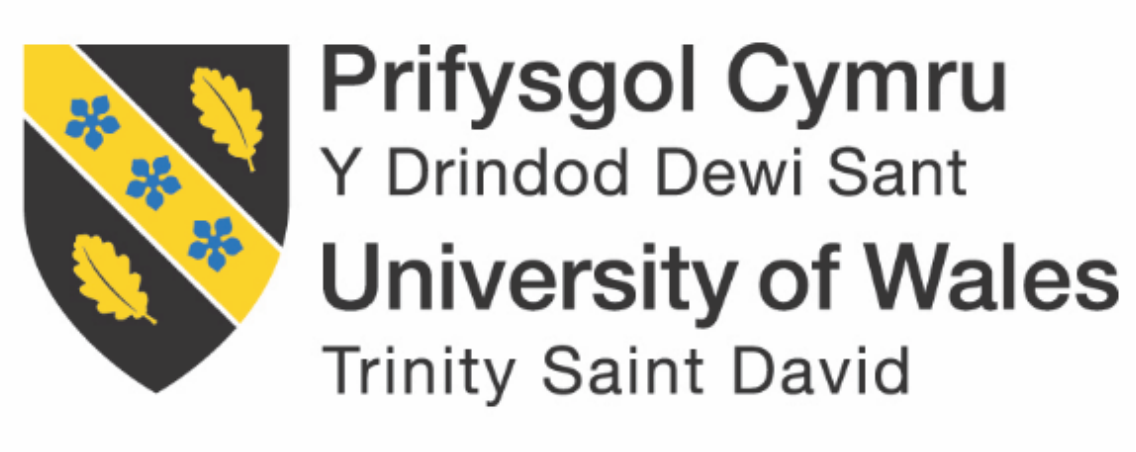


To Research, Design and Implement a Multi Modal Educational Application That Incorporates Interactive Learning Elements.

By

Kadmin John Van Vuuren

Supervisor: Dr Tim Bashford

Project Submitted in Part of the Requirements for the Award of Master of Science: Software Engineering (Web Development)

January 2020

Table of Contents

TABLE OF CONTENTS - 2 -

TABLE OF FIGURES - 8 -

ABSTRACT - 10 -

ACKNOWLEDGEMENTS - 11 -

1. CHAPTER ONE - 12 -

1.1 INTRODUCTION - 12 -

1.1.1 BACKGROUND - 12 -

1.1.2 RESEARCH FOCUS - 14 -

1.1.3 AIMS AND OBJECTIVES - 15 -

1.1.4 VALUE OF RESEARCH - 16 -

1.1.5 OUTLINE STRUCTURE - 17 -

CHAPTER 1: INTRODUCTION - 17 -
CHAPTER 2: REVIEW OF LITERATURE - 17 -
CHAPTER 3: RESEARCH METHODS - 17 -
CHAPTER 4: DESIGN - 17 -
CHAPTER 5: IMPLEMENTATION - 18 -
CHAPTER 6: CONCLUSION - 18 -
CHAPTER 7: REFERENCING - 18 -

2. CHAPTER TWO - 19 -

2.1 REVIEW OF LITERATURE - 19 -

2.1.1 E-LEARNING | AN INTRODUCTION - 19 -

2.1.1.1 E-LEARNING | BRIEF HISTORY - 19 -

2.1.1.2 E-LEARNING | TODAY'S LANDSCAPE **- 20 -**

2.1.2 LEARNING STYLES | AN INTRODUCTION **- 21 -**

2.1.2.1 LEARNING PREFERENCES & STYLES | VARK **- 21 -**

2.1.2.2 LEARNING STYLES | KOLB'S EXPERIENTIAL LEARNING CYCLE **- 24 -**

2.1.3 ECO SYSTEMS | AN INTRODUCTION **- 26 -**

2.1.3.1 ECO SYSTEMS | POTENTIAL PLATFORMS TO TARGET **- 26 -**

2.1.4 SURVEY DATA | AN INTRODUCTION **- 28 -**

2.1.4.1 SURVEY DATA | COLLECTED DATA **- 29 -**

2.1.5 DEVELOPMENT METHODS & TECHNIQUES | AN INTRODUCTION **- 29 -**

2.1.5.1 DEVELOPMENT METHODS & TECHNIQUES | FRAMEWORKS **- 29 -**

2.1.5.2 DEVELOPMENT METHODS & TECHNIQUES | PROJECT APPROACH **- 35 -**

2.1.6 LEARNING MATERIALS | CONSIDERATION **- 37 -**

2.1.7 PROGRAMMING LANGUAGES | AN INTRODUCTION **- 38 -**

2.1.7.1 PROGRAMMING LANGUAGES | JAVA **- 38 -**

2.1.7.2 PROGRAMMING LANGUAGES | C# **- 41 -**

2.1.7.3 PROGRAMMING LANGUAGES | JAVASCRIPT **- 41 -**

2.1.8 UML | AN INTRODUCTION **- 42 -**

2.1.8.1 UML | STRUCTURAL DIAGRAMS **- 42 -**

2.1.8.2 UML | BEHAVIOURAL DIAGRAMS **- 43 -**

2.1.9 SYSTEMS | AN INTRODUCTION **- 43 -**

2.1.9.1 SYSTEMS | WHAT IS A GOOD SYSTEM? **- 44 -**

2.1.9.2 SYSTEMS | PROGRAMMATICALLY GOOD SYSTEM CONSIDERATIONS **- 44 -**

2.1.9.3 SYSTEMS | SO, HOW ARE GOOD SYSTEMS BUILT? **- 45 -**

2.1.9.4 SYSTEMS | LOW COUPLING & HIGH COHESION **- 45 -**

2.1.10 DEVELOPMENT METHODOLOGY | AN INTRODUCTION **- 47 -**

2.1.10.1 DEVELOPMENT METHODOLOGIES | TDD **- 47 -**

2.1.11 DATABASES | AN INTRODUCTION **- 48 -**

2.1.11.1 DATABASES | DATABASE LIFECYCLE **- 48 -**

2.1.11.2 DATABASES | RELATIONAL & NOSQL **- 49 -**

2.1.11.3 DATABASES | SQLITE **- 50 -**

2.1.12 APPLICATION DESIGN | AN INTRODUCTION **- 50 -**

2.1.12.1 APPLICATION DESIGN | JAVAFX **- 50 -**

2.1.12.2 APPLICATION DESIGN | SWING **- 52 -**

2.1.13 DESIGN PATTERNS | CONSIDERATION **- 52 -**

2.1.13.1 DESIGN PATTERNS | SINGLETON PATTERN **- 52 -**

2.1.14 FRAMEWORKS | CONSIDERATION **- 53 -**

2.1.14.1 FRAMEWORKS | MAVEN **- 53 -**

2.1.15 TESTING | CONSIDERATION **- 53 -**

3. CHAPTER THREE **- 55 -**

3.1 RESEARCH METHODS **- 55 -**

3.1.1 RESEARCH STRATEGY | AN INTRODUCTION **- 55 -**

3.1.1.1 RESEARCH STRATEGY | PHILOSOPHIES **- 55 -**

3.1.1.2 RESEARCH STRATEGY | CASE STUDY **- 58 -**

3.1.1.3 RESEARCH STRATEGY | STUDY POPULATION SAMPLE **- 58 -**

3.1.1.4 RESEARCH STRATEGY | DATA COLLECTION METHODS **- 58 -**

3.1.1.5 RESEARCH STRATEGY | RELIABILITY AND VALIDITY **- 59 -**

3.1.1.6 RESEARCH STRATEGY | OBSERVATIONS **- 60 -**

4. CHAPTER FOUR **- 61 -**

4.1 DESIGN AND IMPLEMENTATION **- 61 -**

4.1.1 APPLICATION DESIGN | AN INTRODUCTION **- 61 -**

4.1.1.1 APPLICATION DESIGN | JAVA | JAVAFX | SCENE BUILDER **- 61 -**

4.1.1.2 APPLICATION DESIGN | IDE **- 62 -**

4.1.1.3 APPLICATION DESIGN | BUILDING A PROTOTYPE **- 63 -**

4.1.2 IMPLEMENTING APPLICATION DESIGN | AN INTRODUCTION **- 66 -**

4.1.2.1 IMPLEMENTATION APPLICATION DESIGN | DASHBOARD **- 66 -**

4.1.2.2 IMPLEMENTATION APPLICATION DESIGN | LOGIN REGISTER SPLASH **- 68 -**

4.1.2.3 IMPLEMENTATION APPLICATION DESIGN | ADMIN DASHBOARD **- 69 -**

4.1.2.4 IMPLEMENTATION APPLICATION DESIGN | ADMIN EDITOR **- 69 -**

4.1.3 SYSTEM DESIGN & IMPLEMENTATION | AN INTRODUCTION **- 71 -**

4.1.3.1 SYSTEM DESIGN | USER STORIES **- 72 -**

4.1.3.2 SYSTEM DESIGN | UML **- 72 -**

4.1.3.2 SYSTEM DESIGN | ACTIVITY DIAGRAMS **- 73 -**

4.1.3.3 SYSTEM DESIGN | NOTABLE SYSTEM IMPLEMENTATIONS **- 73 -**

4.1.3.4 SYSTEM DESIGN | MODEL VIEW CONTROLLER **- 74 -**

4.1.3.5 SYSTEM DESIGN | MODELS **- 75 -**

4.1.4 DATABASE DESIGN | AN INTRODUCTION **- 75 -**

4.1.4.1 DATABASE DESIGN & IMPLEMENTATION | ENTITY RELATIONSHIP DIAGRAM **- 76 -**

4.1.5 IMPLEMENTING SINGLETON PATTERN | AN INTRODUCTION **- 77 -**

4.1.6 PROGRAMMING LANGUAGES | HOW LANGUAGES ARE IMPLEMENTED **- 78 -**

5. CHAPTER FIVE **- 80 -**

5.1 RESULTS SUMMARY **- 80 -**

5.1.1 RESULTS SUMMARY | USER FEEDBACK **- 80 -**

5.1.1.1 RESULTS SUMMARY | HAVE YOU USED A SIMILAR TOOL **- 80 -**

5.1.1.2 RESULTS SUMMARY | BACKGROUND IN I.T. **- 81 -**

5.1.1.3 RESULTS SUMMARY | HOW WOULD YOU RATE THIS TOOL **- 82 -**

5.1.1.4 RESULTS SUMMARY | MULTI-MODAL IMPORTANCE **- 83 -**

5.2 CONCLUSION **- 84 -**

6. CHAPTER SIX **- 85 -**

6.1 FUTURE CONSIDERATIONS **- 85 -**

6.1.1 FUTURE CONSIDERATIONS | SOCIAL MEDIA **- 85 -**

6.1.2 FUTURE CONSIDERATIONS | DASHBOARD **- 85 -**

6.1.3 FUTURE CONSIDERATIONS | EXAMS **- 86 -**

6.1.4 FUTURE CONSIDERATIONS | WEB COMPONENT **- 86 -**

6.1.5 FUTURE CONSIDERATIONS | USER INPUT **- 86 -**

7. CHAPTER SEVEN - 88 -

7.1 REFLECTION - 88 -

7.1.1 REFLECTION | POSITIVES | WHAT WENT WELL - 88 -

7.1.1.1 REFLECTION | JAVAFX | FXML | SCENEBUILDER - 88 -

7.1.1.2 REFLECTION | AGILE PRINCIPLES - 88 -

7.1.2 REFLECTION | NEGATIVES | WHAT DIDN'T GO WELL - 89 -

7.1.2.1 REFLECTION | TIME | PLANNING - 89 -

8. CHAPTER EIGHT - 90 -

8.1 REFERENCE LIST - 90 -

READING LIST - 96 -

APPENDIX - 97 -

USER STORIES - 97 -

SINGLETON - 105 -

ACTIVITY DIAGRAMS - 106 -

SEARCHING - 107 -

MVC - 108 -

ADMIN EDITOR - 109 -

USER FEEDBACK - 111 -

REVIEWER NOTES - 116 -

Table of Figures

FIGURE 1: VARK PREFERENCES **[4]** - 23 -
FIGURE 2: KOLB'S EXPERIENTIAL LEARNING CYCLE **[26]** - 24 -
FIGURE 3: LEWINIAN EXPERIENTIAL LEARNING MODEL **[20]** - 25 -
FIGURE 4: USER PREFERRED ECOSYSTEM - 29 -
FIGURE 5: SDLC **[15]** - 31 -
FIGURE 6: WATERFALL SDLC **[19]** - 31 -
FIGURE 7: WATERFALL | ITERATIVE LIFECYCLE **[28]** - 32 -
FIGURE 8: JAVA COMPILER INTERPRETER **[32]** - 40 -
FIGURE 9: SUMMARISE COHESION & COUPLING **[38]** - 46 -
FIGURE 10: COUPLING & COHESION **[39]** - 46 -
FIGURE 11: JAVAFX ARCHITECTURE DIAGRAM **[44]** - 51 -
FIGURE 12: RESEARCH UNION - 56 -
FIGURE 13: POSITIVISM - 57 -
FIGURE 14: POSITIVISM **[8]** - 57 -
FIGURE 15: UNDERSTANDING SCENE BUILDER - 63 -
FIGURE 16: SCENE BUILDER UI PROTOTYPE - 64 -
FIGURE 17: ANCHORING NODES - 64 -
FIGURE 18: ANCHORPANE NODE RESIZING - 65 -
FIGURE 19: DASHBOARD - 66 -
FIGURE 20: LOGIN REGISTER SPLASH - 68 -
FIGURE 21: ADMIN DASHBOARD - 69 -
FIGURE 22: ADMIN EDITOR - 69 -
FIGURE 23: CONTROLLER CLASS SCENEBUILDER - 70 -
FIGURE 24: CONTROLLER CLASS - 70 -
FIGURE 25: CONTENT CONSIDERATIONS - 71 -
FIGURE 26: ERD - 76 -
FIGURE 27:SQLITE DBBROWSER PROJECT DB - 76 -
FIGURE 28: DATABASE SINGLETON - 78 -
FIGURE 29: SURVEY - 81 -
FIGURE 30: SURVEY - 81 -
FIGURE 31: SURVEY - 82 -
FIGURE 32: CENTRAL TENDENCY - 82 -
FIGURE 33: SURVEY - 83 -
FIGURE 34: FUTURE CONSIDERATIONS - 87 -
FIGURE 35: APPENDIX USER STORY BOARD - 97 -
FIGURE 36: APPENDIX USER INTERFACE EPIC STORY - 98 -
FIGURE 37: APPENDIX USER INTERFACE EPIC STORY - 98 -

FIGURE 38: APPENDIX USER INTERFACE EPIC STORY - 99 -
FIGURE 39: APPENDIX USER INTERFACE EPIC STORY - 99 -
FIGURE 40: APPENDIX USER INTERFACE EPIC STORY - 100 -
FIGURE 41: APPENDIX USER LOGIN | REGISTRATION EPIC STORY - 100 -
FIGURE 42: APPENDIX USER LOGIN | REGISTRATION EPIC STORY - 101 -
FIGURE 43: APPENDIX USER LOGIN | REGISTRATION EPIC STORY - 101 -
FIGURE 44: APPENDIX CONTENT CREATOR EPIC STORY - 102 -
FIGURE 45: APPENDIX CONTENT CREATOR EPIC STORY - 102 -
FIGURE 46: APPENDIX CONTENT CREATOR EPIC STORY - 103 -
FIGURE 47: APPENDIX VIEWABLE CONTENT EPIC STORY - 103 -
FIGURE 48: APPENDIX VIEWABLE CONTENT EPIC STORY - 104 -
FIGURE 49: APPENDIX DATABASE EPIC STORY - 104 -
FIGURE 50: APPENDIX SINGLETON USER SESSION - 105 -
FIGURE 51: APPENDIX SINGLETON DATABASE - 105 -
FIGURE 52: APPENDIX ACTIVITY DIAGRAM ALERTING - 106 -
FIGURE 53: APPENDIX ACTIVITY DIAGRAM USER REGISTRATION - 106 -
FIGURE 54: SEARCHING CONTENT - 107 -
FIGURE 55: APPENDIX MVC - 108 -
FIGURE 56: APPENDIX ADMIN EDITOR IMAGE BUTTON - 109 -
FIGURE 57: APPENDIX ADMIN EDITOR CONTENT EXTRACTION - 109 -
FIGURE 58: APPENDIX ADMIN EDITOR FILE ENCODING - 110 -
FIGURE 59: APPENDIX USER FEEDBACK - 111 -
FIGURE 60: APPENDIX USER FEEDBACK - 111 -
FIGURE 61: APPENDIX USER FEEDBACK - 112 -
FIGURE 62: APPENDIX USER FEEDBACK - 112 -
FIGURE 63: APPENDIX USER FEEDBACK - 113 -
FIGURE 64: APPENDIX USER FEEDBACK - 113 -
FIGURE 65: APPENDIX USER FEEDBACK - 114 -
FIGURE 66: APPENDIX USER FEEDBACK - 114 -
FIGURE 67: APPENDIX USER FEEDBACK - 115 -
FIGURE 68: APPENDIX USER FEEDBACK - 115 -

Abstract

With information so readily available to connected users it can be overwhelming when looking to find the right information to educate one's self. Ensuring that this information is presented in ways that benefit all is important when looking to make the most out of one's journey through a topic in which they wish to educate themselves.

This paper looks to understand how as a people, we learn, and more importantly the avenues in which educational material can be presented to ensure the proper learning of said topics. Once this understanding is established, this paper will also look to conduct further research and investigation into the methods and processes one can undertake to develop educational tools such as E-Learning, to provide a multi-modal learning experience through the use of information technology.

This Section Left Intentionally Blank

Acknowledgements

“To my supervisor Dr Tim Bashford, thank you for your ongoing support.”

“To my colleague, friend and mentor Ashley Eatly, thank you for your ongoing support and help throughout my time in Viper Squad.”

1. Chapter One

1.1 Introduction

This section of the report will look to present an introduction to the proposed project. Literature suggests that the structure to a project's introduction should comprise of the following parts:

- Introduction
    - Background
    - Research Focus
    - Overall Research Aim and Individual Research Objectives
    - Value of this Research

All of the above points are important to the proposed project and as such will look to be explored and to provide a detailed understanding of what this project intends to achieve and why **[1]**.

1.1.1 Background

Education plays an important part in developing one's self. How educational material is presented and the medium through which it is deployed, is arguably just as important as the material itself. As such, there is a large area of research that can be undertaken to understand how users learn best and the process by which learning tools can be created to satisfy any educational needs.

Learning today, has had its focus taken away from the classroom, and into almost any environment the learner chooses; whether it be learning whilst commuting, or learning whilst at home or at the workplace, the choice is put into the hands of the learner. By allowing users who seek to consume educational material and further educate themselves

through a means by which we remove the need for classroom-based education, we allow them to study at times and places which ultimately suit them.

Development of eLearning is connected to the development, technical improvement and also better affordability of computers **[3]**. It's important for us to understand some of the more important parts of history surrounding e-learning: If one is to understand the need behind learning tools and what drove their early stage development, an understanding of at least some of its history will play an important part in this projects proposal and understanding of what it hopes to achieve.

This presents a huge challenge to that of those responsible for making these tools. An understanding of what a learner requires and expects from an educational application is essential in ensuring that they get the most out of their experience with the educational tool. This is true for all aspects of an application, especially that of an educational application, from the layout and colour scheme to the flow through the material presented by the application and even through what medium this content is presented; such as audio, text and interactivity. Considerations must be given to every aspect of an application of this nature.

The creation of an application that suits the needs of its learners, presents to us, a huge research area that needs to explored and understood; only then can we be confident in knowing that we have built a reliable, suitable, extensible and flexible application that caters for the needs of all types of learners. This consideration brings us suitably onto another research point of interest; learners, regardless of background, learn differently, and what suits one may not suit another. As such, one can hypothesize that many learning techniques exist, and if so, what are they and how can we use these techniques to help present information to our intended audience.

This project will look to understand these areas of education and how from this understanding we can build a tool that is suitable for many types of learner, regardless of

their learning preference. In order to achieve this, research needs to be undertaken into certain fields of education as well as many fields surrounding information technology; such as tools and techniques used in building an application that can be accessed by many who intend to further their learning within a particular field.

With this in mind, it is important to point out that the main objective of this project is to produce a software deliverable that has taken research topics into consideration, to ensure the best possible learning experience through a technological platform. As such, we should now introduce the concept of E-Learning; a concept that is widely debated as to its meaning, however, for the context of this project, will mean the use of software and information technology techniques and technologies to enable further education to that of the learner. However, regardless of its meaning, this a key component of this project is that of E-Learning, what it is, how it can be used as an educational platform and the processes involved in developing such a platform.

Furthermore, we should also introduce the projects intention to refine that of an educational application, which may have considerable range, in terms of its meaning and implementation, to that of a Learning Management System or LMS. Although the tool will not look to contend with the likes of systems such as Moodle, it is intended to create a framework that further supports the addition, removing and deleting of learning material. As such, this project will look to conclude that its Minimal Viable Product or MVP will be that of a software framework that allows for the addition and removal of educational material, akin to that of an LMS.

1.1.2 Research Focus

The above presents a background to the proposed project. From this, we can comprise a list of what research focus topics this project intends to explore: Something that is also suggested for coverage by **[1]**. The research focus of this project is that of E-Learning, this

topic can then be further broken down into sub research focus groups such as Learning Modalities or how the concept of how individuals prefer to learn. Furthermore, E-Learning in the scope of the proposed application, will form that of a learning tool that can be accessed through computing device such as a Computer or Mobile and due to the nature of wanting to create a Learning Management System, as pointed out in the project background, research will need to be conducted into that of Software Engineering and the SDLC or Software Development Lifecycle as a whole, as well as the other processes involved throughout software engineering projects.

Finally, the project will look to conclude by collecting data and as such another important research topic will be that of Data Gathering techniques and how to present this data.

1.1.3 Aims and Objectives

Before any project can start, a set of aims and objectives need to be identified to help understand what needs to be undertaken for the proposed project as well as provide clear goals and expectations for what the project hopes to achieve. Ultimately this project intends to develop, deploy and present an application that educates users on a range of technological topics. However, it is important to point out that although an application with educational content is desired, an educational framework will look to be created as a base MVP or Minimal Viable Product. This will allow educational materials to be added, updated and/or removed as and when required.

The above aims can be broken down into an achievable set of objectives, such as:

- Investigation of learning modalities
- Investigation into the process involved throughout a Software Engineering project
- To use the above to design and implement an Educational Application

- Finally, to provide evaluation and analysis for the proposed application as well as provide recommendations for future enhancements to the application.

Once concluded, an application should exist that can present educational material to its users. As previously suggested, this material will encompass that of various technologies, with particular emphasis on that of technologies used at the DVLA. Importantly, and as previously suggested, an MVP will look to be achieved that will at least allow for the adding, removing and editing of educational material.

Additionally, the project will look to produce its own set of statistics which illustrate how best users learn and if they feel like the application supports their learning needs. This will further support the hypothesis that individuals learn differently and that in fact, it is best to support multiple learning modalities.

1.1.4 Value of Research

Primarily this research undertaking intends to produce an understanding in the process involved throughout a Software Engineering project. This adds value to the field as it will hopefully present the steps involved throughout a project which produces an application. Much research exists in this field however, it is expected that the research undertaken as part of this project and how this research may be applied to produce an end deliverable will help to consolidate the vast amount of materials that exist separately surrounding this topic into one deliverable.

Furthermore, it is hoped that value will be found in the projects own research statistics, which look to present further evidence of multi modal learning and the idea that material needs to be presented in many different formats in order to completely satisfy a learner's educational needs.

1.1.5 Outline Structure

It is suggested that the presentation of an Outline Structure may aid that of the reader **[1]**. As such this section will look to highlight the Chapters expected throughout this paper and provide a description of the considerations made and purposes of each.

Chapter 1: Introduction

This chapter will help to provide the reader with an insight as to the reasoning behind the project, the research that will look to be undertaken and what the project intends to present. By providing this, it is hoped that the reader will fully understand the projects intention and ultimately

Chapter 2: Review of Literature

This chapter will look to present a review of the literature surrounding the projects proposed topics. This review of literature will span from reviews of topics such as E-Learning to the reviewing of more technical literature such as programming techniques. Furthermore, justification for why certain decisions were made will be present throughout this chapter.

Chapter 3: Research Methods

This chapter will look to discuss and justify the research strategy and data collection techniques that will be used as part of the project. Discussions around the potential problems of said research methods and data collection will also be highlighted so to minimise impact on findings will also be discussed.

Chapter 4: Design

This chapter will look to provide an overview and breakdown the design considerations made whilst using research presented for the application.

Chapter 5: Implementation

This chapter will look to show how previous chapter considerations and works have been implemented. This will allow the author to illustrate how all works up until this point have been considered and implemented within the proposed application.

Chapter 6: Conclusion

Finally, a conclusion chapter will provide concluding points and present an overview of works completed whilst presenting data collected during the projects lifecycle and finally a review of this data and its importance to the project.

Chapter 7: Referencing

This chapter will look to provide the sources used throughout the project. Furthermore, these references will be presented using IEEE referencing standards.

This Section Left Intentionally Blank

2. Chapter Two

2.1 Review of Literature

This section of the report will look to present researched literature surrounding the project.

2.1.1 **E-Learning** | An Introduction

E-Learning is a term used to describe learning over some form of electronic platform; whether it be over the internet and browser based or through an application of some sort, regardless of device. It is otherwise described as, the marshalling of computer and network technologies to the task of education **[1]**. Furthermore, we can see the suggestion of E-Learning holding several definitions, such as, several definitions of E-Learning are common and that some people maintain that it is entirely based upon learning within a Web browser, without the need for other software **[1]**.

2.1.1.1 **E-Learning** | Brief History

We can see that during the late eighties and into the nineties of the last century Computer-Based Training or CBT was starting to take shape and it is this that is considered to be the cornerstone of e-learning **[2]**. It appears that this CBT was, of course, CD based multimedia and much like we can today; learning was capable of leaving the classroom and entering a learner's private space. It's interesting to point out, that in parallel with the development of CBT was the technology developed, that at the end lead into the rise of internet and creation of the web system **[2]**. Importantly, we also see other technologies be created for the above, such as the web browser and a means by which we can display content on computer screens. As such, this trend soon became popular among

vast amounts of people across the world, allowing it to be an affordable interest for many of the population.

The above really sets a picture for where e-learning stemmed from and how it became such an important part in the education of learners as well as playing more and more of an important role in education since.

2.1.1.2 **E-Learning |** Today's Landscape

Today, e-learning is a complex system with multiple forms and different level of interest of learner **[5]**. This is important, because until recently and still ongoing, there is debate as to exactly what e-learning is and what it encompasses. For the purposes of this research topic, e-learning will form a means, by which a learner can use multi-modal application, regardless of operating system or system setup, to educate themselves: Furthermore, the application will not be online facing and therefore remove the need for constant internet access.

Another important observation one can make is that many if not most of these e-learning materials reside on the internet and therefore require constant internet connection in order to access the material. There have been suggestions provided through literature that suggest a major problem behind that of e-learning is the application ease of access **[5]**.

Content is readily available today. As such, there is a plethora of applications that exist which provide users with huge amounts of educational content. However, it is important to build an application that encompasses that of many different approaches to presenting material. This is evident as we can see that an article written by a Mr Little, states that after 90 interviews with UK-based executives involved in the e-learning sector, a key issue within that of e-learning is that of the contextualisation of learning materials **[6]**. This is often one problem that plagues that of learning materials, having to piece together information to contextualise something in which a user is trying to understand.

2.1.2 **Learning Styles** | An Introduction

When considering education and how best to present material to learners via E-Learning, we need to understand that learning modalities exist; of course, it is natural to presume that not everyone learns in the same manner and as such, it is natural to assume that multiple modalities exist. Learning modalities are the sensory channels or pathways through which individuals give, receive and store information **[2]**. It's important to outline at this stage, that this project does not look to use any learning models to distinguish how users of the proposed application learn, rather its purpose is to determine that they exist and provide evidence for this. The LMS will look to provide multiple ways of presenting information, and as such needs a basis of why to do so.

There are many learning models that exist which provide an insight into understanding how individuals learn, therefore, this section will look to identify these models and illustrate that many learning models or ways of learning exist.

2.1.2.1 **Learning Preferences & Styles** | VARK

VARK is an acronym made from the initial letters of Visual, Aural, Read/Write and Kinaesthetic **[4]**. These are further described as "Modes" that are used when learning, whether this be taking in the information or providing it out.

Importantly Flemming **[4]** suggests that VARK itself is not a learning style, further suggesting that learning styles are amalgamations of learning preferences: VARK is one of the preferences that exist within learning styles. Although this preference is not strictly a learning style, it is still worthy of mention and study within the scope of this project as it accurately describes the modes that are frequently encountered during education.

Furthermore, we can see that Flemming, states that learners have preferences for some modes and not for others **[4]**. We can break the VARK modalities down into the following:

- **Visual (V):** The first modality preference in the VARK model is that of Visual. This preference suggests that the individual leans towards visual information depiction that takes the form of maps, spider diagrams, flow charts, maps, spider diagrams, charts, graphs, flow charts, labelled diagrams, and all the symbolic arrows, circles, hierarchies and other devices, that people use to represent what could have been presented in words **[4]**. This is an important discovery as without proper research one would think that visual represents text-based information, however using the VARK model, we can see this is not the case. The important take way from this, is that we can see Visual as being named Graphic and does actually suggest that it should be called this.
- **Aural Auditory (A):** The second modality preference is a perceptual mode that described a leaning towards that of information that is heard or spoken. Aural or Auditory is described as being leaners who have a preference towards lectures, group discussions and even radio **[4]**. What is apparent here is that there are learners who's preferred style is that of listening through topics.
- **Read Write (R):** The third preference for the intake of information by individuals as suggested by VARK, is that of the Reading or Writing modality. This is information displayed in words and is further described as a preference that emphasizes text-based input and output, such as reports, essays or manuals **[4]**. Here we see again that there is indeed a spectrum for additional material modality to be added to that of the application and if we are to present this information to users of the suggested piece of software, this is another modality that should be included.

- **Kinaesthetic:** The third and final learning modality as suggested by VARK is that of Kinaesthetic. This learning preference is by definition is to be able to experience or practice that subject area. Interestingly, this modality is further expected to invoke other modalities **[4]**. This suggests that any type of learning that is presented in this manner should be accompanied with as many of the other modalities as possible. This project, its research and implementation are a strong example of such a learning style.

VARK and its inclusion in this project will assist the author in presenting evidence that suggests learning material, when presented to its users should be in multiple different forms, further allowing for all preferences to be catered for. We can further see this illustrated as **[4]** produced the following statistics after were collected and produced from over 278,000 participants:

Multiple preferences		Single preferences	
VARK Type Two	22.9%	V	4.0%
VARK Transition	5.1%	A	8.8%
VARK Type One	7.4%	R	9.0%
VRK	2.4%	K	14.2%
VAK	4.1%	Subtotal	36.0%
VAR	1.1%		
ARK	5.2%		
VR	1.2%		
VA	0.8%		
VK	2.9%		
AK	6.2%		
RK	2.5%		
AR	2.2%		
Subtotal	64.0%		

Figure 1: VARK Preferences **[4]**

The above helps to illustrate the difference in learning styles and we see that there are those who have more than one preference and those who have singular preferences. There are also those who are considered to switch between preference, depending on the topic in which they are learning, and these can be seen as VARK Type One and Two.

It is also important to understand that learning preferences and styles do not only apply to the learner but also the educator, who too have preferred ways of presenting material. For the purposes of this project the educator can be seen as the Author and as such it is important to remove one's self from any preference of learning style when implementing multi-modal functionality.

2.1.2.2 **Learning Styles** | Kolb's Experiential Learning Cycle

When thinking about education and the phases or stages one might need to encounter to learn a topic, Kolb's Experiential Learning Style theory gives us a solid idea as to what needs to be considered when implementing our own learning platform. Kolb's theory is represented by a four-stage learning cycle which can be represented as such:

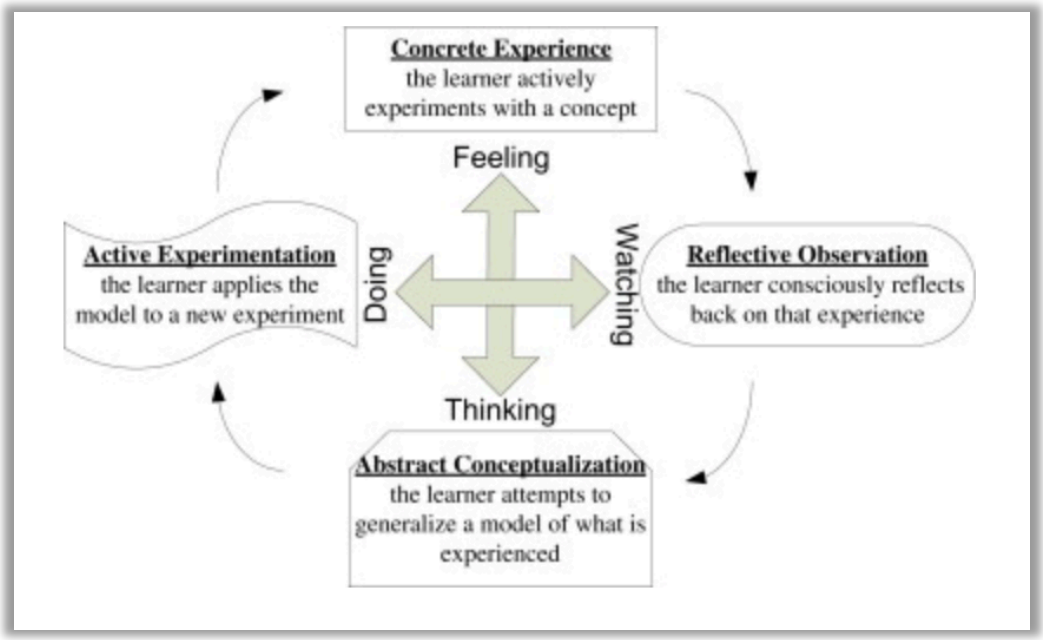


Figure 2: Kolb's Experiential Learning Cycle **[26]**

The above which was influenced by the Lewinian Model and further suggests that the four stages during learning, involve the learner firstly encountering the topic and performing actions based on the topic. They then have a reflective period in which they review the experience and reflect upon it. The learner then goes onto learn from the

experience and concludes upon it; finally, they go on to apply this new learner topic to test what they have learned. It is also important that we understand that the basis behind that of Experiential Learning is that of learning through experience. This draws similarities to that of the Kinaesthetic mode of VARK whereby learners learn best when they perform or practice the topic themselves rather than have the information presented to them through other modalities. We can see how Kolb's model was directly influenced by the Lewinian Model, such as:

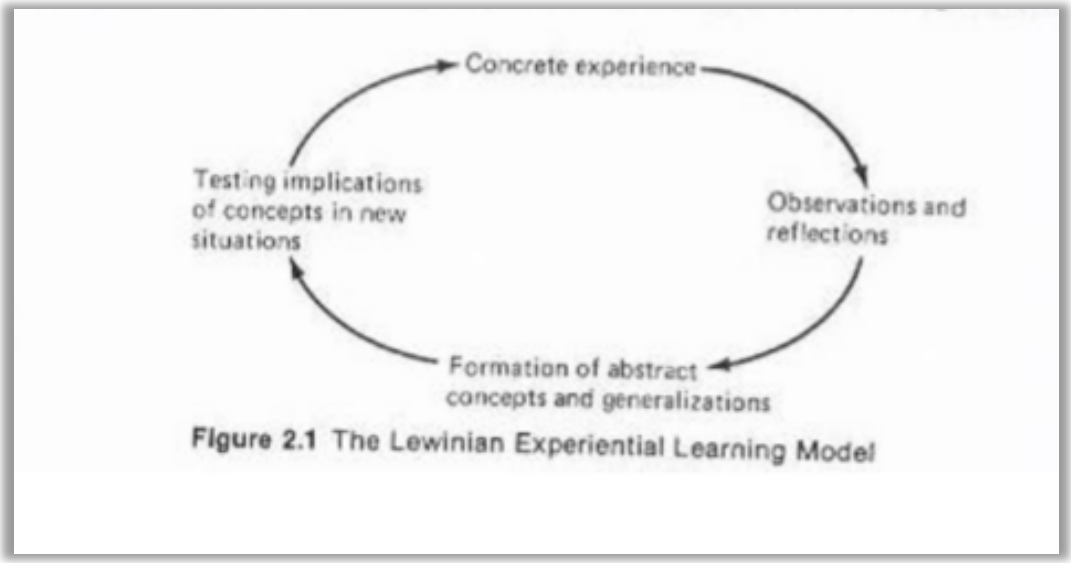


Figure 3: Lewinian Experiential Learning Model **[20]**

This gives an insight into how we can expect to present information to learners, how we can include them in the learning experience and test them on the topic at hand. We see that by following these steps we complete a cycle of learning. Researchers such as Benami **[9]** suggest that we can use the Kolb model in any professional studying matter, further supporting its use and inclusion within this project.

We can summarise this section by highlighting the similarities between learning styles and preferences presented throughout this section. Being able to determine, using multiple sources of research, that similarities exist in both, whereby users can learn by putting into

practice, content that is presented in multiple different ways, is important as it suggests that the proposed application should indeed allow users to absorb information and put this into action, either through interactive features or exam. We can further apply this research use these models to ensure that the educational material presented as part of the proposed application, allows users to cover each one of the steps within the proposed cycles.

2.1.3 **Eco Systems** | An Introduction

Before we can to think about the more technical challenges faced by the proposed project, we need to first, give consideration to the ecosystem we wish to target. Today there exists many types of platforms, all of which of their own uniqueness. This section will look to discover such platforms and decide upon which should be targeted to reach the wider audience.

2.1.3.1 **Eco Systems** | Potential Platforms to Target

Now that we have an understanding of learning styles, the many modalities that exist and E-Learning as a concept, we can begin to try to understand the platforms that exist. LearnToCodeWith suggests that before we can start building an app, you have to decide what kind of application you want to build **[12]**.

- **Desktop:** Desktop apps are arguably one of the most popular choices when looking to target a platform. Desktops are the most natural targeted platform as it arguably offers the richest user experience, allowing developers to the most out of areas of important consideration such as screen real estate. For example, if we are developing interactive features as part of our application, often the user experience of this interactivity may be lost on smaller devices such as mobile devices; however, this is not to say that mobile apps offer no user experience when considering the interactivity of the application. Furthermore, we do have to also consider that there are drawbacks to desktop applications

such as the user having to download applications and run them, as opposed to a web-based environment that allows for instant access and use of features.

Finally, LearnToCodeWith suggests that Because each platform vendor provides their own development tools, creating the user interface of your app will require a different tool for each platform **[12]**. This is interesting as it makes the author wonder about programming languages that cater for multiple platforms, rather than one language for one platform.

- **Web Applications:** As touched upon previously, web applications afford us the positives from a desktop environment, without having to incur any of the negatives. Web apps further allow for us to easily deploy updates for all users, one deployment to a web server in which the application is hosted upon, means all users have access to this. However, one major pitfall of Web applications is that users require stable connections to the internet to be able to access the required material. Although we are in an age where this is widely accessible, there are certainly going to be users who don't have stable connections, who want to carry on using a learning application, so this will need to be considered in detail.
- **Mobile Applications:** Mobile applications offer us a platform that is arguably the most popular or widely used today. **[13]** suggests that in 2019 Apps account for 89% of mobile media time. This shows that mobile consumption is primarily within apps, so developing an app for mobile would be a wise decision when looking to reach a large target audience. Furthermore, statistics show that:

Desktop Owners	Mobile Owners
73%	92%

Table 1: Desktop Vs Mobile

This suggests that there are considerably more mobile phone users than that of desktop users, however it is not clear how many users were surveyed during this data collection.

The above considerations have led to the first realisation that a survey should be put in place to identify what type of application should be created. We can see that there is a lean towards that of mobile based applications, in terms of platform popularity and thus it is reasonable to suggest that the proposed application should be aimed at the mobile ecosystem. However, conducting a survey might provide the data needed to make this decision.

If a desktop application was targeted, we also need to understand that there are numerous display sizes that may be accessing the application. As such it is important to review the most popular screen display sizes.

Screen Size	Popularity %
360x640	14.87%
1920x1080	11.56%
1366x768	11.08%

Table 2: Monitor Size Data [50]

2.1.4 **Survey Data** | An Introduction

Previously the topic of discussion surrounded that of eco systems, and which would be a good suit to the proposed application. Although statistics showed that mobile now drive that of screen time and owner usage, we can collect our own data to solidify our choice. This part will look to produce and illustrate data collected to try and understand how potential users would want to access our content.

2.1.4.1 **Survey Data** | Collected Data

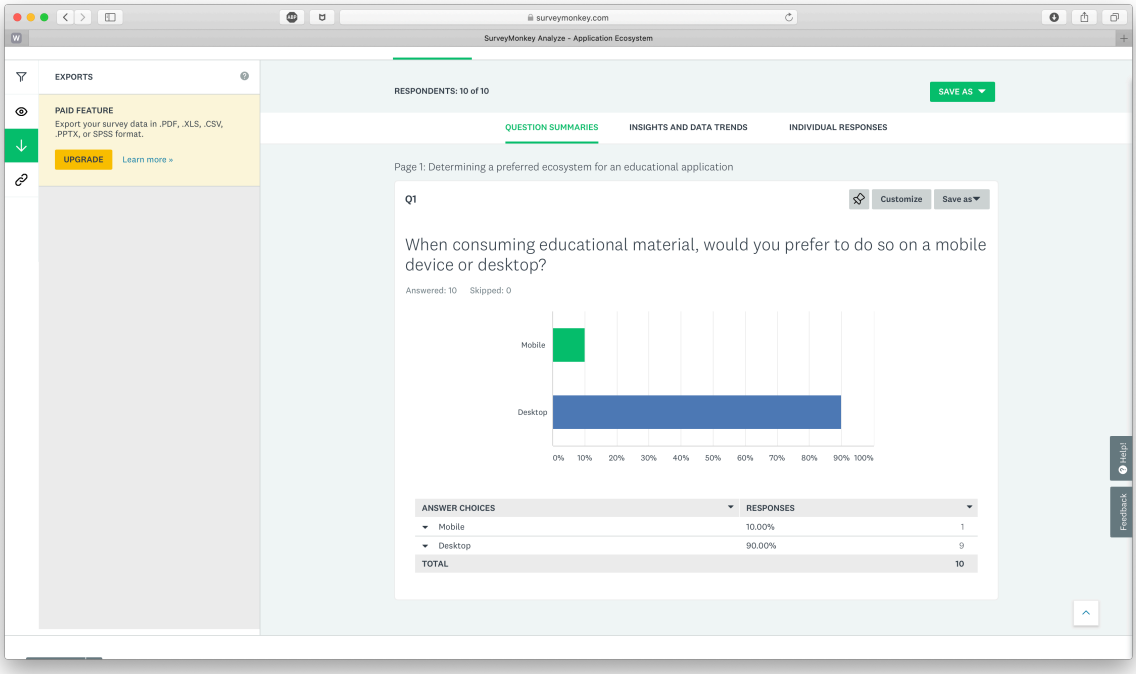


Figure 4: User Preferred Ecosystem

The above illustrates a landslide preference showed towards that of the desktop ecosystem. This is understandable due to the limited screen sizes of mobile devices thus impacting on their learning experience.

2.1.5 **Development Methods & Techniques** | An Introduction

When undertaking such a large project, it is important before we begin that we understand how we are going to approach it. This section will look to provide an understanding of the development methods and techniques that will be used throughout the project's timeline.

2.1.5.1 **Development Methods & Techniques** | Frameworks

A system development methodology refers to the framework that is used to structure, plan, and control the process of developing an information system **[14]**.

Furthermore, when considering what development methods and techniques to use for the creation of a piece of software, there are many choices one can look to utilise. We can categorise these into traditional and modern approaches; a popular comparison that takes place which is suitable to the project being proposed is that of Waterfall and Agile. The following table provides a clear view as to the differences between the two methodologies **[17]**:

Agile	Waterfall
Project software development separated by sprints	Project software development divided into milestones or phases
Incremental	Sequential
Flexible	Structured and rigid
Collection of projects	Single project
Scope for requirements change	No scope once development starts
Testing done in concurrency with software development	Testing done after software development

Table 3: Agile Vs Waterfall

Above illustrates some of the important differences between that of Agile and Waterfall methodologies. Importantly, and what the author believes is the most important comparison, is that of Agile being a methodology that is built upon continuous development and customer interaction. Whereas Waterfall is further described as an approach that will have milestones **[16]**. Furthermore, Waterfall looks to identify a clear set of project deliverables, dates and outcomes with little to no interaction with the customer. Below illustrates Agile software development approach.

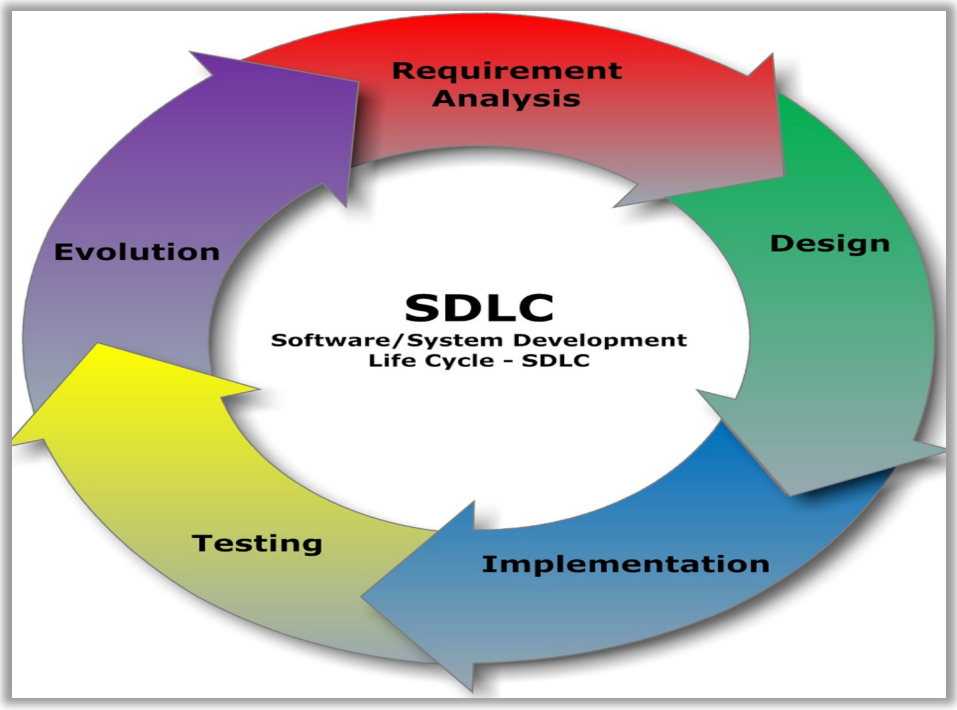


Figure 5: SDLC **[15]**

Above shows the iterative nature of Agile. This differs to that of Waterfall, whereby we can see its SDLC as a more sequential sequence of operations:

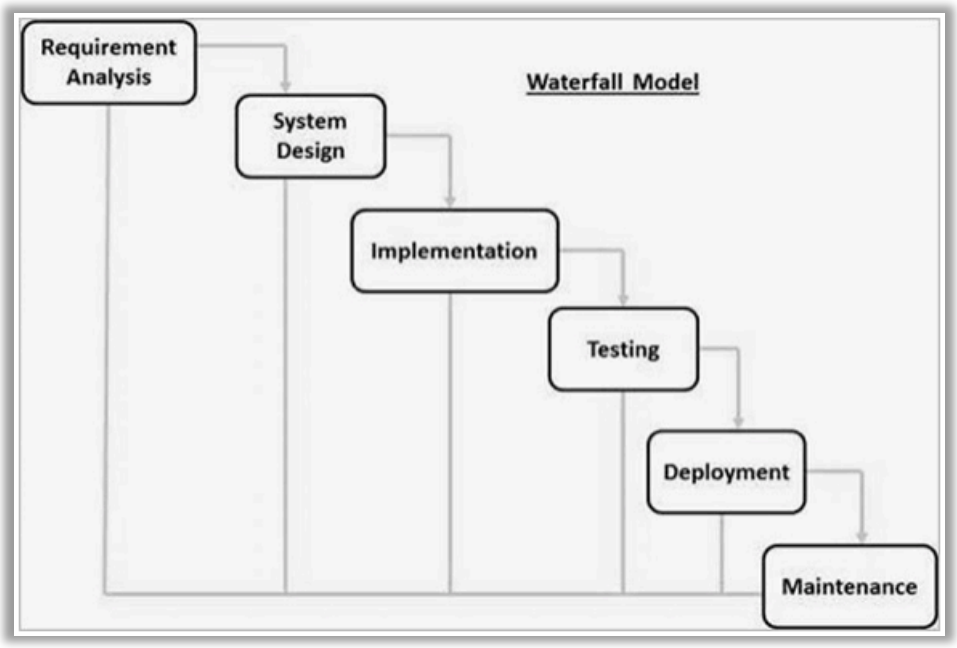


Figure 6: Waterfall SDLC **[19]**

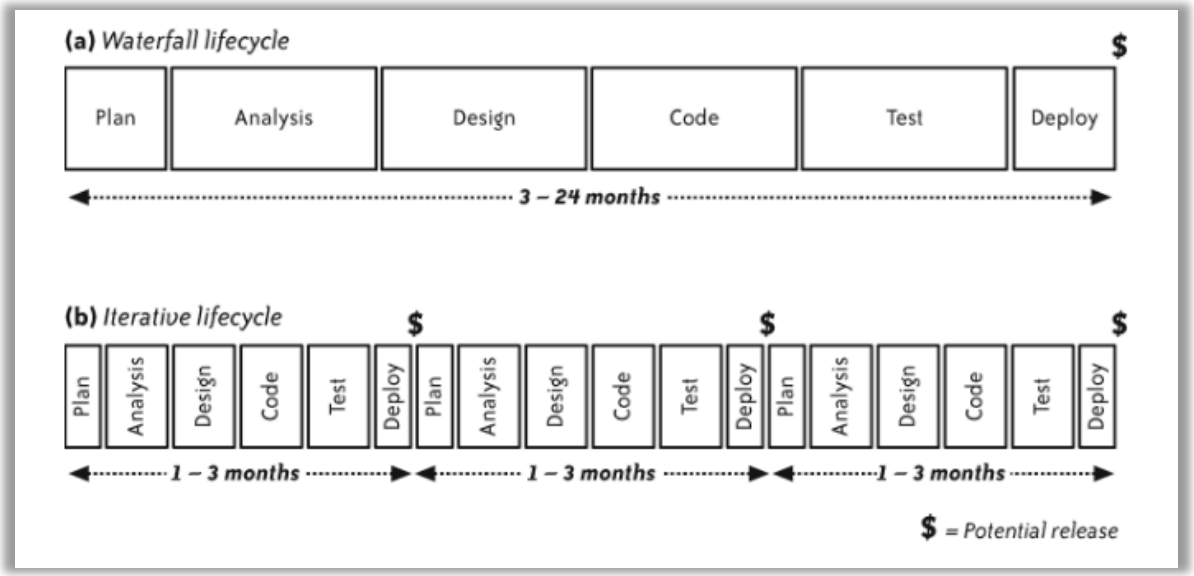


Figure 7: Waterfall | Iterative Lifecycle **[28]**

Above illustrates Waterfall in a somewhat different light and provides us with a better insight into that of iterative waterfall. Key areas of the above will be used throughout this project. Let's now take the opportunity to review each phase and its importance to the proposed project.

- **Requirements Gathering:** The first thing we should do is that of gathering requirements. There are multiple steps one can undertake to complete this phase with the project type determining the steps to be undertaken. Here the environment and project type as suggested will determine how requirements are gathered, however, research **[28]** suggests that customers are responsible for figuring out the requirements for software. From this we can denote that customers are in fact the ones who produce our requirements. Customers are otherwise one of many Stakeholders who provide a set of requirements for us. Another Stakeholder can be seen as that of the user of the system and as such it is the user who will take on the role of the stakeholder for the proposed application.

It is important to point out that due to the nature of this project it will be difficult to get requirements from users and as such it is largely if not entirely the responsibility of the

author to produce user requirements through user stories: Which is one method of gathering requirements. **[29]** states that user stories describe functionality that will be valuable to users of software and/or systems. Furthermore, **[29]** goes on to point out that user stories are made up of three parts: **Card, Conversation and Confirmation**, each providing their own purpose, such as **Card** being that of a written description of the story, **Conversation** being a conversation about the story to flesh out the details and **Confirmation** providing test details to determine when a story is complete. A user story then in the case of the proposed application could be something like:

"*A user can register an account for the application*"

"*A user can login to the application*"

These user stories can provide us with a set of requirements for the application and with adopting an iterative SDLC, these requirements can be built upon and expanded throughout the project's lifecycle. Finally, we can validate the above by highlighting **[35]** who suggests that the user is the single most important aspect of the journey, further suggesting that the user is core: This goes onto validate the idea of creating user stories, with their importance to system design being of the upmost importance. We should also point out that when discussing user stories which is part of the Agile process, we are referring to what the person using the application wants to be able to do and this should not be confused with user requirements which otherwise focuses on what the system should do or its functionality and are most commonly found in the more traditional waterfall methodology.

Another step we can take during the requirements gathering stage is to collect is through Survey's or Structured Interviews. For the purposes of this project, it is likely that Survey data will be collected to provide data surrounding some key elements of this project such as learning modalities and VARK.

Technique	Description
Survey	To seek answers to questions put forward – refer to sections ~2.1.4
Interview	Face to face conversations with those important to the project
User Stories	To understand situations experienced by users of the application

Table 4: Summary of Data Requirement Techniques

- **Design:** During the design phase the author will look to produce design material that will help with the next stage if the SDLC process as seen in implementation. This phase will also look to produce high and low-level designs depicting the proposed applications level design. When thinking about high-level design we are concerning ourselves with the ability to produce documentation and illustrations that depict components expected within the application. These design documents also provide an understanding of the relationships existing within the system: For example, utilising UML here and creating ERD's would help to show these relationships. Once high-level designs have been completed, we can begin to think about any low-level design considerations that need to be made. Here we will look to dive into deeper detail of the high-level considerations such as any error messages produced by the application, input and output of modules and database considerations such as the type and sizes expected.
- **Implementation:** This phase is otherwise known as the coding stage. Here we will look to utilise the decisions made in the previous phase which if done correctly should make this entire process an easier one. This stage will also look to use any tools that have been highlighted as appropriate for the said tasks as well as any programming languages that have been deemed suitable.

- **Testing:** Once programming has been completed and before any deployments on live systems, applications will be tested by the testing team within a test environment. If any defects are found by other QA members or testers, this will be flagged to the developers for fixing before being sent back to retest. Unlikely this approach will be taken for this project as it is unrealistic, however testing will of course be performed to ensure any bugs are kept to a minimum.
- **Iteration:** In an Agile environment, iteration will be performed every few weeks and is otherwise referred to as Sprints. These Sprints allow for work to be highlighted for the sprint period, so developers and testers are aware of the workload.

2.1.5.2 Development Methods & Techniques | Project Approach

Due to the nature of this project and the fact that the project is solely undertaken by the author, it is difficult to choose one approach when thinking about what methodology to use. It is likely that the project will take many different aspects from a number of methodologies. For example, the proposed project is working to a strict deadline, one that cannot be missed or changed, therefore the project will utilise components of the Waterfall model such as working to a fixed deadline. That being said, an Agile approach will also be beneficial as there may be a case for change in project requirements and outcomes, although at this point an MVP has been defined as an educational application framework, further suggesting a Waterfall approach.

Due to this, it is likely that there will be a leaning towards that of the Waterfall model. This is due to a number of requirements being set in the early stages of the project; once set these requirements will form the expected outcome, similar to that of Waterfall. It is difficult to have continuous interaction with customers as the customer in this circumstance, probably doesn't know what they want out of an educational application.

Rather we can collect data early on and build the application using this information: Finally, we can collect feedback data once the application is completed, similar to that of the Waterfall model. This is also suggested by **[17]** whereby stating that the Waterfall model is ideal for projects which have defined requirements, and no changes are expected, whereas Agile is suited to projects with frequent requirement changes. Further literature also supports this and further goes on to state that the differences between traditional methodologies and Agile Methodologies relies on two main assumptions: First, traditional methodologies assume that customers do not know their requirements, hence they need guidance from the developers, but Agile Methodologies assume that both customers and developers do not have full understanding of requirements when the project starts **[19]**. This project has a set deadline and as such it would be unreasonable to think that large, impactful changes can be made at the later stages of the project's lifecycle.

However, this being said, it is still beneficial to use some parts of the Agile SDLC whereby we can continuingly build on what we have and test as and when components have been built, further removing potential bug overheads that may exist at the end of a build. Again though, it is important to recognise that testing practices such as Test-Driven Development are not exclusive to that of Agile Methodologies and can be utilised whilst working within a Waterfall model.

It is important that we understand that there are actually two flavours to Waterfall, such as Traditional Waterfall and Iterative Waterfall. Earlier, Traditional was a very popular model to use, however it does appear that it was too idealistic in its approach, hence the creation and adaption of Iterative Waterfall as illustrated in Figure 6.

An important piece of literature such as that of **[27]** goes on to suggest that the key to great software development is indeed that of iteration and further states that iteration is like a frequent check-up for our software. Looking at figure 4, we may be able to see

iteration as the evolution stage, however for the purposes of this project we'll be referring to this as iteration as it provides us with a clear understand of this part of the process. Furthermore, we can see **[27]** suggest that regardless of the steps involved during any process, iteration can be applied, and is furthered said to be best practice. This observation is important to highlight at this stage as it shows that regardless of choice towards the development methodology used throughout this project, iteration is an important part of the process and one that can be used as part of the project's lifecycle, regardless of methodology.

2.1.6 **Learning Materials |** Consideration

Consideration needs to be given to that of the material that will look to be incorporated into the proposed application. This section will look to research and highlight the topics considered for inclusion within the application. The introduction to this topic also brings to the forefront, the question of "who is this application aimed at", are they beginners, intermediate or advanced in their knowledge of Information Technology? It was hoped that at the initial stages of the project, learning material would be used that was in line with the technical requirements of engineers at the DVLA. Although it is necessary to point out to that this is what would be hoped for, however with an MVP in place which describes a framework for educational material it should be noted that these materials may not be in the application, although something would be used so that tests could be performed on user experience.

To summarise, once finished the proposed application should be able to house content for learners of all varying levels and as suggested throughout, focus lies rather in the application rather than its audience.

2.1.7 **Programming Languages** | An Introduction

There are many programming languages that one can think about using when undertaking a project such as this. The language of choice should be decided on the type of application that is to be developed. For example, we see many games developed using languages such as C and C++ due to considerations such as memory management and its finite nature. We then see a host of other high-level languages such as Python, Ruby, Java and C# all of which are capable of developing interactively rich applications, capable of producing modern applications. As such, this section will look to present suitable languages for the proposed project.

2.1.7.1 **Programming Languages** | Java

The Java programming language is an Object Oriented, Class-Based language that is built on the Write Once Run Anywhere principle or WORA, something in which is made possible through the concept of a JVM or Java Virtual Machine. Many users may be used to the concept of executable files or .EXE, which is otherwise executable compiled code that allows us to run applications. Java takes a different approach: This can be seen as the Java Bytecode, which is described by **[31]** as being a highly optimised set of instructions that are further executed by the JVM, as discussed above.

The importance of the Java JVM and Bytecode come in many areas, notably that of Security and Portability. For the purposes of this project, the benefit to portability will be of particular interest. **[31]** further goes on to suggest this by noting that the translation of Java applications into that of Bytecode is what makes it possible to run the programs on a wide variety of environments: This is because only the JVM needing to be implemented on each type of platform, once this is implemented the Bytecode will run, regardless of

platform. Finally, **[31]** states that execution of bytecode by the JVM is the easiest way to create truly portable programs.

In order to compile Java applications, we need to use Compiler applications known as an IDE, or Integrated Development Environments. For the purposes of this project, if Java is chosen as a suitable language, IDEs such as NetBeans, Eclipse and IntelliJ are all suitable options. We need to use IDEs due to Java being a high-level programming language: When applications are run, they are done so using the devices CPU, which otherwise only understands that of machine language instructions.

It is important to recognise that there are alternatives to compiled languages such as that of Interpreted languages as **[32]** suggests. We can further see that Interpreters, unlike a Compiler runs in a loop cycle reading instructions line by line from a program, further making decisions about what machine-language instructions need to be carried out. The advantage to Interpreted languages is that they can allow for machine-language, which is built for one type of computer ecosystem, to be run on completely different ecosystems.

The above brings us onto our next important point: **[32]** suggests that designers of the Java language decided to use a combination of both Compiled and Interpreted approaches. Java does still compile its source code into a machine-language, which is otherwise known as Java Bytecode: However, this machine doesn't really exist, and it is where the JVM comes in. As previously suggested, the JVM is a virtual machine and it is this that allows Java applications to be platform independent and allows for WORA to be put into practice. We can further see the above illustrated as:

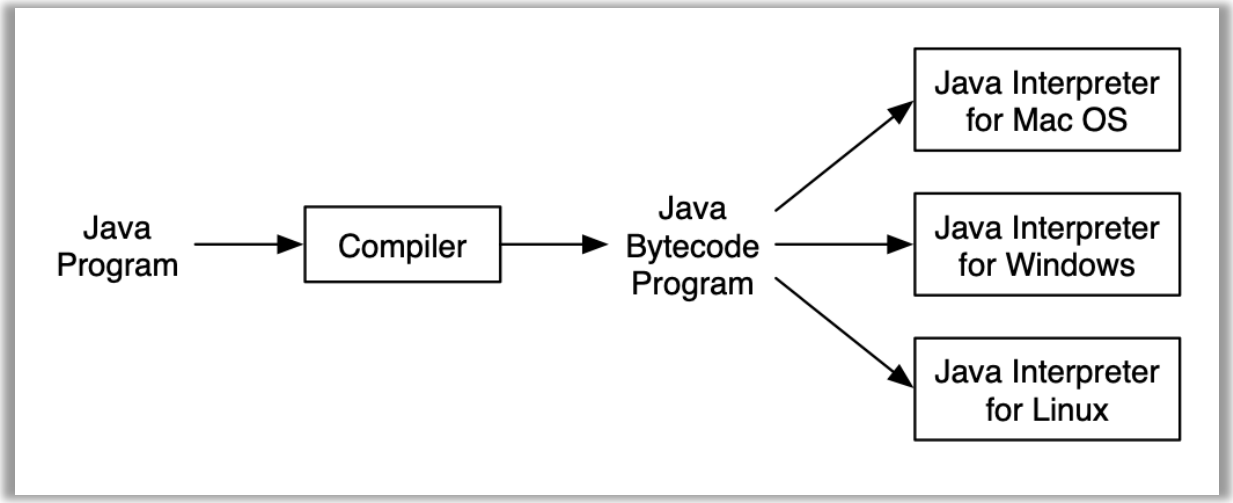


Figure 8: Java Compiler Interpreter **[32]**

This is important as the language provides multiple possibilities for the proposed applications environments. Importantly, unlike languages such as C Java further provides support for that of Automatic Memory Management, thus removing the responsibility of memory management from the developer and saving development time in the process.

The Java language also comes with a rich set of Class libraries, further supporting the developer in development. Libraries of particular importance that would be utilised as part of this project are libraries such as:

Library	Description
UI	JavaFX, Swing
JDBC	Java Database Connectivity

Table 5: Java Libraries

Note that the above library considerations for use is not a comprehensive list of libraries that will be used, rather an insight into the language's capabilities.

Finally, if we take into consideration statistics presented in **Chapter 1**, **Table 1**, we should note that Java is possibly the best choice of programming language. This is largely due to the nature of the JVM and the WORA principle as highlighted above. Being able to

develop an application that can be run on multiple devices allows us to focus on the proposed application rather than make a decision to target platform A vs platform B, such as desktop or mobile.

2.1.7.2 Programming Languages | C#

Another language similar to that of Java is C# or otherwise pronounced C Sharp. When choosing a language that is capable of developing a project such as this, it is hard not to consider that of C#. Unlike Java, C# applications would most likely be written in that of some type of Windows based environment such as Visual Studio or Visual Studio Code.

2.1.7.3 Programming Languages | JavaScript

There is potentially a case for the use of JavaScript as part of the development of the proposed application. With many programming language frameworks such as the .NET framework for that of C# and Java Libraries such as JavaFX offering support for the imbedding of web based technologies such as HTML, CSS and JavaScript through controls such as WebView, it is reasonable to consider the use of a language such as JavaScript. We can further see the importance of its use through literature such as [JavaScript Book] who suggests that due to the language being a Web-enhancing technology on the client side, it can further turn static experiences into that of engaging, interactive and intelligent experiences.

Using controls that allow us to manipulate data as if it were on the web affords us the ability to do more with our software components. An example of this use may be where an admin of the application wants to add new material: This can be done statically by creating templates to hold the necessary information, however if we want a tailored experience such as text, audio, video and images being in different areas of the suggested control, then the use of web based technologies can help us to achieve this.

2.1.8 UML | An Introduction

Previously we discussed the idea of high & low-level design considerations that need to be made as part of the SDLC process. In order to achieve this, we need to understand the concept of UML or Unified Modelling Language. UML allows for the visualisation of a system's ontology, providing a means by which to separate components of a system as well as determine relationships that exist: **[30]** Suggests that UML is a graphical language that allows us to visualize, specify, construct, and document the artefacts software systems. Furthermore, we see **[22]** suggest that since the 1990's UML has become the de facto standard for the modelling of software systems. So, we can denote from this that UML can help us to understand a system on a conceptual level and help others to understand what it is we are looking to achieve through the use of a number of UML diagrams.

2.1.8.1 UML | Structural Diagrams

Importantly we must point out that when looking to produce UML we create either Structural or Behavioural diagrams. Some Structural Diagrams are now discussed.

- **Class Diagrams:** Systems are built from objects, of which can be physical or abstract in their nature. Due to this class diagrams help us to describe the structure of a system **[40]**. Furthermore, the structure of a system is built using classes and the relationships these classes have. As such it is reasonable to suggest that we can use class diagrams to satisfy the need of modelling any systems and its objects. For the purposes of the proposed application we can, for example, create class diagrams that represent databases, users and/or course content such as modules.
- **Object Diagram:** Object diagrams explain and illustrate complex class diagrams **[40]**. As expected, these diagrams are illustrations of objects and any relationships that exist at specific moments.

For the purposes of this application and due to likely time constraints, it is likely that the only requirement of a structural diagram will be that of the Class diagram.

2.1.8.2 **UML** | Behavioural Diagrams

Behavioural diagrams help to highlight the activity scenarios such as a user creating an account or testing themselves.

- **Activity Diagram:** Activity diagrams are described as being used to describe the workflow and operations performed within a class **[40]**. Operations within the proposed system can likely be, for example, how a user creates and account to access the application.
- **Use Case Diagram:** Use case diagrams allow us to capture the functional requirements expected within a system. When we talk about functional requirements, here we are looking to depict a user's or in the case of the use case, an actor is used to show a user, interaction with functions of the application.
- **Sequence Diagram:** Sequence diagrams can help us explore as well as visualize the sequence by which objects of our system interact with each other with these interactions typically illustrating the sequence of messages between objects **[40]**.

2.1.9 **Systems** | An Introduction

Now that we have discussed UML, how to model systems and the diagrams that can be used to help us achieve this, it allows us to think about what a good system actually is. Being able to design a system is good and well but if we don't design and implement something which is fit for purpose, design and implementation of this system means very little. This section will look to highlight literature that outlines what good systems are.

2.1.9.1 **Systems** | What is a Good System?

When thinking about what a good system will be, reason suggests that we can expect a good system to be something that is fit for purpose. **[23]** suggests that ultimately good systems are ones that meets its users' needs. Further going on to suggest that it should be:

- **Usable and Useful:** Software that falls under this category makes the lives of people easier or better. Useful and usable in terms of the proposed application will be determined by if it helps educate the user.
- **Reliable:** The software should be reliable with few bugs.
- **Flexible:** Research further shows as **[23]** suggests that a system needs to be flexible. Users' needs are going to change over time and as such the system should be capable of change without too much impact. This system maintenance shouldn't present problems to the developer.

2.1.9.2 **Systems** | Programmatically Good System Considerations

Above outlined considerations of what is a good system. We now, using this, need to think about what programmatic considerations make the above possible. **[23]** further suggests that we should think of a system as a collection of modules and to further identify dependencies between these modules. In order to do this **[23]** suggests that we must consider the related concepts of dependency, coupling, cohesion, interface, encapsulation and abstraction.

Looking at the above, we can denote that dependency is sometimes otherwise referred to as coupling. That is, if a system had many dependencies it is considered to be highly coupled. Furthermore, if a system has few dependencies it is considered to have low coupling. So, by this we can evaluate that if we require a change to be made to a

component within our system, we do not want to have to make these changes to other parts.

2.1.9.3 **Systems** | So, How Are Good Systems Built?

Pg. 13 When undertaking a project such as this, it is important to ensure that we have considered literature surrounding what type of software engineering approach should be used. **[23]** further goes on to suggest that the following items should be considered:

- **We should** ensure that we have a clearly defined set of processes with a clear set of phases, further ensuring that each has an end-product of some sort. When thinking about this we really want to consider the use of Story Boarding and User Stories when wanting to define processes and end-products. Using this, we can continue to action upon it throughout the design and implementation phases, each of which produce designs and the end deliverable, such as the application. **[23]** does suggest that this end deliverable can even be documentation and doesn't necessarily need to be something that is built however, it can mean both.
- **We should** be concerned with achieving a clear set of requirements which are defined early on. This has been done within sections 1.1.3~ which highlights the aims and objectives of this project as well as any considerations made during the requirements gathering through user stories and/or surveys.

The above literature is important as it further goes on to suggest that early considerations made in this project regarding data collection and testing, is in fact correct.

2.1.9.4 **Systems** | Low Coupling & High Cohesion

It is said that in the past, terms coupling, and cohesion have helped in making the decision of how to group software entities together, and further suggestions state that these concepts were made popular through structured analysis and design in the early 1980's

[37]. We understand that cohesion in software design is used to determine if software entities belong together and should further maximised. Furthermore, we can determine if high cohesion has been achieved in our classes if the class does one thing, and it does this well. For example, we may want for our Database class to only concern itself with tasks that are related to the database: It is further suggested that one way to view cohesion within Object Oriented programming is if a classes methods use private methods **[38]**, if they do we consider this to be high cohesion.

Coupling on the other hand refers to the level by which or rather degree, modules and classes depend on one another. By this we can determine that any classes of the said application can adhere to loose coupling if we ensure dependencies are kept to a minimum and weak.

Design Consideration	Description
Cohesion	How closely related is everything
Coupling	How everything is connected to one another

Figure 9: Summarise Cohesion & Coupling **[38]**

We can further illustrate this with the following:

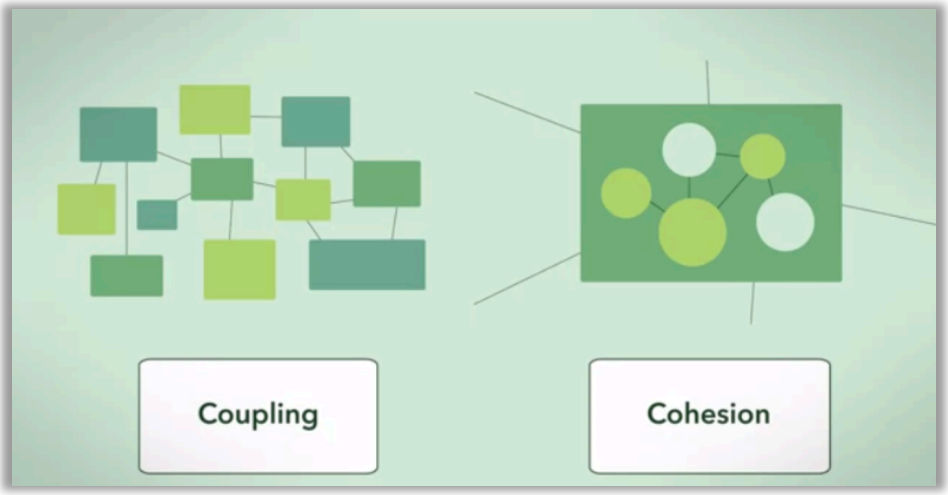


Figure 10: Coupling & Cohesion **[39]**

We should we feasible look to implement such considerations when implementing research and design considerations for the proposed project.

2.1.10 **Development Methodology |** An Introduction

When undertaking such a large development project it is important to decide upon a software development process. Previously, project literature outlined that an Agile approach would be more beneficial to the progress of the project, as such, there are many methodologies one can choose from such as Test-Driven Development or TDD; sometimes referred to as Test-First Development or TFD. Other considerations can be processes such as Behaviour-Driven Development or BDD or Acceptance Test-Driven Development. To choose the correct methodology there needs to be some research into the different approaches, as such this part of the project will look to research the above and make a decision as to what the best approach will be for the proposed project.

2.1.10.1 **Development Methodologies |** TDD

Test-Driven Development is a software development process primarily used by developers of a said application, that further allows for the testing of an application before the code is written. Literature suggests that TDD is the process by which tests are written before any code and when we first write these tests, we purposely want them to fail, which further allows a measure of success to be established **[41]**. This iterative and incremental way of work is otherwise referred to as the Red, Green, Refractor cycle **[41]**.

The above suggests that this approach could suit that of a project such as the proposed and as such efforts will be made to incorporate these development methodologies into the projects SDLC, which has already been touched upon throughout this project.

2.1.11 **Databases** | An Introduction

Project scopes such as the one proposed, require a database to be able to store, modify and delete data that is held at any one time. Data such as users, courses, modules, exams and even user privileges will be stored so that the application can function as suggested. **[24]** suggests that there is a fundamental concept behind database systems: There are business environment data points that need to be stored and these are all related in some way. In the case of this project, business environment data stores may be things like users, courses and modules, which are further related to one another as many modules belong to a single course and users can study many modules and courses. It is these types of considerations that we need to make when thinking about a database.

[24] further goes on to explain that the idea behind database systems is that users of an application, whether that be person or program, such as an API, do not need to concern themselves with how the data is stored. This suggests that users should be able to simply access parts of a system and subsequently interact with it without having to think about the data interaction that is going on. They do this through data manipulation requests which further brings us onto our next point, RDBMS or Relational Database Management Systems: It is these systems which process the request of the user for data stores **[24]**.

Databases are a key component to any application and as such this section will look to highlight important aspects of databases.

2.1.11.1 **Databases** | Database Lifecycle

Importantly we also see that the process by which we are concerned with our database systems and how to implement them, we should consider the Database Development Lifecycle much like the SDLC that has been presented previously. **[36]** suggests that there are six phases in total, many of which resemble that of the SDLC such as, **Strategy** &

Planning, **Requirements Analysis**, **Design, Development**, **Deployment** | **Implementation** & **Operations** and **Maintenance**. For the purposes of this application points such as Requirements Analysis, Design, Development and Implementation will look to be achieved.

2.1.11.2 **Databases |** Relational & NoSQL

When we concern ourselves with Databases, we are often presented with two main categories: Relational or SQL & Non-Relational or NoSQL databases. Relational databases have table-based structures what further enforce strict rules and predefined schemas. In contrast NoSQL can be in multiple structures such as document, graphs and key-value pairs and don't require schemas and importantly data is unstructured in its store **[34]**. We can determine that this means an advantage in NoSQL unstructured approach means that we can build an application without having to first define its schema. Importantly, we can see a database schema as being a description of the actual construction of the database.

In recent times we have seen an increase in the support for and use of NoSQL databases, with NoSQL promoters suggesting that the technology is the future of databases. However, we should note that RDBMS promoters suggest that NoSQL lacks the ability to provide sufficient data integrity **[33]**.

An important point to highlight when thinking about what database architecture to deploy for the project is that of NoSQL being used with big data applications and data-intensive cloud applications **[33]**. NoSQL is also schema-less, and it is suggested that this lack of metadata prevents NoSQL database systems from knowing what data is stored and more importantly how this data is interconnected. As such this does present the problem of querying data, something that will be highly important for the proposed application.

2.1.11.3 **Databases** | SQLite

SQLite is one option that the author can consider when looking to implement a database for the proposed application. There are a number of relational and non-relational database RBDMS tools one can look to use to implement and manage any given database, however due to the nature of the application a lightweight tool is preferred at this time.

SQLite provides this lightweight approach to databases and is therefore worthy of consideration.

2.1.12 **Application Design** | An Introduction

Consideration needs to be made towards that of how one might go about designing a Java application. This section will look to highlight the choices available to developers or designers to design and implement a GUI for the application. **[48]** suggests that for many users the UI is the product and will further judge the quality of the application solely from the perception of the UI.

2.1.12.1 **Application Design** | JavaFX

The Java language has seen many iterations of available ways to design an application. Currently, applications written in Java a designed using the JavaFX graphical and media tools, further allowing for developers to design, create, test, debug and deploy rich client applications **[44]**.

How your JavaFX application looks and feels can be changed through the APIs allowed use of CSS or Cascading Style Sheets, further providing those from backgrounds such as web development to easily style and customise their applications.

Importantly we need to understand that JavaFX applications are built using FXML which is an XML style-based language. This can either be written directly or can be

generated using JavaFX Scene Builder which is a graphical tool that can be used to drag and drop FXML controls to build out a GUI. Key features of JavaFX include:

- **Java APIs:** JavaFX applications are able to reference any Java API libraries
- **FXML & Scene Builder:** As suggested above, Oracle **[44]** suggests that FXML is an XML-based markup language for the constructing of JavaFX user interfaces
- **WebView:** An important feature to that of JavaFX is that of the WebView which is further described as a web component that uses WebKitHTML technology to allow for the imbedding of web pages within JavaFX applications **[44]**.

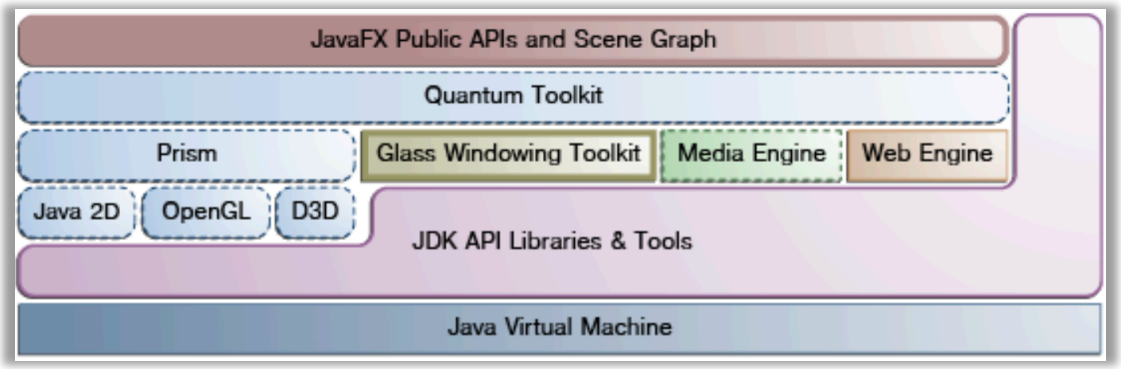


Figure 11: JavaFX Architecture Diagram **[44]**

The above outlines a few of what has been highlighted as potentially important features of JavaFX and what will likely be used during the development of the application. Due to the constraints of time, it is likely that with a tool such as Scene Builder will be used to help the process of designing and implementing a UI for the proposed application: Furthermore, we can see **[47]** suggest that Scene Builder is a must use tool for every JavaFX developer which further helps to speed up development times and provides a way which we can keep a controlled structure.

2.1.12.2 **Application Design** | Swing

Swing, which is a predecessor to JavaFX has been for many years the popular choice for designing Java applications. However, with Swing no longer being supported it is unreasonable to consider such a tool to design the application.

2.1.13 **Design Patterns** | Consideration

One consideration that may be of importance to that of the application is that of Design Patterns. **[43]** suggests that it is the nature of software development to encounter problems on a re-occurring basis, while knowledge is improved after finding solutions to common problems, one can produce design patterns to help overcome these problems again in the future. This is further suggested by **[43]** as it is explained that experience that has been gained in the past can be reused in similar situations thus reducing the effort required to produce systems that are more resilient, effective and more flexible. From this we can conclude that Design Patterns may indeed play an important role in the development of this application and is therefore an important consideration.

2.1.13.1 **Design Patterns** | Singleton Pattern

One pattern of interest is that of the Singleton Pattern. If we ask the question, how does one create a class that only has one instance available at any one time and is globally accessible, then then the Singleton is a patterned solution to this commonly encountered problem. This is a common issue encountered by software engineers, as for example, when looking to implement a database, one may want to restrict database connections, as such this pattern can be of particular use as it will ensure that only one instance of a database connection is made at any one time: Thus, ensuring that proper use is made of the database object being used. Research suggests that **[43]** in some applications it is important that a

class have exactly one instance, as noted above we can see such occurrences may present themselves through database concerns or even users interacting with the application.

2.1.14 **Frameworks** | Consideration

Frameworks provide us with the ability to extend our applications functionality. These are important considerations to make when looking to develop an application as we may need to add framework support for our application so that we can further accomplish more tasks.

2.1.14.1 **Frameworks** | Maven

Maven is a build tool which is described as initially being built to simplify the build processes within the Jakarta Turbine project. Furthermore, Maven is described as a tool that can be used to build and manage Java-based projects **[45]**. Being able to manage and implement further tools into your Java project is a particular benefit to using Maven as it further allows us to import features that enhance our application. Maven dependency considerations can be seen as the following:

- **Junit:** Dependency for Java testing suits
- **JDBC-SQLite:** Dependency for Database interactions

2.1.15 **Testing** | Consideration

One item of consideration should be that of testing. Much has been discussed surrounding the use of testing during the lifecycle of software development as seen in the SDLC as well as that of TDD. As such certain tests should look to be performed whilst developing the application however, consideration should also be given to the time limitations of the project and while important, much time is needed to understand that of testing suites.

This Section Left Intentionally Blank

3. Chapter Three

3.1 Research Methods

Before continuing it is important to understand what Research and Development Methods the project intends to take.

To ensure that the correct research is performed and in a proper manner, Biggam **[7]** further suggests that there are a number of strands to a research methods chapter, typically including the following sub-sections:

- Introduction
- Research Strategy
- Data Collection
- Framework for data analysis
- Limitations and potential problems

The above further provides us with a means by which we can approach our project correctly.

3.1.1 **Research Strategy** | An Introduction

When thinking about what type of strategy to use for the proposed project, there are a number of different ways one can approach their research. A research strategy simply describes how we intend on implementing our own research study **[7]**.

3.1.1.1 **Research Strategy** | Philosophies

However, before we can begin our research, we should decide upon a suitable research philosophy. Figure 1 illustrates the 'Research Onion' which was developed by Saunders et al in 2007. This is a useful tool as it helps researches understand what stages need to be

applied whilst developing a methodology. This is helpful because depending on what type of research is being performed, this will directly influence your research philosophy.

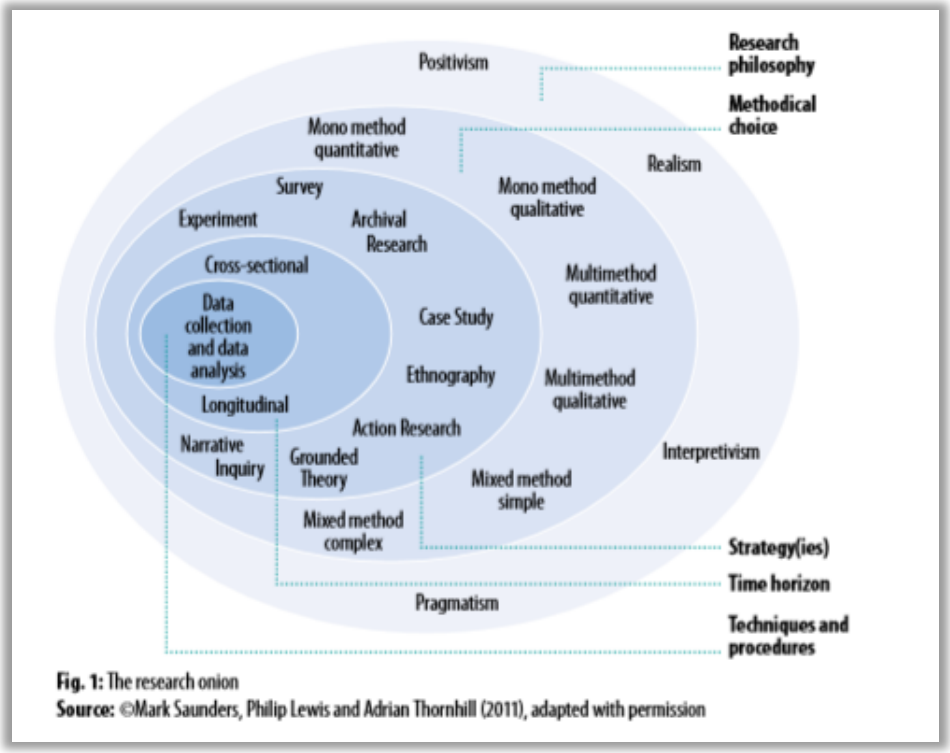


Figure 12: Research Union

The undertaking of this project is to ultimately build a piece of software. As such, it is suggested that the best philosophy for this type of work is that of Positivism. Another good indicator for positivism is the following diagram which helps illustrate the separate pieces of positivism:

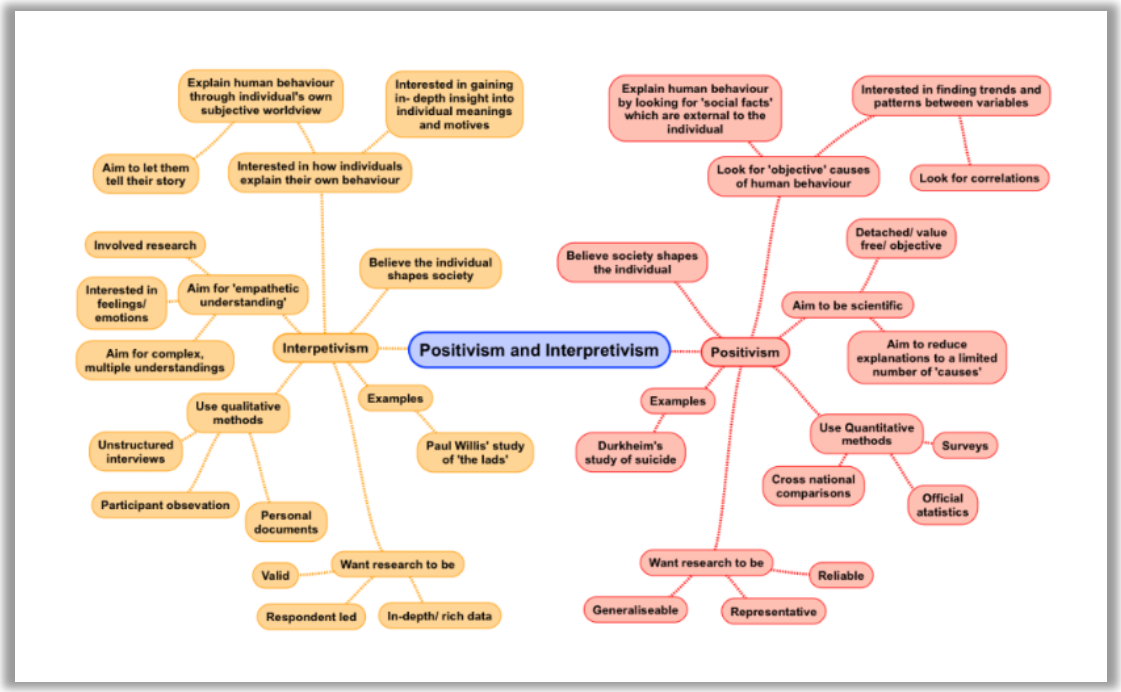


Figure 13: Positivism

The Positivism philosophy takes a scientific approach to research. By basing itself upon observations, Positivism in turn, allows for the collection of data that is otherwise credible and quantifiable. Furthermore, Positivism encapsulates that of both experimental and objectivist paradigms and otherwise suggests that research of its type, is based upon facts.

Positivism			
Real, external, independent One true reality (universalism) Granular (things) Ordered	Scientific method Observable and measurable facts Law-like generalisations Numbers Causal explanation and prediction as contribution	Value-free research Researcher is detached, neutral and independent of what is researched Researcher maintains objective stance	Typically deductive, highly structured, large samples, measurement, typically quantitative methods of analysis, but a range of data can be analysed

Figure 14: Positivism **[8]**

Furthermore, Saunders et al suggests that positivism is typically deductive, further allowing for the use of structured data analysis **[8]**. Now that we have chosen a suitable philosophy, a research approach needs to be decided upon. Both of these approaches can be noted as inductive and deductive.

3.1.1.2 **Research Strategy** | Case Study

The research strategy taken by this research project will be that of a case study. These are well suited to that of software development projects as there is a need for multiple forms of data collection: Such as surveys, observation and structured interviews. By ensuring that multiple means of data collection methods are utilised we can ensure high data validity. Furthermore, a case study is well suited due to the research nature of the project, such as the research into application development, deployment, similar applications etc.

Finally, when making a consideration for that of the time horizon, cross-sectional will be used due to the projects relatively short life span as well as cross-sectional being limited to specific time frames.

3.1.1.3 **Research Strategy** | Study Population Sample

The population for the sample will consist of both students and potentially staff, both of which will be of the School of Applied Computing due to the nature of the project and its contents. Potentially staff at the DVLA may be used as a sample pool however this is yet to be decided.

3.1.1.4 **Research Strategy** | Data Collection Methods

To gather requirements for the required system, staff will be consulted providing a good basis from which to start whilst providing any technical considerations that need to be made.

Once the application has been developed both staff and students will invited to complete anonymous questionnaire, ensuring that questionnaire sites such as Survey Monkey are used to ensure anonymity. Information collected will allow for the realisation

of what the application should provide, how this information is presented and the general look and feel of the intended application.

3.1.1.5 **Research Strategy** | Reliability and Validity

When collecting data, we need to understand that there may be four threats to reliability **[9]**. Of which there are the following:

- Subject or participant error
- Subject or participant bias
- Observer error
- Observer bias

We should also consider that there may be threats to the validity. Validity is concerned with whether the findings are really about what they appear to be about **[9]**. It is suggested that there are six total threats to validity:

- History
- Testing
- Instrumentation
- Morality
- Maturation
- Ambiguity about casual direction

These are all valid reason that can potentially affect our results and are an inevitable part of data collection. However, due to multiple methods of data collection as well as both quantitative and qualitative forms of data, avoiding these, if not minimising the above's impact should be possible.

3.1.1.6 **Research Strategy** | Observations

It may also be suitable to perform observations on the users whilst they are interacting with the application. Being able to see how a user navigates the application and more importantly observing any problems they have may help to developing a more suitable product. However, it should be noted that this may not be necessary as other data collection methods shall be utilised: Although it should be highlighted as a potential avenue to gain valuable insights into user experience.

This Section Left Intentionally Blank

4. Chapter Four

4.1 Design and Implementation

Throughout sections 2~ we presented key literature. Using this, we can look to implement design features expected of the proposed application so that the process of design and implementation can be an easy one. This chapter will look to present design and implementation decisions made while providing critical analysis of why these decisions were made.

4.1.1 **Application Design** | An Introduction

The design of an application is arguably the most important stage in the creation of a tool such as the proposed. When we think about design, we often tend to think of visual aspects such as how does something look. However, there are many other design considerations that need to be made when looking to implement such an application, such as system design or database design. Using research gained throughout Chapter 2, this section will look to present and illustrate how the application has been designed.

4.1.1.1 **Application Design** | Java | JavaFX | Scene Builder

As a result of the Literature Review, it was decided that to build an application that would suit that of the proposed, the Java language would be used. This is due to the language's capability found in that of its JVM and the further ability to implement that WORA principle: Further allowing for one application to be written that can be deployed on multiple devices. We can see this being further suggested as **[46]** suggests that JavaFX complies to the same byte code that java does, resulting in JavaFX being able to run in the

Java VM anywhere Java does. With this in mind, we can see that JavaFX is capable of deploying rich clients that operate consistently across diverse platforms **[44]**.

With this in mind it has been decided that a JavaFX application is the most suited to the project and as such will be used to create the desired application. To design that application, Scene Builder will be used as it offers a powerful set of tools that can be used to implement a feature rich application. As previously mentioned, Swing was overlooked due to it no longer being supported and due to JavaFX providing a host of improvements over Swing, primarily in that of design time, as we can see **[47]** suggest that developers can design interfaces much faster while using JavaFX over Swing.

At this point we should point out that although C# was highlighted as a possible choice of language to be used in the development of the application, Java's ability to be cross platform as well as the points highlighted above have made it hard to not make Java the programming language of choice.

4.1.1.2 Application Design | IDE

With the above considerations we can now decide upon an Integrated Development Environment that will be used throughout the development of the application. Chapter 2 highlighted potential IDEs that can be used when programming with the Java language. Although all platforms offer sufficient tools to complete such an application, IntelliJ was chosen due to the author having access to the Premium version of the IDE: This version provides further programming support to the developer, resulting in less dependence on other applications to complete programmatic tasks. For example, we see the free version of IntelliJ having no support for CSS syntax highlighting, as such developers would have to use other tools such as Atom to ensure CSS is written correctly, further increasing time spent on development tasks. It's important to point out that the decision of what IDE to

use for application design and development is really a preference on behalf the developer and there are multiple IDEs that are capable of developing the proposed application.

4.1.1.3 **Application Design** | Building a Prototype

Having made the above design decisions, we can now look to apply research performed to the building of a user interface.

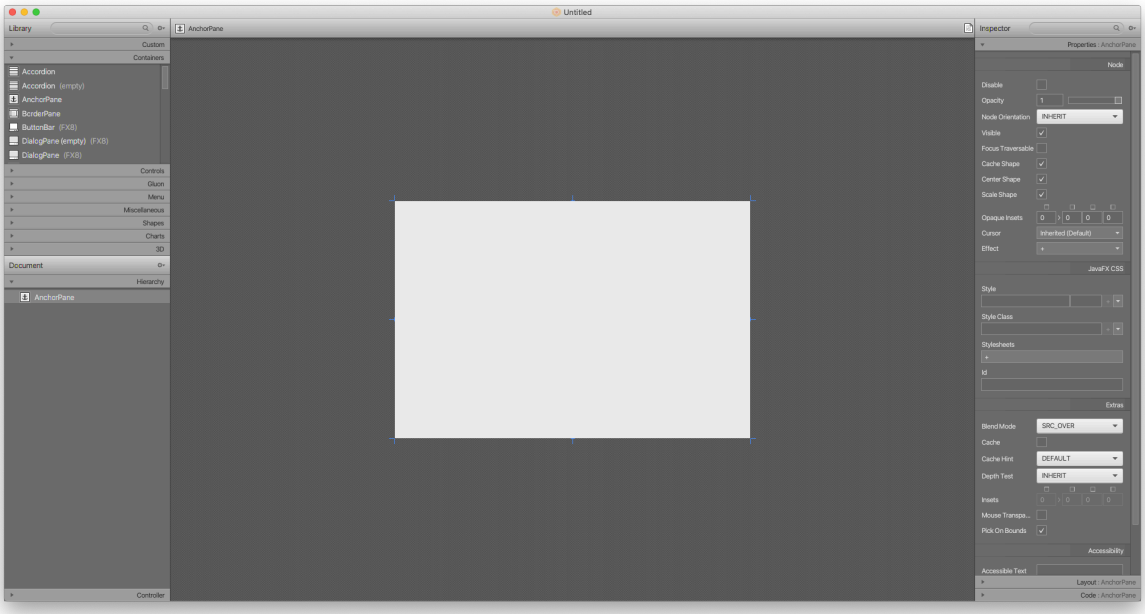


Figure 15: Understanding Scene Builder

We can break up Scene Builder into three parts:

- **Controls:** The left-hand pane provides us with a set of controls that can be used. From here we can choose from a list of layout controls, UI controls and controls that allow us to create animations.
- **Scene:** The main or middle section of the application window is the working area.
- **Properties:** On the right we have the properties section, allowing for the changing of controls such as size and style. More importantly, this is the section one uses to add event handlers to visual controls to provide functionality to that of the user interface.

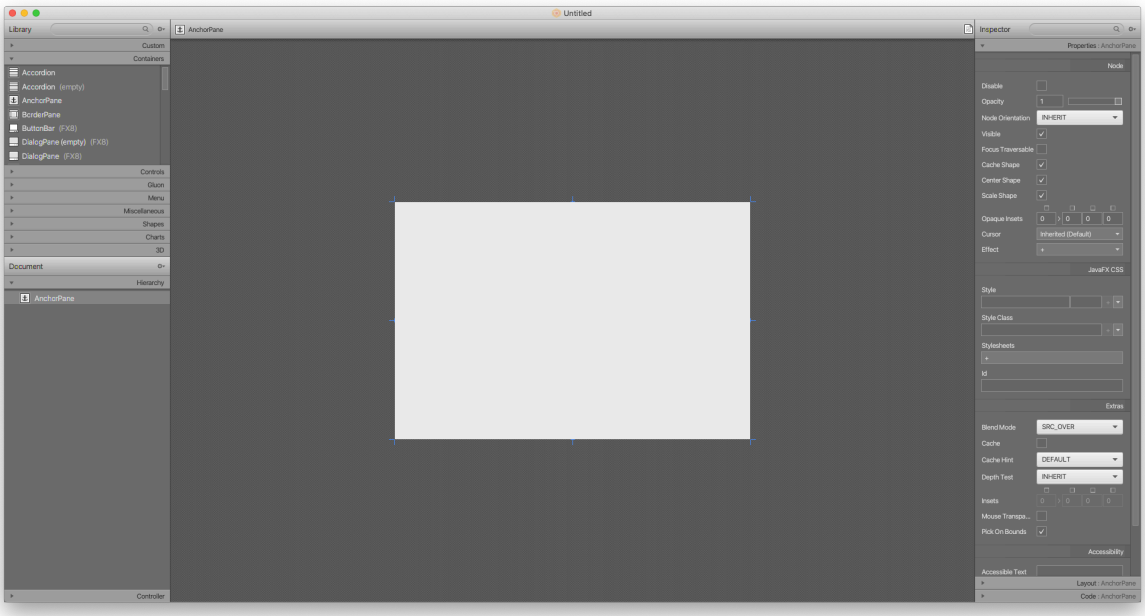


Figure 16: Scene Builder UI Prototype

AnchorPane: The first container we should consider when building a UI is that of the AnchorPane. This layout container as the name suggests allows us to further anchor controls or nodes that are placed within, something that is of particular use when considering the resizing applications. **[44]** suggests that this class enables anchoring of nodes to the top, bottom, left, right and centre of the layout.

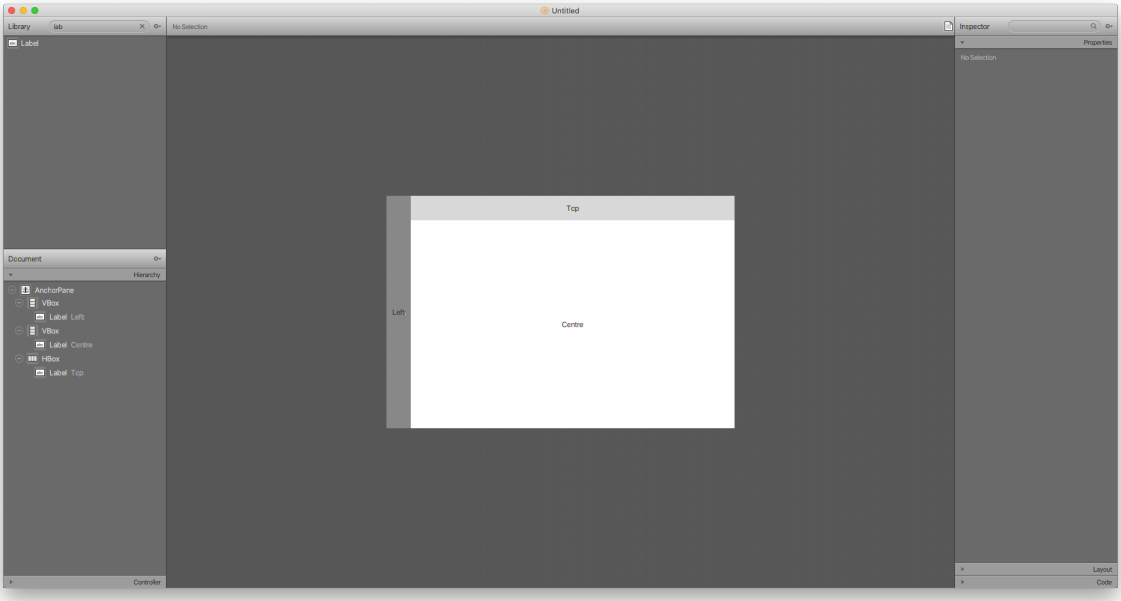


Figure 17: Anchoring Nodes

The above illustrates how nodes, such as a HBOX and VBOX can be anchored within our AnchorPane layout. If we now try and resize the application, we'll note the appropriate resizing take place.

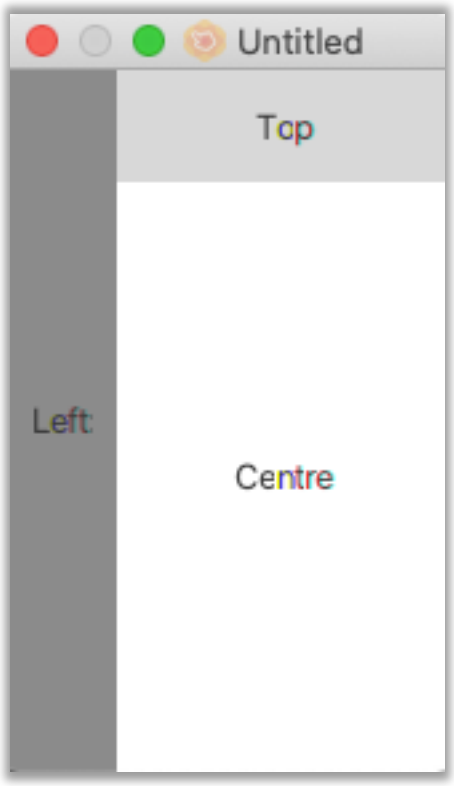


Figure 18: AnchorPane Node Resizing

When designing an application, the above illustrates how we can create a layout that is suitable. Using layout containers such as the AnchorPane allows us to design an application that considers the likelihood of window resizing or device diversity while using the application.

4.1.2 **Implementing Application Design** | An Introduction

Many topics were introduced in Chapter 2 that provided us with an idea as to what design considerations should be made while creating a system design. As such, this section will look to review these decisions.

4.1.2.1 **Implementation Application Design** | Dashboard

Using research and having created a number of templates using JavaFX and SceneBuilder the following dashboard has been created which provides a template for which the rest of the application is designed.

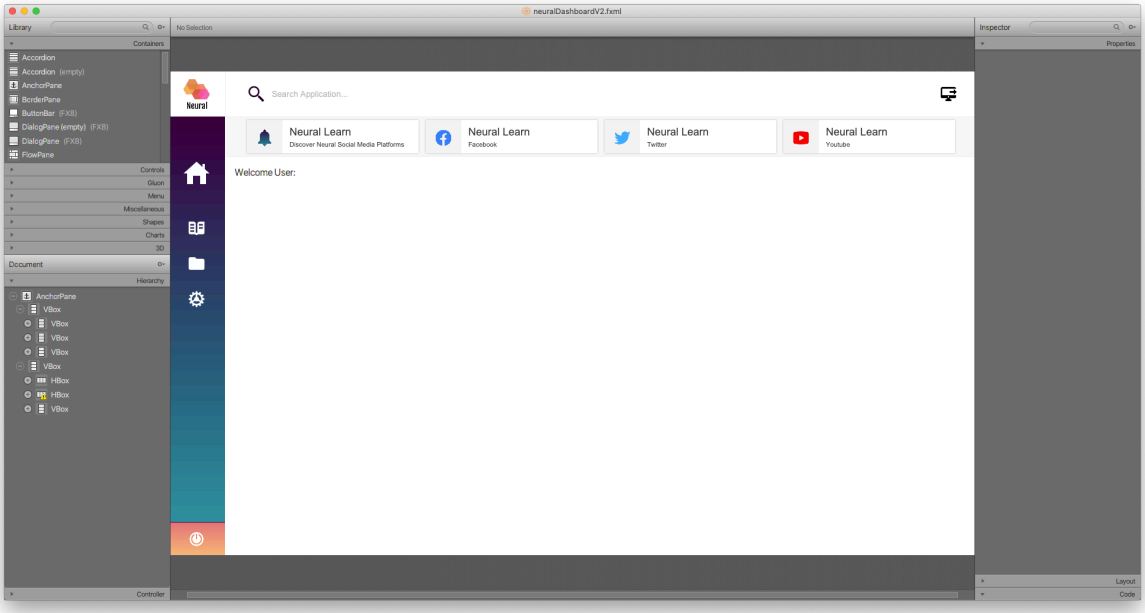


Figure 19: Dashboard

Above illustrates the educational application design and the implementation of this design to create a dashboard for users to further explore the application. Using research from Chapter 2 a display window of 1366x816 has been created. There are a number of key features to the dashboard such as:

- **Navigation:** A navigation pane has been cemented on the left side of the application. This allows users to navigate their way to the available content or if not on the dashboard, navigate their way back to the home screen.
- **Search:** Early release comes with a search feature. At the top of the application window we can find a functional search feature that allows users to specify a course or module that they may be interested in. This provides further enhancements to user experience while allowing users to navigate their way through content more easily.
- **Content Area:** The main area of the window provides a maximised area whereby the content can be displayed. This was given the most screen real estate in the application to allow maximum content is delivered to the user.
- **Social Media:** Early release does not yet have social media for the application: However, this section does intend to provide users with further enhancement to the application while providing alternative routes to finding educational material.
- **Logout | Exit:** The application provides two noteworthy buttons, the logout button located in the top right corner allows users to log out of the application and return them to the login/registration splash screen. Furthermore, in the bottom left hand corner we have the exit application button which allows users to exit. This was added due to a decision being made to produce an UNDECORATED scene window which further removes the standard Mac bar from above the application.

4.1.2.2 **Implementation Application Design** | Login Register Splash

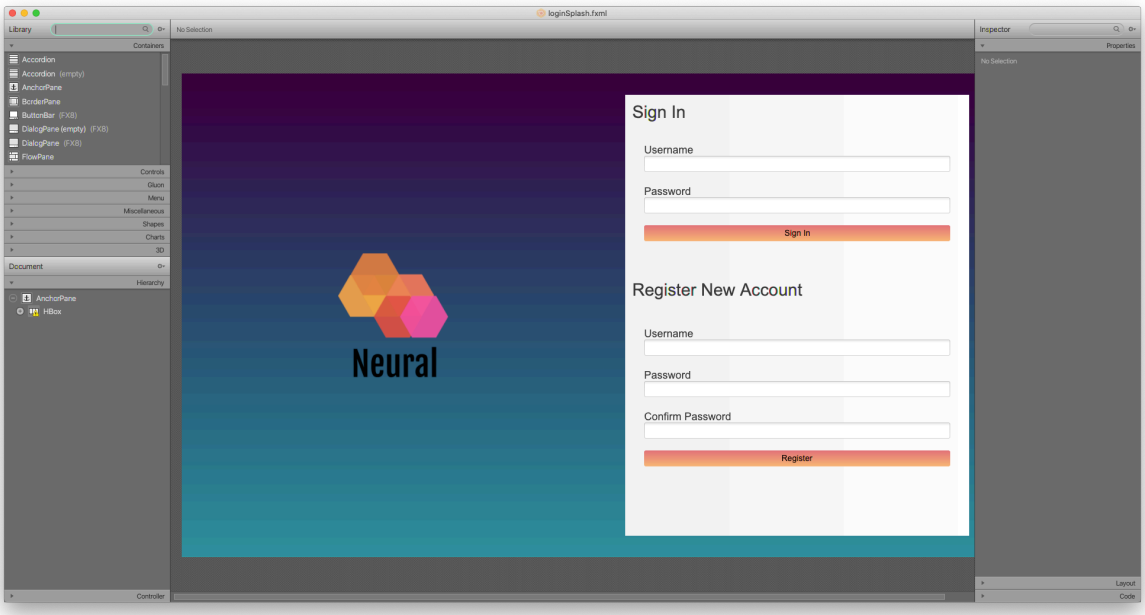


Figure 20: Login Register Splash

Here we can see a number of considerations made towards providing users with a means by which they are presented with a splash screen, login and registration accessibility.

4.1.2.3 **Implementation Application Design** | Admin Dashboard

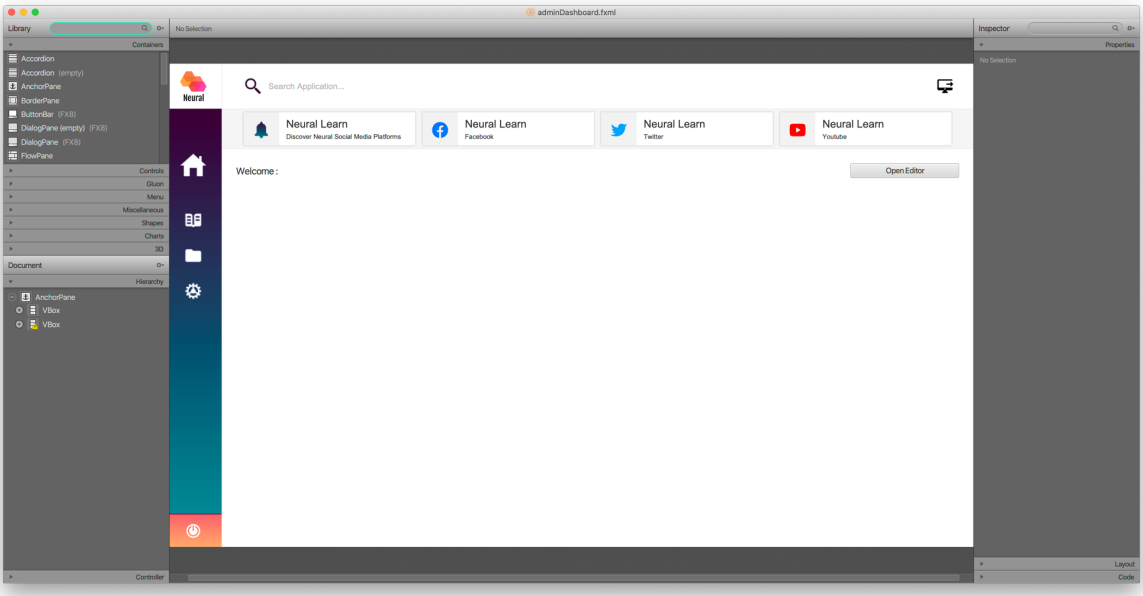


Figure 21: Admin Dashboard

4.1.2.4 **Implementation Application Design** | Admin Editor

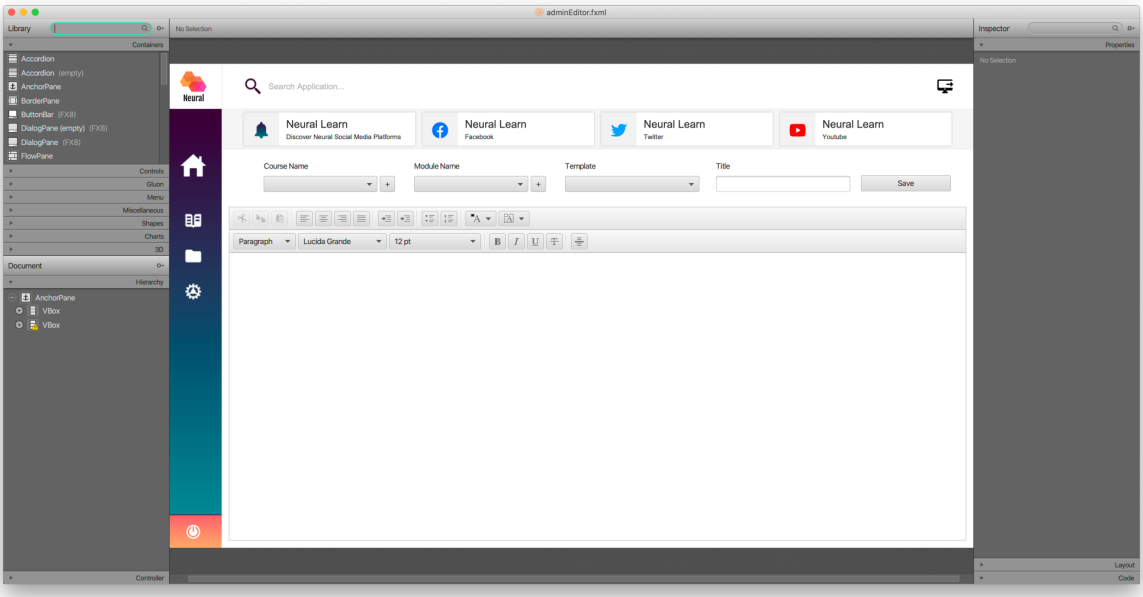


Figure 22: Admin Editor

Earlier we spoke about how FXML and SceneBuilder provide us with a means by which we can link UI elements to that of a controller class which otherwise accesses the

backend models of the application. We can do so by assigning an fxid to the elements in which we want to manipulate, such as:

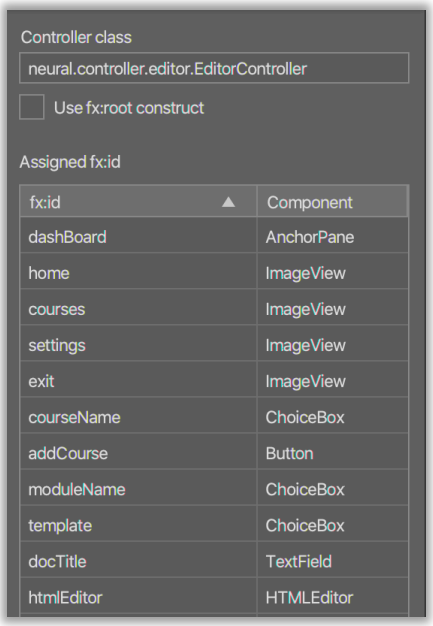


Figure 23: Controller Class SceneBuilder

Above illustrates a number of components set with fxid's so that they can be accessed in the controller class for its belonging fxml document.

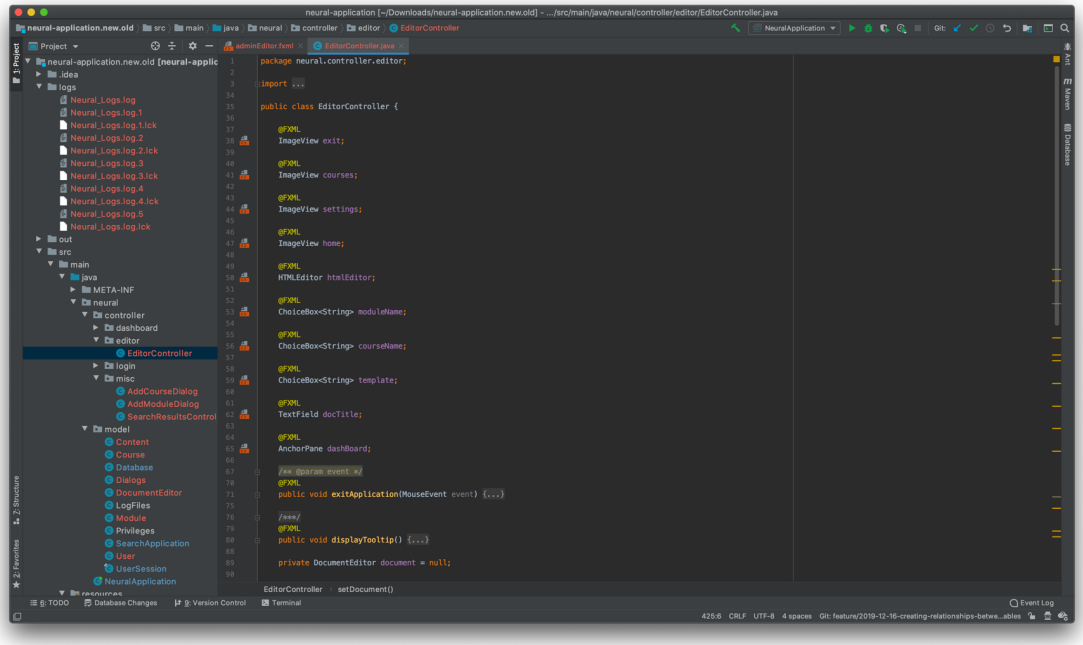


Figure 24: Controller Class

By using the @FXML Annotation we can further create a set of attributes that can be further manipulated. For example, decoration was removed from the applications scene, so the author needed to add a way of exiting the application, as such an event handler was added that uses the @FXML attribute exit, this further allows for any mouse interactions to be handled by the system.

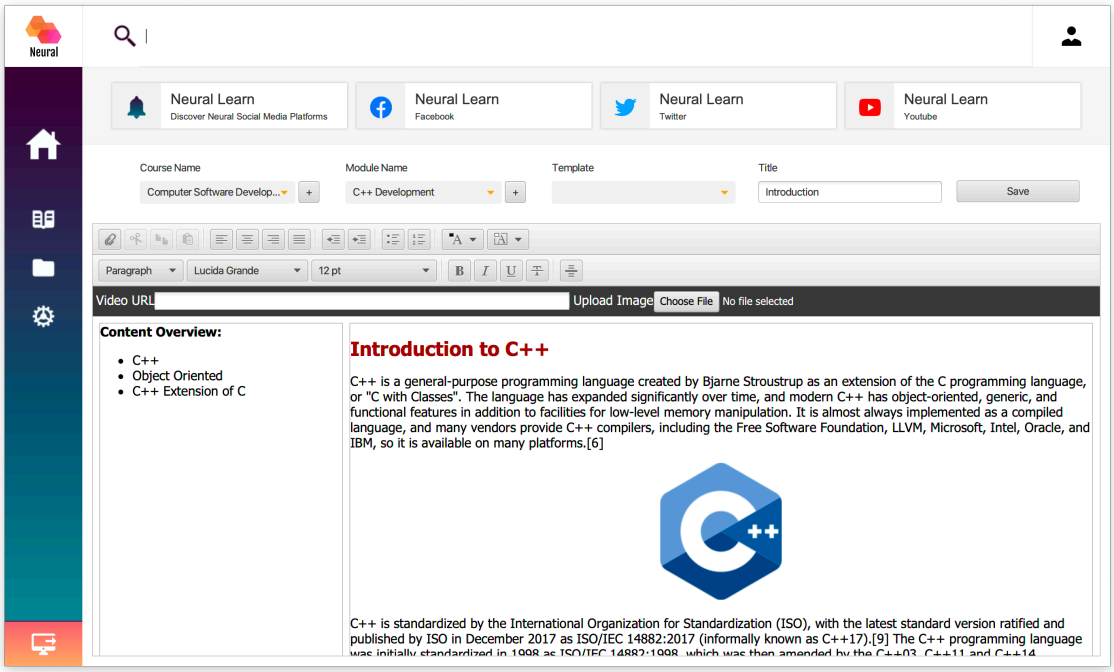


Figure 25: Content Considerations

The above highlights an attempt to allow admin users to design and present content that considers that of Kolb's Experiential Learning Cycle whereby we allow learners a chance to reflect on the content that has been presented: Such as the Content Overview. This bullet pointed section allows users to see the content summarised which helps them in understanding what has been presented to them.

4.1.3 System Design & Implementation | An Introduction

Many topics were introduced in Chapter 2 that provided us with an idea as to what system design considerations should be made. As such, this section will look to review these decisions.

4.1.3.1 **System Design** | User Stories

In Chapter 2 we discussed and reviewed literature surrounding, the requirements stage of the SDLC. We reviewed how when working in an Agile way, user stories are produced to provide us with a set of system design requirements, whereas the more traditional Waterfall produced requirements documents for this.

Using this we can conclude that user stories have been created to further produce system requirements. Furthermore, due to their iterative nature, they can be created as required, further allowing for requirements to be generated throughout the project's lifecycle. There are a number of software applications that exist to develop user story mapping. The author has decided to use Cardboard **[49]** to more accurately capture requirements. We should also point out that it is because of this story boarding that previous sections such as the creating of login and content pages were able to be created: Setting the basis for requirements of these implementations through user stories is hugely beneficial. See **Appendix** User Stories.

4.1.3.2 **System Design** | UML

As a result of the User Stories, an appropriate set of UML diagrams can be created. Depending on the process involved will largely determine the UML diagram, for example:

As a user I can register an account so that I can access the system

The above User Story could be turned into a Use Case diagram to depict their interaction with the system and how at a high level this interaction may work. From here further UML diagrams could be created to show a lower level of interaction such as a Class or Sequence diagram.

4.1.3.2 **System Design** | Activity Diagrams

Multiple Activity diagrams were created to help illustrate and provide conceptual understandings of the system and how it works. Due to the importance being highlighted throughout Chapter 2 these were designed to aid in the implementation of the system processes in which they are concerning. A breakdown of these can be seen in **Appendix** Activity Diagrams.

4.1.3.3 **System Design** | Notable System Implementations

There are a number of important system implementations that can be reviewed.

- **Search:** As highlighted previously, searching has been implemented as system functionality. This provides users with an additional way of navigating the application and further provides user interactivity with parts of the system. Providing users with interactive features was a key part of this project and searching provides users with this interactivity. See **Appendix** Searching.
- **Logging:** Logging was identified as a key component to assist admins in debugging. As such a Logger class was created to further allow any errors that were caught to be logged as well as logging user login attempts and account creations. When we consider user interaction of such a system its important, as highlighted throughout our user story mapping, that we identify admins as users of our system.
- **Content:** Content delivery has been implemented using the JavaFX HTMLEditor control. This control was chosen as it provides flexibility to the way in which we are able to present content: This is due to the control being a web based one which further allows for the implementation of languages such as JavaScript as well as mark-up languages such as HTML. This provides a document like environment whereby allowing users to add content within the editor. An addition to the HTMLEditor was implemented to further

allow for the adding of images to the applications content with research such as **[51]** illustrating how this can be performed. An event handler for the button was also implemented which further provides access to the FileChooser dialog, allowing escalated users such as system admins, to choose an image **[52]**. This then returns a File object for the image that has been selected which is then encoded into a base64 string that is passed to JavaScript and set as the image source data: Research such as **[53, 54]** was helpful in this implementation. See **Appendix** Admin Editor.

Furthermore, due to the HTMLEditor only providing access to setHTMLText and getHTMLText methods, the editor needs to extract the WebView, which is a JavaFX component that is capable of presenting web pages, to process information within the HTMLEditor. Importantly, in order to execute JavaScript functions in the HTMLEditor control we need access to the WebView node within the editor. This provides further access to the WebEngine, which is an internal component of the WebView, and use it to load content from the resources directory, execute JavaScript and perform other tasks which are not accessible directly from an HTML Editor control **[55, 56]**.

Finally, Content is extracted using JavaScript by grabbing the WebView of the editor and calling the implemented JavaScript method getCreatorData.

4.1.3.4 **System Design** | Model View Controller

As part of good system design and implementation, the Model View Controller or MVC has been considered and further applied to the system design. JavaFX applications inherently require the View and Controller to be implemented through fxml views and their respective controllers. As such a Model has also been added to enforce this as well as adhere to good system design practice. See **Appendix** MVC.

4.1.3.5 System Design | Models

A number of Java models have been designed and implemented within the application to provide proper functionality to the application. A list of these can be noted as the following:

- **Database:** Database model to provide database functionality
- **UserSession:** UserSession model to provide session management
- **Course:** Course model to provide course functionality
- **Module:** Module model to provide module functionality
- **Dialogs:** Dialogs model to provide dialog support
- **Content:** Content model to provide content functionality
- **DocumentEditor:** DocumentEditor model to provide support to document retrieval
- **Logfiles:** Logfiles model to provide logging functionality
- **User:** User model to provide user functionality

The above can be expanded in much more detail, however this shall look to be expanded within the JavaDoc commenting throughout the applications source code.

4.1.4 Database Design | An Introduction

Throughout Chapter 2 we discussed Databases, Relational Vs Non-Relational and what RDBMS choices there are for us to design and build any required database as part of the proposed system. Using this research, it was decided that the best approach to a suitable database model for the proposed application would be that of a Relational Database. NoSQL type models focus on large datasets of unorganised information and as already noted relational models ensure strict relationships between entities and is otherwise a good choice for the project.

4.1.4.1 **Database Design & Implementation** | Entity Relationship Diagram

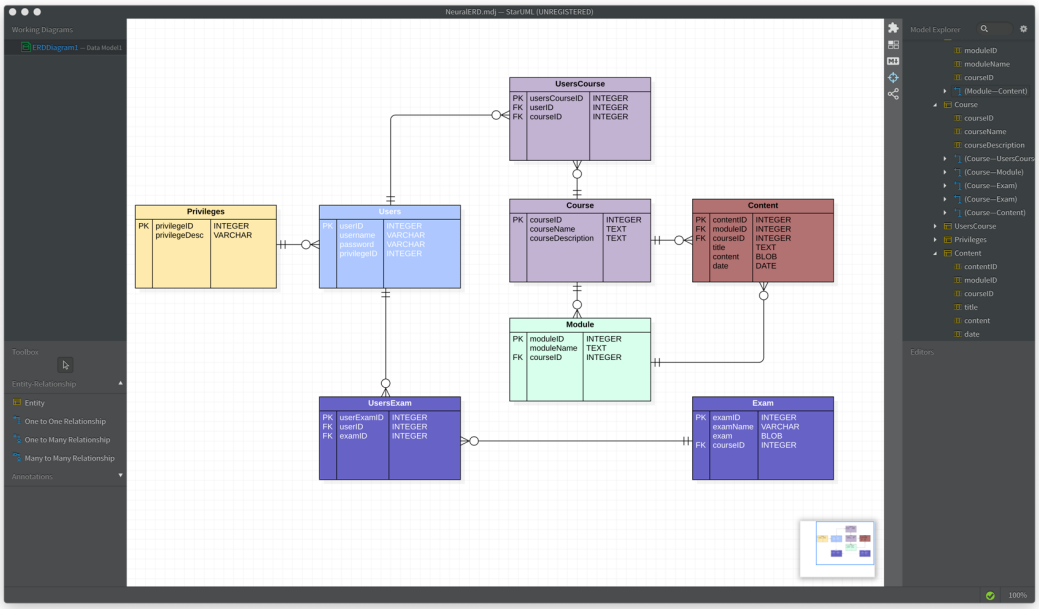


Figure 26: ERD

Above illustrates the database design that has been created to accommodate the applications information stores. We can see this implementation using DBBrowser:

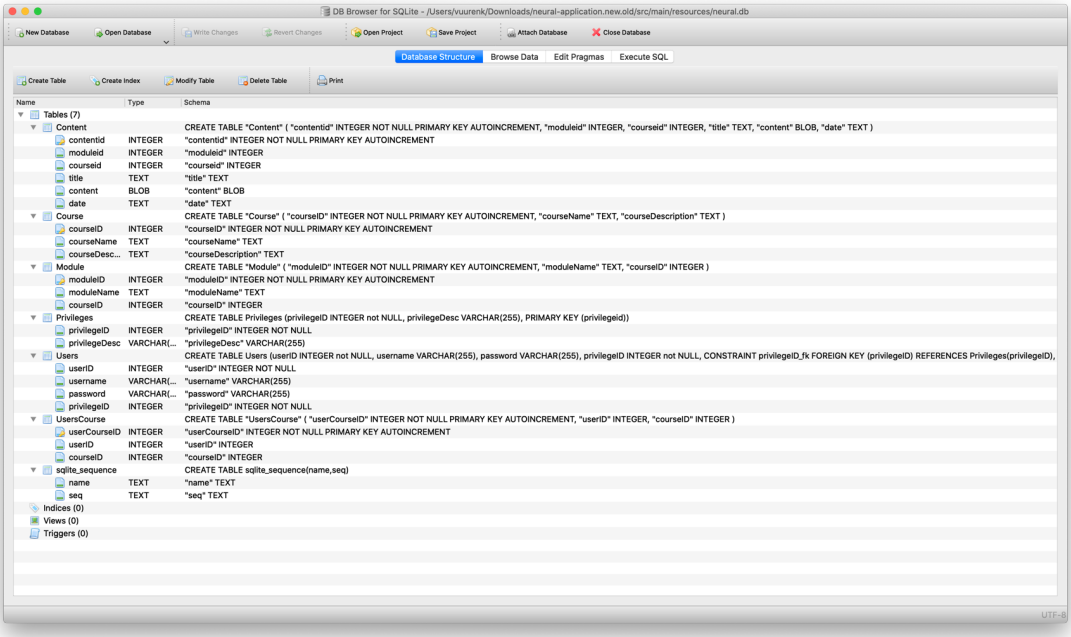


Figure 27:SQLite DBBrowser Project DB

After a process of normalisation an ERD could be created which provided a conceptual understanding of the database needed for the proposed application. Once this had been created it was a simple process of creating the database. This process was performed using DBBrowser which is a databasing tool that allows us to write SQL which is a querying language for the manipulating of database processes. This is then otherwise stored within the project directory which allows for it to be initialised, run and manipulated whenever the application needs it.

4.1.5 **Implementing Singleton Pattern** | An Introduction

Design Patterns such as the Singleton Pattern were proposed throughout Chapter 2 as a reasonable consideration for implementation. As such implementation of such a Design Pattern has been completed. Due to research highlighting its importance and considerations made as to how this could be implemented within the application, this design pattern was used to allow for the proper use of Database objects whereby further ensuring that only one object gets created at any one time.

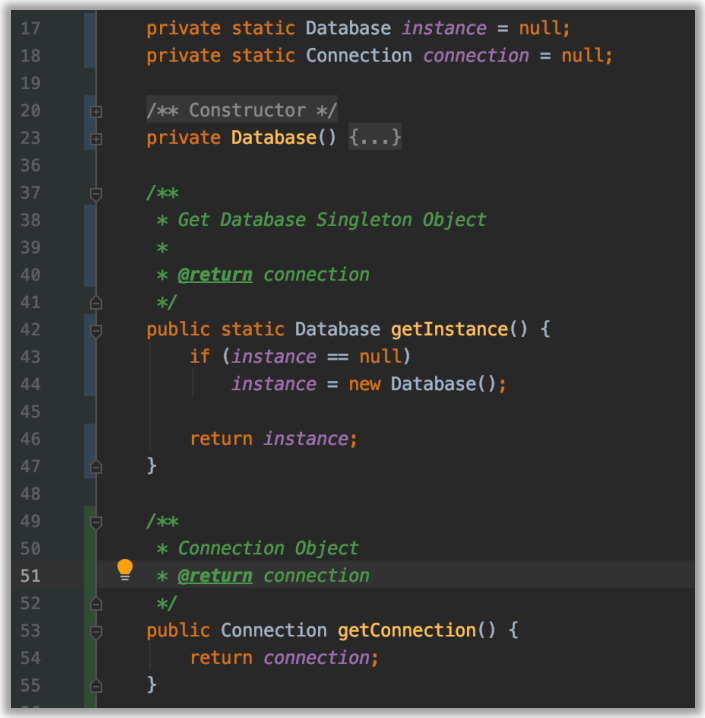


Figure 28: Database Singleton

By providing access to private attributes through a public facing method as seen above we further restrict instantiation of our Database class which has already suggested ensures only one instance of the Database class exists within our JVM. This pattern has also been used in other areas of the application where it was deemed to be suitable such as the creating of user sessions: See **Appendix** Singleton.

4.1.6 **Programming Languages** | How Languages are Implemented

Multiple programming languages have been implemented throughout the project to further allow for the author to design and develop system functionality. Java has been used to be able to provide Model and Controller design. JavaScript has been used to retrieve and pass information from the View to the Model. FXML and HTML has been used to create to create the Views while CSS has allowed us to implement further design decisions on our

View. Finally, SQL queries have been used to interrogate the database and provide the Controller and View the results that have been requested.

The use of multiple languages has allowed the author to create what he feels is a better user experience than if they were not considered.

This Section Left Intentionally Blank

5. Chapter Five

5.1 Results Summary

This section will look to analyse the results collected throughout the duration of the project. The hypothesis or $\boldsymbol{H_1}$ put forward by this project states that:

"Due to learners having multiple preferences to learning materials and how these materials are presented, multi-modal applications are key to providing a complete educational experience".

As such we need to recognise that our $\boldsymbol{H_0}$ would be that multi-modal applications play no part in enhancing educational experiences.

5.1.1 **Results Summary |** User Feedback

Upon completion of the project, participants were given the chance to use the application and were and were consequently questioned about their experience. The following outlines the questions asked and how their answers impact the success of the project and the accuracy of the hypothesis presented.

5.1.1.1 **Results Summary |** Have you used a similar tool

Users were asked if they had used a similar tool to the proposed application before. It was hoped that this would give a good understanding as to the users experience with educational applications. The results suggest:

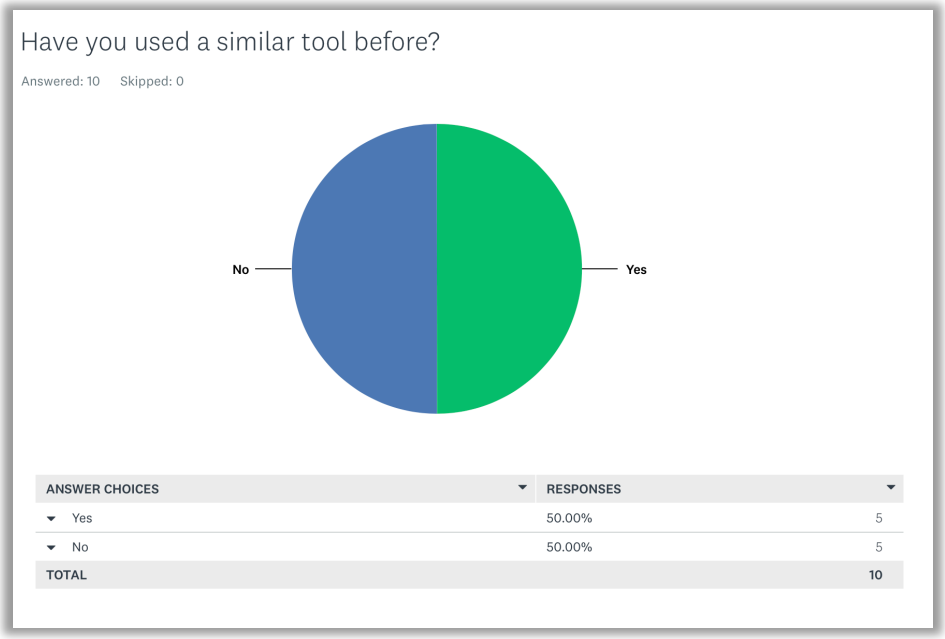


Figure 29: Survey

As we can see out of a total of 10 participants, an equal weighting separated those who had and those who hadn't had experience with a similar tool. This does further suggest that any data collected thereafter shouldn't be skewed or misleading as we have an even number of participants who have and haven't used similar applications.

5.1.1.2 **Results Summary** | Background in I.T.

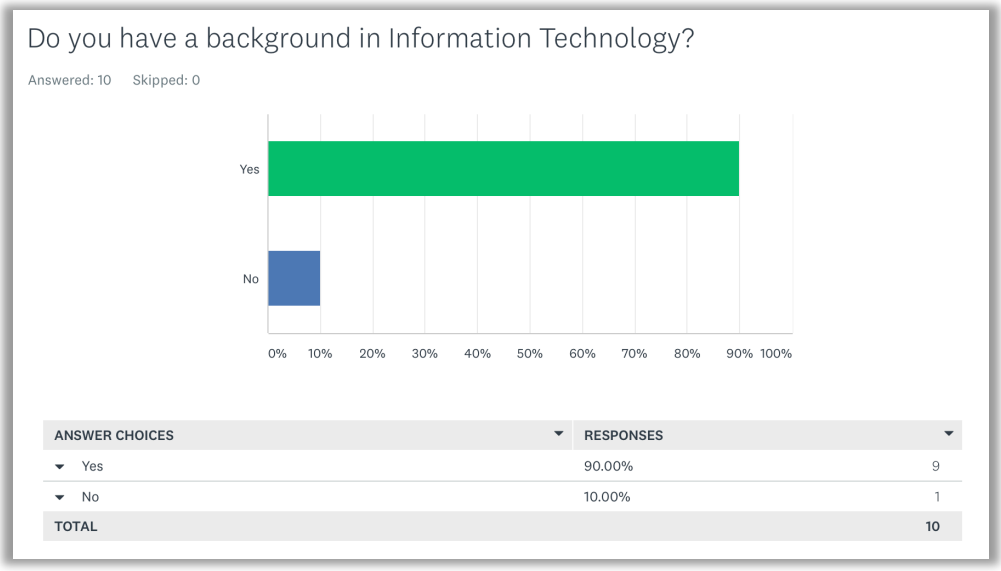
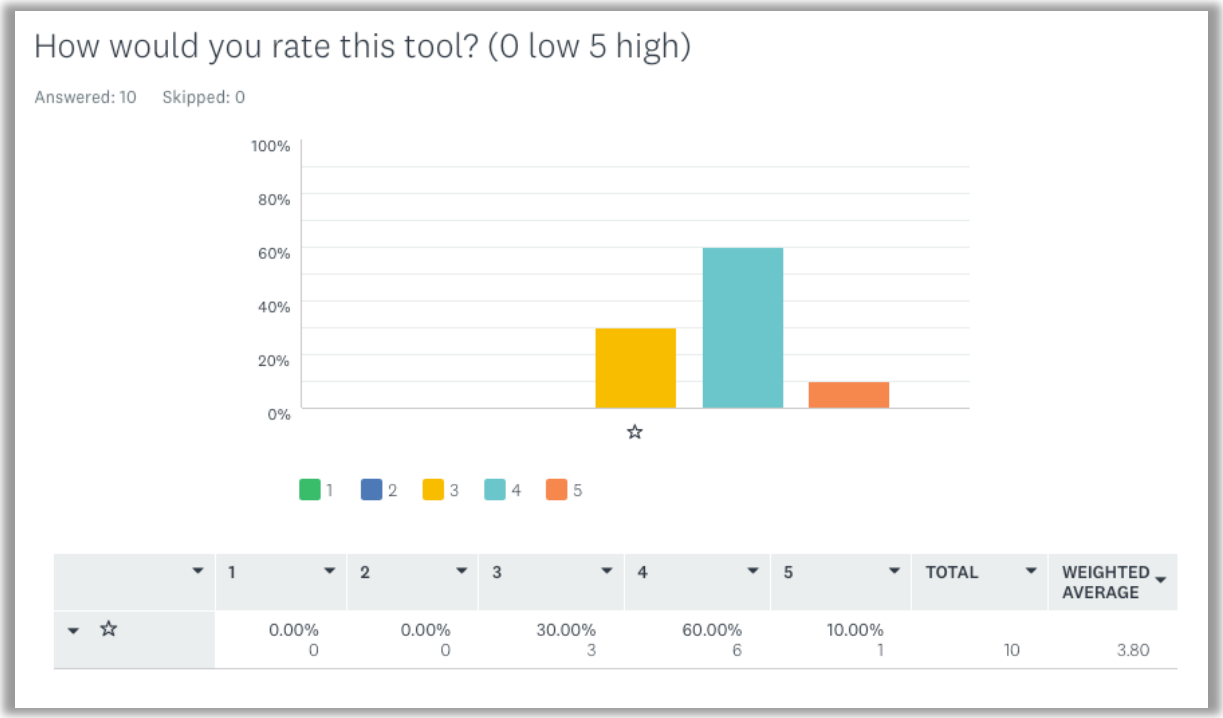


Figure 30: Survey

We also wanted to present and find an answer to users experience in Information Technology. The above allowed us to do just that and we found that 90% of participants did have prior experience within said fields. This is important as it signifies that users may have deeper expectations from a tool such as the proposed.

After understanding users experience in such fields and technologies it was important to understand how they rated the proposed application.

5.1.1.3 **Results Summary** | How would you rate this tool



	1	2	3	4	5	TOTAL	WEIGHTED AVERAGE
☆	0.00% 0	0.00% 0	30.00% 3	60.00% 6	10.00% 1	10	3.80

Figure 31: Survey

The above allows us to further determine a measure of central tendency within our data set by determining the Mean, Median and Mode.

Mean	Median	Mode
3.8	4	4

Figure 32: Central Tendency

The mean suggests that the application scored a 3.8 average. The median which tells us the value that separates the higher half of our data set from the lower half or simply the middle value has come out as 4. Finally, the score that was most chosen by the applications users or the mode has come out as 4.

This suggests that the project was successful in producing an application that could help learners throughout the education. However, it does also suggest that there is room for improvement, something in which will be the topic of discussion in later chapters.

5.1.1.4 **Results Summary** | Multi-Modal Importance

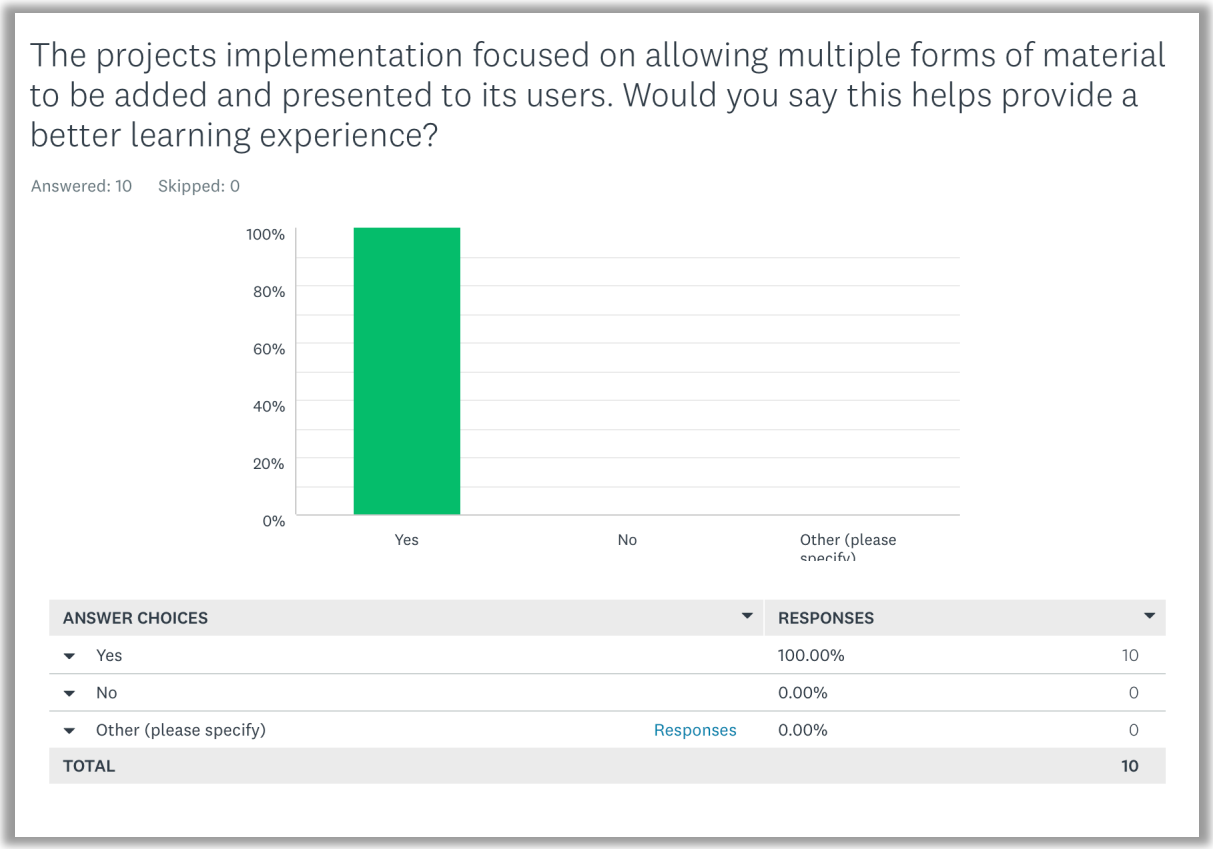


Figure 33: Survey

Finally, and most importantly we wanted to finally determine whether or not our $\boldsymbol{H_1}$ hypothesis was correct, and we can dismiss our $\boldsymbol{H_0}$. The above certainly suggests along with the other data collected, that we can accept the hypothesis put forward by this project.

See **Appendix** User Feedback.

5.2 Conclusion

This project set out to highlight the importance of multi-modal educational applications by further hypothesising that Multi-Modal educational applications are key to producing better learning experiences. Surveys further allowed us to determine whether this hypothesis was correct.

Importantly, we need to recognise that there are two important points to concluding this project. Firstly, we need to ask ourselves if the project has been successful: To determine this we can make reference to the user feedback that was collected. Success in relation to this project is determined in part by how highly users rate the tool and whether or not they think the tool could help with their learning. Both of these have shown to produce positive results with the tool ranking successfully and feedback suggesting that all participants feel like this tool can help them in their education.

Secondly, we needed to provide evidence that the hypothesis put forward by this project was indeed correct and supports validity. To do this, again we put forward a survey to participants to gather important data that could either prove or disprove the hypothesis. All participants said that they felt like a Multi-Modal application is key to providing a better learning experience: This further allows us to reject our $\boldsymbol{H_0}$ and rather accept the $\boldsymbol{H_1}$.

Reviewing both key points allows the author to conclude that the project has been successful and although there are many more considerations that can be made when looking to improve the project, as will be highlighted throughout Chapter 6, the MVP project proposal that was put forward during the early stages of development, has been successful in allowing us to accept the hypothesis put forward as part of this project.

6. Chapter Six

6.1 Future Considerations

This project intended to produce an educational application that acted as a framework for the adding, removing and editing of educational material. In order to achieve this goal, an MVP was proposed that consisted on producing the above. There have been considerations made as part of this framework that further consider any future works desired as part of this project and as such this section will look to identify these.

6.1.1 Future Considerations | Social Media

As part of producing an experience that assisted the learner in as many ways possible, the application was created so that this consideration was highlighted to the learners. It is unreasonable to expect every bit of information surrounding a certain topic to be housed within that of the educational tool. As such it was hoped that the addition of social media would further allow for educators to aid learners by providing further information such as books and other supportive material.

6.1.2 Future Considerations | Dashboard

It was hoped to implement a dashboard that would further be presented to the user upon their login. JavaFX provides a host of drawing tools and diagrams such as charts that can further be used to present data statistics visually. It is thought that presenting a user with graph data showing for example, how many topics they have completed will help them when returning to the application to summarise quickly what they have already covered.

6.1.3 **Future Considerations |** Exams

An important topic of consideration and something that would be required in future considerations is that of exams. Exams play a vital role in determining if a user has grasped the undertaken topic. Further allowing for this would support Kolb's Experiential Learning style as it allows users to take the chance to reflect on what they have learned.

6.1.4 **Future Considerations |** Web Component

A further consideration that was considered to be part of the MVP is that of a web facing component such as a website. Providing this would further provide users of the application with another avenue of pursuing their education. This may also provide a more fleshed out version of the application as hosting a web facing application would allow for more consideration towards the materials stored, such as videos and audio that require relatively large amounts of storage, something that is unrealistic to do as part of a desktop application.

6.1.5 **Future Considerations |** User Input

Further considerations towards that of improvements and future considerations were presented to participants. Below highlights a few of the suggestions that were put forward by users of the application.

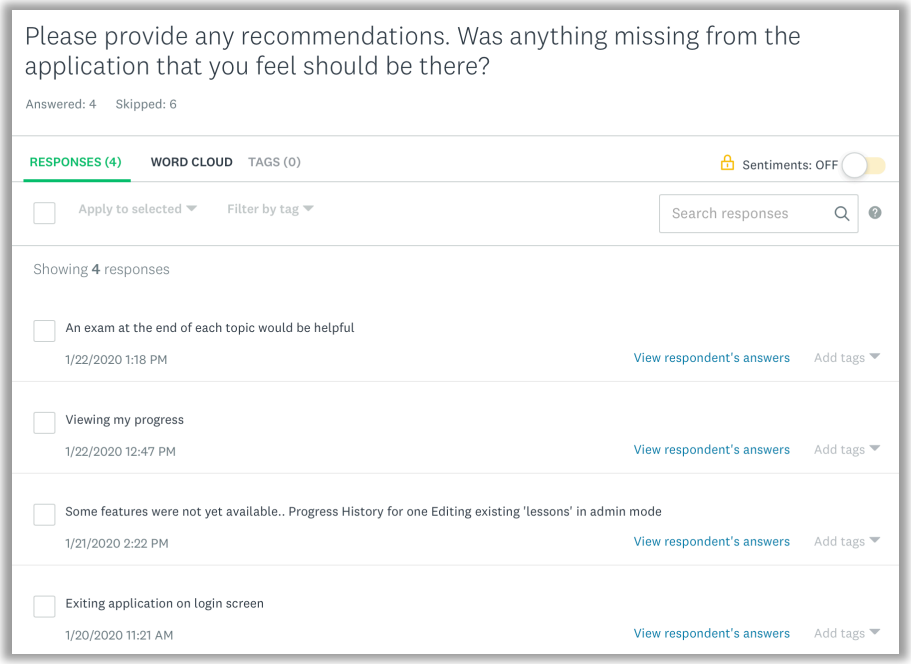


Figure 34: Future Considerations

One suggestion of the content not being editable by Admin users of the application is an example of a future consideration that was fixed before release.

This Section Left Intentionally Blank

7. Chapter Seven

7.1 Reflection

This project has helped in furthering my knowledge on a great deal of things. This section will look to highlight that what went well and things that did not.

7.1.1 **Reflection** | Positives | What Went Well

Firstly, let's consider the positives that have come out of completing such a project.

7.1.1.1 **Reflection** | JavaFX | FXML | SceneBuilder

Throughout the project I have learned a great deal about JavaFX and FXML. Before starting this project, I had little to no experience or knowledge of how to design and develop a visually appealing tool that further allowed for a user to interact with these visual components so that the tool could further be used. Understanding JavaFX applications and the use of FXML along with tools such as SceneBuilder has provided me with a solid grounding and understanding of such technologies which I am sure will allow me to further develop myself in similar areas in the future.

7.1.1.2 **Reflection** | Agile Principles

Having created my own sprint board which allowed for my user story mappings gave me a great practical insight into how we can take epics and deduce them down into smaller manageable pieces of work. Due to working in an Agile environment I wanted to do this so that I could attain a better understanding of these processes and is something I am sure will help in the future.

7.1.2 **Reflection |** Negatives | What Did Not Go Well

Now, let's consider some of the negatives encountered during working on the project.

7.1.2.1 **Reflection |** Time | Planning

Now that the project is complete, once reflected upon, planning is one topic that could have been done better. I spent a lot of time designing multiple user interfaces and could never find myself happy with any created. This ultimately led to me loosing time on work that could have been done to other parts of the application. However, this is something that can be expected due to the nature of the project and the picking up of new tools and technologies: It took me a good amount of time to first understand how one uses FXML along with SceneBuilder to create user interfaces and one understood I was then left with trying to design something that I felt was appropriate.

This Section Left Intentionally Blank

8. Chapter Eight

8.1 Reference List

[1] W. Horton, *E-Learning by Design*, San Francisco: Wiley, 2006.

[2] Cortland, "Modalities" Modalities, March. 2, 2004. [Online] Available: https://web.cortland.edu/andersmd/learning/Modalities.htm. [Accessed Jan. 9, 2019].

[3] H. Sarka. "History and Perspectives of Elearning." *Social and Behavioural Sciences* vol. 191, June 2015.

[4] N. Fleming, C. Bonwell, *VARK How Do I learn Best*. [E-book].

[5] R. Hemant, Rajiv, Lal. Manohar. "E-learning: Issues and Challenges." *International Journal of Computer Applications* vol. 97, July 2014.

[6] B. Little, "Perspective on the e-learning industry" *TJ Training Journal*, Sep 2014. [Online]. Available: https://www.trainingjournal.com/articles/feature/perspective-e-learning-industry. [Accessed May. 25, 2019].

[7] J. Biggam, *Succeeding with your Master's Dissertation*, Glasgow: Open University Press, 2008.

[8] M. Saunders, P. Lewis & A. Thornhill, *Research Methods for Business Students*, 6th Edition, Harlow: Pearson, 2019.

[9] D. Benami, "THE NATURE, THE BEAUTY AND THE DIFFICULTY IN JAVA PROGRAMMING." *Studia Universitatis Moldaviae: Stiinte Exacte Si Economice* vol. 97, 2016.

[10] Simply Psychology "Kolb's Learning Styles and Experiential Learning Cycle" *simplypsychology*, 2017. [Online] Available: https://www.simplypsychology.org/learning-kolb.html [Accessed Sep. 20, 2019].

[11] D. Kolb, *Experiential Learning: Experience as The Source Of Learning And Development*, New Jersey: Pearson, 1984.

[12] Learn to Code with me "Cross Platform Apps" *learntocodewithme*, 2019. [Online] Available: https://learntocodewith.me/posts/cross-platform-apps/ [Accessed Sep. 20, 2019].

[13] 99 Firms "Mobile Marketing Statistics" *99firms*, 2019. [Online] Available: https://99firms.com/blog/mobile-marketing-statistics/ [Accessed Sep. 23, 2019].

[14] CMS Gov, "Selecting Development Approach" *CMS Gov*, 2019. [Online] Available: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf [Accessed May. 26, 2019].

[15] Career Drill, "Software Development Lifecycle" *Career Drill*, 2019. [Online] Available: http://careerdrill.com/blog/wp-content/uploads/2015/11/SDLC_-_Software_Development_Life_Cycle.jpg [Accessed: May. 30, 2019].

[16] Grech, T. (2015) 'The intersection of agile and waterfall', Industrial Engineer: IE, 47(8), pp. 47–49. Available: https://search-ebscohost-com.ezproxy.uwtsd.ac.uk/login.aspx?direct=true&db=bth&AN=109483551&site=ehost-live [Accessed Sep. 24, 2019].

[17] Guru 99 "Waterfall vs Agile" *guru99*, 2019. [Online] Available: https://www.guru99.com/waterfall-vs-agile.html [Accessed September. 24, 2019].

[18] Tutorials Point "SDLC Waterfall Model" *tutorialspoint*, 2019. [Online] Available: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm [Accessed Sep. 24, 2019].

[19] N. Ionel, "AGILE SOFTWARE DEVELOPMENT METHODOLOGIES: AN OVERVIEW OF THE CURRENT STATE OF RESEARCH." *Annals of the University of Oradea: Economic Science* vol. 4.1, p. 381-85, 2009.

[20] D. Kolb, (1984). *Experiential Learning: Experience as The Source of Learning and Development*, New Jersey: Prentice Hall, 1984.

[21] R. Baishakhi, D. Posnett, P. Devanbu, V. Filkov, "A Large-scale Study of Programming Languages and Code Quality in GitHub." *Communications of the ACM* vol. 60, p. 91-100, 2017.

[22] M. R. V. Chaudron, V. Heijstek, A. Nugroho, "How Effective Is UML Modelling?" *Software & Systems Modelling*, vol. 11, no 4, p. 571-80, October 2012.

[23] P. Stevens, R. Pooley, *Using UML: Software Engineering with Objects and Components* 2nd Edition, Harlow: Addison-Wesley, 2006.

[23] J. Garbajosa, X. Wang, A. Aguiar, "Agile Processes in Software Engineering and Extreme Programming" in *Business Information Processing*, Proceedings of the 19th Int. Conference, XP 2018, Porto, Portugal, Springer, 2018.

[24] J. L. Harrington, *Relational Database Design and Implementation: Clearly Explained*. 3rd Edition, Cambridge: Elsevier Science, 2009.

[25] D. Goodman, *JavaScript Bible* 4th Edition, New York: Hungry Minds, 2001.

[26] K, Abdullah. "Using Kolb's Experiential Learning Cycle to Improve Student Learning in Virtual Computer Laboratories." *Computers & Education*, vol. 72, March 2014.

[27] D. Pilone, R. Miles, *Head First Software Development* 1st Edition, Sebastopol: O'Reilly, 2008.

[28] J. Shore, S. Warden, *The Art of Agile Development*, Sebastopol: O'Reilly, 2007.

[29] M. Cohn, *User Stories Applied for Agile Software Development*, Boston: Pearson, 2008.

[30] Spark X Systems "UML Tutorial" *sparkxsystems*, 2020. [Online] Available: https://sparxsystems.com/resources/tutorials/uml/part1.html [Accessed Jan. 01, 2020].

[31] H. Schildt, *Java a Beginners Guide* 6th Edition, New York: McGraw-Hill, 2014.

[32] D. J. Eck, *Introduction to Programming Using Java*, New York: Hobart and William Smith Colleges, 2014.

[33] J. Pokorný, "Integration of Relational and NoSQL Databases." *Vietnam Journal of Computer Science* vol. 6, July 2019.

[34] MongoDB "NoSQL vs Relational Databases" *mongodb*, 2019. [Online] Available: https://www.mongodb.com/scale/nosql-vs-relational-databases [Accessed Dec. 17, 2019].

[35] L. Stephen, "Requirements Gathering: The Journey." *Journal of Decision Systems*, vol. 25, pp. 302–313, 2016.

[36] H. Gould, *Database Design and Implementation: A Practical Introduction Using Oracle SQL*, 1st Edition, Bookboon: 2013.

[37] R. C. Martin, *Designing Object Oriented C++ Applications Using the Booch Method*, New Jersey: Alan Apt, 1995.

[38] StackOverflow "What Does Low Coupling and High Cohesion Mean" *stackoverflow*, 2020. [Online] Available: https://stackoverflow.com/questions/14000762/what-does-low-in-coupling-and-high-in-cohesion-mean [Accessed Dec. 19, 2019].

[39] Coursera "Coupling and Cohesion" *coursera*, 2019. [Online] Available: https://www.coursera.org/lecture/object-oriented-design/1-3-1-coupling-and-cohesion-q8wGt [Accessed Dec. 19, 2019].

[40] H-E. Eriksson, M. Penker, *Business Modelling with UML Business Patterns at Work*, Canada: John Wiley & Sons, Inc, 2000.

[41] D. Pilone, R. Miles, *Head First Software Development*, California: O'Reilly, 2008.

[43] S. Bennett, S. McRobb, R. Farmer, *Object-Oriented Systems Analysis and Design Using UML*, Berkshire: McGraw-Hill Education, 2006.

[44] J. Potts, N. Hildebrandt, J. Gordon, C. Castillo, *JavaFX Getting Started*, Oracle, 2014.

[45] Maven "What is Maven" *maven*, 2019. [Online] Available: https://maven.apache.org/what-is-maven.html [Accessed Sep. 19, 2019].

[46] B. Charles, "Sun Seeks Mobile Edge with JavaFX", *Informationweek*, no. 1185, p.22, May 2008.

[47] G. Kril, O. Alves, L. Molinari, E. Roux, "Best Practices for Efficient Development of JavaFX Applications" *in Accelerator and Large Experimental Control Systems*, Proceedings of the 16th Int. Conference on Accelerator and Large Experimental Control Systems, Geneva, Switzerland, Barcelona, Spain: JACoW Publishing, 2017. pp. 1078-1082.

[48] J. Shore, S. Warden, *The Art of Agile Development*, Sebastopol: O'Reilly, 2007.

[49] Cardboard It "Cardboard It" *cardboard*, 2019. [Online] Available: https://app.cardboardit.com/ [Accessed Sep. 12, 2019].

[50] Reinisfischer "Screen Resolutions" *reinisfischer*, 2019. [Online] Available: https://www.reinisfischer.com/browser-wars-screen-resolutions-2019 [Accessed Oct. 07, 2019].

[51] Github "HTMLEditor Customisation Sample" *github*, 2019. [Online] Available: https://gist.github.com/jewelsea/2514709 [Accessed Nov. 25, 2019].

[52] Jencov "File Chooser" *jencov*, 2019. [Online] Available: http://tutorials.jenkov.com/javafx/filechooser.html [Accessed Nov. 25, 2019].

[53] Oracle "Base64 Encoder" *oracle*, 2019. [Online] Available: https://docs.oracle.com/javase/8/docs/api/java/util/Base64.Encoder.html [Accessed Nov. 26, 2019].

[54] Stackoverflow “How can I set image source with Base64” *stackoverflow*, 2019. [Online] Available: https://stackoverflow.com/questions/16449445/how-can-i-set-image-source-with-base64 [Accessed Nov. 26, 2019].

[55] Jencov “HTMLEditor” *jencov*, 2019. [Online] Available: http://tutorials.jenkov.com/javafx/htmleditor.html [Accessed Nov. 26, 2019].

[56] Jencov “WebView” *jencov*, 2019. [Online]. Available: http://tutorials.jenkov.com/javafx/webview.html [Accessed Nov. 27, 2019].

Reading List

https://www.javatpoint.com/sqlite-like-clause

http://tutorials.jenkov.com/javafx/tableview.html

https://www.geeksforgeeks.org/javafx-choicebox/

https://gist.github.com/jewelsea/2514709

http://tutorials.jenkov.com/javafx/filechooser.html

https://docs.oracle.com/javase/8/docs/api/java/util/Base64.Encoder.html

https://stackoverflow.com/questions/16449445/how-can-i-set-image-source-with-base64

http://tutorials.jenkov.com/javafx/htmleditor.html

http://tutorials.jenkov.com/javafx/webview.html

https://www.w3schools.com/tags/att_global_contenteditable.asp

https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelectorAll

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/replace

https://docs.oracle.com/javase/8/docs/technotes/guides/lang/resources.html

https://books.google.co.uk/books?id=zOzZ7VsQ7skC&pg=PA194&lpg=PA194&dq=java+len+%3D+resourceFile.read(buffer&source=bl&ots=K_mrD5r9cz&sig=ACfU3U1tHBdDLEx1Hm5vArjjq2yTIa9StA&hl=en&sa=X&ved=2ahUKEwiPkZjChIPnAhXUPsAKHW9uC_YQ6AEwAXoECAoQAQ

https://www.sqlitetutorial.net/sqlite-java/sqlite-jdbc-driver/

https://www.sqlitetutorial.net/sqlite-java/insert/

https://www.sqlitetutorial.net/sqlite-java/select/

https://www.tutorialspoint.com/sqlite/sqlite_update_query.htm

https://www.tutorialspoint.com/sqlite/sqlite_using_joins.htm

Appendix

User Stories

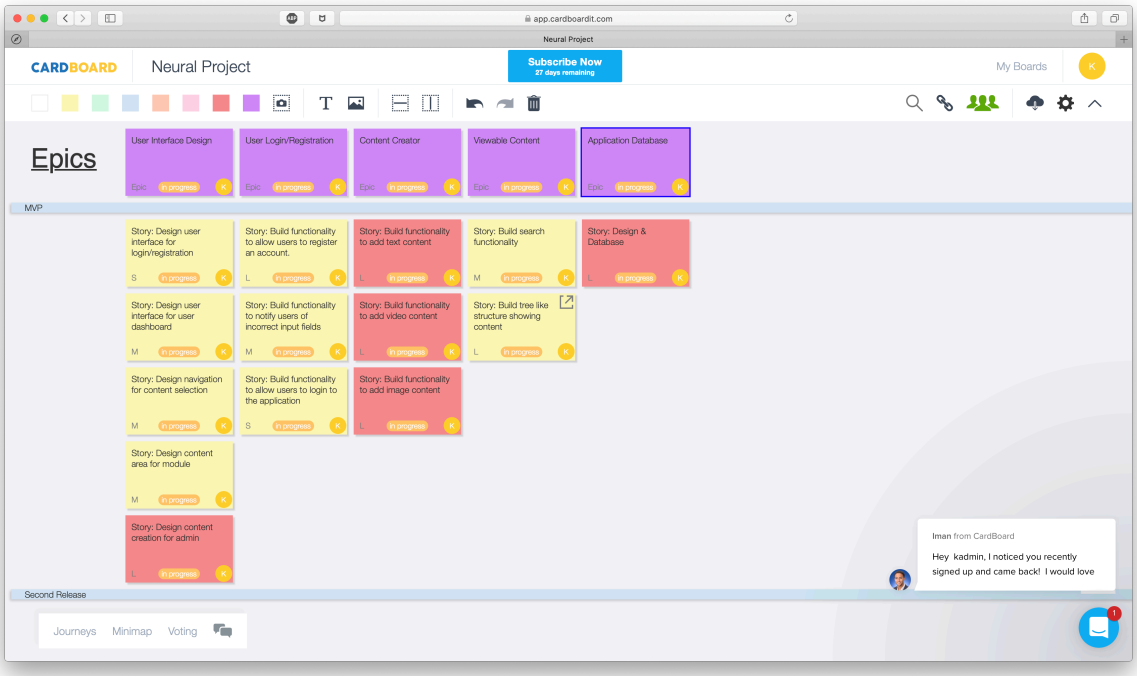


Figure 35: Appendix User Story Board

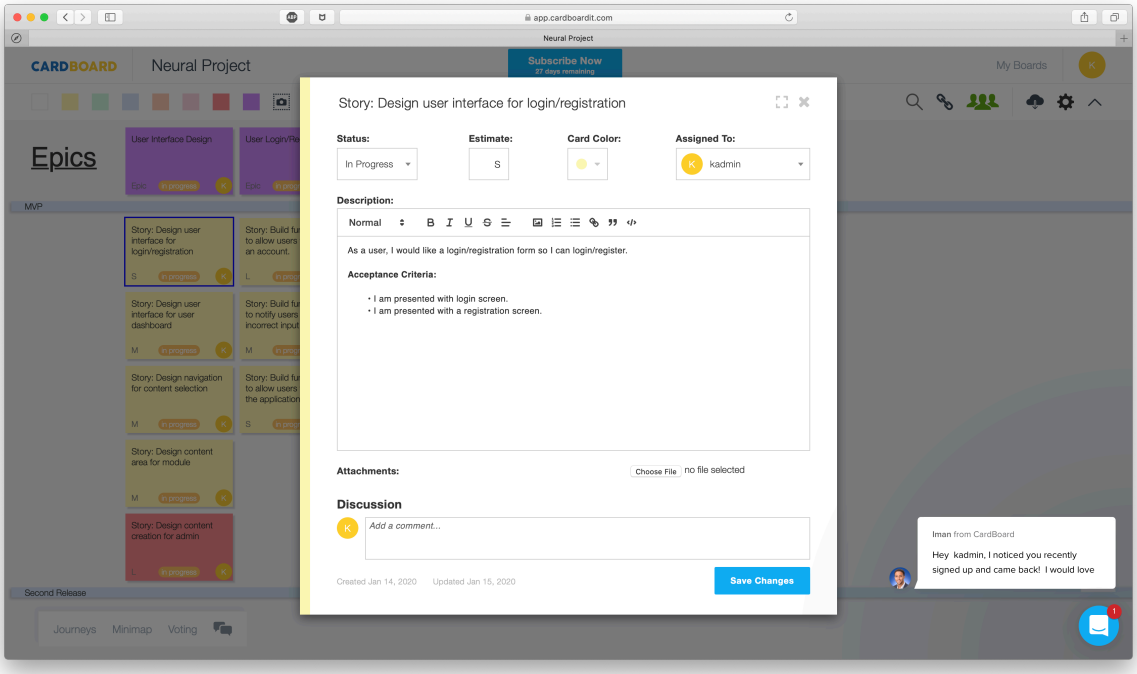


Figure 36: Appendix User Interface Epic Story

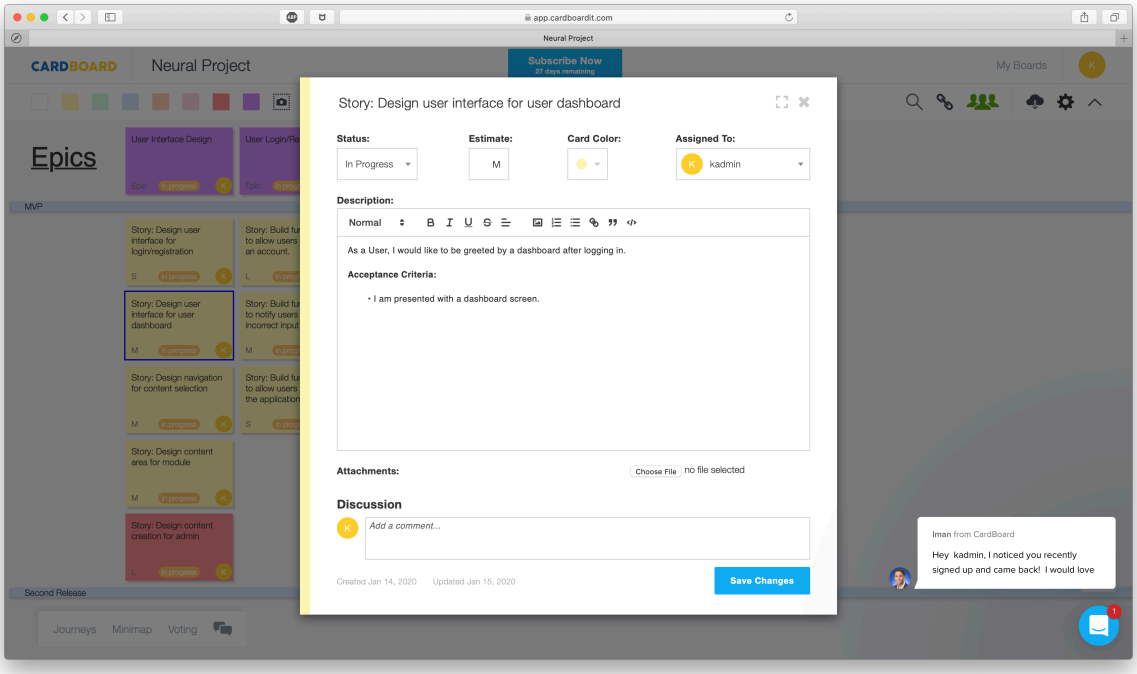


Figure 37: Appendix User Interface Epic Story

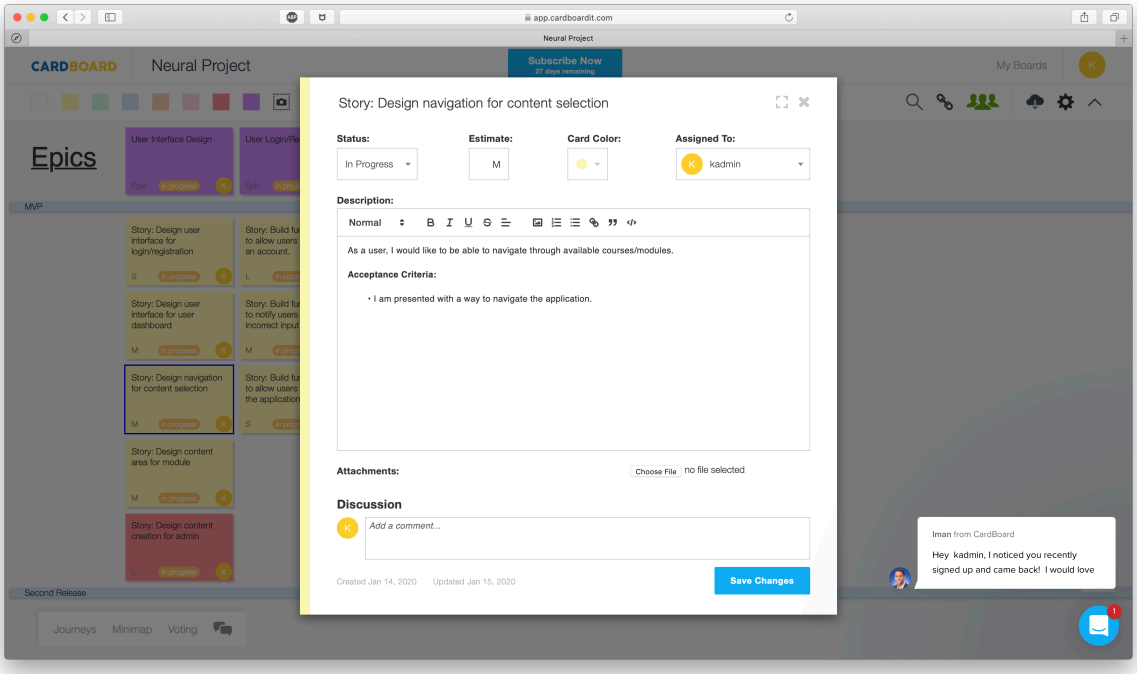


Figure 38: Appendix User Interface Epic Story

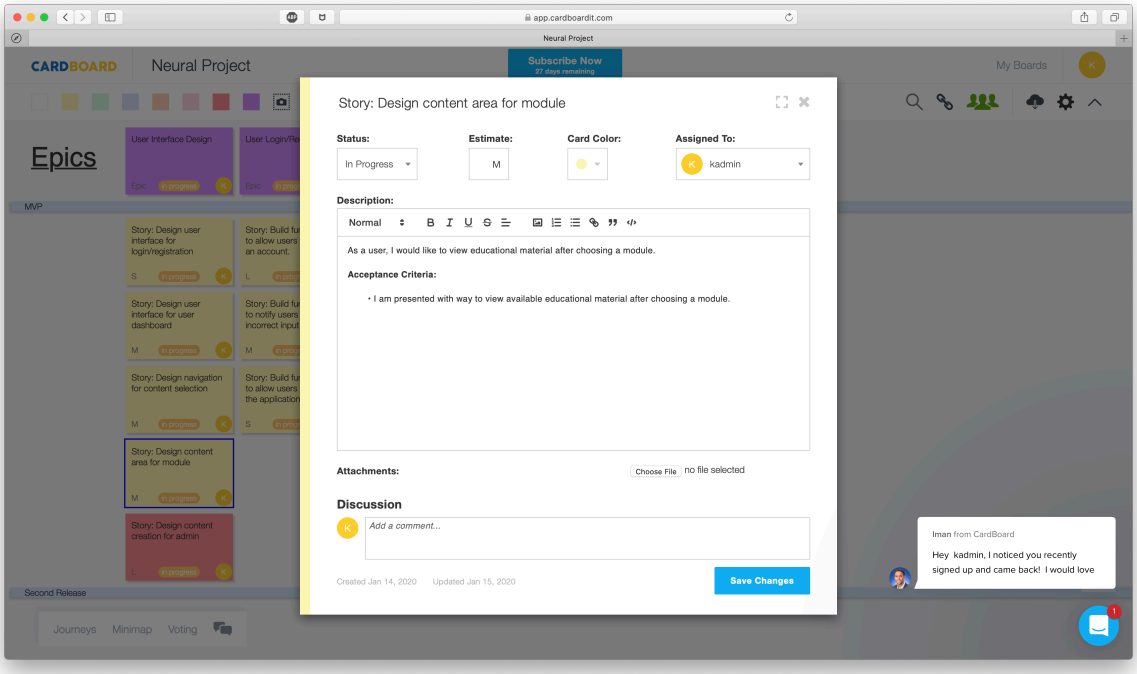


Figure 39: Appendix User Interface Epic Story

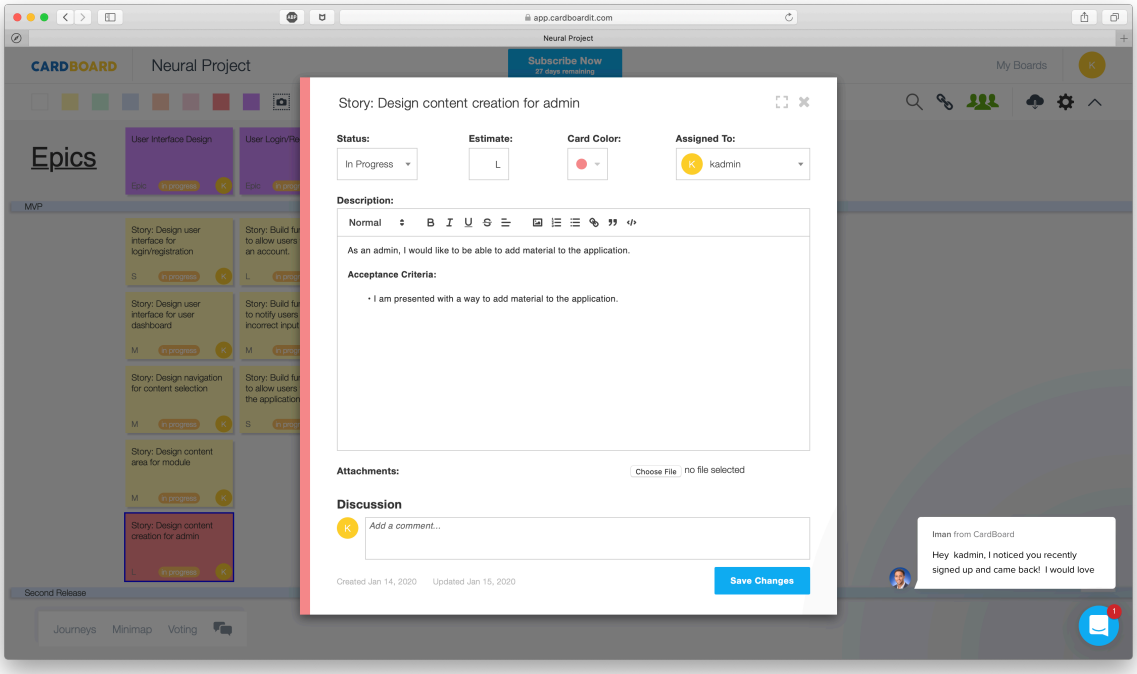


Figure 40: Appendix User Interface Epic Story

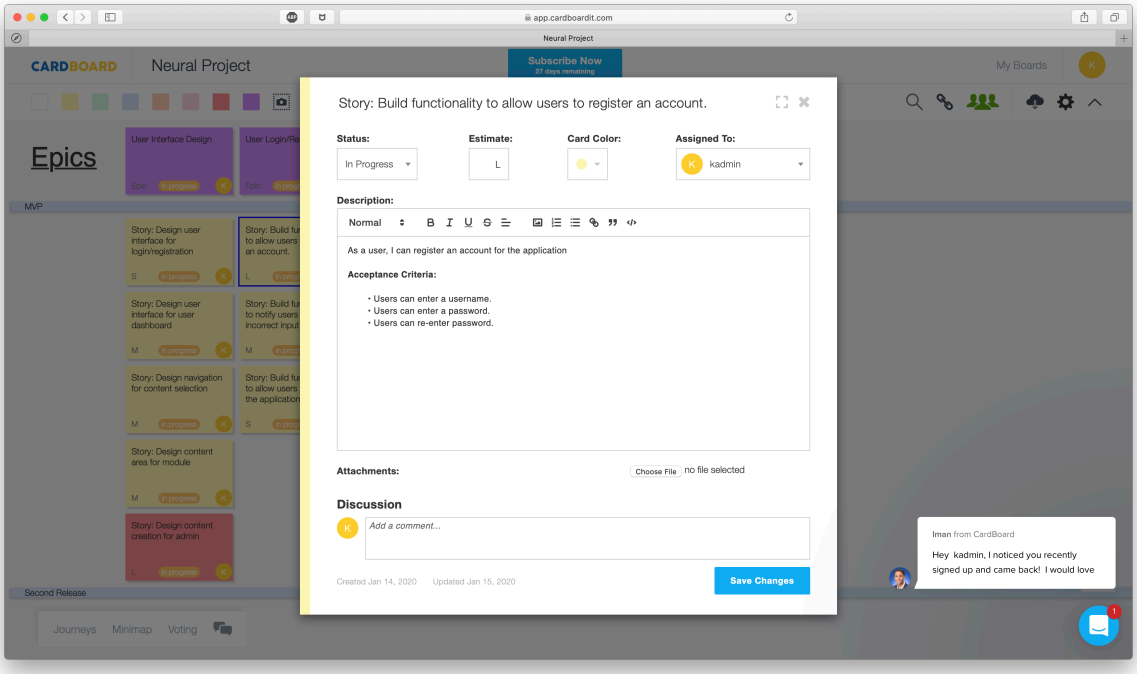


Figure 41: Appendix User Login | Registration Epic Story

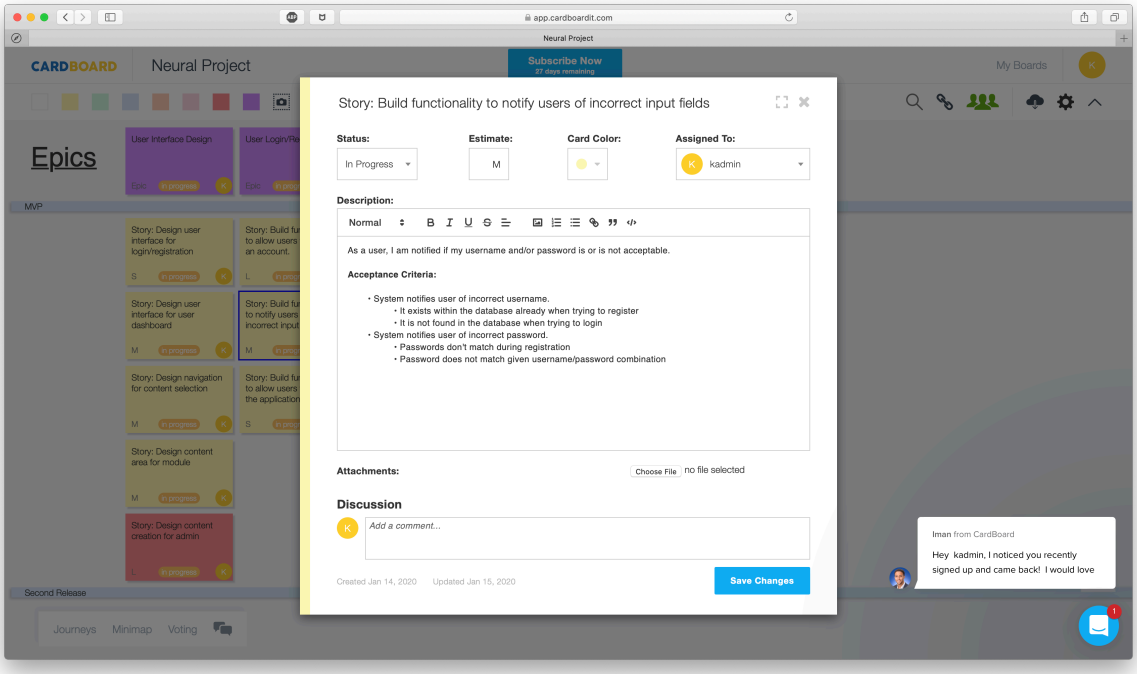


Figure 42: Appendix User Login | Registration Epic Story

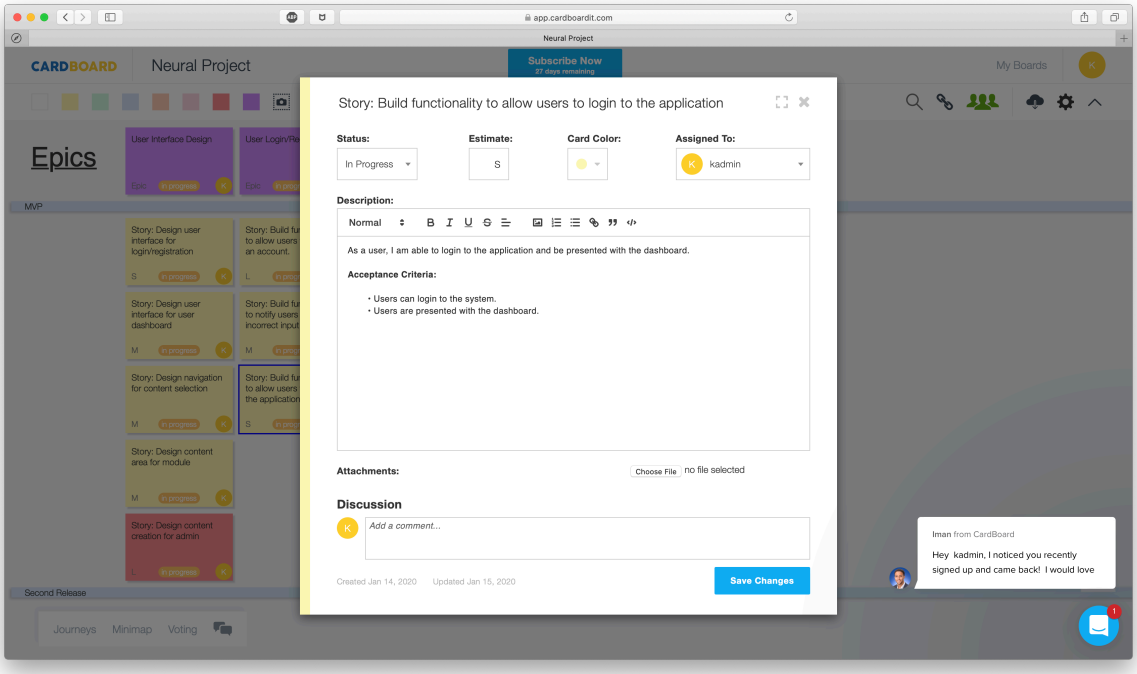


Figure 43: Appendix User Login | Registration Epic Story

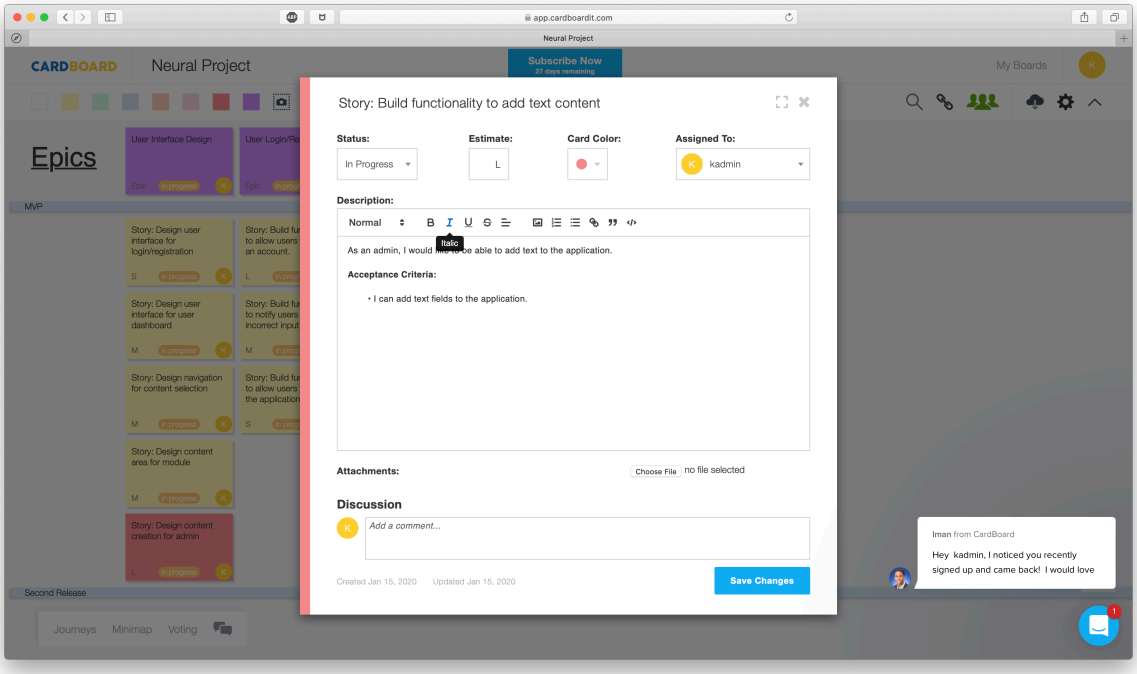


Figure 44: Appendix Content Creator Epic Story

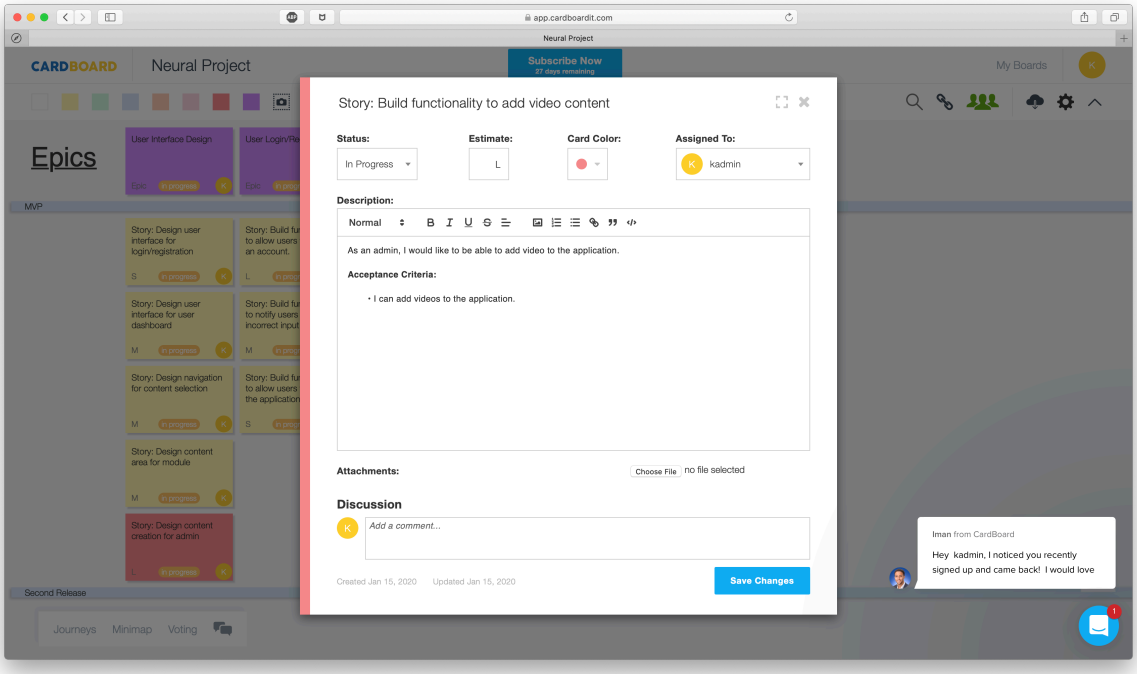


Figure 45: Appendix Content Creator Epic Story

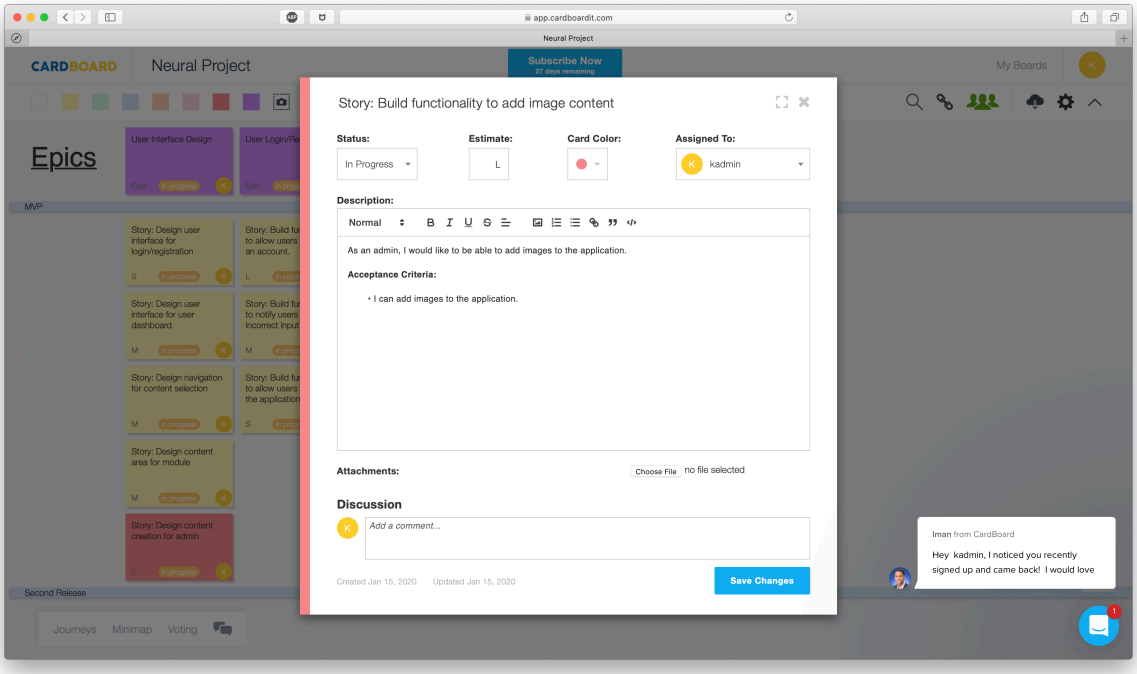


Figure 46: Appendix Content Creator Epic Story

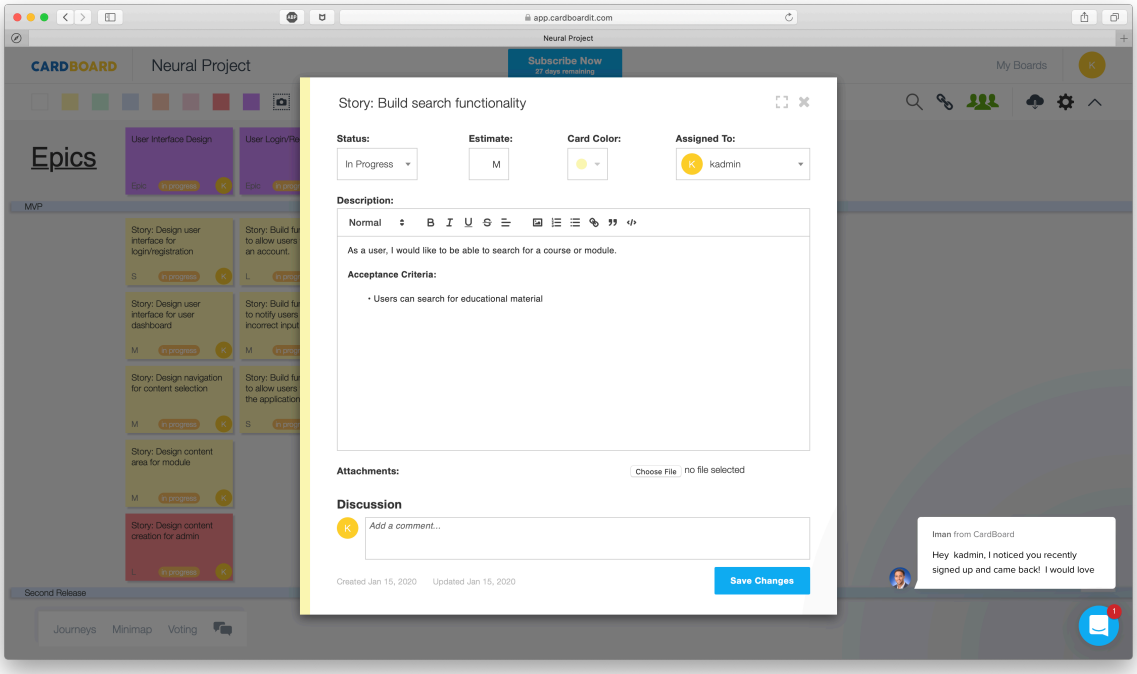


Figure 47: Appendix Viewable Content Epic Story

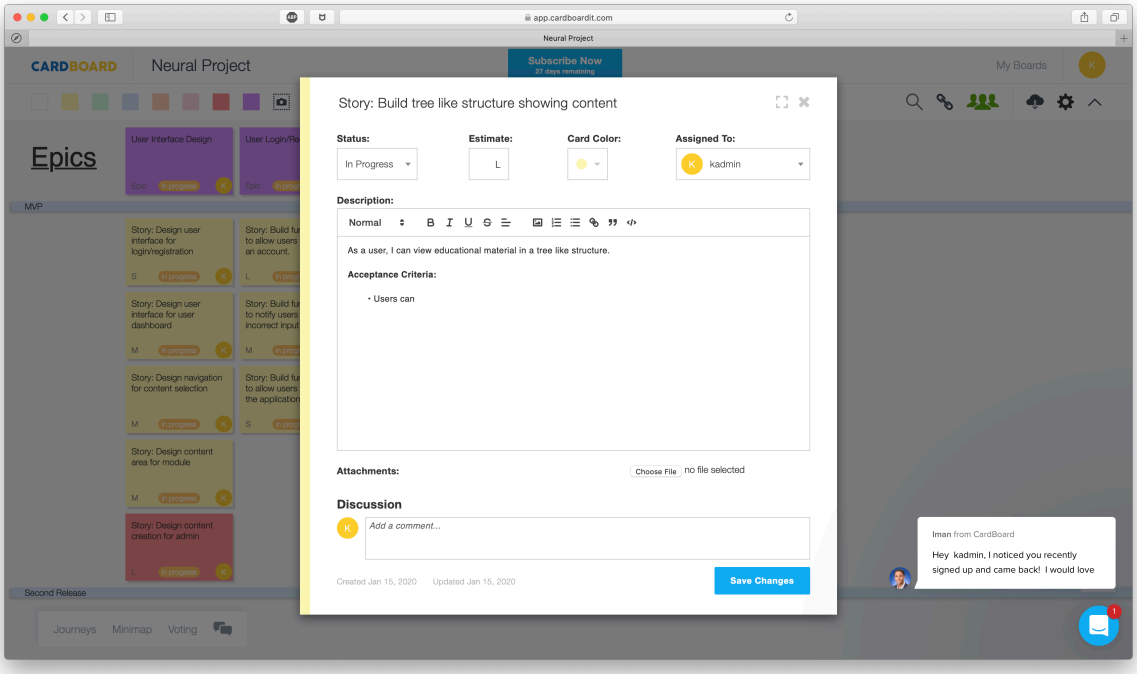


Figure 48: Appendix Viewable Content Epic Story

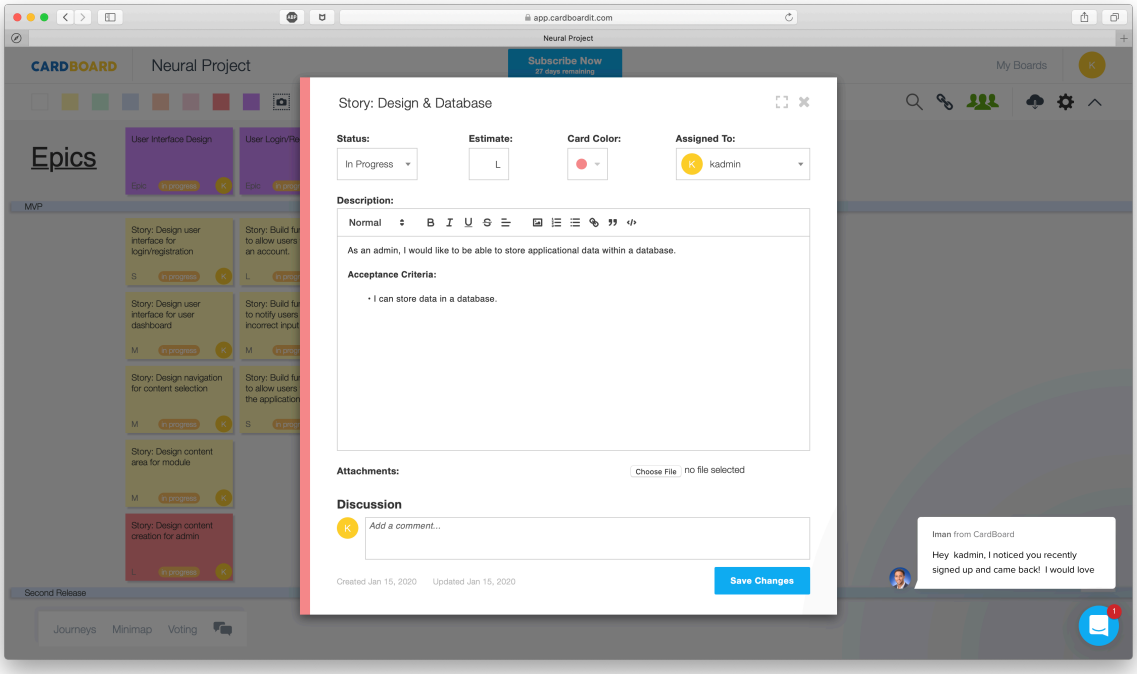


Figure 49: Appendix Database Epic Story

Singleton

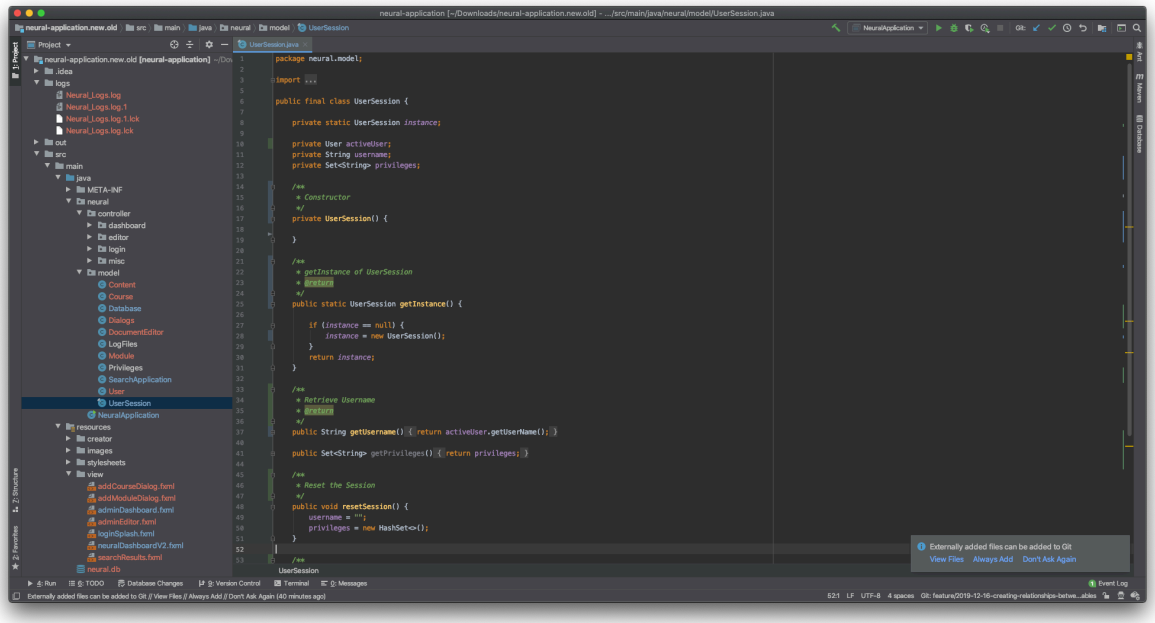


Figure 50: Appendix Singleton User Session

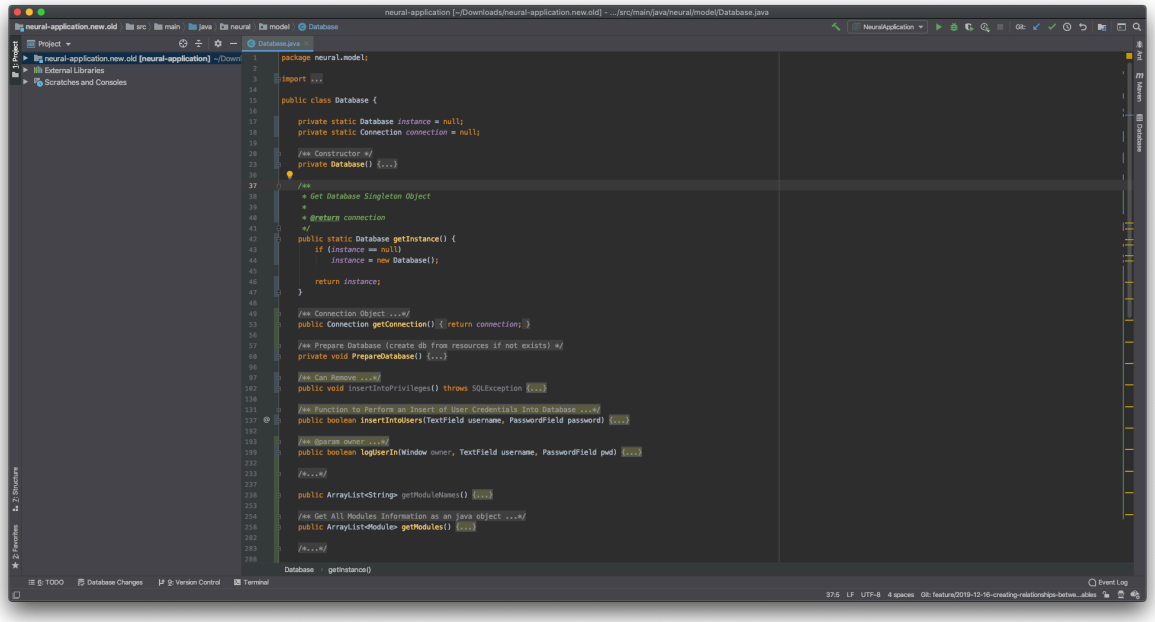


Figure 51: Appendix Singleton Database

Activity Diagrams

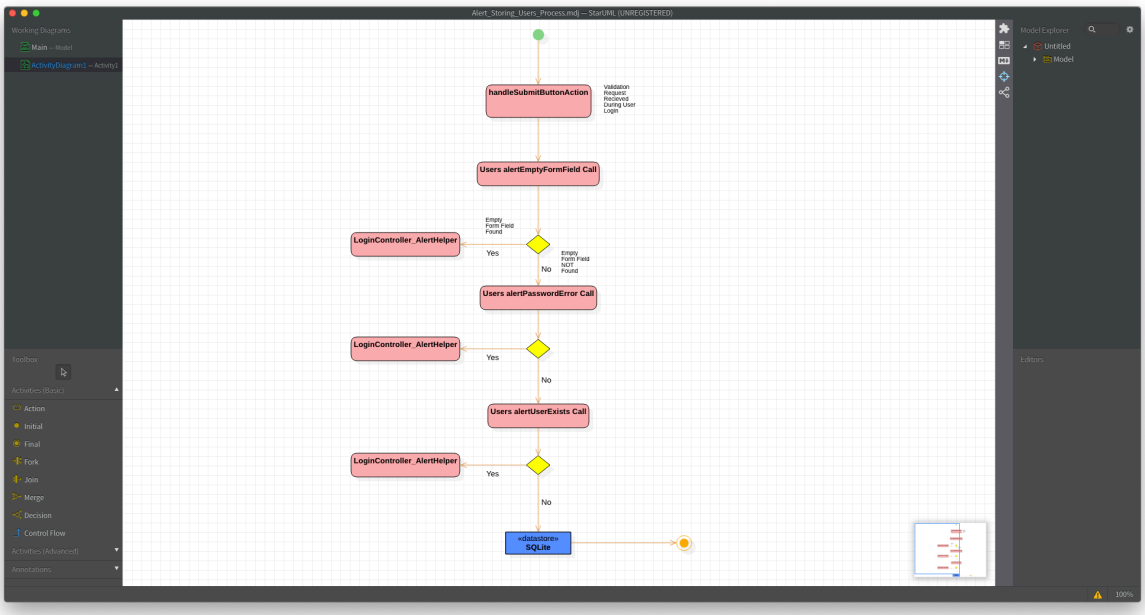


Figure 52: Appendix Activity Diagram Alerting

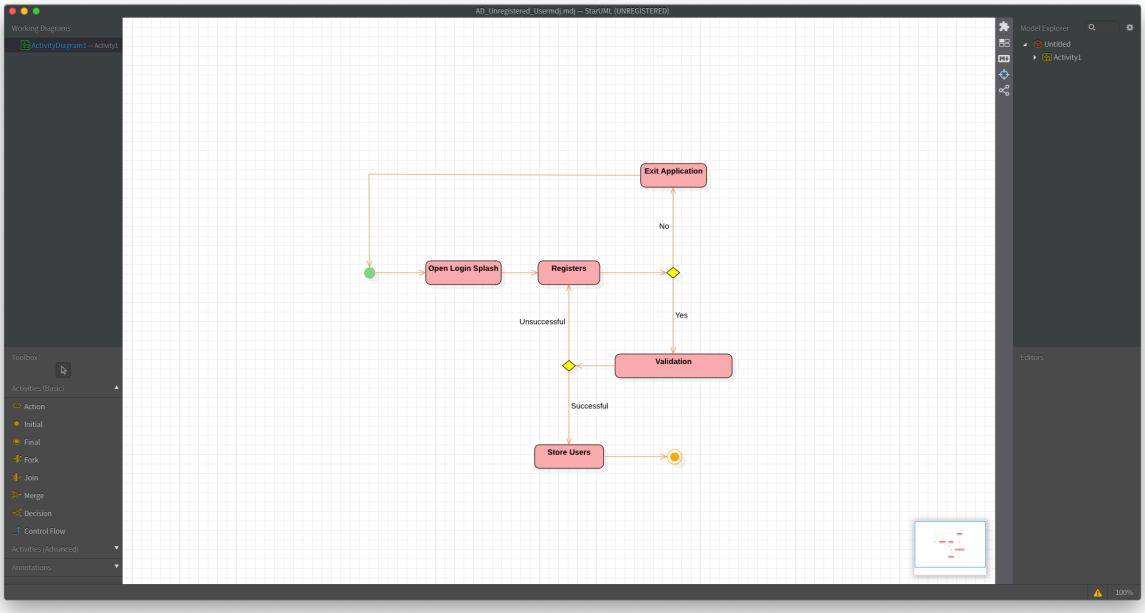


Figure 53: Appendix Activity Diagram User Registration

Searching

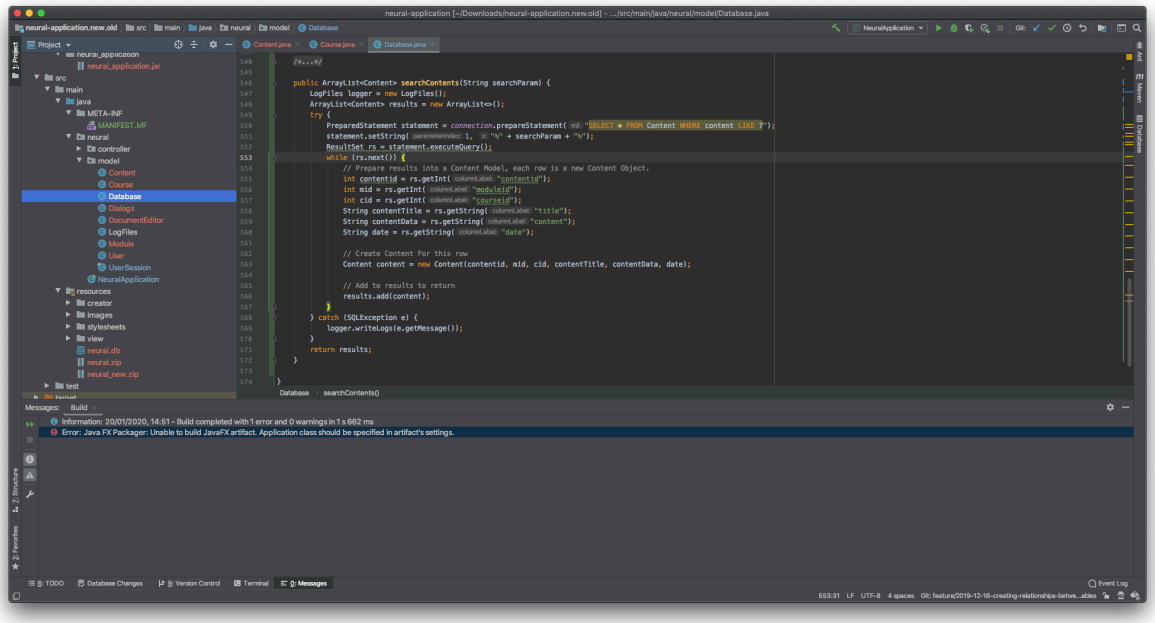


Figure 54: Searching Content

MVC



Figure 55: Appendix MVC

Admin Editor

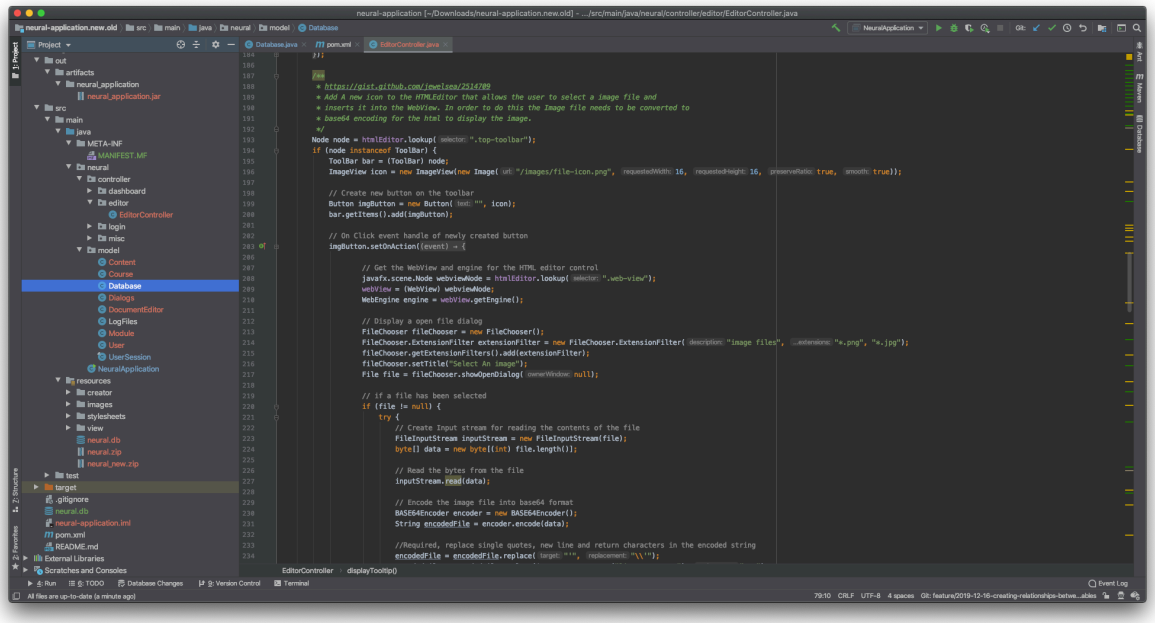


Figure 56: Appendix Admin Editor Image Button

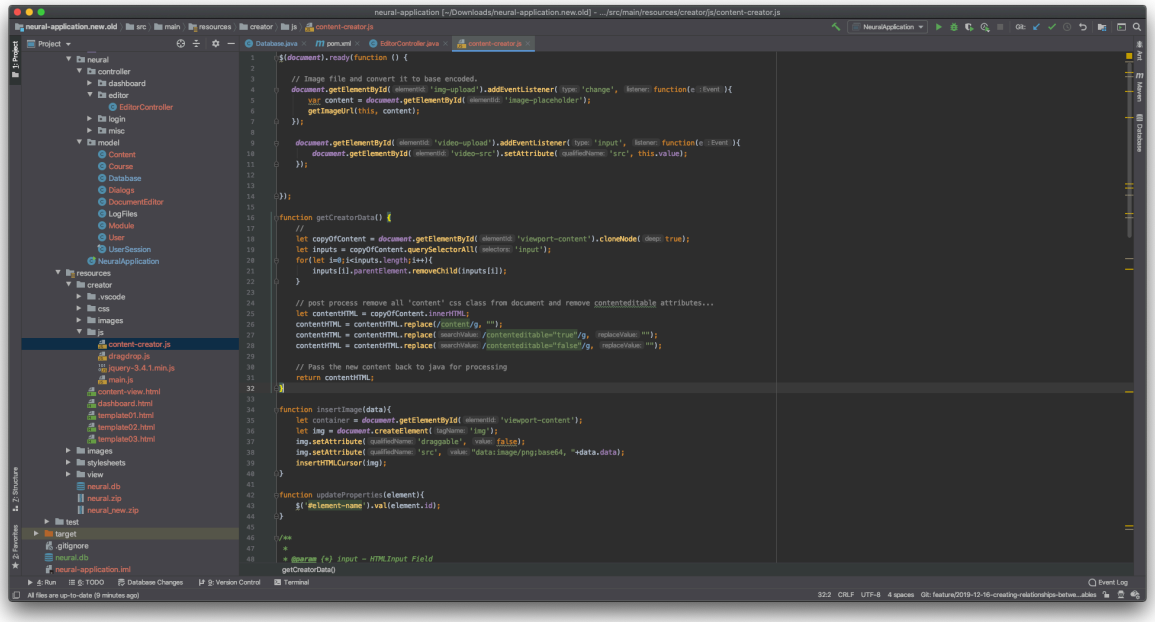


Figure 57: Appendix Admin Editor Content Extraction

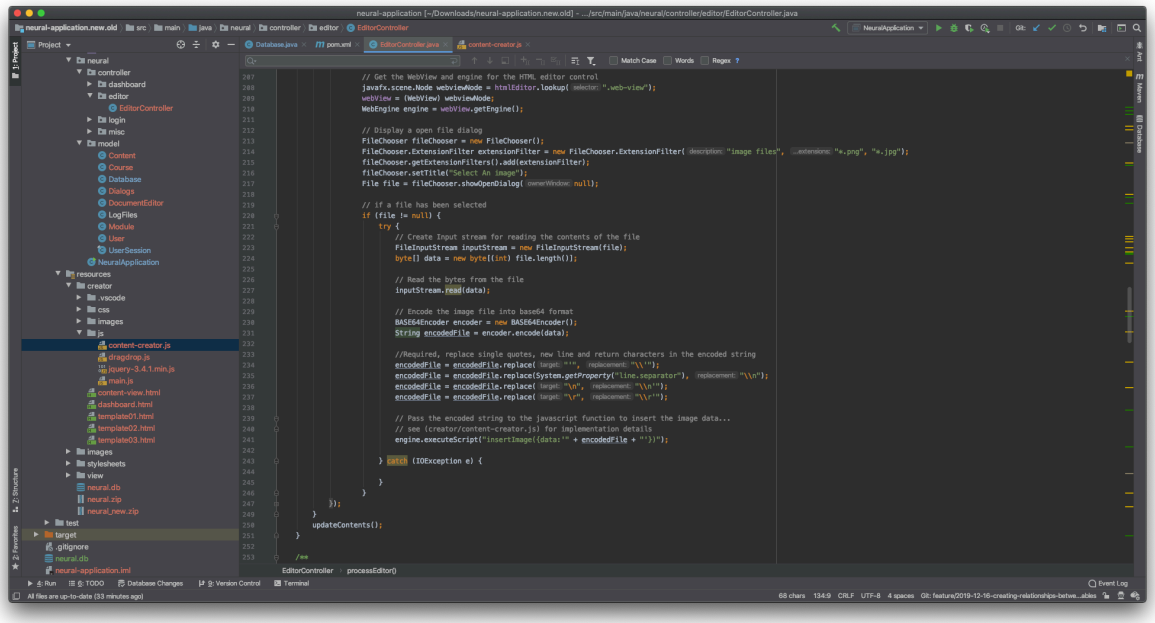


Figure 58: Appendix Admin Editor File Encoding

User Feedback

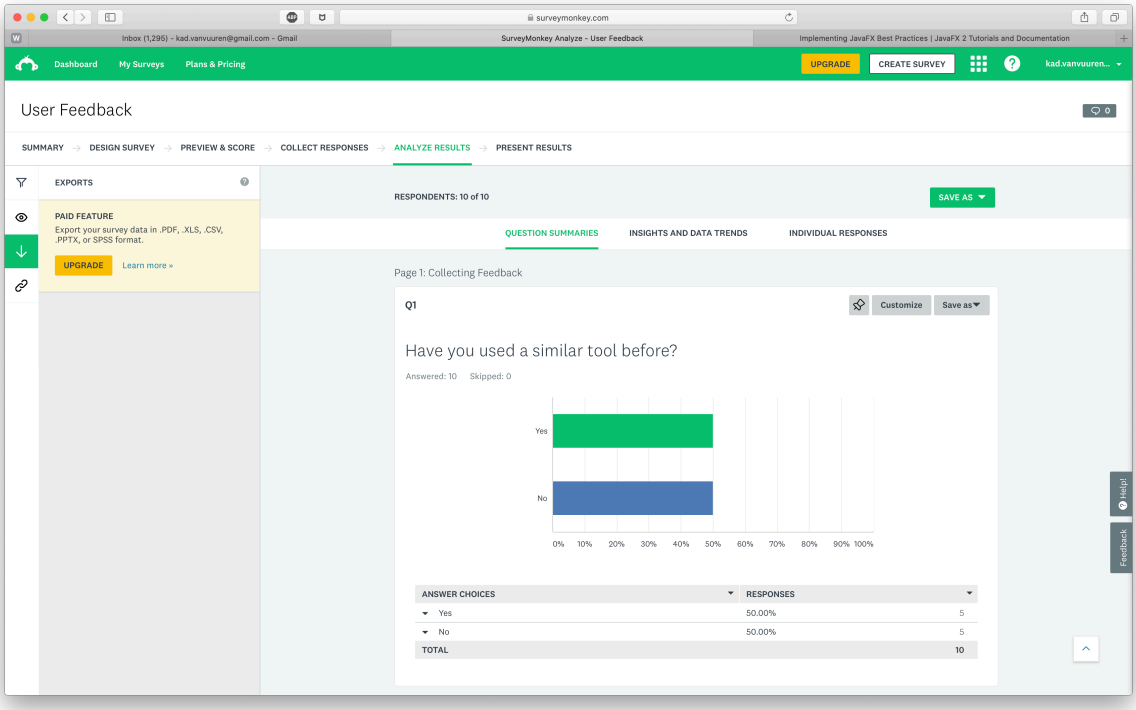


Figure 59: Appendix User Feedback

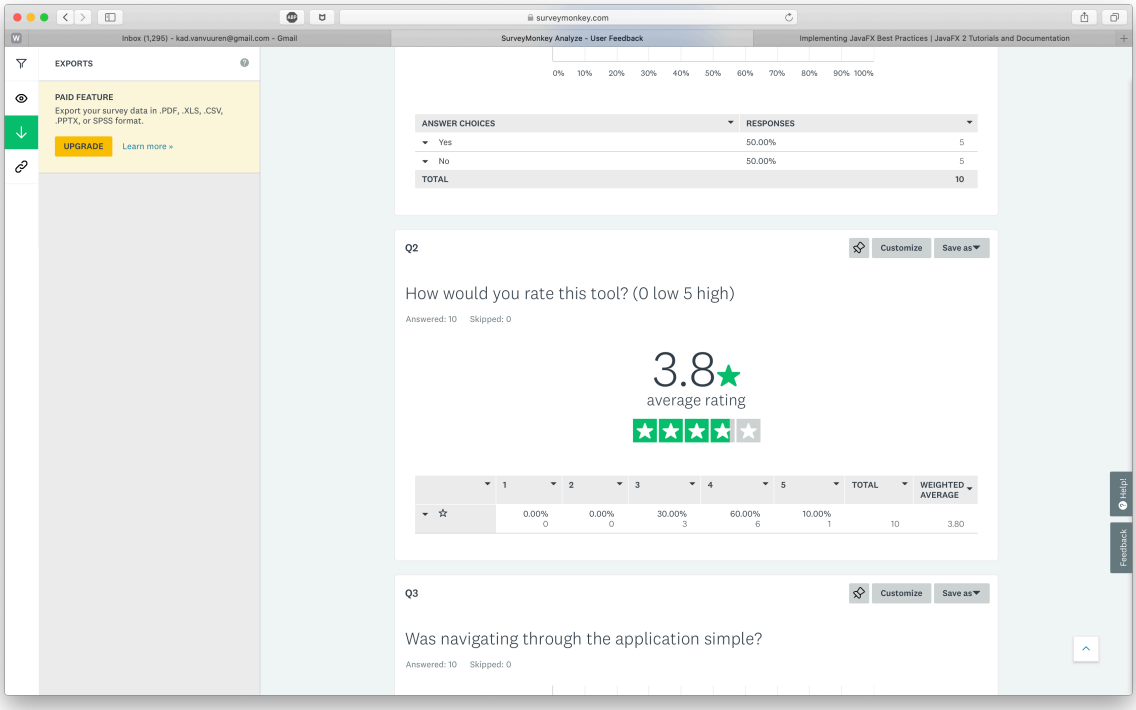


Figure 60: Appendix User Feedback

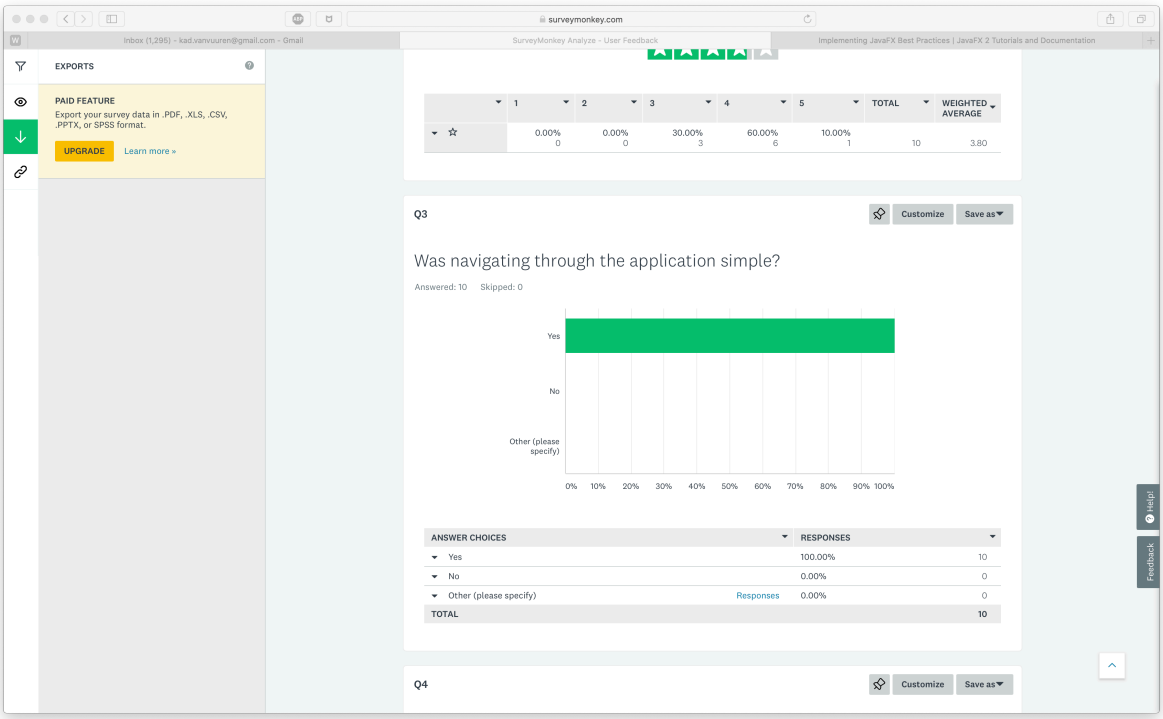


Figure 61: Appendix User Feedback

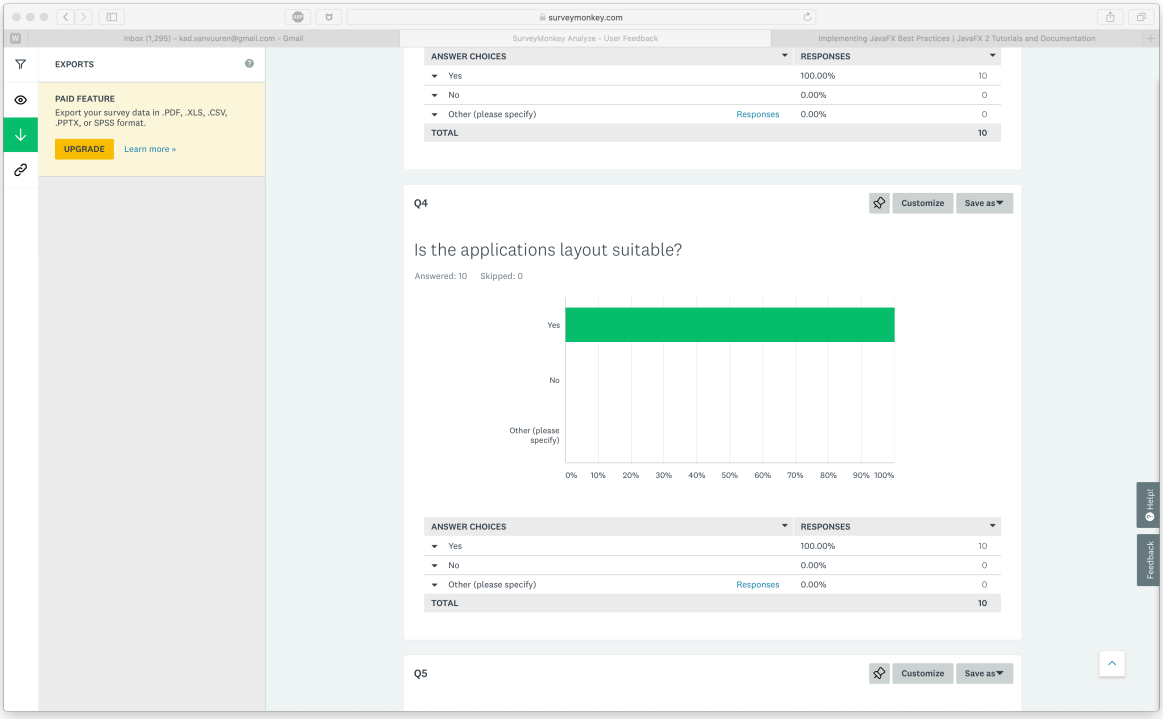


Figure 62: Appendix User Feedback

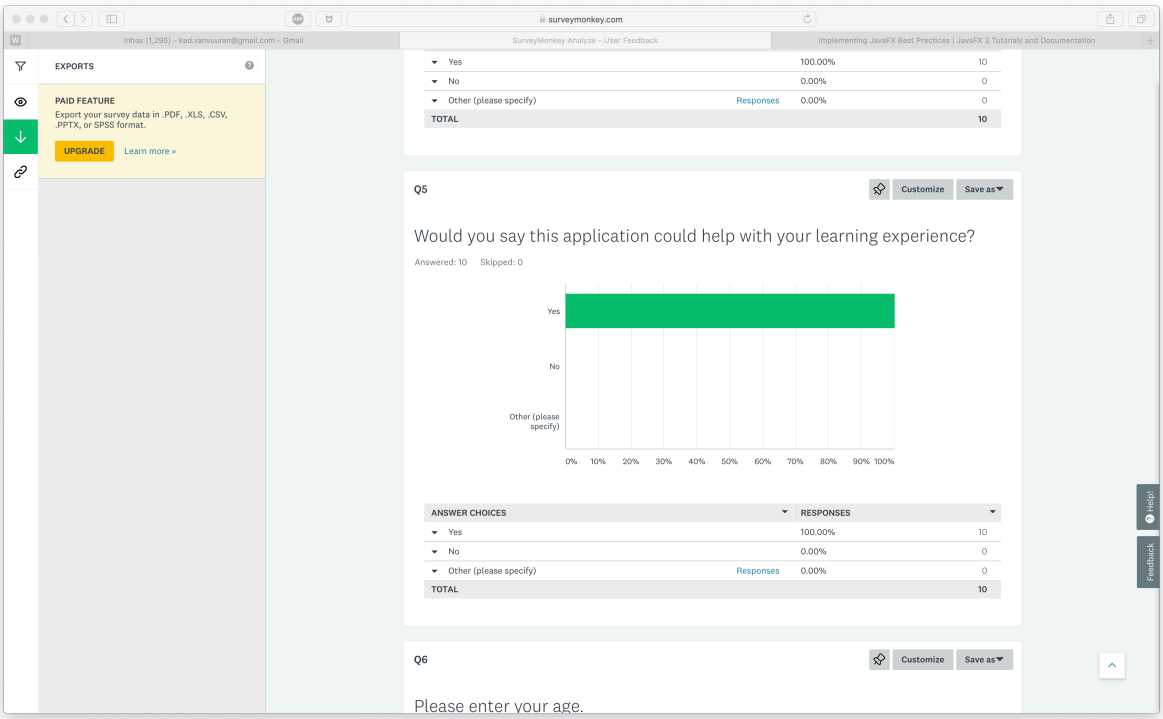


Figure 63: Appendix User Feedback

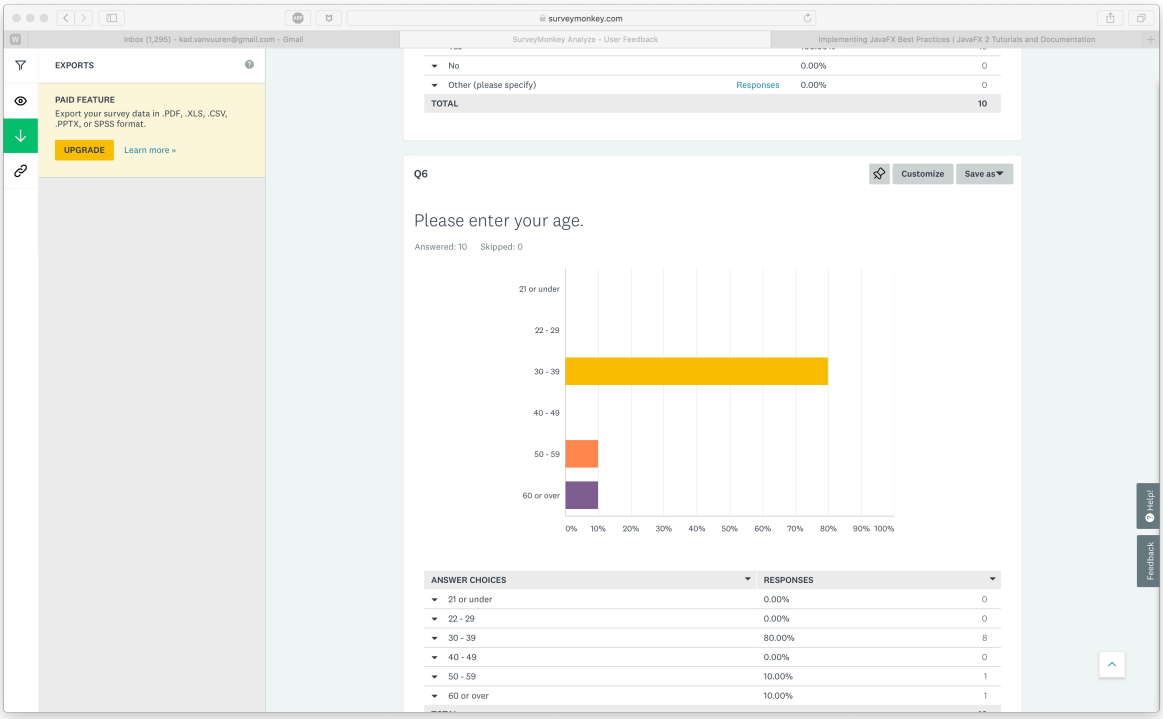


Figure 64: Appendix User Feedback

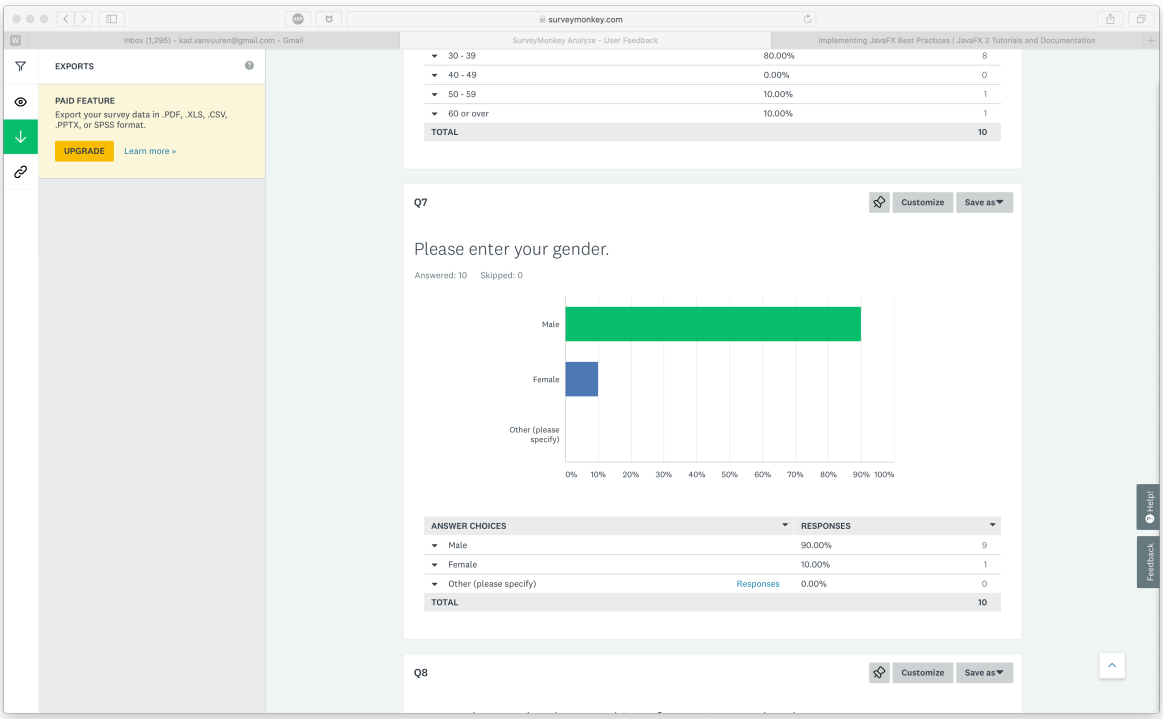


Figure 65: Appendix User Feedback

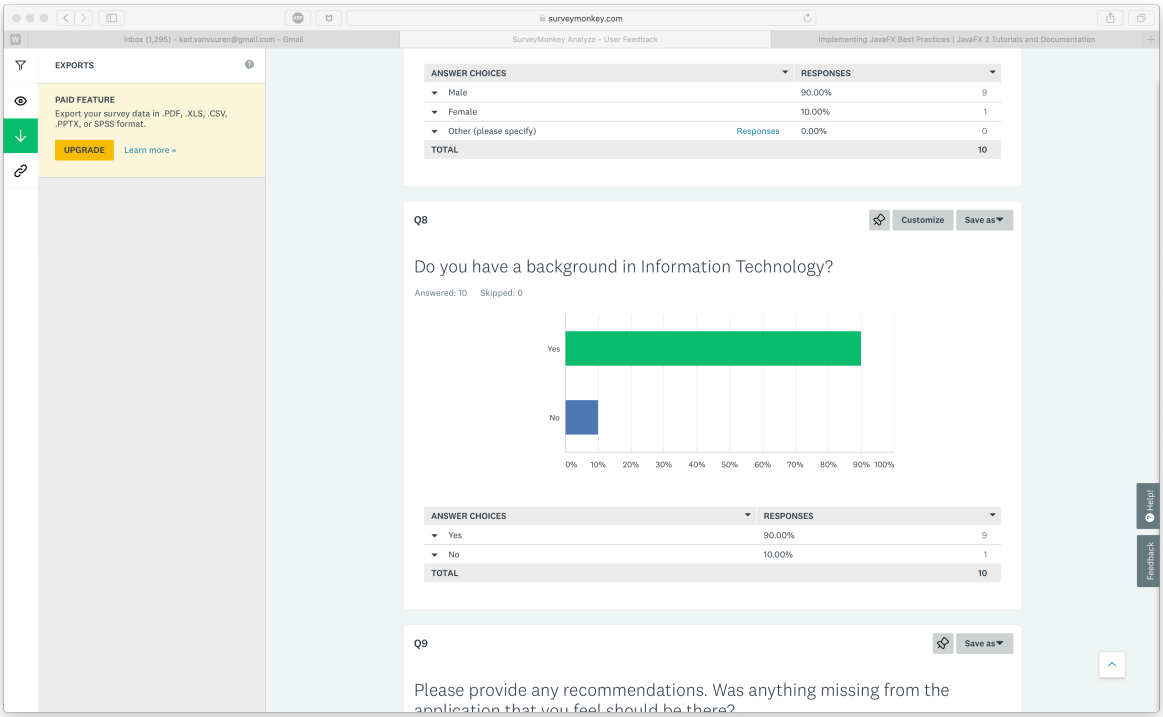


Figure 66: Appendix User Feedback

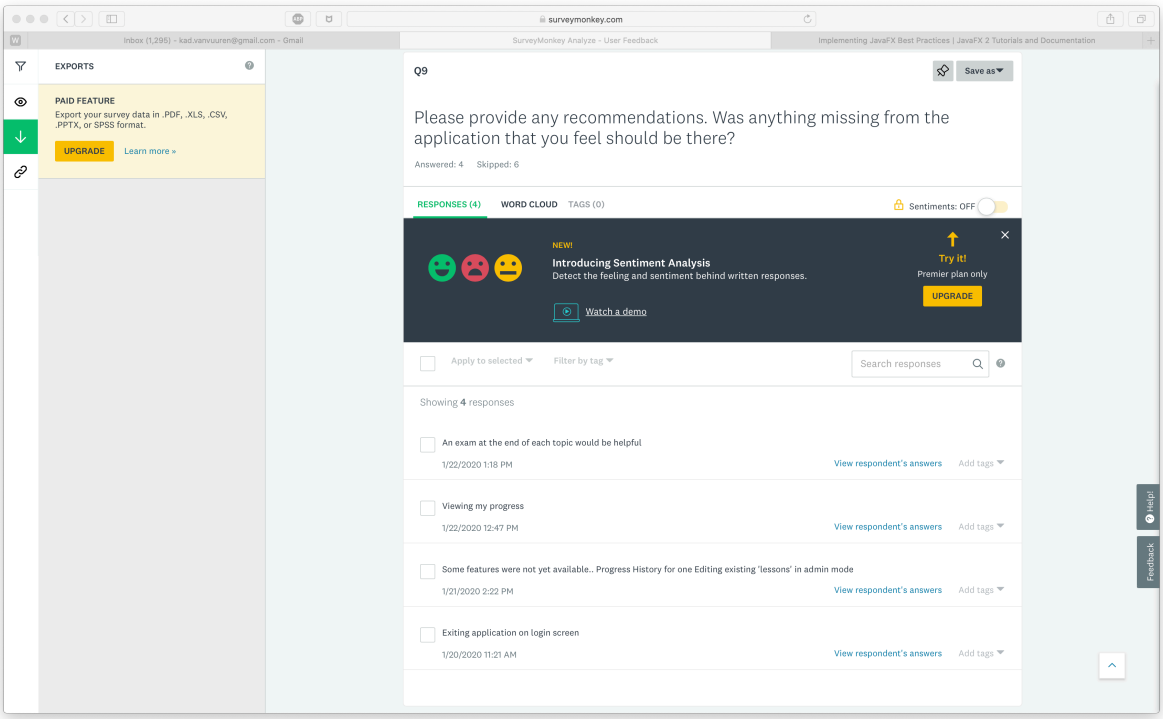
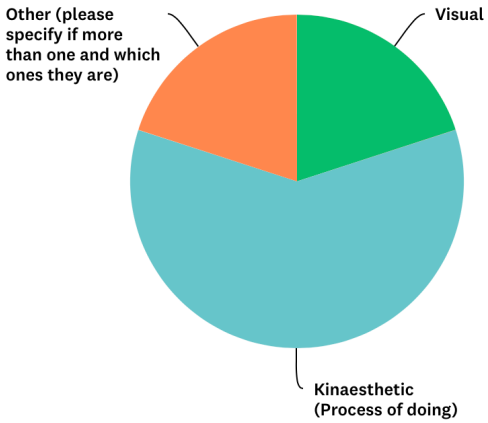


Figure 67: Appendix User Feedback

How would you categorise your own learning preference?

Answered: 10 Skipped: 0



ANSWER CHOICES		RESPONSES	
Visual		20.00%	2
Auditory		0.00%	0
Reading/Writing		0.00%	0
Kinaesthetic (Process of doing)		60.00%	6
Other (please specify if more than one and which ones they are)	Responses	20.00%	2
TOTAL			10

Figure 68: Appendix User Feedback

Reviewer Notes

When testing the application, consider:

- Run within an IDE (JAR production of the application was proving problematic due to what seems to be maven implementation).
- There exist two users within the SQLite database:
    - **Username:** Kadmin (Administrator)
    - **Password:** password
    - **Username:** User (Normal)
    - **Password:** user
- There are multiple databases that exist within the project structure – do not remove the one that exists within the root. This one is created via the database schemas that exist within the resources upon the proper use of running the applications JAR.
- Content can be edited by double clicking the module in the treeview while logged in as an administrator.
- Some database tables are empty as it was intended to use these for future considerations.