

ISOC Reservations Application

by

Anghel Sandra-Stefania

Supervisor: Lecturer Renuga Jayakumar

*Project submitted as part of the requirements for the award of BSc
Cloud Computing*

March 2022

Declaration of Originality

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

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

Signed: Anghel Sandra-Stefania

Date: 18/03/2022

Form of Consent

I, SANDRA-STEFANIA ANGHEL hereby consent that my Project, submitted in candidature for the BSc Cloud Computing degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed: Anghel Sandra-Stefania

Date: 18/03/2022

Abstract

This project analyses how booking systems can impact the restaurants and their customers, and it focuses on developing a restaurant booking system web application hosted in cloud environment.

The system functionalities are defined based on the results of a comparison between several existing similar systems on the market and aligned accordingly to the hospitality businesses major demands.

The project lifecycle is managed following Agile Methodology. The planning phase defines the project schedule, risks, and deliverables and the required technological stack and tools. Analysis phase focuses on gathering and detailing the system requirements the results are transferred to design phase where a “logical” solution for implementing the system is defined. Final phases of the project cover implementation and testing demonstrating that the system meets the expected quality.

Acknowledgement

First, I would like to thank my advisor, Lecturer Renuga Jayakumar, for supervising and rigorously reviewing this project. I also thank her for all the amazing lectures, feedbacks and for being dedicated and supportive during the last three years of study.

Second, I would like to thank our Dean Dr. Audsin Dhas, for accepting me to University of Wales Trinity Saint David, Cloud Computing programme and giving me the opportunity to make a major change in my professional career.

Third, I would like to thank Lecturer Koushik Modak, Lecturer Terry Walcott, Lecturer Suneth Fernando, Lecturer Asim Satti, Lecturer Paul Wallin, Lecturer Joy Botu and Lecturer Anupam Mazumdar for their effort and for every knowledge that helped me to achieve an incredible progress.

I would also like to thank all my colleagues for making this journey a memorable experience.

Finally, I would like to send special thanks to my parents which passed away when I was a teenager. They watched my steps and guide me, and I hope I made them proud.

Table of Content

Abstract.....	2
Acknowledgement	3
List of Figures.....	6
List of Tables	11
List of Abbreviations	13
Chapter 1 – Introduction	14
1.1.Introduction	14
1.2. Aim.....	15
1.3. Objectives	15
1.4. Motivation	15
Chapter 2 - Literature Review.....	16
2.1 Research on the problem domain.....	16
2.2. Existing examples evaluation.....	19
2.3. Required software analysis	20
2.4. Software Development Life Cycle	30
Chapter 3 - Analysis	33
3.1. Requirements analysis.....	33
3.1.1. Functional requirements.....	33
3.1.2. Non-functional requirements.....	35
3.2. UML – use case diagrams	36
3.3. System requirements	39
3.3.1. Hardware requirements.....	39
3.3.2. Software requirements.....	39
Chapter 4 - Design	41
4.1. System architecture	41
4.2. Database design.....	42
Chapter 5 - Implementation.....	44
5.1. Register	44
5.2. Login	47
5.3. Send booking request for user without account.....	51
5.4. Calculate restaurant capacity	54
5.5. Save floor plan	58
5.6. Display restaurant location on map	62

5.7. Azure infrastructure.....	64
5.8. Deployment process.....	65
Chapter 6 - Testing.....	70
6.1. ISOCADMIN functionalities	70
6.2. ADMIN functionalities.....	75
6.3. HOST functionalities.....	99
6.4. CUSTOMER functionalities.....	107
6.5. ANONYMOUS USER functionalities.....	113
6.6. SECURITY	119
Chapter 7 - Evaluation	120
7.1. Personal evaluation.....	120
7.2. System evaluation	121
Chapter 8 - Further enhancement.....	122
Chapter 9 - Conclusion.....	123
References.....	124
Appendices.....	127
Appendix A – Gantt Chart monthly view.....	127
Appendix B – ISOC Reservations Database.....	131
Appendix C – Frontend component project structure.....	133
Appendix D – Backend component project structure	134

List of Figures

FIGURE 1: ISOC RESERVATIONS - AZURE PRICING ESTIMATION	18
FIGURE 2: WINDOWS OPERATING SYSTEM SPECIFICATIONS	21
FIGURE 3: JAVA VERSION COMMAND OUTPUT.....	21
FIGURE 4: SPRING INITIALIZR.....	22
FIGURE 5: ISOC RESERVATIONS JWT EXAMPLE	23
FIGURE 6: INTELLIJ IDEA - STARTING APACHE TOMCAT SERVER.....	23
FIGURE 7: ANGULAR CLI VERSION COMMAND OUTPUT	24
FIGURE 8: ISOC RESERVATIONS APPLICATION NPM PACKAGE DEPENDENCIES	25
FIGURE 9: COMPARISON BETWEEN MICROSOFT AZURE CLOUD MODELS.....	26
FIGURE 10: CONFLUENCE - MAJOR PROJECT SPACE.....	27
FIGURE 11: VISUAL STUDIO CODE SPECIFICATIONS	28
FIGURE 12: INTELLIJ IDEA SPECIFICATIONS	28
FIGURE 13: MYSQL WORKBENCH VERSION	29
FIGURE 14: MYSQL WORKBENCH CONNECTIONS MANAGEMENT	29
FIGURE 15: FILEZILLA SPECIFICATIONS	30
FIGURE 16: ISOC RESERVATIONS GANTT CHART	30
FIGURE 17: AGILE SDLC METHODOLOGY PHASES	31
FIGURE 18: ISOC RESERVATIONS PROJECT - AZURE BOARD.....	32
FIGURE 19: ISOCADMIN USE CASE DIAGRAM.....	37
FIGURE 20: ADMIN USE CASE DIAGRAM.....	37
FIGURE 21: HOST USE CASE DIAGRAM.....	38
FIGURE 22: CUSTOMER AND ANONYMOUS CUSTOMER USE CASE DIAGRAM	38
FIGURE 23: ISOC RESERVATIONS SYSTEM ARCHITECTURE.....	41
FIGURE 24: ISOC RESERVATIONS ERD DIAGRAM	43
FIGURE 25: HIGH-LEVEL DIAGRAM FOR "REGISTER" IMPLEMENTATION	44
FIGURE 26: REGISTER - ANGULAR - SINGUP METHOD IMPLEMENTATION.....	45
FIGURE 27: REGISTER - ANGULAR - AUTHSERVICE IMPLEMENTATION.....	46
FIGURE 28: REGISTER - REST API - AUTHCONTROLLER IMPLEMENTATION.....	46
FIGURE 29: REGISTER - REST API - AUTHSERVICE IMPLEMENTATION.....	47
FIGURE 30: HIGH-LEVEL DIAGRAM FOR "LOGIN" IMPLEMENTATION	47
FIGURE 31: LOGIN - ANGULAR - SIGNIN METHOD IMPLEMENTATION	49
FIGURE 32: LOGIN - ANGULAR - AUTHSERVICE IMPLEMENTATION	49
FIGURE 33: LOGIN - REST API - AUTHCONTROLLER IMPLEMENTATION.....	50
FIGURE 34: LOGIN - REST API - AUTHSERVICE IMPLEMENTATION.....	50
FIGURE 35: LOGIN - REST API - GENERATEJWTOKEN METHOD IMPLEMENTATION.....	50
FIGURE 36: HIGH-LEVEL DIAGRAM FOR "SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT" IMPLEMENTATION.....	51
FIGURE 37: SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT - ANGULAR - OPENDIALOG METHOD IMPLEMENTATION	52
FIGURE 38: SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT - ANGULAR - SENDBOOKING METHOD IMPLEMENTATION	52

FIGURE 39: SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT - BOOKINGSSERVICE IMPLEMENTATION.....	53
FIGURE 40: SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT - REST API - BOOKINGSCONTROLLER IMPLEMENTATION.....	53
FIGURE 41: SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT - REST API - BOOKINGSSERVICE IMPLEMENTATION.....	53
FIGURE 42: HIGH-LEVEL DIAGRAM FOR "CALCULATE RESTAURANT CAPACITY " IMPLEMENTATION.....	54
FIGURE 43: CALCULATE RESTAURANT CAPACITY - ANGULAR - LOADBOOKINGS METHOD IMPLEMENTATION.....	55
FIGURE 44: CALCULATE RESTAURANT CAPACITY - ANGULAR - BOOKINGSSERVICE IMPLEMENTATION.....	55
FIGURE 45: CALCULATE RESTAURANT CAPACITY - ANGULAR - FILTER TODAY BOOKINGS IMPLEMENTATION.....	55
FIGURE 46: CALCULATE RESTAURANT CAPACITY - ANGULAR - CALCULATEBREAKFASTCAPACITY METHOD IMPLEMENTATION.....	56
FIGURE 47: CALCULATE RESTAURANT CAPACITY - ANGULAR - CALCULATELUNCHCAPACITY METHOD IMPLEMENTATION	56
FIGURE 48: CALCULATE RESTAURANT CAPACITY - ANGULAR - CALCULATEDINNERCAPACITY METHOD IMPLEMENTATION	57
FIGURE 49: HIGH-LEVEL DIAGRAM FOR "SAVE FLOOR PLAN" IMPLEMENTATION.....	58
FIGURE 50: SAVE FLOOR PLAN - ANGULAR - DRAWFLOORPLAN METHOD IMPLEMENTATION.....	59
FIGURE 51: SAVE FLOOR PLAN - ANGULAR - PROCESSEATCHART METHOD IMPLEMENTATION.....	59
FIGURE 52: SAVE FLOOR PLAN - ANGULAR - SAVEFLOORPLAN METHOD IMPLEMENTATION.....	60
FIGURE 53: SAVE FLOOR PLAN - ANGULAR - RESTAURANTSSERVICE IMPLEMENTATION	60
FIGURE 54: SAVE FLOOR PLAN - REST API - ENDPOINT IMPLEMENTATION.....	61
FIGURE 55: SAVE FLOOR PLAN - REST API - FLOORPLANSERVICE IMPLEMENTATION.....	61
FIGURE 56: SAVE FLOOR PLAN - REST API - FLOORPLANREPOSITORY IMPLEMENTATION.....	61
FIGURE 57: HIGH-LEVEL DIAGRAM FOR "DISPLAY RESTAURANT LOCATION ON MAP" IMPLEMENTATION.....	62
FIGURE 58: DISPLAY RESTAURANT LOCATION ON MAP - ANGULAR - GETRESTAURANTPROFILE METHOD IMPLEMENTATION	62
FIGURE 59: DISPLAY RESTAURANT LOCATION ON MAP - AZUREMAPSERVICE IMPLEMENTATION.....	63
FIGURE 60: DISPLAY RESTAURANT LOCATION ON MAP - ANGULAR - COORDINATES SETTER IMPLEMENTATION	63
FIGURE 61: DISPLAY RESTAURANT LOCATION ON MAP - ANGULAR - CREATEMAP METHOD IMPLEMENTATION.....	63
FIGURE 62: HIGH-LEVEL DIAGRAM FOR "AZURE INFRASTRUCTURE" IMPLEMENTATION ..	64
FIGURE 63: AZURE PORTAL	64
FIGURE 64: AZURE TOOLS EXTENSION FOR VISUAL STUDIO CODE.....	65
FIGURE 65: RUNNING NG BUILD COMMAND AND DIST DIRECTORY OUTPUT.....	65
FIGURE 66: AZURE TOOLS EXTENSION - CONNECT TO AZURE SUBSCRIPTION.....	66

FIGURE 67: AZURE TOOLS EXTENSION - DEPLOY TO STATIC WEBSITE VIA AZURE STORAGE.....	66
FIGURE 68: INTELLIJ IDE - MAVEN COMMANDS	67
FIGURE 69: TARGET DIRECTORY OUTPUT.....	67
FIGURE 70: AZURE APP SERVICE FTP CREDENTIALS.....	68
FIGURE 71: FILEZILLA CONNECTION TO AZURE APP SERVICE AND ARTIFACT UPLOAD.....	68
FIGURE 72: MYSQL WORKBENCH - EXPORT DATABASE	69
FIGURE 73: INPUT ONBOARD NEW RESTAURANT TEST CASE – INVALID DATA.....	70
FIGURE 74: OUTPUT ONBOARD NEW RESTAURANT TEST CASE – INVALID DATA	70
FIGURE 75: INPUT - ONBOARD NEW RESTAURANT TEST CASE - VALID DATA	71
FIGURE 76: OUTPUT - ONBOARD NEW RESTAURANT TEST CASE - VALID DATA	71
FIGURE 77: INPUT - CREATE ADMIN ACCOUNT TEST CASE - INVALID DATA.....	72
FIGURE 78: OUTPUT - CREATE ADMIN ACCOUNT TEST CASE - INVALID DATA	72
FIGURE 79: INPUT - CREATE ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD	73
FIGURE 80: OUTPUT - CREATE ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD.....	73
FIGURE 81: INPUT - CREATE ADMIN ACCOUNT TEST CASE - VALID DATA	74
FIGURE 82: OUTPUT - CREATE ADMIN ACCOUNT TEST CASE - VALID DATA	74
FIGURE 83: INPUT - ADMIN LOGIN TEST CASE - MISSING CREDENTIALS	75
FIGURE 84: OUTPUT - ADMIN LOGIN TEST CASE - MISSING CREDENTIALS	76
FIGURE 85: INPUT – ADMIN LOGIN TEST CASE – INVALID CREDENTIALS	76
FIGURE 86: OUTPUT - ADMIN LOGIN TEST CASE - INVALID CREDENTIALS	77
FIGURE 87: INPUT - ADMIN LOGIN TEST CASE - VALID CREDENTIALS	77
FIGURE 88: OUTPUT - ADMIN LOGIN TEST CASE - VALID CREDENTIALS	78
FIGURE 89: INPUT - CHANGE PASSWORD TEST CASE - INVALID DATA	78
FIGURE 90: OUTPUT - CHANGE PASSWORD TEST CASE - INVALID DATA	79
FIGURE 91: INPUT - CHANGE PASSWORD TEST CASE - NOT MATCH	79
FIGURE 92: OUTPUT - CHANGE PASSWORD TEST CASE - NOT MATCH	80
FIGURE 93: INPUT - CHANGE PASSWORD TEST CASE - VALID DATA	80
FIGURE 94: OUTPUT - CHANGE PASSWORD TEST CASE - VALID DATA	81
FIGURE 95: INPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID DATA	81
FIGURE 96: OUTPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID DATA	82
FIGURE 97: INPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD	83
FIGURE 98: OUTPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD	83
FIGURE 99: INPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - VALID DATA...	84
FIGURE 100: OUTPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - VALID DATA	84
FIGURE 101: INPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE - INVALID DATA	85
FIGURE 102: OUTPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE - INVALID DATA	85

FIGURE 103: INPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE – INVALID EMAIL OR PASSWORD	86
FIGURE 104: OUTPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD	86
FIGURE 105: INPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE – VALID DATA ...	87
FIGURE 106: OUTPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE – VALID DATA	87
FIGURE 107: INPUT - MODIFY ACCOUNT TEST CASE - INVALID DATA	88
FIGURE 108: OUTPUT - MODIFY ACCOUNT TEST CASE - INVALID DATA	88
FIGURE 109: INPUT - MODIFY ACCOUNT TEST CASE - VALID DATA.....	89
FIGURE 110: OUTPUT - MODIFY ACCOUNT TEST CASE - VALID DATA.....	89
FIGURE 111: INPUT - DELETE ACCOUNT TEST CASE.....	90
FIGURE 112: OUTPUT - DELETE ACCOUNT TEST CASE.....	90
FIGURE 113: INPUT - MODIFY RESTAURANT GENERAL INFORMATION TEST CASE - INVALID DATA.....	91
FIGURE 114: OUTPUT - MODIFY RESTAURANT GENERAL INFORMATION TEST CASE - INVALID DATA.....	92
FIGURE 115: INPUT - MODIFY RESTAURANT PROFILE TEST CASE - VALID DATA	92
FIGURE 116: OUTPUT - MODIFY RESTAURANT PROFILE TEST CASE - VALID DATA	93
FIGURE 117: INPUT - RESTAURANT IMAGE UPLOAD TEST CASE - INVALID SIZE – IMAGE PROPERTIES.....	93
FIGURE 118: INPUT - RESTAURANT IMAGE UPLOAD TEST CASE - INVALID SIZE	94
FIGURE 119: OUTPUT - RESTAURANT IMAGE UPLOAD TEST CASE - INVALID SIZE	94
FIGURE 120: INPUT - RESTAURANT IMAGE UPLOAD TEST CASE - VALID SIZE – IMAGE PROPERTIES.....	94
FIGURE 121: INPUT - RESTAURANT IMAGE UPLOAD TEST CASE - VALID SIZE.....	95
FIGURE 122: OUTPUT - RESTAURANT IMAGE UPLOAD TEST CASE - VALID SIZE.....	95
FIGURE 123: INPUT - RESTAURANT LOCATION ON MAP TEST CASE	96
FIGURE 124: OUTPUT - RESTAURANT LOCATION ON MAP TEST CASE	96
FIGURE 125: INPUT - CREATE RESTAURANT FLOOR PLAN TEST CASE - INVALID CONFIGURATION.....	97
FIGURE 126: OUTPUT - CREATE RESTAURANT FLOOR PLAN TEST CASE - INVALID CONFIGURATION.....	97
FIGURE 127: INPUT - CREATE RESTAURANT FLOOR PLAN TEST CASE - VALID CONFIGURATION.....	98
FIGURE 128: OUTPUT - CREATE RESTAURANT FLOOR PLAN TEST CASE - VALID CONFIGURATION.....	98
FIGURE 129: INPUT - HOST LOGIN TEST CASE - INVALID CREDENTIALS	99
FIGURE 130: OUTPUT - HOST LOGIN TEST CASE - INVALID CREDENTIALS	99
FIGURE 131: INPUT - HOST LOGIN TEST CASE - VALID CREDENTIALS.....	100
FIGURE 132: OUTPUT - HOST LOGIN TEST CASE - VALID CREDENTIALS.....	101
FIGURE 133: INPUT - VIEW TODAY’S BOOKINGS TEST CASE.....	101
FIGURE 134: OUTPUT - VIEW TODAY’S BOOKINGS TEST CASE.....	102
FIGURE 135: INPUT - VIEW ALL BOOKINGS TEST CASE	102
FIGURE 136: OUTPUT - VIEW ALL BOOKINGS TEST CASE	103
FIGURE 137: INPUT - MODIFY TODAY’S BOOKING TEST CASE – BOOKING DETAILS	103
FIGURE 138: OUTPUT - MODIFY TODAY’S BOOKING TEST CASE - BOOKING DETAILS	104

FIGURE 139: INPUT - CANCEL TODAY’S BOOKING TEST CASE	104
FIGURE 140: OUTPUT - CANCEL TODAY’S BOOKING TEST CASE	105
FIGURE 141: INPUT - CUSTOMER ON WAITING-LIST	105
FIGURE 142: OUTPUT - CUSTOMER ON WAITING-LIST	106
FIGURE 143: INPUT - MONITOR RESTAURANT CAPACITY TEST CASE	106
FIGURE 144: OUTPUT - MONITOR RESTAURANT CAPACITY TEST CASE	106
FIGURE 145: INPUT - MODIFY ACCOUNT TEST CASE - INVALID DATA	107
FIGURE 146: OUTPUT - MODIFY ACCOUNT TEST CASE - INVALID DATA	107
FIGURE 147: INPUT - MODIFY ACCOUNT TEST CASE - VALID DATA.....	108
FIGURE 148: OUTPUT - MODIFY ACCOUNT TEST CASE - VALID DATA.....	108
FIGURE 149: INPUT - VIEW BOOKINGS TEST CASE	108
FIGURE 150: OUTPUT - VIEW BOOKINGS TEST CASE	109
FIGURE 151: INPUT - SENT BOOKING REQUEST TEST CASE - INVALID DATA	109
FIGURE 152: OUTPUT - SENT BOOKING REQUEST TEST CASE - INVALID DATA	110
FIGURE 153: INPUT - SEND BOOKING REQUEST TEST CASE - VALID DATA.....	111
FIGURE 154: OUTPUT - SEND BOOKING REQUEST TEST CASE - VALID DATA.....	111
FIGURE 155: INPUT - CANCEL BOOKING TEST CASE.....	112
FIGURE 156: OUTPUT - CANCEL BOOKING TEST CASE.....	112
FIGURE 157: INPUT - CREATE ACCOUNT - INVALID DATA	113
FIGURE 158: OUTPUT - CREATE ACCOUNT - INVALID DATA	114
FIGURE 159: INPUT - CREATE ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD.....	115
FIGURE 160: OUTPUT - CREATE ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD..	115
FIGURE 161: INPUT - CREATE ACCOUNT TEST CASE - VALID DATA.....	116
FIGURE 162: OUTPUT - CREATE ACCOUNT TEST CASE - VALID DATA.....	116
FIGURE 163: INPUT - SEND BOOKING REQUEST TEST CASE - INVALID DATA.....	117
FIGURE 164: OUTPUT - SEND BOOKING REQUEST TEST CASE - INVALID DATA.....	117
FIGURE 165: INPUT SEND BOOKING REQUEST TEST CASE - VALID DATA	118
FIGURE 166: OUTPUT - SEND BOOKING REQUEST TEST CASE - VALID DATA.....	118
FIGURE 167: INPUT - AUTHORIZATION ACCESS TEST CASE	119
FIGURE 168: OUTPUT - AUTHORIZATION ACCESS TEST CASE	119

List of Tables

TABLE 1: COMPARISON BETWEEN MICROSOFT AZURE AND AMAZON WEB SERVICES.....	17
TABLE 2: COMPARISON BETWEEN YELP RESERVATIONS, OPENTABLE AND TABLESREADY SYSTEMS	20
TABLE 3: ISOC RESERVATIONS APPLICATION - USED MICROSOFT AZURE SERVICES	27
TABLE 4: ISOC RESERVATIONS - FUNCTIONAL REQUIREMENTS FOR EACH ROLE	35
TABLE 5: ISOC RESERVATIONS - FUNCTIONAL REQUIREMENTS FOR ANONYMOUS USER	35
TABLE 6: ISOC RESERVATIONS NON-FUNCTIONAL REQUIREMENTS	36
TABLE 7: ISOC RESERVATIONS RISKS ANALYSIS.....	40
TABLE 8: ONBOARD NEW RESTAURANT TEST CASE – INVALID DATA	70
TABLE 9: ONBOARD NEW RESTAURANT TEST CASE – VALID DATA	71
TABLE 10: CREATE ADMIN ACCOUNT TEST CASE - INVALID DATA.....	72
TABLE 11: CREATE ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD.....	73
TABLE 12: CREATE ADMIN ACCOUNT TEST CASE - VALID DATA.....	74
TABLE 13: ADMIN LOGIN TEST CASE - MISSING CREDENTIALS	76
TABLE 14: ADMIN LOGIN TEST CASE - INVALID CREDENTIALS.....	77
TABLE 15: ADMIN LOGIN TEST CASE - VALID CREDENTIALS	78
TABLE 16: CHANGE PASSWORD TEST CASE - INVALID DATA.....	79
TABLE 17: CHANGE PASSWORD TEST CASE - NOT MATCH.....	80
TABLE 18: CHANGE PASSWORD TEST CASE - VALID DATA.....	81
TABLE 19: CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID DATA	82
TABLE 20: CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD.....	83
TABLE 21: CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - VALID DATA.....	84
TABLE 22: CREATE RESTAURANT HOST ACCOUNT TEST CASE - INVALID DATA	85
TABLE 23: CREATE RESTAURANT HOST ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD.....	86
TABLE 24: CREATE RESTAURANT HOST ACCOUNT TEST CASE – VALID DATA.....	87
TABLE 25: MODIFY ACCOUNT TEST CASE - INVALID DATA.....	88
TABLE 26: MODIFY ACCOUNT TEST CASE - VALID DATA	89
TABLE 27: DELETE ACCOUNT TEST CASE	90
TABLE 28: MODIFY RESTAURANT GENERAL INFORMATION TEST CASE - INVALID DATA	92
TABLE 29: MODIFY RESTAURANT PROFILE TEST CASE - VALID DATA.....	93
TABLE 30: RESTAURANT IMAGE UPLOAD TEST CASE - INVALID SIZE	94
TABLE 31: RESTAURANT IMAGE UPLOAD TEST CASE - VALID SIZE	95
TABLE 32: RESTAURANT LOCATION ON MAP TEST CASE.....	96
TABLE 33: CREATE RESTAURANT FLOOR PLAN TEST CASE - INVALID CONFIGURATION	97
TABLE 34: CREATE RESTAURANT FLOOR PLAN TEST CASE - VALID CONFIGURATION	98
TABLE 35: HOST LOGIN TEST CASE - INVALID CREDENTIALS.....	99
TABLE 36: HOST LOGIN TEST CASE - VALID CREDENTIALS	101
TABLE 37: VIEW TODAY’S BOOKINGS TEST CASE	102
TABLE 38: VIEW ALL BOOKINGS TEST CASE	103
TABLE 39: MODIFY TODAY'S BOOKINGS TEST CASE.....	104

TABLE 40: CANCEL TODAY’S BOOKING TEST CASE	105
TABLE 41: CUSTOMER ON WAITING-LIST	106
TABLE 42: MONITOR RESTAURANT CAPACITY TEST CASE	106
TABLE 43: MODIFY ACCOUNT TEST CASE - INVALID DATA.....	107
TABLE 44: MODIFY ACCOUNT TEST CASE - VALID DATA	108
TABLE 45: VIEW BOOKINGS TEST CASE.....	109
TABLE 46: SENT BOOKING REQUEST TEST CASE - INVALID DATA.....	110
TABLE 47: SEND BOOKING REQUEST TEST CASE - VALID DATA	111
TABLE 48: CANCEL BOOKING TEST CASE	112
TABLE 49: CREATE ACCOUNT - INVALID DATA.....	114
TABLE 50: CREATE ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD	115
TABLE 51: CREATE ACCOUNT TEST CASE - VALID DATA	116
TABLE 52: SEND BOOKING REQUEST TEST CASE - INVALID DATA	117
TABLE 53: SEND BOOKING REQUEST TEST CASE - VALID DATA	118
TABLE 54: AUTHORIZATION ACCESS TEST CASE	119

List of Abbreviations

API	Application Programming Interface
CLI	Command-line Interface
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
ERD	Entity Relationship Diagram
FTP	File Transfer Protocol
GB	Gigabyte
GIS	Geographic Information System
HTML	Hyper Text Markup Language
HTTPS	Secure Hyper Text Transfer Protocol
IDE	Integrated Development Environment
JDK	Java Development Kit
JPA	Java Persistence API
JWT	JSON Web Token
NPM	Node Package Manager
OWASP	Open Web Application Security Project
RAM	Random Access Memory
SABRE	Semi-Automated Business Research Environment
SDLC	Software Development Life Cycle
SMART	Specific, Measurable, Achievable, Relevant, Time-Bound
SQL	Structured Query Language
TB	Terabyte
UML	Unified Modeling Language
URL	Uniform Resource Locator

Chapter 1 – Introduction

1.1. Introduction

The concept of “restaurant reservation” exists since 18th century when the guest had the option to reserve a “room” for private meals. Restaurants offered public rooms for dinner or a room with several smaller tables available for the guests to be engaged. The scope, at that moment was to reduce the delays in serving and to have everything prepared when the guests arrive. This practice came with the limitations that the guest had the responsibility to decide in advance the menu, the waiter’s tip and to pay the bill. The reservations were proceeded in person not over the phone and the idea of doing reservations over the internet was utopic for about a century [1].

The 19th century represents a boom in the hospitality area, being marked by the exponential growth of the social phenomena. Massive masses started to eat out, wrote reviews, and gave ratings and rankings to the restaurant. The period 1950-1980 was recorded in history as “the era of computer reservation systems”. In 1964 was developed the first computerized booking system, called SABRE (Semi-automated Business Research Environment) and it was dedicated to airline industry [2]. SABRE was capable to process more than 7000 booking requests per hour without errors and the reservation process became an easy task. Since then, a lot of industries including travel and hospitality exploit the benefits of booking systems. [3]

Nowadays, even there are hundreds of booking systems available on the market, the restaurants must take the decision of either choosing the old manner of using the paper for their entire management or implementing a system. There is no perfect decision, and every booking system has advantages and limitations. The restaurant can avoid over-occupation by accepting reservations only and can have a better planning of the staff allocation and dining room layout, but, on the other side, they can lose some customers as there are people that prefer to not plan a dinner. The restaurant can have a higher profit because reservations might bring large groups of guests, but they also need to assume the cost of no-shows.

1.2. Aim

The aim of this project is to develop a restaurant booking system web application hosted in cloud by using Microsoft Azure cloud provider.

1.3. Objectives

To achieve this aim, I must pursue with the following SMART objectives:

- To do research about several existing booking systems for restaurants (Yelp Reservations, OpenTable and TablesReady)
- To identify the required technological stack, tools, and resources
- To design the normalized database structure and infrastructure for proposed solution
- To analyse OWASP (Open Web Application Security Project) guideline for security aspects
- To learn and enhance required technologies and tools
- To define and develop the functionalities for each type of end-user

1.4. Motivation

The current project is implemented following the main scope of increasing efficiency, profit and the most important to improve the customers satisfaction. It was chosen based on experience on existing similar systems, based on diversity of the proposed technological stack which brings a huge technical skills improvement, and it is desired from career progression perspective.

Chapter 2 - Literature Review

This chapter presents an overview of the problem domain and a comparison between three of the most known existing restaurant reservation systems on the market. The required software list is identified and described, and, in the last section of the chapter, the Software Development Life Cycle (SDLC) is presented.

2.1 Research on the problem domain

As it was already mentioned in Chapter 1, the concept of “restaurant reservation” is not new. The businesses are looking for a solution to increase the profit and optimize the processes, and, on the other hand the customers are looking to improve their dining-out experience. A study performed by OpenTable in 2015 [4], indicates that 85 percent of customers are interested in knowing in advance how long they must wait for a table. This percentage is significative, and it would be difficult to provide an accurate estimation without having a reservations system in place with functionalities for capacity monitoring and waiting list. But guests experience is not the only benefit.

Recent period was marked by a global pandemic caused by COVID-19 when the restaurant reservations systems became more valuable [5]. The restaurants were forced to reduce their capacity to a specific percentage, based on government regulations, and to ensure a social distancing rule. Having a reservation system in place, was essential to improve social and health aspects. The customers were able to book in advance without physically going in the restaurant and getting in contact with the staff or other customers. The long waiting queues were eliminated and the allocation on tables was improved as the capacity was constantly monitored. However, a reservation system should have multiple characteristics to be an effective solution for businesses, including a user-friendly interface, easy setup, and increased security. The traditional way to install the application was to use an internal environment which requires up-front cost for purchasing the hardware components and technical knowledge and can be difficult to maintain. The alternative is represented by using services offered by cloud providers.

Cloud providers are third-party companies specialized in complex cloud-based technologies and services. A study performed by Flexera [6] shows that 91 percentage of businesses are interested in public cloud solutions offered by leaders as Amazon and Microsoft

that have advantages as: reduced cost – eliminate the need of physical hardware cost and pay only for the used resources following pay-as-you-go pricing models, flexibility – extra required resources are immediately provided and the new business demands are meet, insights – tracking and monitoring solutions are provided to offer a better overview and control for data, disaster recovery – cloud providers offers quickly backups and data recovery solutions that reduce the businesses data lose or unpredicted downtime, and security – rigorous security levels, from data centre physical security to application and data security, are implemented and large teams of security experts are allocated by cloud providers [7].

Cloud provider	Advantages	Disadvantages
Microsoft Azure [8]	• More affordable pricing • Pay-as-you-go pricing model • Excellent integration with Microsoft tools • High-level availability • Hybrid cloud provided • Better documentation and support	• Reduced number of supported operating systems • Reduced support for open-source community • Requires platform technical knowledge
Amazon Web Services [9]	• One of the oldest cloud providers on the market • Increased number of availability zones • Support for open-source community • Mature Big Data solutions	• Hybrid cloud is not provided • Technical support fee

TABLE 1: COMPARISON BETWEEN MICROSOFT AZURE AND AMAZON WEB SERVICES

Both Microsoft Azure and Amazon Web Services provides a huge range of services capable to meet all the businesses requirements, and the decision of choosing a specific cloud provider is made based on an analysis of needed resources and cost. Microsoft Azure also provides a useful and easy to use tool to determine the cost called Azure Pricing Calculator.

Current solution was implemented using Microsoft Azure as a cloud provider and the choose was based on technical expertise and effective cost estimation.

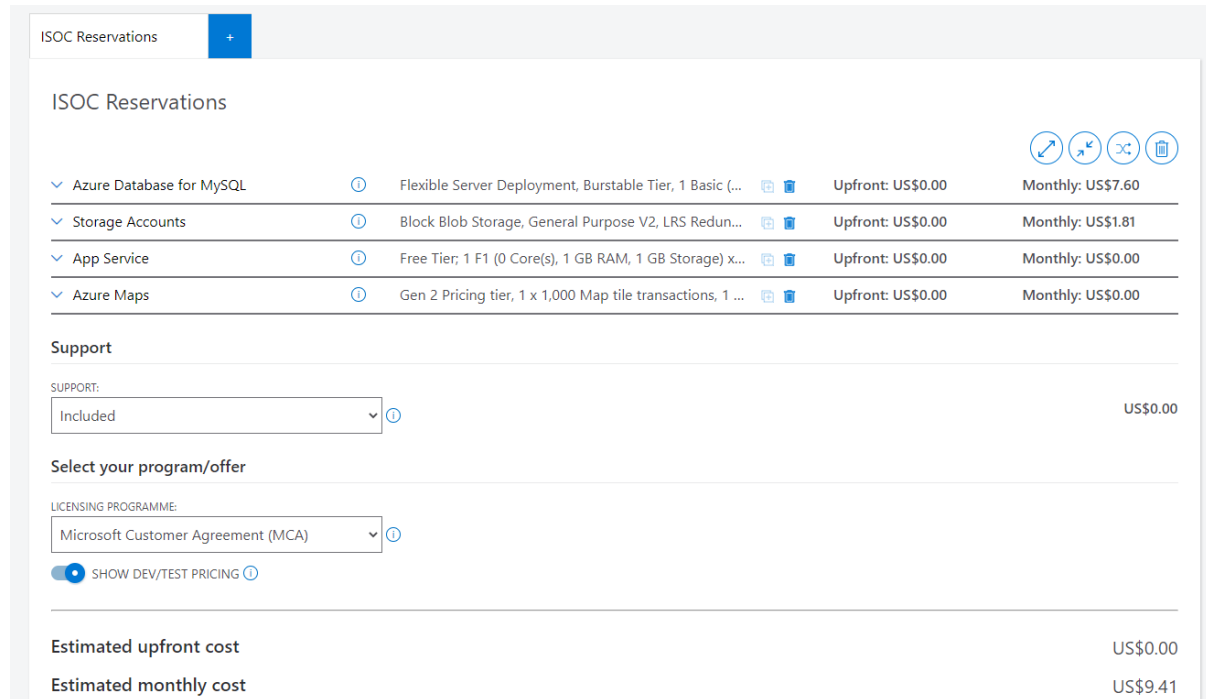


FIGURE 1: ISOC RESERVATIONS - AZURE PRICING ESTIMATION

Having the application hosted in cloud environment, simplifies the setup process, and improves the deployments. When a new version of the application is developed, the restaurants benefit from changes without downtime and more important without any additional work on client-side environments. It also ensures a high level of security. In ISOC Reservations application the security is also implemented following Open Web Application Security Project (OWASP) rules.

OWASP is a dedicated software security organization, that highlights in a document known as OWASP Top 10, the most critical security concerns related to web applications [10]. ISOC Reservations has several of the rules implemented as:

- **Broken Access Control** – users cannot access an area of the application without having permissions. ISOC Reservations application requires role-based access control
- **Cryptographic Failures** – HTTPS protocol is used to secure data in transit and password are encrypted

- **Injection** – SQL injection is handled using Spring JPA framework which construct the queries using parameters
- **Vulnerable and Outdated Components** – dependent components are installed using NPM official repository or Maven official repository

2.2. Existing examples evaluation

Nowadays, the number of restaurant reservation system is constantly increasing, but for the purpose of this comparison, three of the most known systems were chosen: Yelp Reservations, OpenTable and TablesReady.

Yelp Reservations is a system created by Yelp; a company founded in 2004. Its original scope was to offer a platform for customer's reviews, but then the company started to provide a diversity of systems targeting the reservation process as well [11] [12].

OpenTable is a company founded in 1998 with dedicated services for online restaurant reservations. It started with a reduce number of restaurants located in San Francisco and increased massively over time, currently being able to support restaurants from more than 80 countries [13] [14].

TablesReady is a brand-new company, founded in 2011. It is currently a competitor for the other two but offers a reduce number of features [15].

Functional requirement	Yelp Reservations [16]	OpenTable [17]	TablesReady [18]
Capacity monitoring	✗	✓	✗
Waiting-list management	✓	✓	✓
Email marketing	✓	✓	✓
Floor plan management	✗	✓	✗
Accounts management	✓	✓	✓

Pricing	12-month contract \$750 early termination fee	Tier-pricing: ○ Basic ○ Core ○ Pro	Montly plan
----------------	---	---	-------------

TABLE 2: COMPARISON BETWEEN YELP RESERVATIONS, OPENTABLE AND TABLESREADY SYSTEMS

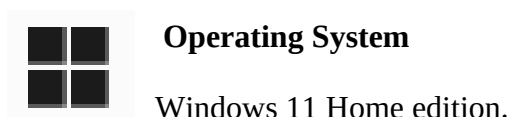
The previous comparison indicates that OpenTable seems to be the most reliable system offering a complete set of functionalities. However, there are limitations or disadvantages that might make the restaurants decision even harder. For example, Yelp Reservations and TablesReady don't provide a built-in option for managing the floor plan or visualizing the restaurant capacity, which becomes an inconvenient for the restaurants as those features are essential for optimizing the cost and improving the staff efficiency by predicting the customers flow and prepare accordingly. Another important aspect is represented by the system acquisition cost. Yelp Reservations and TablesReady offer a more affordable pricing plan, giving the system for a monthly subscription, but still imposing limitations as early termination fee. On the other hand, OpenTable comes in place with a preferable pricing option, providing the features on a tier-pricing model. This approach might be a better solution for restaurants with a smaller budget or start-up businesses, and at the same time, might not satisfy the essential needs that restaurant's flows (reservations, waiting-list, capacity monitoring etc.).

Each system has its own advantages and disadvantages, and the decision of choosing one or another, is part of a detailed analysis for the restaurants to perform to identify their business requirements and to satisfy them on a balanced cost-effective ratio.

2.3. Required software analysis

Software can be classified in two main categories: *system software* - defined as the layer between the physical hardware and applications, and *application software* – defined as a computer program responsible for executing different tasks [19].

The system software used in ISOC Reservations application's development process are:



Windows specifications	
Edition	Windows 11 Home
Version	21H2
Installed on	01/03/2022
OS build	22000.493
Experience	Windows Feature Experience Pack 1000.22000.493.0

FIGURE 2: WINDOWS OPERATING SYSTEM SPECIFICATIONS



Java 8

Java is an object-oriented programming language, developed by Sun Microsystems in 1995. It has the main advantage as being open source, providing a great community and support, and platform independent, which allows it to be executed on any operating system [20]. Backend component of ISOC Reservations application is implemented using Java 8 programming language.



Java Development Kit

Java Development Kit is a software that provides a set of tools, such as compiler, interpreter, archiver, or Java Runtime Environment, required for developing applications using Java programming language. From a development perspective, JDK is essential [21].

```

C:\Users\sandr>java -version
java version "1.8.0_291"
Java(TM) SE Runtime Environment (build 1.8.0_291-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.291-b10, mixed mode)

C:\Users\sandr>

```

FIGURE 3: JAVA VERSION COMMAND OUTPUT



Spring Boot

Spring Boot is a Java-based open-source framework that provides functionalities for developers to quickly configure and generate new Java web applications [22]. ISOC Reservations backend component was generated using spring initializr:



The screenshot shows the Spring Initializr web application interface. It is divided into three main sections: Project, Project Metadata, and Dependencies.

- Project:**
 - Project:** ☒ Maven Project, ☐ Gradle Project
 - Language:** ☒ Java, ☐ Kotlin, ☐ Groovy
 - Spring Boot:** ☐ 3.0.0 (SNAPSHOT), ☐ 3.0.0 (M1), ☐ 2.7.0 (SNAPSHOT), ☐ 2.7.0 (M2), ☐ 2.6.5 (SNAPSHOT), ☐ 2.6.4, ☐ 2.5.11 (SNAPSHOT), ☒ 2.5.10
- Project Metadata:**
 - Group:** uwtsd.isoc
 - Artifact:** isoc-api
 - Name:** isoc-api
 - Description:** ISOC Reservations Rest API
 - Package name:** uwtsd.isoc.isoc-api
 - Packaging:** ☒ Jar, ☐ War
 - Java:** ☐ 17, ☐ 11, ☒ 8
- Dependencies:**
 - Spring Web:** ☒ WEB. Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
 - Spring Data JPA:** ☒ SQL. Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

FIGURE 4: SPRING INITIALIZR



Spring Data JPA

Spring Data JPA is a framework that simplify the connection to the database by introducing an abstraction layer using Java Persistence API (JPA) repositories. Another advantage is represented by the built-in methods that cover CRUD (create, read, update, delete) operations. Tables are easily mapped to Java classes using Entity and Table annotations [23]. The Spring Data JPA dependency is easily added by using Spring Boot initializr and its version can be modified from Spring Boot application configuration file: pom.xml:

```

<dependency>
  <groupId>org.springframework.data</groupId>
  <artifactId>spring-data-jpa</artifactId>
  <version>2.5.5</version>
</dependency>

```



```
PS D:\iscoresreservationapp-repo\IsocReservationsApp\ui> ng version
```

```
Angular CLI
Angular: 12.2.15
... animations, cli, common, compiler, compiler-cli, core, forms
... localize, platform-browser, platform-browser-dynamic, router

Package                                  Version
-----
@angular-devkit/architect                0.1202.15
@angular-devkit/build-angular            12.2.15
@angular-devkit/core                     12.2.15
@angular-devkit/schematics               12.2.15
@schematics/angular                      12.2.15
rxjs                                     6.6.7
typescript                               4.3.5
```

FIGURE 7: ANGULAR CLI VERSION COMMAND OUTPUT



Bootstrap

Bootstrap is an open-source CSS framework used to develop modern and responsive web applications [27]. ISOC Reservations is using Bootstrap version 4 for frontend component which makes it responsive and increase the accessibility non-functional requirement.



NodeJs

NodeJs is an open-source JavaScript runtime environment. NodeJs is not a mandatory dependency for Angular applications, but it must be installed when development tools (e.g., Angular CLI) or external packages are imported into the project using NPM (Node Package Manager) [28].

ISOC Reservations application is using different external packages, installed from the NPM repository, which drives to the necessity of NodeJs.

```

() package.json M x
() package.json > {} dependencies
1  {
2    "name": "isoc",
3    "version": "1.0.0",
4    "description": "ISOC Reservations",
5    "author": "",
6    "license": "",
7    "private": true,
   ▶ Debug
8  > "scripts": { ...
15  },
16  "dependencies": {
17    "@angular/animations": "~12.2.0",
18    "@angular/common": "~12.2.0",
19    "@angular/compiler": "~12.2.0",
20    "@angular/core": "~12.2.0",
21    "@angular/forms": "~12.2.0",
22    "@angular/platform-browser": "~12.2.0",
23    "@angular/platform-browser-dynamic": "~12.2.0",
24    "@angular/router": "~12.2.0",
25    "bootstrap": "^4.6.1",
26    "jsoneditor": "^9.6.0",
27

```

FIGURE 8: ISOC RESERVATIONS APPLICATION NPM PACKAGE DEPENDENCIES



Microsoft Azure

Microsoft Azure is a cloud provider that provides effective solutions as an alternative to the traditional on-premises data centres, being responsible for purchasing and maintaining the hardware components. This characteristic offers a huge benefit to the cloud consumer which translates into increased performance, availability, scalability, efficiency, security, and reduced cost. Microsoft Azure provides services for different cloud deployment models: public, private and hybrid which satisfy a large variety of business requirements [8].

The provided services are classified in three main categories, based on who is responsible for managing the services, known as cloud models:

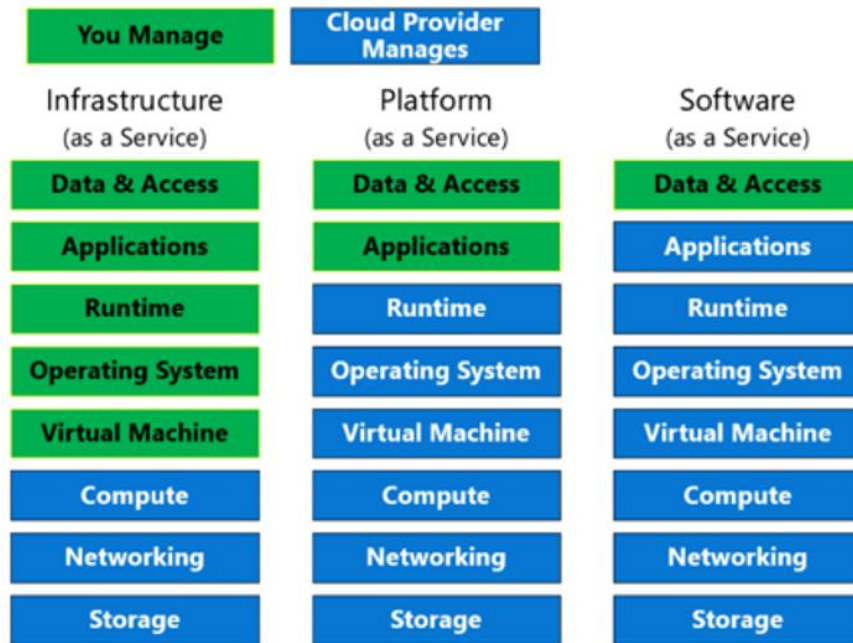


FIGURE 9: COMPARISON BETWEEN MICROSOFT AZURE CLOUD MODELS

ISOC Reservations is using several of the services provided by Microsoft Azure:

Service Name	Cloud Model	Description
Azure App Service	Platform as a Service	A service used to host different types of applications (web, mobile or REST APIs) in Windows or Linux environments without the need of managing the infrastructure by the cloud consumer. [29]
Azure Storage	Platform as a Service	A service that provides impressive scalable storage capabilities. An Azure subscription supports up to 50 storage accounts with a total of 2500 TB storage capacity. [30]
Azure Database for MySQL	Platform as a Service	A service that provides a fully managed MySQL database engine. It has the advantage of high security and availability, efficient monitoring, and alerting mechanisms and automated backups. [31]
Azure Maps	Platform as a Service	A geospatial service that provides APIs for cloud consumers to quickly create

		a geographic information system (GIS). [32]
Azure DevOps	Software as a Service	A set of services that covers the entire lifecycle of the application from planning, collaboration, development to deployment. [33]

TABLE 3: ISOC RESERVATIONS APPLICATION - USED MICROSOFT AZURE SERVICES



Confluence

Confluence is a wiki tool used for improving the collaboration between teams, and to organize the work in one place, being globally accessible over the Internet. The content is organized in “spaces”, which is acting as a main root directory. For ISOC Reservations application, a “Major Project” space was created.

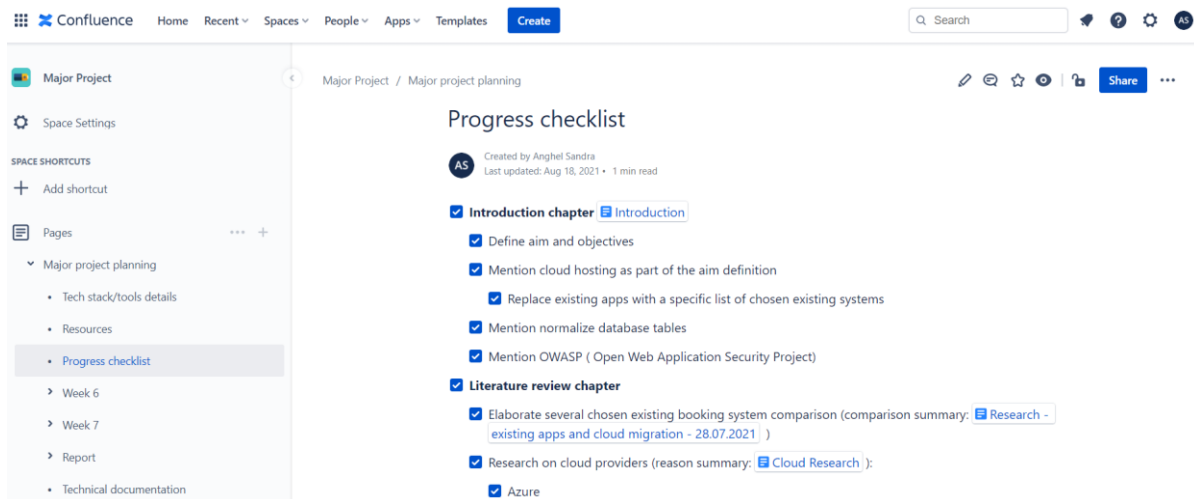


FIGURE 10: CONFLUENCE - MAJOR PROJECT SPACE

The application software used in ISOC Reservations application’s development process are:



Visual Studio Code

Visual Studio Code is a powerful Microsoft source code editor that also supports debugging, intelligent code completion, snippets, and code refactoring for Windows, Linux, and MAC operating systems [34]. In ISOC Reservations application is used to develop, build, and deploy the frontend component.

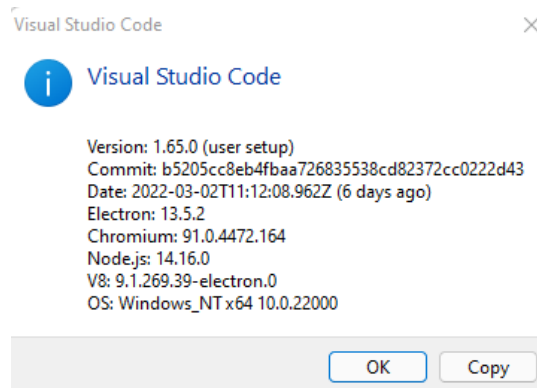


FIGURE 11: VISUAL STUDIO CODE SPECIFICATIONS



IntelliJ IDEA

IntelliJ IDEA is an Integrated Development Environment (IDE) that supports multiple programming languages as Java, Kotlin, JavaScript, TypeScript. It provides functionalities for maximizing the development productivity by providing clever code completion, static code analysis and refactoring [35]. ISOC Reservations backend component is implemented using IntelliJ IDEA Community Edition.

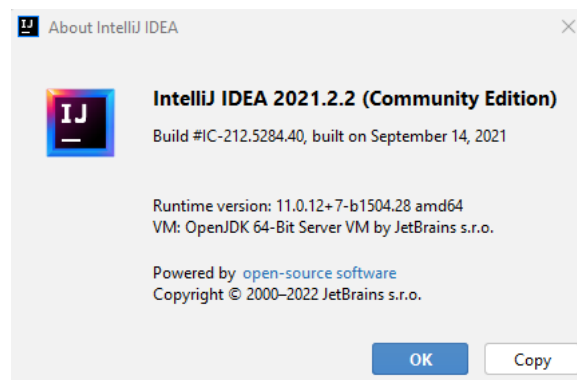


FIGURE 12: INTELLIJ IDEA SPECIFICATIONS



MySQL Workbench

MySQL Workbench is a powerful visual tool that provides functionalities for SQL development, administration, backup, configuration, and management [36]. ISOC Reservations database is design and implemented using MySQL Workbench 8.0 Community

Edition. It is also used for connecting to the Azure Database for MySQL instance from local environment.

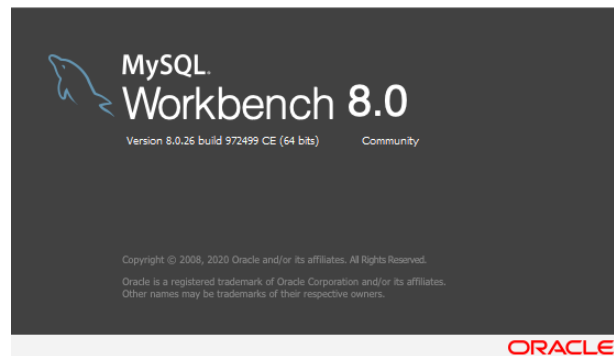


FIGURE 13: MYSQL WORKBENCH VERSION

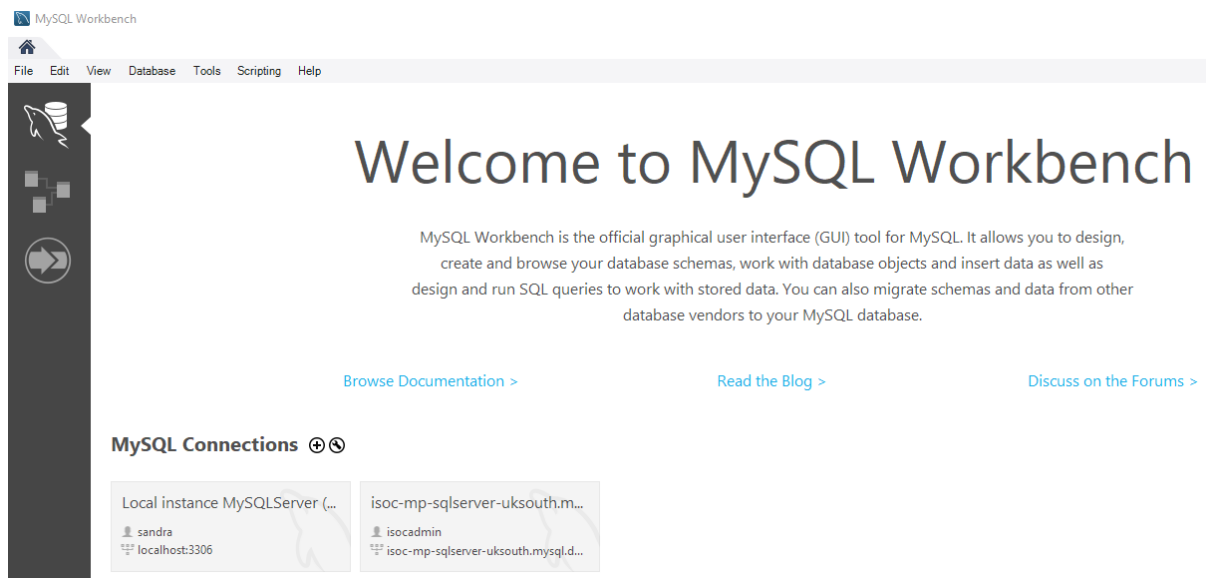


FIGURE 14: MYSQL WORKBENCH CONNECTIONS MANAGEMENT



FileZilla

FileZilla is an open-source tool for simplifying file transfers using File Transfer Protocol [37]. ISOC Reservations application is using FileZilla version 3.58.0 for the deployment process of the backend component.

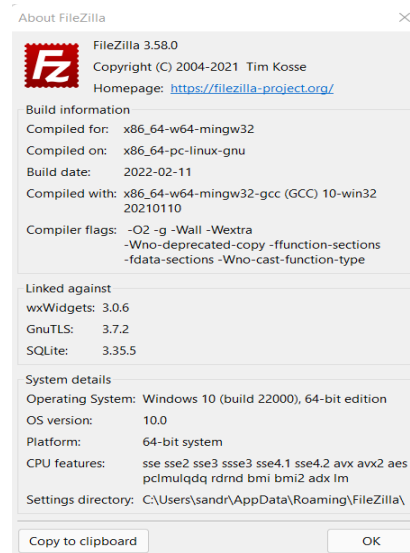


FIGURE 15: FILEZILLA SPECIFICATIONS



Visual Paradigm

Visual Paradigm is a tool used to create UML diagrams by transforming the requirements into accurate design [38]. ISOC Reservations is using Visual Paradigm online free version for creating UML use case diagrams.

2.4. Software Development Life Cycle

The project planning started by building the Gantt Chart which was invented in 1890s and it is used to reflect the required activities displayed against time ([see Appendix A](#)) [39].

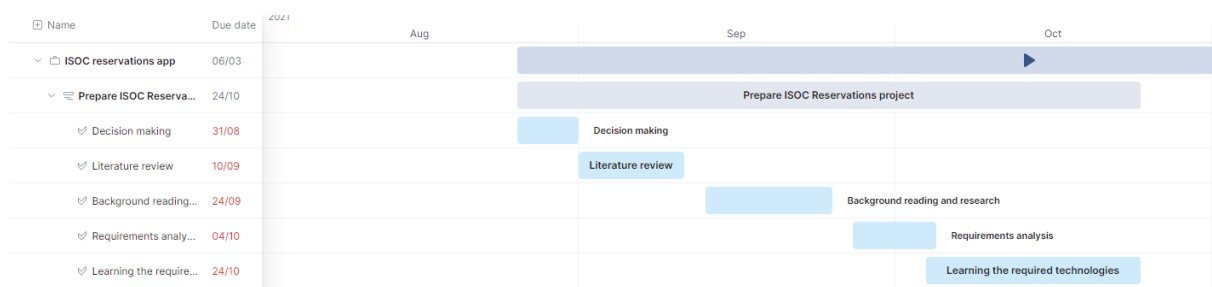


FIGURE 16: ISOC RESERVATIONS GANTT CHART

ISOC Reservations Gantt Chart represents each activity as a horizontal bar which indicates the estimated start and end date. The entire project is estimated to 8 months. It starts with the activities required for preparing the project, including decision making, literature review,

project analysis and continues with activities related to functional requirements. The chart offers an overview, and it is constantly reviewed so the project risks can be easily mitigated.

ISOC Reservations project is further implemented using Agile Software Development Life Cycle Methodology which is an iterative methodology and a way to manage it by breaking up into several phases [40]. The process includes planning, analyse, design, implementation, evaluation and maintenance and two-week sprints are used. Every two weeks, a set of user stories will be planned, analysed, and designed, then developed and tested and in the end, feedback will be obtained.

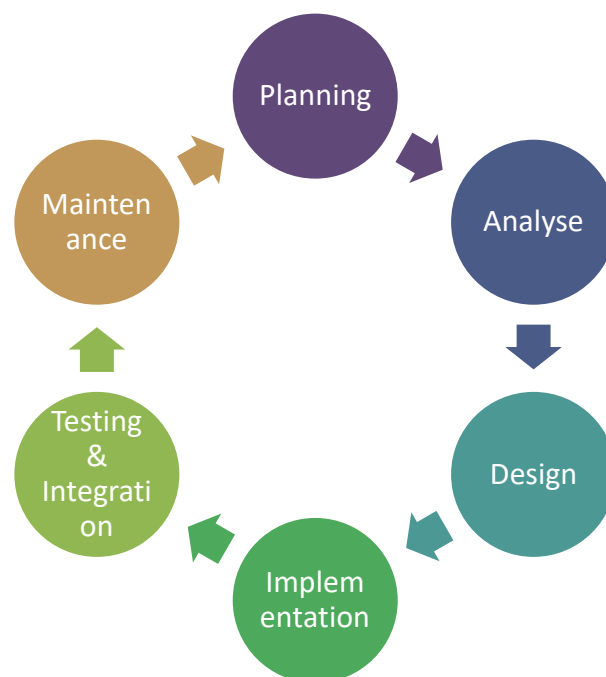


FIGURE 17: AGILE SDLC METHODOLOGY PHASES

This methodology has the advantage of early discovering of bugs and easily and quickly adapt to requirements changes. It also provides predictable delivery dates that aligns with Gantt Chart estimations as per each sprint a set of implemented and tested functionalities can be deployed. The activities are created in Azure Boards in a hierarchical structure represented as epics, features, user stories, tasks, and bugs.

Order	Work Item Type	ID	Title	State	Value Area	Start Date	Target Date
1	Epic	4	> 🏰 ISOC Infrastructure	● New	Architectural	14.10.2021 00:00	25.10.2021 00:00
2	Epic	26	> 🏰 ISOC Business Infrastructure	● New	Architectural	25.10.2021 00:00	19.11.2021 00:00
3	Epic	43	> 🏰 ISOC Authentication & Authorization	● New	Business	18.10.2021 00:00	11.11.2021 00:00
4	Epic	82	▼ 🏰 ISOC_Admin	● New	Business	15.11.2021 00:00	03.12.2021 00:00
	Feature	30	▼ 🍷 App Landing Page	● New	Business	15.11.2021 00:00	18.11.2021 00:00
	User Story	31	▼ 📄 Create landing page	... ● New	Business	15.11.2021 00:00	18.11.2021 00:00
	Task	32	🔗 [UI] App landing page	● Closed		15.11.2021 00:00	15.11.2021 00:00
	Task	33	🔗 [API] Create endpoint for registered restaurants	● New		16.11.2021 00:00	17.11.2021 00:00
	Task	36	🔗 Manual Test	● New		17.11.2021 00:00	17.11.2021 00:00
	Task	42	🔗 UI/UX Design	● New		15.11.2021 00:00	18.11.2021 00:00
	Task	180	🔗 Tech doc	● New		18.11.2021 00:00	18.11.2021 00:00
	Feature	83	▼ 🍷 On-boarding new restaurant	● New	Business	17.11.2021 00:00	26.11.2021 00:00
	User Story	85	> 📄 Create on-boarding admin account	● New	Business	18.11.2021 00:00	23.11.2021 00:00
	User Story	86	> 📄 Create on-boarding restaurant profile	● New	Business	18.11.2021 00:00	26.11.2021 00:00
	Feature	84	> 🍷 Display registered restaurants	● New	Business	17.11.2021 00:00	26.11.2021 00:00
	Feature	96	> 🍷 Delete registered restaurant	● New	Business	22.11.2021 00:00	03.12.2021 00:00
5	Epic	52	> 🏰 Admin	● New	Business	01.12.2021 00:00	14.01.2022 00:00
6	Epic	53	▼ 🏰 Host	● New	Business	17.01.2022 00:00	16.02.2022 00:00
	Feature	60	> 🍷 Bookings management	● New	Business	17.01.2022 00:00	01.02.2022 00:00
	Feature	58	▼ 🍷 Waiting list management	● New	Business	01.02.2022 00:00	08.02.2022 00:00
	User Story	161	▼ 📄 Waiting list management	● New	Business	01.02.2022 00:00	08.02.2022 00:00
	Task	162	🔗 [UI] Create waiting list page	● New		02.02.2022 00:00	07.02.2022 00:00
	Task	163	🔗 [API] Create endpoints for waiting list CRUD	● New		03.02.2022 00:00	07.02.2022 00:00
	Task	164	🔗 [UI] Allocate table for waiting list item	● New		02.02.2022 00:00	07.02.2022 00:00
	Task	165	🔗 [API] Create endpoint for allocating waiting list item	● New		03.02.2022 00:00	07.02.2022 00:00
	Task	166	🔗 Manual Test	● New		08.02.2022 00:00	08.02.2022 00:00
	Task	167	🔗 Tech doc	● New		08.02.2022 00:00	08.02.2022 00:00
	Task	168	🔗 UI/UX Design	● New		03.02.2022 00:00	07.02.2022 00:00
	Feature	59	> 🍷 Capacity monitoring	● New	Business	07.02.2022 00:00	14.02.2022 00:00
7	Epic	54	> 🏰 Customer with account	● New	Business	17.02.2022 00:00	28.02.2022 00:00
8	Epic	130	> 🏰 Anonymous customer	● New	Business	22.02.2022 00:00	28.02.2022 00:00

FIGURE 18: ISOC RESERVATIONS PROJECT - AZURE BOARD

An epic represents a group of features. For example, ISOC_Admin epic covers all the features that a user with ISOCADMIN role can execute in the system. Next level in the hierarchy is represented by features. A feature is a collection of user stories. For example, “On-boarding new restaurant” is a feature that includes two user stories: “Create on-boarding admin account” and “Create on-boarding restaurant profile”. A user story represents a single requirement, and it includes all the tasks needed for implementation. Having the project organized using boards increases the transparency and visibility and improves the tasks management.

Chapter 3 - Analysis

This chapter presents the requirements analysis including functional, non-functional and system requirements. The functional requirements are illustrated using Unified Modeling Language (UML) use case diagrams. In the last section of the chapter, the project risks are listed and analysed.

3.1. Requirements analysis

Karl Wiegers and Joy Beatty define requirements as “a specification of what should be implemented. It describes how the system should behave, or what properties or attributes a system can have” [19]. This definition is eloquent for identifying two main characteristics of a requirement: “what” and “how” and to classify the requirements in several categories: functional requirements and non-functional requirements.

3.1.1. Functional requirements

Functional requirements describe what the system must do in a specific context and define in detail what actions can perform a specific user [19].

ISOC Reservations application has four user roles:

- ISOCADMIN
- ADMIN
- HOST
- CUSTOMER

It also offers functionalities for a user without an assigned role, identified in the system as “Anonymous User”.

Role	Functional requirement
ISOCADMIN	• should be able to login • should be able to register a new restaurant • should be able to view registered restaurants • should be able to create admin account for a specific restaurant • should be able to view created admin accounts

ADMIN	• should be able to login • should be able to view his full name and email details • should be able to change his account's password • should be able to create admin account for the same assigned restaurant • should be able to create host account for the same assigned restaurant • should be able to view existing admin accounts for assigned restaurant • should be able to delete created accounts • should be able to view existing host accounts for assigned restaurant • should be able to modify assigned restaurant's general information • should be able to modify assigned restaurant's business hours • should be able to upload images and menu images for assigned restaurant • should be able to view the assigned restaurant's location on map • should be able to modify assigned restaurant's floor plan
HOST	• should be able to login • should be able to view current day bookings requests • should be able to modify status of a booking • should be able to view assigned restaurant's capacity for breakfast, lunch, and dinner time • should be able to view waiting-list customers • should be able to add a customer on waiting-list • should be able to modify status for a customer from waiting-list • should be able to view all bookings
CUSTOMER	• should be able to create an account • should be able to login • should be able to modify account's full name • should be able to delete the account

	• should be able to view all restaurants • should be able to view a restaurant's details • should be able to send a booking request to a specific restaurant • should be able to view his all bookings • should be able to cancel a booking
--	---

TABLE 4: ISOC RESERVATIONS - FUNCTIONAL REQUIREMENTS FOR EACH ROLE

User without an assigned role	Functional requirement
ANONYMOUS USER	• should be able to create an account • should be able to view all restaurants • should be able to send a booking request to a specific restaurant

TABLE 5: ISOC RESERVATIONS - FUNCTIONAL REQUIREMENTS FOR ANONYMOUS USER

3.1.2. Non-functional requirements

Non-functional requirements describe how the system behaves in a specific context [19]. ISOC Reservations' non-functional requirements includes:

Non-functional requirements		
Usability	Site responsive	Usage of Bootstrap framework
	Design consistency	Usage of same button shapes, colours, tables, forms
	Simple navigation layout	Navigation menu on top and left side of the page
Accessibility	Descriptive HTML elements	Descriptive HTML elements "alt" attribute to describe the images. Usage of HTML descriptive elements: headers (<h3>, <form> and <table>)

Security	OWASP standard	
	SQL Injection	SQL Injection is handled by Spring JPA framework
	Encrypted passwords	Passwords are encrypted and stored in the database
	Authentication and authorization	Authentication is ensured by using Spring Security and JWT mechanism. Authorization is ensured by using role-based access control.

TABLE 6: ISOC RESERVATIONS NON-FUNCTIONAL REQUIREMENTS

3.2. UML – use case diagrams

Michael J. Chonoles defines Unified Modeling Language (UML) as “standardized modeling language consisting of an integrated set of diagrams” used as an abstraction of real world [41]. Use case diagram is one of the UML diagrams that indicates how the user, known as actor, interact with the system.

The following diagrams describe how the actors interact with the system in several scenarios (use cases) and are created based on the functional requirements.

ISOC Reservations actors are:

- ISOCAdmin
- Admin
- Host
- Anonymous Customer
- Customer

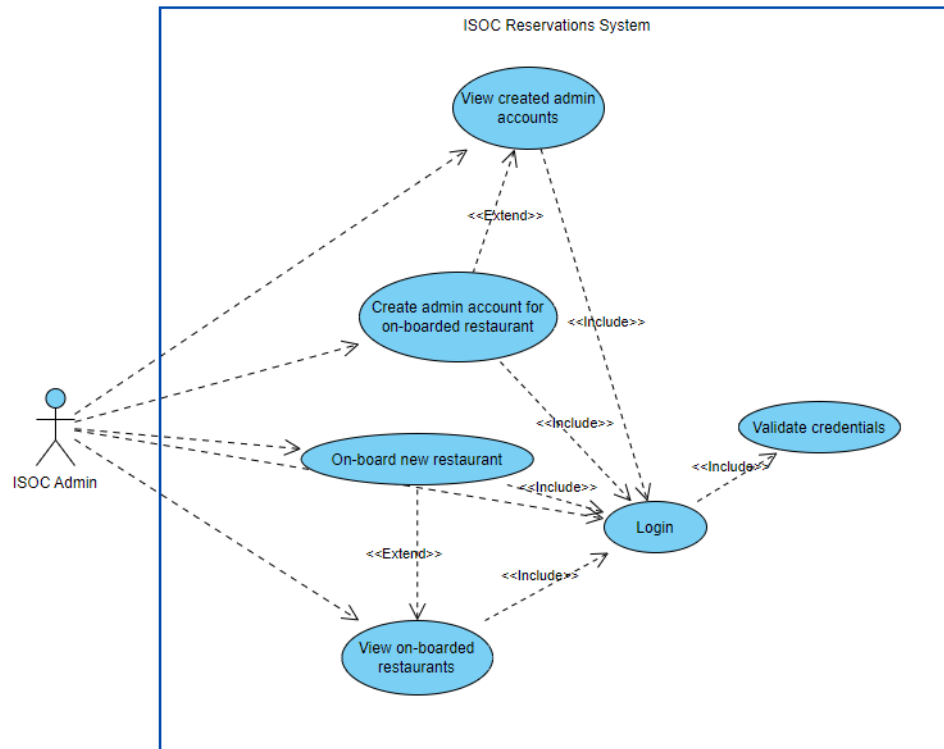


FIGURE 19: ISOCADMIN USE CASE DIAGRAM

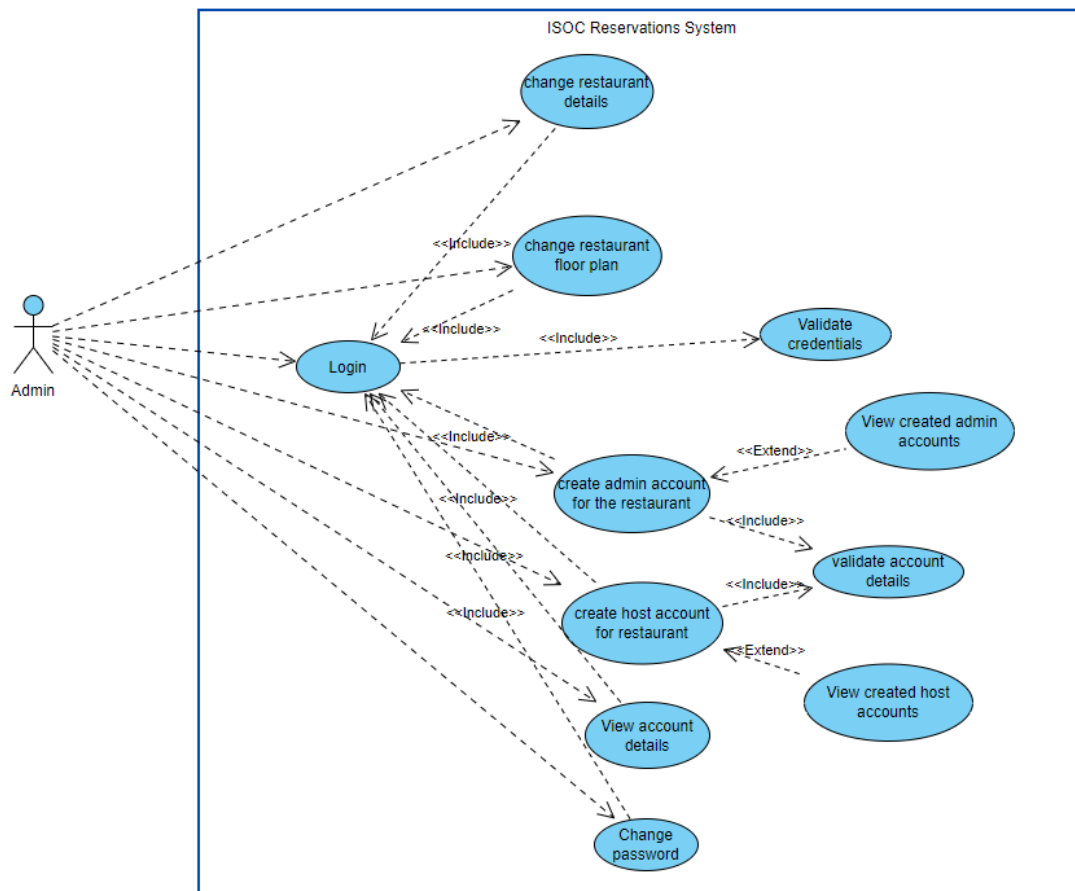


FIGURE 20: ADMIN USE CASE DIAGRAM

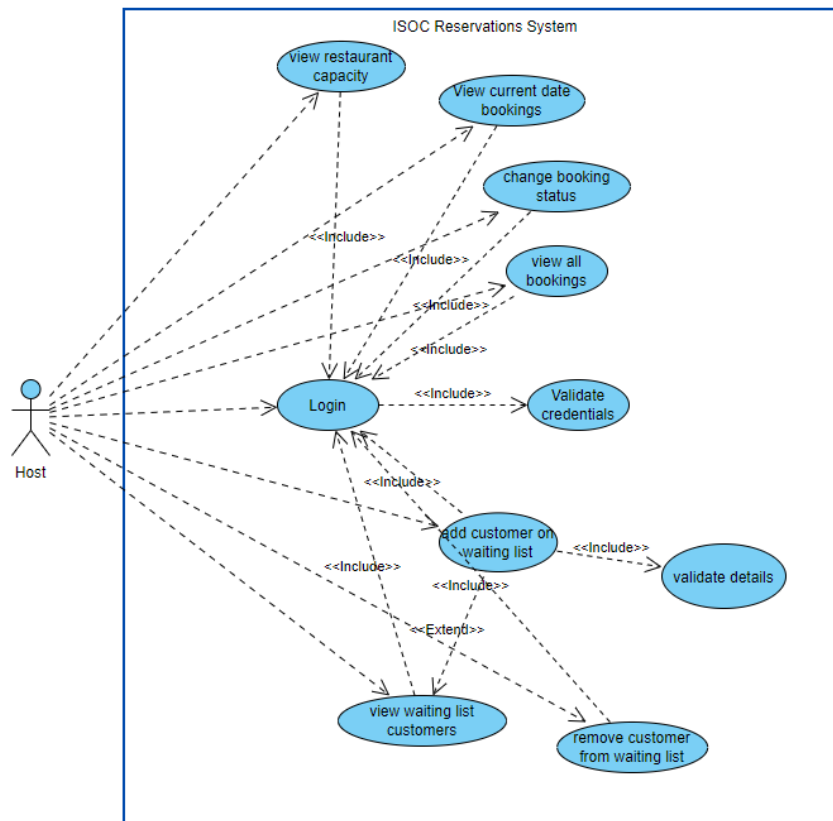


FIGURE 21: HOST USE CASE DIAGRAM

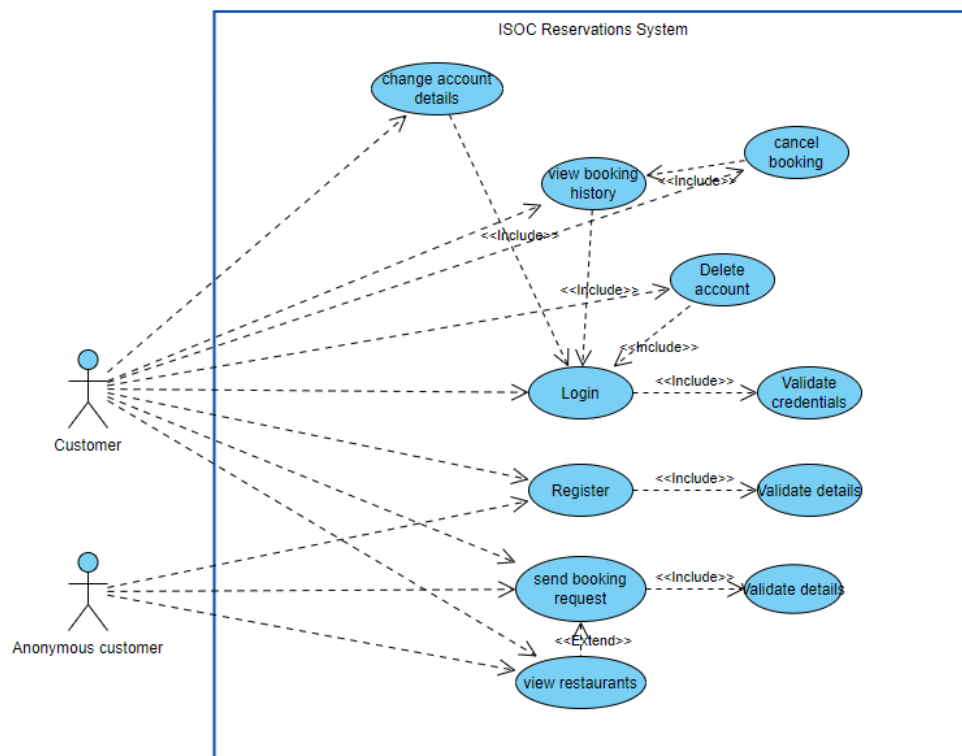


FIGURE 22: CUSTOMER AND ANONYMOUS CUSTOMER USE CASE DIAGRAM

3.3. System requirements

System requirements specify the characteristics needed for a system to successfully run an application [19]. They are classified in:

3.3.1. Hardware requirements

- Processor: Intel i7 (or equivalent)
- Memory: at least 16GB RAM
- Internet connection

3.3.2. Software requirements

- at least Windows 7 Operating System
- at least JDK 8
- NodeJS v14.17.6
- Apache Tomcat 8
- Modern Browser (preferable Microsoft Chrome, Mozilla Firefox)

3.3. Risk analysis

Risk analysis is an essential process used to identify and mitigate different events that could have an impact of the project [42].

ISOC Reservations project risks are:

Risk	Probability of the risk happening	Impact	Mitigation
Hardware failure	Rarely	Could cause delays in delivering the project on time	Purchase new hardware. Ensure regular commits are performed to keep the source code in Azure Repos.
Azure subscription credit fully used	Sometimes	Could increase the cost of deployment	Develop the application using local server and database. Estimate the cost of resources using

			Azure Pricing calculator.
Lack of technical knowledge	Often	Could cause blockage or major delays in delivering the project on time. Could cause increased number of bugs. Could cause missing functionalities.	Create a list of needed technical knowledge (programming languages, frameworks, databases, tools etc.) from the beginning of the project. Perform tutorials to improve the knowledge.
Code issues	Frequently	Could cause major delays in delivering the project.	Investigate existing similar solutions on dedicated programming forums. In case of issues that are taking longer than initially estimated, logged them as “bugs” and revisit as soon as possible.
Frameworks accidentally version updates	Frequently	Could cause major compatibilities issues that might determine an entire rewriting of the application.	Check if a new version is required and make sure the changes from the new version won't affect the current code.
Incorrect planning/estimation	Often	Could cause major delays in delivering the project.	Create a project Gantt-Chart. Review estimations frequently. Adjust estimations based on gained experience during the development phases following Agile Methodology.
Personal health issues	Rarely	Could cause major delays in delivering the project.	Allocate a buffer period in the project planning estimation.

TABLE 7: ISOC RESERVATIONS RISKS ANALYSIS

Chapter 4 - Design

This chapter presents what are the components of ISOC Reservations system and how database was designed following Microsoft best practices recommendations.

4.1. System architecture

System architecture describes what are the main components of a system.

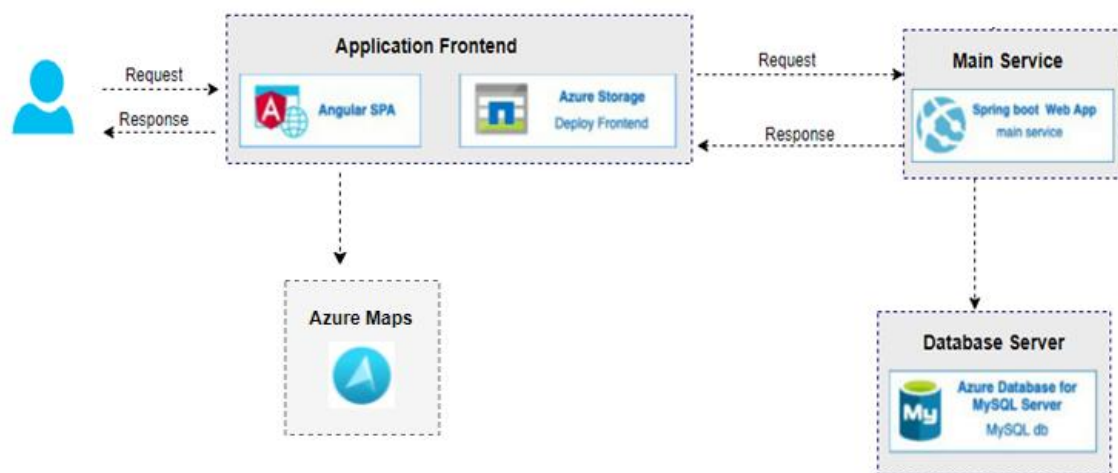


FIGURE 23: ISOC RESERVATIONS SYSTEM ARCHITECTURE

ISOC Reservations systems contains three components: **frontend**, **backend**, and **database**.

Frontend component is implemented mainly using Angular framework and it is hosted in Azure using Azure Storage service. It communicates with backend component and Azure Maps service. ([see Appendix C](#))

Backend component is implemented mainly using Java, Spring Boot, Spring Security and Spring JPA frameworks and it is hosted in Azure using App Service. It is a REST API application that exposes a collection of endpoints, and it is working on a request-response approach. The REST API communicates with the database component using Spring JPA framework which maps the tables into Java classes and increase security by mitigating SQL injection concerns. ([see Appendix D](#))

Database component is implemented using MySQL and it is hosted in Azure using Azure Database for MySQL Server. ([see Appendix B](#))

The components are independent and decoupled and this type of architecture has the advantage of providing a simplified deployment. For example, if frontend component is changing, then it can be independently deployed in Azure without affecting the other components.

4.2. Database design

ISOC Reservations application is using a relational database which stores and provides access to data organized in tables that have specific relationships between them.

According to Microsoft, database design is a process that includes the following steps [43]:

- Identify system's entities
- Translate the entities into tables
- Identify entities properties
- Translate entities properties into columns
- Define relationships between tables

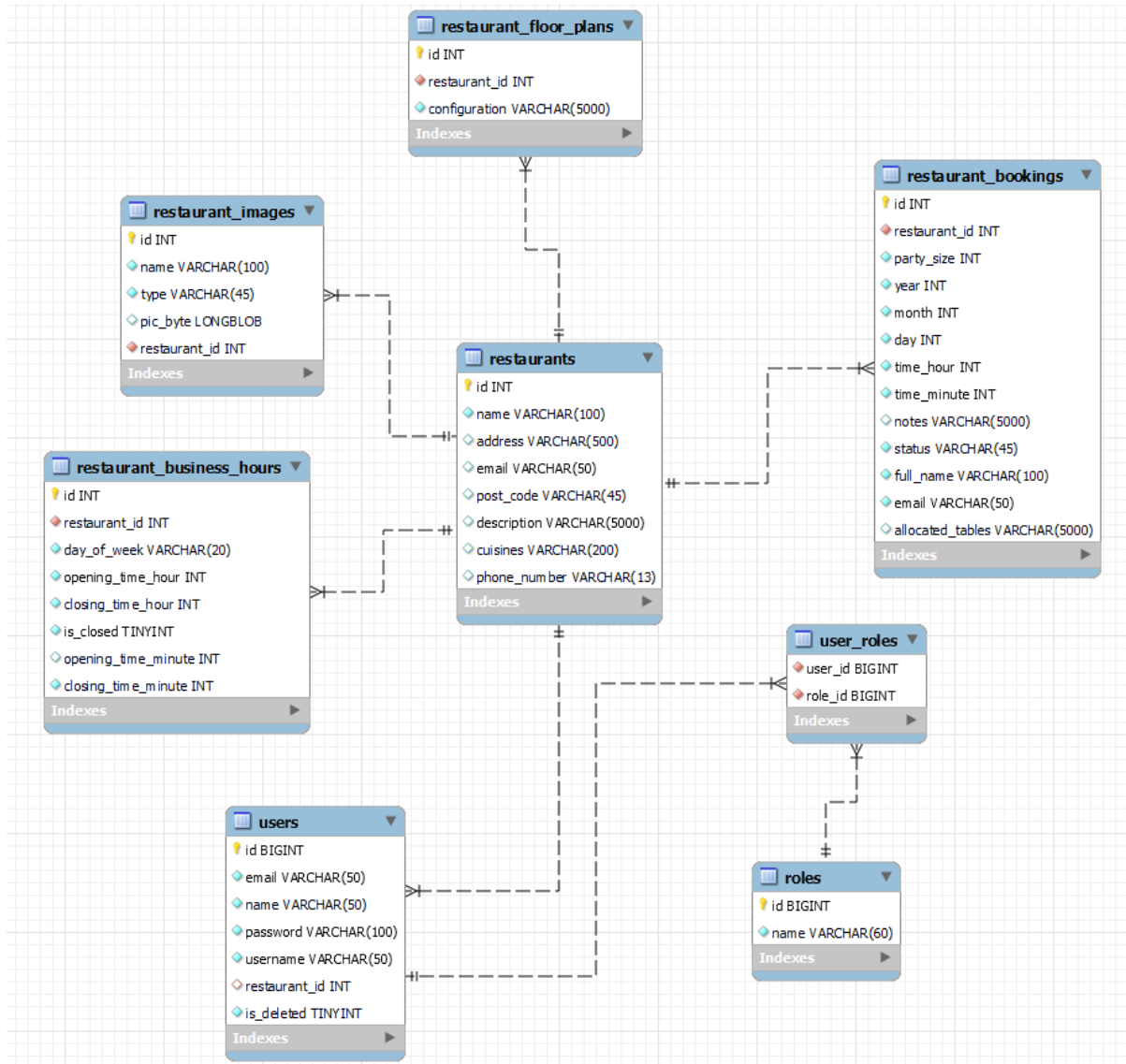


FIGURE 24: ISOC RESERVATIONS ERD DIAGRAM

Database design process is essential to eliminate the data redundancy and ensure its correctness and consistency and the output of it is represented by the Entity Relationship Diagram (ERD).

ISOC Reservations system main entities are user, role, restaurant and booking. Restaurant entity has properties as: image, business hour and floor plan which were divided into separate tables. There is a many-to-many relationship between user and role entities and a one-to-many relationship between restaurant and image, floor plan, business hour and booking entities.

Chapter 5 - Implementation

This chapter presents the implementation of several functionalities of the ISOC Reservations application, Azure infrastructure and deployment process.

5.1. Register

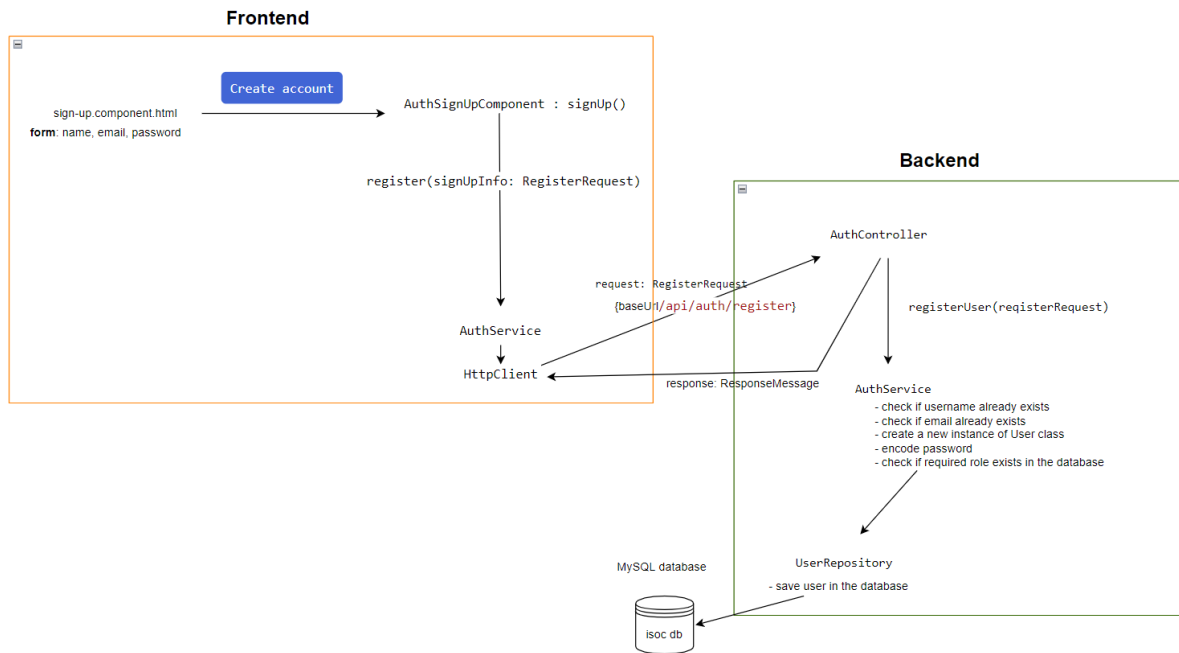


FIGURE 25: HIGH-LEVEL DIAGRAM FOR "REGISTER" IMPLEMENTATION

“Register” flow implementation involves all the components of the ISOC Reservations application. Currently, the username and email have the same value. The user fills a form with the fields: name, email, and password, implemented in `sign-up.component.html`, and then click on “Create account” button. The details from the form are passed into `signUp` method, declared in `AuthSignUpComponent` class, which represents the business logic for the `sign-up.component.html`.

`SignUp` method makes a validation check and if the data is correct then `AuthService`’s `register` method is called. `AuthService` has the responsibility to send a request to the REST API. It has a dependency to `HTTP Client`, which is instantiated and used to call `post` method with an URL and the form data passed from the component. `Post` method sends a request to `AuthController` declared in the backend. The controller has a dependency to `AuthService` injected using `Autowired` annotation and it passes the request data further to the service. The `AuthService` checks if the username and email already exist in the database, then it encodes the password,

create an instance of User class, retrieves the role details from the database based on roleName, assigns the retrieved role to the created User instance and in the end saves the created User instance in the database, using a dependency to UserRepository, which is implemented using Spring Data JPA, and returns the response back to the controller. The controller is returning further the response to frontend in AuthService, which returns it to the component. The component handles the response and displays an error message or makes a redirect to login page.

```
29     signUp(): void {
30
31         if (this.signUpForm.invalid) { return; }
32         this.signUpForm.disable();
33         const roles = new Array<string>();
34         roles.push('customer');
35
36         const signupInfo = new RegisterRequest(
37             this.signUpForm.value.name,
38             this.signUpForm.value.email,
39             this.signUpForm.value.email,
40             this.signUpForm.value.password,
41             roles,
42             null);
43
44         this.authService.register(signupInfo).subscribe(response => {
45             if (response.httpStatus == 'OK') {
46                 this.router.navigate(['auth/sign-in']);
47             } else {
48                 this.toastrService.error('Registration failed!');
49                 this.signUpForm.enable();
50                 this.signUpForm.reset();
51             }
52         });
53     }
```

FIGURE 26: REGISTER - ANGULAR - SINGUP METHOD IMPLEMENTATION

```

13  @Injectable({
14      providedIn: 'root'
15  })
16  export class AuthService {
17
18      private baseUrl = this.appConfig.apiUrl;
19
20      constructor(private http: HttpClient, private appConfig: AppConfigService) { }
21
22  > login(credentials: LoginRequest): Observable<any> { ...
24  }
25
26      register(info: RegisterRequest): Observable<any> {
27          return this.http.post<any>(this.baseUrl + '/api/auth/register', info, httpOptions);
28      }
29
30  > changePassword(info: ChangePasswordRequest) { ...
32  }
33
34  > deleteAccount(email: string){ ...
37  }
38  }

```

FIGURE 27: REGISTER - ANGULAR - AUTHSERVICE IMPLEMENTATION

```

15  @RestController
16  @RequestMapping("/api/auth")
17  @CrossOrigin(origins = "*")
18  public class AuthController {
19
20      @Autowired
21      AuthService authService;
22
23      @PostMapping("/login")
24  32  public JwtResponse authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {...}
25
26      @PostMapping("/register")
27  33  public ResponseMessage registerUser(@Valid @RequestBody RegisterRequest registerRequest) {
28      34      ResponseMessage registerResponse = authService.registerUser(registerRequest);
29      35      return registerResponse;
30      36  }
31  37  }

```

FIGURE 28: REGISTER - REST API - AUTHCONTROLLER IMPLEMENTATION

```

65 public ResponseMessage registerUser(RegisterRequest registerRequest) {
66     if (userRepository.existsByUsername(registerRequest.getUsername())) {...}
67
68     if (userRepository.existsByEmail(registerRequest.getEmail())) {...}
69
70     User user = new User(registerRequest.getName(),
71         registerRequest.getUsername(),
72         registerRequest.getEmail(),
73         encoder.encode(registerRequest.getPassword()),
74         registerRequest.getRestaurantId(),
75         isDeleted: false);
76
77     Set<String> strRoles = registerRequest.getRole();
78     Set<Role> roles = new HashSet<>();
79
80     strRoles.forEach(role -> {
81         switch (role) {
82             case "admin":
83                 Role adminRole = roleRepository.findByName(RoleName.ROLE_ADMIN).orElseThrow(() -> new RuntimeException("User Role Admin not found."));
84                 roles.add(adminRole);
85                 break;
86             case "isocadmin":
87                 Role isocadminRole = roleRepository.findByName(RoleName.ROLE_ISOCADMIN).orElseThrow(() -> new RuntimeException("User Role ISOC Admin not found."));
88                 roles.add(isocadminRole);
89                 break;
90             case "host":
91                 Role pmRole = roleRepository.findByName(RoleName.ROLE_HOST).orElseThrow(() -> new RuntimeException("User Role Host not found."));
92                 roles.add(pmRole);
93                 break;
94             case "customer":
95                 Role userRole = roleRepository.findByName(RoleName.ROLE_CUSTOMER).orElseThrow(() -> new RuntimeException("User Role Customer not found."));
96                 roles.add(userRole);
97                 break;
98             default:
99                 throw new RuntimeException("User Role not found.");
100         }
101     });
102     user.setRoles(roles);
103     userRepository.save(user);
104
105     return new ResponseMessage(HttpStatus.OK, errorMessage: null, payload: "User registered successfully!");
106 }

```

FIGURE 29: REGISTER - REST API - AUTHSERVICE IMPLEMENTATION

5.2. Login

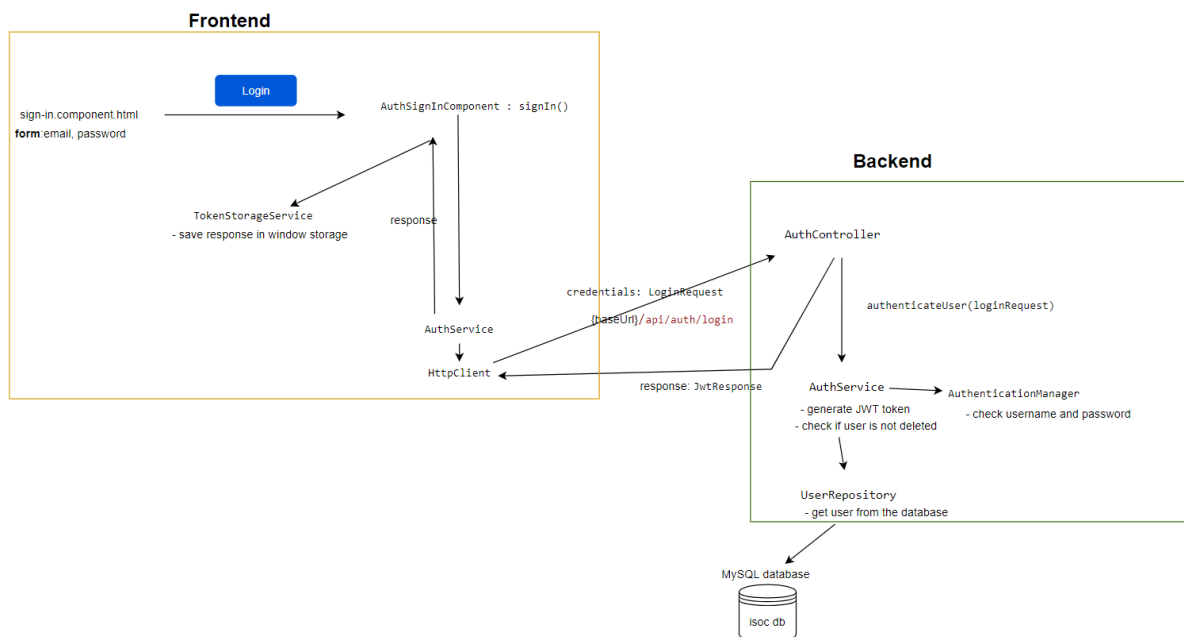


FIGURE 30: HIGH-LEVEL DIAGRAM FOR "LOGIN" IMPLEMENTATION

“Login” flow implementation involves all the components of the ISOC Reservations application. The user fills a form with the fields: email, and password, implemented in `sign-in.component.html`, and then click on “Login” button. The details from the form are passed into `signIn` method, declared in `AuthSignInComponent` class, which represents the business logic for the `sign-in.component.html`.

`SignIn` method makes a validation check and if the data is correct then `authService`’s `login` method is called. `AuthService` has the responsibility to send a request to the REST API. It has a dependency to `HTTP Client`, which is instantiated and used to call `post` method with an URL and the form data passed from the component. `Post` method sends a request to `AuthController` declared in the backend. The controller has a dependency to `AuthService` injected using `Autowired` annotation and it passes the request data further to the service. The `AuthService` tries to authenticate the user using `AuthenticationManager` from Spring Security and sets the authentication in the security context. It also generates the authentication token and checks if the user is not soft deleted from the database and retrieves user details, using an injected dependency of `UserRepository` implemented using Spring Data JPA, to attach `restaurantId` and `user fullname` to the response. The response is returned to the controller which then returns it further to frontend in `AuthService`, which returns it to the component. The component handles the response and saves the token details in the windows storage and based on the user role redirects to a specific dashboard page or displays an error message.

```

34     signIn(): void {
35         if (this.signInForm.invalid) {
36             return;
37         }
38
39         this.signInForm.disable();
40
41         const loginInfo = new LoginRequest(
42             this.signInForm.value.email,
43             this.signInForm.value.password);
44
45         this.authService.login(loginInfo).subscribe(data => {
46             if (data !== null) {
47                 this.tokenStorage.saveToken(data.accessToken);
48                 this.tokenStorage.saveUsername(data.username);
49                 this.tokenStorage.saveRoles(data.authorities);
50                 this.tokenStorage.saveRestaurant(data.restaurantId);
51                 this.tokenStorage.saveFullName(data.fullName);
52
53                 var roles = this.tokenStorage.getRoles();
54
55                 if (roles.includes('ROLE_ISOCADMIN')) {
56                     this.router.navigateByUrl('/redirect-to-isocadmin-dashboard');
57                 } else if (roles.includes('ROLE_ADMIN')) {
58                     this.router.navigateByUrl('/redirect-to-admin-dashboard');
59                 } else if (roles.includes('ROLE_HOST')) {
60                     this.router.navigateByUrl('/redirect-to-host-dashboard');
61                 } else if (roles.includes('ROLE_CUSTOMER')) {
62                     this.router.navigateByUrl('/redirect-to-customer-dashboard');
63                 }
64             } else {
65                 this.signInForm.enable();
66                 this.toastrService.error('Incorrect email or password or you don\'t have an account!');
67             }
68         });
69     }

```

FIGURE 31: LOGIN - ANGULAR - SIGNIN METHOD IMPLEMENTATION

```

13 @Injectable({
14     providedIn: 'root'
15 })
16 export class AuthService {
17
18     private baseUrl = this.appConfig.apiUrl;
19
20     constructor(private http: HttpClient, private appConfig: AppConfigService) { }
21
22     login(credentials: LoginRequest): Observable<any> {
23         return this.http.post<any>(this.baseUrl + '/api/auth/login', credentials, httpOptions);
24     }
25 }

```

FIGURE 32: LOGIN - ANGULAR - AUTHSERVICE IMPLEMENTATION

```

15 @RestController
16 @RequestMapping("/api/auth")
17 @CrossOrigin(origins = "*")
18 public class AuthController {
19
20     @Autowired
21     AuthService authService;
22
23     @PostMapping("/login")
24     public JwtResponse authenticateUser(@Valid @RequestBody LoginRequest loginRequest) {
25         try {
26             JwtResponse tokenResponse = authService.authenticateUser(loginRequest);
27             return tokenResponse;
28         } catch (Exception ex) {...}
29     }
30 }

```

FIGURE 33: LOGIN - REST API - AUTHCONTROLLER IMPLEMENTATION

```

44 @
45 public JwtResponse authenticateUser(LoginRequest loginRequest) throws Exception {
46     Authentication authentication = authenticationManager.authenticate(
47         new UsernamePasswordAuthenticationToken(loginRequest.getUsername(), loginRequest.getPassword()));
48
49     SecurityContextHolder.getContext().setAuthentication(authentication);
50
51     String jwt = jwtProvider.generateJwtToken(authentication);
52     UserDetails userDetails = (UserDetails) authentication.getPrincipal();
53
54     User userFromDb = userRepository.findByUsername((userDetails.getUsername())).get();
55
56     if(userFromDb.getIsDeleted()){
57         throw new Exception("Invalid account");
58     }
59
60     Integer restaurantId = userFromDb.getRestaurantId();
61     String fullName = userFromDb.getName();
62
63     return new JwtResponse(jwt, userDetails.getUsername(), userDetails.getAuthorities(), restaurantId, fullName);
64 }

```

FIGURE 34: LOGIN - REST API - AUTHSERVICE IMPLEMENTATION

```

23 @
24 public String generateJwtToken(Authentication authentication) {
25     UserPrincipal userPrincipal = (UserPrincipal) authentication.getPrincipal();
26     return Jwts.builder()
27         .setSubject((userPrincipal.getUsername()))
28         .setIssuedAt(new Date())
29         .setExpiration(new Date((new Date()).getTime() + jwtExpiration*1000))
30         .signWith(SignatureAlgorithm.HS512, jwtSecret)
31         .compact();
32 }

```

FIGURE 35: LOGIN - REST API - GENERATEJWTOKEN METHOD IMPLEMENTATION

5.3. Send booking request for user without account

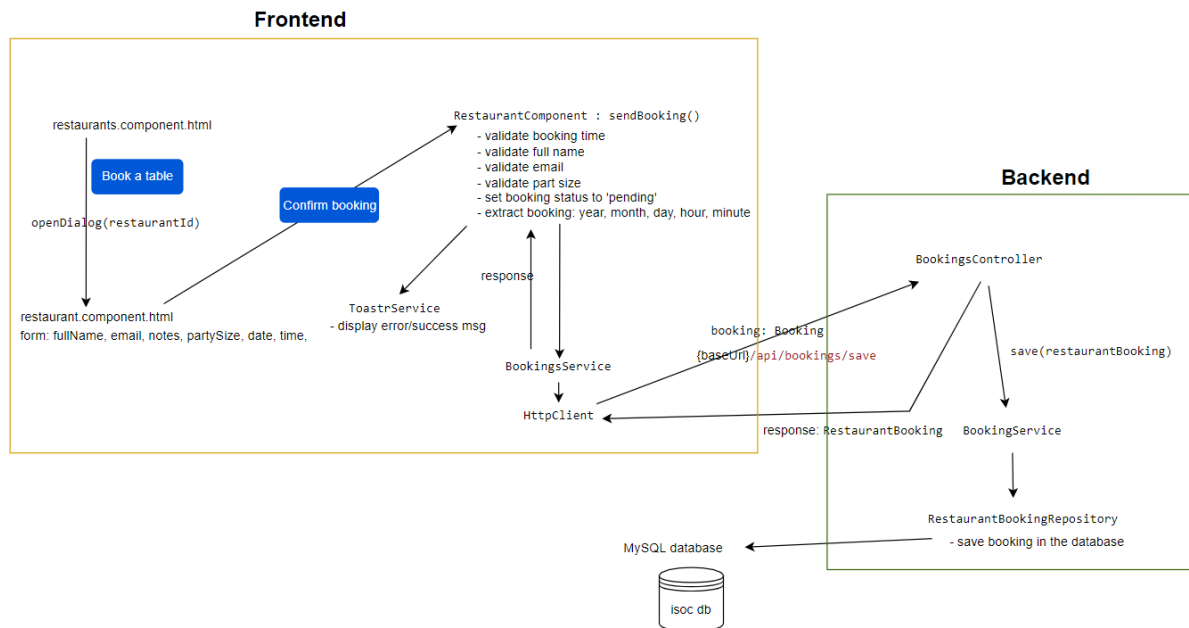


FIGURE 36: HIGH-LEVEL DIAGRAM FOR "SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT" IMPLEMENTATION

“Send booking request for user without account” flow implementation involves all the components of the ISOC Reservations application. The user selects a restaurant from the landing page of the application and clicks on “Book a table” button. A modal component is opened by calling `openDialog` method from the `RestaurantsComponent`. The method sets `restaurantId` and passes it to the modal using the `modalService`, which is a bootstrap built-in modal. Also, the restaurant’s images are filtered based on `restaurantId` and set to the modal.

The opened modal displays details about the restaurant and the user fills a form with details related to the desired booking. Then he clicks on “Confirm” button, which calls `sendBooking` method. `SendBooking` method validates the data (booking time, full name, email, party size) and creates a booking with pending status. The booking data is passed to `BookingsService`. `BookingsService` has the responsibility to send a request to the REST API. It has a dependency to `HTTP Client`, which is instantiated and used to call `post` method with an URL and the form data passed from the component. `Post` method sends a request to `BookingsController` declared in the backend. The controller has a dependency to `BookingService` injected using `Autowired` annotation and it passes the request data further to the service. The `BookingsService` sends the booking object to the `RestaurantBookingRepository`, implemented using `Spring Data JPA`,

which is responsible for saving into the database. The response is returned to the controller which then returns it further to BookingsService, which returns it to the component. The component handles the response displaying a success or error message.

```

24 |     openDialog(id: number) {
25 |         const modalRef = this.modalService.open(RestaurantComponent);
26 |         modalRef.componentInstance.restaurantId = id;
27 |         if (this.userRole === 'ROLE_CUSTOMER') {
28 |             modalRef.componentInstance.userEmail = this.userEmail;
29 |             modalRef.componentInstance.userFullName = this.userFullName;
30 |         }
31 |         const images = this.restaurantsImages.filter(r => r.name.startsWith('restaurant_'+id+'_'));
32 |         if(images && images.length > 0){
33 |             modalRef.componentInstance.restaurantImages = this.restaurantsImages.filter(r => r.name.startsWith('restaurant_'+id+'_'));
34 |         }else{
35 |             const defaultImage = new RestaurantImage();
36 |             defaultImage.name = 'default';
37 |             defaultImage.image = 'assets/images/default-restaurant-image.png';
38 |             modalRef.componentInstance.restaurantImages = [defaultImage];
39 |         }
40 |     }

```

FIGURE 37: SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT - ANGULAR - OPENDIALOG METHOD IMPLEMENTATION

```

103 |     sendBooking() {
104 |         if (this.formModel.selectedTime.hour === null || this.formModel.selectedTime.minute === null) {
105 |             this.toastrService.error('Please select a booking time!');
106 |             return;
107 |         }
108 |         if (!this.formModel.fullName || this.formModel.fullName === '') {
109 |             this.toastrService.error('Please enter your full name');
110 |             return;
111 |         }
112 |         const validEmail = this.validateEmail();
113 |         if (!validEmail) {
114 |             this.toastrService.error('Please enter your email');
115 |             return;
116 |         }
117 |         if (this.formModel.partySize === 0 || this.formModel.partySize < 0) {
118 |             this.toastrService.error('Please select the number of guests (party size)');
119 |             return;
120 |         }
121 |
122 |         const booking = new Booking();
123 |         booking.restaurantId = this.restaurantId;
124 |         booking.year = this.formModel.date.year;
125 |         booking.month = this.formModel.date.month;
126 |         booking.day = this.bookingDate.day;
127 |         booking.timeHour = this.formModel.selectedTime.hour;
128 |         booking.timeMinute = this.formModel.selectedTime.minute;
129 |         booking.partySize = this.formModel.partySize;
130 |         booking.notes = this.formModel.notes;
131 |         booking.status = 'pending';
132 |         booking.email = this.formModel.email;
133 |         booking.fullName = this.formModel.fullName;
134 |
135 |         this.bookingsService.save(booking).subscribe(response => {
136 |             if (response) {
137 |                 this.toastrService.success('Booking request was sent. You will receive a confirmation email!');
138 |                 this.activeModal.close();
139 |             } else {
140 |                 this.toastrService.error('Booking cannot be sent. Please retry or contact the restaurant!');
141 |             }
142 |         });
143 |     }

```

FIGURE 38: SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT - ANGULAR - SENDBOOKING METHOD IMPLEMENTATION

```

14  @Injectable({
15      providedIn: 'root'
16  })
17  export class BookingsService {
18
19      private baseUrl = this.appConfig.apiUrl;
20
21      constructor(private http: HttpClient, private appConfig: AppConfigService) {
22      }
23
24      save(booking: Booking): Observable<any> {
25          return this.http.post<any>(this.baseUrl + '/api/bookings/save', booking, httpOptions);
26      }

```

FIGURE 39: SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT - BOOKINGSSERVICE IMPLEMENTATION

```

14  @RestController
15  @RequestMapping("/api/bookings")
16  @CrossOrigin(origins = "*")
17  public class BookingsController {
18
19      @Autowired
20      private BookingService bookingService;
21
22      @PostMapping("/save")
23      public RestaurantBooking save(@Valid @RequestBody RestaurantBooking restaurantBooking) {
24          try {
25              return bookingService.save(restaurantBooking);
26          } catch (Exception ex) {
27              return null;
28          }
29      }

```

FIGURE 40: SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT - REST API - BOOKINGSCONTROLLER IMPLEMENTATION

```

11  @Service
12  public class BookingService {
13
14      @Autowired
15      private RestaurantBookingRepository bookingRepository;
16
17      public RestaurantBooking save(RestaurantBooking restaurantBooking) {
18          return bookingRepository.save(restaurantBooking);
19      }

```

FIGURE 41: SEND BOOKING REQUEST FOR USER WITHOUT ACCOUNT - REST API - BOOKINGSERVICE IMPLEMENTATION

5.4. Calculate restaurant capacity

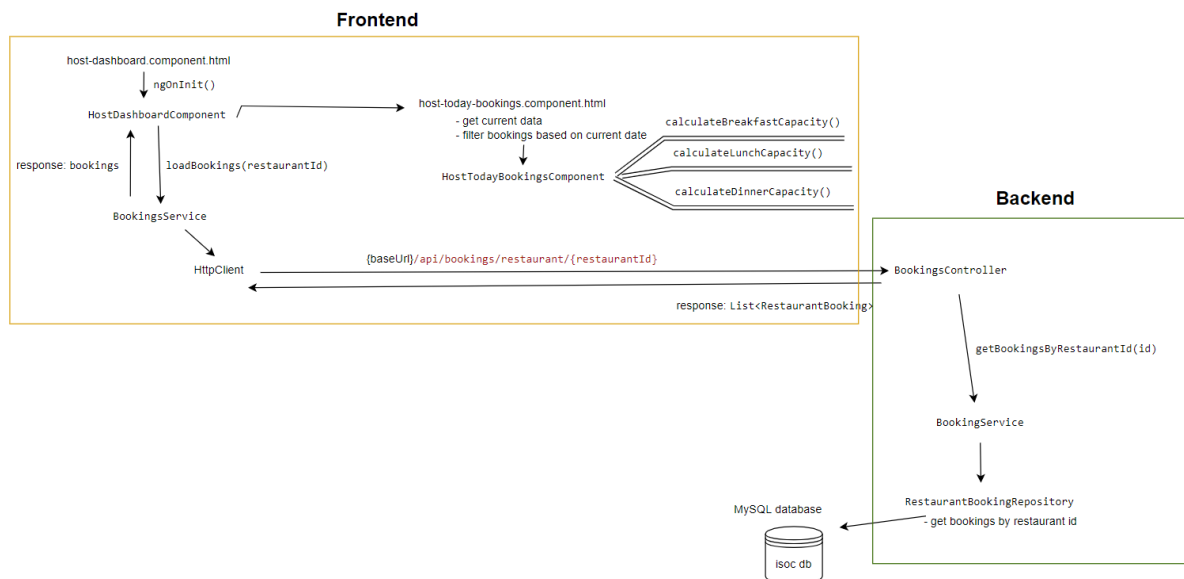


FIGURE 42: HIGH-LEVEL DIAGRAM FOR "CALCULATE RESTAURANT CAPACITY " IMPLEMENTATION

“Calculate restaurant capacity” flow implementation involves indirectly all the components of the ISOC Reservations application, because it needs bookings for the current data. Bookings are retrieved using `loadBookings` method from `HostDashboardComponent` which sends a request to the backed via `BookingsService`. `BookingsService` execute a GET call to the REST API endpoint which returns a collection of bookings as a response. The collection of bookings is further passed into `HostTodayBookingsComponent` and filtered based on current date. The capacity is calculated using methods: `calculateBreakfastCapacity`, `calculateLunchCapacity` and `calculateDinnerCapacity`. `CalculateBreakfastCapacity` retrieves the opening time for the restaurant for the current date and then filters the bookings that have booking time between opening restaurant time and `breakfastTime` (11:30). `CalculateLunchCapacity` filters the bookings that have booking time between `breakfastTime` (11:30) and `lunchTime` (16:30). `CalculateDinnerCapacity` retrieves the closing time for the restaurant for the current date and then filter the bookings that have booking time between `lunchTime` (16:30) and restaurant closing time. Only number of guests from the bookings with status: `seated`, `partial seated` or `confirmed` are extracted from the total capacity of the restaurant.

```

47     private loadBookings(restaurantId: number) {
48         this.bookingsService.getBookingsByRestaurantId(restaurantId).subscribe(response => {
49             if (response) {
50                 this.bookings = response;
51                 this.calendarEvents = this.loadCalendarEvents(this.bookings);
52             }
53         });
54     }

```

FIGURE 43: CALCULATE RESTAURANT CAPACITY - ANGULAR - LOADBOOKINGS METHOD IMPLEMENTATION

```

12 @Injectable({
13     providedIn: 'root'
14 })
15 export class BookingsService {
16
17     private baseUrl = this.appConfig.apiUrl;
18
19     constructor(private http: HttpClient, private appConfig: AppConfigService) {
20     }
21
22 > save(booking: Booking): Observable<any> { ...
24     }
25
26 > getCustomerBookings(email: string): Observable<any> { ...
29     }
30
31 > cancelBooking(bookingId: number): Observable<any> { ...
34     }
35
36     getBookingsByRestaurantId(restaurantId: number): Observable<any> {
37         const url = this.baseUrl + '/api/bookings/restaurant/' + restaurantId;
38         return this.http.get<any>(url);
39     }

```

FIGURE 44: CALCULATE RESTAURANT CAPACITY - ANGULAR - BOOKINGSSERVICE IMPLEMENTATION

```

50     get bookings(): Booking[] {
51         return this._bookings;
52     }
53     @Input() set bookings(value: Booking[]) {
54         if (value) {
55             const currentDate = new Date();
56             this.todayBookings = value.filter(b => b.day === currentDate.getDate() && b.month === (currentDate.getMonth() + 1) && b.year === currentDate.getFullYear());
57             this.waitingListData = this.todayBookings.filter(b => b.status === 'onWaitingList');
58             this.waitingListCollectionSize = this.waitingListData.length;
59             this.getWaitingListData();
60         }
61     }

```

FIGURE 45: CALCULATE RESTAURANT CAPACITY - ANGULAR - FILTER TODAY BOOKINGS IMPLEMENTATION

```

158 calculateBreakfastCapacity() {
159     let allocatedCapacity = 0;
160     if (this.todayBookings && this.restaurantProfile) {
161         const day = new Date().getDate();
162         let openingTime = new TimeModel();
163
164         switch (day) {
165             case 0:
166                 openingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Monday')[0].openingTime;
167                 break;
168             case 1:
169                 openingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Tuesday')[0].openingTime;
170                 break;
171             case 2:
172                 openingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Wednesday')[0].openingTime;
173                 break;
174             case 3:
175                 openingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Thursday')[0].openingTime;
176                 break;
177             case 4:
178                 openingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Friday')[0].openingTime;
179                 break;
180             case 5:
181                 openingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Saturday')[0].openingTime;
182                 break;
183             case 6:
184                 openingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Sunday')[0].openingTime;
185                 break;
186         }
187
188         this.todayBookings.forEach(booking => {
189             const bookingTime = new Date(booking.year, booking.month - 1, booking.day, booking.timeHour, booking.timeMinute, 0, 0);
190             const startTime = new Date(booking.year, booking.month - 1, booking.day, openingTime.hour, openingTime.minute, 0, 0);
191             const breakfastTime = new Date(booking.year, booking.month - 1, booking.day, 11, 30, 0, 0);
192
193             if (booking.status === 'seated' || booking.status === 'partialSeated' || booking.status === 'confirmed') {
194                 if (bookingTime >= startTime && bookingTime <= breakfastTime) {
195                     allocatedCapacity += booking.partySize;
196                 }
197             }
198         });
199     }
200
201     return this.totalCapacity - allocatedCapacity;
202 }

```

FIGURE 46: CALCULATE RESTAURANT CAPACITY - ANGULAR - CALCULATEBREAKFASTCAPACITY METHOD IMPLEMENTATION

```

204 calculateLunchCapacity() {
205     let allocatedCapacity = 0;
206     if (this.todayBookings && this.restaurantProfile) {
207         this.todayBookings.forEach(booking => {
208             const bookingTime = new Date(booking.year, booking.month - 1, booking.day, booking.timeHour, booking.timeMinute, 0, 0);
209             const breakfastTime = new Date(booking.year, booking.month - 1, booking.day, 11, 30, 0, 0);
210             const lunchTime = new Date(booking.year, booking.month - 1, booking.day, 16, 30, 0, 0);
211
212             if (booking.status === 'seated' || booking.status === 'partialSeated' || booking.status === 'confirmed') {
213                 if (bookingTime > breakfastTime && bookingTime <= lunchTime) {
214                     allocatedCapacity += booking.partySize;
215                 }
216             }
217         });
218     }
219
220     return this.totalCapacity - allocatedCapacity;
221 }

```

FIGURE 47: CALCULATE RESTAURANT CAPACITY - ANGULAR - CALCULATELUNCHCAPACITY METHOD IMPLEMENTATION

```

223 calculateDinnerCapacity() {
224     let allocatedCapacity = 0;
225     if (this.todayBookings && this.restaurantProfile) {
226         const day = new Date().getDate();
227         let closingTime = new TimeModel();
228
229         switch (day) {
230             case 0:
231                 closingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Monday')[0].closingTime;
232                 break;
233             case 1:
234                 closingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Tuesday')[0].closingTime;
235                 break;
236             case 2:
237                 closingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Wednesday')[0].closingTime;
238                 break;
239             case 3:
240                 closingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Thursday')[0].closingTime;
241                 break;
242             case 4:
243                 closingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Friday')[0].closingTime;
244                 break;
245             case 5:
246                 closingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Saturday')[0].closingTime;
247                 break;
248             case 6:
249                 closingTime = this.restaurantProfile.businessHours.filter(r => r.dayOfWeek === 'Sunday')[0].closingTime;
250                 break;
251         }
252
253         this.todayBookings.forEach(booking => {
254             const bookingTime = new Date(booking.year, booking.month - 1, booking.day, booking.timeHour, booking.timeMinute, 0, 0);
255             const endTime = new Date(booking.year, booking.month - 1, booking.day, closingTime.hour, closingTime.minute, 0, 0);
256             const lunchTime = new Date(booking.year, booking.month - 1, booking.day, 16, 30, 0, 0);
257
258             if (booking.status === 'seated' || booking.status === 'partialSeated' || booking.status === 'confirmed') {
259                 if (bookingTime > lunchTime && bookingTime <= endTime) {
260                     allocatedCapacity += booking.partySize;
261                 }
262             }
263         });
264     }
265
266     return this.totalCapacity - allocatedCapacity;
267 }

```

FIGURE 48: CALCULATE RESTAURANT CAPACITY - ANGULAR - CALCULATEDINNERCAPACITY METHOD IMPLEMENTATION

5.5. Save floor plan

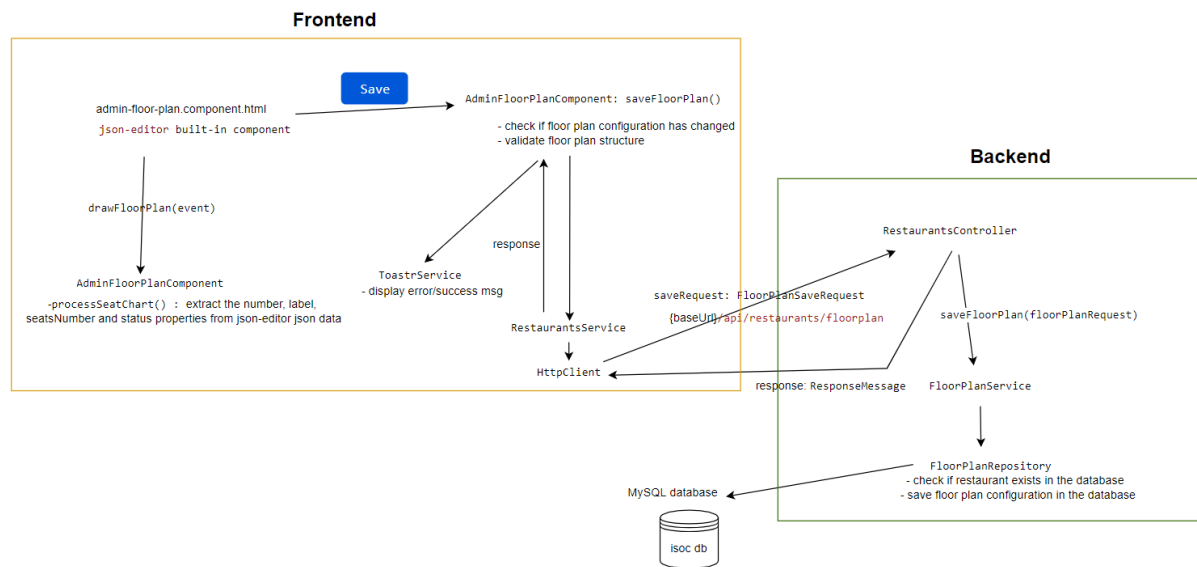


FIGURE 49: HIGH-LEVEL DIAGRAM FOR "SAVE FLOOR PLAN" IMPLEMENTATION

“Save floor plan” flow implementation involves all the components of the ISOC Reservations application. Restaurant floor plan is implemented using a JSON representation and a built-in component from the `json-editor` package installed using NPM. The built-in component exposes a functionality for editing a JSON object which will be displayed by calling the method `drawFloorPlan`. `DrawFloorPlan` iterates each element from the JSON and extracts properties as number, label, seatsNumber, status (for styling purpose). For saving the JSON object, `saveFloorPlan` method is called. It validates the floor plan structure and sends a request to the backend endpoint using `RestaurantsService` which executes a post request to the `RestaurantsController`. The controller has a dependency to `FloorPlanService` injected using `Autowired` annotation and it passes the request data further to the service. The `FloorPlanService` saves the floor plan into the database using an injected dependency of `FloorPlanRepository` implemented using Spring Data JPA. The response is returned to the controller which then returns it further to frontend in `RestaurantsService`, which returns it to the component. The component handles the response and displays a success or error message.

```

41     drawFloorPlan(event: any) {
42         if (event instanceof Event){
43             return;
44         }
45
46         this.floorConfig = [];
47         this.floorConfig = cloneDeep(event);
48         this.processSeatChart(this.floorConfig);
49         this.hasPlanChanged = true;
50     }

```

FIGURE 50: SAVE FLOOR PLAN - ANGULAR - DRAWFLOORPLAN METHOD IMPLEMENTATION

```

84     private processSeatChart(config: any) {
85         if (!config || !(config instanceof Array))
86             return;
87         this.seatmap = [];
88         config.forEach((row) => {
89             var mapObj = { seats: [] };
90             row.forEach((item) => {
91                 var seatObj = {};
92
93                 if (item.seatsNumber != 0) {
94                     seatObj['number'] = item.number;
95                     seatObj['label'] = item.label;
96                     seatObj['seatsNumber'] = item.seatsNumber;
97                     seatObj['status'] = 'available';
98                 } else {
99                     seatObj['label'] = '';
100                 }
101                 mapObj['seats'].push(seatObj);
102             });
103
104             this.seatmap.push(mapObj);
105         });
106     }

```

FIGURE 51: SAVE FLOOR PLAN - ANGULAR - PROCESSEATCHART METHOD IMPLEMENTATION

```

52     saveFloorPlan() {
53         if (!this.hasPlanChanged)
54             return;
55
56         if (!this.floorConfig || !(this.floorConfig instanceof Array) || !(this.floorConfig.length > 0)) {
57             this.toastrService.error('Invalid Floor Plan structure');
58             return;
59         }
60         const request = new FloorPlanSaveRequest();
61         request.floorPlan = JSON.stringify(this.floorConfig);
62         request.restaurantId = this.restaurantId;
63
64         this.restaurantsService.saveFloorPlan(request).subscribe(response => {
65             if (response.httpStatus === 'OK') {
66                 this.toastrService.success('Floor Plan saved!');
67                 this.hasPlanChanged = false;
68             } else {
69                 this.toastrService.error('Floor Plan cannot be saved!Please contact ISOC Administrator');
70             }
71         });
72     }

```

FIGURE 52: SAVE FLOOR PLAN - ANGULAR - SAVEFLOORPLAN METHOD IMPLEMENTATION

```

13 @Injectable({
14     providedIn: 'root'
15 })
16 export class RestaurantsService {
17
18     private baseUrl = this.appConfig.apiUrl;
19
20     constructor(private http: HttpClient, private appConfig: AppConfigService) { }
21
22 >     save(restaurant: Restaurant): Observable<any> { ...
24     }
25
26 >     getAll(): Observable<any> { ...
28     }
29
30 >     getFloorPlan(restaurantId: number): Observable<any> { ...
33     }
34
35     saveFloorPlan(saveRequest: FloorPlanSaveRequest): Observable<any> {
36         const url = this.baseUrl + '/api/restaurants/floorplan';
37         return this.http.post<any>(url, saveRequest, httpOptions);
38     }

```

FIGURE 53: SAVE FLOOR PLAN - ANGULAR - RESTAURANTSSERVICE IMPLEMENTATION

```

61 @PostMapping("/floorplan")
62 public ResponseMessage saveFloorPlan(@Valid @RequestBody FloorPlanSaveRequest floorPlanRequest) {
63     try {
64         int affectedRows = floorPlanService.saveFloorPlan(floorPlanRequest);
65         if (affectedRows == 0) {
66             return new ResponseMessage(HttpStatus.NOT_FOUND, errorMessage: "Floor Plan for restaurant not found", payload: null);
67         }
68         return new ResponseMessage(HttpStatus.OK, errorMessage: null, payload: null);
69     } catch (Exception ex) {
70         return new ResponseMessage(HttpStatus.INTERNAL_SERVER_ERROR, ex.getMessage(), payload: null);
71     }
72 }

```

FIGURE 54: SAVE FLOOR PLAN - REST API - ENDPOINT IMPLEMENTATION

```

9 @Service
10 public class FloorPlanService {
11
12     @Autowired
13     FloorPlanRepository floorPlanRepository;
14
15     public FloorPlan getByRestaurantId(Integer restaurantId) {...}
16
17     @Transactional
18     public int saveFloorPlan(FloorPlanSaveRequest floorPlanRequest) {
19         FloorPlan fp = floorPlanRepository.findByRestaurantId(floorPlanRequest.getRestaurantId());
20         if(fp == null){
21             floorPlanRepository.save(new FloorPlan(floorPlanRequest.getRestaurantId(), floorPlanRequest.getFloorPlan()));
22             return 1;
23         }else {
24             return floorPlanRepository.saveFloorPlan(floorPlanRequest.getRestaurantId(), floorPlanRequest.getFloorPlan());
25         }
26     }
27 }

```

FIGURE 55: SAVE FLOOR PLAN - REST API - FLOORPLANSERVICE IMPLEMENTATION

```

11 @Repository
12 public interface FloorPlanRepository extends JpaRepository<FloorPlan, Long> {
13
14     @Query(value = "SELECT p.* FROM restaurant_floor_plans p " +
15             "WHERE p.restaurant_id = :restaurantId " +
16             "LIMIT 1", nativeQuery = true)
17     FloorPlan findByRestaurantId(@Param("restaurantId") Integer restaurantId);
18
19     @Modifying
20     @Transactional
21     @Query("UPDATE FloorPlan fp " +
22             "SET fp.configuration = :floorPlan " +
23             "WHERE fp.restaurantId = :restaurantId")
24     int saveFloorPlan(@Param("restaurantId") Integer restaurantId, @Param("floorPlan") String floorPlan);
25 }

```

FIGURE 56: SAVE FLOOR PLAN - REST API - FLOORPLANREPOSITORY IMPLEMENTATION

5.6. Display restaurant location on map

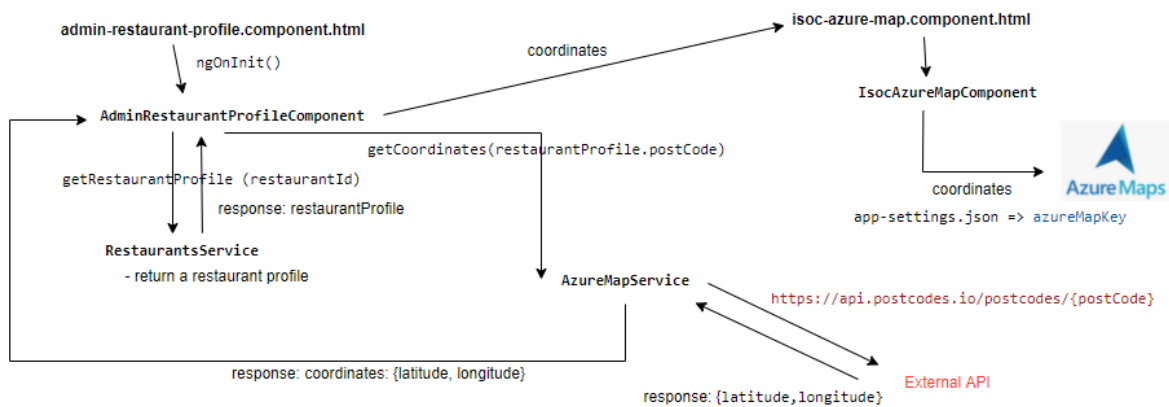


FIGURE 57: HIGH-LEVEL DIAGRAM FOR "DISPLAY RESTAURANT LOCATION ON MAP" IMPLEMENTATION

“Display restaurant location on map” flow implementation involves only frontend component of the ISOC Reservations application and Azure Maps and a third-party API for retrieving the coordinates for a specific post code. The restaurant profile is retrieved using `getRestaurantProfile` method and the response contains the restaurant post code which is passed to `AzureMapService` for calling the third-party API to get the coordinates (latitude and longitude). The response is passed to `IsocAzureMapComponent` to create the map using `createMap` method. This method sets the `azureMapKey`, specified in `app-settings.json` configuration file from assets folder, for establishing the connection to the Azure Maps service and set the coordinates on the map.

```

73 | private getRestaurantProfile(restaurantId: number) {
74 |     this.restaurantsService.getRestProfile(restaurantId).subscribe(response => {
75 |         if (response) {
76 |             this.restaurantProfile = response;
77 |             if (this.restaurantProfile.postCode && this.restaurantProfile.postCode !== '') {
78 |                 this.azureMapService.getCoordinates(this.restaurantProfile.postCode).subscribe(response => {
79 |                     if (response.status === 200) {
80 |                         this.coordinates = new Coordinates();
81 |                         this.coordinates.latitude = response.result.latitude;
82 |                         this.coordinates.longitude = response.result.longitude;
83 |                     }
84 |                 });
85 |             }
86 |         }
87 |     });
88 | }
  
```

FIGURE 58: DISPLAY RESTAURANT LOCATION ON MAP - ANGULAR - GETRESTAURANTPROFILE METHOD IMPLEMENTATION

```

10 @Injectable({
11   providedIn: 'root'
12 })
13 export class AzureMapService {
14
15   constructor(private http: HttpClient, private appConfig: AppConfigService) { }
16
17   getCoordinates(postCode: string): Observable<any> {
18     return this.http.get<any>(this.appConfig.postCodeApi + '/' + postCode, httpOptions);
19   }
20 }

```

FIGURE 59: DISPLAY RESTAURANT LOCATION ON MAP - AZUREMAPSERVICE IMPLEMENTATION

```

6   @Component({
7     selector: 'isoc-azure-map',
8     templateUrl: 'isoc-azure-map.component.html',
9     styleUrls: ['isoc-azure-map.component.scss']
10  })
11  export class IsocAzureMapComponent implements OnInit {
12    private _coordinates: Coordinates;
13
14    @Input()
15    set coordinates(coordinates: Coordinates) {
16      this._coordinates = coordinates;
17      if(this._coordinates){
18        this.createMap();
19      }
20    }

```

FIGURE 60: DISPLAY RESTAURANT LOCATION ON MAP - ANGULAR - COORDINATES SETTER IMPLEMENTATION

```

29   private createMap() {
30
31     let azureMapKey = this.appConfig.azureMapKey;
32     if (azureMapKey !== '') {
33       var map = new Map(this.mapContainer.nativeElement, {
34         center: [this._coordinates.longitude, this._coordinates.latitude],
35         zoom: 16,
36         view: "Auto",
37         authOptions: {
38           authType: AuthenticationType.subscriptionKey,
39           subscriptionKey: azureMapKey,
40         },
41       });
42
43       map.events.add('ready', () => {
44         var dataSource = new source.DataSource();
45         map.sources.add(dataSource);
46         var point = new Shape(new data.Point([this._coordinates.longitude, this._coordinates.latitude]));
47         dataSource.add([point]);
48         map.layers.add(new layer.SymbolLayer(dataSource, null));
49       });
50     }
51   }

```

FIGURE 61: DISPLAY RESTAURANT LOCATION ON MAP - ANGULAR - CREATEMAP METHOD IMPLEMENTATION

5.7. Azure infrastructure

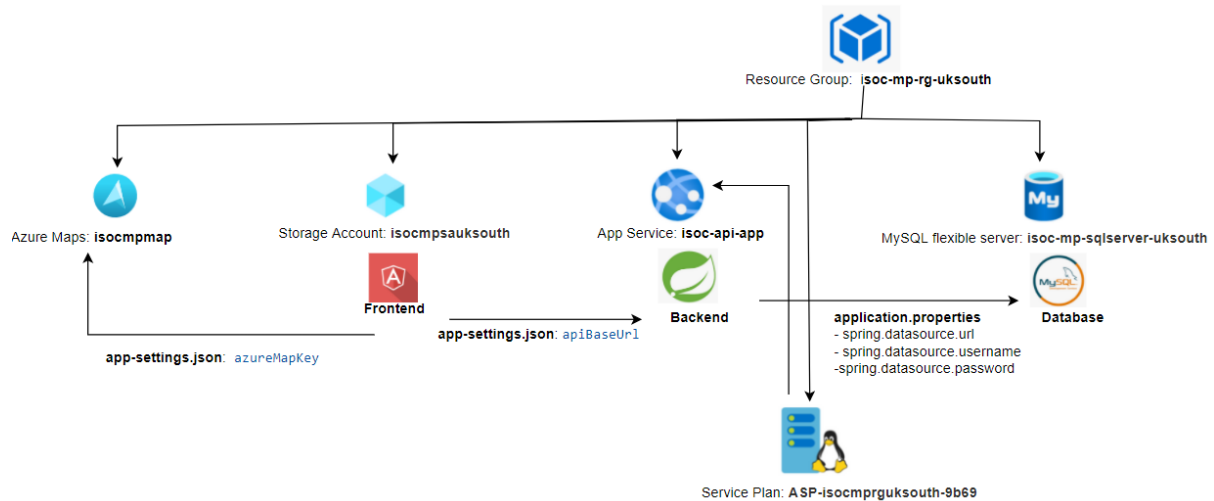


FIGURE 62: HIGH-LEVEL DIAGRAM FOR "AZURE INFRASTRUCTURE" IMPLEMENTATION

Azure infrastructure was created using Azure Portal:

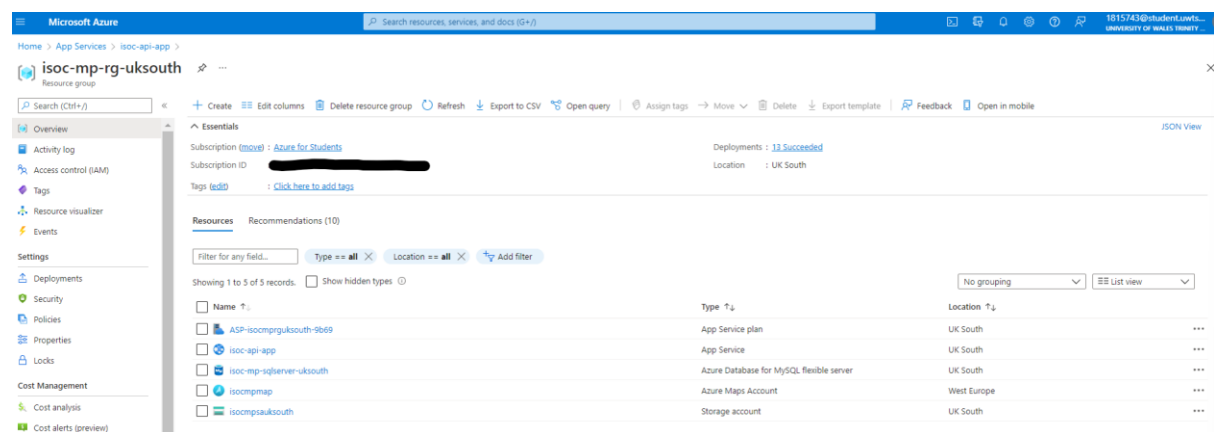


FIGURE 63: AZURE PORTAL

All Azure Services must be created under a resource group that indicates in which region is located the Microsoft Azure data centre. Azure Storage Account contains the frontend artifact and communicates with Azure Maps and App Service, which contains the backend artifact, using configurations defined in `app-settings.json` file. App Service is communicating with the Azure Database for MySQL using configurations defined in `application.properties` file. App Service also needs an App Service Plan for the creation process which defines the compute resources capacity used to run the web application.

5.8. Deployment process

Deployment process involves all the components of the ISOC Reservations application.

Frontend deployment is performed using Azure Tools extension for Visual Studio Code.

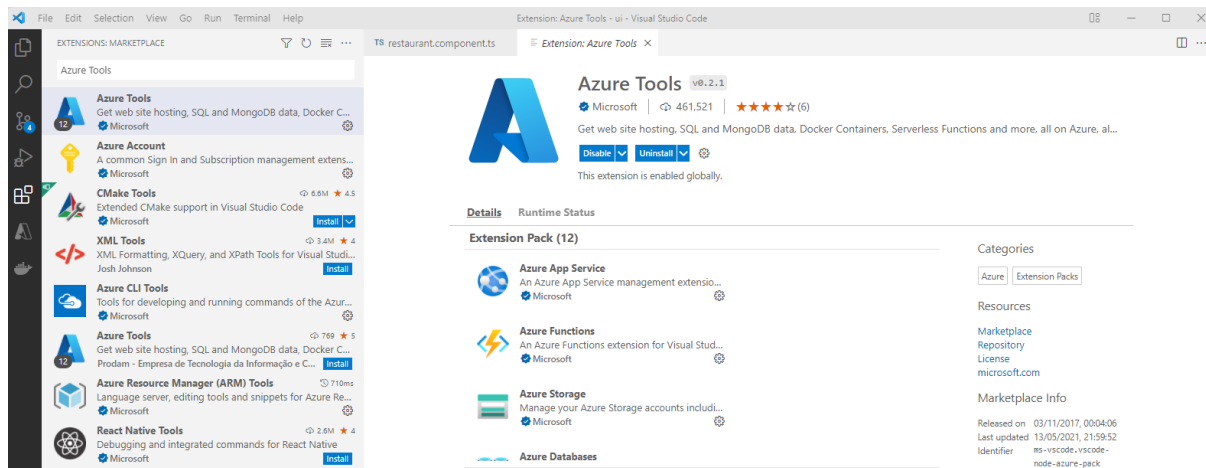


FIGURE 64: AZURE TOOLS EXTENSION FOR VISUAL STUDIO CODE

First step of the deployment is represented by running in Visual Studio Code terminal the command: `ng build` which creates an output in `dist` directory.

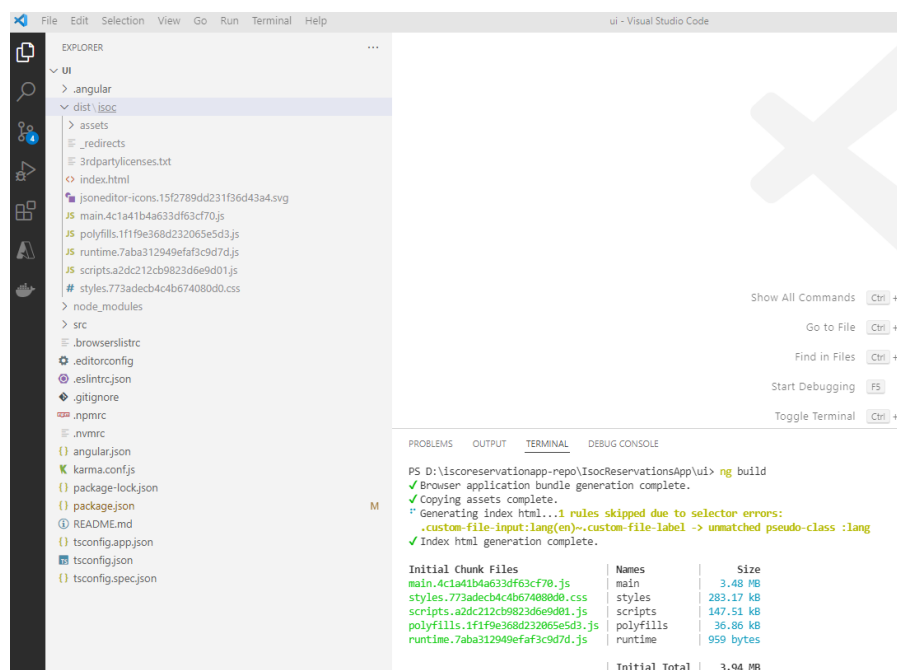


FIGURE 65: RUNNING NG BUILD COMMAND AND DIST DIRECTORY OUTPUT

Second step of the deployment is represented by navigating to Azure Tools extension in Storage section and connect to Azure Subscription, then select “Deploy to Static Website via Azure Storage...” and choose dist/isoc directory.

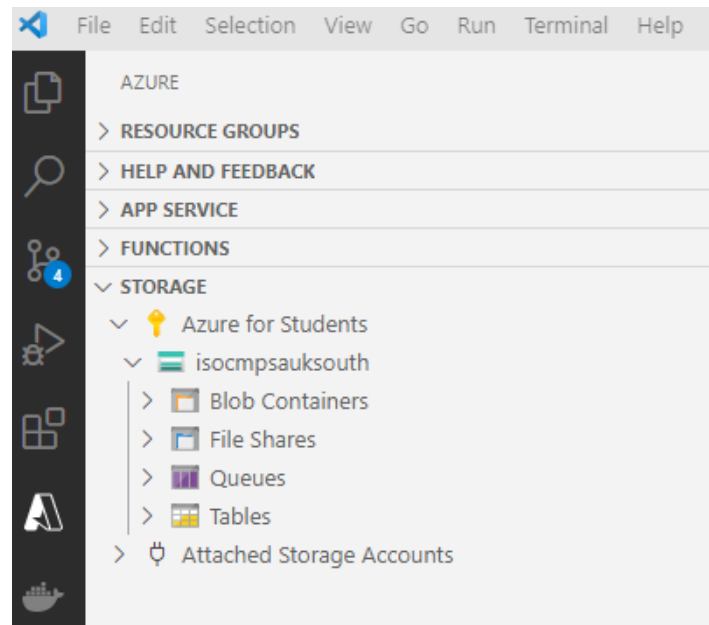


FIGURE 66: AZURE TOOLS EXTENSION - CONNECT TO AZURE SUBSCRIPTION

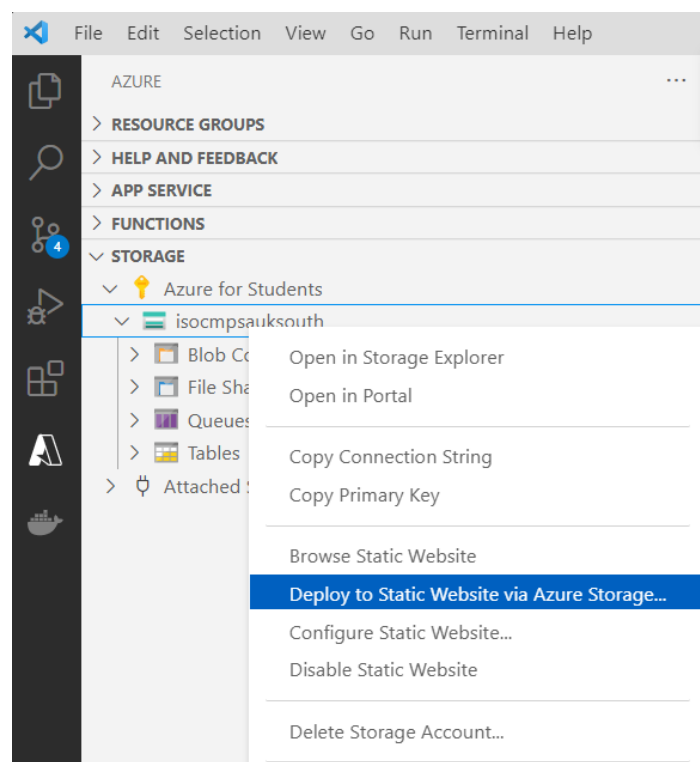


FIGURE 67: AZURE TOOLS EXTENSION - DEPLOY TO STATIC WEBSITE VIA AZURE STORAGE

Backend deployment is performed using FileZilla.

First step of the deployment is represented by running in IntelliJ IDE the commands: *mvn clean* and *mvn package build* which creates an output in *target* directory.

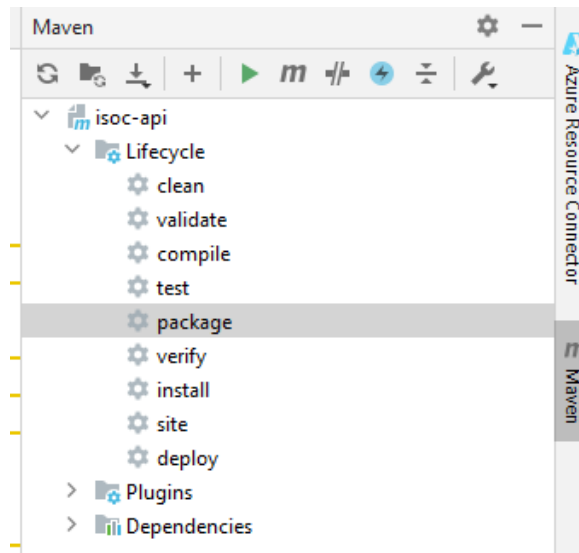


FIGURE 68: INTELLIJ IDE - MAVEN COMMANDS

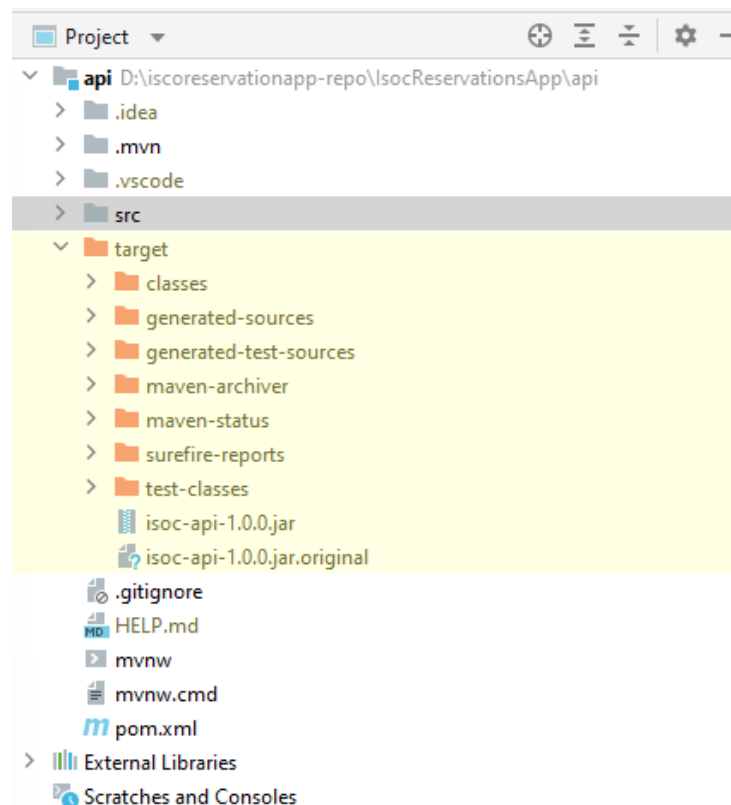


FIGURE 69: TARGET DIRECTORY OUTPUT

Second step of the deployment is represented by connecting FileZilla to Azure App Service using the FTP Credentials.

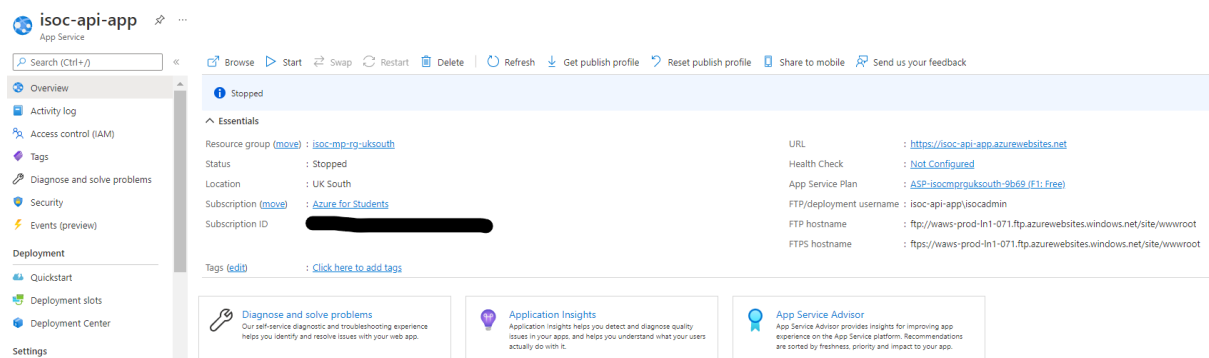


FIGURE 70: AZURE APP SERVICE FTP CREDENTIALS

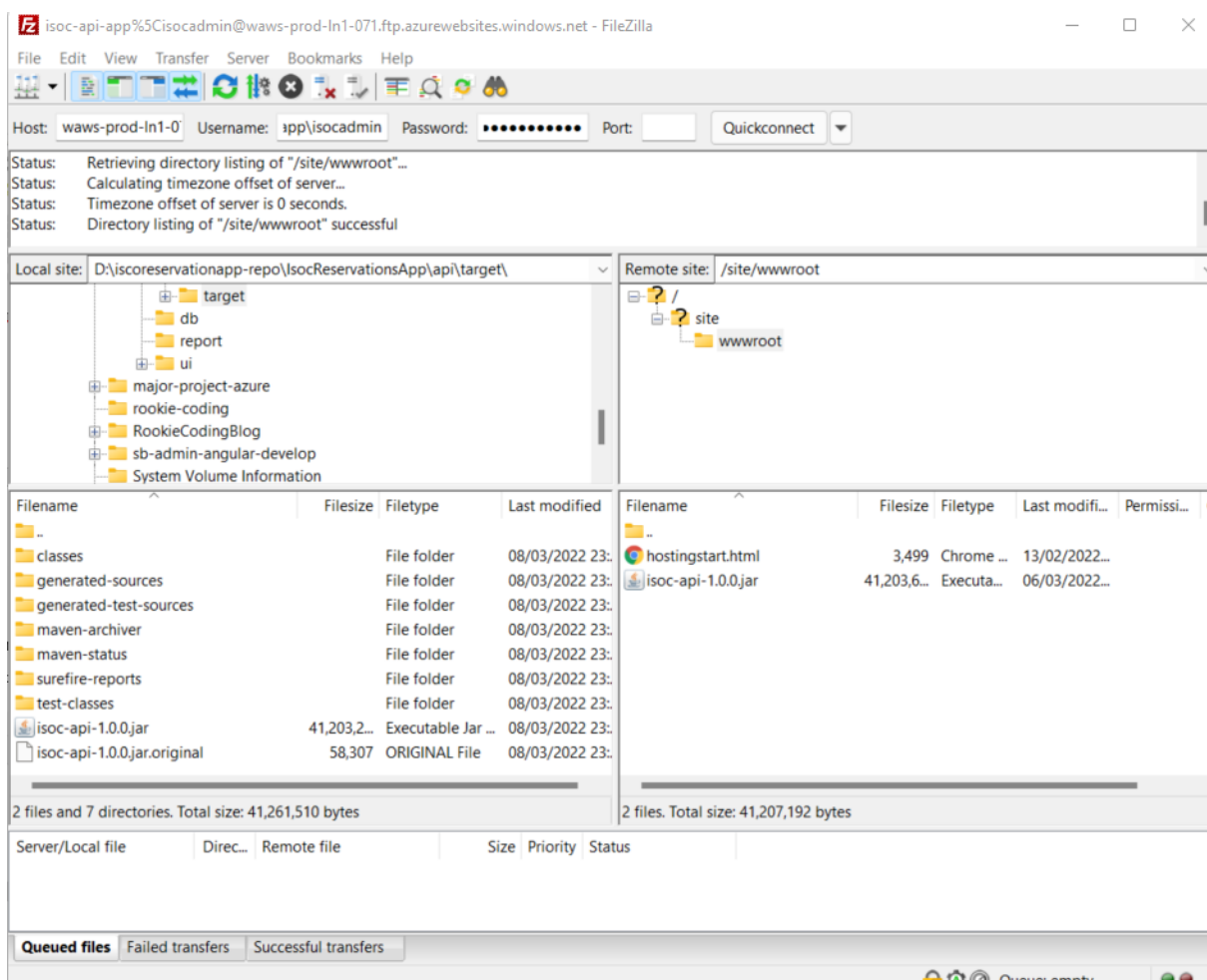


FIGURE 71: FILEZILLA CONNECTION TO AZURE APP SERVICE AND ARTIFACT UPLOAD

In the end, the isoc-api-1.0.0.jar file is uploaded from target directory into wwwroot directory.

Database deployment is performed using MySQL Workbench.

The development of the database is made on the local instance and the scripts are exported using MySQL Workbench.

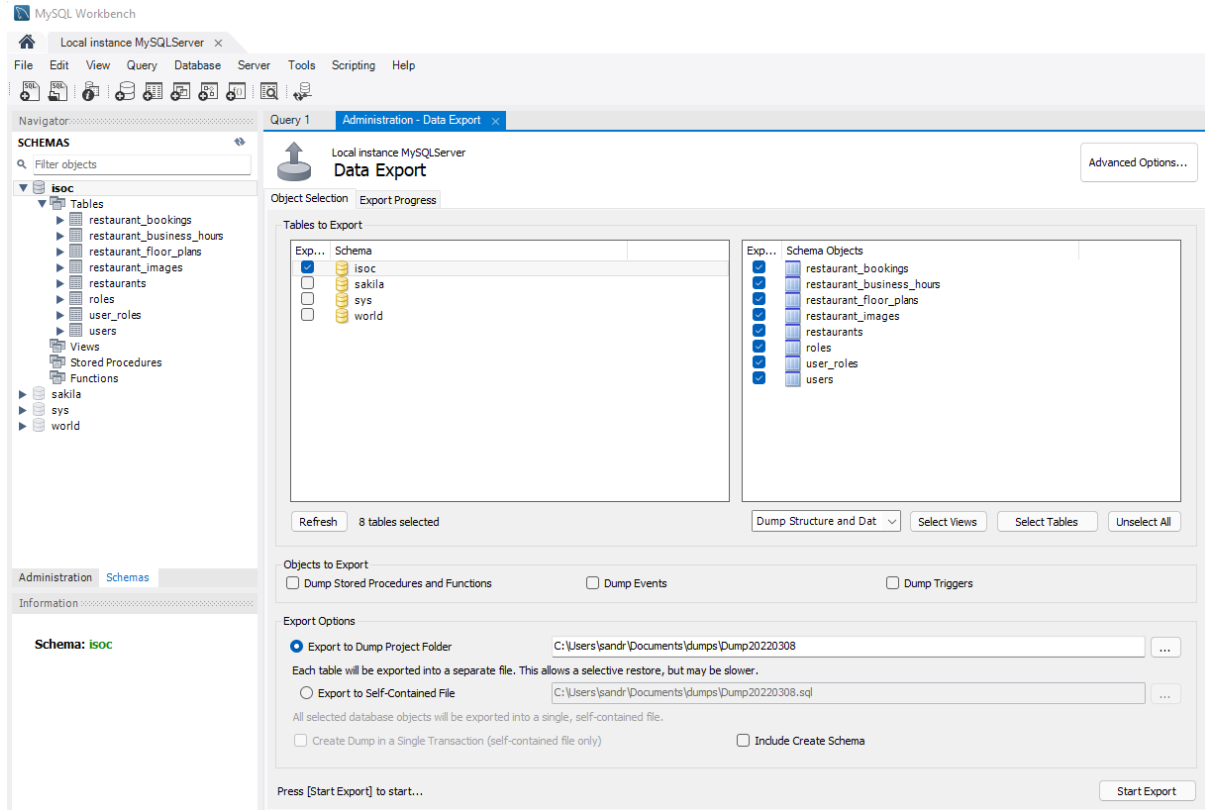


FIGURE 72: MYSQL WORKBENCH - EXPORT DATABASE

Generated scripts will be executed against the Azure Database for MySQL instance.

Chapter 6 - Testing

This chapter presents the manual testing of the ISOC Reservations application. The testing plan is organized into “testing areas” defined based on the functionalities associated to each role of the applications and additionally security area.

6.1. ISOCADMIN functionalities

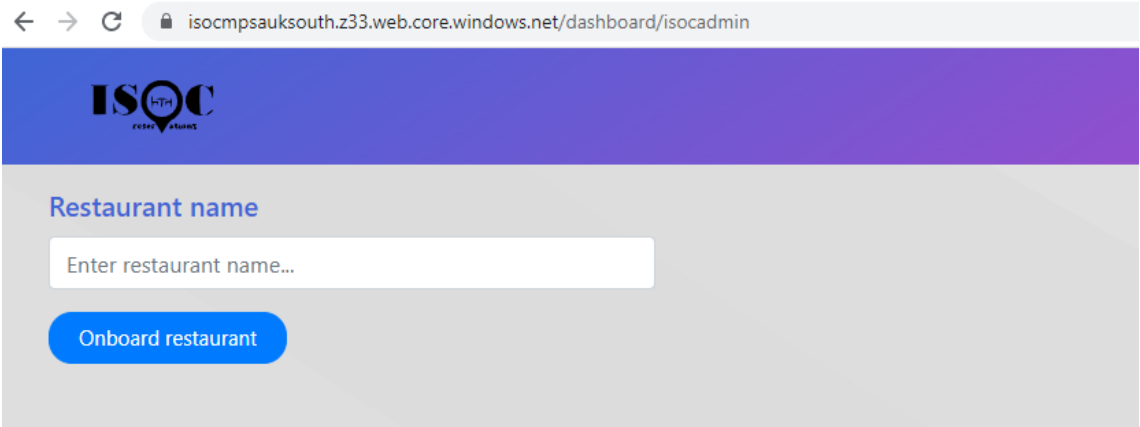
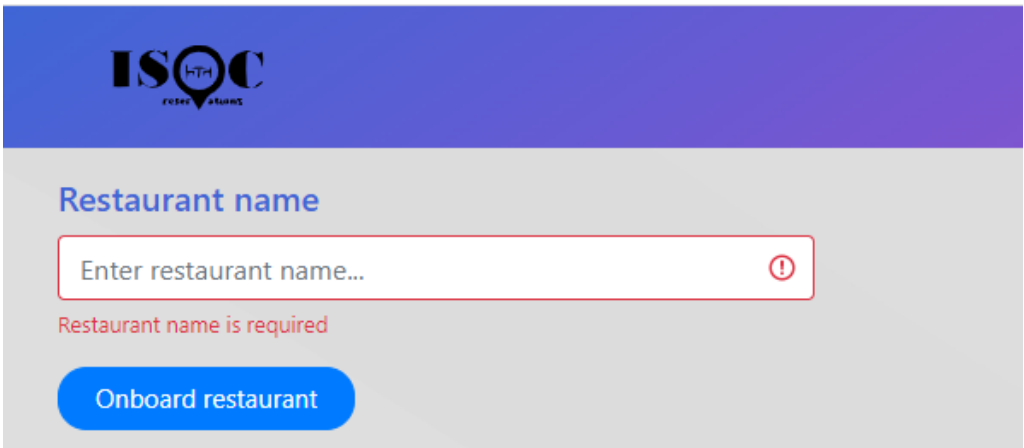
Test case	Onboard new restaurant – invalid data
Purpose of test case	Onboard new restaurant without specifying a restaurant name
Input	
	
FIGURE 73:INPUT ONBOARD NEW RESTAURANT TEST CASE – INVALID DATA	
Output	
	
FIGURE 74: OUTPUT ONBOARD NEW RESTAURANT TEST CASE – INVALID DATA	
Result	PASS

TABLE 8: ONBOARD NEW RESTAURANT TEST CASE – INVALID DATA

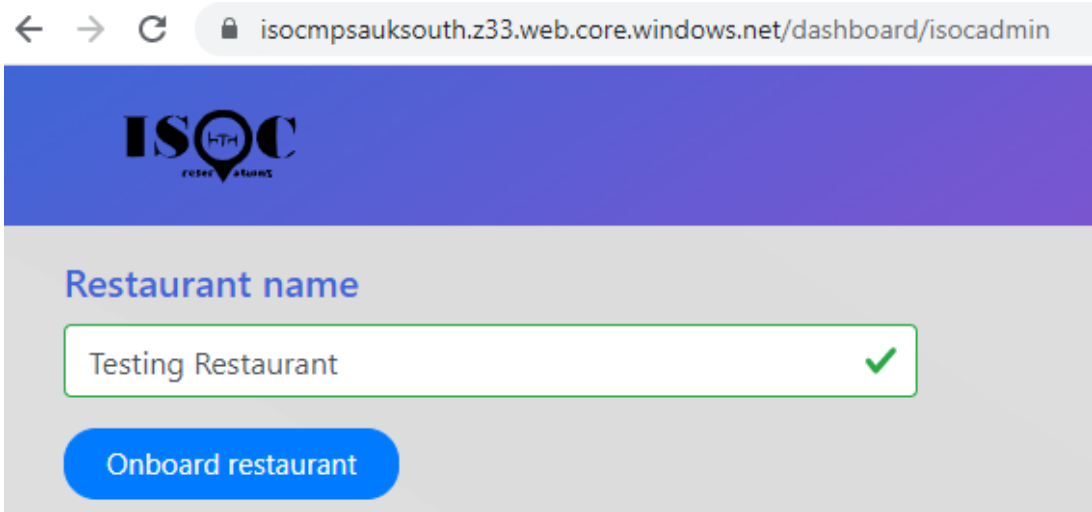
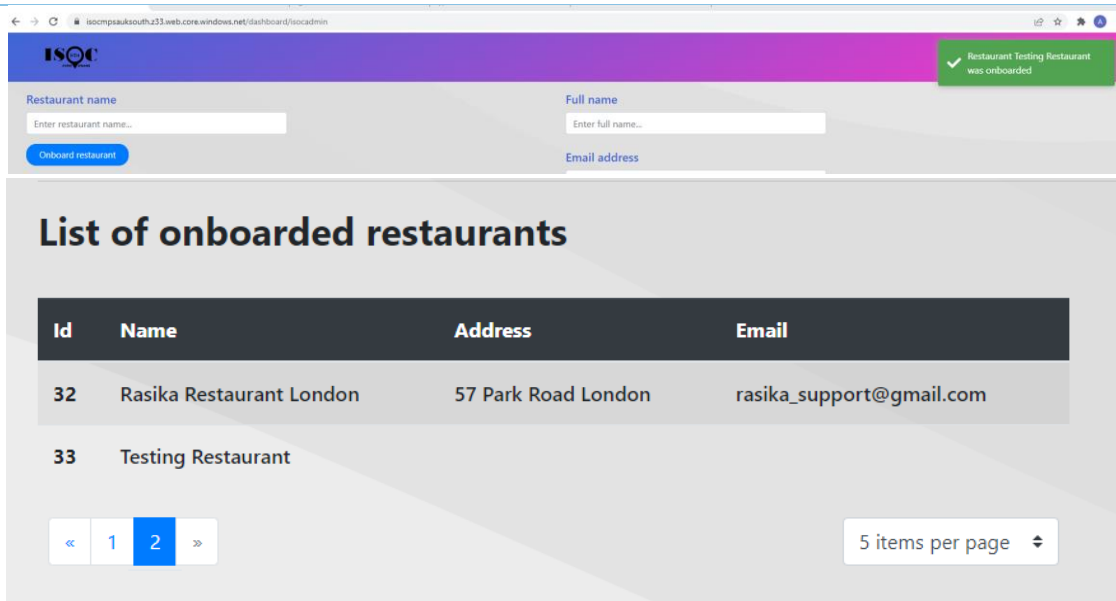
Test case	Onboard new restaurant – valid data
Purpose of test case	Onboard new restaurant with a valid restaurant name
Input	
	
FIGURE 75: INPUT - ONBOARD NEW RESTAURANT TEST CASE - VALID DATA	
Output	
	
FIGURE 76: OUTPUT - ONBOARD NEW RESTAURANT TEST CASE - VALID DATA	
Result	PASS

TABLE 9: ONBOARD NEW RESTAURANT TEST CASE – VALID DATA

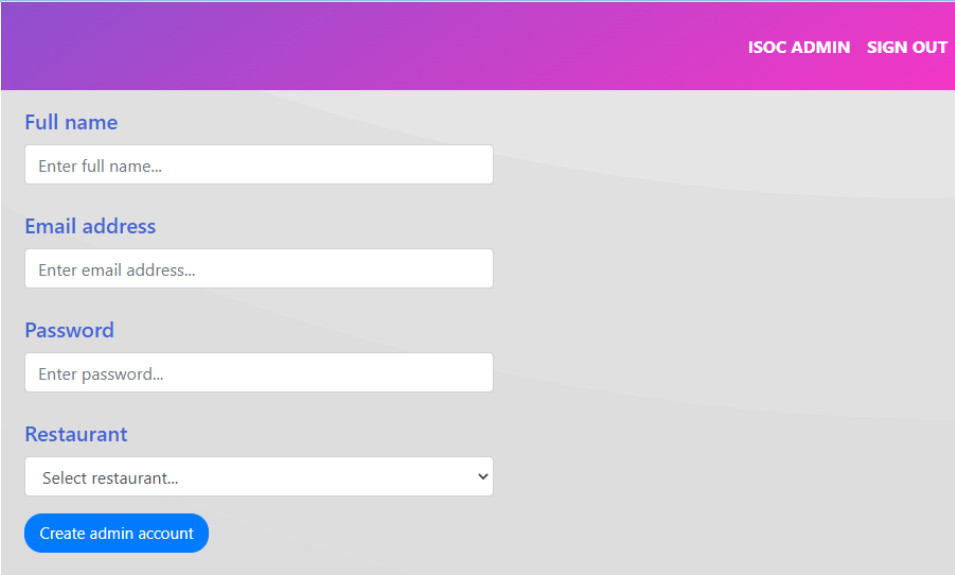
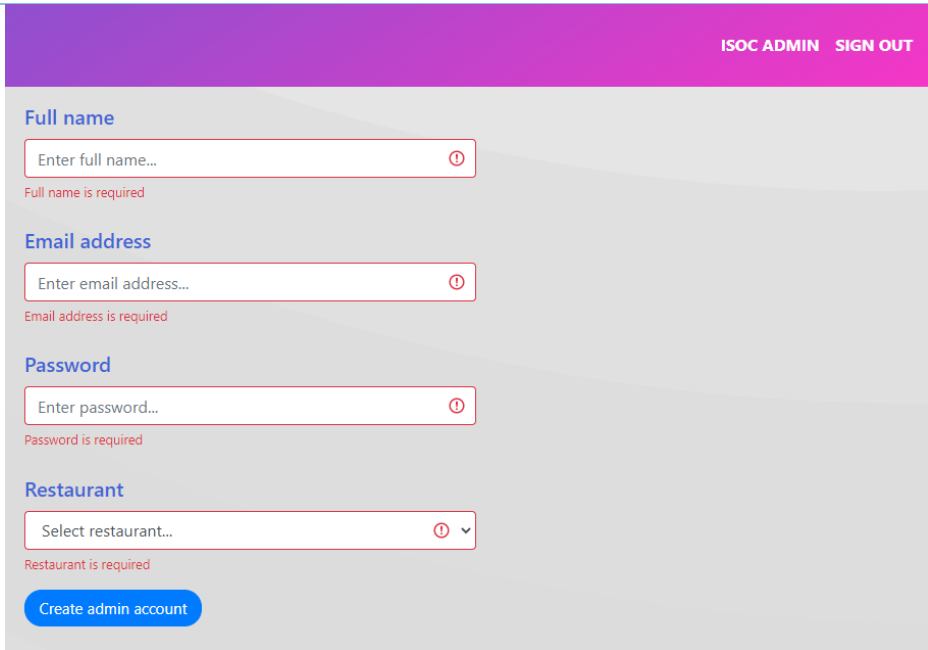
Test case	Create admin account – invalid data
Purpose of test case	Create admin account for a specific restaurant with invalid data
Input	
 FIGURE 77: INPUT - CREATE ADMIN ACCOUNT TEST CASE - INVALID DATA	
Output	
 FIGURE 78: OUTPUT - CREATE ADMIN ACCOUNT TEST CASE - INVALID DATA	
Result	PASS

TABLE 10: CREATE ADMIN ACCOUNT TEST CASE - INVALID DATA

Test case	Create admin account – invalid email or password
Purpose of test case	Create admin account for a specific restaurant with invalid email or password
Input	

The screenshot shows a web interface for creating an admin account. At the top right, there are links for 'ISOC ADMIN' and 'SIGN OUT'. The form contains four fields: 'Full name' with the value 'Admin Testing Restaurant' and a green checkmark; 'Email address' with the value 'admin_testing_restaurant' and a red error icon; 'Password' with masked characters '.....' and a red error icon; and 'Restaurant' with the value 'Testing Restaurant' and a green checkmark. Below the 'Email address' field, the text 'Invalid email address' is displayed in red. Below the 'Password' field, the text 'Password must be at least 6 characters' is displayed in red. At the bottom of the form is a blue button labeled 'Create admin account'.

FIGURE 79: INPUT - CREATE ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD

Output

This screenshot is identical to Figure 79, showing the 'Create admin account' form with validation errors for the email and password fields. The 'Full name' and 'Restaurant' fields are correctly filled and validated. The 'Email address' field shows an 'Invalid email address' error, and the 'Password' field shows a 'Password must be at least 6 characters' error. The 'Create admin account' button is visible at the bottom.

FIGURE 80: OUTPUT - CREATE ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD

Result	PASS
---------------	-------------

TABLE 11: CREATE ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD

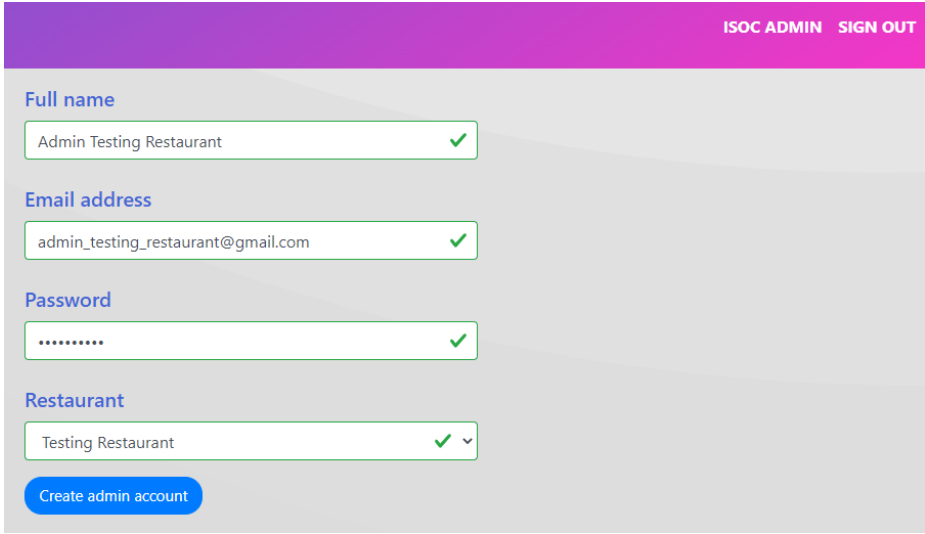
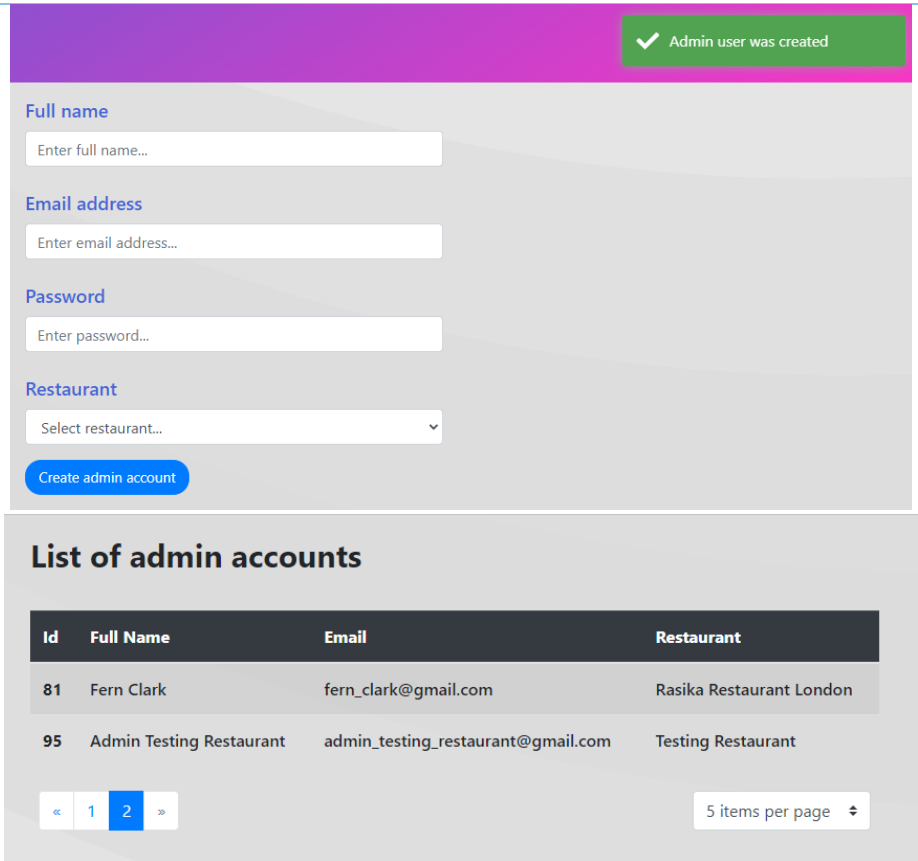
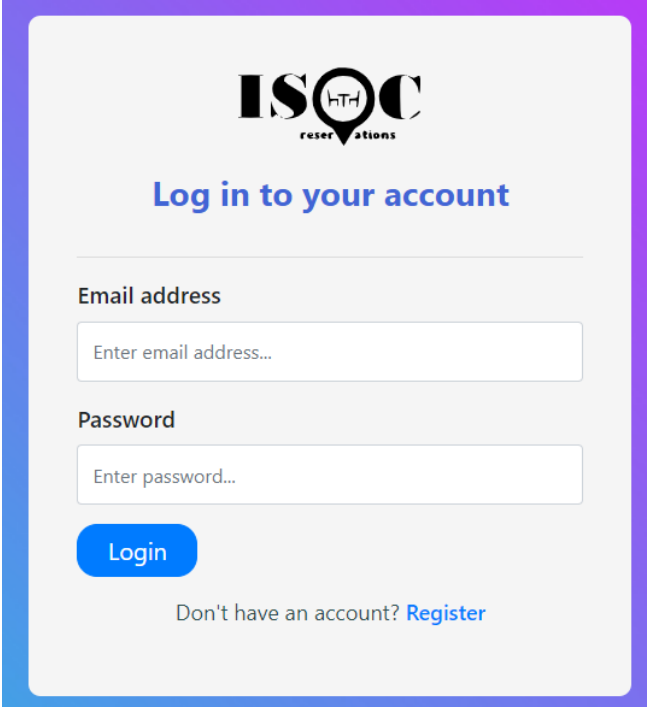
Test case	Create admin account - valid data
Purpose of test case	Create admin account for a specific restaurant with valid data
Input	 FIGURE 81: INPUT - CREATE ADMIN ACCOUNT TEST CASE - VALID DATA
Output	 FIGURE 82: OUTPUT - CREATE ADMIN ACCOUNT TEST CASE - VALID DATA
Result	PASS

TABLE 12: CREATE ADMIN ACCOUNT TEST CASE - VALID DATA

6.2. ADMIN functionalities

Test case	Admin login – missing credentials
Purpose of test case	Login with missing credentials
Input	 FIGURE 83: INPUT - ADMIN LOGIN TEST CASE - MISSING CREDENTIALS
Output	

ISOC
reservations

Log in to your account

Email address

Email address is required

Password

Password is required

Login

Don't have an account? [Register](#)

FIGURE 84: OUTPUT - ADMIN LOGIN TEST CASE - MISSING CREDENTIALS

Result	PASS
---------------	-------------

TABLE 13: ADMIN LOGIN TEST CASE - MISSING CREDENTIALS

Test case	Admin login – invalid credentials
Purpose of test case	Login with invalid credentials
Input	

ISOC
reservations

Log in to your account

Email address

Password

Login

Don't have an account? [Register](#)

FIGURE 85: INPUT – ADMIN LOGIN TEST CASE – INVALID CREDENTIALS

Output

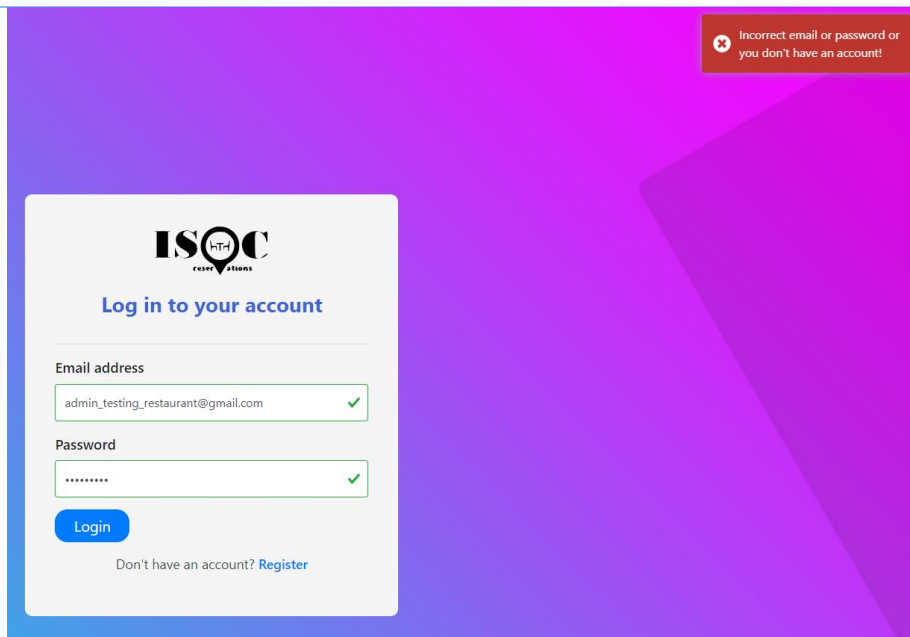


FIGURE 86: OUTPUT - ADMIN LOGIN TEST CASE - INVALID CREDENTIALS

Result

PASS

TABLE 14: ADMIN LOGIN TEST CASE - INVALID CREDENTIALS

Test case	Admin login – valid credentials
-----------	---------------------------------

Purpose of test case	Login with valid credentials
----------------------	------------------------------

Input

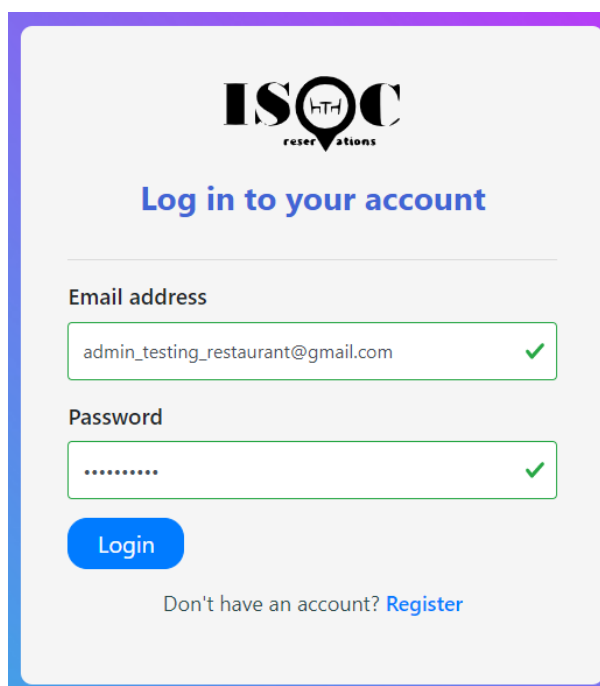


FIGURE 87: INPUT - ADMIN LOGIN TEST CASE - VALID CREDENTIALS

Output

ISOC

ADMIN TESTING RESTAURANT SIGN OUT

Accounts

Restaurant Profile

Floor Plan

Full name: Admin Testing Restaurant

Email: admin_testing_restaurant@gmail.com

New Password

Enter new password...

Confirm Password

Confirm password...

Change Password

Full name

Enter full name...

Email address

Enter email address...

Password

Enter password...

Role

Administrator

Create account

List of accounts

Id	Full Name	Email	Role
95	Admin Testing Restaurant	admin_testing_restaurant@gmail.com	Administrator

1

5 items per page

© ISOC Reservations 2021. All Rights Reserved.

FIGURE 88: OUTPUT - ADMIN LOGIN TEST CASE - VALID CREDENTIALS

Result	PASS
--------	------

TABLE 15: ADMIN LOGIN TEST CASE - VALID CREDENTIALS

Test case	Change password - invalid data
Purpose of test case	Change password with invalid data
Input	

Full name: Admin Testing Restaurant

Email: admin_testing_restaurant@gmail.com

New Password

....

!

New Password must be at least 6 characters

Confirm Password

Confirm password...

Change Password

FIGURE 89: INPUT - CHANGE PASSWORD TEST CASE - INVALID DATA

78

Output

Full name: Admin Testing Restaurant
Email: admin_testing_restaurant@gmail.com

New Password

New Password must be at least 6 characters

Confirm Password

Confirm Password is required

Change Password

FIGURE 90: OUTPUT - CHANGE PASSWORD TEST CASE - INVALID DATA

Result

PASS

TABLE 16: CHANGE PASSWORD TEST CASE - INVALID DATA

Test case

Change password - not match

Purpose of test case

Change password with passwords not matching

Input

Full name: Admin Testing Restaurant
Email: admin_testing_restaurant@gmail.com

New Password

Confirm Password

Change Password

FIGURE 91: INPUT - CHANGE PASSWORD TEST CASE - NOT MATCH

Output

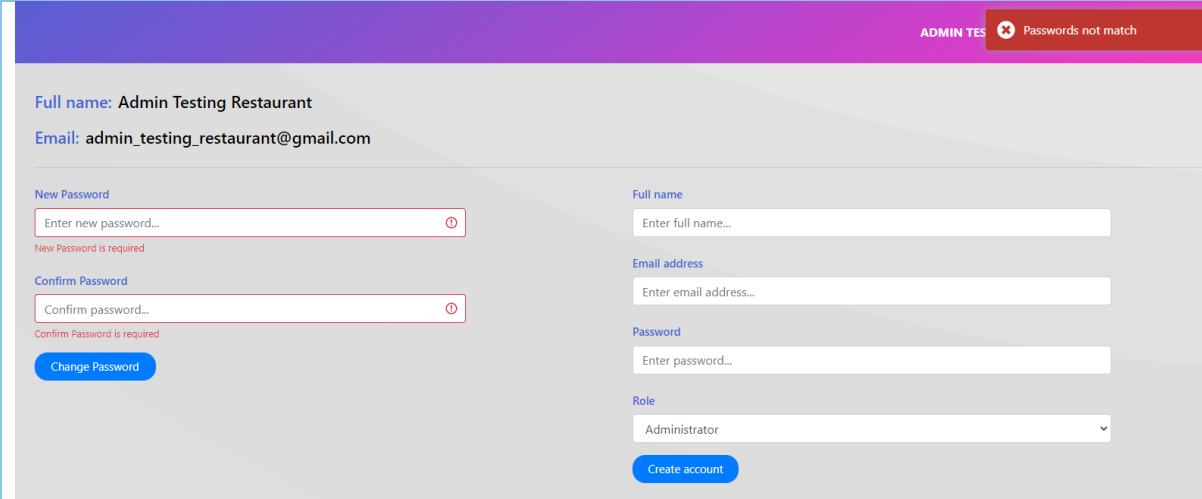
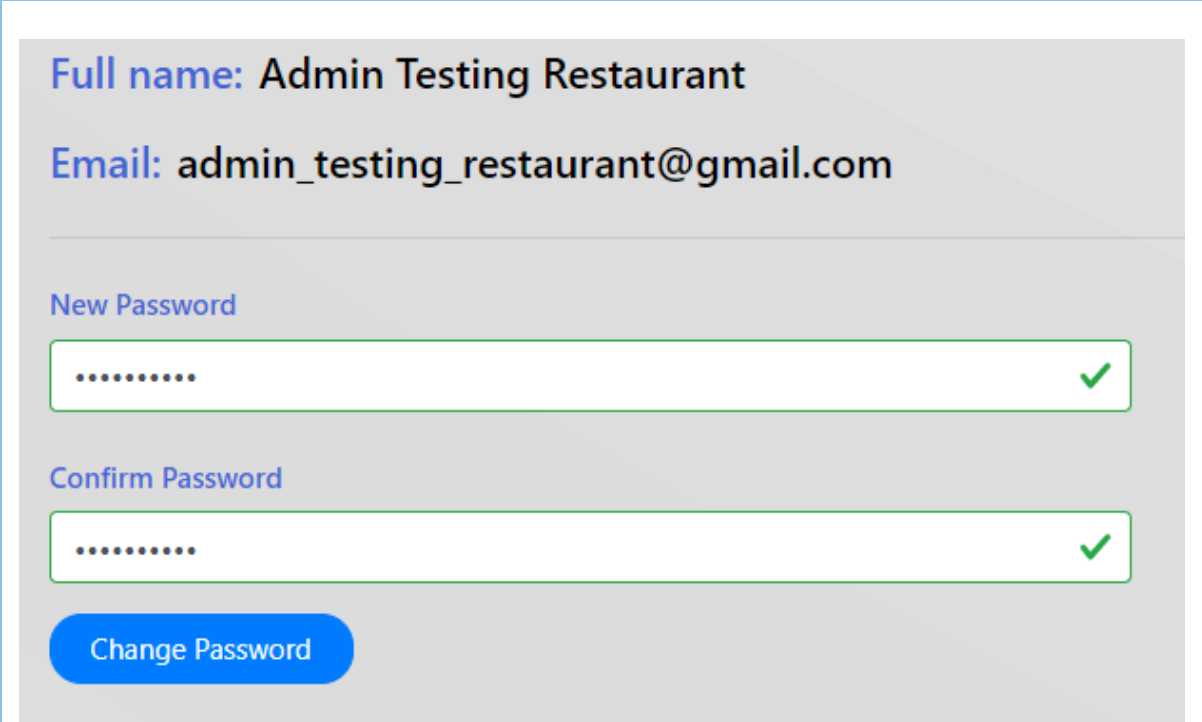
	
FIGURE 92: OUTPUT - CHANGE PASSWORD TEST CASE - NOT MATCH	
Result	PASS

TABLE 17: CHANGE PASSWORD TEST CASE - NOT MATCH

Test case	Change password – valid data
Purpose of test case	Change password with valid data
Input	
	
FIGURE 93: INPUT - CHANGE PASSWORD TEST CASE - VALID DATA	

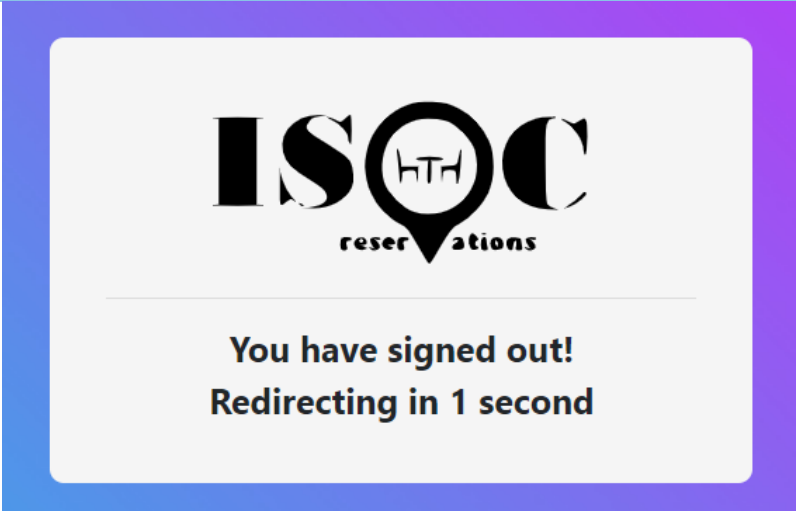
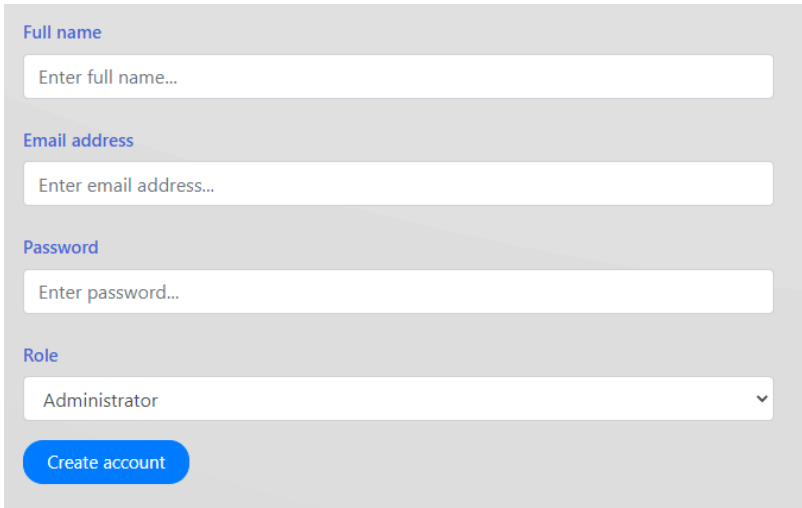
Output	
	
FIGURE 94: OUTPUT - CHANGE PASSWORD TEST CASE - VALID DATA	
Result	PASS

TABLE 18: CHANGE PASSWORD TEST CASE - VALID DATA

Test case	Create restaurant admin account – invalid data
Purpose of test case	Create restaurant admin account with invalid data
Input	
	
FIGURE 95: INPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID DATA	
Output	

Full name

Enter full name...

Full name is required

Email address

Enter email address...

Email address is required

Password

Enter password...

Password is required

Role

Administrator

Create account

FIGURE 96: OUTPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID DATA

Result	PASS
--------	------

TABLE 19: CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID DATA

Test case	Create restaurant admin account – invalid email or password
Purpose of test case	Create restaurant admin account with invalid email or password
Input	

The screenshot shows a web form for creating a restaurant admin account. It contains four input fields: 'Full name' (with value 'admin2 testing restaurant' and a green checkmark), 'Email address' (with value 'admin2_testing_restaurant' and a red error message 'Invalid email address'), 'Password' (with masked characters '.....' and a red error message 'Password must be at least 6 characters'), and 'Role' (a dropdown menu with 'Administrator' selected and a green checkmark). A blue 'Create account' button is at the bottom.

FIGURE 97: INPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD

Output

This screenshot is identical to Figure 97, showing the 'Create account' form with validation errors for the email and password fields. The 'Full name' and 'Role' fields are correctly filled, and the 'Create account' button is visible at the bottom.

FIGURE 98: OUTPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD

Result	PASS
---------------	-------------

TABLE 20: CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD

Test case	Create restaurant admin account – valid data
Purpose of test case	Create restaurant admin account with valid data
Input	

Full name
admin2 testing restaurant ✓

Email address
admin2_testing_restaurant@gmail.com ✓

Password
..... ✓

Role
Administrator ✓

Create account

FIGURE 99: INPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - VALID DATA

Output

ADMIN TEST ✓ Admin user was created

Full name
Enter full name...

Email address
Enter email address...

Password
Enter password...

Role
▼

Create account

List of accounts

Id	Full Name	Email	Role
95	Admin Testing Restaurant	admin_testing_restaurant@gmail.com	Administrator
96	admin2 testing restaurant	admin2_testing_restaurant@gmail.com	Administrator

< 1 >

5 items per page

FIGURE 100: OUTPUT - CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - VALID DATA

Result	PASS
---------------	-------------

TABLE 21: CREATE RESTAURANT ADMIN ACCOUNT TEST CASE - VALID DATA

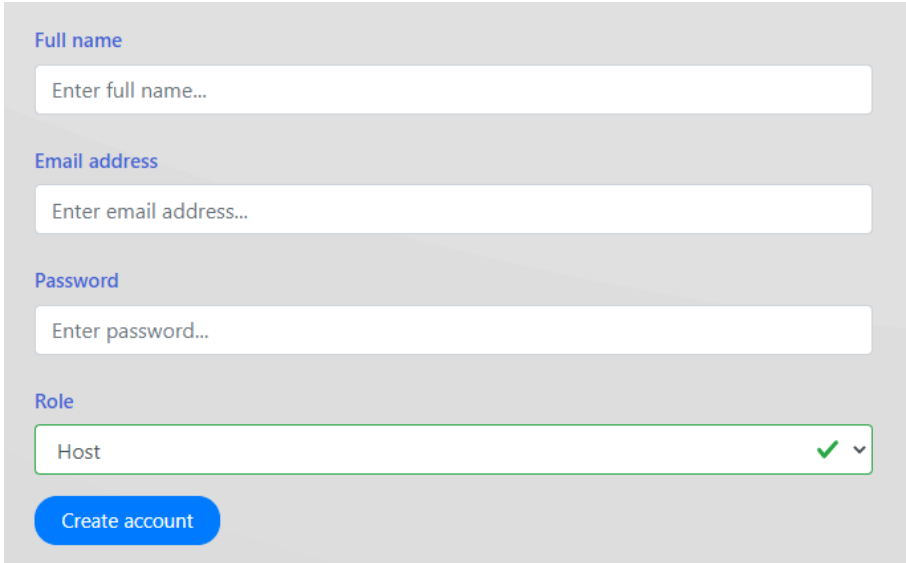
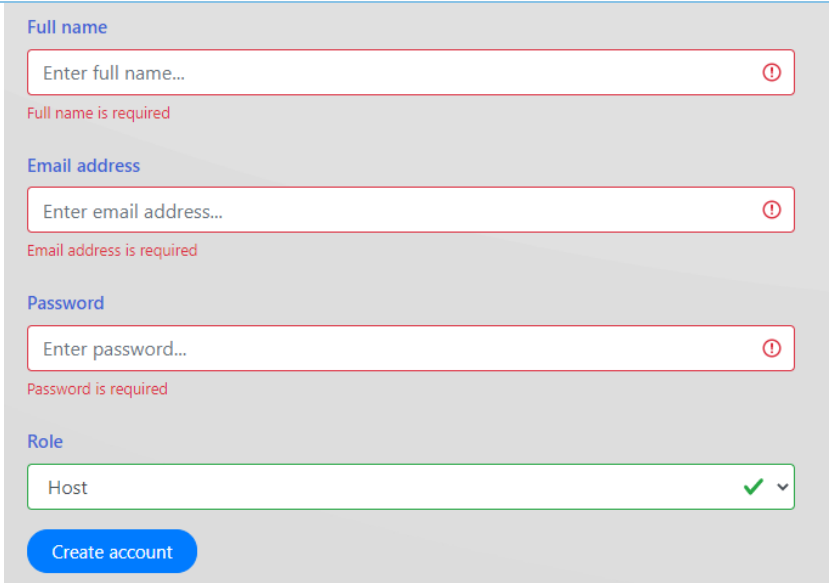
Test case	Create restaurant host account – invalid data
Purpose of test case	Create restaurant admin account with invalid data
Input	
	
FIGURE 101: INPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE - INVALID DATA	
Output	
	
FIGURE 102: OUTPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE - INVALID DATA	
Result	PASS

TABLE 22: CREATE RESTAURANT HOST ACCOUNT TEST CASE - INVALID DATA

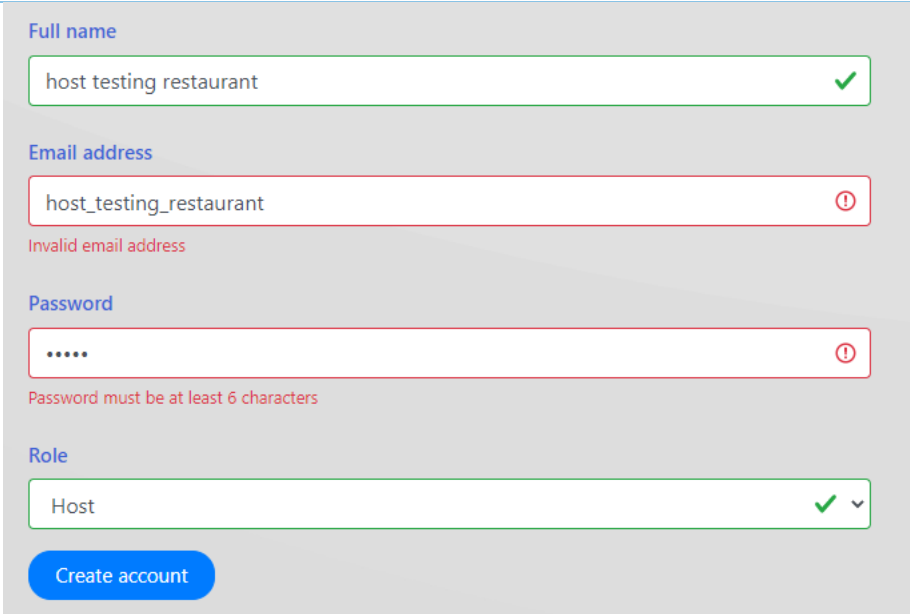
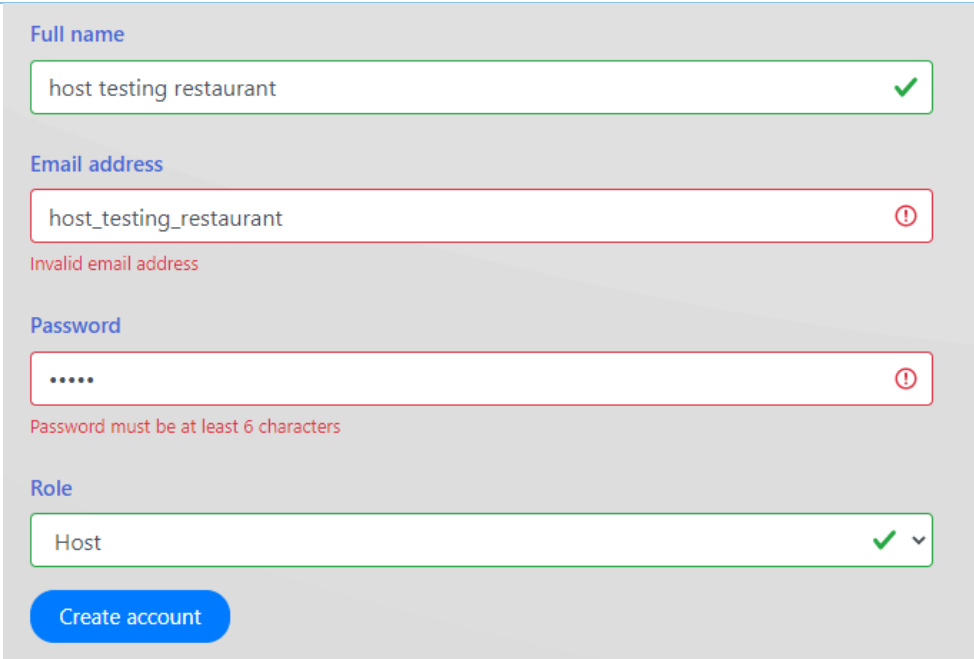
Test case	Create restaurant host account – invalid email or password
Purpose of test case	Create restaurant host account with invalid email or password
Input	
 The screenshot shows a 'Create account' form with the following fields and validation status: Full name: 'host testing restaurant' (Valid, green checkmark) Email address: 'host_testing_restaurant' (Invalid, red border and exclamation mark icon. Error message: 'Invalid email address') Password: '.....' (Invalid, red border and exclamation mark icon. Error message: 'Password must be at least 6 characters') Role: 'Host' (Valid, green checkmark and dropdown arrow) A blue 'Create account' button is at the bottom.	
FIGURE 103: INPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE – INVALID EMAIL OR PASSWORD	
Output	
 This screenshot is identical to Figure 103, showing the 'Create account' form with validation errors for the email and password fields.	
FIGURE 104: OUTPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD	
Result	PASS

TABLE 23: CREATE RESTAURANT HOST ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD

Test case	Create restaurant host account – valid data																				
Purpose of test case	Create restaurant admin account with valid data																				
Input																					
Full name host testing restaurant Email address host_testing_restaurant@gmail.com Password Role Host Create account																					
FIGURE 105: INPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE – VALID DATA																					
Output																					
ADMIN TEST Host user was created Full name Enter full name... Email address Enter email address... Password Enter password... Role Create account List of accounts <table><thead><tr><th>Id</th><th>Full Name</th><th>Email</th><th>Role</th><th></th></tr></thead><tbody><tr><td>95</td><td>Admin Testing Restaurant</td><td>admin_testing_restaurant@gmail.com</td><td>Administrator</td><td></td></tr><tr><td>96</td><td>admin2 testing restaurant</td><td>admin2_testing_restaurant@gmail.com</td><td>Administrator</td><td></td></tr><tr><td>97</td><td>host testing restaurant</td><td>host_testing_restaurant@gmail.com</td><td>Host</td><td></td></tr></tbody></table> <1> 5 items per page		Id	Full Name	Email	Role		95	Admin Testing Restaurant	admin_testing_restaurant@gmail.com	Administrator		96	admin2 testing restaurant	admin2_testing_restaurant@gmail.com	Administrator		97	host testing restaurant	host_testing_restaurant@gmail.com	Host	
Id	Full Name	Email	Role																		
95	Admin Testing Restaurant	admin_testing_restaurant@gmail.com	Administrator																		
96	admin2 testing restaurant	admin2_testing_restaurant@gmail.com	Administrator																		
97	host testing restaurant	host_testing_restaurant@gmail.com	Host																		
FIGURE 106: OUTPUT - CREATE RESTAURANT HOST ACCOUNT TEST CASE – VALID DATA																					
Result	PASS																				

TABLE 24: CREATE RESTAURANT HOST ACCOUNT TEST CASE – VALID DATA

Test case	Modify account – invalid data
Purpose of test case	modify account from “list of accounts” with invalid data
Input	

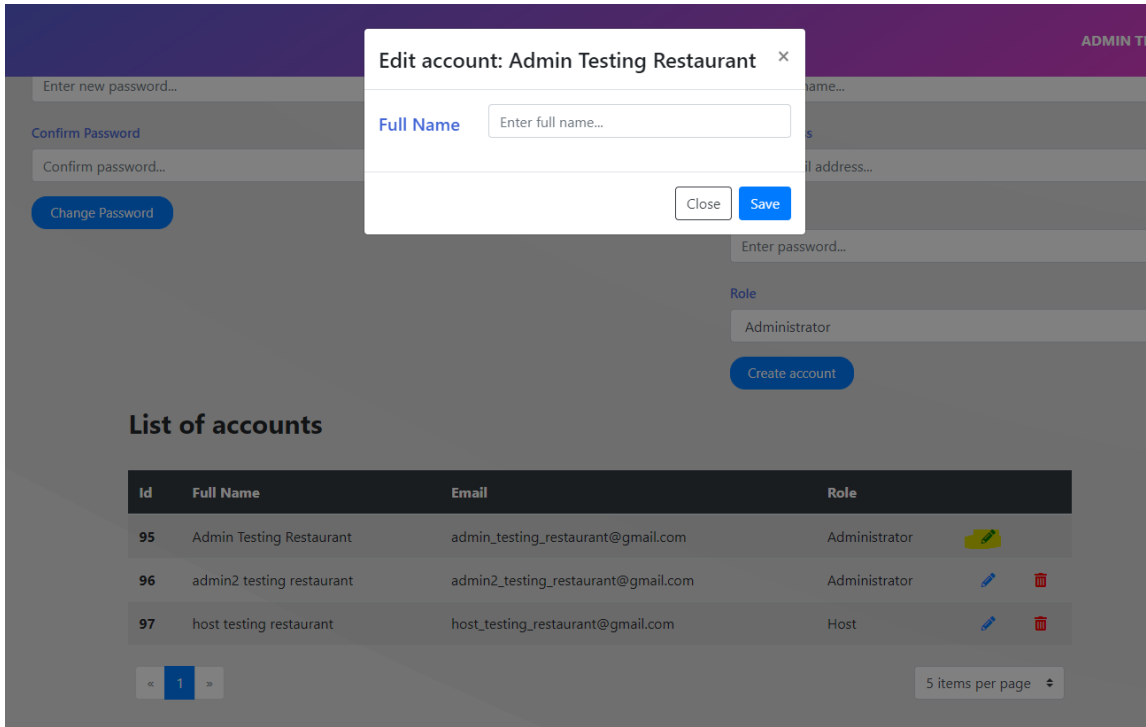


FIGURE 107: INPUT - MODIFY ACCOUNT TEST CASE - INVALID DATA

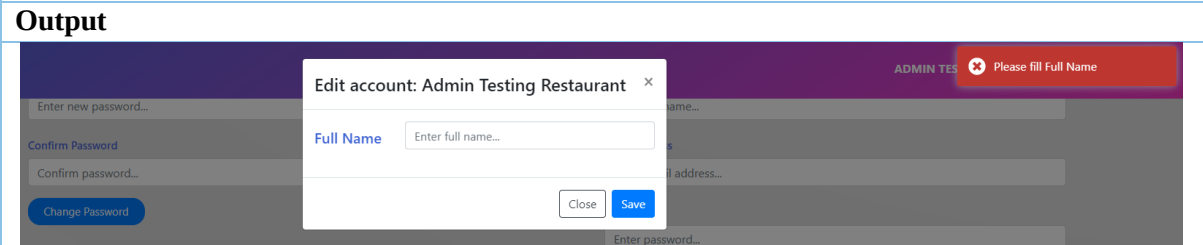


FIGURE 108: OUTPUT - MODIFY ACCOUNT TEST CASE - INVALID DATA

Result	PASS
---------------	-------------

TABLE 25: MODIFY ACCOUNT TEST CASE - INVALID DATA

Test case	Modify account – valid data
Purpose of test case	modify account from “list of accounts” with valid data

Input

FIGURE 109: INPUT - MODIFY ACCOUNT TEST CASE - VALID DATA

Output

FIGURE 110: OUTPUT - MODIFY ACCOUNT TEST CASE - VALID DATA

Result	PASS
---------------	-------------

TABLE 26: MODIFY ACCOUNT TEST CASE - VALID DATA

Test case	Delete account
Purpose of test case	Delete an account from “list of accounts” except current logged account

Input

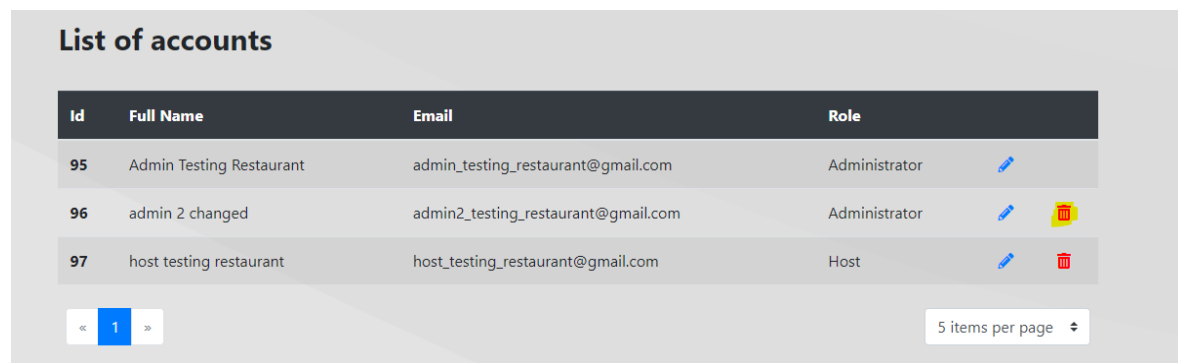


FIGURE 111: INPUT - DELETE ACCOUNT TEST CASE

Output

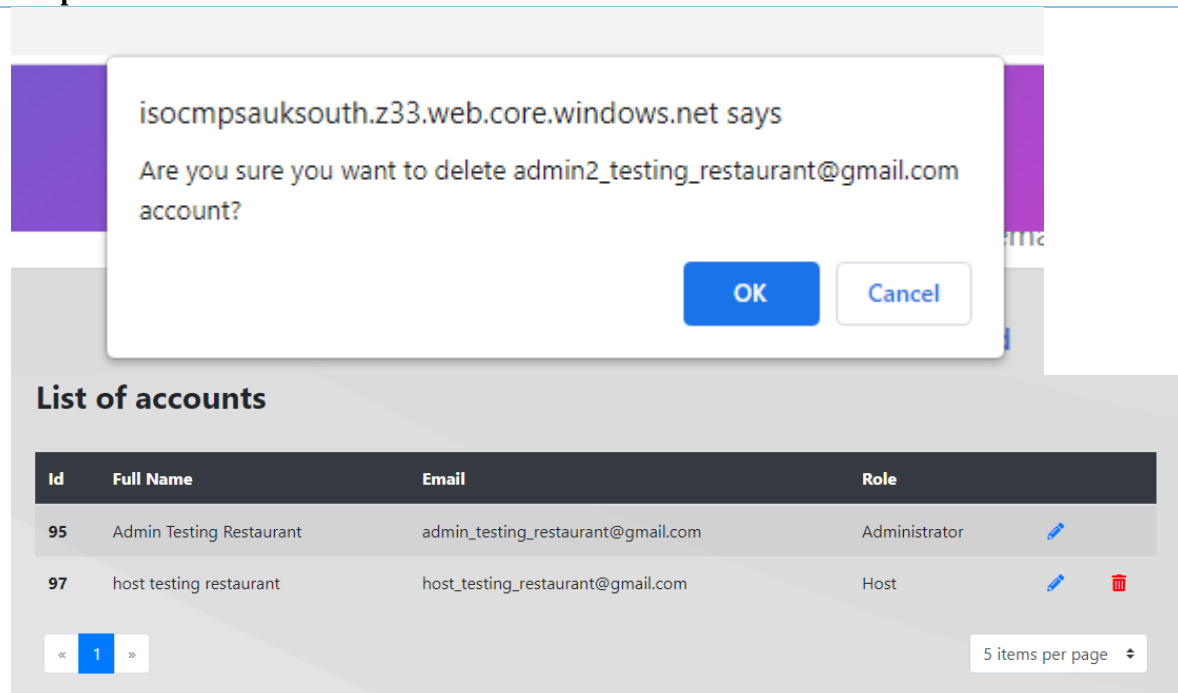


FIGURE 112: OUTPUT - DELETE ACCOUNT TEST CASE

Result	PASS
---------------	-------------

TABLE 27: DELETE ACCOUNT TEST CASE

Test case	Modify restaurant general information – invalid data
Purpose of test case	modify restaurant general information with invalid data
Input	

General Information

Address

testing restaurant address, London

Phone Number

+447777

Email

testing_restaurant

Post Code

E153JN

Cuisines

Mediterranean

Description

This is a testing restaurant description

FIGURE 113: INPUT - MODIFY RESTAURANT GENERAL INFORMATION TEST CASE - INVALID DATA

ADMIN TEST

Invalid phone number! Format expected: +441234567890

General Information

Address

testing restaurant address, London

Phone Number

+447777

Email

testing_restaurant

Post Code

E153JN

Cuisines

Mediterranean

Description

This is a testing restaurant description

Business hours

Monday

00 : 00

00 : 00

00 : 00

00 : 00

Tuesday

00 : 00

00 : 00

00 : 00

00 : 00

Wednesday

00 : 00

00 : 00

00 : 00

00 : 00

Thursday

00 : 00

00 : 00

00 : 00

00 : 00

Friday

00 : 00

00 : 00

00 : 00

00 : 00

ADMIN TEST		Invalid email!
General Information Address testing restaurant address, London Phone Number +441234567890 Email testing_restaurant Post Code E153JN Cuisines Mediterranean Description This is a testing restaurant description		Business hours Monday 00 : 00 00 : 00 Tuesday 00 : 00 00 : 00 Wednesday 00 : 00 00 : 00 Thursday 00 : 00 00 : 00 Friday 00 : 00 00 : 00

TABLE 28: MODIFY RESTAURANT GENERAL INFORMATION TEST CASE - INVALID DATA

Test case	Modify restaurant profile – valid data
Purpose of test case	Modify restaurant information and business hours with valid data
Input	
General Information Address testing restaurant address, London Phone Number +441234567890 Email testing_restaurant@gmail.com Post Code E153JN Cuisines Mediterranean Description This is a testing restaurant description	Business hours Monday 08 : 00 23 : 00 Tuesday 07 : 00 23 : 00 Wednesday 08 : 00 23 : 00 Thursday 08 : 00 22 : 00 Friday 08 : 00 23 : 00 Saturday Closed Sunday Closed

FIGURE 115: INPUT - MODIFY RESTAURANT PROFILE TEST CASE - VALID DATA

Output

FIGURE 115: INPUT - MODIFY RESTAURANT PROFILE TEST CASE - VALID DATA

ADMIN TEST

Restaurant profile updated

General Information

Address

testing_restaurant.address.London

Phone Number

+441234567890

Email

testing_restaurant@gmail.com

Post Code

E153JN

Cuisines

Mediterranean

Description

This is a testing restaurant description

Save

Business hours

Monday	08 : 00	23 : 00	☐
Tuesday	07 : 00	23 : 00	☐
Wednesday	08 : 00	23 : 00	☐
Thursday	08 : 00	22 : 00	☐
Friday	08 : 00	23 : 00	☐
Saturday	Closed	Closed	☒
Sunday	Closed	Closed	☒

FIGURE 116: OUTPUT - MODIFY RESTAURANT PROFILE TEST CASE - VALID DATA

Result **PASS**

TABLE 29: MODIFY RESTAURANT PROFILE TEST CASE - VALID DATA

Test case	Restaurant image upload – invalid size
Purpose of test case	Upload a restaurant image with a size greater than 1MB
Input	The screenshot shows the 'Properties' dialog box for a file named 'testing_invalid_image'. The 'General' tab is selected, showing the file is a 'JPG File (.jpg)' located at 'D:\University\Year 3\Major Project\Final Report\der'. The file size is 3.88 MB (4,077,975 bytes) and the size on disk is 3.89 MB (4,079,616 bytes). The file was created on 24 February 2022 at 11:09:28 and modified on the same day at 11:09:29. It was accessed on 06 March 2022 at 14:16:45. The 'Attributes' section shows 'Read-only' and 'Hidden' are unchecked. The 'Security' section shows a message: 'This file came from another computer and might be blocked to help protect this computer.' with an 'Unblock' checkbox.

FIGURE 117: INPUT - RESTAURANT IMAGE UPLOAD TEST CASE - INVALID SIZE – IMAGE PROPERTIES

Upload images

Image size must be less than 1MB!

Upload

testing_invalid_image.jpg

Browse

FIGURE 118: INPUT - RESTAURANT IMAGE UPLOAD TEST CASE - INVALID SIZE

Output

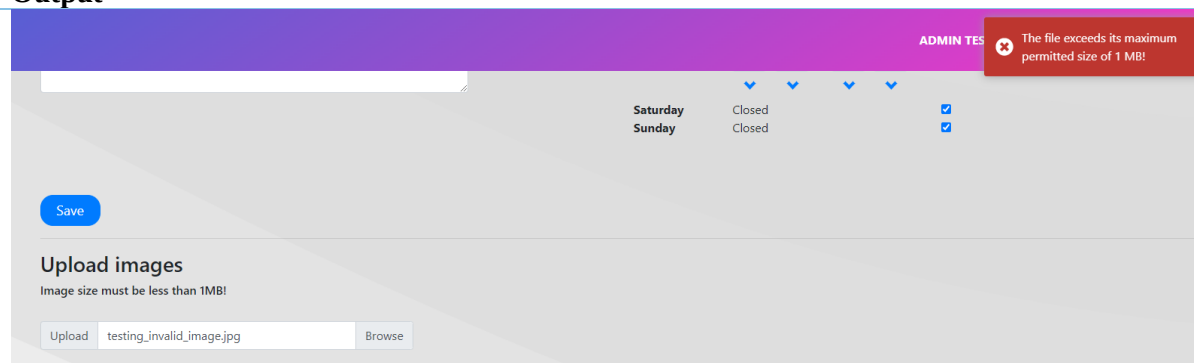


FIGURE 119: OUTPUT - RESTAURANT IMAGE UPLOAD TEST CASE - INVALID SIZE

Result

PASS

TABLE 30: RESTAURANT IMAGE UPLOAD TEST CASE - INVALID SIZE

Test case	Restaurant image upload – valid size
Purpose of test case	Upload a restaurant image with a size less than 1MB
Input	

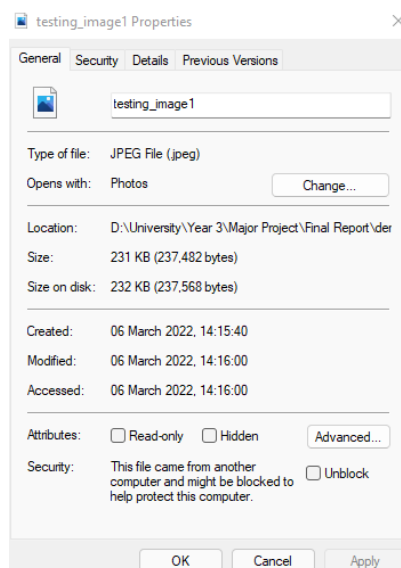


FIGURE 120: INPUT - RESTAURANT IMAGE UPLOAD TEST CASE - VALID SIZE – IMAGE PROPERTIES

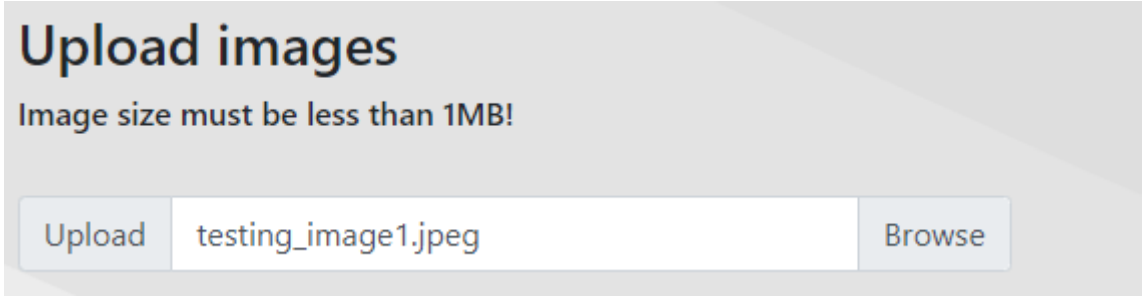


FIGURE 121: INPUT - RESTAURANT IMAGE UPLOAD TEST CASE - VALID SIZE

Output

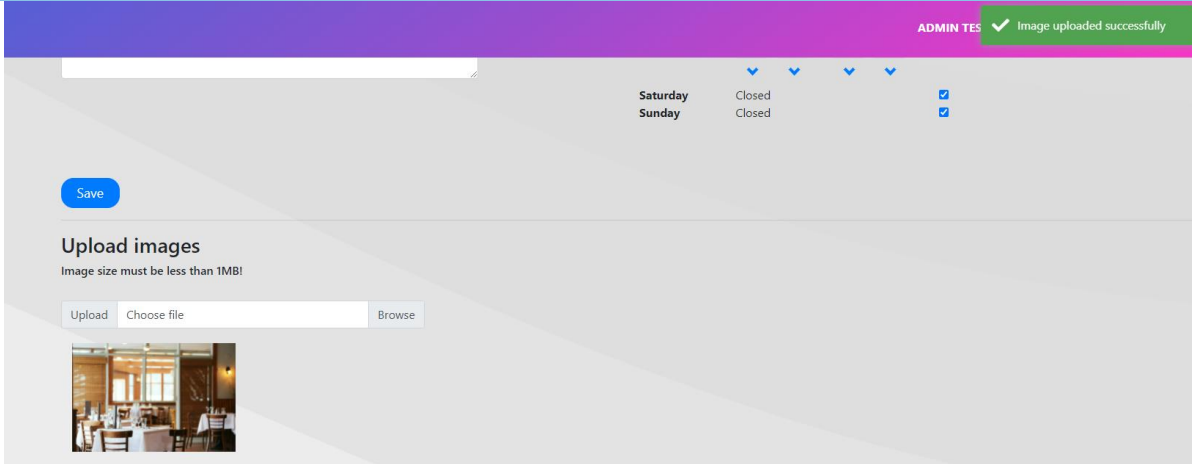


FIGURE 122: OUTPUT - RESTAURANT IMAGE UPLOAD TEST CASE - VALID SIZE

Result

PASS

TABLE 31: RESTAURANT IMAGE UPLOAD TEST CASE - VALID SIZE

Test case	Restaurant location on map
Purpose of test case	Display restaurant location on map
Input	

General Information

Address

Phone Number

Email

Post Code

Cuisines

Description

FIGURE 123: INPUT - RESTAURANT LOCATION ON MAP TEST CASE

Output

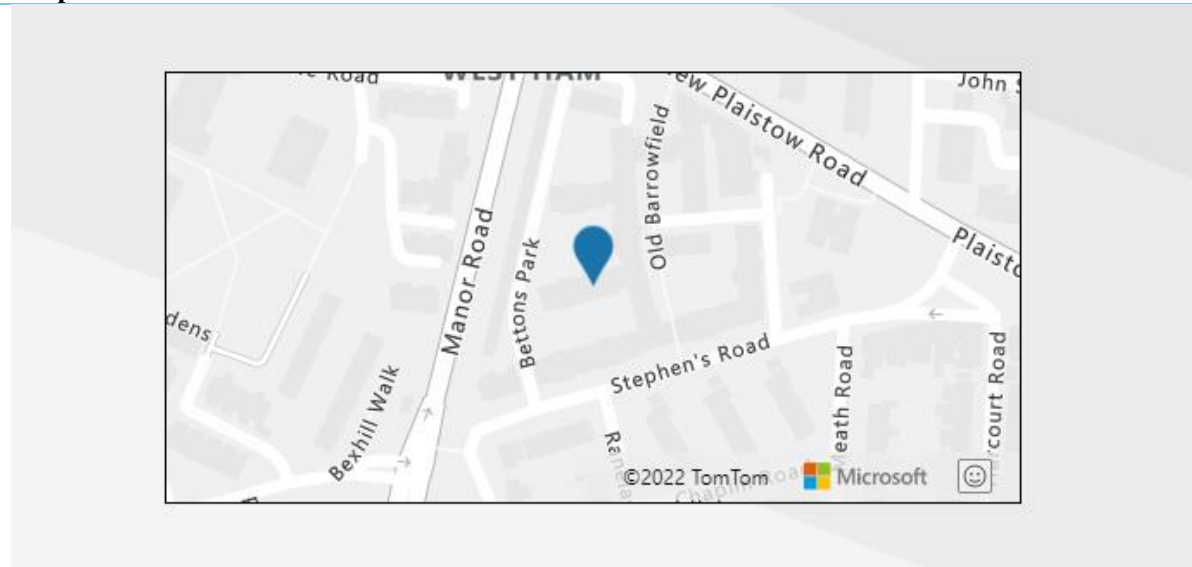


FIGURE 124: OUTPUT - RESTAURANT LOCATION ON MAP TEST CASE

Result	PASS
---------------	-------------

TABLE 32: RESTAURANT LOCATION ON MAP TEST CASE

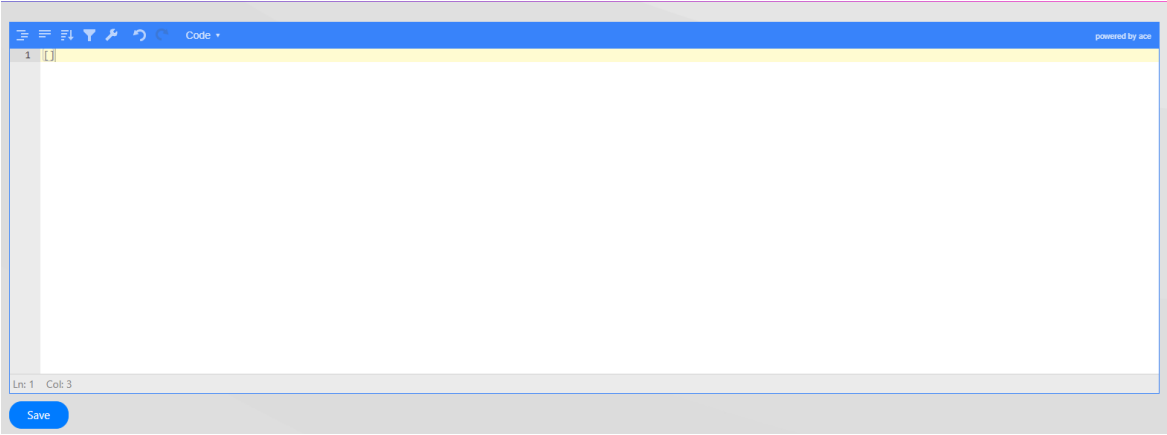
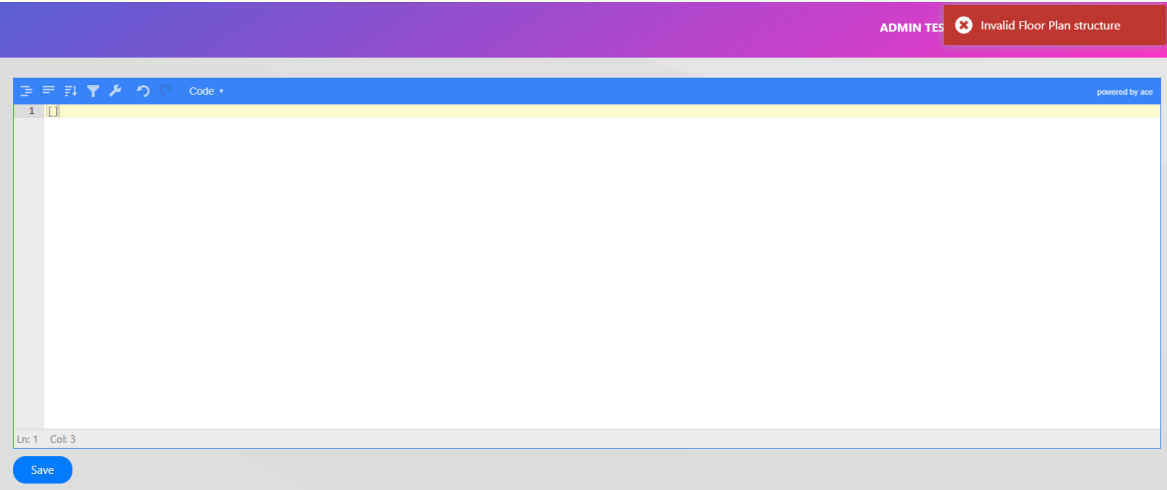
Test case	Create restaurant floor plan – invalid configuration
Purpose of test case	Create restaurant floor plan with invalid configuration
Input	
	
FIGURE 125: INPUT - CREATE RESTAURANT FLOOR PLAN TEST CASE - INVALID CONFIGURATION	
Output	
	
FIGURE 126: OUTPUT - CREATE RESTAURANT FLOOR PLAN TEST CASE - INVALID CONFIGURATION	
Result	PASS

TABLE 33: CREATE RESTAURANT FLOOR PLAN TEST CASE - INVALID CONFIGURATION

Test case	Create restaurant floor plan – valid configuration
Purpose of test case	Create restaurant floor plan with valid configuration
Input	

```

1- [
2- [
3- {
4-   "number": 1,
5-   "label": "Table 1",
6-   "seatsNumber": 2
7- },
8- {
9-   "seatsNumber": 0
10- },
11- {
12-   "number": 2,
13-   "label": "Table 2",
14-   "seatsNumber": 4
15- }
16- ]
17- ]

```

FIGURE 127: INPUT - CREATE RESTAURANT FLOOR PLAN TEST CASE - VALID CONFIGURATION

Output

ADMIN TEST ✓ Floor Plan saved!

```

1- [
2- [
3- {
4-   "number": 1,
5-   "label": "Table 1",
6-   "seatsNumber": 2
7- },
8- {
9-   "seatsNumber": 0
10- },
11- {
12-   "number": 2,
13-   "label": "Table 2",
14-   "seatsNumber": 4
15- }
16- ]
17- ]

```

Save

Table 1
2 seats

Table 2
4 seats

FIGURE 128: OUTPUT - CREATE RESTAURANT FLOOR PLAN TEST CASE - VALID CONFIGURATION

Result	PASS
---------------	-------------

TABLE 34: CREATE RESTAURANT FLOOR PLAN TEST CASE - VALID CONFIGURATION

6.3. HOST functionalities

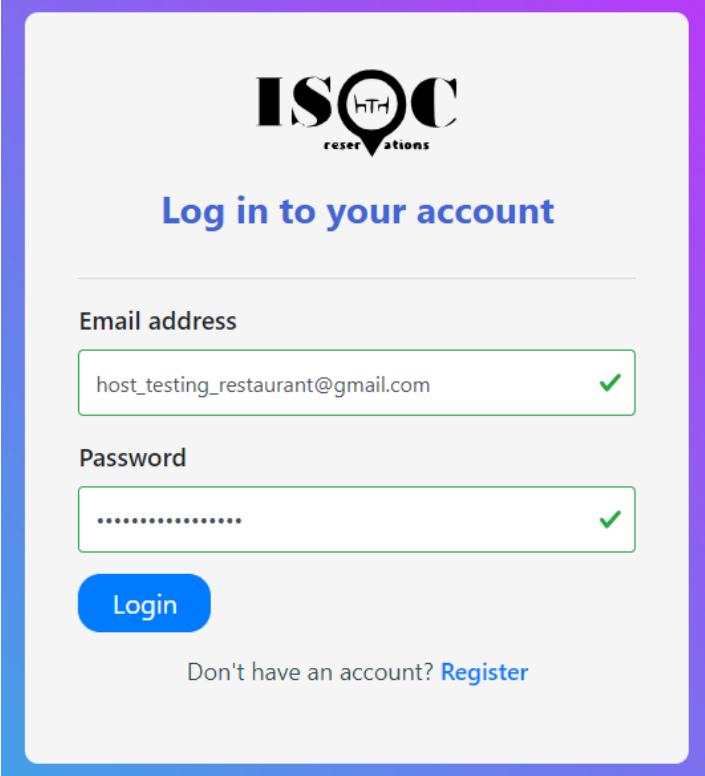
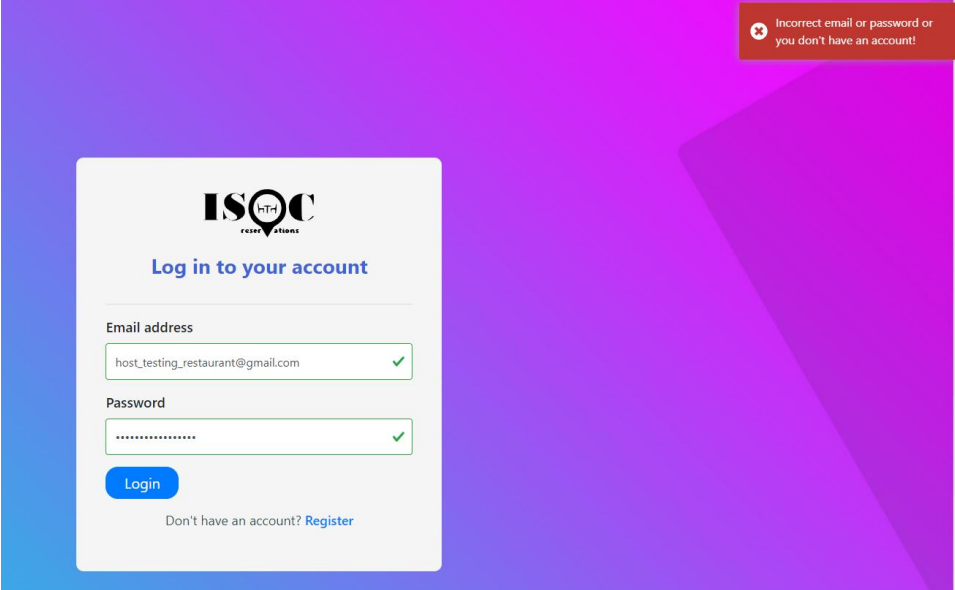
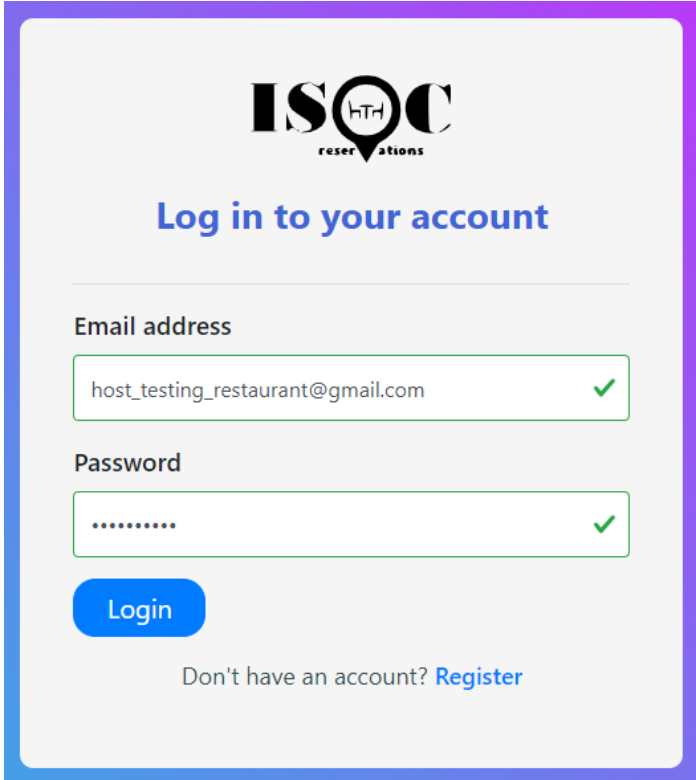
Test case	Host login – invalid credentials
Purpose of test case	Login host account with invalid credentials
Input	
Output	
Result	PASS

TABLE 35: HOST LOGIN TEST CASE - INVALID CREDENTIALS

Test case	Host login – valid credentials
Purpose of test case	Login host account with valid credentials
Input	 FIGURE 131: INPUT - HOST LOGIN TEST CASE - VALID CREDENTIALS
Output	

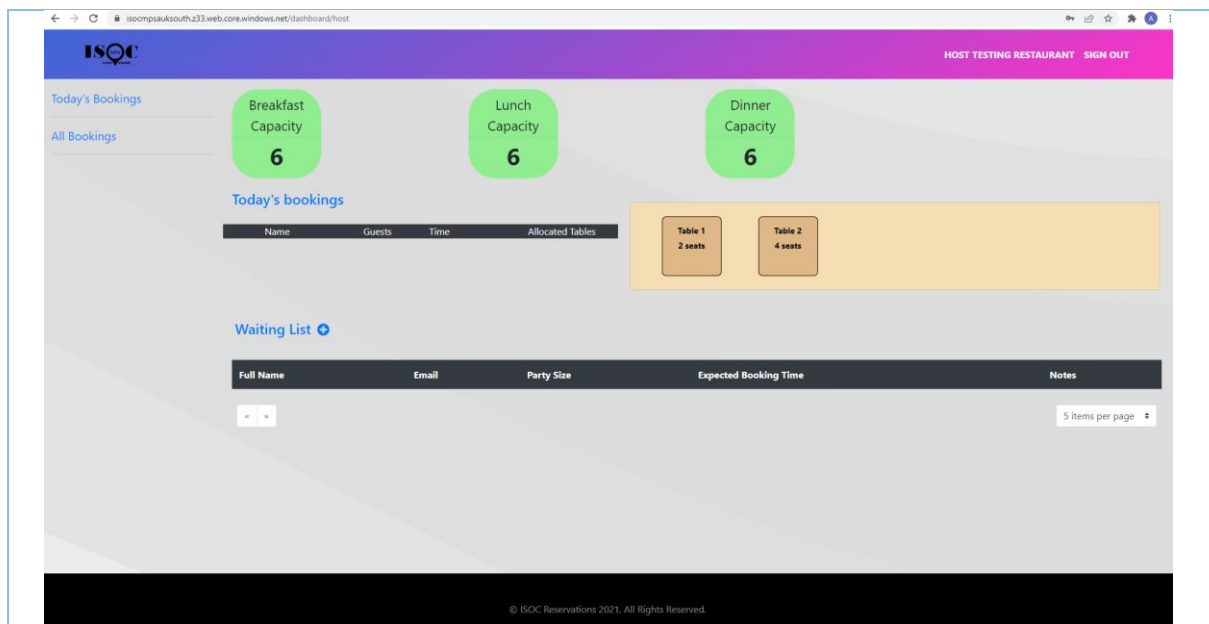


FIGURE 132: OUTPUT - HOST LOGIN TEST CASE - VALID CREDENTIALS

Result	PASS
---------------	-------------

TABLE 36: HOST LOGIN TEST CASE - VALID CREDENTIALS

Test case	View today's bookings
Purpose of test case	Display bookings for the current date
Input	The screenshot shows the ISOC Host Dashboard. The 'Today's Bookings' link is highlighted in yellow. The 'All Bookings' link is also visible. The 'Breakfast Capacity' button is also visible, displaying the number '6'.
Output	

FIGURE 133: INPUT - VIEW TODAY'S BOOKINGS TEST CASE

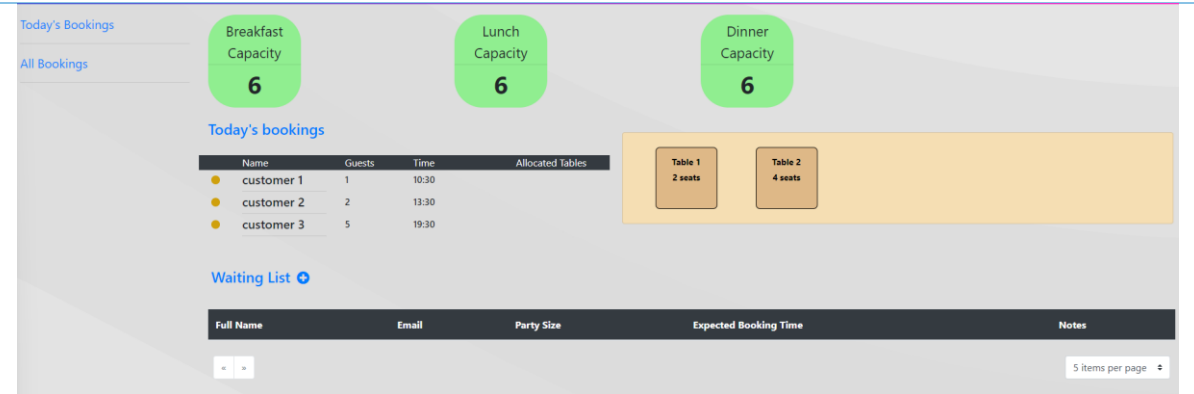


FIGURE 134: OUTPUT - VIEW TODAY’S BOOKINGS TEST CASE

Result	PASS
--------	------

TABLE 37: VIEW TODAY’S BOOKINGS TEST CASE

Test case	View all bookings
Purpose of test case	Display all bookings
Input	
Output	

FIGURE 135: INPUT - VIEW ALL BOOKINGS TEST CASE

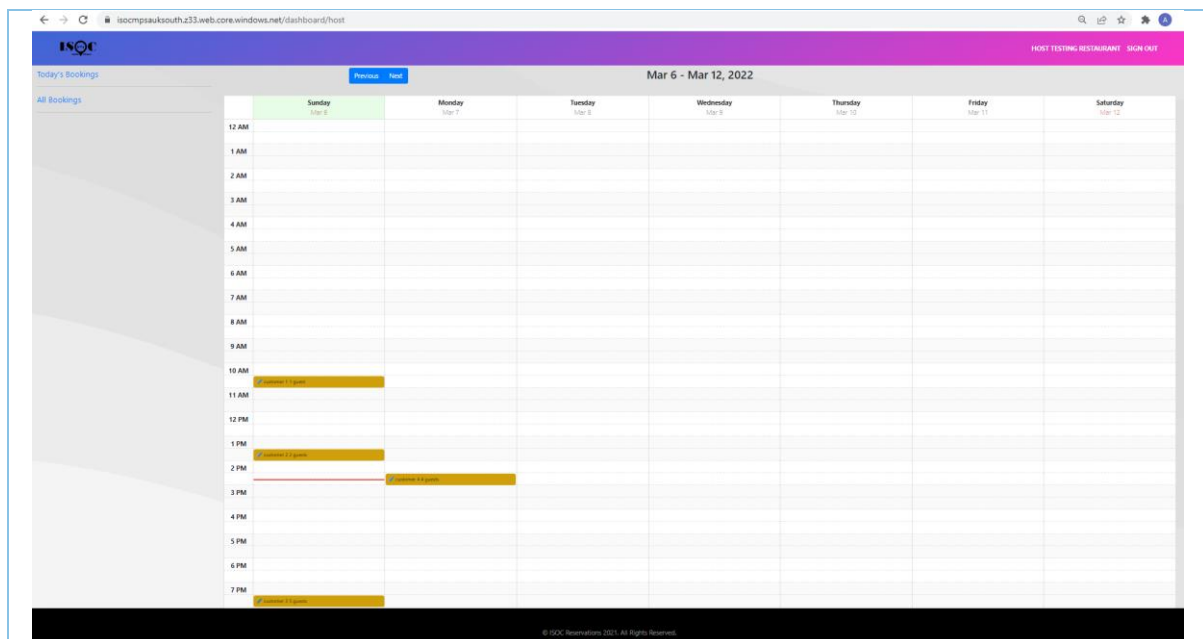


FIGURE 136: OUTPUT - VIEW ALL BOOKINGS TEST CASE

Result

PASS

TABLE 38: VIEW ALL BOOKINGS TEST CASE

Test case **Modify today's booking – booking details**

Purpose of test case Modify a today's booking details

Input

FIGURE 137: INPUT - MODIFY TODAY'S BOOKING TEST CASE – BOOKING DETAILS

Output

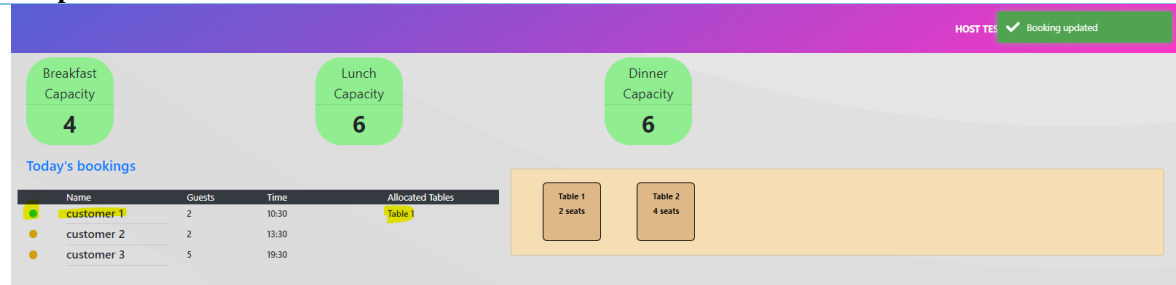


FIGURE 138: OUTPUT - MODIFY TODAY'S BOOKING TEST CASE - BOOKING DETAILS

Result

PASS

TABLE 39: MODIFY TODAY'S BOOKINGS TEST CASE

Test case

Cancel today's booking

Purpose of test case

Cancel a today's booking

Input

The screenshot shows a modal form titled 'Edit booking for guest: customer 1'. The form contains the following fields:

- Date/Time:** 6.3.2022/10:30
- Party size:** 2
- Additional notes:** test notes
- Allocated Tables:** Table 1
- Status:** Canceled (dropdown menu)

At the bottom right, there are two buttons: 'Close' and 'Save'.

FIGURE 139: INPUT - CANCEL TODAY'S BOOKING TEST CASE

Output

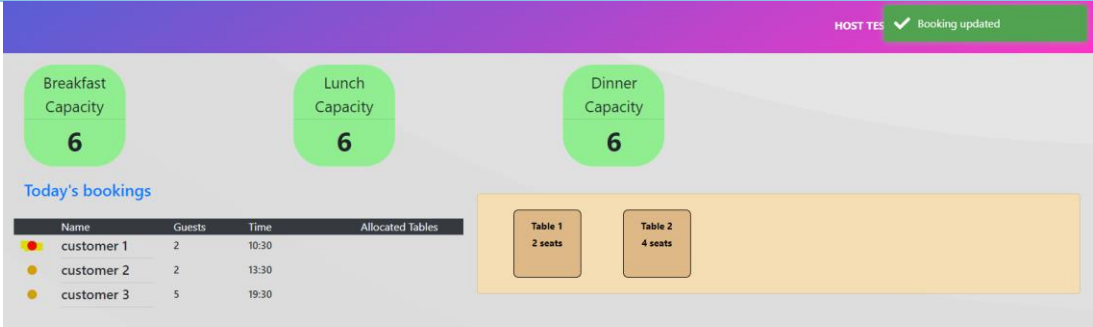
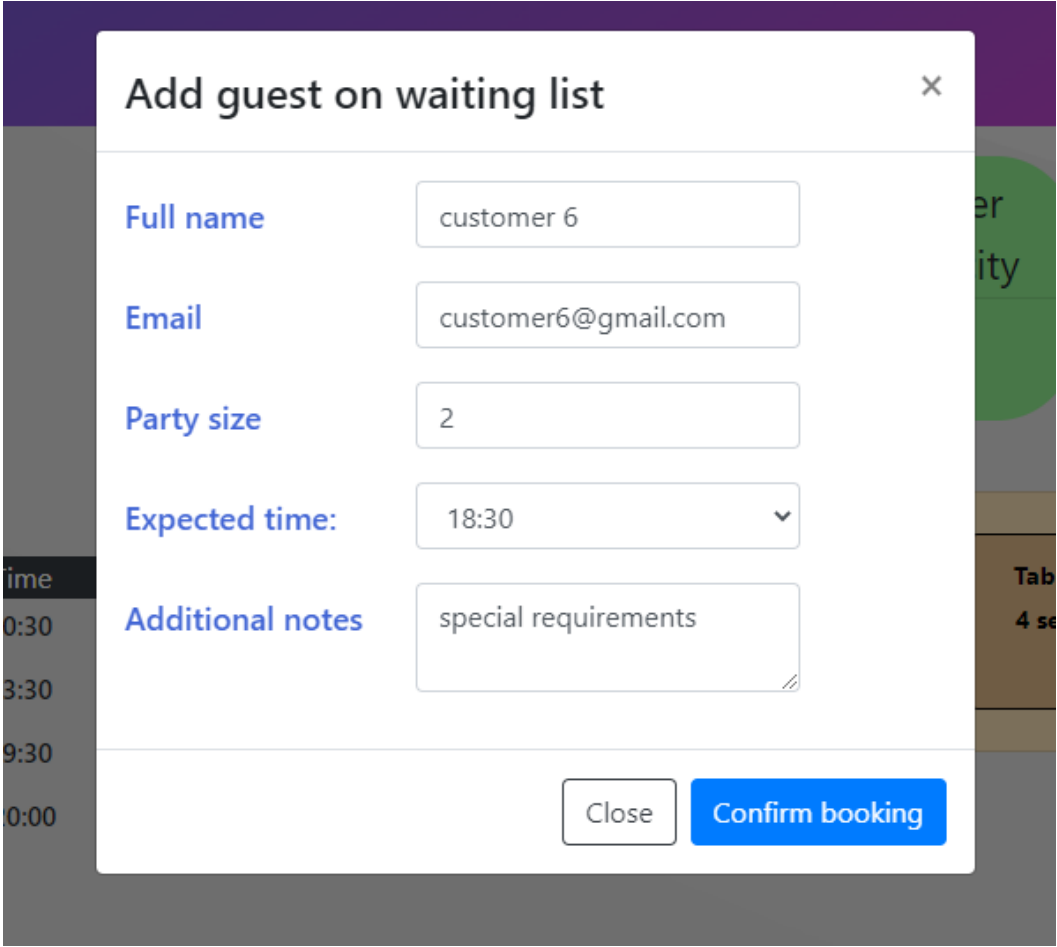
	
FIGURE 140: OUTPUT - CANCEL TODAY’S BOOKING TEST CASE	
Result	PASS

TABLE 40: CANCEL TODAY’S BOOKING TEST CASE

Test case	Customer on waiting-list
Purpose of test case	Add a customer on waiting-list
Input	
	
FIGURE 141: INPUT - CUSTOMER ON WAITING-LIST	
Output	

The screenshot displays a restaurant booking system interface. At the top, there are three green buttons labeled 'Breakfast Capacity 6', 'Lunch Capacity 6', and 'Dinner Capacity 6'. Below these, the 'Today's bookings' section contains a table with columns: Name, Guests, Time, and Allocated Tables. The table lists five bookings, with 'customer 6' highlighted in yellow. To the right of the bookings table, there are two orange boxes representing 'Table 1 2 seats' and 'Table 2 4 seats'. At the bottom, the 'Waiting List' section features a table with columns: Full Name, Email, Party Size, Expected Booking Time, and Notes. The table lists two customers, with 'customer 6' highlighted in yellow and noted as having 'special requirements'. A green notification bar at the top right states 'Customer added on waiting list!'.

FIGURE 142: OUTPUT - CUSTOMER ON WAITING-LIST

Result

PASS

TABLE 41: CUSTOMER ON WAITING-LIST

Test case	Monitor restaurant capacity																														
Purpose of test case	View restaurant capacity monitoring for breakfast, lunch, and dinner period																														
Input	Today's bookings Pending WaitingList Confirmed Partial Seated Seated Canceled <table><thead><tr><th></th><th>Name</th><th>Guests</th><th>Time</th><th>Allocated Tables</th></tr></thead><tbody><tr><td>Partial Seated</td><td>customer 1</td><td>2</td><td>10:30</td><td>Table 1</td></tr><tr><td>Seated</td><td>customer 2</td><td>2</td><td>13:30</td><td>Table 2</td></tr><tr><td>Canceled</td><td>customer 3</td><td>5</td><td>19:30</td><td></td></tr><tr><td>Confirmed</td><td>customer 5</td><td>12</td><td>20:00</td><td>Table 1</td></tr><tr><td>WaitingList</td><td>customer 6</td><td>2</td><td>18:30</td><td></td></tr></tbody></table>		Name	Guests	Time	Allocated Tables	Partial Seated	customer 1	2	10:30	Table 1	Seated	customer 2	2	13:30	Table 2	Canceled	customer 3	5	19:30		Confirmed	customer 5	12	20:00	Table 1	WaitingList	customer 6	2	18:30	
	Name	Guests	Time	Allocated Tables																											
Partial Seated	customer 1	2	10:30	Table 1																											
Seated	customer 2	2	13:30	Table 2																											
Canceled	customer 3	5	19:30																												
Confirmed	customer 5	12	20:00	Table 1																											
WaitingList	customer 6	2	18:30																												
Output	ISQC Today's Bookings All Bookings Breakfast Capacity 4 Lunch Capacity 4 Dinner Capacity -6																														
Result	PASS																														

TABLE 42: MONITOR RESTAURANT CAPACITY TEST CASE

6.4. CUSTOMER functionalities

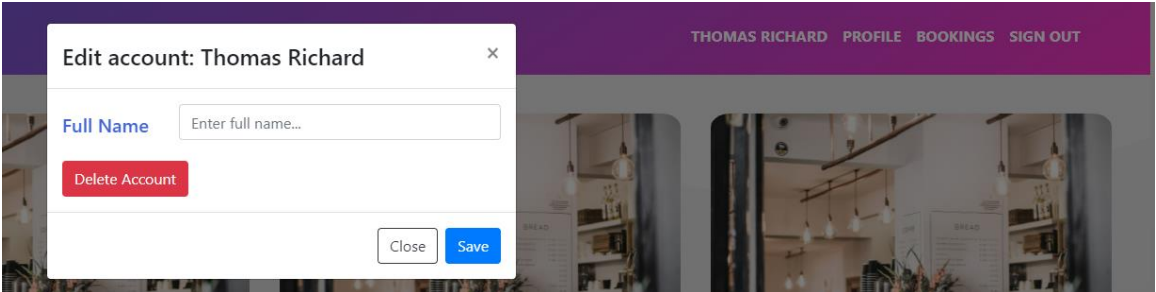
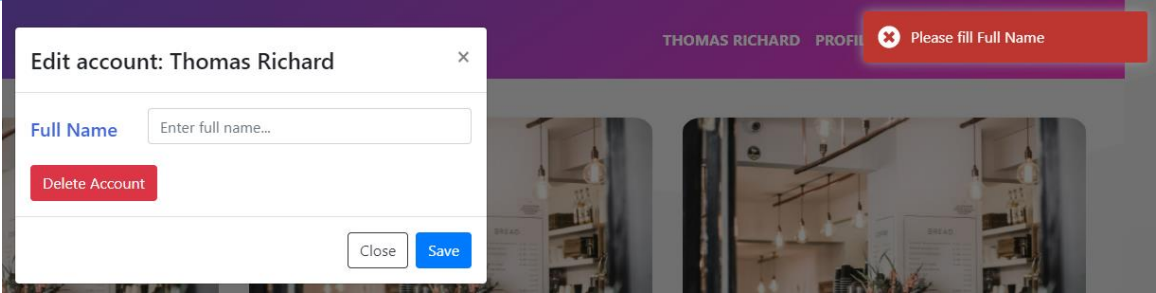
Test case	Modify account – invalid data
Purpose of test case	Modify account with invalid full name
Input	
	
FIGURE 145: INPUT - MODIFY ACCOUNT TEST CASE - INVALID DATA	
Output	
	
FIGURE 146: OUTPUT - MODIFY ACCOUNT TEST CASE - INVALID DATA	
Result	PASS

TABLE 43: MODIFY ACCOUNT TEST CASE - INVALID DATA

Test case	Modify account – valid data
Purpose of test case	Modify account with valid full name
Input	

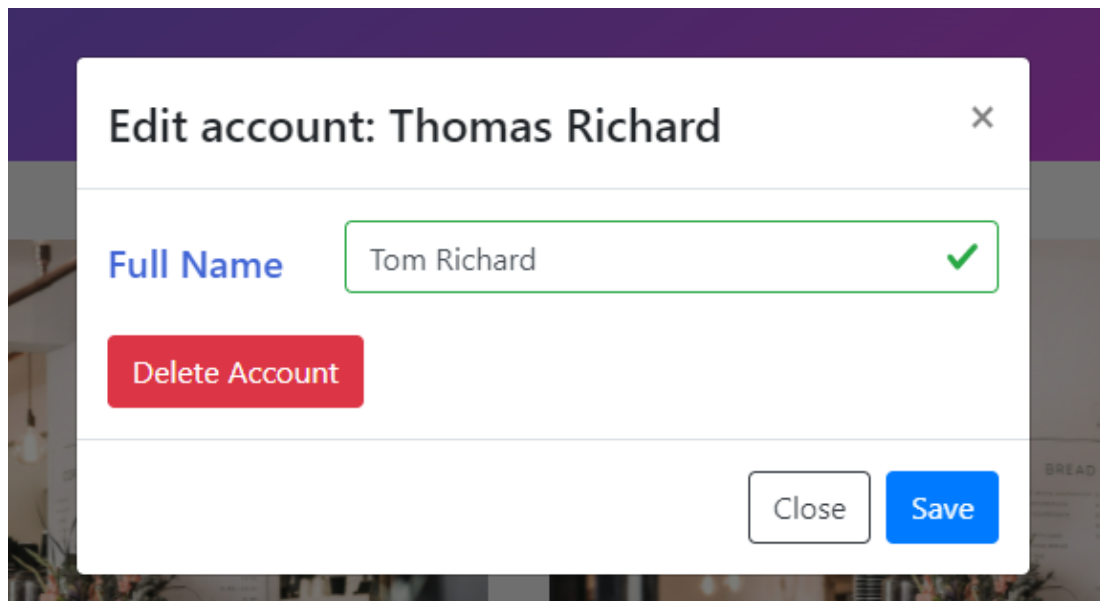


FIGURE 147: INPUT - MODIFY ACCOUNT TEST CASE - VALID DATA

Output	
---------------	--

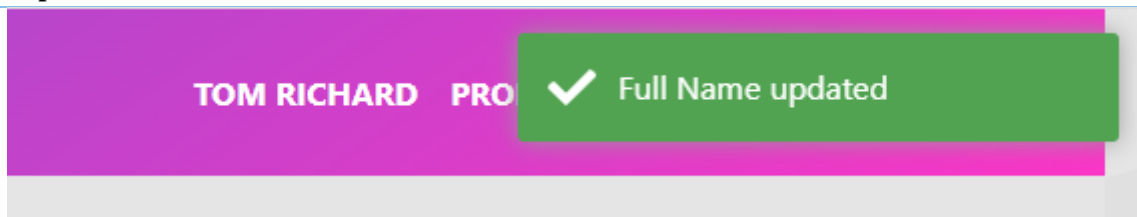


FIGURE 148: OUTPUT - MODIFY ACCOUNT TEST CASE - VALID DATA

Result	PASS
---------------	-------------

TABLE 44: MODIFY ACCOUNT TEST CASE - VALID DATA

Test case	View bookings
Purpose of test case	Display customer's bookings
Input	

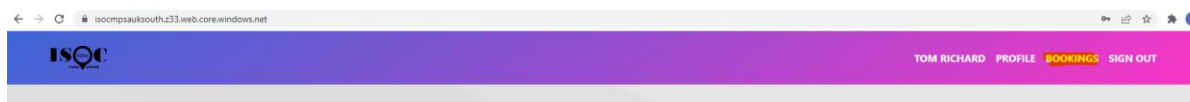


FIGURE 149: INPUT - VIEW BOOKINGS TEST CASE

Output	
---------------	--

Booking history						×
Restaurant	Party Size	Date & Time	Notes	Status		
Atera London	2	3.3.2022 / 19:0		● pending	Cancel booking	
Barclay Prime Restaurant	2	4.3.2022 / 23:0		● canceled		
« 1 » 5 items per page ▾						
FIGURE 150: OUTPUT - VIEW BOOKINGS TEST CASE						

Result	PASS
--------	------

TABLE 45: VIEW BOOKINGS TEST CASE

Test case	Send booking request – invalid data
Purpose of test case	Send a booking request with invalid data
Input	Party size 0 Date < Mar 2022 > Mo Tu We Th Fr Sa Su 28 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 Time 6:00 > Additional notes Close Confirm booking
Output	FIGURE 151: INPUT - SENT BOOKING REQUEST TEST CASE - INVALID DATA

The menu specializes in Northern Italian cuisine with seasonal changes to reflect freshness of ingredients with a focus on locally grown produce.

Party size

Date

	Mo	Tu	We	Th	Fr	Sa	Su
28	1	2	3	4	5	6	
7	8	9	10	11	12	13	
14	15	16	17	18	19	20	
21	22	23	24	25	26	27	
28	29	30	31	1	2	3	
4	5	6	7	8	9	10	

Time

Additional notes

Please select the number of guests (party size)

FIGURE 152: OUTPUT - SENT BOOKING REQUEST TEST CASE - INVALID DATA

Result

PASS

TABLE 46: SENT BOOKING REQUEST TEST CASE - INVALID DATA

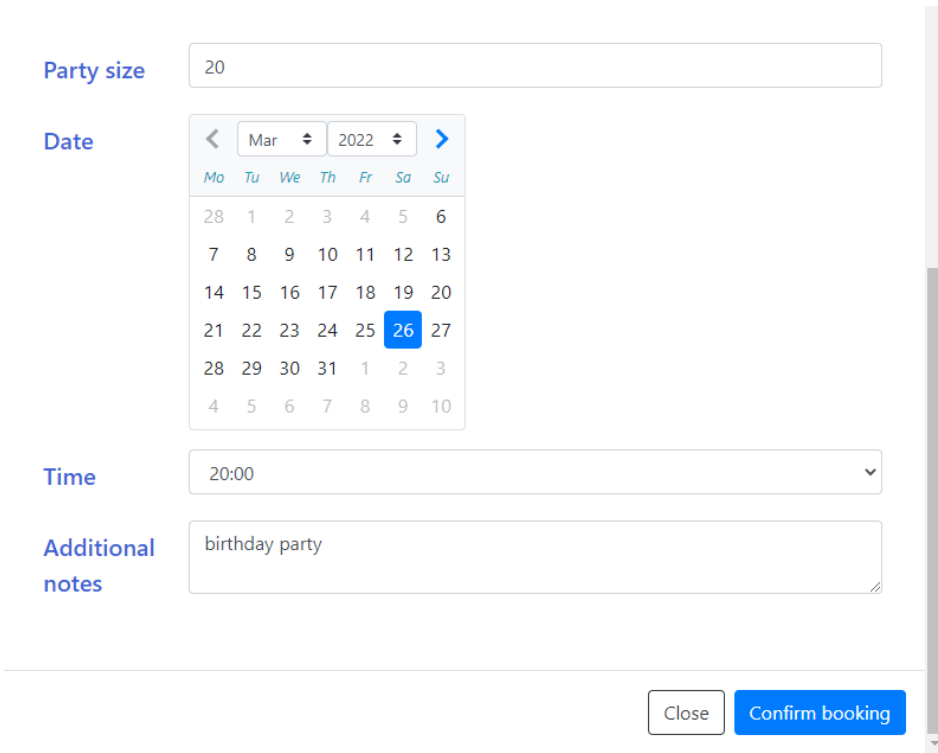
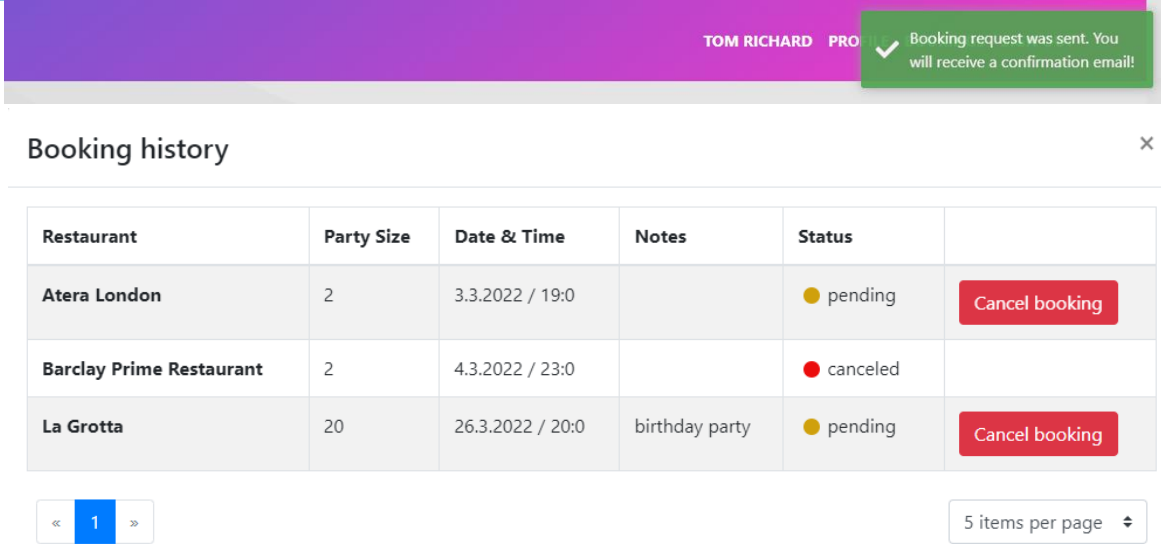
Test case	Send booking request – valid data
Purpose of test case	Send a booking request with valid data
Input	
	
FIGURE 153: INPUT - SEND BOOKING REQUEST TEST CASE - VALID DATA	
Output	
	
FIGURE 154: OUTPUT - SEND BOOKING REQUEST TEST CASE - VALID DATA	
Result	PASS

TABLE 47: SEND BOOKING REQUEST TEST CASE - VALID DATA

Test case	Cancel booking																								
Purpose of test case	Cancel a booking																								
Input																									
Booking history <table border="1"> <thead> <tr> <th>Restaurant</th> <th>Party Size</th> <th>Date & Time</th> <th>Notes</th> <th>Status</th> <th></th> </tr> </thead> <tbody> <tr> <td>Atera London</td> <td>2</td> <td>3.3.2022 / 19:0</td> <td></td> <td>● pending</td> <td>Cancel booking</td> </tr> <tr> <td>Barclay Prime Restaurant</td> <td>2</td> <td>4.3.2022 / 23:0</td> <td></td> <td>● canceled</td> <td></td> </tr> <tr> <td>La Grotta</td> <td>20</td> <td>26.3.2022 / 20:0</td> <td>birthday party</td> <td>● pending</td> <td>Cancel booking</td> </tr> </tbody> </table> << 1 >> 5 items per page		Restaurant	Party Size	Date & Time	Notes	Status		Atera London	2	3.3.2022 / 19:0		● pending	Cancel booking	Barclay Prime Restaurant	2	4.3.2022 / 23:0		● canceled		La Grotta	20	26.3.2022 / 20:0	birthday party	● pending	Cancel booking
Restaurant	Party Size	Date & Time	Notes	Status																					
Atera London	2	3.3.2022 / 19:0		● pending	Cancel booking																				
Barclay Prime Restaurant	2	4.3.2022 / 23:0		● canceled																					
La Grotta	20	26.3.2022 / 20:0	birthday party	● pending	Cancel booking																				

FIGURE 155: INPUT - CANCEL BOOKING TEST CASE

Output																													
Are you sure you want to cancel your booking? OK Cancel <table border="1"> <thead> <tr> <th>Restaurant</th> <th>Party Size</th> <th>Date & Time</th> <th>Notes</th> <th>Status</th> <th></th> </tr> </thead> <tbody> <tr> <td>Atera London</td> <td>2</td> <td>3.3.2022 / 19:0</td> <td></td> <td>● pending</td> <td>Cancel booking</td> </tr> <tr> <td>Barclay Prime Restaurant</td> <td>2</td> <td>4.3.2022 / 23:0</td> <td></td> <td>● canceled</td> <td></td> </tr> <tr> <td>La Grotta</td> <td>20</td> <td>26.3.2022 / 20:0</td> <td>birthday party</td> <td>● pending</td> <td>Cancel booking</td> </tr> </tbody> </table> << 1 >> 5 items per page						Restaurant	Party Size	Date & Time	Notes	Status		Atera London	2	3.3.2022 / 19:0		● pending	Cancel booking	Barclay Prime Restaurant	2	4.3.2022 / 23:0		● canceled		La Grotta	20	26.3.2022 / 20:0	birthday party	● pending	Cancel booking
Restaurant	Party Size	Date & Time	Notes	Status																									
Atera London	2	3.3.2022 / 19:0		● pending	Cancel booking																								
Barclay Prime Restaurant	2	4.3.2022 / 23:0		● canceled																									
La Grotta	20	26.3.2022 / 20:0	birthday party	● pending	Cancel booking																								
Booking history <table border="1"> <thead> <tr> <th>Restaurant</th> <th>Party Size</th> <th>Date & Time</th> <th>Notes</th> <th>Status</th> <th></th> </tr> </thead> <tbody> <tr> <td>Atera London</td> <td>2</td> <td>3.3.2022 / 19:0</td> <td></td> <td>● pending</td> <td>Cancel booking</td> </tr> <tr> <td>Barclay Prime Restaurant</td> <td>2</td> <td>4.3.2022 / 23:0</td> <td></td> <td>● canceled</td> <td></td> </tr> <tr> <td>La Grotta</td> <td>20</td> <td>26.3.2022 / 20:0</td> <td>birthday party</td> <td>● canceled</td> <td></td> </tr> </tbody> </table> << 1 >> 5 items per page						Restaurant	Party Size	Date & Time	Notes	Status		Atera London	2	3.3.2022 / 19:0		● pending	Cancel booking	Barclay Prime Restaurant	2	4.3.2022 / 23:0		● canceled		La Grotta	20	26.3.2022 / 20:0	birthday party	● canceled	
Restaurant	Party Size	Date & Time	Notes	Status																									
Atera London	2	3.3.2022 / 19:0		● pending	Cancel booking																								
Barclay Prime Restaurant	2	4.3.2022 / 23:0		● canceled																									
La Grotta	20	26.3.2022 / 20:0	birthday party	● canceled																									

FIGURE 156: OUTPUT - CANCEL BOOKING TEST CASE

Result	PASS
---------------	-------------

TABLE 48: CANCEL BOOKING TEST CASE

6.5. ANONYMOUS USER functionalities

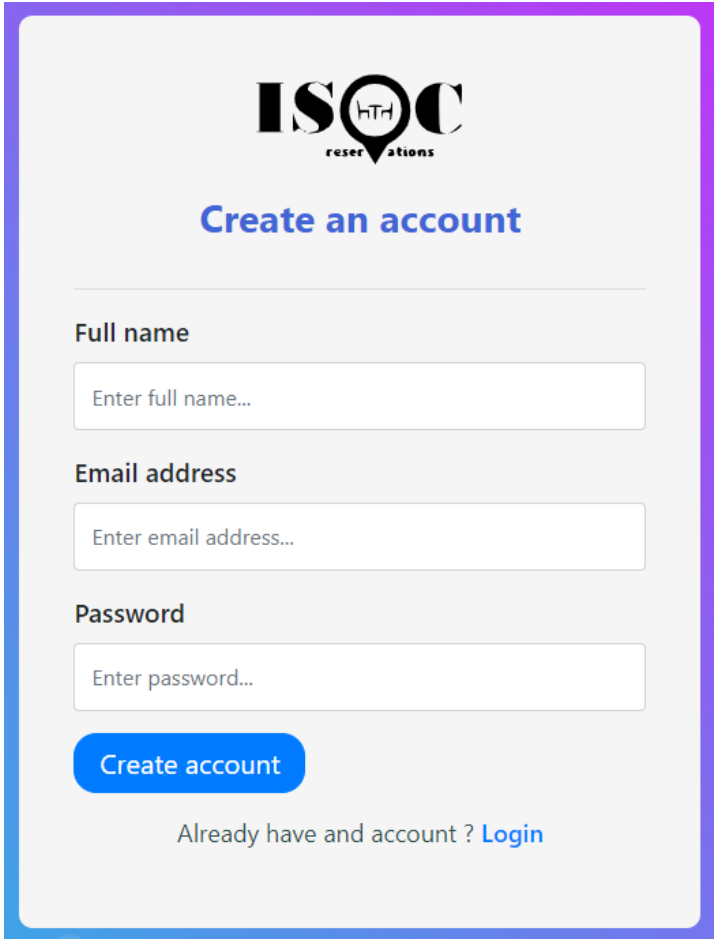
Test case	Create account – invalid data
Purpose of test case	Create a new customer account with invalid data
Input	
Output	

FIGURE 157: INPUT - CREATE ACCOUNT - INVALID DATA



Create an account

Full name

Enter full name...

Full name is required

Email address

Enter email address...

Email address is required

Password

Enter password...

Password is required

Create account

Already have an account ? [Login](#)

FIGURE 158: OUTPUT - CREATE ACCOUNT - INVALID DATA

Result	PASS
--------	------

TABLE 49: CREATE ACCOUNT - INVALID DATA

Test case	Create account – invalid email or password
Purpose of test case	Create a new customer account with invalid email or password
Input	

The screenshot shows the 'ISOC reservations' logo at the top, followed by the heading 'Create an account'. Below this are three input fields: 'Full name' with the value 'testing customer' and a green checkmark; 'Email address' with the value 'testing_customer' and a red error message 'Invalid email address'; and 'Password' with the value '****' and a red error message 'Password must be at least 6 characters'. At the bottom of the form is a blue 'Create account' button and a link that says 'Already have an account ? Login'.

FIGURE 159: INPUT - CREATE ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD

Output

This screenshot is identical to the one in Figure 159, showing the 'ISOC reservations' logo, the 'Create an account' heading, and the three input fields with their respective values and error messages. The 'Full name' field is correct, while the 'Email address' and 'Password' fields show validation errors. The 'Create account' button and the 'Login' link are also visible at the bottom.

FIGURE 160: OUTPUT - CREATE ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD

Result	PASS
---------------	-------------

TABLE 50: CREATE ACCOUNT TEST CASE - INVALID EMAIL OR PASSWORD

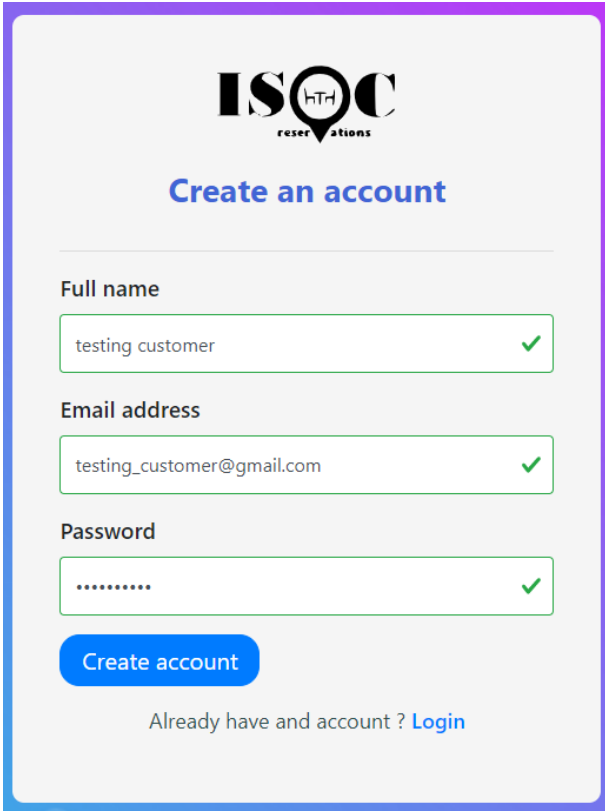
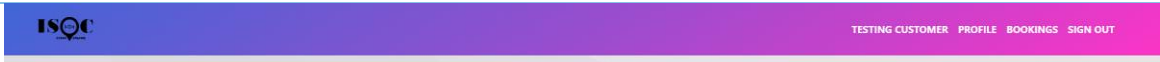
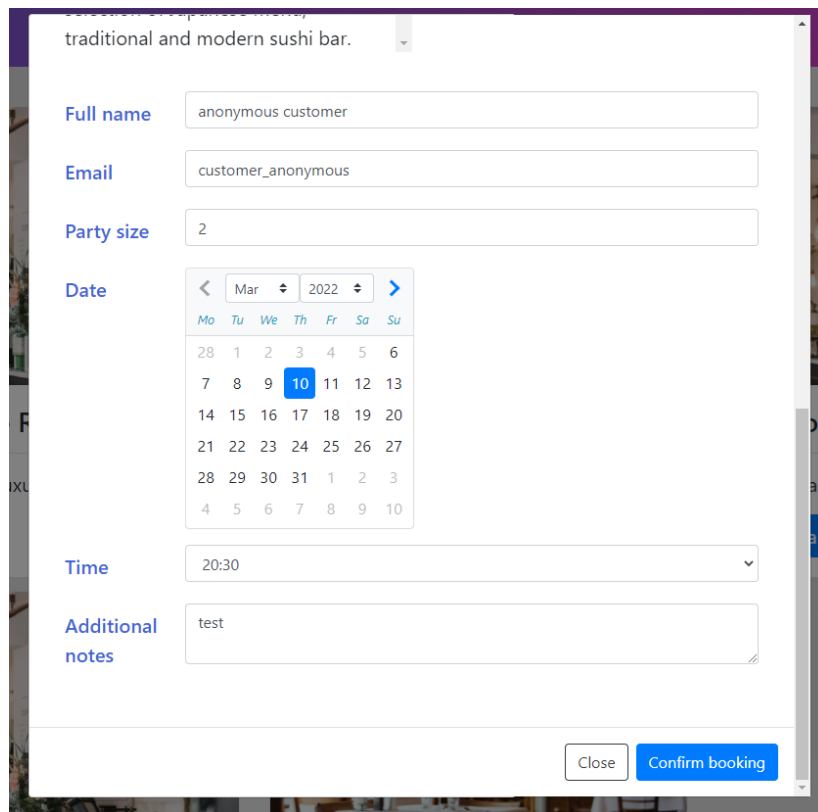
Test case	Create account – valid data
Purpose of test case	Create a new customer account with valid data
Input	 FIGURE 161: INPUT - CREATE ACCOUNT TEST CASE - VALID DATA
Output	 FIGURE 162: OUTPUT - CREATE ACCOUNT TEST CASE - VALID DATA
Result	PASS

TABLE 51: CREATE ACCOUNT TEST CASE - VALID DATA

Test case	Send booking request – invalid data
Purpose of test case	Send a booking request with invalid data

Input



traditional and modern sushi bar.

Full name anonymous customer

Email customer_anonymous

Party size 2

Date Mar 2022

Mo	Tu	We	Th	Fr	Sa	Su
28	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

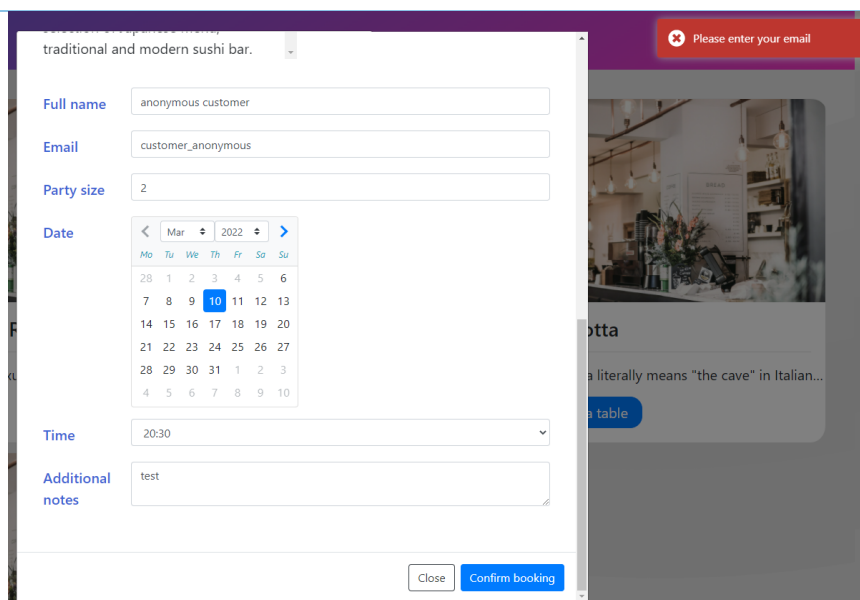
Time 20:30

Additional notes test

Close Confirm booking

FIGURE 163: INPUT - SEND BOOKING REQUEST TEST CASE - INVALID DATA

Output



traditional and modern sushi bar.

Full name anonymous customer

Email customer_anonymous

Party size 2

Date Mar 2022

Mo	Tu	We	Th	Fr	Sa	Su
28	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

Time 20:30

Additional notes test

Close Confirm booking

Please enter your email

FIGURE 164: OUTPUT - SEND BOOKING REQUEST TEST CASE - INVALID DATA

Result	PASS
---------------	-------------

TABLE 52: SEND BOOKING REQUEST TEST CASE - INVALID DATA

Test case	Send booking request – valid data
Purpose of test case	Send a booking request with valid data
Input	

traditional and modern sushi bar.

Full name anonymous customer

Email customer_anonymous@gmail.com

Party size 2

Date Mar 2022

Time 20:30

Additional notes test

Close Confirm booking

FIGURE 165: INPUT SEND BOOKING REQUEST TEST CASE - VALID DATA

Output

✓ Booking request was sent. You will receive a confirmation email!

BONDST London
BONDST, is a trendy downtown restaurant ...
Book a table

La Grotta
La Grotta literally means "the cave" in Italian...
Book a table

FIGURE 166: OUTPUT - SEND BOOKING REQUEST TEST CASE - VALID DATA

Result	PASS
---------------	-------------

TABLE 53: SEND BOOKING REQUEST TEST CASE - VALID DATA

6.6. SECURITY

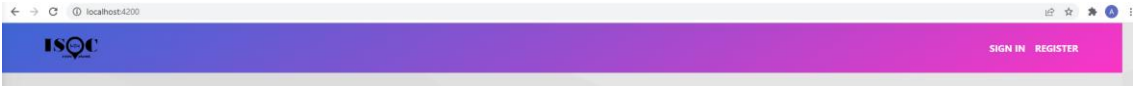
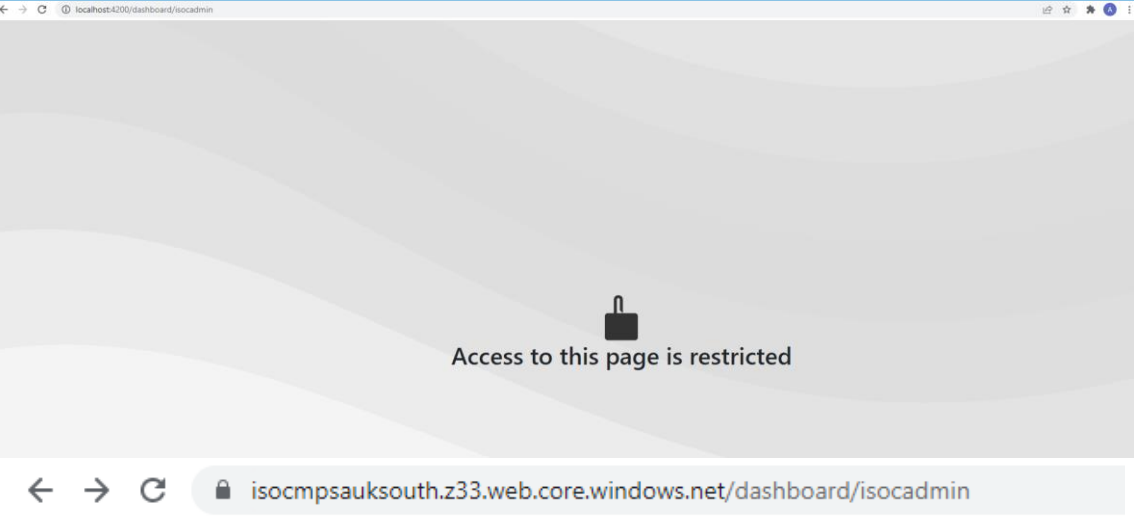
Test case	Authorization access
Purpose of test case	Access isocadmin/admin/host or customer dashboard without permissions
Input	
 FIGURE 167: INPUT - AUTHORIZATION ACCESS TEST CASE	
Output	
 FIGURE 168: OUTPUT - AUTHORIZATION ACCESS TEST CASE	
Result	PASS

TABLE 54: AUTHORIZATION ACCESS TEST CASE

Chapter 7 - Evaluation

This chapter presents a detailed evaluation of ISOC Reservations system and a personal evaluation highlighting the achievements obtained during the execution of the project.

7.1. Personal evaluation

During this project, a set of soft and hard skills have considerably improved. The achievements that demonstrate this progress are listed as following:

- I improved my research and analytical capabilities by rigorously reviewing multiple reference sources and extracting the most relevant information
- I managed my time effectively by delivering the project on expected deadline and estimating and prioritising the required activities
- I improved my decision-making and adaptability skills by handling the blocking situations and looking for alternative working solutions
- I improved my communication skills by actively listening for feedback
- I improved my presentation skills by describing my work every week
- I improved my technical skills by learning new programming languages, frameworks, and tools
 - I passed Microsoft Azure Fundamentals certification
 - I followed tutorials using e-learning platforms as LinkedIn Learning, Pluralsight, Code Academy and official documentations, books, and forums

Overall, I became confident, and I was involved in a project for the entire process. Hard work and commitment were the characteristics that drove me during this experience and the gained benefits will be valuable for my personal and professional life.

7.2. System evaluation

The aim and objectives of the project were met as expected. Current version of the system has all the functional and non-functional requirements implemented. Chapter 6 testing evidence shows that the system is fully functional. The system offers a user-friendly interface, a set of features that covers the business requirements for restaurants and for customers, a simplified setup process and an increased level of security. These factors indicate the successful project completion.

Chapter 8 - Further enhancement

The current version of ISOC Reservations system satisfies the most important requirements for restaurant reservations processes; however, the following improvements can be addressed in the future:

- Send email confirmation for booking requests
- Add “customer reviews” functionality
- Add functionality for removing a restaurant from the system
- Add functionality for password recovery
- Improve security by storing the database connection details in Azure Key Vault
- Improve security by adding multi-factor authentication
- Improve application monitoring by using Azure App Insights
- Automate the deployment process

Chapter 9 - Conclusion

The aim of this project was to develop a restaurant booking system web application hosted in cloud by using Microsoft Azure cloud provider. Chapter 1 introduced the background and motivations for this project and listed the objectives required for a successful completion. Based on the literature review conveyed in Chapter 2, it can be concluded that restaurant reservations systems are a valuable solution for increasing restaurants profits and customers satisfaction and their provided features are essential. A comparison between several existing similar systems was mandatory for developing ISOC Reservations application with all the required characteristics to be an efficient and sustainable solution, and it drove the analysis process to establish a list of functional, non-functional and system requirements as well as technologies and tools.

Chapter 4 and 5 were focused on designing and implementing the ISOC Reservations application, by illustrating how two of the main advantages of the system, security, and cloud deployment, were integrated. The result of testing phase performed in Chapter 6 demonstrated that all the scenarios are working as expected and the system is ready to satisfy restaurants business requirements. However, during this testing phase, a set of enhancements were identified and considered in Chapter 8 for further implementation.

As a final observation, the project was successfully completed, and the achievements are valuable from both system and personal perspectives.

References

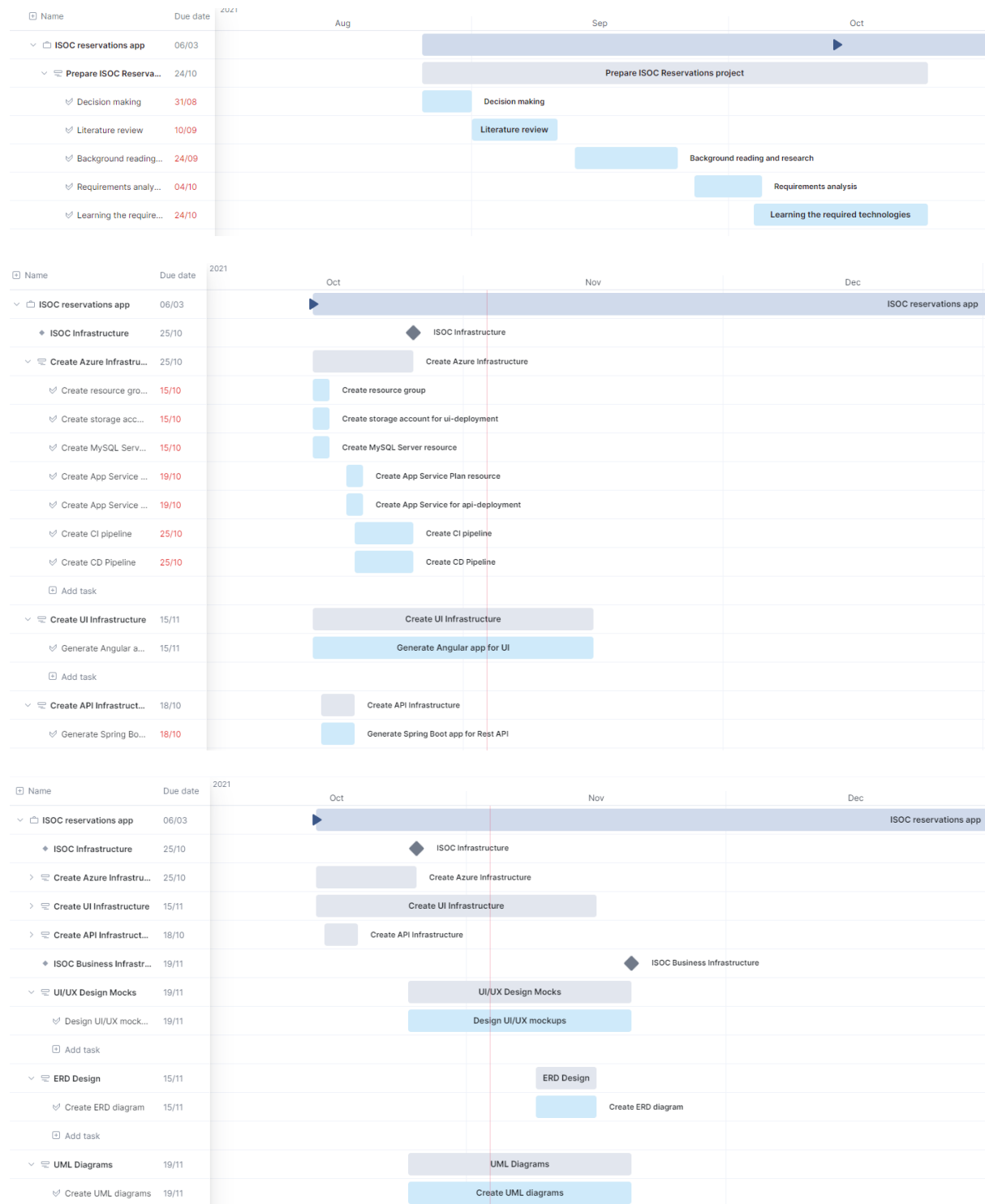
- [1] R. L. Spang, *The Invention of the Restaurant, Paris and modern gastronomic culture*, Cambridge, Massachusetts, and London, England: Harvard University Press, 2000.
- [2] Altexsoft, "History of Flight Booking: CRSs, GDS Distribution, Travel Agencies, and Online Reservations," Altexsoft, 12 04 2019. [Online]. Available: <https://www.altexsoft.com/blog/travel/history-of-flight-booking-crss-gds-distribution-travel-agencies-and-online-reservations/>.
- [3] IBM Corporation, "Sabre, The First Online Reservation System," IBM Corporation, 07 03 2012. [Online]. Available: <https://www.ibm.com/ibm/history/ibm100/us/en/icons/sabre/>.
- [4] OpenTable Inc, "Technology and Dining Out 2015," OpenTable, San Francisco, 2015.
- [5] OpenTable Inc, "What diners expect in 2021," OpenTable, United Kingdom, 2021.
- [6] Flexera, "Flexera 2022 State of the Cloud Report," Flexera, 2022.
- [7] B. Gupta, P. Mittal and T. Mufti, "A Review on Amazon Web Service (AWS), Microsoft Azure & Google Cloud Platform (GCP) Services," in *ICIDSSD 2020*, New Delhi, India, 2020.
- [8] T. Warner, *Microsoft Azure for dummies*, Hoboken, New Jersey: John Wiley & Sons Inc., 2020.
- [9] B. Golden, *Amazon Web Services For Dummies*, John Wiley & Sons Inc., 2013.
- [10] OWASP, "Official documentation - OWASP Top 10 - 2021," OWASP, 2021. [Online]. Available: <https://owasp.org/Top10/>.
- [11] H. D. Canavan, "Yelp Turns 10: From Startup to Online Review Dominance," 05 08 2014. [Online]. Available: <https://www.eater.com/2014/8/5/6177213/yelp-turns-10-from-startup-to-online-review-dominance>.
- [12] M. Luca, "Reviews, Reputation, and Revenue: The Case of Yelp.Com," Harvard Business School, 2016.
- [13] OpenTable Inc, "Official website - Our story," OpenTable Inc, [Online]. Available: <https://www.opentable.com/about/?lang=en>.
- [14] S. Lacy, "How OpenTable Could Actually Matter," 03 05 2009. [Online]. Available: <https://techcrunch.com/2009/05/03/how-opentable-could-actually-matter/?guccounter=1>.

- [15] Table's Ready, LLC, "TablesReady - LinkedIn official page," Table's Ready, LLC, [Online]. Available: <https://www.linkedin.com/company/table's-ready/about/>.
- [16] Yelp Inc, "Official website - products," Yelp Inc, [Online]. Available: <https://restaurants.yelp.com/products/>.
- [17] OpenTable Inc, "Official website - solutions," OpenTable Inc, [Online]. Available: <https://restaurant.opentable.co.uk/products/>.
- [18] Table's Ready, LLC, "Official website - features," Table's Ready, LLC, [Online]. Available: <https://www.tablesready.com/features>.
- [19] K. W. a. Seilevel, Software Requirements, 3rd Edition, Microsoft, 2013.
- [20] T. S. Raffles, A History of Java, Cambridge University Press, 2012.
- [21] Oracle, "Java™ Platform Standard Edition 17 Development Kit - JDK™ 17," Oracle, [Online]. Available: <https://www.oracle.com/java/technologies/javase/jdk17-readme-downloads.html>.
- [22] VMware, Inc., "Spring Boot," [Online]. Available: <https://spring.io/projects/spring-boot>.
- [23] VMware, Inc, "Spring Data JPA," [Online]. Available: <https://spring.io/projects/spring-data-jpa>.
- [24] VMware, Inc, "Spring Security," [Online]. Available: <https://spring.io/projects/spring-security>.
- [25] Apache Software Foundation, "Official documentation - Apache Tomcat," [Online]. Available: <https://tomcat.apache.org/>.
- [26] Google , "Official documentation - Angular," [Online]. Available: <https://angular.io/docs>.
- [27] B. team, "Official documentation - Bootstrap," [Online]. Available: <https://getbootstrap.com/docs/5.1/getting-started/introduction/>.
- [28] OpenJS Foundation, "Official documentation - NodeJs," [Online]. Available: <https://nodejs.org/en/about/>.
- [29] Microsoft , "Official documentation - App Service," [Online]. Available: <https://azure.microsoft.com/en-us/services/app-service/#overview>.
- [30] Microsoft, "Official documentation - Azure Storage," [Online]. Available: <https://docs.microsoft.com/en-us/azure/storage/common/storage-introduction>.
- [31] Microsoft , "Official documentation - Azure SQL Database," [Online]. Available: <https://azure.microsoft.com/en-us/products/azure-sql/database/#overview>.
- [32] Microsoft, "Official documentation - Azure Maps," [Online]. Available: <https://azure.microsoft.com/en-us/services/azure-maps/#azuremaps-overview>.

- [33] Microsoft , “Official documentation - Azure DevOps,” [Online]. Available: <https://azure.microsoft.com/en-gb/services/devops/>.
- [34] Microsoft, “Official website - Visual Studio Code,” [Online]. Available: <https://code.visualstudio.com/>.
- [35] JetBrains s.r.o, “Official website - IntelliJ IDEA,” [Online]. Available: <https://www.jetbrains.com/idea/features/>.
- [36] Oracle Corporation, “Official website - MySQL Workbench,” [Online]. Available: <https://www.mysql.com/products/workbench/>.
- [37] “Official website - FileZilla,” [Online]. Available: <https://filezilla-project.org/>.
- [38] Visual Paradigm, “Official website - Visual Paradigm,” [Online]. Available: <https://www.visual-paradigm.com/>.
- [39] “Official website - Gantt,” [Online]. Available: <https://www.gantt.com/>.
- [40] C. Drumond, “Agile Manifesto,” [Online]. Available: <https://www.atlassian.com/agile/manifesto>.
- [41] M. J. Chonoles and J. A. Schardt, UML 2 for Dummie, Hungry Minds, 2003.
- [42] J. McManus, Risk Management in Software Development Projects, New York: Routledge, 2011.
- [43] Microsoft , “Database design basics,” [Online]. Available: <https://support.microsoft.com/en-us/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5>.

Appendices

Appendix A – Gantt Chart monthly view



Name	Due date	2021	Oct	Nov
◆ ISOC Authentication &...	11/11			◆ ISOC Authentication & Authorization
▼ Create login page	03/11		Create login page	
✓ Create login page UI	25/10		Create login page UI	
✓ UI/UX login page	03/11			UI/UX login page
✓ Create login page A...	28/10		Create login page API endpoints	
✓ Manual Testing	30/10		Manual Testing	
✓ Tech doc	30/10		Tech doc	
+	Add task			
▼ Create register page	11/11			Create register page
✓ Create register pag...	08/11			Create register page UI
✓ UI/UX register page	05/11			UI/UX register page
✓ Create register pag...	09/11			Create register page API endpoints
✓ Manual Testing	11/11			Manual Testing
✓ Tech doc	10/11			Tech doc

Name	Due date	2021	Oct	Nov	Dec
◆ System_Admin	03/12				◆ System_Admin
▼ Create landing page	18/11			Create landing page	
✓ [UI] App landing page	15/11			[UI] App landing page	
✓ [API] Create endpoi...	17/11			[API] Create endpoint for registered restaurants	
✓ Manual Test	17/11			Manual Test	
✓ UI/UX Design	18/11			UI/UX Design	
✓ Tech doc	18/11			Tech doc	
+	Add task				
▼ Create on-boarding a...	23/11			Create on-boarding admin account	
✓ [UI] Create on-boar...	22/11			[UI] Create on-boarding admin account for specific registered restaurant	
✓ [API] Create on-bo...	22/11			[API] Create on-boarding admin account for specific registered restaurant	
✓ Manual Test	23/11			Manual Test	
✓ Tech doc	22/11			Tech doc	
✓ UI/UX Design	23/11			UI/UX Design	

Name	Due date	2021	Oct	Nov	Dec
▼ Create on-boarding re...	24/11			Create on-boarding restaurant profile	
✓ [UI] Create on-boar...	22/11			[UI] Create on-boarding restaurant profile	
✓ [API] Create on-bo...	22/11			[API] Create on-boarding restaurant profile	
✓ Manual Test	24/11			Manual Test	
✓ Tech doc	24/11			Tech doc	
✓ UI/UX Design	23/11			UI/UX Design	
+	Add task				
▼ Display register restau...	26/11			Display register restaurants	
✓ [UI] Display register...	25/11			[UI] Display registered restaurants	
✓ [API] Create endpoi...	26/11			[API] Create endpoint for registered restaurants	
✓ Manual Test	26/11			Manual Test	
✓ Tech doc	26/11			Tech doc	
✓ UI/UX Design	25/11			UI/UX Design	

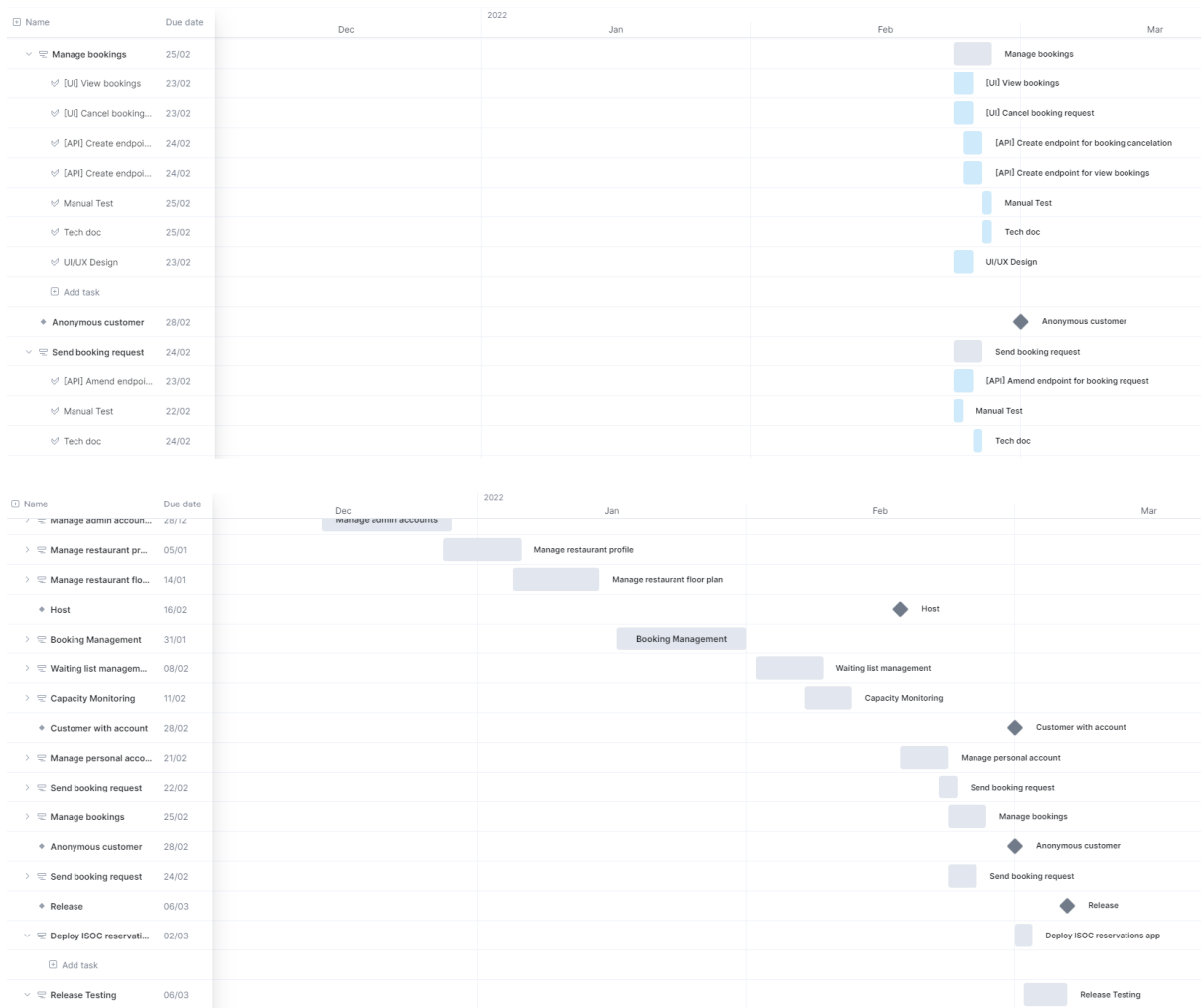
Name	Due date	Nov	Dec	2022	Jan
▼ Delete registered rest...	03/12		Delete registered restaurant		
✓ [UI] Delete registre...	26/11		[UI] Delete registered restaurant		
✓ [API] Create endpoi...	01/12		[API] Create endpoint for deleting registered restaurant		
✓ UI/UX Design	26/11		UI/UX Design		
✓ Manual Test	02/12		Manual Test		
✓ Tech doc	03/12		Tech doc		
Add task					
◆ Admin	14/01				Admin
▼ Manage host accounts	13/12		Manage host accounts		
✓ [API] Create endpoi...	10/12		[API] Create endpoint for CRUD operations on host accounts		
✓ [UI] Display host ac...	08/12		[UI] Display host accounts		
✓ [UI] Create host acc...	08/12		[UI] Create host account		
✓ [UI] Edit host account	08/12		[UI] Edit host account		
✓ [UI] Delete host acc...	08/12		[UI] Delete host account		
✓ Manual Test	13/12		Manual Test		
✓ Tech doc	10/12		Tech doc		
✓ UI/UX Design	08/12		UI/UX Design		

Name	Due date	Nov	Dec	2022	Jan
▼ Manage admin accoun...	28/12		Manage admin accounts		
✓ [API] Create endpoi...	28/12		[API] Create endpoint for CRUD operations on admin accounts		
✓ [UI] Display admin a...	21/12		[UI] Display admin accounts		
✓ [UI] Create admin a...	21/12		[UI] Create admin accounts		
✓ [UI] Edit admin acco...	21/12		[UI] Edit admin accounts		
✓ [UI] Delete admin a...	21/12		[UI] Delete admin accounts		
✓ Manual Test	28/12		Manual Test		
✓ Tech doc	28/12		Tech doc		
✓ UI/UX Design	21/12		UI/UX Design		
Add task					
▼ Manage restaurant pr...	05/01			Manage restaurant profile	
✓ [UI] Edit restaurant ...	03/01			[UI] Edit restaurant profile (details)	
✓ [API] Create endpoi...	04/01			[API] Create endpoint for editing restaurant profile (details)	
✓ Manual Test	05/01			Manual Test	
✓ Tech doc	05/01			Tech doc	
✓ UI/UX Design	03/01			UI/UX Design	

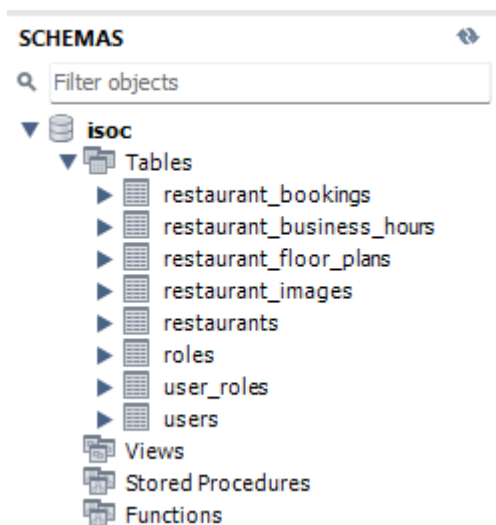
Name	Due date	Dec	2022	Jan	Feb
▼ Manage restaurant flo...	14/01			Manage restaurant floor plan	
✓ [UI] Create restaura...	11/01			[UI] Create restaurant floor plan	
✓ [UI] Edit restaurant ...	05/01			[UI] Edit restaurant floor plan	
✓ [UI] Delete restaura...	11/01			[UI] Delete restaurant floor plan	
✓ [API] Create endpoi...	13/01			[API] Create endpoints for restaurant floor plan	
✓ Manual Test	14/01			Manual Test	
✓ Tech doc	14/01			Tech doc	
✓ UI/UX Design	05/01			UI/UX Design	
□ Add task					
◆ Host	16/02				Host
▼ Booking Management	31/01			Booking Management	
✓ [UI] Create booking...	24/01			[UI] Create bookings page	
✓ [API] Create endpoi...	28/01			[API] Create endpoints for bookings CRUD	
✓ Manual Test	31/01			Manual Test	
✓ Tech doc	31/01			Tech doc	
✓ UI/UX Design	24/01			UI/UX Design	

Name	Due date	Dec	2022	Jan	Feb
▼ Waiting list managem...	08/02				Waiting list management
✓ [UI] Create waiting l...	07/02				[UI] Create waiting list page
✓ [API] Create endpoi...	07/02				[API] Create endpoints for waiting list CRUD
✓ [UI] Allocate table f...	07/02				[UI] Allocate table for waiting list item
✓ [API] Create endpoi...	07/02				[API] Create endpoint for allocating waiting list item
✓ Manual Test	08/02				Manual Test
✓ Tech doc	08/02				Tech doc
✓ UI/UX Design	07/02				UI/UX Design
□ Add task					
▼ Capacity Monitoring	11/02				Capacity Monitoring
✓ [UI] Display capaciti...	11/02				[UI] Display capacity monitoring
✓ [API] Create endpoi...	11/02				[API] Create endpoint for capacity monitoring
✓ Manual Test	07/02				Manual Test
✓ Tech doc	07/02				Tech doc
✓ UI/UX Design	11/02				UI/UX Design

Name	Due date	Dec	2022	Jan	Feb	Mar
◆ Customer with account	28/02					Customer with account
▼ Manage personal acco...	21/02				Manage personal account	
✓ [UI] Display persona...	18/02				[UI] Display personal account details	
✓ [API] Create endpoi...	18/02				[API] Create endpoint for editing personal account details	
✓ [UI] Delete personal...	18/02				[UI] Delete personal account	
✓ [API] Create endpoi...	18/02				[API] Create endpoint for deleting personal account	
✓ Manual Test	21/02				Manual Test	
✓ Tech doc	21/02				Tech doc	
✓ UI/UX Design	21/02				UI/UX Design	
□ Add task						
▼ Send booking request	22/02				Send booking request	
✓ [UI] Change restaur...	21/02				[UI] Change restaurants page and add book btn	
✓ [API] Create endpoi...	21/02				[API] Create endpoint for booking request	
✓ Manual Test	22/02				Manual Test	
✓ Tech doc	22/02				Tech doc	



Appendix B – ISOC Reservations Database



1 • SELECT * FROM isoc.users;

Result Grid							
Filter Rows:							
Edit: Export/Import: Wrap Cell Content: I A							
	id	email	name	password	username	restaurant_id	is_deleted
▶	74	IsocAdmin@gmail.com	Isoc Admin	\$2a\$10\$RvM8dmzRq5A.QQrknQkUeDHrLBqmJoUULU...	IsocAdmin@gmail.com	NULL	0
	75	paige_fox@gmail.com	Paige Fox	\$2a\$10\$uzl2g1EAYSUoFuogkefJuhRwWBkhAW5IPVp...	paige_fox@gmail.com	27	0
	76	alfie_medina@gmail.com	Alfie Medina	\$2a\$10\$Hl4F3ovtVT.8vp3CnhM9f.w/Q5aieqlkGFIK1...	alfie_medina@gmail.com	28	0
	77	otto_medeira@gmail.com	Otto Medeira	\$2a\$10\$BOLgMQhDGNbPpv0kE10hMONZYZVE.4KB1b...	otto_medeira@gmail.com	29	0
	78	austin_price@gmail.com	Austin Price	\$2a\$10\$No7theQzyeR9yjCGYeSP1upQCaRdD18rVPg0...	austin_price@gmail.com	30	0
	80	alice_morris@gmail.com	Alice Morris	\$2a\$10\$VzAY4rANzpjehsTyTmxy1.52iarso5PN8ZtnLJs...	alice_morris@gmail.com	31	0
	81	fern_clark@gmail.com	Fern Clark	\$2a\$10\$QyNGc40325VLQTVLhzx8oOjqdHS.mrhbuSrs...	fern_clark@gmail.com	32	0
	82	ava_bailey@gmail.com	Ava Bailey	\$2a\$10\$L3bRscCbU18ue.bF1.Yfe81.OpC8aOf7TJUCA...	ava_bailey@gmail.com	27	0
	83	georgina_hill@gmail.com	Georgina Hill	\$2a\$10\$oaqCbU18ue.bF1.Yfe81.OpC8aOf7TJUCA...	georgina_hill@gmail.com	27	0
	84	cleo_barnes@gmail.com	Cleo Barnes	\$2a\$10\$gsK8miEdHO93jswN2GoA.dpTzEd8NbmqDh...	cleo_barnes@gmail.com	28	0
	85	cora_garcia@gmail.com	Cora Garcia	\$2a\$10\$fgoDUKbG14Dy8Y0CYmXUbOo.5D2pV5sNf.hj...	cora_garcia@gmail.com	28	0
	86	alicia_baker@gmail.com	Alicia Baker	\$2a\$10\$TMDVvw38ujtszaiRMRBq.FwTQ32pMjEnnHH0...	alicia_baker@gmail.com	29	0
	87	theo_young@gmail.com	Theo Young	\$2a\$10\$4d/YmBnQADWvVpGQwcmusO8GTnCLDHmSZ...	theo_young@gmail.com	29	0
	88	brian_wallace@gmail.com	Brian Wallace	\$2a\$10\$6LbsnXponTvbDwh.YywhN.prn4TL6hASHLUu1...	brian_wallace@gmail.com	30	0
	89	leo_spencer@gmail.com	Leo Spencer	\$2a\$10\$RS.GjrgJ.LDJURdLaAg/Obz7kAWDEFWfzVZU...	leo_spencer@gmail.com	30	0
	90	eric_willis@gmail.com	Eric Willis	\$2a\$10\$FAHS2DZQf2gr02wmS8o0j.0JfyudmxE0rG47...	eric_willis@gmail.com	31	0
	91	carmen_ross@gmail.com	Carmen Ross	\$2a\$10\$wpAWmy3fMmkLBGybA3w8Lu0BE42nFwhzSS...	carmen_ross@gmail.com	31	0
	92	sebastian_wood@gmail....	Sebastian W...	\$2a\$10\$YgRYIkG78rNyYdeKrA6pu7azBqBjNd6MwJ...	sebastian_wood@gmail....	32	0
	93	ivan_allen@gmail.com	Ivan Allen	\$2a\$10\$Sp5VBuv2cNdAw9yNaKQ0q1.6ep.gQsGus/Ity...	ivan_allen@gmail.com	32	0
	94	thomas_richard@gmail.c...	Thomas Rich...	\$2a\$10\$HrMinkEYqxN NabzczPH7ueOrQXSZhf3wXdVi...	thomas_richard@gmail.c...	NULL	0
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Appendix C – Frontend component project structure

```

  UI
  ├── .angular
  ├── dist
  ├── node_modules
  └── src
      ├── app
      │   ├── modules
      │   │   ├── admin
      │   │   ├── auth
      │   │   ├── azure-map
      │   │   ├── customer-bookings
      │   │   ├── edit-account
      │   │   ├── edit-booking
      │   │   ├── host
      │   │   ├── isocadmin
      │   │   ├── landing
      │   │   ├── models
      │   │   └── services
      │   ├── app.component.html
      │   ├── app.component.scss
      │   ├── app.component.ts
      │   ├── app.module.ts
      │   ├── app.routing.ts
      │   ├── assets
      │   ├── environments
      │   ├── styles
      │   ├── _redirects
      │   ├── global.d.ts
      │   ├── index.html
      │   ├── main.ts
      │   ├── polyfills.ts
      │   ├── test.ts
      │   ├── .browserslistrc
      │   ├── .editorconfig
      │   ├── .eslintrc.json
      │   ├── .gitignore
      │   ├── .npmrc
      │   ├── .nvmrc
      │   ├── angular.json
      │   ├── karma.conf.js
      │   ├── package-lock.json
      │   ├── package.json
      │   ├── README.md
      │   ├── tsconfig.app.json
      │   ├── tsconfig.json
      │   └── tsconfig.spec.json
      └── ...
  
```

Appendix D – Backend component project structure

