Hotel Booking Management

By

Saira Mubeen	**2021-GCUF-058619**
Maria Rasheed	**2021-GCUF-058632**
Bisma Rehman	**2021-GCUF-058609**

Project submitted in partial fulfillment

of the requirements for the degree of

BACHELOR OF SCIENCE

IN

COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

Government College University Faisalabad

2021-2025


بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ
In the name of Allah, the most gracious, the most merciful

DECLARATION

This project, a die-hard work, is productized out by Saira Mubeen, Maria Rasheed and Bisma Rehman under the supervision of Mr. Yasir Arfat (HOD, Computer Department) and Mr. Yasir Arfat (Lecturer, Computer Department) Govt Graduate College Chowk Azam, GC University, Faisalabad, Pakistan. We feel please to declare that the project and contents of this project is the productive result of our hardworking, studies and research and no part of this is copied from any published source. This work has been conducted under the practical atmosphere of our studies not for the award of any other degree / diploma. The University may take action if the information provided is found guilty at any stage. Any external sources of information used in this project, including references, have been duly acknowledged through proper citations and bibliographical references. The project has not been previously submitted for any other degree or examination at any other institution. Any contributions made by others to this project, including guidance and support from faculty members, have been duly acknowledged. The software code, documentation, and any other materials presented as part of this project are the result of my own work, unless otherwise acknowledged. I take full responsibility for the authenticity and originality of the content presented in this project. I understand that any misrepresentation or falsification of information in this declaration will have serious consequences, including the possibility of disciplinary action by the institution.

Signature of Student:

Saira Mubeen _____

Registration No. 2021-GCUF-058619

Maria Rasheed_____

Registration No. 2021-GCUF-058632

Bisma Rehman _____

Registration No. 2021-GCUF-058609

Dedication

I dedicate this project to God Almighty, my Creator, my strong pillar, and my source of inspiration, wisdom, knowledge, and understanding. He has been the source of my strength throughout this program, and it is on His wings alone that I have soared. I also dedicate this work to my friends, who have encouraged me along the way and whose support ensured that I gave my all to complete what I started. This project, titled **"SelfOrder Hotel Booking Management,"** is dedicated to the individuals who have been a source of inspiration and support throughout this journey. To my family, for their unwavering love, encouragement, and understanding. Your belief in me has been my driving force, and I am deeply grateful for your constant support. To my project supervisor, **Mr. Yasir Arfat**, for his guidance, expertise, and valuable insights. His mentorship has been invaluable in shaping this project and my growth as a developer. To my friends and classmates, for their camaraderie, motivation, and the countless hours spent brainstorming ideas and discussing challenges. Your presence made this project more enjoyable and fulfilling. To the staff and faculty members at **Govt Graduate College Chowk Azam**, for providing a conducive learning environment and resources that contributed to my education and the successful completion of this project. To all the users and stakeholders who participated in the testing and evaluation of the app, providing valuable feedback and helping me improve its functionality and user experience. Lastly, I would like to express my gratitude to everyone who played a role, big or small, in this project. Your support and belief in my abilities have been instrumental in its completion. This project is dedicated to each and every one of you. Thank you for being a part of my journey and making it a truly enriching experience.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciation to all those who have contributed to the successful completion of my Final Year Project titled "SelfOrder Hotel Booking Website." This project has been an enriching and rewarding experience, and I would like to acknowledge the individuals and organizations for their support. My project supervisor, **Mr. Yasir Arfat**, for their invaluable guidance, expertise, and continuous support throughout the project. Their knowledge and insights have been instrumental in shaping the direction of this work. The faculty members of **Govt Graduate College Chowk Azam**, for providing me with a conducive learning environment and valuable resources. I am grateful for their dedication to education and their efforts in imparting knowledge and skills. My family and friends, for their unwavering support, encouragement, and understanding. Their belief in me and their motivation have been a constant source of inspiration, driving me to overcome challenges and strive for excellence. The participants who willingly volunteered their time and provided feedback during the testing and evaluation phases of the project. Their insights and suggestions have been invaluable in refining the functionality and user experience of the app. The developers and contributors of open-source libraries, frameworks, and tools that were utilized in the development of this project. Their efforts have significantly expedited the development process and enhanced the overall quality of the application. I would also like to extend my gratitude to all the users and restaurant owners who may benefit from this Hotel Booking website. Their needs and expectations have been a driving force behind the development of this solution. Lastly, I want to express my appreciation to all the individuals, both mentioned and unnamed, who have directly or indirectly contributed to this project. Your support and assistance have played an integral role in its successful completion. In conclusion, I am truly grateful to everyone who has been a part of this project. Your contributions, encouragement, and support have been invaluable, and I am honored to have had the opportunity to work on this project with such wonderful individuals.

Saira Mubeen **(Roll No. 058619)**
Maria Rasheed **(Roll No. 058632)**
Bisma Rehman **(Roll No. 058609)**

To

The Controller of Examinations,

Government Graduate College,

Chowk Azam

I, the Supervisor, certifies that the contents and form of the project submitted by

Saria Mubeen **(Roll No. 058619)**

Maria Rasheed **(Roll No. 058632)**

Bisma Rehman **(Roll No. 058609)**

Have been found satisfactory for the award of the degree.

Internal Examiner

Name:_____

Signature:_____

Lecturer: Govt Graduate College Chowk Azam

Department of Computer Science

External Examiner

Name:_____

Signature:_____

Table of Contents

List of Figures

Abstract:

The rise of online food delivery services has significantly transformed the way people access meals, offering convenience and efficiency that have become integral to modern living SelfOrder addresses the need for a streamlined Hotel Booking platform that benefits both Hotel and consumers. The project involves developing a comprehensive Hotel Booking system composed of two main components: a website for Hotel Booking and a website for users. The website, built using ReactJS and allows hotel owners to manage their profiles, menus, and booking with ease, while the website, developed with React, empowers users to browse hotel, place bookings, and manage their carts. The backend, built with MySQL, ensures reliable data management and seamless interaction between users, hotel, and booking through RESTful APIs. By leveraging modern technologies, SelfOrder aims to create a robust, efficient, and user-friendly platform that simplifies the process of bookings room, enhances hotel management, and offers a seamless experience for customers. The system is designed to handle high volumes of concurrent users while ensuring data security and smooth communication across all components.

Keywords

Room Bookings, Hotel Management, Website, ReactJS, AdonisJS, MySQL, RESTful APIs

Chapter-1

Introduction to the Problem

1.1 Introduction

The Hotel Booking Management System is a comprehensive, user-friendly, and secure software solution developed to digitalize and automate the hotel reservation process. In today's fast-paced and highly competitive hospitality industry, efficient management of bookings and customer service is critical for success. This system addresses these needs by providing an all-in-one platform that caters to both guests and hotel administrators.The system primarily serves two types of users. First, guests who seek a quick, hassle-free, and convenient way to search for, book, and manage hotel room reservations online. Second, hotel management staff who require a centralized system to track room availability, manage guest details, process payments, handle cancellations, generate reports, and oversee financial transactions with ease.By offering real-time information on room availability, pricing, special offers, and services, the system significantly enhances the customer experience. Guests can view detailed descriptions and images of rooms, select from various room types based on their preferences, and make bookings from anywhere at any time using their devices.For hotel management, the system streamlines operations by automating routine tasks such as check-in/check-out processes, invoice generation, and room assignment. It reduces the reliance on manual record-keeping, minimizes errors, and saves valuable time and resources. Furthermore, with detailed analytics and reporting features, hotel managers can make data-driven decisions to optimize pricing, forecast demand, and improve service quality.

Background

Traditionally, hotel reservations were managed manually using physical logbooks, telephone calls, or email communications. Hotel staff had to record guest details, reservation dates, and payment information by hand, leading to high chances of human error.Miscommunication between staff members often resulted in incorrect bookings, duplicate entries, or even lost reservations. Overbooking was a common problem because real-time room inventory tracking was difficult without a centralized system.Manual methods also demanded extensive administrative efforts. Staff had to continuously monitor availability, update room status manually, and confirm bookings individually. This led to delays, customer dissatisfaction, and operational inefficiencies.

1.2 Purpose

The primary purpose of a Hotel Booking System is to digitalize and streamline the reservation process for both guests and hotel management. In today's competitive hospitality industry, manual reservation methods are no longer efficient or reliable. A booking system provides a centralized, automated platform to manage room availability, customer details, payments, and special requests effectively.One key purpose is to **enhance customer convenience**. Guests can easily check room availability, view hotel facilities, select preferred dates, and complete bookings anytime, anywhere through an online platform. Instant confirmation reduces uncertainty and improves customer satisfaction.

For hotel management, the system **reduces administrative workload**. It automates routine tasks like updating room status, managing cancellations, generating invoices, and tracking payments, allowing staff to focus more on delivering quality service to guests.

1.3 Scope

The Hotel Booking System is designed to streamline the entire hotel reservation process by offering online booking capabilities for guests and efficient management tools for hotel staff.It allows guests to view available rooms, select room types, check prices, and make secure online reservations anytime, from anywhere.The system manages real-time room availability, ensuring that overbooking and double-bookingissues are eliminated.It supports multiple payment options, including credit cards, online wallets, and bank transfers, with secure transaction processing.Hotel administrators can easily add, edit, or remove room information, pricing, and promotional offers through a user-friendly backend system.The system provides automated booking confirmations, email notifications, and booking reminders to enhance customer communication.It offers multilingual and multi-currency support to cater to international travelers and widen the hotel's reach. Customer feedback and review management are also integrated to maintain service quality and build trust. The system includes a loyalty program feature to encourage repeat bookings by offering discounts and rewards. Reporting tools and analytics are provided to help hotel management make informed decisions about pricing, marketing, and service improvements.

1.4 Objective

The specific objectives of the system are as follows:

- **Simplify Booking Process** Offer a seamless online platform for hotel room reservations.
- Reduce manual tasks for hotel staff, enabling them to focus on guest services.Implement a robust backend system using **AdonisJS** with a **MySQL** database.
- Ensure seamless communication between the frontend applications and the backend through **RESTful APIs**.
- Provide real-time updates, easy payments, and flexible booking options.
- Implement secure authentication, payment processing, and data protection measures.
- Boost hotel sales through online visibility, seasonal offers, and special deals. Help hotel management make better decisions through reporting and analytics tools.

1.5 Intended Audience and Reading Suggestions

This documentation is intended for the following audiences:

1. **Developers** – To understand system structure, backend/frontend frameworks, and database designs.
2. **Hotel Owners** – To learn how to use the portal to manage rooms, customers, and finances.
3. **Users** – To explore how easy and secure the booking process is.
4. **Project Managers and Stakeholders** – To review system scope, architecture, and project success metrics.
5. **RAD Activities:**

1. **Requirements Planning:** Interviews with hotel managers.Online surveys from frequent travelers.Finalization of mandatory features
2. **User Design:** Wireframes and mockups for customer portal and admin panel.Iterative feedback from sample users.User testing on early prototypes
3. **Construction:** Development using ReactJS (frontend) and NodeJS (backend).Database setup using MySQL.Payment gateway integration (Stripe/PayPal)
4. **Cutover:** Final system testing,Deployment to live server,Staff training and customer onboarding guides.

1.7 Document Conventions

The documentation follows the following conventions to maintain clarity and consistency:

1.7.1 Title: The document sections are clearly titled to reflect their respective content.

1.7.2 Introduction: Each section begins with an introduction that provides an overview of the topics covered.

1.7.3 Getting started: This section should provide step-by-step instructions for downloading and installing the app, as well as any initial setup steps required.

1.7.4 User interface: The document should include screenshots or illustrations of the website user interface to help users.

1.7.5 Features: This section should provide a detailed description of the website features, including what they do and how to use them.

1.7.6 Troubleshooting: The document should include a section on troubleshooting common issues that users may encounter, along with solutions or workarounds.

Chapter-2

Software Requirement Specification

2.1 Overall Description

2.1.1 Product Perspective

The Hotel Booking Management System is a standalone application designed to streamline the reservation process and manage hotel operations such as room availability, customer check-ins/outs, payments, and reporting. This system is intended to replace traditional manual booking processes with an automated, user-friendly digital platform.

2.1.2 Product Features

The **Hotel Booking Management System** provides the following key features to facilitate efficient hotel operations and enhance the customer booking experience:

Restaurant Module:

- Login
- Manage Booking
  - Add/update/delete/view
  - Enable/disable booking
- Bookings
  - Pending Bookings:
    - List of bookings which are yet to be booking
    - Approve/Reject/Cancel
    - Assign Bookings Person
    - Update status
    - Today's Bookings:
    - List of all Today's Bookings irrespective of status
  - All Bookings:
    - List of all the Bookings
- Filter by date

User Module:

- Register/Login
- Profile
  - Hotel Browsing
  - Menu Browsing
  - Cart Management
    - Add/Update/Delete Room in the cart.
    - Proceed to checkout and payment.

- ♦ Booking Management
- ♦ Booking Tracking
- ♦ Notifications

2.1.3 Design and Implementation Constraints

The following constraints impact the design and implementation of SelfOrder:

- **Technology Stack:**
  - Frontend: ReactJS (Web), React Native (Mobile)
  - Backend: AdonisJS (Node.js framework) with MySQL as the database.
  - APIs: RESTful APIs for communication between mobile, web, and backend.
- **Platform Requirements:**
  - The website must be compatible with both iOS and Android devices.
  - The web application must support all modern browsers (Chrome, Firefox, Safari).
- **Performance Considerations:**
  - The system must handle simultaneous users efficiently, especially during peak hours
  - Ensure minimum loading times for the mobile and web interfaces.

2.1.4 Assumptions and Dependencies

Following are the assumptions:

- **AS-1** Hotel staff and administrators have internet-enabled devices to manage bookings and rooms availability.
- **AS-2** Customers must have internet access to place booking and receive updates.
- **AS-3** Trained personnel will be available to handle room cleaning, check-in, customer service.

The app will be dependent on certain external factors such as:

- **DE-1** Compliance with local government regulations related to hospitality, data privacy and digital transactions (e.g., Pakistan Personal Data Protection Act or similar).
- **DE-2** The app will use third-party cloud services for data storage and server management.

2.2 System Features

The System Feature are Following:

2.2.1 User Registration and Login

Description and Priority

The registration process allows new users to create an account by providing their name, email,

password, and contact details. The login feature allows returning users to access their account by entering their credentials..

Functional Requirements

- Input: User's name, email, password.
- Output: Successful login, personalized user experience.

2.2.2 Hotel Registration and Login

Description and Priority

Hotel administrators/staff must register their establishments on the platform to start providing details. They need to provide their hotel name, location, pricing, available room types and other relevant details.This is a high –priority feature to enable hotel onboarding and system use..

Functional Requirements

- Input: Hotel name, Hotel owner, email, and other credentials.
- Output: Successful login, personalized user experience.

2.2.3 Add to Cart

Description and Priority

Users can view detailed booking information, including the selected room(s), booking duration, check-in/check-out dates, booking status, and total amount paid. This is a critical feature to help users confirm their reservation and plan their stay accordingly.

Functional Requirements

- Input: Selected Room and quantities.
- Output: Updated cart with total cost and estimated booking time.

2.2.4 Booking details

Description and Priority

Users can view detailed booking information, including the room details, Booking status (e.g., confirmed, cancelled, checked-in), check-in/check-out time

Functional Requirements

- Input: User's Booking ID.
- Output: Booking status, estimated Booking time, and personnel contact.

2.3 External Interface Requirements

2.3.1 User Interfaces

The system will use following User interfaces:

- Main Interface

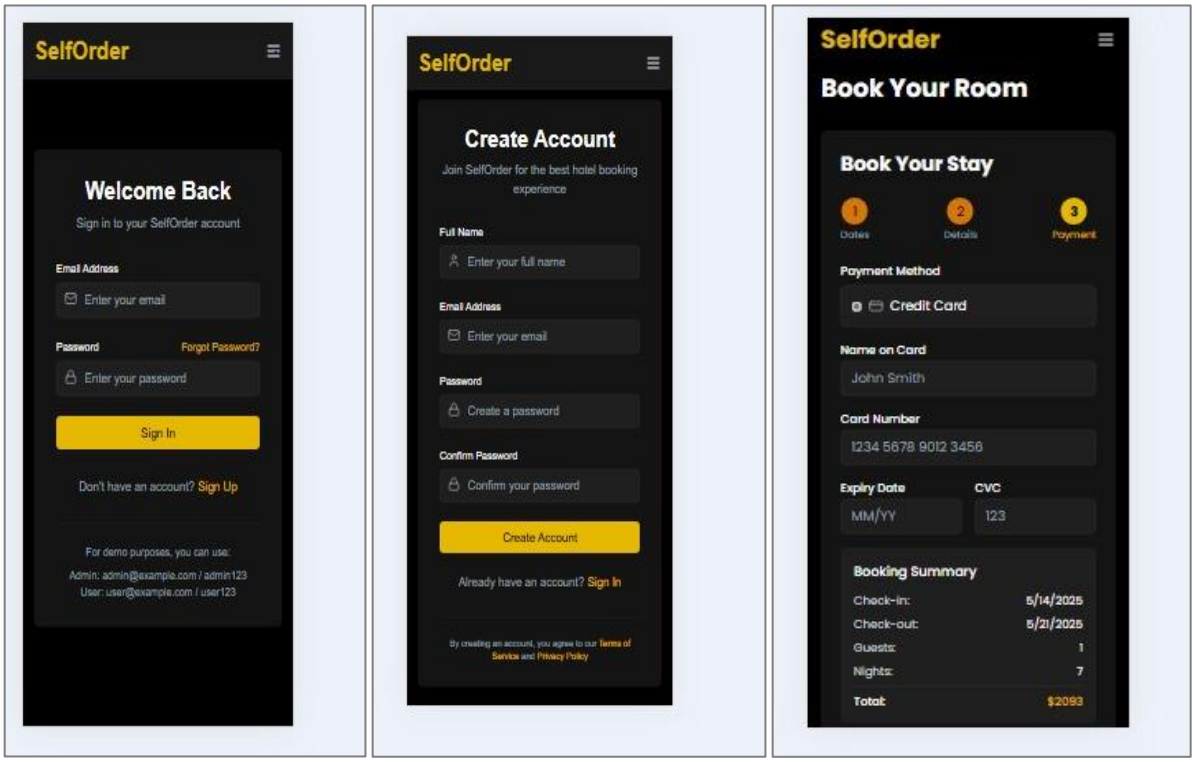

Fig.1 Main Interface

- User Login/Register Screen

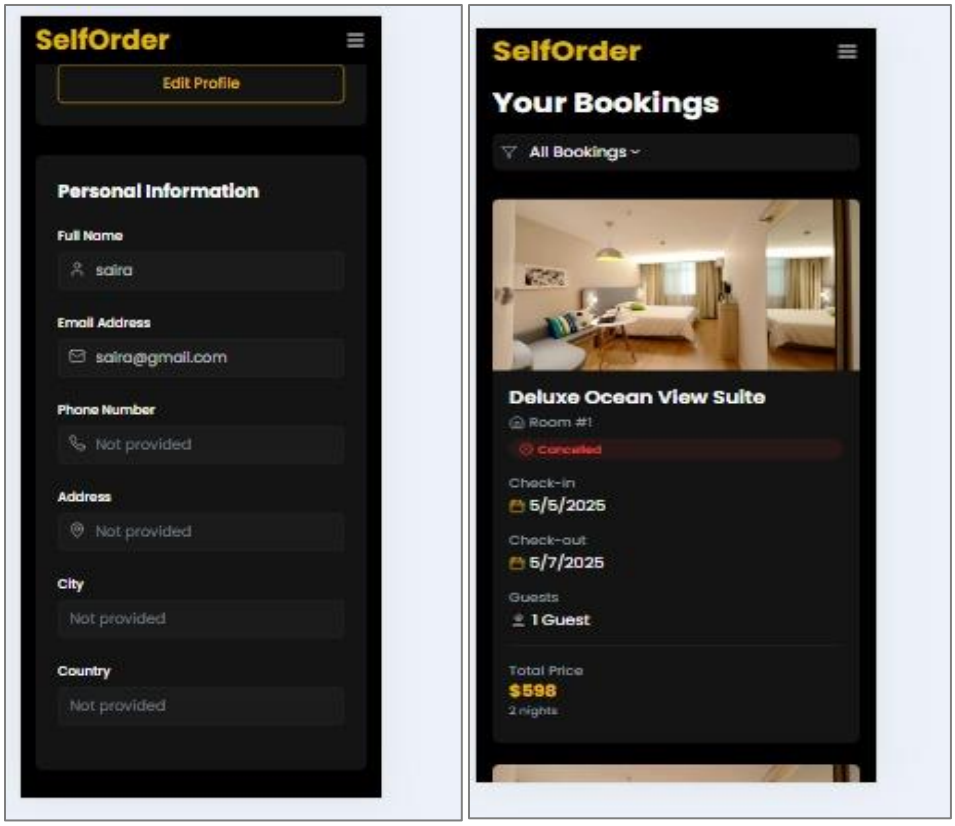

Fig.2 user login/signup

- Hotel Login/Register Screen

Fig.3Hotel login

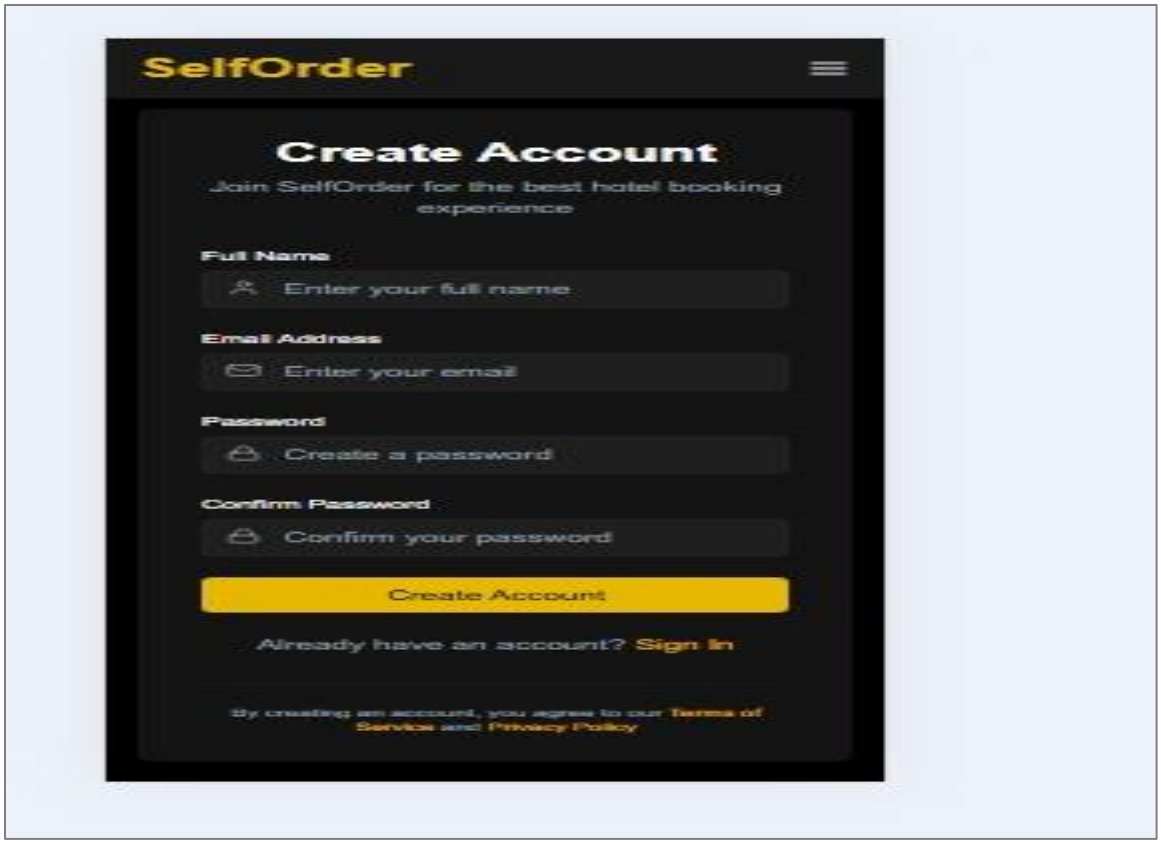

Fig.4 Hotel signup.1

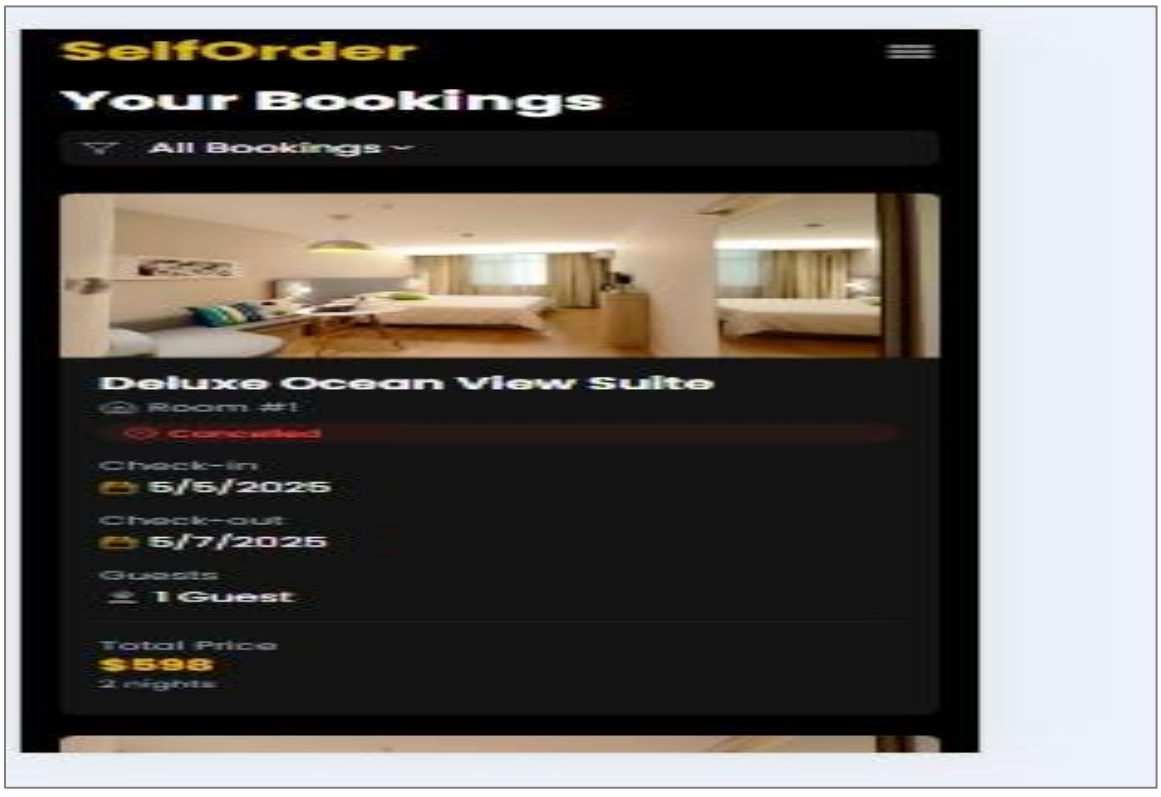

Fig.5 Hotel signup.2

- Booking Interface

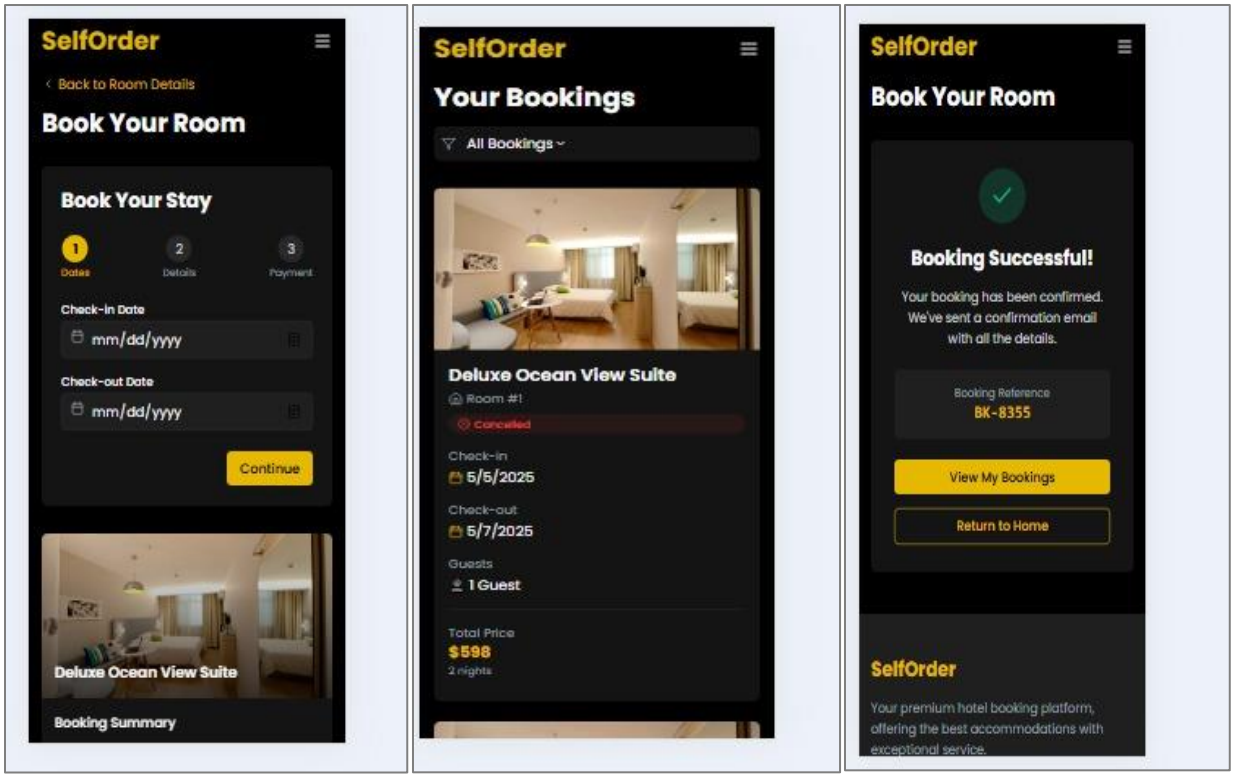

Fig.6 Booking Interface

2.3.2 Hardware Interfaces

The system will use following communication interfaces:

- **Mobile Devices:** The mobile app must support Android and iOS devices with at least 4 GB RAM.
- **Web Application:** Accessible via modern browsers on devices with at least 8 GB RAM and 128 GB storage.

2.3.3 Software Interfaces

The system will use following Software interfaces:

- **Operating System:** Android, iOS, and cross-platform support for web browsers
- **Development Environment:** Android Studio (Mobile), Visual Studio Code (Web).
- **Database:** MySQL for handling large-scale data operations.

2.3.4 Communications Interfaces

The system will use following communication interfaces:

- **Internet Connection:** Stable internet for both mobile and web apps to function.
- **Push Notifications:** For booking updates, promotions, and alerts.
- **Email Notifications:** For booking confirmations and account updates.

2.4 Other Nonfunctional Requirements

2.4.1 Performance Requirements

The performance requirements of this application are as following:

- This system will enable customers to place their booking in a fast manner.
- This system will be available 24/7 every day.
- 92% of the queries shall be completed in approximately 3.5-4 seconds.
- Easy to use
- Error free

2.4.2 Safety Requirements

The safety requirements of this application are as following:

- Sensitive data isn't distributed among third party mediators.
- Sensitive data is not stored outside the app's storage system.
- The application should comply with relevant local privacy and data protection laws.

2.4.3 Security Requirements

The security requirements of this application are as following:

- Every user must change his initial password after first successful login.
- Personal Data (Personal Information) of user is protected.
- Payment Methods are safe.
- Ensure compliance with relevant data protection laws in Pakistan, such as the Pakistan Personal Data Protection Act (PDPA) or any other applicable regulations. Understand the legal obligations regarding data collection, storage, and processing.
- Implement secure authentication mechanisms, such as strong password policies, password hashing, and encryption, to protect user login credentials and prevent unauthorized access.

Chapter-3

Analysis (Use case Model)

3.1 Identifying Actors and Use Cases using Textual Analysis

3.1.1 Use Case Name: Hotel Owner

DC#	1	Ref:	Req#1
UC Name	Signup as Hotel Owner.		
Level	Detailed		
Description	Create Hotel owner profile.		
Actor	Hotel Owner		
Stakeholders	Hotel Owner		
Preconditions	New account is created.		
Success Guarantee	Signup successfully.		
Main Success Scenario	**Action**	**Response**	
	1. Enter the basic details.		
		2. Account created successfully.	
Extensions	If something went wrong, system respond accordingly		
Special requirements	Application should be running		
Frequency of Occurrence	Any Time when the Hotel owner needs to Login		

3.1.2 Use Case Name: Customer

DC#	2	Ref:	Req#2
UC Name	Signup as Customer.		
Level	Detailed		
Description	Create Customer profile.		
Actor	Customer		
Stakeholders	Customer		
Preconditions	New account is created.		
Success Guarantee	Signup successfully**.**		
Main Success Scenario	**Action**	**Response**	
	1. Enter the basic details.		
		2. Account created successfully.	
Extensions	If something went wrong, system respond accordingly		
Special requirements	Application should be running		
Frequency of Occurrence	Login Any Time the customer uses the App		

3.2 Forming Use Case Diagram with Candidate and Use Cases

3.2.1 Use Case diagram: Hotel Owner

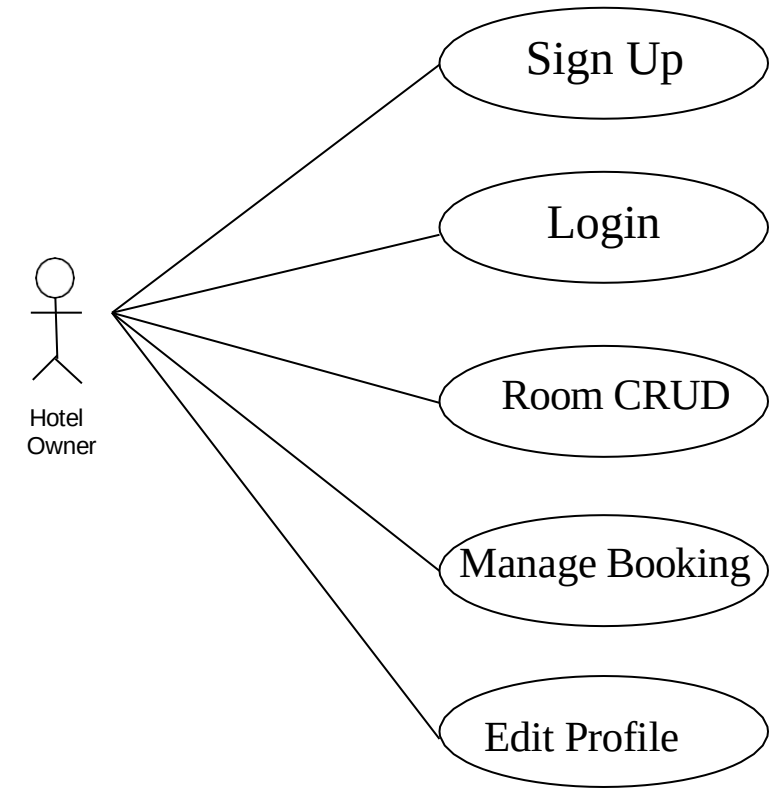

Fig.7 Use case Hotel Owner

3.2.2 Use Case diagram: Customer

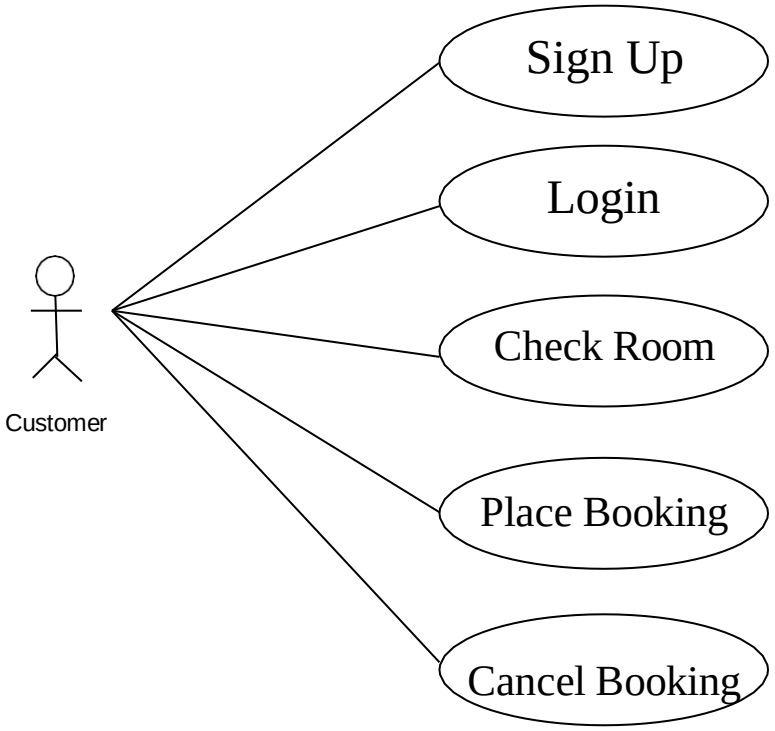

Fig.8 Use case Customer

3.3 Describe the Events Flow for Use Case

The event flow explains the interactions between the actor(s) and the system, detailing what the system actually does in response to the user's actions.

3.3.1 Addition of Hotel Owner

- **Precondition:**

  Hotel Owner must create an account.

- **Main Flow:**
  - Enter full name
  - Add booking charges
  - Add mobile number
  - Enter email address
  - Add Hotel details(name, location, license key)
  - Add time schedule
  - Submit

3.3.2 Addition of Customer

- **Precondition:**

  Customer must create an account.

- **Main Flow:**
  - Enter full name
  - Set password
  - Add mobile number
  - Enter address
  - Submit

Chapter-4

Design

4.1 Architecture Diagram

4.1.1 Definition:

An architecture diagram is a visual representation of all the elements that constitute part or all of a system. It helps engineers, designers, stakeholders, and others involved in the project understand the structure and layout of the system.

4.1.2 Diagram:

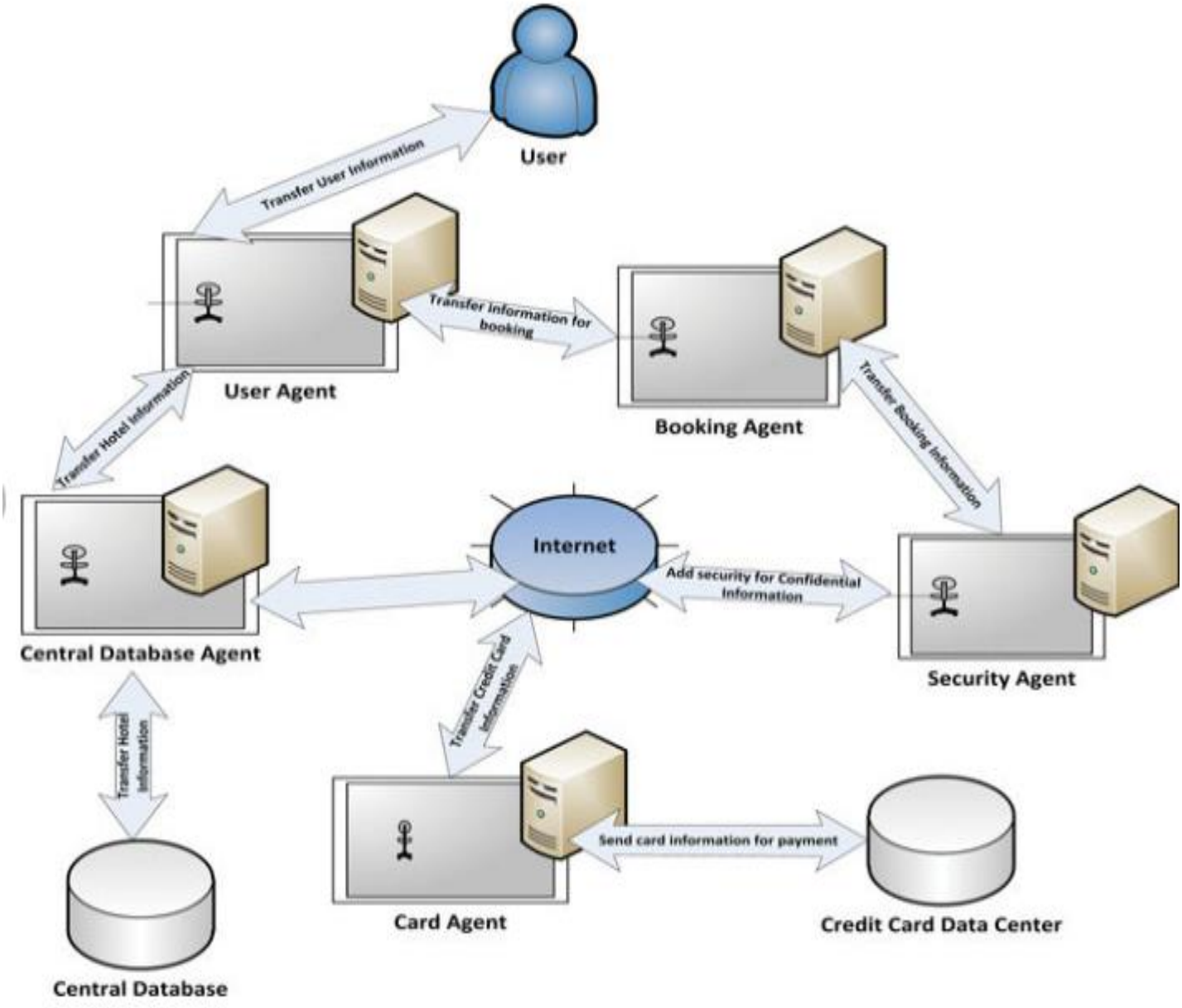

Fig.9 Architecture Diagram

4.2 ERD Diagram with data dictionary

4.2.1 Definition:

An Entity Relationship (ER) Diagram is a flowchart that shows relationships between entities like people, things, or concepts inside a system. ER Diagrams are used in software engineering and business information systems to design relational databases. The ER diagram in the *SelfOrder* system shows how the Hotel Owner creates booking rooms, the Customer checks and booking room.

4.2.2 Diagram:

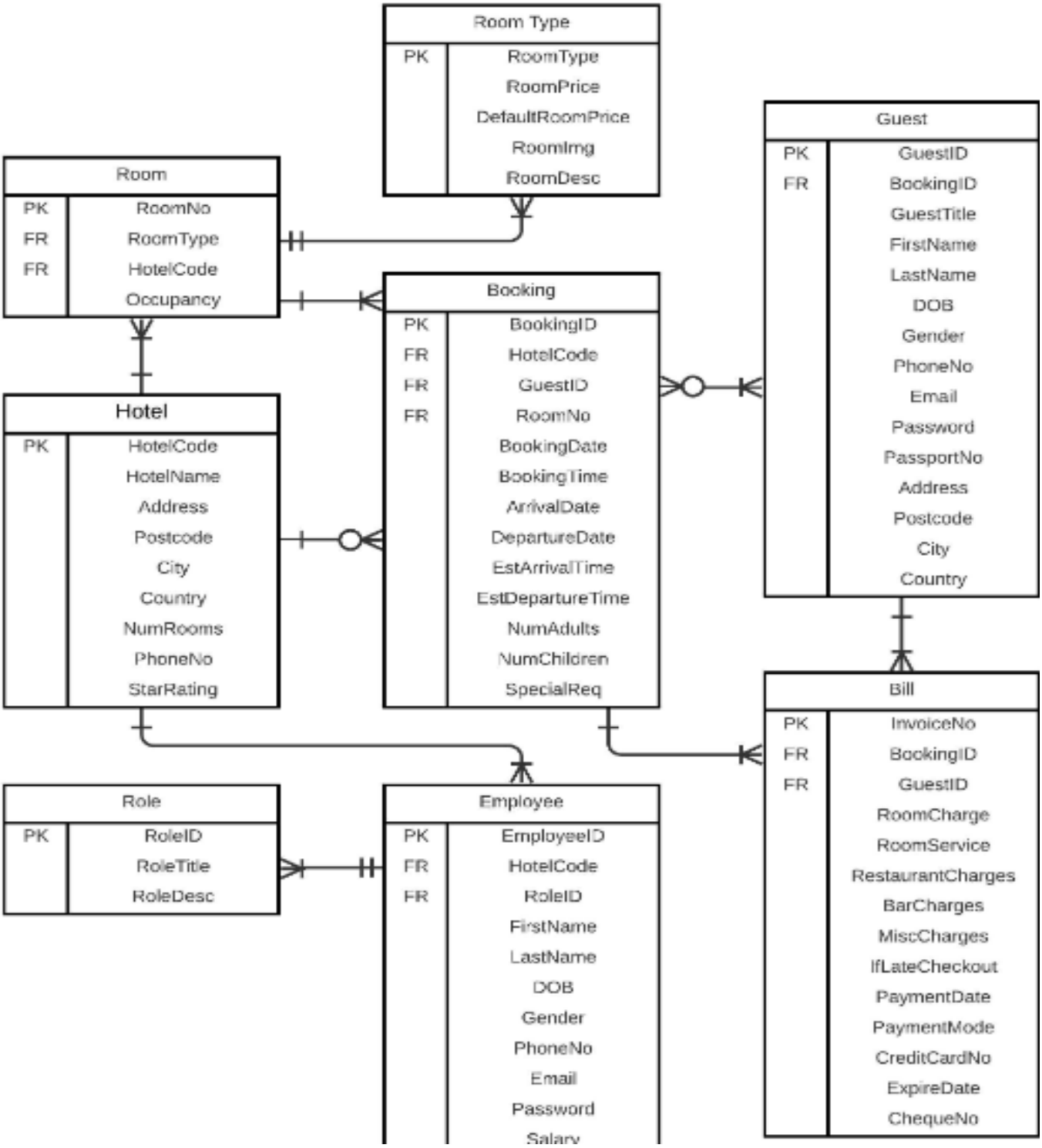

Fig.10 ERD

4.3 Class Diagram

4.3.1 Definition:

A class diagram in UML illustrates the classes, attributes, operations, and interactions between objects in a system. In *SelfOrder*, the primary classes include the Hotel Owner and Customer with subclasses for specific functionalities like Cart and menu.

4.3.2 Diagram:

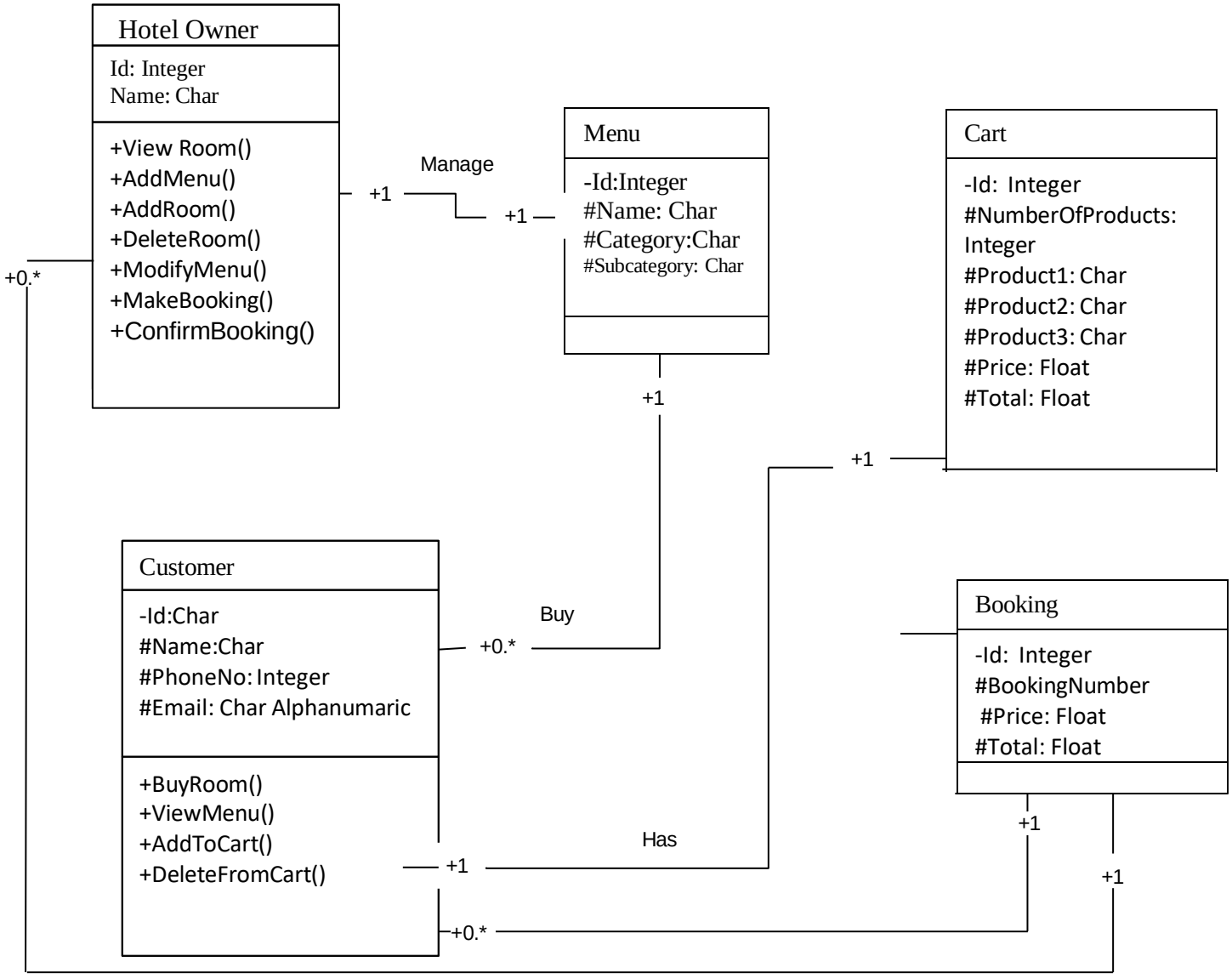

Fig.11 Class Diagram

4.4 Object Diagram

4.4.1 Definition:

An object diagram is a runtime instance of a class diagram that shows objects and their relationships at a specific moment. It captures the detailed state of a system at a particular point in time.

4.4.2 Diagram:

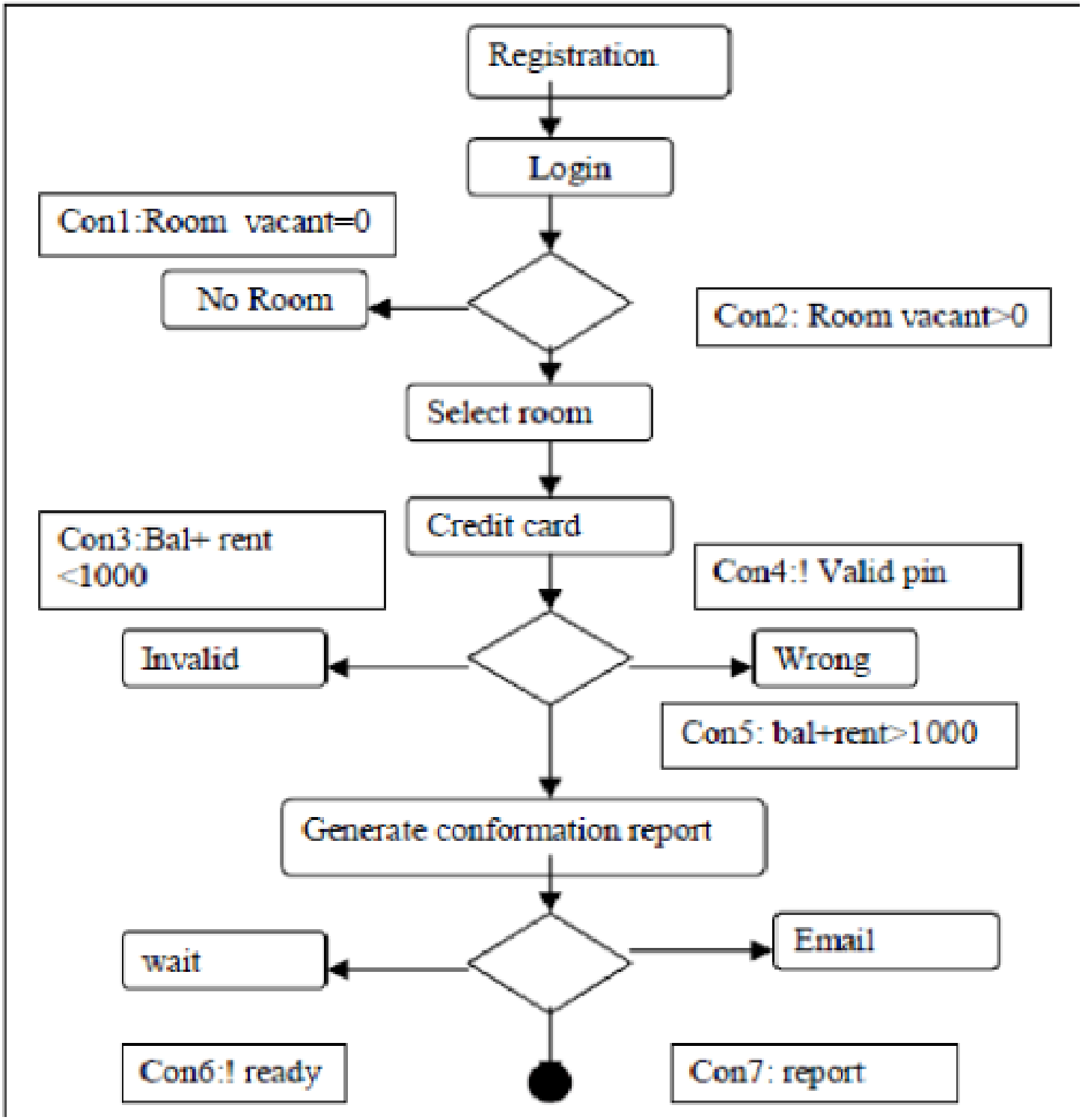

Fig.12 Object Diagram

4.5 Sequence Diagram

4.5.1 Definition:

A sequence diagram shows the flow of messages sent and received by objects during an interaction. In *SelfOrder*, the sequence begins when the user logs into the application. The customer selects Room and Booking, hotel view the booking, and booking it to the customer.

4.5.2 Diagram:

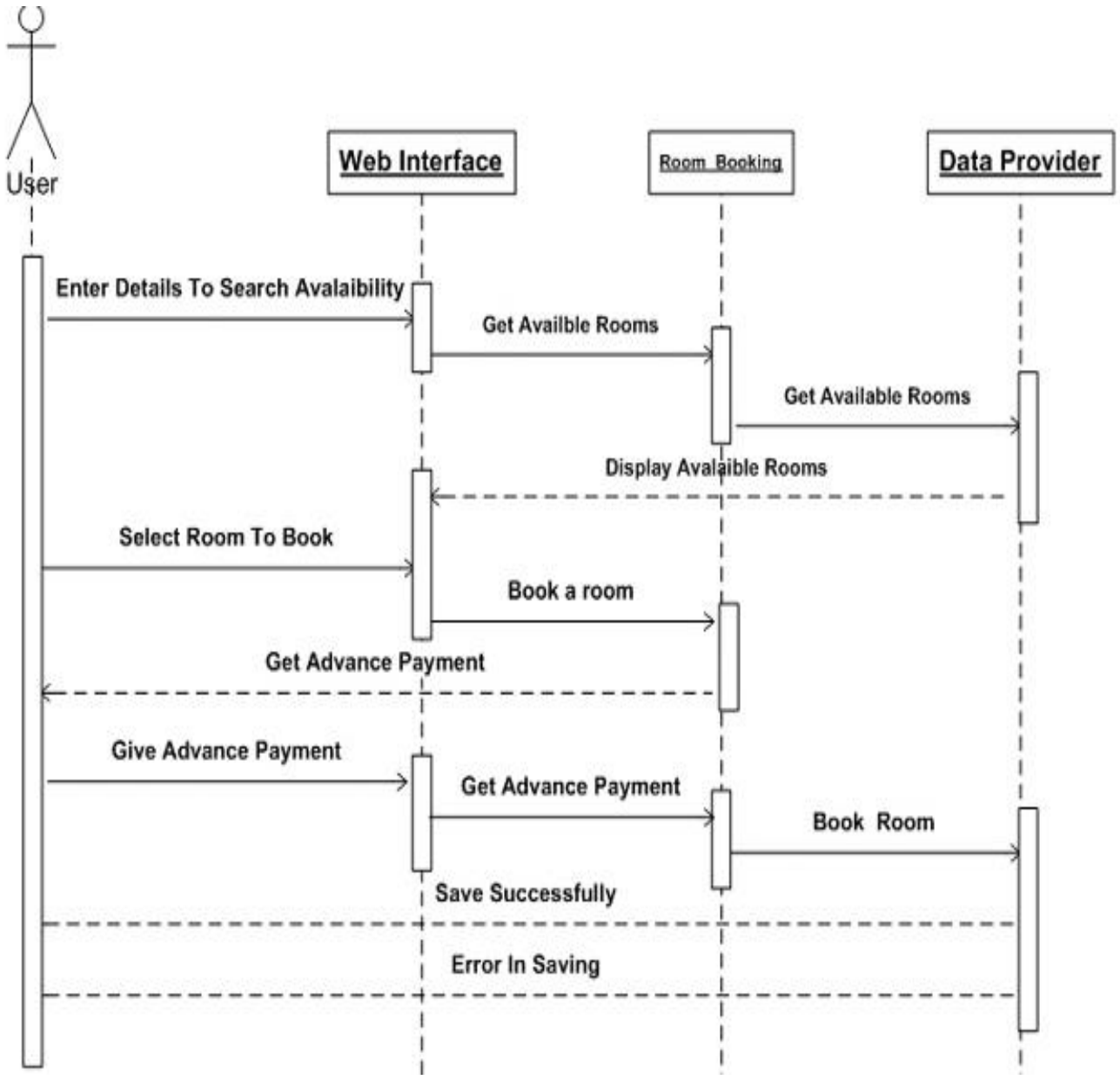

Fig.13 Sequence Diagram

4.6 Activity Diagram

4.6.1 Definition:

An activity diagram visualizes business and software processes as a succession of actions. These functions can be carried out by individuals, hardware, or software. In *SelfOrder*, the activity diagram shows the processes of booking room and booking it to the customer.

4.6.2 Diagrams:

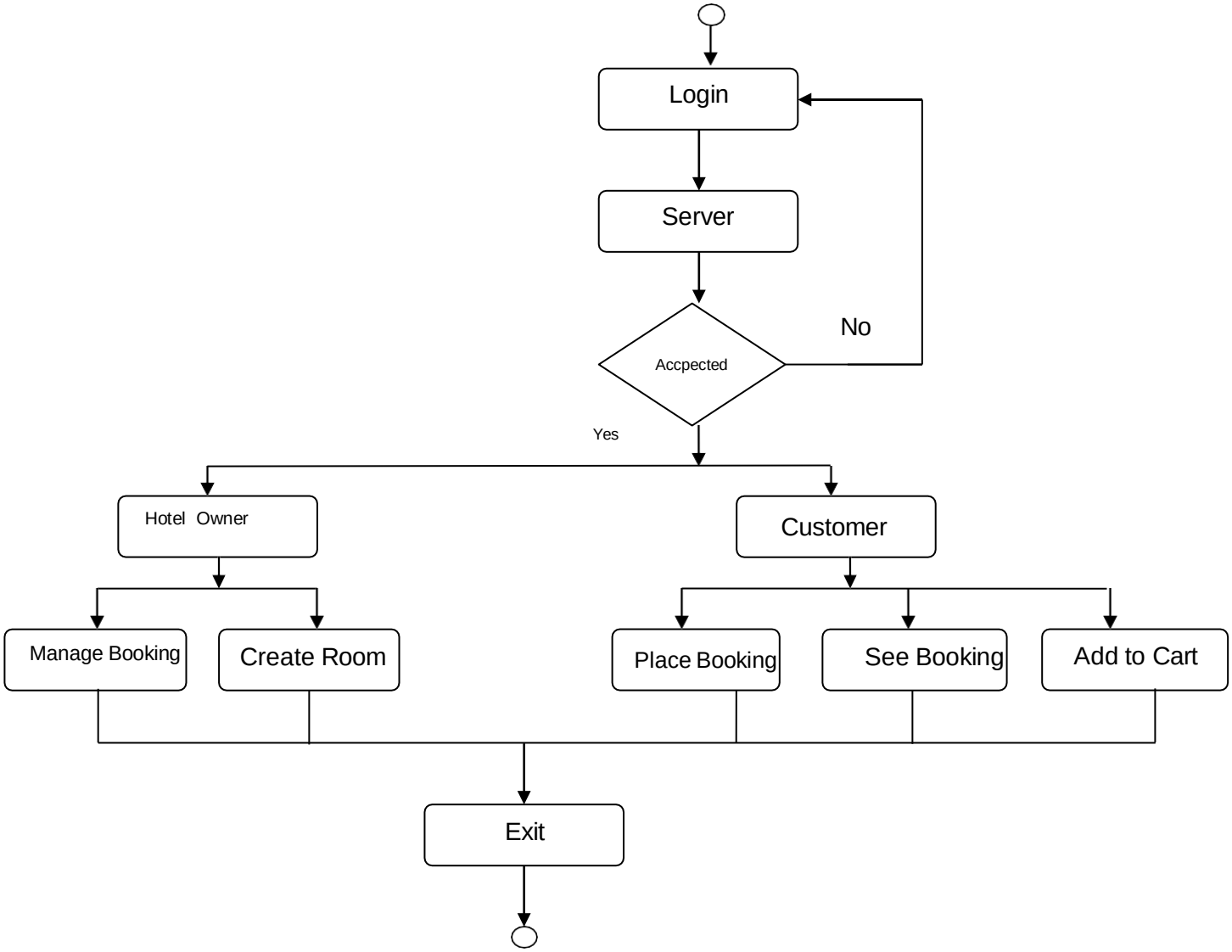

Fig.14 Activity Diagram

4.7 Collaboration Diagram

4.7.1 Definition:

A collaboration (communication) diagram illustrates the connections and interactions between software elements. In *SelfOrder*, it shows how the customer chooses room, places booking, and how the Hotel view the booking.

4.7.2 Diagram:

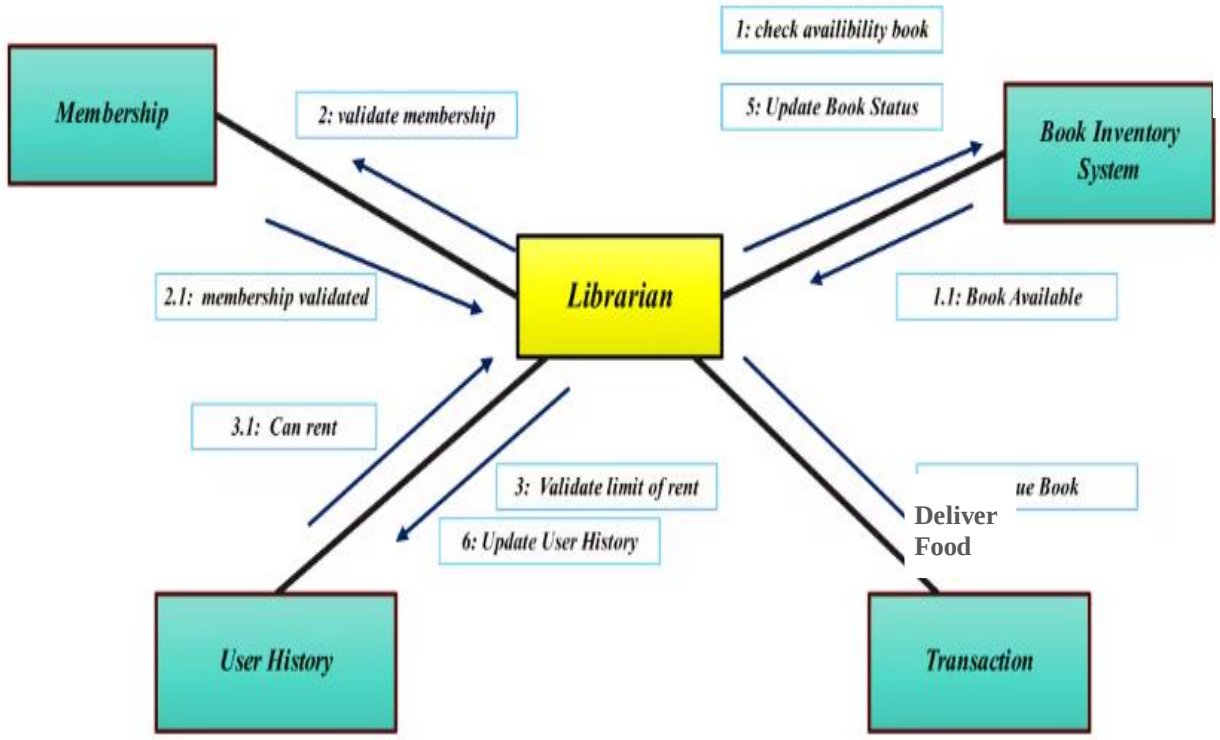

Fig.15 Collaboration Diagram

4.8 State Transition Diagram

4.8.1 Definition:

A state-transition diagram shows the states an object can be in and the conditions under which it changes states. For *SelfOrder*, the system starts in an idle state. When the customer interacts with the menu, the system transitions between different states based on input, such as placing or canceling an booking.

4.8.2 Diagram:

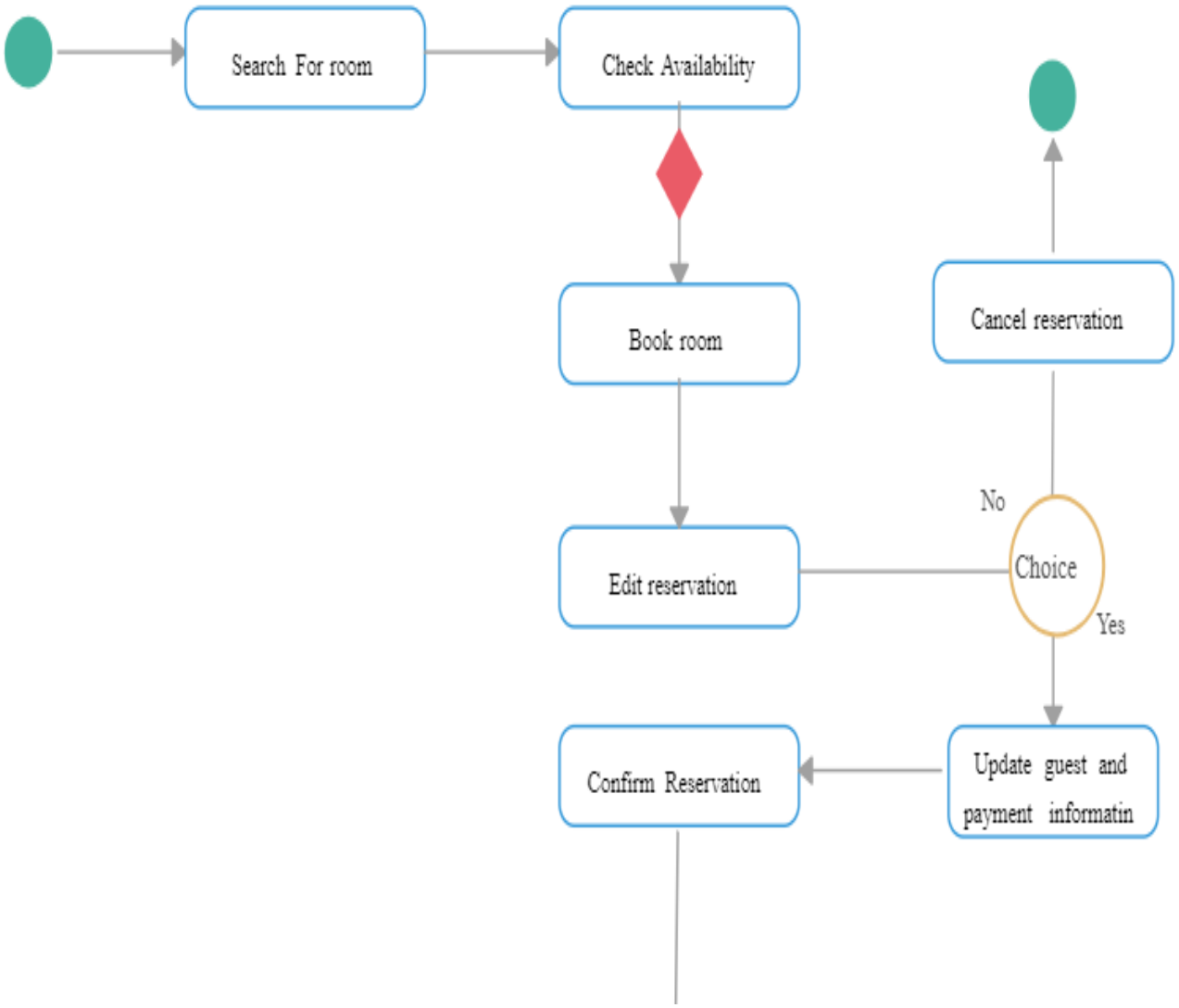

Fig.16 State Transmission Diagram

Chapter-5

Development

5.1 Operating System

When developing the SelfOrder room booking website, several operating systems can be considered, depending on the target audience and devices used.

5.1.1 Available Operating System

Here are some commonly used OS options for developing a SelfOrder room booking website:

- **Android:** Android is an open-source mobile OS, widely used in smartphones, tablets, and other devices. As the majority of users own Android devices, it becomes an essential platform for SelfOrder's web app development. It offers a flexible development environment with access to various tools and libraries.
- **iOS:** Apple's iOS is popular among iPhone and iPad users. Targeting iOS would allow SelfOrder to reach a significant user base. However, development requires adherence to Apple's strict guidelines and the use of the Swift programming language.
- **Web-based:** Developing a web-based app makes the application accessible across various platforms and devices through a web browser. This ensures that SelfOrder can reach users regardless of whether they use Android, iOS, or even desktop platforms.
- **Cross-platform frameworks:** Cross-platform frameworks like React Native and Flutter enable the development of applications for both Android and iOS with a single codebase. This approach saves development time and effort while maintaining the app's quality across platforms.

5.1.2 Selected Operating System

- **Android:** For *SelfOrder*, the Android operating system is chosen as the primary platform for the web application. Since most potential users (customers) use Android-based smartphones, this makes Android a suitable and cost-effective choice for initial deployment.
- **Web-based:** For the SelfOrder Hotel web application, a web-based operating system approach has been selected. This allows the application to be platform-independent, making it accessible from any device with a web browser, including desktops, laptops, tablets, and smartphones. Furthermore, this choice supports a responsive design that ensures the web app adapts seamlessly to different screen sizes and devices.

5.2 Development Approach

Development approach refers to the methodology or strategy followed during the software development process. Different development approaches provide guidelines on how to plan, design, develop, and deliver software projects.

5.2.1 Available Development Approach

Some common development approaches include:

- **Waterfall:** The waterfall approach follows a sequential and linear development process, with distinct phases such as requirements gathering, design, development, testing, and deployment. Each phase is completed before moving on to the next, and changes made in earlier phases may be challenging to accommodate later.
- **Agile:** Agile is an iterative and flexible development approach that emphasizes collaboration, adaptability, and customer feedback. It involves breaking down the project into small increments called sprints, where each sprint delivers a functional piece of software. Agile methodologies include Scrum, Kanban, and Extreme Programming (XP).
- **Spiral:** The spiral approach combines elements of the waterfall and iterative approaches. It involves multiple iterations of development, where each iteration builds upon the previous one. The spiral approach focuses on risk management and incorporates prototyping and customer feedback.
- **Rapid Application Development (RAD):** RAD emphasizes rapid prototyping and quick iterations to develop software. It involves close collaboration between developers and users to gather requirements, build prototypes, receive feedback, and iterate on the solution.
- **DevOps:** DevOps is an approach that emphasizes collaboration and integration between development and operations teams. It involves continuous integration, delivery, and deployment to streamline the software development lifecycle and ensure faster and more reliable releases.
- **Lean:** The lean approach aims to eliminate waste and focus on delivering value to the customer. It emphasizes efficiency, continuous improvement, and reducing unnecessary steps or activities in the development process.
- **Incremental:** The incremental approach involves dividing the project into smaller increments or modules, each of which is developed and delivered independently. This approach allows for early delivery of functioning features and facilitates ongoing customer feedback and requirements refinement

5.2.2 Selected Development Approach

For the development of **SelfOrder**, we have selected the **Agile** approach:

Requirements Gathering: In this phase, the functional requirements were identified, including hotel management, menu updates and user roles for both the mobile app (users) and web app (hotel).

System Design: The system's architecture, including the web interface for hotel and the mobile app for customers, was designed. The backend APIs were structured with a focus on scalability and performance.

Implementation: The Website is developed using React Native for cross-platform compatibility, and the web application is implemented with React.js. Backend services, including authentication and booking management, were handled using Adonis.js.

Testing: Continuous integration ensured thorough testing of each feature, including unit testing, integration testing, and user acceptance testing. The system was iteratively tested at each sprint.

Deployment: The app will be deployed on both the Google Play Store (for Android users) and the web platform for hotel owners, ensuring a smooth rollout.

Operation and Maintenance: Post-deployment, the system will undergo regular updates and maintenance, including bug fixes, performance improvements, and security updates.

5.3 Programming Language

A programming language is a formal language that provides a set of instructions for a computer to perform specific tasks. It allows programmers to write code and communicate with computers to develop software applications, scripts, or algorithms. Programming languages serve as a means for humans to convey their instructions to computers effectively.

5.3.1 Available Programming Language

- **Node.js:** Node.js is a JavaScript runtime built on Chrome's V8 engine that enables developers to run JavaScript code outside the browser. It is widely used for building server-side applications due to its event-driven, non-blocking I/O model, which allows it to handle multiple requests efficiently. This makes Node.js particularly suitable for real-time applications, APIs, and microservices. Its lightweight architecture and scalability have contributed to its popularity among developers. Additionally, Node.js has a vast ecosystem of open-source libraries and tools available through npm (Node Package Manager), which speeds up development and adds flexibility.
- **Express.js:** Express.js is a minimal and flexible Node.js web application framework that provides robust features for building web and mobile applications. It simplifies the process of handling HTTP requests, routing, and middleware integration, making server-side development more efficient. With Express, developers can quickly set up APIs and web servers using clean and organized code. It supports various templating engines and integrates easily with databases and other tools. Express.js is widely used due to its simplicity, performance, and strong community support.
- **JavaScript:** For web development, mobile development and handling backend logic, JavaScript is a widely used, event-driven language that can manage multiple operations efficiently.
- **TypeScript:** TypeScript is a superset of JavaScript that provides static typing. It's often used with React.js and Node.js for better code quality and maintainability.
- **React:** React is a popular JavaScript library developed by Meta (formerly Facebook) for building user interfaces, particularly single-page applications. It follows a component-based architecture, allowing developers to create reusable UI components that manage their own state. React uses a virtual DOM to

efficiently update and render only the parts of the page that change, improving performance. This makes it ideal for building fast, interactive, and dynamic web applications. With a large ecosystem and strong community support, React has become a go-to choice for modern front-end development.

5.3.2 Selected Programming Language

- **JavaScript (React.js):** The website will be developed using **React** and JavaScript, ensuring cross-platform compatibility. The web application for Hotel owners will be built using **React.js** for fast, efficient, and scalable frontend development.

- **Adonis.js (Node.js for Backend): Adonis.js** will be used for backend development. It leverages **Node.js**, providing an efficient framework for handling asynchronous operations, database connections, and API requests.

5.4 Platform

5.1.1 Available Platform

For mobile and web application development, various platforms are available. Some common choices include:

- **Android Studio:** Android Studio is an integrated development environment (IDE) specifically for Android app development. It offers various tools for debugging, testing, and UI design.
- **AdonisJS:** Adonis.js is a Node.js framework for backend development that offers a well-organized structure for API creation, database connections, and user authentication.
- **ReactJS:** React.js is a popular JavaScript library used for building dynamic web applications. It provides efficient rendering and smooth user interfaces.

5.1.2 Selected Platform

- **Android Studio:** Android Studio is an integrated development environment (IDE) specifically for Android app development. It offers various tools for debugging, testing, and UI design. We use for android studio for mobile app development in SelfOrder project.

- **React.js:** The web-based system for restaurant owners and administrators will be developed using **React.js**, offering fast performance and scalability.

- **Adonis.js (Node.js Framework):** The backend services will be managed using **Adonis.js**, a powerful Node.js framework that provides an efficient environment for API development and database management.

5.4 Case Tool

CASE (Computer-Aided Software Engineering) tools, also known as CASE tools or CASE environments, are software applications that provide support for various activities involved in the software development process. These tools are designed to assist software developers, analysts, and project managers in tasks such as analysis, design, coding, testing, and maintenance of software systems.

5.4.1 Available Case tools

There are several CASE (Computer-Aided Software Engineering) tools available that can aid in the development of a train room booking website on the Android platform. Here are some commonly used CASE tools that can be helpful in different stages of the software development lifecycle:

- **Unified Modeling Language (UML) Tools:** UML tools provide support for creating visual models and diagrams to represent the structure, behavior, and relationships of a software system. They help in designing the app's architecture, data flow, and user interface. Some popular UML tools include Enterprise Architect, Lucidchart, and Visual Paradigm.
- **Integrated Development Environments (IDEs):** IDEs are comprehensive software development environments that offer a range of tools and features to support coding, debugging, and testing. Android Studio is the official IDE for Android app development, providing essential features like code editing, compilation, debugging, and deployment.
- **Version Control Systems (VCS):** Version control systems are used to manage source code and track changes made during development. They allow developers to collaborate, manage different versions of the codebase, and handle conflicts efficiently. Git, which is widely used in conjunction with platforms like GitHub and Bitbucket, is a popular version control system.
- **Issue Tracking and Project Management Tools:** These tools help manage tasks, track issues, and facilitate collaboration within development teams. They allow you to assign tasks, set priorities, track Progress, and communicate with team members. Examples of such tools include Jira, Trello, and Asana.

5.4.2 Selected Case Tools

- **Unified Modeling Language (UML) Tools:** Unified Modeling Language (UML) tools offer several benefits in the software development process. One major advantage is their ability to provide a visual representation of the system being developed. UML diagrams allow developers to create models and diagrams that illustrate the structure, behavior, and relationships of the software. This visual representation makes it easier for team members, stakeholders, and clients to understand and communicate complex concepts and design decisions. UML tools also provide a standardized notation, ensuring consistency and clarity in communication across the development team. Another benefit is the documentation aspect.

5.5 Data Base

A database is an organized collection of structured data that is stored and accessed electronically. It is designed to efficiently store, retrieve, and manage large amounts of data. Databases are widely used in various applications and industries to store and organize data in a structured manner.

5.5.1 Available Data Base

Here are some commonly used databases that can be suitable for a train room booking web app:

- **Relational Databases:**
    - **MySQL:** MySQL is a popular open-source relational database management system (RDBMS) known for its scalability, performance, and wide community support. It is well-suited for applications with structured data and complex querying needs.
    - **PostgreSQL:** PostgreSQL is another powerful open-source RDBMS that offers advanced features, extensibility, and strong data integrity. It provides support for complex queries, JSON data, and spatial data, making it suitable for applications requiring advanced data manipulation capabilities.
- **NoSQL Databases:**
    - **MongoDB:** MongoDB is a popular open-source document-oriented NoSQL database. It offers flexible schema design, scalability, and high performance for handling unstructured or semi-structured data.
    - **Firebase Real-time Database:** Firebase is a cloud-based platform that provides a NoSQL database, among other services. Firebase Realtime Database offers real-time data synchronization and offline support, making it suitable for real-time collaboration and synchronization in the room booking website.

5.5.2 Selected Database

Firebase Real-time Database: It offers features for backend infrastructure, real-time database, user authentication, cloud messaging, analytics, and more. This integrated approach simplifies the development process by providing a unified platform to handle multiple aspects of app functionality. Secondly, Firebase is known for its real-time database capabilities. The Firebase Real-time Database enables real-time data synchronization across clients, allowing instant updates and changes to be propagated in real-time. This is particularly useful for applications that require real-time collaboration or instant data updates, such as chat apps or collaborative document editing.

Chapter-6

Testing (Software Quality Attributes)

6.1 Test Case Specification

This section outlines the testing phase of the **SelfOrder** Room Booking system. Testing ensures that the web applications for Hotel and users function as intended, meet system requirements, and offer an efficient and effective user experience.

6.2 Black Box Test Case

Black Box Testing focuses on testing the system's functionality without knowing the internal workings or code structure. Below are the methods used for black box testing in **SelfOrder**.

6.2.1 BVA or Boundary Value Analysis:

Boundary Value Analysis is a black box test design technique used to identify errors at the boundaries of input domains. It focuses on values at the edges of the allowable input range (minimum, maximum, and values just inside and outside these boundaries). For **SelfOder**, test cases were designed to ensure that data inputs such as user login credentials, menu item prices, and booking quantities work correctly at their boundaries. For instance, when inputting the number of items in an booking, tests were performed for 0, 1, maximum number allowed, and beyond the maximum to ensure error handling.

6.2.2 Equivalence Class Partitioning

In Equivalence Class Partitioning, the input data domain is divided into valid and invalid partitions. This method reduces the number of test cases by categorizing inputs into classes that are expected to exhibit similar behavior. For example, in the **SelfOrder** system, valid user credentials (email and password) are classified into valid and invalid input classes, where valid inputs include correctly formatted email addresses and strong passwords, while invalid inputs might include incorrect formats or missing information.

6.2.3 State Transition Testing

State Transition Testing examines how the system transitions between different states based on the input. This method checks the behavior of the system when certain actions are performed in specific states.

In **SelfOrder** examples of state transitions include:

- A hotel owner logs in and transitions from the login page to the hotel dashboard.
- A customer selects a hotel, adds food items to the cart, and transitions to the booking confirmation page.

6.2.4 Decision Table Testing

Decision Table Testing uses a table to represent inputs and the expected outcomes based on various conditions. This is particularly useful in testing scenarios where multiple conditions can lead to different outputs.

In **SelfOrder**, examples of state transitions include:

- A hotel owner logs in and transitions from the login page to the hotel dashboard.
- A customer selects a hotel, adds food items to the cart, and transitions to the order confirmation page.

6.2.5 Graph Base Testing

Graph-based testing involves creating a cause-effect graph that links inputs to their respective outputs. This ensures all potential combinations of inputs are tested.

For **SelfOrder**, this technique was used to test:

- Cause: A customer places an booking.
- Effect: The system triggers notifications to the hotel and updates booking status, and deducts the payment.

6.3 White Box Testing

6.3.1 Statement Coverage

Statement Coverage ensures that all statements in the source code are executed at least once. This helps detect unreachable or dead code and ensures that all features have been thoroughly tested.

For **SelfOrder**, statement coverage was achieved by executing all actions like:

- User login and registration
- Adding and removing items from the cart
- Viewing bookings from the hotel and delivery dashboards

6.3.2 Branch Coverage

Branch Coverage ensures that each decision point in the code is tested, including both true and false outcomes. This ensures all branches are exercised.

In **SelfOrder**, branch coverage tests were conducted on:

- Conditional statements in the booking processing logic (e.g., if a payment fails or succeeds different branches of the code are executed).
- Conditional rendering of UI elements based on user roles (e.g., hotel owner vs. customer).

6.3.3 Path Coverage

Path Coverage involves testing all possible paths through the system's code. Each possible flow of the application is executed at least once.

In **SelfOrder**, path coverage testing ensured that all paths through:

- User registration, login, adding items to the cart, placing an booking, and checking out
- Hotel owners updating the menu, viewing pending bookings, and marking bookings as completed.

6.4 Test Cases

Test Case 1:

Date: 1 May, 2025	
System: Android & iOS (React)	
Objective: Login as Customer	Test ID:1
Version:1	
Input: Customer credentials.	
Expected Result: Display customer dashboard	
Actual Result: Passed	

Test Case 2:

Date: 2 May, 2025	
System: Web (React.js)	
Objective: Login as Hotel Owner	Test ID:2
Version:1	
Input: Hotel owner credentials	
Expected Result: Display Hotel owner dashboard	
Actual Result: Passed	

Test Case 3:

Date: 3 May, 2025	
System: Android & iOS (React)	
Objective: Register as Customer	Test ID:3
Version:1	
Input: Email, password, and other registration details.	
Expected Result: Successful registration, display customer dashboard	
Actual Result: Passed	

Test Case 4:

Date: 4 May, 2025	
System: Web (React.js)	
Objective: Add Room Items to Hotel Booking	Test ID:4
Version:1	
Input: rooms no, description, price	
Expected Result: New Room item is added to the room	
Actual Result: Passed	

Test Case 5:

Date: 5 May, 2025	
System: Android & iOS (React Native)	
Objective: Add Item to Cart	Test ID:5
Version:1	
Input: Select Room item and add to cart	
Expected Result: Room item added to cart	
Actual Result: Passed	

Test Case 6:

Date: 6 May, 2025	
System: Android & iOS (React)	
Objective: Place booking	Test ID:6
Version:1	
Input: Click on booking now button	
Expected Result: booking is successfully placed	
Actual Result: Passed	

Test Case 7:

Date: 7 May, 2025	
System: Web (React.js)	
Objective: View bookings as Hotel Owner	Test ID:7
Version:1	
Input: Click on "View Hotel"	
Expected Result: Display Booking	
Actual Result: Passed	

Test Case 8:

Date: 8 May, 2025	
System: Web (React.js)	
Objective: Mark booking as Completed	Test ID:8
Version:1	
Input: Click on "Mark as Completed"	
Expected Result: Booking is marked as completed	
Actual Result: Passed	

Test Case 9:

Date: 9 May, 2025	
System: Android & iOS (React)	
Objective: View Booking as Customer	Test ID:9
Version:1	
Input: Click on "View Order"	
Expected Result: Display booking details and status	
Actual Result: Passed	

Test Case 10

Date: 10 May, 2025	
System: Web (React.js)	
Objective: View Pending Bookings as Booking	Test ID:10
Version:1	
Input: Click on "View Pending Bookings"	
Expected Result: Display pending booking.	
Actual Result: Passed	

Chapter-7

Implementation

7.1 Component Diagram

7.1.1 Definition

A component diagram, often called a UML component diagram, shows how the physical parts of a system are wired up and organized. To represent implementation specifics and ensure that all necessary functionalities of the system are covered by planned development, component diagrams are frequently created.

7.1.2 Diagram

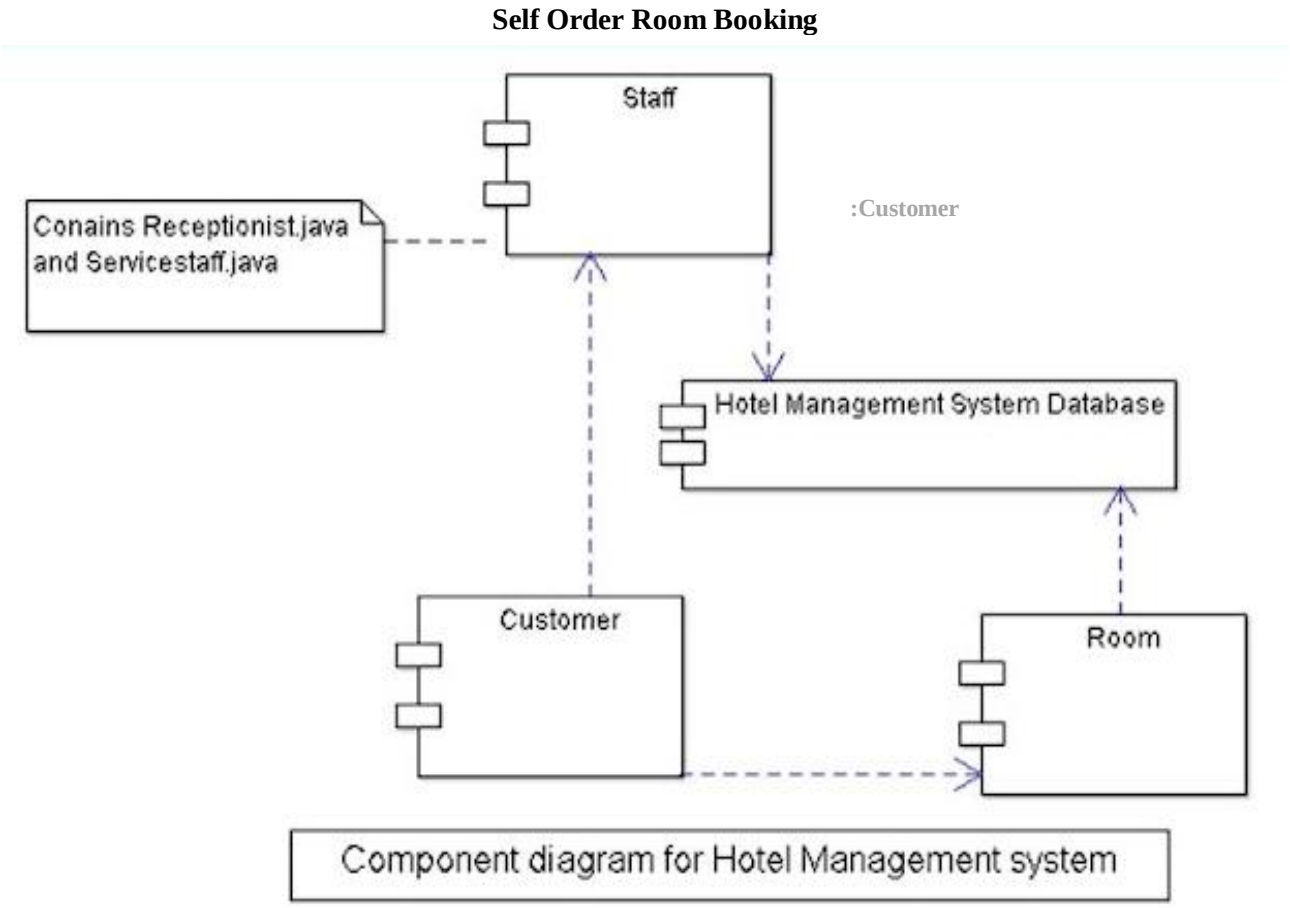

Fig.17 Component Diagram

7.2 Deployment Diagram

7.2.1 Definition

Deployment diagrams represent a system's physical architecture. The links between the system's hardware and software components as well as the physical distribution of the processing are displayed in deployment diagrams.

7.2.2 Diagram

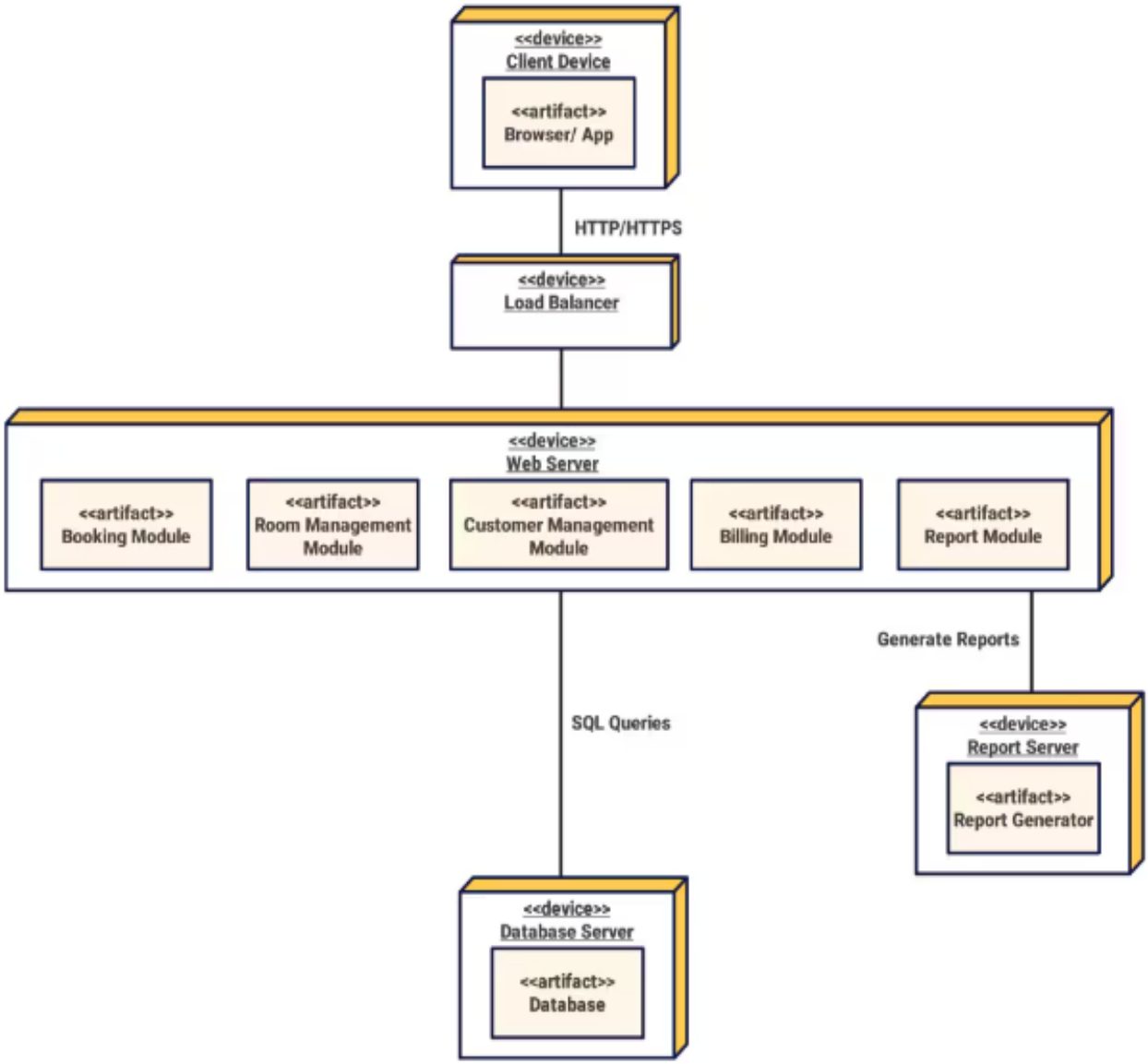

Fig.18 Deployment Diagram

7.3 Database Architecture (1- Tier, 2-Tier, 3- Tier Architecture)

7.3.1 Definition

A DBMS design is represented by database architecture. It aids in the creation, growth, use, and maintenance of the database management system. The database system may be divided into separate parts that can be independently adjusted, changed, replaced, and altered thanks to DBMS design. Understanding a database's components is also beneficial.

7.3.2 Diagram

1- Tier Architecture

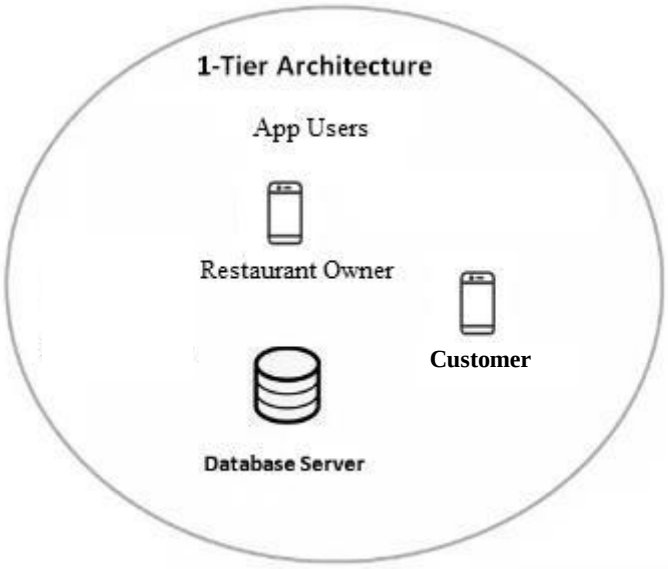

Fig.19 1-Tier Architecture

2- Tier Architecture

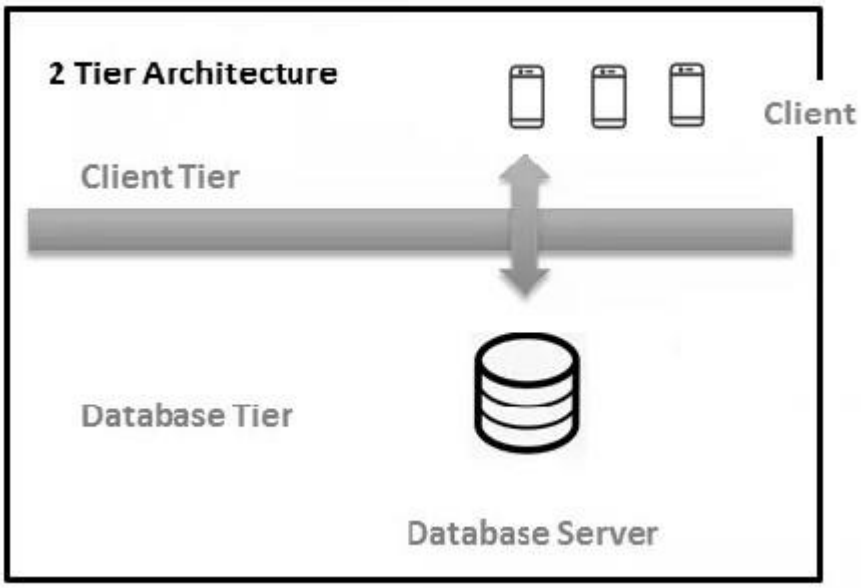

Fig.20 2-Tier Architecture

3- Tier Architecture

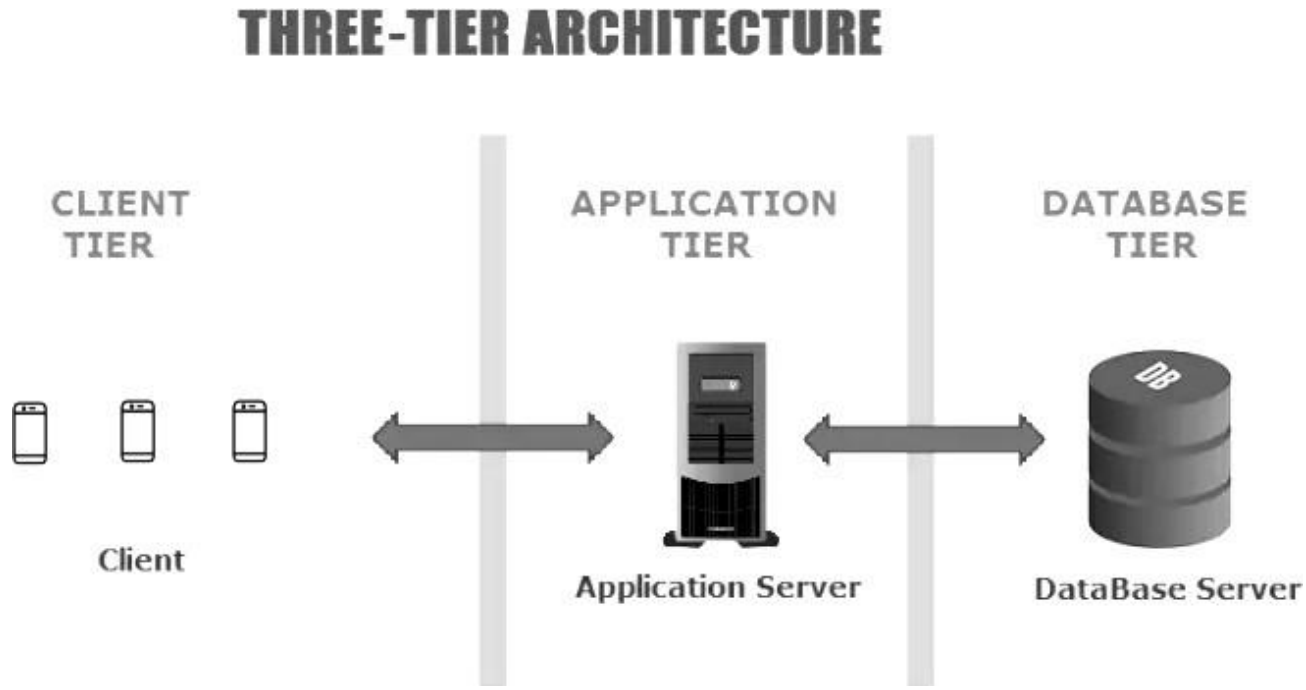

Fig.21 3-Tier Architecture

Chapter-8

Tools and Technologies

8.1 Programming Languages

The programming languages used for the development of **SelfOrder** are modern and suited for building scalable and efficient web and mobile applications.

- **JavaScript**:
  JavaScript is the core language used in both frontend and backend development for **SelfOrder**. It offers flexibility and scalability due to its large ecosystem of libraries and frameworks. As a versatile language, JavaScript allows developers to write code for the client-side, server-side, and even web applications using the same syntax.
- **React.js**:
  React.js is a JavaScript library primarily used for building user interfaces. For **SelfOrder**, React.js was chosen for the development of the web-based hotel dashboard. React provides fast rendering, component-based architecture, and easy scalability, which makes it perfect for building an interactive and high-performance web application.
- **AdonisJS**:
  AdonisJS is a Node.js web framework used for building the backend of **SelfOrder**. It is based on the MVC (Model-View-Controller) architecture and provides a rich set of tools for handling HTTP requests, data management, and authentication. The backend developed in AdonisJS is responsible for processing bookings, managing Hotel data, and handling user authentication.

8.2 Markup and Styling

- **JSX (JavaScript XML)**:
  JSX is a syntax extension of JavaScript used in React.js and React Native. It allows developers to write HTML-like code within JavaScript files, making the UI development process more intuitive and easier to maintain. JSX was used to design the interactive components in both the mobile and web applications of **SelfOrder**.
- **SCSS (Sass)**:
  SCSS is a preprocessor scripting language that is interpreted or compiled into CSS. For **SelfOrder**, SCSS was used to create modular and maintainable styles for the web application, making it easier to implement responsive designs across different devices.

8.3 Databases

- **MySQL**:
  MySQL is an open-source relational database management system used to store and manage data for **SelfOrder**. It is responsible for storing information about users, Bookings, hotel, room items, and transaction history. MySQL offers scalability and data integrity, making it an ideal choice for handling the large datasets in **SelfOrder**.

8.4 Operating Environment

The operating environment for **SelfOrder** includes both mobile and web platforms to cater to different user bases.

- **Android & iOS (React.js)**:
  The mobile application for **SelfOrder** is built using React.js, which supports both Android and iOS operating systems. This allows for a cross-platform solution where both Android and iOS users can access the application with the same codebase. By using React.js, the app can easily be updated, maintained, and scaled.
- **Web (React.js)**:
  The restaurant owners and administrators access **SelfOrder** via a web application developed with React.js. The web environment is designed to be responsive, allowing hotel owners to manage their Bookings and menus from various devices, including desktops, laptops, and tablets.

8.5 Development Tools

Several development tools were employed during the creation of **SelfOrder** to streamline the coding, testing, and deployment processes.

- **Visual Studio Code**:
  Visual Studio Code (VS Code) is an open-source code editor widely used in JavaScript development. It offers a rich ecosystem of extensions that make coding more efficient. It was used as the primary Integrated Development Environment (IDE) for coding the frontend, backend, and mobile components of **SelfOrder**.
- **Postman**:
  Postman is a collaboration platform used for API testing. During the development of **SelfOrder**, Postman was used to test the APIs created in AdonisJS, ensuring that the backend services were functioning correctly and returning the expected data.
- **Git**:
  Git is a version control system that allows for tracking changes and collaborating with multiple developers.

Throughout the development of **SelfOrder**, Git was used for version control, allowing developers to push and pull changes, resolve conflicts, and manage code efficiently.

- **GitHub**:

  GitHub is a web-based platform for hosting code repositories. The codebase for **SelfOrder** was hosted on GitHub, enabling the development team to manage the project, track issues, and collaborate seamlessly.

APPENDIX A:

USER DOCUMENTATION

User Documentation for Hotel Owner

Introduction

- Welcome to the SelfOrder Room Booking web application user documentation for hotel owners.
- This guide will help you understand how to use the app to manage your hotel and fulfill Room bookings for customers.

Getting Started

- Download and install the SelfOrder Room Booking web application from your respective app store.
- Launch the web application and sign up for a new account or log in if you already have one.
- Provide your hotel details, including name, contact information, and menu items.

Menu Management

- Add, edit, or remove menu items available for the customers.
- Include detailed descriptions, prices, and customization options for each menu item.
- Update menu availability based on stock or time restrictions.

Booking Management

- Receive and process incoming room booking from customers.
- Get notified of new bookings in real-time and review booking details including room items, quantities, and special instructions.
- Confirm or decline bookings based on availability.

Managing Special Offers and Discounts

- Create special offers or discounts for customers to attract more bookings.
- Specify terms and conditions for promotional offers.

User Documentation for Customer

Introduction

- Welcome to the SelfOrder Room Booking website user documentation.
- This guide will help you understand how to use the app as a customer to booking room .

Getting Started

- Download and install the SelfOrder Room Booking website from your respective app store.
- Launch the web app and sign up for a new account or log in if you already have one.
- Enter your personal details such as name, contact number, and email address.

Menu Selection

- Explore the available hotel and menus on the website.
- Browse through the menu items and select the desired room items by adding them to your cart.

Check-In and Enjoying Your Stay

- Once the hotel staff welcomes you at the reception, proceed with your check-in process..
- Verify your booking details and room allocation, then relax and enjoy your stay..

APPENDIX B:

References

1. Official documentation for React Native, the framework used for building the cross-platform web application for **SelfOrder**. Available at: https://reactnative.dev/docs/getting-started
2. The official documentation for React.js, used for building the web-based hotel dashboard in **SelfOrder**. Available at: https://reactjs.org/docs/getting-started.html
3. Official AdonisJS documentation, the framework used for building the backend of **SelfOrder**. Available at: https://docs.adonisjs.com
4. Documentation for MySQL, the relational database management system used to store user, hotel, and booking data for **SelfOrder**. Available at: https://dev.mysql.com/doc/
5. Firebase's official documentation, used for push notifications and real-time data synchronization in the **SelfOrder** app. Available at: https://firebase.google.com/docs
6. Heroku is used for deploying the backend services of **SelfOrder**. Official documentation on deploying applications to Heroku. Available at: https://devcenter.heroku.com/
7. Node.js is used to develop the server-side logic of **SelfOrder**. Documentation for Node.js is available at: https://nodejs.org/en/docs/
8. A comprehensive guide to learning React.js, which was used to build the web application for hotel owners in **SelfOrder**.