



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Investigating the Technologies Available for Guided Tours

by
Ewan Richards (1802187)

Supervisor: Dr. Stephen Hole
Second Marker: Dr. Tim Bashford

Project submitted as part of the requirements for
the award of MSc Software Engineering

February 2023

Declaration Form and Form of Consent

Declaration Form

I, Ewan James Richards, declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed: Ewan James Richards

Date: Monday 20th February 2023

Form of Consent

I, Ewan James Richards, hereby consent that my Project, submitted in candidature for the M.Sc. Software Engineering degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: Ewan James Richards

Date: Monday 20th February 2023

Abstract

This project looks at how Bluetooth, Beacon Technology and QR codes can aid University building tours, and a self-guided tour application prototype has been created. QR codes have become popular over the last few years because of them being contactless, which is of paramount importance within a post-Covid environment.

The Java programming language, and the Android Development Studio software, have been used to create the self-guided tour app. The evaluation methods used throughout the project were interviews prior to implementation and testing after implementation.

The Android prototype was not finished within the time frame, however a wireframe diagram detailing the user interface has been completed and evaluated.

A re-design was made starting from the bottom up using Java and SQL Server through use of Java Database Connection (JDBC) for both a version 1 console application and a second implementation using a graphical user interface (GUI) that integrated images. The hope would be to take this prototype back into the Android phase given more time.

This research has highlighted the utility of QR Codes and Bluetooth Beacons for guided tour apps through their ability to provide location-based information.

Contents

Declaration Form and Form of Consent	2
Declaration Form	2
Form of Consent.....	2
Abstract	3
Table of Figures.....	7
List of Tables	10
CHAPTER 1 - INTRODUCTION	11
1.1 Aims & Objectives	11
1.2 Report Structure	11
CHAPTER 2 – LITERATURE REVIEW.....	13
2.1 Physical Web	13
2.2 Location Based Services	14
2.3 Geographic Information Systems (GIS)	15
2.4 Global Positioning System (GPS)	18
2.5 Indoor Positioning Systems (IPS)	20
2.6 Bluetooth / RFID.....	21
2.7 Bluetooth Beacons and Beacon Technology	22
2.8 Android Studio	23
2.9 Examples of App.....	25
2.10 User Experience (UX)	26
2.11 Mobile App Design	26
2.12 QR Codes	28
CHAPTER 3 – RESEARCH AND DEVELOPMENT METHODOLOGY	30
3.1 Research Methodology	30
3.1.1 Research Onion	30
3.2 Development Methodology	32
3.2.1 Agile	32
3.2.2 Waterfall	37
3.2.3 Prototyping Development Methodology	39
3.2.4 Choice of Development Methodology	40
3.3 Using Bluetooth Beacons for the UWTSD Self-Guided Tour Application.	41
3.4 Testing.....	41
3.5 Data Collection	41
3.5.1 Pre-Implementation	41

3.5.2 Post Implementation	42
3.6 Limitations.....	42
CHAPTER 4 – PRE-IMPLEMENTATION MARKET RESEARCH.....	44
4.1 Existing Application Testing	44
4.1.1 The Gower Heritage Centre	44
4.1.2 The Glynn Vivian Art Gallery	48
4.2 Pre-Implementation Interviews	52
4.2.1 Interview Results	54
CHAPTER 5 – DESIGN AND IMPLEMENTATION	62
5.1 Design.....	62
5.2 Implementation	66
5.2.1 Prototypes.....	73
5.3 Graphical User Interface.....	81
5.3.1 Add Data.....	89
5.3.2 Creating a User.....	98
5.3.3 Connecting to the Database	99
CHAPTER 6 – RESULTS, ANALYSIS AND EVALUATION	106
6.1 Analysis During App Development.....	106
6.2 Post Implementation	108
6.2.1 Usability Tests – Wireframe Diagram.....	109
6.2.2 Virtual Emulator Device (Android Studio).....	112
6.2.3 Physical Device: Huawei P8 lite 2017	118
6.2.4 Researcher Self Testing of Java Application	122
6.2.5 Java Application Results	130
CHAPTER 7 – CONCLUSION AND RECOMMENDATIONS.....	136
CHAPTER 8 – REFLECTION.....	139
CHAPTER 9 – REFERENCES	145
CHAPTER 10 – APPENDICES	152
Appendix A – Gantt Chart.....	152
Appendix B – Pre-Implementation Interview Questions	153
Appendix C – Prototype Interview Questions	156
Appendix D – Java Application Interview Questions	157
Appendix E – Android Studio Code.....	159
Barcode (QR Code).....	159
Camera.....	167
Landmarks (Rooms)	198

App Functions	206
Landmark Activity.....	208
Landmark Collection Java Class	212
Main Activity Java Class	215
Search Landmark Activity Java Class.....	222
Activity Main Layout Design (Home Page Design)	228
Barcode Capture Design (Capturing the QR Code)	230
Landmark Page Design	231
Search Landmark Page Design	233
Areas.....	236
Country	240
Locations	244
Region.....	248
Object ID Formatter Java Class	250
Object Logging Java Class.....	252
Appendix F – Console App.....	255
Appendix G – Java Application	260

Table of Figures

Figure 1: How the Physical Web works [3].....	13
Figure 2: Location Based Services System components [5].	14
Figure 3: GPS vs IPS [29]	21
Figure 4: How the Beacons Work [34].....	23
Figure 5: Unified Gradle Test Runner [38]	24
Figure 6: American Museum of Natural History Explorer (AMNH) [39]	25
Figure 7: How QR codes work [46]	28
Figure 8: Research Onion	30
Figure 9: Waterfall vs Agile [56].....	32
Figure 10: Scrum [61]	34
Figure 11: Waterfall Method [67]	37
Figure 12: Relative Cost of Fixing Defects [69].....	38
Figure 13: Prototype Development Methodology [72].....	39
Figure 14: Welcome screen	44
Figure 15: Map of the Heritage Centre	45
Figure 16: Key for the Map	46
Figure 17: Places by Distance	47
Figure 18: App coincides with Actual Location.....	48
Figure 19: Home page with Accessibility Settings	49
Figure 20: The Rooms	50
Figure 21: Stress Test	51
Figure 22: Incorrect Room Example	52
Figure 23: Attending an Open Day prior to commencing studies.	54
Figure 24: finding information.....	55
Figure 25: were campus tours provided?	55
Figure 26: what did you see on a campus tour?	56
Figure 27: would participants feel that the app would benefit from?.....	56
Figure 28: potential features and places to include.....	58
Figure 29: future developments	60
Figure 30: using and recommending the app	61
Figure 31: Home Page.....	63
Figure 32: Student Services	64
Figure 33: Contact Page	65
Figure 34: Solution Explorer in Android Studio	67
Figure 35: Camera2 Code [81].....	68
Figure 36: importing the camera Android hardware.....	68
Figure 37: attempt of fixing the camera errors.....	69
Figure 38: base code for the landmark factory class.....	70
Figure 39: new landmark factory code.....	70
Figure 40: changes to the app functions java class.....	71
Figure 41: alert dialog	72
Figure 42: home page (wireframe diagram).....	74
Figure 43: QR code scanner (wireframe diagram)	75
Figure 44: search page (wireframe diagram)	76
Figure 45: room page (wireframe diagram)	77
Figure 46: prototype 1 (ionic framework)	79

Figure 47: prototype 2	80
Figure 48: prototype 3 – the final version	80
Figure 49: Initial Design of Java Application.....	81
Figure 50: Database Diagram (Visio)	82
Figure 51: Database Diagram (SQL Server)	83
Figure 52: Creating the Software Table	84
Figure 53: Creating the Lecturer Table	84
Figure 54: Creating the Department Table.....	85
Figure 55: Creating the Room Type Table.....	85
Figure 56: Creating the Sessions Table.....	86
Figure 57: Creating the Hardware Table	87
Figure 58: Creating the Room Table.....	87
Figure 59: Creating the Hardware in Room Table.....	88
Figure 60: Creating the Software in Room Table	88
Figure 61: Adding Data into the Department Table	89
Figure 62: Adding Data to the Room Type Table.....	90
Figure 63: Adding Data to the Lecturer Table	91
Figure 64: Adding Data to the Sessions Table	92
Figure 65: Resolving the error on the Sessions Table	92
Figure 66: Inserting Data into the Software Table.....	93
Figure 67: resolving an error in the Software table.....	93
Figure 68: Adding Data into the Room Usage Table	94
Figure 69: Adding Data into the Hardware Table.....	95
Figure 70: resolving the error in the Hardware table.	95
Figure 71: Adding data into the Hardware In Room Table	96
Figure 72: Adding Data into the Room Usage table.	97
Figure 73: Error with Photos in SQL Server.....	97
Figure 74: Creating a User.....	98
Figure 75: assigning the user permissions	98
Figure 76: Logging into SQL Server.....	99
Figure 77: Showing the Data in the Lecturer Table.....	100
Figure 78: Showing Data from the Lecturer Table	100
Figure 79: Showing Room Data, Hardware and Software Description	101
Figure 80: Further Data Being Shown.....	101
Figure 81: Version 2.1	102
Figure 82: Version 2.2	103
Figure 83: SQL Inject	103
Figure 84: Try Catch Error Code.....	104
Figure 85: Room not registered on app	105
Figure 86: unused import statement on Search Landmark Activity page	106
Figure 87: depreciated code	107
Figure 88: reverting to the older Gradle versions.....	108
Figure 89: testing the features on the wireframe diagram	109
Figure 90: participants rating the app.....	110
Figure 91: outcome of test 1	114
Figure 92: outcome of test 1	119
Figure 93: Evidence of Test 1.1	123
Figure 94: Evidence of Test 1.2	124

Figure 95: Evidence of Test 1.3	125
Figure 96: Evidence of Test 1.4	126
Figure 97: Evidence of Test 1.5	127
Figure 98: Evidence of Test 1.6	128
Figure 99: Evidence of Test 1.7	129
Figure 100: Evidence of Test 1.8.....	130
Figure 101: Chart for question 1: testing Room IQ104	130
Figure 102: Chart for question 2: testing room FF002	131
Figure 103: chart for question 3 (testing rom IQ310) and testing question 4 (testing FF105)	132
Figure 104: Chart of App Ratings	134
Figure 105: Original Gantt Chart	152
Figure 106: Updated Gantt Chart	152

List of Tables

Table 1: GIS Geography [7].....	15
Table 2: Wireless Solutions [30].....	22
Table 3: differences of utility, usability, and useful [41].....	26
Table 4: 3 Scrum Core Principles.....	35
Table 5: 5 Scrum Values.....	35
Table 6: sprint workflow.....	36
Table 7: feedback from users regarding potential app features	59
Table 8: future work suggestions from participants	61
Table 9: Gradle version	73
Table 10: Relationships	83
Table 11: test plan for the virtual device	113
Table 12: QR Codes on Virtual Device Camera	115
Table 13: Search by ID.....	116
Table 14: Search by Name	117
Table 15: test plan for the physical device.....	119
Table 16: outcome of testing the QR code.....	120
Table 17: Search by ID.....	121
Table 18: Search by Name	122
Table 19: test plan for test 1.1.....	123
Table 20: test plan for test 1.2.....	124
Table 21: test plan for test 1.3.....	125
Table 22: test plan for test 1.4.....	125
Table 23: test plan for test 1.5.....	126
Table 24: test plan for test 1.6.....	127
Table 25: test plan for test 1.7.....	128
Table 26: test plan for test 1.8.....	129

CHAPTER 1 - INTRODUCTION

1.1 Aims & Objectives

The aim of the project is to build a university tour application for the UWTSD IQ building which utilises location-based technologies to provide the user with locational information.

The objectives of the project are:

1. To research the use of GIS, Bluetooth and QR codes for providing locational information.
2. To research similar apps to see how they work as well as the technology that is used and analyse the user interface.
3. To choose three locations of the ground and first floors of the IQ Building to place the QR codes.
4. Set up the three QR codes at the locations and test them to ensure that they work.
5. Build the application using Java and collect user feedback.

1.2 Report Structure

Chapter One illustrates an overview of the report. It details the aims and objectives that once achieved will bring success to this research.

Chapter Two reviews the literature on the technologies available for self-guided tour apps.

Chapter Three details the research approach and design, such as the research onion, which will demonstrate how the app will be created, implemented, and used.

Chapter Four reviews the market research pre-implementation prior to creating the app.

Chapter Five focuses on the design and implementation of the app.

Chapter Six concentrates on the results, analysis, and evaluation of the application.

This includes the testing done of the app, as well as any issues encountered during development, and the results from the usability tests based on the wireframe diagrams.

Chapter Seven details on the conclusion and recommendations, particularly on future work of the app.

Chapter Eight focuses on reflecting on the project.

Chapter Nine outlines the references and sources used to aid this research project.

Chapter Ten is the appendices of the project which includes the original and updated Gantt chart, pre-implementation and post-implementation questions, and code for development.

CHAPTER 2 – LITERATURE REVIEW

2.1 Physical Web

By using Bluetooth beacons, the physical web locates online pages that have a specific location. The URLs can be displayed in a variety of internet browsers on mobile devices. [1]

The physical web enables rapid communication with real-world sites. Users can interact with any area by using the physical web, and an interaction is only a touch away. The ability to broadcast any information and places is one advantage of the physical web. Any item that can be read by a Bluetooth Beacon will broadcast the physical web ready URLs being shared. After passing through a proxy, web browsers like Google Chrome can display URLs. [2]

Google Physical Web : how it works

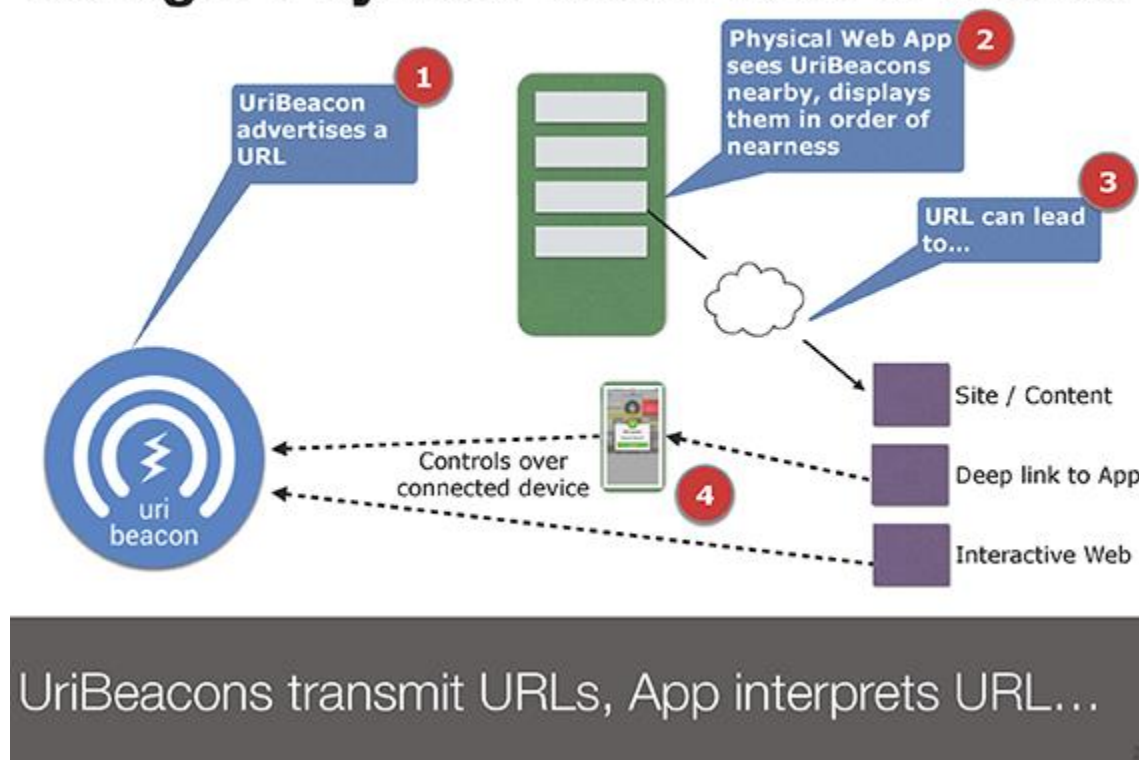


Figure 1: How the Physical Web works [3]

Figure 1: How the Physical Web works demonstrates how the Bluetooth Beacon reads a URL, the application using the physical web then reads the near beacon and it then shows them in the order of which they are located and finally, the URL discovered by

the beacon is then lead to the relevant page, website or link to the application or section of the application. [3]

2.2 Location Based Services

Location, mobility, and connection are essential components of location-based services (LBS), which are all crucial to computational managerial events. LBS are now necessities for both daily life and business because of improved access in the modern day. With the use of location-based applications, security risks have increased significantly. [4]

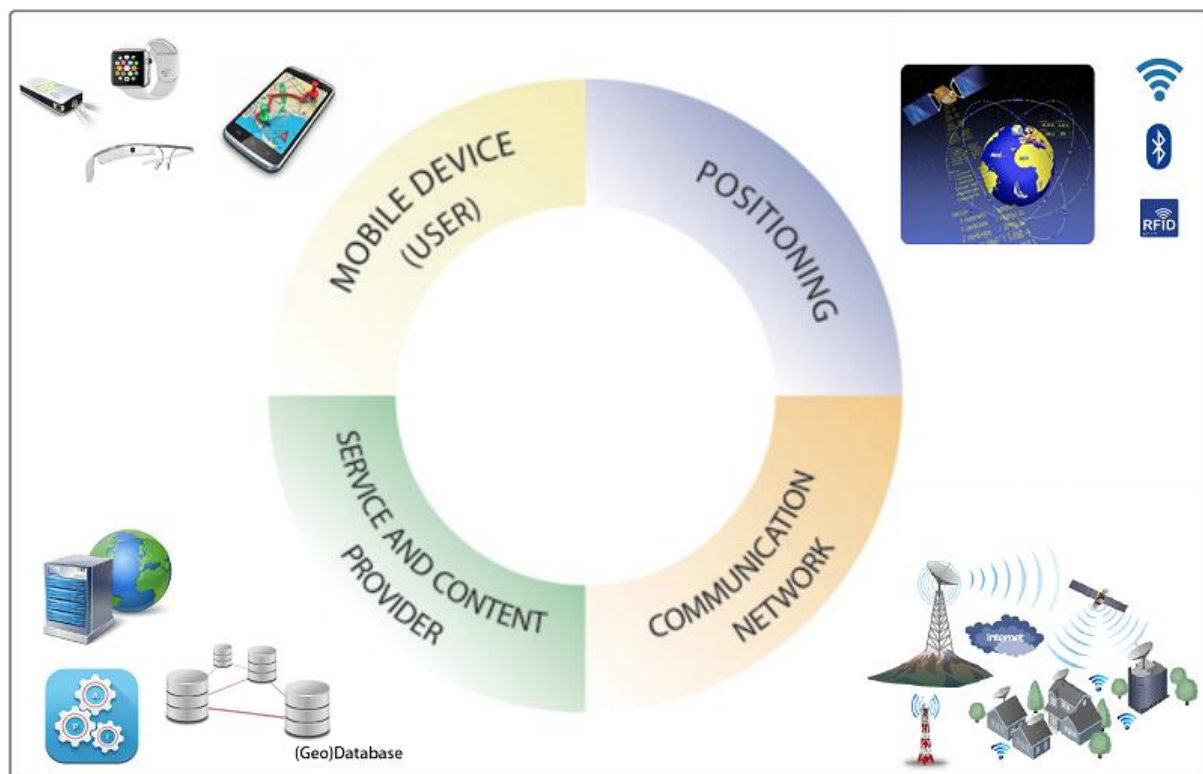


Figure 2: Location Based Services System components [5].

Figure 2 (above) demonstrates the many elements of location-based services. When a device asks the user for their present position, the response can be displayed either visually with a map or verbally with textual instructions and a description of where they are. The placement element determines the precise location of the users. Map matching methods are applied when the GPS signals are unreliable. Wi-Fi, Bluetooth, and RFID are the technologies that are used for indoor location-based service applications. The request from the user utilising the mobile device and the data is transferred to the content provider via the communication network, such as Wi-Fi. The service and content supplier gathers user requirements before providing the necessary

information. It frequently uses sources to get the data needed to handle the service request. [5].

The foundation of location-based services is the necessity for people to find their way in uncharted territory. A computer service known as a "location-based service" offers a particular feature based on the present position of a mobile entity. In certain circumstances, the mobile entity is a mobile device whose location is linked to the location of the user. In other situations, such as fleet management systems or logistics, it is necessary to pinpoint and track the location of some things. [6]

The three fundamental components of location-based services are as follows. First, location-based services make use of people's ability to have their whereabouts determined dynamically. This definitional phrase relates to the idea of place. Although limited to the communication of place, the location feature is incorporated. Finally, the word "utilising" includes the geoinformation aspect or the value aspect. [6]

2.3 Geographic Information Systems (GIS)

GIS technology can design internet connection architecture as well as telecommunications networks for consumer electronics. Once the data has been compiled and analysed across all data kinds, it is combined with a map and the locational data is integrated with all the descriptive data. [7]

GIS collects and allows visualisations of information about specific locations within the Earth. Much information can be demonstrated on a single map, thanks to GIS. [7]

GIS is used to map positions and specific objects and it allows to see what is happening within a specific area. It is also use, more specifically for the following, as shown below in Table 1: GIS Geography .

Geography	Details
Environmental	This tests the impact the environment has on users.
Physical	This looks at the atmosphere components.
Emergency	It provides live data about the current location to the emergency responders.
Health	It studies health issues (illness) which uses geographic data.
Economic	It looks at the Earths economic events.

Transportation	It looks at the longitudinal communications between users and objects.
----------------	--

Table 1: GIS Geography [7]

GIS expenditures are not cheap, especially with maintenance fees, but it can help organisations express their visual representations more effectively. Additionally, any modifications can be recorded, making choosing simple. [7]

GIS includes attributes with synthesis and analysis information as well as functions for its figures. The topology relationship between visual elements and the function modules of map matching, path query, and path lead would also be displayed on a map, in addition to data in coordinate format. [8]

Geospatial information is visualised via GIS mapping. The creation of geographic data, its management in a database, its analysis and pattern-finding, and its display on a map are the four central concepts of GIS. We can use GIS to improve our decision-making because viewing and analysing data on maps interacts with our comprehension of the data. Our understanding of what is there is improved, making the analysis easier. [9]

GIS can help with the creation of informational visual plans. Google Maps is a wonderful illustration of this; it is a navigation programme that people use as a mapping solution. GIS data can be used to solve transportation issues. Many organisations can plan a limited number of routes for their transportation services since GIS may combine topical and environmental data utilising a GIS software platform. [10]

There are many software and platforms available which utilises GIS such as MapInfo, ArcGIS, Google Earth, Google Maps and Quantum Geographic Information System (QGIS)

Many seasoned geospatial analysts utilise MapInfo, a desktop GIS programme. It offers raster data manipulation, spatial querying, thematic mapping, and map layout. It has a helpful user interface and is quite efficient. [11]

Users may access workflow-specific apps, maps, and data from across the world, tools for being mobile in the field, and other resources using ArcGIS, a cloud-based mapping and analysis solution. Users can create maps, analyse data, share, and collaborate, and utilise ArcGIS to produce maps and analyse data. [12, 13]

By using ArcGIS, users can easily share data, maps, apps, and other items with teams departments and the public as well as users being able to visualise the data spatially through web maps and apps that can be accessed from anywhere at any time. [14]

Versions 9.3 and 10 of ArcGIS are the most recent iterations. The expense of the software licences for the core programming and necessary extensions like Spatial Analyst is the key reason restricting the use of ArcGIS. [15]

A well-known and popular GIS programme is Google Earth. It enables users to produce, store, view, and interact with location-related data. It is a commonly used GIS platform and is more effective for displaying routes and travel times. [16]

GIS maps show you everything that is above, below, and unseen, as well as historical data so you can forecast the future. Google Maps only displays what the human eye can see on the surface. [17]

Manifold is an example of a GIS solution that is more affordable than other commercial GIS software solutions. Although it offers many of the same features as more expensive GIS software packages, you must make sure you have the proper licences in order to use all the GIS capabilities you might require. Additionally, compared to most other GIS applications, it is founded on a fundamentally different GIS methodology. Because of this, some GIS experts could find it challenging to transition from the Manifold GIS methodology to that employed by other GIS software products. [15]

QGIS is an open-source software that allows users to create, edit, visualise, analyse, and publish geospatial information. There are many benefits to using GIS, such as the software offers free online resources and maps available to download. [18, 19]

Clark Labs developed a line of software products called IDRISI, the most recent of which being IDRISI Taiga. It offers a lot of the same GIS capability as ESRI products and, in some cases, might even offer more. Particularly, many users discover that using IDRISI software makes it simpler to work with time series data and data from remote sensing sources. Although the requisite licences are less expensive than those for ESRI products and some academic users are eligible for reductions, they may still be too expensive for some customers. [15]

2.4 Global Positioning System (GPS)

At essence, GPS is a radio navigation system that constantly determines the user's location and time in that area. GPS can still locate the user no matter where they are in the world. [20]

In order to provide users with navigational guidance and routes to locations, Google Maps incorporates GPS in its application. This allows it to read the user's current location in real time. Since it can recognise bus stops and train stations, it also provides real-time schedules for public transportation, including buses and trains. Numerous health applications monitor users' whereabouts as they work out. It can record the distance they have travelled while exercising and gather information about the terrain they have covered. [21]

Satellites in the GPS system give fine accuracy. The latitude, longitude, and height may normally be read out by four satellites. The GPS receiver gathers information from satellites and updates it in real time so that it can analyse movement direction and speed. [8]

Currently, GPS provides two levels of service – a Standard Positioning Service (SPS) for general civilian use and an encoded Precise positioning service (PPS) primarily

intended for use by the Department of Defence. The SPS signal is not as accurate as the PPS signal. While PPS pinpoints the location of an object within 3 feet, SPS can only pinpoint an object to within 110 yards. [22]

Web-based GIS demonstrated, in conjunction with SDIs, that the GIS sector can take a flawlessly straightforward web browser and transform it into a fiendishly tricky and sophisticated map system. Up until Google Maps eventually put us out of our agony in 2005, this went through several iterations. [23]

A GPS receiver connects with satellites that circle the Earth via radio waves. Currently, there are 27 GPS satellites in orbit, of which 24 are actively used and 3 serve as backups in the event of satellite failure. Trilateration is used by the receiver to pinpoint your precise location. It creates a sphere around each of the three satellites it is capable of occulting. One of the points where these spheres connect is in space, and the other is on the earth. In wireless-assisted systems, the phone uses data from the mobile phone's signal along with information from the GPS satellites in orbit. Wireless-assisted GPS, also known as improved GPS, can frequently pinpoint the user's location more quickly than a GPS-only receiver. [24]

A GNSS or GPS antenna is a tool used to capture, amplify, and transform the radio signals sent by GNSS satellites at certain frequencies into an electrical signal that can be used by a GNSS or GPS receiver. A GNSS or GPS receiver that can compute the position receives the output of the GNSS or GPS antenna. [25]

The Trimble GIS antenna is a well-known GIS antenna. The Trimble GIS antennas are top-notch GNSS antennas for usage in challenging GPS situations and in areas with heavy multipath, like under cover of trees and in urban canyons. They work with Trimble GeoExplorer series handhelds and Trimble Pro series receivers and are perfect for all mapping applications. [26]

Low-power transmitters built into cell phones enable them to connect with the nearby tower [24].

With GPS timing receivers, the two most common antenna types are roof-mounted and window-mounted. The more precise GPS clocks need the roof-mounted antenna because timing precision, which is normally measured in nanoseconds to UTC, depends on having a constant view of three satellites. The window-mounted antenna is appropriate for network time servers that run with a lower level of precision, often microseconds or occasionally low milliseconds to UTC, and can work with as few as one sporadic satellite in view. [27]

A smartphone is an advanced two-way radio. Radio signals are sent and received by tower and base stations that are grouped into a network of cells. The towers transmit your signal from one cell to the next as you walk from one to the other. Towers in isolated areas may be too far apart to produce a reliable signal, and even when there are many towers, mountains can cut off transmissions. Based on measures of your signal, such as its angle of approach to the cell towers, the time it takes the signal to reach several towers, and the strength of your signal when it does, a computer may pinpoint your location. [24]

2.5 Indoor Positioning Systems (IPS)

For indoor positioning, technologies like Bluetooth and RFID have been used. Because of the challenging environment that makes it difficult to locate wireless signals, indoor positioning is now in an early stage of adoption. [28]

A technology like GPS hasn't yet been publicly developed, and each indoor positioning system has drawbacks that prevent widespread usage. Yu, Liu, Chi, and Peng say that iBeacons are Based on Bluetooth Low Energy (also known as BLE) devices, the beacon is initially intended to provide push notifications to clients when they are nearby.

Bluetooth-based positioning technology has been applied in a variety of contexts for active positioning or passive positioning (tracking). [28]

Only indoor or nearly indoor situations, such as those seen on a university campus, are considered by IPS. Unless the system is turned off by the user, an IPS should always function and provide tracking of persons or objects in the intended region within the permitted time delay. [29]



Figure 3: GPS vs IPS [29]

Figure 3: GPS vs IPS (above) depicts a typical indoor coverage scenario for GPS vs. IPS systems. While the IPS system can show the floor and shop the user is on, the GPS locating system can only follow the user as far as which area of the building they are in. [29]

2.6 Bluetooth / RFID

With the advent of short-range wireless technologies over the past ten years, including Bluetooth and Radio Frequency Identification (RFID), a new transmission structure has started to emerge, with numerous products and services benefiting from this development. Numerous location-aware applications in a range of industries and professions, including tourism, healthcare, logistics, and manufacturing, have been developed based on these wireless technologies. [30]

Bluetooth is a wireless technology that does not require cables to share data over a short distance. On your mobile device, Bluetooth can be used to transfer files or establish connections with other Bluetooth-capable gadgets. [31]

Radio Frequency Identification (RFID) is a technology that utilises radio waves to transmit data when an RFID tag is scanned. The most basic kind of RFID chips only output a unique ID when it is scanned, while others contain encryption and control-sequences so that only authorised scanners with receive its ID. [32]

Tracking systems based on active RFID tags works in the same way as Wi-Fi based positioning systems. In passive RFID systems, the readers need to be placed at the multiple points-of-interest (PoI) for the readers to detect if a passive RFID tag is near. [33]

Wireless Solution	Accuracy	Cost	Indoor or Outdoor
Bluetooth	2 to 3 metres	Medium	Both
RFID	Under 3 metres	Low	Both

Table 2: Wireless Solutions [30]

There is a significant difference between these solutions. With Bluetooth, this has a higher accuracy than RFID however it could potentially cost more to have a Bluetooth wireless solution. With RFID, its accuracy is under 3 metres however its cost is relatively low to reflect this. Both Bluetooth and RFID can work indoors and outdoors, which would be beneficial for this project as there would be some sensors location on the outside of the building main entrance, so it is good that these can work in outdoor environments. [30]

2.7 Bluetooth Beacons and Beacon Technology

A Bluetooth Beacon is a wireless gadget that relies on Bluetooth Low Energy, also known as BLE. Other technology can detect the signal that it produces. A radio signal with Bluetooth on it is made up of a string of letters and numbers that can reach a Beacon once it has been made visible to the mobile device itself. [34]

Eli Zimmerman has said that

"Universities see a benefit of beacon technology in recruitment" [35]. He then goes onto say that "University IT teams installed 20 Aruba beacons at points along the tour" and "when prospective students open the self-guided tour applications, it enables location services on their phones. Then, as students pass by the beacons, they receive notifications based on nearby landmarks" [35].

By having this type of self-guided tour, it can assist with students becoming familiar with their new surroundings and it can engage with the community further.



Figure 4: How the Beacons Work [34]

Figure 4 demonstrates how Bluetooth beacons work. Once the signal has reached the mobile device, it pings where the user is currently located, and the device details are sent to the cloud. An example is that if the user is in a university building and the user is walking past a specific room, the beacon located in the area would alert the user that they are currently at this department. [34]

According to Aneta Ciurkot, "Bluetooth provides the infrastructure for the entire beacon ecosystem" that it is "a standard for sending data over short distances, a wireless technology not so dissimilar from Wi-Fi." [34]

2.8 Android Studio

You can create apps for Android phones, tablets, Android Wear, Android TV, and Android Auto using Android Studio, which offers a single development environment. You can break your project into functional pieces that you can separately create, test, and debug using structured code modules. [36]

The official integrated development environment (IDE) for developing Android applications is called Android. It combines the code editing and development capabilities from IntelliJ IDEA, a Java integrated development environment for applications. [37]

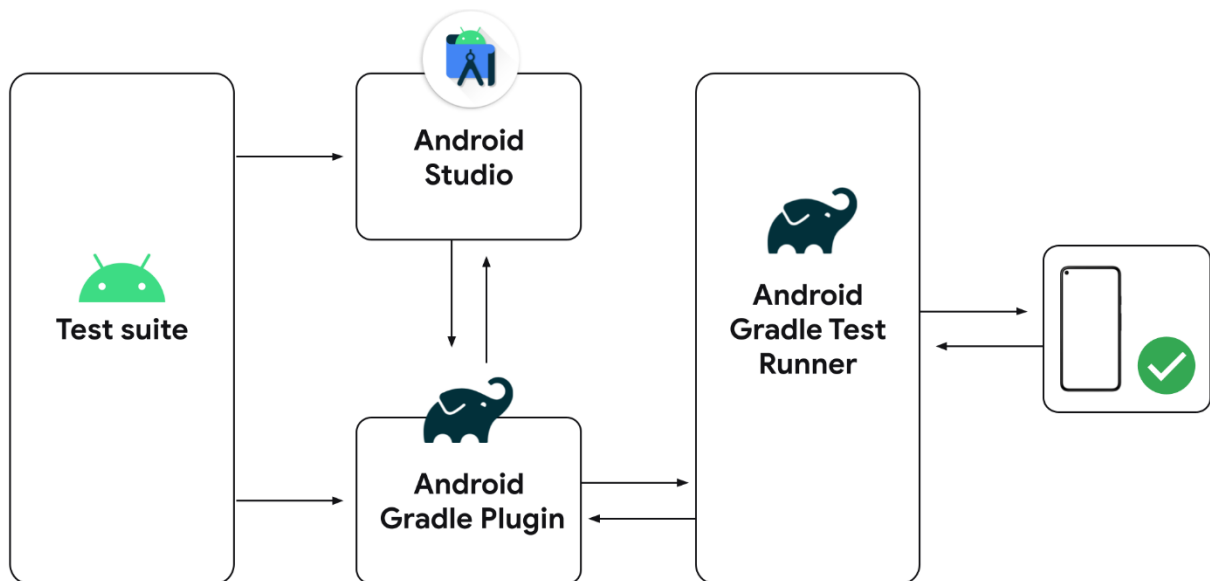


Figure 5: Unified Gradle Test Runner [38]

Instrumented tests are run using Gradle's own version of the Android instrumented test runner in the Android Gradle plugin 7.1.0 with Android Studio Bumblebee and higher. Results are likely to be consistent if you run using AGP from the command line, such as on a continuous integration server, or from Android Studio if you use the same test runner. [38]

2.9 Examples of App



(A) AMNH's home screen

(B) AMNH's navigation

Figure 6: American Museum of Natural History Explorer (AMNH) [39]

Figure 6: American Museum of Natural History Explorer (AMNH) shows screenshots of the AMNH application. According to the museum, The AMNH is the first indoor GPS navigation application [39].

A few of the services provided by the AMNH, which was introduced in 2010, include turn-by-turn instructions to displays, customizable museum tours, and details on 140 specimens and artefacts. The application uses 300 Wi-Fi access points to triangulate users' positions inside the museum over a total area of 500,000 square feet. As with MazeMap and Meridian, Cisco MSE serves as the foundational technology. Only Apple products, including the iPad, iPod Touch, and iPhone, can use the application. [29]

Wireless Trondheim and NUNU collaborated to produce MazeMap, formerly known as CampusGarden. The software-as-a-service (SaaS) programme was initially created to make it easier for students and visitors to navigate interior and outdoor areas of Gloschaugen, the main campus of NTNU. Glosgaugen has 60 buildings with 13,000 rooms totalling 350,000 square metres of floor space. The MazeMap Application Programming Interface (API) is used by the location-based service MazeMap to determine users' whereabouts and the shortest route between places. [29]

2.10 User Experience (UX)

UX focuses on understanding users' needs, requirements, principles, abilities, and limitations. UX is a field that is constantly growing. Useful, useable, desirable, findable, accessible, and believable are the guiding principles of user experience. [40]

The material on the app should be original, have a purpose, be simple to use, and be there to do so. The information on an app should be simple to find, and media, branding, and other design elements should be leveraged to inspire passion and comprehension. Consumers with disabilities should be able to access the app's content, and users should trust the information the website provides. [41]

Utility, which indicates the effectiveness of the design, is a crucial component. Both usability and utility are crucial, and they can be used to determine whether something is useful. The same user study methodologies that improve usability may be applied to understanding a device's service. Table 3: differences of utility, usability, and useful (below) includes the key distinctions between useful, usable, and utility. [41]

Word	Definition
Utility	Whether it provides the components the user needs.
Usability	How easy and pleasurable it is to use these components.
Useful	Usability and utility.

Table 3: differences of utility, usability, and useful [41]

2.11 Mobile App Design

The task of designing mobile applications is known as mobile app design. Mobile apps can be used for a wide range of purposes, but they all share the need for the best usability, accessibility, engagement, and overall user experience. [42]

The user experience of your app is affected by the mobile app design, which is why it is crucial to the success of your app. The way your app appears, and functions has a big impact on how a user interacts with it. [43]

Colour, typeface, icons, drawings, brand design, and navigation are the 6 essential components for creating the ideal mobile application. These have the greatest impact on producing a fantastic user interface. By incorporating these essential components, the mobile app user interface would give users a first image of the company, and the better the app user interface is, the greater user experience and satisfaction they will have. [44]

2.12 QR Codes

Website connections, account details, phone numbers, and even vouchers can all be found in QR codes, a type of barcode or scannable pattern. Menus, social media, and notice boards all contain QR codes, but their use has increased since the Coronavirus outbreak because of their contactless nature. [45]

The QR Code was created in 1994 by Nasahiro Hara, head engineer of Denso Wave, a Japanese business and division of Toyota. Initially, it was used to track components and vehicles as they were manufactured. Quick Response (QR) codes are a barcode that can be read by smartphones and other digital devices. In contrast to traditional barcodes, which can only be read from top to bottom, they contain information as a series of pixels in a square grid that can be read from both the top and bottom and the right and left. [45]



Figure 7: How QR codes work [46]

Figure 7: How QR codes work (above) demonstrates on how QR codes work. No matter how much information they contain, a QR code should rapidly give the user access to that information when it is scanned, which is why it is named a Quick Response code. [47].

On most cameras-equipped devices, including phones, smartphones, and tablets, you may scan QR codes. Once you have done so, you are free to open or share the URL of the website the QR code points to. [48]

A type of 3D barcode known as a QR code can hold a variety of data. It is quite simple to encode a website URL and other things that you do not have space for within a normal barcode because QR codes may hold thousands of pieces of data. [49]

Dynamic QR codes are trackable, meaning that once they are completed, records of their usage will start being tracked. This includes information such as the location of the scan, the number of scans, what time the scans took place, as well as the operating system of the device used. This allows you to see data in real-time so you can plan campaigns more easily or even change your QR code entirely if something is not working. [50]

The operating system, location, total number of scans, unique scans, and time are the key metrics for QR codes. Most users use mobile devices, each of which runs a specific operating system, to scan QR codes. The counties and cities that make up the total number of scan locations can be split down. Based on an individual's IP address, this data is provided. The total number of scans reveals the total number of times a QR code was scanned, even if it was done so by a single person more than once. Particularly when contrasted to other data like time and location, this information is helpful. Additionally, there are unique scans, which differ slightly from total scans in that they represent the number of times a single user has scanned a QR code. The time is divided into the month or day in the metrics for QR code scanning. [50]

CHAPTER 3 – RESEARCH AND DEVELOPMENT METHODOLOGY

3.1 Research Methodology

3.1.1 Research Onion

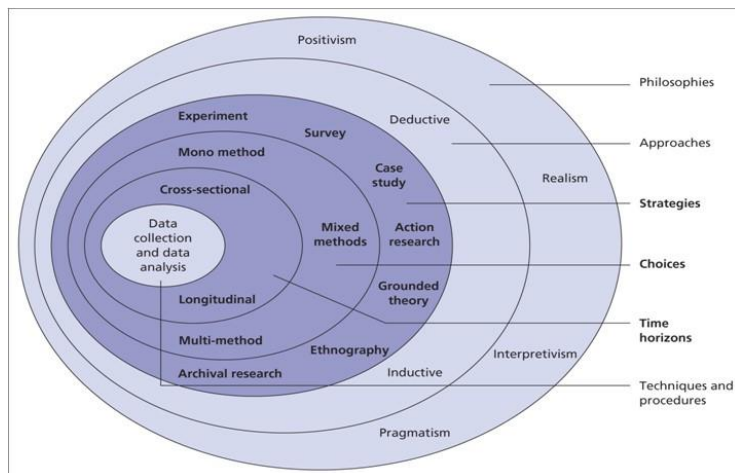


Figure 8: Research Onion

Figure 8: Research Onion (above) demonstrates the research onion. The research onion was developed in 2007, and one advantage of employing it is that it results in a list of phases of various data collection techniques that may be recognised. It also demonstrates the procedures required for interpreting a study approach.

Positivism is the best approach for the project as, once the application gets fully implemented, users will be asked to test it to ensure that all the functionality works, their actions would be observed – this is to determine that if the user is finding a particular task difficult this can be evidenced to the developer. By using observation, the developer would see what features the user likes using as well as the features that do not work particularly well. According to the Business Research Methodology, “positivism adheres to the view that only factual knowledge gained through observation is trustworthy.” It then goes on to say that “positivism depends on quantifiable observations that lead to statistical analysis.” [51]

With the aid of empirical facts, the deductive research approach offers a better understanding of numerous social obligations. It offers the observations needed to build the conclusions from the findings to accomplish the objectives of the study. To accomplish this, primary research is carried out utilising a standardised questionnaire to understand study field observations. [52]

For the project, the positivism philosophy will be used, and the deductive approach will be implemented. The best strategy for this project is positivism because after the application is fully implemented, users will be asked to test it to make sure that everything functions as intended. Their actions will also be watched, allowing the developer to see if the user is having trouble with a particular task. The developer would utilise observation to learn which features the user enjoys using and which features function very well. [51]

The mixed method is based on the use of two or more research methods and frequently refers to the use of qualitative and quantitative methodology. This was chosen because, both results from numerical questions and interviews were conducted, so both quantitative and qualitative results would be collected. [53]

The survey strategy has been used and the mixed method choice has also been used alongside the strategy because this would assist with the analysis of results, based on the responses received and quantitative results would be produced. The case study approach combined with a survey constitutes the research strategy. The research is focused on a specific contextual environment in this research technique. [54]

Cross-sectional time horizon will be used along with data collection and data analysis techniques as the research strategy has been explained, with primary and secondary results being analysed. Where a predetermined period of time is set aside for data collecting, that is when the cross-sectional time horizon occurs. [55]

Data Analysis, techniques and processes make up the research onion's last layer. This is the part of the essay where you should clearly state how and why you are conducting the research. This can apply to either primary or secondary data. You must clearly explain your research strategy and provide solid explanations for each step. This offers a framework that details the factors considered when choosing the best methodology, how respondents were chosen, and how the data will be analysed. [55]

Data will be collected in the form of interviews prior to the implementation being completed, to aid in gathering requirements. The data will be analysed to see which features users would like to see the most. Data will also be collected post-implementation to test whether the app works and what future work could be implemented to improve the app.

3.2 Development Methodology

3.2.1 Agile

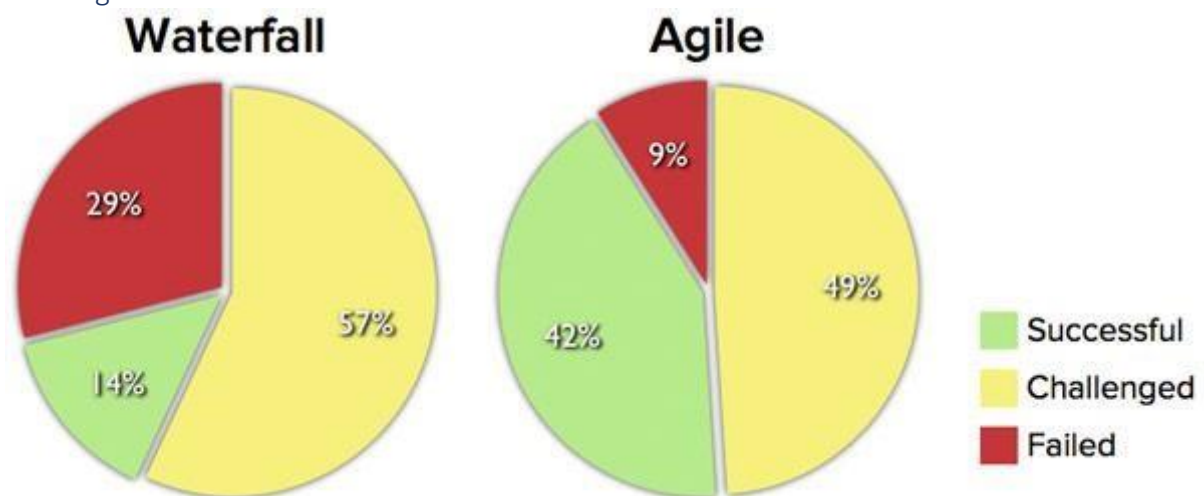


Figure 9: Waterfall vs Agile [56]

Figure 9 demonstrates the two different methodologies of waterfall and agile. Within the waterfall methodology, 29% of projects fail which is 20% higher than the agile methodology however, 57% of projects done using the waterfall methodology has been successful, 8% higher than the agile methodology. Within the agile methodology, 42% of the projects have been challenged and just 14% within the waterfall methodology. This statistic demonstrates that, 28%, there was more projects challenged in agile than in waterfall. [56]

Agile assists to create what the client wants as the final prototype, this approach can developer in small bits. The requirements are evaluated constantly as well as the results to see what is working and what is not with the application. The development team can also review the requirements to see whether the objectives set are realistic within the project timeframe. [57]

Agile initiatives benefit from teamwork and can swiftly adapt to new requirements. In the 14th Annual State of Agile report from digital.ai, 95% of respondents claimed that their companies used Agile best practises, and 61% of the 40,000 respondents had been utilising this technique for more than three years [58]. If this strategy is followed, stakeholders are frequently involved with the development team and can offer input to the team to ensure that their needs are met, resulting in a greater overall quality of the software produced. The flexibility and predictability of the agile methodology are two of its primary benefits. Agile teams can deploy changes with little to no disturbance and can do it fast. The development team can examine and adjust the implementation consistently if the product owner's needs change. All of this is possible because agile technique is flexible. Agile teams typically work in tiny stages on projects. They are known as sprints. Typically, these sprints last two weeks, and project managers can easily see the results. [57, 58, 59]

Agile approach training is necessary for developers to ensure that any programme may be implemented successfully. Many developers lack the knowledge and abilities necessary to use this process. Using the agile technique when developers are working on a big project is challenging. Some organisations may need to change some of their processes for the agile methodology to function well; this may necessitate that developers collaborate and build partnerships. [60]

3.2.1.1 Scrum

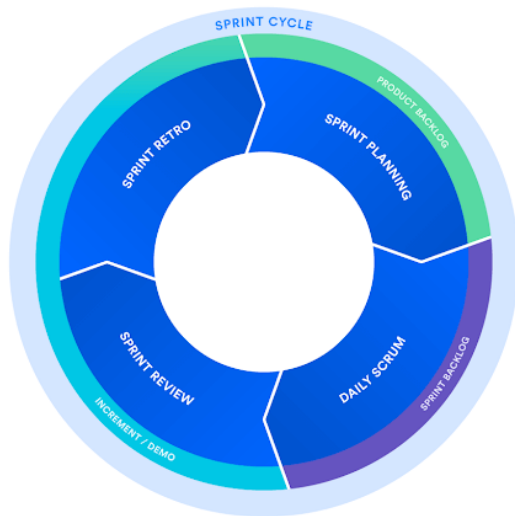


Figure 10: Scrum [61]

Figure 10: Scrum (above) shows how the Scrum methodology is followed and implemented.

Every new product release during a sprint went through one or more iterations of development. Iteration indicates that we can edit our work before it is turned into a new product increment. The code may need to be refactored in a subsequent sprint. [62]

A concrete step toward the product goal that satisfies the definition of doneness is called an increment. [62]

There are three core principles within Scrum, which is transparency, inspection, and adaption [63].

Scrum Core Principles	What it means
Transparency	All pertinent project elements must be clearly defined and made apparent to any team member who is responsible for them.
Inspection	To track advancement toward Sprint Goals, Scrum artefacts must be routinely examined.
Adaption	If an inspector determines that a project component isn't accomplishing the desired results, that component needs to be changed as soon as possible.

Table 4: 3 Scrum Core Principles

There are also five scrum values: commitment, courage, focus, openness, and respect [64].

Scrum Values	What the values are
Commitment	Every sprint, each team member makes a personal commitment to achieve the team's objectives.
Courage	Members of the team are confident in their ability to resolve disputes and difficulties amicably so they may act morally.
Focus	There should be no work done outside of the Sprint Backlog; instead, team members should be concentrating solely on their team goals and the Sprint Backlog.
Openness	Both team members and their stakeholders commit to being open and honest about their efforts and any difficulties they encounter.
Respect	Team members value each other's technical proficiency and sincere motives in their job.

Table 5: 5 Scrum Values

The Sprint Backlog is created in planning, and it contains user stories and tasks to complete a goal. It is a team focus on achieving the goal, and the spring backlog is owned by the scrum team – only they can change it. The sprint backlog is the source

of progress information and should be publicly visible. A good backlog is when it is detailed appropriately, estimated, emergent and prioritised (DEEP for short).

A sprint is a condensed period used to build or "finish" an iteration that may be released. The following sprint starts right away after the previous one concludes. The team chooses related user stories that can be finished quickly and represent a feature or a fix for a problem. [65]

There are four main stages of a sprint workflow: sprint planning, daily scrum (stand up), sprint review and sprint retrospective. [66]

Stages	Description
Sprint Planning	Set the upcoming sprint's sprint goal. In a sprint planning meeting, it should be discussed what can be done within the following month and how it can be done.
Daily Scrum (Stand Up)	The development team meets daily to go over the work that will be done the following day.
Sprint Review	After a Sprint is over, the team analyses the results of the increment and modifies the Product Backlog, as necessary.
Sprint Retrospective	The team meets after the sprint review but before the following sprint to discuss any changes that could be made to practises or processes for the following sprint.

Table 6: *sprint workflow*

3.2.2 Waterfall

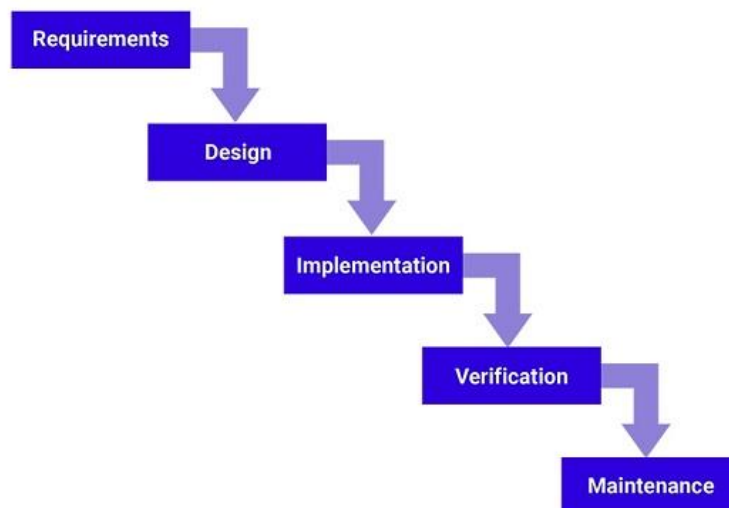


Figure 11: Waterfall Method [67]

The waterfall technique, which Herbert D. Benington created and first defined in 1956, is most frequently employed for software development that takes place over time. Analysing the requirements comes first in the model, while testing comes last. The drawback of this approach is that it is challenging to climb back up the waterfall, and testing is done later in the development process, which increases the likelihood that developers may discover missing requirements late in the process.

The waterfall methodology is a much-used methodology. It consists of five stages that are concentrated on certain goals. Before proceeding to the next stage, each section needs to be finished in its entirety. Every stage has a specified flow into the stage below it. [67, 68]

In the requirements stage, tests will be carried out on an existing University tour app to evaluate its features. It would be investigated potential users to establish what their requirements would be. In the design stage, mind maps would be created to demonstrate what needed to be implemented as well as its purpose. The programming languages for the application will also be included as well as the software required to create the application. In the implementation stage, the software will be coded in a programming environment and the application will be viewed at different stages of development to view what has been done and to see whether anything needs altering. Tests will be carried out to ensure that any issues are discovered straight away, and they can be remedied straight away. Tests will be performed using expected results

and comparing to actual results and through user testing and evaluation. In the verification stage, to verify that the application works, users will be asked to test a part of the app, and it would be tested to ensure stability and efficiency. In the maintenance stage, if there are any issues discovered within the verification tests these would be maintained and the application will be re-tested. [68]

This approach has the advantage of being simple to comprehend. This could be the better methodology to use for the project after clear goals have been established. Assessing the progress of a project is simple. [67, 68]

The model is sluggish while utilising this methodology because of its strict design. Financial restrictions may prevent going back to earlier stages and rebuilding the app if the user is not satisfied with it during the verify phase. [67]

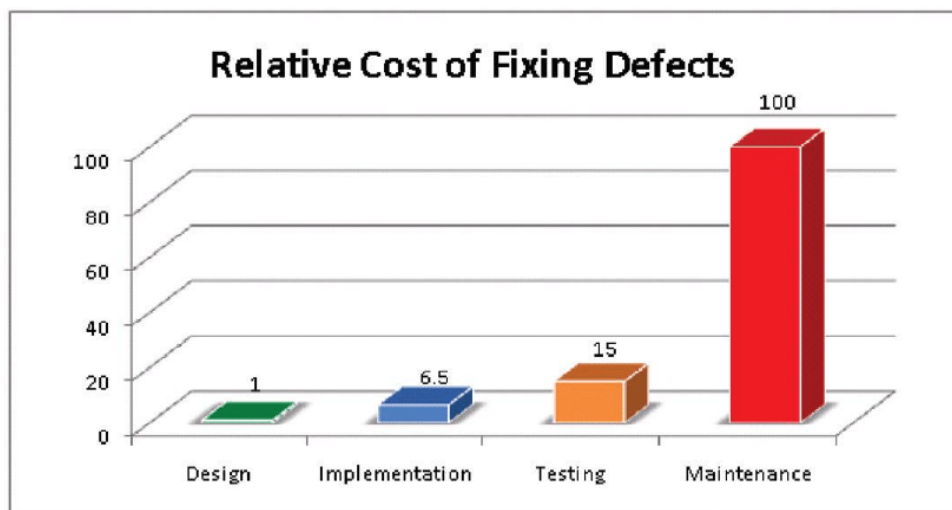


Figure 12: Relative Cost of Fixing Defects [69]

Figure 12 (above) demonstrates how costs for errors difference between the start of the project over the end of the project. If any errors with the software are found within the design stage of the project, costs would be minimal with errors found during the implementation stage, it would cost 6.5 time more. At the developer goes through the project, if there were errors discovered within the testing stages of development, it would cost 15 times to fix more than if they were found in the design stage and if errors were found within the maintenance stage, it would cost 100 times more than if they were discovered in the design stage. [69]

3.2.3 Prototyping Development Methodology

A prototype is made, tested, and then revised as necessary under the prototyping model, a systems development approach, in order to reach an acceptable result from which the whole system or product can be created. [70]

A prototype might be anything from an intricate pen-and-paper drawing to a fully functional version of the product. Prototype development is thus nothing more than a set of procedures the manufacturer uses to create the prototype. [71]

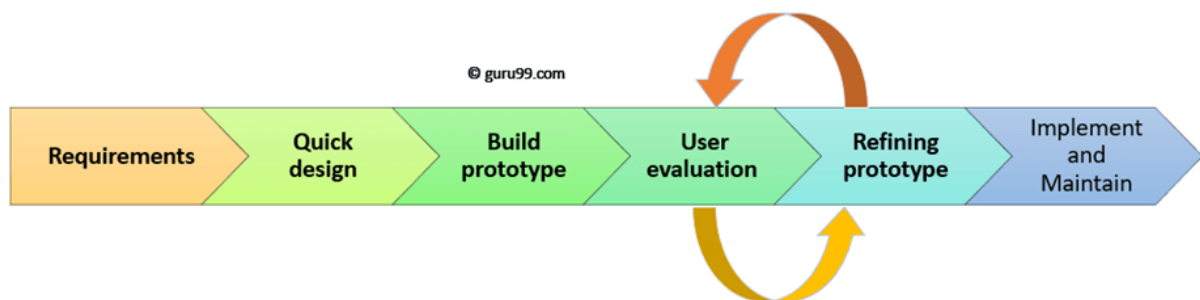


Figure 13: Prototype Development Methodology [72]

Figure 13: Prototype Development Methodology (above) shows the prototype development methodology. The 5 main steps for this model are requirements gathering and analysis, quick design, build a prototype, initial user evaluation and implement product and maintain. [72]

Gathering and analysing requirements is the first step. Analysis of needs is the first step in a prototyping model. The system requirements are precisely determined during this phase. Users of the system are questioned throughout the process to learn what they anticipate from it. [72]

A preliminary design, sometimes known as a fast design, is the second stage. At this stage, the system's basic design is created. It is not, however, a finished design. It gives the user a quick overview of the system. The prototype development is aided by the rapid design. [72]

A real prototype is created during the "make a prototype" stage using the data acquired during fast design. It is a scaled-down version of the necessary system. [72]

The proposed system is offered to the client for an initial evaluation during the initial user evaluation stage. It is useful to identify the working model's advantages and disadvantages. Customer feedback and suggestions are gathered and given to the developer. [72]

In the refining prototype stage, if the user is unhappy with the present prototype, you must modify the prototype in accordance with their comments and recommendations. This phase won't be complete until all of the user-specified requirements have been satisfied. A final system is created based on the final prototype that has been accepted once the user is satisfied with the developed prototype. [72]

The stage of product implementation and maintenance is the last. After being created using the final prototype as a base, the system is thoroughly tested before going into production. Routing maintenance is performed on the system to reduce downtime and guard against catastrophic failures. [72]

3.2.4 Choice of Development Methodology

Based on the research conducted, it has been decided that the prototyping development methodology will be used for the project because of the process of gathering requirements, creating a design based on these gathering of requirements to then building a prototype, getting feedback from users, refining the prototype (or noting them as future work) and then completing the final implementation and maintaining has been used throughout the project.

The waterfall model is more of a software development model rather than a prototyping model, the involvement of users is only at the beginning, rather than high within the prototyping model. The complexity of an error increases as the nature of the model each phase is sequential of the other, and flexibility to change in waterfall model is difficult. [73]

Agile is less suited for the project as that if changes need to be made, it could cause issues with the cycle in which the agile methodology works, it is more aimed at

teamwork, and if tasks get delayed it could create a large series of sprints, and cause time constraints. [74]

3.3 Using Bluetooth Beacons for the UWTSD Self-Guided Tour Application.

The UWTSD IQ building is a multi-story structure with a variety of venues including computer rooms, classrooms, student services, and private study areas. Even if there is no link to any place for it to appear on the app, extra beacons may still need to be added inside the structure due to its height and size. This would guarantee that the app would continue to work, and that the Bluetooth Beacon's signal would continue to be active and accessible on a mobile device. The alternative prototype implements QR codes.

3.4 Testing

A user experience (UX) interview is a practical testing technique. The interviews will be performed online using Zoom and Microsoft Forms, and there will be between 10 and 20 participants. This will assess a page's appeal by determining if visitors find it appealing. Users are learning how to utilise the app and checking the program's accuracy as a result. The effectiveness of the app would also be evaluated, including how easily the user could resolve any problems with the app without exerting more effort than necessary and how quickly the software would adapt to new tasks. The user would assess the app's dependability by being asked if they felt in control of the communications. Users' opinions on the design might be sought out. [75]

3.5 Data Collection

3.5.1 Pre-Implementation

To collect the requirements of the application, participants have been asked questions about whether they attended an open day whilst applying to study at university, whether they had campus tours and asked what types of features should be included on the app.

The Student Hwb at UWTSD have been interviewed to collect data on things such as what students ask whilst attending an open day at the university.

The Student Services (Learning Support) team at UWTSD have also been questioned to assist with the collection of requirements in preparation for the application. They were asked about what types of questions they get from open days and new students, general feedback on how a student feels on arrival at university and feedback on accessibility for all and wheelchair users as well as the most difficult thing for disabled students at present.

3.5.2 Post Implementation

Once the application has been fully implemented, further tests have been performed to make sure that the application works properly and that the objectives have been met. The benefit of testing is that the clients would be satisfied with the end-product as any bugs would be found early in the implementation and could be fixed before being released to clients. [76]

3.6 Limitations

Due to the time constraints of the project, it may not be possible to fully complete the final version of the app. The researcher has set a budget of £300 to spend on the project which will cover for any premium access of software, books, and equipment.

The researcher does have some prior experience of developing applications within Android Studio and the Java programming language however, due to not programming in the Android Studio software for some time, however both the software and programming language are regularly update to incorporate new features and functions, the coding becoming depreciated with older versions of software, which could cause some problems during implementation.

Due to the ongoing restrictions due to Coronavirus, testing would be done online. This cause come technological issues during the testing, especially for the wireframe diagram. Control was given to the participants of the researcher's screen which caused a security risk because, by doing this, this gave access to participants to the researchers' files which, if got to, could have caused significant delays in getting the report and software submitted on time.

GDPR is also a potential limitation with testing as details must remain secure and have to consent to have this personal details stored alongside with being recorded for the purpose of the interview.

CHAPTER 4 – PRE-IMPLEMENTATION MARKET RESEARCH

4.1 Existing Application Testing

4.1.1 The Gower Heritage Centre

The app guides users around the heritage centre, the users learn about the inner workings of the mill and then explore nearby sites, such as Three Cliffs Bay. Walking directions to nearby sites are provided in the app. The app uses Bluetooth Beacons to alert users to the key places of interest within the Gower Heritage Centre. It also makes use of GPS to guide users to places of Interest within walking distance of the Heritage Centre. [77]

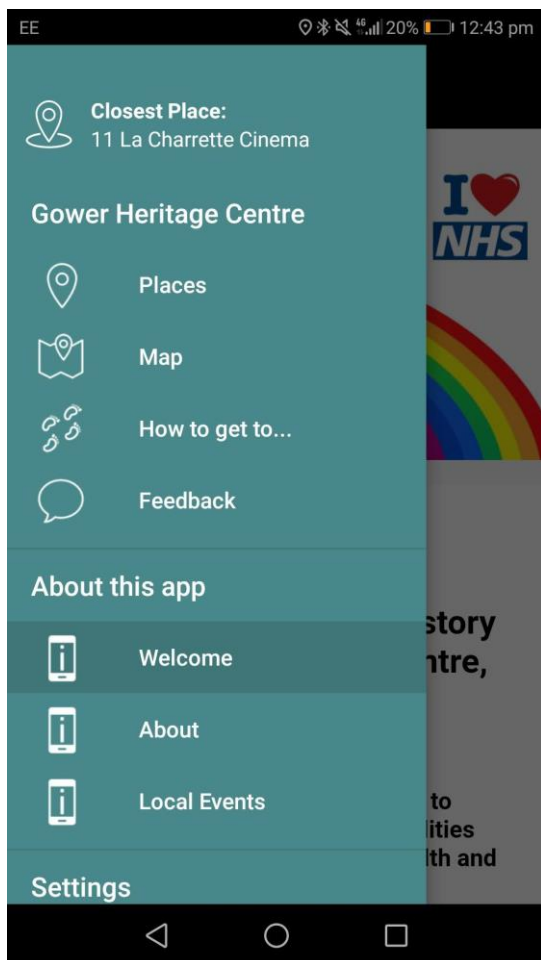


Figure 14: Welcome screen

Figure 14: Welcome screen (above) shows the first page users see when using the app as well as a sidebar menu. The menu on the side gives users the options to view the places without having to go near the beacon for that location, and users can view the places in order of wherever is nearest to where they are currently located (as shown in Figure 17: Places by Distance), so they could do their own self-guided tour if they wished.

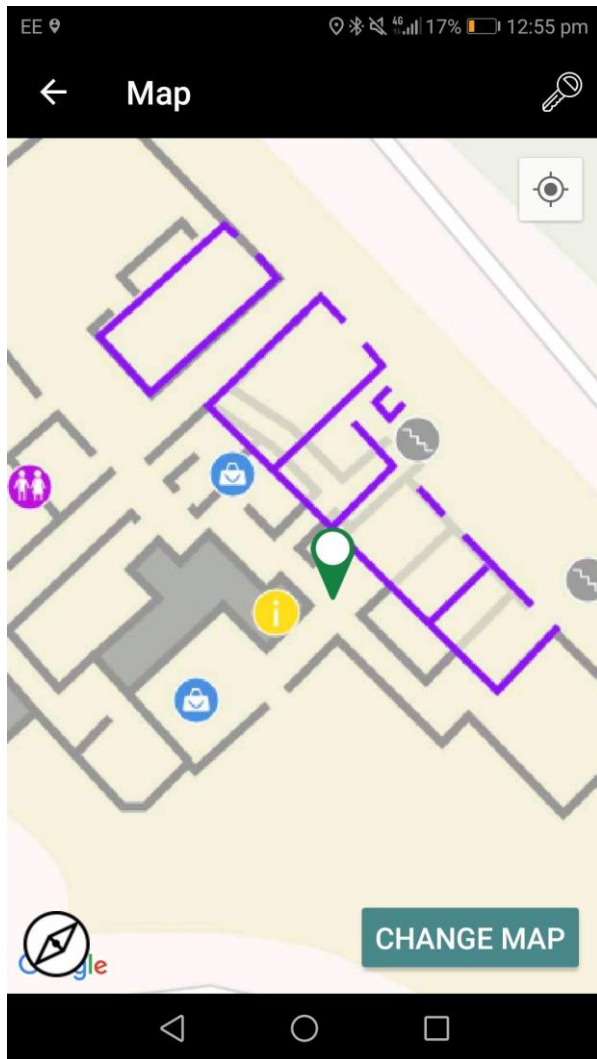


Figure 15: Map of the Heritage Centre

Figure 15: Map of the Heritage Centre (above) demonstrates one level of the Gower Heritage Centre. This was a particularly good feature as it shows where specific point, such as cafes, information points, and places of interests within the centre are located. There was a ground floor and first floor plan available on the app.

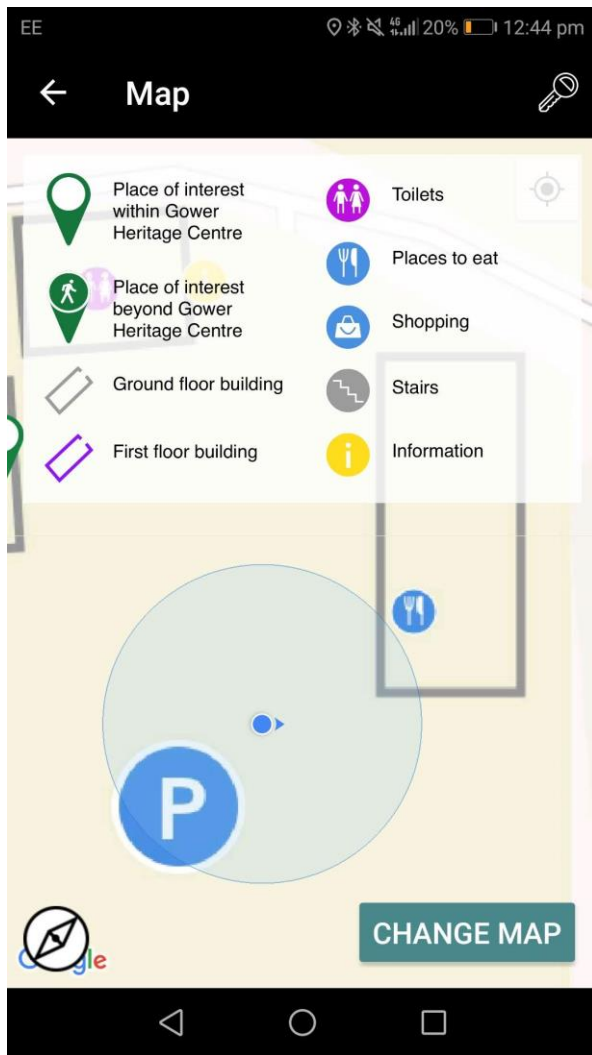


Figure 16: Key for the Map

Figure 16: Key for the Map (above) shows the key for the map. A key was provided for the map, which shows key areas such as car parking, food and beverage outlets, shopping, stairs, and many other types of landmarks within the centre. This is a good feature as, if users had not visited the Gower Heritage Centre previously, it would give you a good indication of where different landmarks and services are located.

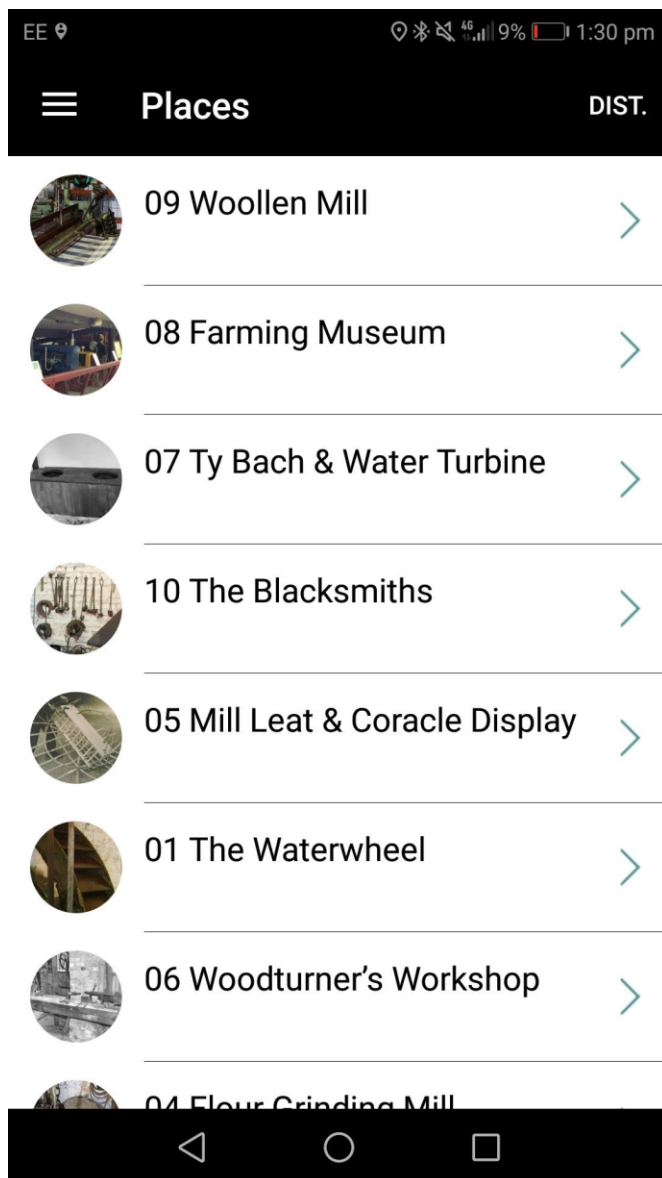


Figure 17: Places by Distance

Figure 17: Places by Distance (above) shows the places nearest to where the researcher was located.



Figure 18: App coincides with Actual Location

Figure 18: App coincides with Actual Location (above) shows that, while the researcher was at the woollen mill, the app in Figure 17: Places by Distance stated that the user was by the woollen mill. The problem with this is that it took a long time to update from the previous nearby location.

4.1.2 The Glynn Vivian Art Gallery

The app helps users to plan their visit and provides access to online content enabling a great understanding of the collections and exhibitions within the art gallery itself. The app enables users to zoom into high resolution images, read the exhibition description and listen to audio commentaries to expand knowledge, choose from a selection of tours, view upcoming events in the gallery and access videos and other additional content, including audio commentaries from curators, artists, and academics. [78]

One of the problems with this app is that the tours do not coincide with the room numbers within the art gallery.

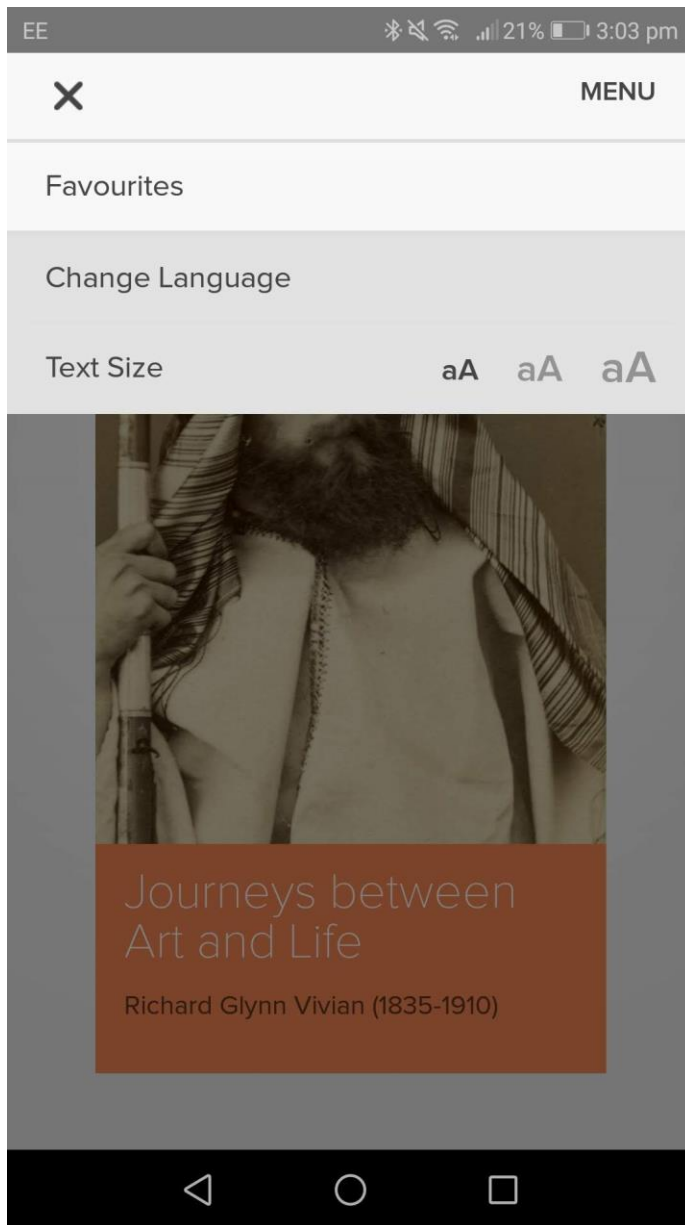


Figure 19: Home page with Accessibility Settings

Figure 19: Home page with Accessibility Settings (above) shows the home page of the app, with accessibility settings also being shown. This is an important aspect to the app, that it must be accessible to all users.

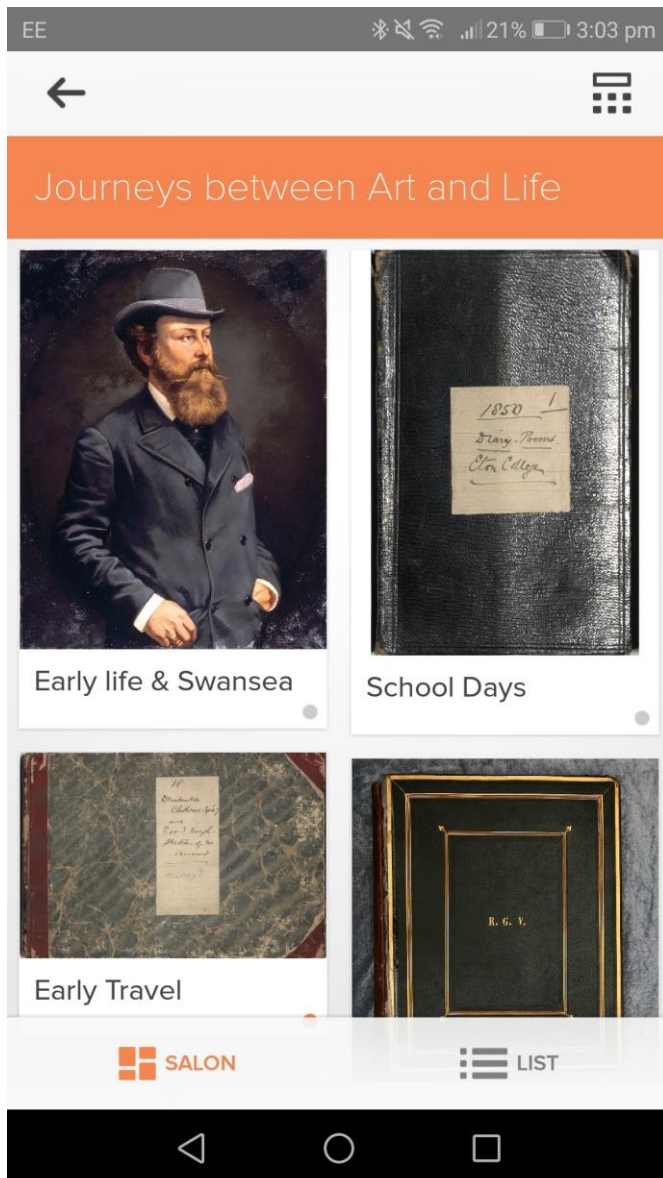


Figure 20: The Rooms

Figure 20: The Rooms (above) is the start page for the tour. It allows users to either pick the room they want to visit, or enter in the room number manually, as shown in Figure 21: Stress Test (below).

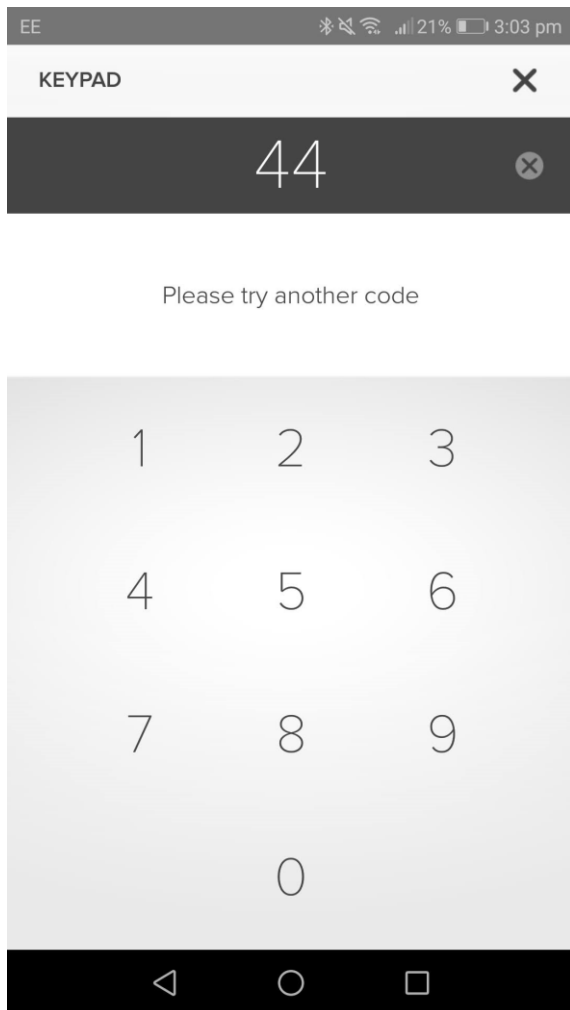


Figure 21: Stress Test

Some tests were performed whilst using the app, the room number '44' was entered, the researcher knew this number did not exist within the gallery but wanted to see whether an error message was produced. This test worked; an error message appeared.

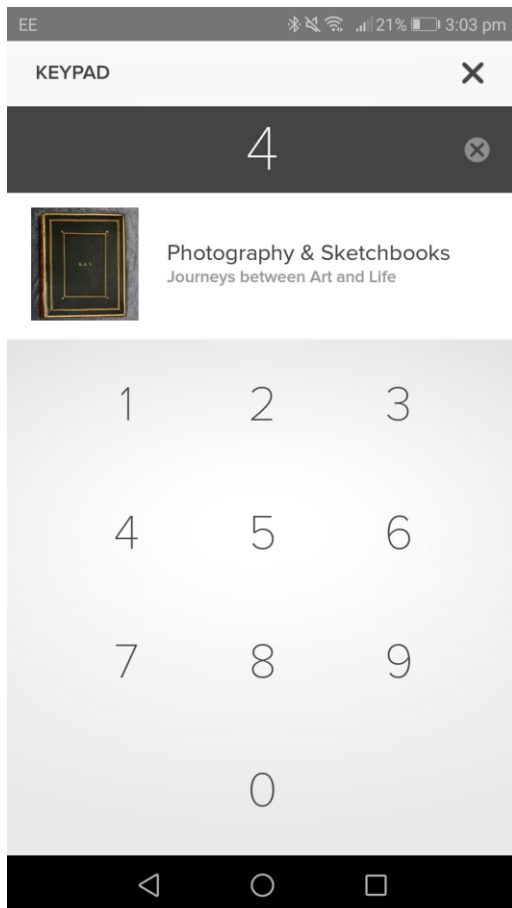


Figure 22: Incorrect Room Example

The main problem was that the room would provide users a number to be inputted into the application and the application would give a room, but it was not the correct room, it was not even similar. An example would be that if the room was room 4 (Photography and Sketchbooks) the app would have it as totally given exhibition.

4.2 Pre-Implementation Interviews

When the Student Hwb were questioned, they said that based on the purpose of the app, they think the main focus is where the facilities are, opening hours of the campus buildings and the Library as well as the accessibility of the facilities and buildings.

Student Services were then questioned, and their responses are below:

Some of the typical questions asked on open days are:

1. Where are Student Services?
2. Where is Learning Support?
3. Where is the Finance Office?

4. Where are the toilets located?
5. What support can I access?
6. Is there campus accommodation? Where is student accommodation located?
7. Is there disabled parking? Do I need a parking permit and how to access it?
8. What time does the IQ building close?
9. Where is the Library located?
10. Where is the café? Are there other places I can access food and hot drinks on campus?
11. How do I get hold of lecturing staff?
12. How do I borrow a laptop?
13. Where do I get my Student ID / Library Card?
14. Who are the Hwb? What do they do?
15. Is there a student communal room? Is there a quiet space?
16. Where is the Prayer Room?
17. Where are the Fire Assembly Points?
18. Who can I talk to if I am struggling, and need help or feeling down?
19. Where do I access my timetable?
20. Can I access a standing desk?

Whilst conducting the pre-implementation interviews, many of the participants concurred with what both Learning Support and Student Hwb teams said such as opening and closing hours, what support is available, where is the accommodation and where is the prayer room.

Some general feedback on how a student feels on arrival at university is that they can often feel overwhelmed and a bit lost. Home sickness is common in the early months. They can also feel incredibly lonely, can experience a culture shock and struggle with language if English is not their first language.

The feedback on accessibility for all and wheelchair users is that the disabled car park is difficult to navigate with the stone chippings, in term time the staff / student car park can be difficult to find parking spaces, there are no automated doors internally

and there is currently no standing desk in learning support. The lifts are small to navigate a wheelchair, from the lift there are 5 heavy doors to open to get to learning support and the internal doors are heavy and difficult to open if you have poor mobility or in a wheelchair.

The most difficult thing for disabled students at present is that there is a lot of form filling for disabled to access support, which can be particularly difficult for students with Specific Learning Difficulties (SpLD).

It is hoped that on open days, a student can go by a particular lab and have a rundown of what is being taught in that room. In the library, the app informs the user of the opening times. Student Services would also be interviewed and be questions on what students with disabilities what the most difficult thing for them is presently at the University. For those students who are wheelchair bound, what is lift access like and are they accessible to those students with disabilities.

4.2.1 Interview Results

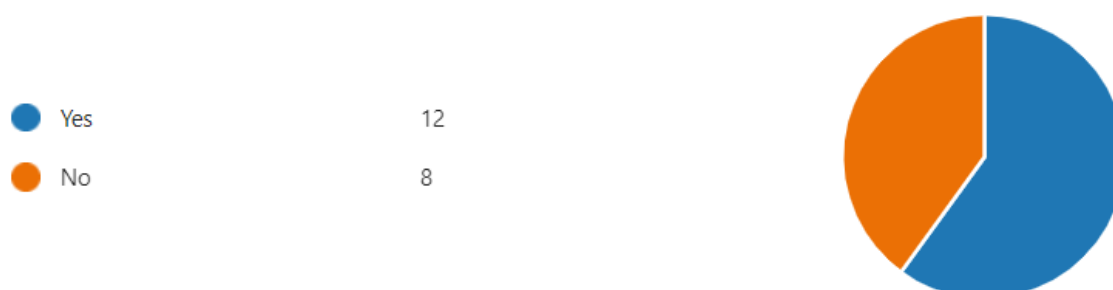


Figure 23: Attending an Open Day prior to commencing studies.

Figure 23: Attending an Open Day prior to commencing (above) shows whether participants attended an open day prior to commencing at university. 60% of participants said that they had attended an open day, whilst 40% said that they had not attended an open day prior to starting University. The participants who were unable to attend an open day shared their thoughts and said that there were video tours available online so did not feel the need to attend an in-person event, some of the open days were fully booked so were unable to attend, coronavirus restrictions meant that in-person open days could not take place for some participants, instead they were virtual.

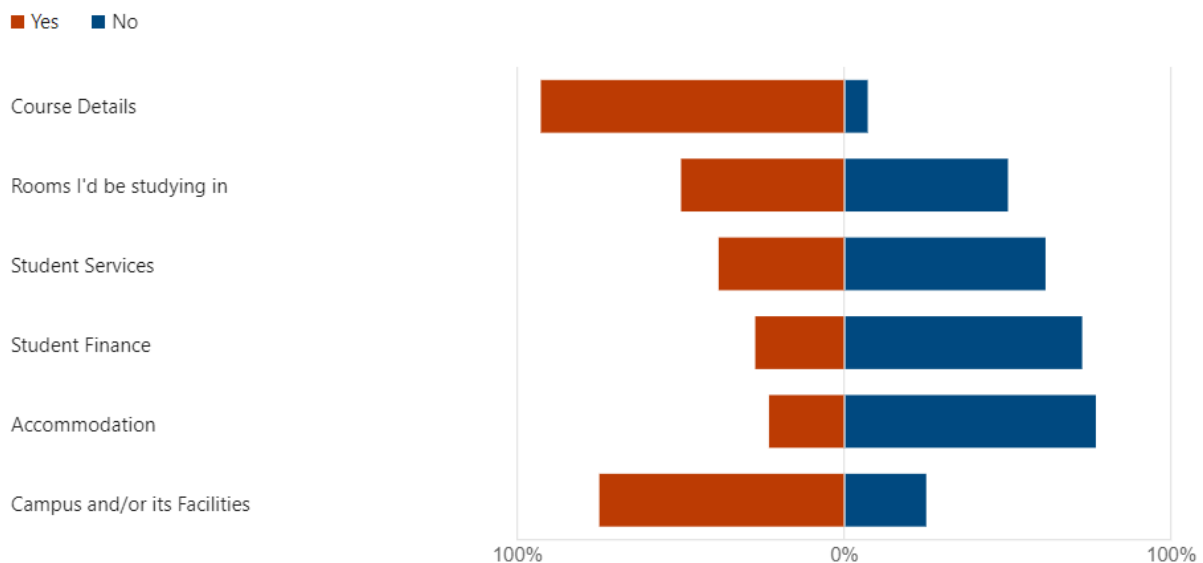


Figure 24: finding information.

Figure 24: (above) shows what information applicants wanted to find out about whilst attending a university open day. 92.9% of applicants wanted to find out details about the course they were applying for, with 75% wanting to find out about the campus they would be studying on as well as the facilities the campuses could provide. 38.5% wanted to find out information about the support available from Student Services, with 23.1% wanting to view, or access, potential accommodation and 27.3% finding out information about Student Finance.

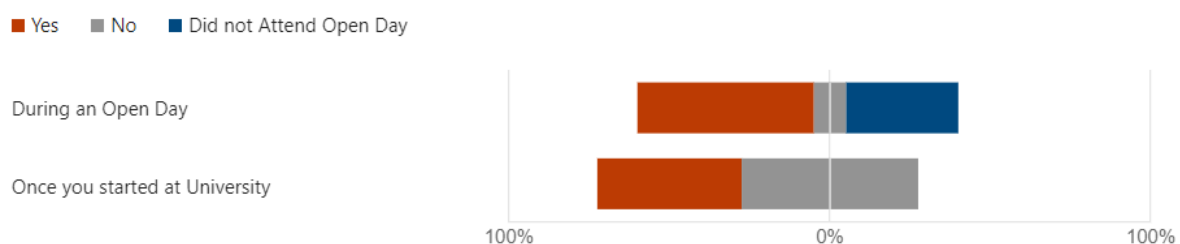


Figure 25: were campus tours provided?

Figure 25: were campus tours provided? (above) shows the results of the question to participants 'were you provided with an open day as part of either an open day and/or once the applicants had started at the University. 55% received a tour of the campus during an open day, and 45% of students receiving a campus tour once they had started at the University. The Coronavirus pandemic has impacted this result as open days were not being performed with the regularity that they did pre-covid.

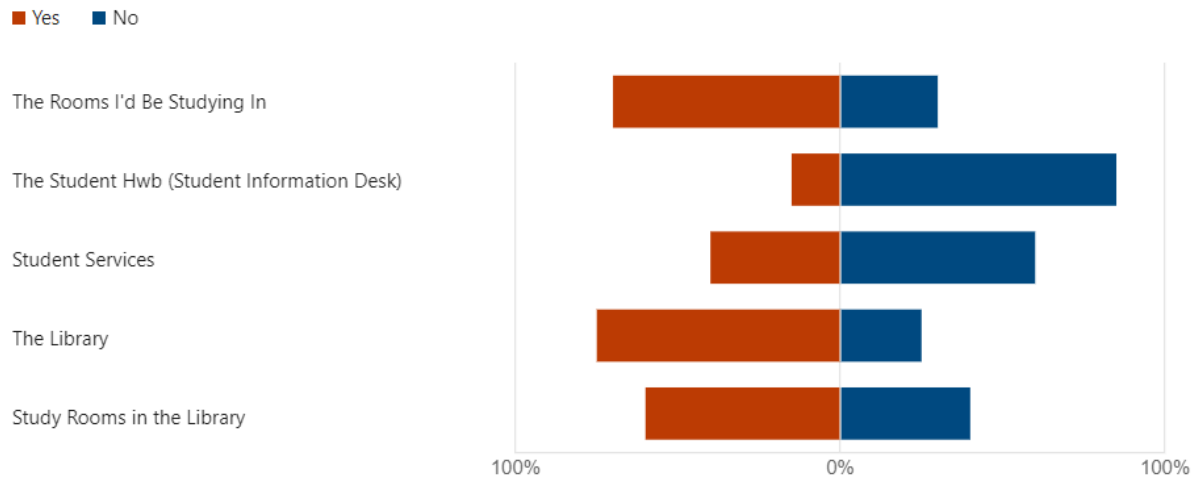


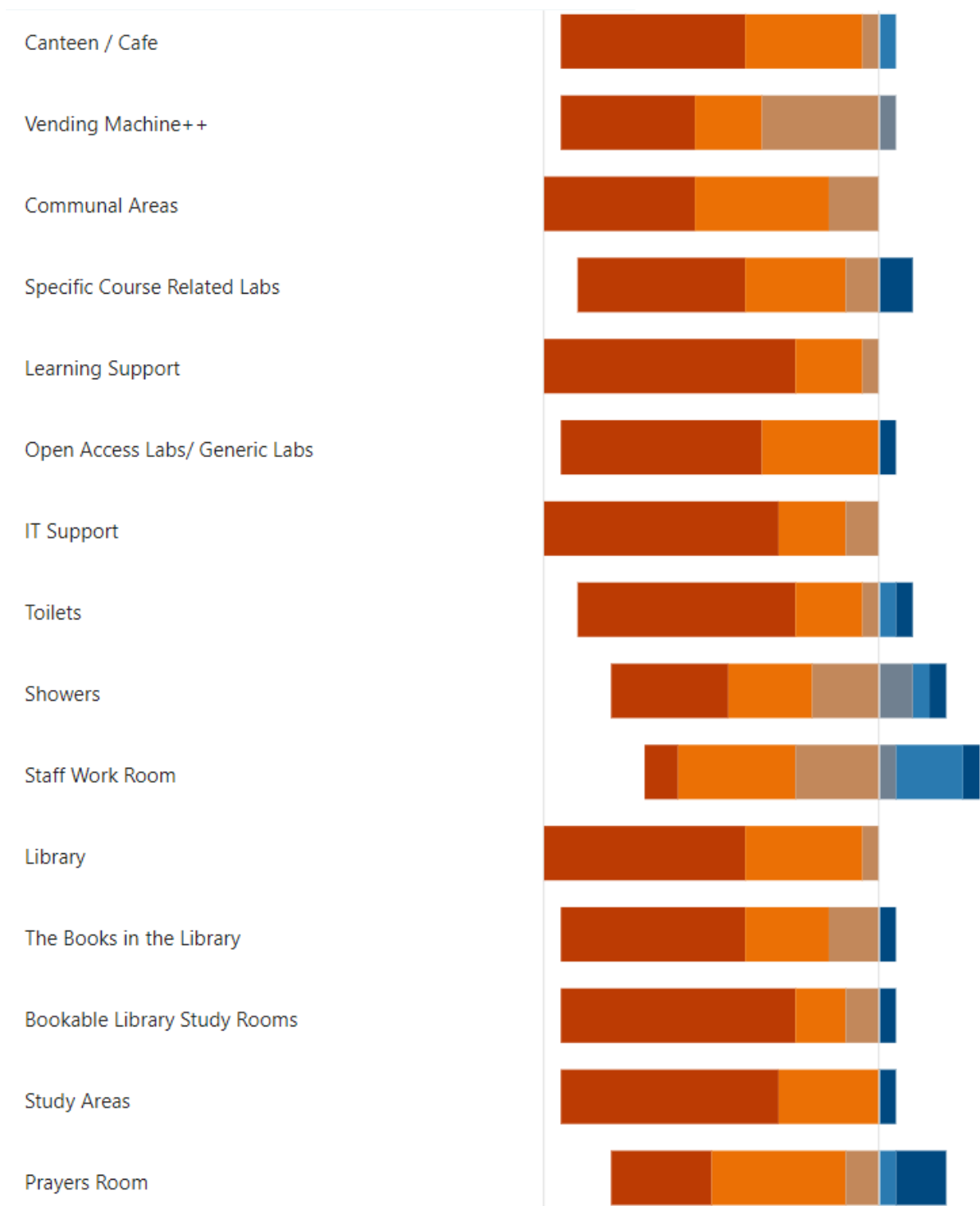
Figure 26: what did you see on a campus tour?

Figure 26: what did you see on a campus tour? (above) shows the results for the question 'what did you see on a campus tour'. 75% of participants said that they saw the library as part of the tour, with 70% of participants said that they saw the rooms they would be studying in. 60% of participants said that they saw the study rooms within the library, 40% of participants saw where Student Services are located and 15% of the participants saw the Student Hwb desk.



Figure 27: would participants feel that the app would benefit from?

Figure 27: would participants feel that the app would benefit from? (above) shows the results for the question 'would you benefit from using this app'. 90% of participants feel that the app would be beneficial for them, with 10% thinking not.



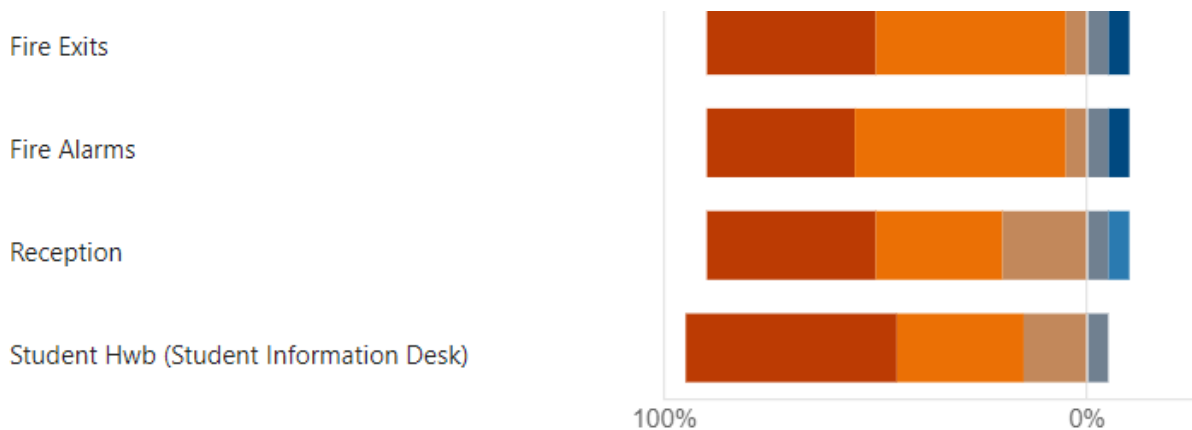


Figure 28: potential features and places to include

Figure 28: potential features and places to include (above) shows what features the participants would want to see on the app. 55% of participants asked strongly to feel that the café would be beneficial to have on the app, with 40% feeling that vending machines would be beneficial. 45% of participants strongly feel that the communal areas should be included on the app, with 50% of the participants feeling that specific course labs would be popular and 75% of participants wanting to see information on Student Services on the app. 60% of participants would want to see where the open access labs are located and 70% of participants would find it beneficial to have the location of IT support on the app. 65% of participants would want to see where the toilets are located on the app, with just 35% of participants requiring to know where the showers are within the campus building. 60% of participants would want to see where the library is, with 55% wanting to see where the different categories of books are within the library and 70% wanting to know where the library study rooms are. 65% of participants would want to know where the campus-based study rooms are, with 30% wanting to see where the prayer room was based. 25% of participants strongly disagree that technicians office should be on the app, and 20% of participants saying that they do not feel that the staff work room should be on the app. 40% of participants feel that the location of fire exits should be demonstrated on the app, with 35% of participants feeling that fire alarms, especially in disabled toilet facilities, should be included. 40% of participants feel that the reception desk should be highlighted on the app, with 50% of participants feeling that the Student Hwb desk should be shown on the app.

Some participants gave specific feedback on this question, which have been listed below in Table 7: feedback from users regarding potential app features.

Area	Feedback
Café and Catering	The options for dietary requirements should be shown and the microwaves should be made visible. Information such as opening times, menu, where it is and how long the queues would be beneficial.
Student Services	This could show what type of support is available, and what they can support you with.
Library	The click and collect facility should be advertised.
Offices (staff and technician)	Useful for new students.
Open Access Areas	Timetables for when lectures are taking place and when the rooms are available to use.
Specific course labs	Get an idea of what technology have on each of the computers. Would be useful to know, but not beneficial for a
Library	Opening times and availability of study spaces and rooms within the library.
Printers	Would like to see where the printers are located
Other features	Would like to see where the water fountains are located as well as the stairs and lift. Disabled access, such as ramps.

Table 7: feedback from users regarding potential app features

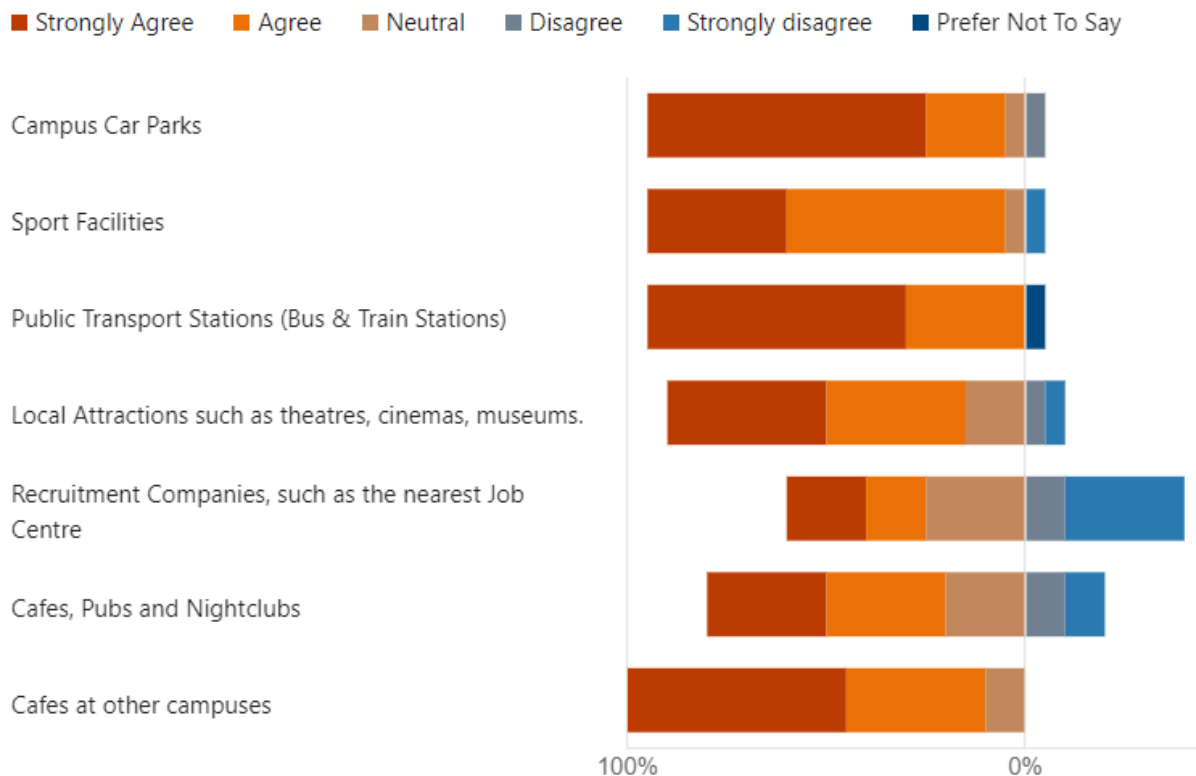


Figure 29: future developments

Figure 29: future developments (above) shows the chart of what participants would like to see during future implementation. The researcher expressed to participants that there would be some features that would be implemented in future works, and not during the time frame of the project. Features were put forward to participants, and the outcome is shown in the chart above. 70% of participants strongly feel that the campus car parks should be made available, with 35% finding it beneficial to have the sports facilities discoverable. 65% strongly feel that the nearest public transport stations, such as the nearest bus and train stations, should be implemented on the app as future work and 40% of participants thought that local attractions, such as theatres, cinemas and museums, should be made available. 30% of participants feel that it is not necessary to have recruitment companies, such as the nearest job centre, on the ap and 30% of participants favour the nearest cafes, pubs and nightclubs to the campus. 55% of participants strongly feel that having cafes at other campuses should be visible on the app.

Some participants gave specific feedback on this question, which have been listed below in Table 8: future work suggestions from participants.

Area	Feedback
Cafes, Bars and Nightclubs	Refer to the UWTSD Student Union bars, cafes, and nightclubs.
Public Transport Stations	Beneficial to those students located in the Swansea Business Campus.
Local Attractions	Useful for students, not for staff unless they have relocated.

Table 8: future work suggestions from participants

■ Yes ■ No

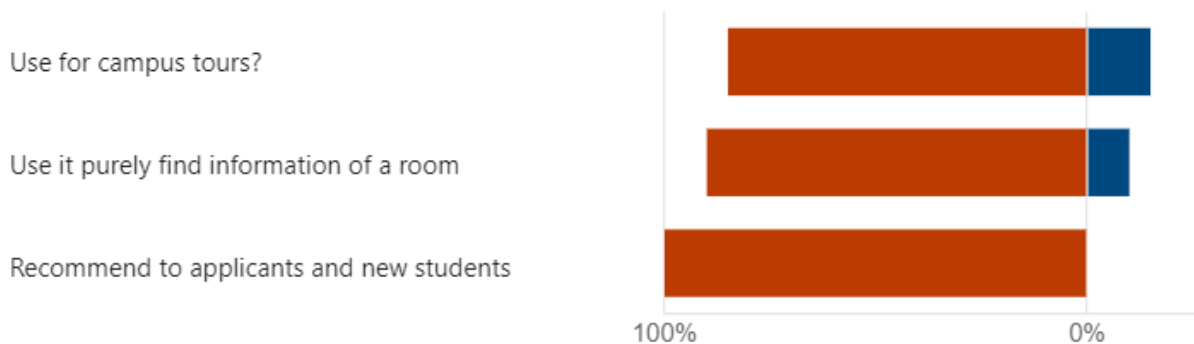


Figure 30: using and recommending the app

Figure 30: using and recommending the app (above) shows the responses for the final three questions. 85% of participants would use the app for campus tours, with 90% of participants using the app to find out information and rooms. 100% of participants would recommend the app to applicants and new students.

CHAPTER 5 – DESIGN AND IMPLEMENTATION

5.1 Design

It has been decided to use the Java programming language for the project because it is the recommended language to use when developing applications. Java has many advantages over other languages, making it appropriate for most programming tasks. The benefits of Java include its ease of learning, ease of usage, and ease of writing, compiling, debugging, as compared to other programming languages. [79]

The researcher has recently studied the java programming language within the Object-Oriented Programming module at the University, so he will be able to use the skills and experience gained from this module to assist with the development of the application for the project.

The Android Development Studio software has been used because it “provides a robust analysis framework” and “it provides several quick fixes that help you address issues in various categories, such as performance, security, and correctness” [80].

The researcher also has some experience of using Android Studio, from previous studies 2 years ago whilst studying an application development module.



Figure 31: Home Page

Figure 31: Home Page (above) shows the proposed home page interface for the app. There would be the UWTSD logo and a menu on the header of all pages on the app, alongside a visual which would be included on each page of the app. For the home page, there would be an image of the external of the IQ building as this prototype will be based on the SA1 Campus only.

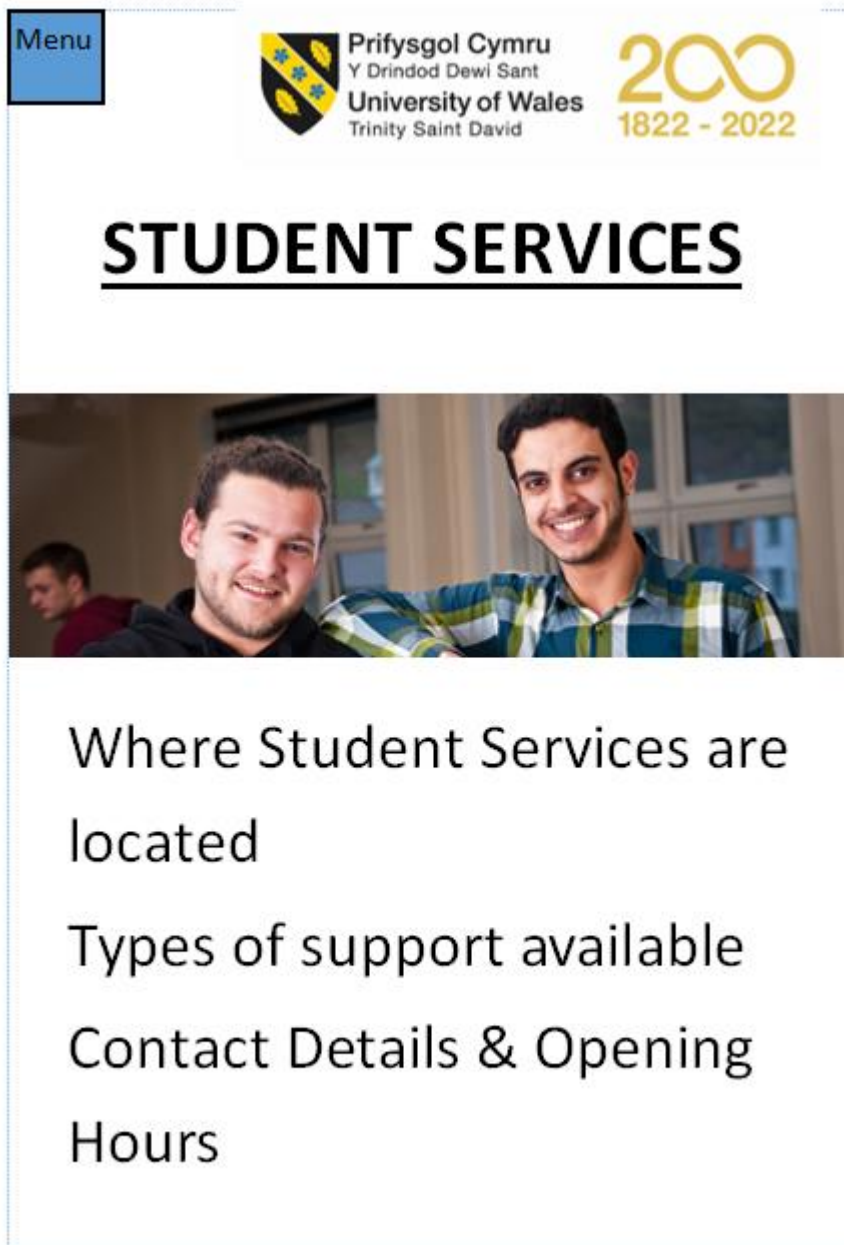


Figure 32: Student Services

Figure 32: Student Services (above) demonstrates how the individual room pages will look while the student is using the app whilst navigating their way around the IQ building. The illustrations shown above is for Student Services, but the other pages will be created in a very similar way, just the content and visuals would need updating.



CONTACT US



Address: UWTSD, IQ Building,
Swansea, SA1 8EW

E-mail: info@uwtsd.ac.uk

Website: [https://
uwtsd.ac.uk/contact-us/](https://uwtsd.ac.uk/contact-us/)

Figure 33: Contact Page

Figure 33: Contact Page (above) shows the layout of the contact us page on the app. It is hoped that, in future developments, when the app, covers more of the UWTSD campuses, that the image on this page will be a montage of all campuses in the UK. On the contact us page, it will include the telephone number and e-mail address for the SA1 campus, as well as a hyperlink to the contact us page on the UWTSD website, where it includes the addresses of the other UWTSD campuses.

The wireframe diagrams will be tested by interview participants, they will look at the diagram, navigate around it, and see how the app should work and point out any errors that are in the interface. They will also test buttons to see how they operate.

5.2 Implementation

The base code that has been used, was given as an exercise whilst learning about app development some years ago, it was a template to a QR Historic Landmarks app. This used QR codes, and there was a QR code assigned to each landmark on the app. On a practical level, the users would be in a location, and if they can see a landmark which has a QR code, they can scan the QR code, and they can view which landmark they have scanned on the QR code. They would then click on a button to open the landmark page, which includes detailed information and historical facts about that location. Alternatively, if users want to search whether there is a certain landmark on the app, or if the QR code does not work for any specific details, they can enter in the unique ID of the landmark or enter in the landmark name, and then the landmark page would appear on the app.

This base code was chosen because the concept of the idea was similar to what was aiming to be achieved, the main difference was that the room information would be accessed on the app using QR codes, rather than Bluetooth Beacons. On the base code app, there was a page about historical landmarks, providing some facts and information about each landmark. In this app that has been developed, it would read the QR code, direct you to its correct room page, and provide you with specific information, such as availability, support available, course information, and other information that is relevant to the room being shown on the app.

There are multiple classes and interfaces, many of which are in their own folders within Android Studio, making them more accessible to locate within the project files. This kind of encapsulation is a key component of the object-oriented approach as it allows modules from one project to be re-used in future developments.

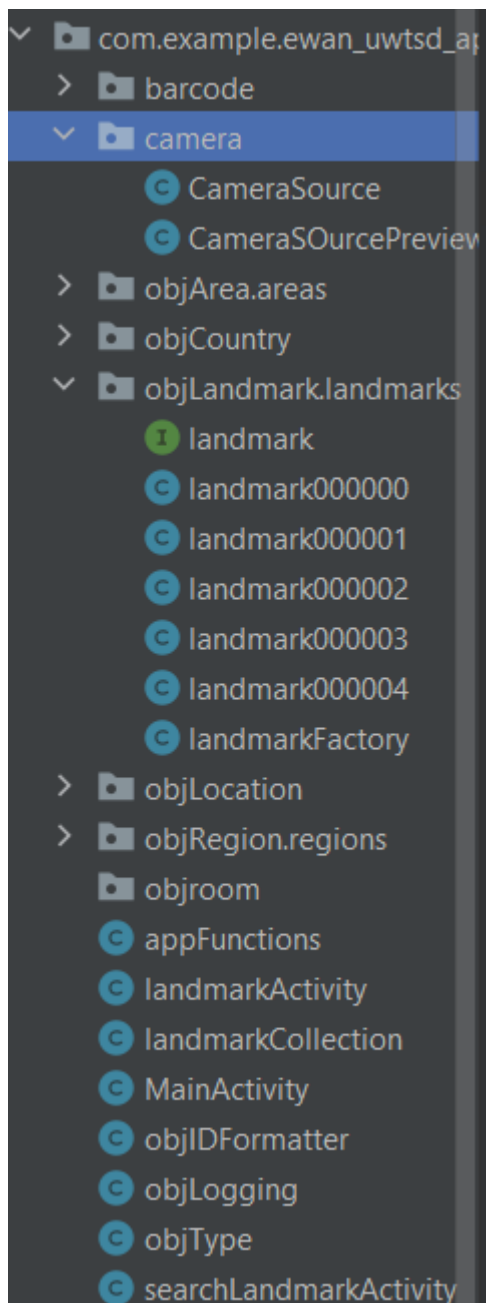


Figure 34: Solution Explorer in Android Studio

```

public abstract class CameraDevice implements AutoCloseable {

    /**
     * Create a request suitable for a camera preview window. Specifically, this
     * means that high frame rate is given priority over the highest-quality
     * post-processing. These requests would normally be used with the
     * {@link CameraCaptureSession#setRepeatingRequest} method.
     * This template is guaranteed to be supported on all camera devices.
     *
     * @see #createCaptureRequest
     */
    public static final int TEMPLATE_PREVIEW = 1;

    /**
     * Create a request suitable for still image capture. Specifically, this
     * means prioritizing image quality over frame rate. These requests would
     * commonly be used with the {@link CameraCaptureSession#capture} method.
     * This template is guaranteed to be supported on all camera devices except
     * {@link CameraMetadata#REQUEST_AVAILABLE_CAPABILITIES_DEPTH_OUTPUT} devices
     * that are not {@link CameraMetadata#REQUEST_AVAILABLE_CAPABILITIES_BACKWARD_COMPATIBLE}
     * BACKWARD_COMPATIBLE}.
     * @see #createCaptureRequest
     */
    public static final int TEMPLATE_STILL_CAPTURE = 2;

```

Figure 35: Camera2 Code [81]

Figure 35: Camera2 Code (above) was some code, which was on the camera2 hardware, was attempted. When created a number of errors then appeared which could not be resolved within the timeframe, so the code was removed from the app.

```

import android.Manifest;
import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.content.Context;
import android.graphics.ImageFormat;
import android.graphics.SurfaceTexture;
import android.hardware.camera2;
import android.os.Build;
import android.os.SystemClock;
// import android.support.annotation.Nullable;
// import android.support.annotation.RequiresPermission;
// import android.support.annotation.StringDef;
import android.util.Log;
import android.view.Surface;

```

Figure 36: importing the camera Android hardware

Figure 36: importing the camera Android hardware (above) demonstrates that the newest version of the camera functions, camera2, was imported however to no avail as this came back as an error message saying "cannot resolve symbol 'camera2'". Other errors were then discovered throughout the code to say, "cannot resolve symbol 'camera'". When attempting to resolve these errors, they were investigated and the main options were to either import the class, rename the reference or to create a constant field 'Camera' in 'Focus Mode' in 'CameraSource.' There was no actual class called camera, so did not feel that this would resolve the error and renaming the reference did not change the fact that it kept the name the same, so the error was not resolved. The create a constant field 'Camera' in 'Focus Mode' in 'CameraSource' fix was attempted and, although the recommended code came up, it still did not resolve the errors, as shown below in Figure 37: attempt of fixing the camera errors.

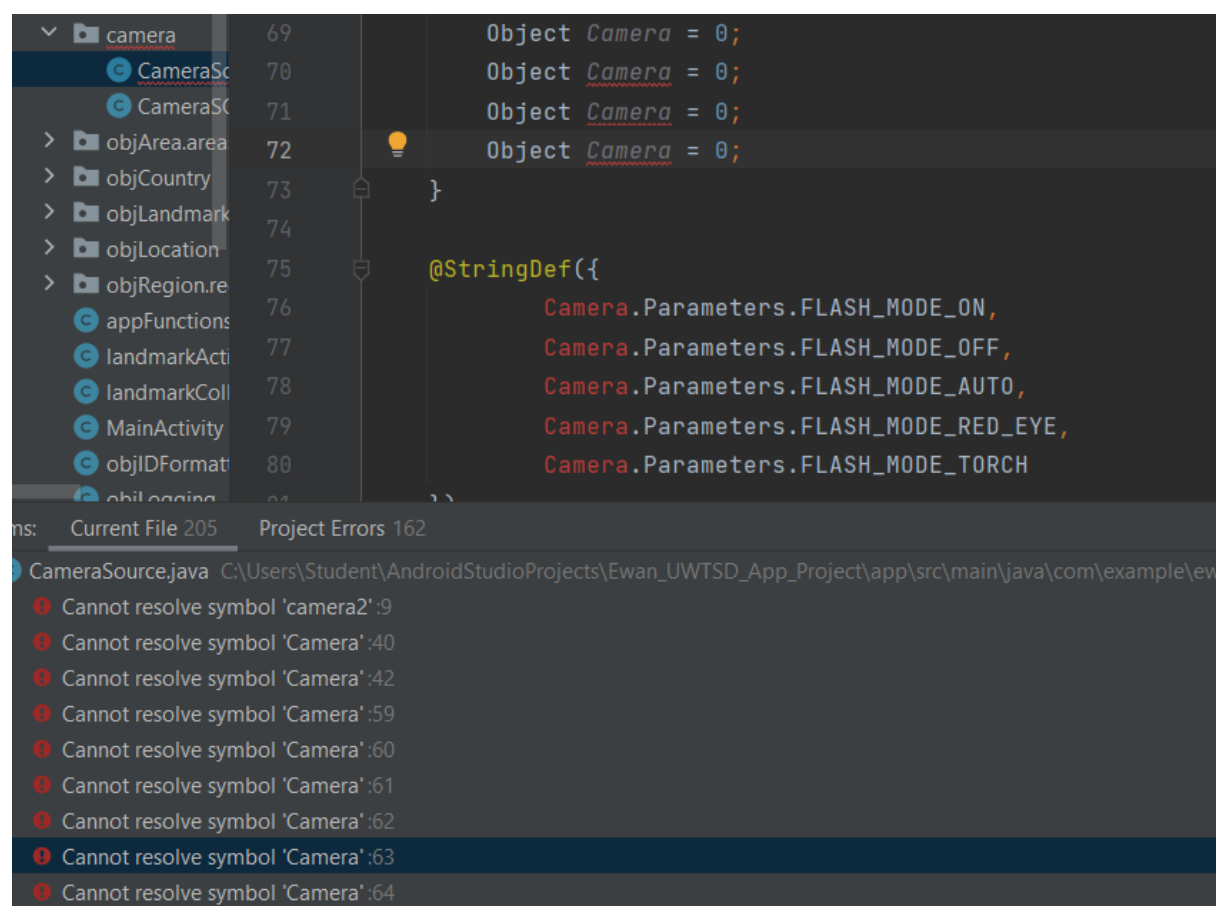


Figure 37: attempt of fixing the camera errors

```

public class landmarkFactory {
    // Declares string for log tag. Used for debugging purposes.
    public static final String LOG_TAG = landmarkFactory.class.getSimpleName();

    // Declares static landmark method to create new landmark object.
    // Parameter: Landmark ID.
    public static landmark newLandmark(int landmarkID) {
        // Switch compares landmark ID to cases..
        switch(landmarkID) {
            // If landmark ID matches the following cases, then creates one of the following landmark objects.
            case 1:
                return landmark000001.createLandmark();
            case 2:
                return landmark000002.createLandmark();
            case 3:
                return landmark000003.createLandmark();
            case 4:
                return landmark000004.createLandmark();
            // If landmark ID does not matches above cases, then returns default/base landmark object.
            default:
                return landmark000000.createLandmark();
        }
    }
}

```

Figure 38: base code for the landmark factory class

Figure 38: base code for the landmark factory class (above) shows the base code for the landmark factory class. This code had been written to read each room ID, compare them, and if it matches then create the object for the room. If the ID of the room does not match a room already created, it returns the default object which, in this case, is 000000. Whilst testing with this code, it was causing the application to crash and close unexpectedly, so the code was changed to something much smaller, as shown in Figure 39: new landmark factory code.

```

package com.example.ewan_uwtsd_app_project.objLandmark.landmarks;

import com.example.ewan_uwtsd_app_project.objLandmark.landmarks.*;

public class landmarkFactory {
    public static landmark newLandmark(int landID) { return null; }
}

```

Figure 39: new landmark factory code

The new landmark factory is much shorter than the original code, and tells us the same thing, it can read the landmark (room) ID and if there are none that matches the exact ID, it returns as no value.

```

public class appFunctions {
    // Declares static String method to get object ID.
    // Parameter: Object type enum value.
    public static String getObjectID(objType obj) {
        // Switch compares passed object type enum value to cases.
        // If object type is landmark, then..
        if (objType.Landmark.equals(obj)) { // Returns the following.
            return "landmark";
        } // If object type is area, then..
        else if (objType.Area.equals(obj)) {
            return "area";
        } // If object type is location, then..
        else if (objType.Location.equals(obj)) {
            return "location";
        } // If object type is country, then..
        else if (objType.Country.equals(obj)) {
            return "country";
        } // If object type is region then..
        else if (objType.Region.equals(obj)) {
            return "region";
        } // If object type does not match cases, then return empty string.
        return "";
    }
}

```

Figure 40: changes to the app functions java class

Figure 40: changes to the app functions java class (above) shows the main changes done to the app functions java class. In the base code, it compared the past object type enum values into cases and if then the object type is landmark, would return landmark and if the object type is area, then the area case would return area. This repeats for the location, country, and region. For the app, if the object type equals to landmark, it returns the landmark. Else if, if the object types equal to area, it returns the area. Same as the base code, this is repeated for the location, country, and region. The main difference is that rather than using cases, it uses an else if statement.

The changes, as mentioned above, were all carried out within the project time frame however, many issues appeared during development, such as unused import statements and depreciated code for the camera code. Due to these errors, and what the functions were doing, the app could not be fully completed within the project time frame.

```
class showDialog implements View.OnClickListener {  
    @Override  
    public void onClick(View view) {  
        // Calls method to check search results.  
        DialogFragment dialog = new MainActivity.MyDialogFragment();  
        dialog.show(getSupportFragmentManager(), tag: "MyDialogFragmentTag");  
    }  
}
```

Figure 41: alert dialog

Figure 41: alert dialog (above) shows the alert dialog that was created within the Main Activity java class. The code above was also inserted into the search landmark activity java class and was going to be used to send out a notification of where the user was within the IQ building. In an attempt to get the code to work, an alert dialog was made so then if a valid ID number and/or room name was entered into the search, an alert dialog would become visible to say that you were at a certain room. This was attempted but failed upon testing.

Current Version	Old Version
Android Gradle Plugin Version 7.2.2 Gradle Version 7.3.3	Android Gradle Plugin Version 5.0.0 Gradle Version 5.0.0

Table 9: Gradle version

Table 9: Gradle version (above) shows that it was attempted to revert to an older Gradle version. When switched to the older version, an error message appeared saying that the specific Gradle distribution does not exist. Although changing the Gradle version, it is still showing the code as depreciated, so this test has not been successful. The Gradle version has now been reverted back to the current version. Changes made to update the code for new version of Android Studio.

With the unused import statement errors, the files commonly affected by this was within the objLandmark folder within the repository. Within other java files, it was not discovering any of the interfaces of classes within the landmark folder, essentially with and of the landmark code not working at all, which was causing the individual rooms not to show when the QR code was being scanned or when the room was being searched.

5.2.1 Prototypes

5.2.1.1 Prototype 1 – Wireframe diagram of interface

Usability testing will be conducted on the interface, to ensure that all the buttons direct the user to where they should do, and to make sure that the interface and the content of the app are what the expectations of the app are.

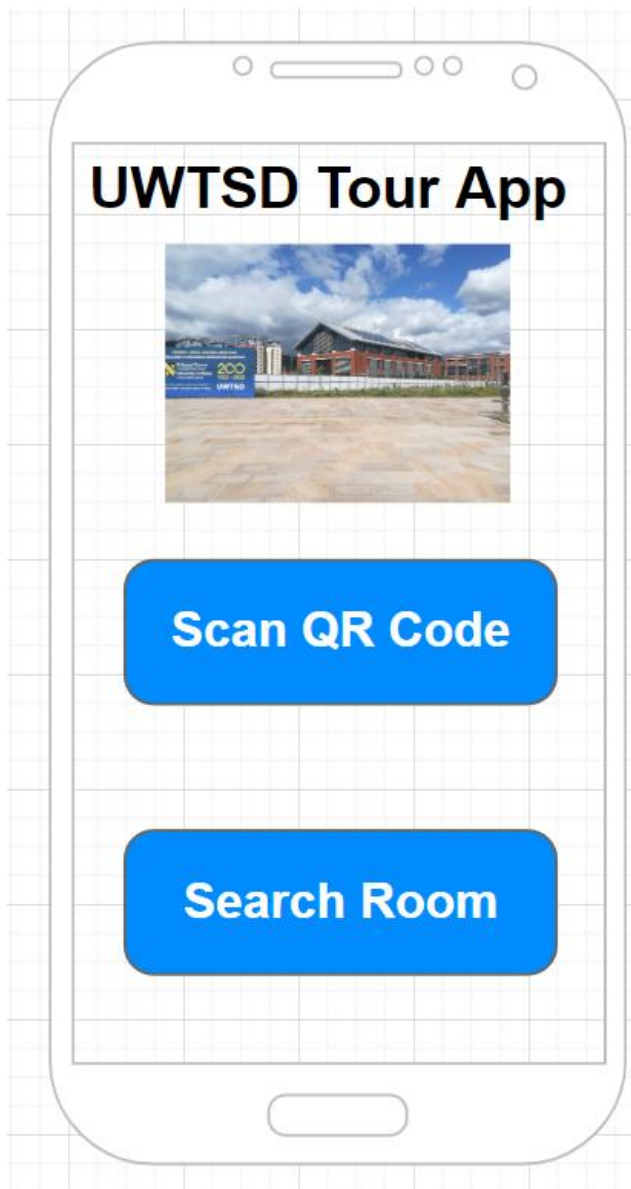


Figure 42: home page (wireframe diagram)

Figure 42: home page (wireframe diagram) (above) show the home page that was created for the wireframe diagram. There is an image of the IQ building, as planned within the design stage, with two buttons to scan a QR code or to search for a room.

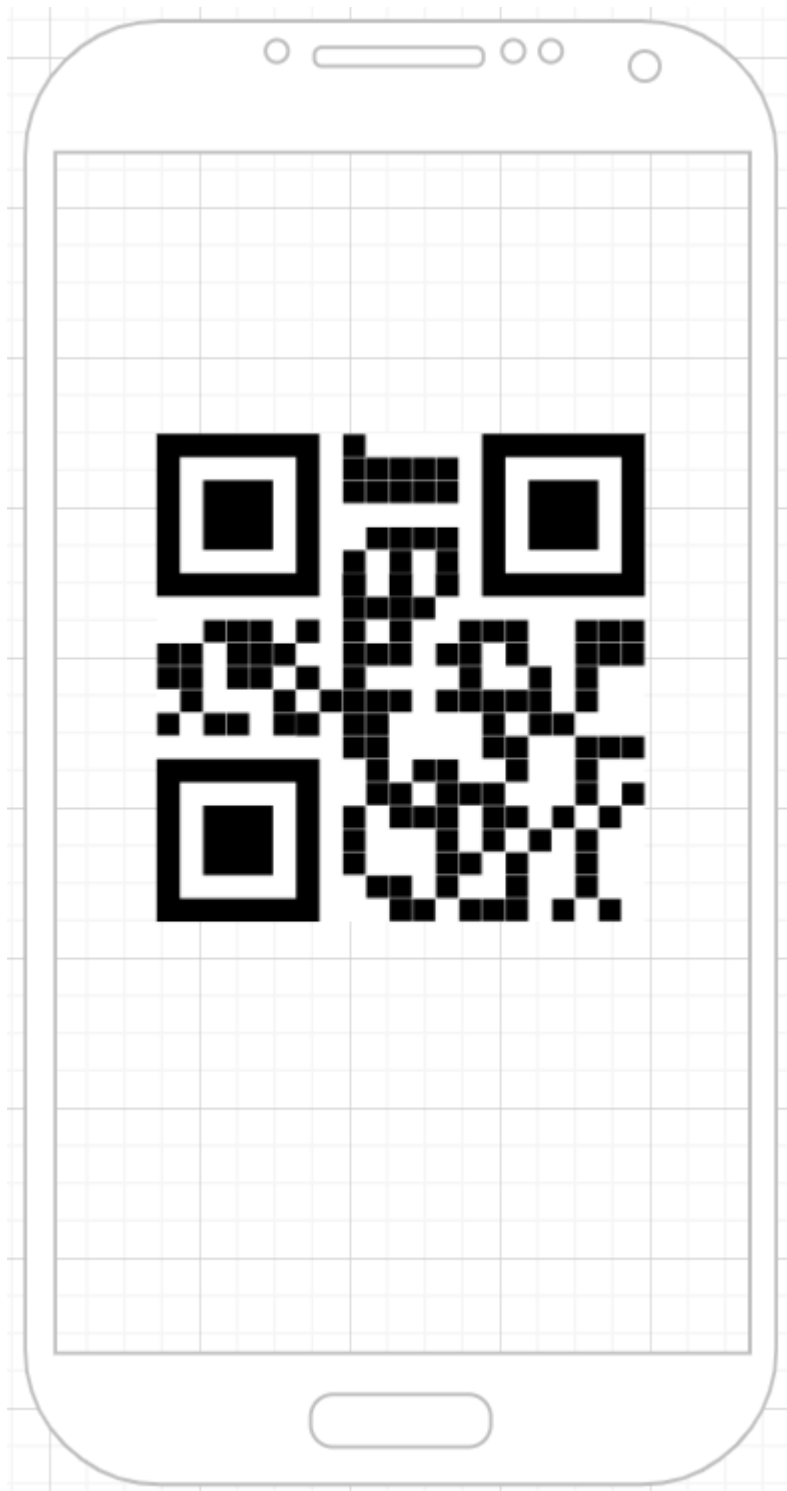


Figure 43: QR code scanner (wireframe diagram)

Figure 43: QR code scanner (above) shows how the QR code page looks like on the wireframe diagram. With the wireframe diagram, users are unable to scan a physical QR code as the software does not provide camera functionality, a QR code was inserted for demonstration purposes so then users could click on the QR code and then take them to the room page it was linked up to.

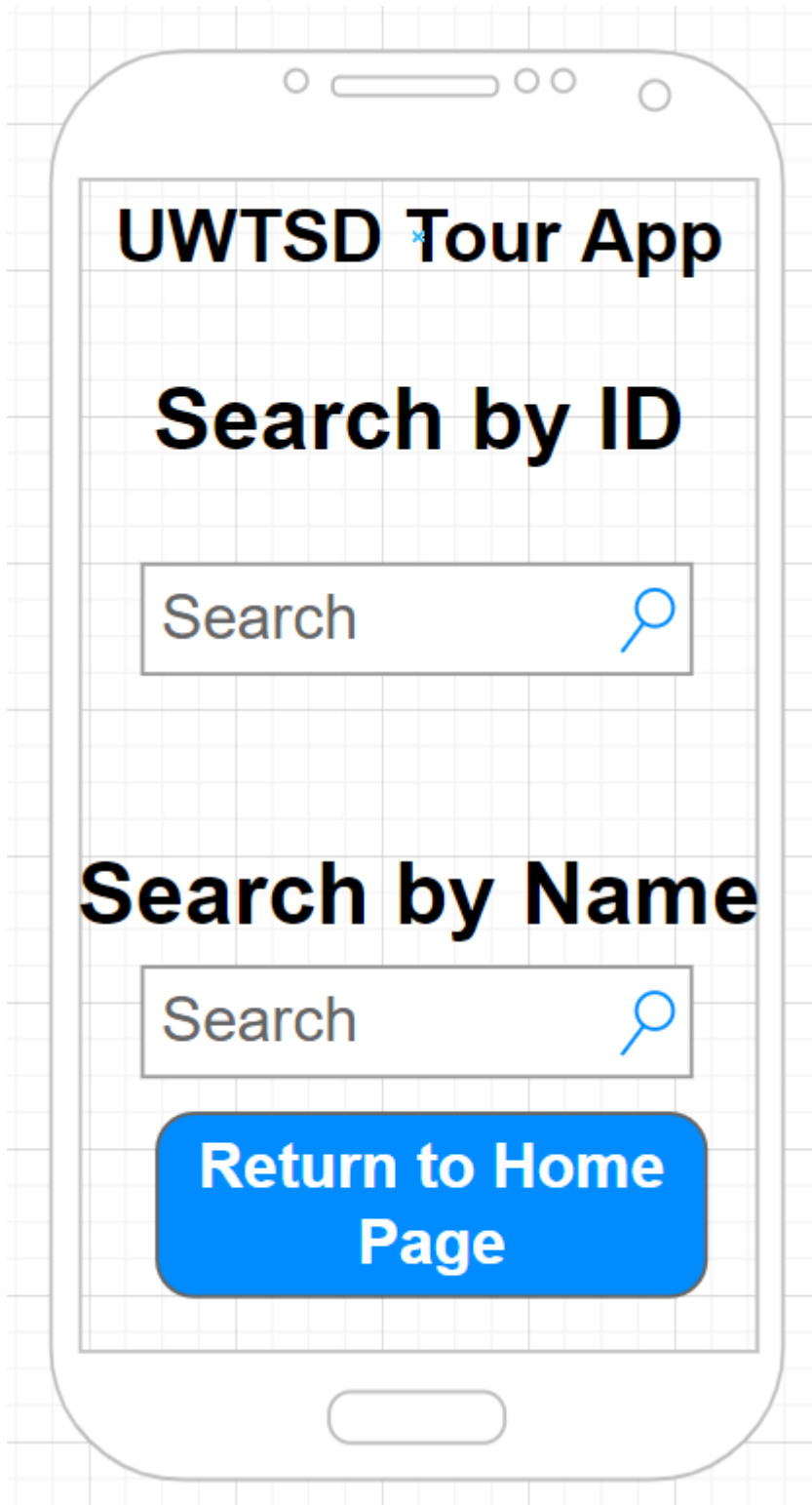


Figure 44: search page (wireframe diagram)

Figure 44: search page (wireframe diagram) (above) shows how the search page for the different rooms look on the wireframe diagram. By using the 'Search By ID' search box, users will be able to search for a room by its ID number assigned on the app, and by using the 'Search by Name, users will be able to search for a room by its name.

The user can then click on its relevant search button and then if the ID number and/or name matches what has been created on the app, users would be directed to that page.

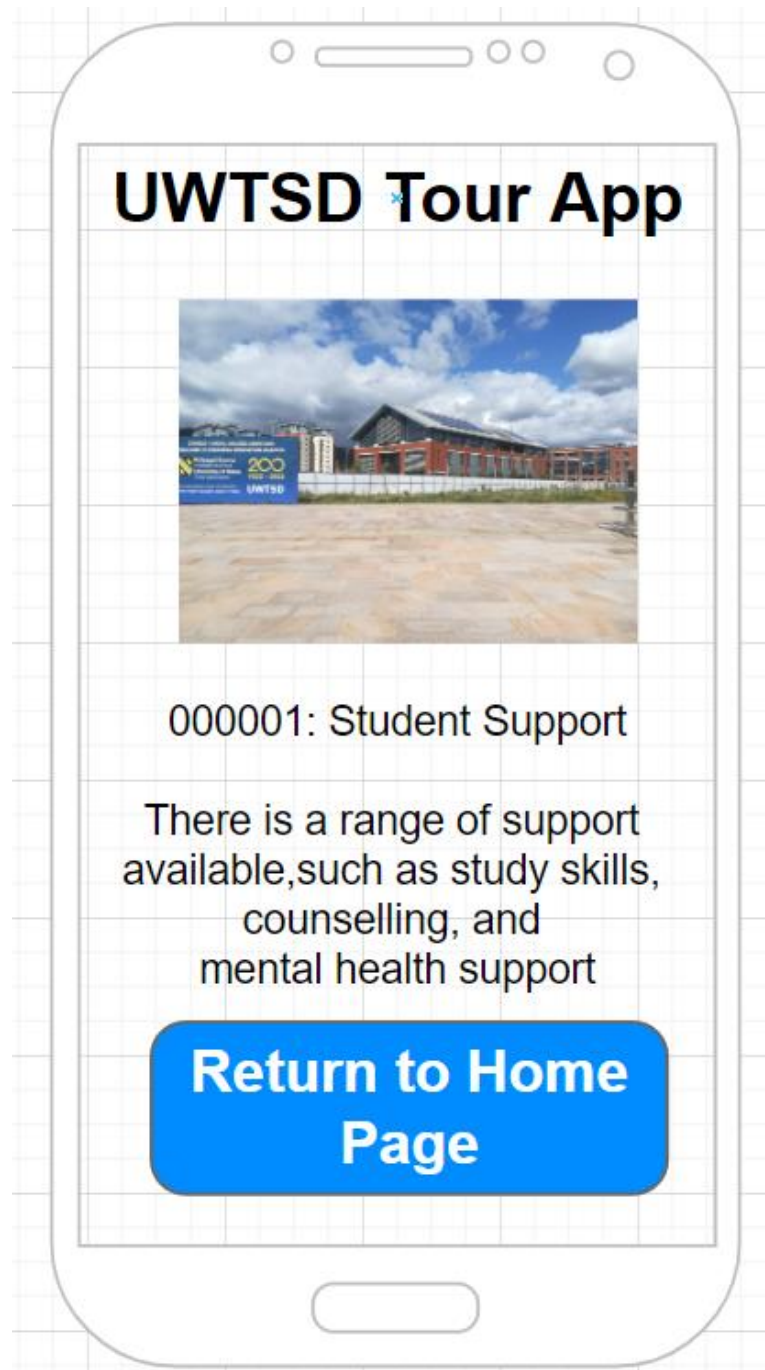


Figure 45: room page (wireframe diagram)

Figure 45: room page (wireframe diagram) (above) shows the individual room page on the wireframe diagram. This is what each room page layout looks, for the purpose of the prototype the student support page is being demonstrated and tested. On the page itself, it includes an image of the campus of where the Swansea student support

offices are based, what support is provided and a 'return to home page' button has been created. The room number itself has not been placed on the app, as the QR codes would be placed outside each room, hence why they are not included on the app.

5.2.1.2 Prototype 2 – initial app using Ionic

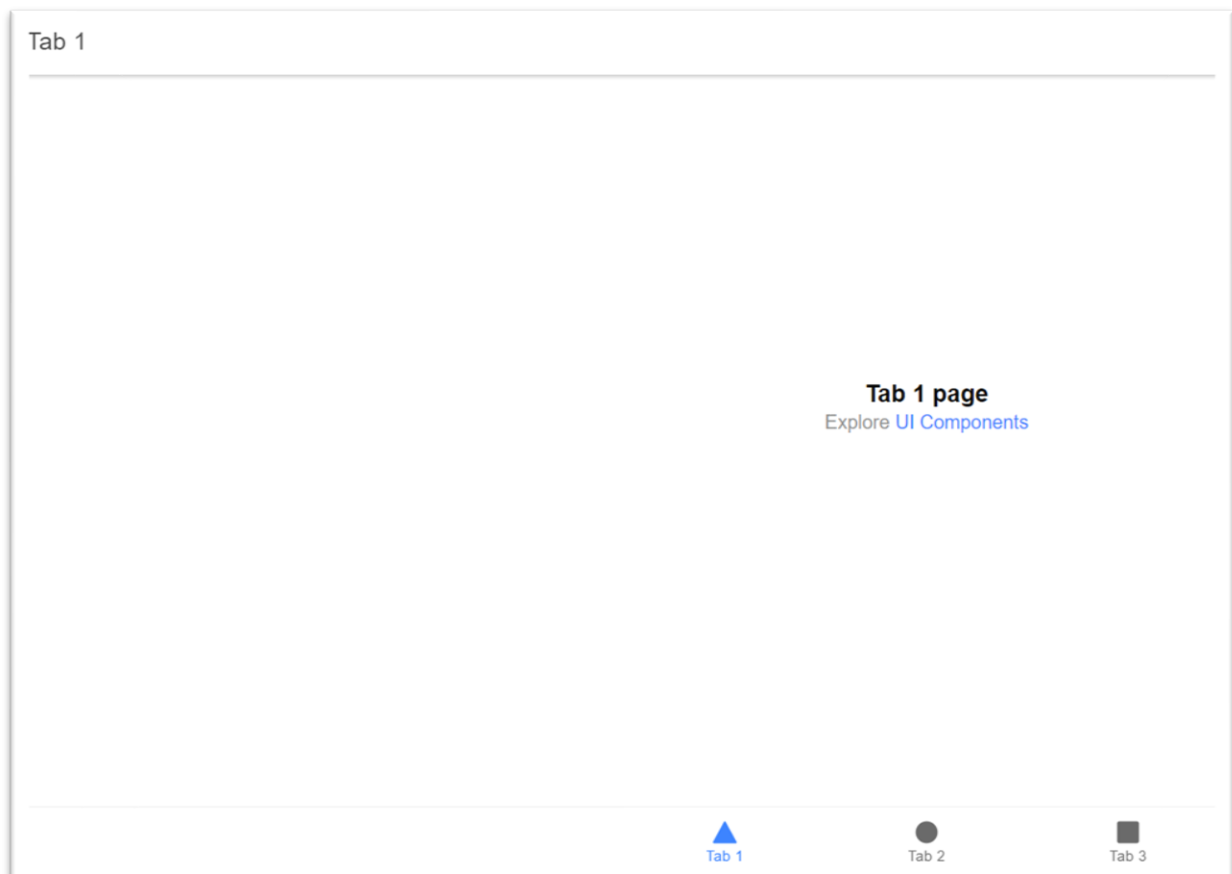


Figure 46: prototype 1 (ionic framework)

Figure 46: prototype 1 (above) shows the first prototype that was completed, which was implemented using the ionic framework. Initially, the ionic framework was chosen as there were resources available for the app to be implemented using Bluetooth, and there was potential for beacons to be built into the app so then when users were navigating around the IQ building, the app would automatically provide them with locational data. After following tutorials, conducting further research into this framework, and implementing different versions of code, it was decided that using the ionic framework was not adequate for the needs of the app because it was unable to detect Bluetooth beacons as expected and the code provided for Bluetooth was not deemed relevant to the requirements of the application therefore, an alternative software had to be found.

5.2.1.3 Prototype 3

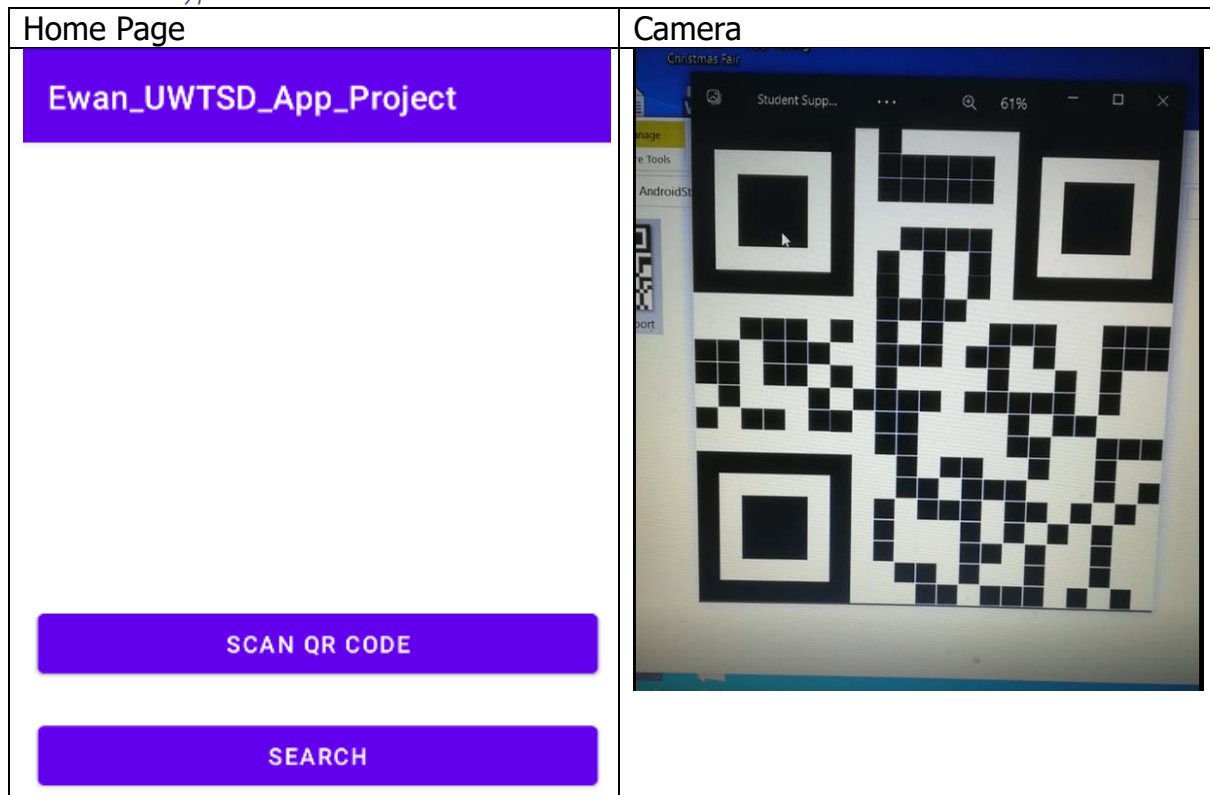


Figure 47: prototype 2

Figure 47: prototype 2 (above) shows the second prototype that was completed. The home page has been fully implemented, and the camera has been completed, and should be able to scan QR codes.

5.2.1.4 Prototype 4 – the final version

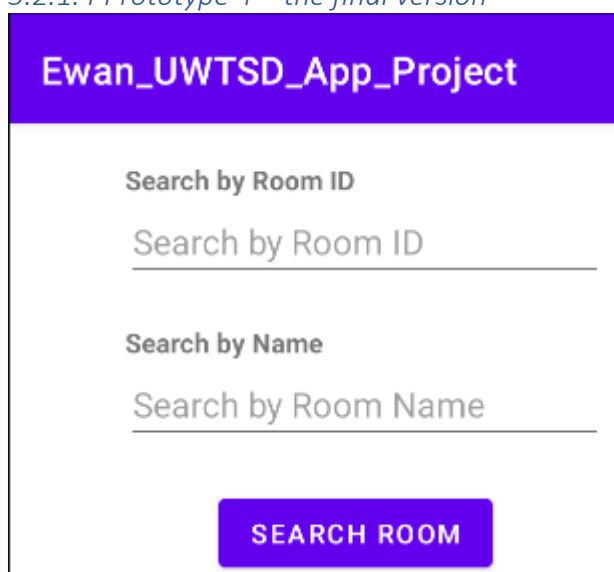


Figure 48: prototype 3 – the final version

Ewan_UWTSD_App_Project

Search by Room ID

Search by Room ID

Search by Name

Search by Room Name

SEARCH ROOM

Figure 48: prototype 3 (above) shows the third, and final, prototype that was completed. Alongside the already completed home page and camera, the search room page has now been completed, so users will be able to search for rooms by their assigned ID and/or name.

The main changes that were going to be implemented would be the data held on the app, switching the landmarks to individual rooms within the IQ building. The images would also be updated to reflect the rooms that were being added to the application. Much of the code was also changed, the functions, within the landmark factory, camera and app functions classes. The Gradle version was also updated to be on the latest version possible, and an alert dialog was created for the search room page.

5.3 Graphical User Interface

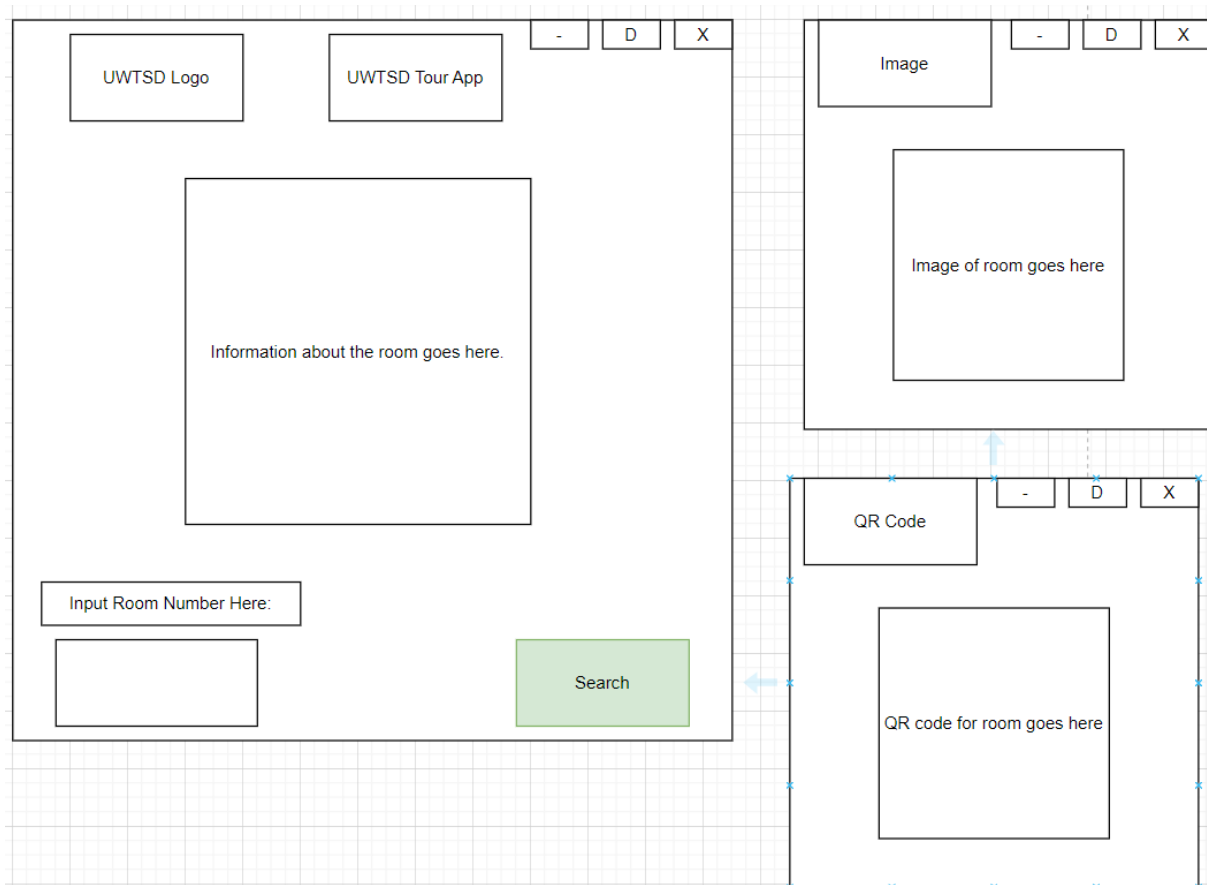


Figure 49: Initial Design of Java Application

Figure 49: Initial Design of Java Application (above) shows the initial design of the refined application. The search box to search for the rooms would be at the bottom of the form with the text area showing information about the room being placed in the centre of the screen. The University logo would be inserted into the top of the application with a label stating the name of the app next to it. The image and QR code of the room would both be in their own separate windows so that they are clearer to view and access.

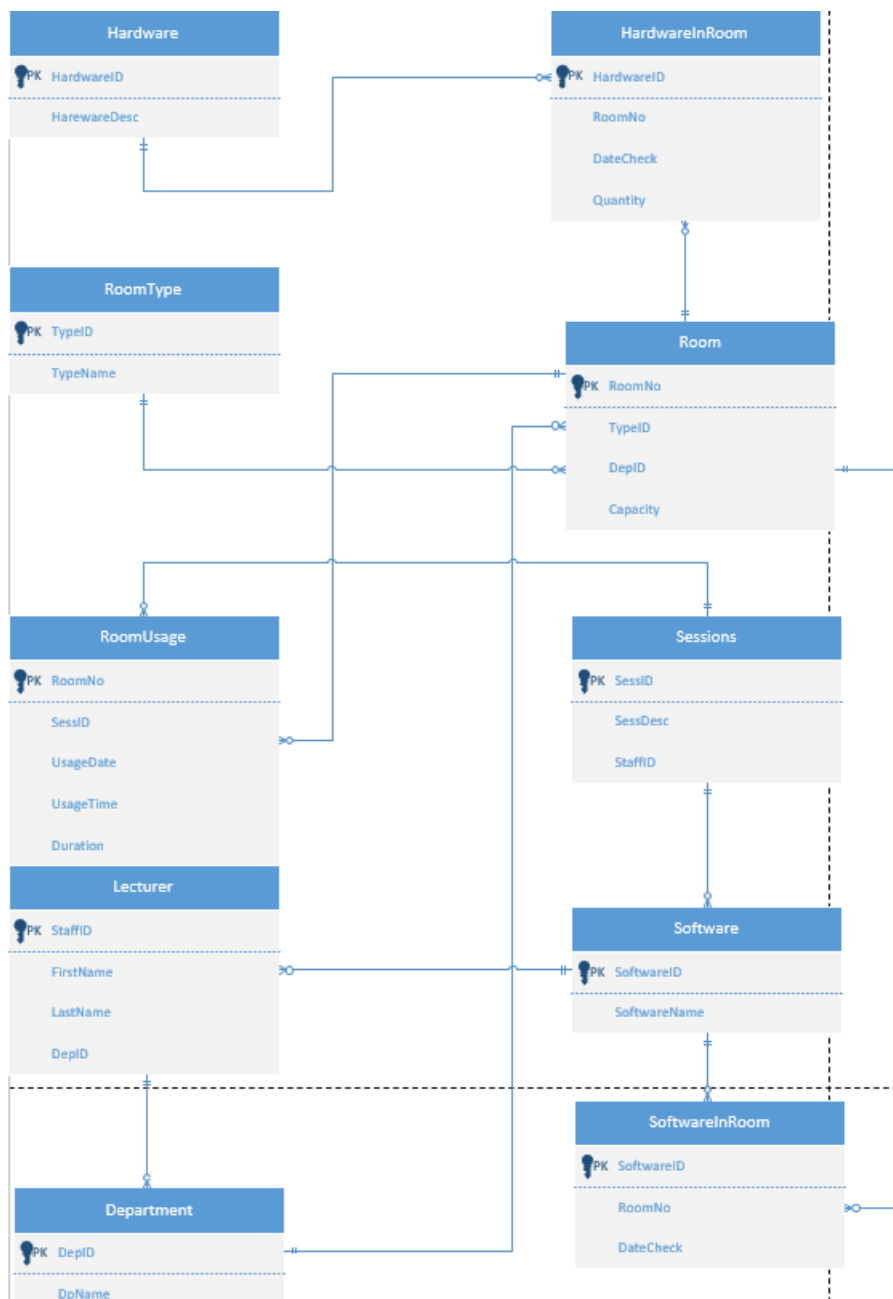


Figure 50: Database Diagram (Visio)

Figure 50: Database Diagram (Visio) demonstrates the database diagram created using Visio. There would be a total of eight tables within the database itself. There would be multiple one-to-many relationships, which are shown below in Table 10: Relationships.

Table	Keys	Table	Keys
Hardware	HardwareID	Hardware in Room	HardwareID
Room	RoomNo	RoomUsage	RoomNo

Room Type	TypeID	Room	TypeID
Department	DepID	Room	DepID
Sessions	StaffID	Lecturer	StaffID
Software	SoftwareID	Software in Room	SoftwareID

Table 10: Relationships

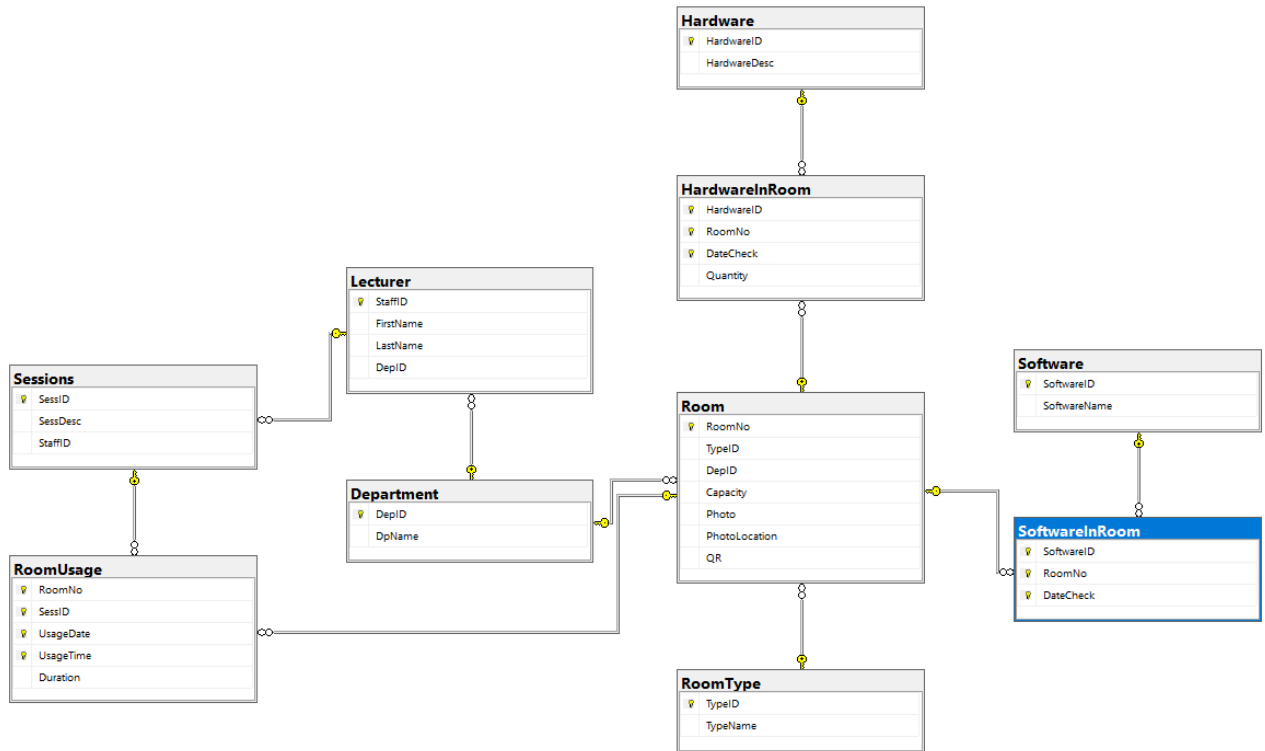


Figure 51: Database Diagram (SQL Server)

Figure 51: Database Diagram (SQL Server) demonstrates the diagram created in SQL server.

```
CREATE TABLE Software
(
  SoftwareID VARCHAR(4) NOT NULL,
  SoftwareName VARCHAR(20) NOT NULL,

  CONSTRAINT [PK_Software] PRIMARY KEY (SoftwareID)
)
```

Messages

Commands completed successfully.

Completion time: 2022-12-21T15:18:38.8718896+00:00

Figure 52: Creating the Software Table

Figure 52: Creating the Software Table shows that the command to create the software table was successful. The primary key set up for this table is SoftwareID.

```
CREATE TABLE Lecturer
(
  StaffID INTEGER NOT NULL,
  FirstName VARCHAR(20) NOT NULL,
  LastName VARCHAR(20) NOT NULL,
  DepID VARCHAR(5) NOT NULL,

  CONSTRAINT [PK_LECTURER] PRIMARY KEY (StaffID)
)
```

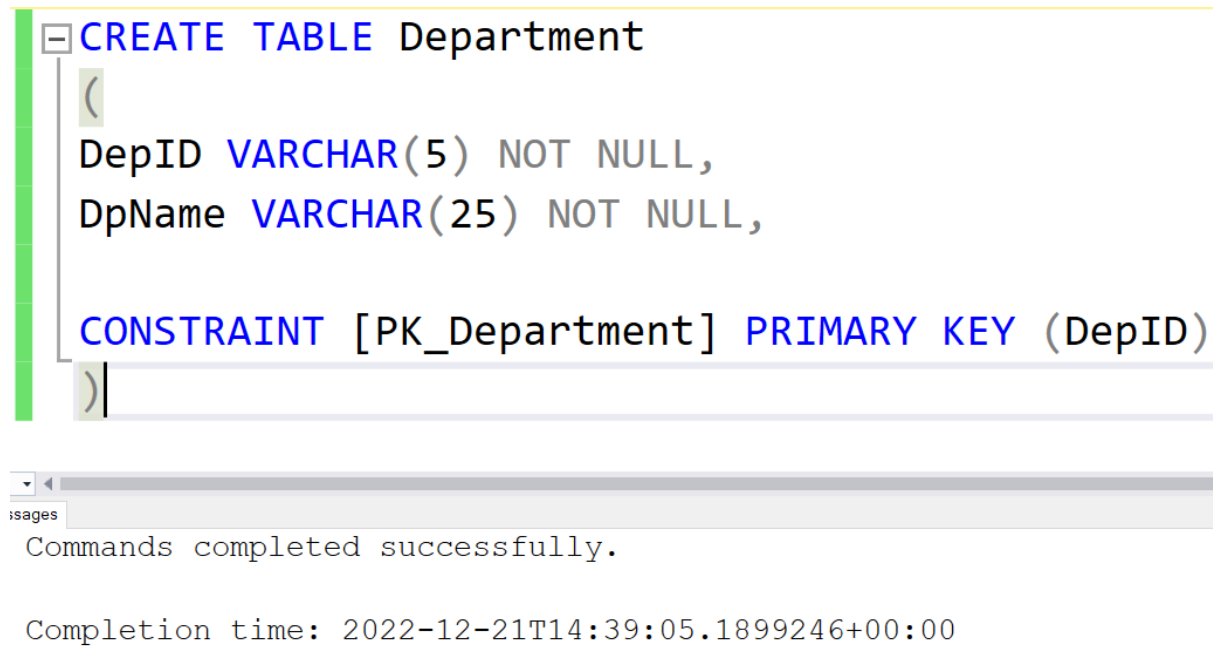
Messages

Commands completed successfully.

Completion time: 2022-12-21T15:09:26.7416813+00:00

Figure 53: Creating the Lecturer Table

Figure 53: Creating the Lecturer Table shows the command to create the lecturer table was successful. The primary key set up for this table is StaffID.



```
CREATE TABLE Department
(
  DepID VARCHAR(5) NOT NULL,
  DpName VARCHAR(25) NOT NULL,

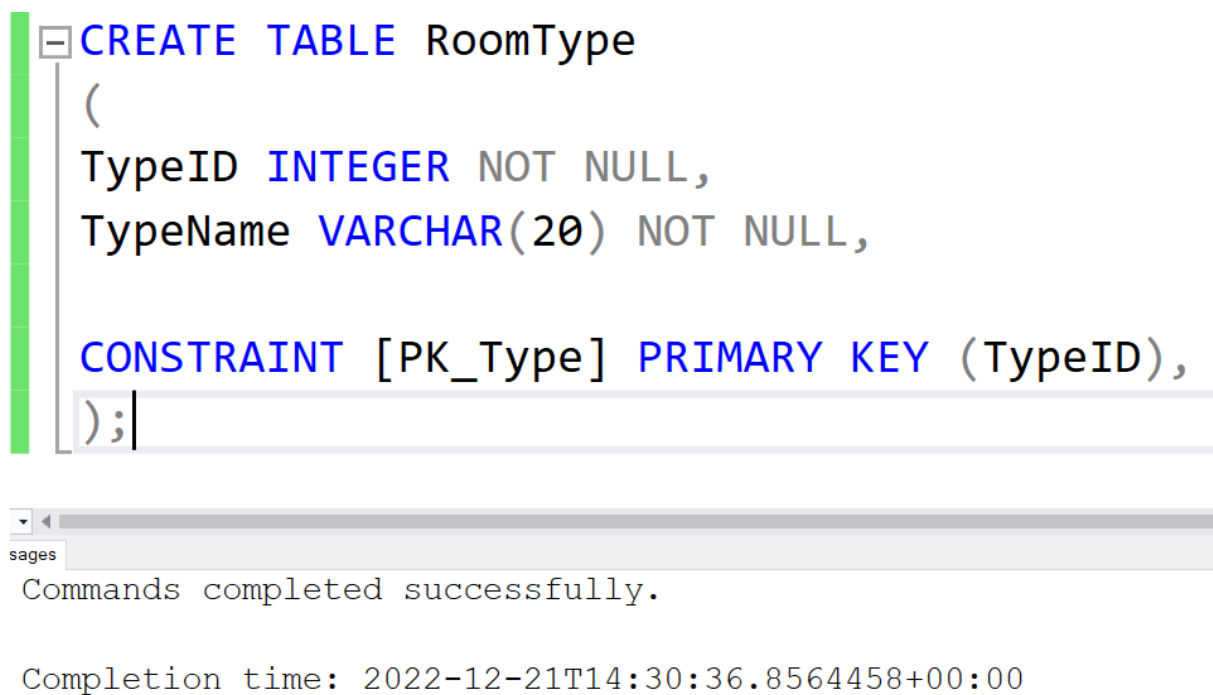
  CONSTRAINT [PK_Department] PRIMARY KEY (DepID)
);
```

Commands completed successfully.

Completion time: 2022-12-21T14:39:05.1899246+00:00

Figure 54: Creating the Department Table

Figure 54: Creating the Department Table shows the commands to create the department table was successful. The primary key set up for this table is DepID.



```
CREATE TABLE RoomType
(
 TypeID INTEGER NOT NULL,
  TypeName VARCHAR(20) NOT NULL,

  CONSTRAINT [PK_Type] PRIMARY KEY (TypeID),
);
```

Commands completed successfully.

Completion time: 2022-12-21T14:30:36.8564458+00:00

Figure 55: Creating the Room Type Table

Figure 55: Creating the Room Type Table shows the commands to create the room type table was successful. The primary key set up for this table isTypeID.

```
CREATE TABLE Sessions
(
  SessID INTEGER NOT NULL,
  SessDesc VARCHAR(20) NOT NULL,
  StaffID INTEGER NOT NULL,

  CONSTRAINT [PK_Session] PRIMARY KEY (SessID),
  CONSTRAINT [FK_Staff_Session] FOREIGN KEY (StaffID) REFERENCES Lecturer (StaffID)
)
```

Messages

Commands completed successfully.

Completion time: 2022-12-21T15:45:23.2415717+00:00

Figure 56: Creating the Sessions Table

Figure 56: Creating the Sessions Table shows the commands to create the sessions table was successful. The primary key set up for this table is SessID, with a foreign key being created between the StaffID on the lecturer and sessions tables as there would be a relationship made between these two tables.

```
CREATE TABLE Hardware
(
  HardwareID VARCHAR(5) NOT NULL,
  HardwareDesc VARCHAR(25) NOT NULL,

  CONSTRAINT [PK_Hardware] PRIMARY KEY (HardwareID)
)
```

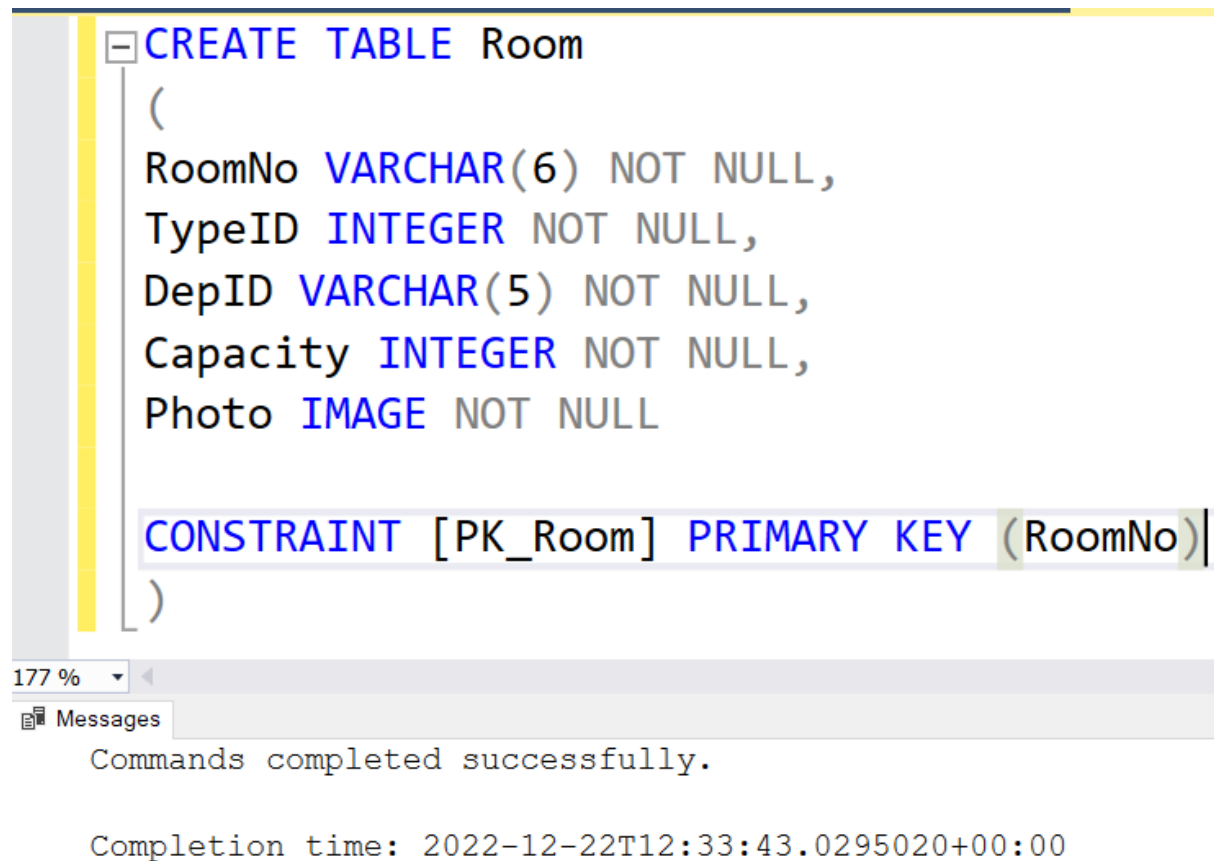
Messages

Commands completed successfully.

Completion time: 2022-12-21T15:48:44.9379366+00:00

Figure 57: Creating the Hardware Table

Figure 57: Creating the Hardware Table shows the commands to create the hardware table was successful. The primary key set up for this table is HardwareID.



```
CREATE TABLE Room
(
  RoomNo VARCHAR(6) NOT NULL,
 TypeID INTEGER NOT NULL,
  DepID VARCHAR(5) NOT NULL,
  Capacity INTEGER NOT NULL,
  Photo IMAGE NOT NULL

  CONSTRAINT [PK_Room] PRIMARY KEY (RoomNo)
)
```

177 %

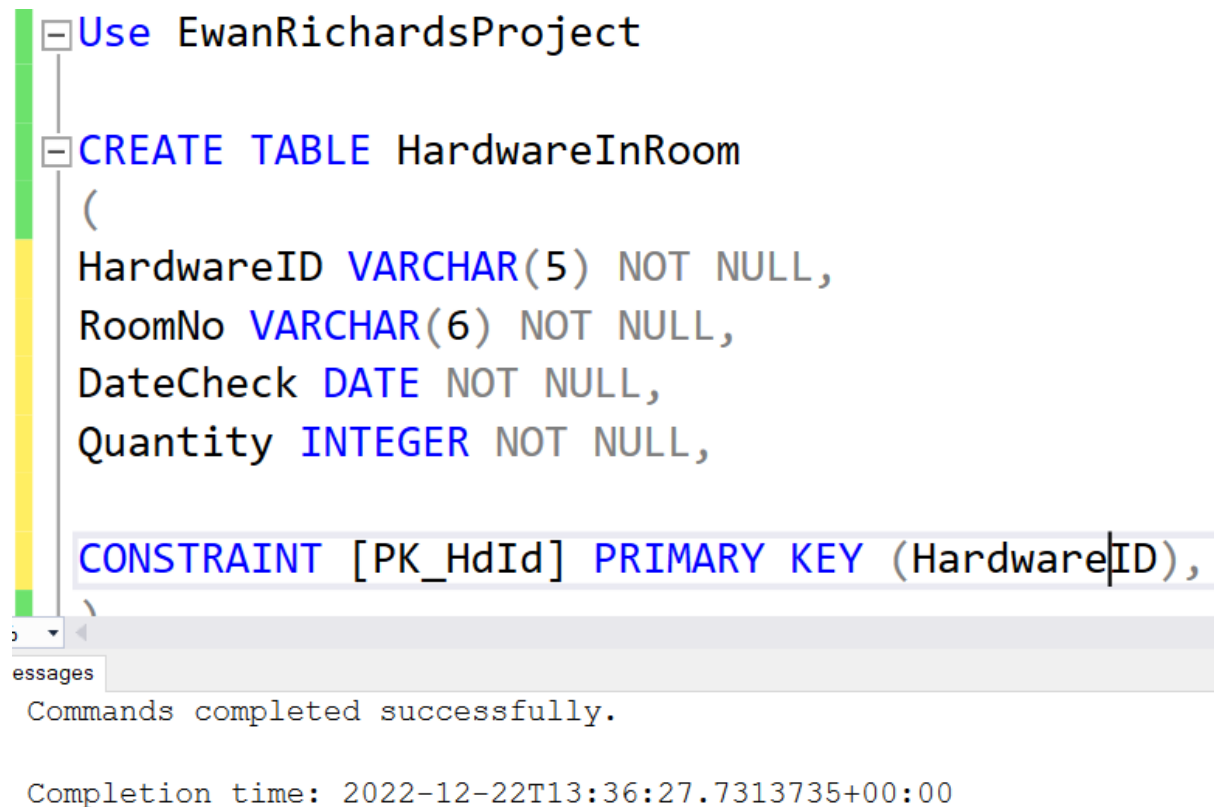
Messages

Commands completed successfully.

Completion time: 2022-12-22T12:33:43.0295020+00:00

Figure 58: Creating the Room Table

Figure 58: Creating the Room Table shows the commands to create the room table was successful. The primary key set up for this table is RoomNo.



```
-- Use EwanRichardsProject

CREATE TABLE HardwareInRoom
(
  HardwareID VARCHAR(5) NOT NULL,
  RoomNo VARCHAR(6) NOT NULL,
  DateCheck DATE NOT NULL,
  Quantity INTEGER NOT NULL,

  CONSTRAINT [PK_HdId] PRIMARY KEY (HardwareID),
)
```

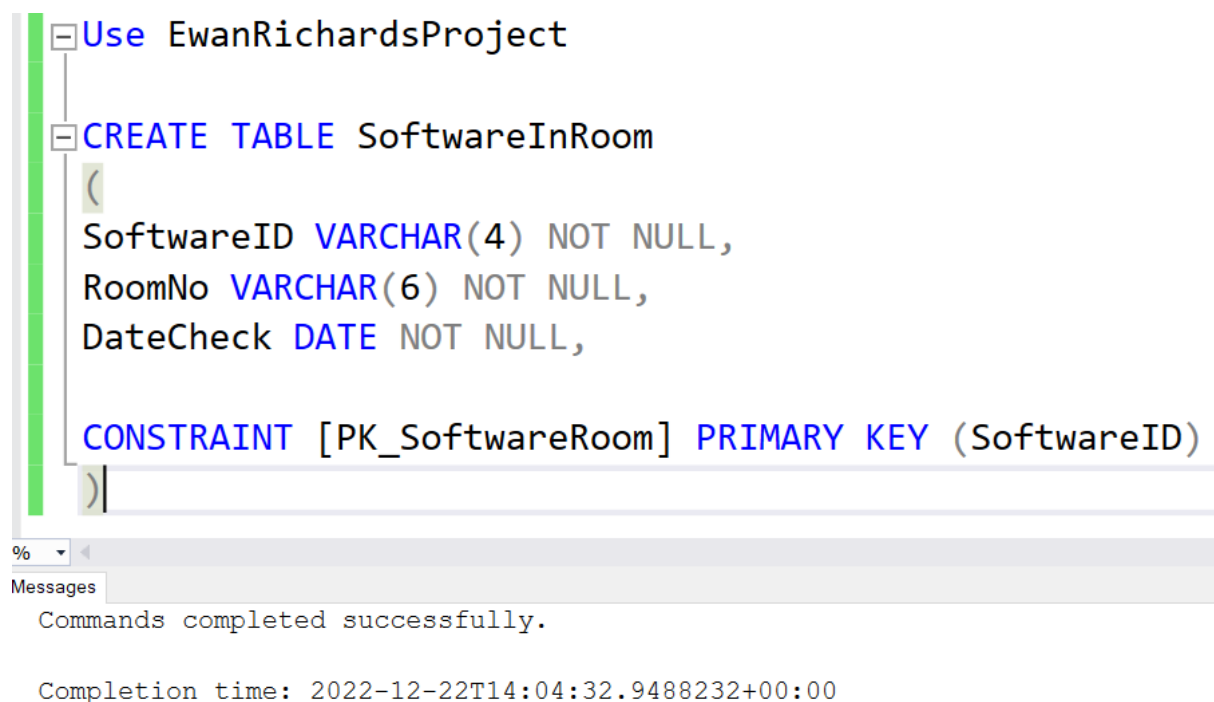
Messages

Commands completed successfully.

Completion time: 2022-12-22T13:36:27.7313735+00:00

Figure 59: Creating the Hardware in Room Table

Figure 59: Creating the Hardware in Room Table shows the commands to create the hardware in room table was successful. The primary key set up for this table is HardwareID.



```
-- Use EwanRichardsProject

CREATE TABLE SoftwareInRoom
(
  SoftwareID VARCHAR(4) NOT NULL,
  RoomNo VARCHAR(6) NOT NULL,
  DateCheck DATE NOT NULL,

  CONSTRAINT [PK_SoftwareRoom] PRIMARY KEY (SoftwareID)
)
```

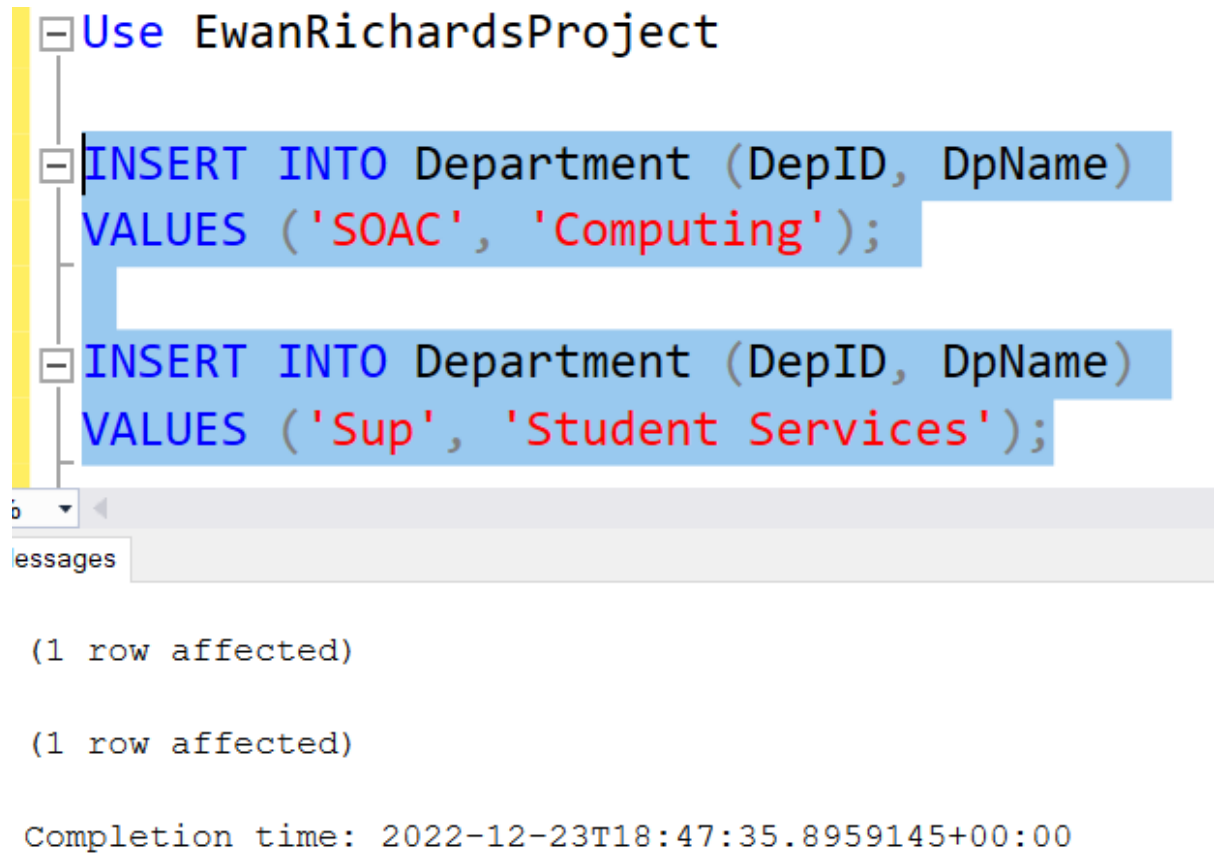
Messages

Commands completed successfully.

Completion time: 2022-12-22T14:04:32.9488232+00:00

Figure 60: Creating the Software in Room Table

5.3.1 Add Data



The screenshot shows a database client window titled "Use EwanRichardsProject". It displays two SQL statements executed in sequence, each highlighted in blue. The first statement is `INSERT INTO Department (DepID, DpName) VALUES ('SOAC', 'Computing');` and the second is `INSERT INTO Department (DepID, DpName) VALUES ('Sup', 'Student Services');`. Below the statements, the "Messages" tab is active, showing two messages: "(1 row affected)" and "(1 row affected)". At the bottom, the completion time is displayed as "Completion time: 2022-12-23T18:47:35.8959145+00:00".

```
-- Use EwanRichardsProject
--
-- INSERT INTO Department (DepID, DpName)
-- VALUES ('SOAC', 'Computing');
--
-- INSERT INTO Department (DepID, DpName)
-- VALUES ('Sup', 'Student Services');
--
--
-- Messages
--
-- (1 row affected)
--
-- (1 row affected)
--
-- Completion time: 2022-12-23T18:47:35.8959145+00:00
```

Figure 61: Adding Data into the Department Table

Figure 61: Adding Data into the Department Table demonstrates that data was added to the department table successfully.

```
INSERT INTO RoomType (TypeID, TypeName)
VALUES ('1', 'SOAC Lab');

INSERT INTO RoomType (TypeID, TypeName)
VALUES ('2', 'Uni Lab');

INSERT INTO RoomType (TypeID, TypeName)
VALUES ('3', 'Studio');

INSERT INTO RoomType (TypeID, TypeName)
VALUES ('4', 'Support');
```

Messages

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

Figure 62: Adding Data to the Room Type Table

Figure 62: Adding Data to the Room Type Table demonstrates that data was added to the room type table successfully.

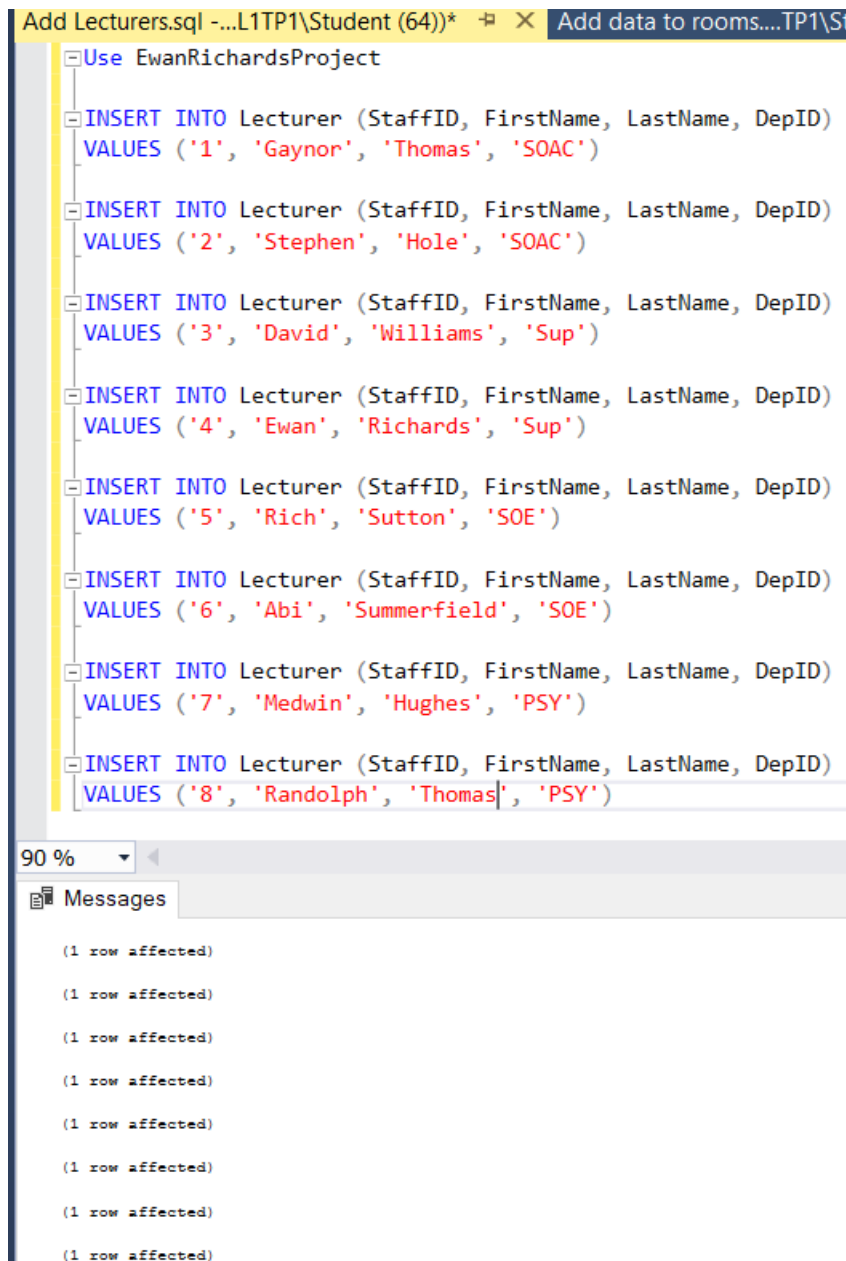


Figure 63: Adding Data to the Lecturer Table

Figure 63: Adding Data to the Lecturer Table demonstrates that data was added to the lecturer table successfully.

```
Use EwanRichardsProject

INSERT INTO Sessions (SessID, SessDesc, StaffID)
VALUES ('1', 'Lecture', '1')

INSERT INTO Sessions (SessID, SessDesc, StaffID)
VALUES ('2', 'Apprenticeship Introduction', '2')

INSERT INTO Sessions (SessID, SessDesc, StaffID)
VALUES ('3', 'Support', '3')

INSERT INTO Sessions (SessID, SessDesc, StaffID)
VALUES ('4', 'Support', '4')

INSERT INTO Sessions (SessID, SessDesc, StaffID)
VALUES ('5', 'Lecture', '5')

INSERT INTO Sessions (SessID, SessDesc, StaffID)
VALUES ('6', 'Lecture', '6')

INSERT INTO Sessions (SessID, SessDesc, StaffID)
VALUES ('7', 'Conference', '7')

INSERT INTO Sessions (SessID, SessDesc, StaffID)
VALUES ('8', 'Council Meeting', '8')
```

82 %

Messages

(1 row affected)
Msg 2628, Level 16, State 1, Line 6
String or binary data would be truncated in table 'EwanRichardsProject.dbo.Sessions', column 'SessDesc'. Truncated value: 'Apprenticeship Intro'.
The statement has been terminated.

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

Figure 64: Adding Data to the Sessions Table

Figure 64: Adding Data to the Sessions Table demonstrates that data was added to the sessions table successfully. An error was displayed whilst inserting data onto the sessions table, and the way it was resolved can be seen below in Figure 65: Resolving the error on the Sessions Table.

```
INSERT INTO Sessions (SessID, SessDesc, StaffID)
VALUES ('2', 'Apprenticeship Intro', '2')
```

(1 row affected)

Completion time: 2023-01-03T16:47:55.2574153+00:00

Figure 65: Resolving the error on the Sessions Table



```
INSERT INTO Software (SoftwareID, SoftwareName)
VALUES ('Unty', 'Unity')

INSERT INTO Software (SoftwareID, SoftwareName)
VALUES ('AnSt', 'Android Studio')

INSERT INTO Software (SoftwareID, SoftwareName)
VALUES ('SQSv', 'SQL Server')

INSERT INTO Software (SoftwareID, SoftwareName)
VALUES ('Mtlb', 'Matlab')

INSERT INTO Software (SoftwareID, SoftwareName)
VALUES ('PsST', 'Psychology Software Tools')
```

120 %

Messages

(1 row affected)

(1 row affected)

(1 row affected)

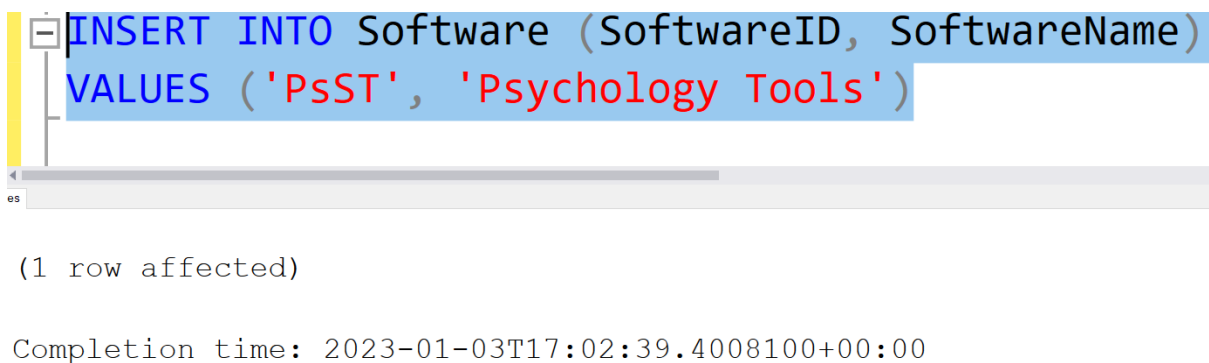
(1 row affected)

Msg 2628, Level 16, State 1, Line 15
String or binary data would be truncated in table 'master.dbo.Software', column 'SoftwareName'. Truncated value: 'Psychology Software '.
The statement has been terminated.

Completion time: 2023-01-03T16:55:58.2535864+00:00

Figure 66: Inserting Data into the Software Table

Figure 66: Inserting Data into the Software Table demonstrates that data was added to the software table successfully. An error was displayed whilst inserting data onto the software table, and the way it was resolved can be seen below in Figure 67: resolving an error in the Software .

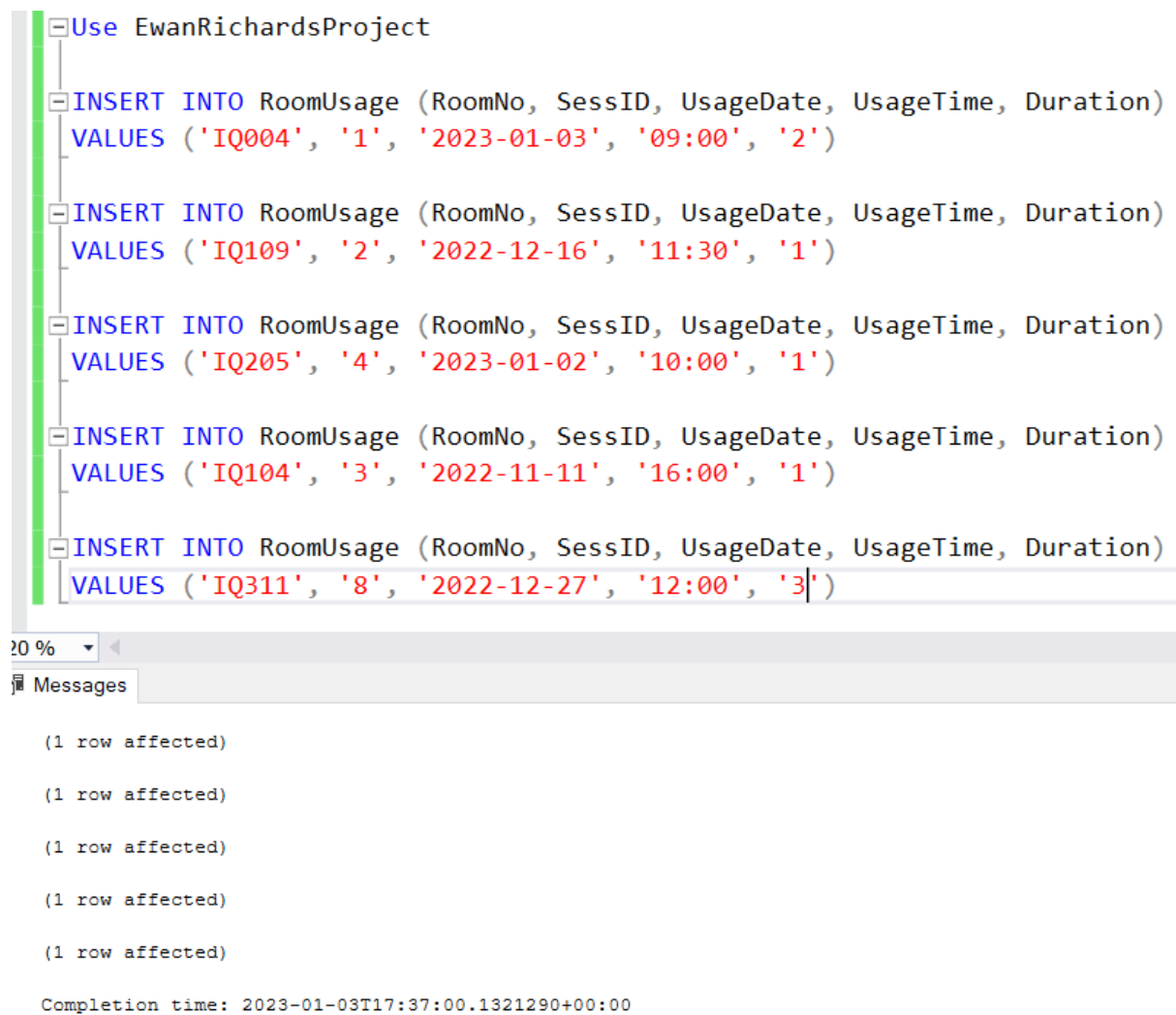


```
INSERT INTO Software (SoftwareID, SoftwareName)
VALUES ('PsST', 'Psychology Tools')
```

(1 row affected)

Completion time: 2023-01-03T17:02:39.4008100+00:00

Figure 67: resolving an error in the Software table.



```
Use EwanRichardsProject

INSERT INTO RoomUsage (RoomNo, SessID, UsageDate, UsageTime, Duration)
VALUES ('IQ004', '1', '2023-01-03', '09:00', '2')

INSERT INTO RoomUsage (RoomNo, SessID, UsageDate, UsageTime, Duration)
VALUES ('IQ109', '2', '2022-12-16', '11:30', '1')

INSERT INTO RoomUsage (RoomNo, SessID, UsageDate, UsageTime, Duration)
VALUES ('IQ205', '4', '2023-01-02', '10:00', '1')

INSERT INTO RoomUsage (RoomNo, SessID, UsageDate, UsageTime, Duration)
VALUES ('IQ104', '3', '2022-11-11', '16:00', '1')

INSERT INTO RoomUsage (RoomNo, SessID, UsageDate, UsageTime, Duration)
VALUES ('IQ311', '8', '2022-12-27', '12:00', '3')
```

20 %

Messages

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2023-01-03T17:37:00.1321290+00:00

Figure 68: Adding Data into the Room Usage Table

Figure 68: Adding Data into the Room Usage Table demonstrates that data was added to the room usage table successfully.

```
Use EwanRichardsProject
GO
INSERT INTO Hardware (HardwareID, HardwareDesc)
VALUES ('PJ', 'Projector')
GO
INSERT INTO Hardware (HardwareID, HardwareDesc)
VALUES ('P1', 'PC General Usage')
GO
INSERT INTO Hardware (HardwareID, HardwareDesc)
VALUES ('P2', 'PC Games Spec')
GO
INSERT INTO Hardware (HardwareID, HardwareDesc)
VALUES ('P3', 'Podium PC')
GO
INSERT INTO Hardware (HardwareID, HardwareDesc)
VALUES ('SU', 'Speaker Units and Ampli')
GO
INSERT INTO Hardware (HardwareID, HardwareDesc)
VALUES ('CH', 'Chairs')
GO
```

Messages

```
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
Msg 2628, Level 16, State 1, Line 15
String or binary data would be truncated in table 'EwanRichardsProject.dbo.Hardware', column 'HardwareDesc'. Truncated value: 'Speaker Units and Amplifi'.
The statement has been terminated.
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
(1 row affected)
```

Figure 69: Adding Data into the Hardware Table

Figure 69: Adding Data into the Hardware Table demonstrates that data was added to the hardware table successfully. An error was displayed whilst inserting data onto the hardware table, and the way it was resolved can be seen below in Figure 70: resolving the error in the Hardware .

```
INSERT INTO Hardware (HardwareID, HardwareDesc)
VALUES ('SU', 'Speaker Units and Amp')
```

Messages

```
(1 row affected)

Completion time: 2023-01-04T17:29:16.5358517+00:00
```

Figure 70: resolving the error in the Hardware table.

```
INSERT INTO HardwareInRoom (HardwareID, RoomNo, DateCheck, Quantity)
VALUES ('SU', 'IQ210', '2022-12-06', '2')

INSERT INTO HardwareInRoom (HardwareID, RoomNo, DateCheck, Quantity)
VALUES ('CH', 'IQ210', '2022-12-06', '10')

INSERT INTO HardwareInRoom (HardwareID, RoomNo, DateCheck, Quantity)
VALUES ('TB', 'IQ210', '2022-12-06', '5')

INSERT INTO HardwareInRoom (HardwareID, RoomNo, DateCheck, Quantity)
VALUES ('SM', 'IQ210', '2022-12-06', '2')

INSERT INTO HardwareInRoom (HardwareID, RoomNo, DateCheck, Quantity)
VALUES ('EE', 'IQ210', '2022-12-06', '10')

INSERT INTO HardwareInRoom (HardwareID, RoomNo, DateCheck, Quantity)
VALUES ('PJ', 'IQ311', '2022-12-07', '1')

INSERT INTO HardwareInRoom (HardwareID, RoomNo, DateCheck, Quantity)
VALUES ('P1', 'IQ311', '2022-12-07', '30')

INSERT INTO HardwareInRoom (HardwareID, RoomNo, DateCheck, Quantity)
VALUES ('P3', 'IQ311', '2022-12-07', '1')

INSERT INTO HardwareInRoom (HardwareID, RoomNo, DateCheck, Quantity)
```

81 %

Messages

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

(1 row affected)

Figure 71: Adding data into the Hardware In Room Table

Figure 71: Adding data into the Hardware In Room Table demonstrates that data was added to the hardware in room table successfully.

```

Use EwanRichardsProject

INSERT INTO RoomUsage (RoomNo, SessID, UsageDate, UsageTime, Duration)
VALUES ('FF002', '3', '2023-01-08', '09:00', '2')

INSERT INTO RoomUsage (RoomNo, SessID, UsageDate, UsageTime, Duration)
VALUES ('FF101', '8', '2023-01-05', '12:00', '3')

INSERT INTO RoomUsage (RoomNo, SessID, UsageDate, UsageTime, Duration)
VALUES ('FF201', '7', '2023-01-03', '10:30', '4')

```

%

Messages

(1 row affected)

(1 row affected)

(1 row affected)

Completion time: 2023-01-09T10:21:03.7145383+00:00

Figure 72: Adding Data into the Room Usage table.

Figure 72: Adding Data into the Room Usage demonstrates that data was added to the room usage table successfully.

	RoomNo	TypeID	DepID	Capacity	Photo	PhotoLocation	QR
▶	FF002	5	LIB	30	NULL	D:\Images\Photo\FF002.jpg	D:\Images\QR\FF002.png
	FF101	5	LIB	6	NULL	D:\Images\Photo\FF101.jpg	D:\Images\QR\FF101.png
	FF201	5	LIB	1	NULL	D:\Images\Photo\FF201.jpg	D:\Images\QR\FF201.png
	IQ002	2	SOE	50	NULL	D:\Images\Photo\IQ002.jpg	D:\Images\QR\IQ002.png
	IQ004	1	SOAC	20	NULL	D:\Images\Photo\IQ004.jpg	D:\Images\QR\IQ004.png
	IQ009	2	SOE	20	NULL	D:\Images\Photo\IQ009.jpg	D:\Images\QR\IQ009.png
	IQ104	4	Sup	10	NULL	D:\Images\Photo\IQ104.jpg	D:\Images\QR\IQ104.png
	IQ109	1	SOAC	20	NULL	D:\Images\Photo\IQ109.jpg	D:\Images\QR\IQ109.png
	IQ205	1	SOAC	20	NULL	D:\Images\Photo\IQ205.jpg	D:\Images\QR\IQ205.png
	IQ209	1	SOAC	20	NULL	D:\Images\Photo\IQ209.jpg	D:\Images\QR\IQ209.png
	IQ210	1	SOAC	20	NULL	D:\Images\Photo\IQ210.jpg	D:\Images\QR\IQ210.png
	IQ311	3	PSY	50	NULL	D:\Images\Photo\IQ310.jpg	D:\Images\QR\IQ311.png
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 73: Error with Photos in SQL Server

Due to time constraints, the images could not be inserted into the database so the developer went down the avenue of storing the image locations for both the photographs and QR codes. A problem was encountered when trying to remove the photo field so the developer felt that the best route would be to leave it in the table as it is the intention to insert these image as a BLOB in future iterations (a second image field will be required for the QR codes).

5.3.2 Creating a User

Login Properties - testUser

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Script Help

Login name: testUser Search...

☐ Windows authentication

☒ SQL Server authentication

Password:

Confirm password:

☐ Specify old password

Old password:

☐ Enforce password policy

☐ Enforce password expiration

☐ User must change password at next login

☐ Mapped to certificate

☐ Mapped to asymmetric key

☐ Map to Credential Add

Mapped Credentials

Credential	Provider
------------	----------

Remove

Default database: EwanRichardsProject

Default language: English - us_english

Connection

Server: DESKTOP-KIL1TP1

Connection: testUser

[View connection properties](#)

Progress

Ready

Figure 74: Creating a User

An administrator user was added to the database, as shown in Figure 74: Creating a User. User privileges were assigned in Figure 75: assigning the user permissions and Figure 76: Logging into SQL Server shows the user logging into SQL Server.

Login Properties - testUser

Select a page

- General
- Server Roles
- User Mapping
- Securables
- Status

Script Help

Server role is used to grant server-wide security privileges to a user.

Server roles:

- ☒ bulkadmin
- ☒ dbcreator
- ☒ diskadmin
- ☒ processadmin
- ☒ public
- ☒ securityadmin
- ☒ serveradmin
- ☒ setupadmin
- ☒ sysadmin

Figure 75: assigning the user permissions

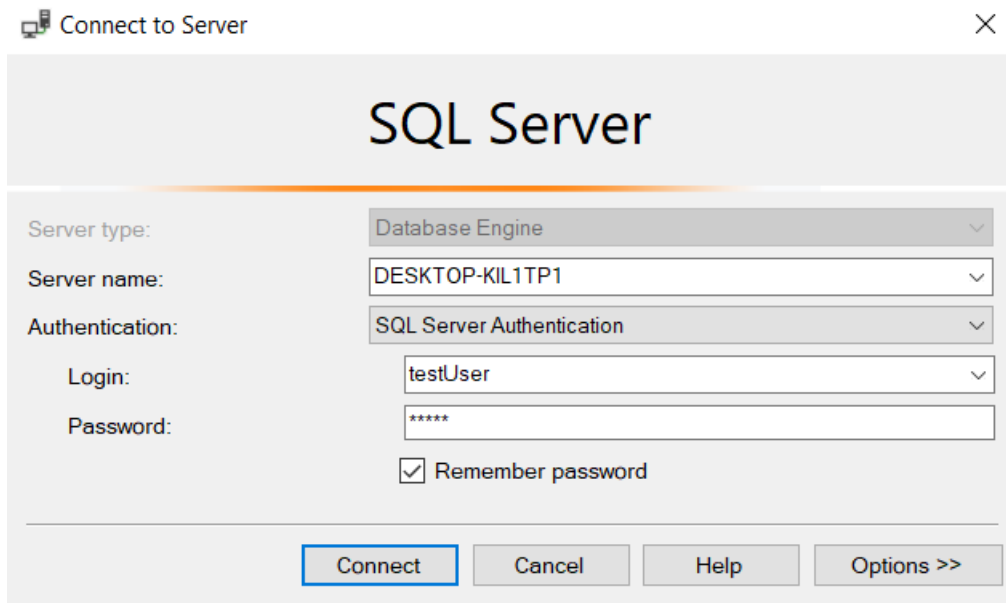


Figure 76: Logging into SQL Server

In order for the data connection string to work properly a SQL Server authentication user had to be created within the database as problems were encountered when using Windows authentication. Naturally, there is scope for security issues here, however, for the purpose of a prototype or proof of concept, this approach was considered adequate to produce demonstratable and usable code. Future iterations would need to focus on individual logins, however, owing to the fact that the desired app does not allow users to input data to the database and only read it, the security risk of this can be considered fairly low.

5.3.3 Connecting to the Database

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */

/**
 *
 * @author Student
 */

import java.sql.*;

public class DatabaseTest {
    static final String JDBC_DRIVER = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    static final String DB_URL = "jdbc:sqlserver://DESKTOP-KIL1TP1; databaseName=EwanRichardsProject; trustServerCertificate=true; encrypt=true;";
    static final String USER = "testUser";
    static final String PASS = "test1";

    DatabaseTest > JDBC_DRIVER >
}

```

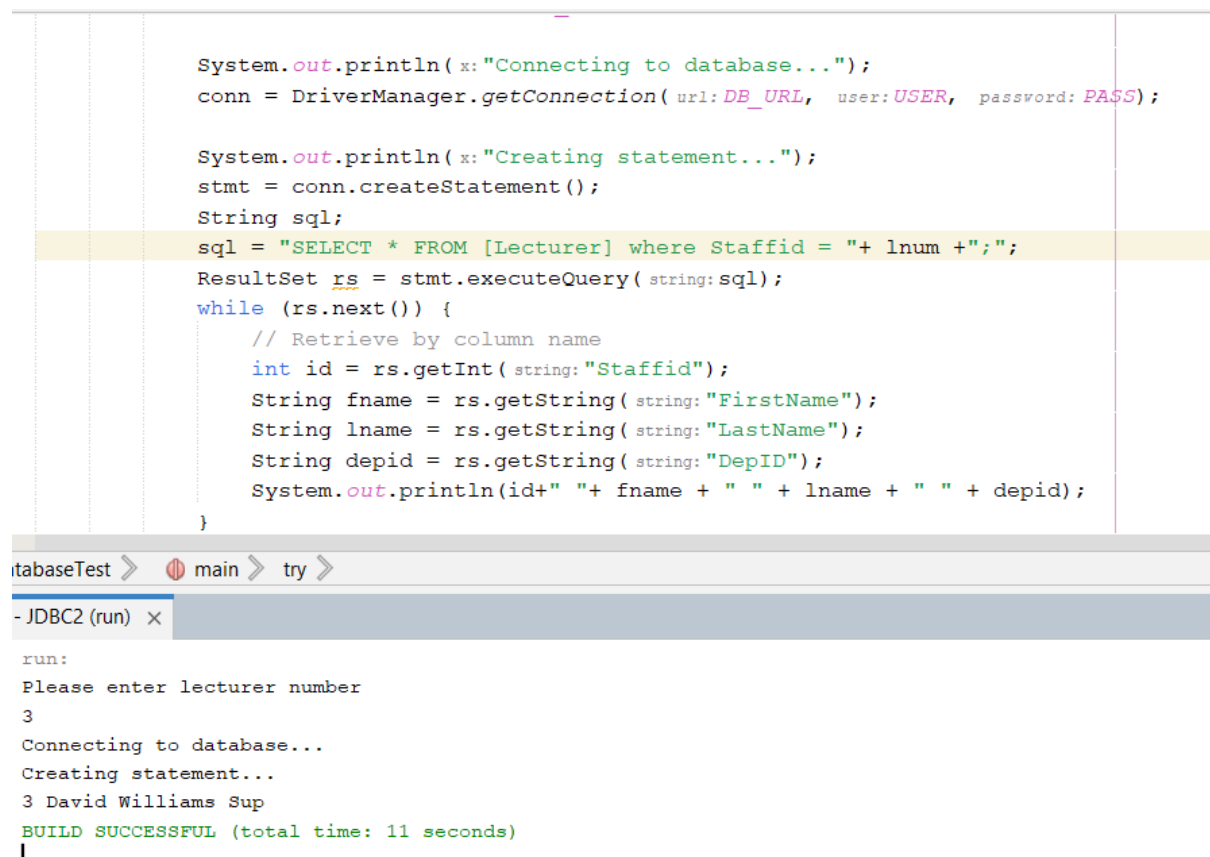
```

put - JDBC2 (run) x
run:
Connecting to database...
Creating statement...
1 Gaynor Thomas SOAC
2 Stephen Hole SOAC
3 David Williams Sup
4 Ewan Richards Sup
5 Rich Sutton SOE
6 Abi Summerfield SOE
7 Medwin Hughes PSY
8 Randolph Thomas PSY
BUILD SUCCESSFUL (total time: 1 second)

```

Figure 77: Showing the Data in the Lecturer Table

Figure 77: Showing the Data in the Lecturer Table demonstrates that a connection was successfully made, and data is being outputted from the lecturer table.



```
System.out.println(x:"Connecting to database...");
conn = DriverManager.getConnection( url:DB_URL, user:USER, password:PASS);

System.out.println(x:"Creating statement...");
stmt = conn.createStatement();
String sql;
sql = "SELECT * FROM [Lecturer] where Staffid = "+ lnum +";";
ResultSet rs = stmt.executeQuery( string:sql);
while (rs.next()) {
    // Retrieve by column name
    int id = rs.getInt( string:"Staffid");
    String fname = rs.getString( string:"FirstName");
    String lname = rs.getString( string:"LastName");
    String depid = rs.getString( string:"DepID");
    System.out.println(id+" "+ fname + " " + lname + " " + depid);
}
```

DatabaseTest > main > try >

- JDBC2 (run) x

run:
Please enter lecturer number
3
Connecting to database...
Creating statement...
3 David Williams Sup
BUILD SUCCESSFUL (total time: 11 seconds)

Figure 78: Showing Data from the Lecturer Table

Figure 78: Showing Data from the Lecturer Table demonstrates that a connection was successfully made, and data is being outputted from the lecturer table. The main difference between Figure 77: Showing the Data in the Lecturer Table and Figure 78: Showing Data from the Lecturer Table is that for Figure 78: Showing Data from the Lecturer Table, users can search for a lecturer using a unique ID whereas for Figure 77: Showing the Data in the Lecturer Table shows a whole list of staff.

```

    conn = DriverManager.getConnection( url:DB_URL, user:USER, password:PASS);

    stmt = conn.createStatement();

    ring sql = "Select  s.softwarename, sr.datecheck from software s, softwareInRoom sr where s.softwareid=sr.softwareid and sr.roomNo='"+rnum+"'";

    ResultSet rs = stmt.executeQuery( string:sql);
    while (rs.next()) {
        // Retrieve by column name
        String sd = rs.getString( string:"SoftwareName");
        String sdate = rs.getString( string:"DateCheck");
        System.out.println("Software Description: " +sd+" " + sdate);
    }
    rs.close();
    stmt.close();
    conn.close();

```

DatabaseTest > main > try > while (rs.next()) >

ut - JDBC2 (run) x

Please enter room number
IQ104
Connecting to database...
Creating statement...
Room Description: IQ104 10 Support Student Services
Hardware Description: Chairs 15 2022-12-01
Hardware Description: PC General Usage 30 2022-12-01
Hardware Description: Podium PC 1 2022-12-01
Hardware Description: Projector 1 2022-12-01
Hardware Description: Sockets 30 2022-12-09
Hardware Description: Tables 35 2022-12-01
Software Description: Dragon 2022-12-09
Software Description: Office 365 2022-12-09
Software Description: Read & Write 2022-12-09
Software Description: Sonocent 2022-12-09
Software Description: Unity 2022-12-02
BUILD SUCCESSFUL (total time: 8 seconds)

Figure 79: Showing Room Data, Hardware and Software Description

Figure 79: Showing Room Data, Hardware and Software Description demonstrates that a connection has been successfully made to the database and it is showing information in relation to the room being searched which is IQ104 (Student Services).

```

String sql = "Select  ru.ssessid, ru.usagedate, ru.usagetime, ru.duration, s.SessDesc, l.firstname,l.lastname, d.dpname from roomusage ru,

    ResultSet rs = stmt.executeQuery( string:sql);
    while (rs.next()) {
        // Retrieve by column name
        int sid = rs.getInt( string:"SessID");
        String udate = rs.getString( string:"UsageDate");
        String utime = rs.getString( string:"UsageTime");
        int dur = rs.getInt( string:"Duration");
        String sdesc = rs.getString( string:"SessDesc");
        String fname = rs.getString( string:"FirstName");
        String lname = rs.getString( string:"LastName");
        String dep = rs.getString( string:"DpName");

        System.out.println("Session Time: " + sid + " " + udate + " " + utime + " " + dur);
    }

```

DatabaseTest > main > try > sql >

ut - JDBC2 (run) x

Connecting to database...
Creating statement...
Room Description: IQ104 10 Support Student Services
Hardware Description: Chairs 15 2022-12-01
Hardware Description: PC General Usage 30 2022-12-01
Hardware Description: Podium PC 1 2022-12-01
Hardware Description: Projector 1 2022-12-01
Hardware Description: Sockets 30 2022-12-09
Hardware Description: Tables 35 2022-12-01
Software Description: Dragon 2022-12-09
Software Description: Office 365 2022-12-09
Software Description: Read & Write 2022-12-09
Software Description: Sonocent 2022-12-09
Software Description: Unity 2022-12-02
Session Time: 3 2022-11-11 16:00:00.0000000 1
Session Description: Support David Williams Student Services
BUILD SUCCESSFUL (total time: 7 seconds)

Figure 80: Further Data Being Shown

Figure 80: Further Data Being Shown is demonstrating that further data is now being outputted for the room, and it is now showing session dates and times.



Figure 81: Version 2.1

Once the initial console version had been successfully a basic form was created, with a text box created so that users can search a room within the building. As shown in Figure 81: Version 2.1, room IQ104 has been searched for and it shows information about the room, software and hardware descriptions and session (timetable) information.

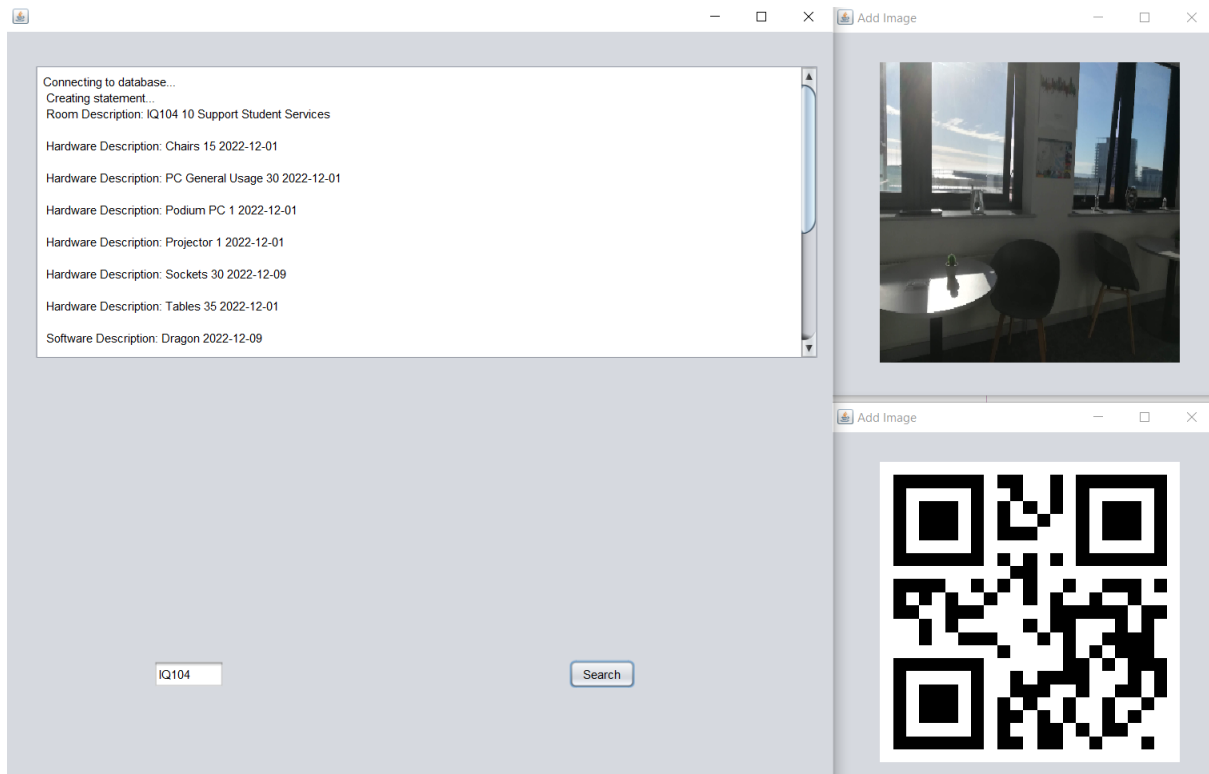


Figure 82: Version 2.2

In addition to the work being carried out in Figure 81: Version 2.1, Figure 82: Version 2.2 shows that a picture of the room has been added alongside the QR code for the room.

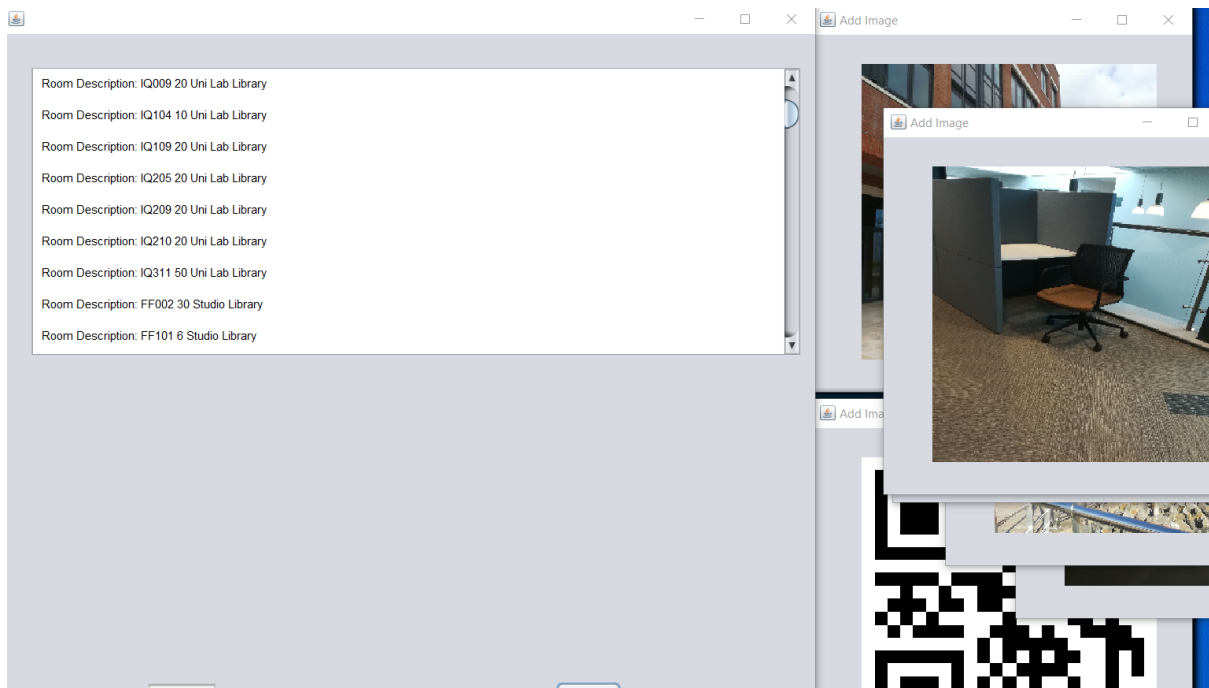


Figure 83: SQL Inject

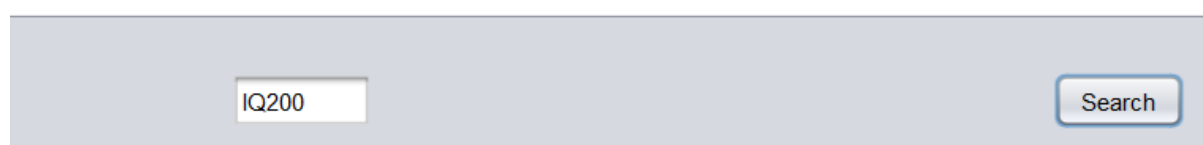
One of the issues that was discovered with this system is that users could potentially SQL inject it, particularly as the application is linked to a SQL database. A future version would require the use of stored procedures on the SQL Server which can be used by the code to eradicate the possibility of this occurring.

```
} catch (SQLException se) {
    jTextArea1.setText( t: "SQL ERROR");
    se.printStackTrace();
} catch (Exception e) {
    jTextArea1.setText( t: "EXCEPTION");
    e.printStackTrace();
} finally {
    try {
        if (stmt != null)
            stmt.close();
    } catch (SQLException se2) {
        jTextArea1.setText( t: "NULL ERROR");
    }
    try {
        if (conn != null)
            conn.close();
    } catch (SQLException se) {
        jTextArea1.setText( t: "CONNECTION ERROR");
        se.printStackTrace();
    }
}
```

Figure 84: Try Catch Error Code

If an unknown value is entered, it hangs rather than displaying information. Despite that error catches have been put in place, these do not seem to be reflected in the text area, this would have to be addressed in any future sprint. An example of this is on Figure 85: Room not registered on app, with a code snippet of this being demonstrated on Figure 84: Try Catch Error Code.

Connecting to database...
Creating statement...



A horizontal search bar with a light gray background. On the left, there is a text input field containing the text "IQ200". On the right, there is a rounded rectangular button with a blue border and the text "Search".

Figure 85: Room not registered on app

CHAPTER 6 – RESULTS, ANALYSIS AND EVALUATION

6.1 Analysis During App Development

During development, several errors within the code. Many of the errors were import statements, where it was not recognising the class that was being imported.

```
import com.example.ewan_uwtsd_app_project.objLandmark.*;
import com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmark;
import com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmarkFactory;

public class searchLandmarkActivity extends AppCompatActivity {
    // Declares string for log tag. Used for debugging purposes.
    public static final String LOG_TAG = searchLandmarkActivity.class.getSimpleName();
    // Declares static final String for package/context directory of variable. Used when changing activities.
    public static final String LANDMARK_ID = "com.csg.qrHistoricApp.LANDMARK_ID";
```

Figure 86: unused import statement on Search Landmark Activity page

One of the unused import statements was discovered on the search landmark activity java page. It was not able to import the classes within the objLandmark folder within the repository, not reading any of the interfaces or classes within that folder. Although there were references made to the landmark classes within the search landmark activity page relating to the landmark interfaces and classes.

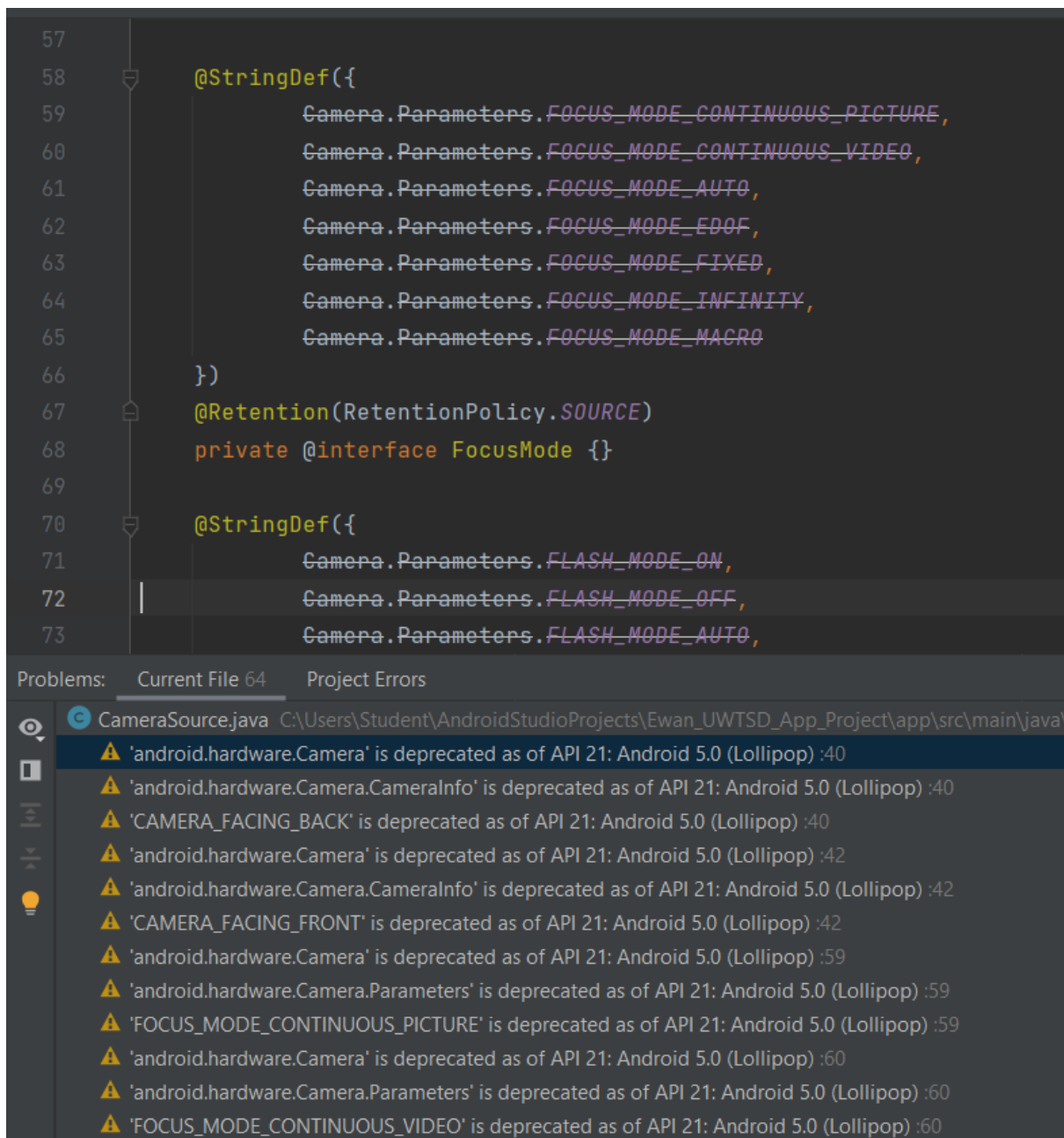


Figure 87: depreciated code

Figure 87: depreciated code (above) shows some of the code within the Camera Source java class. This is one of the main reasons that the QR codes are not being scanned, the code was based on an older version of Android Studio, the code has become depreciated, and it will not work with this current version of code.

Current Version	Old Version
Android Gradle Plugin Version 7.2.2 Gradle Version 7.3.3	Android Gradle Plugin Version 5.0.0 Gradle Version 5.0.0

Figure 88: reverting to the older Gradle versions

Figure 88: reverting to the older Gradle versions (above) shows that it was attempted to revert to an older Gradle version. When switched to the older version, an error message appeared saying that the specific Gradle distribution does not exist. Although changing the Gradle version, it is still showing the code as depreciated, so this test has not been successful. The Gradle version has now been reverted back to the current version.

Once the coding had been completed, there were many issues with the functionality with the app, such as the room search page. The ID number for the Student Services room would be entered and, rather than going straight to the student support page on the app itself.

6.2 Post Implementation

As the prototype was developed within the Android Development Studio, the software provided a virtual emulator device. For testing, both a virtual emulator device and the researchers' actual phone was used.

Two stages of usability tests were planned, firstly an interaction usability test using the wireframes from the design phase and a second usability test for the final implementation. However, owing to the fact that the back end of the application is still not fully functional at the time of submission, the developer decided that the second round of usability tests would not provide any meaningful results and, as such, these tests were abandoned.

The developer went back to the drawing board and developed a Graphical User Interface using Java and JDBC whilst integrating SQL Server as the back-end

development. The development was done incrementally across three sprints, the first one being to create the initial database in SQL Server and inserting data into the tables, the second sprint using JDBC and returning data via a console app when searching for a room and the final sprint was generating images and using a GUI, QR codes and enhancing the design further. Once the three sprints had been developed, it was decided to run some user tests to ensure that the application ran effectively, whilst collecting feedback about the application to gather future work and improvements.

6.2.1 Usability Tests – Wireframe Diagram

1. Did these features work?

[More Details](#)

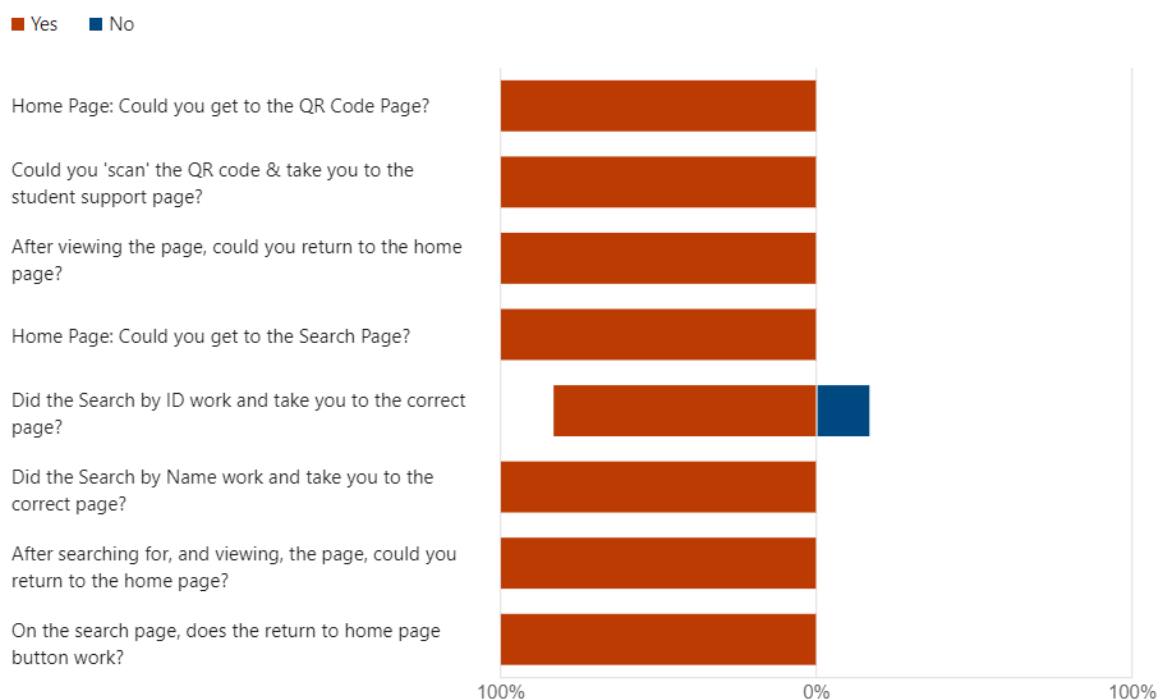


Figure 89: testing the features on the wireframe diagram

Figure 89: testing the features on the wireframe diagram (above) shows the results of the testing of the app. All the features worked for all participants, with exception to one participant where the Search by ID did not work as the numbers could not be entered into the search box.

When participants were asked whether there were any features, they particularly liked, many liked the QR code aspect and said that the incorporation of QR codes was cool

and it saves people putting in the URL of a specific page. Others felt that “the bold text and the images” were good as it is “vibrant and quite appealing” It was also “easy to use with the layout and the text was easy to read.” The app itself was “simple and clear and does not have additional clutter” it is “to the point and does what it says.”

When asked for any future work, participants said they looked forward to reviewing the final product once it was completed. Participants also mentioned that more visuals are needed as there was only 2 pictures, which were identical to each other, it would be useful to include all UWTSD campuses and utilise Google Maps to have a more accurate location within the building.

When asking the participants would the app assist them with having a tour of the campus, 100% agreed that it would assist. Participants said that “if I was looking for something / a room that would be great and the QR code is great if you are wondering what room it is and finding more information about it.” Others said that “if someone has a social anxiety, this app might appeal more to them” and “having this as an option would rather having the app than asking someone for a tour,” so the accessibility of the app has been praised.

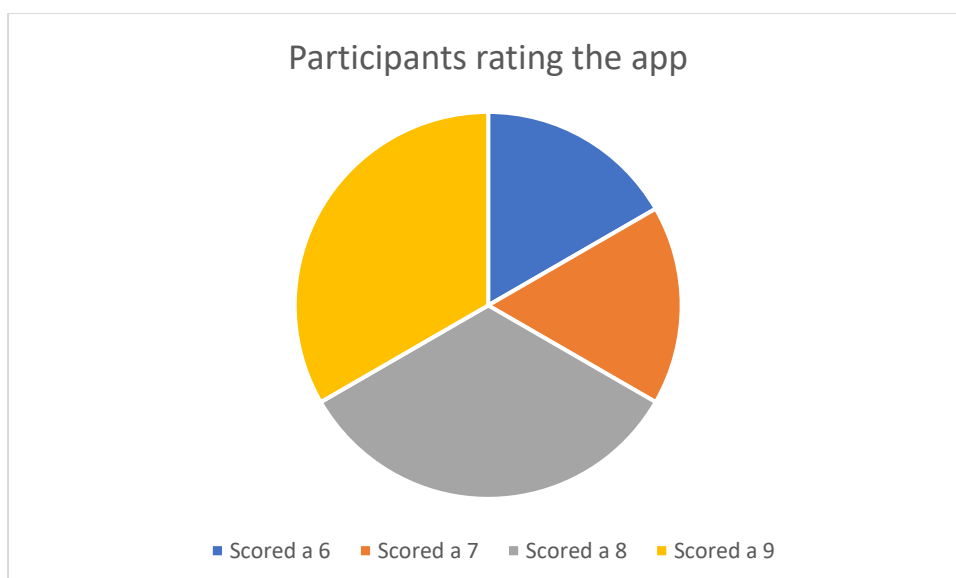


Figure 90: participants rating the app

Figure 90: participants rating the app (above) shows the chart of responses for the question. Users were then asked to rate the app out of 10, with 1 being bad and 10

being excellent. The average score was 7.8, with 1 participant scoring the app a 6, 1 participant scoring it a 7, 2 participants scoring the app an 8 and 2 participants scoring it a 9. Participants said that “more visuals would be great², they did not “give it a 10 as it is still a work in progress” and that the app generally was “good but there is room for improvement”. These comments will be investigated during development and attempted to be fixed.

6.2.2 Virtual Emulator Device (Android Studio)

Test Number	What is Being Tested	Expected Outcome	Actual Outcome
1	Launch the App	The expected outcome of this result is that the app should launch on the virtual device without any errors	The app was launched, the test was successful.
2	QR Code	Due to this being a virtual device, the camera would only work within a virtual room, which was provided by Android Studio. Within the settings itself for the device, the QR code for the Student Support page have been inserted on the room wall and table. The QR code should be able to be scanned from either the wall or table, and then redirect the user to the Student Support page.	The QR code for the Student Support page did not appear on the wall or table within the virtual room, so this test was unsuccessful.
3	Search by ID	The expected outcome of this test is that the user will be able to search for the room by its assigned ID number and the room page should appear. For the test, the room ID 000001 would be tested	As shown in Table 13: <i>Search by ID</i> , many ways of testing the room name was entered, however, either an error message appeared or it looped back to the home page, so this test was unsuccessful.
4	Search by Name	The expected outcome of this test	As shown in Table 14: <i>Search by Name</i> ,

		is that the user will be able to search for the room by its name and the room page should appear. For the test, Student Support will be tested.	many ways of testing the room name was entered, however, either an error message appeared or it looped back to the home page, so this test was unsuccessful.
5	Search by ID and Name	The expected outcome of this test is that the user will be able to search for the room by its assigned ID number and the room name, the page should then appear. For the test, the room ID 000001, and the room name Student Support, will be tested.	As shown in Table 14: <i>Search by Name</i> , many ways of testing the room name was entered, however, either an error message appeared or it looped back to the home page, so this test was unsuccessful.

Table 11: test plan for the virtual device

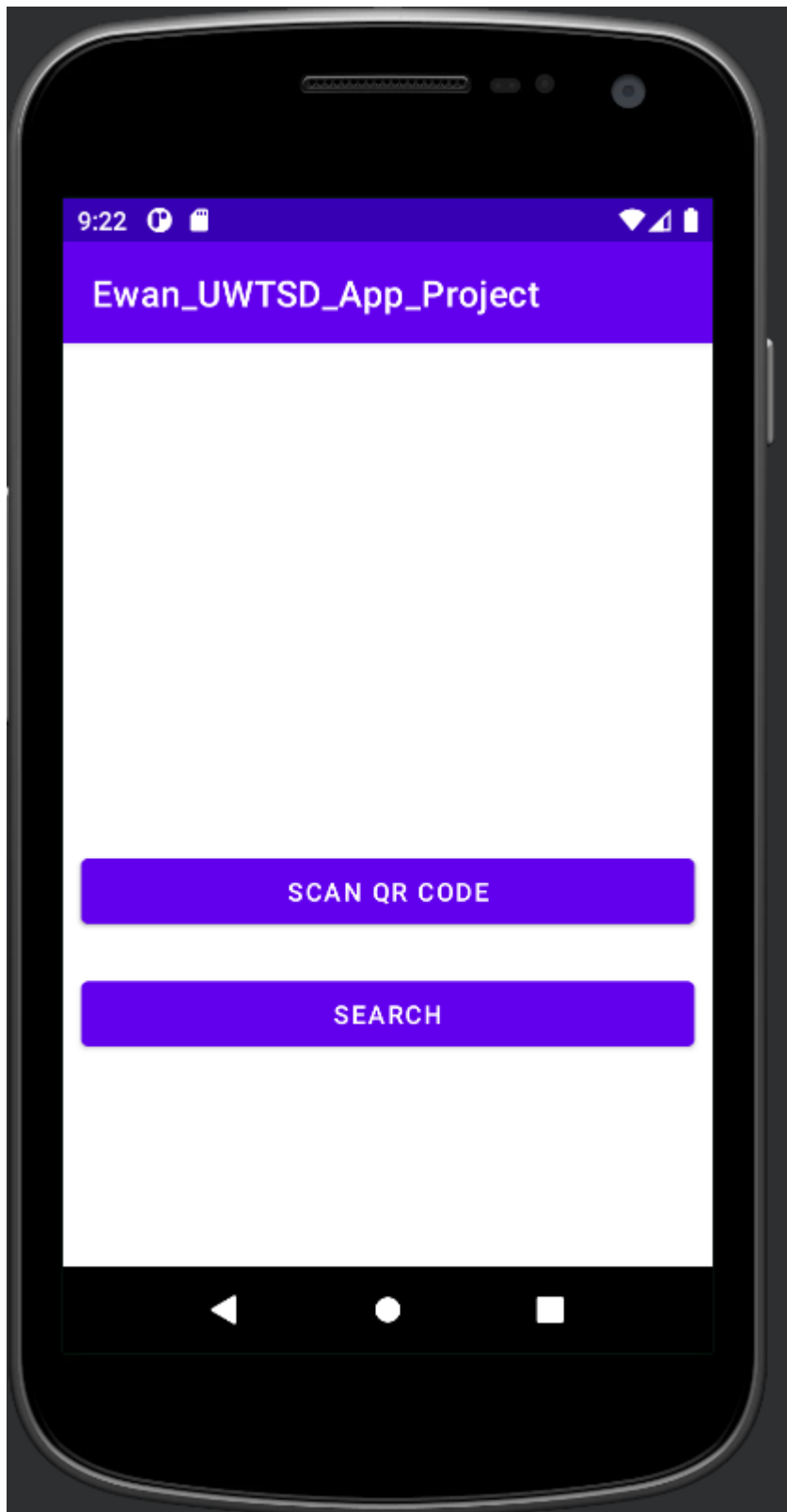


Figure 91: outcome of test 1

Figure 91: outcome of test 1 (above) shows the actual outcome for the first test (launching the app).

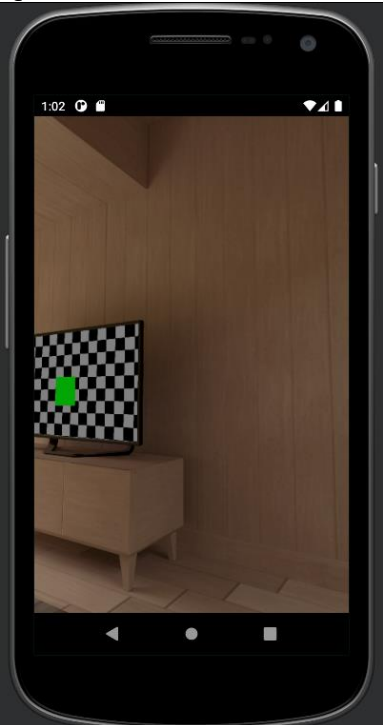
QR Code on the Wall	QR Code on the Table
	

Table 12: QR Codes on Virtual Device Camera

Table 12: QR Codes on Virtual Device Camera (above) shows the virtual room on the device, the wall and table were looked at and the QR code did not appear.

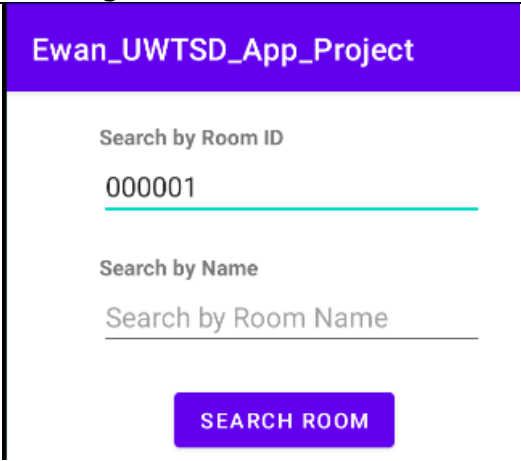
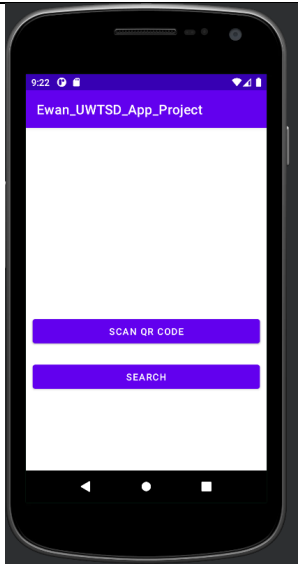
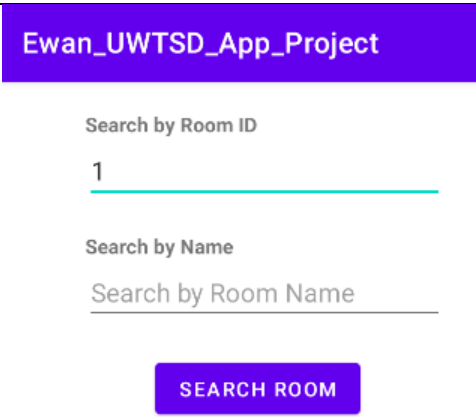
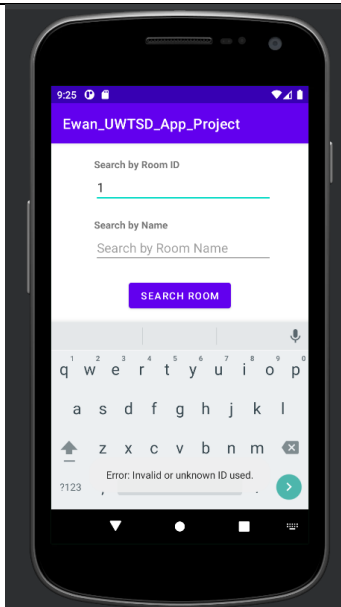
Test	Entering the ID	Outcome
Searching for Room 000001. Upon testing, the room associated with this ID did not appear, and the app looped back to the home page.		
Searching for Room 1.		

Table 13: Search by ID

Table 13: Search by ID (above) demonstrates the various ways the 'Search by ID' function was tested. Unfortunately, this function did not work as it should have done and it kept on looping back to the home page, and not taking the user directly to the room page.

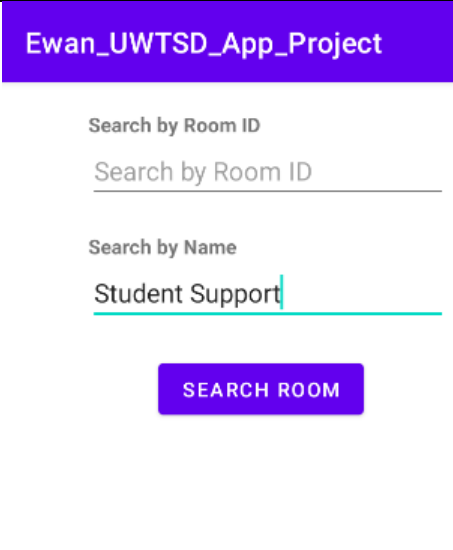
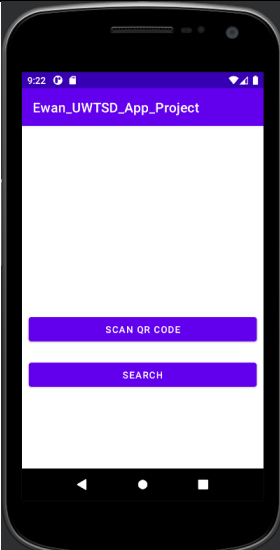
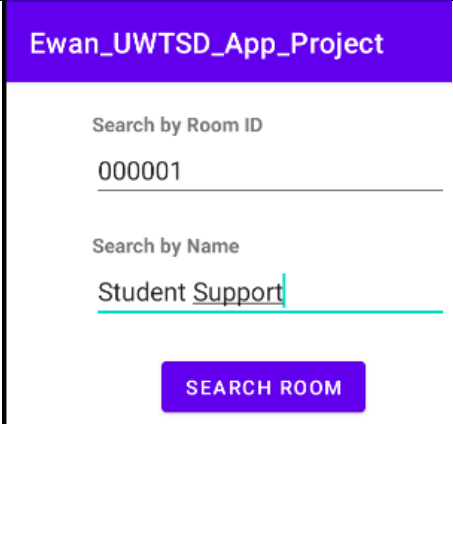
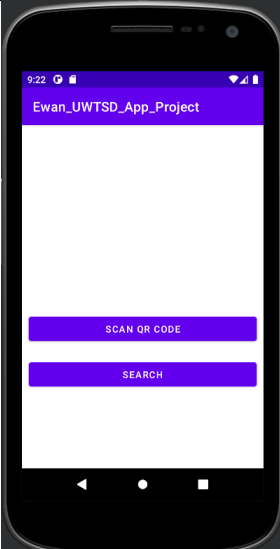
Test	Entering the room name	Outcome
Searching for room name 'Student Support.' Unfortunately, it did not work as it looped back to the homepage and provided an error message.		
Both Room ID (000001) and room name (Student Support) was entered into their relevant text boxes, and the test as unsuccessful as it looped back to the home page.		

Table 14: Search by Name

Table 14: Search by Name (above) shows the various ways the search by name was tested. Unfortunately, all tests were unsuccessful.

6.2.3 Physical Device: Huawei P8 lite 2017

Test Number	What is Being Tested	Expected Outcome	Actual Outcome
1	Launch the App	The expected outcome of this result is that the app should launch on the virtual device without any errors	The app was launched, the test was successful.
2	QR Code	A QR code has been created for the student support page on the app, the expected outcome of the test is that once the user scans the QR code, it would re-direct them to the student support page.	The QR code for the Student Support page did not appear on the wall or table within the virtual room, so this test was unsuccessful.
3	Search by ID	The expected outcome of this test is that the user will be able to search for the room by its assigned ID number and the room page should appear. For the test, the room ID 000001 would be tested	As shown in Table 17: <i>Search by ID</i> , many ways of testing the room name was entered, however, either an error message appeared or it looped back to the home page, so this test was unsuccessful.
4	Search by Name	The expected outcome of this test is that the user will be able to search for the room by its name and the room page should appear. For the test, Student Support will be tested.	As shown in Table 18: <i>Search by Name</i> , many ways of testing the room name was entered, however, either an error message appeared or it looped back to the home page, so this

			test was unsuccessful.
5	Search by ID and Name	The expected outcome of this test is that the user will be able to search for the room by its assigned ID number and the room name, the page should then appear. For the test, the room ID 000001, and the room name Student Support, will be tested.	As shown in Table 18: <i>Search by Name</i> , many ways of testing the room name was entered, however, either an error message appeared or it looped back to the home page, so this test was unsuccessful.

Table 15: test plan for the physical device

Ewan_UWTSD_App_Project

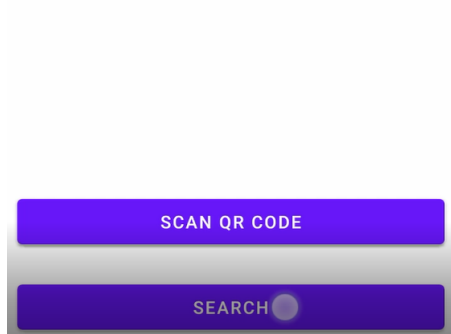


Figure 92: outcome of test 1

Figure 92: outcome of test 1 (above) shows the outcome of test 1, the app launched on the physical device so the test was successful.

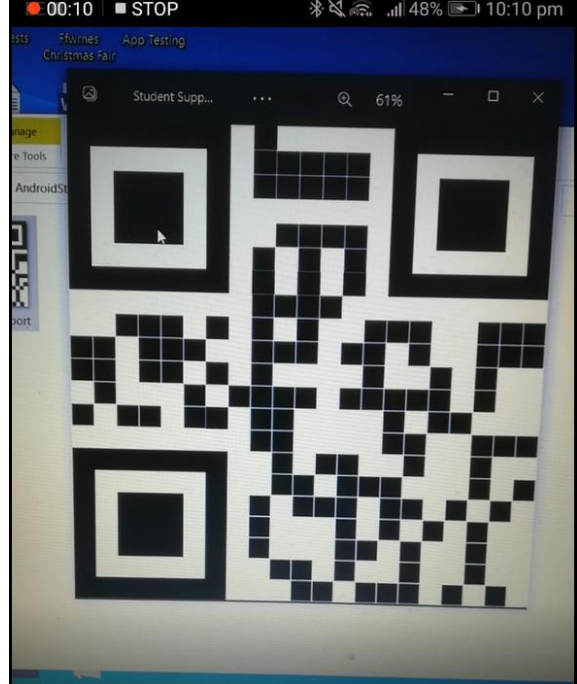
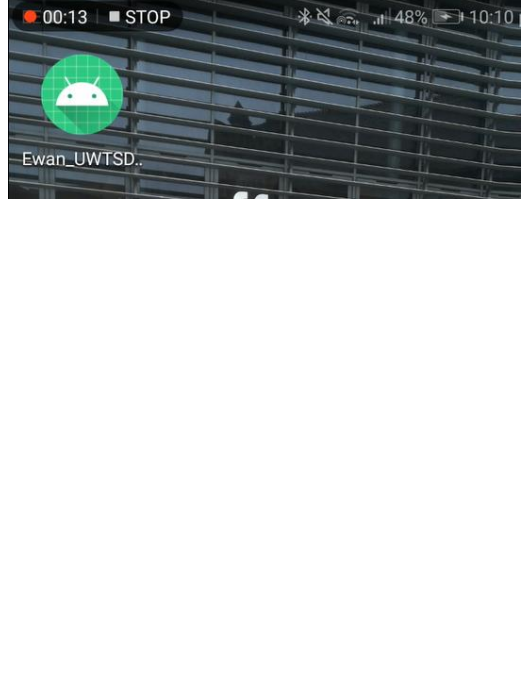
Scanning the QR code	The Result
	

Table 16: outcome of testing the QR code

Table 16: outcome of testing the QR code (above) shows the outcome of the QR code test. The QR code was scanned on the physical device however, upon scanning the QR code, the app crashed and return to the home page of the device.

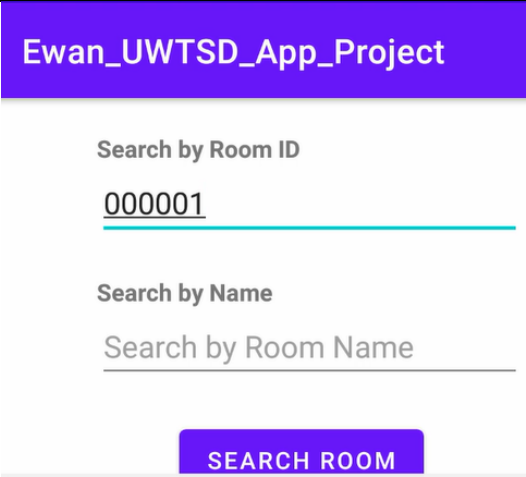
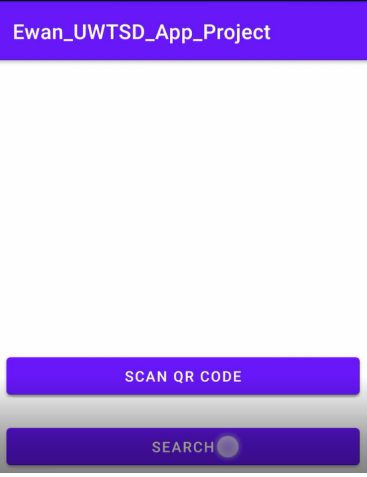
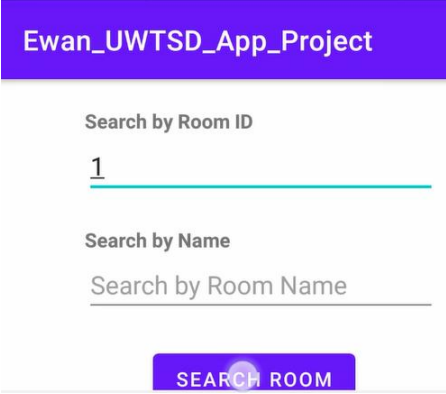
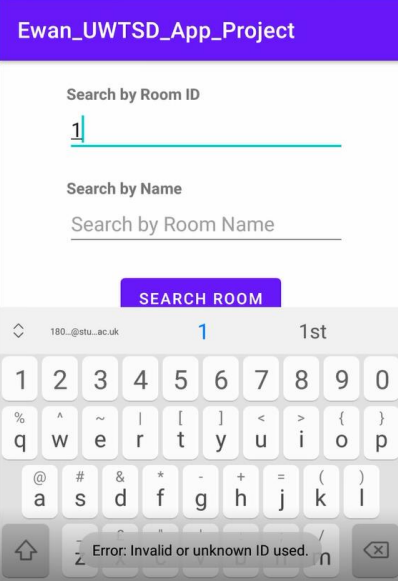
Test	Entering the ID	Outcome
Searching for Room 000001. Upon testing, the room associated with this ID did not appear, and the app looped back to the home page.		
Searching for Room 1. Upon testing, the room associated with this ID did not appear, and the app looped back to the home page.		

Table 17: Search by ID

Table 17: Search by ID (above) demonstrates the various ways the 'Search by ID' function was tested. Unfortunately, this function did not work as it should have done and it kept on looping back to the home page, and not taking the user directly to the room page.

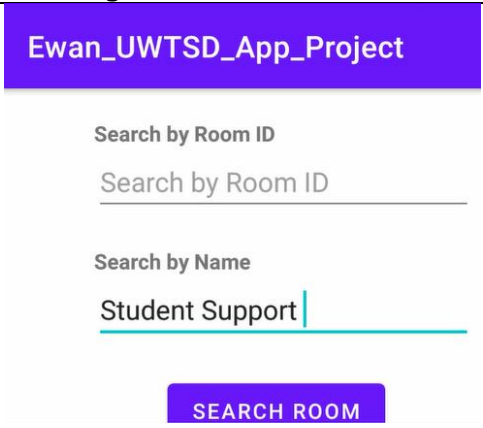
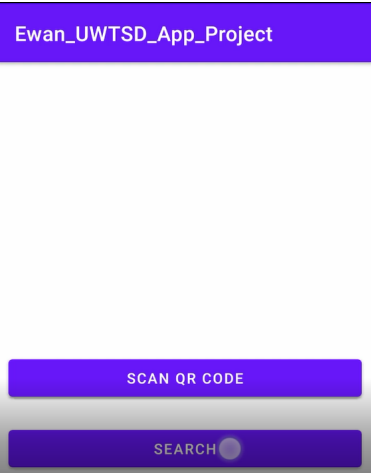
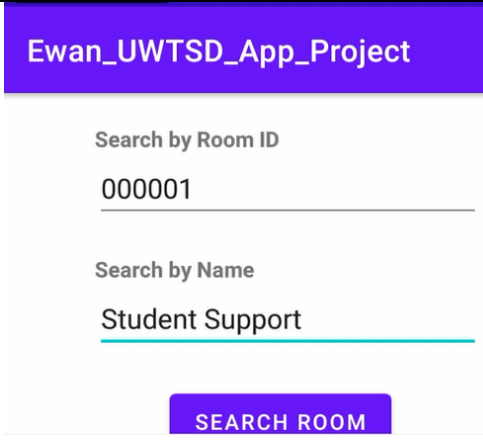
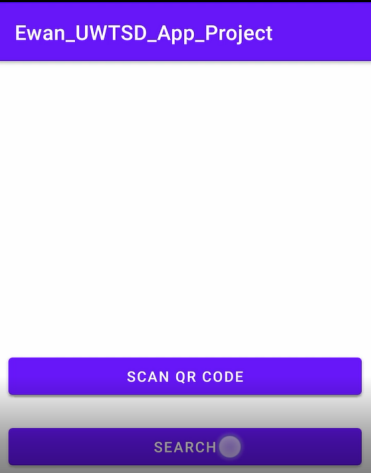
Test	Entering the room name	Outcome
Searching for room name 'Student Support.' Unfortunately, it did not work as it looped back to the homepage and no error message was provided.		
Both room ID (000001) and room name (Student Support) was entered into their relevant text boxes, and the test as unsuccessful as it looped back to the home page.		

Table 18: Search by Name

Table 18: Search by Name (above) shows the various ways the search by name was tested. Unfortunately, all tests were unsuccessful.

6.2.4 Researcher Self Testing of Java Application

Step Description	Test 1.1: test to...
Input Data	IQ104 into console application
Expected Results	Information, such as room details, software and hardware description and session (timetable) details should appear.
Actual Results	Information, such as room details, software and hardware description and session (timetable) details appear, the test is successful.
Result	

Step No: 1.1	Pass/Fail: Pass	Issues: n/a
	Correction Needed n/a	Fixed ☐ Inits/ Date n/a

Table 19: test plan for test 1.1

```

Please enter room number
IQ104
Connecting to database...
Creating statement...
Room Description: IQ104 10 Support Student Services
Hardware Description: Chairs 15 2022-12-01
Hardware Description: PC General Usage 30 2022-12-01
Hardware Description: Podium PC 1 2022-12-01
Hardware Description: Projector 1 2022-12-01
Hardware Description: Sockets 30 2022-12-09
Hardware Description: Tables 35 2022-12-01
Software Description: Dragon 2022-12-09
Software Description: Office 365 2022-12-09
Software Description: Read & Write 2022-12-09
Software Description: Sonocent 2022-12-09
Software Description: Unity 2022-12-02
Session Time: 3 2022-11-11 16:00:00.0000000 1
Session Description: Support David Williams Student Services
BUILD SUCCESSFUL (total time: 7 seconds)

```

Figure 93: Evidence of Test 1.1

Figure 93: Evidence of Test 1.1 (above) shows evidence of test 1.1 being carried out, this test was successful.

Step Description	Test 1.2: test to...
Input Data	FF002 in console application
Expected Results	Information, such as room details, software and hardware description and session (timetable) details should appear.
Actual Results	Information, such as room details, software and hardware description and session (timetable) details appear, the test is successful.

Result		
Step No: 1.1	Pass/Fail: Pass	Issues: n/a
	Correction Needed n/a	Fixed ☐ Inits/ Date n/a

Table 20: test plan for test 1.2

```

Please enter room number
FF002
Connecting to database...
Creating statement...
Room Description: FF002 30 Library Library
Hardware Description: Chairs 30 2022-12-08
Hardware Description: Podium PC 1 2022-12-08
Hardware Description: Projector 1 2022-12-08
Hardware Description: Sockets 20 2022-12-09
Hardware Description: Speaker Units and Amp 1 2022-12-08
Hardware Description: Tables 15 2022-12-08
Software Description: Office 365 2022-12-06
Session Time: 1 2023-02-09 11:00:00.0000000 2
Session Description: Lecture Gaynor Thomas Computing
Session Time: 3 2023-01-08 09:00:00.0000000 2
Session Description: Support David Williams Student Services
BUILD SUCCESSFUL (total time: 6 seconds)
|

```

Figure 94: Evidence of Test 1.2

Figure 94: Evidence of Test 1.2 (above) shows evidence of test 1.8 being carried out, this test was successful.

Step Description	Test 1.3: test to...
Input Data	IQ400 in console application
Expected Results	This room does not exist on the database as there is no room IQ400 at the University campus, so there should be no information about the room appearing.
Actual Results	The test is successful as no information about the room appears.

Result	Test successful.	
	Pass/Fail: Pass	Issues: n/a
	Correction Needed n/a	Fixed ☐ Inits/ Date n/a

Table 21: test plan for test 1.3

```

Please enter room number
IQ400
Connecting to database...
Creating statement...
BUILD SUCCESSFUL (total time: 4 seconds)

```

Figure 95: Evidence of Test 1.3

Figure 95: Evidence of Test 1.3 (above) shows evidence of test 1.3 being carried out, this test was successful.

Step Description	Test 1.4: test to...	
Input Data	FF100 in console application	
Expected Results	This room does not exist on the database as there is no room FF100 at the University campus, so there should be no information about the room appearing.	
Actual Results	The test is successful as no information about the room appears.	
Result	Test successful.	
	Pass/Fail: Pass	Issues: n/a
	Correction Needed n/a	Fixed ☐ Inits/ Date n/a

Table 22: test plan for test 1.4

```

Please enter room number
FF100
Connecting to database...
Creating statement...
BUILD SUCCESSFUL (total time: 5 seconds)

```

Figure 96: Evidence of Test 1.4

Figure 96: Evidence of Test 1.4 (above) shows evidence of test 1.4 being carried out, this test was successful.

Step Description	Test 1.5: test to...	
Input Data	IQ104 into search box	
Expected Results	Image, QR code and information (hardware, software and session details) relating to room IQ104.	
Actual Results	Image, QR code and information (hardware, software and session details) appears as expected.	
Result	Test successful.	
	Pass/Fail: Pass	Issues: n/a
	Correction Needed n/a	Fixed ☐ Inits/ Date n/a

Table 23: test plan for test 1.5

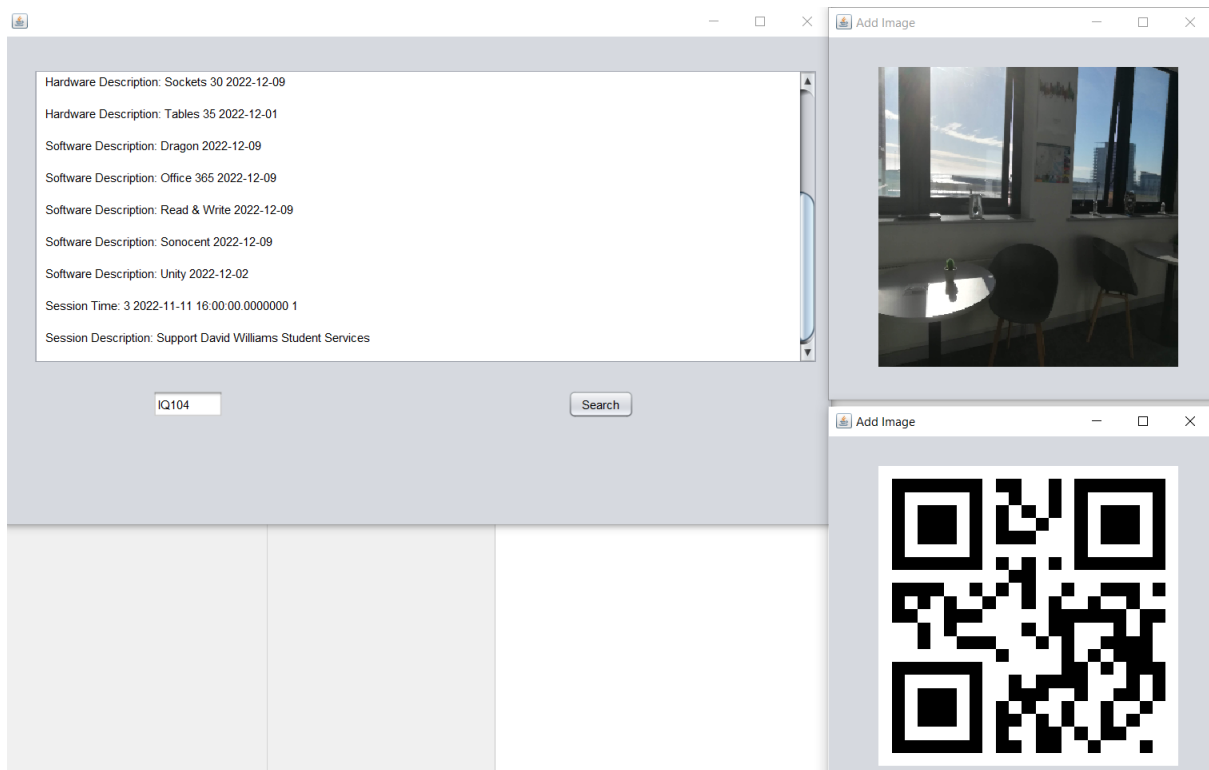


Figure 97: Evidence of Test 1.5

Figure 97: Evidence of Test 1. (above) shows evidence of test 1.5 being carried out, this test was successful.

Step Description	Test 1.6: test to...	
Input Data	FF002 in search box	
Expected Results	Image, QR code and information (hardware, software and session details) relating to room FF002.	
Actual Results	Image, QR code and information (hardware, software and session details) appears as expected.	
Result	Test successful.	
	Pass/Fail: Pass	Issues: n/a
	Correction Needed n/a	Fixed ☐ Inits/ Date n/a

Table 24: test plan for test 1.6

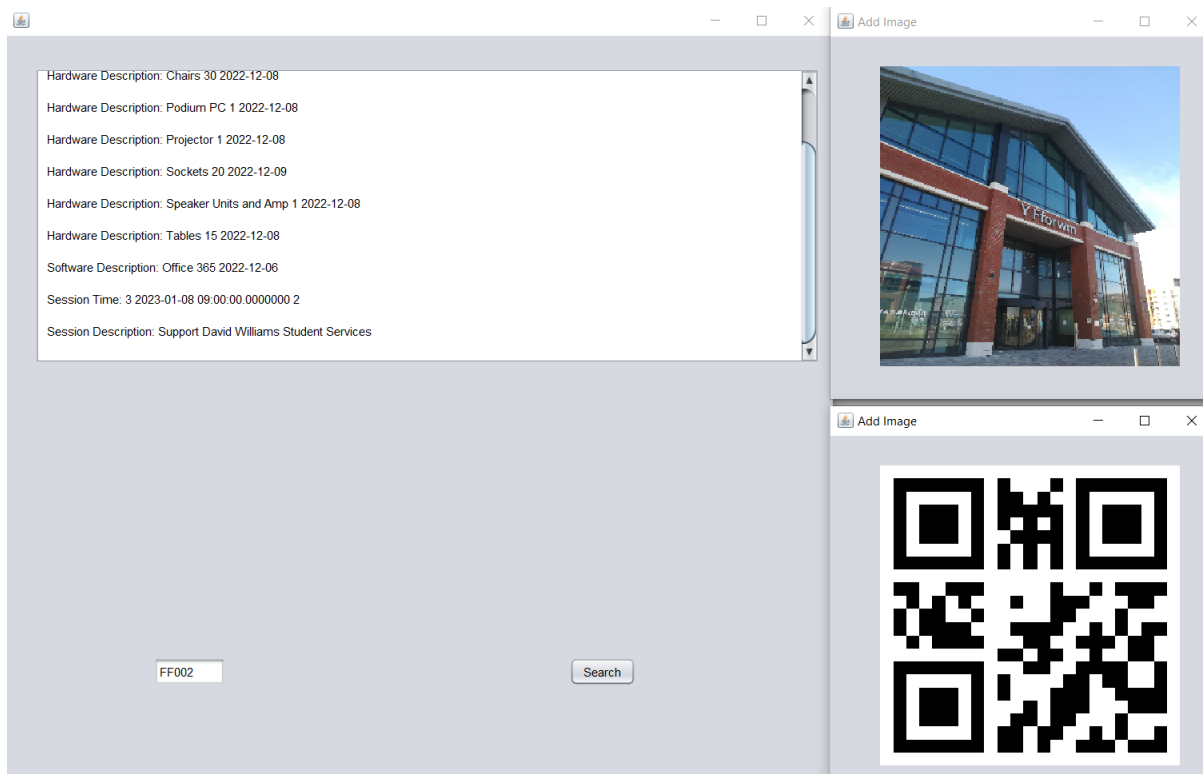


Figure 98: Evidence of Test 1.6

Figure 98: Evidence of Test 1. (above) shows evidence of test 1.6 being carried out, this test was successful.

Step Description	Test 1.7: test to...	
Input Data	IQ400 in search box	
Expected Results	This room does not exist on the database as there is no room IQ400 at the University campus, so there should be no text (information about the room), image or QR code appearing.	
Actual Results	The test is successful as no text, images or QR codes appears.	
Result	Test successful.	
	Pass/Fail: Pass	Issues: n/a
	Correction Needed n/a	Fixed ☐ Inits/ Date n/a

Table 25: test plan for test 1.7

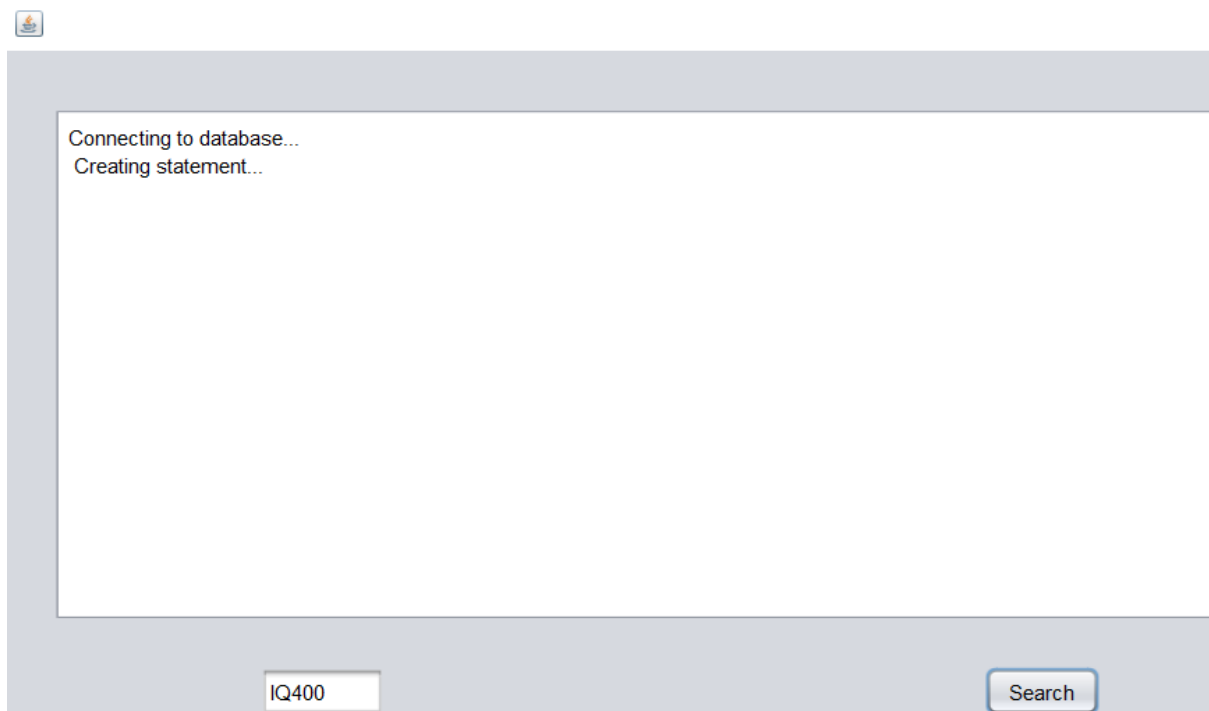


Figure 99: Evidence of Test 1.7

Figure 99: Evidence of Test 1. (above) shows evidence of test 1.7 being carried out, this test was successful.

Step Description	Test 1.8: test to...	
Input Data	FF100 in search box	
Expected Results	This room does not exist on the database as there is no room FF100 at the University campus, so there should be no text (information about the room), image or QR code appearing.	
Actual Results	The test is successful as no text, images or QR codes appears.	
Result	Test successful.	
	Pass/Fail: Pass	Issues: n/a
	Correction Needed n/a	Fixed ☐ Initis/ Date n/a

Table 26: test plan for test 1.8

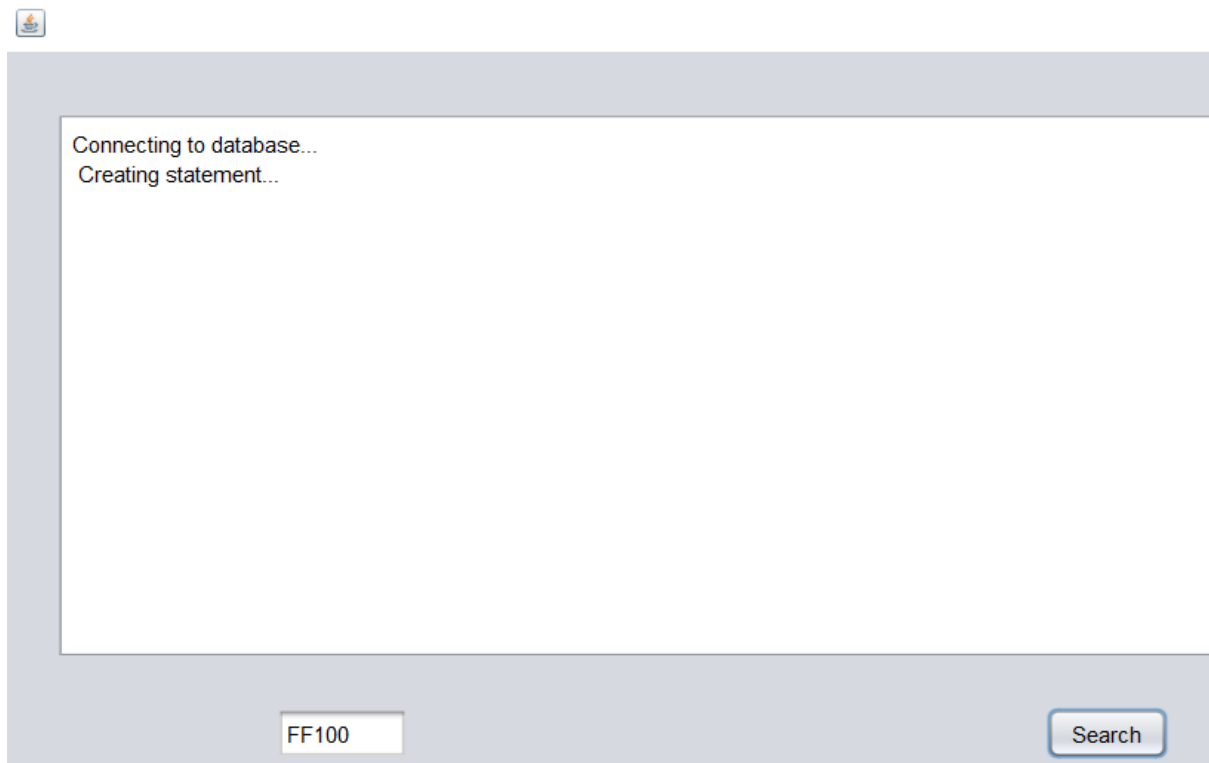


Figure 100: Evidence of Test 1.8

Figure 100: Evidence of Test 1. (above) shows evidence of test 1.8 being carried out, this test was successful.

6.2.5 Java Application Results

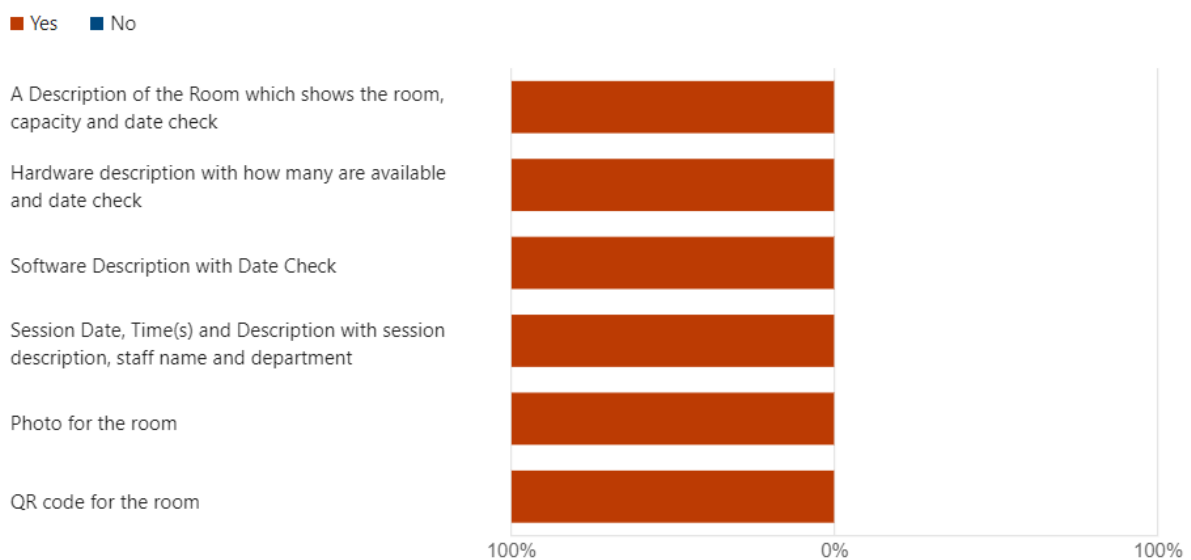


Figure 101: Chart for question 1: testing Room IQ104

Figure 101: Chart for question 1: testing Room IQ104 is a chart showing the responses for question 1, where participants were entering room IQ104 into the app. Information about the room, photo of the room and QR code should appear on the screen.

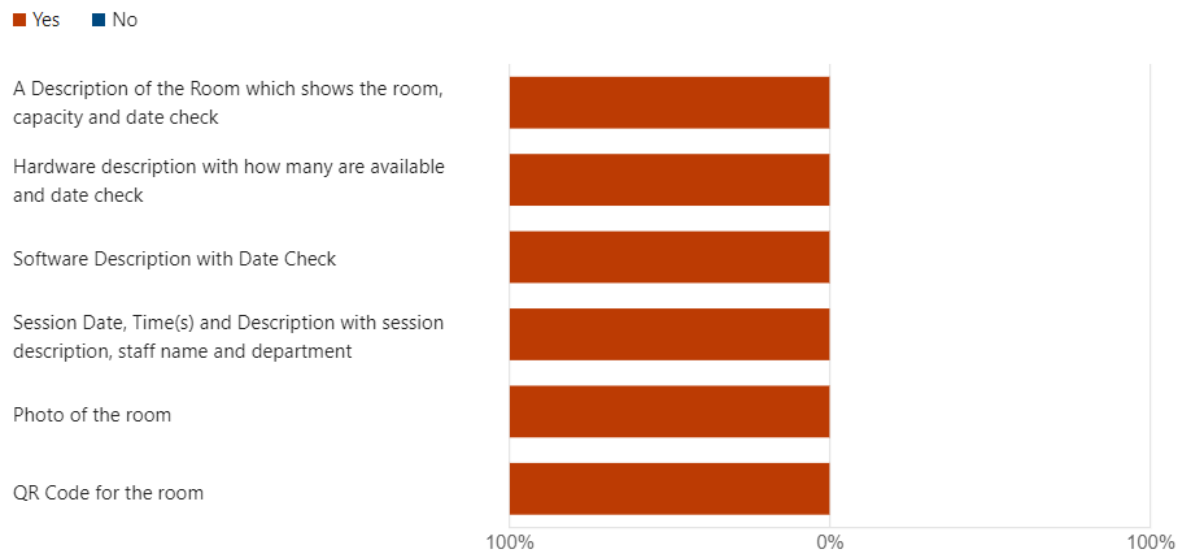


Figure 102: Chart for question 2: testing room FF002

Figure 102: Chart for question 2: testing room FF002 is a chart showing the responses for question 2, where participants were entering room FF002 into the app. Information about the room, photo of the room and QR code should appear on the screen.

3. Try entering in room IQ310, this room has not yet been added to the application, so should thrown back an error. Does it throw back an error or is anything displayed?

[More Details](#)

● Yes 0
● No 14



4. Try entering in room FF105, this room has not yet been added to the application, so should thrown back an error. Does it throw back an error or is anything displayed?

[More Details](#)

● Yes 0
● No 14



Figure 103: chart for question 3 (testing rom IQ310) and testing question 4 (testing FF105)

Figure 103: chart for question 3 (testing rom IQ310 is a chart showing the responses for question 2, where participants were entering room IQ400 into the app. There are no rooms called IQ400 and FF105 have not yet been inserted onto the database. Nothing should be displayed when entering these rooms.

Participants were then asked was there any features, or anything specific with the app, that they particularly like. The images are visually appealing, they give a good visual description of the room. Having information such as how may sockets, the capacity of the room – knowing this type of information is very useful. According to the interview participants, they thought that the app itself was “very responsive” and that it “opened up additional windows for the images and QR codes”, it was “good to see those separated from the text”. Some participants also thought that the app was “simplistic” and the app “implements all different types of learning styles”. The app also informs users of where you are and where the room is on a map, by seeing a visual helps you navigate around the campus. Some participants also stated that to

“see when the next sessions it” they thought it was “quite handy especially when you’re working out if that is your next session so that is quite useful”.

The next question was ‘do you have any improvements you would like to suggest?’. Some participants feel that having more than one photo per room would be better, with the potential of having one photo from outside and one of the inside of the room. Some features were not that easy to use, such as the inserting of room numbers into the text box. This should be rectified by a simple label to say ‘input room number here’. Some colours could be added to make it more interesting as, although it is good the way it is as it is minimal, but do not want to make it over the top. The information box should be more visually pleasing, rather than all the text be in the same font, make some in bold, and some in italic. The title of the page and information about the room should be in a different font, this would make it more eye catching. Make sure that the information is translated, due to the University being an organisation being based in Wales. For buildings such as IQ (IQ Building) or FF (Y Fforwm Library), have a page where these are explained so then new students and staff members know what this mean and where they are located.

Participants were then asked would the app assist them with having a tour of the campus. For most of the participants, they believe that this app would assist them with having a tour of the campus. The room search feature is useful as you could give a more detailed response to visitor enquiries as you can see the capacity, software and hardware description and timetabled sessions as well as it is assisting the user with navigating where to go, especially if they are not use to attending the campus. The visuals are of huge benefit and to improve, include the refuge points and access arrangements for wheelchair users. Some supplementary materials would be required such as if a guide to the rooms were provided, that would be useful to students and staff. Some participants feel that if there was an initial landing page with all the rooms listed on there, that would help users in using the app and understanding where the rooms are located.

The next question asked to participants was 'was the look and feel of the app acceptable for its purpose?'. Many participants feel that the look and feel of the app is acceptable for its purpose. Some feel that to make it more acceptable, the University branding could be added alongside some design changes. By making these changes, it would make the app more corporate. Some participants feel that the text box where users enter in the room number could be make clearer and bigger alongside the text being a bigger font and size but, other than that, participants feel that the look and feel of the app is acceptable.

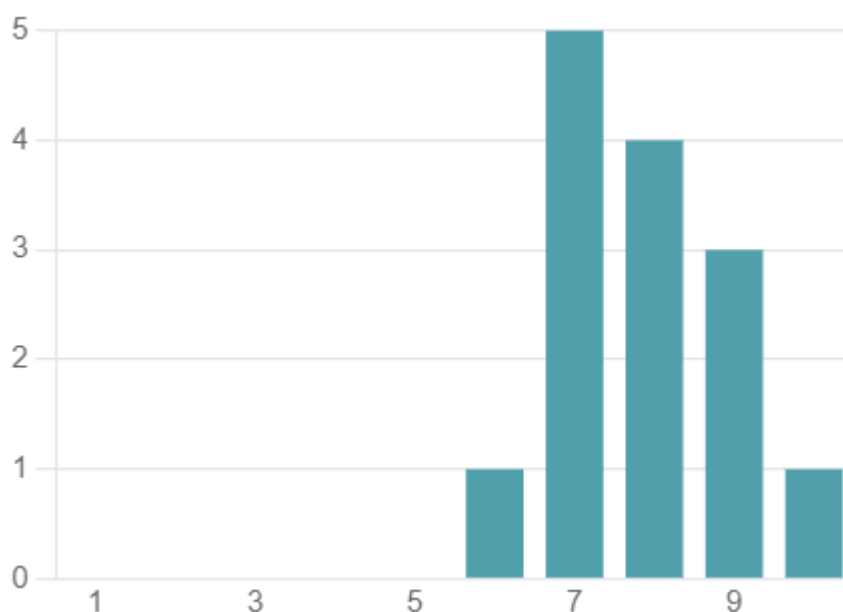


Figure 104: Chart of App Ratings

Figure 104: Chart of App Ratings (above) shows the responses given for the app rating question. The final question was, with 1 being poor and 10 being excellent, how would participants rate the app. The average score given for this question was 7.89, with 7% of participants scoring it a 6, 36% of participants scoring the app a 7, 29% of participants scoring it an 8, 21% of participants scoring the app a 9 and 7% of participants scoring it a 10.

Some future works have been identified from the interviews, some of which have been identified below:

- Make the text clearer.
- Use the University branding to make it more corporate, such as font, size, colours and logo.
- Implement the scanning of QR codes.
- Implement stored procedures to eliminate SQL injection.
- Make it more mobile friendly.
- Have more than one image per room.
- Store images in the database rather than image locations.
- Have a map of each campus on the app.
- Mention accessibility information, such as where the lifts are located.
- Make the form easier to use and clearer.

CHAPTER 7 – CONCLUSION AND RECOMMENDATIONS

A wireframe diagram was created as an initial prototype, and user feedback was collected. Although the feedback from interview participants were positive, there were some areas that users would like to see changed, particularly with visuals and colours to match the University branding.

The pre implementation interviews (chapter four) revealed some of the reasons in why some people may not have attended University open days and, for those who did, what caused them to have an unenjoyable user experience. This also discusses what features, in the views of the participants interviewed, would like to be seen on the application itself.

Once the app had been completed, it was tested to see whether the features worked. The home page interface worked, and the buttons the individual pages worked successfully however, the QR codes were not scanned as they should have done and the search functions for searching for rooms just looped back to the home page, so these tests were unsuccessful as the functions did not do as they were supposed to do.

As a result of the issues discovered during both development and testing, the researcher went back to the drawing board to review what needed to be changed. Because of time constraints at this point of the project, it was decided to create a scaled down version of the app using JDBC and create an app where users could only search for a room, rather than the intended QR code scan and view.

A database, with relational tables in third normal form, was created which contained data about the rooms in the IQ Building, staff name and department, session information and pictures. Using SQL code and database connection information, the code was successfully connecting to the database. At this point, a basic console application was developed which allowed users to enter in a room number and it shows details in relation to that room such as capacity, hardware and software information as well as timetabled sessions.

After the console application had been created, it was decided to enhance the application further and add images as well as improving the design, by creating a form to show the data in a clearer format. After further implementation, a text area was created to show all the details in relation to that room as well as images for each room and the associated QR image. A QR code was created for each room in preparation to move onto the next version of the app.

Not all the aims and objectives have been met. The use of GIS, Bluetooth and QR codes have been researched and two similar applications to see how they work have been analysed. Just one location was chosen within the IQ Building, with one QR code created for that location. The application was built using the Java programming language but, unfortunately, it could not be set up or tested during the time frame of the project due to a number of errors within the back-end development of the code.

Prior to commencing implementation, an analysis of both the Glynn Vivian Art Gallery and Gower Heritage Centre self-guided tour apps were conducted. Both apps contrasted each other as although there were problems using each of the apps, there were some differences in the types of errors being given due to the way the app is used and was setup. With the Glynn Vivian Art Gallery app, the ID numbers on the app of each room did not line up to the actual room numbers in the gallery itself. The Gower Heritage Centre regularly gave the incorrect location whilst using the app, it uses Bluetooth Beacons across the site and, due to potentially poor signal or the beacons not responding in time, I could be elsewhere within the centre and it says where the user was a few minutes ago, so there was a time delay in providing the current location.

The front end development went well, but there were problems with the back-end developing initially with the implementation of Bluetooth and then eventually with QR codes, which then affected the gathering of post-implementation results. It should be noted, however, that the negative results gained from this project have acted as a positive learning curve for the author.

Wireframe diagrams were implemented during the design phase and these were used as a method to gather results from usability testing with several individuals. Improvements have been suggested from interview participants and have been noted for potential future development.

CHAPTER 8 – REFLECTION

I have utilised Gibbs Reflective Cycle (Gibbs, 1988, cited in Jasper, 2013) for this reflection [82].

A prototype has been created however, many of its functions are not working as expected. The main concepts can be shown, but in terms of functionality, both the QR codes and search features do not work. I am quite disappointed that the functions did not work as planned but, due to the number of errors that came up, it was difficult to resolve them all by the project deadline. The good outcome from the experience is that I managed to develop my skills further within both Android Studio and the Java programming language. Java was not a programming language I was that familiar with but, following the Object-Oriented Programming module as well as this development for the project, I have refined my skills further. If anything else could have been done, it would have been allowing some more time using Android Studio, as I switched from the Ionic Framework to Android Studio quite quickly and I had not used this software for some time. If the project arose again, I would conduct reading of literature, create simple applications on Android Studio to get used to operating the software and continue to test existing applications to see how they work.

At the start of the project, it was hoped that the app could be used with Bluetooth Beacons. Because of its capability to work with Bluetooth, the Ionic Framework was researched and online tutorials were attempted. Due to the Ionic Framework being mostly front-end development work, rather than implementing code on the back-end of the app, so it was becoming quite difficult to program the Bluetooth Beacons to their assigned pages on the room as well as to create functionality to create a camera function to scan QR codes.

It was then decided to purchase an Arduino device and program it from scratch, Arduino has its own integrated development environment (IDE) however there were some initial problems with this. The first problem was that I realised that additional equipment was required to make the Arduino device run as a beacon, whilst exploring for this additional equipment from the Arduino store, a student kit was discovered,

however it was sold out and was unable to be purchase these items. Also, due to not understanding the IDE environment or the code it was decided to change direction with the project.

Other Bluetooth Beacons were researched and found on online stores however, time was not on my side and at this point, it was the beginning of August and very little of the practical work still was not being achieved, so an alternative project was thought about, which included the Google Maps features to create a tour app of local landmarks within the Swansea area. There were advantages of creating this app, mainly that there were more learning resources available however, due to travel constraints that I could not travel to the different locations and testing could be a problem so this idea was not done.

It was then decided to switch to QR codes rather than Bluetooth Beacons as this could be achieved within the timeframe of the project. By using QR codes, these could be created to link up to the page in the app so should be easy to implement.

There were a number of problems coding within the Android Studio environment, one of the biggest problems was with the camera. Many of the code within the camera classes used was depreciated within the current version of Android Studio, it was attempted to updated the Gradle version of the app to make it the newest version, however the error remained the same. The code was compatible with Gradle version 5, it was also attempted to revert to level 5 however, this caused further complications and more errors were noticed. If I had more time, I would have used a different technology, and it would have been Visual Studio. This technology would have been used because I am more familiar with it and although Android Studio is recommended to be used for mobile app development.

Overall, I am pleased that a working prototype using JDBC that enables users to search for a room within the SA1 Campus (in future work it could work on any campus with the same logic) was successfully built. This successfully broadcasts data in relation to the room such as the type of room it is, which department it is in, description of the

hardware, equipment and software as well as timetabled sessions in that room. Some errors did occur during development such as not being able to connect with SQL Server and some issues within the Java programming language which took some time to fix. The application development went well, with workable code and code that can be demonstrated being created. Three sprints have been completed, one being to create the database, the second being to create the app and the third generating the images and QR codes in preparation for further development to incorporate scanning QR codes into the application. My next iteration would be to try and get this app working in Android Studio and incorporate the use of QR codes via a mobile phone camera. Two further issues that would be beneficial would be to firstly, storage of the images directly into the database rather than the file locations. Secondly, the ability to use stored procedures on the back-end (SQL Server) rather than straight SQL to eradicate the possibility of SQL injection, however, if the search was conducted via a scan rather than the entry of a code this might not be such a large concern.

My research skills have developed immensely during this project. Over the past year, I have learnt about the different types of resources that are out there, especially with journal resources and library databases, as I have used these for past modules during the Masters degree at UWTSD, I was confident in using the online library systems and other online book sites, and found some sound academic sources. During the project, I have learnt about how Bluetooth is used and how it has developed, how QR codes are effective in day-to-day life and how both these technologies play their part in the modern world. I have also developed my skills in the physical web, how GIS, GPS, and IPS work and have considered the principles of mobile app design, ensuring that my final application meets these principles.

If anything else could have been done different, I think finding more journals on how self-guided tour apps are used, and can help with campus familiarisation and student recruitment within universities would have assisted I find out recommendations of where they could be improved as well as having more closed questions within the pre-implementation interviews to determine app requirements.

Testing was carried out on the prototype to determine what features worked and what improvements could be made.

As a result of my ability to handle the tasks that needed to be finished for the project, my personal motivation and time management have improved. I set aside time close to the project's due date to try to remedy any mistakes. There were times when I thought some faults were insurmountable, but I managed to stay focused and fixed some of the mistakes that were found prior to the usability testing.

Because of my weekly meetings with my project supervisor, when I listened to what they had to say about my work and replied to their suggestions for changes, my communication and teamwork skills have improved. I also spoke to other computing lecturers to assist with resolving errors, listening to how they would go around resolving the errors. Along with listening to interviewees, I also addressed their suggestions for how to strengthen the application. This would help me in my future profession because there would be management inside an organisation that would assign duties and need careful listening. As I spoke with participants about my project and questioned them about my application during virtual interviews, my communication skills have grown throughout the project. I struggle with communication because of my learning challenges, but I have gotten over it. Expand on this

My academic writing has continued to improve during this academic year. My referencing skills have particularly improved. I have improved my engagement with literature skills as I feel that I have engaged with a wide range of research-informed literature and my references have been noted as relevant and credible, with good use of referencing during my supervision meetings. Due to the adherence to academic writing, this has also benefited my writing for other assignments.

As I had to gather information from the interviews along with interpreting and analysing it to draw conclusions about what was good and negative about what the participant was rating, my research and analytical skills have increased. I used to find

this challenging because I had not applied this ability frequently during my academic studies, but now that I have learned it, I will be able to utilise it in my future job should any further data or feedback need to be analysed at work.

My Java skills have been improved as, even though it was already learned in a prior module, new functions and coding were learned for the prototype. I was eager to study this because I am passionate about application development. My main weakness is problem solving, therefore when issues developed during the rollout of the self-guided tour app, I personally worried about them.

The initial tasks of the project, such as the introduction, aims and objectives, literature review and research and development methodologies were mostly completed within the original timeframe set out in the research proposal. Due to the analysis of existing applications taking longer than expected, this impacted the implementation stages. The pre-implementation interviews also took longer than anticipated due to initially not finding enough participants and then a lot becoming available at once, which meant doing a lot of interviews within a pace of just a week and then writing up the results. The implementation was heavily impacted due to the number of errors within the back-end development which caused the app not to be fully functional, making pre-implementation a near impossible task. The app was tested however it was discovered that only the interfaces of the app worked, whilst the main functions did not. The conclusion and project reflection was completed quicker than initially planned, so some time was gained from those tasks. Due to the number of issues within development, it was agreed that the project would be completed by Wednesday 14th September, with the final viva taking place week commencing Monday 19th September. This is an example of scope creep.

Overall, the project has been moderately successful and I have been able to follow the timeline of work as set out on the Gantt chart and, conduct interviews where requirements have been collected. I have benefited from this project as I have learnt new skills and have learnt how Bluetooth, Beacon technology and QR codes can aid with University tour apps.

CHAPTER 9 – REFERENCES

- [1] Physical Web, "Setting up Physical Web in Chrome," Physical Web, 2022. [Online]. Available: <https://google.github.io/physical-web/try-physical-web#:~:text=The%20Physical%20Web%20enables%20you,working%20to%20>. [Accessed 9 June 2022].
- [2] Physical Web, "The Physical Web," Physical Web, 2022. [Online]. Available: <https://google.github.io/physical-web/>. [Accessed 9 June 2022].
- [3] S. O. Sullivan, "Google's Physical Web vs Apple's iBeacon," Radar, 27 April 2015. [Online]. Available: <http://radar.oreilly.com/2015/04/googles-physical-web-vs-apples-ibeacon.html>. [Accessed 15 March 2022].
- [4] H. A. Karimi, *Advanced Location-Based Technologies And Services*, New York: Taylor And Frances, 2013.
- [5] University Consortium for Geographic Information Science GIS&T Body of Knowledge, "CP-12 - Location-Based Services," University Consortium for Geographic Information Science GIS&T Body of Knowledge, 2022. [Online]. Available: <https://gistbok.ucgis.org/bok-topics/location-based-services>. [Accessed 13 March 2022].
- [6] M. Werner, *Indoor Location-Based Services: Prerequisites and Foundations*, Munich: Springer, 2014.
- [7] National Geographic, "GIS (Geographic Information System)," National Geographic, 2022. [Online]. Available: <https://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/#:~:text=GIS%20can%20show%20many%20different,and%20understand%20patterns%20and%20relationships..> [Accessed 10 March 2022].
- [8] H. Li, F. Jin-hong and F. Xiao-ling, "Research and Implementation of Mobile Navigation System Based on GPS and GIST," in *The 1st International Conference on Information Science and Engineering*, Beijing, 2009.
- [9] GISGeography, "What is Geographic Information Systems (GIS)?," GISGeography, 20 June 2022. [Online]. Available: <https://gisgeography.com/what-gis-geographic-information-systems/>. [Accessed 12 July 2022].
- [10] L. Tate, "20 Ways GIS Data is Used in Business and Everyday Life," Nobel Systems, [Online]. Available: https://nobelsystemsblog.com/gis-data-business/#1_Mapping. [Accessed 16 March 2022].
- [11] Korem, "MapInfo | Location Data Software," Korem, 2022. [Online]. Available: <https://www.korem.com/product/mapinfo/#:~:text=MapInfo%20Pro%20is%20the%20premier,a%20familiar%20Windows%20user%20interface..> [Accessed 7 July 2022].
- [12] Esri, "What is ArcGIS Online," Esri, 2022. [Online]. Available: <https://doc.arcgis.com/en/arcgis-online/get-started/what-is-arcgis.htm#:~:text=ArcGIS%20Online%20is%20a%20cloud,being%20mobile%20in%20the%20field..> [Accessed 4 July 2022].

- [13] D. Smith, N. Strout, C. Harder, S. Moore, T. Ormsby and T. Balstrom, Understanding GIS: An ArcGIS Pro Project Workbook, 3rd Edition, 2017.
- [14] C. Dore, "Benefits of working with ArcGIS Online," Esri Canada, 16 April 2020. [Online]. Available: <https://resources.esri.ca/getting-technical/benefits-of-using-arcgis-online>. [Accessed 4 July 2022].
- [15] GIS In Ecology, "GIS Software," GIS In Ecology, 2022. [Online]. Available: <https://gisinecology.com/gis-software/>. [Accessed 19 July 2022].
- [16] Brooklyn College Library, "Geographic Information System and Data Visualization Applications: Google Earth and Google Maps," Brooklyn College Library, 25 April 2022. [Online]. Available: https://libguides.brooklyn.cuny.edu/GIS/google_earth_maps#:~:text=Google%20Maps%20is%20probably%20the,of%20routes%20and%20journey%20times.. [Accessed 25 July 2022].
- [17] A. Jin, "Google Maps vs GIS Maps - What to say at your next Dinner Party," LinkedIn, 26 June 2019. [Online]. Available: <https://www.linkedin.com/pulse/google-maps-vs-gis-what-say-your-next-dinner-party-alice-jin#:~:text=Google%20maps%20only%20show%20you,you%20can%20predict%20the%20future..> [Accessed 11 July 2022].
- [18] Eric L Sparks, "QGIS: An Introduction to an Open-Source Geographic Information System," Mississippi State University, 2022. [Online]. Available: [http://extension.msstate.edu/publications/qgis-introduction-open-source-geographic-information-system#:~:text=QGIS%20\(Quantum%20Geographic%20Information%20System,and%20maps%20available%20to%20download..](http://extension.msstate.edu/publications/qgis-introduction-open-source-geographic-information-system#:~:text=QGIS%20(Quantum%20Geographic%20Information%20System,and%20maps%20available%20to%20download..) [Accessed 4 July 2022].
- [19] A. Graser, B. Mearns, A. Mandel and V. O. F. A. Bruy, QGIS: Becoming a GIS Power User, Birmingham - Mumbai: Packt Publishing, 2017.
- [20] GPS 2 GPS, "What is GPS?," GPS 2 GPS, 2022. [Online]. Available: <https://www.gis2gps.com/GPS/GPSDEF/gpsdef.html#:~:text=GPS%2C%20which%20stands%20for%20Global,conditions%2C%20anywhere%20in%20the%20world..> [Accessed 16 March 2022].
- [21] C. Gullo, "Leveraging location in consumer health apps," Movi Health News, 14 October 2011. [Online]. Available: <https://www.mobihealthnews.com/13755/leveraging-location-in-consumer-health-apps>. [Accessed 25 March 2022].
- [22] L. Rosencrance, "Military vs. Civilian GPS," Computer World, 3 April 2000. [Online]. Available: <https://www.computerworld.com/article/2594378/military-vs---civilian-gps.html#:~:text=Currently%2C%20GPS%20provides%20two%20levels,by%20the%20Department%20of%20Defense..> [Accessed 4 July 2022].
- [23] T. Gregorius, "GIS is not as simple as it used to be," DM Insights On Location, 31 October 2013. [Online]. Available: <https://www.directionsmag.com/article/1508>. [Accessed 19 July 2022].

- [24] T. V. Wilson, "How GPS Phones Work," How Stuff Works, 2022. [Online]. Available: <https://electronics.howstuffworks.com/gps-phone.htm>. [Accessed 18 July 2022].
- [25] Hexagon NovAtel, "GPS & GNSS Antennas," Hexagon NovAtel, 2022. [Online]. Available: <https://novatel.com/products/gps-gnss-antennas>. [Accessed 19 July 2022].
- [26] Hyper-Tech Systems, "GIS Antennas," Hyper-Tech Systems, 2022. [Online]. Available: <https://www.hypertech.co.il/product/gps-for-mapping-gis/gis-antennas/#:~:text=The%20Trimble%20GIS%20antennas%20are,and%20Trimble%20Pro%20series%20receivers..> [Accessed 19 July 2022].
- [27] Symmetricom, "General GPS Antenna Information," [Online]. Available: <https://www.microsemi.com/133676-gps-antenna-basics>. [Accessed 18 July 2022].
- [28] L. Yu, Y. Liu, T. Chi and L. Peng, "An iBeacon-based Indoor and Outdoor Positioning System for the Fire Emergency Command," in *Forum on Cooperative Positioning and Service*, Beijing, 2017.
- [29] T. Thingstad and H. T. Tran, A Location-based Mobile Service utilizing Anonymous Indoor User Data: An Empirical Study, Norway: Norwegian University of Science and Technology, 2014.
- [30] J. Shang, S. Yu and L. Zhu, Location-Aware Systems for Short-range Wireless Networks, Wuhan: Chind University of Geosciences and University of Science and Technology.
- [31] Samsung, "What is Bluetooth and how do I use it?," Samsung, 2022. [Online]. Available: <https://www.samsung.com/uk/support/mobile-devices/what-is-bluetooth/>. [Accessed 11 July 2022].
- [32] S. Amsler and S. Shea, "RFID (radio frequency identification)," Tech Target, March 2021. [Online]. Available: <https://www.techtarget.com/iotagenda/definition/RFID-radio-frequency-identification>. [Accessed 11 July 2022].
- [33] Identiv, "What Are RFID Tags and How Are They Used?," Identiv, 12 February 2021. [Online]. Available: <https://www.identiv.com/community/2021/02/12/what-are-rfid-tags-and-how-are-they-used/>. [Accessed 11 July 2022].
- [34] A. Ciurkot, "What is a Beacon? The Complete Guide to Bluetooth Beacons," Kontakt, 2022. [Online]. Available: <https://kontakt.io/what-is-a-beacon/#:~:text=Bluetooth%20provides%20the%20infrastructure%20for,beacon%20hardware%20can%20be%20simple..> [Accessed 15 March 2022].
- [35] E. Zimmerman, "How Can Beacon Technology Help Universities Boost Student Engagement?," EdTech, 10 December 2018. [Online]. Available: <https://edtechmagazine.com/higher/article/2018/12/how-can-beacon-technology-help-universities-boost-student-engagement-perfcon>. [Accessed 15 March 2022].
- [36] Android, "Android Studio features - Android Developers," Android, 2022. [Online]. Available: <https://developer.android.com/studio/features>. [Accessed 26 July 2022].

- [37] Tech Target, "What is Android Studio?," Tech Target, October 2018. [Online]. Available: <https://www.techtarget.com/searchmobilecomputing/definition/Android-Studio>. [Accessed 26 July 2022].
- [38] Android Developers, "Test in Android Studio," Android, 2022. [Online]. Available: <https://developer.android.com/studio/test/test-in-android-studio>. [Accessed 26 July 2022].
- [39] American Museum of Natural History, "Explorer App," American Museum of Natural History, 2022. [Online]. Available: <https://www.amnh.org/plan-your-visit/explorer>. [Accessed 17 July 2022].
- [40] Usability.Gov, "User Experience Basics," Usability.Gov, 9 September 2020. [Online]. Available: <https://www.usability.gov/what-and-why/user-experience.html>. [Accessed 4 August 2022].
- [41] J. Nielsen, "Usability 101: Introduction to Usability," Nielsen Norman Group, 3 January 2012. [Online]. Available: <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>. [Accessed 4 August 2022].
- [42] In Vision, "Mobile app design," In Vision, 2022. [Online]. Available: <https://www.invisionapp.com/defined/mobile-app-design>. [Accessed 6 August 2022].
- [43] Duckma, "Why Mobile App Design is Critical to Your App's Success," Duckma, 2022. [Online]. Available: <https://blog.duckma.com/en/sample-blog/mobile-app-design-critical/>. [Accessed 6 August 2022].
- [44] App Samurai, "App Samurai," 6 Necessary Elements For Designing A Perfect Mobile App User Interface, 3 August 2017. [Online]. Available: https://appsamurai.com/6-necessary-elements-for-designing-a-perfect-mobile-app-user-interface/#6_Necessary_Elements_For_Designing_A_Perfect_Mobile_App_User_Interface. [Accessed 6 August 2022].
- [45] "What is a QR code? A guide to the barcode's basics, why you're seeing it everywhere, and how to scan one," Insider, 3 May 2021. [Online]. Available: <https://www.businessinsider.com/what-is-a-qr-code?r=US&IR=T>. [Accessed 4 August 2022].
- [46] W. Cherisien, "17 Creative Ways to Use QR Codes," Mention Memo, 16 June 2022. [Online]. Available: <https://mention.com/en/blog/creative-ways-to-use-qr-codes/>. [Accessed 4 August 2022].
- [47] Kaspersky, "QR Code Security: What are QR codes and are they safe to use?," Kaspersky, 2022. [Online]. Available: <https://www.kaspersky.co.uk/resource-center/definitions/what-is-a-qr-code-how-to-scan>. [Accessed 4 August 2022].
- [48] W. Antonelli, "How to scan a QR code on your Android phone or tablet," Insider, 19 November 2021. [Online]. Available: <https://www.businessinsider.com/scan-qr-code-android?r=US&IR=T>. [Accessed 4 August 2022].
- [49] Expert Labels, "QR Codes – How They Work and What You Can Do with Them.," Expert Labels, 21 January 2022. [Online]. Available: <https://expertlabels.co.uk/news/what-you-can-do-with-qr->

codes/#:~:text=It%27s%20a%20form%20of%20D,Denso%20in%20Japan%20developed%20them.. [Accessed 4 August 2022].

- [50] A. Stein, "How Does QR Code Tracking Work?," QR Code Generator Powered by Bitly, 10 January 2020. [Online]. Available: <https://www.qr-code-generator.com/blog/how-does-qr-code-tracking-work/#:~:text=Dynamic%20QR%20Codes%20are%20trackable,system%20of%20the%20device%20used..> [Accessed 6 August 2022].
- [51] Business Research Methodology, "Positivism Research Philosophy," Business Research Methodology, 2022. [Online]. Available: <https://research-methodology.net/research-philosophy/positivism/>. [Accessed 10 March 2022].
- [52] C. Oden, "Research Methodology: Using Research Onion as a Framework," Project Topics, 2022. [Online]. Available: <https://www.projecttopics.org/research-methodology-using-research-onion-as-a-framework.html#:~:text=The%20onion%20research%20approach%20is%20deductive%2C%20making%20it,provides%20a%20better%20understanding%20of%20various%20social%20responsibilities..> [Accessed 13 June 2022].
- [53] 15Writers, "Understanding the Research Onion," 15Writers, 2022. [Online]. Available: <https://15writers.com/research-onion/#:~:text=The%20research%20onion%20suggests%20mono,of%20qualitative%20and%20quantitative%20methodology..> [Accessed 20 July 2022].
- [54] C. Rafel, "Understanding The Research Onion: An Illustration," Research Clue, 29 December 2016. [Online]. Available: <https://nairaproject.com/blog/understanding-the-research-onion-methodology.html>. [Accessed 25 July 2022].
- [55] Tourism Teacher, "Research Onion," Tourism Teacher, 31 August 2021. [Online]. Available: <https://tourismteacher.com/research-onion-beginners/#:~:text=Layers%20of%20the%20research%20onion&text=These%20layers%20are%20broken%20up,horizons%3B%20and%20techniques%20and%20procedures..> [Accessed 19 July 2022].
- [56] The Standish Group, "CHAOS Manifesto 2012," The Standish Group, 2012. [Online]. Available: <https://cs.calvin.edu/courses/cs/262/kvlinden/resources/CHAOSManifesto2012.pdf>. [Accessed 3 March 2022].
- [57] Atlassian Agile Coach, "What is Agile?," Atlassian Agile Coach, 2022. [Online]. Available: <https://www.atlassian.com/agile>. [Accessed 16 March 2022].
- [58] Digital.ai, "15th State of Agile Report," Digital.ai, 2022. [Online]. Available: <https://digital.ai/resource-center/analyst-reports/state-of-agile-report#ufh-i-615706098-14th-annual-state-of-agile-report/7027494>. [Accessed 17 March 2022].
- [59] Wrike, "The Benefits and Advantages of Agile," Wrike, 2022. [Online]. Available: <https://www.wrike.com/agile-guide/benefits-of-agile/>. [Accessed 17 March 2022].
- [60] C. Cobb, "What Are the Advantages and Disadvantages of Agile and Scrum?," High Impact Project Management, Inc, 2022. [Online]. Available:

<https://managedagile.com/what-are-the-advantages-and-disadvantages-of-agile-scrum/>. [Accessed 19 March 2022].

- [61] C. Drumond, "What is scrum?," Atlassian, 2022. [Online]. Available: <https://www.atlassian.com/agile/scrum>. [Accessed 10 August 2022].
- [62] Active Collab, "Sprint vs. Iteration: What's the Difference?," Active Collab, 23 March 2021. [Online]. Available: <https://activecollab.com/blog/project-management/sprint-vs-iteration-difference>. [Accessed 10 August 2022].
- [63] P. Kajzer and D. Winczewski, "The Three Pillars of Scrum and Why You Should Understand Them," STX Next, 2022. [Online]. Available: <https://www.stxnext.com/blog/three-pillars-scrum/>. [Accessed 10 August 2022].
- [64] Knowledge Hut upGrad, "Top 5 Scrum Values & Principles," Knowledge Hut upGrad, 2022. [Online]. Available: <https://www.knowledgehut.com/tutorials/scrum-tutorial/scrum-values>. [Accessed 10 August 2022].
- [65] M. Rehopf, "Scrum Sprints," Atlassian Agile Coach, 2022. [Online]. Available: <https://www.atlassian.com/agile/scrum/sprints>. [Accessed 10 August 2022].
- [66] A. Taouab, "Scrum Workflow - Key Stages of Product Development," LinkedIn, 10 December 2021. [Online]. Available: https://www.linkedin.com/pulse/scrum-workflow-key-stages-product-development-alae-taouab/?trk=public_profile_article_view. [Accessed 10 August 2022].
- [67] Synopsys Editorial Team, "Top 4 software development methodologies," Synopsys, 28 March 2017. [Online]. Available: Top 4 software development methodologies. [Accessed 28 February 2022].
- [68] Project Manager, "The Ultimate Guide ... Waterfall Model," Project Manager, 2022. [Online]. Available: <https://www.projectmanager.com/waterfall-methodology>. [Accessed 28 February 2022].
- [69] M. Dawson, D. Burrell, E. Rahim and S. Brewster, "Integrating Software Assurance into Software Development Life Cycle," *Journal of Information Systems Technology and Planning*, vol. 3, pp. 49 - 53, 2010.
- [70] S. Lewis, "Prototyping Model," Tech Target, September 2019. [Online]. Available: <https://www.techtarget.com/searchcio/definition/Prototyping-Model>. [Accessed 11 August 2022].
- [71] Rapid Direct, "Prototype Development: How To Make A Rapid Prototype," Rapid Direct, 11 November 2021. [Online]. Available: <https://www.rapiddirect.com/blog/prototype-development/>. [Accessed 11 August 2022].
- [72] M. Martin, "Prototyping Model in Software Engineering: Methodology, Process, Approach," Guru99, 9 July 2022. [Online]. Available: <https://www.guru99.com/software-engineering-prototyping-model.html>. [Accessed 11 August 2022].
- [73] Geeks for Geeks, "Difference between Waterfall model and Prototype model," Geeks for Geeks, 24 September 2020. [Online]. Available:

<https://www.geeksforgeeks.org/difference-between-waterfall-model-and-prototype-model/>. [Accessed 2 September 2022].

- [74] Yodiz Project Management, "Agile vs Prototyping," Yodiz Project Management, 24 November 2016. [Online]. Available: <https://www.yodiz.com/blog/agile-vs-prototyping/>. [Accessed 2 September 2022].
- [75] B. Laugwitz, T. Held and M. Schrepp, "User Experience Questionnaire," UEQ, 2022. [Online]. Available: <https://www.ueq-online.org/>. [Accessed 7 March 2022].
- [76] I. Sommerville, Software Engineering, 10th ed., Essex: Pearson Education, 2016.
- [77] Google Play, "Gower Heritage Centre," AT Creative Limited, 11 June 2018. [Online]. Available: https://play.google.com/store/apps/details?id=uk.co.audiotrails.gowerheritagecentre&hl=en_GB&gl=US. [Accessed 14 June 2022].
- [78] Google Play, "Glynn Vivian Multimedia Guide," Acoustiguide Limited, 23 August 2019. [Online]. Available: https://play.google.com/store/apps/details?id=com.acoustiguideamobile.am_glynn&hl=en_GB&gl=US. [Accessed 14 June 2022].
- [79] IBM, "Advantages of Java," IBM, 18 May 2022. [Online]. Available: <https://www.ibm.com/docs/en/aix/7.1?topic=monitoring-advantages-java>. [Accessed 1 September 2022].
- [80] Android Developers, "Android Studio features," Android Developers, [Online]. Available: <https://developer.android.com/studio/features#:~:text=Android%20Studio%20provides%20a%20robust,correctness%2C%20with%20a%20single%20click..> [Accessed 1 September 2022].
- [81] Google Git, "CameraDevice.java," Google Git, 2013. [Online]. Available: <https://android.googlesource.com/platform/frameworks/base/+/-/master/core/java/android/hardware/camera2/CameraDevice.java>. [Accessed 5 September 2022].
- [82] M. Jasper, Beginning Reflective Practice, Andover: Cengage Learning, 2013.

CHAPTER 10 – APPENDICES

Appendix A – Gantt Chart

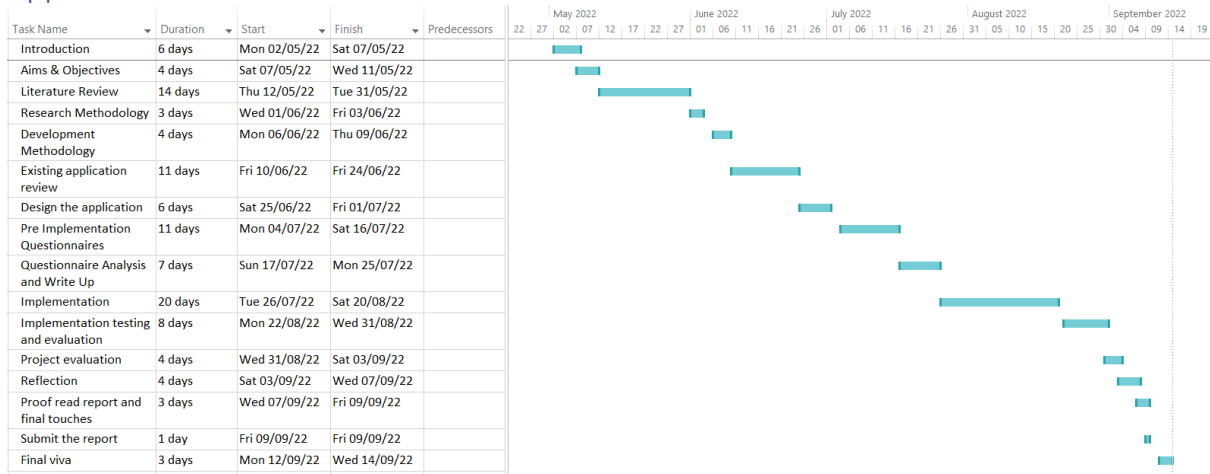


Figure 105: Original Gantt Chart

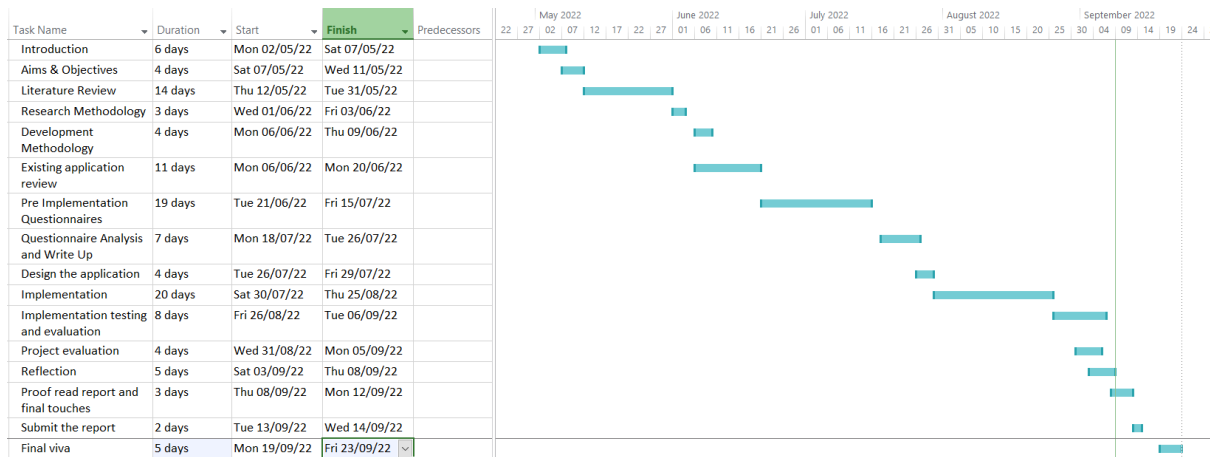


Figure 106: Updated Gantt Chart

Appendix B – Pre-Implementation Interview Questions

Did you attend a University open day prior to you starting?

Yes	No

If not, what are your reasons for not attending an open day? Did anything put you off attending, or could you find out information without needing to visit?

--

Please tick the area(s) you wanted to find more information about?

Area	Found Out More
Course Details	
Rooms I'd Be Learning In	
Student Services	
Student Finance	
Accommodation	
Campus and/or Facilities	

Is there anything that I've missed out on here which you wanted to find out about?

--

Were you provided a campus tour as part of the open day?

Yes	No

Once you started at the University, did you then have a campus tour?

Yes	No

If you did receive a campus tour, what did you see?

Facility	Did you See It?
The rooms you'd be studying in?	
The Hwb	
Student Services	
The Library	
Study Rooms in the Library	

Did you view anything else not mentioned in the previous question?

--

If you were given an option of having a self-guided tour app of the campus, is this something you would benefit from?

Yes	No

Please expand further

--

I would like to see

Area	Strongly agree	Agree	Neutral	Disagree	Strongly disagree	Prefer not to say
Canteen						
Vending machines						
Communal Areas						
Specific labs / rooms (e.g. games, networking, computing, engineering)						
Learning Support						
Open Access Areas / Generic Labs						
IT Support						
Toilets						
Showers						
Staff work room						
Library						
The Books						
Bookable library rooms						
Study areas						
Prayer Room						
Technician's Office						
Fire Exits						
Fire Alarms						
Reception						
Hwb						

It is hoped that, in future developments, that all UWTSD campuses, alongside with local places of interest, attractions and nightlife places, will be included on the

application. If this was to be done, which features you like to see. Again, you can agree / not agree to any of these options.

Area	Strongly agree	Agree	Neutral	Disagree	Strongly disagree	Prefer not to say
Campuses car parks						
Sports facilities						
Public Transport stations (bus & train stations)						
Local attractions						
Recruitment Companies						
Cafes, Pubs & Nightclubs						
Cafes at other campuses						

If this app had of been developed whilst you were applying for Universities, would it have helped with familiarisation of the campus prior to you commencing your studies?

Going by the description of the app, and the potential features it would include, is this an app you would

Answer	Yes	No
Use for campus tours?		
Use it purely to find rooms and not use it for self-guided tour purposes?		
Recommend to applicants and new students?		

Are there any other comments you'd like to make about the application, or make any comments which you'd like to make which haven't been covered in the other questions?

Appendix C – Prototype Interview Questions

Did these features work?

Feature	Yes	No
Home Page: Could you get to the QR code page?		
Could you 'scan' the QR code & take you to the student support page?		
After viewing the page, could you return to the home page?		
Did the Search by ID work and take you to the correct page?		
Did the Search by Name work and take you to the correct page?		
After viewing the page, could you return to the home page?		
On the search page, does the 'return to home page' button work?		

Are there any features, or anything specific with the app, that you particularly like?

Do you have any improvements you'd like to suggest?

Was the look and feel of the app acceptable for its purpose?

How would you rate the app prototype?

1 (poor)	2	3	4	5	6	7	8	9	10 (excellent)

If there are any other comments you would like to make, please say.

Appendix D – Java Application Interview Questions

Please enter the room IQ104 into the Search Box, when you search for this room, when you search for this room, which of these appears?

Testing	Yes	No
A description of the room which shows the room, capacity and date check		
Hardware description with how many are available and date check		
Software description with date check		
Session date, time(s) and description with session description, staff name and department		
Photo for the room		
QR code for the room		

Please enter the room FF002 into the search box, when you search for this room, which of these appears?

Testing	Yes	No
A description of the room which shows the room, capacity and date check		
Hardware description with how many are available and date check		
Software description with date check		
Session date, time(s) and description with session description, staff name and department		
Photo for the room		
QR code for the room		

Try entering in room IQ310, this room has not yet been added to the application so should throw back an error. Does it throw back an error or is anything displayed?

Yes	No

Try entering in room FF105, this room has not yet been added to the application so should throw back an error. Does it throw back an error or is anything displayed?

Yes	No

Are there any features, or anything specific with the app, that you particularly like?

Do you have any improvements you'd like to suggest?

Does the app assist you with having a tour of the campus?

Was the look and feel of the app acceptable for its purpose?

How would you rate the app prototype?

1	2	3	4	5	6	7	8	9	10

Appendix E – Android Studio Code

Barcode (QR Code)

All the code in the barcode (QR code) classes explain how the QR codes would work with the app.

Barcode Capture Activity Java Class

```
package com.example.ewan_uwtsd_app_project.barcode;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.os.Build;
import android.os.Bundle;
//import android.support.annotation.NonNull;
//import android.support.v4.app.ActivityCompat;
//import android.support.v7.app.AppCompatActivity;
import android.util.DisplayMetrics;
import android.util.Log;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;
import com.google.android.gms.common.api.CommonStatusCodes;
import com.google.android.gms.vision.MultiProcessor;
import com.google.android.gms.vision.barcode.Barcode;
import com.google.android.gms.vision.barcode.BarcodeDetector;

//Imports CameraSource and CameraSourcePreview classes as libraries.
import com.example.ewan_uwtsd_app_project.R;
import com.example.ewan_uwtsd_app_project.camera.CameraSource;
import com.example.ewan_uwtsd_app_project.camera.CameraSourcePreview;

import java.io.IOException;

public class BarcodeCaptureActivity extends AppCompatActivity
    implements BarcodeTracker.BarcodeGraphicTrackerCallback {
```

```

// Log tag for barcode reader.
private static final String TAG = "Barcode-reader";

// Intent request code to handle updating play services if needed.
private static final int RC_HANDLE_GMS = 9001;

// Permission request codes need to be < 256
private static final int RC_HANDLE_CAMERA_PERM = 2;

// Constants used to pass extra data in the intent
public static final String BarcodeObject = "Barcode";

// Declares new instances of CameraSource and CameraSourcePreview.
// Used to capture and preview camera for scanning QR codes.
private CameraSource mCameraSource;
private CameraSourcePreview mPreview;

/**
 * Initializes the UI and creates the detector pipeline.
 */
@Override
// Sets onCreate event for BarcodeCaptureActivity.
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // Sets content view as barcode_capture layout.
    setContentView(R.layout.barcode_capture);

    // Sets ID for CameraSourcePreview.
    mPreview = (CameraSourcePreview) findViewById(R.id.preview);

    // Declares and sets boolean for auto focus and flash functions for Camera.
    boolean autoFocus = true;
    boolean useFlash = false;

    // Check for the camera permission before accessing the camera. If the
    // permission is not granted yet, request permission.
    int rc = ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA);
    // If permission is granted, then..
    if (rc == PackageManager.PERMISSION_GRANTED) {
        //Creates new camera sources. Passes booleans for settings.
        createCameraSource(autoFocus, useFlash);
        //Else (If permission is denied)
    } else {
        //Calls method to request camera permission.
        requestCameraPermission();
    }
}

```

```

}

// Overrides abstract method when QR code is detected.
@Override
public void onDetectedQrCode(Barcode barcode) {
    // If scanned barcode is not null (not empty), then..
    if (barcode != null) {
        // Declares new intent.
        Intent intent = new Intent();
        // Stores barcode object and scanned barcode as Extra.
        intent.putExtra(BarcodeObject, barcode);
        // Sets result using success status code and intent.
        setResult(CommonStatusCodes.SUCCESS, intent);
        // Finishes activity.
        finish();
    }
}

// Handles the requesting of the camera permission.
private void requestCameraPermission() {
    // Prints following to log. Requesting permission to access camera.
    Log.w(TAG, "Camera permission is not granted. Requesting permission");

    // Declares final string array for permissions. Stores answer.
    final String[] permissions = new String[]{Manifest.permission.CAMERA};

    // Requests permission based on the following.
    if (!ActivityCompat.shouldShowRequestPermissionRationale(this,
        Manifest.permission.CAMERA)) {
        ActivityCompat.requestPermissions(this, permissions,
RC_HANDLE_CAMERA_PERM);
    }
}

/**
 * Creates and starts the camera.
 *
 * Suppressing InlinedApi since there is a check that the minimum version is met
before using
 * the constant.
 */
@SuppressWarnings("InlinedApi")
// Declares method to create CameraSource. Used to preview and capture QR
codes.
// Parameters: boolean values for autoFocus and useFlash settings.
private void createCameraSource(boolean autoFocus, boolean useFlash) {
    // Declares and assigns context for application.
    Context context = getApplicationContext();

```

```

        // A barcode detector is created to track barcodes. An associated multi-
processor instance
        // is set to receive the barcode detection results, track the barcodes, and
maintain
        // graphics for each barcode on screen. The factory is used by the multi-
processor to
        // create a separate tracker instance for each barcode.
        BarcodeDetector barcodeDetector = new BarcodeDetector.Builder(context)
            .setBarcodeFormats(Barcode.ALL_FORMATS)
            .build();
        // Declares new instance of BarcodeTrackerFactory. Used to create new
instances of BarcodeTracker.
        BarcodeTrackerFactory barcodeFactory = new BarcodeTrackerFactory(this);
        barcodeDetector.setProcessor(new
MultiProcessor.Builder<>(barcodeFactory).build());

        if (!barcodeDetector.isOperational()) {
            // Note: The first time that an app using the barcode or face API is installed
on a
            // device, GMS will download a native libraries to the device in order to do
detection.
            // Usually this completes before the app is run for the first time. But if that
            // download has not yet completed, then the above call will not detect any
barcodes
            // and/or faces.
            //
            // isOperational() can be used to check if the required native libraries are
currently
            // available. The detectors will automatically become operational once the
library
            // downloads complete on device.
            Log.w(TAG, "Detector dependencies are not yet available.");

            // Check for low storage. If there is low storage, the native library will not
be
            // downloaded, so detection will not become operational.
            IntentFilter lowstorageFilter = new
IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
            boolean hasLowStorage = registerReceiver(null, lowstorageFilter) != null;

            if (hasLowStorage) {
                Toast.makeText(this, R.string.low_storage_error,
                    Toast.LENGTH_LONG).show();
                Log.w(TAG, getString(R.string.low_storage_error));
            }
        }
    }
}

```

```

        // Creates and starts the camera. Note that this uses a higher resolution in
        comparison
        // to other detection examples to enable the barcode detector to detect small
        barcodes
        // at long distances.
        DisplayMetrics metrics = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics(metrics);

        // Builds new CameraSource.
        CameraSource.Builder builder = new
        CameraSource.Builder(getApplicationContext(), barcodeDetector)
            .setFacing(CameraSource.CAMERA_FACING_BACK)
            .setRequestedPreviewSize(metrics.widthPixels, metrics.heightPixels)
            .setRequestedFps(24.0f);

        // make sure that auto focus is an available option
        if (Build.VERSION.SDK_INT >=
        Build.VERSION_CODES.ICE_CREAM_SANDWICH) {
            builder = builder.setFocusMode(
                autoFocus ?
        Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE : null);
        }

        mCameraSource = builder
            .setFlashMode(useFlash ? Camera.Parameters.FLASH_MODE_TORCH :
        null)
            .build();
    }

    // Restarts the camera
    @Override
    protected void onResume() {
        super.onResume();
        startCameraSource();
    }

    // Stops the camera
    @Override
    protected void onPause() {
        super.onPause();
        if (mPreview != null) {
            mPreview.stop();
        }
    }

    /**
     * Releases the resources associated with the camera source, the associated
     * detectors, and the

```

```

    * rest of the processing pipeline.
    */
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (mPreview != null) {
            mPreview.release();
        }
    }

    /**
     * Callback for the result from requesting permissions. This method
     * is invoked for every call on {@link #requestPermissions(String[], int)}.
     * <p>
     * <strong>Note:</strong> It is possible that the permissions request interaction
     * with the user is interrupted. In this case you will receive empty permissions
     * and results arrays which should be treated as a cancellation.
     * </p>
     *
     * @param requestCode The request code passed in {@link
     #requestPermissions(String[], int)}.
     * @param permissions The requested permissions. Never null.
     * @param grantResults The grant results for the corresponding permissions
     * which is either {@link
     PackageManager#PERMISSION_GRANTED}
     * or {@link PackageManager#PERMISSION_DENIED}. Never null.
     * @see #requestPermissions(String[], int)
     */
    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           @NonNull String[] permissions,
                                           @NonNull int[] grantResults) {
        if (requestCode != RC_HANDLE_CAMERA_PERM) {
            Log.d(TAG, "Got unexpected permission result: " + requestCode);
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
            return;
        }

        if (grantResults.length != 0 && grantResults[0] ==
            PackageManager.PERMISSION_GRANTED) {
            Log.d(TAG, "Camera permission granted - initialize the camera source");
            // we have permission, so create the camera source
            boolean autoFocus = true;
            boolean useFlash = false;
            createCameraSource(autoFocus, useFlash);
            return;
        }
    }

```

```

        Log.e(TAG, "Permission not granted: results len = " + grantResults.length +
            " Result code = " + (grantResults.length > 0 ? grantResults[0] :
            "(empty)"));

        DialogInterface.OnClickListener listener = new
        DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                finish();
            }
        };

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Multitracker sample")
            .setMessage(R.string.no_camera_permission)
            .setPositiveButton(R.string.ok, listener)
            .show();
    }

    /**
     * Starts or restarts the camera source, if it exists. If the camera source doesn't
     * exist yet
     * (e.g., because onResume was called before the camera source was created),
     * this will be called
     * again when the camera source is created.
     */
    private void startCameraSource() throws SecurityException {
        // check that the device has play services available.
        int code = GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(
            getApplicationContext());
        if (code != ConnectionResult.SUCCESS) {
            Dialog dlg =
                GoogleApiAvailability.getInstance().getErrorDialog(this, code,
RC_HANDLE_GMS);
            dlg.show();
        }

        if (mCameraSource != null) {
            try {
                mPreview.start(mCameraSource);
            } catch (IOException e) {
                Log.e(TAG, "Unable to start camera source.", e);
                mCameraSource.release();
                mCameraSource = null;
            }
        }
    }
}

```

Barcode Tracker Java Class

```
package com.example.ewan_uwtsd_app_project.barcode;

import android.content.Context;

import com.google.android.gms.vision.Tracker;
import com.google.android.gms.vision.barcode.Barcode;

public class BarcodeTracker extends Tracker<Barcode> {
    // Declare private barcode graphic tracker callback.
    private BarcodeGraphicTrackerCallback mListener;

    // Declare interface for BarcodeGraphicTrackerCallback.
    public interface BarcodeGraphicTrackerCallback {
        // Declare abstract method when QR code is detected.
        void onDetectedQrCode(Barcode barcode);
    }

    //Default constructor/
    BarcodeTracker(Context listener) {
        mListener = (BarcodeGraphicTrackerCallback) listener;
    }

    /* Overrides method when new item is scanned.
     * Parameters include id and barcode item. Not sure if ID is used. */
    @Override
    public void onNewItem(int id, Barcode item) {
        // If item is not empty, then..
        if (item.displayValue != null) {
            // Calls abstract method for when QR code is detected.
            mListener.onDetectedQrCode(item);
        }
    }
}
```

Barcode Tracker Factory Java Class

```
package com.example.ewan_uwtsd_app_project.barcode;

import android.content.Context;

import com.google.android.gms.vision.MultiProcessor;
import com.google.android.gms.vision.Tracker;
import com.google.android.gms.vision.barcode.Barcode;

public class BarcodeTrackerFactory implements MultiProcessor.Factory<Barcode> {
    private Context mContext;

    BarcodeTrackerFactory(Context context) {
```

```

        mContext = context;
    }

    //Creates new instance of BarcodeTracker.
    @Override

    public Tracker<Barcode> create(Barcode barcode) {
        //Returns new instance of BarcodeTracker.
        return new BarcodeTracker(mContext);
    }
}

```

Camera

All the code in the camera classes show the functions implemented to make the camera work on the app.

Camera Source Java Class

```

package com.example.ewan_uwtsd_app_project.camera;

import android.Manifest;
import android.annotation.SuppressLint;
import android.annotation.TargetApi;
import android.content.Context;
import android.graphics.ImageFormat;
import android.graphics.SurfaceTexture;
import android.hardware.Camera;
import android.os.Build;
import android.os.SystemClock;
// import android.support.annotation.Nullable;
// import android.support.annotation.RequiresPermission;
// import android.support.annotation.StringDef;
import android.util.Log;
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.WindowManager;

import androidx.annotation.Nullable;
import androidx.annotation.RequiresPermission;
import androidx.annotation.StringDef;

import com.google.android.gms.common.images.Size;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.Frame;

import java.io.IOException;
import java.lang.annotation.Retention;

```

```

import java.lang.annotation.RetentionPolicy;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class CameraSource {
    @SuppressWarnings("InlinedApi")
    public static final int CAMERA_FACING_BACK =
Camera.CameraInfo.CAMERA_FACING_BACK;
    @SuppressWarnings("InlinedApi")
    public static final int CAMERA_FACING_FRONT =
Camera.CameraInfo.CAMERA_FACING_FRONT;

    private static final String TAG = "OpenCameraSource";

    /**
     * The dummy surface texture must be assigned a chosen name. Since we never
     use an OpenGL
     * context, we can choose any ID we want here.
     */
    private static final int DUMMY_TEXTURE_NAME = 100;

    /**
     * If the absolute difference between a preview size aspect ratio and a picture
     size aspect
     * ratio is less than this tolerance, they are considered to be the same aspect
     ratio.
     */
    private static final float ASPECT_RATIO_TOLERANCE = 0.01f;

    @StringDef({
        Camera.Parameters.FOCUS_MODE_CONTINUOUS_PICTURE,
        Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO,
        Camera.Parameters.FOCUS_MODE_AUTO,
        Camera.Parameters.FOCUS_MODE_EDOF,
        Camera.Parameters.FOCUS_MODE_FIXED,
        Camera.Parameters.FOCUS_MODE_INFINITY,
        Camera.Parameters.FOCUS_MODE_MACRO
    })
    @Retention(RetentionPolicy.SOURCE)
    private @interface FocusMode {}

    @StringDef({
        Camera.Parameters.FLASH_MODE_ON,
        Camera.Parameters.FLASH_MODE_OFF,
        Camera.Parameters.FLASH_MODE_AUTO,

```

```

        Camera.Parameters.FLASH_MODE_RED_EYE,
        Camera.Parameters.FLASH_MODE_TORCH
    })
    @Retention(RetentionPolicy.SOURCE)
    private @interface FlashMode {}

    private Context mContext;

    private final Object mCameraLock = new Object();

    // Guarded by mCameraLock
    private Camera mCamera;

    private int mFacing = CAMERA_FACING_BACK;

    /**
     * Rotation of the device, and thus the associated preview images captured from
     the device.
     * See {@link Frame.Metadata#getRotation()}.
     */
    private int mRotation;

    private Size mPreviewSize;

    // These values may be requested by the caller. Due to hardware limitations, we
    may need to
    // select close, but not exactly the same values for these.
    private float mRequestedFps = 30.0f;
    private int mRequestedPreviewWidth = 1024;
    private int mRequestedPreviewHeight = 768;

    private String mFocusMode = null;
    private String mFlashMode = null;

    // These instances need to be held onto to avoid GC of their underlying resources.
    Even though
    // these aren't used outside of the method that creates them, they still must have
    hard
    // references maintained to them.
    private SurfaceView mDummySurfaceView;
    private SurfaceTexture mDummySurfaceTexture;

    /**
     * Dedicated thread and associated runnable for calling into the detector with
     frames, as the
     * frames become available from the camera.
     */

```

```

private Thread mProcessingThread;
private FrameProcessingRunnable mFrameProcessor;

/**
 * Map to convert between a byte array, received from the camera, and its
 * associated byte
 * buffer. We use byte buffers internally because this is a more efficient way to
 * call into
 * native code later (avoids a potential copy).
 */
private Map<byte[], ByteBuffer> mBytesToByteBuffer = new HashMap<>();

//=====
// Builder

//=====

/**
 * Builder for configuring and creating an associated camera source.
 */
public static class Builder {
    private final Detector<?> mDetector;
    private CameraSource mCameraSource = new CameraSource();

    /**
     * Creates a camera source builder with the supplied context and detector.
     Camera preview
     * images will be streamed to the associated detector upon starting the camera
     source.
     */
    public Builder(Context context, Detector<?> detector) {
        if (context == null) {
            throw new IllegalArgumentException("No context supplied.");
        }
        if (detector == null) {
            throw new IllegalArgumentException("No detector supplied.");
        }

        mDetector = detector;
        mCameraSource.mContext = context;
    }

    /**
     * Sets the requested frame rate in frames per second. If the exact requested
     value is not

```

```

    * not available, the best matching available value is selected. Default: 30.
    */
    public Builder setRequestedFps(float fps) {
        if (fps <= 0) {
            throw new IllegalArgumentException("Invalid fps: " + fps);
        }
        mCameraSource.mRequestedFps = fps;
        return this;
    }

    public Builder setFocusMode(@FocusMode String mode) {
        mCameraSource.mFocusMode = mode;
        return this;
    }

    public Builder setFlashMode(@FlashMode String mode) {
        mCameraSource.mFlashMode = mode;
        return this;
    }

    /**
     * Sets the desired width and height of the camera frames in pixels. If the
     exact desired
     * values are not available options, the best matching available options are
     selected.
     * Also, we try to select a preview size which corresponds to the aspect ratio of
     an
     * associated full picture size, if applicable. Default: 1024x768.
     */
    public Builder setRequestedPreviewSize(int width, int height) {
        // Restrict the requested range to something within the realm of possibility.
        The
        // choice of 1000000 is a bit arbitrary -- intended to be well beyond
        resolutions that
        // devices can support. We bound this to avoid int overflow in the code
        later.
        final int MAX = 1000000;
        if ((width <= 0) || (width > MAX) || (height <= 0) || (height > MAX)) {
            throw new IllegalArgumentException("Invalid preview size: " + width +
"x" + height);
        }
        mCameraSource.mRequestedPreviewWidth = width;
        mCameraSource.mRequestedPreviewHeight = height;
        return this;
    }

    /**
     * Sets the camera to use (either {@link #CAMERA_FACING_BACK} or

```

```

        * {@link #CAMERA_FACING_FRONT}). Default: back facing.
    */
    public Builder setFacing(int facing) {
        if ((facing != CAMERA_FACING_BACK) && (facing !=
CAMERA_FACING_FRONT)) {
            throw new IllegalArgumentException("Invalid camera: " + facing);
        }
        mCameraSource.mFacing = facing;
        return this;
    }

    /**
     * Creates an instance of the camera source.
     */
    public CameraSource build() {
        mCameraSource.mFrameProcessor = mCameraSource.new
FrameProcessingRunnable(mDetector);
        return mCameraSource;
    }
}

//=====
// Bridge Functionality for the Camera1 API
//=====

/**
 * Callback interface used to signal the moment of actual image capture.
 */
public interface ShutterCallback {
    /**
     * Called as near as possible to the moment when a photo is captured from the
sensor. This
     * is a good opportunity to play a shutter sound or give other feedback of
camera operation.
     * This may be some time after the photo was triggered, but some time before
the actual data
     * is available.
     */
    void onShutter();
}

/**
 * Callback interface used to supply image data from a photo capture.
 */

```

```

public interface PictureCallback {
    /**
     * Called when image data is available after a picture is taken. The format of
the data
     * is a jpeg binary.
     */
    void onPictureTaken(byte[] data);
}

/**
 * Callback interface used to notify on completion of camera auto focus.
 */
public interface AutoFocusCallback {
    /**
     * Called when the camera auto focus completes. If the camera
     * does not support auto-focus and autoFocus is called,
     * onAutoFocus will be called immediately with a fake value of
     * <code>success</code> set to <code>true</code>.
     * <p/>
     * The auto-focus routine does not lock auto-exposure and auto-white
     * balance after it completes.
     *
     * @param success true if focus was successful, false if otherwise
     */
    void onAutoFocus(boolean success);
}

/**
 * Callback interface used to notify on auto focus start and stop.
 * <p/>
 * <p>This is only supported in continuous autofocus modes -- {@link
 * Camera.Parameters#FOCUS_MODE_CONTINUOUS_VIDEO} and {@link
 * Camera.Parameters#FOCUS_MODE_CONTINUOUS_PICTURE}. Applications can
show
 * autofocus animation based on this.</p>
 */
public interface AutoFocusMoveCallback {
    /**
     * Called when the camera auto focus starts or stops.
     *
     * @param start true if focus starts to move, false if focus stops to move
     */
    void onAutoFocusMoving(boolean start);
}

//=====
=====

```

```

// Public

//=====

/**
 * Stops the camera and releases the resources of the camera and underlying
 * detector.
 */
public void release() {
    synchronized (mCameraLock) {
        stop();
        mFrameProcessor.release();
    }
}

/**
 * Opens the camera and starts sending preview frames to the underlying
 * detector. The preview
 * frames are not displayed.
 *
 * @throws IOException if the camera's preview texture or display could not be
 * initialized
 */
@RequiresPermission(Manifest.permission.CAMERA)
public CameraSource start() throws IOException {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            return this;
        }

        mCamera = createCamera();

        // SurfaceTexture was introduced in Honeycomb (11), so if we are running
and
        // old version of Android. fall back to use SurfaceView.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
            mDummySurfaceTexture = new
SurfaceTexture(DUMMY_TEXTURE_NAME);
            mCamera.setPreviewTexture(mDummySurfaceTexture);
        } else {
            mDummySurfaceView = new SurfaceView(mContext);
            mCamera.setPreviewDisplay(mDummySurfaceView.getHolder());
        }
        mCamera.startPreview();

        mProcessingThread = new Thread(mFrameProcessor);
        mFrameProcessor.setActive(true);
    }
}

```

```

        mProcessingThread.start();
    }
    return this;
}

/**
 * Opens the camera and starts sending preview frames to the underlying
 * detector. The supplied
 * surface holder is used for the preview so frames can be displayed to the user.
 *
 * @param surfaceHolder the surface holder to use for the preview frames
 * @throws IOException if the supplied surface holder could not be used as the
 * preview display
 */
@RequiresPermission(Manifest.permission.CAMERA)
public CameraSource start(SurfaceHolder surfaceHolder) throws IOException {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            return this;
        }

        mCamera = createCamera();
        mCamera.setPreviewDisplay(surfaceHolder);
        mCamera.startPreview();

        mProcessingThread = new Thread(mFrameProcessor);
        mFrameProcessor.setActive(true);
        mProcessingThread.start();
    }
    return this;
}

/**
 * Closes the camera and stops sending frames to the underlying frame detector.
 * <p/>
 * This camera source may be restarted again by calling {@link #start()} or
 * {@link #start(SurfaceHolder)}.
 * <p/>
 * Call {@link #release()} instead to completely shut down this camera source
 * and release the
 * resources of the underlying detector.
 */
public void stop() {
    synchronized (mCameraLock) {
        mFrameProcessor.setActive(false);
        if (mProcessingThread != null) {
            try {
                // Wait for the thread to complete to ensure that we can't have

```

```

multiple threads
    // executing at the same time (i.e., which would happen if we called
start too
    // quickly after stop).
    mProcessingThread.join();
} catch (InterruptedException e) {
    Log.d(TAG, "Frame processing thread interrupted on release.");
}
mProcessingThread = null;
}

// clear the buffer to prevent oom exceptions
mBytesToByteBuffer.clear();

if (mCamera != null) {
    mCamera.stopPreview();
    mCamera.setPreviewCallbackWithBuffer(null);
    try {
        // We want to be compatible back to Gingerbread, but SurfaceTexture
        // wasn't introduced until Honeycomb. Since the interface cannot use
a SurfaceTexture, if the
        // developer wants to display a preview we must use a SurfaceHolder.
If the developer doesn't
        // want to display a preview we use a SurfaceTexture if we are
running at least Honeycomb.

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB)
{
            mCamera.setPreviewTexture(null);

        } else {
            mCamera.setPreviewDisplay(null);
        }
    } catch (Exception e) {
        Log.d(TAG, "Failed to clear camera preview: " + e);
    }
    mCamera.release();
    mCamera = null;
}
}

/**
 * Returns the preview size that is currently in use by the underlying camera.
 */
public Size getPreviewSize() {
    return mPreviewSize;
}

```

```

/**
 * Returns the selected camera; one of {@link #CAMERA_FACING_BACK} or
 * {@link #CAMERA_FACING_FRONT}.
 */
public int getCameraFacing() {
    return mFacing;
}

```

```

public int doZoom(float scale) {
    synchronized (mCameraLock) {
        if (mCamera == null) {
            return 0;
        }
        int currentZoom = 0;
        int maxZoom;
        Camera.Parameters parameters = mCamera.getParameters();
        if (!parameters.isZoomSupported()) {
            Log.w(TAG, "Zoom is not supported on this device");
            return currentZoom;
        }
        maxZoom = parameters.getMaxZoom();

        currentZoom = parameters.getZoom() + 1;
        float newZoom;
        if (scale > 1) {
            newZoom = currentZoom + scale * (maxZoom / 10);
        } else {
            newZoom = currentZoom * scale;
        }
        currentZoom = Math.round(newZoom) - 1;
        if (currentZoom < 0) {
            currentZoom = 0;
        } else if (currentZoom > maxZoom) {
            currentZoom = maxZoom;
        }
        parameters.setZoom(currentZoom);
        mCamera.setParameters(parameters);
        return currentZoom;
    }
}

```

```

/**
 * Initiates taking a picture, which happens asynchronously. The camera source
 * should have been
 * activated previously with {@link #start()} or {@link #start(SurfaceHolder)}.
 * The camera
 * preview is suspended while the picture is being taken, but will resume once

```

picture taking is

```
* done.
*
* @param shutter the callback for image capture moment, or null
* @param jpeg the callback for JPEG image data, or null
*/
public void takePicture(ShutterCallback shutter, PictureCallback jpeg) {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            PictureStartCallback startCallback = new PictureStartCallback();
            startCallback.mDelegate = shutter;
            PictureDoneCallback doneCallback = new PictureDoneCallback();
            doneCallback.mDelegate = jpeg;
            mCamera.takePicture(startCallback, null, null, doneCallback);
        }
    }
}

/**
 * Gets the current focus mode setting.
 *
 * @return current focus mode. This value is null if the camera is not yet
 * created. Applications should call {@link
 * #autoFocus(AutoFocusCallback)} to start the focus if focus
 * mode is FOCUS_MODE_AUTO or FOCUS_MODE_MACRO.
 * @see Camera.Parameters#FOCUS_MODE_AUTO
 * @see Camera.Parameters#FOCUS_MODE_INFINITY
 * @see Camera.Parameters#FOCUS_MODE_MACRO
 * @see Camera.Parameters#FOCUS_MODE_FIXED
 * @see Camera.Parameters#FOCUS_MODE_EDOF
 * @see Camera.Parameters#FOCUS_MODE_CONTINUOUS_VIDEO
 * @see Camera.Parameters#FOCUS_MODE_CONTINUOUS_PICTURE
 */
@Nullable
@FocusMode
public String getFocusMode() {
    return mFocusMode;
}

/**
 * Sets the focus mode.
 *
 * @param mode the focus mode
 * @return {@code true} if the focus mode is set, {@code false} otherwise
 * @see #getFocusMode()
 */
public boolean setFocusMode(@FocusMode String mode) {
    synchronized (mCameraLock) {
```

```

        if (mCamera != null && mode != null) {
            Camera.Parameters parameters = mCamera.getParameters();
            if (parameters.getSupportedFocusModes().contains(mode)) {
                parameters.setFocusMode(mode);
                mCamera.setParameters(parameters);
                mFocusMode = mode;
                return true;
            }
        }

        return false;
    }
}

/**
 * Gets the current flash mode setting.
 *
 * @return current flash mode. null if flash mode setting is not
 * supported or the camera is not yet created.
 * @see Camera.Parameters#FLASH_MODE_OFF
 * @see Camera.Parameters#FLASH_MODE_AUTO
 * @see Camera.Parameters#FLASH_MODE_ON
 * @see Camera.Parameters#FLASH_MODE_RED_EYE
 * @see Camera.Parameters#FLASH_MODE_TORCH
 */
@Nullable
@FlashMode
public String getFlashMode() {
    return mFlashMode;
}

/**
 * Sets the flash mode.
 *
 * @param mode flash mode.
 * @return {@code true} if the flash mode is set, {@code false} otherwise
 * @see #getFlashMode()
 */
public boolean setFlashMode(@FlashMode String mode) {
    synchronized (mCameraLock) {
        if (mCamera != null && mode != null) {
            Camera.Parameters parameters = mCamera.getParameters();
            if (parameters.getSupportedFlashModes().contains(mode)) {
                parameters.setFlashMode(mode);
                mCamera.setParameters(parameters);
                mFlashMode = mode;
                return true;
            }
        }
    }
}

```

```

    }

    return false;
}

}

/**
 * Starts camera auto-focus and registers a callback function to run when
 * the camera is focused. This method is only valid when preview is active
 * (between {@link #start()} or {@link #start(SurfaceHolder)} and before
 * {@link #stop()} or {@link #release()}).
 * <p/>
 * <p>Callers should check
 * {@link #getFocusMode()} to determine if
 * this method should be called. If the camera does not support auto-focus,
 * it is a no-op and {@link AutoFocusCallback#onAutoFocus(boolean)}
 * callback will be called immediately.
 * <p/>
 * <p>If the current flash mode is not
 * {@link Camera.Parameters#FLASH_MODE_OFF}, flash may be
 * fired during auto-focus, depending on the driver and camera hardware.<p>
 *
 * @param cb the callback to run
 * @see #cancelAutoFocus()
 */
public void autoFocus(@Nullable AutoFocusCallback cb) {
    synchronized (mCameraLock) {
        if (mCamera != null) {
            CameraAutoFocusCallback autoFocusCallback = null;
            if (cb != null) {
                autoFocusCallback = new CameraAutoFocusCallback();
                autoFocusCallback.mDelegate = cb;
            }
            mCamera.autoFocus(autoFocusCallback);
        }
    }
}

/**
 * Cancels any auto-focus function in progress.
 * Whether or not auto-focus is currently in progress,
 * this function will return the focus position to the default.
 * If the camera does not support auto-focus, this is a no-op.
 *
 * @see #autoFocus(AutoFocusCallback)
 */
public void cancelAutoFocus() {
    synchronized (mCameraLock) {

```

```

        if (mCamera != null) {
            mCamera.cancelAutoFocus();
        }
    }
}

/**
 * Sets camera auto-focus move callback.
 *
 * @param cb the callback to run
 * @return @code true if the operation is supported (i.e. from Jelly Bean),
 * @code false otherwise
 */
@TargetApi(Build.VERSION_CODES.JELLY_BEAM)
public boolean setAutoFocusMoveCallback(@Nullable AutoFocusMoveCallback cb)
{
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.JELLY_BEAM) {
        return false;
    }

    synchronized (mCameraLock) {
        if (mCamera != null) {
            CameraAutoFocusMoveCallback autoFocusMoveCallback = null;
            if (cb != null) {
                autoFocusMoveCallback = new CameraAutoFocusMoveCallback();
                autoFocusMoveCallback.mDelegate = cb;
            }
            mCamera.setAutoFocusMoveCallback(autoFocusMoveCallback);
        }
    }

    return true;
}

//=====
//=====
// Private

//=====
//=====

/**
 * Only allow creation via the builder class.
 */
private CameraSource() {
}

```

```

/**
 * Wraps the camera1 shutter callback so that the deprecated API isn't exposed.
 */
private class PictureStartCallback implements Camera.ShutterCallback {
    private ShutterCallback mDelegate;

    @Override
    public void onShutter() {
        if (mDelegate != null) {
            mDelegate.onShutter();
        }
    }
}

/**
 * Wraps the final callback in the camera sequence, so that we can automatically
turn the camera
 * preview back on after the picture has been taken.
 */
private class PictureDoneCallback implements Camera.PictureCallback {
    private PictureCallback mDelegate;

    @Override
    public void onPictureTaken(byte[] data, Camera camera) {
        if (mDelegate != null) {
            mDelegate.onPictureTaken(data);
        }
        synchronized (mCameraLock) {
            if (mCamera != null) {
                mCamera.startPreview();
            }
        }
    }
}

/**
 * Wraps the camera1 auto focus callback so that the deprecated API isn't
exposed.
 */
private class CameraAutoFocusCallback implements Camera.AutoFocusCallback {
    private AutoFocusCallback mDelegate;

    @Override
    public void onAutoFocus(boolean success, Camera camera) {
        if (mDelegate != null) {
            mDelegate.onAutoFocus(success);
        }
    }
}

```

```

    }

    /**
     * Wraps the camera1 auto focus move callback so that the deprecated API isn't
     exposed.
     */
    @TargetApi(Build.VERSION_CODES.JELLY_BEAM)
    private class CameraAutoFocusMoveCallback implements
    Camera.AutoFocusMoveCallback {
        private AutoFocusMoveCallback mDelegate;

        @Override
        public void onAutoFocusMoving(boolean start, Camera camera) {
            if (mDelegate != null) {
                mDelegate.onAutoFocusMoving(start);
            }
        }
    }
}

/**
 * Opens the camera and applies the user settings.
 *
 * * @throws RuntimeException if the method fails
 */
@SuppressLint("InlinedApi")
private Camera createCamera() {
    int requestedCameraId = getIdForRequestedCamera(mFacing);
    if (requestedCameraId == -1) {
        throw new RuntimeException("Could not find requested camera.");
    }
    Camera camera = Camera.open(requestedCameraId);

    SizePair sizePair = selectSizePair(camera, mRequestedPreviewWidth,
mRequestedPreviewHeight);
    if (sizePair == null) {
        throw new RuntimeException("Could not find suitable preview size.");
    }
    Size pictureSize = sizePair.pictureSize();
    mPreviewSize = sizePair.previewSize();

    int[] previewFpsRange = selectPreviewFpsRange(camera, mRequestedFps);
    if (previewFpsRange == null) {
        throw new RuntimeException("Could not find suitable preview frames per
second range.");
    }

    Camera.Parameters parameters = camera.getParameters();

```

```

        if (pictureSize != null) {
            parameters.setPictureSize(pictureSize.getWidth(), pictureSize.getHeight());
        }

        parameters.setPreviewSize(mPreviewSize.getWidth(),
mPreviewSize.getHeight());
        parameters.setPreviewFpsRange(
            previewFpsRange[Camera.Parameters.PREVIEW_FPS_MIN_INDEX],
            previewFpsRange[Camera.Parameters.PREVIEW_FPS_MAX_INDEX]);
        parameters.setPreviewFormat(ImageFormat.NV21);

        setRotation(camera, parameters, requestedCameraId);

        if (mFocusMode != null) {
            if (parameters.getSupportedFocusModes().contains(
                mFocusMode)) {
                parameters.setFocusMode(mFocusMode);
            } else {
                Log.i(TAG, "Camera focus mode: " + mFocusMode + " is not supported
on this device.");
            }
        }

        // setting mFocusMode to the one set in the params
        mFocusMode = parameters.getFocusMode();

        if (mFlashMode != null) {
            if (parameters.getSupportedFlashModes().contains(
                mFlashMode)) {
                parameters.setFlashMode(mFlashMode);
            } else {
                Log.i(TAG, "Camera flash mode: " + mFlashMode + " is not supported on
this device.");
            }
        }

        // setting mFlashMode to the one set in the params
        mFlashMode = parameters.getFlashMode();

        camera.setParameters(parameters);

        // Four frame buffers are needed for working with the camera:
        //
        // one for the frame that is currently being executed upon in doing detection
        // one for the next pending frame to process immediately upon completing
detection
        // two for the frames that the camera uses to populate future preview images
        camera.setPreviewCallbackWithBuffer(new CameraPreviewCallback());

```

```

        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));
        camera.addCallbackBuffer(createPreviewBuffer(mPreviewSize));

        return camera;
    }

    /**
     * Gets the id for the camera specified by the direction it is facing. Returns -1 if
     no such
     * camera was found.
     *
     * @param facing the desired camera (front-facing or rear-facing)
     */
    private static int getIdForRequestedCamera(int facing) {
        Camera.CameraInfo cameraInfo = new Camera.CameraInfo();
        for (int i = 0; i < Camera.getNumberOfCameras(); ++i) {
            Camera.getCameraInfo(i, cameraInfo);
            if (cameraInfo.facing == facing) {
                return i;
            }
        }
        return -1;
    }

    /**
     * Selects the most suitable preview and picture size, given the desired width and
     height.
     * <p/>
     * Even though we may only need the preview size, it's necessary to find both the
     preview
     * size and the picture size of the camera together, because these need to have
     the same aspect
     * ratio. On some hardware, if you would only set the preview size, you will get a
     distorted
     * image.
     *
     * @param camera the camera to select a preview size from
     * @param desiredWidth the desired width of the camera preview frames
     * @param desiredHeight the desired height of the camera preview frames
     * @return the selected preview and picture size pair
     */
    private static SizePair selectSizePair(Camera camera, int desiredWidth, int
    desiredHeight) {
        List<SizePair> validPreviewSizes = generateValidPreviewSizeList(camera);

        // The method for selecting the best size is to minimize the sum of the

```

differences between

// the desired values and the actual values for width and height. This is certainly not the

// only way to select the best size, but it provides a decent tradeoff between using the

// closest aspect ratio vs. using the closest pixel area.

SizePair selectedPair = null;

int minDiff = Integer.MAX_VALUE;

for (SizePair sizePair : validPreviewSizes) {

Size size = sizePair.previewSize();

int diff = Math.abs(size.getWidth() - desiredWidth) +

Math.abs(size.getHeight() - desiredHeight);

if (diff < minDiff) {

selectedPair = sizePair;

minDiff = diff;

}

}

return selectedPair;

}

/**

** Stores a preview size and a corresponding same-aspect-ratio picture size. To avoid distorted*

** preview images on some devices, the picture size must be set to a size that is the same*

** aspect ratio as the preview size or the preview may end up being distorted. If the picture*

** size is null, then there is no picture size with the same aspect ratio as the preview size.*

**/*

private static class SizePair {

private Size mPreview;

private Size mPicture;

public SizePair(Camera.Size previewSize,

Camera.Size pictureSize) {

mPreview = new Size(previewSize.width, previewSize.height);

if (pictureSize != null) {

mPicture = new Size(pictureSize.width, pictureSize.height);

}

}

public Size previewSize() {

return mPreview;

}

@SuppressWarnings("unused")

```

    public Size pictureSize() {
        return mPicture;
    }
}

/**
 * Generates a list of acceptable preview sizes. Preview sizes are not acceptable
 if there is
 * not a corresponding picture size of the same aspect ratio. If there is a
 corresponding
 * picture size of the same aspect ratio, the picture size is paired up with the
 preview size.
 * <p/>
 * This is necessary because even if we don't use still pictures, the still picture
 size must be
 * set to a size that is the same aspect ratio as the preview size we choose.
 Otherwise, the
 * preview images may be distorted on some devices.
 */
private static List<SizePair> generateValidPreviewSizeList(Camera camera) {
    Camera.Parameters parameters = camera.getParameters();
    List<Camera.Size> supportedPreviewSizes =
        parameters.getSupportedPreviewSizes();
    List<Camera.Size> supportedPictureSizes =
        parameters.getSupportedPictureSizes();
    List<SizePair> validPreviewSizes = new ArrayList<>();
    for (Camera.Size previewSize : supportedPreviewSizes) {
        float previewAspectRatio = (float) previewSize.width / (float)
previewSize.height;

        // By looping through the picture sizes in order, we favor the higher
        resolutions.
        // We choose the highest resolution in order to support taking the full
        resolution
        // picture later.
        for (Camera.Size pictureSize : supportedPictureSizes) {
            float pictureAspectRatio = (float) pictureSize.width / (float)
pictureSize.height;
            if (Math.abs(previewAspectRatio - pictureAspectRatio) <
ASPECT_RATIO_TOLERANCE) {
                validPreviewSizes.add(new SizePair(previewSize, pictureSize));
                break;
            }
        }
    }
}

// If there are no picture sizes with the same aspect ratio as any preview sizes,
allow all

```

```

        // of the preview sizes and hope that the camera can handle it. Probably
        unlikely, but we
        // still account for it.
        if (validPreviewSizes.size() == 0) {
            Log.w(TAG, "No preview sizes have a corresponding same-aspect-ratio
picture size");
            for (Camera.Size previewSize : supportedPreviewSizes) {
                // The null picture size will let us know that we shouldn't set a picture
size.
                validPreviewSizes.add(new SizePair(previewSize, null));
            }
        }

        return validPreviewSizes;
    }

    /**
     * Selects the most suitable preview frames per second range, given the desired
frames per
     * second.
     *
     * @param camera the camera to select a frames per second range from
     * @param desiredPreviewFps the desired frames per second for the camera
preview frames
     * @return the selected preview frames per second range
     */
    private int[] selectPreviewFpsRange(Camera camera, float desiredPreviewFps) {
        // The camera API uses integers scaled by a factor of 1000 instead of floating-
point frame
        // rates.
        int desiredPreviewFpsScaled = (int) (desiredPreviewFps * 1000.0f);

        // The method for selecting the best range is to minimize the sum of the
differences between
        // the desired value and the upper and lower bounds of the range. This may
select a range
        // that the desired value is outside of, but this is often preferred. For example,
if the
        // desired frame rate is 29.97, the range (30, 30) is probably more desirable
than the
        // range (15, 30).
        int[] selectedFpsRange = null;
        int minDiff = Integer.MAX_VALUE;
        List<int[]> previewFpsRangeList =
camera.getParameters().getSupportedPreviewFpsRange();
        for (int[] range : previewFpsRangeList) {
            int deltaMin = desiredPreviewFpsScaled -
range[Camera.Parameters.PREVIEW_FPS_MIN_INDEX];

```

```

        int deltaMax = desiredPreviewFpsScaled -
range[Camera.Parameters.PREVIEW_FPS_MAX_INDEX];
        int diff = Math.abs(deltaMin) + Math.abs(deltaMax);
        if (diff < minDiff) {
            selectedFpsRange = range;
            minDiff = diff;
        }
    }
    return selectedFpsRange;
}

/**
 * Calculates the correct rotation for the given camera id and sets the rotation in
the
 * parameters. It also sets the camera's display orientation and rotation.
 *
 * @param parameters the camera parameters for which to set the rotation
 * @param cameraId the camera id to set rotation based on
 */
private void setRotation(Camera camera, Camera.Parameters parameters, int
cameraId) {
    WindowManager windowManager =
        (WindowManager)
mContext.getSystemService(Context.WINDOW_SERVICE);
    int degrees = 0;
    int rotation = windowManager.getDefaultDisplay().getRotation();
    switch (rotation) {
        case Surface.ROTATION_0:
            degrees = 0;
            break;
        case Surface.ROTATION_90:
            degrees = 90;
            break;
        case Surface.ROTATION_180:
            degrees = 180;
            break;
        case Surface.ROTATION_270:
            degrees = 270;
            break;
        default:
            Log.e(TAG, "Bad rotation value: " + rotation);
    }

    Camera.CameraInfo cameraInfo = new Camera.CameraInfo();
    Camera.getCameraInfo(cameraId, cameraInfo);

    int angle;
    int displayAngle;

```

```

        if (cameraInfo.facing == Camera.CameraInfo.CAMERA_FACING_FRONT) {
            angle = (cameraInfo.orientation + degrees) % 360;
            displayAngle = (360 - angle); // compensate for it being mirrored
        } else { // back-facing
            angle = (cameraInfo.orientation - degrees + 360) % 360;
            displayAngle = angle;
        }

        // This corresponds to the rotation constants in {@link Frame}.
        mRotation = angle / 90;

        camera.setDisplayOrientation(displayAngle);
        parameters.setRotation(angle);
    }

    /**
     * Creates one buffer for the camera preview callback. The size of the buffer is
     based off of
     * the camera preview size and the format of the camera image.
     *
     * @return a new preview buffer of the appropriate size for the current camera
     settings
     */
    private byte[] createPreviewBuffer(Size previewSize) {
        int bitsPerPixel = ImageFormat.getBitsPerPixel(ImageFormat.NV21);
        long sizeInBits = previewSize.getHeight() * previewSize.getWidth() *
bitsPerPixel;
        int bufferSize = (int) Math.ceil(sizeInBits / 8.0d) + 1;

        //
        // NOTICE: This code only works when using play services v. 8.1 or higher.
        //

        // Creating the byte array this way and wrapping it, as opposed to using
        .allocate(),
        // should guarantee that there will be an array to work with.
        byte[] byteArray = new byte[bufferSize];
        ByteBuffer buffer = ByteBuffer.wrap(byteArray);
        if (!buffer.hasArray() || (buffer.array() != byteArray)) {
            // I don't think that this will ever happen. But if it does, then we wouldn't
be
            // passing the preview content to the underlying detector later.
            throw new IllegalStateException("Failed to create valid buffer for camera
source.");
        }

        mBytesToByteBuffer.put(byteArray, buffer);
        return byteArray;
    }

```

```

    }

//=====
//=====
    // Frame processing

//=====
//=====

/**
 * Called when the camera has a new preview frame.
 */
private class CameraPreviewCallback implements Camera.PreviewCallback {
    @Override
    public void onPreviewFrame(byte[] data, Camera camera) {
        mFrameProcessor.setNextFrame(data, camera);
    }
}

/**
 * This runnable controls access to the underlying receiver, calling it to process
frames when
 * available from the camera. This is designed to run detection on frames as fast
as possible
 * (i.e., without unnecessary context switching or waiting on the next frame).
 * <p/>
 * While detection is running on a frame, new frames may be received from the
camera. As these
 * frames come in, the most recent frame is held onto as pending. As soon as
detection and its
 * associated processing are done for the previous frame, detection on the mostly
recently
 * received frame will immediately start on the same thread.
 */
private class FrameProcessingRunnable implements Runnable {
    private Detector<?> mDetector;
    private long mStartTimeMillis = SystemClock.elapsedRealtime();

    // This lock guards all of the member variables below.
    private final Object mLock = new Object();
    private boolean mActive = true;

    // These pending variables hold the state associated with the new frame
awaiting processing.
    private long mPendingTimeMillis;
    private int mPendingFrameId = 0;
    private ByteBuffer mPendingFrameData;

```

```

FrameProcessingRunnable(Detector<?> detector) {
    mDetector = detector;
}

/**
 * Releases the underlying receiver. This is only safe to do after the associated
thread
 * has completed, which is managed in camera source's release method above.
 */
@SuppressWarnings("Assert")
void release() {
    assert (mProcessingThread.getState() == Thread.State. TERMINATED);
    mDetector.release();
    mDetector = null;
}

/**
 * Marks the runnable as active/not active. Signals any blocked threads to
continue.
 */
void setActive(boolean active) {
    synchronized (mLock) {
        mActive = active;
        mLock.notifyAll();
    }
}

/**
 * Sets the frame data received from the camera. This adds the previous
unused frame buffer
 * (if present) back to the camera, and keeps a pending reference to the frame
data for
 * future use.
 */
void setNextFrame(byte[] data, Camera camera) {
    synchronized (mLock) {
        if (mPendingFrameData != null) {
            camera.addCallbackBuffer(mPendingFrameData.array());
            mPendingFrameData = null;
        }

        if (!mBytesToByteBuffer.containsKey(data)) {
            Log.d(TAG,
                "Skipping frame. Could not find ByteBuffer associated with the
image " +
                "data from the camera.");
            return;
        }
    }
}

```

```

    }

    // Timestamp and frame ID are maintained here, which will give
downstream code some
    // idea of the timing of frames received and when frames were dropped
along the way.
    mPendingTimeMillis = SystemClock.elapsedRealtime() - mStartTimeMillis;
    mPendingFrameId++;
    mPendingFrameData = mBytesToByteBuffer.get(data);

    // Notify the processor thread if it is waiting on the next frame (see
below).
    mLock.notifyAll();
}
}

/**
 * As long as the processing thread is active, this executes detection on frames
 * continuously. The next pending frame is either immediately available or
hasn't been
 * received yet. Once it is available, we transfer the frame info to local
variables and
 * run detection on that frame. It immediately loops back for the next frame
without
 * pausing.
 * <p/>
 * If detection takes longer than the time in between new frames from the
camera, this will
 * mean that this loop will run without ever waiting on a frame, avoiding any
context
 * switching or frame acquisition time latency.
 * <p/>
 * If you find that this is using more CPU than you'd like, you should probably
decrease the
 * FPS setting above to allow for some idle time in between frames.
 */
@Override
public void run() {
    Frame outputFrame;
    ByteBuffer data;

    while (true) {
        synchronized (mLock) {
            while (mActive && (mPendingFrameData == null)) {
                try {
                    // Wait for the next frame to be received from the camera, since
we
                    // don't have it yet.

```

```

        mLock.wait();
    } catch (InterruptedException e) {
        Log.d(TAG, "Frame processing loop terminated.", e);
        return;
    }
}

if (!mActive) {
    // Exit the loop once this camera source is stopped or released. We
check      // this here, immediately after the wait() above, to handle the case
where      // setActive(false) had been called, triggering the termination of this
           // loop.
           return;
}

outputFrame = new Frame.Builder()
    .setImageData(mPendingFrameData, mPreviewSize.getWidth(),
        mPreviewSize.getHeight(), ImageFormat.NV21)
    .setId(mPendingFrameId)
    .setTimestampMillis(mPendingTimeMillis)
    .setRotation(mRotation)
    .build();

    // Hold onto the frame data locally, so that we can use this for
detection  // below. We need to clear mPendingFrameData to ensure that this
buffer isn't // recycled back to the camera before we are done using that data.
            data = mPendingFrameData;
            mPendingFrameData = null;
        }

    // The code below needs to run outside of synchronization, because this
will allow  // the camera to add pending frame(s) while we are running detection on
the current // frame.

    try {
        mDetector.receiveFrame(outputFrame);
    } catch (Throwable t) {
        Log.e(TAG, "Exception thrown from receiver.", t);
    } finally {
        mCamera.addCallbackBuffer(data.array());
    }
}

```

```

    }
}

```

Camera Source Preview Java Class

```

package com.example.ewan_uwtsd_app_project.camera;

import android.Manifest;
import android.content.Context;
import android.content.res.Configuration;
//import android.support.annotation.RequiresPermission;
import android.util.AttributeSet;
import android.util.Log;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.ViewGroup;

import androidx.annotation.RequiresPermission;

import com.google.android.gms.common.images.Size;

import java.io.IOException;

public class CameraSourcePreview extends ViewGroup {
    private static final String TAG = CameraSourcePreview.class.getSimpleName();

    private Context mContext;
    private SurfaceView mSurfaceView;
    private boolean mStartRequested;
    private boolean mSurfaceAvailable;
    private CameraSource mCameraSource;

    public CameraSourcePreview(Context context, AttributeSet attrs) {
        super(context, attrs);
        mContext = context;
        mStartRequested = false;
        mSurfaceAvailable = false;

        mSurfaceView = new SurfaceView(context);
        mSurfaceView.getHolder().addCallback(new SurfaceCallback());
        addView(mSurfaceView);
    }

    @RequiresPermission(Manifest.permission.CAMERA)
    public void start(CameraSource cameraSource) throws IOException,
        SecurityException {
        if (cameraSource == null) {

```

```

        stop();
    }

    mCameraSource = cameraSource;

    if (mCameraSource != null) {
        mStartRequested = true;
        startIfReady();
    }
}

public void stop() {
    if (mCameraSource != null) {
        mCameraSource.stop();
    }
}

public void release() {
    if (mCameraSource != null) {
        mCameraSource.release();
        mCameraSource = null;
    }
}

@RequiresPermission(Manifest.permission.CAMERA)
private void startIfReady() throws IOException, SecurityException {
    if (mStartRequested && mSurfaceAvailable) {
        mCameraSource.start(mSurfaceView.getHolder());
        mStartRequested = false;
    }
}

private class SurfaceCallback implements SurfaceHolder.Callback {
    @Override
    public void surfaceCreated(SurfaceHolder surface) {
        mSurfaceAvailable = true;
        try {
            startIfReady();
        } catch (SecurityException se) {
            Log.e(TAG, "Do not have permission to start the camera", se);
        } catch (IOException e) {
            Log.e(TAG, "Could not start camera source.", e);
        }
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder surface) {
        mSurfaceAvailable = false;
    }
}

```

```

    }

    @Override
    public void surfaceChanged(SurfaceHolder holder, int format, int width, int
height) {
    }
}

@Override
protected void onLayout(boolean changed, int left, int top, int right, int bottom) {
    int previewWidth = 320;
    int previewHeight = 240;
    if (mCameraSource != null) {
        Size size = mCameraSource.getPreviewSize();
        if (size != null) {
            previewWidth = size.getWidth();
            previewHeight = size.getHeight();
        }
    }

    // Swap width and height sizes when in portrait, since it will be rotated 90
degrees
    if (isPortraitMode()) {
        int tmp = previewWidth;
        previewWidth = previewHeight;
        previewHeight = tmp;
    }

    final int viewWidth = right - left;
    final int viewHeight = bottom - top;

    int childWidth;
    int childHeight;
    int childXOffset = 0;
    int childYOffset = 0;
    float widthRatio = (float) viewWidth / (float) previewWidth;
    float heightRatio = (float) viewHeight / (float) previewHeight;

    // To fill the view with the camera preview, while also preserving the correct
aspect ratio,
    // it is usually necessary to slightly oversize the child and to crop off portions
along one
    // of the dimensions. We scale up based on the dimension requiring the most
correction, and
    // compute a crop offset for the other dimension.
    if (widthRatio > heightRatio) {
        childWidth = viewWidth;
        childHeight = (int) ((float) previewHeight * widthRatio);
    }

```

```

        childYOffset = (childHeight - viewHeight) / 2;
    } else {
        childWidth = (int) ((float) previewWidth * heightRatio);
        childHeight = viewHeight;
        childXOffset = (childWidth - viewWidth) / 2;
    }

    for (int i = 0; i < getChildCount(); ++i) {
        // One dimension will be cropped. We shift child over or up by this offset
and adjust
        // the size to maintain the proper aspect ratio.
        getChildAt(i).layout(
            -1 * childXOffset, -1 * childYOffset,
            childWidth - childXOffset, childHeight - childYOffset);
    }

    try {
        startIfReady();
    } catch (IOException e) {
        Log.d(TAG, "Could not start camera source.", e);
    } catch (SecurityException se) {
        Log.d(TAG, "Does not have permission to start the camera.", se);
    }
}

private boolean isPortraitMode() {
    int orientation = mContext.getResources().getConfiguration().orientation;
    if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
        return false;
    }
    if (orientation == Configuration.ORIENTATION_PORTRAIT) {
        return true;
    }

    Log.d(TAG, "isPortraitMode returning false by default");
    return false;
}
}

```

Landmarks (Rooms)

All the code in the landmark (room) classes show the functions implemented to make the camera work on the app. It also shows what text will be displayed on each individual room page.

Landmark Interface

```
package com.example.ewan_uwtsd_app_project.objLandmark.landmarks;
```

```

import com.example.ewan_uwtsd_app_project.objArea.area.area;

public interface landmark extends area {

    int getLandmarkID();

    int getIDLength();

    String getLandmarkName();

    String getLandmarkDesc();

    int getPastLandmarkID();

    int getFutureLandmarkID();

    String getResourceID();

    String getImgResource();

    String getImgResource(String imgResourceID);
}

```

Landmark (Room) 000000 Java Class

```

package com.example.ewan_uwtsd_app_project.objLandmark.landmarks;

import com.example.ewan_uwtsd_app_project.objArea.area.area00000;
import com.example.ewan_uwtsd_app_project.objIDFormatter;
import
com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmarkFactory;
import com.example.ewan_uwtsd_app_project.objType;

import java.util.ArrayList;
import java.util.List;

public class landmark000000 {

    // Declares static landmark method to create landmark object. Returns new
instance of this class.
    public static landmark createLandmark() {
        return (landmark) new landmark000000();
    }

    // Declares default constructor for landmark000000 class.
    public landmark000000() {
        super();
        setLandmarkData(
            0,

```

```

        "Default Landmark",
        "Description of " + mStrLandmarkName,
        0,
        0);
    }

    // int to store landmark ID.
    private int mIntLandmarkID;
    // Declares String to store length of landmark ID. Calls method to get ID length
    based on object.
    public static final int mIntIDLength =
    objIDFormatter.getIDLength(objType.landmark);

    // String to store landmark name.
    private String mStrLandmarkName;
    // String to store landmark description.
    private String mStrLandmarkDesc;

    // int to store past landmark ID.
    private int mIntPastLandmarkID;
    // int to store future landmark ID.
    private int mIntFutureLandmarkID;

    // String to store resource ID of landmark.
    private String mStrResourceID;
    // list to store list of image resource IDs.
    private List <String> mListImgResources = new ArrayList<>();

    // int method to get landmark ID.
    public int getLandmarkID() {
        return mIntLandmarkID;
    }
    // method to set landmark ID. Parameter: Landmark ID as int.
    private void setLandmarkID(int landmarkID) {
        mIntLandmarkID = landmarkID;
    }
    // Declares int method to get length of valid landmark ID.
    public int getIDLength() {
        return mIntIDLength;
    }
    // Declares String to get landmark name.
    public String getLandmarkName() {
        return mStrLandmarkName;
    }
    // Declares method to set landmark name. Parameter: Name of landmark.
    private void setLandmarkName(String landmarkName) {
        mStrLandmarkName = landmarkName;
    }
}

```

```

// Declares String method to get landmark description.
public String getLandmarkDesc() {
    return mStrLandmarkDesc;
}
// Declares method to set landmark description. Parameter: Description of
landmark.
private void setLandmarkDesc(String landmarkDesc) {
    mStrLandmarkDesc = landmarkDesc;
}
// Declares int method to get past landmark ID.
public int getPastLandmarkID() {
    return mIntPastLandmarkID;
}
// Declares method to set past landmark ID. Parameter: Past landmark ID.
private void setPastLandmarkID(int pastLandmarkID) {
    mIntPastLandmarkID = pastLandmarkID;
}
// Declares int method to get future landmark ID.
public int getFutureLandmarkID() {
    return mIntFutureLandmarkID;
}
// Declares method to set future landmark ID. Parameter: Future landmark ID.
private void setFutureLandmarkID(int futureLandmarkID) {
    mIntFutureLandmarkID = futureLandmarkID;
}
// Declares String to get resource ID of landmark.
public String getResourceID() {
    return mStrResourceID;
}
// Declares method to set resource ID of landmark.
private void setResourceID() {
    // Concatenates the following string with the landmark ID. Calls method to
format ID appropriately.
    mStrResourceID = "landmark" + objIDFormatter.formatID(objType.landmark,
mIntLandmarkID);
}

// Declares String method to get image resource ID of a landmark.
public String getImgResource() {
    // Defaults to returning first image resource ID in list. To be updated in future.
    return mListImgResources.get(0);
}

// Declares String method to get image resource for landmark. (Overloaded
method)
// Parameter: Image resource to get based on ID.
public String getImgResource(String imgResourceID) {
    // If size of image resource list is greater than 0, then..

```

```

    if (mListImgResources.size() > 0) {
        // Loops based on number of image resources in list.
        for (int i = 0; i < mListImgResources.size(); i++) {
            // If image resource in list matches ID, then..
            if (mListImgResources.get(i).equals(imgResourceID)) {
                // Returns with image resource ID from item in list.
                return mListImgResources.get(i);
            }
        }
        // Returns null if not found
        return null;
    }
    else {
        // Returns null if list is empty.
        return null;
    }
}

// Declares method to set image resource in list. Parameter: Image resource ID
// to add.
private void setImgResources(String resourceID) {
    mListImgResources.add(resourceID);
}

// Declares method to clear image resource list.
private void clearImgResourceList() {
    // If list size is greater than 0, then clears list.
    if (mListImgResources.size() > 0) {
        mListImgResources.clear();
    }
}

// Declares method to set landmark data.
// Parameters: Landmark ID, name, and description, as well as past and future
// landmark IDs.
protected void setLandmarkData(int landmarkID, String landmarkName, String
landmarkDesc, int pastLandmarkID, int futureLandmarkID) {

    // Sets current landmark details.
    setLandmarkID(landmarkID);
    setLandmarkName(landmarkName);
    setLandmarkDesc(landmarkDesc);
    // Sets past and future landmark IDs.
    setPastLandmarkID(pastLandmarkID);
    setFutureLandmarkID(futureLandmarkID);
    //Sets image resources.
    setResourceID();
    clearImgResourceList();
}

```

```

        setImgResources(getResourceID());
    }

    // Declares method to set landmark of an object.
    // Parameter: ID of landmark to set.
    protected void setLandmark(int landmarkID) {

        // Declares new landmark object using ID passed.
        landmark updatedLandmark = landmarkFactory.newLandmark(landmarkID);
        // Declares int to store new landmark's ID.
        int id = updatedLandmark.getLandmarkID();
        // Declares String to store new landmark's name.
        String name = updatedLandmark.getLandmarkName();
        // Declares String to store new landmark's description.
        String desc = updatedLandmark.getLandmarkDesc();
        // Declares int to store past landmark ID's new landmark object.
        int past = updatedLandmark.getPastLandmarkID();
        // Declares int to store future landmark ID's new landmark object.
        int future = updatedLandmark.getFutureLandmarkID();
        // Calls method to update landmark data using variables above.
        setLandmarkData(id, name, desc, past, future);
    }
}

```

Landmark (Room) 000001 Java Class

```
package com.example.ewan_uwtsd_app_project.objLandmark.landmarks;
```

```
public class landmark000001 extends landmark000000 {
```

```

    public static landmark createLandmark() {
        return (landmark) new landmark000001();
    }

```

```

    landmark000001() {
        setLandmarkData(
            000001,
            "Student Support",
            "Help is Here!",
            2,
            0);
        setArea(1);
        setLocation(1);
        setCountry(1);
        setRegion(1);
    }

```

```

    private void setCountry(int i) {
    }

```

```

private void setRegion(int i) {
}

private void setLocation(int i) {
}

private void setArea(int i) {
}
}

```

Landmark (Room) 000002 Java Class

```
package com.example.ewan_uwtsd_app_project.objLandmark.landmarks;
```

```

public class landmark000002 extends landmark000000 {

    public static landmark createLandmark() {
        return (landmark) new landmark000002();
    }

    landmark000002() {
        setLandmarkData(2,
            "Location 2",
            "Location 2.",
            0,
            1);
        setArea(1);
        setLocation(1);
        setCountry(1);
        setRegion(1);
    }

    private void setRegion(int i) {
    }

    private void setCountry(int i) {
    }

    private void setLocation(int i) {
    }

    private void setArea(int i) {
    }

}

```

Landmark (Room) 000003 Java Class

```
package com.example.ewan_uwtsd_app_project.objLandmark.landmarks;
```

```
public class landmark000003 extends landmark000000 {
```

```
    public static landmark createLandmark() {  
        return (landmark) new landmark000003();  
    }
```

```
    landmark000003() {  
        setLandmarkData(  
            3,  
            "Location 3.",  
            "Location 3.",  
            0,  
            0);  
        setArea(2);  
        setLocation(1);  
        setCountry(1);  
        setRegion(1);  
    }
```

```
    private void setRegion(int i) {  
    }
```

```
    private void setCountry(int i) {  
    }
```

```
    private void setLocation(int i) {  
    }
```

```
    private void setArea(int i) {  
    }  
}
```

Landmark (Room) 000004 Java Class

```
package com.example.ewan_uwtsd_app_project.objLandmark.landmarks;
```

```
public class landmark000004 {
```

```
    public static landmark createLandmark() {  
        return (landmark) new landmark000004();  
    }
```

```
    landmark000004() {  
        setLandmarkData(  
            4,  
            "Location 4",
```

```

        "Location 4.",
        0,
        0);
    setArea(3);
    setLocation(1);
    setCountry(1);
    setRegion(1);
}

private void setRegion(int i) {
}

private void setCountry(int i) {
}

private void setLocation(int i) {
}

private void setArea(int i) {
}

private void setLandmarkData(int i, String stradey_castle, String s, int i1, int i2) {
}
}

```

Landmark (Room) Factory

```

package com.example.ewan_uwtsd_app_project.objLandmark.landmarks;

import com.example.ewan_uwtsd_app_project.objLandmark.landmarks.*;

public class landmarkFactory {
    public static landmark newLandmark(int landID) {
        return null;
    }
}

```

App Functions

All the code in the app function class show the functions created to ensure that the app works, this includes showing the area, region, country and other functions including parameters.

```

package com.example.ewan_uwtsd_app_project;

import android.content.Context;

public class appFunctions {

```

```

// Declares static String method to get object ID.
// Parameter: Object type enum value.
public static String getObjectID(objType obj) {
    // Switch compares passed object type enum value to cases.
    // If object type is landmark, then..
    if (objType.landmark.equals(obj)) { // Returns the following.
        return "landmark";
        // If object type is area, then..
    } else if (objType.Area.equals(obj)) {
        return "area";
        // If object type is location, then..
    } else if (objType.Location.equals(obj)) {
        return "location";
        // If object type is country, then..
    } else if (objType.Country.equals(obj)) {
        return "country";
        // If object type is region then..
    } else if (objType.Region.equals(obj)) {
        return "region";
        // If object type does not match cases, then return empty string.
    }
    return "";
}

// Declares static int to get image resource.
public static int getImgResourceID(Context context, String pVariableName, String
pResourcename, String pPackageName) {

    // Declares int to store resource ID. Calls method to get it.
    int id = getResourceId(context, pVariableName,
pResourcename,pPackageName);

    // If id is not 0, then..
    if (id != 0) {
        // Returns ID.
        return id;
    }
    else {
        // Returns default/placeholder image resource.
        return getResourceId(context, "landmark000000",
pResourcename,pPackageName);
    }

}

// Declares static int method that gets the resource ID.
// Parameters: Context of activity, image resource name as String, resource
directory name, and package name.

```

```

    private static int getResourceId(Context context, String pVariableName, String
pResourcename, String pPackageName) throws RuntimeException {
        try {
            // Returns result from getting resource using passed parameters.
            return context.getResources().getIdentifier(pVariableName,
pResourcename, pPackageName);
        } catch (Exception e) {
            // Returns runtime error if value returned is null/empty/invalid.
            throw new RuntimeException("Error getting Resource ID.", e);
        }
    }
}

```

Landmark Activity

All the code in the camera classes show the functions implemented to make the rooms show correctly on the app, as well as code to assist with debugging in the event of an error occurring.

```

package com.example.ewan_uwtsd_app_project;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

// Imports the following Landmark object packages.
import androidx.appcompat.app.AppCompatActivity;

import com.example.ewan_uwtsd_app_project.objLandmark.landmarks.*;

public class landmarkActivity extends AppCompatActivity {

    // Declares string for log tag. Used for debugging purposes.
    public static final String LOG_TAG = landmarkActivity.class.getSimpleName();
    // Declares ImageView for displaying preview image of landmark.
    public ImageView mImgLandmarkPreview;
    // Declares TextView to display landmark name.
    public TextView mTxtLandmarkName;
    // Declares TextViews to display landmark ID.
    public TextView mTxtLandmarkID;
    // Declares TextViews to display landmark's position.
    public TextView mTxtLandmarkPosition;
    // Declares TextView to display description of landmark.
    public TextView mTxtLandmarkDesc;

```

```

// Declares Buttons to allow users to view past or future landmarks.
public Button mBtnPastLandmark;
public Button mBtnFutureLandmark;
// Declares String to store copy of landmark ID.
public String mStrLandmarkID;
// Declares Landmark object to get data of selected landmark.
public landmark mObjLandmark;

/* onCreate event for mainActivity.
 * Sets content view, assigns ID for layout elements, and onClick events.
 *
 * Parameters pass in savedInstanceState.
 */
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // Sets content view for landmarkActivity to landmark page layout.
    setContentView(R.layout.landmark_page);
    // Calls method to initialise intent. Used to get landmark ID from mainActivity.
    initialiseIntent();
    // Calls method to initialise layout components, such as widgets (buttons,
    textviews, etc.).
    initialiseLayoutComponents();
    // Calls method to update landmark object being used.
    updateLandmarkObject();
}

// Declares method to initialise intent. Used to get values from mainActivity.
public void initialiseIntent() {
    // Declares Intent and assigns by calling method to get intent from
    mainActivity.
    Intent intent = getIntent();
    // Gets landmark ID and stores it here.
    mStrLandmarkID = intent.getStringExtra(MainActivity.landmarkID);
}

// Declares method to initialise layout components, such as widgets (Buttons,
TextViews, etc.).
public void initialiseLayoutComponents() {
    // Sets ID for ImageView that display preview image of landmark.
    mImgLandmarkPreview = findViewById(R.id.landmark_imageview);
    // Sets ID for TextViews displaying the landmark's ID and position
    mTxtLandmarkID = findViewById(R.id.landmark_id_textview);
    mTxtLandmarkPosition = findViewById(R.id.landmark_position_textview);
    // Sets ID for TextViews displaying the landmark's name and its description.
    mTxtLandmarkName = findViewById(R.id.landmark_name_textview);
    mTxtLandmarkDesc = findViewById(R.id.landmark_desc_textview);
    // Sets ID for Buttons that scroll through past and future landmarks.

```

```

        mBtnPastLandmark = findViewById(R.id.past_landmark_button);
        mBtnFutureLandmark = findViewById(R.id.future_landmark_button);
        // Sets OnClickListener events for buttons.
        mBtnPastLandmark.setOnClickListener(new viewPastLandmark());
        mBtnFutureLandmark.setOnClickListener(new viewFutureLandmark());
    }

    // Declares method that updates landmark timeline.
    public void updateLandmarkTimeline(String pastLandID, String futureLandID) {
        // Calls method to update landmark timeline buttons based on IDs found in
        landmark object.

        updateLandmarkButtons(mBtnPastLandmark, checkPastFutureLandmarks(pastLandID
        ));

        updateLandmarkButtons(mBtnFutureLandmark, checkPastFutureLandmarks(futureLa
        ndID));
    }

    // Declares method to update state of landmark buttons.
    public void updateLandmarkButtons(Button btn, boolean check) {
        // If ID value exists, then..
        if (check) {
            // Enables the button and makes it visible.
            btn.setVisibility(View.VISIBLE);
            btn.setEnabled(true);
        }
        // Else if value does not exist, then..
        else {
            // Disables the button and makes it invisible.
            btn.setVisibility(View.GONE);
            btn.setEnabled(false);
        }
    }

    // Declares boolean method that checks if past/future IDs exist in landmark
    object.
    public boolean checkPastFutureLandmarks(String landmarkID) {
        // If past/future landmark ID is not null/empty, then..
        if (landmarkID != null) {
            // Returns true
            return true;
        }
        else {
            // Returns false if past/future landmark ID is empty/null.
            return false;
        }
    }
}

```

```

// Declares method that updates landmark object when called.
// Default method does not require parameters.
public void updateLandmarkObject() {

    // Assigns new landmark object based on new landmark ID. Parses String into
    int.
    mObjLandmark =
    landmarkFactory.newLandmark(Integer.parseInt(mStrLandmarkID));
    // Calls method to update landmark page to display new landmark info.
    updateLandmarkPage();
}

// Declares method that updates landmark object when called. (Overloaded
method)
// Parameter: landmark ID as string.
public void updateLandmarkObject(String landmarkID) {
    // Assigns new landmark object based on passed landmark ID. Parses String
    into int.
    mObjLandmark =
    landmarkFactory.newLandmark(Integer.parseInt(landmarkID));
    // Calls method to update landmark page to display new landmark info.
    updateLandmarkPage();
}

// Declares method that updates content on landmark page.
public void updateLandmarkPage() {
    // Declares string to store image resource name.
    String strObjID = mObjLandmark.getImgResource();
    // Declares int to get image resource ID. Calls method to get image resource
    ID using resource name from landmark object.
    int intObjID = appFunctions.getImgResourceID(landmarkActivity.this, strObjID,
    "drawable",getPackageName());
    // Sets image resource for ImageView.
    mImgLandmarkPreview.setImageResource(intObjID);
    // Sets text for Landmark information into following TextViews.
    mTxtLandmarkName.setText(mObjLandmark.getLandmarkName());
    // Declares String to concatenate and display ID. Calls method to format ID
    appropriately.
    String formattedID = "ID: " +
    objIDFormatter.formatID(objType.landmark,mObjLandmark.getLandmarkID());
    // Sets text for ID.
    mTxtLandmarkID.setText(formattedID);
    // Declares String to concatenate strings for landmark's position. Concatenates
    area, location, country, and region.
    String position = mObjLandmark.getAreaName() + ", " +
    mObjLandmark.getLocationName() + ", " +
    mObjLandmark.getCountryName() + ", " +

```

```

mObjLandmark.getRegionName();
    // Sets text as landmark's position.
    mTxtLandmarkPosition.setText(position);
    // Sets text to display description of landmark.
    mTxtLandmarkDesc.setText(mObjLandmark.getLandmarkDesc());
    // Calls method to update landmark timeline.
    updateLandmarkTimeline(
        // Gets past/future IDs from landmark object.
        objIDFormatter.formatID(objType.landmark
,mObjLandmark.getPastLandmarkID()),
        objIDFormatter.formatID(objType.landmark
,mObjLandmark.getFutureLandmarkID()));
    }

    // Declares class to view past landmarks. Used as OnClickListener event.
    class viewPastLandmark implements View.OnClickListener {
        // Overrides onClick event.
        @Override
        public void onClick(View view) {
            // Calls method to past landmark object using future landmark ID. Calls
            method to format ID appropriately.

            updateLandmarkObject(objIDFormatter.formatID(objType.landmark,mObjLandmark.
            getPastLandmarkID()));
        }
    }

    // Declares class to view future landmarks. Used as OnClickListener event.
    class viewFutureLandmark implements View.OnClickListener {
        // Overrides onClick event.
        @Override
        public void onClick(View view) {
            // Calls method to update landmark object using future landmark ID. Calls
            method to format ID appropriately.

            updateLandmarkObject(objIDFormatter.formatID(objType.landmark,mObjLandmark.
            getFutureLandmarkID()));
        }
    }
}

```

Landmark Collection Java Class

Inheriting from the landmark factory, within the landmark class, all the code in the landmark collection class show the functions implemented to make the camera work on the app.

```
package com.example.ewan_uwtsd_app_project;
```

```

// import com.example.ewan_uwtsd_app_project.objLandmark.landmarks;
import com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmark;
import
com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmarkFactory;
// import
com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmarkFactory;

// Imports the following utilities.
import android.util.Log;
import java.util.ArrayList;
import java.util.List;

public class landmarkCollection {

    // Declares string for log tag. Used for debugging purposes.
    public static final String LOG_TAG = landmarkCollection.class.getSimpleName();

    // Declares static List that will contain all landmark objects generated here.
    private static List<landmark> mListLandmarkCollection = new ArrayList<>();
    // Declares static method to initialise list.
    private static void initialiseList() {
        // Loop that adds landmark objects to list. Stops based on number of
landmark objects in app.
        // Starts at 0, continues looping when less than X, and increments by 1 each
loop.
        for (int i = 0; i < 4; i++) {
            // Declares int to store landmark ID
            int landID = i + 1;
            // Declares temporary landmark object to store new generated landmark
based on ID.
            landmark tempLand = landmarkFactory.newLandmark(landID);
            /* Adds newly generated landmark object to list.
            * Objects added are saved to list after loop and when temporary landmark
object is overwritten. */
            mListLandmarkCollection.add(tempLand);
        }
    }

    // Declares static int method that gets the size of landmark list.
    private static int getListSize() {
        // Returns size of list as int.
        return mListLandmarkCollection.size();
    }

    // Declares static String method that compares name entered in EditText with
landmark name value.
    // Parameter: Name inputted by user as String.
    private static String compareSearchName(String searchName) {

```

```

        // Declares string to store passed String in lower case. Prevents errors if user
types in upper case.
        String sName = searchName.toLowerCase();

        // Loop that checks each item in the list and compares to search name passed
in.
        // Starts at 0, continues looping when less than X, and increments by 1 each
loop.
        for (int i = 0; i < 4; i++) {

            // Declares int to store landmark ID
            int listItem = i + 1;
            // Declares String to store copy of landmark name from current landmark
object.
            String itemName =
mListLandmarkCollection.get(i).getLandmarkName().toLowerCase();
            // If landmark name matches name user inputted, then..
            if (sName.equals(itemName) ) {
                // Returns landmark ID from current landmark object. Calls method to
format ID appropriately.
                return
objIDFormatter.formatID(objType.landmark,mListLandmarkCollection.get(i).getLand
markID());
            }
        }
        // Returns empty string if nothing was found.
        return "000000";
    }

    // Declares static String that searches landmark list of names.
    public static String searchByName(String searchName) {
        // Calls method to initialise list.
        initialiseList();
        // Declares int that stores length of list.
        int listLength = getListSize();
        // Prints following to log.
        Log.d(LOG_TAG,Integer.toString(listLength));
        // If length of list is greater than 0, then..
        if(listLength > 0) {
            // Returns landmark ID by calling method to compare user inputted name
with list.
            return compareSearchName(searchName);
        }
        // Returns the following if no landmark objects available.
        return "000000";
    }
}

```

Main Activity Java Class

The Main Activity Java class is where most functions are located, showing how the main features on the interface would work and what events would take place when the QR codes are scanned.

```
package com.example.ewan_uwtsd_app_project;

import android.annotation.SuppressLint;
import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Point;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.DialogFragment;

import com.example.ewan_uwtsd_app_project.barcode.BarcodeCaptureActivity;
import com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmark;
import
com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmarkFactory;
import com.google.android.gms.common.api.CommonStatusCodes;
import com.google.android.gms.vision.barcode.Barcode;

public class MainActivity extends AppCompatActivity {
    // Declares string for log tag. Used for debugging purposes.
    private static final String LOG_TAG = MainActivity.class.getSimpleName();
    // Declares static final int for storing barcode reader request code. Needed to
    handle requests to scan barcode.
    private static final int BARCODE_READER_REQUEST_CODE = 1;
    // Declares static final String for package/context directory of variable. Used
    when changing activities.
    public static final String LANDMARK_ID =
    "com.example.ewan_uwtsd_app_project";
    public static String landmarkID;

    // Declares TextView for displaying the text content of a scanned barcode.
    public TextView mResultTextView;

    // Declares ImageView for displaying preview image of landmark.
```

```

public ImageView mImgLandmarkPreview;

// Declares Buttons for accessing or using different parts of the app.
public Button mShowDialogButton;
public Button mViewLandmarkButton; // Views landmark scanned via QR code.
public Button mScanBarcodeButton; // Uses camera to scan QR code.
public Button mSearchManualButton; // Allows user to manually search for
landmark.

//Declares String to store contents of QR code.
private String mStrQRCode;

//Declares static String to store landmark ID. Used when changing activities to
view landmark page.
public static String mStrLandmarkID;
//Declares landmark interface object to store landmark user wants to
view/scanned via QR code.
public landmark mObjLandmark;

/* onCreate event for mainActivity.
 * Sets content view, assigns ID for layout elements, and onClick events.
 *
 * Parameters pass in savedInstanceState.
 */
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // Sets content view to be barcode scanner layout. Used to scan and display
content of a barcode.
    setContentView(R.layout.activity_main);
    // Calls method to initialise layout components.
    initialiseLayoutComponents();

    mObjLandmark = landmarkFactory.newLandmark(0);
}

// Declares method to that sets IDs and events for components (buttons, text,
etc.).
private void initialiseLayoutComponents() {
    // Sets ID of TextView
    mResultTextView = findViewById(R.id.result_textview);
    // Sets ID for ImageView
    mImgLandmarkPreview = findViewById(R.id.result_imageview);
    // Sets image resource for ImageView. Uses default/placeholder image.
    mImgLandmarkPreview.setImageResource(R.drawable.landmark000000);
    mShowDialogButton = findViewById(R.id.show_dialog_button);

    // Sets ID for buttons.

```

```

mViewLandmarkButton = findViewById(R.id.view_landmark_button);
mScanBarcodeButton = findViewById(R.id.scan_barcode_button);
mSearchManualButton = findViewById(R.id.search_manual_button);

// Calls method to toggle landmark results. Set to false by default.
toggleLandmarkResult(false);

//Sets onClickListener event for buttons.
mViewLandmarkButton.setOnClickListener(new viewLandmarkPage()); //
Allows user to view landmark pages.
mScanBarcodeButton.setOnClickListener(new scanBarcode()); // Allows
user to scan a barcode.
mSearchManualButton.setOnClickListener(new searchLandmark()); // Allows
user to manually search landmarks.
mShowDialogButton.setOnClickListener(new showDialog());
}

// Declares method that toggles layout components used to show landmark
found when scanning QR codes.
// Parameter: boolean value that toggles components visibility and
enables/disables buttons.
private void toggleLandmarkResult(boolean toggle) {
    // If boolean value
    if (toggle) {
        // Enables the following..
        mViewLandmarkButton.setEnabled(true); //Button is enabled.
        mViewLandmarkButton.setVisibility(View.VISIBLE); //Button becomes
visible.
        mResultTextView.setVisibility(View.VISIBLE); //TextView becomes
visible.
    }
    else {
        // Disables the following..
        mViewLandmarkButton.setEnabled(false); //Button is disabled.
        mViewLandmarkButton.setVisibility(View.GONE); //Button becomes
invisible.
        mResultTextView.setVisibility(View.GONE); //TextView becomes
invisible.
    }
}

// Declares method that checks and validates the contents of the QR code.
// Parameter: Contents of QR code as String.
public void qrCodeValidator(String qrCode) {

    Log.d(LOG_TAG,"Validating QR Code..");
    // Declares String that contains the format that valid QR codes use.
    String strCodeMatch = "\\d\\d\\d\\d\\d\\d\\d\\d";

```

```

        // Declares Int to store length of a valid id in a QR code. Calls method to get
length.
        int intMaxCodeLen = mObjLandmark.getIDLength();

        // Prints log message to check contents of code.
        Log.d(LOG_TAG, "Code " + qrCode + " is being checked..");
        // Declares int to store length of string stored in QR code.
        int mIntStrLen = qrCode.length();

        // If contents of QR code match length of landmark ID length, then..
        if (mIntStrLen == intMaxCodeLen) {
            // Prints following to log if correct length.
            Log.d(LOG_TAG, "QR code is correct length.");
            // If format of contents in QR code matches the following, then..
            if (qrCode.matches(strCodeMatch)) {
                //Prints following to log.
                Log.d(LOG_TAG, "Compatible QR code found!");
                // Stores content of QR code as String
                mStrQRCode = qrCode;
            }
            else {
                // Prints following to log if unsuccessful.
                Log.d(LOG_TAG, "Error: Code form is incompatible with app.");
                mStrQRCode = "";
            }
        }
        else {
            // Prints the following to log and toasts a message
            Log.d(LOG_TAG, "Incompatible QR code found. Cannot be used!");
            mStrQRCode = "";
            Toast.makeText(this, "QR code is incompatible with app.",
Toast.LENGTH_SHORT).show();
        }
    }

    // Declares method to store QR data.
    // Parameter: Landmark ID stored in QR code.
    public void storeQRData(String qrCodeData) {

        // Prints following to log.
        Log.d(LOG_TAG, "Storing QR Code data..");
        // Declares int to parse value in QR code from String to int.
        int landmarkID = Integer.parseInt(qrCodeData);
        // Prints following to log.
        Log.d(LOG_TAG, "Parsed Int: " + Integer.toString(landmarkID));
        // Assigns new landmark object by passing ID from QR code.
        mObjLandmark = landmarkFactory.newLandmark(landmarkID);
        // Stores copy of Landmark ID from landmark object. Calls method to format

```

ID appropriately.

```
    mStrLandmarkID = objIDFormatter.formatID(objType.landmark,
mObjLandmark.getLandmarkID());
    // Calls method to display data stored inside landmark object.
    displayLandmarkData();
}

// Declares method for displaying landmark data.
public void displayLandmarkData() {
    // Prints following to log.
    Log.d(LOG_TAG, "Displaying landmark name..");
    // Declares string and stores landmark name.
    String landmarkName = mObjLandmark.getLandmarkName();
    // Sets text of TextView to display landmark name.
    mResultTextView.setText(landmarkName);
    // Declares string for storing image resource id by name. Calls method to
return it.
    String strObjID = mObjLandmark.getImgResource();
    // Declares int that stores image
    int intObjID = appFunctions.getImgResourceID(MainActivity.this, strObjID,
"drawing",getPackageName());
    // Sets image resource of ImageView using ID
    mImgLandmarkPreview.setImageResource(intObjID);
    // Calls method to toggle landmark results.
    toggleLandmarkResult(true);
}

// Declares method to get QR code data.
@SuppressWarnings("StringFormatInvalid")
public void getQRCodeData(int requestCode, int resultCode, Intent data) {
    // If passed request code matches request code set in mainActivity, then..
    if (requestCode == BARCODE_READER_REQUEST_CODE) {
        // If result code equals success status code, then..
        if (resultCode == CommonStatusCodes.SUCCESS) {
            // If data from intent is not null, meaning not empty, then..
            if (data != null) {
                // Declare barcode and stores parsed barcode data.
                Barcode barcode =
data.getParcelableExtra(BarcodeCaptureActivity.BarcodeObject);
                // Declares Point array to store corner points data of scanned
barcode?
                Point[] p = barcode.cornerPoints;
                // Sets TextView to display value from scanned barcode.
                qrCodeValidator(barcode.displayValue);
                if (mStrQRCode != null && !mStrQRCode.isEmpty()) {
                    storeQRData(mStrQRCode);
                }
                // Else (if data is empty), sets TextView to say no barcode was
```

```

captured.
        } else mResultTextView.setText(R.string.no_barcode_captured);
        // Else (if result code is not a success), prints error to log.
    } else Log.e(LOG_TAG,
String.format(getString(R.string.barcode_error_format),
        CommonStatusCodes.getStatusCodeString(resultCode)));
        // Else (if request code does not match), calls onActivityResult from base.
    } else super.onActivityResult(requestCode, resultCode, data);
}

/* Sets onActivityResult event for mainActivity. When new intent is declared and
starts activity to get a result.
* Verifies barcode reader request code and returns data from barcode.
*
* Parameters required are the request code finalised in mainActivity, the result
code received from changing activity,
* and the data collected from the scanned barcode.
*/
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Switch compares request code with cases.
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
        // If request code is 1, then..
        case 1:
            // Gets QR data.
            getQRCodeData(requestCode, resultCode, data);
            // If request code is none of the above, then do nothing.
        default:
            break;
    }
}

// Declares class to view landmark page. Used as OnClickListener event.
class viewLandmarkPage implements View.OnClickListener {
    // Overrides onClick event.
    @Override
    public void onClick(View view) {
        // Declares new intent. Changes activity to the following.
        Intent intent = new Intent(MainActivity.this, landmarkActivity.class);
        // Stores landmark ID as extra when changing activity.
        intent.putExtra(LANDMARK_ID, mStrLandmarkID);
        // Starts the activity.
        startActivity(intent);
    }
}

// Declares class to scan QR codes. Used as OnClickListener event.

```

```

class scanBarcode implements View.OnClickListener {

    @Override
    public void onClick(View view) {
        /* Declares new intent. Takes user from mainActivity (sets layout to
barcode scanner)
        * to BarcodeCaptureActivity (sets layout to barcode capture).
        */
        Intent intent = new Intent(getApplicationContext(),
BarcodeCaptureActivity.class);
        // Starts activity and waits for a result when returns.
        startActivityForResult(intent, BARCODE_READER_REQUEST_CODE);
    }
}

// Declares class to search landmarks manually. Used as OnClickListener event.
class searchLandmark implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        // Declares new intent. Changes activity to the following.
        Intent intent = new Intent(MainActivity.this, searchLandmarkActivity.class);
        // Starts the activity.
        startActivity(intent);
    }
}

// Declares onSaveInstanceState event. Stores following state when orientation
changes.
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    // Stores contents of QR code.
    outState.putString("qrCode", mStrQRCode);
}

// Declares onRestoreInstanceState event. Recovers following state when
orientation changes.
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);

    // Assigns restored QR code.
    mStrQRCode= savedInstanceState.getString("qrCode");
    // If QR code stored in variable is not empty, then..
//    if (!mStrQRCode.isEmpty()) {
    // Calls method to store QR data, and display landmark.
    //storeQRData(mStrQRCode);
    //}
}

```

```

    }
    public class MyDialogFragment extends DialogFragment {
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
            // Use the Builder class for convenient dialog construction
            AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
            builder.setTitle("App Title");
            builder.setMessage("This is an alert with no consequence");
            builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int id) {
                    // You don't have to do anything here if you just want it dismissed
when clicked
                }
            });

            // Create the AlertDialog object and return it
            return builder.create();
        }
    }

    class showDialog implements View.OnClickListener {
        @Override
        public void onClick(View view) {
            // Calls method to check search results.
            DialogFragment dialog = new MainActivity.MyDialogFragment();
            dialog.show(getSupportFragmentManager(), "MyDialogFragmentTag");
        }
    }
}

```

Search Landmark Activity Java Class

Inheriting from the landmark interface within the landmark repository, all the code in the search landmark activity class show the functions implemented to make the searching for rooms work on the app.

```
package com.example.ewan_uwtsd_app_project;
```

```

import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

```

```

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.DialogFragment;

import com.example.ewan_uwtsd_app_project.objLandmark.*;
import com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmark;
import
com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmarkFactory;

public class searchLandmarkActivity extends AppCompatActivity {
    // Declares string for log tag. Used for debugging purposes.
    public static final String LOG_TAG =
searchLandmarkActivity.class.getSimpleName();
    // Declares static final String for package/context directory of variable. Used
when changing activities.
    public static final String LANDMARK_ID =
"com.example.ewan_uwtsd_app_project.LANDMARK_ID";

    // Declares EditText to track user input.
    public EditText mEditSearchID;    // EditText for landmark ID.
    public EditText mEditSearchName;  // EditText for landmark name.

    // Declares Button for searching for landmarks based on user input.
    public Button mBtnSearchLandmark;

    // Declares TextView for displaying landmark name.
    public TextView mTxtLandmarkName;
    // Declares ImageView for displaying preview image of landmark.
    public ImageView mImgLandmarkPreview;
    // Declares Button to go to landmark's page.
    public Button mBtnViewLandmark;

    // Declares Button to return user to mainActivity.
    public Button mBtnReturnMain;

    // Declares Strings to store user inputs.
    public String mStrSearchID;    // Stores Landmark ID inputted by user.
    public String mStrSearchName;  // Stores Landmark Name inputted by user.

    // Declares String to store copy of landmark ID from successful searches.
    public String mStrLandmarkID;
    // Declares landmark to store landmark object.
    public landmark mObjLandmark;

    /* onCreate event for mainActivity.
    * Sets content view, assigns ID for layout elements, and onClick events.
    *
    * Parameters pass in savedInstanceState.

```

```

*/
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    // Sets content view to be the search landmark page layout.
    setContentView(R.layout.search_landmark_page);
    // Calls method to initialise layout components, such as widgets (Buttons,
    EditTexts, etc.).
    initialiseLayoutComponents();
}

// Declares method for initialising layout components, such as widgets (Buttons,
    EditTexts, etc.).
    public void initialiseLayoutComponents() {

        // Sets IDs for EditTexts for user input.
        mEditSearchID = findViewById(R.id.search_id_edittext);
        mEditSearchName = findViewById(R.id.search_name_edittext);
        // Sets ID for Button to search for landmarks.
        mBtnSearchLandmark = findViewById(R.id.search_landmark_button);
        // Sets ID for the following used to display found landmarks.
        mTxtLandmarkName = findViewById(R.id.result_landmark_name_textview);
        mImgLandmarkPreview = findViewById(R.id.result_landmark_imageview);
        mBtnViewLandmark = findViewById(R.id.result_landmark_button);
        // Sets ID for Button used to return to main menu/mainActivity.
        mBtnReturnMain = findViewById(R.id.return_main_button);
        // Calls method to toggle result view. Disables result view.
        toggleResultView(false);
        // Sets the onClickListener events for the following buttons.
        mBtnSearchLandmark.setOnClickListener(new searchLandmark());
        mBtnViewLandmark.setOnClickListener(new viewLandmark());
        mBtnReturnMain.setOnClickListener(new returnMain());
    }

    // Declares method used to check search results and compare with list of
    landmarks on app.
    public void checkSearch() {

        mObjLandmark = null;

        // Following gets user input for both landmark ID and landmark name.
        Converted to string.
        mStrSearchID = mEditSearchID.getText().toString();
        mStrSearchName = mEditSearchName.getText().toString();

        // If EditText for inputting landmark ID is not empty, then..
        if (!mStrSearchID.isEmpty()) {
            // Calls method to compare landmark ID with landmark objects.

```

```

        compareLandmarkID(mStrSearchID);
    }
    // If EditText for inputting landmark name is not empty, then..
    if (!mStrSearchName.isEmpty() && !resultFound()) {
        // Calls method to compare landmark name with landmark objects.
        compareLandmarkName(mStrSearchName);
    }

    // If result has not been found, then..
    if (!resultFound()) {
        // Calls method to hide result view.
        toggleResultView(false);
    }
}

// Declares boolean method to check if result was found.
public boolean resultFound() {
    // Returns boolean value from following check.
    return (mObjLandmark != null);
}

// Declares method for comparing landmark id inputted by user with landmark
objects.
// Parameter: landmark id inputted by user as string.
public void compareLandmarkID(String landmarkID) {

    // Declares String to contain format for valid landmark ID.
    String idFormat = "\\d\\d\\d\\d\\d\\d";

    // If landmarkID passed matches correct format, then..
    if (landmarkID.matches(idFormat)) {
        // Assigns new landmark object using landmark ID inputted by user.
        mObjLandmark =
landmarkFactory.newLandmark(Integer.parseInt(landmarkID));

        // IF landmark object does not equal 0, then..
        if (mObjLandmark.getLandmarkID() != 0) {

            // Stores landmark ID from landmark object.
            mStrLandmarkID =
objIDFormatter.formatID(objType.landmark,mObjLandmark.getLandmarkID());
            // Sets text of TextView to display landmark name.
            mTxtLandmarkName.setText(mObjLandmark.getLandmarkName());
            // Declares string to store name of resource id from landmark object.
            String strObjID = mObjLandmark.getImgResource();
            // Declares int to get image resource ID for landmark. Passes in
resource ID got from landmark object.

```

```

        int intObjID =
appFunctions.getImgResourceID(searchLandmarkActivity.this, strObjID,
"drawable",getPackageName());
        // Sets image resource using ID.
        mImgLandmarkPreview.setImageResource(intObjID);
        // Calls method to toggle landmark result view.
        toggleResultView(true);

    }
    else {
        // Toasts the following message if search result is invalid.
        Toast.makeText(this, "Error: Invalid or unknown ID used.",
Toast.LENGTH_SHORT).show();
    }
}
else {
    // Toasts the following message if search result is invalid.
    Toast.makeText(this, "Error: Invalid or unknown ID used.",
Toast.LENGTH_SHORT).show();
}
}

// Declares method for comparing landmark name inputted by user with
landmark objects.
// Parameter: landmark name inputted by user as string.
public void compareLandmarkName(String landmarkName) {

    // Declares string to store id of landmark. Calls method to search and
compare landmark name with landmark objects.
    String id = landmarkCollection.searchByName(landmarkName);

    // Assigns new landmark object using landmark ID from successful searches.
Parses String to int.
    mObjLandmark = landmarkFactory.newLandmark(Integer.parseInt(id));

    // If landmark ID does not equal the following, then..
    if (mObjLandmark.getLandmarkID() != 0) {

        // Stores landmark ID from landmark object. Calls method to format ID
appropriately.
        mStrLandmarkID =
objIDFormatter.formatID(objType.landmark,mObjLandmark.getLandmarkID());
        // Sets text of TextView to display landmark name.
        mTxtLandmarkName.setText(mObjLandmark.getLandmarkName());
        // Declares string to store name of resource id from landmark object.
        String strObjID = mObjLandmark.getImgResource();
        // Declares int to get image resource ID for landmark. Passes in resource
ID got from landmark object.

```

```

        int intObjID =
appFunctions.getImgResourceID(searchLandmarkActivity.this, strObjID,
"drawable",getPackageName());
        // Sets image resource using ID.
        mImgLandmarkPreview.setImageResource(intObjID);
        // Calls method to toggle landmark result view.
        toggleResultView(true);

    }
    else {
        // Toasts the following message if search result is invalid.
        Toast.makeText(this, "Error: Invalid or unknown ID used.",
Toast.LENGTH_SHORT).show();
    }

}

// Declares method to toggle landmark search results.
// Parameter: boolean value to show/hide results.
public void toggleResultView(boolean toggle) {
    // If passed boolean is true, then..
    if (toggle) {
        // Enables the following..
        mTxtLandmarkName.setVisibility(View.VISIBLE);    //Sets landmark name
visible.
        mImgLandmarkPreview.setVisibility(View.VISIBLE);    //Sets landmark
image preview visible.
        mBtnViewLandmark.setVisibility(View.VISIBLE);    //Sets view landmark
button visible.
        mBtnViewLandmark.setEnabled(true);                //Enables view landmark
button.
    }
    else {
        // Disables the following..
        mTxtLandmarkName.setVisibility(View.GONE);    //Sets landmark name
invisible.
        mImgLandmarkPreview.setVisibility(View.GONE);    //Sets landmark image
preview invisible.
        mBtnViewLandmark.setVisibility(View.GONE);    //Sets view landmark
button invisible.
        mBtnViewLandmark.setEnabled(false);                //Disables view landmark
button.
    }
}

// Declares class to search for landmarks. Acts as OnClickListener event.
class searchLandmark implements View.OnClickListener {
    @Override

```

```

        public void onClick(View view) {
            // Calls method to check search results.
            checkSearch();
        }
    }

    // Declares class to view landmark page. Acts as OnClickListener event.
    class viewLandmark implements View.OnClickListener {
        @Override
        public void onClick(View view) {
            // Declares new intent. Changes from searchLandmarkActivity to
            landmarkActivity. To display landmark page.
            Intent intent = new Intent(searchLandmarkActivity.this,
            landmarkActivity.class);
            // Stores reference to landmark ID. To be used in landmark activity.
            intent.putExtra(LANDMARK_ID, mStrLandmarkID);
            // Starts the activity.
            startActivity(intent);
        }
    }

    // Declares class to return to main menu. Acts as OnClickListener event.
    class returnMain implements View.OnClickListener {
        @Override
        public void onClick(View view) {
            // Declares new intent. Changes from searchLandmarkActivity to
            mainActivity. To display main menu.
            Intent intent = new Intent(searchLandmarkActivity.this, MainActivity.class);
            // Starts the activity.
            startActivity(intent);
        }
    }
}

```

Activity Main Layout Design (Home Page Design)

This code shows how the home page would look.

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".3MainActivity">

    <ImageView
        android:id="@+id/result_imageview"

```

```

        android:layout_width="match_parent"
        android:layout_height="250dp"
        android:layout_marginStart="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="10dp"
        android:contentDescription="@string/result_imageview_desc"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:srcCompat="@tools:sample/avatars[2]" />

<!-- TextView displays content of QR code as text. -->
<TextView
    android:id="@+id/result_textview"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:layout_marginStart="10dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="10dp"
    android:autoLink="all"
    android:freezesText="true"
    android:gravity="center"
    android:text="@string/no_barcode_captured"
    android:textSize="22sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/result_imageview" />

<!-- Button when pushed allows user to scan QR codes. -->

<Button
    android:id="@+id/view_landmark_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="10dp"
    android:text="View Room Page"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/result_textview" />

<Button
    android:id="@+id/show_dialog_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"

```

```

        android:layout_marginTop="20dp"
        android:layout_marginEnd="10dp"
        android:text="Show Dialog"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/result_textview" />

```

```

<Button
    android:id="@+id/scan_barcode_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/result_textview"
    android:layout_marginStart="10dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="10dp"
    android:text="Scan QR Code"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/view_landmark_button" />

```

```

<Button
    android:id="@+id/search_manual_button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="10dp"
    android:text="Search"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/scan_barcode_button" />

```

```

</androidx.constraintlayout.widget.ConstraintLayout>

```

Barcode Capture Design (Capturing the QR Code)

This code explains shows the design for when the QR code gets captured.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/topLayout"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:keepScreenOn="true">

```

```

    <!-- Camera Source Preview. Captures QR Code. -->

```

```

<!-- Window layout width should match parent. Be full screen. -->
<com.example.ewan_uwtsd_app_project.camera.CameraSourcePreview
    android:id="@+id/preview"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1">
</com.example.ewan_uwtsd_app_project.camera.CameraSourcePreview>
</LinearLayout>

```

Landmark Page Design

This code explains how the individual landmark (room) pages would look.

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:id="@+id/landmark_imageview"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:layout_marginStart="10dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="10dp"
        android:contentDescription="@string/result_imageview_desc"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:srcCompat="@tools:sample/avatars[2]" />

    <TextView
        android:id="@+id/landmark_name_textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Room Name"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/landmark_imageview" />

    <LinearLayout
        android:id="@+id/linearlayout2"

```

```

        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="10dp"
        android:orientation="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/landmark_name_textview">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="25dp"
    android:layout_marginBottom="5dp"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/landmark_id_textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Landmark ID" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="25dp"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/landmark_position_textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/object_landmark_position" />

</LinearLayout>
</LinearLayout>

<RelativeLayout
    android:id="@+id/relativeLayout"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:layout_marginStart="10dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="10dp"
    app:layout_constraintEnd_toEndOf="parent"

```

```

app:layout_constraintHorizontal_bias="0.0"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/linearLayout2">

<TextView
    android:id="@+id/landmark_desc_textview"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:scrollbars="vertical"
    android:text="@string/object_landmark_description" />
</RelativeLayout>

<Button
    android:id="@+id/past_landmark_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginBottom="10dp"
    android:text="Past Landmark"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

<Button
    android:id="@+id/future_landmark_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="10dp"
    android:layout_marginBottom="10dp"
    android:text="Future Landmark"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

[Search Landmark Page Design](#)

This code shows how the search landmark page would look.

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="250dp"

```

```

        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="20dp"
        android:orientation="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <TextView
            android:id="@+id/search_id_textview"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Search by Room ID"
            android:textStyle="bold"/>

        <EditText
            android:id="@+id/search_id_edittext"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName"
            android:text=""
            android:hint="Search by Room ID" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout3"
        android:layout_width="250dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="20dp"
        android:orientation="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout">

        <TextView
            android:id="@+id/search_name_textview"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Search by Name"
            android:textStyle="bold" />

        <EditText
            android:id="@+id/search_name_edittext"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"

```

```

        android:inputType="textPersonName"
        android:text=""
        android:hint="Search by Room Name"/>
</LinearLayout>

<Button
    android:id="@+id/search_landmark_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="10dp"
    android:text="Search Room"
    android:textSize="14sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout3" />

<Button
    android:id="@+id/return_main_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:text="Return to the Home Page"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />

<LinearLayout
    android:id="@+id/linearLayout4"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="10dp"
    android:gravity="center"
    android:orientation="vertical"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/search_landmark_button">

    <TextView
        android:id="@+id/result_landmark_name_textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="5dp"
        android:text="Landmark Name"
        android:textStyle="bold"

```

```

        android:textSize="18sp" />

        <ImageView
            android:id="@+id/result_landmark_imageview"
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:layout_marginBottom="5dp"
            android:contentDescription="@string/result_imageview_desc"
            tools:srcCompat="@tools:sample/avatars[2]" />

        <Button
            android:id="@+id/result_landmark_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="View Landmark" />
    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

```

Areas

All the code in the area interface and classes show the functions implemented to make the area show on each individual room page on the app. It also shows what area will be linked to, and displayed, on each individual room page.

Area

```

package com.example.ewan_uwtsd_app_project.objArea.areas;

public interface area extends
com.example.ewan_uwtsd_app_project.objLocation.locations.location {

    int getAreaID();

    String getAreaName();

}

```

Area 00000 Java Class

```

package com.example.ewan_uwtsd_app_project.objArea.areas;

import com.example.ewan_uwtsd_app_project.objArea.areas.areaFactory;
import com.example.ewan_uwtsd_app_project.objLocation.locations.*;

public class area00000 extends
com.example.ewan_uwtsd_app_project.objLocation.locations.location0000
implements com.example.ewan_uwtsd_app_project.objArea.areas.area {

    // Declares static area method to create area object. Returns new instance of this

```

```

class.
    public static area00000 createArea() {
        return new area00000();
    }

    // Declares default constructor for area00000 class.
    public area00000() {
        super();
        setAreaData(0,"Default Area");
    }

    // Declares int to store area ID.
    private int mIntAreaID;
    // Declares String to store area name.
    private String mStrAreaName;

    // Declares int method to get area ID.
    public int getAreaID() {
        return mIntAreaID;
    }
    // Declares method to set area ID. Parameter: Area ID as int.
    private void setAreaID(int areaID) {
        mIntAreaID = areaID;
    }
    // Declares String method to get area name.
    public String getAreaName() {
        return mStrAreaName;
    }
    // Declares method to set area name. Parameter: Name of area as string.
    private void setAreaName(String areaName) {
        mStrAreaName = areaName;
    }

    // Declares method to set area data.
    // Parameters: Area ID as int and area name as string.
    public void setAreaData(int areaID, String areaName) {
        setAreaID(areaID);
        setAreaName(areaName);
    }

    // Declares method to set area of an object.
    // Parameter: ID of area to set.
    protected void setArea(int areaID) {
        // Declares new area object using ID passed.
        areaFactory areaFactory = null;
        com.example.ewan_uwtsd_app_project.objArea.area.area updatedArea =
        areaFactory.newArea(areaID);
        // Declares int to store new area's ID.

```

```

        int id = updatedArea.getAreaID();
        // Declares String to store new area's name.
        String name = updatedArea.getAreaName();
        // Calls method to update area data using variables above.
        setAreaData(id, name);
    }
}

```

Area 00001 Java Class

```
package com.example.ewan_uwtsd_app_project.objArea.areas;
```

```

public class area00001 extends area00000 {

    public static area00000 createArea() {
        return (area00000) new area00001();
    }

    area00001() {
        setAreaData(1,"King's Road");
        setLocation(1);
        setCountry(1);
        setRegion(1);
    }

    public void setRegion(int i) {
        System.out.println("Swansea");
    }

    public void setCountry(int i) {
        System.out.println("Wales");
    }

}

```

Area 00002 Java Class

```
package com.example.ewan_uwtsd_app_project.objArea.areas;
```

```

public class area00002 extends area00000 {

    public static area00000 createArea() {
        return new area00002();
    }

    area00002() {
        setAreaData(
            2,
            "Alexandria Road");
    }
}

```

```

        setLocation(1);
        setCountry(1);
        setRegion(1);
    }

    public void setRegion(int i) {
        System.out.println("Swansea");
    }

    public void setCountry(int i) {
        System.out.println("Wales");
    }
}

```

Area 00003 Java Class

```
package com.example.ewan_uwtsd_app_project.objArea.areas;
```

```

public class area00003 extends area00000 {

    public static area00000 createArea() {
        return (area00000) new area00003();
    }

    area00003() {
        setAreaData(
            3,
            "De le beche Road");
        setLocation(1);
        setCountry(1);
        setRegion(1);
    }

    public void setRegion(int i) {
        System.out.println("Swansea");
    }

    public void setCountry(int i) {
        System.out.println("Wales");
    }
}

```

Area Factory

```
package com.example.ewan_uwtsd_app_project.objArea.areas;
```

```
import com.example.ewan_uwtsd_app_project.objArea.areas.*;
```

```

import com.example.ewan_uwtsd_app_project.objArea.area00001;

public class areaFactory {
    // Declares string for log tag. Used for debugging purposes.
    public static final String LOG_TAG = areaFactory.class.getSimpleName();
    // Declares static area method to create new area object.
    // Parameter: Area ID.
    public static area00000 newArea(int areaID) {
        // Switch compares area ID to cases..
        switch(areaID) {
            // If Area ID matches the following cases, then creates one of the following
            // area objects.
            case 1:
                area00001 area00001 = null;
                return area00001.createArea();
            case 2:
                return area00002.createArea();
            case 3:
                return area00003.createArea();
            // If Area ID does not matches above cases, then returns default/base area
            // object.
            default:
                return area00000.createArea();
        }
    }
}

```

Country

All the code in the country interface and classes show the functions implemented to make the country show on each individual room page on the app. It also shows what area will be linked to, and displayed, on each individual room page.

Country Interface

```

package com.example.ewan_uwtsd_app_project.objCountry.countries;

import com.example.ewan_uwtsd_app_project.objRegion.regions.region;

public interface country extends region {

    //Abstract String method used to get the country ID of a class.
    int getCountryID();
    //Abstract String method used to get the name of a country of a class.
    String getCountryName();

}

```

Country 000 Java Class

```
package com.example.ewan_uwtsd_app_project.objCountry.countries;

import com.example.ewan_uwtsd_app_project.objCountry.countryFactory;
import com.example.ewan_uwtsd_app_project.objRegion.regions.region00;

public class country000 extends region00 implements country {

    // Declares static country method to create country object. Returns new instance
    of this class.
    public static country000 createCountry() {
        //Returns new instance of country000 class.
        return new country000();
    }

    // Declares default constructor for country000 class.
    public country000() {
        super();
        setCountryData(0, "Default Country");
    }

    // Declares int to store country ID.
    private int mIntCountryID;
    // Declares Strings to store country name.
    private String mStrCountryName;

    // Declares int method that returns the country ID.
    public int getCountryID() {
        return mIntCountryID;
    }

    // Declares private method that sets the country ID. Parameter: Country ID as
    int.
    void setCountryID(int countryID) {
        mIntCountryID = countryID;
    }

    // Declares String method that returns the name of the country.
    public String getCountryName() {
        return mStrCountryName;
    }

    // Declares method that sets the name of the country. Parameter: Country name
    as string.
    void setCountryName(String countryName) {
        mStrCountryName = countryName;
    }
}
```

```

// Declares method to set country data.
// Parameter: country ID and country name
void setCountryData(int countryID, String countryName) {
    // Calls methods to set country data..
    setCountryID(countryID);
    setCountryName(countryName);
}

// Declares method to set region of an object.
// Parameter: ID of region to set.
protected void setCountry(int countryID) {
    // Declares new country object using ID passed.
    country updatedCountry = countryFactory.newCountry(countryID);
    // Declares int to store new country's ID.
    int id = updatedCountry.getCountryID();
    // Declares String to store new country's name.
    String name = updatedCountry.getCountryName();
    // Calls method to update country data using variables above.
    setCountryData(id, name);
}
}

```

Country 001 Java Class

```

package com.example.ewan_uwtsd_app_project.objCountry.countries;

public class country001 extends country000 {

    public static country001 createCountry() {
        return new country001();
    }

    private country001() {
        setCountryData(
            1,
            "Wales");
        setRegion(1);
    }
}

```

Country 002 Java Class

```

package com.example.ewan_uwtsd_app_project.objCountry.countries;

public class country002 extends country000 {

    public static country000 createCountry() {
        return new country002();
    }
}

```

```

        private country002() {
            setCountryData(
                2,
                "England");
            setRegion(1);
        }
    }
}

```

Country 003 Java Class

```
package com.example.ewan_uwtsd_app_project.objCountry.countries;
```

```

public class country003 extends country000 {

    public static country000 createCountry() {
        return new country003();
    }

    private country003() {
        setCountryData(
            3,
            "Scotland");
        setRegion(1);
    }
}

```

Country Factory

```
package com.example.ewan_uwtsd_app_project.objCountry;
```

```

import com.example.ewan_uwtsd_app_project.objCountry.countries.country000;
import com.example.ewan_uwtsd_app_project.objCountry.countries.country001;
import com.example.ewan_uwtsd_app_project.objCountry.countries.country002;
import com.example.ewan_uwtsd_app_project.objCountry.countries.country003;

```

```

public class countryFactory {

    // Declares string for log tag. Used for debugging purposes.
    public static final String LOG_TAG = countryFactory.class.getSimpleName();
    // Declares static country method to create new region object.
    // Parameter: Country ID.
    public static country001 newCountry(int countryID) {
        // Switch compares country ID to cases..
        switch (countryID) {
            // If Country ID matches the following cases, then creates one of the
            following country objects.
            case 1:

```

```

        return country001.createCountry();
    case 2:
        return (country001) country002.createCountry();
    case 3:
        return (country001) country003.createCountry();
    // If country ID does not matches above cases, then returns default/base
country object.
    default:
        return (country001) country000.createCountry();
    }
}
}

```

Locations

All the code in the location interface and classes show the functions implemented to make the location show on each individual room page on the app. It also shows what area will be linked to, and displayed, on each individual room page.

Location Interface

```

package com.example.ewan_uwtsd_app_project.objLocation.locations;

import com.example.ewan_uwtsd_app_project.objCountry.countries.country;

public interface location extends country {

    int getLocationID();

    String getLocationName();
}

```

Location 0000 Java Class

```

package com.example.ewan_uwtsd_app_project.objLocation.locations;

import com.example.ewan_uwtsd_app_project.objCountry.countries.country000;
import com.example.ewan_uwtsd_app_project.objLocation.locationFactory;

public class location0000 extends country000 implements location {

    // Declares static location method to create location object. Returns new instance
of this class.
    public static location createLocation() {
        return new location0000();
    }

    // Declares default constructor for location0000 class.
    public location0000() {
        super();
    }
}

```

```

        setLocationData(0,"Default Location");
    }

    // Declares int to store location ID.
    private int mIntLocationID;
    // Declares String to store location name.
    private String mStrLocationName;

    // Declares int method to get location ID.
    public int getLocationID() {
        return mIntLocationID;
    }
    // Declares method to set location ID. Parameter: Location ID as int.
    private void setLocationID(int locationID) {
        mIntLocationID = locationID;
    }
    // Declares String method to get location name.
    public String getLocationName() {
        return mStrLocationName;
    }
    // Declares method to set location name. Parameter: Name of location as string.
    private void setLocationName(String locationName) {
        mStrLocationName = locationName;
    }

    // Declares method to set location data.
    // Parameters: Location ID as int and location name as string.
    void setLocationData(int locationID, String locationName) {
        setLocationID(locationID);
        setLocationName(locationName);
    }

    // Declares method to set location of an object.
    // Parameter: ID of location to set.
    protected void setLocation(int locationID) {
        // Declares new location object using ID passed.
        Location updatedLocation = locationFactory.newLocation(locationID);
        // Declares int to store new location's ID.
        int id = updatedLocation.getLocationID();
        // Declares String to store new location's name.
        String name = updatedLocation.getLocationName();
        // Calls method to update location data using variables above.
        setLocationData(id, name);
    }
}

```

Location 0001 Java Class

```
package com.example.ewan_uwtsd_app_project.objLocation.locations;
```

```
public class location0001 {

    public static location createLocation() {
        return (location) new location0001();
    }

    location0001() {
        setLocationData(
            1,
            "Swansea");
        setCountry(1);
        setRegion(1);
    }

    private void setRegion(int i) {
    }

    private void setCountry(int i) {
    }

    private void setLocationData(int i, String swansea) {
    }
}
```

Location 0002 Java Class

```
package com.example.ewan_uwtsd_app_project.objLocation.locations;
```

```
public class location0002 {

    public static location createLocation() {
        return (location) new location0002();
    }

    location0002() {
        setLocationData(
            2,
            "Swansea");
        setCountry(1);
        setRegion(1);
    }

    private void setRegion(int i) {
    }

    private void setCountry(int i) {
    }
}
```

```

    }

    private void setLocationData(int i, String swansea) {
    }
}

```

Location 0003 Java Class

```
package com.example.ewan_uwtsd_app_project.objLocation.locations;
```

```
public class location0003 extends location0000 {
```

```
    public static location createLocation() {
        return new location0003();
    }

```

```
    location0003() {
        setLocationData(
            3,
            "Swansea");
        setCountry(1);
        setRegion(1);
    }
}

```

Location Factory

```
package com.example.ewan_uwtsd_app_project.objLocation;
```

```
import com.example.ewan_uwtsd_app_project.objLocation.locations.*;
```

```
public class locationFactory {
    // Declares string for log tag. Used for debugging purposes.
    public static final String LOG_TAG = locationFactory.class.getSimpleName();
    // Declares static location method to create new region object.
    // Parameter: Location ID.
    public static location newLocation(int locationID) {
        // Switch compares location ID to cases..
        switch(locationID) {
            // If Location ID matches the following cases, then creates one of the
            following location objects.
            case 1:
                return location0001.createLocation();
            case 2:
                return location0002.createLocation();
            case 3:
                return location0003.createLocation();
            // If location ID does not matches above cases, then returns default/base
            location object.

```

```

        default:
            return location0000.createLocation();
    }
}

```

Region

All the code in the region interface and classes show the functions implemented to make the region show on each individual room page on the app. It also shows what area will be linked to, and displayed, on each individual room page.

Region Interface

```

package com.example.ewan_uwtsd_app_project.objRegion.regions;

public interface region {

    int getRegionID();

    String getRegionName();
}

```

Region 00 Java Class

```

package com.example.ewan_uwtsd_app_project.objRegion.regions;

import com.example.ewan_uwtsd_app_project.objRegion.regions.regionFactory;

public class region00 implements region {

    // Declares static region method to create region object. Returns new instance of
    this class.
    public static region createRegion() {
        return new region00();
    }

    // Declares default constructor for region00 class.
    protected region00() {
        // Calls method to set the following default values.
        setRegionData(0,"Default Region");
    }

    // Declares int to store region ID.
    private int mIntRegionID;
    // Declares String to store region name.
    private String mStrRegionName;
    // Declares int method to get region ID.
    public int getRegionID() {

```

```

        return mIntRegionID;
    }
    // Declares method to set region ID. Parameter: Region ID as int.
    private void setRegionID(int regionID) {
        mIntRegionID = regionID;
    }
    // Declares String method to get region name.
    public String getRegionName() {
        return mStrRegionName;
    }
    // Declares method to set region name. Parameter: Name of region as string.
    private void setRegionName(String regionName) {
        mStrRegionName = regionName;
    }

    // Declares method to set region data.
    // Parameters: Region ID as int and region name as string.
    public void setRegionData(int regionID, String regionName) {
        // Calls methods to set region data..
        setRegionID(regionID);
        setRegionName(regionName);
    }

    // Declares method to set region of an object.
    // Parameter: ID of region to set.
    protected void setRegion(int regionID) {
        // Declares new region object using ID passed.
        region updatedRegion = regionFactory.newRegion(regionID);
        // Declares int to store new region's ID.
        int id = updatedRegion.getRegionID();
        // Declares String to store new region's name.
        String name = updatedRegion.getRegionName();
        // Calls method to update region data using variables above.
        setRegionData(id,name);
    }
}

```

Region 01 Java Class

```

package com.example.ewan_uwtsd_app_project.objRegion.regions;

public class region01 extends region00 {

    public static region createRegion() {
        return new region01();
    }

    private region01() {
        setRegionData(1,"Europe");
    }
}

```

```

    }
}

```

Region Factory Java Class

```

package com.example.ewan_uwtsd_app_project.objRegion.regions;

import com.example.ewan_uwtsd_app_project.objRegion.regions.*;

public class regionFactory {
    // Declares string for log tag. Used for debugging purposes.
    public static final String LOG_TAG = regionFactory.class.getSimpleName();
    // Declares static region method to create new region object.
    // Parameter: Region ID.
    public static region newRegion(int regionID) {
        // Switch compares region ID to cases..
        switch (regionID) {
            // If Region ID matches the following cases, then creates one of the
            // following region objects.
            case 1:
                return region01.createRegion();
            // If region ID does not matches above cases, then returns default/base
            // region object.
            default:
                return region00.createRegion();
        }
    }
}

```

Object ID Formatter Java Class

This class is showing how each object, such as the individual rooms, would be formatted, and shows the area IF statement.

```

package com.example.ewan_uwtsd_app_project;

import android.util.Log;

public class objIDFormatter {

    public static final String LOG_TAG = objIDFormatter.class.getSimpleName();

    // Declares static String to format IDs of objects.
    // Paramteres: Object ID to format as enum, ID of object.
    public static String formatID(objType type, int id) {
        Log.d(LOG_TAG, "Formatting ID " + id + "..");
        // If ID is 0 or less, then return null.
        if (id <= 0) {
            // Prints following to log.

```

```

        Log.d(LOG_TAG,"Passed ID is 0 or less, returning null..");
        return null;
    }

    // Declares String to contain ID as String.
    String strID = Integer.toString(id);
    // Declares int to get length of ID for object. Calls method to check this.
    int length = getIDLength(type);
    // Declares int to store length of loop.
    // Formula: Length of object ID - length of passed ID
    int loopLength = length - strID.length();
    // Declares String and assigns to be empty.
    String strFormatID = "";
    // Loops based on length of passed ID compared to object ID.
    // Default starts at 0, continues looping when less than loop length, and
    increments by 1 each loop.
    for (int i = 0; i < loopLength; i++) {
        // If length of formatted string is greater than 0, then.
        if (strFormatID.length() > 0) {
            // Concat string and adds a 0 to the end of it.
            strFormatID = strFormatID + "0";
        }
        // If length of formatted string is 0, then..
        else {
            // Stores "0" into formatted ID string first.
            strFormatID = "0";
        }
    }
    // Concatenates formatted ID string with passed ID at the end.
    strFormatID = strFormatID + Integer.toString(id);
    // Prints the following to log.
    Log.d(LOG_TAG,strFormatID);
    // Returns formatted ID as string.
    return strFormatID;
}

// Declares static int to get length of object ID.
// Parameter: Object type as enum.
public static int getIDLength(objType object) {
    // Compares object type to cases.
    // If object type matches any of the following, returns a specific value.
    if (objType.landmark.equals(object)) {
        return 6;
    } else if (objType.Area.equals(object)) {
        return 5;
    } else if (objType.Location.equals(object)) {
        return 4;
    }
}

```

```

    } else if (objType.Country.equals(object)) {
        return 3;
    } else if (objType.Region.equals(object)) {
        return 2;
        // Returns 0 if object type does not match any of the above cases.
    }
    return 0;
}
}

```

Object Logging Java Class

The coding for this class shows how the data shown in the different objects (landmark, region, country, location and area) will be logged.

```

package com.example.ewan_uwtsd_app_project;

import android.util.Log;

import com.example.ewan_uwtsd_app_project.objLandmark.landmarks.landmark;
import com.example.ewan_uwtsd_app_project.objRegion.regions.region;
import com.example.ewan_uwtsd_app_project.objCountry.countries.country;
import com.example.ewan_uwtsd_app_project.objLocation.locations.location;
import com.example.ewan_uwtsd_app_project.objArea.areas.area;

public class objLogging {

    // Declares static final String used for containing the log tag name.
    public static final String LOG_TAG = objLogging.class.getSimpleName();

    // Declares static method for printing object information to log.
    // Parameters: log name and region object.
    static void logObjInfo(String name, region objRegion) {

        // Prints the following to log.
        Log.d(LOG_TAG, "--- " + name + " ---");
        Log.d(LOG_TAG, "Region Values");
        Log.d(LOG_TAG, "Region ID: " + objRegion.getRegionID());
        Log.d(LOG_TAG, "Region Name: " + objRegion.getRegionName());
    }

    // Declares static method for printing object information to log.
    // Parameters: log name and country object.
    static void logObjInfo(String name, country objCountry) {

        // Prints the following to log.
        Log.d(LOG_TAG, "--- " + name + " ---");
        Log.d(LOG_TAG, "Country Values");
        Log.d(LOG_TAG, "Country ID: " + objCountry.getCountryID());
    }
}

```

```

    Log.d(LOG_TAG, "Country Name: " + objCountry.getCountryName());
    Log.d(LOG_TAG, "Region Values");
    Log.d(LOG_TAG, "Region ID: " + objCountry.getRegionID());
    Log.d(LOG_TAG, "Region Name: " + objCountry.getRegionName());
}

// Declares static method for printing object information to log.
// Parameters: log name and location object.
static void logObjInfo(String name, location objLocation) {

    // Prints the following to log.
    Log.d(LOG_TAG, "--- " + name + " ---");
    Log.d(LOG_TAG, "Location Values");
    Log.d(LOG_TAG, "Location ID: " + objLocation.getLocationID());
    Log.d(LOG_TAG, "Location Name: " + objLocation.getLocationName());
    Log.d(LOG_TAG, "Country Values");
    Log.d(LOG_TAG, "Country ID: " + objLocation.getCountryID());
    Log.d(LOG_TAG, "Country Name: " + objLocation.getCountryName());
    Log.d(LOG_TAG, "Region Values");
    Log.d(LOG_TAG, "Region ID: " + objLocation.getRegionID());
    Log.d(LOG_TAG, "Region Name: " + objLocation.getRegionName());
}

// Declares static method for printing object information to log.
// Parameters: log name and area object.
static void logObjInfo(String name, area objArea) {

    // Prints the following to log.
    Log.d(LOG_TAG, "--- " + name + " ---");
    Log.d(LOG_TAG, "Area Values");
    Log.d(LOG_TAG, "Area ID: " + objArea.getAreaID());
    Log.d(LOG_TAG, "Area Name: " + objArea.getAreaName());
    Log.d(LOG_TAG, "Location Values");
    Log.d(LOG_TAG, "Location ID: " + objArea.getLocationID());
    Log.d(LOG_TAG, "Location Name: " + objArea.getLocationName());
    Log.d(LOG_TAG, "Country Values");
    Log.d(LOG_TAG, "Country ID: " + objArea.getCountryID());
    Log.d(LOG_TAG, "Country Name: " + objArea.getCountryName());
    Log.d(LOG_TAG, "Region Values");
    Log.d(LOG_TAG, "Region ID: " + objArea.getRegionID());
    Log.d(LOG_TAG, "Region Name: " + objArea.getRegionName());
}

// Declares static method for printing object information to log.
// Parameters: log name and landmark object.
static void logObjInfo(String name, landmark objLandmark) {

    // Prints the following to log.

```

```

    Log.d(LOG_TAG, "--- " + name + " ---");
    Log.d(LOG_TAG, "Landmark Values");
    Log.d(LOG_TAG, "Landmark ID: " + objLandmark.getLandmarkID());
    Log.d(LOG_TAG, "Landmark Name: " + objLandmark.getLandmarkName());
    Log.d(LOG_TAG, "Area Values");
    Log.d(LOG_TAG, "Area ID: " + objLandmark.getAreaID());
    Log.d(LOG_TAG, "Area Name: " + objLandmark.getAreaName());
    Log.d(LOG_TAG, "Location Values");
    Log.d(LOG_TAG, "Location ID: " + objLandmark.getLocationID());
    Log.d(LOG_TAG, "Location Name: " + objLandmark.getLocationName());
    Log.d(LOG_TAG, "Country Values");
    Log.d(LOG_TAG, "Country ID: " + objLandmark.getCountryID());
    Log.d(LOG_TAG, "Country Name: " + objLandmark.getCountryName());
    Log.d(LOG_TAG, "Region Values");
    Log.d(LOG_TAG, "Region ID: " + objLandmark.getRegionID());
    Log.d(LOG_TAG, "Region Name: " + objLandmark.getRegionName());
}
}

```

Object Type Java Class

```
package com.example.ewan_uwtsd_app_project;
```

```

public class objType {
    public static objType landmark;
    public static objType Region;    //Region Object
    public static objType Country;    //Country Object
    public static objType Location;    //Location Object
    public static objType Area;        //Area Object

    public static objType room;
}

```

Appendix F – Console App

```
/**
 *
 * @author Ewan Richards 1802187
 */

import java.sql.*;
import java.util.Scanner;

public class DatabaseTest {
    static final String JDBC_DRIVER =
"com.microsoft.sqlserver.jdbc.SQLServerDriver";
    static final String DB_URL = "jdbc:sqlserver://DESKTOP-KIL1TP1;
databaseName=EwanRichardsProject; trustServerCertificate=true; encrypt=true;";
    static final String USER = "testUser";
    static final String PASS = "test1";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        Scanner scan = new Scanner(System.in);
        System.out.println("Please enter room number");
        String rnum = scan.next();
        try {
            Class.forName(JDBC_DRIVER);

            System.out.println("Connecting to database...");
            conn = DriverManager.getConnection(DB_URL, USER, PASS);

            System.out.println("Creating statement...");
            stmt = conn.createStatement();
            String sql;
            sql = "Select r.RoomNo, r.capacity, rt.TypeName, d.DpName from Room r,
Roomtype rt, Department d where r.typeid=rt.typeid and r.depid=d.depid and
r.roomNo='"+rnum+"'";

            ResultSet rs = stmt.executeQuery(sql);
            while (rs.next()) {
                // Retrieve by column name
                String id = rs.getString("RoomNo");
                int cap = rs.getInt("Capacity");
                String tname = rs.getString("TypeName");
                String dp = rs.getString("DpName");
                System.out.println("Room Description: " +id+" " + cap + " " + tname + "
" + dp);
            }
            rs.close();
        }
    }
}
```

```

        stmt.close();
        conn.close();
    } catch (SQLException se) {
        se.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (stmt != null)
                stmt.close();
        } catch (SQLException se2) {
        }
        try {
            if (conn != null)
                conn.close();
        } catch (SQLException se) {
            se.printStackTrace();
        }
    }

    try {
        Class.forName(JDBC_DRIVER);

        conn = DriverManager.getConnection(DB_URL, USER, PASS);

        stmt = conn.createStatement();

```

```

        String sql = "Select h.hardwareDesc, hr.Quantity, hr.datecheck from hardware
h, hardwareInRoom hr where h.hardwareid=hr.hardwareid and
hr.roomNo='"+rnum+"'";

```

```

        ResultSet rs = stmt.executeQuery(sql);
        while (rs.next()) {
            // Retrieve by column name
            String hd = rs.getString("HardwareDesc");
            int quan = rs.getInt("Quantity");
            String hdate = rs.getString("DateCheck");
            System.out.println("Hardware Description: " +hd+" "+ quan + " " +
hdate);
        }
        rs.close();
        stmt.close();
        conn.close();
    } catch (SQLException se) {
        se.printStackTrace();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {

```

```

    try {
        if (stmt != null)
            stmt.close();
    } catch (SQLException se2) {
    }
    try {
        if (conn != null)
            conn.close();
    } catch (SQLException se) {
        se.printStackTrace();
    }
}
}
try {
    if (conn != null)
        conn.close();
} catch (SQLException se) {
    se.printStackTrace();
}

```

```

try {
    Class.forName(JDBC_DRIVER);

```

```

    conn = DriverManager.getConnection(DB_URL, USER, PASS);

```

```

    stmt = conn.createStatement();

```

```

    String sql = "Select s.softwarename, sr.datecheck from software s,
softwareInRoom sr where s.softwareid=sr.softwareid and sr.roomNo='"+rnum+"';";

```

```

    ResultSet rs = stmt.executeQuery(sql);
    while (rs.next()) {
        // Retrieve by column name
        String sd = rs.getString("SoftwareName");
        String sdate = rs.getString("DateCheck");
        System.out.println("Software Description: " +sd+" " + sdate);
    }
    rs.close();
    stmt.close();
    conn.close();
} catch (SQLException se) {
    se.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (stmt != null)

```

```

        stmt.close();
    } catch (SQLException se2) {
    }
    try {
        if (conn != null)
            conn.close();
    } catch (SQLException se) {
        se.printStackTrace();
    }
}

```

```

try {
    Class.forName(JDBC_DRIVER);

```

```

    conn = DriverManager.getConnection(DB_URL, USER, PASS);

```

```

    stmt = conn.createStatement();

```

```

    String sql = "Select ru.sessid, ru.usagedate, ru.usagetime, ru.duration,
s.SessDesc, l.firstname,l.lastname, d.dpname from roomusage ru, sessions s,
lecturer l, department d where ru.sessid=s.sessid and s.staffID=l.staffid and l.depid
= d.depid and ru.roomNo='"+rnum+"'";

```

```

    ResultSet rs = stmt.executeQuery(sql);
    while (rs.next()) {
        // Retrieve by column name
        int sid = rs.getInt("SessID");
        String udate = rs.getString("UsageDate");
        String utime = rs.getString("UsageTime");
        int dur = rs.getInt("Duration");
        String sdesc = rs.getString("SessDesc");
        String fname = rs.getString("FirstName");
        String lname = rs.getString("LastName");
        String dep = rs.getString("DpName");

        System.out.println("Session Time: " + sid + " " + udate + " " + utime + "
" + dur);
        System.out.println("Session Description: " +sdesc+" " + fname + " " +
lname + " " + dep);
    }
    rs.close();
    stmt.close();
    conn.close();
} catch (SQLException se) {
    se.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
}

```

```
    } finally {  
        try {  
            if (stmt != null)  
                stmt.close();  
        } catch (SQLException se2) {  
        }  
        try {  
            if (conn != null)  
                conn.close();  
        } catch (SQLException se) {  
            se.printStackTrace();  
        }  
    }  
}  
  
}
```

Appendix G – Java Application

```
package jdbcv3;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;
import java.sql.*;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.event.*;
import java.awt.*;

/**
 *
 * @author ER
 */
public class DtBeForm extends javax.swing.JFrame {

    static final String JDBC_DRIVER =
"com.microsoft.sqlserver.jdbc.SQLServerDriver";
    static final String DB_URL = "jdbc:sqlserver://DESKTOP-KIL1TP1;
databaseName=EwanRichardsProject; trustServerCertificate=true; encrypt=true;";
    static final String USER = "testUser";
    static final String PASS = "test1";

    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JTextField jTextField1;
    JTextField SearchBox;
    JTextArea Output;

    /**
     * Creates new form DtBeForm
     */
    public DtBeForm() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
}
```

```

@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jTextField1 = new javax.swing.JTextField();
    jButton1 = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTextArea1 = new javax.swing.JTextArea();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jButton1.setLabel("Search");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });

    jTextArea1.setColumns(20);
    jTextArea1.setRows(5);
    jScrollPane1.setViewportView(jTextArea1);

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(148, 148, 148)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
71, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(198, Short.MAX_VALUE)
            )
            .addGroup(layout.createSequentialGroup()
                .addGap(29, 29, 29)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 785,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(14, Short.MAX_VALUE)
            )
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(33, 33, 33)

```

```

        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 295,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 26,
Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE
)
        .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jButton1))
        .addGap(107, 107, 107))
);

pack();
} // </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    Connection conn = null;
    Statement stmt = null;
    String rnum = jTextField1.getText();

    try {
        Class.forName(JDBC_DRIVER);

        jTextArea1.setText("Connecting to database...");
        conn = DriverManager.getConnection(DB_URL, USER, PASS);

        jTextArea1.append("\n Creating statement...");
        stmt = conn.createStatement();
        String sql;
        sql = "Select r.PhotoLocation from Room r where r.roomNo='"+rnum+"'";

        ResultSet rs = stmt.executeQuery(sql);
        while (rs.next()) {
            // Retrieve by column name
            String pl = rs.getString("PhotoLocation");
            JFrame frame = new JFrame(); //JFrame
            frame.setTitle("Add Image"); //Add the title to frame
            frame.setLayout(null); //Terminates default flow layout
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Terminate
program on close button
            frame.setBounds(825, 1, 400, 400); //Sets the position of the frame
            Container c = frame.getContentPane(); //Gets the content layer

```

```

JLabel label = new JLabel(); //JLabel Creation
/*label.setIcon(new ImageIcon(pl)); */
label.setIcon(new ImageIcon(new
ImageIcon(pl).getImage().getScaledInstance(300, 300,
Image.SCALE_DEFAULT))); //Sets the image to be displayed as an icon
Dimension size = label.getPreferredSize(); //Gets the size of the image
label.setBounds(50, 30, size.width, size.height); //Sets the location of the
image
c.add(label); //Adds objects to the container
frame.setVisible(true); // Exhibit the frame
}
rs.close();
stmt.close();
conn.close();
} catch (SQLException se) {
    se.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (stmt != null)
            stmt.close();
    } catch (SQLException se2) {
    }
    try {
        if (conn != null)
            conn.close();
    } catch (SQLException se) {
        se.printStackTrace();
    }
}

try {
    Class.forName(JDBC_DRIVER);

    jTextArea1.setText("Connecting to database...");
    conn = DriverManager.getConnection(DB_URL, USER, PASS);

    jTextArea1.append("\n Creating statement...");
    stmt = conn.createStatement();
    String sql;
    sql = "Select r.QR from Room r where r.roomNo='"+rnum+"'";

    ResultSet rs = stmt.executeQuery(sql);
    while (rs.next()) {
        // Retrieve by column name
        String qr = rs.getString("QR");
        JFrame frame = new JFrame(); //JFrame

```

```

        frame.setTitle("Add Image"); //Add the title to frame
        frame.setLayout(null); //Terminates default flow layout
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Terminate
program on close button
        frame.setBounds(825, 400, 400, 400); //Sets the position of the frame
        Container c = frame.getContentPane(); //Gets the content layer
        JLabel label = new JLabel(); //JLabel Creation
        /*label.setIcon(new ImageIcon(pl)); */
        label.setIcon(new ImageIcon(new
ImageIcon(qr.getImage().getScaledInstance(300, 300,
Image.SCALE_DEFAULT)));//Sets the image to be displayed as an icon
        Dimension size = label.getPreferredSize(); //Gets the size of the image
        label.setBounds(50, 30, size.width, size.height); //Sets the location of the
image
        c.add(label); //Adds objects to the container
        frame.setVisible(true); // Exhibit the frame
    }
    rs.close();
    stmt.close();
    conn.close();
} catch (SQLException se) {
    se.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (stmt != null)
            stmt.close();
    } catch (SQLException se2) {
    }
    try {
        if (conn != null)
            conn.close();
    } catch (SQLException se) {
        se.printStackTrace();
    }
}

try {
    Class.forName(JDBC_DRIVER);

    jTextArea1.setText("Connecting to database...");
    conn = DriverManager.getConnection(DB_URL, USER, PASS);

    jTextArea1.append("\n Creating statement...");
    stmt = conn.createStatement();
    String sql;

```

```

        sql = "Select r.RoomNo, r.capacity, rt.TypeName, d.DpName from Room r,
Roomtype rt, Department d where r.typeid=rt.typeid and r.depid=d.depid and
r.roomNo='"+rnum+"';";

```

```

        ResultSet rs = stmt.executeQuery(sql);
        while (rs.next()) {
            // Retrieve by column name
            String id = rs.getString("RoomNo");
            int cap = rs.getInt("Capacity");
            String tname = rs.getString("TypeName");
            String dp = rs.getString("DpName");
            jTextArea1.append("\n Room Description: " + id + " " + Integer.toString
(cap) + " " + tname + " " + dp + "\n");
        }
        rs.close();
        stmt.close();
        conn.close();
    } catch (SQLException se) {
        jTextArea1.setText("SQL ERROR");
        se.printStackTrace();
    } catch (Exception e) {
        jTextArea1.setText("EXCEPTION");
        e.printStackTrace();
    } finally {
        try {
            if (stmt != null)
                stmt.close();
        } catch (SQLException se2) {
            jTextArea1.setText("NULL ERROR");
        }
        try {
            if (conn != null)
                conn.close();
        } catch (SQLException se) {
            jTextArea1.setText("CONNECTION ERROR");
            se.printStackTrace();
        }
    }

    try {
        Class.forName(JDBC_DRIVER);

        conn = DriverManager.getConnection(DB_URL, USER, PASS);

        stmt = conn.createStatement();

```

```
String sql = "Select h.hardwareDesc, hr.Quantity, hr.datecheck from hardware
h, hardwareInRoom hr where h.hardwareid=hr.hardwareid and
hr.roomNo='"+rnum+"'";
```

```
ResultSet rs = stmt.executeQuery(sql);
while (rs.next()) {
    // Retrieve by column name
    String hd = rs.getString("HardwareDesc");
    int quan = rs.getInt("Quantity");
    String hdate = rs.getString("DateCheck");
    jTextArea1.append("\n Hardware Description: " +hd+" "+
Integer.toString(quan) + " " + hdate + "\n");
}
rs.close();
stmt.close();
conn.close();
} catch (SQLException se) {
    se.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (stmt != null)
            stmt.close();
    } catch (SQLException se2) {
    }
    try {
        if (conn != null)
            conn.close();
    } catch (SQLException se) {
        se.printStackTrace();
    }
}
}
try {
    if (conn != null)
        conn.close();
} catch (SQLException se) {
    se.printStackTrace();
}

try {
    Class.forName(JDBC_DRIVER);

    conn = DriverManager.getConnection(DB_URL, USER, PASS);

    stmt = conn.createStatement();
```

```
String sql = "Select s.softwarename, sr.datecheck from software s,  
softwareInRoom sr where s.softwareid=sr.softwareid and sr.roomNo='"+rnum+"'";
```

```
ResultSet rs = stmt.executeQuery(sql);  
while (rs.next()) {  
    // Retrieve by column name  
    String sd = rs.getString("SoftwareName");  
    String sdate = rs.getString("DateCheck");  
    jTextArea1.append("\n Software Description: " +sd+" " + sdate + "\n");  
}  
rs.close();  
stmt.close();  
conn.close();  
} catch (SQLException se) {  
    se.printStackTrace();  
} catch (Exception e) {  
    e.printStackTrace();  
} finally {  
    try {  
        if (stmt != null)  
            stmt.close();  
    } catch (SQLException se2) {  
    }  
    try {  
        if (conn != null)  
            conn.close();  
    } catch (SQLException se) {  
        se.printStackTrace();  
    }  
}
```

```
try {  
    Class.forName(JDBC_DRIVER);
```

```
conn = DriverManager.getConnection(DB_URL, USER, PASS);
```

```
stmt = conn.createStatement();
```

```
String sql = "Select ru.sessid, ru.usagedate, ru.usagetime, ru.duration,  
s.SessDesc, l.firstname,l.lastname, d.dpname from roomusage ru, sessions s,  
lecturer l, department d where ru.sessid=s.sessid and s.staffID=l.staffid and l.depid  
= d.depid and ru.roomNo='"+rnum+"'";
```

```
ResultSet rs = stmt.executeQuery(sql);  
while (rs.next()) {
```

```

        // Retrieve by column name
        int sid = rs.getInt("SessID");
        String udate = rs.getString("UsageDate");
        String utime = rs.getString("UsageTime");
        int dur = rs.getInt("Duration");
        String sdesc = rs.getString("SessDesc");
        String fname = rs.getString("FirstName");
        String lname = rs.getString("LastName");
        String dep = rs.getString("DpName");

        jTextArea1.append("\n Session Time: " + Integer.toString(sid) + " " +
        udate + " " + utime + " " + Integer.toString(dur) + "\n");
        jTextArea1.append("\n Session Description: " + sdesc + " " + fname + " "
+ lname + " " + dep + "\n");
    }
    rs.close();
    stmt.close();
    conn.close();
} catch (SQLException se) {
    se.printStackTrace();
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (stmt != null)
            stmt.close();
    } catch (SQLException se2) {
    }
    try {
        if (conn != null)
            conn.close();
    } catch (SQLException se) {
        se.printStackTrace();
    }
}

}

} }

}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">

```

```

        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
        * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
        */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(DtBeForm.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(DtBeForm.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(DtBeForm.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(DtBeForm.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
        }
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new DtBeForm().setVisible(true);
        }
    });
}
/*
// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JTextField jTextField1;
// End of variables declaration

*/

```

}