**SDLC Activity Sheets**

**ACTIVITY SHEET 1 – Requirements Gathering (SRS Development)**

**Project Title:**  
**RoomRush – Hotel Booking Management System**

**Team Members:**  
Haripriya, Srujan, Lekha sai, Hemavarshini, Sampath, Harsha

**Task 1: Stakeholder Interview**

**Identified Stakeholders**

1. Hotel Manager
2. Receptionist / Staff
3. Customers (Guests)
4. System Administrator
5. Hotel Owner
6. Team Members

**Interview Questions**

1. What details are required from customers during room booking?
2. What types of rooms are available in the hotel (AC, Non-AC, Deluxe, etc.)?
3. How is room availability currently managed?
4. Should customers be allowed to cancel bookings online?
5. What payment methods should the system support?
6. Do staff members need login access to manage bookings?
7. Should booking confirmation be generated automatically?
8. What reports are required (daily bookings, revenue, occupancy)?
9. How should customer data be stored and secured?
10. Should the system support future expansion (more hotels/rooms)?

**Task 2: Extract Requirements**

|  |  |  |
| --- | --- | --- |
| **Requriement ID** | **TYPE(F/NF)** | **Requriement Description** |
| **R1** | **Functional** | **The system shall allow customers to book rooms based on availability.** |
| **R2** | **Functional** | **The system shall allow customers to cancel or modify bookings.** |
| **R3** | **Non -Functional** | **The system shall ensure data security and user authentication.** |

**Task 3: User Stories**

1. **As a customer, I want to view available rooms, so that I can choose a suitable room.**

**2. As a customer, I want to book a room online, so that I can reserve my stay easily.**

**3. As a customer, I want to cancel my booking, so that I can change my plans if needed.**

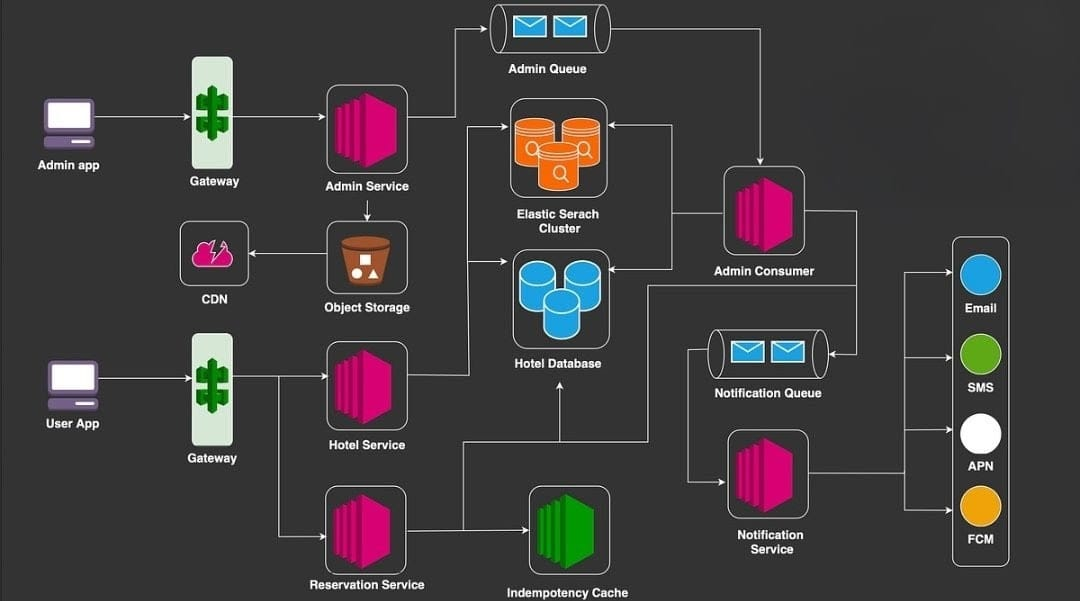
**4. As a receptionist, I want to manage bookings, so that room allocation is accurate.**

**5. As a user, I want to see room prices, so that I can choose within my budget.**

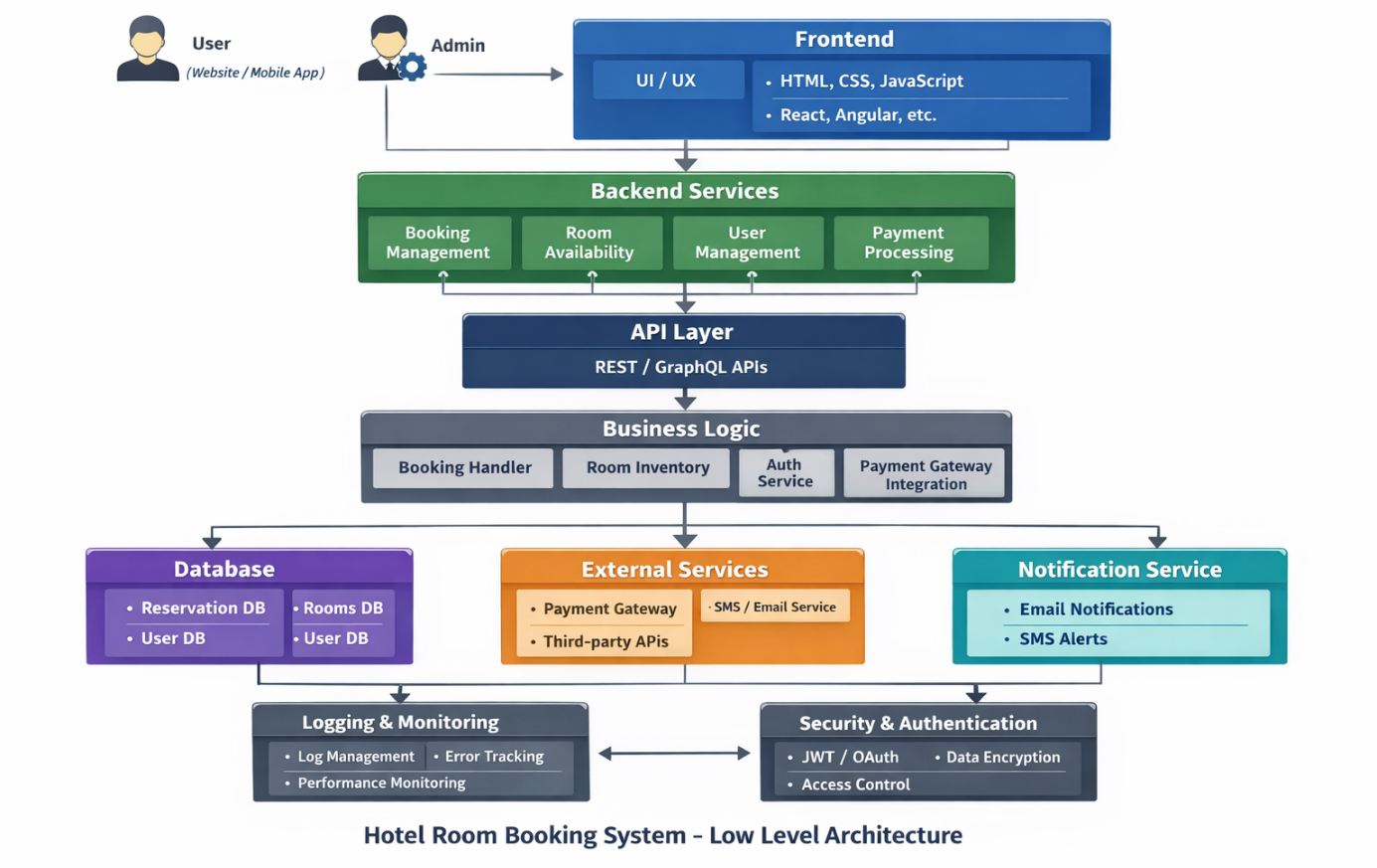
**ACTIVITY SHEET 2 — System Design (High-Level & Low-Level)**

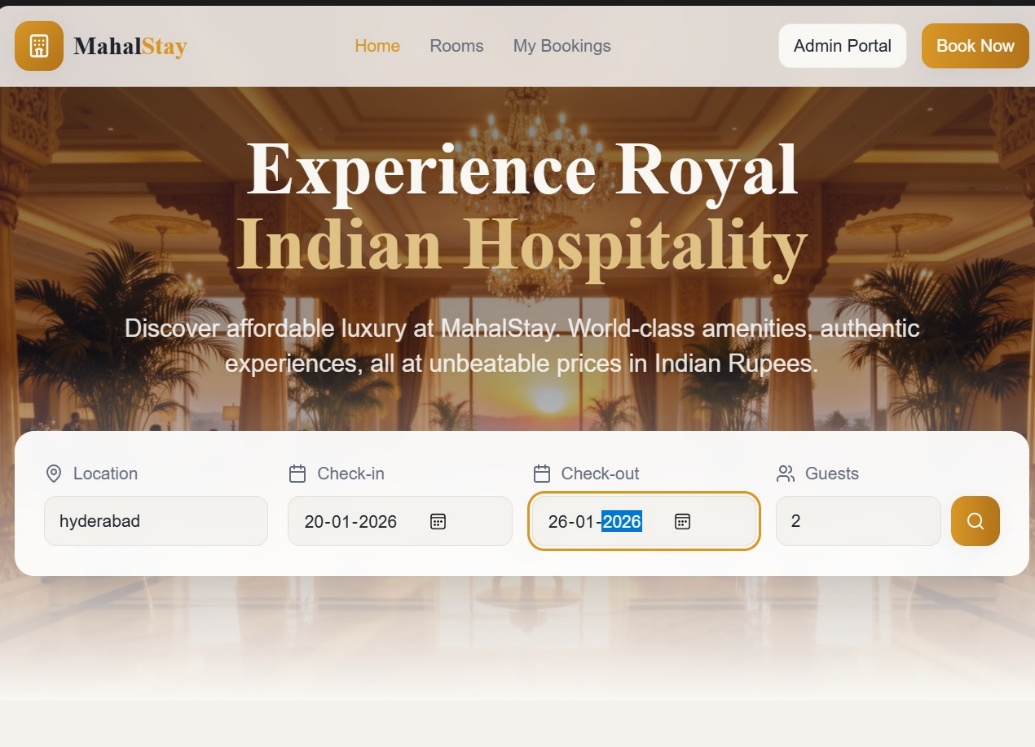
**Project: MAHALSTAY : HOTEL ROOM BOOKING SYSTEM**

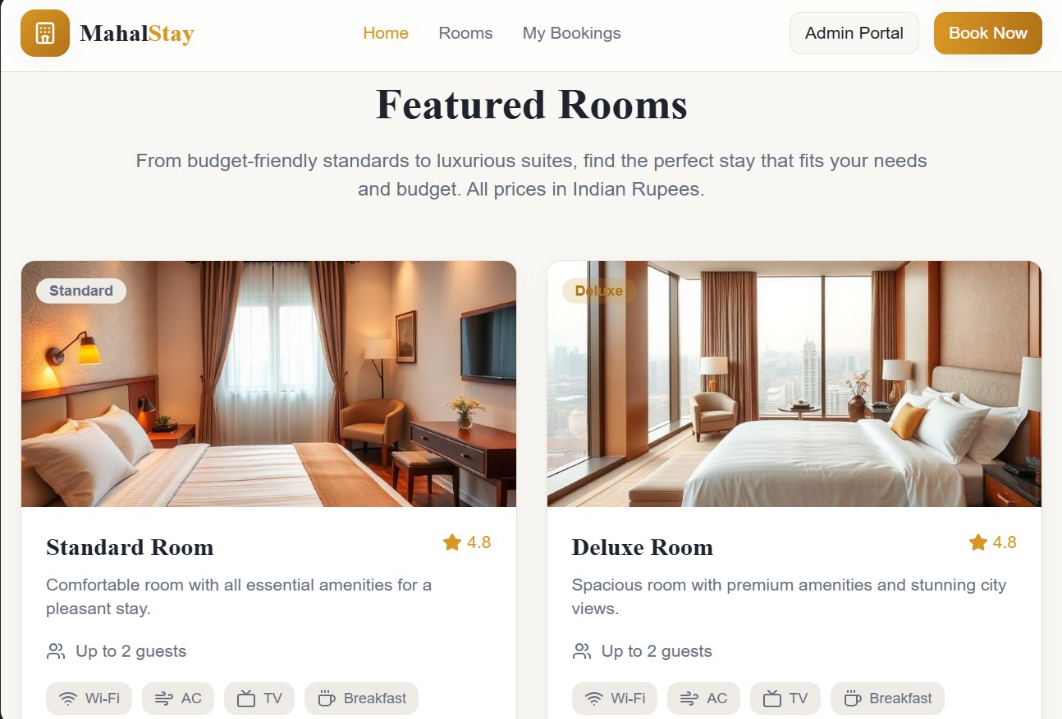
**Task 1: System Architecture Diagram**

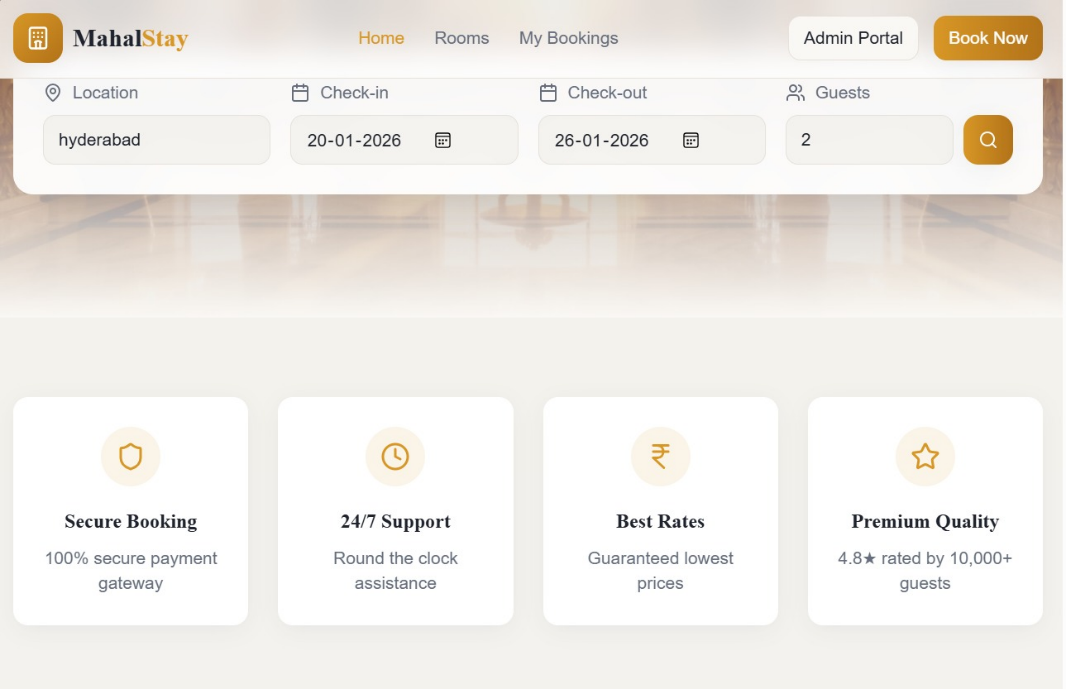
****

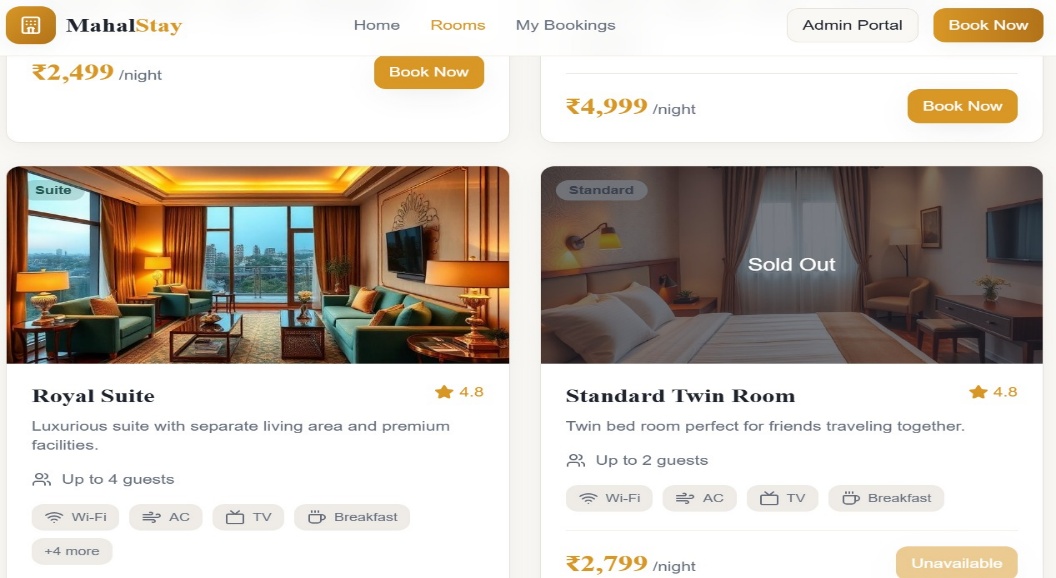
**LOW LEVEL DIAGRAM:**

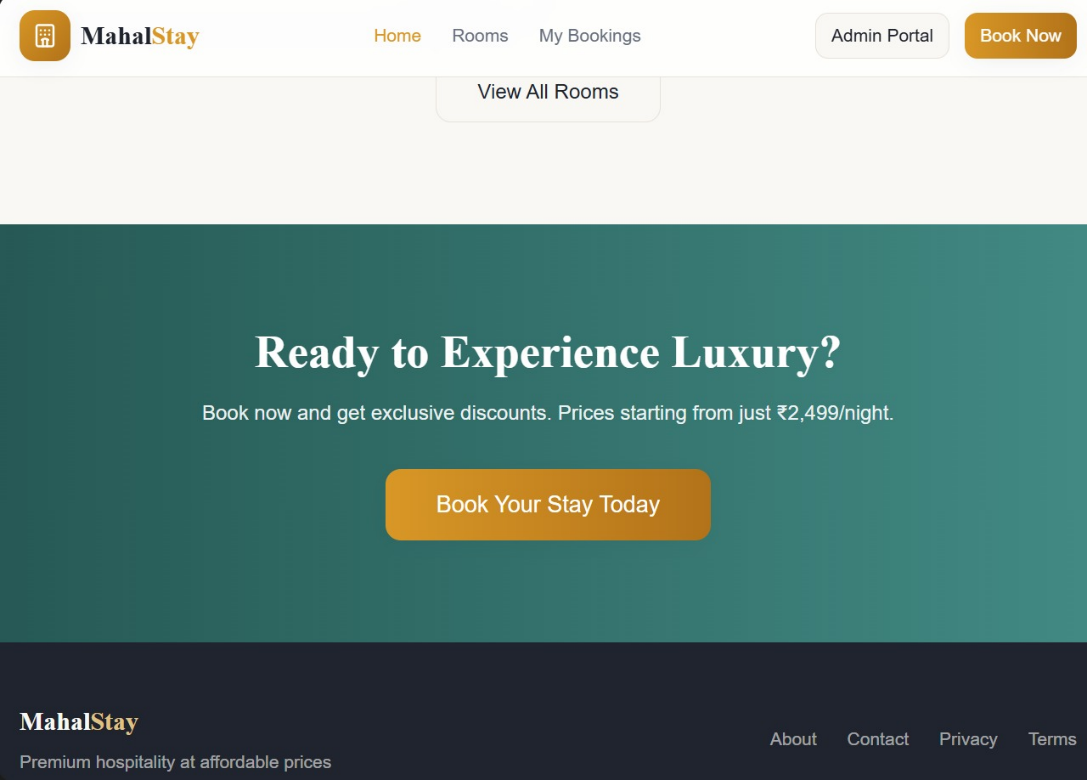


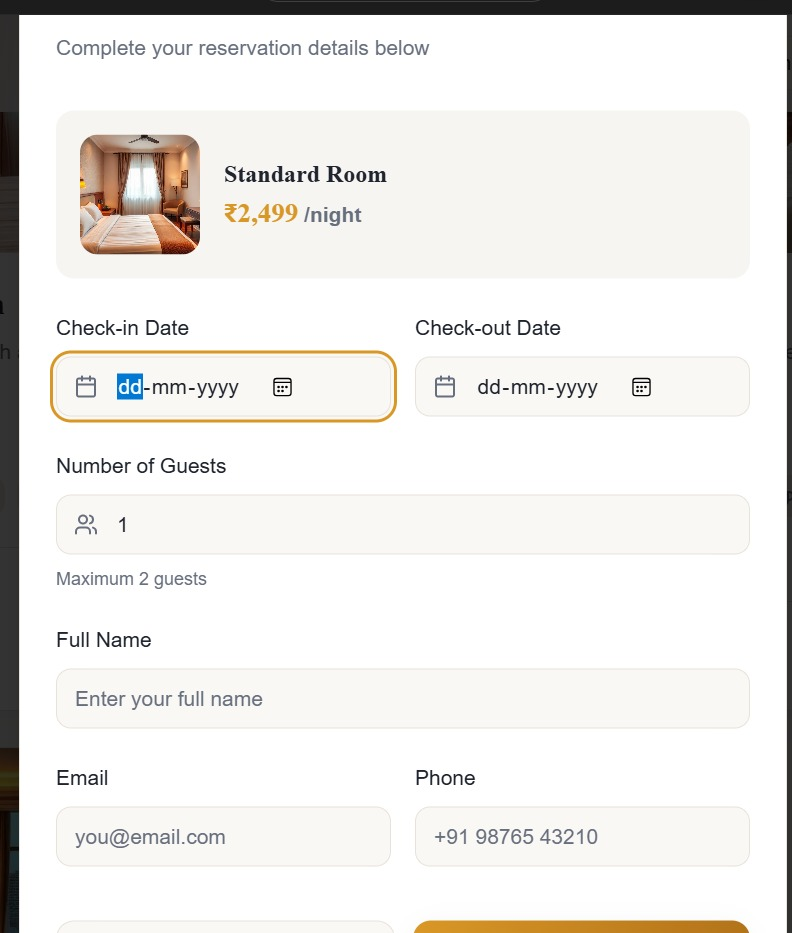
**Task 2: Create UI Wireframes : user phase**

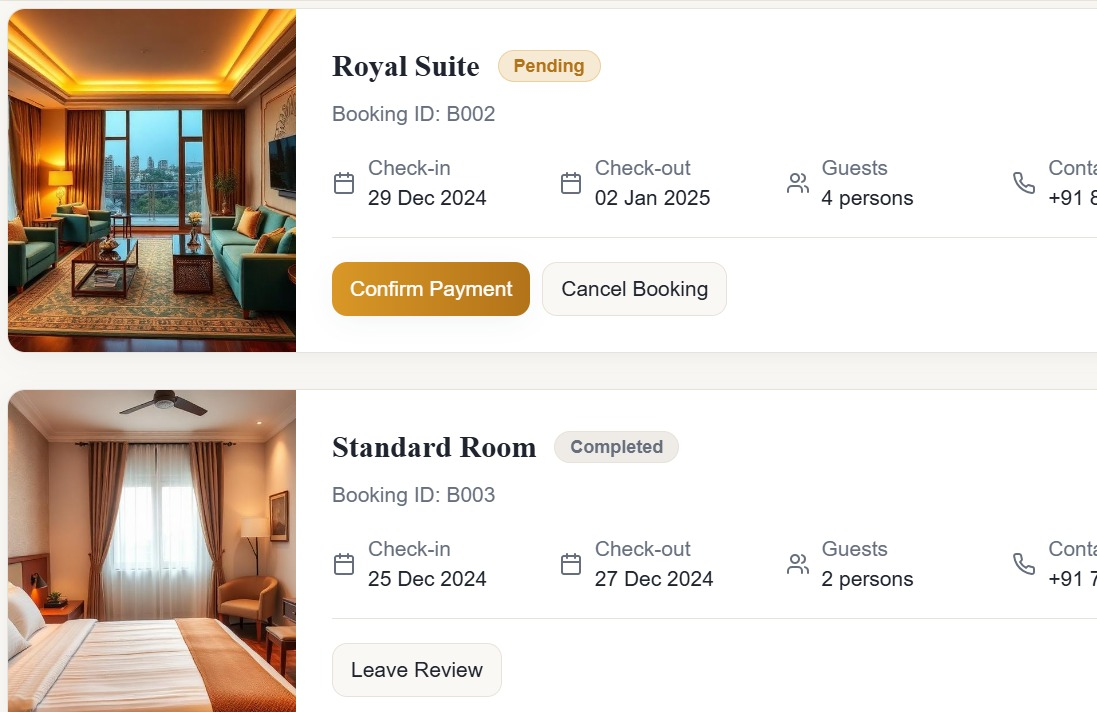
****



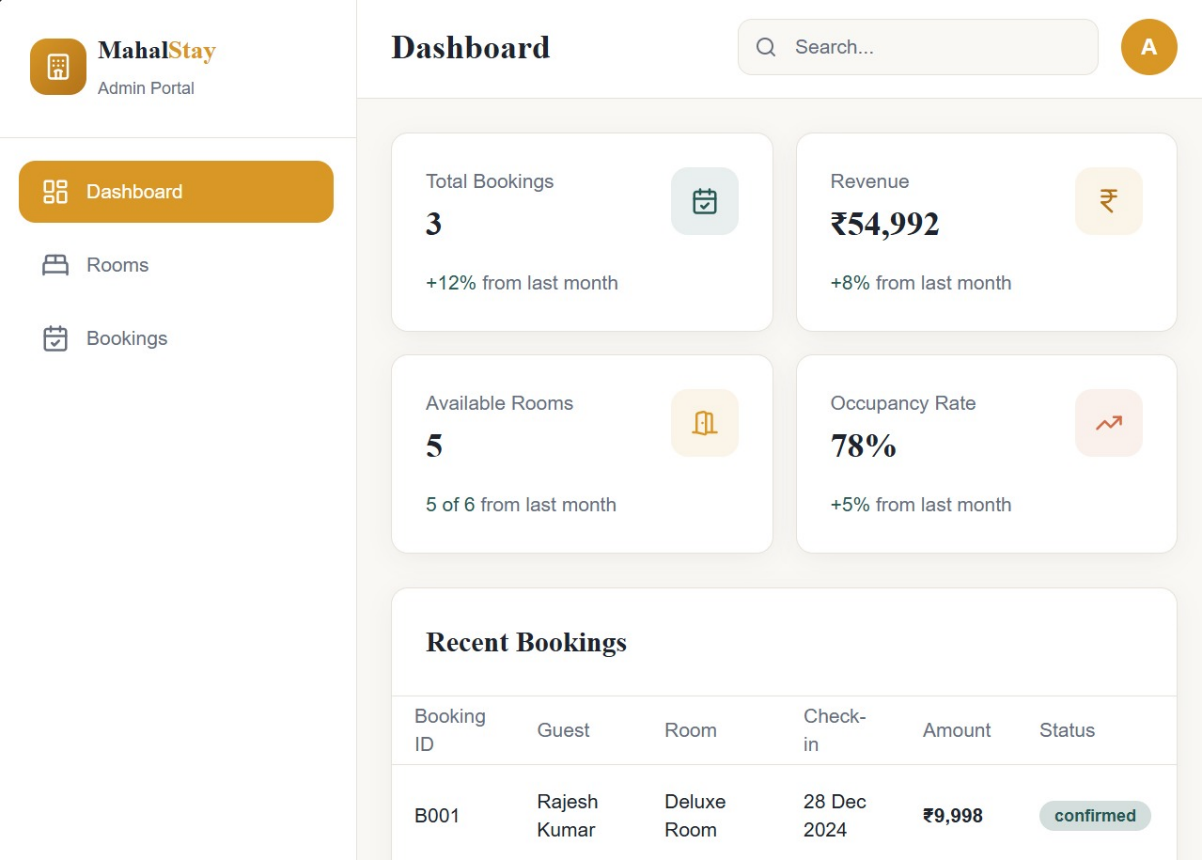




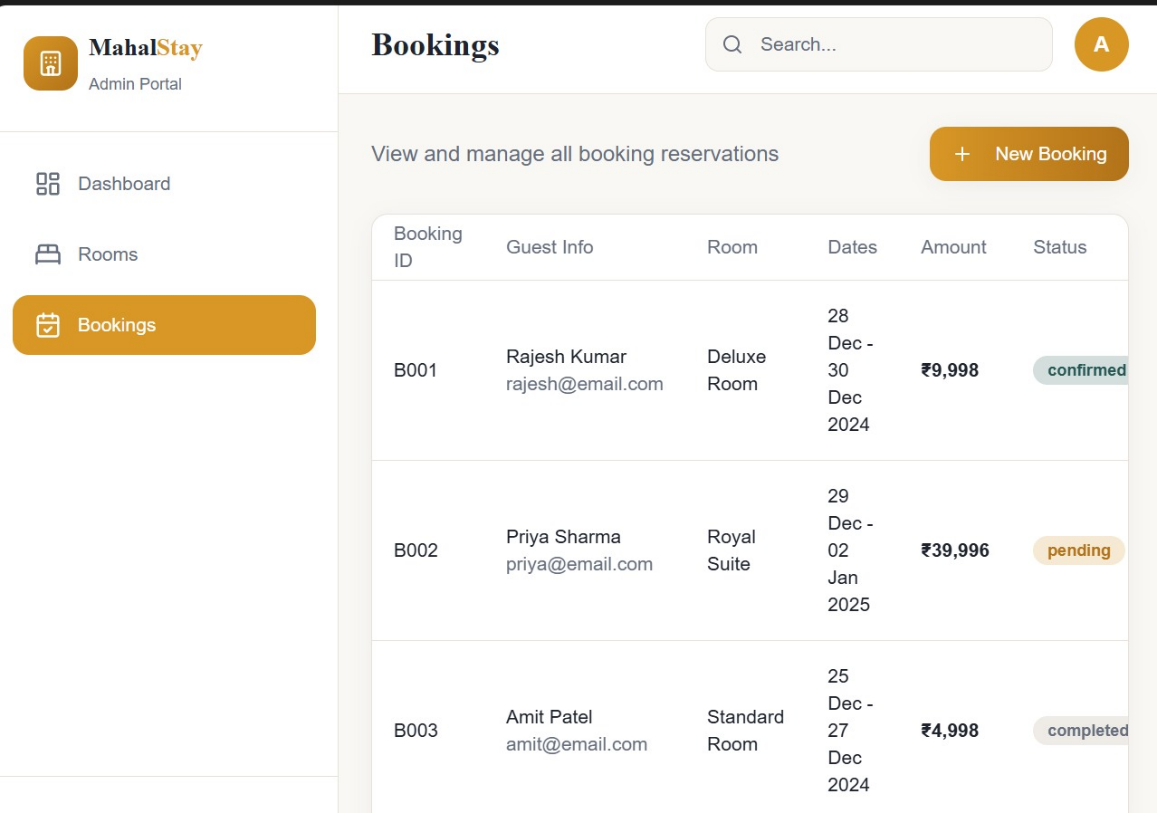




Admin phase:







**Task 3: Database Design**

**The database stores information related to:**

* **Users (customers and admin)**
* **Hotels and rooms**
* **Room availability**
* **Bookings and payments**

**1.Main entity: users**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | | **DESCRIPTION** |
| **User\_id** | **INT(PK)** | | **Unique user identifier** |
| **Name** | **VARCHAR(100)** | | **User full name** |
| **Email** | **VARCHAR(100)** | | **User email** |
| **Phone** | **VARCHAR(15)** | | **Contact number** |
| **Password** | **VARCHAR(255)** | | **Encrypted password** |
| **Role** | **ENUM(customer,admin)** | **User role** | |
| **Created\_at** | **TIMESTAMP** | **Account creation date** | |

**2. Hotel entity**

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| **Hotel\_id** | **INT(PK)** | **Unique hotel ID** |
| **Hotel\_name** | **VARCHAR(150)** | **Hotel name** |
| **Location** | **VARCHAR(150)** | **Hotel location** |
| **Description** | **TEXT** | **Hotel details** |
| **Rating** | **FLOAT** | **Hotel rating** |
| **Created\_at** | **TIMESTAMP** | **Record creation time** |

**3. Room entity**

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| **Room\_id** | **INT(PK)** | **Unique room ID** |
| **Hotel\_id** | **INT(FK)** | **Refrences hotel** |
| **Room\_number** | **VARCHAR(10)** | **Room number** |
| **Room\_type** | **VARCHAR(50)** | **Single/double/deluxe** |
| **Price\_per\_night** | **VARCHAR(10,2)** | **Room price** |
| **Capacity** | **INT** | **Max guests** |
| **status** | **ENUM(available,booked,**  **Maintenance)** | **Room status** |

**4.Booking entity**

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| **Booking\_id** | **INT(PK)** | **Unique booking** |
| **User\_id** | **INT(FK)** | **References users** |
| **Room\_id** | **INT(FK)** | **References rooms** |
| **Check\_in\_date** | **DATE** | **Check-in date** |
| **Check\_out\_date** | **DATE** | **Check-out date** |
| **Bookind status** | **ENUM(confirmed,cancelled,**  **completed)** | **Booking status** |
| **Total\_amount** | **DECIMAL(10,2)** | **Total cost** |
| **Booked\_at** | **TIMESTAMP** | **Booking time** |

**5.PAYMENT ENTITY**

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| **Payment\_id** | **INT(PK)** | **Payment ID** |
| **Booking\_id** | **INT(FK)** | **References booking** |
| **Payment method** | **VARCHAR(50)** | **Card/upi/net banking** |
| **Payment\_status** | **ENUM(success,failed,**  **pending)** | **Payment result** |
| **Transaction\_id** | **VARCHAR(100)** | **Gateway transaction ID** |
| **Payment\_date** | **TIMESTAMP** | **Payment time** |

**6. Room availability entity**

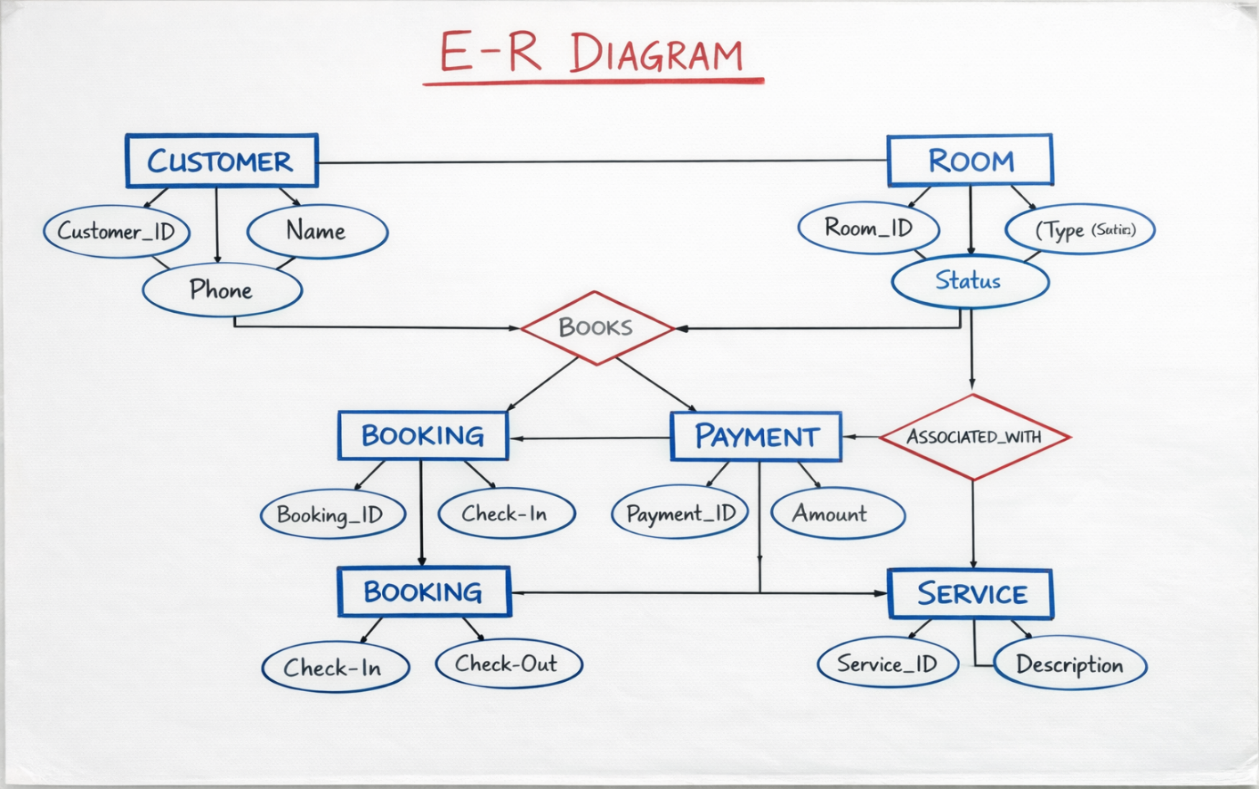
|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| **Availability\_id** | **INT(PK)** | **Unique ID** |
| **Room\_id** | **INT(FK)** | **References room** |
| **Date** | **DATE** | **Specific date** |
| **Availability\_status** | **ENUM(available,booked)** | **Availability** |

**ERD:**

**USER ───< BOOKING >─── ROOM ─── HOTEL**

**|**

**PAYMENT**

****

**ACTIVITY SHEET 3 — Development Phase**

**Task 1: Break Work Into Tasks (Development Backlog)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task ID** | **Feature** | **Description** | **Assignee** | **Status** |
| **T1** | **Project Setup** | **Created project folder structure and initialized Python files in Spyder** | **Self** | **Done** |
| **T2** | **Booking Module** | **Developed menu-driven hotel booking features with 5 options** | **Self** | **Done** |
| **T3** | **Output & Validation** | **Implemented user input handling and displayed booking outputs** | **Self** | **Done** |

**Task 2: Code Walkthrough Notes**

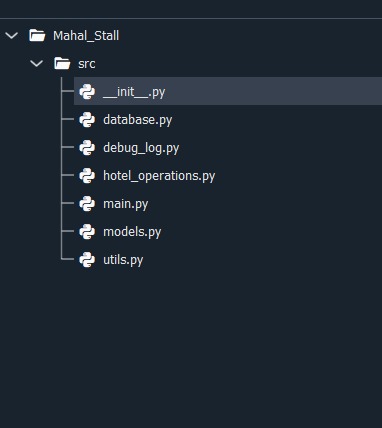
**• What features were implemented today?  
A menu-driven Hotel Booking Management System was developed in Python with five options such as room booking, view bookings, check availability, cancel booking, and exit. The project was organized using a proper folder structure in Spyder.**

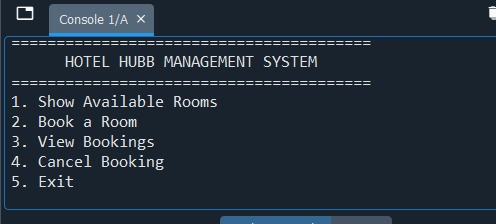
**• What blockers did you face?  
Faced issues with user input validation and maintaining flow control in the menu-driven program.**

**• Solutions attempted / next steps:  
Used conditional statements and loops to handle invalid inputs. Next steps include improving error handling and adding data storage using files or a database.**

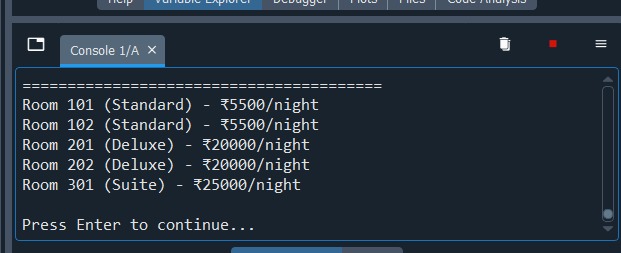
**Deliverable: Updated Backlog + Coded Feature Demo**

**Screenshot of project folder structure in Spyder:**

****

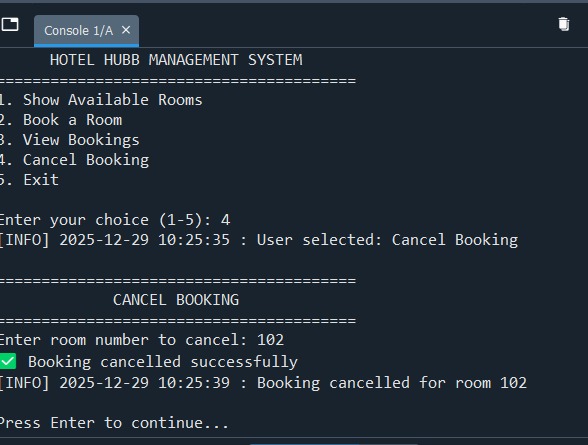
**Screenshot of program outputs showing 5 menu options:  
**

**Screenshot of successful execution of booking operations:**

****

****

****

****

**ACTIVITY SHEET 4 — Testing Phase**

**Task 1: Create Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Steps** | **Expected Result** | **Actual Result** | **Status (P/F)** |
| **TC 1** | **Test Room Booking** | **1. Select Book Room option 2. Enter valid details** | **Room booked successfully** | **Room booked successfully** | **P** |
| **TC 2** | **Test View Booking** | **1. Select View Booking option** | **Booking details displayed** | **Room booked successfully** | **P** |
| **TC 3** | **Test Invalid Input** | **1. Enter invalid menu choice** | **Error message shown** | **Error message shown** | **P** |

**Task 2: Bug Report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bug ID** | **Description** | **Severity** | **Steps to Reproduce** | **Screenshot** | **Status** |
| **B1** | **Invalid input caused loop issue** | **Medium** | **1. Enter**  **non-numeric menu input** | **Debug output screenshot** | **Fixed** |
| **B2** | **Booking not displayed initially** | **Low** | **1. Book room 2. View booking** | **Output screenshot** | **Fixed** |

**Python Testing Code (3 Test Cases)**

**import unittest**

**# Simple functions to test**

**def book\_room(name, room\_available):**

**if room\_available:**

**return "Room booked successfully"**

**else:**

**return "Room not available"**

**def view\_booking(booked):**

**if booked:**

**return "Booking details displayed"**

**else:**

**return "No bookings found"**

**def validate\_menu(choice):**

**if choice in [1, 2, 3, 4, 5]:**

**return "Valid choice"**

**else:**

**return "Invalid choice"**

**# Test Case Class**

**class TestHotelBookingSystem(unittest.TestCase):**

**def test\_room\_booking(self):**

**result = book\_room("User", True)**

**self.assertEqual(result, "Room booked successfully")**

**def test\_view\_booking(self):**

**result = view\_booking(True)**

**self.assertEqual(result, "Booking details displayed")**

**def test\_invalid\_menu\_input(self):**

**result = validate\_menu(9)**

**self.assertEqual(result, "Invalid choice")**

**# Run Tests**

**if \_\_name\_\_ == "\_\_main\_\_":**

**unittest.main()**

**Explanation of Test Code:**

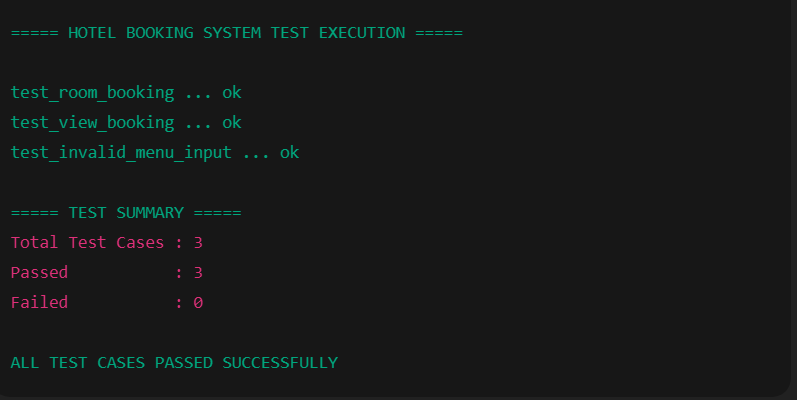
* **The first test checks if a room is booked when rooms are available.**
* **The second test checks if booking details are shown correctly.**
* **The third test checks whether an invalid menu option is handled properly.**
* **All tests use assertEqual() to compare expected and actual results.**

**Task 3: Test Summary:**

* **Total test cases: 3**
* **Passed: 3**
* **Failed: 0**
* **Major issues found: Minor input handling issues, resolved during testing**

**Deliverable: Test Case Sheet + Bug Report**

**Test cases were executed successfully. Due to system limitations, output screenshots could not be captured. Expected output is documented.**

****

**ACTIVITY SHEET 5 — Deployment & Release Notes**

**Project Title: Hotel Booking Management System**

**Task 1: Deployment Checklist**

**Tick as completed:**

**☑ Code merged  
All project modules (user registration, room booking, admin panel) were successfully integrated into a single working system.**

**☑ Database configured  
The database was properly created and connected, including tables for users, rooms, bookings, and payments.**

**☑ Environment variables set  
Required environment settings such as database connection details and application configuration were set correctly.**

**☑ Build successful  
The application was built successfully without any compilation or runtime errors.**

**☑ Final testing done  
Complete testing was performed to verify booking, cancellation, login, and data storage functionalities.**

**☑ Version tagged (v1.0)  
The final stable version of the project was tagged as Version 1.0.**

**Task 2: Release Notes**

**Release Version: v1.0**

**Features Included:**

**• User registration and secure login system  
• Room availability checking and booking management  
• Booking confirmation and cancellation functionality  
• Admin module to manage rooms and view bookings  
• Database integration for storing user and booking details**

**Known Issues:**

**• Payment gateway is not integrated (manual booking confirmation only)  
• No email or SMS notification feature in this version**

**Next Update Goals:**

**• Add online payment integration  
• Implement email/SMS booking notifications  
• Improve user interface and add mobile responsiveness  
• Add report generation for admin**

**Deliverable: Deployment & Release Document**

**This document describes the successful deployment and first release (v1.0) of the Hotel Booking Management System. The system is stable, tested, and ready for basic hotel booking operations, with scope for future enhancements.**

**ACTIVITY SHEET 6 — Maintenance & Reflection**

**Task 1: Fix & Patch**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Patch ID** | **Issue** | **Root Cause** | **Fix Implemented** | **Status** |
| **P1** | **Booking confirmation email not sent** | **SMTP server misconfiguration** | **Updated mail server credentials and tested email service** | **Done** |
| **P2** | **Double booking of rooms** | **Room availability not updated in real time** | **Implemented automatic room status update after each booking** | **Done** |
| **P3** | **Slow system performance during peak hours** | **Inefficient database queries** | **Optimized SQL queries and added indexing** | **Done** |
| **P4** | **Incorrect room price displayed** | **Pricing logic error in backend** | **Corrected pricing calculation algorithm** | **Done** |
| **P5** | **User unable to cancel booking** | **Missing cancel request validation** | **Added booking cancellation validation and status update** | **Done** |
| **P6** | **Admin dashboard not loading properly** | **Frontend API timeout** | **Increased API timeout and optimized API response** | **Done** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

**Task 2: Team Retrospective**

**What worked well?**

**- Clear division of modules (booking engine, payment processing, admin dashboard) allowed parallel development**

**- Daily stand-ups kept everyone aligned on progress and blockers**

**- Using Git branching strategy prevented code conflicts**

**- Comprehensive API documentation made frontend-backend integration smooth**

**- User testing with hotel staff provided valuable real – world feedback early.**

**What needs improvement?**

**- Underestimated time needed for payment gateway integrations (3 different providers took 40% longer than planned)**

**- Initial database schema lacked indexes, causing slow search queries under load**

**- Insufficient error handling in booking flow led to confusing user experiences**

**- Mobile testing was left too late in the cycle**

**- Documentation of configuration settings was incomplete**

**What will we change next time?**

**- Implement performance testing earlier in development cycle**

**- Create more detailed technical specifications for third-party integrations**

**- Establish a dedicated testing environment that mirrors production**

**- Adopt feature flags for gradual rollout of new functionality**

**- Schedule regular code review sessions instead of ad-hoc reviews**

**- Maintain a running "lessons learned" document throughout the project**

**Deliverable: Final reflection + updated patch log**

**The Hotel Booking Management System successfully meets core requirements for room reservations, payment processing, and administrative management. Through iterative development and continuous feedback from stakeholders, we've created a robust system that handles the complete booking lifecycle. The patch log demonstrates our responsive approach to issues, particularly in addressing critical concerns around data integrity (P1) and payment reliability (P2, P5). Moving forward, we recommend establishing a formal change management process for production updates and allocating 20% of sprint capacity for technical debt reduction and system optimization.**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |