Software Engineering Semester Project

Hotel Management System

Presented by:
Haleema Tahir - 22i-0937
Muhammad Daniyal Aziz - 22i-0753
Moaz Farooq - 22i-1173

Presented to:
Sir Basharat Hussain

Course Number: CS3009

Contents

1. Project Introduction

The Hotel Management System is a comprehensive software solution designed to streamline hotel operations, enhance guest experiences, and improve administrative efficiency. It will provide a centralized platform for booking management, room allocation, billing, and reporting, catering to the needs of hotel staff, managers, and guests.

The system aims to replace manual processes with an automated, user-friendly interface, ensuring accuracy, speed, and scalability. By leveraging modern technologies, the Hotel Management System will serve as a reliable tool for hotels to manage their daily operations effectively.

Project Vision

Our vision is to create a robust, scalable, and intuitive Hotel Management System that simplifies hotel operations and enhances the guest experience. The system will empower hotel staff to manage bookings, rooms, and payments efficiently while providing guests with a seamless booking and check-in process.

Scope

The Hotel Booking System will allow users to search for available rooms, make reservations, process payments, manage bookings, and provide customer feedback. The system will also offer administrative functionalities such as room management, user management, and reporting.

Intended Audience and Reading Suggestions

This document is intended for:

- Developers
- Project Managers
- System Architects
- Testers
- Clients/Stakeholders

Intended Use of the System

Stakeholders and Their Needs

Hotel Guests:

Needs: Easy booking, real-time room availability, secure payment options, and personalised experiences.

Hotel Staff:

Needs: Efficient room management, quick check-in/check-out, billing, and guest information access.

Hotel Managers:

Needs: Reporting, analytics, inventory management, and staff performance tracking.

How the System Will Be Used

Guests: Guests will use the system to book rooms, view room details, and make payments online.

Staff: Staff will use the system to manage bookings, allocate rooms, process check-ins/check-outs, and generate bills.

Managers: Managers will use the system to monitor occupancy rates, generate reports, and manage inventory.

Features and Overall Functionality

What the System Will Do

Room Booking: Guests can book rooms online, view availability, and select room types.

Check-In/Check-Out: Staff can manage guest arrivals and departures efficiently.

Billing and Payments: Generate and process bills with multiple payment options.

Reporting and Analytics: Provide managers with insights into occupancy rates, revenue, and guest preferences.

Inventory Management: Track and manage hotel inventory (e.g., room supplies, and amenities).

How the System Will Help Users

Guests: Simplifies the booking process and provides a seamless experience.

Staff: Reduces manual work and improves efficiency in managing rooms and guests.

Managers: Offers data-driven insights for better decision-making.

2. Team Introduction

Team Name:

CheckIn Crew

Team Logo:

HOTEL
CHECKIN
CREW
CHECKIN CREW - ELEVATE HOSPITALITY, SIMPLIFY MANAGEMENT!

Group Picture:

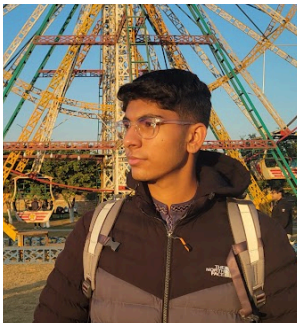
Team Leadership

- Team Lead: Haleema Tahir Malik

2.1 Team Member Introductions

Member 1

Muhammad Daniyal Aziz



Biography:

Daniyal is a passionate Computer Science student at FAST NUCES Islamabad, currently in his 6th semester. With a strong foundation in software development, he has worked on various projects, including inventory management systems, web applications, and game development. His expertise spans front-end and back-end development, with proficiency in **Java, C++, Python, SQL, and modern web technologies like React**.

Beyond academics, Daniyal has contributed as a **mentor in the FAST Computing Society** and serves as the **Vice Head of the Parho and Parhao program** for GDGOC. He enjoys problem-solving, UI/UX design, and exploring data science and AI.

GitHub Username: Daniyal Aziz

Member 2

Moaz Farooq

Biography:

Moaz Farooq is a passionate Computer Science student at FAST NUCES Islamabad with a strong background in **artificial intelligence, backend development, and database management.** He has gained hands-on experience through **internships** at **Huawei Pakistan and Ezi Tech,** working on process automation, database optimization, and software development. His expertise includes **Python, Java, C/C++,SQL, and multithreading,** along with experience in **deep learning, numerical methods, and large-scale data processing.**

Beyond academics, Moaz is a Team AI/DS member in the **Google Developer Group of Campus (GDGoC)**, where he contributes to AI and data science projects. He has also been involved in **community service**, tutoring students to promote education. With a keen interest in **problem-solving, system optimization, and AI research**, he continues to explore new technologies and innovative solutions.

GitHub Username: Dizzy1s

Member 3

Haleema Tahir Malik

Biography:

Haleema Tahir Malik is a dedicated Computer Scientist with expertise in Java, C++, Python, and database management. She has a strong background in **game development, web applications, and system design**. Passionate about problem-solving and innovation, she has built **interactive games, web-based platforms, and AI-driven applications**, excelling in both **technical and business domains**.

GitHub Username: HaleemaMalik

2.2 Team Roles and Responsibilities

Role	Assigned Member
Product Owner / Project Manager	Moaz Farooq
Scrum Master	Haleema Tahir
Requirement Analyst / Architect	M. Daniyal Aziz
Developers	M. Daniyal Aziz, Moaz Farooq, Haleema Tahir Malik
Testers	M. Daniyal Aziz, Moaz Farooq, Haleema Tahir Malik
UI Designer	M. Daniyal Aziz, Haleema Tahir

2.3 Team Agreement

Communication

Methods: Email, WhatsApp, Google Meet sessions.

Response Time: Within 24 hours for general inquiries, ASAP for urgent tasks.

Meetings

Frequency: Daily Stand-ups, Mid-week reviews.

Format: Online (Zoom/Google Meet) / In-Person / Hybrid.

Attendance: Mandatory for all members unless otherwise stated.

Meeting Notes: Haleema will take notes and store them in Google Drive.

Work Organization

Version Control: GitHub (Branching Strategy, Commit Message Format).

Task Division: Tasks assigned using Trello/Jira based on workload and expertise.

Assignment Submission: Reviewed by M. Daniyal Aziz before final submission.

Contingency Plan

If a team member drops out: Redistribute tasks and notify the instructor.

If a team member frequently misses meetings: Discuss internally and escalate if necessary.

If academic dishonesty occurs: Report to the instructor and take necessary action.

3. Functional Requirements

User Roles

- **Guest User**: Can browse hotels and check availability.
- **Registered Customer**: Can book rooms, make payments, view booking history, and provide feedback.
- **Administrator**: Can manage room listings, users, bookings, and reports.

System Features

1. **User Authentication**
    - Users must register and log in to access booking services.
    - Admin has a separate authentication system.
2. **Room Search and Filtering**
    - Users can search rooms based on location, price, type, and availability.
3. **Room Booking**
    - Users can select check-in/check-out dates and confirm bookings.
4. **Payment Processing**
    - Users can make secure payments via credit/debit card, PayPal, or other gateways.
    - A confirmation email is sent upon successful payment.
5. **Booking Management**
    - Users can modify or cancel bookings before the check-in date.
6. **Admin Management**
    - Admin can add/edit/remove rooms and update availability.
    - Admin can generate reports on revenue and occupancy rates.
7. **Customer Feedback & Support**
    - Users can submit reviews and ratings for their stay.
    - A support chat or contact form will be available.

4. Non-Functional Requirements

Product Requirements

- **Performance:** The system should handle 100 concurrent users.
- **Security:** User data and transactions must be encrypted.
- **Usability:** The interface should be intuitive and responsive.

Organizational Requirements

- **Compliance:** Must follow GDPR for data protection.
- **Scalability:** The system should support future expansion.

External Requirements

- **Regulatory Compliance:** Must comply with PCI DSS for payment security.
- **Integration:** Should integrate with third-party payment gateways and SMS/email APIs.

5. ERD:


rooms
id int
room_number varchar
type varchar
price decimal
status enum
description text
created_at timestamp
bookings
id int
user_id int
room_id int
check_in date
check_out date
status enum
payment_method enum
total_price decimal
booking_date timestamp
staff
id int
user_id int
position varchar
salary decimal
hire_date date
users
id int
name varchar
email varchar
password_hash varchar
role enum
created_at timestamp
admins
id int
user_id int
access_level enum
complaints_requests
id int
user_id int
booking_id int
type varchar
message text
status varchar
created_at timestamp
resolved_at timestamp

6. Use Case Diagram:

Haleema Tahir Malik 22i-0937
Muhammad Daniyal Aziz 22i-0753
Moaz Farooq 22i-1173

6.1 User Stories from Use Case Diagram:

- **Guest User Stories**

1. **Search Rooms**
   As a guest user, I want to search for available rooms so that I can find suitable options for booking.

- **Pre-condition:** The system must have room data available.
- **Post-condition:** The user sees a list of available rooms.

2. **View Hotel Details**
   As a guest user, I want to view hotel details so that I can make an informed decision before booking.

- **Pre-condition:** The selected hotel exists in the system.
- **Post-condition:** The user sees hotel descriptions, amenities, and pricing.

3. **Register an Account**
   As a guest user, I want to register an account so that I can make bookings and track my history.

- **Pre-condition:** The user must provide valid credentials.
- **Post-condition:** The user receives confirmation of registration.

- **Registered Customer Stories**

4. Search Rooms

As a registered customer, I want to search for available rooms so that I can find suitable options for booking.

- **Pre-condition:** The system must have room data available.
- **Post-condition:** The user sees a list of available rooms.

5. Login Account

As a user, I want to log into the system so that I can access my account and use available features.

- **Pre-conditions:**
  - The user has a registered account.
  - The system has a login interface.
- **Post-conditions:**
  - The user is authenticated and granted access to their account.
  - The system denies access if the credentials are incorrect.

6. View Hotel Details

As a registered customer, I want to view hotel details so that I can make an informed decision before booking.

- **Pre-condition:** The selected hotel exists in the system.
- **Post-condition:** The user sees hotel descriptions, amenities, and pricing.

7. Book a Room

As a registered customer, I want to book a room so that I can reserve accommodation for my stay.

- **Pre-condition:** Available rooms must exist in the system.
- **Post-condition:** The room is booked, and a confirmation is generated.

8. Modify/Cancel Booking

As a registered customer, I want to modify or cancel my booking so that I can adjust my travel plans.

- **Pre-condition:** The booking must exist in the system.
- **Post-condition:** The booking is updated or canceled.

9. Make Payment

As a registered customer, I want to make a payment so that I can confirm my booking.

- **Pre-condition:** The booking must be created and valid.
- **Post-condition:** The payment is processed successfully.

10. View Booking History

As a registered customer, I want to view my booking history so that I can keep track of my past and upcoming stays.

- **Pre-condition:** The user must have made previous bookings.
- **Post-condition:** The booking records are displayed.

11. Provide Feedback

As a registered customer, I want to provide feedback so that I can share my experience

with the hotel.

- **Pre-condition:** The user must have completed a booking.
- **Post-condition:** The feedback is stored and visible for review.

🔹 **Staff Stories**

12. Manage Room Availability

As a staff member, I want to manage room availability so that I can ensure correct occupancy status is maintained.

- **Pre-condition:** Rooms exist in the system.
- **Post-condition:** Room availability is updated.

13. Update Check-in/Check-out Status

As a staff member, I want to update the check-in/check-out status so that I can keep track of guest arrivals and departures.

- **Pre-condition:** A valid booking must exist.
- **Post-condition:** The system reflects updated room occupancy.

14. Handle Customer Complaints or Special Requests

As a staff member, I want to handle customer complaints or special requests so that I can improve guest satisfaction.

- **Pre-condition:** A complaint or request must be submitted.
- **Post-condition:** The issue is resolved, and the system logs the response.

15. Modify/Cancel Booking

As a staff member, I want to modify or cancel my customer's booking so that I can improve guest satisfaction.

- **Pre-condition:** The booking must exist in the system.
- **Post-condition:** The booking is updated or canceled.

16. View Booking History

As a staff member, I want to view my customer's booking history so that I can keep track of my customer's past and upcoming stays.

- **Pre-condition:** The user must have made previous bookings.
- **Post-condition:** The booking records are displayed.

🔹 Administrator Stories

17. Manage Room Listings

As an administrator, I want to manage room listings so that I can update room availability and descriptions.

- **Pre-condition:** The system must allow room modifications.
- **Post-condition:** The updated listings are saved.

18. Manage User and Staff Accounts

As an administrator, I want to manage user and staff accounts so that I can regulate access, remove inactive users, assign roles, and control system permissions.

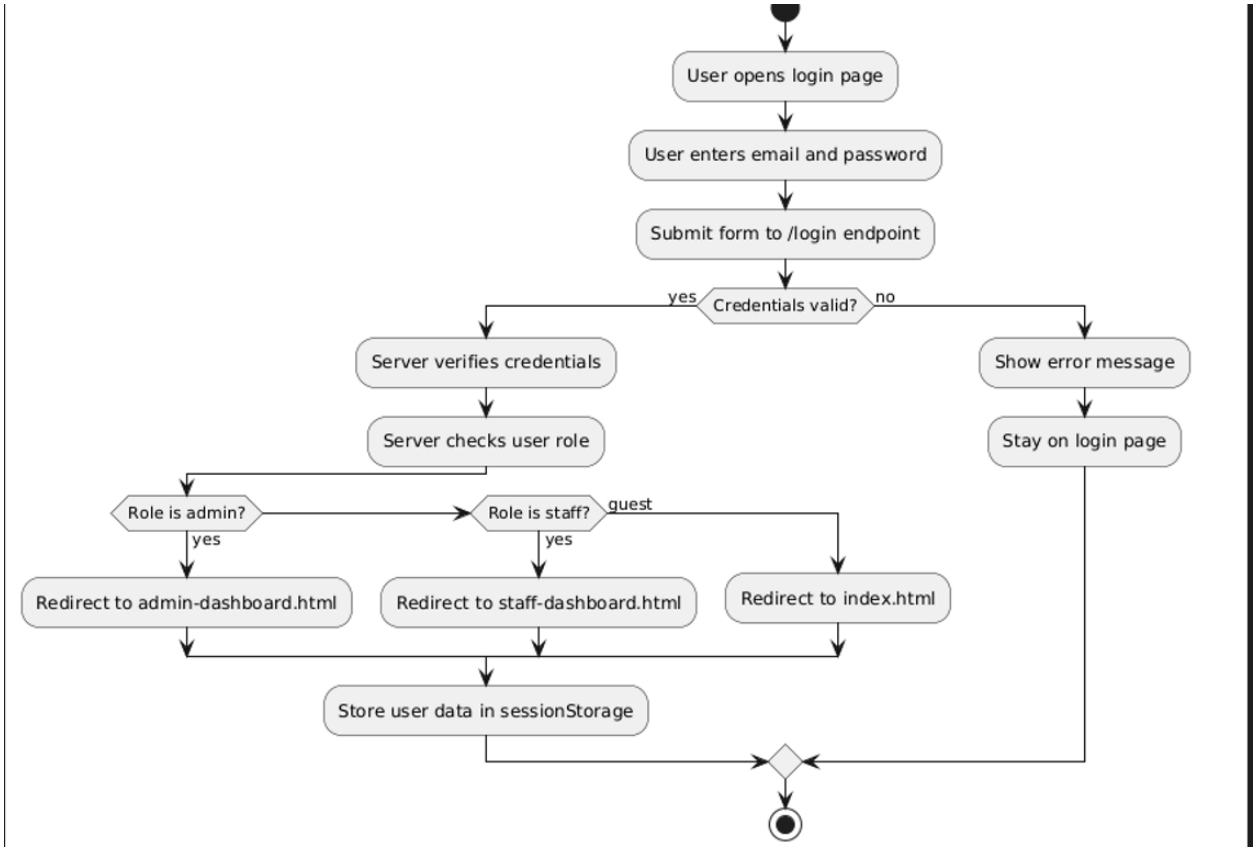
- **Pre-conditions:**
  User and staff accounts exist in the system.
  The system supports staff role management.

- **Post-conditions:**
  The system updates user and staff access as required.
  The system assigns or modifies staff roles and permissions.

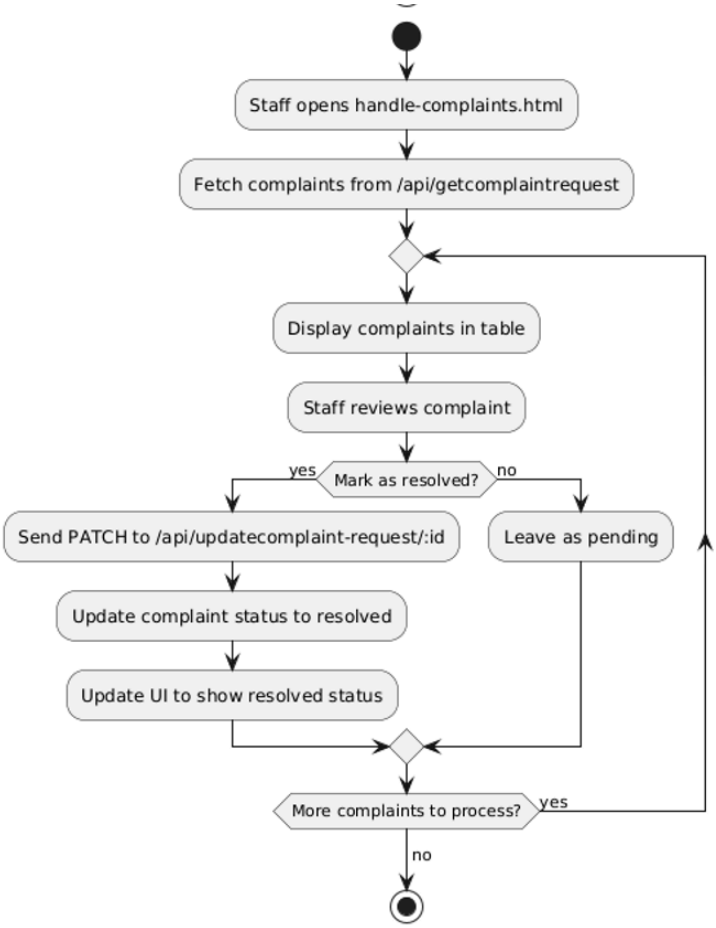
19. **Generate Booking Reports**
    As an administrator, I want to generate booking reports so that I can analyze booking trends and system performance.
- **Pre-condition:** The system must store booking records.
- **Post-condition:** A report is generated and available for review.

Activity Diagram:

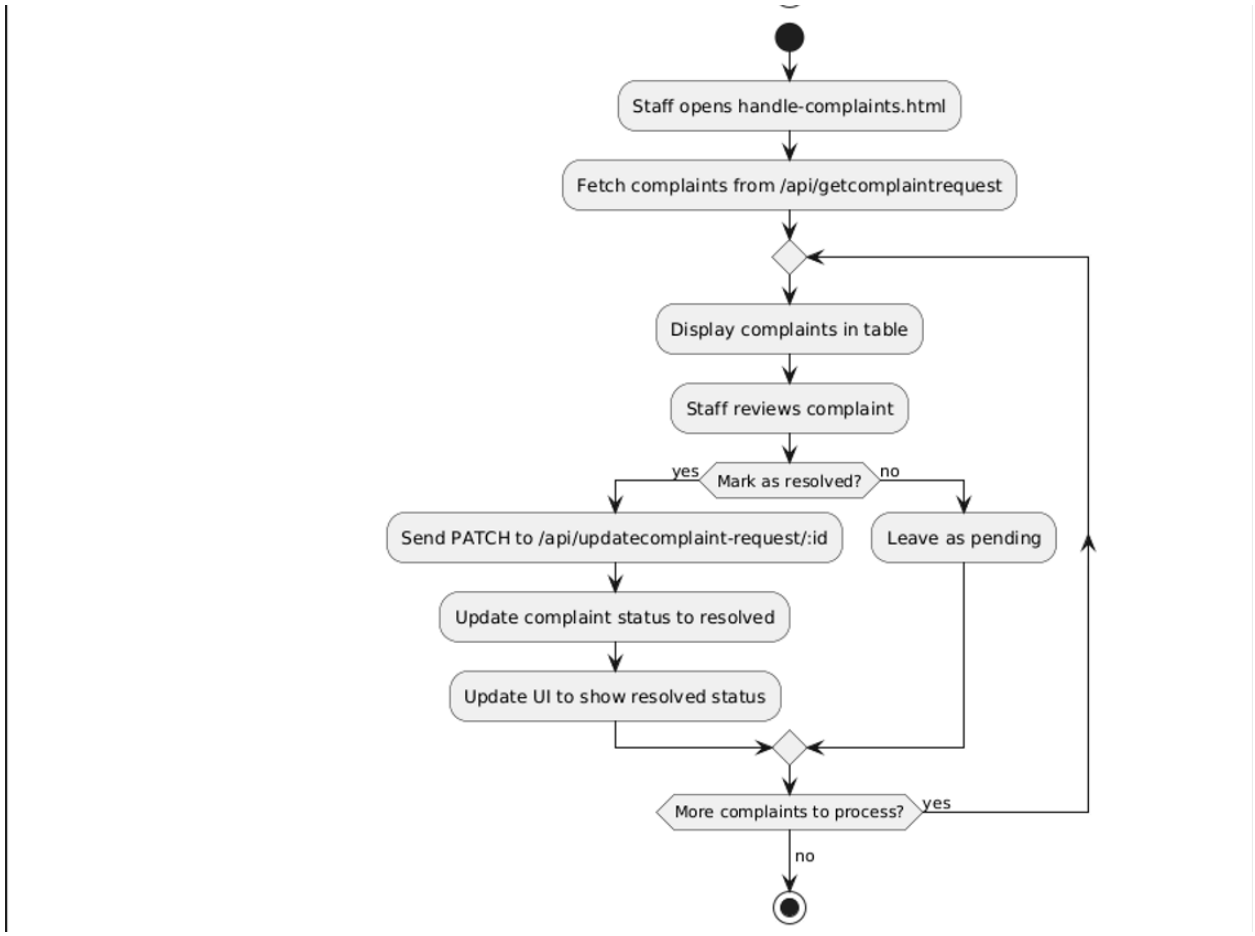
- **Login**



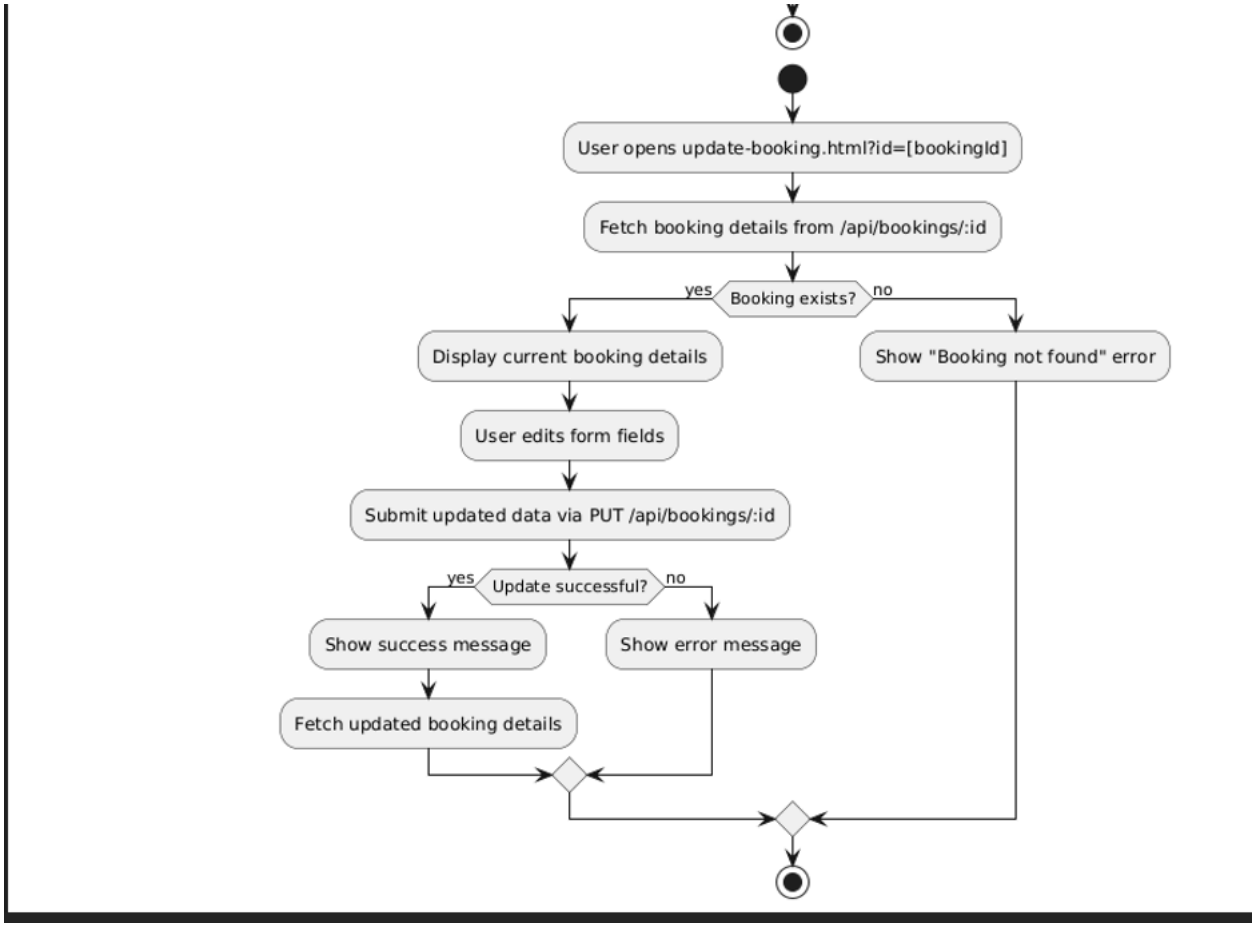
- **Handle Complaints**



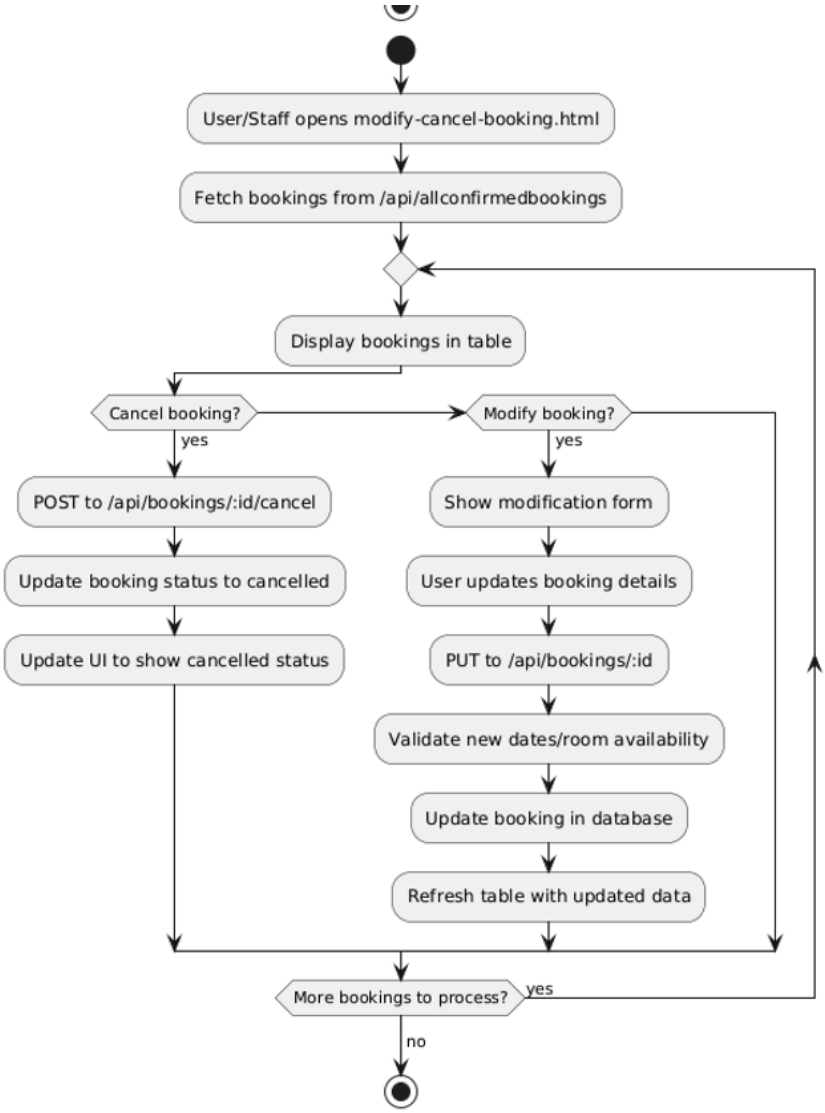
- **Manage Checkin / Checkout**



- **Modify Booking**



- **Modify/ Cancel Booking**



7. Product Backlog

User Story	Description	Priority	Status
Search Rooms (Guest User) US1	As a guest, I want to search for rooms by date, room type, and filters like price or amenities, so I can find a suitable room for my stay.	High	Completed (Sprint 1)
View Hotel Details US2+US6	As a guest, I want to view hotel details (e.g., name, location, amenities) and additional information like photos and reviews, so I can make an informed decision.	High	Completed (Sprint 1)
Register Account US3	As a user, I want to register an account with my name, email, password, and phone number, so I can access personalized features like booking history.	High	Completed (Sprint 1)
Search Rooms (Registered User) US4	As a registered user, I want to search for rooms with additional filters (e.g., saved preferences), so I can quickly find rooms that match my needs.	High	Completed (Sprint 1)
Login Account US5	As a registered user, I want to log in with my email and password, so I can access my account and manage my bookings securely.	High	Completed (Sprint 2)
Book a Room US7	As a guest, I want to book a room by selecting dates, entering guest details, and choosing a payment method, so I can secure my stay.	High	Completed (Sprint 2)
Modify/Cancel Booking US8	As a guest, I want to modify or cancel my booking (e.g., change dates, cancel with a refund), so I can adjust my plans as needed.	Medium	Completed (Sprint 2)
Make Payment US9	As a guest, I want to make a secure payment for my booking using options like credit card or PayPal, so I can complete my reservation.	High	Completed (Sprint 2)

View Booking History US10	As a registered user, I want to view my booking history with filters (e.g., past, upcoming), so I can track my reservations.	Medium	Completed (Sprint 2)
Provide Feedback US11	As a guest, I want to provide feedback on my stay with a rating and comments, so I can share my experience with the hotel.	Low	Completed (Sprint 2)
Manage Room Availability US12	As a staff member, I want to manage room availability (e.g., mark rooms as available or unavailable), so I can ensure accurate booking information.	Medium	Completed (Sprint 3)
Update Check-In/Check-Out Status US13	As a staff member, I want to update the check-in and check-out status of guests, so I can track room occupancy and manage housekeeping.	Medium	Completed (Sprint 3)
Handle Customer Complaints or Special Requests US14	As a staff member, I want to handle customer complaints or special requests (e.g., extra towels, late check-out), so I can improve guest satisfaction.	Medium	Completed (Sprint 3)
Modify/Cancel Booking (Staff) US15	As a staff member, I want to modify or cancel a guest's booking on their behalf, so I can assist with changes or cancellations.	Medium	Completed (Sprint 3)
View Booking History US16	As a staff member, I want to view booking history for all guests, so I can analyze trends and manage operations.	Low	Completed (Sprint 3)
Manage Room Listing US17	As a staff member, I want to manage room listings (e.g., add, update, or remove rooms), so I can keep the inventory up to date.	Low	Completed (Sprint 3)
Manage User and Staff Accounts US18	As an admin, I want to manage user and staff accounts (e.g., create, update, delete accounts), so I can control access to the system.	Low	Completed (Sprint 3)

Generate Booking Reports US19	As an admin, I want to generate booking reports (e.g., occupancy rates, revenue), so I can analyze business performance.	Low	Completed (Sprint 3)

8. Sprint Backlog

8.1 Sprint 1: March 6 - March 15

Sprint Goal: Establish foundational user functionalities, enabling users to search for rooms, view hotel details, and register accounts as the initial building blocks of the system.

Task	Description	Planned Date	Assignee	Status
Search Rooms US1+US4 (Frontend)	Develop a responsive frontend interface allowing users to search rooms by date, room type, and filters like price or amenities. Includes input fields, search button, and results display.	Mar 6 - Mar 8	Haleema Tahir	Completed
View Hotel US2+US5 (Frontend)	Create frontend pages to display hotel details (e.g., name, location, amenities), including a gallery for photos and a section for user reviews with ratings.	Mar 9 - Mar 11	Haleema Tahir	Completed
Register Account US3 (Frontend)	Implement a user registration form with fields for name, email, password, and phone number, including client-side validation for email format and password strength.	Mar 12 - Mar 14	Haleema Tahir	Completed
Register Account US3 (Backend)	Set up a backend API to process registration requests, validate input data (e.g., unique email), hash passwords, and return success/error responses.	Mar 13 - Mar 14	Daniyal Aziz	Completed
Register Account US3 (DB)	Design and implement a database schema with a "users" table including columns for id, name, email, hashed password, and phone number, with indexing on email for uniqueness.	Mar 14 - Mar 15	Moaz Farooq	Completed

8.1 Sprint 2: March 16 - March 22

Sprint Goal: Implement advanced functionalities to enable user authentication, room booking, payment processing, booking management, and feedback submission, enhancing the system's usability and interactivity.

Task	Description	Planned Date	Assignee	Status
Login Account US5 (Frontend)	Develop a login page with fields for email and password, client-side validation, and error messages for invalid credentials or server issues.	Mar 16 - Mar 17	Haleema Tahir	Completed
Login Account US5 (Backend)	Implement backend authentication logic using tokens, verify user credentials, manage sessions, and handle logout functionality.	Mar 18 - Mar 19	Daniyal Aziz	Completed
Login Account US5 (DB)	Ensure the database supports secure credential storage (e.g., hashed passwords) and efficient retrieval for authentication queries.	Mar 18 - Mar 19	Moaz Farooq	Completed
Book a Room US7 (Frontend)	Create a booking interface with room selection, date picker, guest details form, and payment method options (credit card, PayPal), with real-time availability updates.	Mar 17 - Mar 19	Haleema Tahir	Completed
Book a Room US7 (Backend)	Handle booking logic: check room availability, reserve the room, process payment via API calls, and confirm booking with a unique ID.	Mar 19 - Mar 20	Daniyal Aziz	Completed
Book a Room US7 (DB)	Update the database with a "bookings" table, linking to users and rooms tables.	Mar 19 - Mar 20	Moaz Farooq	Completed
Modify/Cancel Booking US8 (Frontend)	Provide a UI for users to view bookings, edit details (e.g., dates, guests), or	Mar 20 - Mar 22	Haleema Tahir	Completed

	cancel with confirmation prompts and success/error feedback.			
Modify/Cancel Booking US8 (Backend)	Implement APIs to update booking details (e.g., new dates) or mark bookings as canceled, with validation for availability and refund eligibility.	Mar 21 - Mar 22	Daniyal Aziz	Completed
Modify/Cancel Booking US8 (DB)	Update the "bookings" table to reflect modifications or cancellations, maintaining a status column (e.g., active, canceled) and logging changes.	Mar 21 - Mar 22	Moaz Farooq	Completed
Make Payment US9 (Frontend)	Integrate payment gateways (e.g., credit card, PayPal) into the frontend, with a secure checkout form, loading states, and success/error notifications.	Mar 17 - Mar 19	Haleema Tahir	Completed
Make Payment US9 (Backend)	Process payments server-side: validate payment details, communicate with gateways, log transactions, and return confirmation to the frontend.	Mar 17 - Mar 19	Daniyal Aziz	Completed
Make Payment US9 (DB)	Store payment records in a "payments" table updating "bookings" with payment status.	Mar 17 - Mar 19	Moaz Farooq	Completed
View Booking History US10 (Frontend)	Display a user’s booking history with filters (e.g., past, upcoming), showing details like dates, room type, and status in a tabular or card format.	Mar 20 - Mar 21	Haleema Tahir	Completed
View Booking History US10 (Backend)	Create an API to query the database for a user’s booking history, supporting filtering and pagination for scalability.	Mar 21 - Mar 22	Daniyal Aziz	Completed
Provide Feedback US11 (Frontend)	Build a feedback form with fields for rating (1-5 stars), comments, and optional photos, including validation and submission confirmation.	Mar 21 - Mar 22	Haleema Tahir	Completed

Provide Feedback US11 (Backend)	Handle feedback submissions via an API, validate input (e.g., rating range), store data, and return a success response to the frontend.	Mar 21 - Mar 22	Daniyal Aziz	Completed
Provide Feedback US11 (DB)	Create a "feedback" table to store submissions, linked to users and bookings.	Mar 21 - Mar 22	Moaz Farooq	Completed

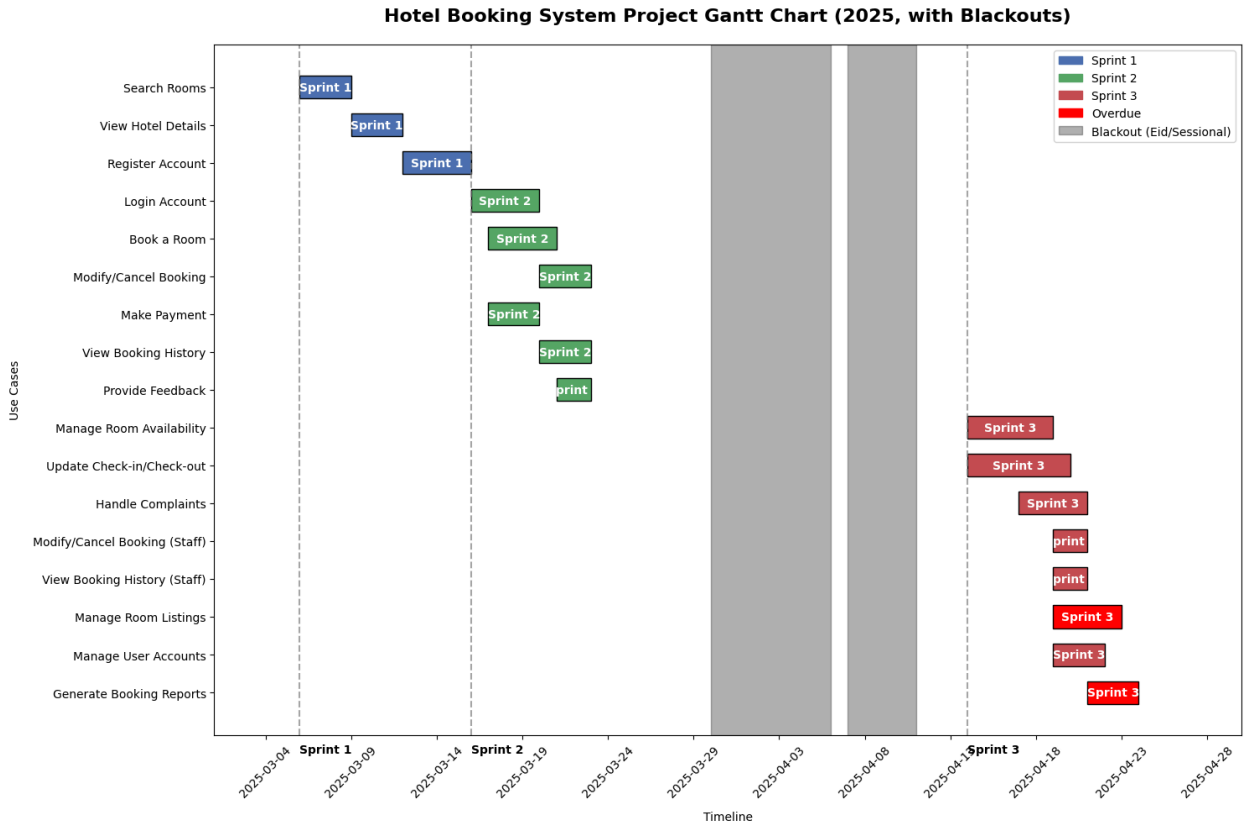
---

9. Project Plan

9.1 Work Breakdown Structure (WBS)

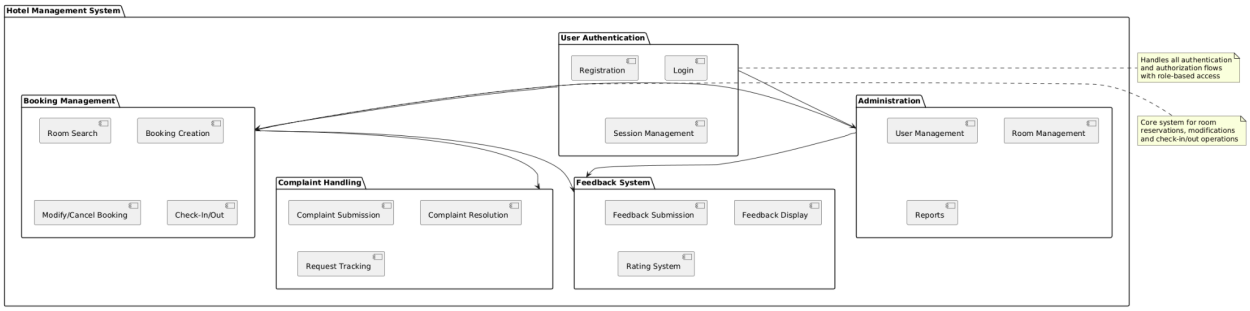
Phase	Task	Sub-Task/Deliverable
1. Requirements Gathering	- Define system requirements - Gather user stories - Create use case diagrams	Functional and Non-Functional Requirements Use Case Diagram
2. Design	- Design system architecture - Database schema design - UI/UX wireframes	Architecture Diagram Database Tables Initial Page Designs
3. Development (Sprint 1 & 2)	- Frontend development (HTML, CSS, Tailwind) - Backend setup (API, database connection) - User Authentication - Room browsing and booking functionality	Working prototype with Login/Register, Bookings, Search Rooms
4. Sprint 3 and Enhancements	- Staff dashboard - Admin dashboard - Review, Feedback, Complaint handling pages	Fully functional Staff/Admin roles and features
5. Testing	- Unit testing - Integration testing - User acceptance testing (UAT)	Test Cases (Black Box and White Box Testing)
6. Finalization	- Report compilation - Presentation preparation	Final Report (this one) Presentation Slides

9.2 Gantt Chart



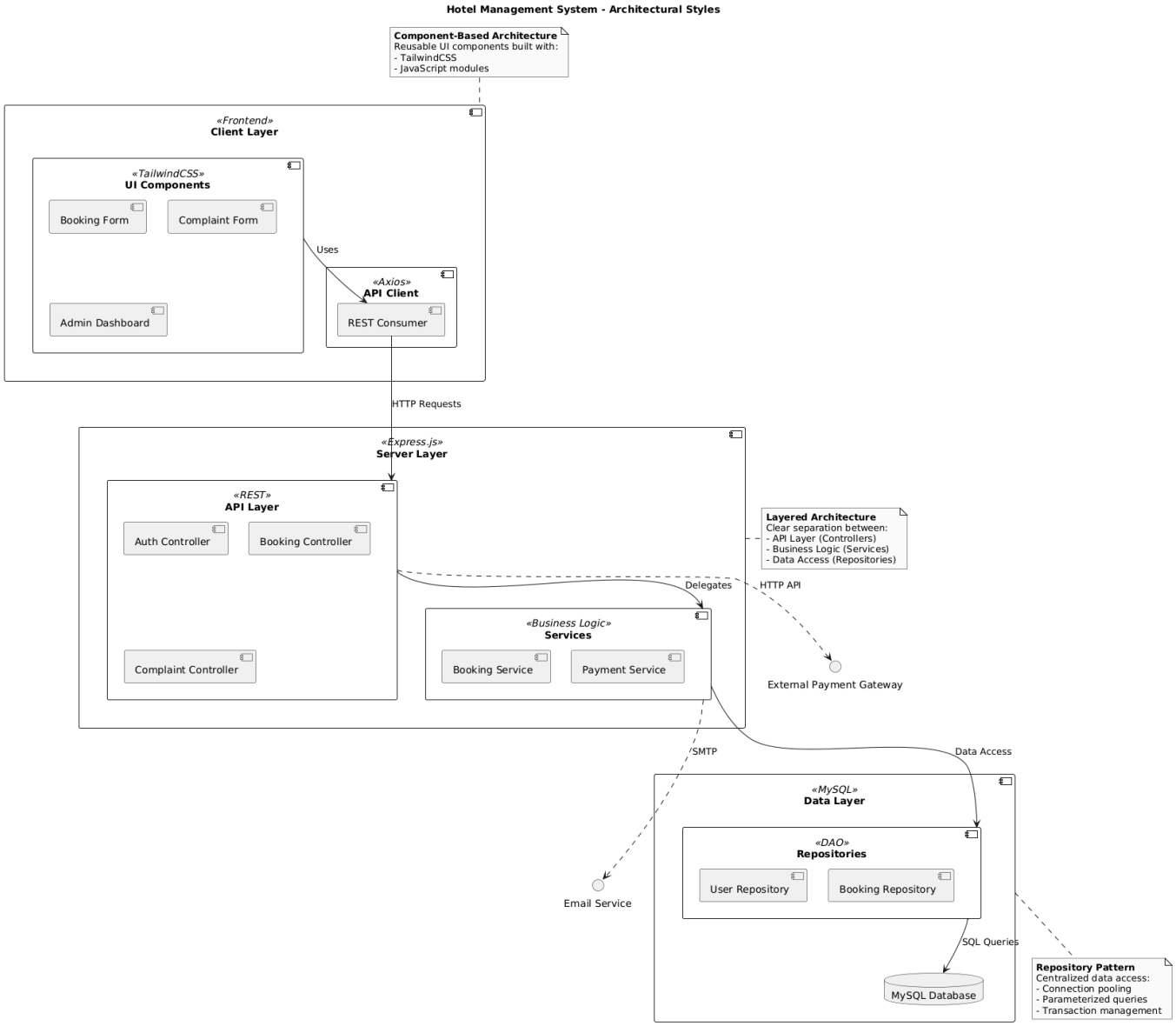
10. System Architecture Diagrams

10.1 Identifying Subsystems - UML package diagram



- **User Authentication**: Handles login, registration, and session management with role-based access control (guest, staff, admin)
- **Booking Management**: Core reservation system including search, booking creation, modifications, and check-in/out operations
- **Complaint Handling**: Processes guest complaints and special requests with tracking and resolution workflows
- **Feedback System**: Manages guest feedback collection, ratings, and display
- **Administration**: Backend management for users, rooms, and reporting

10.2 Architecture Styles



Hotel Management System - Architectural Styles Description

The system is designed using a **multi-layered architecture** that ensures modularity, scalability, and maintainability.

Client Layer:

- Built with TailwindCSS and JavaScript.
- Composed of reusable UI components (Booking Form, Complaint Form, Admin Dashboard).
- Communicates with the server through an Axios-based API client.
- Follows a **Component-Based Architecture**.

Server Layer:

- Implemented using Express.js.
- Divided into an API Layer (Controllers) and Services Layer (Business Logic).
- The API Layer handles HTTP requests, while Services manage core logic such as booking and payment processing.
- Adopts a **Layered Architecture** to separate concerns clearly.

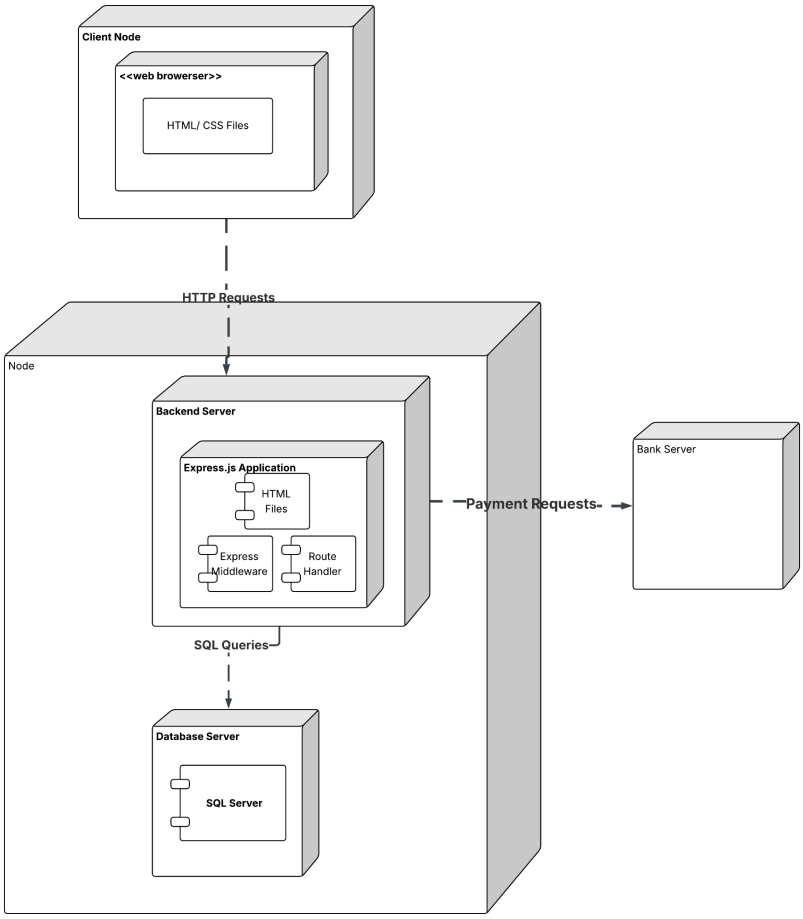
Data Layer:

- Utilizes MySQL for data storage.
- Accessed through Repositories (User Repository, Booking Repository) following the **Repository Pattern**.
- Ensures centralized data access, transaction management, and query security.

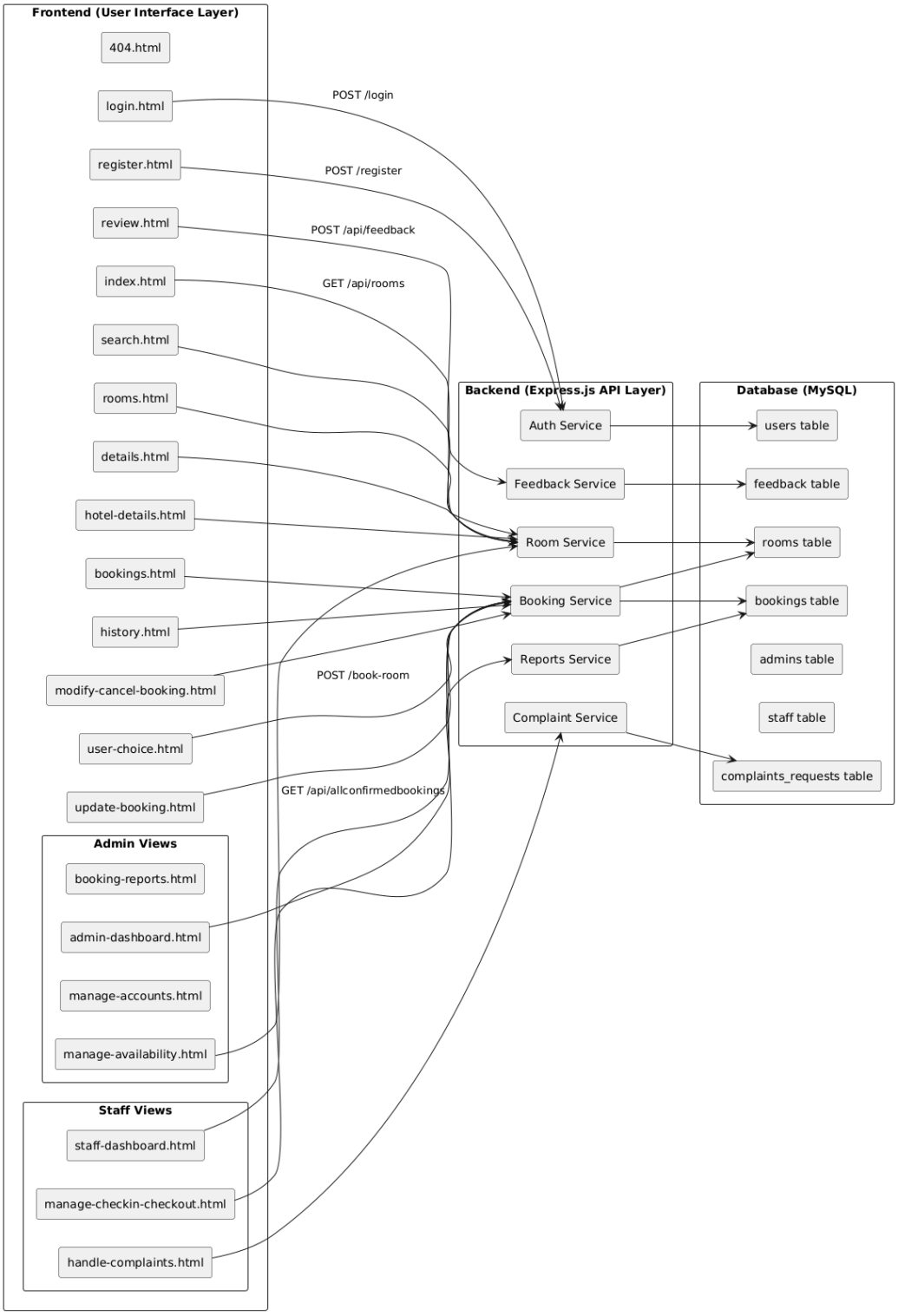
External Dependencies:

- Integrates with an external Payment Gateway for transaction processing.
- Utilizes an Email Service (SMTP) for customer communications.

10.3 Deployment diagram for client deployments



10.4 Component diagrams



11. Design (all sprint 3 items)

Task	Description	Planned Date	Assignee
Manage Room Availability (US12)	Develop page to allow staff to mark rooms as available or unavailable.	April 15-18	Haleema Tahir Malik (Frontend), M. Daniyal Aziz (Backend), Moaz Farooq (Database)
Update Check-In/Check-Out Status (US13)	Build functionality to update guest check-in/check-out status.	April 15-19	Haleema Tahir Malik (Frontend), M. Daniyal Aziz (Backend), Moaz Farooq (Database)
Handle Customer Complaints or Special Requests (US14)	Implement a page for staff to manage customer complaints or special service requests.	April 18-20	Haleema Tahir Malik (Frontend), M. Daniyal Aziz (Backend), Moaz Farooq (Database)
Modify/Cancel Booking (Staff) (US15)	Create functionality for staff to modify or cancel bookings on behalf of users.	April 20	Haleema Tahir Malik (Frontend), M. Daniyal Aziz (Backend), Moaz Farooq (Database)
View Booking History (US16)	Develop page to allow staff to view all guest booking history.	April 21	Haleema Tahir Malik (Frontend), M. Daniyal Aziz (Backend), Moaz Farooq (Database)
Manage Room Listings (Admin) (US17)	Build an admin page to add, update, and remove room listings.	April 20-22	Haleema Tahir Malik (Frontend), M. Daniyal Aziz (Backend), Moaz Farooq (Database)
Manage User and Staff Accounts (Admin) (US18)	Develop functionality for admin to create, update, and delete user/staff accounts.	April 20-23	Haleema Tahir Malik (Frontend), M. Daniyal Aziz (Backend), Moaz Farooq (Database)

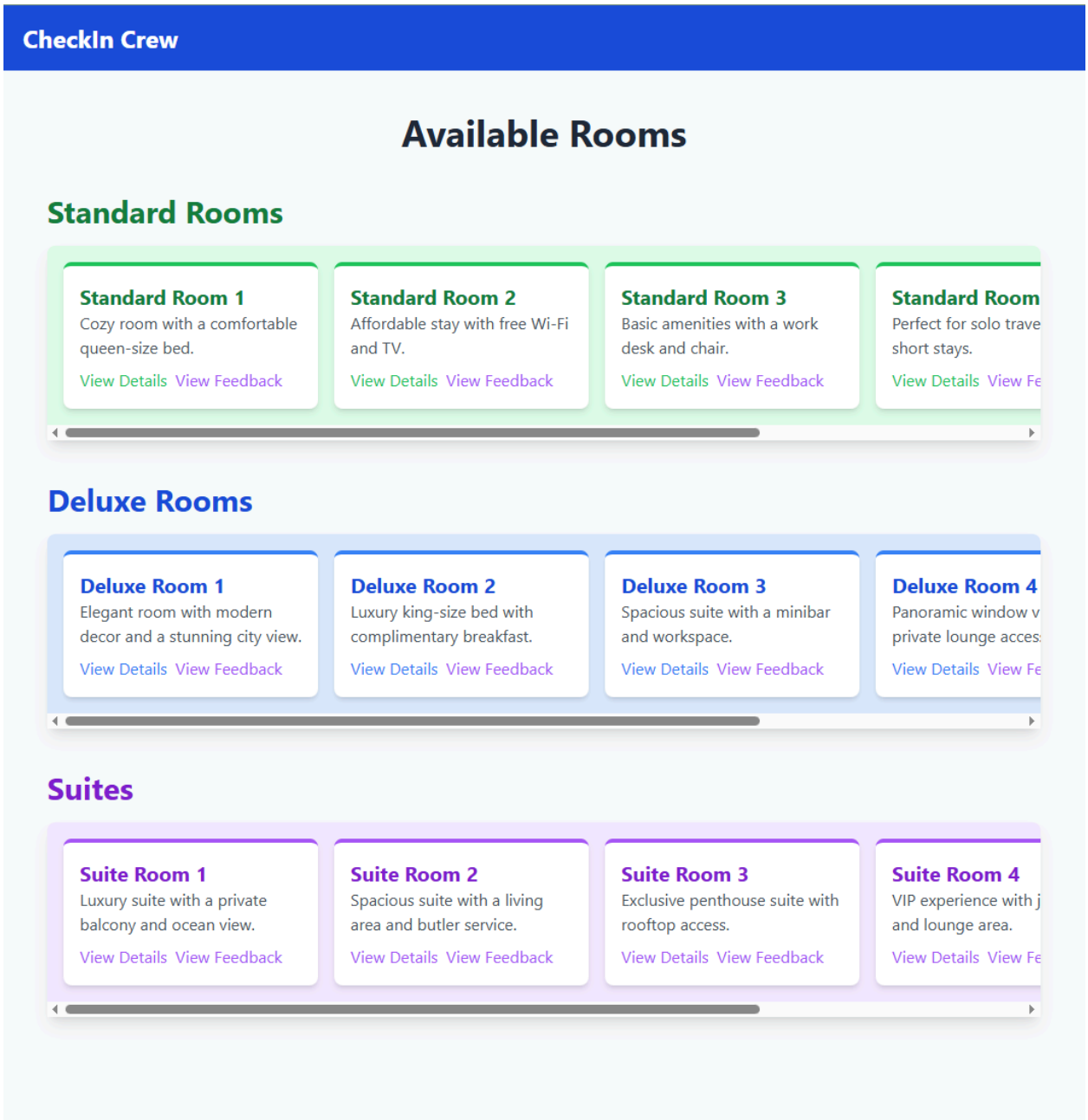
Generate Booking Reports (Admin) (US19)	Implement a system to generate booking performance and occupancy reports.	April 22-23	Haleema Tahir Malik (Frontend), M. Daniyal Aziz (Backend), Moaz Farooq (Database)

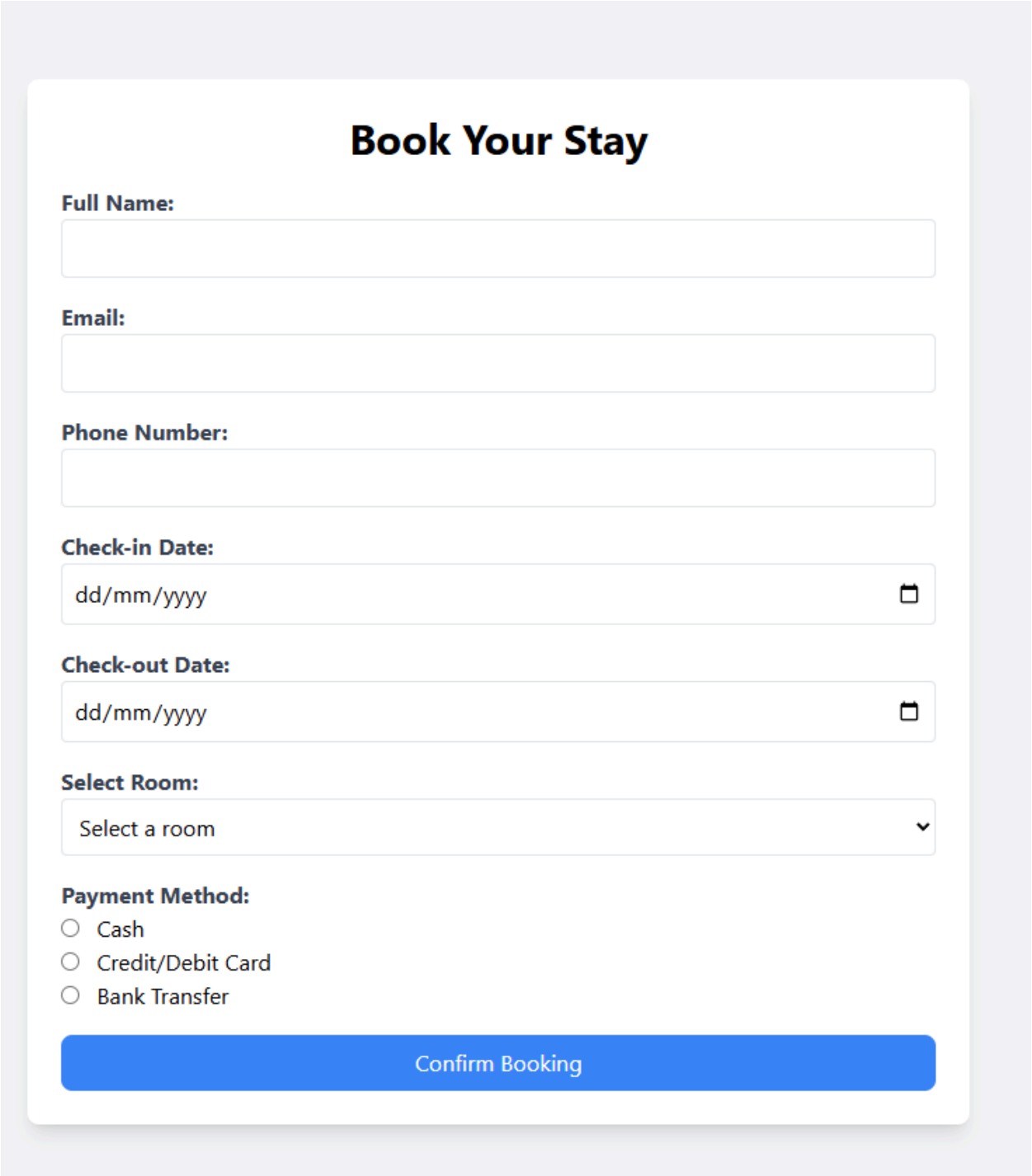
Github Link to code:

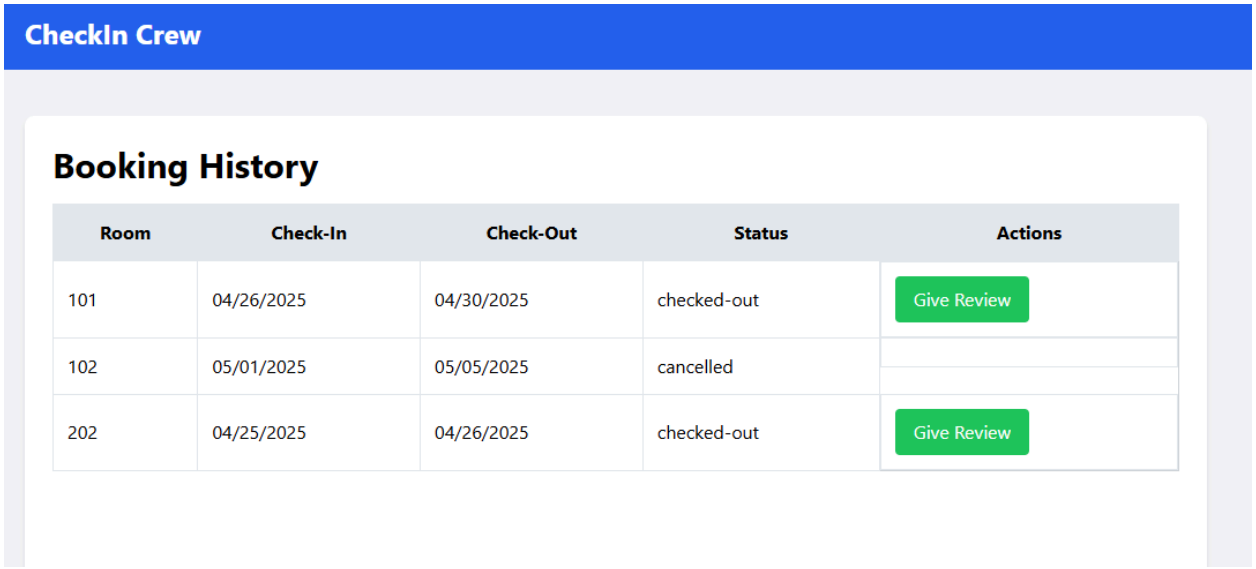
github.com/Daniyal1234-alt/checkin-crew

12. Actual implementation screenshots

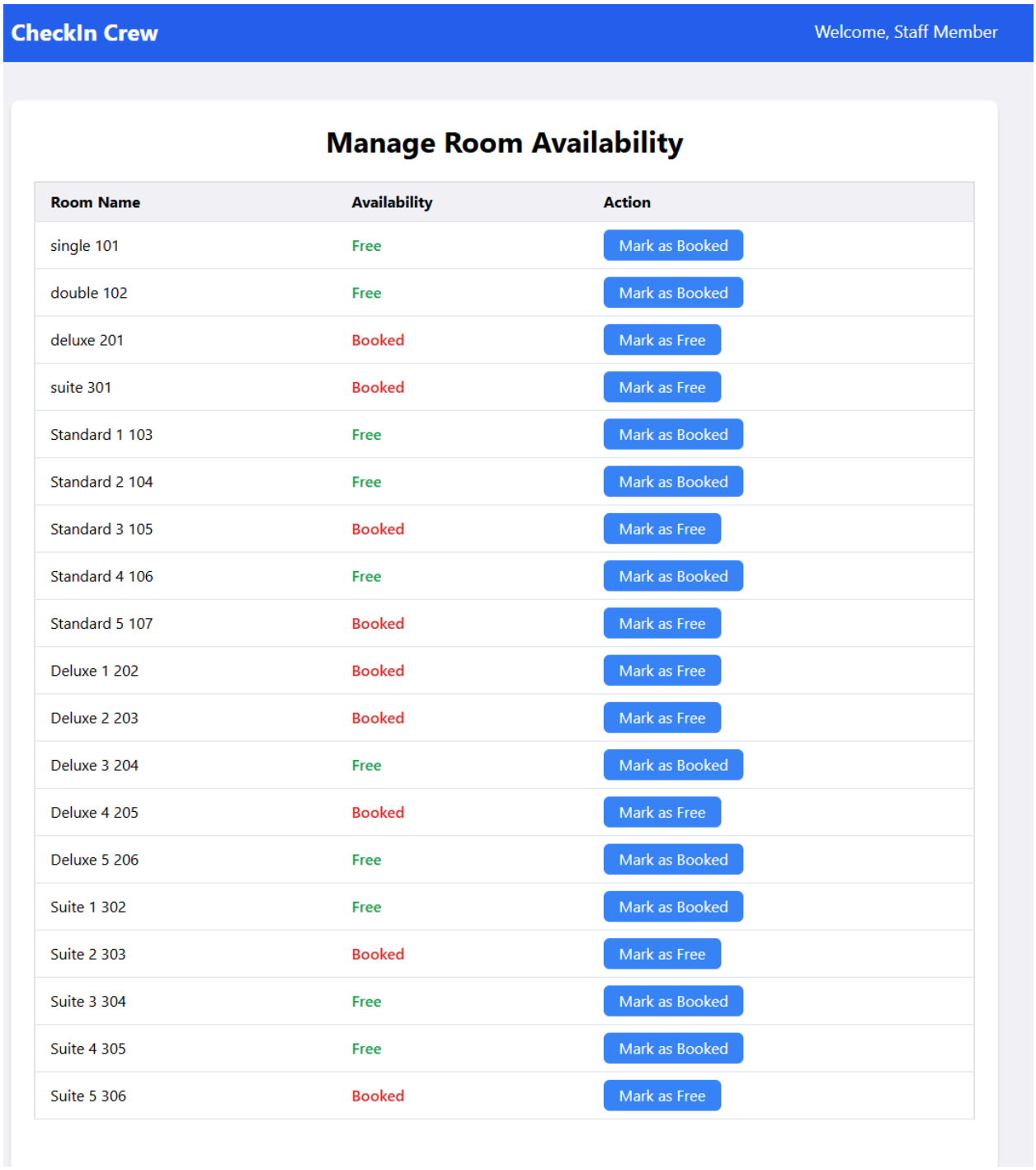
Guest:

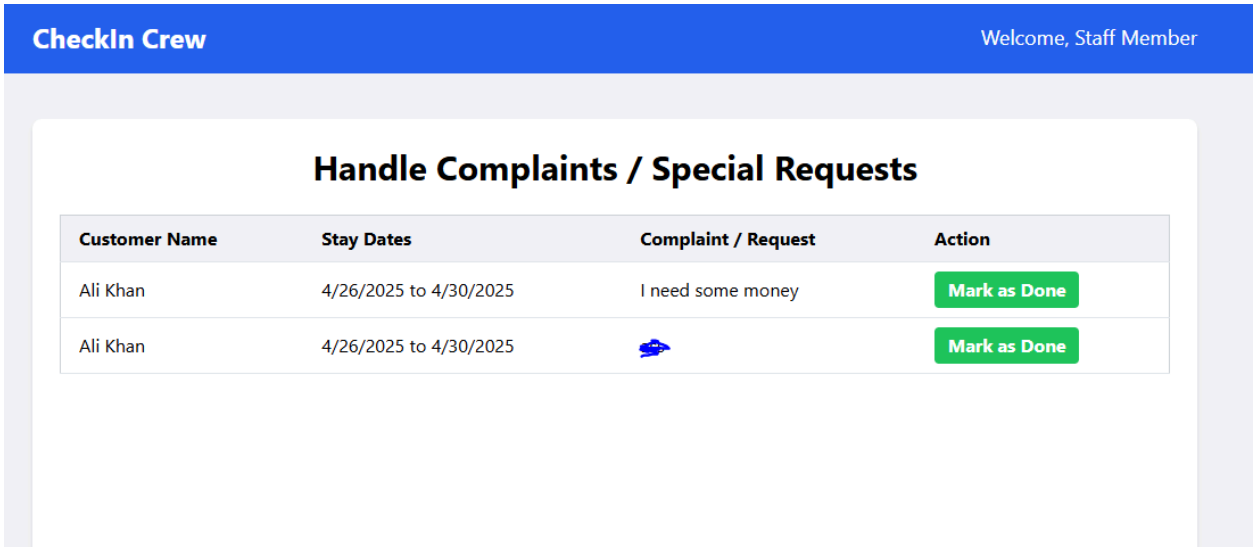


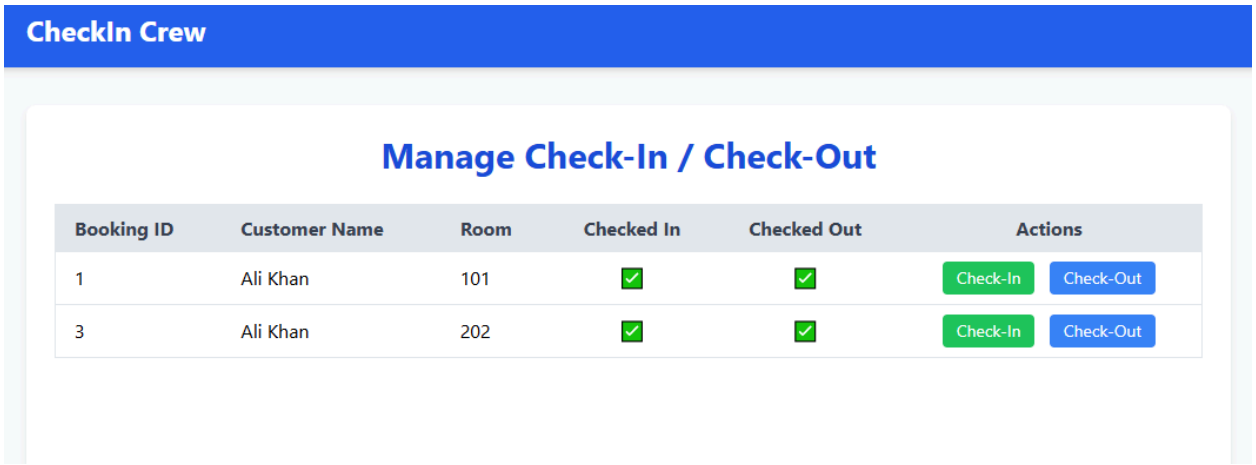

Book Your Stay
Full Name:
Email:
Phone Number:
Check-in Date:
dd/mm/yyyy
Check-out Date:
dd/mm/yyyy
Select Room:
Select a room
Payment Method:
Cash
Credit/Debit Card
Bank Transfer
Confirm Booking


CheckIn Crew
Booking History
Room
Check-In
Check-Out
Status
Actions
101
04/26/2025
04/30/2025
checked-out
Give Review
102
05/01/2025
05/05/2025
cancelled
202
04/25/2025
04/26/2025
checked-out
Give Review

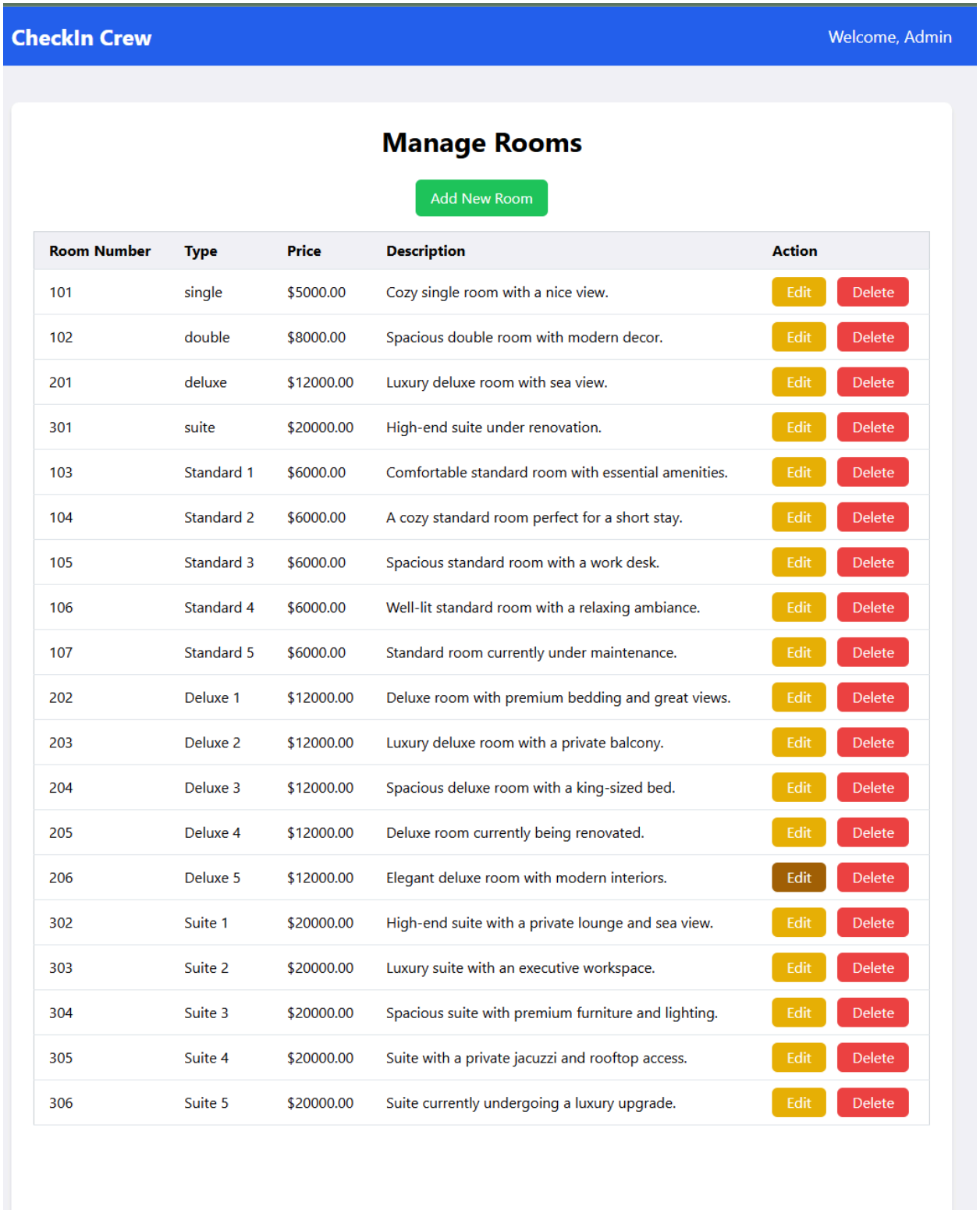
Staff:

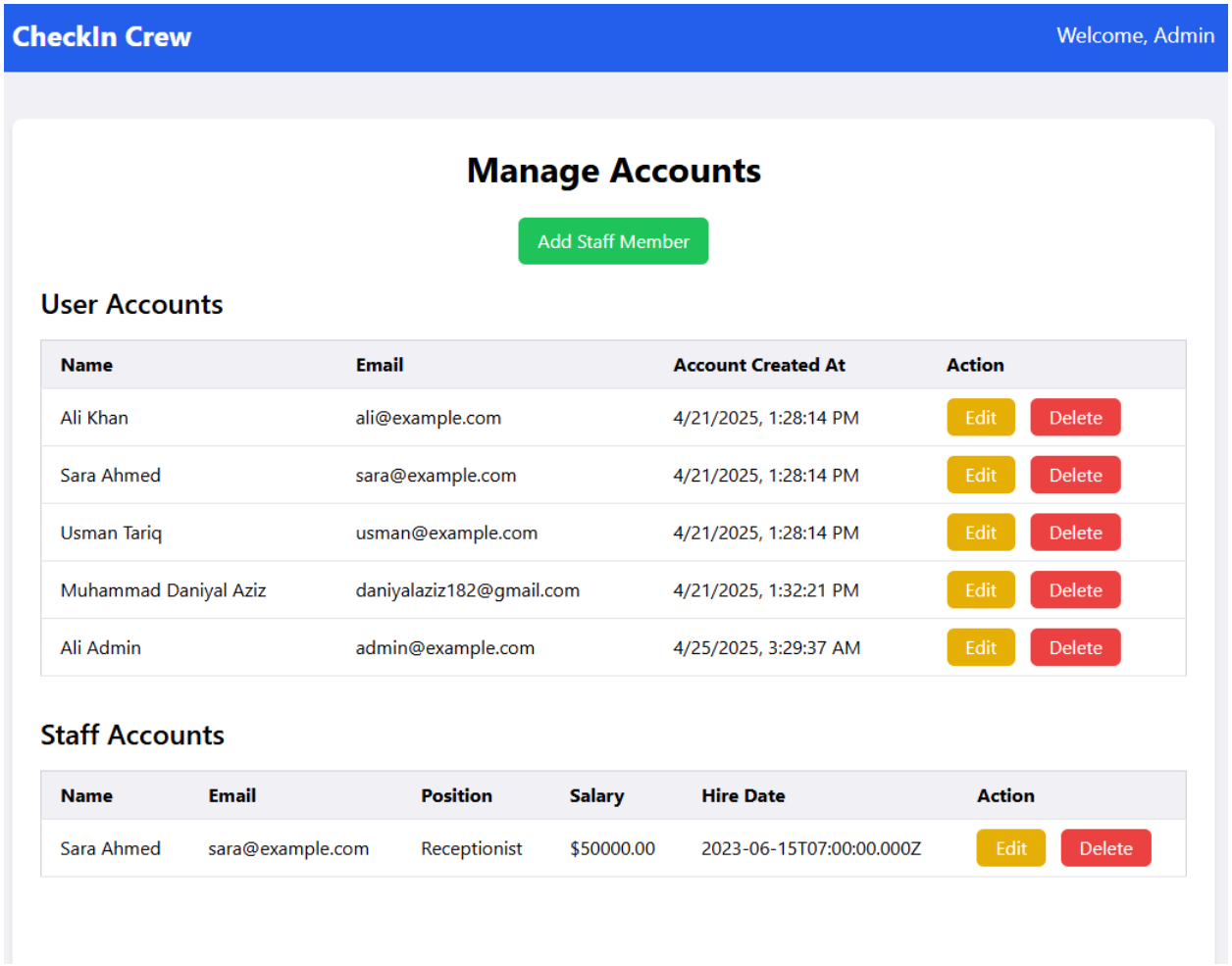

CheckIn Crew
Welcome, Staff Member
Manage Room Availability
Room Name
Availability
Action
single 101
Free
Mark as Booked
double 102
Free
Mark as Booked
deluxe 201
Booked
Mark as Free
suite 301
Booked
Mark as Free
Standard 1 103
Free
Mark as Booked
Standard 2 104
Free
Mark as Booked
Standard 3 105
Booked
Mark as Free
Standard 4 106
Free
Mark as Booked
Standard 5 107
Booked
Mark as Free
Deluxe 1 202
Booked
Mark as Free
Deluxe 2 203
Booked
Mark as Free
Deluxe 3 204
Free
Mark as Booked
Deluxe 4 205
Booked
Mark as Free
Deluxe 5 206
Free
Mark as Booked
Suite 1 302
Free
Mark as Booked
Suite 2 303
Booked
Mark as Free
Suite 3 304
Free
Mark as Booked
Suite 4 305
Free
Mark as Booked
Suite 5 306
Booked
Mark as Free


CheckIn Crew
Welcome, Staff Member
Handle Complaints / Special Requests
Customer Name
Stay Dates
Complaint / Request
Action
Ali Khan
4/26/2025 to 4/30/2025
I need some money
Mark as Done
Ali Khan
4/26/2025 to 4/30/2025
Mark as Done


CheckIn Crew
Manage Check-In / Check-Out
Booking ID
Customer Name
Room
Checked In
Checked Out
Actions
1
Ali Khan
101
Check-In
Check-Out
3
Ali Khan
202
Check-In
Check-Out

Admin:


CheckIn Crew
Welcome, Admin
Manage Rooms
Add New Room
Room Number Type Price Description Action
101 single $5000.00 Cozy single room with a nice view. Edit Delete
102 double $8000.00 Spacious double room with modern decor. Edit Delete
201 deluxe $12000.00 Luxury deluxe room with sea view. Edit Delete
301 suite $20000.00 High-end suite under renovation. Edit Delete
103 Standard 1 $6000.00 Comfortable standard room with essential amenities. Edit Delete
104 Standard 2 $6000.00 A cozy standard room perfect for a short stay. Edit Delete
105 Standard 3 $6000.00 Spacious standard room with a work desk. Edit Delete
106 Standard 4 $6000.00 Well-lit standard room with a relaxing ambiance. Edit Delete
107 Standard 5 $6000.00 Standard room currently under maintenance. Edit Delete
202 Deluxe 1 $12000.00 Deluxe room with premium bedding and great views. Edit Delete
203 Deluxe 2 $12000.00 Luxury deluxe room with a private balcony. Edit Delete
204 Deluxe 3 $12000.00 Spacious deluxe room with a king-sized bed. Edit Delete
205 Deluxe 4 $12000.00 Deluxe room currently being renovated. Edit Delete
206 Deluxe 5 $12000.00 Elegant deluxe room with modern interiors. Edit Delete
302 Suite 1 $20000.00 High-end suite with a private lounge and sea view. Edit Delete
303 Suite 2 $20000.00 Luxury suite with an executive workspace. Edit Delete
304 Suite 3 $20000.00 Spacious suite with premium furniture and lighting. Edit Delete
305 Suite 4 $20000.00 Suite with a private jacuzzi and rooftop access. Edit Delete
306 Suite 5 $20000.00 Suite currently undergoing a luxury upgrade. Edit Delete


CheckIn Crew
Welcome, Admin
Manage Accounts
Add Staff Member
User Accounts
Name
Email
Account Created At
Action
Ali Khan
ali@example.com
4/21/2025, 1:28:14 PM
Edit
Delete
Sara Ahmed
sara@example.com
4/21/2025, 1:28:14 PM
Edit
Delete
Usman Tariq
usman@example.com
4/21/2025, 1:28:14 PM
Edit
Delete
Muhammad Daniyal Aziz
daniyalaziz182@gmail.com
4/21/2025, 1:32:21 PM
Edit
Delete
Ali Admin
admin@example.com
4/25/2025, 3:29:37 AM
Edit
Delete
Staff Accounts
Name
Email
Position
Salary
Hire Date
Action
Sara Ahmed
sara@example.com
Receptionist
$50000.00
2023-06-15T07:00:00.000Z
Edit
Delete

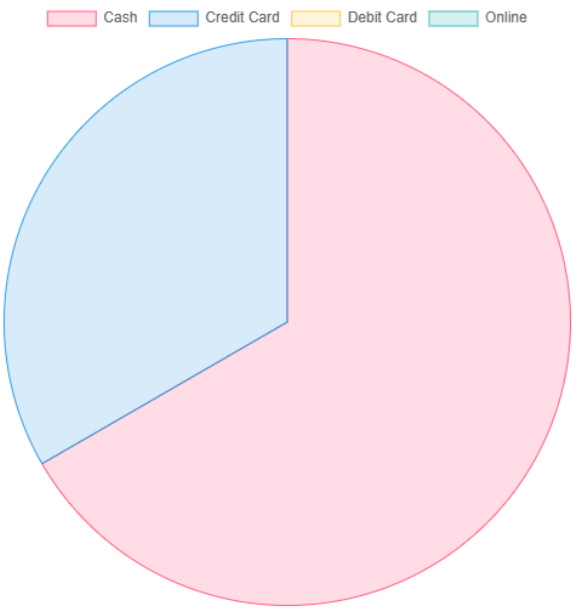
CheckIn Crew Welcome, Admi

Booking Reports & Trends

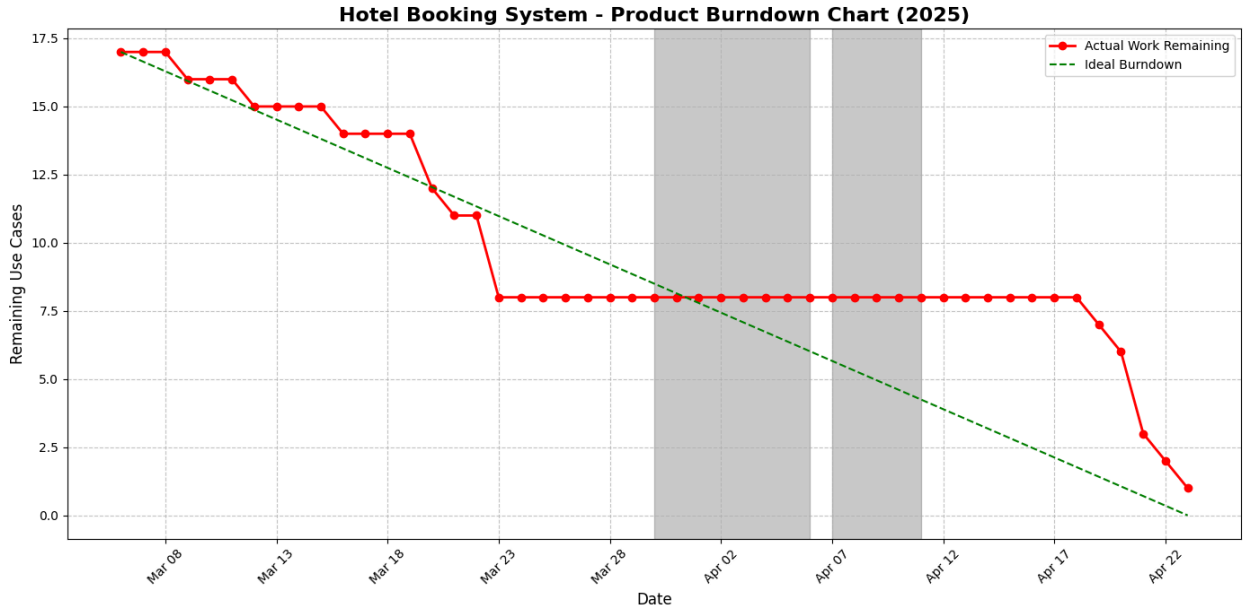
Booking ID	User Name	Room Number	Check-in Date	Check-out Date	Status	Payment Method	Total Price	Booking Date
1	Ali Khan	101	4/26/2025	4/30/2025	checked-out	credit card	$25000.00	4/21/2025, 1:28:14 PM
2	Ali Khan	102	5/1/2025	5/5/2025	cancelled	cash	$32000.00	4/21/2025, 1:28:14 PM
3	Ali Khan	202	4/25/2025	4/26/2025	checked-out	cash	$12000.00	4/25/2025, 3:10:25 AM

Booking Trends (Last 6 Months)

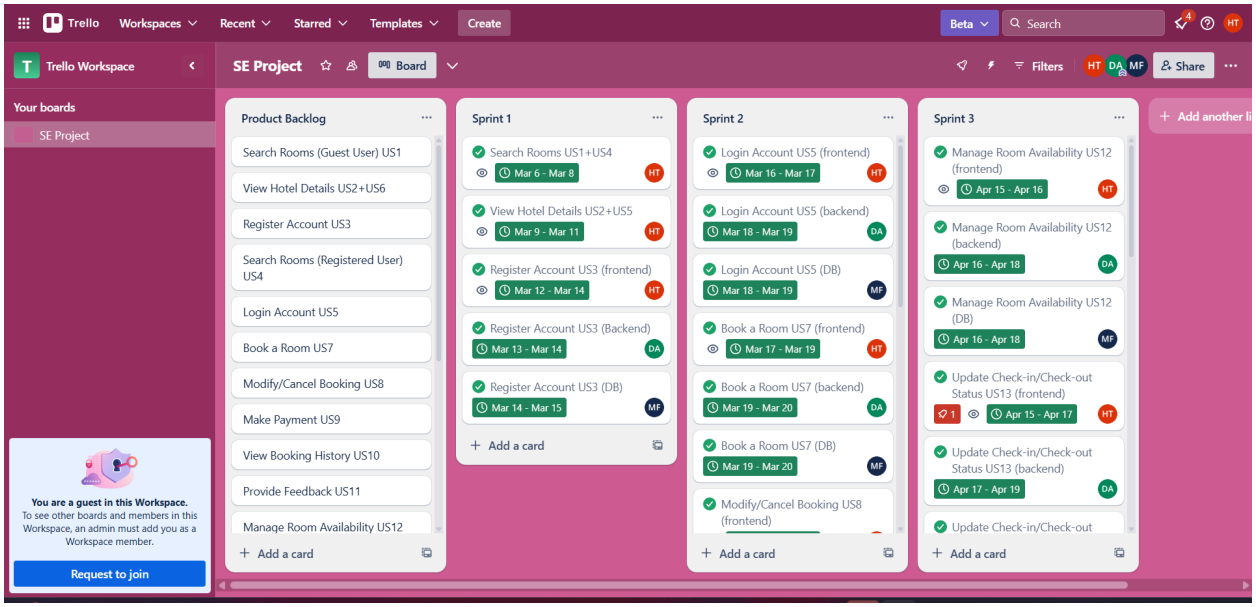
Payment Method Distribution



13. Product Burndown chart for the project



14. Trello board screen shot



15. Test Cases -Black box

15.1 Equivalence Class Partitioning Table

Input	Valid Classes	Invalid Classes
Room Price	Price > 0	Price ≤ 0
Number of Guests	1 to 5 guests	0 guests, more than 5 guests
Check-In Date	Future date (today or later)	Past date
Check-Out Date	After Check-In Date	Same as or before Check-In Date
Email Address	Valid email (e.g., abc@example.com)	Missing "@" or "." symbols
Password	Length between 6-20 characters	Less than 6 or more than 20 characters
Search Room Input	Location entered	Location field empty
Booking Amount	Booking amount > 0	Booking amount ≤ 0
Feedback Rating	1 to 5 stars	Less than 1 or more than 5 stars
Feedback Comment	Text entered	No text entered

15.2 Boundary Value Analysis Table

Input	Boundary Values
Number of Guests	0 (invalid), 1 (valid min), 3 (normal), 5 (valid max), 6 (invalid)
Room Price ($)	0 (invalid), 1 (valid min), 100 (normal), 999 (high valid), 1000+ (invalid)
Check-In Date	Yesterday (invalid), Today (valid), +1 day (valid)
Password Length	5 (invalid), 6 (valid min), 10 (normal), 20 (valid max), 21 (invalid)
Feedback Rating	0 (invalid), 1 (valid min), 3 (normal), 5 (valid max), 6 (invalid)

15.3 Test Cases Table

Test Case ID	Description	Preconditions	Input Data	Expected Result	Status
TC-1	Register a new user	Registration page open	Valid Email + Password	User registered successfully	Pass
TC-2	Register with invalid email	Registration page open	Invalid Email + Password	Show "Invalid Email" error	Pass
TC-3	Search available rooms	Home page loaded	Location = "New York"	Display list of available rooms	Pass
TC-4	Book a room with valid data	Logged in user	Room = Deluxe, Guests = 2	Booking successful	Pass
TC-5	Attempt booking with 0 guests	Logged in user	Guests = 0	Show error: "Invalid number of guests"	Pass
TC-6	Give 5-star feedback	Booking completed	Rating = 5, Comment = "Excellent"	Feedback submitted	Pass

TC-7	Submit feedback without comment	Booking completed	Rating = 4, Empty comment	Show error: "Comment required"	Pass
TC-8	Cancel a booking	User has existing booking	Click "Cancel"	Booking marked as cancelled	Pass
TC-9	Login with wrong password	Login page open	Email = correct, Password = wrong	Show error: "Incorrect credentials"	Pass
TC-10	View booking history	Logged in user	None	Booking history displayed correctly	Pass
TC-11	Staff Login Successful	Staff account exists	Valid Staff Email + Password	Staff redirected to dashboard	Pass
TC-12	Staff views booking history	Staff dashboard open	Click "View Booking History"	List of all bookings displayed	Pass
TC-13	Staff handles customer complaint	Staff dashboard open	Click "Handle Complaints", select request	Complaint details displayed	Pass
TC-14	Staff updates Check-In status	Staff dashboard open	Booking ID + Change status	Check-In status updated	Pass
TC-15	Staff cancels booking	Staff dashboard open	Click "Cancel" on booking	Booking marked as "Cancelled"	Pass
TC-16	Admin Login Successful	Admin account exists	Valid Admin Email + Password	Admin redirected to dashboard	Pass
TC-17	Admin adds new room listing	Admin dashboard open	Room details filled	Room added to system	Pass
TC-18	Admin removes user account	Admin dashboard open	Select user → Click "Delete"	User account deleted	Pass
TC-19	Admin generates booking report	Admin dashboard open	Click "Generate Report"	Report file downloaded or displayed	Pass

TC-20	Customer provides feedback after stay	Booking status completed	Select booking + Leave feedback	Feedback stored in system	Pass
TC-21	Customer modifies upcoming booking	Upcoming booking exists	Modify check-in/check-out dates	Booking details updated successfully	Pass
TC-22	Customer cancels upcoming booking	Upcoming booking exists	Click "Cancel"	Booking marked cancelled	Pass
TC-23	Guest searches room without logging in	Home page loaded	Enter search query	Available rooms shown	Pass
TC-24	Registration fails with weak password	Registration form open	Password < 6 characters	Error: "Password too weak"	Pass
TC-25	Feedback rating outside 1-5 not allowed	Feedback form open	Rating = 6	Error: "Invalid rating"	Pass

16. Test Cases -White box

16.1 Code Coverage:

All files checkin-crew/routes

67.12% Statements 296/441 **61.86%** Branches 73/118 **78.04%** Functions 32/41 **67.27%** Lines 296/440

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
authRoutes.js	71.79%	28/39	52.94%	9/17	100%	2/2	71.79%	28/39
bookingRoutes.js	68.75%	110/160	66%	33/50	91.66%	11/12	68.75%	110/160
feedbackRoutes.js	66.66%	38/57	70%	7/10	83.33%	5/6	66.66%	38/57
roomRoutes.js	62.79%	54/86	70.37%	19/27	77.77%	7/9	63.52%	54/85
staffRoutes.js	75.51%	37/49	50%	4/8	100%	4/4	75.51%	37/49
staticRoutes.js	61.11%	11/18	0%	0/2	20%	1/5	61.11%	11/18
userRoutes.js	56.25%	18/32	25%	1/4	66.66%	2/3	56.25%	18/32

All files checkin-crew/db

100% Statements 4/4 **100%** Branches 0/0 **100%** Functions 0/0 **100%** Lines 4/4

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
dbConfig.js	100%	1/1	100%	0/0	100%	0/0	100%	1/1
pool.js	100%	3/3	100%	0/0	100%	0/0	100%	3/3

All files checkin-crew

100% Statements 16/16 **100%** Branches 0/0 **100%** Functions 0/0 **100%** Lines 16/16

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File	Statements		Branches		Functions		Lines	
Server.js	100%	16/16	100%	0/0	100%	0/0	100%	16/16

16.2 Unit Testing and Coverage Report

Overview

This report outlines the unit testing strategy and coverage analysis performed for the backend system of the Hotel Management Application. The objective was to ensure functional reliability and maintainability through comprehensive test coverage using a modern JavaScript testing framework.

Testing Framework

- Tool Used: Jest
- Rationale: Jest is a robust testing framework tailored for Node.js environments. It supports mocking, assertions, and built-in coverage reporting, making it an ideal choice for backend API testing.

Scope of Testing

The test suite was designed to cover all core functionalities including:

- Room Management: Creating, retrieving, updating, and deleting room records
- Booking Operations: Booking rooms, updating booking details, checking availability, and cancelling bookings
- User Services: User login and registration
- Staff Management: Staff record operations (CRUD)
- Complaint and Feedback Handling: Submission and status tracking

Coverage Results

The test coverage metrics obtained are as follows:

Coverage Metric	Percentage
Statement Coverage	~70%
Branch Coverage	~30-40%
Function Coverage	~53%
Line Coverage	~70%

Note: The values are based on the Jest coverage report generated during the latest test run and can be found in the `coverage/lcov-report/index.html` file.

Coverage Highlights

Well-Covered Areas:

- All REST API endpoints were tested with both valid and invalid scenarios.
- Key business logic and response validations were verified.
- Transaction rollbacks and commits were tested using mock database connections.

Areas Not Covered:

- Third-party libraries such as `mysql2` and `bcrypt` are excluded from testing scope.
- Static file serving via `res.sendFile` was not tested due to limited logic involvement.
- Some edge-case branches in nested asynchronous logic were not reached in the current test set.

17. Work Division between group members

In our project, responsibilities were divided clearly to ensure smooth workflow and efficient development:

- **Haleema Tahir Malik**
  Role: Frontend Developer
  Responsibility:
  Haleema was responsible for designing and implementing the frontend of the Hotel Management System using HTML, Tailwind CSS, and basic JavaScript. She developed the user interfaces for guest users, registered users, staff members, and administrators, maintaining consistency in the theme, responsiveness, and user experience across the entire system.

- **Muhammad Daniyal Aziz**
  Role*:* Backend Developer
  Responsibility:
  Daniyal was responsible for building the backend logic of the system. He implemented server-side functionality using Node.js and Express.js, set up APIs for bookings, authentication, room management, and handled user requests by connecting frontend components with database operations.

- **Moaz Farooq**
  Role: Database Administrator
  Responsibility:
  Moaz was responsible for designing and managing the database schema using MySQL. He created all the required tables (users, rooms, bookings, feedback, etc.), wrote SQL queries for data retrieval and manipulation, and ensured smooth database connectivity and data integrity for the backend operations.

We closely coordinated using GitHub for version control and Trello for task management, ensuring efficient collaboration.

18. Lesson learnt by group

Throughout the course of this project, we collectively learned valuable technical, professional, and team collaboration lessons:

- **Team Coordination and Communication:**
  We learned the importance of effective communication and clear role division when working in a team. Regular meetings and task updates helped us stay on track.

- **Technical Skill Enhancement:**

  - Haleema improved her expertise in responsive frontend development using Tailwind CSS and JavaScript.
  - Daniyal strengthened his knowledge of backend APIs, authentication mechanisms, and server management.
  - Moaz gained hands-on experience in SQL database design, writing optimized queries, and ensuring backend database integration.

- **Problem Solving:**
  We faced challenges like merging frontend with backend properly, handling user session states, and managing booking cancellations dynamically. We learned to troubleshoot and adapt quickly.

- **Software Engineering Best Practices:**
  Applying the Scrum methodology, maintaining a clear sprint backlog, writing user stories, and testing systematically helped us develop a real-world project in a professional way.

- **Importance of Testing:**
  We realized that comprehensive testing (Black-box testing, Boundary Value Analysis) is critical to ensuring a robust and error-free system before final delivery.

- **Version Control and Collaboration Tools:**
  Working with GitHub and GitHub Desktop improved our ability to manage branches, pull requests, and handle team code merges smoothly.

In conclusion, this project not only improved our technical knowledge but also gave us a taste of real-world software development practices, preparing us for future professional projects.