Project Documentation

ISTE-330 Database Connectivity and Access

Hotel Reservations

Document (filename): HotelReservations.docx

Author(s): Kristijan Ninčević, Noa Mišić, Josef Arieatas, Josip Pavlinić

Date (last change): 2024-04-18

Version (last version):1.6

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Description of Change* | *Author(s)* | *Date* |
| 1.0 | Adding Overview and Purpose and Scope | Josip Pavlinić | 2024-03-24 |
| 1.1 | Adding functional and non-functional requirements | Noa Mišić | 2024-03-24 |
| 1.2 | Background, References, Document Overview | Kristijan Nincevic | 2024-03-24 |
| 1.3 | Problem Description and Solution Architecture | Josef Arieatas | 2024-03-24 |
| 1.4 | Schema and data dictionary | Josef Arieatas | 2024-03-28 |
| 1.5 | Expand Problem Description and Solution Architecture, User Documentation, Installation | Noa Misic | 2024-04-05 |
| 1.6 | Installation, Configuration and Acceptance Testing, Final Remarks and Conclusion, Areas of Particular Concern | Josip Pavlinic, Kristijan Nincevic | 2024-04-18 |

Table of contents

[1 Introduction 3](#_Toc163414264)

[1.1 Overview 3](#_Toc163414265)

[1.2 Purpose and Scope 3](#_Toc163414266)

[1.3 Background 3](#_Toc163414267)

[1.4 References 3](#_Toc163414268)

[1.5 Document Overview 4](#_Toc163414269)

[2 Problem Description and Solution Architecture 5](#_Toc163414270)

[2.1 Problem Description 5](#_Toc163414271)

[2.2 Technologies and Architectural Design 5](#_Toc163414272)

[2.3 Database Layer and Database Connectivity Layer 7](#_Toc163414273)

[2.4 Business Layer 12](#_Toc163414274)

[2.5 Presentation Layer 13](#_Toc163414275)

[2.6 Areas of particular concern 14](#_Toc163414276)

[3 Requirements 15](#_Toc163414277)

[3.1 Context 15](#_Toc163414278)

[3.2 Functional Requirements 15](#_Toc163414279)

[3.3 Other (Non-Functional) Requirements 16](#_Toc163414280)

[4 User Documentation 17](#_Toc163414281)

[4.1 Introduction 17](#_Toc163414282)

[4.2 Initial setup 17](#_Toc163414283)

[4.3 Navigating the Application 17](#_Toc163414284)

[4.4 Graphical User Interface Design 18](#_Toc163414285)

[4.5 User Manual 20](#_Toc163414286)

[4.6 Introduction 20](#_Toc163414287)

[5 Installation, Configuration and Acceptance Testing 21](#_Toc163414288)

[5.1 Installation 21](#_Toc163414289)

[5.1.1 Hardware requirements: 21](#_Toc163414290)

[5.1.2 Installation Process 21](#_Toc163414291)

[5.2 Configuration 21](#_Toc163414292)

[5.3 Acceptance Testing 21](#_Toc163414293)

[6 Final Remarks and Conclusion 22](#_Toc163414294)

# Introduction

## Overview

The Hotel Reservation application is a direct competitor to Booking.com Our idea is to make the best online hotel reservation platform that will enhance the user experience of reserving a hotel room, as well as enhance a hotels online presence. A user will be able to log in to his account, find a hotel based on some filter criteria, sort the hotels based on some criterias like price or rating, and reserve a hotel. Hotel owners will be able to post their hotel on the platform with information regarding their hotel and rooms. Users will be able to see their reservations, and hotel owners will be able to see how many reservations their hotel has.

## Purpose and Scope

The direct purpose of this project is to enhance online hotel reservation experience. Intended clients will be both hotel owners to list their hotels, and tourists that are looking for a hotel at some destination. The goal is to offer a seamless experience in reserving a hotel room. The document is intended for developers of the project, as it contains technical information about the project. It will contain project description, technologies used, layer of development and functional requirements for the project. The document is not intended for users of the project, as users should not be able to see the technologies used, and architectural design. The document will also contain user manuals, requirements for installation, and user guides but the purpose of that is so that the developers can get a better understanding of what each part should accomplish, and how.

## Background

The Hotel Reservations – Project Documentation is produced by highly sophisticated team members who include: Noa Misic, Josip Pavlinic, Kristijan Nincevic & Josef Arieatas. Our team wanted to break the monopoly booking.com and other several platforms have created for hotel reservations by creating a better, cheaper, and more user-friendly platform. Our idea is to innovate the way bookings are handled and implement new technologies by incorporating Artificial Intelligence. By combining Artificial Intelligence with our platform we reduce the cost of human resources needed for maintaining the platform which directly cuts our costs and allows us to be affordable to users. We want our product to be recognized globally as the number 1 Commercial Accommodation leader.

## References

The references used to create our platform were mainly based on Java. Java was the most optimal language to use to achieve our goal.

Java Development Kit (JDK): *Offers the best tools and with its built-in libraries it serves as the best option for development.* Official Java Documentation (https://docs.oracle.com/javase/8/docs/)

MySQL Database Management: *Chosen because of its scalability and reliability. That is why it serves as our database backbone*. MySQL Official Documentation (<https://dev.mysql.com/doc/>)

User Interface Development: *We chose JavaFX because it has just too many features to ignore. It is the best option for creating a good visual user interface.* JavaFX Documentation (<https://openjfx.io/javadoc/11/>)

MySQL Connector/J (JDBC Driver for MySQL): Serves as our connector to ensure Java and MySQL can interact with each other. MySQL Connector/J Documentation (https://dev.mysql.com/downloads/connector/j/)

## Document Overview

**Chapter 1**: Chapter 1 captures the basic overview of who we are and what we deliver. In the introduction we give a brief overview of how the our application works. In the scope we explain our vision and final goal. In the background we provide a story about who we are and what we want to achieve.

**Chapter 2**: Problem Description and Solution Architecture focus on the problems the project will fix and propose the solutions in the architecture. As part of this process, a we will provide a detailed examination of the technologies used for the project together and the reasons behind their usage.

**Chapter 3**: The Requirements section sets out both functional and non-functional requisites of the application which also includes clear enough and valid framework that highlights priorities as well as the scope of the project.

**Chapter 4**: User Documentation is responsible for guiding the user through the whole process of the application. This ensures that the user has a positive experience while interacting and learning about the application. The user documentation also includes detailed user interface designs as well as a comprehensive user manual.

**Chapter 5**: Installation, Configuration, and Acceptance Testing provides all the technical instructions needed in order to install and configure the application. We also provide acceptance testing norms to make sure the application meets all specified requirements.

**Chapter 6**: Final Remarks and Conclusion looks back at the journey and process of the application. Reflecting on the positive and negative challenges we had to overcome through the journey, while also searching for potential improvement for future development.

# Problem Description and Solution Architecture

## Problem Description

While the focus of this project is to compete directly with Booking.com, most existing hotel reservation platforms have similar issues. From clunky, cluttered user interfaces which are a chore to navigate, to bombarding users with seemingly useless, unrelated information. All of these issues combine to give the users a needlessly stressful experience when simply trying to get a reservation, leaving them anxious in the end, wondering if something was done wrong. Our solution seeks to redefine this process by focusing on a logical separation of concerns. With this, we plan on only showing the users exactly what they are looking for in a flow where they can easily navigate back and make any changes desired.

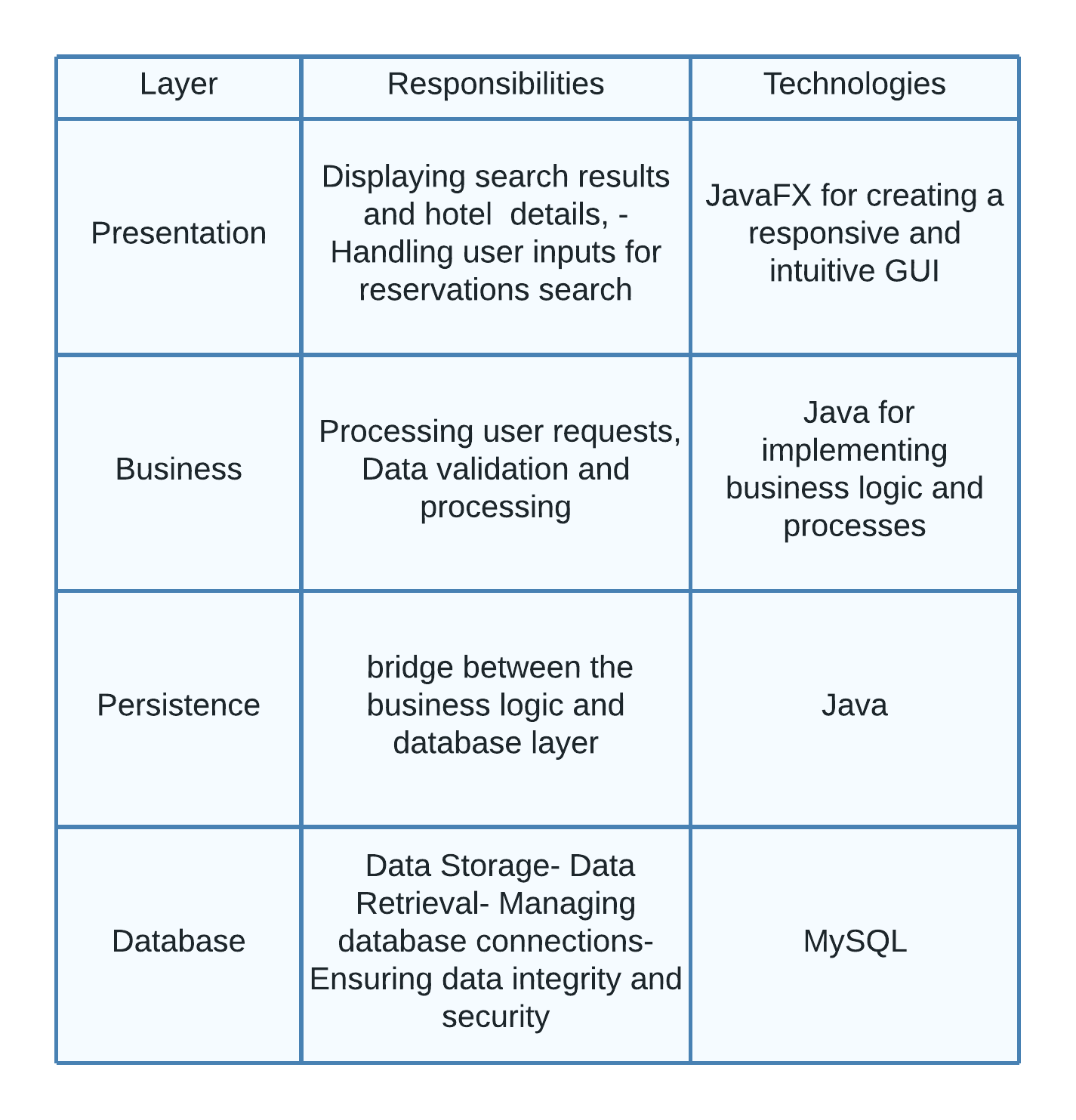
## Technologies and Architectural Design

**Presentation Layer:** Being the part most interacted with by the user, we chose to use JavaFX as it allows us to provide the user with a clean interface and seamless user experience. This has the ability to dynamically update based on user interaction with backend processes.

**Business Layer:** Core logic center of the application, this is where all processing related to business rules, user requests, and data manipulation takes place. Using Java’s refined features, we can achieve complex logical operations, maintain a high security status and easily integrate with lower levels.

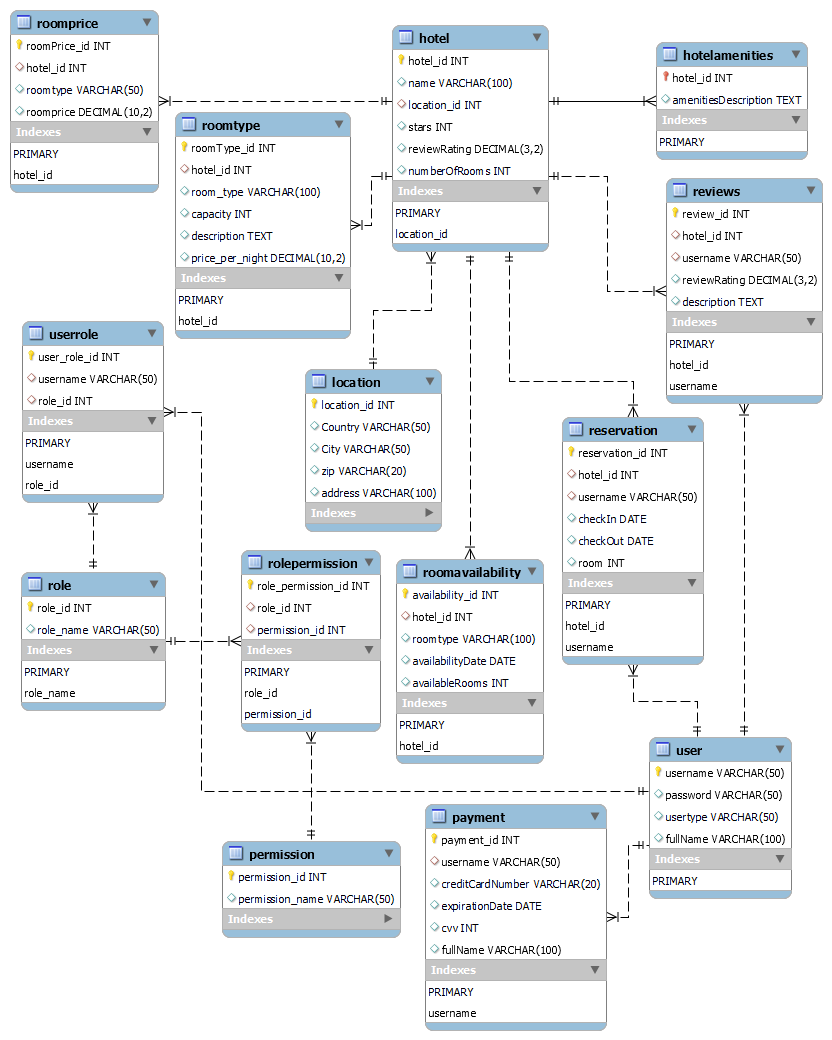
**Persistence Layer**: This layer acts as a bridge between the Business and Database layers, ensuring a smooth transition of business objects to database entities and vice versa.

**Database Layer:** This is the foundation of all information for the application, handling data storage, retrieval and management. Here we decided to make use of MYSQL for its industry famous reliability, scalability and close compatibility with Java-based applications.



## Database Layer and Database Connectivity Layer

**ERD / database schema**

**

**Data dictionary**

**User Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| username | VARCHAR | 50 | PRIMARY KEY | Unique identifier for a user account. |
| password | VARCHAR | 50 |  | Encrypted password for the user account. |
| usertype | VARCHAR | 50 |  | Type of user, e.g., admin, guest, staff. |
| fullName | VARCHAR | 100 |  | Full name of the user. |

**Role Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| role\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each role. Increments automatically with each new entry. |
| role\_name | VARCHAR | 50 | UNIQUE | The name of the role, such as 'Admin', 'User', 'Guest', etc. This field is unique, ensuring no two roles have the same name. |

**UserRole Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| user\_role\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each user-role assignment. Increments automatically with each new entry. |
| username | VARCHAR | 50 | FOREIGN KEY | Username of the user. References the username in the User table, establishing a relationship between a user and their assigned roles. |
| role\_id | INT |  | FOREIGN KEY | Identifier for the role. References the role\_id in the Role table, linking the user to a specific role within the system. |

**Permission Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| permission\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each permission. Increments automatically with each new entry. |
| permission\_name | VARCHAR | 50 | UNIQUE | The name of the permission. This field is unique, ensuring no two permissions have the same name. |

**RolePermission Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| role\_permission\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each role-permission assignment. Increments automatically with each new entry. |
| role\_id | INT |  | FOREIGN KEY | The identifier for the role. This field references the role\_id in the Role table, establishing a link to the specific role being granted permissions. |
| permission\_id | INT |  | FOREIGN KEY | Identifier for the permission. This field references the permission\_id in the Permission table, linking the role to specific permissions within the system. |

**Location Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| location\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each location. Automatically incremented with each new entry. |
| Country | VARCHAR | 50 |  | The country where the location is situated. |
| City | VARCHAR | 50 |  | The city where the location is situated. |
| zip | VARCHAR | 20 |  | The ZIP/postal code for the location. |
| address | VARCHAR | 100 |  | The street address of the location. |

**Hotel Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| hotel\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | A unique identifier for each hotel record. Increments automatically with each new hotel entry. |
| name | VARCHAR | 100 |  | Name of the hotel. |
| location\_id | INT |  | FOREIGN KEY | A reference to the Location table, indicating where the hotel is situated. |
| stars | INT |  |  | The star rating of the hotel. (scale from 1 to 5) |
| reviewRating | DECIMAL | (3, 2) |  | The average review rating of the hotel. |
| numberOfRooms | INT |  |  | The total number of rooms available in the hotel. |

**RoomPrice Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| roomPrice\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each room price entry. Increments automatically with each new entry. |
| hotel\_id | INT |  | FOREIGN KEY | A reference to the Hotel table, linking the room price to a specific hotel. |
| roomtype | VARCHAR | 50 |  | The type of room (Standard, Deluxe, Suite, etc). |
| roomprice | DECIMAL | (10, 2) |  | The cost of the room per night. |

**Reviews Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| review\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each review entry. Increments automatically with each new entry. |
| hotel\_id | INT |  | FOREIGN KEY | A reference to the Hotel table, linking the review to a specific hotel. |
| username | VARCHAR | 50 | FOREIGN KEY | A reference to the User table, identifying the user who wrote the review. |
| reviewRating | DECIMAL | (3, 2) |  | The rating given by the user in the review. |
| description | TEXT |  |  | The full text content of the user's review. |

Payment Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| payment\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each payment transaction. Automatically increments with each new entry. |
| username | VARCHAR | 50 | FOREIGN KEY | Reference to the User table, identifying the user making the payment. |
| creditCardNumber | VARCHAR | 20 |  | The credit card number used for the payment. Stored securely. |
| expirationDate | DATE |  |  | Expiration date of the credit card. |
| cvv | INT |  |  | The card verification number/security code of the credit card. |
| fullName | VARCHAR | 100 |  | Full name of cardholder. |

**Reservation Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| reservation\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each reservation. Increments automatically with each new entry. |
| hotel\_id | INT |  | FOREIGN KEY | A reference to the Hotel table, linking the reservation to a specific hotel. |
| username | VARCHAR | 50 | FOREIGN KEY | A reference to the User table, identifying the user who made the reservation. |
| checkIn | DATE |  |  | The date when the guest will check into the hotel. |
| checkOut | DATE |  |  | The date when the guest will check out of the hotel. |
| room | INT |  |  | The room number assigned to the reservation/ guest. |

**HotelAmenities Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| hotel\_id | INT |  | PRIMARY KEY, FOREIGN KEY | Unique identifier that references the Hotel table, linking amenities to a specific hotel. This field acts as both a primary key and a foreign key. |
| amenitiesDescription | TEXT |  |  | A detailed description of the amenities offered by the hotel. |

**RoomAvailability Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| availability\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each room availability record. Increments automatically with each new entry. |
| hotel\_id | INT |  | FOREIGN KEY | A reference to the Hotel table, linking the room availability to a specific hotel. |
| roomtype | VARCHAR | 100 |  | The type of room (e.g., Standard, Deluxe, Suite) for which availability is being recorded. |
| availabilityDate | DATE |  |  | The date(s) for which the room is available. |
| availableRooms | INT |  |  | The number of rooms of the specified type that are available for booking on the availability date. |

**RoomType Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Length** | **Constraint** | **Description** |
| roomType\_id | INT |  | AUTO\_INCREMENT, PRIMARY KEY | Unique identifier for each room type entry. It increments automatically with each new entry. |
| hotel\_id | INT |  | FOREIGN KEY | A reference to the Hotel table, linking the room type to a specific hotel. |
| room\_type | VARCHAR | 100 |  | The name or category of the room type (e.g., Standard, Deluxe, Suite). |
| capacity | INT |  |  | The maximum number of guests that can be accommodated in the specific type of room. |
| description | TEXT |  |  | A detailed description of the room type, including size, amenities, view, etc. |
| price\_per\_night | DECIMAL | (10, 2) |  | The cost of staying in this type of room for one night. |

## Business Layer

The Business Layer of our Hotel Reservation application is designed to implement the application's core logic and rules. It acts as the brain of the application, processing user requests, executing business operations, and making logical decisions based on the data received from the Database Connectivity Layer. This layer is responsible for tasks such as validating user inputs, managing reservations, calculating prices, and handling user authentication and authorization.

HotelManager: Manages hotel-related operations including adding hotels, updating hotel details, and retrieving hotel amenities.

Payment: Handles the secure processing and storage of payment information, including the encryption of credit card numbers, expiration dates, and CVV codes.

Permission: Facilitates the management of user access control by representing and handling permissions in the application. This class provides the functionality to retrieve a list of permissions from the database.

Reservation: Manages the booking process within the hotel reservation system, allowing users to make, view, and cancel reservations. Supporting the creation of reservation records in the database with details such as hotel ID, username, check-in and check-out dates, and room number.

Review: Handles user reviews for hotels, allowing users to leave ratings and comments about their stays. This class adds new reviews to a hotel and shows all previous reviews the hotel has received.

Role: Manages user roles within the system, allowing for the definition and assignment of access levels. Allows the addition of new roles to the database, supporting flexible and dynamic access control tailored to various user responsibilities and permissions.

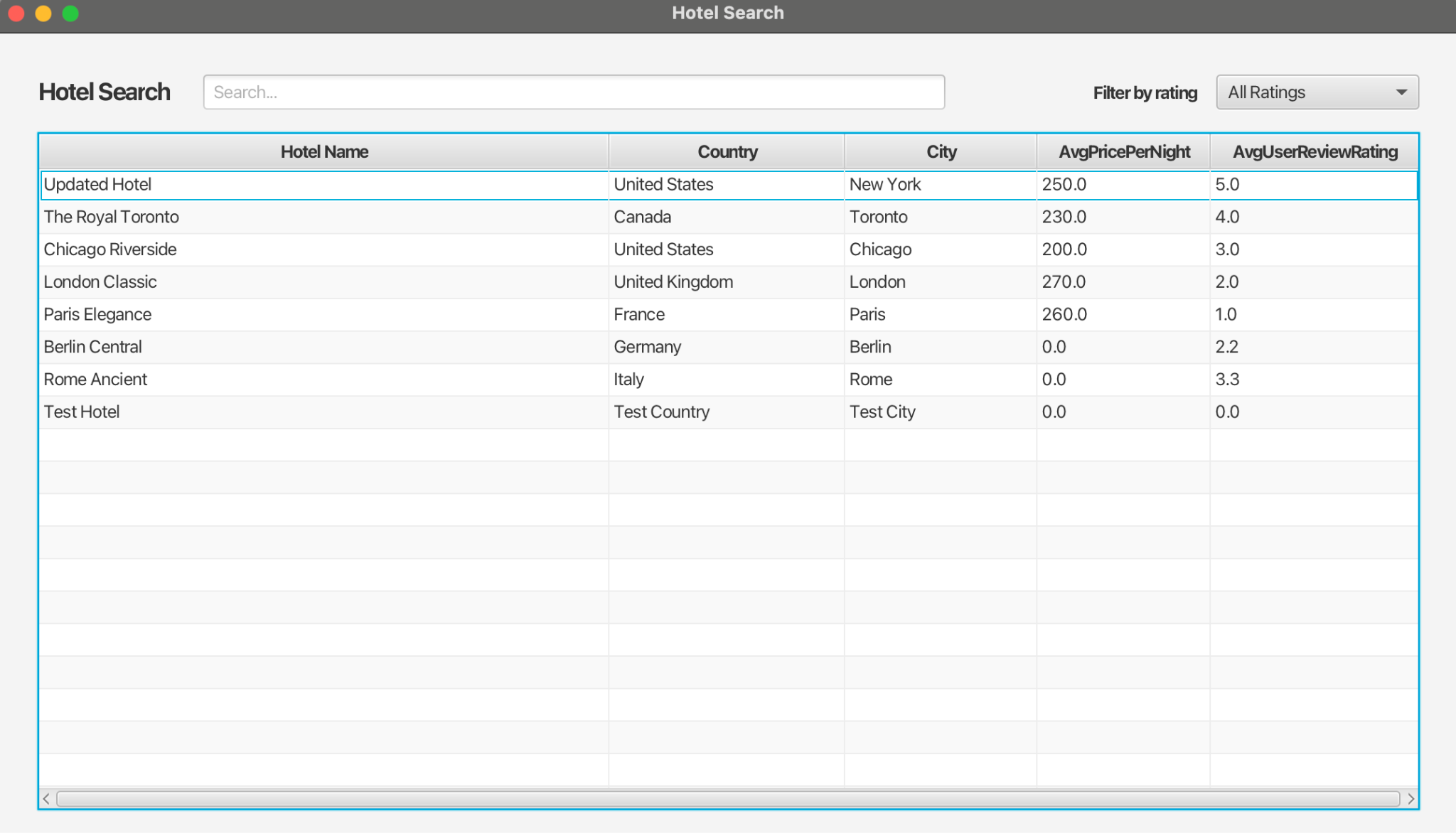
RolePermission: Manages the mapping between roles and permissions in the application, facilitating the implementation of access controls based on user roles.

User: Manages user accounts, including registration, login, and maintaining user profiles. It securely handles user credentials through password hashing, supports the addition of new users to the system, and verifies user logins by matching hashed passwords.

UserRole: Manages the assignment of roles to users within the system, ensuring that each user is granted the correct permissions according to their role.

## Presentation Layer

The Presentation Layer of our Hotel Reservation application is designed to offer a user-friendly, intuitive, and visually appealing interface that will not confuse the user. It provides the users with interactive elements to navigate through the application, search for hotels, make reservations, view their booking history, and leave reviews. This layer separates the complex underlying business logic and presents information in an accessible and easily understandable format.

HotelSearch example:  


## Areas of particular concern

**Areas for concern:** Processing and storing user data and payments methods needs to be done in a secure way so that sensitive user information cannot become compromised.

As the application grows in user base and data volume, there might be scalability issues affecting performance and user experience.

Scalability and Performance:

**Risk**: Increased user load may overwhelm the initial architecture, causing performance bottlenecks.

**Mitigation**: Perform load and stress testing to optimize performance. Use scalable cloud infrastructure to handle growth efficiently.

**Security and Data Privacy:**

**Risk**: Sensitive data handling necessitates stringent security to prevent breaches and comply with regulations like GDPR.

**Mitigation**: Implement robust encryption, regular security updates, and compliance audits.

**Technology Dependencies:**

**Risk**: Dependence on specific technologies like JavaFX and MySQL could limit flexibility.

**Mitigation**: Maintain a modular architecture to facilitate updates and integrations without extensive overhauls.

**Integration Complexities:**

**Risk**: Integration with external APIs (e.g., payment gateways) can introduce failures.

**Mitigation**: Utilize reliable APIs and establish comprehensive error handling processes.

**Regulatory Compliance:**

**Risk**: Non-compliance with laws for online transactions can result in penalties.

**Mitigation**: Regularly update compliance protocols and engage with legal experts to verify software adherence to legal standards.

**Human Resources:**

**Risk**: Project success is highly dependent on the development team's stability and expertise.

**Mitigation**: Provide continuous training and maintain clear communication paths within the team.

# Requirements

## Context

Hotel Reservation is an application that will allow users to search for hotels or list their own. In case the user wants to search for a hotel they can select some parameters to filter their search. For example, Destination (town and country where they are going), date of Check-in, Check out, Number of People in the Room, Price of the room per night, and Rating of the Hotel. The more parameters the user chooses the more specific search they will get. In case the user wants to list their hotel, they will need to provide information about the hotel. The pieces of information are: Location, Number of Rooms Total, Number of Rooms for a certain amount of people (for example 50 rooms for one person, 20 rooms for two people…) Hotel Rating, and Price of a certain room per night. The user needs to have a specific account for a hotel listing. Those types of accounts will need to pay a monthly fee, while “regular” user accounts (which just want to view hotels) are free.

## Functional Requirements

1. User Registration – The user needs to register before using the application. User will specify their first name, last name, email, username, password, and type of account they want.

2. Log in – If the user registered for an account they can log in to use the application.

3. Hotel Search – Function that will allow users to search for the hotels.

4. Hotel Filtering – Function that will allow users to filter parameters to get specific hotels they had in mind.

5. Booking – If users like the hotel and want to book it they can do it. This will remove any number of rooms they booked from the available rooms.

6. Payment – After the user books the hotel room/s they will need to enter credit/debit card information to pay for the room.

7. Cancellation – Option to cancel hotel reservation if the user has changed their mind.

8. Hotel listing – for the users that have listing accounts. This will allow those users to list their hotels. This function will also place the hotel in the list with all other available hotels for the search.

9. Modify listing – The user can modify some information about the hotel (for example, if the price of the hotel has gone up user can change that price and it will apply from that moment onwards.

10. User Review and Rating View – The users can view reviews and ratings of other users for a certain hotel.

11. User Review and Rating Enter – The user can leave a Review and Rating for a certain hotel.

12. Top amenities – The top amenities of the hotel will be listed if the user clicks on the hotel.

13. Sort Hotel Search – The user can sort all the hotels the search finds by different parameters. Those parameters are Price and Rating

14. Reservation Check – The user can check the reservation/s they have made.

15. User Profile Management – User can change their profile information.

## Other (Non-Functional) Requirements

Localization - The whole application for be written in English for now. If the app gains many users and the need for more languages shows we will add them.

Performance – When a user is searching for the hotels the more hotels are available for their search the longer will take to list them all. However, even if there are many hotels, we will list them the quickly as we can.

Usability – We will have a navigation bar where users can clearly choose what they need. If the user has a listing account they will be able to access more functions (as they need to be able to add the hotels). The app will be very simple and easy to use without much complications. When a user searches for hotels and they are listed if the user clicks on any hotel it will open. Also, there will be a back button that allows the user to go back whenever they want.

Reliability – If the system fails we will fix it in under 15 minutes. Also, there will always be a backup if something goes wrong with the database.

Security – User sensible information such as password will be stored in hash and sent to the server. Whenever the user wants to log in their password will again be stored in hash and compared to the hash on the server. If those two hashes are the same user will be allowed to enter the app. Also, Credit/Debit card information will be encrypted before it is sent to the server. All other sensitive information will also be encrypted.

Scalability – The application will be made for a low amount of people at the start. When the application scales it will be adjusted to be able to hold more users at the same time.

Backup and Recovery – anything goes wrong there will be a full backup done every week. Also, there will be Differential backup every day.

# User Documentation

## Introduction

This portion of the documentation is designed and intended to provide the end-user with a detailed guide on how to navigate and use this application efficiently and effectively. The purpose of this documentation is to address any possible confusion and be a point of reference for the end-user if they seek any further information/ clarification.

## Initial setup

After launching the application, the user will be prompted to either register or sign in to their existing account. After this stage, the user will be able to navigate the main part of the application and chose what they intend to interact with

## Navigating the Application

After logging into the application the user will have access to the following screens:  
**Dashboard**(main page): This will display what the user can immediately access after logging in. It includes a display of their profile and username.

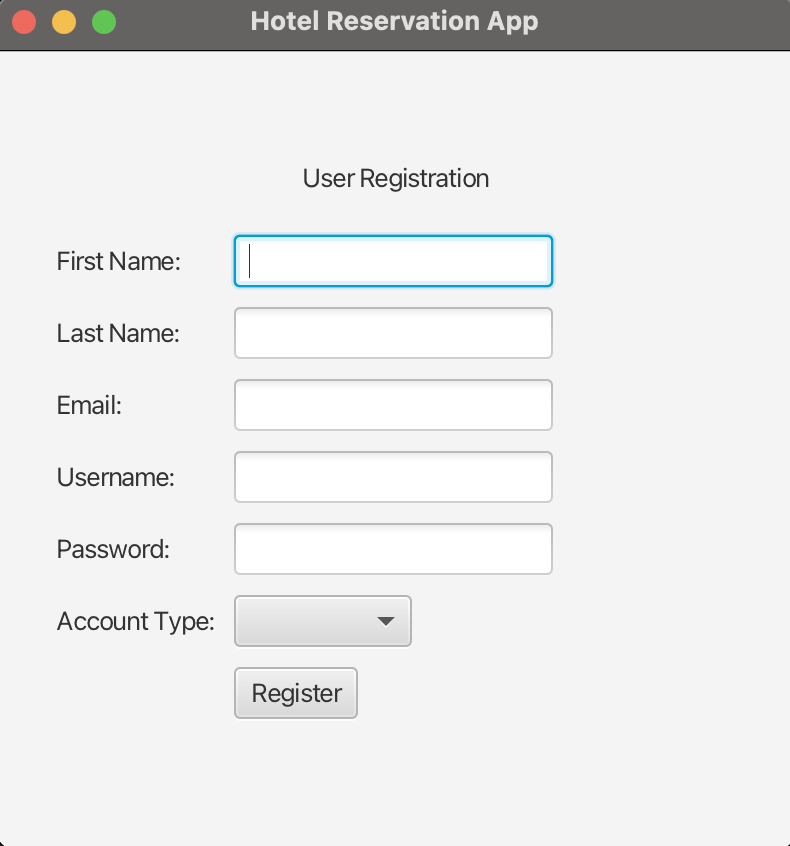
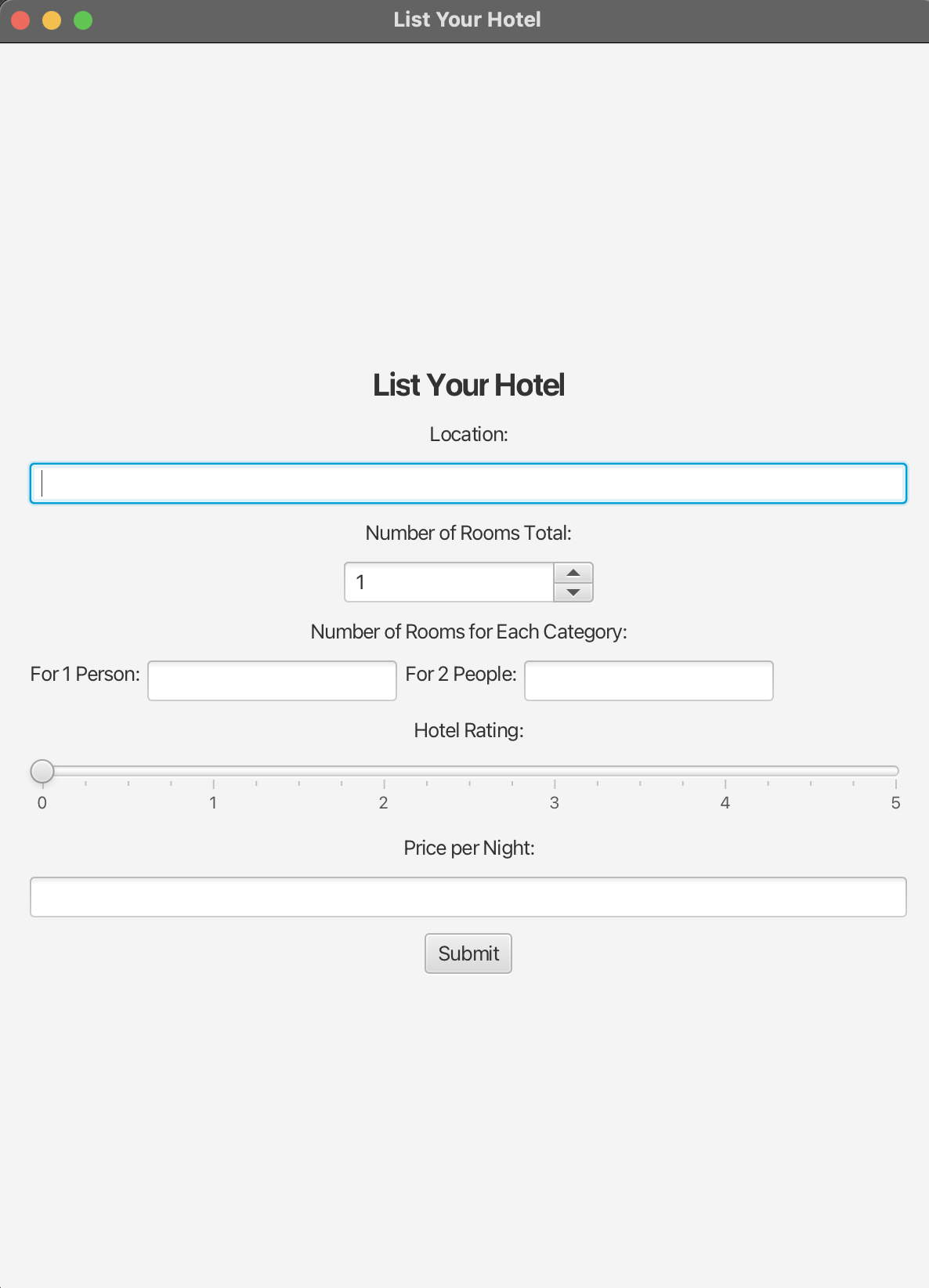
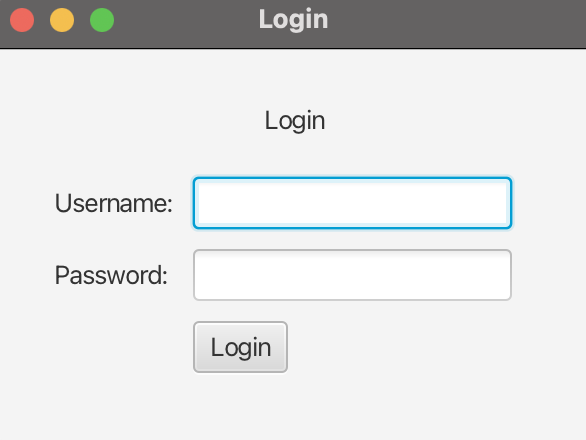
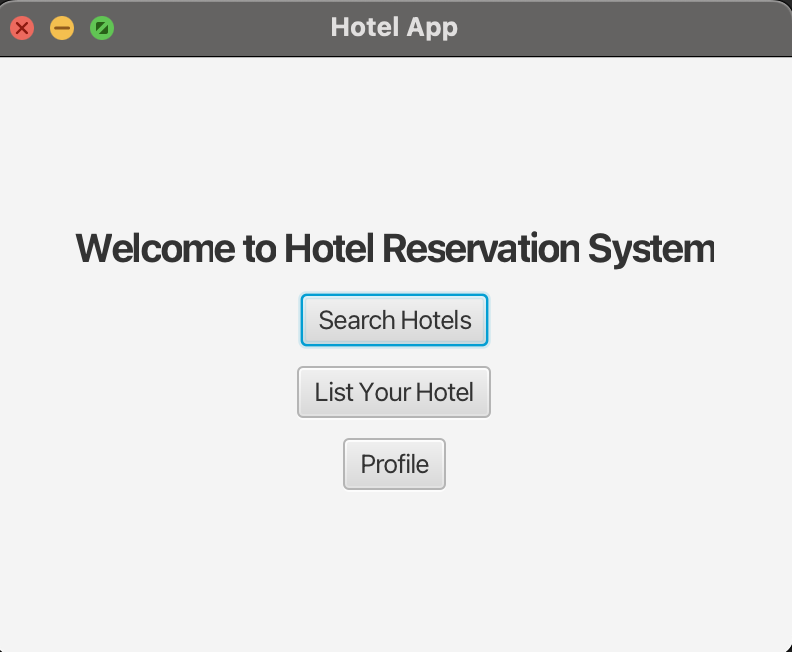
**Hotel Search**: Here the user will be able to search a list of existing hotels, and view information pertaining to each.

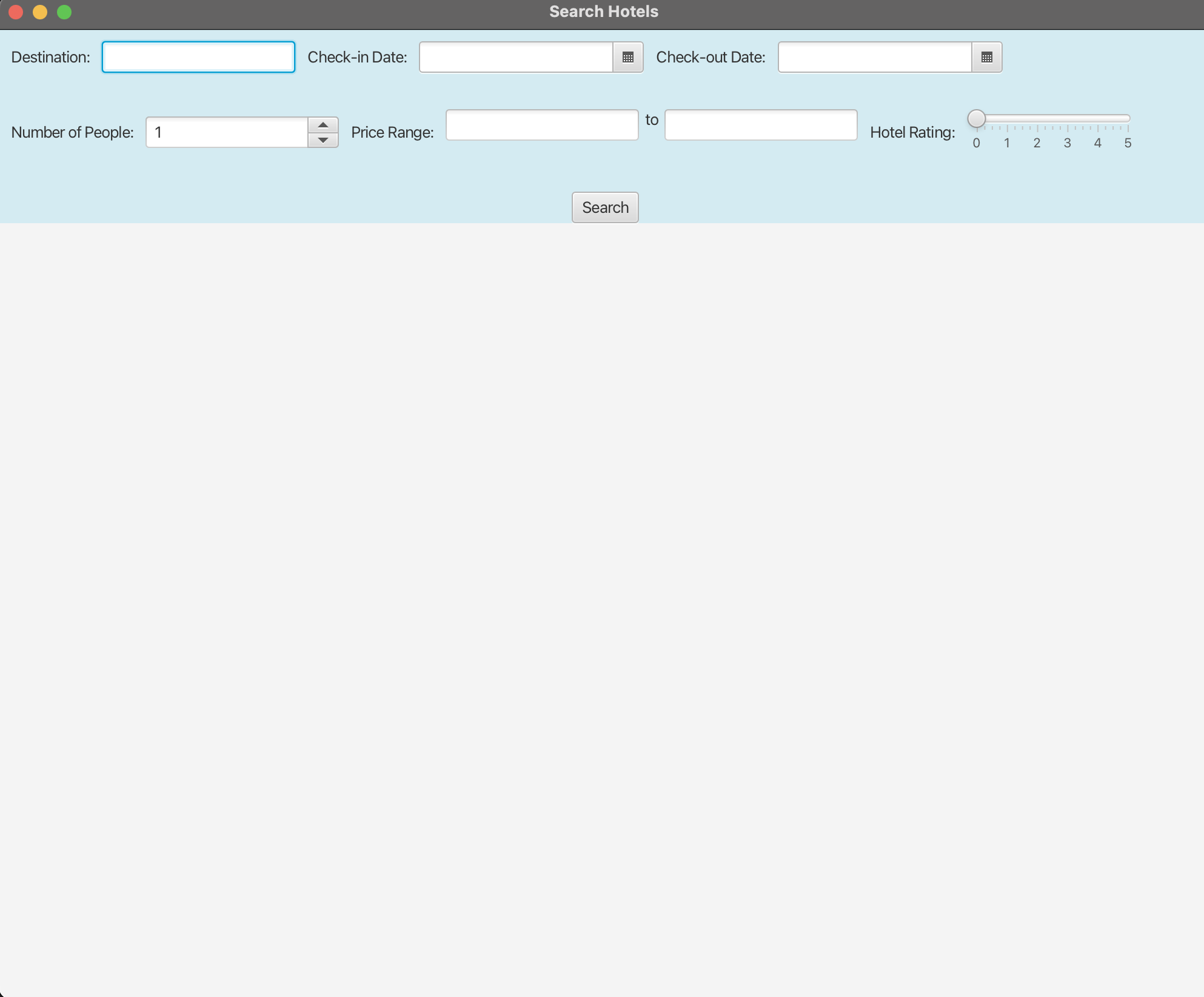
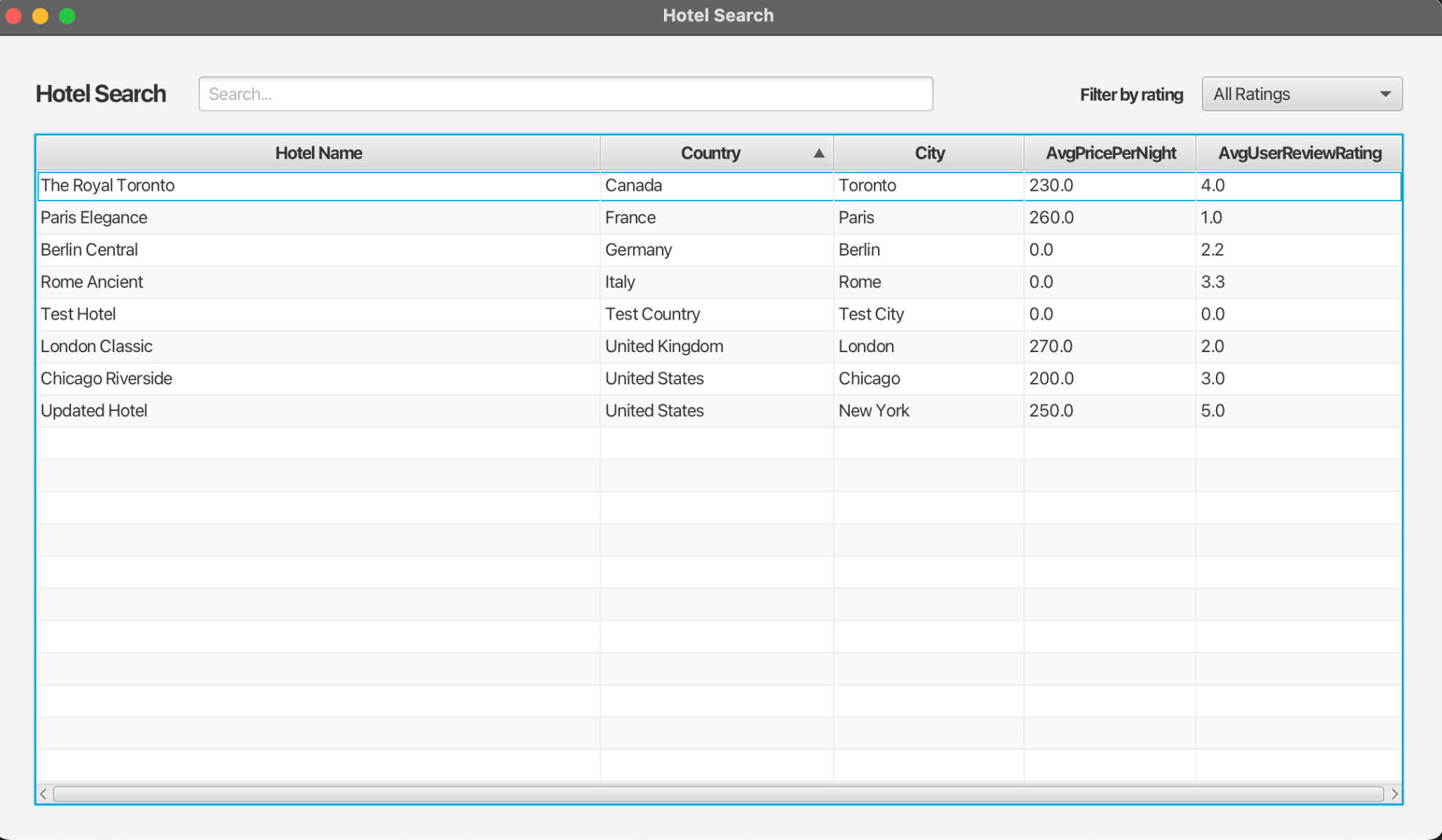
**Reservations**: Here the user will be able to select a hotel in which they are interested and proceed with creating a reservation.

**Account Management**: How to update user profile details, manage preferences, and view past activity.

## Graphical User Interface Design

This is still being prototyped, not the final result.





## User Manual

## Introduction

Welcome to the User Manual for Hotel Reservation Application your comprehensive guide to booking and managing hotel reservations with ease. This manual provides instructions and tips to help you make the most of the application's features.

Table of Contents

1. Getting Started

* Installation
* Creating an Account

2. General Navigation

3. User Roles

* Guests
* Hotel Managers
* Administrators

4. Making a Reservation (For Guests)

5. Managing Hotel Listings (For Hotel Managers)

6. Admin Dashboard (For Administrators)

7. FAQs

8. Troubleshooting

9. Contact Support

# Installation, Configuration and Acceptance Testing

## Installation

**Operating system**: Windows Vista or above, macOS Catalina or newer, Ubuntu 5.10 or above.

### Hardware requirements:

CPU: Minimum 2 GHz processor.

Memory: Minimum 2 GB RAM.

Disk Space: Minimum 1 GB of free disk space.

**Software Dependencies:**

Java Runtime Environment 17 or newer.

Database Server MySQL 8.0

JavaFX 17 or newer.

**Permission:** Administrative or root access on the machine where the installation will occur.

### Installation Process

Step 1: Download the Application

Step 2: Initialize dependencies. (JavaFX and Database Driver)

Step 3: in databaseInfo.txt change the data for connection with the db

Step 4: Run the script from the hotelDatabase.sql - that creates the databse and table with the admin user (login info are admin, admin)

Step 5: Run the application

Step 6: Log in with admin or register

Step 7: Admin can both list and search for hotels, Listing user can only list hotels(and other actions like edit listing...), and Regular user can only search hotels(and do other actions like book...)

## Configuration

**Database Configuration:**

URL: jdbc:mysql://localhost:3306/hotel

Username: root (Change as required)

Password: ritcroatia (Change as required)

These settings can be adjusted in the databaseInfo.txt or directly within the application's configuration module.

**Security Settings:**

Configure the SSL in order to achieve secure data transmission if the application will be used on the internet.

Set up firewalls and restrict database access to only the application servers.

**Logging:**

By setting the logging properties in the logging.properties file, you can enable application logging. Application operation logs, access logs, and error logs should all be included in this.

**JavaFX Configuration:**

To fix any JavaFX-related runtime errors, make sure the JavaFX lib directory is included in the PATH variable.

## Acceptance Testing

**User Registration and Login:**

Test the registration process with both valid and invalid data inputs.

Ensure the login process authenticates existing users correctly and restricts access upon failure.

**Hotel Search and Reservation:**

Verify the search functionality returns the correct results based on various filters (e.g., location, price, ratings).

Check the reservation process, including constraints like attempting to book more rooms than are available.

**Data Management:**

Confirm that new hotels can be added, and existing listings can be updated or deleted.

Test data backup and recovery procedures to ensure data integrity is maintained.

**Navigation and User Interface:**

Test the navigational flow of the application, ensuring that users can easily move from one section to another without issues.

Validate the responsiveness and aesthetic of the graphical user interface on different devices and resolutions.

# Final Remarks and Conclusion

6.1 Summary of Experiences

As we come to the end of our project documentation for our Hotel Reservation application, we will be recollecting the challenges and the rewarding experiences we have gone through this rollercoaster of a journey!

Positive Experiences:

Collaboration and Teamwork: The team overcame obstacles together with their collaborative skills and determination. Effective communication was the key to bringing the application to life!

Learning Curve: It was a challenge for the team to work with new tools such as JavaFX which is responsible for the presentation layer and incorporating MySQL database management into the project itself. The integration of the two technologies allowed our application to work effortlessly.

User-Centric Design: The team focused on providing the users with a clean and intuitive interface that has been given positive feedback in user acceptance testing.

Challenges Encountered:

Scalability Issues: The biggest challenge overall for the team was to scale the application for a larger number of users. The application encountered issues when trying to connect with 100 or over clients, but with determination, the team overcame this obstacle and delivered an application that overcomes all boundaries.

Password Hashing: The team struggled to successfully hash passwords for each user effectively, but in the end, the team figured out that they were trying to encrypt the passwords on multiple occasions during one session which led to complications during the process which sent the information to the database. The team debugged for hours and solved the issue.

6.2 Project Deliverables

Software Requirements Specification: Detailed functional and non-functional requirements.

Technical Documentation: Including installation guides, user manuals, and system architecture descriptions.

Source Code: Fully commented and version-controlled source code repository.

Testing Reports: Comprehensive testing results, including unit, integration, and acceptance testing.

Deployment and Maintenance Guides: Guidelines for deploying and maintaining the application.

6.3 Missing Functionalities and Future Work

Multi-language Support: Currently the application only uses English as its primary and only language. In the future, the team might add the functionality to provide users with more languages to interact with.

Mobile Compatibility: The application currently only works on a desktop. In the future, the team might consider developing a mobile application.

Advanced Payment Features: The team will consider implementing features such as payment via digital wallets and direct bank transfers.

6.4 Effort Estimation

The project involved approximately 100 person-hours, distributed as follows:

Planning and Design: 10 hours

Implementation: 30 hours

Testing and Debugging: 50 hours

Documentation: 10 hours

The team distributed the roles based on the person's knowledge and experience working with different technologies. By doing so the team ensured that all team members could contribute effectively:

Front-end Development: Focused on user interface design and interactions.

Back-end Development: Handled database interactions, business logic, and security measures.

Quality Assurance: Ensured the application met all requirements through thorough testing and feedback implementation.

6.5 Conclusion

The Hotel Reservation application was a challenge at first and while the team is happy the goal came to an end, it was the journey that leaves the best of memories. The Hotel Reservation Application was made to revolutionize hotel reservations which the team successfully delivered. The experiences gained from the development of the application will be held as a core for future team projects. The team will focus on providing future improvements to the application. We believe that continuous improvement and adaptation to emerging technologies will be key to maintaining the relevance and efficiency of the Hotel Reservation application. We think that the Hotel Reservation application's continued relevance and effectiveness will depend on its ability to adapt to new technologies and undergo constant improvement.