Hotel Booking Management System

By

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

Supervised By: Prof. Yasir Arfat

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 الرَّحْمَنِ الرَّحِيمِ
In the name of Allah, the most gracious, the most merciful

DECLARATION

This project, a result of dedicated effort and teamwork, is the outcome of tireless work by **Saira Mubeen, Maria Rasheed**, and **Bisma Rehman** under the supervision of **Prof. Yasir Arfat** (Lecturer, Computer Department) at **Govt. Graduate College Chowk Azam, GC University Faisalabad, Pakistan**.We are pleased to declare that the **" Hotel Booking Management System"** and its content are the productive results of our own hard work, academic knowledge, and research. No part of this project has been copied from any published source. This work has been conducted in the practical context of our academic studies and is not submitted for the award of any other degree or diploma.The University is entitled to take necessary action if any part of the information provided is later found to be false or misrepresented. All external sources of information used in this project, including references and contributions, have been properly acknowledged through appropriate citations and bibliographical references.This project has not been previously submitted for any examination or academic qualification at any other institution. Any guidance or support provided by faculty members or external contributors has been duly acknowledged.The software code, documentation, and other materials submitted as part of this project are original and developed by us unless stated otherwise. We take full responsibility for the authenticity and originality of the content presented in this project. We understand that any form of misrepresentation or falsification of this declaration may result in 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

We dedicate this project to God Almighty, our Creator — the ultimate source of strength, wisdom, knowledge, and inspiration. It is only through His boundless mercy and grace that we have been able to accomplish this significant milestone. His divine guidance has carried us through every challenge and triumph encountered along the way.This project, titled **"Hotel Booking Management System,"** is also dedicated to the many individuals who have supported, motivated, and encouraged us throughout this academic and developmental journey.To our families, for their unconditional love, continuous support, and understanding — your faith in us has been a constant source of strength, pushing us to persevere and succeed.To our project supervisor, **Prof. Yasir Arfat**, whose expert guidance, insightful feedback, and mentorship were instrumental in shaping this project and helping us grow both technically and personally. Your contribution has left a lasting impact.To our friends and classmates, thank you for your companionship, collaboration, and uplifting spirit. Your encouragement made this journey more enjoyable and less stressful.To the faculty and staff of **Govt. Graduate College Chowk Azam**, we are grateful for the knowledge, resources, and academic environment that empowered us to bring this project to life.To the testers, users, and stakeholders who interacted with and reviewed our system — your valuable feedback played a key role in enhancing the system's functionality, reliability, and user-friendliness.Finally, we extend our heartfelt thanks to everyone who contributed in any form — through prayers, kind words, suggestions, or support. This project is a shared achievement, and we dedicate it to all who stood by us in this rewarding endeavor.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude and heartfelt appreciation to all those who have contributed to the successful completion of our Final Year Project titled **"Hotel Booking Management System."** This journey has been both challenging and rewarding, and we are deeply thankful to the individuals and institutions that supported us throughout.First and foremost, our deepest thanks go to our respected project supervisor, **Prof. Yasir Arfat** (Lecturer, Computer Department), for his invaluable guidance, encouragement, and continuous support. His expertise and insights played a crucial role in shaping the direction of our work and helping us grow both academically and professionally.We extend our sincere thanks to the faculty members of **Govt. Graduate College Chowk Azam**, **GC University Faisalabad**, for providing us with a stimulating and supportive academic environment. Their dedication to education and the resources they provided significantly contributed to our learning and development.Our heartfelt appreciation goes to our **families and friends** for their unwavering love, encouragement, and understanding throughout this journey. Their faith in us has been our motivation, helping us stay focused and determined even during the toughest phases of the project.We are also grateful to the **participants and users** who offered their valuable feedback during the testing and evaluation stages. Their suggestions were crucial in improving the system’s functionality, usability, and effectiveness.We would like to acknowledge the **developers and contributors of open-source libraries, frameworks, and tools** that were instrumental in the development of this project. Their work significantly accelerated the development process and improved the overall quality of our application.A special thanks to the **hotel managers, staff, and stakeholders** whose needs and insights inspired the features of this system. Their real-world requirements helped us design a platform that is practical, efficient, and user-friendly.

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

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

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 digital platforms has significantly transformed the way people search for and book hotel accommodations, offering convenience, efficiency, and transparency that have become essential in the hospitality industry. The **Hotel Booking Management System** addresses the growing need for a streamlined, user-friendly platform that benefits both hotel administrators and customers. This project involves developing a comprehensive room booking management system comprising two main components: a **web application** for administrators and a **mobile application** for customers.The web application, developed using **ReactJS and MobX**, enables hotel staff to efficiently manage room listings, customer profiles, bookings, and payment records. On the other hand, the mobile application, built with **React Native and MobX**, empowers customers to browse available rooms, make reservations, view booking history, and manage their profiles with ease.The backend, powered by **AdonisJS and MySQL**, ensures secure data handling and provides seamless integration between customers, hotel staff, and payment gateways via **RESTful APIs**. By leveraging modern technologies, the Hotel Room Booking Management System aims to deliver a reliable, responsive, and scalable platform that enhances user experience, improves booking efficiency, and ensures smooth coordination across all system components. The system is capable of handling high volumes of concurrent users while maintaining data integrity and security.

Keywords
Hotel Booking, Reservation System, Web Application, ReactJS, React Native, MySQL, RESTful APIs.

Chapter 1

Introduction to the Problem

1.1 Introduction

In today's interconnected and fast-paced world, technology plays a pivsotal role in reshaping how services are delivered and consumed, especially in the hospitality industry. Traditional methods of hotel room booking and management are being rapidly replaced by digital solutions that prioritize convenience, efficiency, and transparency.The **Hotel Room Booking Management System (HRBMS)** is a comprehensive digital platform designed to simplify and modernize the hotel reservation process for both customers and hotel administrators. It is a dual-component system comprising a **web-based application** for hotel staff and administrators, and a **mobile application** for end users (guests). The system bridges the gap between guests and hotel operators by offering a seamless, real-time, and user-friendly experience for both parties.The customer-facing mobile application, developed using **React Native and MobX**, allows users to browse available rooms, view amenities, make bookings, and manage their reservation history. The administrator panel, built using **ReactJS and MobX**, enables hotel staff to manage room availability, reservations, customer profiles, and financial records with ease.The backend infrastructure, developed with **AdonisJS and MySQL**, ensures secure and scalable data handling, with **RESTful APIs** facilitating communication between frontend components and backend services. The system is built to handle high volumes of concurrent users, offering speed, reliability, and robust performance.Ultimately, the Hotel Room Booking Management System aims to revolutionize hotel operations by enhancing efficiency, boosting customer satisfaction, and eliminating the need for outdated, manual booking processes.

1.1 Background

The need for efficient and reliable hotel management systems has grown significantly as customer expectations shift towards fast, secure, and personalized service. Traditional hotel booking systems, often reliant on paper-based records or outdated software, are prone to issues such as double bookings, data errors, lack of real-time availability, and administrative delays.

In an era where mobile and online services dominate, guests expect to book accommodations instantly, receive confirmation quickly, and have full visibility into their reservations. Simultaneously, hotel operators require tools that streamline operations, manage bookings in real time, monitor financial performance, and ensure accurate communication.

The **Hotel Booking Management System** addresses these needs by offering a unified digital platform that centralizes room listings, reservation data, user interactions, and payment transactions. Built with **AdonisJS** for backend operations and **MySQL** for data management, the system ensures secure, fast, and scalable performance. RESTful APIs support real-time interaction across frontend and backend, ensuring a consistent and smooth user experience.

This modern solution replaces fragmented workflows with integrated features—improving accountability, operational efficiency, and the overall guest experience.

1.2 Purpose

The core purpose of the **Hotel Room Booking Management System** is to transform the hotel reservation experience into one that is simple, secure, and efficient—for both customers and hotel staff.

From the guest perspective, the platform aims to deliver a smooth booking experience through an intuitive mobile application. Guests can browse room options, make reservations, and manage their bookings—all without needing direct human assistance.

From the hotel administrator's side, the platform offers a powerful dashboard to handle reservations, update room availability, monitor financial transactions, and view customer activity, thereby reducing operational workload and human errors.

Key features such as secure online payment processing (e.g., JazzCash, PayPal, and other integrated gateways), real-time room availability, and automated confirmation systems are included to increase trust, efficiency, and convenience.

Ultimately, the system is designed not just as a booking tool, but as a complete hotel management solution that promotes better customer service, enhances organizational performance, and supports business growth.

1.3 Scope

The scope of the **Hotel Booking Management System** (HRBMS) includes the following modules for **Guests (Customers)** and **Hotel Administrators**:

Scope for Guests:

- **User Registration and Login:**
  Guests can register and securely log in to the system, ensuring that personal information is protected.
- **Room Browsing and Filtering:**
  Users can explore available rooms with descriptions, pricing, images, amenities, and filters (e.g., room type, price range, availability).
- **Online Booking:**
  Guests can book rooms in real-time, select check-in/check-out dates, and process payments using integrated gateways like JazzCash or PayPal.
- **Booking History:**
  A personal log of past and current bookings is available for each user, offering transparency and self-service capabilities.
- **Responsive Mobile Interface:**
  The mobile application is responsive and optimized for various screen sizes and platforms.

Scope for Administrators:

- **Admin Dashboard:**
  A centralized dashboard displaying key metrics such as total bookings, available rooms, revenue, and customer insights.
- **Room Management:**
  Add, edit, or remove room listings; update prices, availability, and features.
- **Reservation Management:**
  View, confirm, cancel, or modify guest bookings and handle overbooking scenarios.
- **User Management:**
  Manage customer records, user roles, and access permissions.
- **Reports and Analytics:**
  Generate reports based on date, room category, occupancy rates, and revenue to assist in decision-making and performance tracking.

Technical Scope:

- **Backend Development (AdonisJS):**
  Provides the server-side logic, authentication, and data processing.
- **Database Integration (MySQL):**
  Ensures efficient storage and retrieval of user, booking, and transaction data.
- **RESTful APIs:**
  Manage communication between frontend and backend, promoting modularity and scalability.
- **Security:**
  Strong input validation, role-based access control, encrypted transactions, and secure login mechanisms are implemented throughout the system.

1.4 Objective

Functional Objectives:

- **User-Friendly Interface:**
  Design a platform that is simple to navigate, visually appealing, and intuitive across devices.
- **Real-Time Room Booking:**
  Enable users to view availability and book rooms instantly, with real-time updates on room status.
- **Integrated Payment Gateways:**
  Support secure, multi-option payment processing for a smooth checkout experience.
- **Transaction Confirmation:**
  Provide instant notifications and confirmations upon successful booking and payment.

Technical Objectives:

- **Scalable Backend:**
  Build a modular backend using AdonisJS and MVC architecture to handle growing user bases and data loads.
- **Efficient Data Storage:**
  Use MySQL to manage room listings, user profiles, and transaction history securely.

- **Responsive Frontend:**
  Ensure seamless user interaction across mobile and desktop platforms.
- **Secure APIs:**
  Leverage RESTful APIs for safe, modular communication between frontend and backend systems.
- **Enhanced Security:**
  Implement encryption, secure login, and access control to protect user and financial data.

1.5 Intended Audience and Reading Suggestions

Audience & Purpose:

- **Developers:**
  Understand system architecture, API integrations, database schemas, and coding standards.
- **Hotel Administrators/Managers:**
  Learn to use the admin dashboard for managing bookings, room availability, and customer data.
- **Guests/Customers:**
  Find guidance on browsing, booking, and managing room reservations through the mobile app.
- **Project Managers/Stakeholders:**
  Evaluate system functionality, deployment status, and business alignment.

Reading Suggestions:

- **Start with the Introduction and Features** to get an overview of system capabilities.
- **Developers** should move to the **System Design, API, and Implementation** sections.
- **Hotel Staff/Admins** should focus on the **Admin Panel and Reservation Management** guides.
- **Guests** can refer to the **Mobile Interface and Booking Process** sections.
- **Stakeholders** may find the **Reports, Analytics, and Performance Metrics** sections most relevant.

1.6 Process Model

The **Hotel Room Booking Management System** follows the **Rapid Application Development (RAD)** model for quick iterations and continuous feedback.

RAD Phases:

- **Requirements Planning:**
  Interviews and meetings with hotel managers and guests to identify core functionalities such as room booking, real-time availability, and payment processing.
- **User Design:**
  Prototypes and wireframes created for mobile and admin interfaces; feedback used to improve UX before development.
- **Construction:**
  Simultaneous frontend and backend development using ReactJS/React Native, AdonisJS, and RESTful APIs. Regular testing ensures stability and alignment with user needs.

- **Cutover:**
  Deployment of the system to production, database migration, final QA, and training for hotel staff. Security audit and performance checks are also completed.

1.7 Document Conventions

Element	**Convention**
Headings	Bold and numbered hierarchically (e.g., 1.1, 1.2)
Code/Commands	Monospaced font (e.g., `npm install`)
Images/Diagrams	Labeled as “Figure X.X” with captions
Tables	Clearly labeled, centered, and structured
References	Listed in footnotes or at the end of the document
Navigation	Internal hyperlinks for electronic viewing
Terminology	Consistent use of terms like "guest," "admin," "room," "reservation"

Chapter-2

Software Requirement Specification

2.1 Overall Description

2.1.1 Product Perspective

The **Hotel Booking Management System (HRBMS)** is a web and mobile-based application designed to streamline hotel reservations, room management, and customer service. It modernizes the traditional booking process by offering separate, dedicated interfaces for guests and hotel administrators. Guests can register, browse available rooms, book accommodations, and receive booking confirmations. Administrators manage room listings, monitor reservations, process check-ins/check-outs, and generate reports.

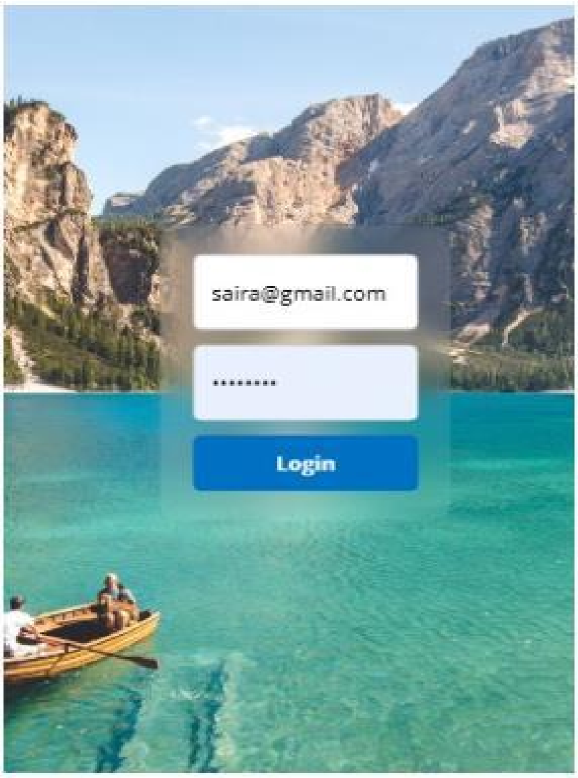
Unlike traditional paper-based or standalone systems, HRBMS is an integrated solution that ensures efficient handling of real-time room availability, secure online payments, customer data management, and automated notifications.

2.1.2 Product Features

The system is divided into two main modules:

Admin Module:

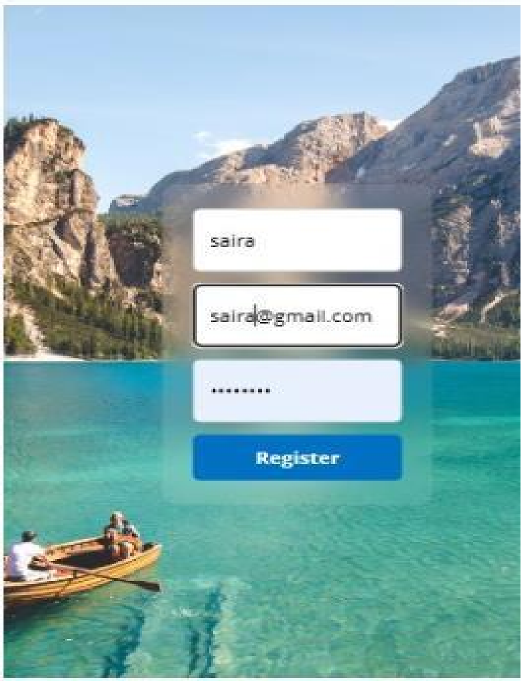
- **Login**



- **Room Management**
  - Add/Update/Delete/View room listings
  - Enable/Disable room types or services
- **Booking Management**
  - View all current bookings
  - Confirm, cancel, or reschedule reservations
  - Update booking status (confirmed, checked-in, completed)
- **Customer Management**
  - View customer profiles
  - Handle customer feedback and complaints
- **Reports and Analytics**
  - Filter bookings by date, room type, customer
  - Generate occupancy and revenue reports

Guest Module:

- **Register/Login**


saira
saira@gmail.com
••••••••

- **Profile Management**
  - Update contact, identity, and preferences
- **Browse Rooms**
  - Filter by price, type, availability, services
- **Add to Cart / Book Room**
  - Select rooms, choose dates, and confirm booking
- **Booking History and Tracking**
  - View current and past bookings
- **Notifications**

    - Email/SMS alerts for booking confirmation, reminders, or changes

2.1.3 Design and Implementation Constraints

- **Technology Stack:**
    - **Frontend:** ReactJS (Web), React Native (Mobile)
    - **Backend:** Node.js, Express.js
    - **Database:** MySQL
    - **APIs:** RESTful APIs for frontend-backend communication
- **Platform Requirements:**
    - Mobile app compatible with Android and iOS
    - Web app supported on modern browsers (Chrome, Firefox, Safari)
- **Performance:**
    - Must handle multiple concurrent bookings during peak seasons
    - Quick response times on all devices

2.1.4 Assumptions and Dependencies

- **Assumptions:**
    - Guests have access to the internet and smart devices
    - Hotel staff is trained to use the admin dashboard
    - Hotels have stable internet and devices to run the system
- **Dependencies:**
    - Compliance with hospitality and data protection laws
    - Dependence on third-party services for hosting and payment gateways

2.2 System Features

2.2.1 Guest Registration and Login

Description & Priority:
Users must create an account or log in to manage bookings and access personalized services.

Functional Requirements:

- **Input:** Name, email, password
- **Output:** Access to guest dashboard

2.2.2 Admin Registration and Login

Description & Priority:
Hotel administrators access the management dashboard via secure login.

Functional Requirements:

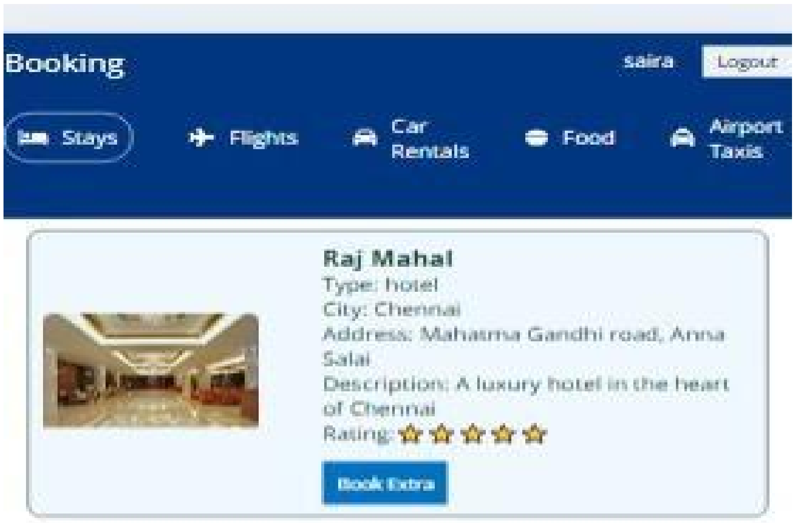
- **Input:** Admin credentials (name, email, password)
- **Output:** Access to admin dashboard

2.2.3 Add Room Booking to Cart

Description & Priority:
Guests select available rooms and add them to a cart before confirming booking.

Functional Requirements:

- **Input:** Room type, check-in/check-out dates, number of guests
- **Output:** Booking summary with total cost



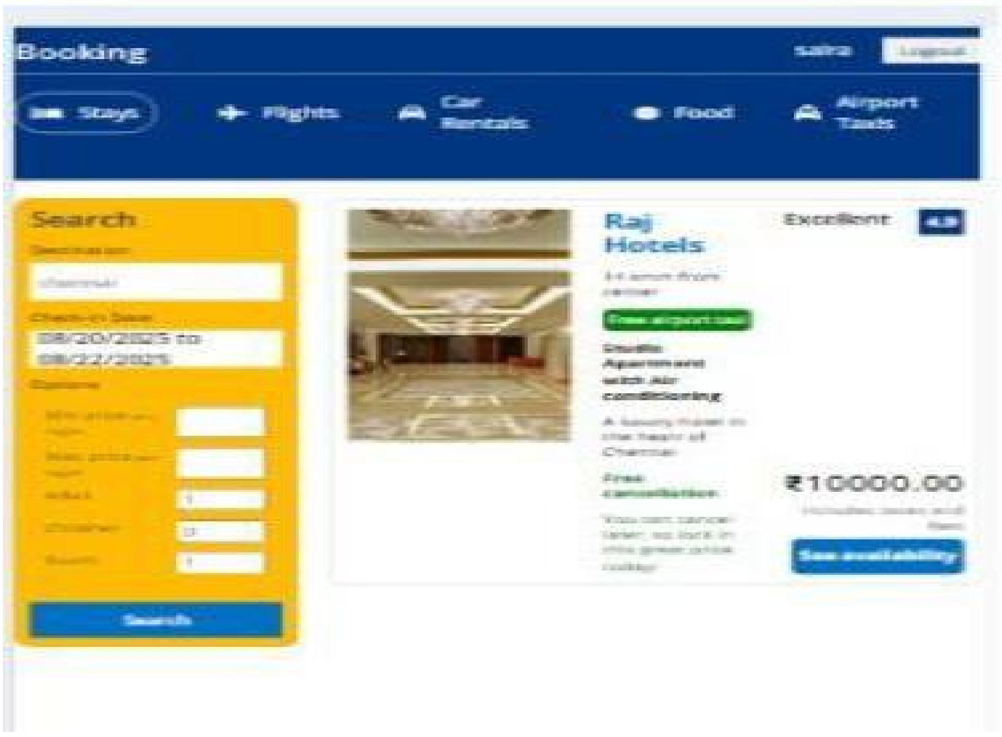
Raj Mahal
Type: hotel
City: Chennai
Address: Mahasma Gandhi road, Anna Salai
Description: A luxury hotel in the heart of Chennai
Rating: ★★★★★

2.2.4 View Booking Details

Description & Priority:
Guests and admins can view the status and details of individual bookings.

Functional Requirements:

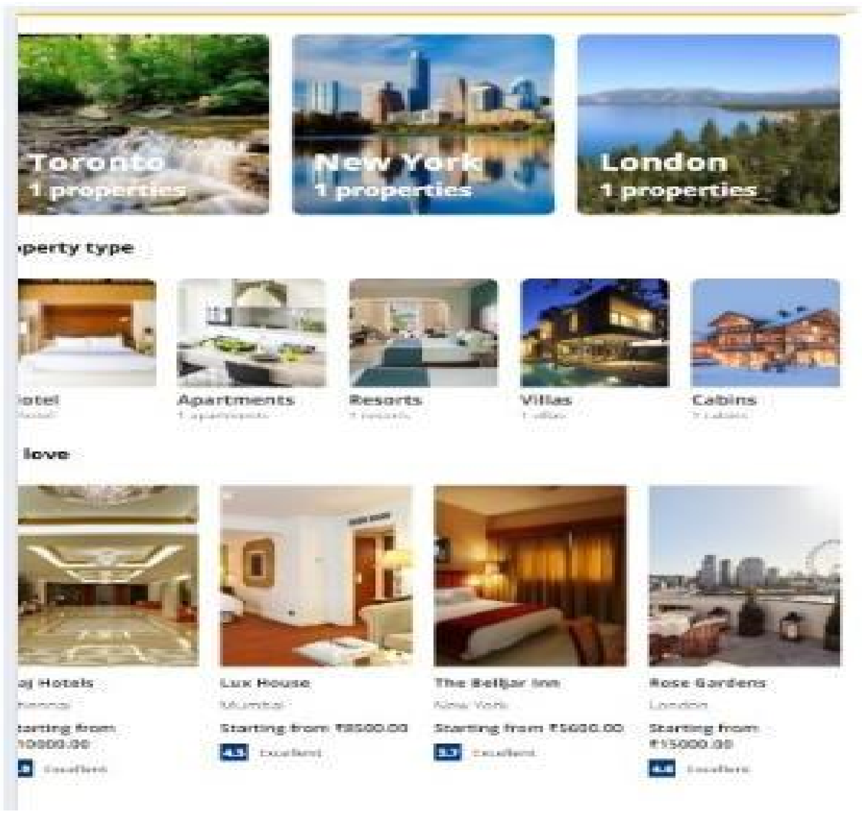
- **Input:** Booking ID or user ID
- **Output:** Booking status, room information, payment status



2.3 External Interface Requirements

2.3.1 User Interfaces

- Guest Dashboard
- Admin Dashboard
- Login/Registration Screens
- Room Listing and Booking Pages
- Booking Confirmation and Tracking Pages



2.3.2 Hardware Interfaces

- **Mobile Devices:** Android/iOS, minimum 4GB RAM
- **Web App Devices:** Desktop/laptops, 8GB RAM, modern browser, minimum 128GB storage

2.3.3 Software Interfaces

- **Operating Systems:** Android, iOS, Windows/macOS
- **Development Tools:** Android Studio (Mobile), VS Code (Web)
- **Database:** MySQL
- **Backend Framework:** Node.js with Express.js

2.3.4 Communication Interfaces

- **Internet:** Required for booking and management features
- **Push Notifications:** Booking status, promotions
- **Email Notifications:** Booking confirmations, password resets, reminders

2.4 Other Nonfunctional Requirements

2.4.1 Performance Requirements

- Real-time room availability updates
- 24/7 platform uptime
- Quick booking and payment processing
- User-friendly and responsive design

2.4.2 Safety Requirements

- All guest and staff data is encrypted and securely stored
- No sharing of personal data with third parties
- System adheres to hotel industry safety standards and data policies

2.4.3 Security Requirements

- First-time password reset required
- End-to-end encryption of sensitive data
- Secure authentication (JWT, OAuth2)
- Strong password enforcement and regular security audits
- Full compliance with Pakistan’s Personal Data Protection Act (PDPA)

Chapter 3:

Analysis (Use Case Model)

1.1 Identifying Actors and Use Cases Using Textual Analysis

1.1.1 Use Case Name: Room booking

DC#	**1**
UC Name	Signup as Room booking
Level	Detailed
Description	Create a booking profile.
Actor	Room booking
Stakeholders	Room booking
Preconditions	New account is created.
Success Guarantee	Signup successful.

Main Success Scenario Action:

- The recipient fills out the registration form with:
- Full Name (e.g., Ali Khan)
- Contact Information (Phone: +92 300 1234567, Email: ali@example.com)
- Address (House #123, Street 5, Islamabad)
- Type of Aid Needed (funds, etc.)
- Urgency Level (Low/Medium/High)

Response:

- The system validates the entered details.
- If all data is correct, it creates a new recipient account.

Displays a success message:

- ✅"Registration Successful! You can now request aid."
- Extensions (Alternative Flows)

Case 1: Missing Required Fields

- **System Response:**

1) ❌"Please fill all mandatory fields (*) to proceed."

- **User Action:**

1) The recipient corrects the missing details and resubmits.

Case 2: Invalid Contact Information

- **System Response**:

1) ❌"Invalid phone/email format. Please correct it."

- **User Action**:

1) The recipient updates the contact details.

Special Requirements

- The Hotel Room booking Management System (HRBMS) must be online and accessible 24/7.
- Recipients must have a stable internet connection to submit requests.
- Data privacy compliance (e.g., GDPR/PDPA) must be ensured.
- Frequency of Occurrence
- High – Occurs daily as new recipients register when in need.

Additional Enhancements (Optional)

- OTP Verification (For Security)
- After registration, the system sends an SMS/Email OTP for account activation.
- Automated Needs Assessment

1.1.2 Use Case Name: Recipient

DC#	3
UC Name	Signup as Recipient
DC#	3

Level	Detailed
Description	Create recipient profile for aid requests.
Actor	Recipient
Stakeholders	Recipient, Room booking System
Preconditions	New account required.
Success Guarantee	Recipient profile create

Recipients can specify:

- Family size (to estimate aid quantity).
- Preferred aid distribution method (Pickup/Delivery).
- Multilingual Support
- Registration form available in Urdu/English for wider accessibility.

1.2 Use Case Diagrams

1.2.1 Use Case Diagram: Room Booking

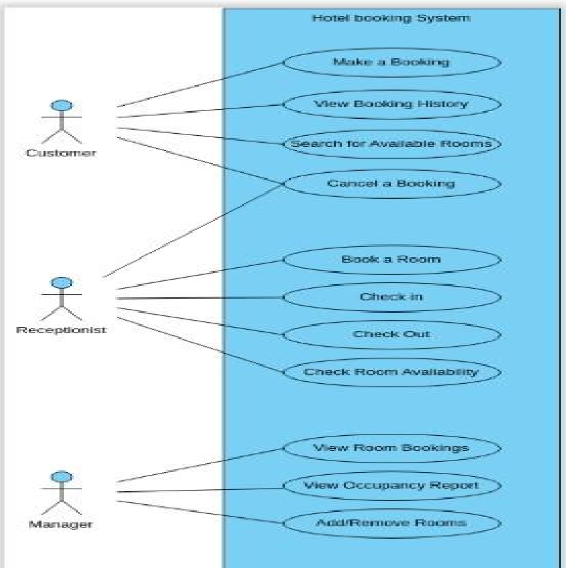


Fig.7 Room booking

Key Use Cases Explained:

1. **Register**

- New Room booking create accounts by providing basic details.
- System validates and confirms via email/SMS.

2. **Login**

Existing Room booking access accounts using credentials.

- System authenticates and grants dashboard access.

3. **Room booking Items**

- Room booking specify Room booking type (cash/items) and details.
- System processes and generates receipts.

4. **Track Room booking**

- Room booking view real-time status (Pending/Received).
- Filterable by date and donation type.

5. **Update Profill**

- Edit contact info or preferences.
- Password changes require verification.

6. **View History**

- See all past Room booking in summary.
- Export options available for records.

1.2.2 Use Case Diagram: Hotel Booking

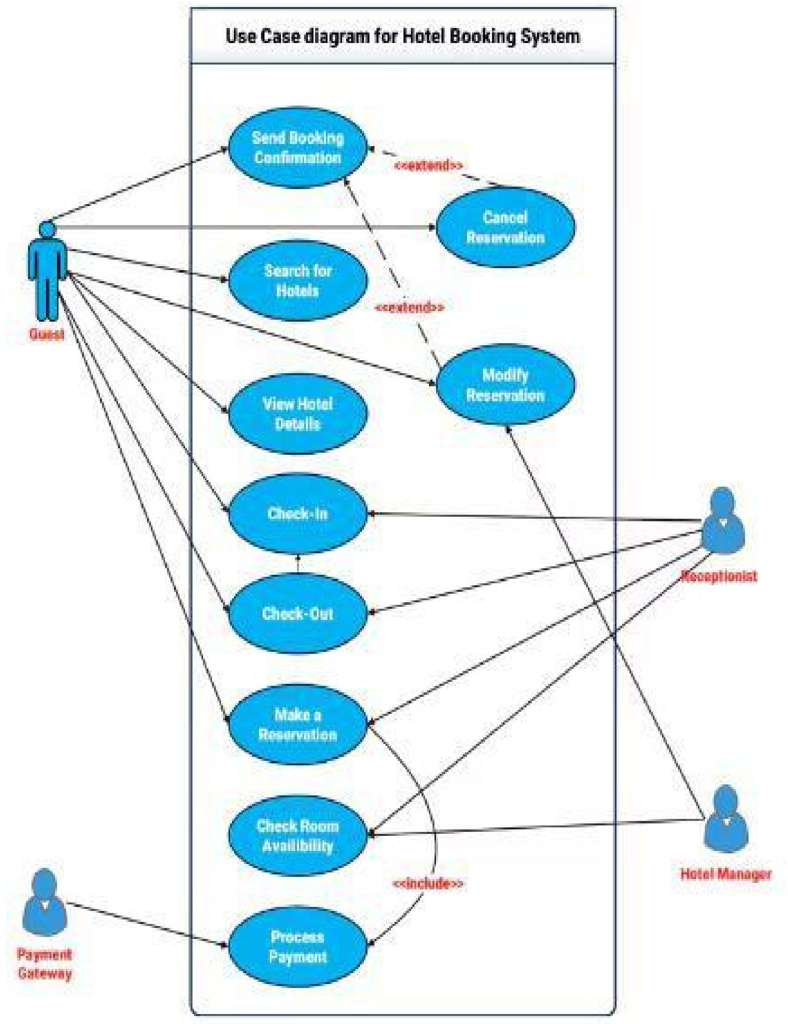


Fig.8 Hotel Booking

1.3 Event Flow for Use Cases

1.3.1 Room booking Registration Precondition: Donor must sign up. **Main Flow:**

- Enter full name.
- Provide contact details (email/phone).

- Specify donation types (funds/etc.).
- Submit.

1.3.2 Guest Registration

Precondition: Guest must sign up. **Main Flow:**

- Enter full name.
- Add contact info.
- Describe needs (urgent/regular).
- Submit.

Chapter-4

Design

4.1 Architecture Diagram

This document outlines the architecture of a **Hotel Room Booking Management System**, detailing its major components and the flow of interaction between them. The system is designed to allow guests to book rooms in hotels seamlessly while giving hotel administrators full control over room availability, booking status, and user management. The architecture emphasizes usability, scalability, and security.

Diagram

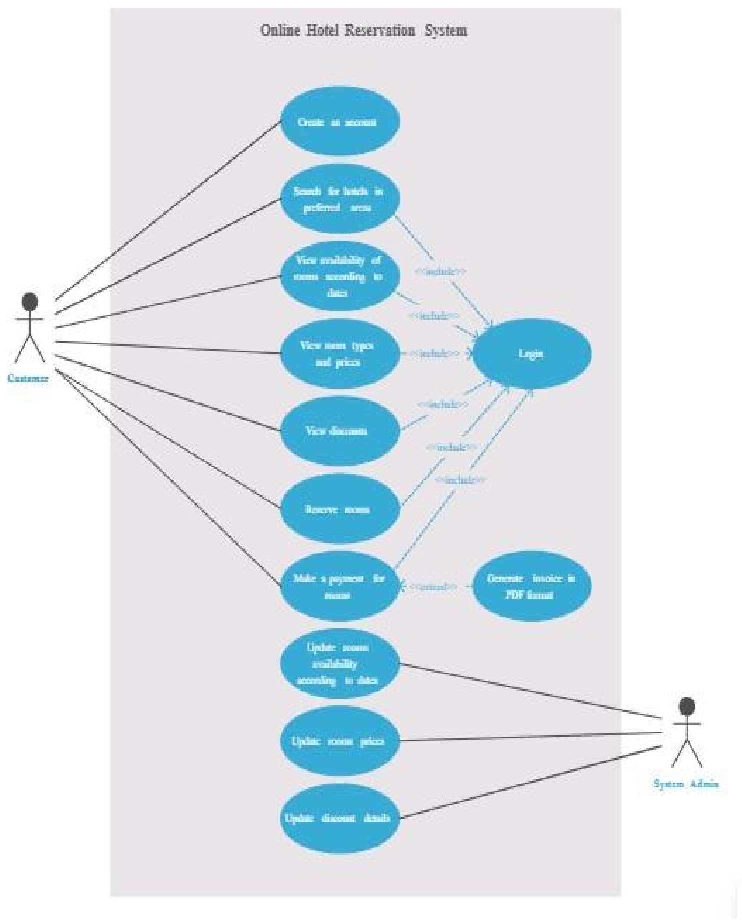

Online Hotel Reservation System
Create an account
Search for hotels in preferred areas
View availability of rooms according to dates
View room types and prices
View discounts
Reserve rooms
Make a payment for rooms
Generate invoice in PDF format
Update rooms availability according to dates
Update rooms prices
Update discount details
Login
<<include>>
<<extend>>
Customer
System Admin

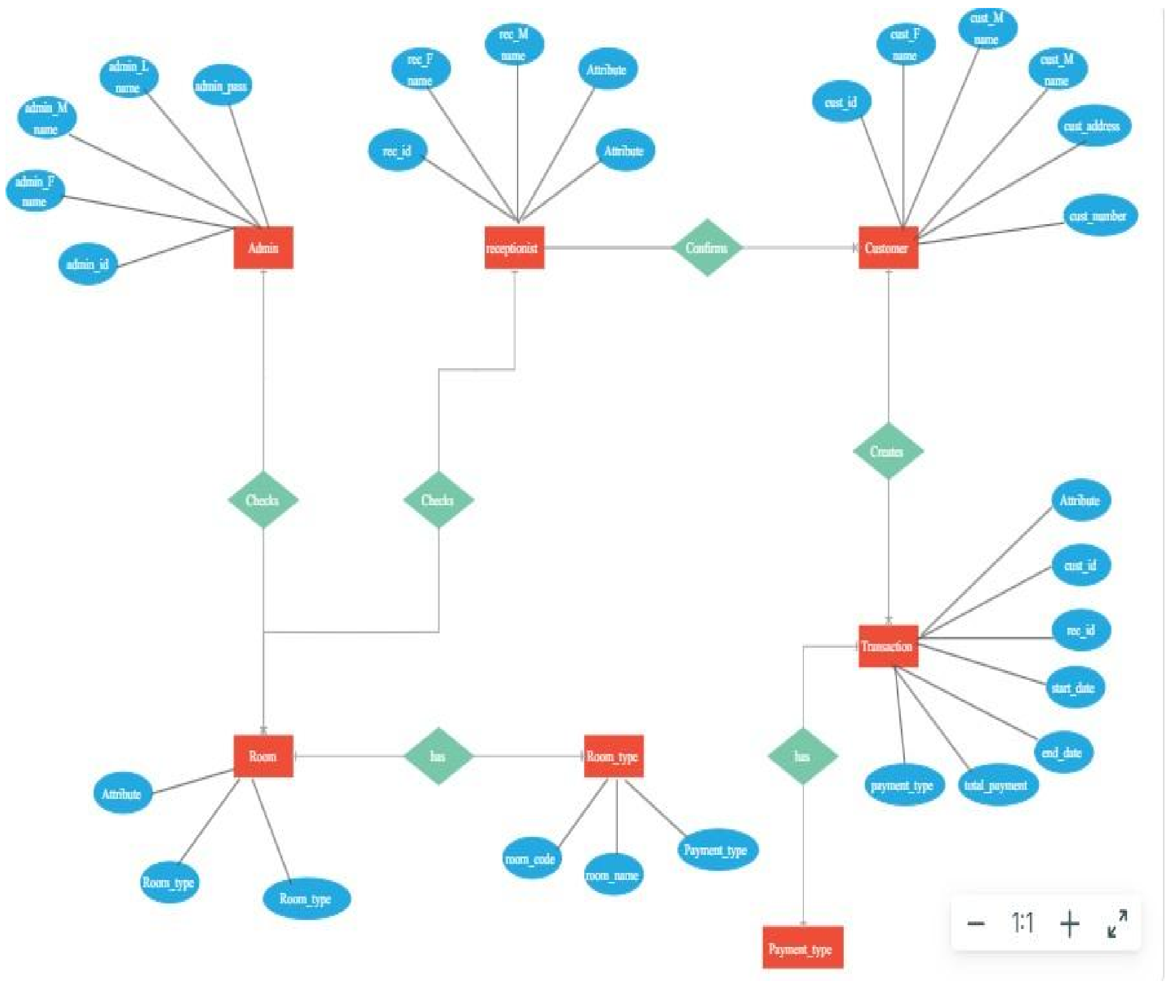
Fig.9Architecture Diagram

4.2 ER Diagram with data dictionary

4.1.1 Definition:

A **Class Diagram** in UML (Unified Modeling Language) illustrates the **classes**, their **attributes**, **methods (operations)**, and the **relationships** between different objects in a system. In the context of the **Hotel Room Booking Management System**, the key entities (classes) revolve around users (Guests and Admins), rooms, bookings, and payments.

Fig.10 ER Diagram



4.2 Class Diagram

4.2.1 Definition

A **Class Diagram** is a UML (Unified Modeling Language) representation of the system's structure, depicting the classes, their attributes, methods (operations), and the relationships among them..

4.2.2 Class Diagram

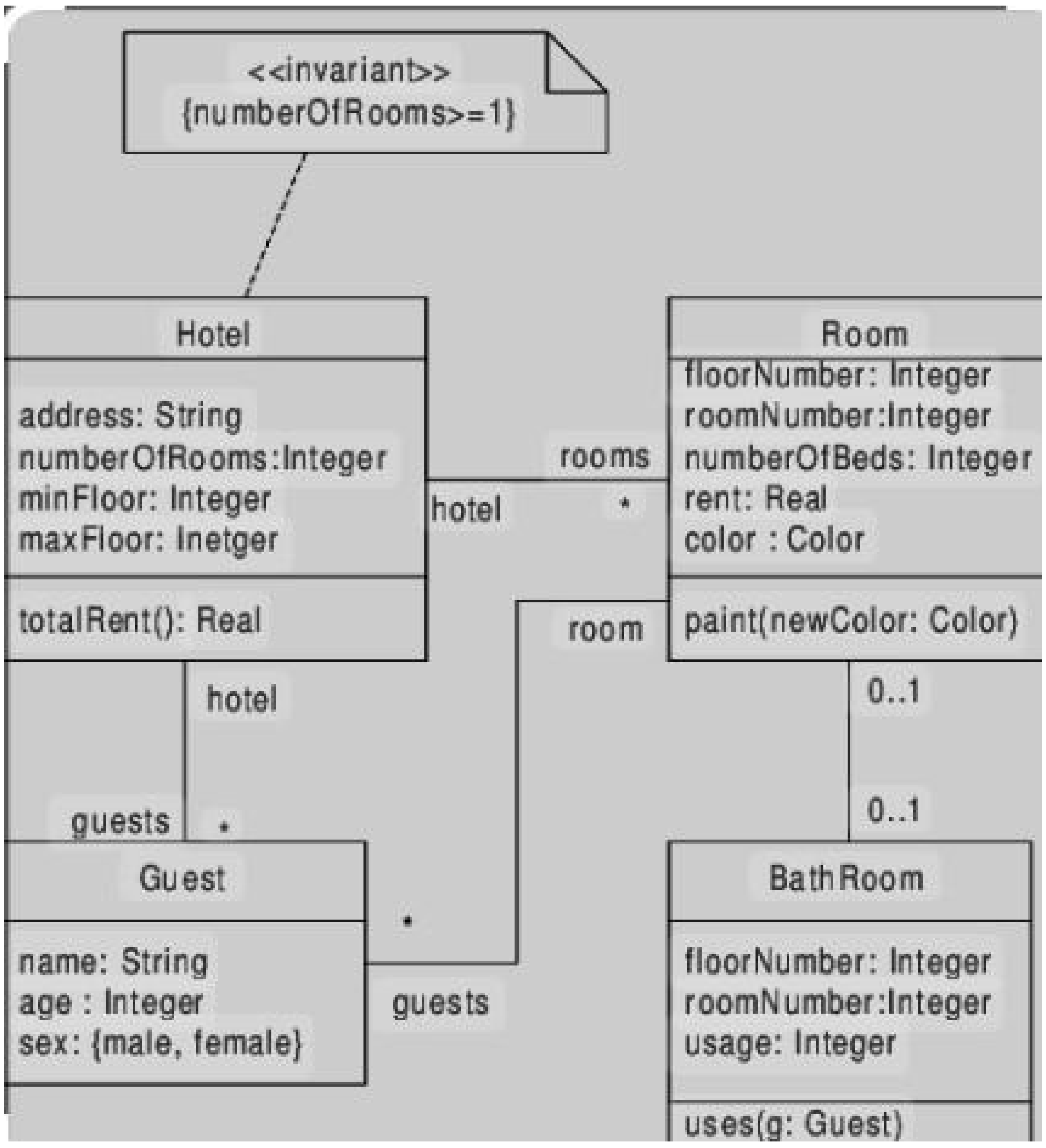


Fig.11 Class Diagram

4.3 Object Diagram

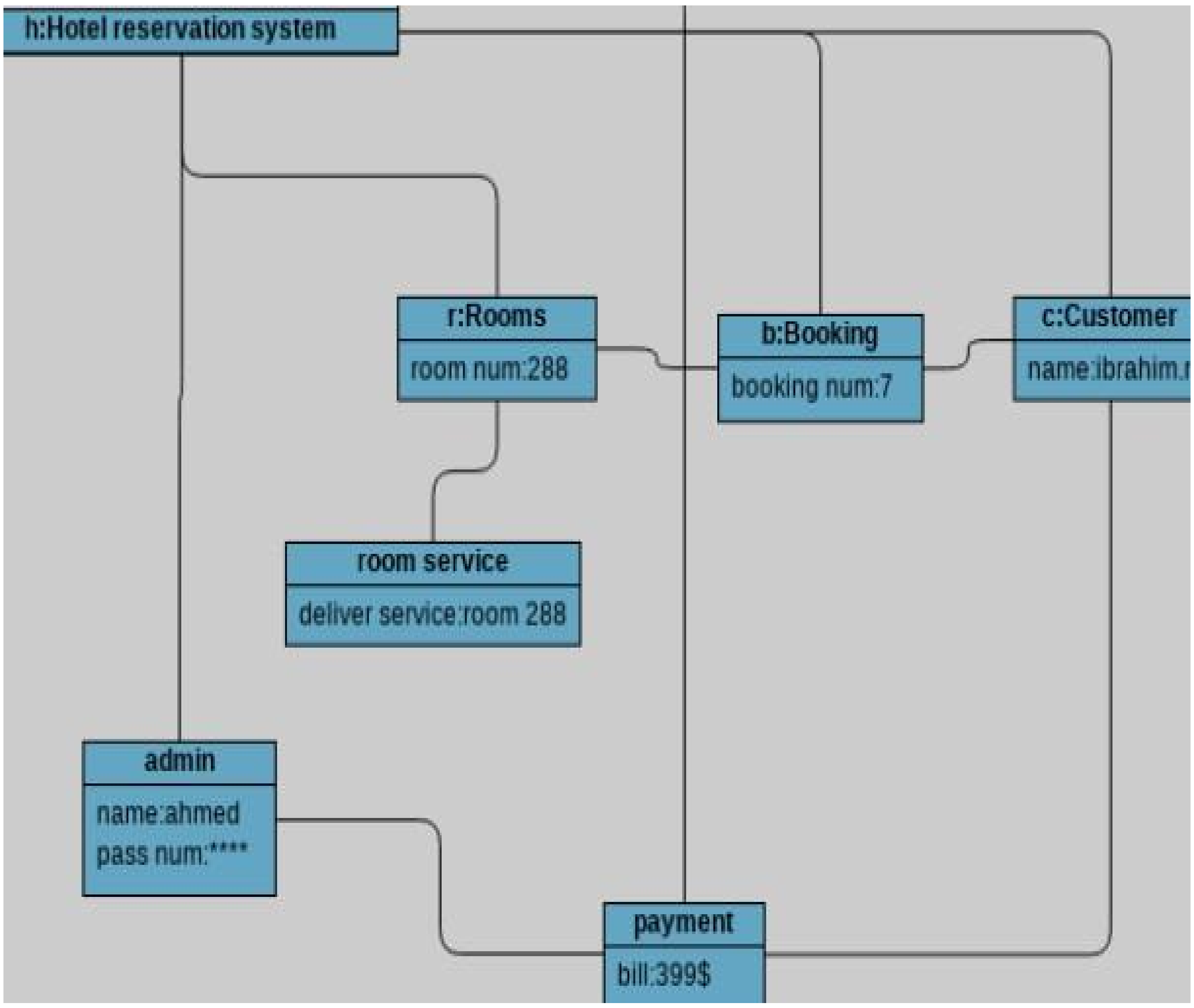
4.3.1 Definition

An Object Diagram **is a snapshot of a system's runtime state, showing the instances of**

classes (objects) and their relationships at a specific moment in time.

4.3.2 Object Diagram

Object Diagram (Fig. 12)



4.4 Sequence Diagram

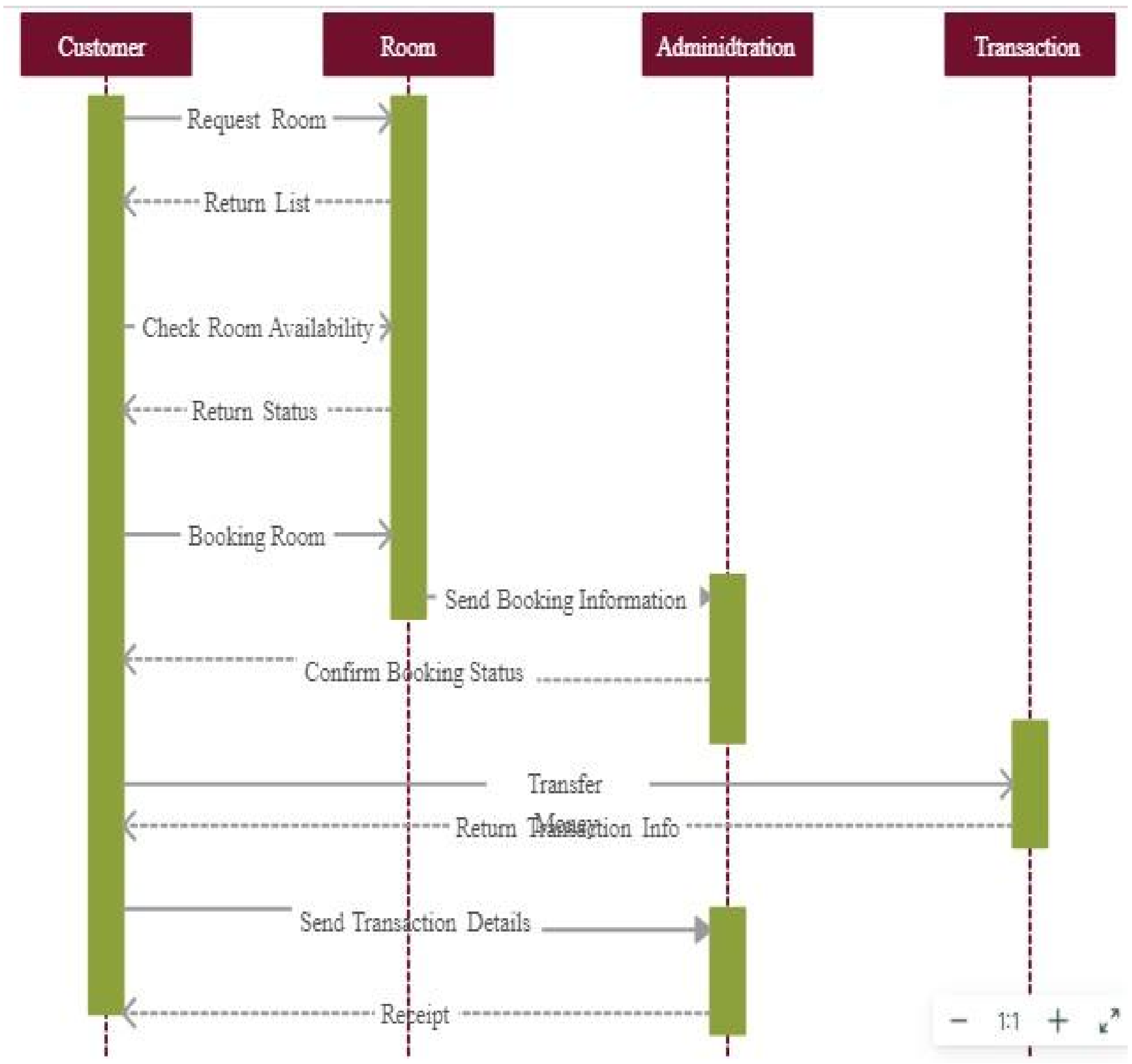
4.4.1 Definition

A **Sequence Diagram** illustrates the dynamic interactions between objects in a system, representing the chronological order of messages exchanged during a specific use-case scenario.

In the context of the **Hotel Room Booking Management System**, this diagram captures the complete flow from **guest login and room search to booking confirmation and payment processing**, including **admin validation** and **room status updates**.

4.4.2 Sequence Diagram

Sequence Diagram (Fig. 13)



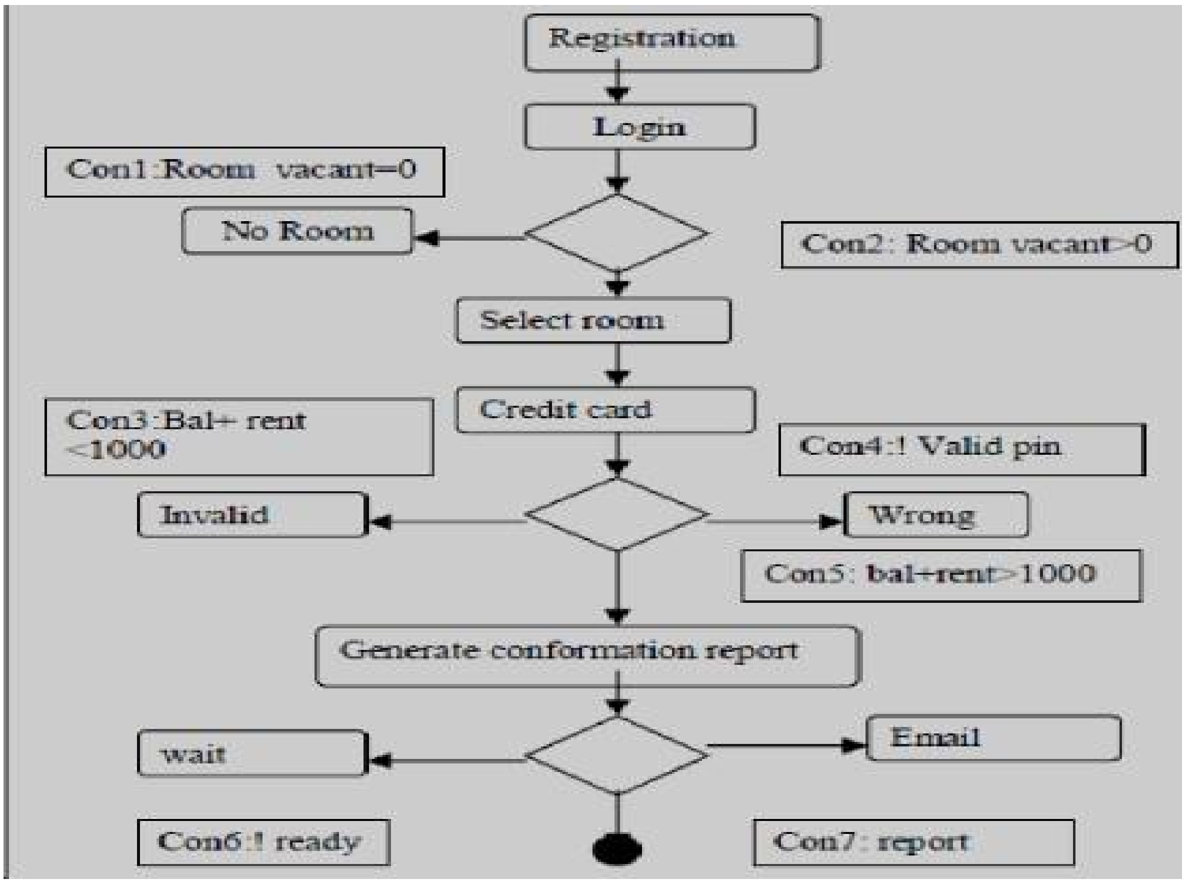
4.5 Activity Diagram

4.5.1 Definition

An Activity Diagram represents the workflow of activities and actions in a system, showing the control flow from one activity to another. For the Room Booking Management System, it illustrates the complete process from donor registration to donation fulfillment, including admin verification and recipient allocation.

4.5.2 Activity Diagram (Fig. 14)

Activity Diagram (Fig. 14)

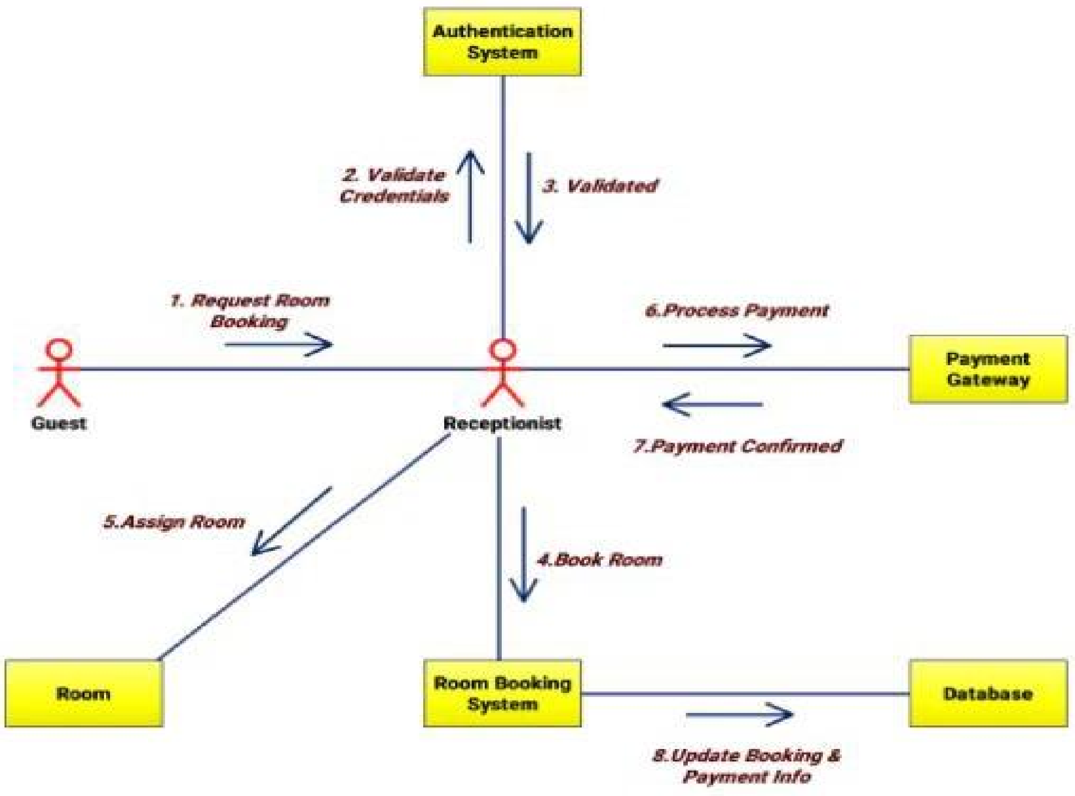


4.6 Collaboration Diagram

4.6.1 Definition

A Collaboration Diagram (Communication Diagram) demonstrates how objects interact to perform specific use cases by showing message exchanges and relationships. For the Donation Management System, it visualizes the interactions between Room Booking, admins, recipients, and payment systems during the Room Booking process.

4.6.2 Collaboration Diagram



Collaboration Diagram (Fig. 15)

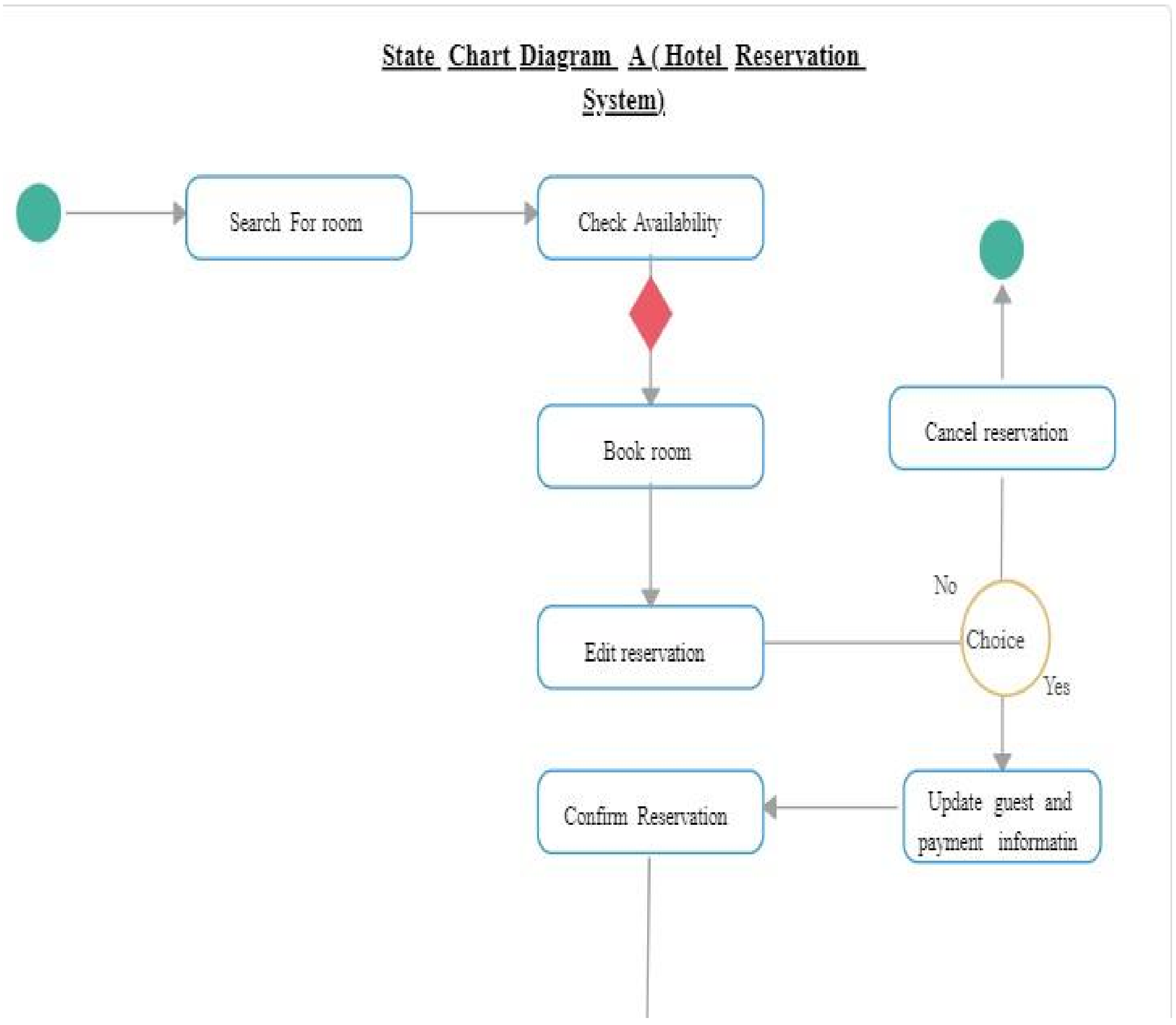
4.7 State Transition Diagram

4.7.1 Definition

A State Transition Diagram models the lifecycle of an object by showing its various states and the events that cause transitions between them. For the Room Booking Management System, this diagram illustrates how a donation progresses from creation to completion, including all possible states (e.g., Pending, Verified, Failed) and triggering events (e.g., admin approval, payment processing).

4.7.2 State Transition Diagram

State Transition Diagram (Fig. 16



Chapter-5

Development

5.1 Operating System

When developing the Fund Room Booking Management System, several operating systems can be considered based on the target audience and devices used.

5.1.1 Available Operating System

Here are some commonly used OS options for developing the system:

- **Android:**
  Android is an open-source mobile OS widely used in smartphones and tablets. Since many Room Booking and administrators may use Android devices, developing an Android app ensures accessibility for a large user base. It provides a flexible development environment with extensive tools and libraries.

- **iOS:**
  Apple's iOS is popular among iPhone and iPad users. Developing an iOS version of the app would help reach donors who prefer Apple devices. However, it requires adherence to Apple's strict guidelines and the use of Swift or Objective- C.

- **Web-based:**
  A web-based application ensures platform independence, allowing Room Booking and administrators to access the system from any device (desktops, laptops, tablets, or smartphones) via a browser. This approach eliminates OS restrictions and simplifies maintenance.

- **Cross-platform frameworks:**
  Frameworks like React Native and Flutter allow the development of a single codebase for both Android and iOS, reducing development time and cost while ensuring consistency across platforms.

5.1.2 Selected Operating System

- **Android:**
  The Android OS is selected for the mobile application due to its widespread adoption, especially in regions where fund Room Booking campaigns are highly active. This ensures maximum reach among potential Room Booking.

- **Web-based:**
  For the administrative dashboard and Room Booking portal, a web-based approach is chosen. This allows NGOs, charities, and administrators to access the system from any device with a browser, ensuring flexibility and ease of use. Additionally, responsive design ensures compatibility across different screen sizes.

5.2 Development Approach

The development approach refers to the methodology or strategy followed during the software development process. Different approaches provide guidelines on planning, designing, developing, and delivering software projects effectively.

5.2.1 Available Development Approach

Common development approaches include:

- **Waterfall:**
  A sequential, linear process with distinct phases (requirements, design, development, testing, deployment). Changes after a phase is completed are difficult to implement.
- **Agile:**
  An iterative, flexible approach emphasizing collaboration, adaptability, and user feedback. Work is divided into sprints, each delivering a functional increment. Includes Scrum, Kanban, and XP.
- **Spiral:**
  Combines waterfall and iterative methods, focusing on risk management. Each iteration involves prototyping, user feedback, and refinement.
- **Rapid Application Development (RAD):**
  Prioritizes quick prototyping and iterations. Developers and stakeholders collaborate closely to refine requirements and deliver fast solutions.
- **DevOps:**
  Integrates development and operations teams for continuous integration/delivery (CI/CD). Ensures faster, reliable releases through automation.
- **Lean:**
  Focuses on eliminating waste (e.g., unnecessary code, delays) and maximizing value for end-users. Emphasizes efficiency and continuous improvement.
- **Incremental:**
  Breaks the project into standalone modules delivered incrementally. Allows early feature releases and iterative feedback.

5.2.2 Selected Development Approach

For the Fund Room Booking Management System, the Agile methodology (Scrum) is selected due to its flexibility, stakeholder collaboration, and adaptability to changing Room Booking /administrator needs.

- **Phases:**

1. Requirements Gathering:
2. Identified functional needs:
3. Donor portal (web) for campaigns, Room Booking, and tracking.
4. Admin dashboard for NGOs (Room Booking analytics, user management, reporting).
5. Payment gateway integration (Stripe, PayPal).

- **System Design:**

1. Frontend: React.js (web admin panel)
2. Backend: Node.js/Express.js for scalable APIs.
3. Database: MongoDB (NoSQL for flexible data models).
4. Security: JWT authentication, encryption for sensitive data.

- **Implementation:**

1. Web app (React.js) for NGOs to manage funds, generate reports.
2. APIs for real-time Room Booking tracking, notifications, and payment processing.

- **Testing:**

1. Unit/Integration Testing: Jest, Mocha.
2. User Acceptance Testing (UAT): Feedback from NGOs and donors.
3. Security Testing: Penetration testing for payment systems.

- **Deployment:**

1. Web app hosted on AWS/Heroku for global accessibility.

- **Operation & Maintenance:**

1. Regular updates (new features, compliance with payment regulations).
2. Performance optimization and bug fixes based on user feedback.

5.3 Programming Language

A programming language is a formal language used to write instructions for computers to perform tasks. It enables developers to build software applications, scripts, and algorithms efficiently.

5.3.1 Available Programming Language

- **Java:**
  A robust, object-oriented language suitable for Android development. Known for portability and strong community support.
- **Kotlin:**
  A modern alternative to Java for Android, with concise syntax and null-safety features. Officially supported by Google.
- **JavaScript:**
  A versatile language for web/mobile development (frontend and backend). Essential for dynamic, cross-platform applications.
- **TypeScript:**
  A typed superset of JavaScript that improves code maintainability, ideal for large- scale projects like donation systems.
- **Python:**
  Used for backend logic, data analytics (Room Booking trends), and scripting due to its simplicity and rich libraries.

5.3.2 Selected Programming Language

JavaScript (React.js):

- React.js for the NGO/admin web portal (dynamic UI, real-time updates).
- Node.js (Express.js/Adonis.js for Backend):
- Handles APIs, authentication, and payment processing.

- Adonis.js provides structured backend development with built-in security features.

5.4 Platform

5.4.1 Available Platform

- **AdonisJS:**
  Node.js backend framework for scalable APIs and database management.
- **ReactJS:**
  Library for building responsive web interfaces (admin dashboards).
- **Firebase:**
  For real-time Room Booking tracking, notifications, and authentication.

5.4.2 Selected Platform

- **React.js:**
  For the NGO web portal (real-time analytics, campaign management).
- **Adonis.js (Node.js):**
  Backend APIs for Room Booking , user roles, and reporting.
- **Firebase:**
  Additional backend support for push notifications (Room Booking receipts) and cloud functions.

5.4 CASE Tools

CASE (Computer-Aided Software Engineering) tools are software applications that assist in various stages of the software development lifecycle (SDLC), including analysis, design, coding, testing, and maintenance. These tools help improve efficiency, collaboration, and documentation in software projects.

5.4.1 Available CASE Tools

For the Fund Room Booking Management System (web-based), the following CASE tools can be utilized:

- **Unified Modeling Language (UML) Tools**

1. Used for visualizing system architecture, workflows, and database design.
2. Helps in creating use case diagrams, class diagrams, and sequence diagrams for better planning.
3. Examples: Lucidchart, Visual Paradigm, Draw.io

- **Integrated Development Environments (IDEs)**

1. Essential for writing, debugging, and testing the web application code.
2. Examples:

➢ Visual Studio Code (VS Code) – Lightweight, supports JavaScript/TypeScript, React.js, and Node.js.

➢ WebStorm – Advanced IDE for JavaScript-based web development.

- **Version Control Systems (VCS)**

1. Helps manage source code changes, collaboration, and version history.
2. **2.** Git (with platforms like GitHub, GitLab, or Bitbucket) is widely used for tracking

Facilitates task assignment, progress tracking, and team collaboration.

3. Examples:

- ➢ Jira – Advanced issue tracking & Agile project management.
- ➢ Trello – Simple Kanban-style task management.
- ➢ Asana – Useful for tracking feature development and bug fixes.

- **Database Design & Management Tools**

1. Helps in designing, querying, and managing the database schema.
2. Examples:

- ➢ MySQL Workbench (for relational databases).
- ➢ MongoDB Compass (for NoSQL databases).

- **API Development & Testing Tools**

Ensures backend APIs are well-documented and functional. Examples:

Postman – API testing and documentation.

Swagger (OpenAPI) – API specification and interactive docs.

5.4.2 Selected CASE Tools

For the Fund Room Booking Management System, the following tools will be used:

- **UML Tool**: Lucidchart / Draw.io

1. Used for:

- ➢ System architecture design (component diagrams, ER diagrams).
- ➢ Workflow modeling (Room Booking process, admin dashboard interactions).

2. Benefits:

- ➢ Visual clarity for stakeholders.
- ➢ Standardized documentation for future scalability.

- **IDE:** Visual Studio Code (VS Code)

1. Why selected?

- ➢ Supports React.js (frontend) + Node.js (backend).
- ➢ Extensions for debugging, Git integration, and live server testing.
- ➢ Version Control: Git (GitHub / GitLab)

2. Ensures:

- ➢ Collaboration among developers.
- ➢ Code backup & version history for rollback if needed.

- **Project Management:** Jira

1. **Used for:**

➢ Agile sprint planning (Scrum/Kanban boards).

➢ Bug tracking & task prioritization.

2. **Database Tool**: MySQL Workbench / MongoDB Compass

➢ MySQL Workbench (if using SQL) for schema design.

➢ MongoDB Compass (if using NoSQL) for document-based data management.

- **API Testing:** Postman

1. **Ensures:**

➢ Backend API reliability (Room Booking processing, user authentication).

➢ Automated testing for payment gateway integrations.

5.4 CASE Tools

5.4.1 Available CASE Tools

CASE (Computer-Aided Software Engineering) tools assist in software development by supporting analysis, design, coding, testing, and maintenance. Below are some useful CASE tools for a Room Booking Management System:

- **Unified Modeling Language (UML) Tools**

1. Used for visualizing system architecture, workflows, and data models.
2. Examples: Enterprise Architect, Lucidchart, Visual Paradigm.

- **Integrated Development Environments (IDEs)**

1. Essential for coding, debugging, and deployment.
2. Examples: Visual Studio Code, Eclipse, IntelliJ IDEA.

- **Version Control Systems (VCS)**

1. Helps manage source code changes and team collaboration.
2. Examples: Git (GitHub, GitLab, Bitbucket).

- **Issue Tracking & Project Management Tools**

1. Facilitates task management and progress tracking.
2. Examples: Jira, Trello, Asana.

5.4.2 Selected CASE Tools

- **UML Tools (e.g., Lucidchart)**

➢ **Benefits:**

1. Visual representation of system structure (use case diagrams, ER models).
2. Improves stakeholder communication.
3. Standardized documentation for consistency.

- **Git (GitHub)**

1. Ensures version control and collaborative development.

- **Jira**

1. Helps track bugs, manage sprints, and assign tasks efficiently.

5.5 Database

5.5.1 Available Databases

For a Room Booking Management System, the following databases can be considered:

- **Relational Databases (RDBMS)**

➢ **MySQL**: Open-source, scalable, and widely used for structured data.

➢ **PostgreSQL**: Advanced features, strong data integrity.

- **NoSQL Databases**

➢ **MongoDB**: Schema-less, flexible for unstructured data.

➢ **Firebase Realtime DB**: Cloud-based, real-time sync (useful for live updates).

5.5.2 Selected Database: MySQL (via MySQL Workbench) Why MySQL Workbench?

Structured Data Handling – Ideal for Room Booking records, user details, and transaction logs.
Scalability – Supports large datasets efficiently.
Security – Role-based access control for sensitive donor data.
Integration – Works well with web & mobile backends (PHP, Python, Java). **Visual Modeling** – MySQL Workbench provides ER diagram tools for database design.

Chapter 6

Testing (Software Quality Attributes)

6.1 Test Case Specification

This section defines the systematic approach to validate the Room Booking Management System against functional and non-functional requirements. The testing process ensures:

- **Functionality:** All features work as intended.
- **Usability:** Seamless experience for donors, admins, and beneficiaries.
- **Security:** Protection of sensitive data (e.g., donor details, payment info).
- **Performance:** Efficient handling of concurrent users and large datasets.

Test Deliverables:

- Test Plans
- Test Cases (Manual/Automated)
- Bug Reports
- Test Summary Reports

6.2 Black Box Test Cases

6.2.1 Boundary Value Analysis (BVA) Objective: Validate inputs at edge cases.
Test Cases:

Field	Min Value	Max Value	Invalid Values	Expected Result
Room Booking Amount	$1	$10,000	0,−0,−10, $10,001	Error for invalid; accept valid.
Name (Char Limit)	1 character	50 characters	51+ chars, empty	Truncate/reject overflow.
Phone Number	10 digits	15 digits	9 digits, letters	Reject malformed entries

6.2.2 Equivalence Class Partitioning

Objective: Group inputs to reduce redundant tests.
Test Cases:

Input Type	**Valid Classes**	**Invalid Classes**	**Expected Result**
Email Address	user@domain.com	user@.com, user.com	Accept valid; rejec invalid.
Password	8+ chars, mixed case, symbols	Less than 8 chars, all lowercase	Enforce complexity rules.
Payment Card	16 digits, valid expiry	15 digits, expired card	Reject invalid payments.

6.2.3 State Transition Testing

Objective: Verify workflows between system states.
Test Scenarios:

- **Room Booking Journey:**
  - State 1: Guest views Room Booking campaigns.
  - State 2: Logs in → Redirects to dashboard.
  - State 3: Room Booking → Payment gateway → Confirmation page.
- **Admin Workflow:**
  - State 1: Login → Dashboard.
  - State 2: Approves Room Booking → Updates beneficiary records.
  - State 3: Generates reports → Exports CSV.

6.2.4 Decision Table Testing

Objective: Test multi-condition logic.
Example: Tax Exemption Rules

Condition	Donation ≥ \$100?	Recurring?	Tax Exempt?	Outcome
1	Yes	Yes	Yes	Apply exemption.
2	Yes	No	No	Standard processing.

6.2.5 Graph-Based Testing

- **Objective**: Map cause-effect relationships. Example: Room Booking Submission
- **Cause**: Donor clicks "Submit Room Booking ."
- **Effects:**

➢ Payment gateway processes transaction.

➢ Database records donation.

➢ Confirmation email sent.

➢ Admin dashboard updates in real-time.

6.3 White Box Testing

6.3.1 Statement Coverage

Objective: Execute every code statement.

Tested Modules:

- **Room Booking Registration:**

➢ SQL queries, validation checks, email triggers.

- **Payment Processing:**

➢ API calls, success/failure handlers.

- Admin Reports:

➢ Data aggregation, PDF/CSV generation.

6.3.2 Branch Coverage

Objective: Test all decision branches (True/False).
Critical Branches: Payment Flow:

```
if payment_success:
   update_database()
   seid_email()
else:
   show_error()
```

Test both successful and failed payments

User Access Control:

```
if user_role == "admin":
   show_dashboard()
```

```
else:
    redirect_to_home()
```

6.3.3 Path Coverage

Objective: Test all possible execution paths.
Example Workflows:

- **Donor Path:**
  - ➢ Homepage → Login → Select Campaign → Donate → Payment → Confirmation.
- **Alternate Path:**
  - ➢ Guest → Donate → Forced Login → Resume donation.
- **Admin Path:**
  - ➢ Login → Audit Room Booking → Filter by Date → Export → Logout.

6.4 Additional Test Types (Recommended)

- **Performance Testing:**
  - ➢ Simulate 1000+ concurrent donors.
  - ➢ Measure page load times (<2 sec).
- **Security Testing:**
  - ➢ SQL injection, XSS attacks on forms.
  - ➢ OWASP ZAP for vulnerability scans.
- **Usability Testing:**
  - ➢ User feedback on navigation, form clarity.

6.4 Test Cases

Test Case 1: Login as Room Booking

Field	Details
Test ID	1
Date	1 Oct 2024
System	Web (React.js)
Objective	Login as Donor
Version	1

Field	Details
Input	Room Booking credentials
Steps	Open login page; Enter username and password; Click login
Expected Result	Room Booking dashboard displayed
Actual Result	Passed
Comments	Login works correctly

Test Case 2: Login as Admin

Field	Details
Test ID	2
Date	2 Oct 2024
System	Web (React.js)
Objective	Login as Admin
Version	1
Input	Admin credentials
Steps	Open login; Enter username/password; Click login
Expected Result	Admin dashboard displayed
Actual Result	Passed
Comments	Working as expected

Test Case 3: Register Room Booking

Field	Details
Test ID	3
Date	3 Oct 2024

Field	Details
System	Web (React.js)
Objective	Register Donor
Version	1
Input	Email, password, contact details
Steps	Open registration; Enter details; Submit form
Expected Result	Successful registration
Actual Result	Passed
Comments	Registration successful

Test Case 4: Create Campaign

Field	Details
Test ID	4
Date	4 Oct 2024
System	Web (React.js)
Objective	Create new campaign
Version	1
Input	Campaign name, description, goal
Steps	Admin login; Navigate to create; Enter data; Submit
Expected Result	Campaign created
Actual Result	Passed
Comments	Creation flow good

Test Case 5: Make Donation

Field	Details
Test ID	5
Date	5 Oct 2024
System	Web (React.js)
Objective	View campaigns
Version	1
Input	N/A
Steps	Login; Navigate to campaigns
Expected Result	Campaign list displayed
Actual Result	Passed
Comments	Correct listings

Test Case 6: View Donations

Field	Details
Test ID	6
Date	6 Oct 2024
System	Web (React.js)
Objective	Make donation
Version	1
Input	Campaign selected, amount
Steps	Login; Select campaign; Enter amount; Confirm
Expected Result	Room Booking accepted

Field	Details
Actual Result	Passed
Comments	Confirmation shown

Test Case 7: Approve Disbursement

Field	Details
Test ID	7
Date	7 Oct 2024
System	Web (React.js)
Objective	Admin views Room Booking
Version	1
Input	N/A
Steps	Login; Open Room Booking page
Expected Result	Donations shown
Actual Result	Passed
Comments	Accessible data

Test Case 8: View History

Field	Details
Test ID	8
Date	8 Oct 2024
System	Web (React.js)
Objective	Mark Room Booking completed
Version	1
Input	Select Room Booking to mark

Field	Details
Steps	Login; Find Room Booking; Mark completed
Expected Result	Status changed
Actual Result	Passed
Comments	Status updated

Test Case 9: Generate Report

Field	Details
Test ID	9
Date	9 Oct 2024
System	Web (React.js)
Objective	View Room Booking history
Version	1
Input	N/A
Steps	Login; Go to history
Expected Result	Past Room Booking listed
Actual Result	Passed
Comments	History shown correctly

Test Case 10: Filter Campaigns

Field	Details
Test ID	10

Date	10 Oct 2024
Field	**Details**
System	Web (React.js)
Objective	View pending Room Booking
Version	1
Input	N/A
Steps	Login; Open pending Room Booking
Expected Result	Pending Room Booking listed
Actual Result	Passed
Comments	Filtering works correctly

Chapter 7

Implementation

7.1 Component Diagram

7.1.1 Definition

A **Component Diagram** breaks down the **Hotel Room Booking Management System** into reusable **modules** and their dependencies. It ensures:

- **Modularity** (Separation of concerns)
- **Scalability** (Easy addition of new features)
- **Maintainability** (Independent component updates)

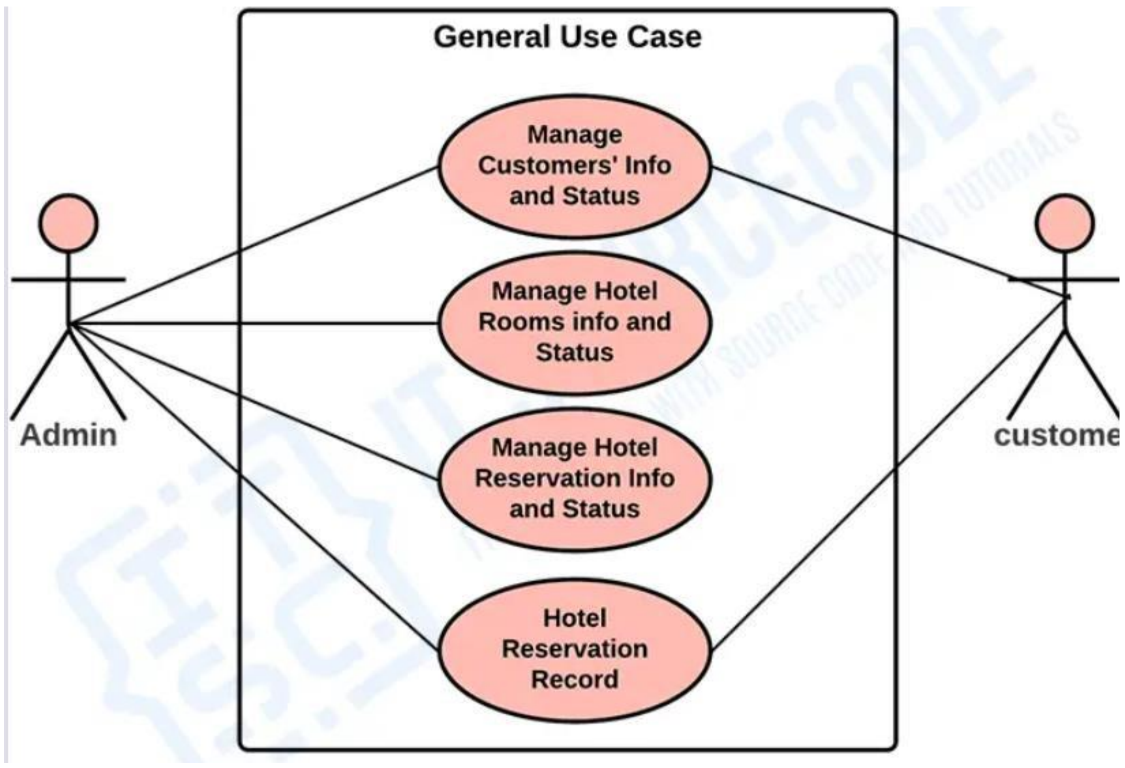

General Use Case
Manage Customers' Info and Status
Manage Hotel Rooms info and Status
Manage Hotel Reservation Info and Status
Hotel Reservation Record
Admin
custome

Fig.17: Component Diagram

7.1.2 Diagram Details Key Components:

- **User Interface (UI) Component**
  - Handles Room Booking /admin interactions (React/Angular).
  - Connects to Authentication & Payment Gateway.
- **Authentication Component**
  - Manages user login/signup (JWT/OAuth).
- **Payment Gateway Component**
  - Integrates Stripe/PayPal for donations.
- **Room Booking Management Component**
  - Management, room booking, transactions.
- **Database Component**
  - Stores data in MySQL/MongoDB.
- **Dependencies:**
  - UI → Authentication → Payment → Database.
  - Admin Dashboard → Room Booking Management → Reports.

7.2 Deployment Diagram

7.2.1 Definition

The Deployment Diagram visualizes the physical hardware and software nodes involved in the system, along with their communication protocols. It ensures:

- **Scalability** (Horizontal/Vertical scaling of servers).
- **Security** (Isolation of tiers, HTTPS encryption).
- **Performance** (Load balancing, optimized server roles).

Key Nodes & Responsibilities

Node	Technology Examples	**Role**
Client Device	Browser (Chrome/Firefox), Mobile (iOS/Android)	Renders the UI, sends HTTP requests t the web server.
Web Server	NGINX, Apache, AWS CloudFront	Hosts static files (HTML/CSS/JS), handles SSL/TLS, routes API requests
Application Server	Node.js, Django, Spring Boot	Processes business logic (Room Booking, auth), connects to database.
Database Server	MySQL (AWS RDS), MongoDB Atlas	Stores Room Booking data, transactions, and campaign records.
External Services	Stripe/PayPal, SendGrid (Email)	Handles payments and email notifications (SMTP/API).

7.2.2 Diagram Details

Connections & Protocols:

- **Client → Web Server**
  - **Protocol:** HTTPS (Port 443) for secure communication.
  - **Data Flow:** Static files (index.html, bundle.js) + API requests.
- **Web Server → Application Server**
  - **Protocol:** REST API (HTTP/HTTPS) or gRPC for microservices.
  - **Load Balancer:** Optional (e.g., AWS ALB) to distribute traffic.
- **Application Server → Database Server**
  - **Protocol:** SQL (TCP/IP) or NoSQL (MongoDB Wire Protocol).
  - **Security:** VPN/Private Subnet (AWS VPC) to restrict public access.
- **Application Server → External Services**
  - **Payment Gateway:** Stripe API (HTTPS with API keys).
  - **Email Service:** SMTP (Port 587) or SendGrid API.

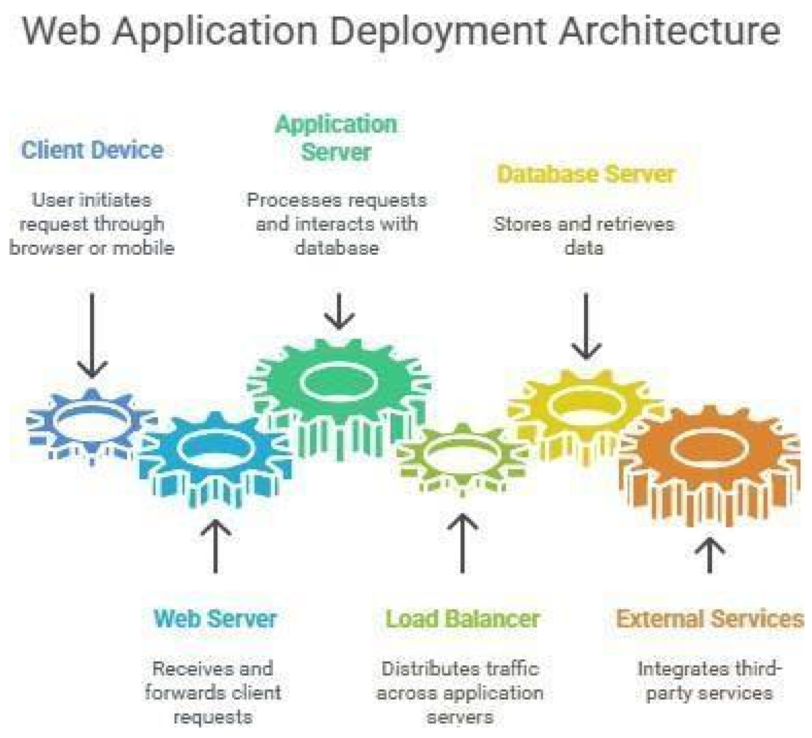

Web Application Deployment Architecture
Client Device
User initiates request through browser or mobile
Application Server
Processes requests and interacts with database
Database Server
Stores and retrieves data
Web Server
Receives and forwards client requests
Load Balancer
Distributes traffic across application servers
External Services
Integrates third-party services

Fig.18: Deployment Diagram

7.3 Database Architecture (1-Tier, 2-Tier, 3-Tier) for Room Booking Management System

7.3.1 Definition

Database architecture defines how the Room Booking Management System stores, retrieves, and manages data for Room Booking, recipients, and administrators. The system can follow 1- Tier, 2- Tier, or 3-Tier architecture, but 3-Tier is recommended for security, scalability, and performance.

7.3.2 Database Architectures (Explained for Donation System)

1. 1-Tier Architecture (Single-Tier)

Definition: All components (UI, Business Logic, Database) run on a single machine (e.g., a local PC or a simple server).
Use Case: Only suitable for **small-scale testing** (not for production).
Components:

- **Donor/Admin UI** (HTML, JavaScript)
- **Business Logic** (PHP/Python scripts)
- **Database** (SQLite/MySQL on the same machine)
- **Limitations:**
- **No security separation** (Database exposed to UI).
- **Not scalable** (Cannot handle multiple users efficiently).

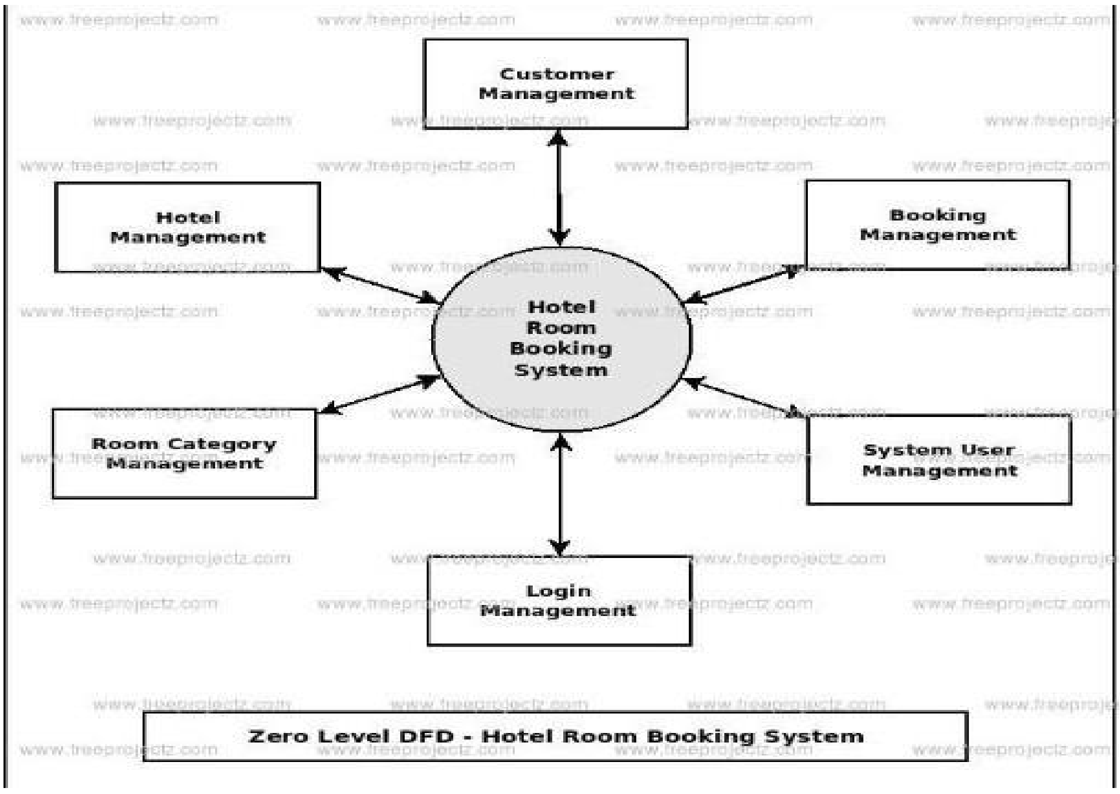


Fig.19: 1-Tier Architecture for Hotel Booking System

2. 2-Tier Architecture (Client-Server Model)

Definition: Separates UI (Client) from Database (Server).
Use Case: Suitable for small web apps but not ideal for high-security Room Booking systems.

- **Components:**
  - ➢ **Client Tier (Tier 1):** Web Browser (Room Booking /Admin Dashboard).
  - ➢ **Database Tier (Tier 2):** MySQL/PostgreSQL (Room booki, recipient, and transaction data).
- **How It Works:**
  - ➢ Room Booking submits a Room Booking → Browser sends SQL query directly to the database.
  - ➢ Admin views reports → Fetches data directly from the DB.
- **Limitations:**
  - ➢ **Security Risk:** UI directly accesses the database (SQL injection risks).
  - ➢ **No Business Logic Layer** (Hard to enforce rules like Room Booking limits).

○

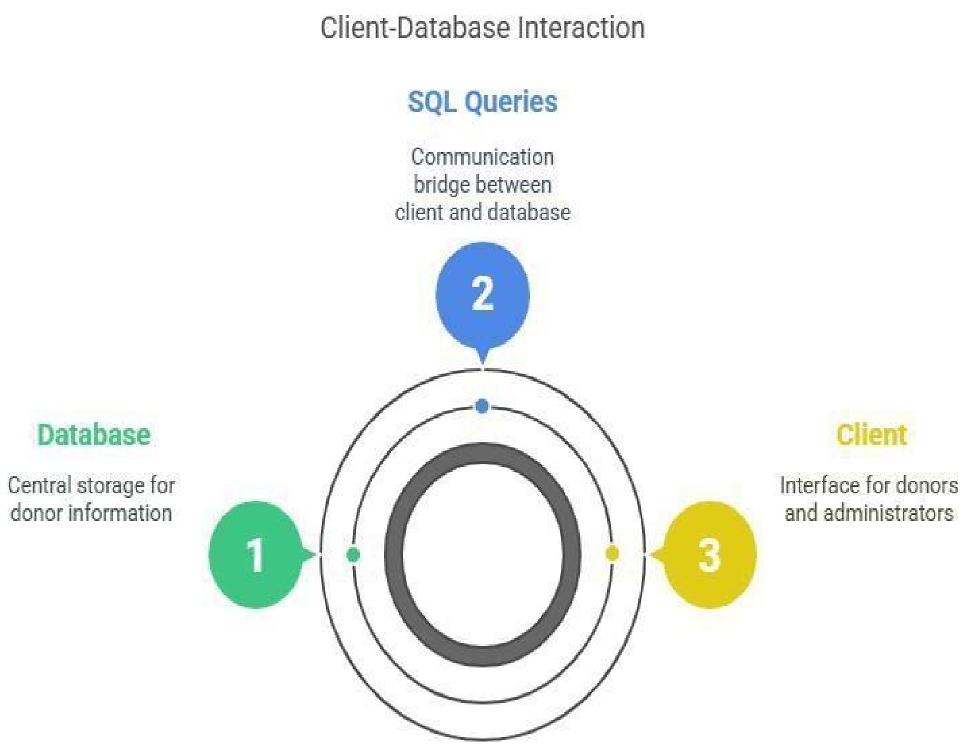

Client-Database Interaction
SQL Queries
Communication bridge between client and database
2
Database
Central storage for donor information
1
Client
Interface for donors and administrators
3

Fig.20: 2-Tier Architecture for Room Booking System

3. 3-Tier Architecture (Recommended for Room Booking System

Definition: Separates UI, Business Logic, and Database for better security and scalability.
Use Case: Best for production-ready Room Booking systems with multiple users.
Components:

- **Presentation Tier (Tier 1):** Website (React/Angular/HTML).
- **Application Tier (Tier 2):** Backend (Node.js/Django) → Handles:
  - ➢ Room Booking registration
  - ➢ Payment processing (Stripe/PayPal)
  - ➢ Admin approval of recipients
  - ➢ Reporting & analytics
- **Database Tier (Tier 3):** Secure DB (MySQL/MongoDB) → Stores:
  - ➢ Room Booking details (name, email, Room Booking history)
  - ➢ Recipient requests (charity info, verification status)
  - ➢ Transactions (amount, date, payment method)
- **Advantages:**
  - ➢ **Secure:** Database is not directly exposed to the internet.
  - ➢ **Scalable:** Each tier can be upgraded independently (e.g., adding more app servers).
  - ➢ **Flexible:** Easy to integrate APIs (SMS, Email, Payment Gateways).

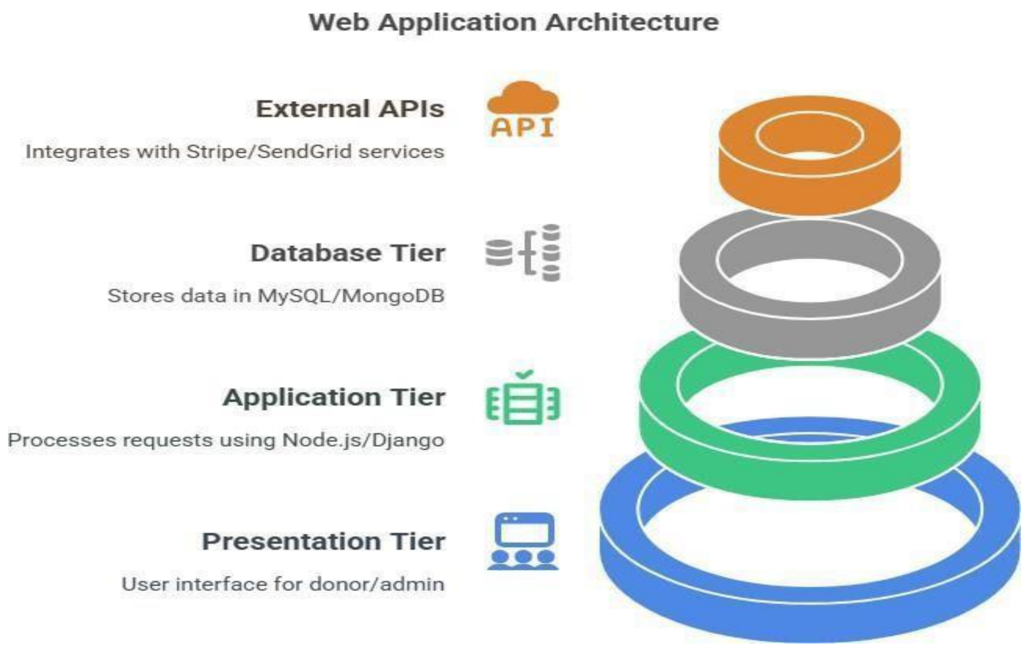

Web Application Architecture
External APIs
Integrates with Stripe/SendGrid services
API
Database Tier
Stores data in MySQL/MongoDB
Application Tier
Processes requests using Node.js/Django
Presentation Tier
User interface for donor/admin

Fig.21: 3-Tier Architecture for Room Booking System

Chapter 8:

Tools and Technologies

8.1 Programming Languages

The Hotel Room Booking Management System is built using modern, scalable, and secure programming languages suited for **web-based** Room Booking **platforms**.

1. JavaScript

- **Role:** Core language for **frontend and backend** development.
- **Why?**
  - Unified syntax for full-stack development.
  - Extensive libraries (e.g., Stripe.js for payments).
  - Asynchronous handling for Room Booking transactions.

2. React.js

- **Role:** Frontend library for building the **Room Booking and admin dashboards.**
- **Why?**
  - **Component-based UI:** Reusable components for donation forms, campaign listings, and reports.
  - **Fast rendering:** Critical for real-time Room Booking tracking.
  - **State management:** Redux/Context API for handling donor sessions and admin privileges.

3. Node.js (Backend)

- **Role:** Server-side logic for processing Room Booking, user auth, and APIs.
- **Why?**
  - **Non-blocking I/O:** Handles concurrent Room Booking requests efficiently.
  - **NPM ecosystem:** Libraries like Express.js for REST APIs and Helmet.js for security.

8.2 Markup and Styling

1. JSX (JavaScript XML)

- **Role:** Used with React.js to create **dynamic UI components**.
- **Examples:**
  - Donation form with real-time validation.
  - Interactive campaign cards with progress bars.

2. SCSS (Sass)

- **Role:** CSS preprocessor for **modular and responsive designs**.
- **Advantages:**
  - Variables for consistent theming (e.g., Room Booking buttons, alerts).
  - Mixins for reusable media queries (mobile/desktop compatibility).

8.3 Databases

MySQL (Relational Database)

- **Role:** Stores structured data for rooms, transactions, and campaigns.
- **Tables:**

➢ Room Booking (id, name, email, password_hash)
➢ campaigns (id, title, target_amount, current_amount)
➢ transactions (id, donor_id, campaign_id, amount, status)

Why MySQL?

- ACID compliance for secure transactions.
- Scalable for high-volume rooms decoration .

8.4 Operating Environment

Web-Based Platform

- **Frontend:** React.js (Deployed on **Netlify/Vercel**).
- **Backend:** Node.js/AdonisJS (Hosted on **AWS EC2/Heroku**).
- **Database:** MySQL (AWS RDS for managed services).

8.5 Development Tools

1. Visual Studio Code (VS Code)

- **Extensions Used:**

➢ ESLint (JavaScript linting).
➢ Prettier (Code formatting).
➢ REST Client (API testing).

2. Postman

- **Role:** API testing for endpoints like:

➢ /api/ Room Booking (POST Room Booking).
➢ /api/campaigns (GET active campaigns).

3. Git & GitHub

- **Workflow:**

➢ Feature branches for Room Booking /Admin modules.
➢ Pull requests for code reviews.
➢ GitHub Actions for CI/CD (auto-deploy to staging).

8.6 Additional Technologies

1. Payment Gateway Integration

- **Stripe.js:**

➢ Secure credit card processing.
➢ Webhooks for Room Booking confirmation emails.

2. Authentication

- **JWT (JSON Web Tokens):**
  - ➢ Secures Room Booking /admin login sessions.
  - ➢ OAuth 2.0 for social logins (Google/Facebook).

3. Security

- **Helmet.js:** HTTP header protection.
- **CORS:** Restricts API access to trusted domains.

APPENDIX A: USER DOCUMENTATION User Documentation for Admin

Introduction

Welcome to the Fund Hotel Room Booking Management System user documentation for admins.

This guide will help you understand how to use the system to manage Room Booking, donors, and fundraising campaigns effectively.

Getting Started

- Access the Fund Hotel Room Booking Management System through your web browser.
- Log in with your admin credentials or sign up if you are a new admin.
- Set up your profile with essential information, including contact details.

Room Booking Management

- Add, edit, or remove Room Booking categories, including Fund, Medical, Education, and Animal Room Booking.
- Monitor ongoing Room Booking and track Room Booking amounts.
- Review and approve incoming Room Booking requests from Room Booking.

Room deoration Management

- View and manage registered Room Booking.
- Track Room Booking history and Room Booking activities.
- Update Room Booking information as needed.

Campaign Management

- Create, edit, or delete fundraising campaigns.
- Add descriptions, images, and target amounts for each campaign.
- Monitor the progress of active campaigns and adjust as needed.
- Managing Special Offers and Promotions
- Create special offers or promotional messages for Room Booking to encourage contributions.
- Specify terms and conditions for each promotion.

User Documentation for Donor

Introduction

Welcome to the Fund Hotel Room Booking Management System user documentation for hotel guests.

This guide will help you understand how to use the system to make Room Booking and support various causes.

Getting Started

- Access the Fund Room Booking Management System through your web browser.
- Create a new Room Booking account or log in if you already have one.
- Provide your personal details, including name, contact number, and email address.

Making Donations

- Explore available Room Booking categories, including Fund, Medical, Education, and Animal Room Booking.
- Select your preferred category and enter the Room Booking amount.
- Complete the Room Booking process using the available payment options.

Tracking Donations

- View your Room Booking history and track the impact of your contributions.
- Receive notifications and updates on your supported campaigns.

APPENDIX B: References

1. Official documentation for React.js, used for building the web-based front end of the Fund Room Booking Management System. Available at: https://reactjs.org/docs/getting-started.html
2. Official documentation for Node.js, used to develop the server-side logic. Available at: https://nodejs.org/en/docs/
3. MySQL official documentation, used for database management. Available at: https://dev.mysql.com/doc/
4. Heroku documentation for deploying backend services. Available at: https://devcenter.heroku.com/
5. GitHub for version control and collaboration. Available at: https://github.com/
6. Firebase documentation, used for real-time notifications and data synchronization. Available at: https://firebase.google.com/docs