Final Report - Hotel Booking System

Group Name: TISM Tech
Ayaan Khan - i220832
Ayaan Mughal - i220861
Mishal Ali - i221291

---

1. Project Introduction

The **Hotel Booking Microservices System** is a web-based platform designed to simplify hotel reservations for guests, hotel management, and travel agents. Developed using the **MERN stack** and **Docker**, it enables scalable hotel room booking, secure payments, customer feedback collection, and administrative reporting.
The project follows the **Scrum methodology** with iterative development and sprint-based delivery.

---

2. Functional and Non-Functional Requirements

Functional Requirements:

The system will support the following functionalities:

- **Search & Filter:**
    - Locate hotels by location, date, and preferences (e.g., amenities, price, rating).
- **Reservation Engine:**
    - Enable secure room bookings with real-time availability and pricing.
    - Support selection of additional services (room services, pick and drop).
- **Booking Management:**
    - Allow guests to view, modify, or cancel reservations.
    - Send notifications on booking changes.
- **Payment Integration:**
    - Process payments securely via multiple payment options.
    - Notify the Hotel Management and Guests on payment confirmation
- **User Accounts:**

    - Manage personal profiles, view booking history, and participate in loyalty programs for guests.
    - Display the success rate and booking history of the travel agents.
- **Administer Rooms/Hotels:**
    - Provide dashboards for hotel staff to update room availability, adjust prices, manage inventory, and generate detailed reports.
- **Feedback System:**
    - Enable guests to submit reviews and ratings (with optional attachments– pictures, videos).

Non-Functional Requirements:

Product Requirements

- **Performance:**
    - Pages load within 2.5 seconds under normal conditions.
- **Reliability:**
    - System uptime of at least 99.5% monthly.
- **Scalability:**
    - Support a growing user base without performance degradation.
- **Security:**
    - Adhere to industry standards (e.g. GDPR) for data and payment processing.
- **Usability:**
    - Provide an intuitive, accessible interface on both desktop and mobile devices, and across multiple platforms– apple, android.
- **Maintainability:**
    - Code must be modular, well-documented, and easy to update.

Organizational Requirements

- **Development Methodology:**
    - Use Scrum Method for iterative delivery and continuous stakeholder feedback.
- **Collaboration Tools:**
    - Utilize Trello for sprint tracking and GitHub for version control.
- **Team Roles:**
    - Clearly defined roles such as **Team Lead** (Ayaan Khan), **UI Designe**r (Ayaan Mughal), Requirement Analyst/Architect and **Developer** (Mishal Ali).

External Requirements

- **Legal and Regulatory:**

    - Comply with GDPR and other relevant data privacy regulations.
- **Payment Processing:**
    - Integrate with trusted payment gateways (e.g., Stripe, PayPal, EasyPaisa) that comply with PCI-DSS.
- **Interoperability:**
    - Allow future integrations with external systems (e.g., travel agent systems).

---

3. User Stories

The project is composed of 16 user stories, with each team member implementing approximately 5. Here is the complete list below:

- **US01: Searching and Filtering Hotels**
    - **Role:** All Users
    - **Goal:** To search for hotels using criteria such as location, dates, and filters (e.g., price range, amenities).
    - **Reason:** To quickly and easily find the best accommodation.
    - **Pre-Conditions:**
        - User is on the home page with access to the search interface.
        - Valid search criteria are available (location, dates, etc.).
    - **Post-Conditions:**
        - A list of hotels matching the criteria is displayed.
        -
- **US02: Booking a Hotel Room**
    - **Role:** Guest
    - **Goal:** A guest books a room by selecting a hotel, choosing a room, entering booking details, and completing payment.
    - **Reason:** Confirm desired accommodation with personalized services if needed.
    - **Pre-Conditions:**
        - A hotel and room have been selected.
        - The hotel room is available.
        - Payment details are available.
    - **Post-Conditions:**
        - Booking is confirmed and a receipt is generated.
        -
- **US03: Managing Bookings**
    - **Role:** Hotel Management
    - **Goal:** To update, cancel and modify bookings of their respective hotels.
    - **Reason:** To avoid miscommunication or overburden at the hotel.

    - **Pre-Conditions:**
        - Hotel is there in the system.
        - Hotel Management is logged in the system.
        - Existing bookings are available.
    - **Post-Conditions:**
        - Booking changes are saved and a confirmation is sent.
        -
- **US04: Providing Feedback**
    - **Role:** All Users
    - **Goal:** To submit reviews and ratings for a successful booking and stay.
    - **Reason:** Enhance service quality and help future users.
    - **Pre-Conditions:**
        - Guest has completed a stay with a valid booking record, or Travel agent has successfully booked a group and Hotel Management is able to withhold these bookings .
    - **Post-Conditions:**
        - Feedback is recorded and linked to the respective booking.
        - Rating is updated.
        -
- **US05: Facilitating Group Bookings**
    - **Role:** Travel Agent
    - **Goal:** To manage bulk room reservations for a group of guests or its customers.
    - **Reason:** Efficiently arrange group or corporate bookings.
    - **Pre-Conditions:**
        - Travel agent is logged in and has provided group details.
        - Required number of rooms are available.
    - **Post-Conditions:**
        - Bulk booking is confirmed with individual room assignments if required and confirmation is sent to the agent.
        -
- **US06: Selecting Favourites/ Adding to the WishList**
    - **Role:** Guests
    - **Goal:** To select and save hotels or specific rooms for future bookings.
    - **Reason:** This reduces searching and filtering time for regular guests and provides an easier booking methodology.
    - **Pre-Conditions:**
        - Guests are logged into the system.
        - Required Hotel or room is available.
    - **Post-Conditions:**
        - Hotel/ Room is successfully added to the favourites tab.
        - Favourites can be accessed in the future.
        -

- **US07: Enrolling in Loyalty Program**
    - **Role:** Guests
    - **Goal:** Guests enroll in and participate in the loyalty program
    - **Reason:** To earn and redeem points or coupons for future bookings.
    - **Pre-Conditions:**
        - Guests are logged into the system.
        - Guests have at least used the system 4 times to be considered in the program
    - **Post-Conditions:**
        - Guest is successfully part of the Loyalty Program
        - Guests successfully get notified of special discounts and earn points.

- **US08: Register/ Login**
    - **Role:** All Users
    - **Goal:** Users authenticate themselves by logging in or create a new account through signup.
    - **Reason:** To book a room.
    - **Pre-Conditions:**
        - System should be running
        - Valid credentials should be entered.
    - **Post-Conditions:**
        - Users are notified of successful login.
        - Users are notified with a verification pin in case of sign up
        - Users can view the Homepage when successfully logged in.

- **US09: Managing Rooms**
    - **Role:** Admin
    - **Goal:** To add, update, and remove rooms in the system for optimal usage and cost management
    - **Reason:** To optimise room usage and costs
    - **Pre-Conditions:**
        - The Admin must be logged into the system.
        - Necessary permissions to modify room records must be granted.
    - **Post-Conditions:**
        - New rooms are successfully added to the system with full details.
        - Existing rooms are updated with the latest information.
        - Rooms no longer in service are removed from the listings.

- **US10: Viewing Booking Data**
    - **Role:** Admin

- **Goal:** To get detailed information for a selected booking record.
- **Reason:** To view all relevant details of a specific booking, such as guest contact information, payment status, and any special requests.
- **Pre-Conditions:**
  - The admin is logged into the system.
  - The booking record to be viewed exists in the system.
- **Post Conditions:**
  - The admin is presented with a detailed view of the selected booking, including all associated metadata and transactional history.
  - The admin has the option to print, export, or update the booking details if required.

- **US11: Manage Hotels**
  - **Role:** Admin
  - **Goal:** To add, update, or remove hotel listings within the platform.
  - **Reason:** Ensure hotel information is current, accurate, and reflective of available inventory.
  - **Pre-Conditions:**
    - The admin is logged into the system.
    - The admin possesses the necessary permissions to modify hotel records.
  - **Post-Conditions:**
    - Hotel listings are updated in the database and reflected across the system.
    - Changes such as new entries, modifications, or deletions are logged and accessible for audit purposes.

- **US12: Manage Accounts**
  - **Role:** Admin
  - **Goal:** To add, update, or remove user accounts.
  - **Reason:** Ensure that account information is accurate, secure, and up-to-date.
  - **Pre-Conditions:**
    - The admin is logged into the system.
    - The admin possesses the necessary permissions to modify user accounts.
  - **Post-Conditions:**
    - The user account information is updated in the database.
    - Changes such as account creation, modification, or deletion are logged for audit purposes.

- **US13: Adjust Prices**
  - **Role:** Hotel Management

    - **Goal:** To modify room pricing based on seasonal trends, promotions, or occupancy rates.
    - **Reason:** Maintain competitive pricing, maximize occupancy, and manage revenue effectively.
    - **Pre-Conditions:**
        - The admin is logged into the system.
        - The current pricing data is accessible and editable by the admin.
    - **Post-Conditions:**
        - Updated pricing information is saved in the system and reflected across all booking channels.
        - A log entry is created for any price changes to maintain a history of adjustments.

- **US14: Generate Reports**
    - **Role:** Hotel Management
    - **Goal:** To generate comprehensive reports on bookings, revenue, occupancy rates, and other key performance metrics.
    - **Reason:** Enable hotel management to analyze operational performance, make data-driven decisions, and strategize improvements.
    - **Pre-Conditions:**
        - The system is connected to the relevant databases.
        - Hotel management is logged into the system.
        - Sufficient historical data exists to generate the required reports.
    - **Post-Conditions:**
        - Detailed reports are generated in the desired format and made available to hotel management.
        - Reports can be filtered or customized based on the selected criteria, providing actionable insights.

- **US15: Process Payment**
    - **Role:** Hotel Management
    - **Goal:** To process and verify payment transactions related to hotel bookings.
    - **Reason:** Ensure that all financial transactions are handled securely and accurately, maintaining the integrity of the hotel's financial records.
    - **Pre-Conditions:**
        - Hotel management is logged into the system.
        - Payment gateway integrations are active and functioning.
    - **Post-Conditions:**
        - Payment transactions are processed and recorded accurately in the system.
        - Successful transactions update booking statuses accordingly, while failed transactions trigger alerts for further review.
        - Detailed logs of all payment activities are maintained for auditing and reconciliation purposes.

- **US16: Express Check-In for Returning Guests**
    - **Role:** Guest
    - **Goal:** To check in online before arriving at the hotel, allowing them to skip the front desk and go straight to their room.
    - **Reason:**Saves time and enhances convenience for frequent travelers.
    - **Pre-Conditions:**
        - The guest has an existing confirmed booking.
        - The system can verify the booking and confirm identity via OTP (One-Time Password) sent via email/SMS.
        - If payment isn't fully settled, the guest must complete it before check-in
    - **Post-Conditions:**
        - The guest receives room access details (such as a digital key code or pickup instructions).
        - The hotel staff is notified of the guest's online check-in.
        - The guest can bypass the front desk and access their room directly.

---

4. Product Backlog

High Priority

- **US11: Manage Hotels:** As an Admin, I want to add and manage hotels within the system so that hotel information is accurate and reflective of available inventory

    - **US11.1:** Provide an intuitive interface for admins to add new hotels with necessary details like name, location, contact information, and amenities.
    - **US11.2:** Enable updating and editing of existing hotel information to ensure accuracy.
    - **US11.3:** Allow removal of hotel listings to maintain an up-to-date inventory.

- **US09: Managing Rooms:** As an Admin, I want to add and manage rooms so that I can optimise usage and costs
    - **US09.1:** Allow detailed entry of new room information including type, price, capacity, and special features.
    - **US09.2:** Facilitate easy updates to existing room details to keep information current.
    - **US09.3:** Provide functionality to remove rooms that are no longer available or in service.

- **US12: Manage Accounts:** As an Admin, I want to add, update, or remove user accounts so that I can ensure account information is accurate, secure, and up-to-date.
    - **US12.1:** Allow the creation of new user accounts so that I can grant system access to authorized users.
    - **US12.2:** Enable updating of existing user details so that I can keep account information current and accurate.
    - **US12.3:** Provide functionality to remove user accounts so that I can maintain security and restrict access when necessary.
    - **US12.4:** Ensure that all account modifications are logged so that I can track changes for auditing purposes.

- **US02: Booking a Hotel Room:** As a Guest, I want to securely book a hotel room in real-time so that I have my accommodation confirmed to my liking

    - **US02.1:** Display comprehensive room details, including images, descriptions, pricing, and available dates.
    - **US02.2:** Offer guests the ability to select additional services such as breakfast, airport transportation, or special accommodations.
    - **US02.3:** Issue immediate booking confirmations and receipts upon successful payment transactions to assure guests of their reservations.

- **US08: Register/Login:** As a User, I want to authenticate myself by logging in or creating a new account through signup so that I can book a room:
    - **US08.1:** Provide a registration option for new users so that I can create an account to access hotel booking services.
    - **US08.2:** Enable secure login with valid credentials so that I can access my account safely.
    - **US08.3:** Implement verification via a PIN for new signups so that I can ensure account security.
    - **US08.4:** Notify users of successful login so that I can confirm access to my account.
    - **US08.5:** Redirect users to the homepage upon successful authentication so that I can start browsing and booking rooms immediately.

Medium Priority

- **US01: Searching and Filtering Hotels:** As a Guest, I want to search and filter hotels so that I can quickly find suitable accommodation.

    - **US01.1:** Allow guests to search hotels by entering location, check-in/check-out dates, and specific preferences (room type, budget, etc.).
    - **US01.2:** Enable filtering of search results based on amenities (Wi-Fi, breakfast, swimming pool, etc.) to enhance guest convenience.

    - **US01.3:** Provide sorting options by price, user ratings, or proximity to major landmarks or airports to streamline decision-making.

- **US13: Adjust Prices:** As Hotel Management, I want to modify room pricing so that I can maintain competitive pricing and manage revenue effectively.
    - **US13.1:** Allow real-time adjustments to room pricing so that I can respond to changing market conditions.
    - **US13.2:** Provide an option to set automatic pricing rules based on predefined conditions so that I can streamline pricing updates.
    - **US13.3:** Ensure price changes are logged and documented so that I can maintain a history of adjustments for auditing and analysis.

- **US03: Managing Bookings:** As a Guest, I want to manage my bookings so that I can adjust my travel plans easily and avoid miscommunication with the Hotel

    - **US03.1:** Provide guests with easy access to view their booking history and upcoming reservation details.
    - **US03.2:** Implement automated notifications via email or SMS to inform guests promptly of any booking changes, modifications, or cancellations.

- **US010: Viewing Booking Data:** As Hotel Staff, I want to access detailed booking data so that I can improve customer service and operational management.

    - **US010.1:** Facilitate quick retrieval of booking information based on guest name, booking dates, or room numbers for efficient customer service.
    - **US010.2:** Provide access to detailed guest profiles to personalize guest services and improve customer satisfaction.

- **US15: Process Payment:** As Hotel Management, I want to process and verify payments so that I can ensure all financial transactions are handled securely and accurately.
    - **US15.1:** Integrate with secure payment gateways so that I can process transactions reliably.
    - **US15.2:** Validate and confirm payment details before processing so that I can prevent errors or fraud.
    - **US15.3:** Update booking statuses automatically upon successful payments so that I can ensure accurate records.
    - **US15.4:** Maintain detailed logs of all transactions so that I can support auditing and financial reconciliation.

Low Priority

- **US04: Providing Feedback:** As a Guest, I want to submit feedback about my stay so that service quality can be enhanced

    - **US04.1:** Allow guests to submit detailed reviews and ratings for hotels and rooms post-stay.
    - **US04.2:** Enable uploading of multimedia (photos/videos) with reviews to enhance feedback authenticity and usefulness.
    - **US04.3:** Provide the option for guests to edit or update previously submitted reviews.

- **US014: Generating Reports:** As Hotel Management, I want to generate detailed performance reports so that I can monitor progress

    - **US014.1:** Generate comprehensive occupancy and revenue reports to help management analyze performance.
    - **US014.2:** Allow reports to be exported in widely used formats such as Excel and PDF for better accessibility and review.

- **US05: Facilitating Group Bookings:** As a Travel Agent, I want to manage bulk room reservations so that I can efficiently arrange group bookings

    - **US05.1:** Enable travel agents to manage bulk reservations efficiently.
    - **US05.2:** Allow easy assignment and management of individual rooms within group reservations.
    - **US05.3:** Implement a flexible system for applying and managing group discounts.

- **US06: Selecting Favourites/ Adding to the WishList:** As a Guest, I want to select and save hotels or specific rooms for future bookings so that I can reduce searching and filtering time
    - **US06.1:** Allow guests to add hotels or rooms to their favourites so that I can quickly access preferred options for future bookings.
    - **US06.2:** Enable easy removal of hotels or rooms from the favourites list so that I can manage my preferences efficiently.
    - **US06.3:** Ensure favourites are accessible in a dedicated tab so that I can conveniently review my saved options anytime.

- **US07: Enrolling in Loyalty Program:** As a Guest, I want to enroll in and participate in the loyalty program so that I can earn and redeem points or coupons for future bookings.
    - **US07.1:** Provide an option for eligible guests to join the loyalty program so that I can start accumulating rewards.
    - **US07.2:** Notify guests about their loyalty status and available rewards so that I can stay informed about my benefits.
    - **US07.3:** Allow guests to redeem earned points or coupons during bookings so that I can take advantage of my loyalty benefits.

- **US08: Express Check-In for Returning Guests:** As a Returning Guest, I want to check in online before arriving at the hotel so that I can skip the front desk and go straight to my room.
    - **US08.1:** Allow returning guests with a confirmed booking to check in online via the hotel's website or mobile app.
    - **US08.2:** Verify the guest's booking and identity through an OTP sent via email/SMS..
    - **US08.3:** If payment is incomplete, prompt the guest to complete the payment before proceeding with check-in.
    - **US08.4:** Provide room access details (digital key code or pickup instructions) after successful check-in.

Structured Specifications

US01: Searching and Filtering Hotels

US01.1: Searching Hotels

Pre-condition: User inputs valid search criteria.

Post-condition: Hotels matching criteria are displayed.

Normal Flow: Enter location, dates → Search → Display results.

Alternative Flow: Notify if no results are found.

US01.2: Filtering Hotels by Amenities

Pre-condition: Hotels are listed from the initial search.

Post-condition: Filtered hotel results are displayed.

Normal Flow: Select amenities → Apply filter → Display updated results.

Alternative Flow: Alert if no matching amenities are found.

US01.3: Sorting Hotels

Pre-condition: Hotels are listed.

Post-condition: Hotels are sorted as per user choice.

Normal Flow: Choose sorting criteria → Display sorted results.
Alternative Flow: Default sorting if criteria are unspecified.

US02: Booking a Hotel Room

US02.1: Display Room Details
Pre-condition: User selects a hotel.
Post-condition: Comprehensive room details are displayed.
Normal Flow: Select hotel → View room details → Display images, descriptions, and pricing.
Alternative Flow: Notify if no rooms are available.

US02.2: Selecting Additional Services
Pre-condition: User is in the booking process.
Post-condition: Selected additional services are added to the booking.
Normal Flow: Choose additional services → Confirm selection → Update booking summary.
Alternative Flow: Notify if service is unavailable.

US02.3: Booking Confirmation
Pre-condition: Payment is successfully processed.
Post-condition: Booking confirmation and receipt are generated.
Normal Flow: Process payment → Generate confirmation → Send receipt via email/SMS.
Alternative Flow: Notify if payment fails.

US03: Managing Bookings

US03.1: Viewing Booking History
Pre-condition: User is logged in.
Post-condition: Booking history is displayed.
Normal Flow: Navigate to bookings → View past and upcoming reservations.
Alternative Flow: Notify if no booking history is available.

US03.2: Automated Notifications
Pre-condition: A booking is modified.
Post-condition: User receives a notification.
Normal Flow: Update booking → Send notification via email/SMS.
Alternative Flow: Notify if the message fails to send.

US04: Providing Feedback

US04.1: Submitting Reviews
Pre-condition: User has completed their stay.
Post-condition: Review is submitted successfully.
Normal Flow: Select stay → Enter review → Submit.
Alternative Flow: Notify if submission fails.

US04.2: Uploading Multimedia
Pre-condition: User is submitting a review.
Post-condition: Photos/videos are attached to the review.
Normal Flow: Select media → Upload → Attach to review.
Alternative Flow: Notify if the upload fails.

US04.3: Editing Reviews
Pre-condition: User has submitted a review.
Post-condition: Updated review is saved.
Normal Flow: Locate review → Edit content → Save changes.
Alternative Flow: Notify if editing is restricted.

US05: Facilitating Group Bookings

US05.1: Bulk Reservations
Pre-condition: Travel agent is logged in.
Post-condition: Group reservation is successfully created.
Normal Flow: Enter booking details → Assign rooms → Confirm booking.
Alternative Flow: Notify if room allocation is unavailable.

US05.2: Managing Assigned Rooms
Pre-condition: Group booking exists.
Post-condition: Individual room assignments are updated.
Normal Flow: Select booking → Update room assignment → Save changes.
Alternative Flow: Notify if rooms are unavailable.

US05.3: Managing Group Discounts
Pre-condition: Bulk reservation is eligible for a discount.

Post-condition: Discount is applied.
Normal Flow: Apply discount → Confirm price adjustment → Save booking.
Alternative Flow: Notify if the discount is invalid.

US06: Selecting Favourites/Adding to WishList

US06.1: Adding Hotels or Rooms to Favourites
Pre-condition: User is logged in.
Post-condition: Hotel/room is saved to favourites.
Normal Flow: Select hotel/room → Click add to favourites → Save.
Alternative Flow: Notify if the action fails.

US06.2: Removing Hotels or Rooms from Favourites
Pre-condition: User has items in favourites.
Post-condition: Item is removed from favourites.
Normal Flow: Open favourites → Select item → Remove.
Alternative Flow: Notify if the removal fails.

US06.3: Accessing Favourites
Pre-condition: User is logged in.
Post-condition: Favourites list is displayed.
Normal Flow: Navigate to favourites → View saved hotels/rooms.
Alternative Flow: Notify if the favourites list is empty.

US07: Enrolling in Loyalty Program

US07.1: Joining the Loyalty Program
Pre-condition: User meets eligibility criteria.
Post-condition: User is enrolled.
Normal Flow: Accept invitation → Enroll in program → Receive confirmation.
Alternative Flow: Notify if eligibility criteria are not met.

US07.2: Receiving Loyalty Status Updates
Pre-condition: User is enrolled in the program.
Post-condition: Loyalty status and rewards are displayed.

Normal Flow: Log in → Navigate to loyalty section → View status.
Alternative Flow: Notify if no loyalty data is available.

US07.3: Redeeming Points or Coupons
Pre-condition: User has available points.
Post-condition: Discount is applied to the booking.
Normal Flow: Select booking → Apply points/coupons → Confirm.
Alternative Flow: Notify if insufficient points.

US08: Register/Login

US08.1: User Registration
Pre-condition: User provides valid registration details.
Post-condition: Account is successfully created.
Normal Flow: Enter registration details → Submit → Receive verification PIN → Verify PIN → Account created.
Alternative Flow: Notify user if registration details are invalid or if the email/phone number is already in use.

US08.2: Secure Login
Pre-condition: User has a registered account.
Post-condition: User is successfully logged in.
Normal Flow: Enter login credentials → Authenticate → Redirect to homepage.
Alternative Flow: Notify user if login credentials are incorrect.

US08.3: PIN Verification for Signup
Pre-condition: User has successfully submitted registration details.
Post-condition: Account is verified and activated.
Normal Flow: System sends verification PIN → User enters PIN → Account activated.
Alternative Flow: Notify user if PIN is incorrect or expired.

US08.4: Successful Login Notification
Pre-condition: User has entered correct credentials.
Post-condition: User receives confirmation of login.
Normal Flow: Authenticate user → Notify successful login → Redirect to homepage.
Alternative Flow: Alert if login is from a new device and require additional verification.

US08.5: Homepage Redirection
Pre-condition: User is successfully authenticated.
Post-condition: Homepage is displayed.
Normal Flow: Authenticate → Redirect to homepage.
Alternative Flow: Redirect user to an onboarding page if first-time login.

US09: Managing Rooms

US09.1: Adding New Rooms
Pre-condition: Admin has the necessary permissions.
Post-condition: Room is successfully added to the system.
Normal Flow: Enter room details → Save → Confirm addition.
Alternative Flow: Notify if required fields are missing.

US09.2: Updating Room Details
Pre-condition: Room exists in the system.
Post-condition: Updated details are saved.
Normal Flow: Select room → Edit details → Save changes.
Alternative Flow: Notify if update fails due to missing or incorrect data.

US09.3: Removing Rooms
Pre-condition: Admin has necessary permissions.
Post-condition: Room is removed from active listings.
Normal Flow: Select room → Remove → Confirm deletion.
Alternative Flow: Notify if the room cannot be deleted due to active bookings.

US10: Viewing Booking Data

US10.1: Retrieving Booking Information
Pre-condition: Hotel staff is logged in.
Post-condition: Booking details are displayed.
Normal Flow: Search by guest name, booking dates, or room number → Display results.
Alternative Flow: Notify if no matching records are found.

US10.2: Accessing Guest Profiles
Pre-condition: Booking exists for the guest.

Post-condition: Detailed guest profile is displayed.
Normal Flow: Select guest name from booking list → View profile.
Alternative Flow: Notify if the profile is incomplete or missing.

US11: Manage Hotels

US11.1: Adding New Hotels
Pre-condition: Admin has necessary permissions.
Post-condition: Hotel is successfully added to the system.
Normal Flow: Enter hotel details → Save.
Alternative Flow: Notify if details are incomplete or if the hotel already exists.

US11.2: Updating Hotel Details
Pre-condition: Hotel exists in the system.
Post-condition: Updated details are saved.
Normal Flow: Select hotel → Edit details → Save changes.
Alternative Flow: Notify if update fails.

US11.3: Removing Hotels
Pre-condition: Admin has necessary permissions.
Post-condition: Hotel is removed from active listings.
Normal Flow: Select hotel → Remove → Confirm deletion.
Alternative Flow: Notify if hotel has active bookings and cannot be removed.

US12: Manage Accounts

US12.1: Creating User Accounts
Pre-condition: Admin is logged in.
Post-condition: User account is created.
Normal Flow: Enter user details → Assign role → Save.
Alternative Flow: Notify if details are invalid.

US12.2: Updating User Details
Pre-condition: User account exists.
Post-condition: Updated details are saved.
Normal Flow: Select user → Edit details → Save.

Alternative Flow: Notify if the update fails.

US12.3: Removing User Accounts
Pre-condition: Admin has necessary permissions.
Post-condition: Account is removed.
Normal Flow: Select user → Remove account → Confirm deletion.
Alternative Flow: Notify if the user has active reservations or permissions preventing deletion.

US12.4: Logging Account Changes
Pre-condition: Any modification is made to a user account.
Post-condition: Change is recorded in the system.
Normal Flow: Admin modifies an account → System logs change.
Alternative Flow: Notify if logging fails due to a system issue.

US13: Adjust Prices

US13.1: Modifying Room Pricing
Pre-condition: Admin has necessary permissions.
Post-condition: Price changes are reflected in the system.
Normal Flow: Adjust pricing → Confirm changes → Save.
Alternative Flow: Notify if the update fails.

US13.2: Setting Automated Pricing Rules
Pre-condition: Admin has access to pricing management.
Post-condition: Automatic pricing rules are applied.
Normal Flow: Define conditions → Save → System updates prices accordingly.
Alternative Flow: Notify if rules conflict with existing settings.

US13.3: Logging Price Adjustments
Pre-condition: Price changes are made.
Post-condition: Adjustments are recorded in the system.
Normal Flow: Modify price → Save → Log entry created.
Alternative Flow: Notify if logging fails.

US14: Generating Reports

US14.1: Generating Occupancy and Revenue Reports
Pre-condition: Admin is logged in.
Post-condition: Report is generated.
Normal Flow: Select report type → Generate → Display.
Alternative Flow: Notify if report generation fails.

US14.2: Exporting Reports
Pre-condition: Report is generated.
Post-condition: Report is downloaded in the selected format.
Normal Flow: Choose format (PDF/Excel) → Export → Download.
Alternative Flow: Notify if export fails.

US15: Process Payment

US15.1: Processing Transactions
Pre-condition: Secure payment gateway is available.
Post-condition: Payment is successfully processed.
Normal Flow: Enter payment details → Process transaction → Confirm booking.
Alternative Flow: Notify if payment fails.

US15.2: Validating Payment Details
Pre-condition: User enters payment details.
Post-condition: Payment is validated.
Normal Flow: System checks card details → Validate with bank → Confirm.
Alternative Flow: Notify if card is declined.

US15.3: Updating Booking Status
Pre-condition: Payment is successful.
Post-condition: Booking status is updated.
Normal Flow: Confirm payment → Update booking record → Notify user.
Alternative Flow: Notify if status update fails.

US15.4: Maintaining Payment Logs
Pre-condition: A transaction is completed.
Post-condition: Transaction is logged for auditing.
Normal Flow: Process payment → Save transaction details.
Alternative Flow: Notify if logging fails.

US16: Express Check-In for Returning Guests

US16.1: Online Check-In for Returning Guests
Pre-condition: Guest has a confirmed booking.
Post-condition: Guest is checked in online.
Normal Flow: Enter booking details → Verify identity via OTP → Complete check-in.
Alternative Flow: Display error if booking is invalid or OTP verification fails.

US16.2: Identity Verification via OTP
Pre-condition: Guest initiates online check-in.
Post-condition: Guest identity is verified.
Normal Flow: System sends OTP → Guest enters OTP → Verification success.
Alternative Flow: Prompt for OTP resend if incorrect or expired.

US16.3: Payment Completion Before Check-In
Pre-condition: Guest has an outstanding payment.
Post-condition: Payment is successfully processed.
Normal Flow: Display outstanding amount → Guest completes payment → Proceed with check-in.
Alternative Flow: Restrict check-in until payment is completed.

US16.4: Room Access Details Delivery
Pre-condition: Guest has successfully checked in online.
Post-condition: Guest receives room access details.
Normal Flow: Check-in confirmed → System sends digital key or pickup instructions.
Alternative Flow: Notify front desk if digital access fails.

---

5. Sprint 1 and Sprint 2 Backlog

Sprint 1 Backlog: Project Initialization Module

- **US02: Booking a Hotel Room:** As a Guest, I want to securely book a hotel room in real-time so that I have my accommodation confirmed to my liking

    - **US02.1:** Display comprehensive room details, including images, descriptions, pricing, and available dates.

    - **US02.2:** Offer guests the ability to select additional services such as breakfast, airport transportation, or special accommodations.
    - **US02.3:** Issue immediate booking confirmations and receipts upon successful payment transactions to assure guests of their reservations.

- **US08: Register/Login:** As a User, I want to authenticate myself by logging in or creating a new account through signup so that I can book a room:
    - **US08.1:** Provide a registration option for new users so that I can create an account to access hotel booking services.
    - **US08.2:** Enable secure login with valid credentials so that I can access my account safely.
    - **US08.3:** Implement verification via a PIN for new signups so that I can ensure account security.
    - **US08.4:** Notify users of successful login so that I can confirm access to my account.
    - **US08.5:** Redirect users to the homepage upon successful authentication so that I can start browsing and booking rooms immediately.

- **US09: Managing Rooms:** As an Admin, I want to add and manage rooms so that I can optimise usage and costs
    - **US09.1:** Allow detailed entry of new room information including type, price, capacity, and special features.
    - **US09.2:** Facilitate easy updates to existing room details to keep information current.
    - **US09.3:** Provide functionality to remove rooms that are no longer available or in service.

- **US11: Manage Hotels:** As an Admin, I want to add and manage hotels within the system so that hotel information is accurate and reflective of available inventory

    - **US11.1:** Provide an intuitive interface for admins to add new hotels with necessary details like name, location, contact information, and amenities.
    - **US11.2:** Enable updating and editing of existing hotel information to ensure accuracy.
    - **US11.3:** Allow removal of hotel listings to maintain an up-to-date inventory.

- **US12: Manage Accounts:** As an Admin, I want to add, update, or remove user accounts so that I can ensure account information is accurate, secure, and up-to-date.
    - **US12.1:** Allow the creation of new user accounts so that I can grant system access to authorized users.
    - **US12.2:** Enable updating of existing user details so that I can keep account information current and accurate.

- **US12.3:** Provide functionality to remove user accounts so that I can maintain security and restrict access when necessary.
- **US12.4:** Ensure that all account modifications are logged so that I can track changes for auditing purposes.

Sprint 2 Backlog: Feature Enhancement

- **US01: Searching and Filtering Hotels:** As a Guest, I want to search and filter hotels so that I can quickly find suitable accommodation.

    - **US01.1:** Allow guests to search hotels by entering location, check-in/check-out dates, and specific preferences (room type, budget, etc.).
    - **US01.2:** Enable filtering of search results based on amenities (Wi-Fi, breakfast, swimming pool, etc.) to enhance guest convenience.
    - **US01.3:** Provide sorting options by price, user ratings, or proximity to major landmarks or airports to streamline decision-making.

- **US13: Adjust Prices:** As Hotel Management, I want to modify room pricing so that I can maintain competitive pricing and manage revenue effectively.
    - **US13.1:** Allow real-time adjustments to room pricing so that I can respond to changing market conditions.
    - **US13.2:** Provide an option to set automatic pricing rules based on predefined conditions so that I can streamline pricing updates.
    - **US13.3:** Ensure price changes are logged and documented so that I can maintain a history of adjustments for auditing and analysis.

- **US03: Managing Bookings:** As a Guest, I want to manage my bookings so that I can adjust my travel plans easily and avoid miscommunication with the Hotel

    - **US03.1:** Provide guests with easy access to view their booking history and upcoming reservation details.
    - **US03.2:** Implement automated notifications via email or SMS to inform guests promptly of any booking changes, modifications, or cancellations.

- **US010: Viewing Booking Data:** As Hotel Staff, I want to access detailed booking data so that I can improve customer service and operational management.

    - **US010.1:** Facilitate quick retrieval of booking information based on guest name, booking dates, or room numbers for efficient customer service.
    - **US010.2:** Provide access to detailed guest profiles to personalize guest services and improve customer satisfaction.

- **US15: Process Payment:** As Hotel Management, I want to process and verify payments so that I can ensure all financial transactions are handled securely and accurately.
    - **US15.1:** Integrate with secure payment gateways so that I can process transactions reliably.
    - **US15.2:** Validate and confirm payment details before processing so that I can prevent errors or fraud.
    - **US15.3:** Update booking statuses automatically upon successful payments so that I can ensure accurate records.
    - **US15.4:** Maintain detailed logs of all transactions so that I can support auditing and financial reconciliation.

---

6. Project Plan

Introduction / Overview

- **Project Name**: Hotel Booking Microservices System

- **Description**: A microservices-based hotel booking platform supporting guest reservations, hotel management, and travel agent bookings, designed using the MERN stack and containerized with Docker.

- **Need Addressed**: Enables scalable hotel room booking, management of room availability, secure payment simulation, and customer feedback collection.

- **Sponsor**: TISM Tech

- **Deliverables**:

    - Deliverable 1: Planning Documents, Setup, Initial Sprints

    - Deliverable 2: Detailed Documentation + Sprint 1 and Sprint 2 Implementation

    - Deliverable 3: Sprint 3 Implementation + Testing + Deployment

---

Project Organization

- **Ayaan Khan -** Scrum Master, Project Manager, Tester
- **Ayaan Mughal -** UI Designer, Developer
- **Mishal Ali:** Requirement Architect, Developer

Management and Technical Approach

- **Management Objectives**:
    - Deliver a scalable, modular hotel booking solution.
    - Achieve sprint goals timely with continuous integration and testing.

- **Project Control**:
    - Scrum-based sprint planning and monitoring.
    - Trello board for task management.
    - GitHub for version control and branching.

- **Risk Management**:
    - Early integration testing to avoid service mismatches.
    - Regular backups during Dockerization to avoid deployment risks.

- **Technical Processes**:
    - Node.js Microservices with Express
    - MongoDB database per service
    - React.js Web Client
    - Docker and Docker Compose for orchestration

Project Scope

- **Major Work Packages**:

    - Deliverable 1 (Planning and Documentation)

    - Deliverable 2 (Sprint 1 and Sprint 2 Implementation)
    - Deliverable 3 (Sprint 3 Implementation, Testing, Deployment)

- **Key Deliverables**:
    - Project Reports
    - Source Code for all Sprints
    - Final Deployment (Dockerized)

Work Breakdown Structure (WBS)

The Work Breakdown Structure (WBS) for the Hotel Booking Microservices Project organizes the entire scope of work into hierarchical levels, making it easier to manage, track, and complete tasks systematically.

At the top level, the project is divided into three major deliverables corresponding to the academic deliverable phases:

- **Deliverable 1**: Focused on planning activities, setting up documentation, preparing the project management environment (Trello, GitHub), and defining user stories and initial backlog items.

- **Deliverable 2**: Emphasized the development and implementation of Sprint 1 (basic CRUD operations) and Sprint 2 (advanced features like filtering, pricing adjustments, and payment simulation), along with refinement of project documentation.

- **Deliverable 3**: Concentrated on completing **Sprint 3**, conducting thorough black-box and white-box testing, and preparing the final Dockerized deployment of the project.

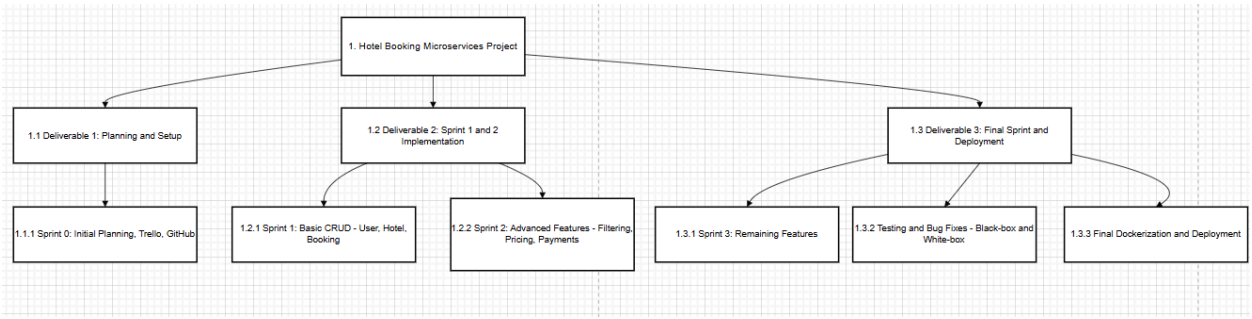
Each deliverable is further broken down into sprints, activities, and tasks at multiple levels (1.1, 1.1.1, etc.).
This structured breakdown ensures that the team can manage progress incrementally while maintaining alignment with project goals and deadlines.

WBS Table

	A	B	C
1	**Level**	**Work Package / Task**	**Details**
2	**1**	Hotel Booking Microservices Project	Entire project lifecycle
3	**1.1**	Deliverable 1: Planning and Setup	Project team info, user stories, initial documents
4	1.1.1	Sprint 0: Initial Planning, Trello, GitHub	Set up project repositories and task management tools
5	**1.2**	Deliverable 2: Sprint 1 and 2 Implementation	Complete SRS, Sprint 1 and Sprint 2 implementation
6	1.2.1	Sprint 1: Basic CRUD (User, Hotel, Booking)	CRUD operations and basic system functionalities
7	1.2.2	Sprint 2: Advanced features (Filtering, Pricing, Payments)	Implement filtering, pricing adjustments, and dummy payments
8	**1.3**	Deliverable 3: Final Sprint and Deployment	Sprint 3, system completion, and deployment
9	1.3.1	Sprint 3: Remaining Features	Complete pending user stories, system refinements
10	1.3.2	Testing and Bug Fixes (Black-box and White-box Testing)	Conduct Black-box and White-box testing
11	1.3.3	Final Dockerization and Deployment	Docker Compose setup for production deployment

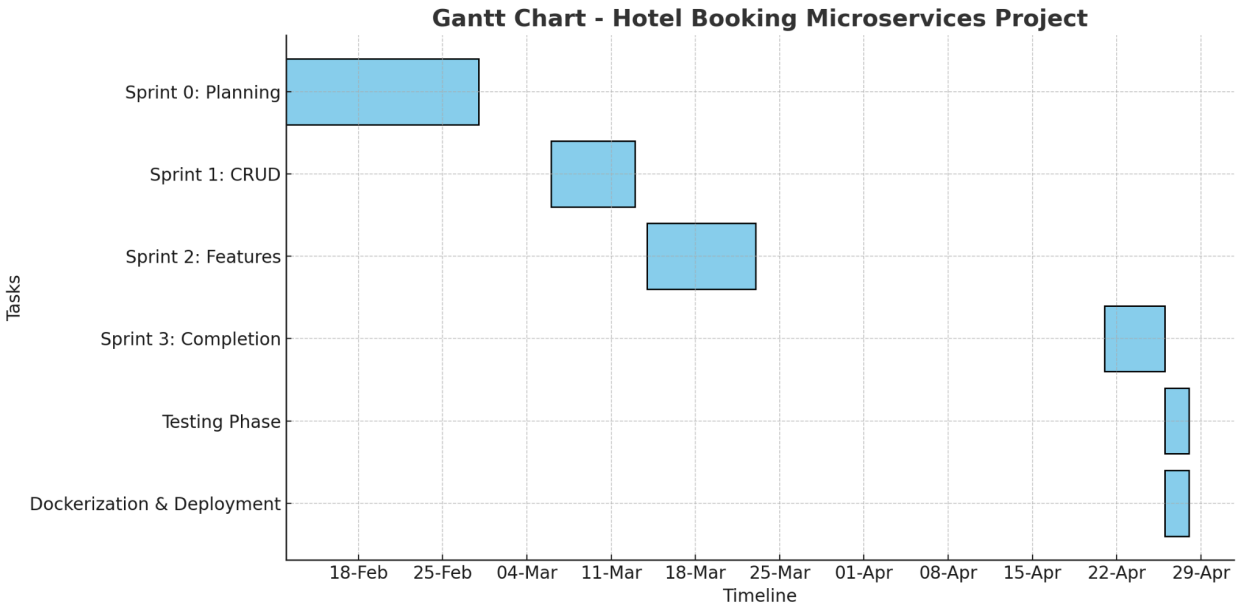
---

WBS Diagram (Structure)



Project Schedule

Gantt Chart Timeline



Textual Summary:

- **Feb 12–28**: Deliverable 1 (Planning, Setup)
- **March 6–23**: Deliverable 2 (Sprints 1 & 2 Coding)
- **April 21–28**: Deliverable 3 (Sprint 3 Coding, Testing, Final Submission)

Budget

Category	**Percentage(%)**
Developer Resources	80
DevOps Tools	10
Testing	10

---

7. Architecture Diagram

The following architectural styles have been adopted in our system:

Microservices Architecture

Each core functionality (User, Hotel, Booking) is implemented as a **separate microservice**, ensuring modularity, scalability, and ease of deployment. Each service owns its data and operates independently.

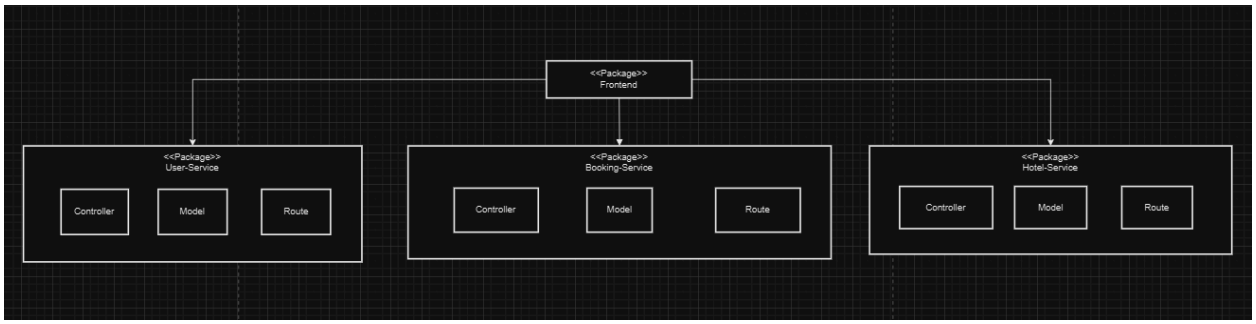
- **User Service**: Handles user authentication, registration, login, and loyalty features.
- **Booking Service**: Manages room bookings, payment simulation, and reservation tracking.
- **Hotel Service**: Manages hotel and room listings, search and filter functionality, pricing updates, and reviews.
- **Frontend**: A React-based client application interfacing with backend services through REST APIs.

Layered (3-tier) Architecture within Services

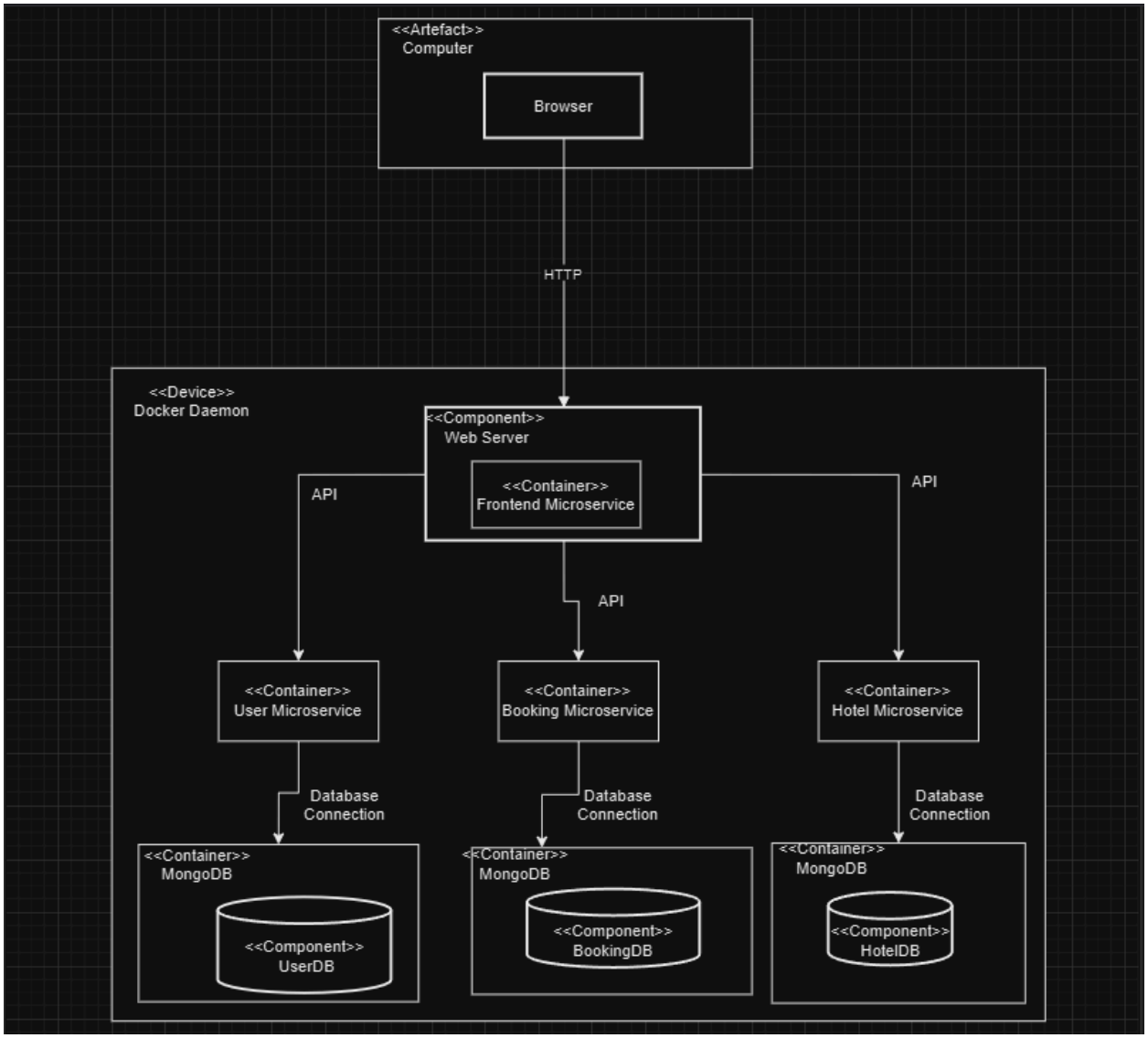
Each service follows a **layered structure**:

- **Controller**: Handles logic and validation
- **Route Layer**: Manages request routing
- **Model Layer**: Defines and interacts with MongoDB collections

Architecture Diagram



Deployment Diagram with Containers and Databases:



---

8. Design (Sprint 3)

Sprint 3 Backlog: Remaining Features

- **US04: Providing Feedback:** As a Guest, I want to submit feedback about my stay so that service quality can be enhanced
    - **US04.1:** Allow guests to submit detailed reviews and ratings for hotels and rooms post-stay.
    - **US04.2:** Enable uploading of multimedia (photos/videos) with reviews to enhance feedback authenticity and usefulness.

- **US04.3:** Provide the option for guests to edit or update previously submitted reviews.

- **US014: Generating Reports:** As Hotel Management, I want to generate detailed performance reports so that I can monitor progress

    - **US014.1:** Generate comprehensive occupancy and revenue reports to help management analyze performance.
    - **US014.2:** Allow reports to be exported in widely used formats such as Excel and PDF for better accessibility and review.

- **US05: Facilitating Group Bookings:** As a Travel Agent, I want to manage bulk room reservations so that I can efficiently arrange group bookings

    - **US05.1:** Enable travel agents to manage bulk reservations efficiently.
    - **US05.2:** Allow easy assignment and management of individual rooms within group reservations.
    - **US05.3:** Implement a flexible system for applying and managing group discounts.

- **US06: Selecting Favourites/ Adding to the WishList:** As a Guest, I want to select and save hotels or specific rooms for future bookings so that I can reduce searching and filtering time
    - **US06.1:** Allow guests to add hotels or rooms to their favourites so that I can quickly access preferred options for future bookings.
    - **US06.2:** Enable easy removal of hotels or rooms from the favourites list so that I can manage my preferences efficiently.
    - **US06.3:** Ensure favourites are accessible in a dedicated tab so that I can conveniently review my saved options anytime.

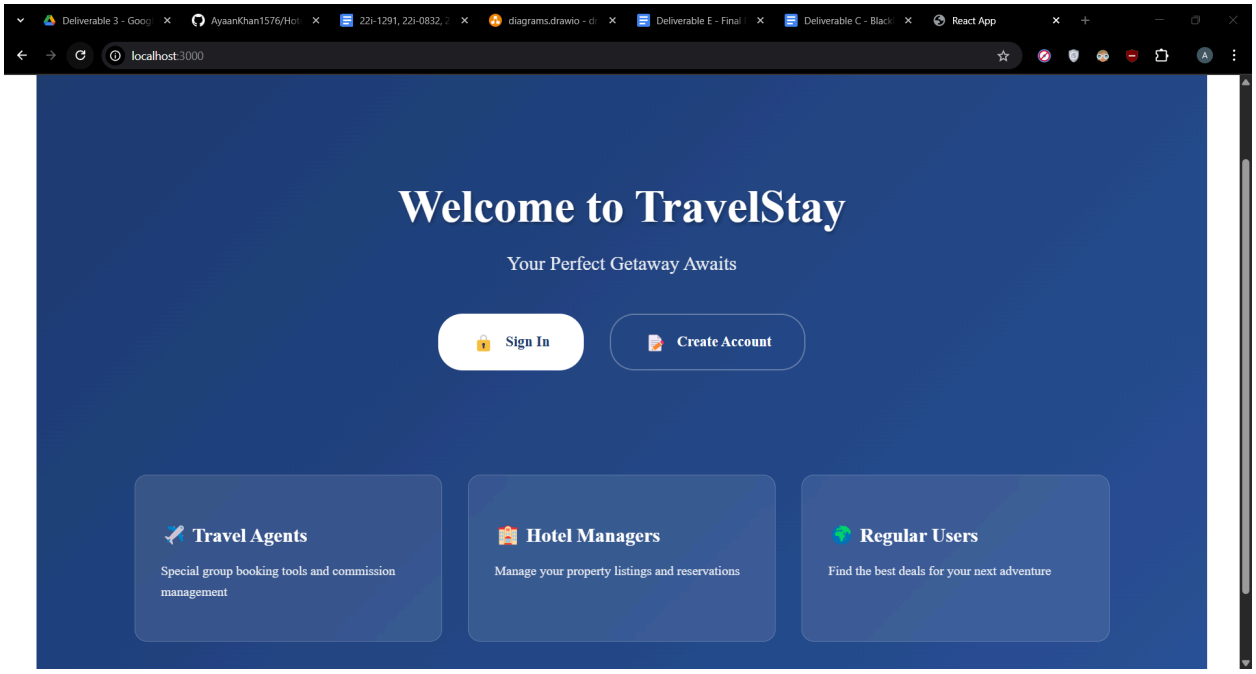
- **US07: Enrolling in Loyalty Program:** As a Guest, I want to enroll in and participate in the loyalty program so that I can earn and redeem points or coupons for future bookings.
    - **US07.1:** Provide an option for eligible guests to join the loyalty program so that I can start accumulating rewards.
    - **US07.2:** Notify guests about their loyalty status and available rewards so that I can stay informed about my benefits.
    - **US07.3:** Allow guests to redeem earned points or coupons during bookings so that I can take advantage of my loyalty benefits.

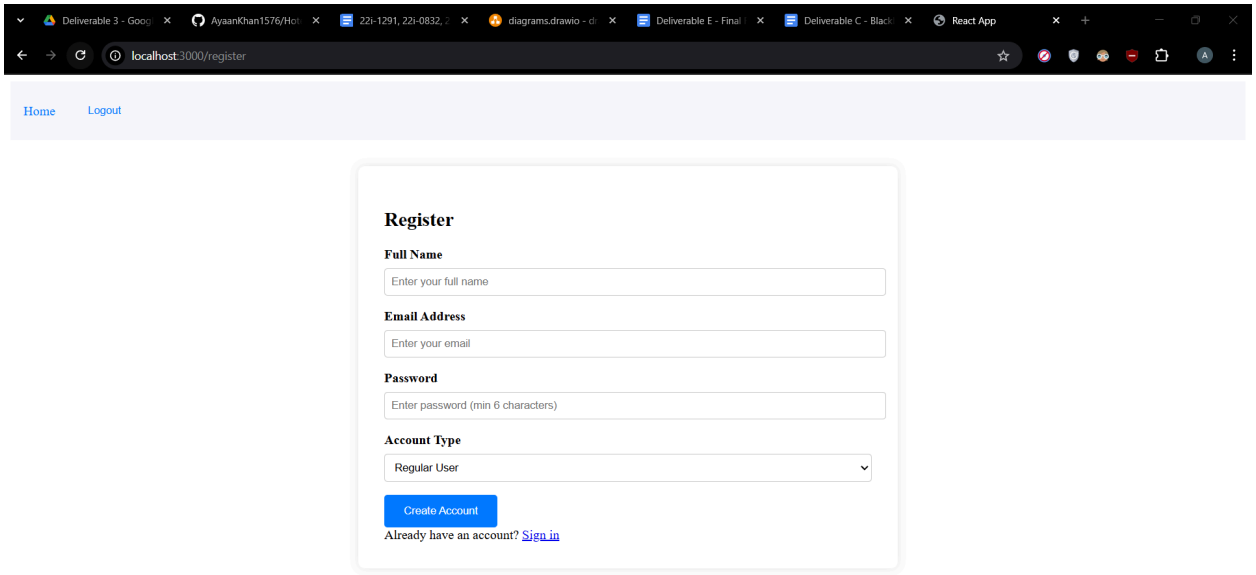
- **US08: Express Check-In for Returning Guests:** As a Returning Guest, I want to check in online before arriving at the hotel so that I can skip the front desk and go straight to my room.
    - **US08.1:** Allow returning guests with a confirmed booking to check in online via the hotel's website or mobile app.
    - **US08.2:** Verify the guest's booking and identity through an OTP sent via email/SMS..
    - **US08.3:** If payment is incomplete, prompt the guest to complete the payment before proceeding with check-in.
    - **US08.4:** Provide room access details (digital key code or pickup instructions) after successful check-in.

---

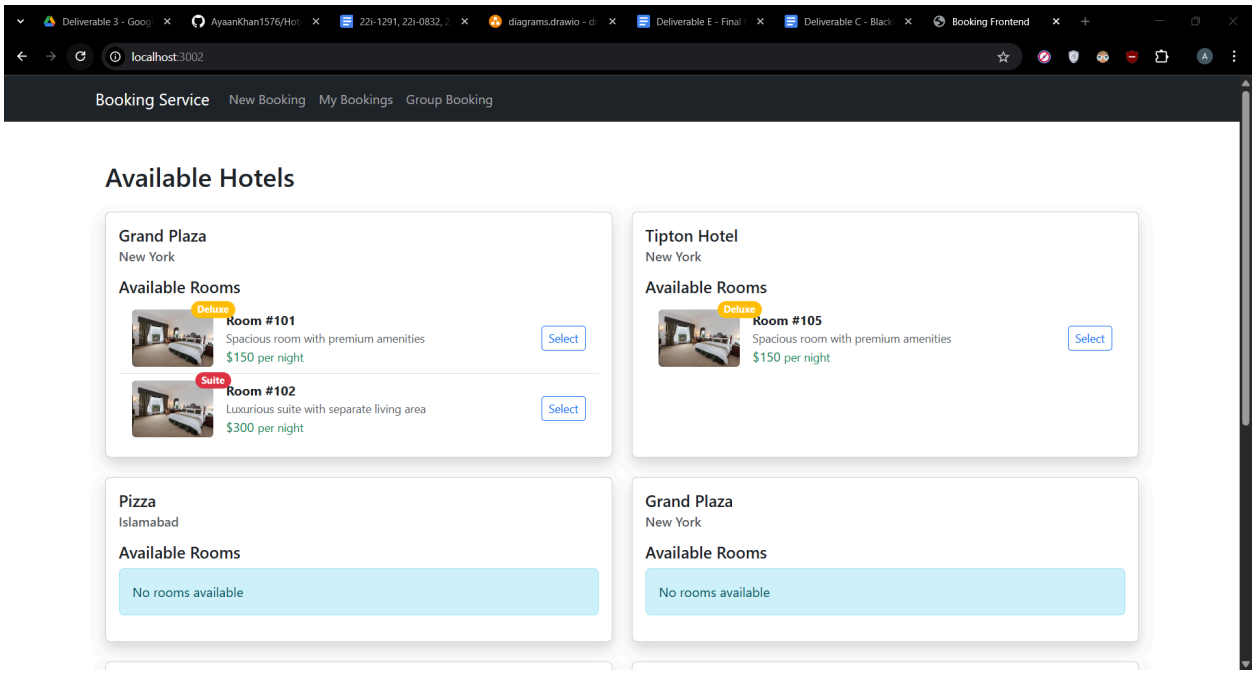
9. Actual Implementation Screenshots

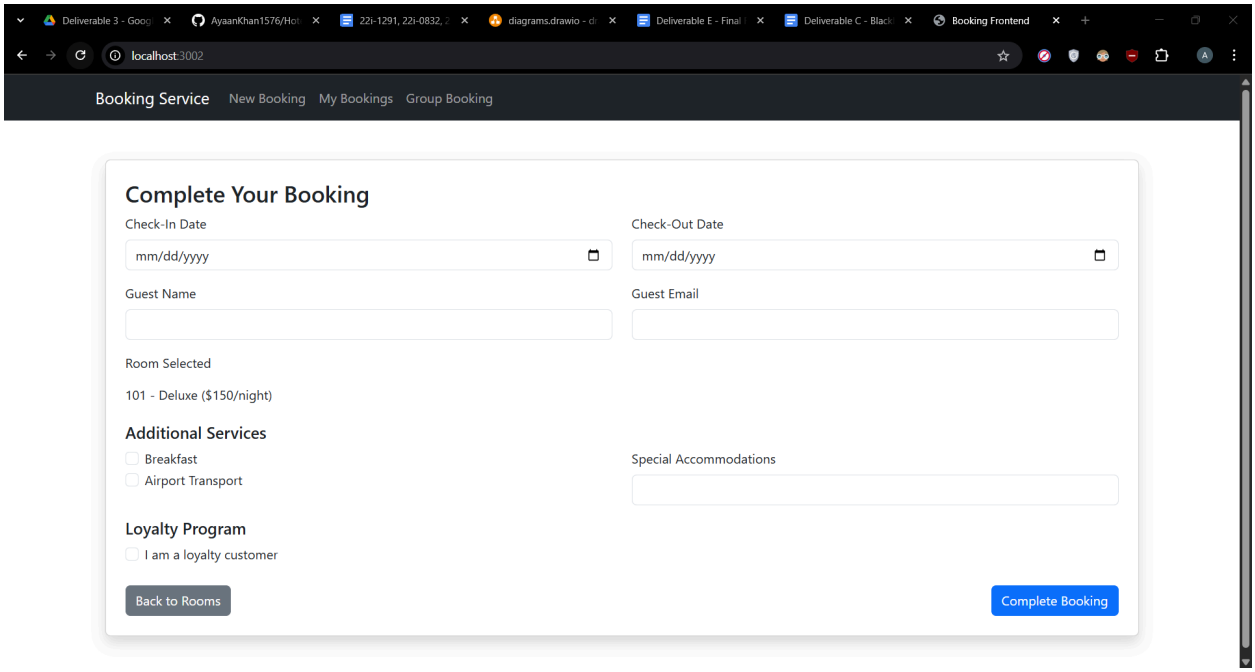
User:

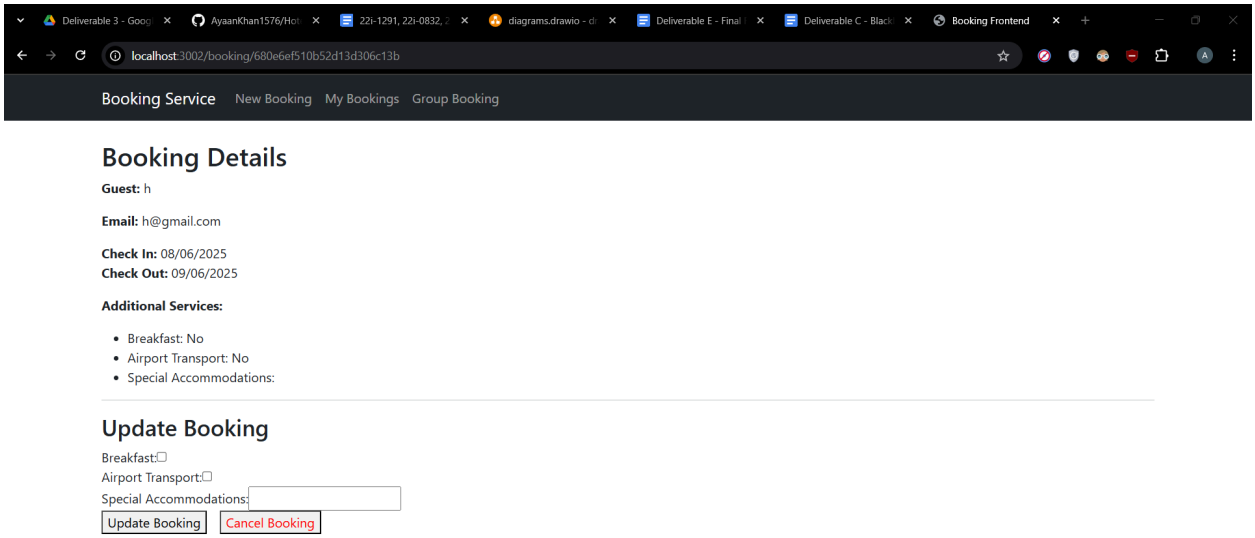


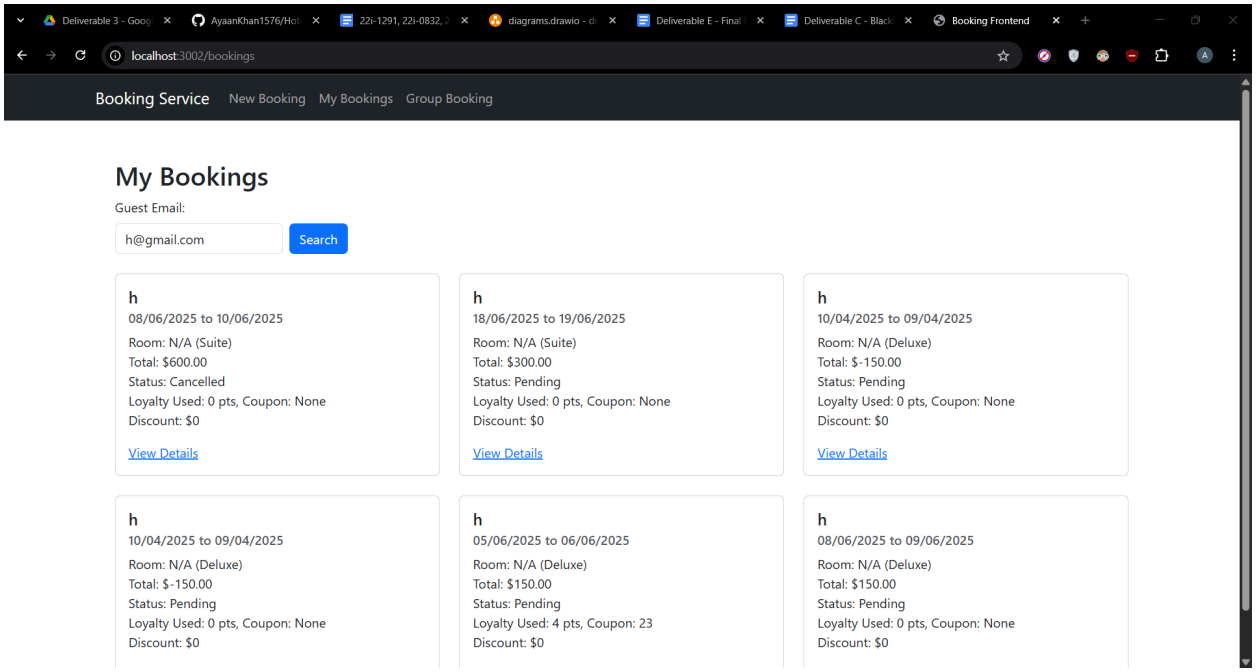

Deliverable 3 - Goog
AyaanKhan1576/Hot
22i-1291, 22i-0832, 2
diagrams.drawio - di
Deliverable E - Final
Deliverable C - Black
React App
localhost:3000/register
Home
Logout
Register
Full Name
Enter your full name
Email Address
Enter your email
Password
Enter password (min 6 characters)
Account Type
Regular User
Create Account
Already have an account? Sign in

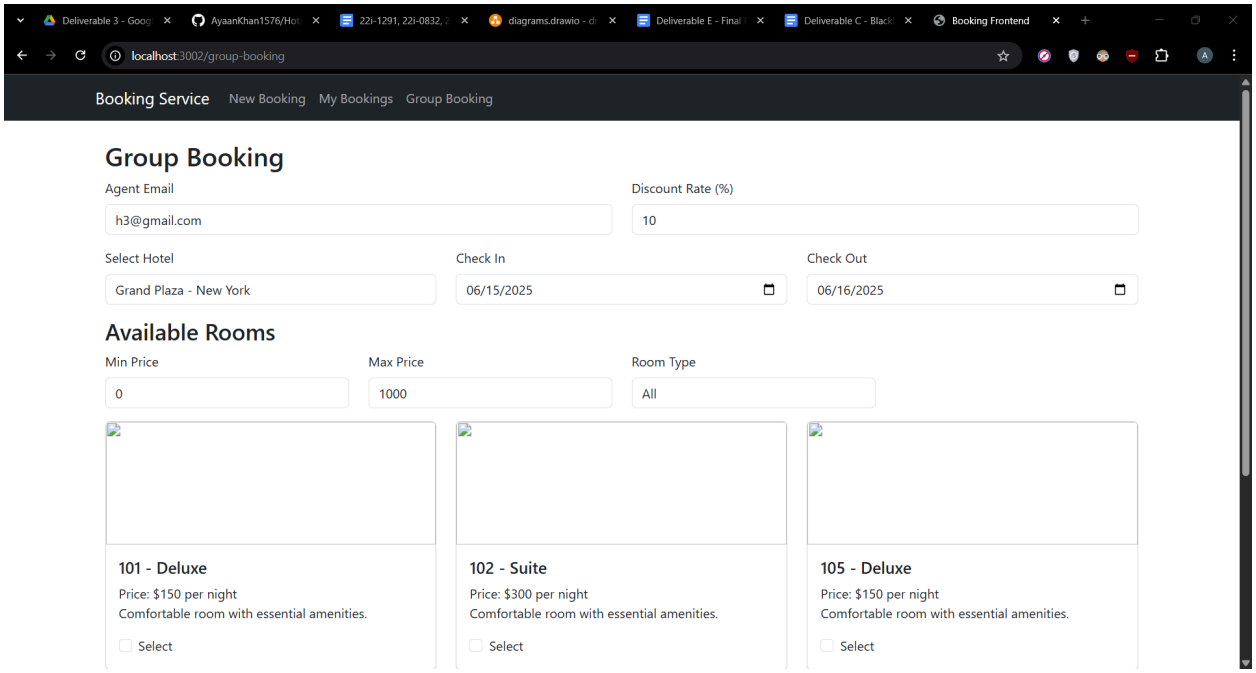
Booking


Deliverable 3 - Goog
AyaanKhan1576/Hot
22i-1291, 22i-0832, 2
diagrams.drawio - di
Deliverable E - Final
Deliverable C - Black
Booking Frontend
localhost:3002
Booking Service
New Booking
My Bookings
Group Booking
Available Hotels
Grand Plaza
New York
Available Rooms
Deluxe
Room #101
Spacious room with premium amenities
$150 per night
Select
Suite
Room #102
Luxurious suite with separate living area
$300 per night
Select
Tipton Hotel
New York
Available Rooms
Deluxe
Room #105
Spacious room with premium amenities
$150 per night
Select
Pizza
Islamabad
Available Rooms
No rooms available
Grand Plaza
New York
Available Rooms
No rooms available

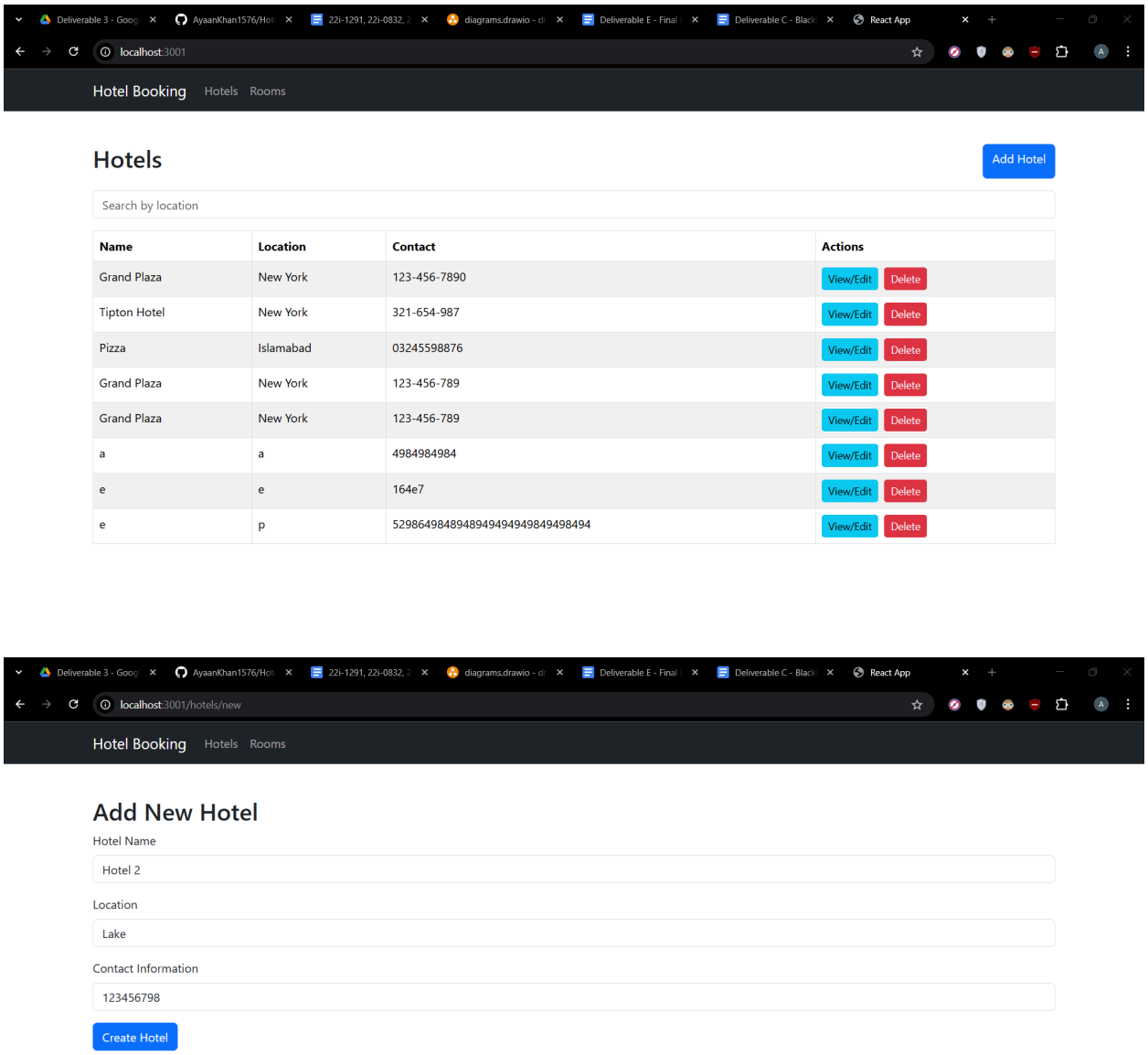

Booking Service
New Booking
My Bookings
Group Booking
Complete Your Booking
Check-In Date
mm/dd/yyyy
Check-Out Date
mm/dd/yyyy
Guest Name
Guest Email
Room Selected
101 - Deluxe ($150/night)
Additional Services
Breakfast
Airport Transport
Special Accommodations
Loyalty Program
I am a loyalty customer
Back to Rooms
Complete Booking


Booking Service
New Booking
My Bookings
Group Booking
Booking Details
Guest: h
Email: h@gmail.com
Check In: 08/06/2025
Check Out: 09/06/2025
Additional Services:
Breakfast: No
Airport Transport: No
Special Accommodations:
Update Booking
Breakfast:
Airport Transport:
Special Accommodations:
Update Booking
Cancel Booking

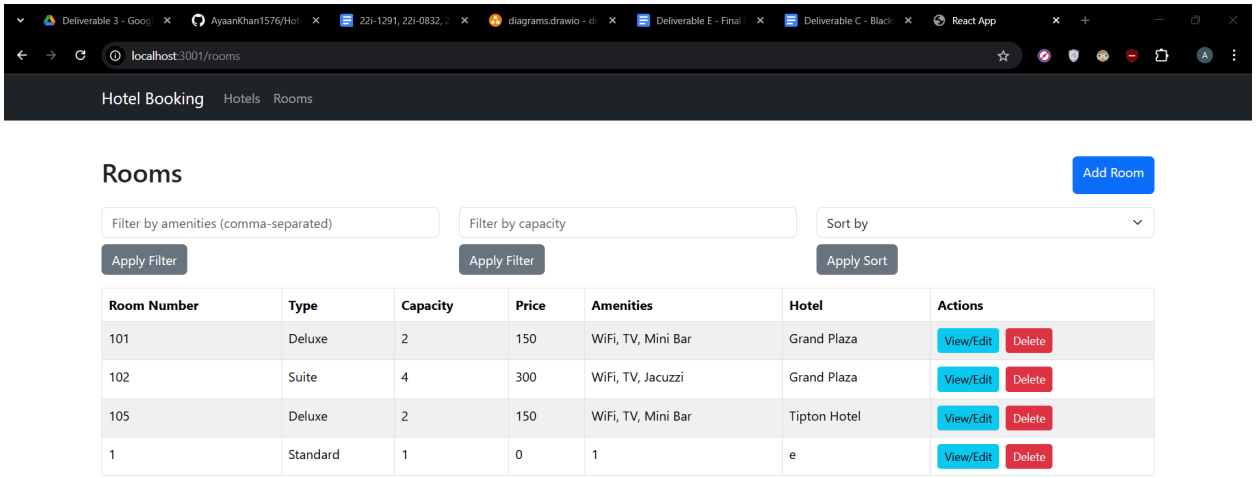

Booking Frontend
localhost:3002/bookings
Booking Service
New Booking
My Bookings
Group Booking
My Bookings
Guest Email:
h@gmail.com
Search
h
08/06/2025 to 10/06/2025
Room: N/A (Suite)
Total: $600.00
Status: Cancelled
Loyalty Used: 0 pts, Coupon: None
Discount: $0
View Details
h
18/06/2025 to 19/06/2025
Room: N/A (Suite)
Total: $300.00
Status: Pending
Loyalty Used: 0 pts, Coupon: None
Discount: $0
View Details
h
10/04/2025 to 09/04/2025
Room: N/A (Deluxe)
Total: $-150.00
Status: Pending
Loyalty Used: 0 pts, Coupon: None
Discount: $0
View Details
h
10/04/2025 to 09/04/2025
Room: N/A (Deluxe)
Total: $-150.00
Status: Pending
Loyalty Used: 0 pts, Coupon: None
Discount: $0
h
05/06/2025 to 06/06/2025
Room: N/A (Deluxe)
Total: $150.00
Status: Pending
Loyalty Used: 4 pts, Coupon: 23
Discount: $0
h
08/06/2025 to 09/06/2025
Room: N/A (Deluxe)
Total: $150.00
Status: Pending
Loyalty Used: 0 pts, Coupon: None
Discount: $0

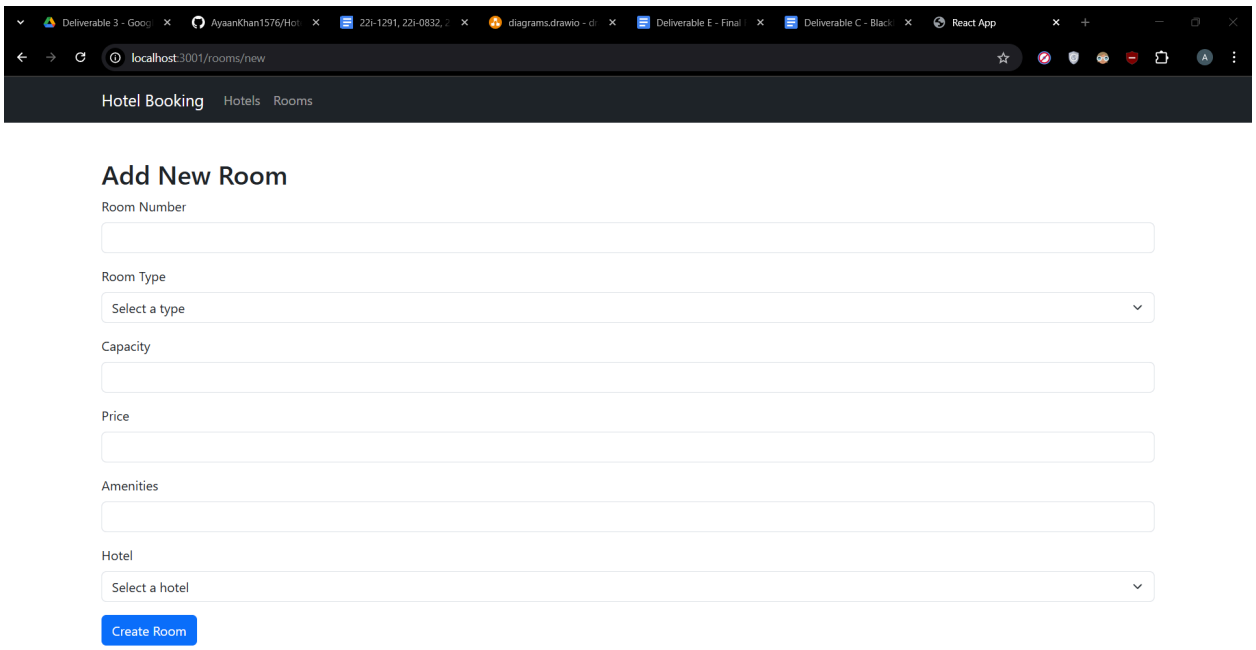

Booking Frontend
localhost:3002/group-booking
Booking Service
New Booking
My Bookings
Group Booking
Group Booking
Agent Email
h3@gmail.com
Discount Rate (%)
10
Select Hotel
Grand Plaza - New York
Check In
06/15/2025
Check Out
06/16/2025
Available Rooms
Min Price
0
Max Price
1000
Room Type
All
101 - Deluxe
Price: $150 per night
Comfortable room with essential amenities.
Select
102 - Suite
Price: $300 per night
Comfortable room with essential amenities.
Select
105 - Deluxe
Price: $150 per night
Comfortable room with essential amenities.
Select

Hotel


Deliverable 3 - Goog
AyaanKhan1576/Hot
22i-1291, 22i-0832, 2
diagrams.drawio - di
Deliverable E - Final
Deliverable C - Black
React App
localhost:3001
Hotel Booking
Hotels
Rooms
Hotels
Add Hotel
Search by location
Name
Location
Contact
Actions
Grand Plaza
New York
123-456-7890
View/Edit
Delete
Tipton Hotel
New York
321-654-987
View/Edit
Delete
Pizza
Islamabad
03245598876
View/Edit
Delete
Grand Plaza
New York
123-456-789
View/Edit
Delete
Grand Plaza
New York
123-456-789
View/Edit
Delete
a
a
4984984984
View/Edit
Delete
e
e
164e7
View/Edit
Delete
e
p
52986498489489494949849498494
View/Edit
Delete

Deliverable 3 - Goog
AyaanKhan1576/Hot
22i-1291, 22i-0832, 2
diagrams.drawio - di
Deliverable E - Final
Deliverable C - Black
React App
localhost:3001/hotels/new
Hotel Booking
Hotels
Rooms
Add New Hotel
Hotel Name
Hotel 2
Location
Lake
Contact Information
123456798
Create Hotel

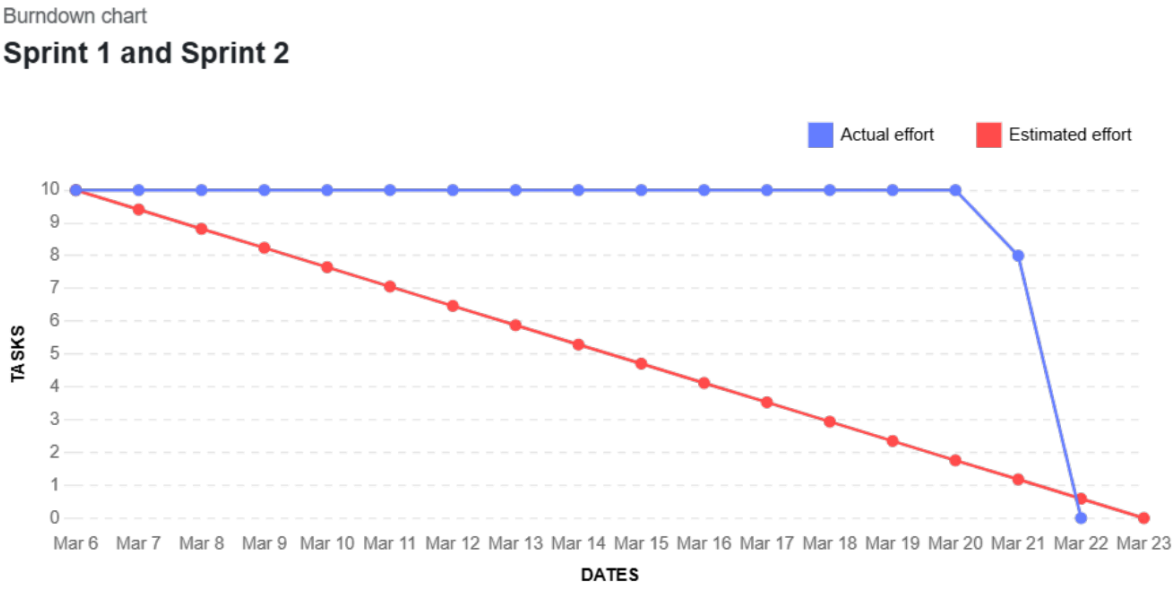




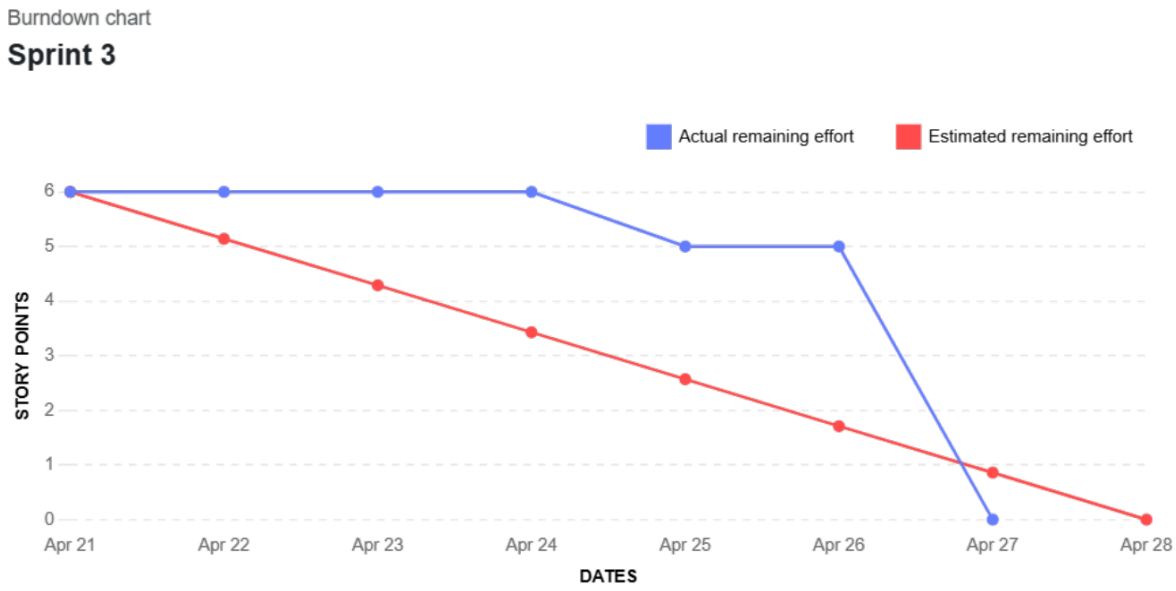
---

10. Product Burndown Chart

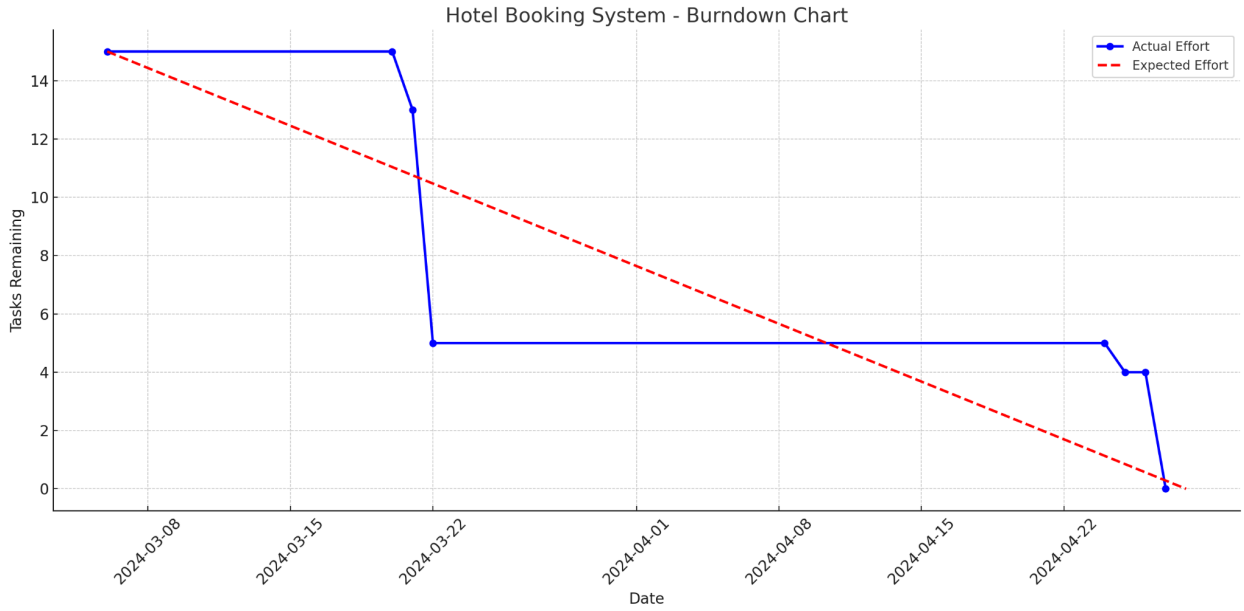
Sprint 1 and Sprint 2:



Sprint 3:



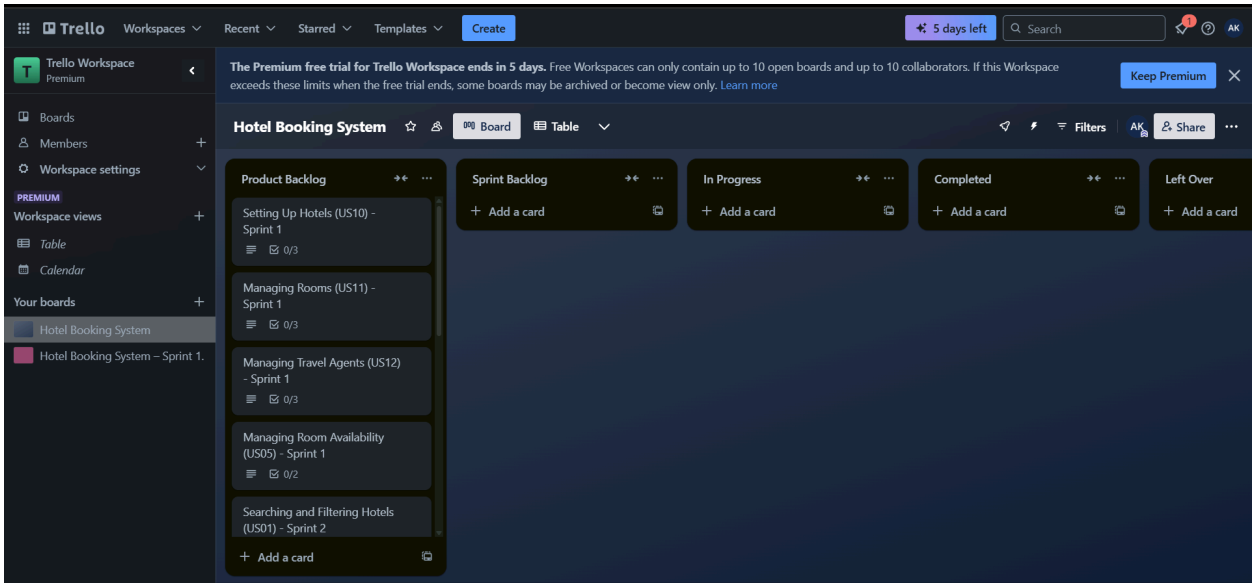
Overall Project:



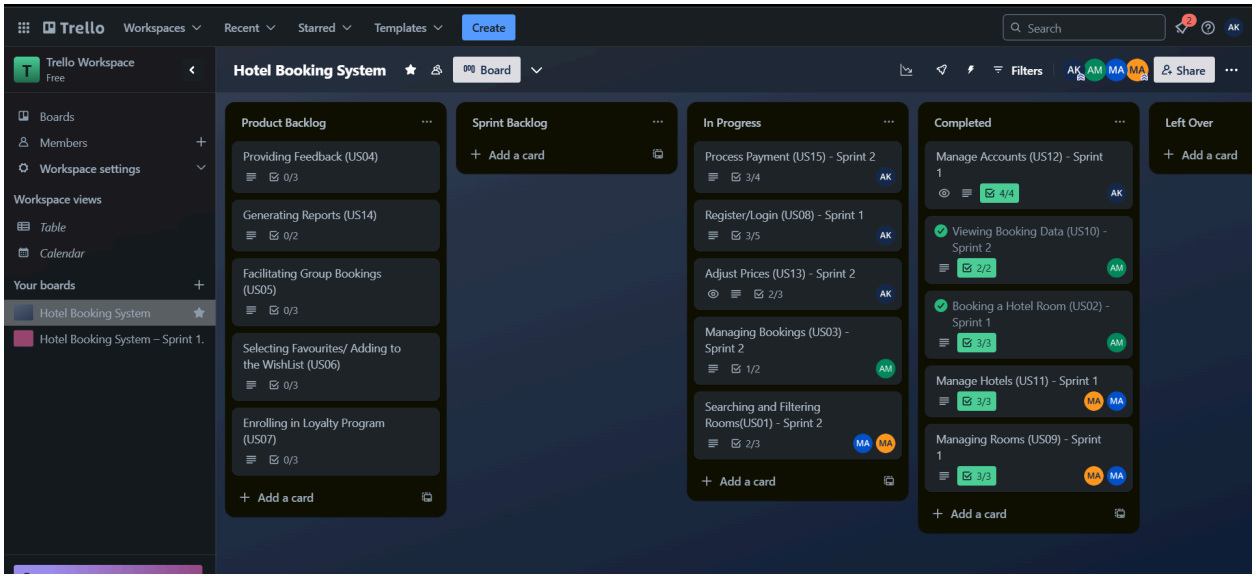
---

11. Trello Board Screenshots

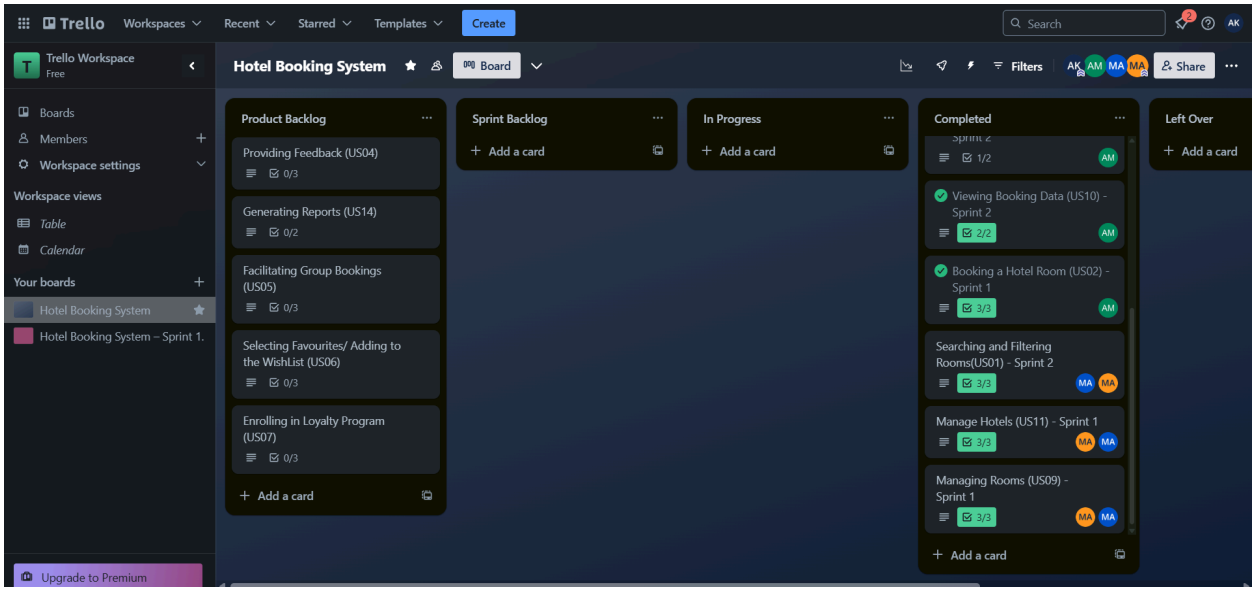
Stage 1 (0% Work Done)



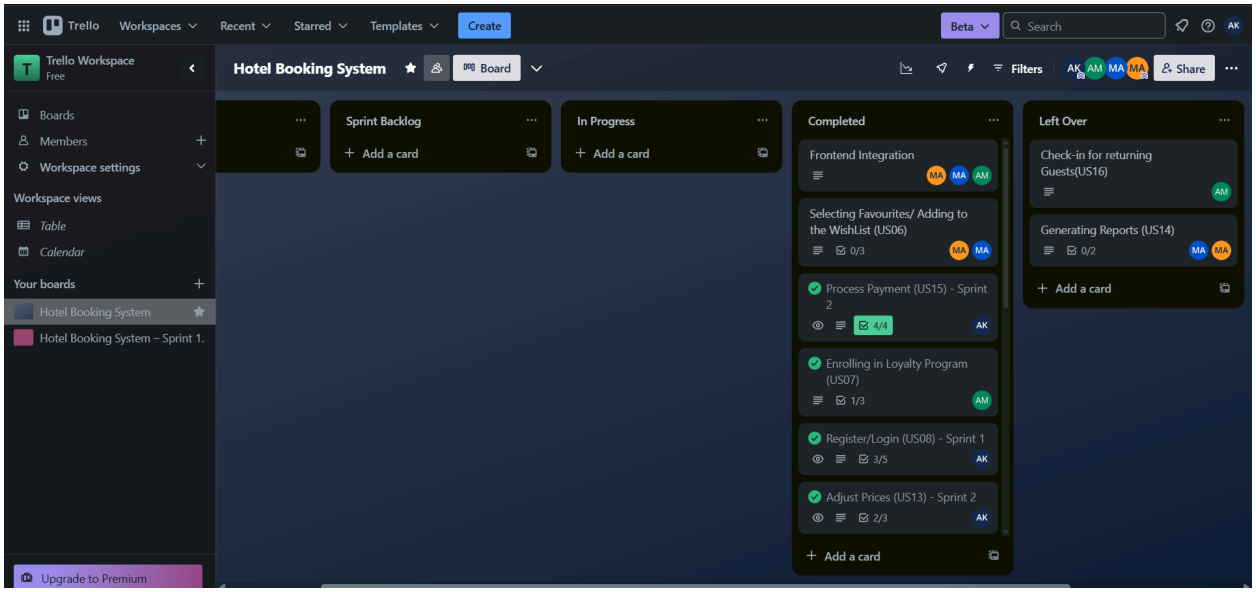
Stage 2 (50% of Sprints 1 and 2 done)



Stage 3 (Sprints 1 and 2 Completed)



Stage 4 (100% Project Completed)



---

12. Test Cases - Black Box

US02: Booking a Hotel Room

Test Case ID: TC-US02-01

- **Test Case Name**: Booking with valid data
- **User Story**: US02: Booking a Hotel Room
- **Input**: Valid guest information, valid card details, valid check-in/check-out dates
- **Expected Output**: Booking is confirmed and receipt is generated
- **Actual Output**: Booking is confirmed and receipt is generated
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Navigate to a hotel listing.
    2. Select an available room.
    3. Fill in valid guest details.
    4. Provide correct payment information.
    5. Click "Book Now".
    6. Verify booking confirmation and receipt.

Test Case ID: TC-US02-02

- **Test Case Name**: Booking with missing invalid name
- **User Story**: US02: Booking a Hotel Room
- **Input**: Guest details filled, Name set to someone else
- **Expected Output**: System shows "Enter correct guest details" error
- **Actual Output**: Booking Confirmed
- **Status**: Failed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Navigate to a hotel listing.
    2. Select a room.
    3. Fill in guest information, leave payment details empty.
    4. Attempt to book.
    5. Verify that an error appears.

Test Case ID: TC-US02-03

- **Test Case Name**: Booking with invalid email
- **User Story**: US02: Booking a Hotel Room
- **Input**: Valid guest information, email doesn’t follow format “hgmail.com”
- **Expected Output**: System rejects transaction with "Invalid email syntax" message
- **Actual Output**: Rejects booking and gives error
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Select a hotel and a room.
    2. Enter valid guest details.
    3. Enter invalid credit card number (e.g., 1234).
    4. Attempt to book.
    5. Verify that payment is rejected.

Test Case ID: TC-US02-04

- **Test Case Name**: Booking with incomplete guest details
- **User Story**: US02: Booking a Hotel Room
- **Input**: Missing fields in guest info e.g missing name
- **Expected Output**: System displays validation error
- **Actual Output**: Validation error
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:

1. Select hotel and room.
2. Leave name field blank.
3. Attempt to book.
4. Observe system validation.

Test Case ID: TC-US02-05

- **Test Case Name**: Booking on current date (minimum valid check-in)
- **User Story**: US02: Booking a Hotel Room
- **Input**: Check-in date = Today's date
- **Expected Output**: Booking is allowed
- **Actual Output**: Room not available
- **Status**: Failed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Book hotels with today's date.
    2. Select room.
    3. Complete booking process.
    4. Verify booking success.

Test Case ID: TC-US02-06

- **Test Case Name**: Booking with check-in date in the past (invalid)
- **User Story**: US02: Booking a Hotel Room
- **Input**: Check-in date = Yesterday's date
- **Expected Output**: System should reject with "Invalid date" error or Can’t select date
- **Actual Output**: Can’t select date
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Book hotels with check-in = yesterday.
    2. Attempt to proceed.
    3. Observe system error.

Test Case ID: TC-US02-07

- **Test Case Name**: Booking minimum stay (1 night)
- **User Story**: US02: Booking a Hotel Room
- **Input**: Check-in = 10-May, Check-out = 11-May
- **Expected Output**: Booking allowed (1-night stay)
- **Actual Output**: Booking allowed

- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Select check-in = 10-May, check-out = 11-May.
    2. Book a room.
    3. Verify success.

Test Case ID: TC-US02-08

- **Test Case Name**: Booking checkout allowed same day
- **User Story**: US02: Booking a Hotel Room
- **Input**: checkout = checkin
- **Expected Output**: Booking not allowed
- **Actual Output**: Booking not allowed
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Select a check-in and check-out gap of 30 nights.
    2. Complete booking.
    3. Verify booking accepted.

US11: Manage Hotels

Test Case ID: TC-US11-01

- **Test Case Name**: Adding a new hotel with all valid information
- **User Story**: US11: Manage Hotels
- **Input**: Hotel name, location, contact info all valid
- **Expected Output**: Hotel successfully added to the system
- **Actual Output**: Hotel successfully added to system
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Login as Admin.
    2. Navigate to "Add Hotel" form.
    3. Fill all fields correctly.
    4. Click "Save".
    5. Verify that hotel appears in listings.

Test Case ID: TC-US11-02

- **Test Case Name**: Adding hotel with missing required fields
- **User Story**: US11: Manage Hotels
- **Input**: Missing hotel name or contact info
- **Expected Output**: System displays validation error
- **Actual Output**: Validation error
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Leave hotel name blank.
    2. Attempt to submit form.
    3. Verify error for missing field.

Test Case ID: TC-US11-03

- **Test Case Name**: Adding a duplicate hotel
- **User Story**: US11: Manage Hotels
- **Input**: Hotel name already exists
- **Expected Output**: System rejects duplicate entry
- **Actual Output**: Hotel successfully added to system
- **Status**: Failed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Add a hotel with an existing name.
    2. Submit.
    3. Confirm error message about duplication.

Test Case ID: TC-US11-04

- **Test Case Name**: Hotel name minimum characters
- **User Story**: US11: Manage Hotels
- **Input**: Hotel name = 1 character
- **Expected Output**: Hotel added to system
- **Actual Output**: Hotel added to system
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Input a 1-character name.

2. Save.
3. Check result.

Test Case ID: TC-US11-05

- **Test Case Name**: Hotel contact number has letter in it
- **User Story**: US11: Manage Hotels
- **Input**: Hotel contact number has letter in it
- **Expected Output**: Rejected
- **Actual Output**: Accepted
- **Status**: Failed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Input a number with a letter
    2. Save.
    3. Confirm correct behavior.

Test Case ID: TC-US11-06

- **Test Case Name**: Contact phone number boundary test
- **User Story**: US11: Manage Hotels
- **Input**: Phone number with minimum and maximum allowed digits
- **Expected Output**: Validation success or failure based on phone rules
- **Actual Output**: Hotel Booked
- **Status**: Failed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Enter less than 7-digit number (minimum boundary).
    2. Enter more than 15-digit number (maximum boundary).
    3. Try to save and observe validations.

US09: Manage Rooms

Test Case ID: TC-US09-01

- **Test Case Name**: Adding a new room with all valid fields
- **User Story**: US09: Manage Rooms
- **Input**: Room type, price, capacity, special features correctly filled

- **Expected Output**: Room added successfully
- **Actual Output**: Room added successfully
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Navigate to Room Management.
    2. Fill all room fields correctly.
    3. Save.
    4. Verify room listing update.

Test Case ID: TC-US09-02

- **Test Case Name**: Adding a room with missing price field
- **User Story**: US09: Manage Rooms
- **Input**: Room fields filled but price left blank
- **Expected Output**: System asks to add price
- **Actual Output**: Asks to enter price
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Fill room details.
    2. Leave price blank.
    3. Save.
    4. Verify error handling.

Test Case ID: TC-US09-03

- **Test Case Name**: Updating room details successfully
- **User Story**: US09: Manage Rooms
- **Input**: Existing room edited (new price)
- **Expected Output**: Room details updated successfully
- **Actual Output**: Room details updated successfully
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Find existing room.
    2. Edit price.
    3. Save changes.
    4. Confirm updates.

Test Case ID: TC-US09-04

- **Test Case Name**: Removing a room
- **User Story**: US09: Manage Rooms
- **Input**: Select an existing room and remove
- **Expected Output**: Room removed successfully
- **Actual Output**: Room removed
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Open Room List.
    2. Select a room.
    3. Remove.
    4. Verify deletion.

Test Case ID: TC-US09-05

- **Test Case Name**: Room capacity minimum limit
- **User Story**: US09: Manage Rooms
- **Input**: Capacity = 1 person
- **Expected Output**: Room added successfully
- **Actual Output**: Room added
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Add room with capacity = 1.
    2. Save.
    3. Confirm acceptance.

Test Case ID: TC-US09-06

- **Test Case Name**: Room capacity negative number
- **User Story**: US09: Manage Rooms
- **Input**: Capacity = -10
- **Expected Output**: Validation error
- **Actual Output**: Validation error
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Enter capacity = -1

2. Try saving.
3. Confirm error.

US08: Register/Login

Test Case ID: TC-US08-01

- **Test Case Name**: Successful user registration
- **User Story**: US08: Register/Login
- **Input**: Valid email , strong password
- **Expected Output**: Account created successfully
- **Actual Output**: Account created
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Open the registration page.
    2. Enter valid email, strong password
    3. Click "Register."
    4. Observe that account is created.

Test Case ID: TC-US08-02

- **Test Case Name**: Login with valid credentials
- **User Story**: US08: Register/Login
- **Input**: Registered email and correct password
- **Expected Output**: User successfully logged in and redirected to homepage
- **Actual Output**: User logged in and redirected to homepage
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Open login page.
    2. Enter registered email and correct password.
    3. Click "Login."
    4. Verify that user is logged in and redirected.

Test Case ID: TC-US08-03

- **Test Case Name**: Registration with invalid email

- **User Story**: US08: Register/Login
- **Input**: Invalid email format (e.g. usergmail.com) and strong password
- **Expected Output**: Error message: "Enter valid email"
- **Actual Output**: Error message
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Open registration form.
    2. Enter invalid email format.
    3. Submit the form.
    4. Check that validation error appears.

Test Case ID: TC-US08-04

- **Test Case Name**: Password minimum length boundary
- **User Story**: US08: Register/Login
- **Input**: Password = exactly 6 characters
- **Expected Output**: Password accepted, registration successful
- **Actual Output**: Registration successful
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Open registration page.
    2. Enter password exactly 6 characters long.
    3. Register.
    4. Verify success.

Test Case ID: TC-US08-05

- **Test Case Name**: Password just below minimum length
- **User Story**: US08: Register/Login
- **Input**: Password = 5 characters
- **Expected Output**: Validation error "Password too short"
- **Actual Output**: Error
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Try registering with 5-character password.
    2. Submit.
    3. Verify error shown.

Test Case ID: TC-US08-06

- **Test Case Name**: Login attempt with wrong password
- **User Story**: US08: Register/Login
- **Input**: Correct email, wrong password
- **Expected Output**: "Incorrect password" error
- **Actual Output**: Error
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Open login page.
    2. Enter valid email but wrong password.
    3. Try to login.
    4. Observe error message.

US01: Searching and Filtering Hotels

Test Case ID: TC-US01-01

- **Test Case Name**: Valid hotel search
- **User Story**: US01: Searching and Filtering Hotels
- **Input**: City = "New York"
- **Expected Output**: List of hotels matching criteria
- **Actual Output**: “Grand Plaza”, “Tipton Hotel”
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Open the search page.
    2. Enter “New York”
    3. Click Search.
    4. Observe hotel listings.

Test Case ID: TC-US01-02

- **Test Case Name**: Invalid city search
- **User Story**: US01: Searching and Filtering Hotels
- **Input**: City = "@#$%^", valid dates
- **Expected Output**: Error "No hotels found" or Empty

- **Actual Output**: Empty
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Enter invalid characters in city.
    2. Click Search
    3. Observe Output

Test Case ID: TC-US01-03

- **Test Case Name**: Search with minimum characters
- **User Story**: US01: Searching and Filtering Hotels
- **Input**: City = "N”
- **Expected Output**: Hotels in Cities starting from N
- **Actual Output**: “Grand Plaza”, “Tipton Hotel”
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Search using 1-letter city.
    2. Click Search.
    3. Verify behavior.

Test Case ID: TC-US01-04

- **Test Case Name**: Search with maximum input length
- **User Story**: US01: Searching and Filtering Hotels
- **Input**: City = long name (50+ chars)
- **Expected Output**: Empty
- **Actual Output**: Empty
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Enter maximum length city.
    2. Search.
    3. Check if system processes or truncates input.

US12: Manage Accounts

Test Case ID: TC-US12-01

- **Test Case Name**: Creating a new user account successfully
- **User Story**: US12: Manage Accounts
- **Input**: Valid username, email, and password
- **Expected Output**: User account created successfully
- **Actual Output**: User account created successfully
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Admin logs in.
    2. Go to API using postman
    3. Fill all fields correctly.
    4. Save.
    5. Verify account in Database

Test Case ID: TC-US12-02

- **Test Case Name**: Username minimum length validation
- **User Story**: US12: Manage Accounts
- **Input**: Username = 1 character
- **Expected Output**: User registered
- **Actual Output**: User registered
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Try creating a user with 1-character username.
    2. Save.
    3. Verify

US13: Adjust Prices

Test Case ID: TC-US13-01

- **Test Case Name**: Successfully updating room price
- **User Story**: US13: Adjust Prices
- **Input**: Update price from $100 to $120
- **Expected Output**: Price updated successfully
- **Actual Output**: Price updated
- **Status**: Passed

- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Admin logs in.
    2. Edit a room's price field.
    3. Save changes.
    4. Verify updated price shown.

Test Case ID: TC-US13-02

- **Test Case Name**: Price minimum value validation
- **User Story**: US13: Adjust Prices
- **Input**: Enter price = 0
- **Expected Output**: Validation error
- **Actual Output**: Room price changed
- **Status**: Failed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Set room price = 0.
    2. Try to save.
    3. Confirm system error.

US03: Managing Bookings

Test Case ID: TC-US03-01

- **Test Case Name**: Cancel Booking
- **User Story**: US03: Managing Bookings
- **Input**: Logged-in user with bookings
- **Expected Output**: Booking no longer in list
- **Actual Output**: Booking still in list
- **Status**: Failed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Login as user.
    2. Open "My Bookings".
    3. Cancel booking
    4. Verify

Test Case ID: TC-US03-02

- **Test Case Name**: Update Booking details
- **User Story**: US03: Managing Bookings
- **Input**: logged in user with bookings
- **Expected Output**: Booking details updated
- **Actual Output**: Booking details updated
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Login with a new account.
    2. Open "My Bookings".
    3. Select Booking and update details
    4. Verify

US10: Viewing Booking Data

Test Case ID: TC-US10-01

- **Test Case Name**: Retrieve guest booking details successfully
- **User Story**: US10: Viewing Booking Data
- **Input**: Search by valid guest name
- **Expected Output**: Guest bookings displayed
- **Actual Output**: Bookings displayed
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Staff logs in.
    2. Search guest by name.
    3. Confirm booking data displayed.

Test Case ID: TC-US10-02

- **Test Case Name**: Search with empty input (boundary)
- **User Story**: US10: Viewing Booking Data
- **Input**: Empty search field
- **Expected Output**: Validation error
- **Actual Output**: Validation error
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis

- **Steps to Execute**:
    1. Leave search field empty.
    2. Try searching.
    3. Verify error.

US05: Facilitating Group Bookings

Test Case ID: TC-US05-01

- **Test Case Name**: Successfully book multiple rooms
- **User Story**: US05: Facilitating Group Bookings
- **Input**: 2 rooms booked together
- **Expected Output**: Group booking success
- **Actual Output**: Group booking success
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Login as Travel Agent.
    2. Select 2 rooms.
    3. Complete group booking.

Test Case ID: TC-US05-02

- **Test Case Name**: Booking minimum allowed rooms
- **User Story**: US05: Facilitating Group Bookings
- **Input**: 0 rooms booked
- **Expected Output**: Booking rejected
- **Actual Output**: Booking rejected
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Book exactly 0 rooms.
    2. Complete booking.
    3. Confirm system allows.

US07: Enrolling in Loyalty Program

Test Case ID: TC-US07-01

- **Test Case Name**: Enroll into loyalty program after eligibility
- **User Story**: US07: Enrolling in Loyalty Program
- **Input**: User meets booking criteria
- **Expected Output**: Enrollment success
- **Actual Output**: Enrollment success
- **Status**: Passed
- **Testing Method**: Equivalence Class Partitioning
- **Steps to Execute**:
    1. Complete required number of bookings.
    2. Enroll in loyalty program.
    3. Verify confirmation.

Test Case ID: TC-US07-02

- **Test Case Name**: Loyalty points redemption boundary
- **User Story**: US07: Enrolling in Loyalty Program
- **Input**: Redeem exactly 500 points (minimum)
- **Expected Output**: Discount applied successfully
- **Actual Output**: Discount applied successfully
- **Status**: Passed
- **Testing Method**: Boundary Value Analysis
- **Steps to Execute**:
    1. Try redeeming 500 points.
    2. Complete booking.
    3. Confirm discount applied.

---

13. Test Cases - White Box

Overview

Whitebox testing was conducted across all backend microservices: **User Service**, **Hotel Service**, and **Booking Service**. The goal was to achieve high coverage by testing critical functions, APIs, database interactions, and logic flows using unit and integration tests.
Testing was performed using **Jest**, **Supertest.**

What is Covered

- **Core Business Logic**:
    - Critical flows such as user registration, login, loyalty enrollment/rewards, hotel management (CRUD), room availability, booking creation, cancellation, payment processing, and group booking management were extensively tested.

- **API Endpoints**:
    - All major HTTP endpoints were unit-tested and integration-tested using **Supertest**.
    - Request validation, success responses, and error handling paths were covered.

- **Database Operations**:
    - MongoDB model interactions (using Mongoose) such as creating, updating, deleting, and querying documents were tested.
    - Mocked database behavior was used where appropriate to simulate edge cases and failures.

- **Branch Coverage**:
    - Multiple conditions such as success paths, missing data, invalid inputs, not found errors, and server errors were handled in tests.
    - Both positive and negative test scenarios were covered for major controllers.

- **Function and Statement Coverage**:
    - Most controllers, utility functions, and models have above **70%+** function and statement coverage, with some services exceeding **80%**

What is Not Covered and Why

- **Third-party Libraries**:
    - Built-in third-party libraries such as **Mongoose**, **Axios**, **bcrypt**, and **jsonwebtoken** were **mocked** or **assumed to work** correctly without direct unit testing their internals, since these are external and already well-tested libraries.

- **Asynchronous Cron Jobs**:
    - Scheduled jobs (cron jobs) that automatically free up room availability after checkout were not exhaustively tested because they run on background schedulers, and mocking long-running timed processes was not prioritized.

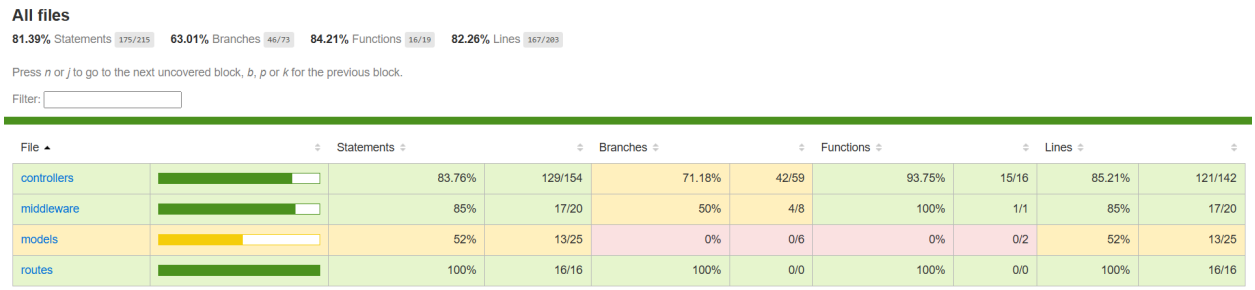
- **Error Paths for External Services**:
    - Some error handling paths involving external services (like Room Service or User Service) were not deeply simulated with network errors or timeouts.

    - While API call failures were mocked, full resilience against external service crashes was not fully tested.

- **Frontend**:
    - **No frontend whitebox testing** was performed in this scope. The React frontend was excluded, and only backend services were tested.

Screenshots

- ## User Service



All files

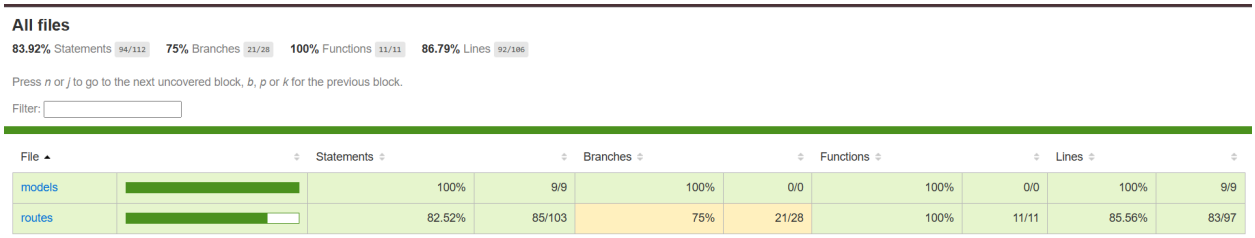
81.39% Statements 175/215 **63.01%** Branches 46/73 **84.21%** Functions 16/19 **82.26%** Lines 167/203

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	
controllers	83.76%	129/154	71.18%	42/59	93.75%	15/16	85.21%	121/142
middleware	85%	17/20	50%	4/8	100%	1/1	85%	17/20
models	52%	13/25	0%	0/6	0%	0/2	52%	13/25
routes	100%	16/16	100%	0/0	100%	0/0	100%	16/16

- ## Hotel Service



All files

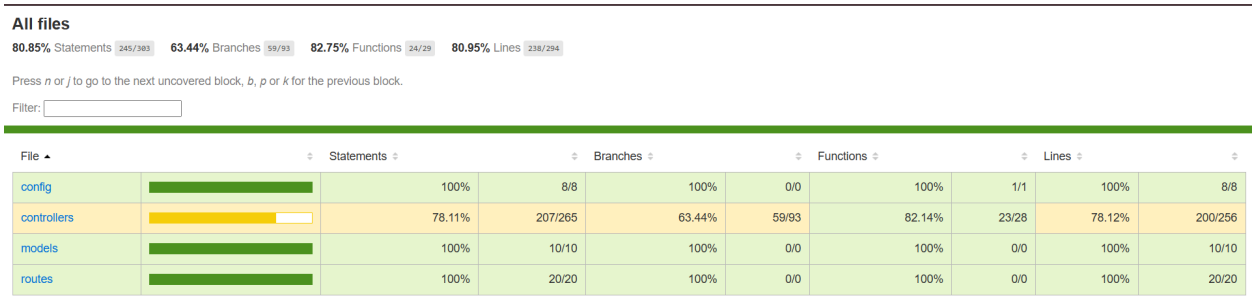
83.92% Statements 94/112 **75%** Branches 21/28 **100%** Functions 11/11 **86.79%** Lines 92/106

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	
models	100%	9/9	100%	0/0	100%	0/0	100%	9/9
routes	82.52%	85/103	75%	21/28	100%	11/11	85.56%	83/97

- ## Booking Service



Each report shows the achieved levels of:

- **Statement Coverage**
- **Branch Coverage**
- **Function Coverage**
- **Line Coverage**

---

14. Work Division Between Group Members

Ayaan Khan

Scrum Master, Project Manager, Tester

Assigned user microservice related user stories and implemented them. Role in testing and deployment. Responsible for setting up project and trello board. Worked on Deliverable 3 documentation.

Ayaan Mughal

UI Designer, Developer

Assigned booking microservice related user stories and implemented them. Role in frontend and testing. Responsible for services integration, and UI design. Worked on Deliverable 1 documentation.

Mishal Ali

Requirements Architect, Developer

Assigned hotel microservice related user stories and implemented them. Role in frontend and deployment. Responsible for requirements analysis and Deliverable 2 documentation. Also responsible for UI design and setting up project.

---

15. Lessons Learned

- Importance of sprint planning and flexible backlog grooming.
- Blackbox and Whitebox (Jest) testing and advantages to find bugs
- Managing version control and team collaboration effectively (GitHub, Trello).

- Testing at each layer (Frontend, API, Database) avoids last-minute bugs.
- Criticality of containerized deployment (Docker) for modular development.
- Advantage of microservices architecture over monolithic architecture in terms of modularity

---