

**American International University-Bangladesh (AIUB)**  
 Department of Computer Science

Faculty of Science & Technology (FST)

***Travel and Ticket Management System***

**Semester: Fall 25-26**

|  |  |  |
| --- | --- | --- |
| ***Group:*** | | ***Section:*** |
| **SL** | **Student Name** | **Student ID** |
| 1 | Md. Rasel Hossain | 23-55576-3 |
| 2 | Mohammad Hasib Khandaker | 23-55572-3 |
| 3 | Md. Taslimul Mahbub Jitu | 23-55552-3 |
| 4 |  |  |
| 5 |  |  |

Table of Contents

[1. Project Proposal xx](#_Toc214214201)

[1.1 Background to the Problem](#_Toc214214203) xx

[1.2 Selection of Process Model xx](#_Toc214214204)

[2. Software Requirements Specification (SRS) / PRD xx](#_Toc214214202)

[2.1 Scopes and Features xx](#_Toc214214203)

[2.2 User Story Table xx](#_Toc214214204)

[2.3 Requirements Traceability Matrix xx](#_Toc214214205)

[2.3.1 Functional Requirements xx](#_Toc214214212)

[2.3.2 Non-Functional Requirements xx](#_Toc214214216)

[3. Software Design xx](#_Toc214214202)

[3.1 System Deisgn xx](#_Toc214214203)

[3.2 UI Deisgn using Figma xx](#_Toc214214203)

[4. Git Workflow xx](#_Toc214214202)

[5. Software Testing xx](#_Toc214214202)

[6. Conclusion xx](#_Toc214214202)

**1. PROJECT PROPOSAL**



* 1. **Background to the Problem**

The rapid growth of the global travel and aviation industry has significantly increased the need for fast, accurate, and reliable ticket management solutions. Despite this advancement, many local and small-scale travel agencies still depend on traditional, paper-based, or semi-manual processes for booking, managing, and storing flight ticket information. These traditional methods often lead to critical issues such as inaccurate data handling, duplicate records, operational delays, loss of important information, and overall inefficient customer service. As a result, customers face long waiting times, limited access to real-time flight information, and poor transparency in ticketing procedures.

Furthermore, the manual system provides limited support for managing large volumes of customers or group bookings. Agents struggle to handle data consistently, while administrators lack a centralized way to monitor user activity, maintain records, or evaluate performance. The absence of a secure and integrated platform makes the system vulnerable to errors, unauthorized access, and data inconsistency. In modern digital environments, such limitations hinder customer satisfaction and reduce operational reliability.

To address these challenges, there is a need for a **centralized, automated, and user-friendly** travel and ticket management system that improves operational workflow and provides real-time, secure access to data. The proposed **Travel and Ticket Management System** aim to fill this gap by offering a complete digital platform built using C# Windows Forms and SQL Server. This system not only automates the entire ticket management process but also ensures transparency, accuracy, and efficient handling of flight information. Through a multi-layered, object-oriented architecture, the system provides different access levels Customer, Agent, and Admin each designed to support specific roles and responsibilities while maintaining overall system integrity.

The system empowers customers with capabilities such as registration, login, search for flights, ticket booking, history tracking, and submitting reviews or reports. Agents are given enhanced tools to manage up to ten tickets at a time, update client information, process bookings, and respond to customer feedback. Administrators, functioning as the highest-level authority, can manage all users, oversee booking activities, monitor reviews, and maintain database records. This structured control ensures that every operation from booking and updating tickets to user management and system monitoring is handled efficiently and securely.

By providing an automated alternative to the traditional workflow, the system reduces human error, accelerates ticket processing time, ensures secure data storage, and enhances the user experience. Moreover, it supports scalability and future improvements, making it suitable for travel agencies of various sizes.

## ****Target Group of Users****

The proposed system is designed for three main user groups:

### ****Customers (Travelers)****

Individuals who want to book flight tickets, check availability, view ticket history, update passenger details, submit reviews, and manage personal profiles. This group benefits from fast, transparent, and secure booking processes.

### ****Agents (Ticketing/Travel Agency Staff)****

Staff members responsible for handling customer bookings, managing group reservations, maintaining passenger information, and generating reports. Agents require efficiency, accuracy, and tools to manage multiple bookings simultaneously.

### ****Administrators (System/Agency Managers)****

Admin users who oversee system operations, manage all user accounts, handle booking oversight, process feedback, and ensure overall system security and performance. They require complete control and access to all system resources.

* 1. **Selection of Process Model**

1. **Explanation of Selected Process Model – Agile (Scrum) Process Model**

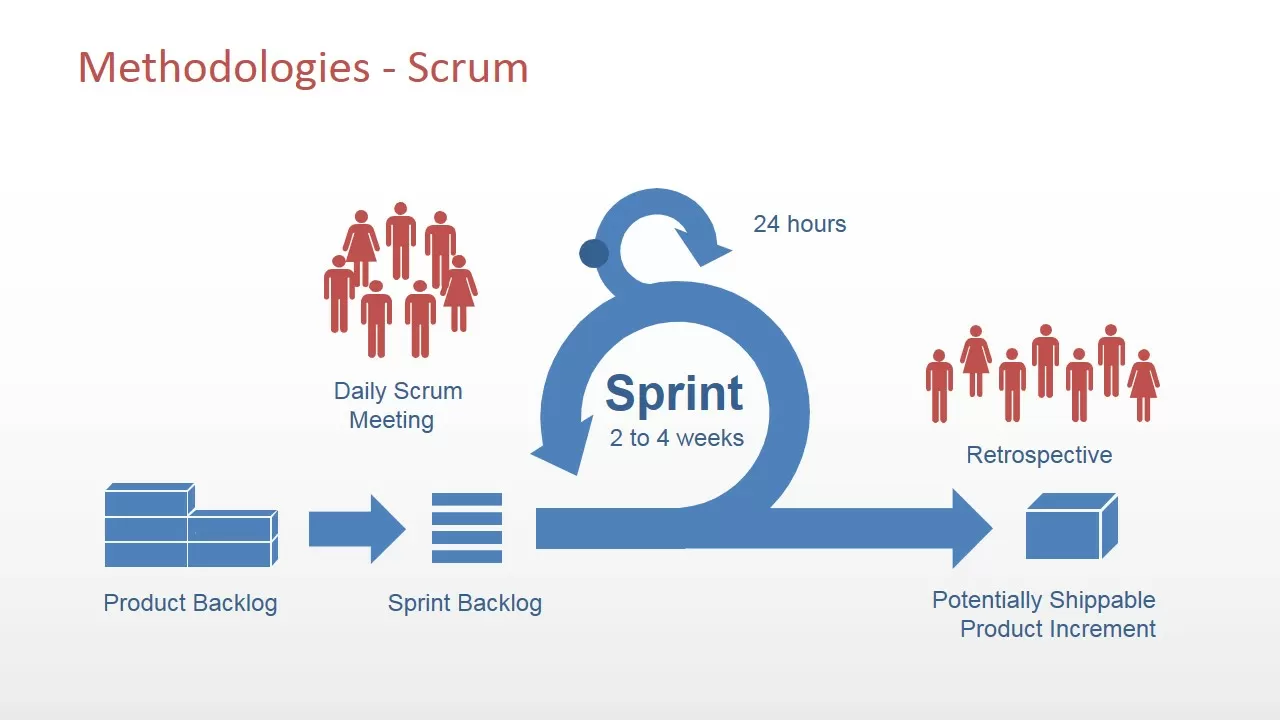
For the **Travel and Ticket Management System**, the **Agile Process Model**, particularly the **Scrum framework**, has been selected as the most suitable software development process model.

Agile is an **iterative and incremental development approach** where the system is developed in small, manageable phases called **sprints**. Each sprint delivers a functional and testable part of the system within a short time frame (usually **1–2 weeks**). Instead of building the entire system in a single cycle, Agile allows continuous development, testing, and improvement throughout the project lifecycle.

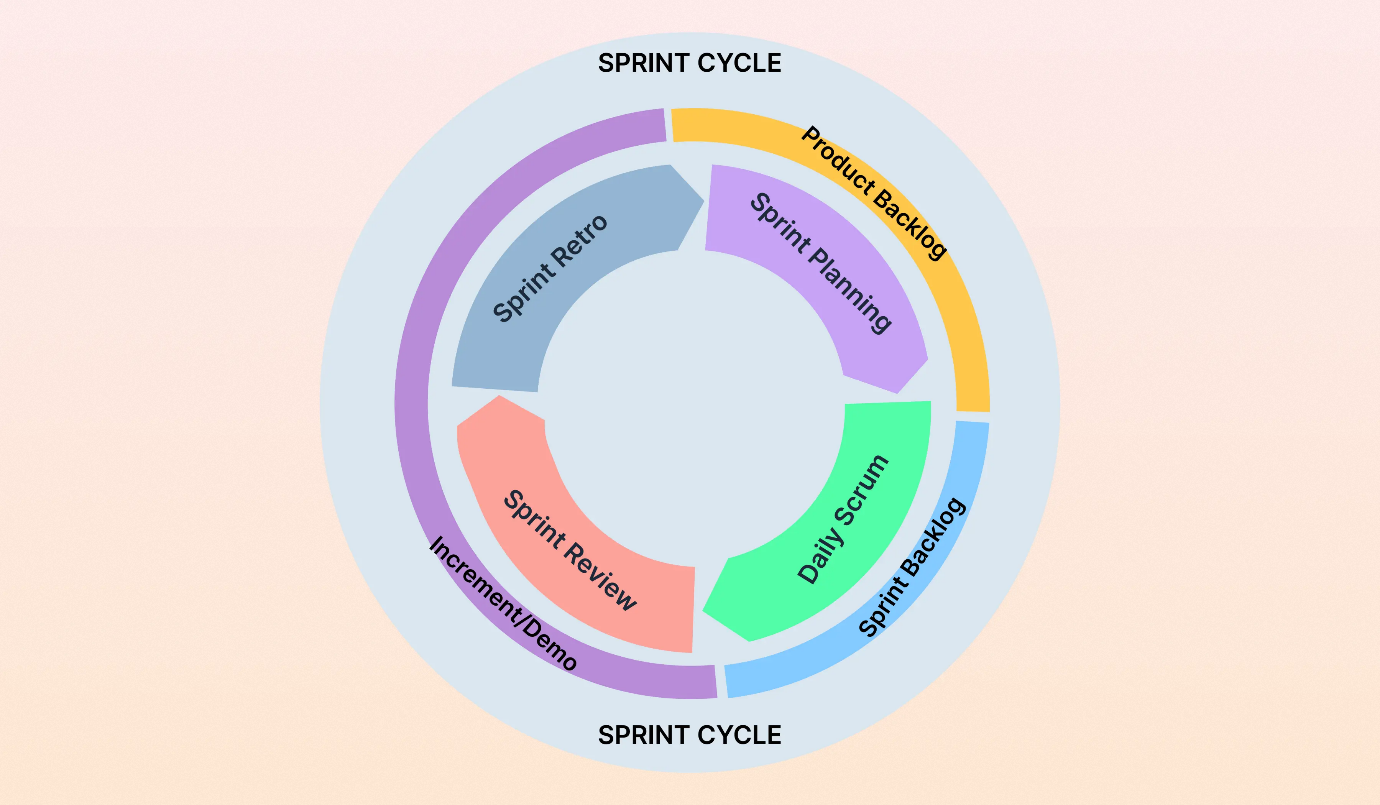
In the Scrum-based Agile model:

* System requirements are maintained in a **Product Backlog**, which includes all functional and non-functional requirements such as user authentication, ticket booking, role-based access, and reporting.
* The project is divided into **modules**, including:
  1. User Registration & Login
  2. Flight Search & Booking
  3. Ticket Management
  4. Customer Review & Feedback
  5. Agent Operations
  6. Admin Management & Monitoring
* Each sprint includes the following activities:
  1. Sprint Planning
  2. Design & Development
  3. Testing & Debugging
  4. Sprint Review
  5. Integration and Improvement

At the end of every sprint, a **working increment** of the Travel and Ticket Management System is produced, such as a completed booking module or a fully functional admin dashboard. This allows early verification of system behavior and usability.







The Agile (Scrum) model emphasizes **continuous user feedback, collaboration, and adaptability**, making it highly suitable for a **multi-user, database-driven, and role-based system** like the Travel and Ticket Management System.

**ii) Why We Selected This Process Model**

The **Agile (Scrum) Process Model** was selected for the following reasons:

**1. Frequently Changing and Expanding Requirements**

The Travel and Ticket Management System involves multiple user roles (Customer, Agent, Admin) and complex functionalities such as booking, ticket history, feedback, and system monitoring. As development progresses, new requirements or improvements may be identified. Agile supports changes at any stage without affecting the entire system structure.

**2. Modular Development Structure**

Each major feature of the system—such as flight search, booking management, and admin control—can be developed independently in separate sprints. This aligns perfectly with Agile’s incremental development approach and ensures better organization and clarity.

**3. Improved Collaboration and Task Distribution**

Since this is a group-based software project, Agile supports teamwork through sprint planning and task distribution. Each team member can work on assigned modules while maintaining synchronization with the overall system.

**4. Early Testing and Error Reduction**

Continuous testing after each sprint helps detect errors early. This is critical for a system that handles **user data, booking records, and authentication**, where accuracy and data integrity are essential.

**5. Better Progress Monitoring**

Agile allows the project supervisor or instructor to review progress frequently. Instead of waiting until the final submission, the system can be demonstrated incrementally with working features.

**6. Scalability and Future Enhancement**

The Travel and Ticket Management System may be extended in the future to include online payment gateways, mobile integration, or analytics. Agile supports easy scalability without redesigning the entire system.

* **Benefits of Using Scrum in This Project**

1. **Transparency** – Progress is visible through sprint backlogs and completed modules.
2. **Inspection** – Bugs and logic errors are identified early during sprint reviews.
3. **Adaptation** – System features can be modified easily based on feedback or new requirements.

* **Why Agile (Scrum) is Best for This System**

The Travel and Ticket Management System is:

1. Multi-user and role-based
2. Database-intensive
3. Feature-rich
4. Expected to evolve during development

Therefore, the **Agile (Scrum) Process Model** is a natural and effective choice for ensuring successful development, timely delivery, and high system quality.

**Common Scrum Terminologies Used**

* **Sprint** – A short development cycle (1–2 weeks)
* **Product Backlog** – Prioritized list of system features
* **Sprint Backlog** – Tasks selected for a specific sprint
* **Increment** – A functional part of the system delivered after a sprint
* **User Story** – Requirement described from the user’s perspective
* **Sprint Planning** – Meeting to define sprint goals
* **Sprint Review** – Demonstration of completed features
* **Scrum Master** – Facilitates Scrum activities
* **Product Owner** – Manages system requirements
* **Development Team** – Team members who build the system

## 2. SOFTWARE REQUIREMENTS SPECIFICATIONS (SRS) / PRODUCT REQUIREMENTS DOCUMENT (PRD)

**2.1 Scopes and Features**

**Scope 1: User Management & Authentication**

**Features:**

* Customers can register using name, email, username, and password
* Admin can manually create Customer and Agent accounts
* Secure login system for all users
* Forgot password functionality using email or username
* Users can update personal profile information
* Admin can activate, deactivate, edit, or delete user accounts
* Role-based access control (Customer, Agent, Admin)

**Scope 2: Flight Search & Ticket Booking**

**Features:**

* Customers and Agents can search flights using source and destination cities
* Display of flight details (date, time, price, seat availability)
* Customers can book individual tickets
* Agents can book up to **10 tickets at a time**
* Passenger details can be entered and modified before confirmation
* Automatic booking confirmation generation
* Users can view booking history anytime

**Scope 3: Passenger & Booking Management**

**Features:**

* Edit passenger information before booking confirmation
* Admin can view, update, or cancel any booking
* Agents can manage group reservations
* Travel cost estimation between cities
* All booking records stored in a centralized SQL Server database

**Scope 4: Review & Reporting System**

**Features:**

* Customers can submit travel reviews
* Customers can submit complaints or reports
* Agents and Admin can view reviews and reports
* Agents can respond to customer feedback
* Admin can analyze service quality using reviews

**Scope 5: Administrative Control Panel**

**Features:**

* Full CRUD operations on users
* DataGridView used for displaying database records
* Admin can cancel any ticket
* Admin can monitor agent performance
* Booking statistics and system logs available

**Scope 6: Security & System Control**

**Features:**

* Secure login and logout
* Password encryption
* Session management
* Role-based authorization (ACL)
* Safe system exit and shutdown

**2.2 User Story Table**

**2.2.1 Customer User Stories**

|  |  |  |  |
| --- | --- | --- | --- |
| As a User | I want to | So that | Acceptance Criteria |
| Customer | Register an account | I can access the system | Valid data, unique email/username |
| Customer | Login securely | I can use my dashboard | Valid credentials required |
| Customer | Recover password | I regain access | Reset link sent successfully |
| Customer | Search flights | I find suitable options | Results displayed correctly |
| Customer | Book tickets | I confirm travel | Seat availability verified |
| Customer | Update passenger info | Data remains accurate | Changes saved |
| Customer | View booking history | I track past trips | History displayed |
| Customer | Write reviews | Share experience | Review saved |
| Customer | Submit complaints | Admin is informed | Complaint recorded |
| Customer | Update profile | Data stays current | Update successful |
| Customer | Logout | Account remains secure | Session terminated |

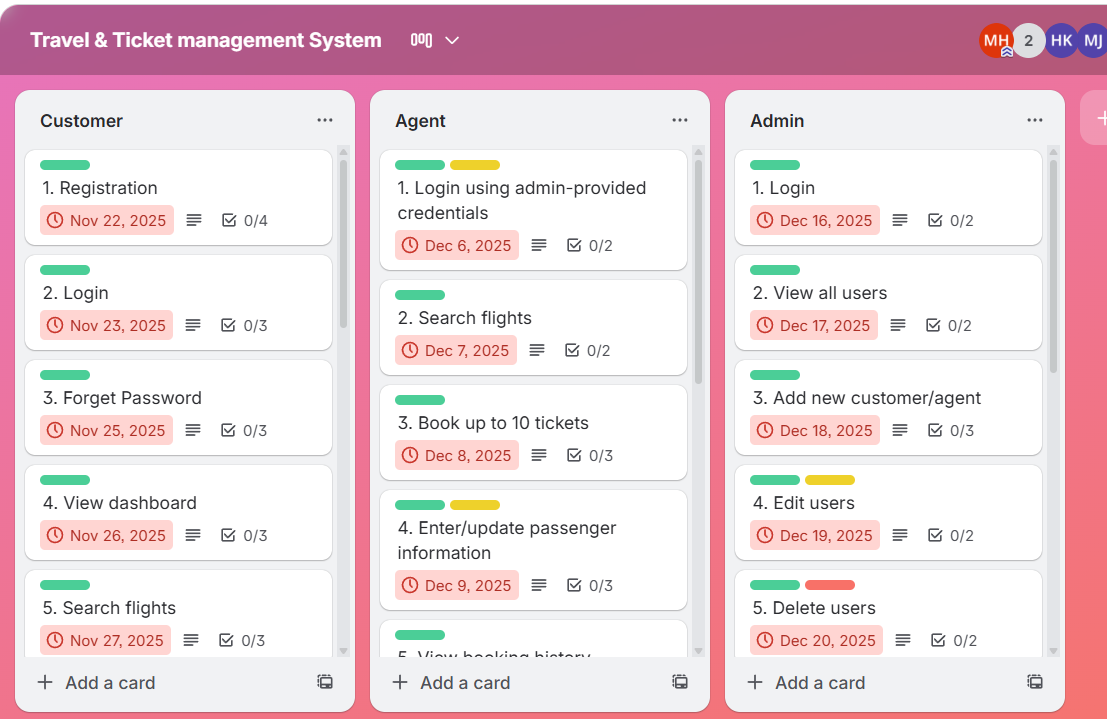
**2.2.2 Agent User Stories**

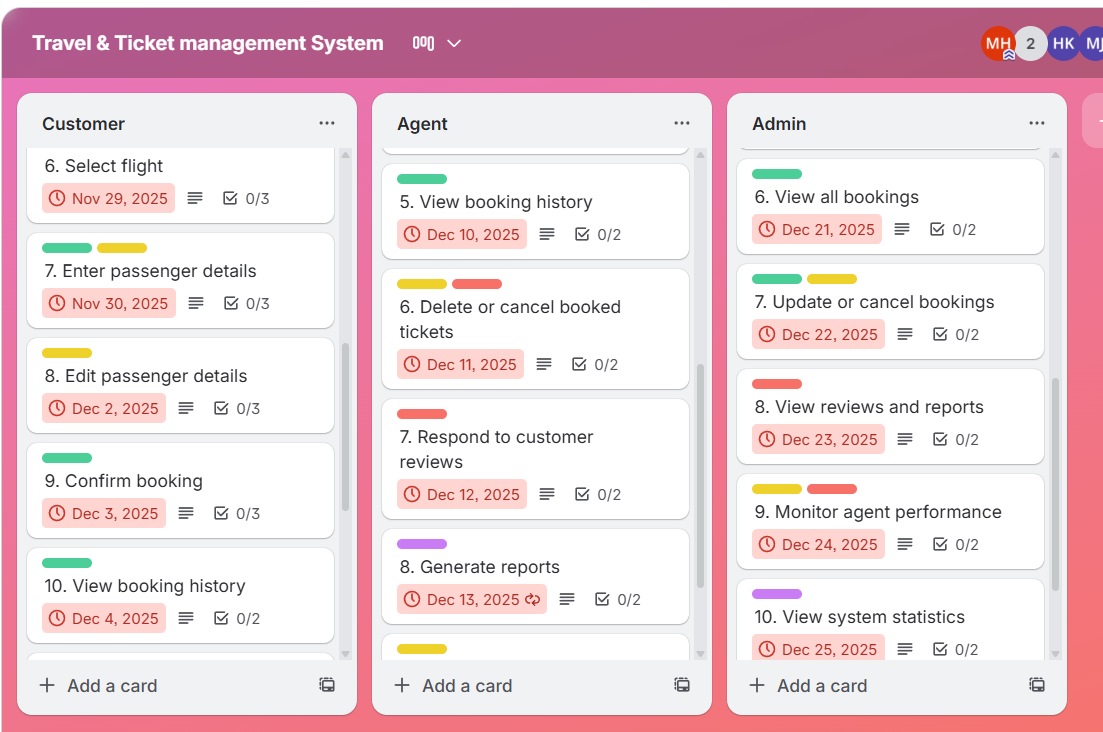
|  |  |  |  |
| --- | --- | --- | --- |
| As a User | I want to | So that | Acceptance Criteria |
| Agent | Login | Manage bookings | Secure access |
| Agent | Search flights | Assist customers | Accurate results |
| Agent | Book tickets | Confirm customer travel | Booking successful |
| Agent | Book group tickets | Handle bulk bookings | Max 10 tickets |
| Agent | Update passenger data | Avoid errors | Data saved |
| Agent | View booking history | Track bookings | History accessible |
| Agent | Respond to reviews | Improve service | Response saved |
| Agent | Logout | Secure session | Logout successful |

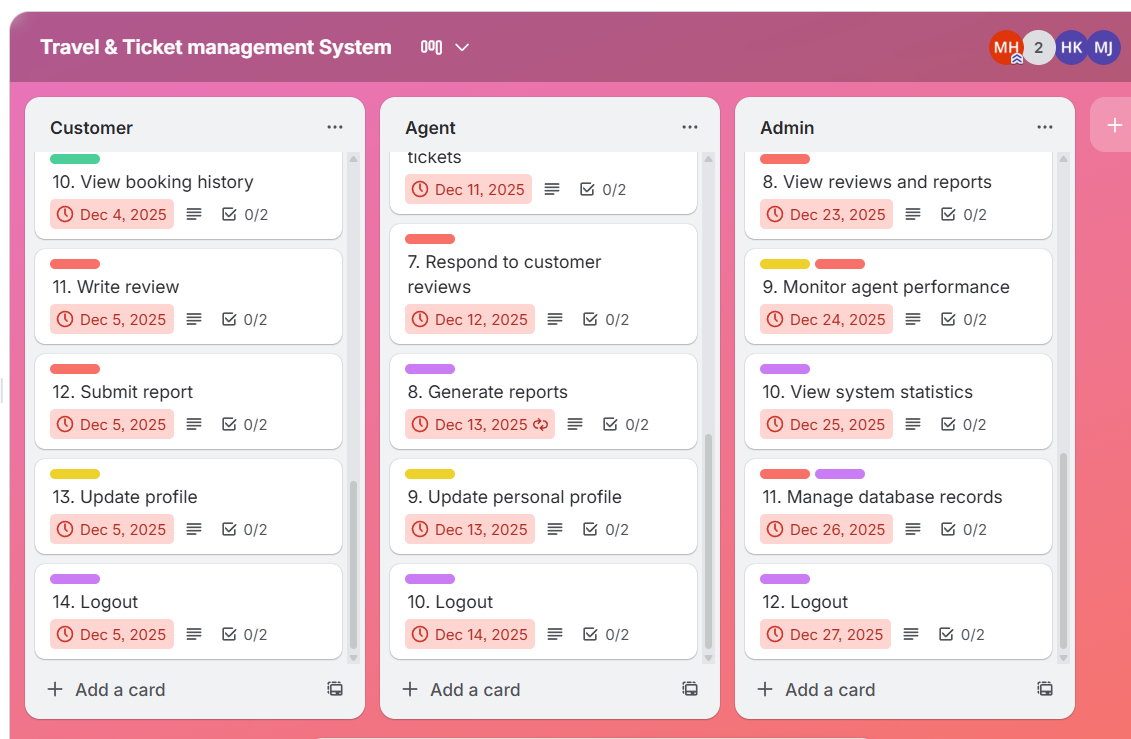
**2.2.3 Admin User Stories**

|  |  |  |  |
| --- | --- | --- | --- |
| As a User | I want to | So that | Acceptance Criteria |
| Admin | Login | Control system | Secure access |
| Admin | Manage users | System stays organized | CRUD operations |
| Admin | Manage bookings | Data remains accurate | Update/cancel allowed |
| Admin | View reports | Address issues | Reports visible |
| Admin | Monitor agents | Evaluate performance | Statistics shown |
| Admin | Control system settings | Ensure stability | Changes applied |
| Admin | Logout | Prevent misuse | Session terminated |

**Trello Scheduling**

****

****

****

**2.3 Requirements Traceability Matrix (RTM)**

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | Requirement Description | User Role | Module |
| FR-01 | User Registration | Customer | User Management |
| FR-02 | Secure Login | All | Authentication |
| FR-03 | Flight Search | Customer/Agent | Booking |
| FR-04 | Ticket Booking | Customer/Agent | Booking |
| FR-05 | Group Booking | Agent | Booking |
| FR-06 | Passenger Management | Customer/Agent | Booking |
| FR-07 | Booking History | All | Records |
| FR-08 | Review Submission | Customer | Review |
| FR-09 | Complaint Handling | Admin | Reports |
| FR-10 | User Management | Admin | Admin Panel |

**2.3.1 Functional Requirements**

**Customer Functional Requirements**

1. Register
2. Login
3. Forgot Password
4. View Dashboard
5. Search Flights
6. Select Flight
7. Enter Passenger Details
8. Edit Passenger Details
9. Confirm Booking
10. View Booking History
11. Submit Review
12. Submit Complaint
13. Update Profile
14. Logout

**Agent Functional Requirements**

1. Login
2. Search Flights
3. Book Up to 10 Tickets
4. Manage Passenger Data
5. View Booking History
6. Cancel or Modify Tickets
7. Respond to Reviews
8. Generate Reports
9. Update Profile
10. Logout

**Admin Functional Requirements**

1. Login
2. View All Users
3. Add/Edit/Delete Users
4. View All Bookings
5. Cancel or Update Bookings
6. View Reviews & Complaints
7. Monitor Agent Performance
8. View System Statistics
9. Manage Database Records
10. Logout

**2.3.2 Non-Functional Requirements**

* **Performance Requirements**

1. System should respond within **2 seconds**
2. Booking confirmation generated instantly

* **Security Requirements**

1. Encrypted passwords
2. Role-based access control
3. Secure session handling

* **Usability Requirements**

1. Simple and user-friendly interface
2. Easy navigation for all user roles

* **Reliability Requirements**

1. Data consistency ensured
2. No data loss during failure

* **Scalability Requirements**

1. Supports future features like online payment
2. Can handle increased users

* **Maintainability Requirements**

1. Modular architecture
2. Easy bug fixing and updates

# 3. SOFTWARE DESIGN

**3.1 System Design:**

**Description-**

The Use Case Diagram of the **Travel and Ticket Management System** illustrate the complete interaction between the system and its three primary actors: **Customer**, **Agent**, and **Admin**. Each actor communicates with the system based on their assigned role and access privileges. The **Customer** represents a general traveller who uses the system to perform essential ticketing activities. The customer can register for a new account, log in securely, and recover their password if forgotten. After authentication, the customer can search for available flights by selecting origin and destination locations, view ticket details such as price, time, and date, and select a preferred flight. Before confirming a booking, the customer can enter and update passenger information, ensuring accuracy of the booking data. Once the booking is finalized, the customer receives a confirmation summary and can later view all previous booking records. Additionally, customers can share their travel experience through reviews and submit complaints or reports to the admin for service improvement. They also have access to profile settings where they can update their personal information and log out for secure session termination.

The **Agent** acts as a travel agency staff member and interacts with the system using more advanced privileges than a regular customer. An agent can search for flights, view detailed ticket information, and perform group bookings of up to ten tickets in a single session. They can manage passenger records for each booking, update or correct information before confirmation, and review all booking histories they have processed. Agents are also responsible for cancelling or modifying tickets when necessary and can view customer reviews to understand service quality. In some cases, agents may respond directly to customer feedback, enhancing communication and customer satisfaction. Like customers, agents can update their own profile information and log out of the system securely.

The **Admin** is the highest-authority user and interacts with the system to maintain overall control and ensure smooth operation. The admin can view all registered users, including customers and agents, and has full capability to add, update, or delete user accounts. The admin oversees all booking operations and can update or cancel any booking if errors or conflicts arise. They have complete access to customer reviews, submitted reports, and agent performance statistics. Through the administrative dashboard, the admin can monitor system activity, view booking trends, analyse ticket sales, and ensure database accuracy using DataGrid View control. The admin also maintains system security by managing login credentials, user permissions, and ensuring secure access and logout procedures.

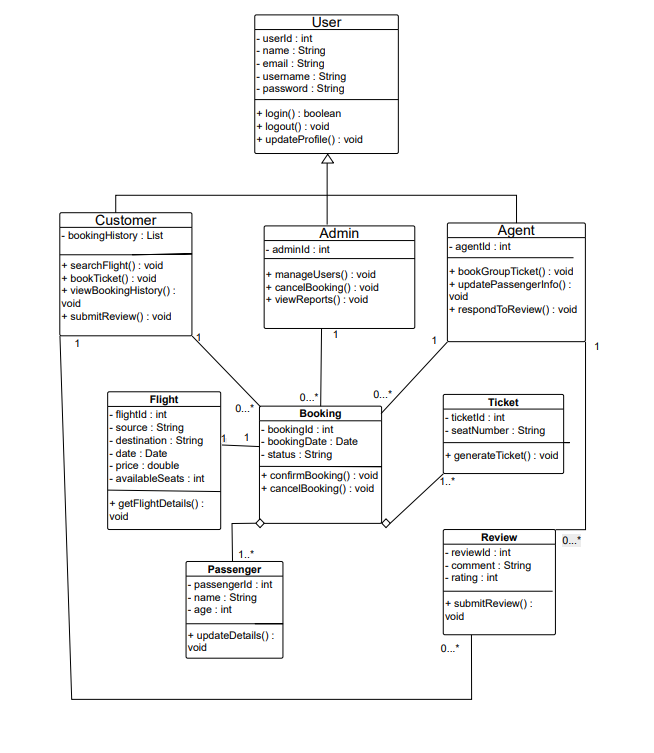
Overall, the Use Case Diagram demonstrates how the Travel and Ticket Management System support three interconnected roles, each performing different functions while maintaining a consistent workflow. By defining clear actor boundaries and responsibilities, the diagram helps visualize system behaviour, improves understanding of requirements, and ensures that every core feature such as registration, booking, managing users, and reviewing feedback is properly supported and logically connected within the system.

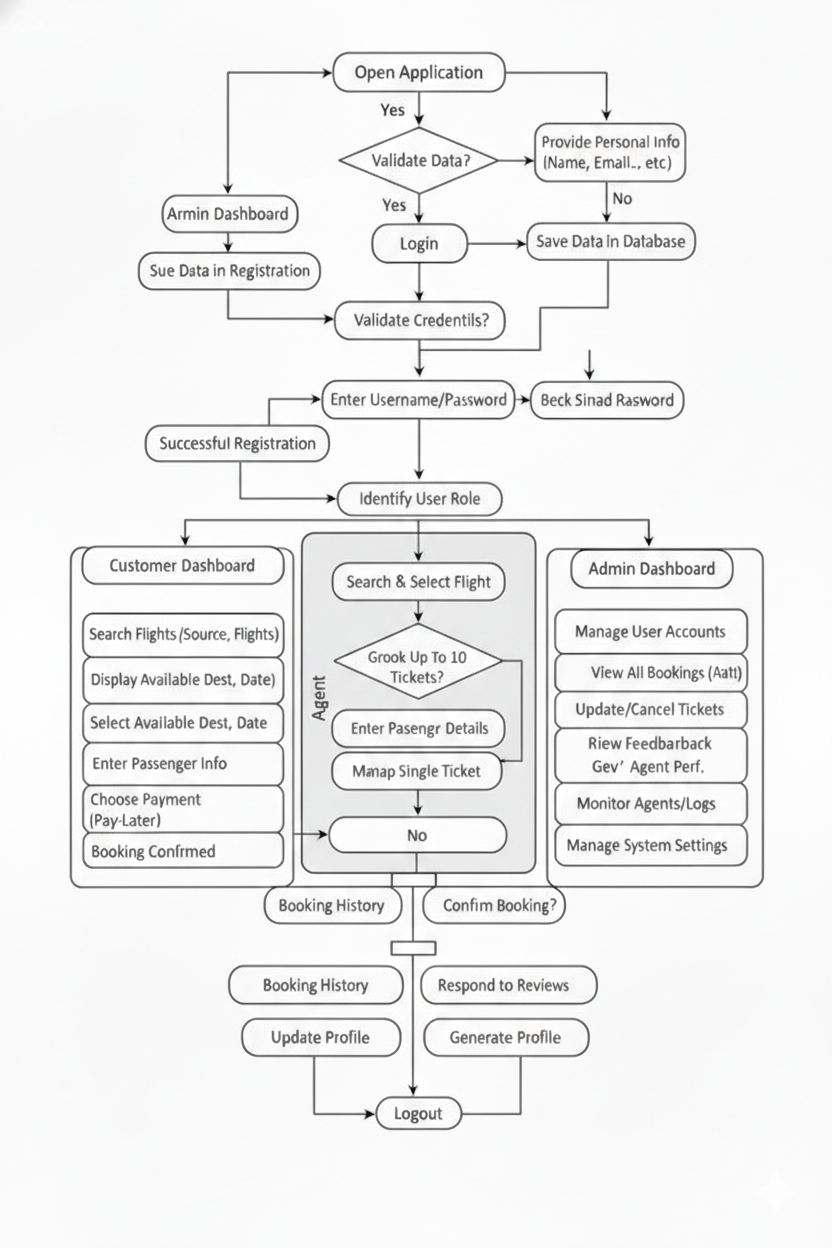
# USE CASE DIAGRAM

A black screen with white text

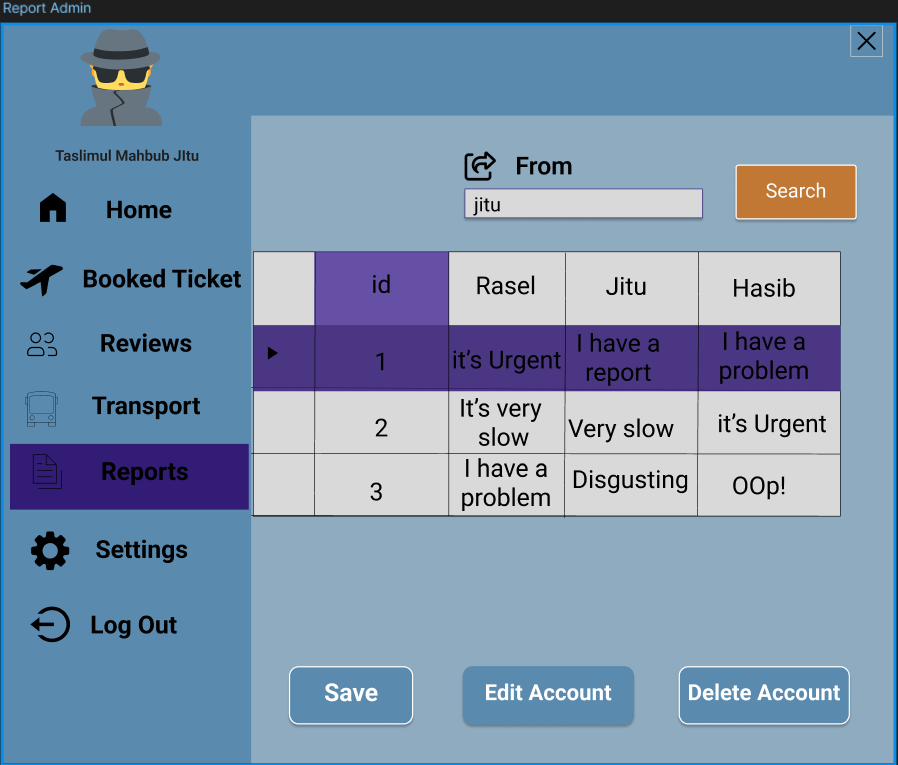
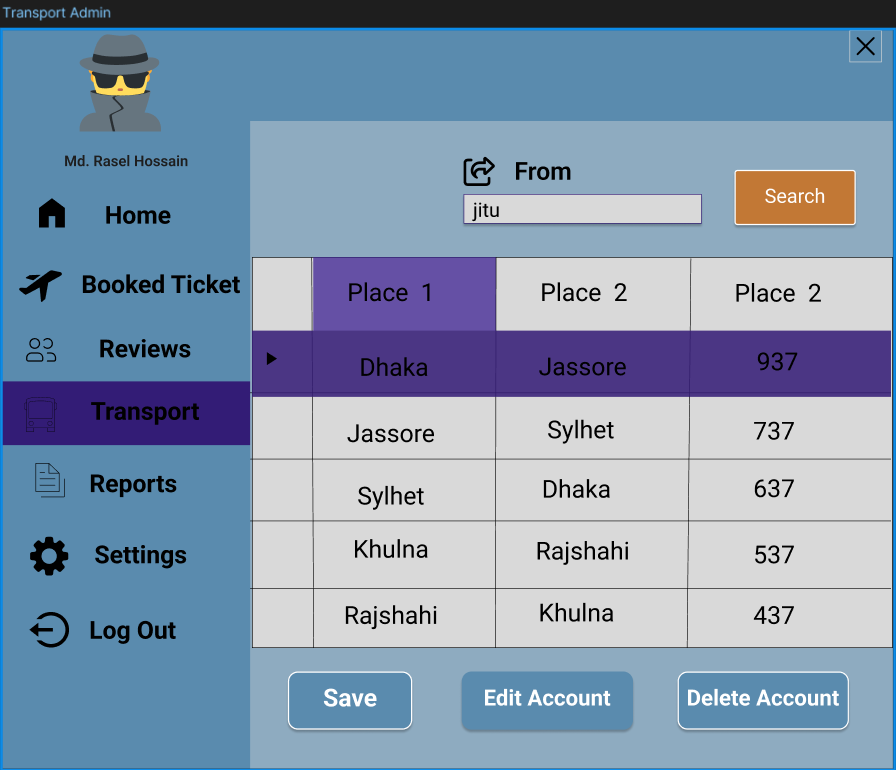
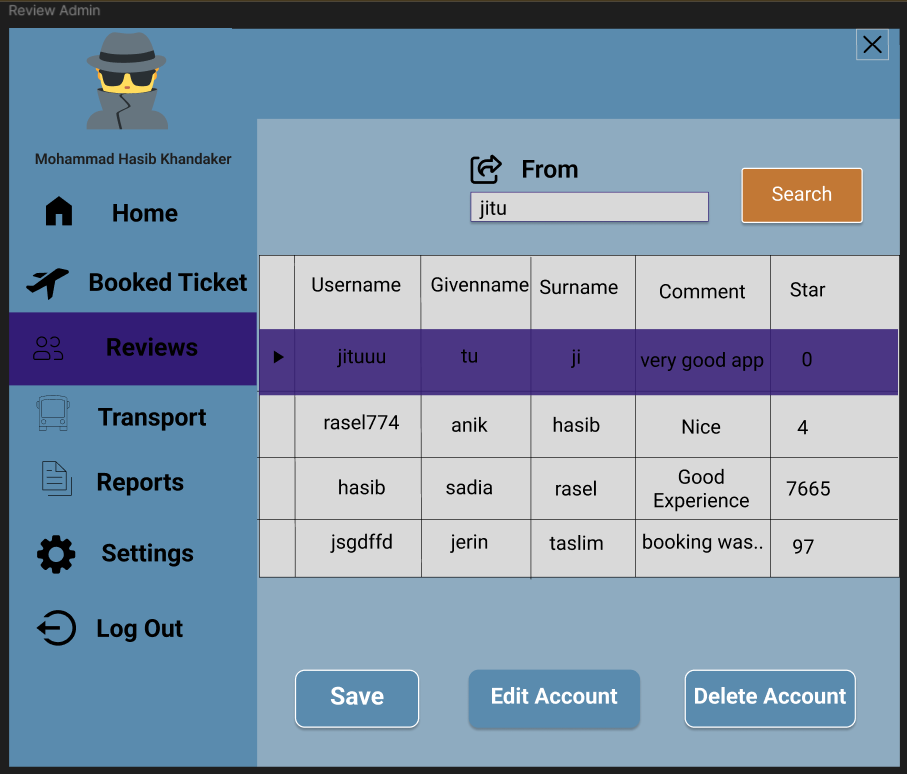
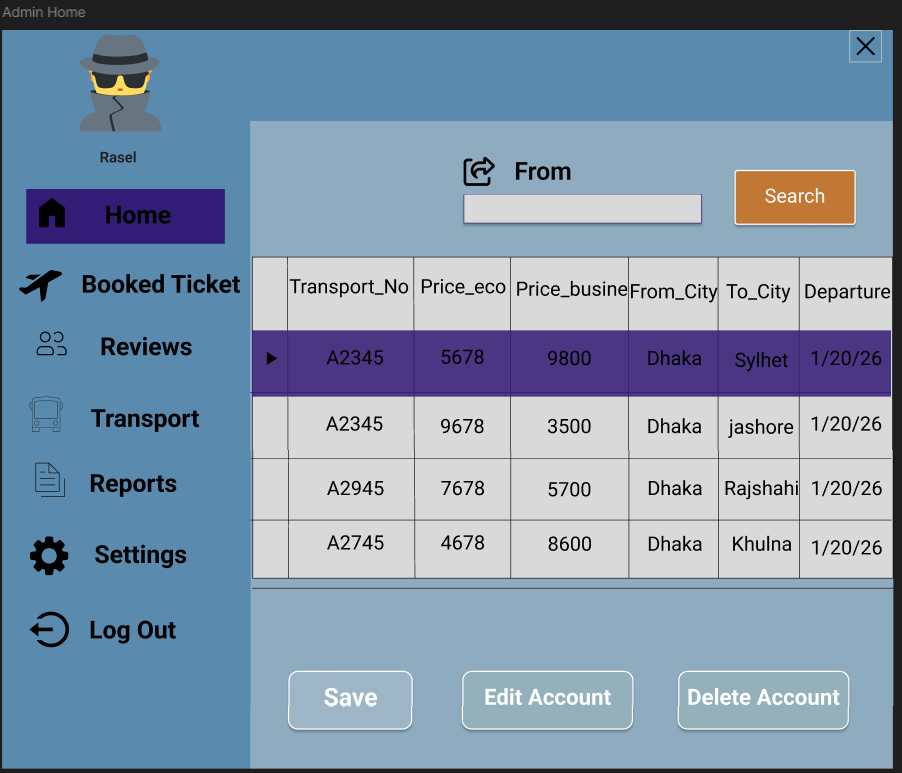
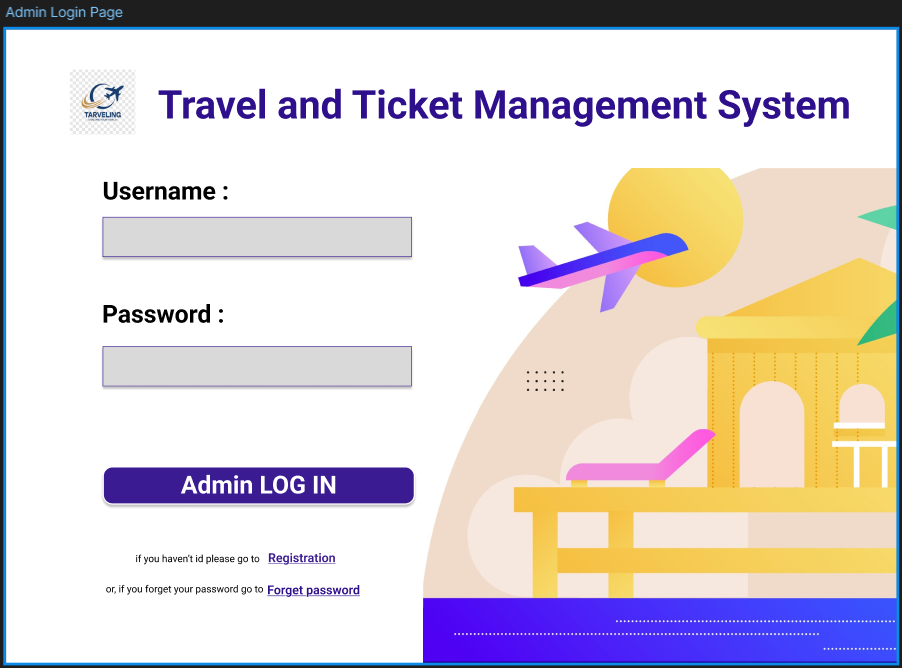
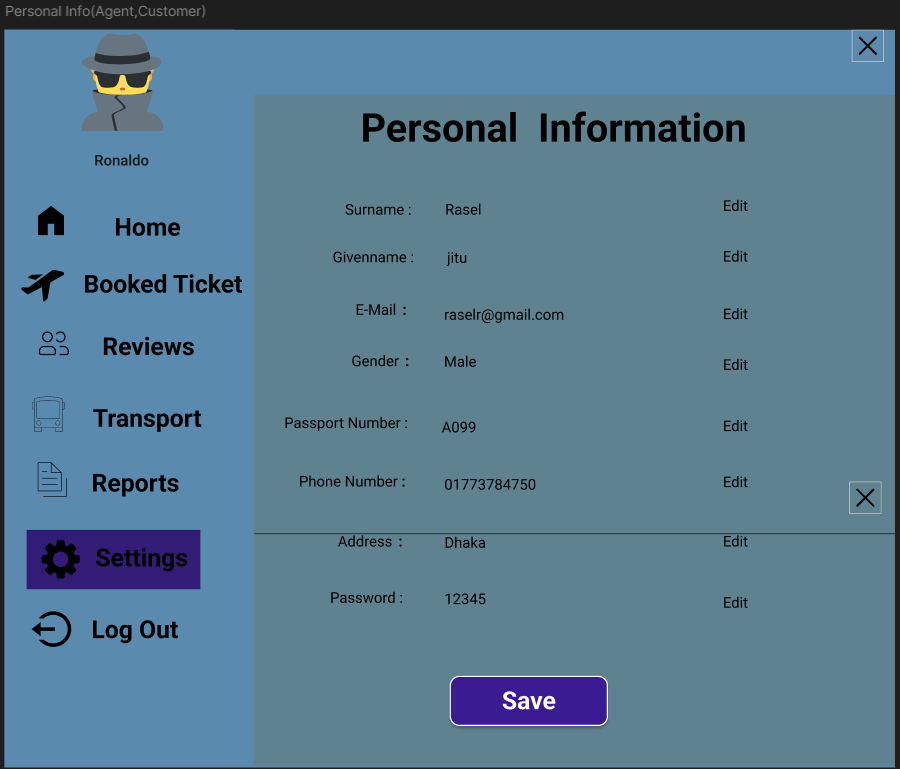
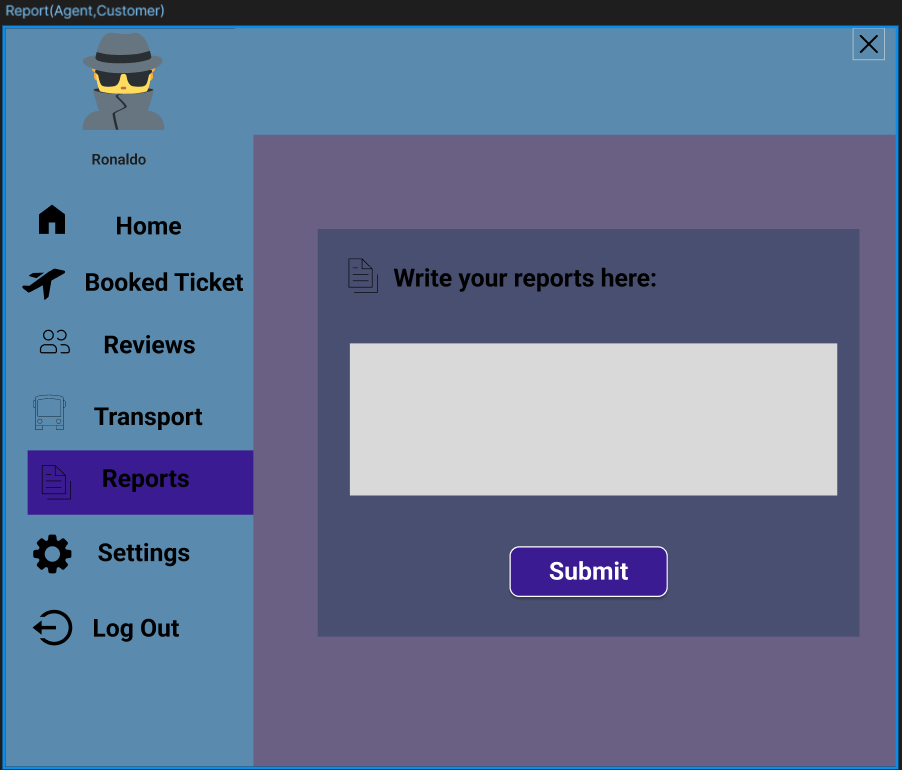
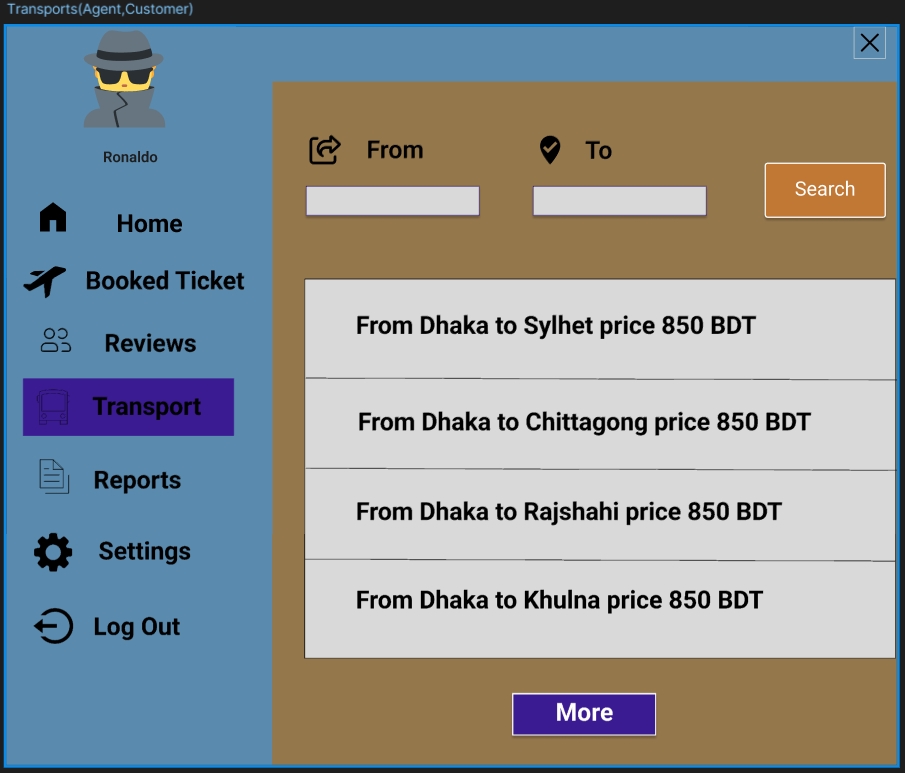
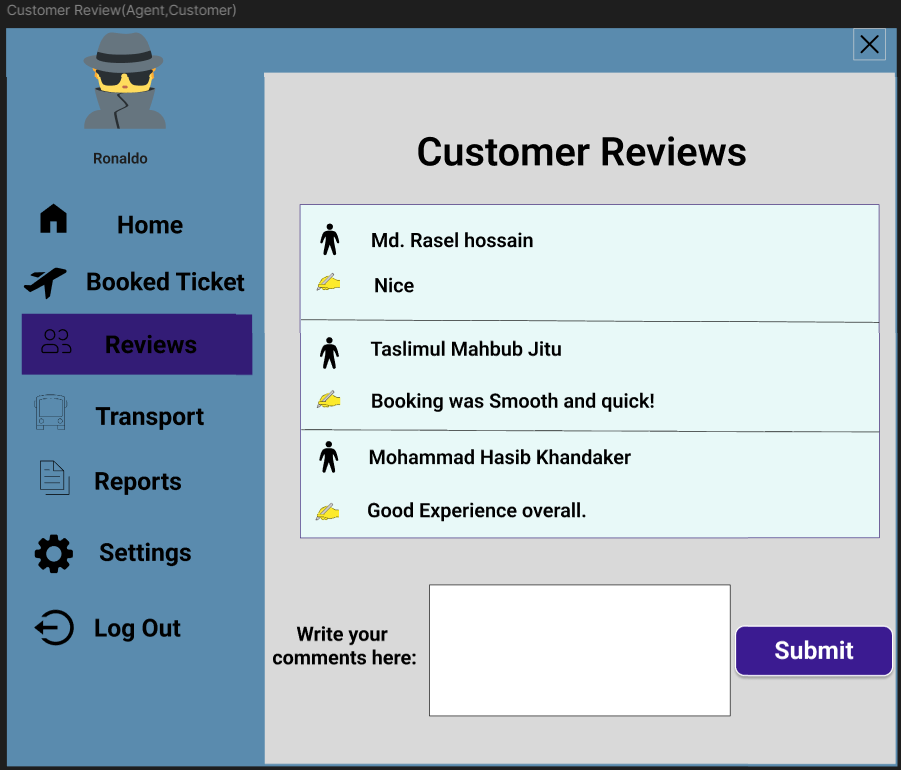
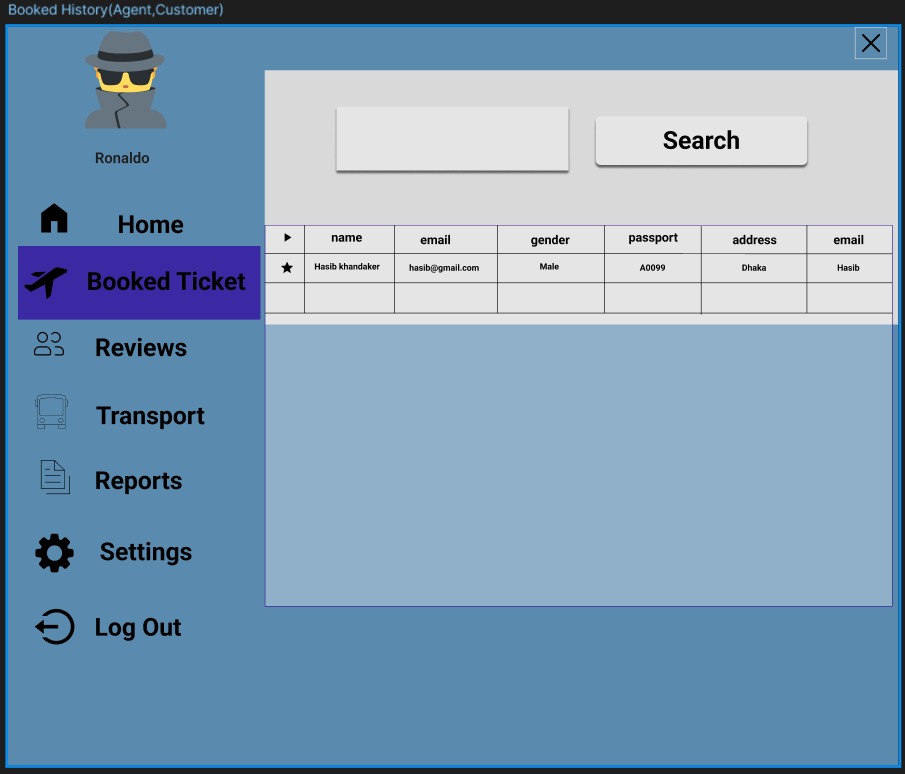
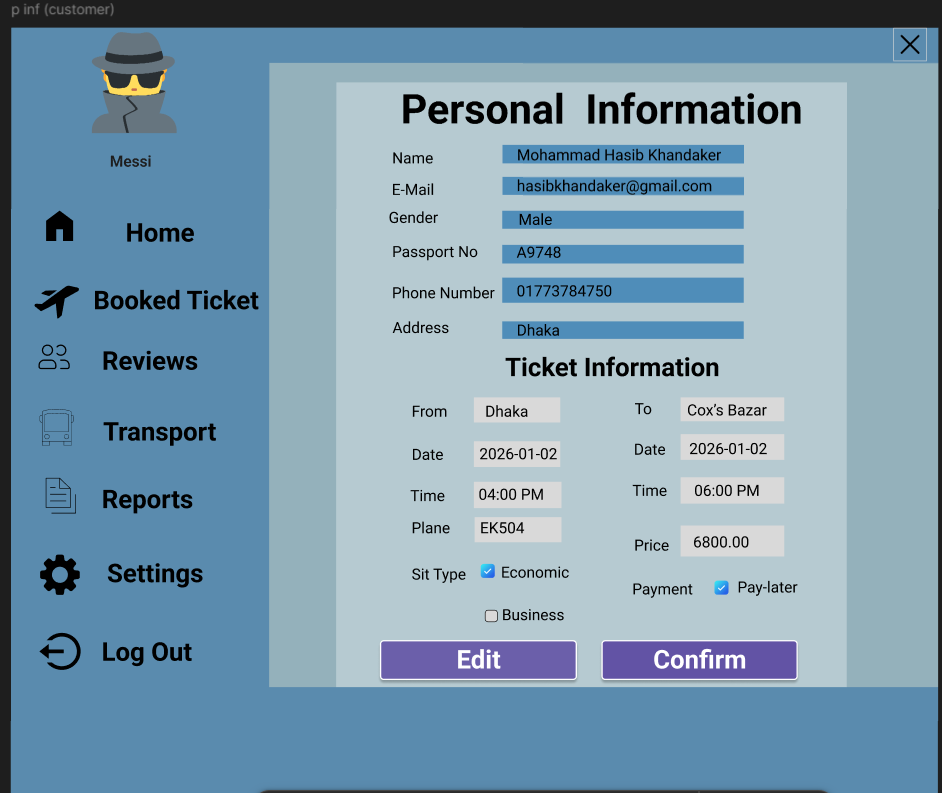
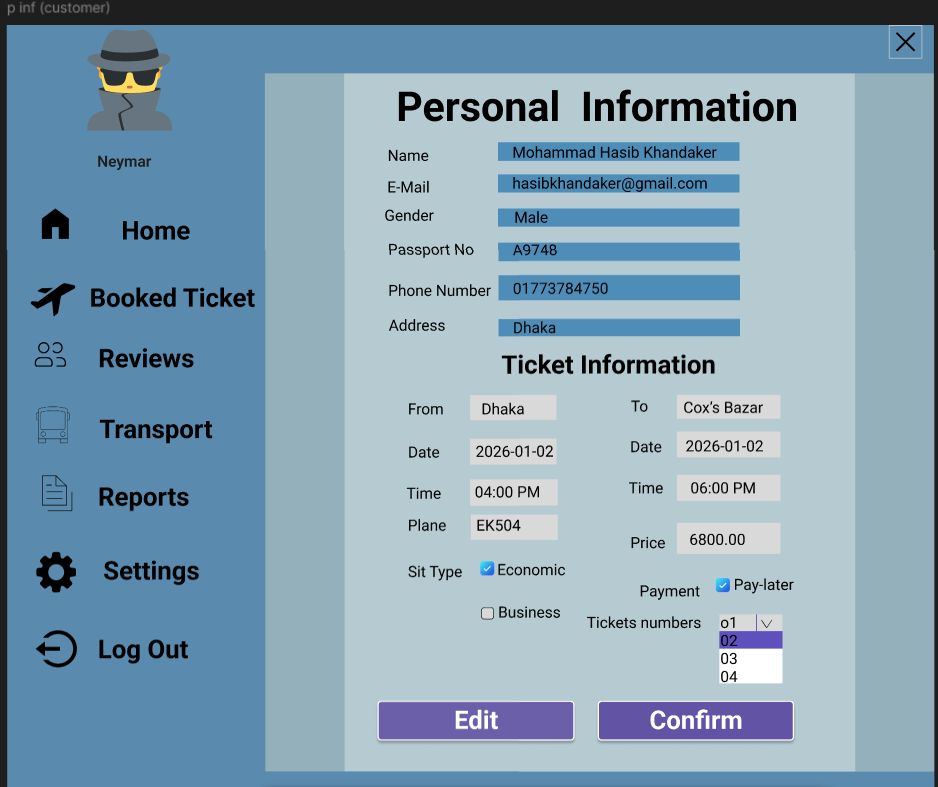
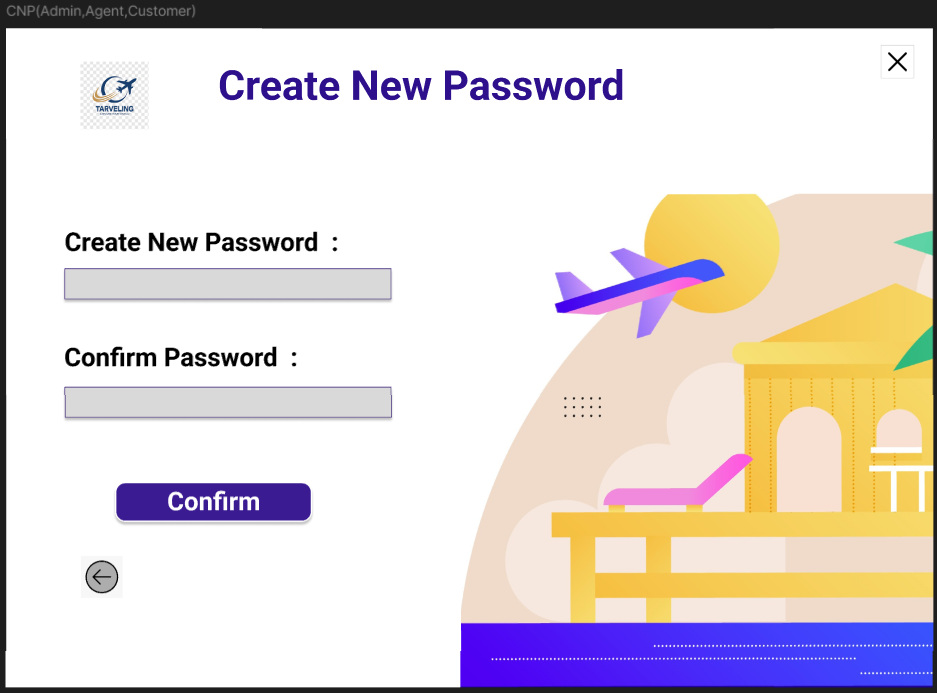
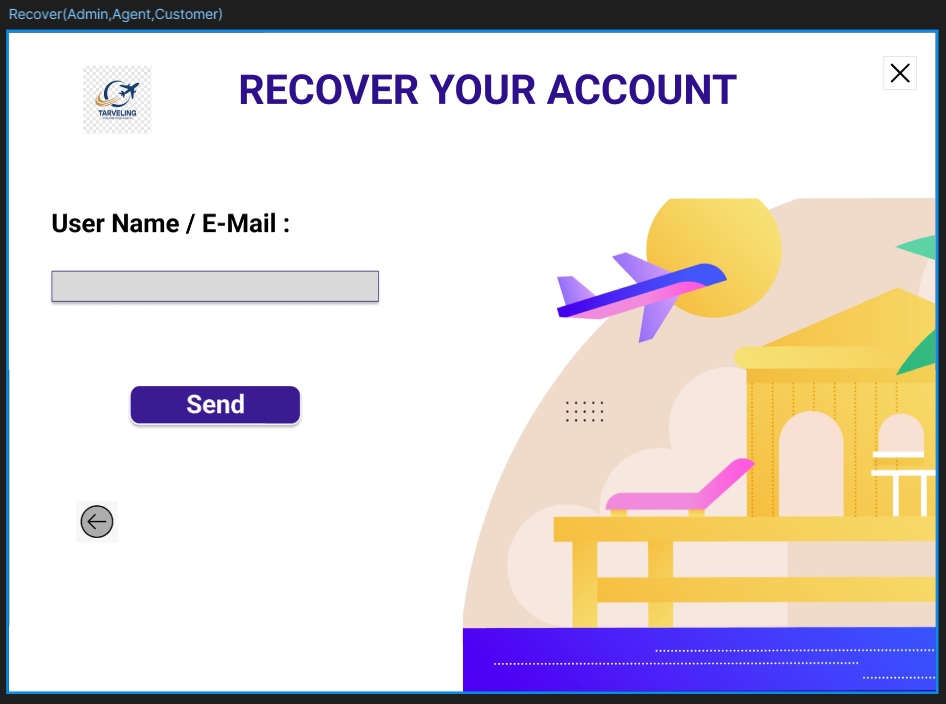
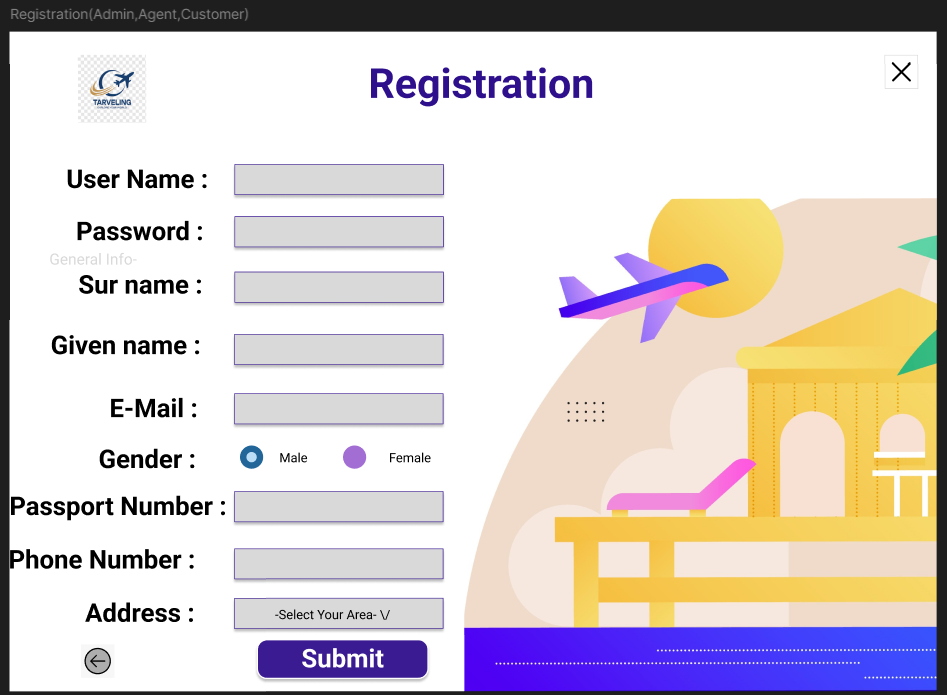
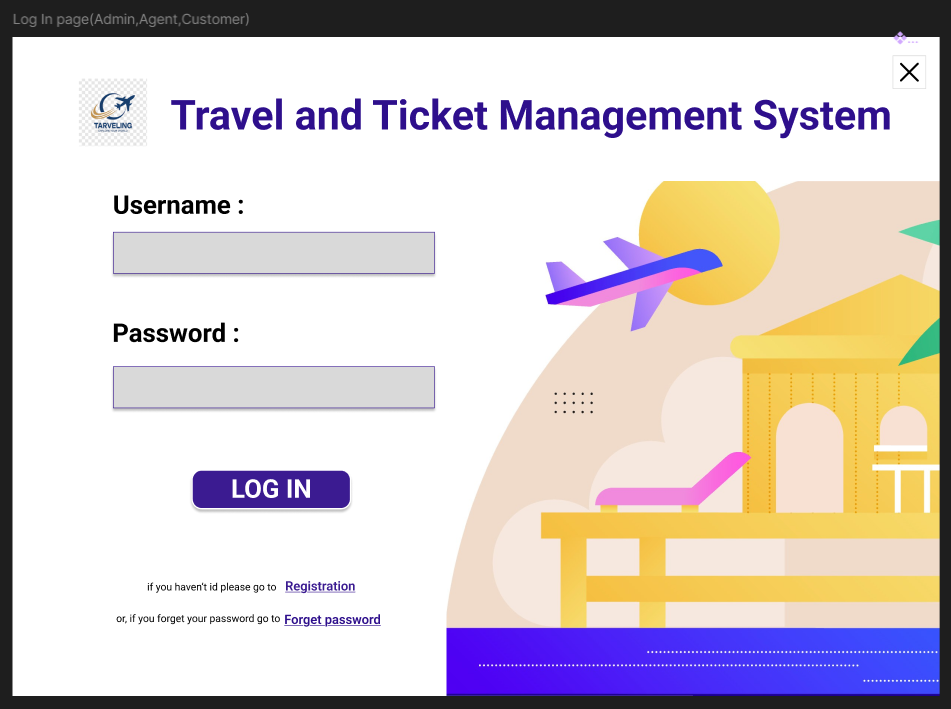
AI-generated content may be incorrect.

**Class Diagram**

****

**Activity Diagram**

## UI / Wireframe Design using Figma



# 4. GIT WORKFLOW

1. Create a central repository for the project on GitHub and set the **master (or main) branch** as the primary branch for integration.
2. Each member should clone the repository and create their own **feature branches** for assigned tasks. Work on new features or fixes within these branches.
3. Add files, stage them and commit changes with clear messages that describe the purpose of each update.
4. Push commits from the feature branches to the remote repository so other members can see progress.
5. Use **pull** to fetch and integrate changes from the remote repository into local copies, ensuring everyone stays updated.
6. Merge feature branches into the **master/main branch** only after the work is tested and reviewed, resolving any conflicts that occur.
7. Show evidence of collaboration by maintaining a clear commit history (using logs) with multiple commits, merges and contributions from all group members.
8. Keep the repository organized with a clean history that tracks the project workflow from initialization to completion.

# 5. SOFTWARE TESTING (Travel & Ticket Management System)

**5.1 Testing Methods to Use in the Testing Phase**

To ensure the **Travel & Ticket Management System** is correct, secure, and reliable, the following testing methods will be used during the testing phase:

**1) Unit Testing**

Unit testing will be performed on individual functions and methods to ensure each small unit works correctly in isolation.  
**Example:** password validation, seat limit check (agent max 10), fare calculation, input validation methods.

**2) Integration Testing**

Integration testing will verify that different modules work correctly when combined.  
**Example:** UI forms (WinForms) → Business Logic Layer → SQL Server database operations (insert booking, update seat count, show booking history).

**3) Smoke Testing**

Smoke testing (build verification testing) will be executed after each new build to confirm that the main features run without crashing.  
**Example:** app launches, login works, flight search opens, booking form loads, admin dashboard loads.

**4) Black-Box Testing**

Black-box testing will validate system behavior based on inputs and expected outputs without considering internal code structure.  
**Example:** login with correct/incorrect credentials, booking with available/unavailable seats, registration with duplicate email/username.

**5) White-Box Testing**

White-box testing will be used to test internal code logic, conditions, loops, and paths to ensure correctness.  
**Example:** validation logic branches (empty fields, invalid format), role-based checks (Customer cannot open Admin panel), booking conditions.

**6) Higher Order Testing**

Higher order testing will include system-level evaluations such as **System Testing** and **Acceptance Testing** to confirm the complete system meets requirements.  
**Example:** end-to-end workflow: Register → Login → Search → Book → Booking Summary → History → Logout.

**7) Regression Testing**

Regression testing will be conducted after bug fixes or feature updates to ensure existing modules still work correctly.  
**Example:** after improving booking module, re-test login, search, history, admin CRUD.

**8) OO Testing (Object-Oriented Testing)**

Object-oriented testing will focus on class behavior, object interaction, inheritance, and method correctness in the OOP architecture.  
**Example:** testing classes like User, Customer, Agent, Admin, Booking, Flight, Review; verifying object interactions and data passing.

**9) Validation and Verification (V & V)**

* **Verification:** Ensures we built the system correctly according to SRS (checking design, code, and test results).
* **Validation:** Ensures we built the correct system that satisfies user needs (checking usability and real workflow).  
  **Example:** verifying role-based access matches SRS; validating that users can easily complete booking and see confirmation/history.

**TEST CASE – TC\_01**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Md. Rasel Hossain |
| **Test Case ID:** TC\_01 | **Test Designed date:** 02/07/2025 |
| **Test Priority (Low, Medium, High):** High | **Test Executed by:** Md. Rasel Hossain |
| **Module Name:** Login Session | **Test Execution date:** 22/08/2025 |
| **Test Title:** Verify login with valid username and password |  |
| **Description:** Test customer login page |  |
| **Precondition:** The user has a valid username and password |  |
| **Dependence:** **Database connection available** |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Open application  2. Enter username  3. Enter password 4. Click Login | Username: user01 Password: User@123 | User should successfully log in | As expected | Pass | | |

**TEST CASE – TC\_02**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Hasib |
| **Test Case ID:** TC\_02 | **Test Designed date:** 02/07/2025 |
| **Test Priority (Low, Medium, High):** High | **Test Executed by:** Hasib |
| **Module Name:** Login Session | **Test Execution date:** 22/08/2025 |
| **Test Title:** Verify login fails with invalid password |  |
| **Description:** Test login with wrong password |  |
| **Precondition:** User exists in database |  |
| **Dependencies:** Authentication module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Open login page  2. Enter username  3. Enter wrong password  4. Click Login | Username: user01 Password: wrong123 | Login denied with error message | As expected | Pass | | |

**TEST CASE – TC\_03**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Hasib |
| **Test Case ID:** TC\_03 | **Test Designed date:** 02/07/2025 |
| **Test Priority (Low, Medium, High):** Medium | **Test Executed by:** Hasib |
| **Module Name:** User Registration | **Test Execution date:** 22/08/2025 |
| **Test Title:** Verify new user registration with valid details |  |
| **Description:** Test customer registration |  |
| **Precondition:** Email and username must be unique |  |
| **Dependencies:** Database validation |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Open registration form  2. Enter details  3. Click Register | Email: [test@mail.com](mailto:test@mail.com) Username: test01 Password: Test@123 | Registration successful | As expected | Pass | | |

**TEST CASE – TC\_04**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Hasib |
| **Test Case ID:** TC\_04 | **Test Designed date:** 02/07/2025 |
| **Test Priority (Low, Medium, High):** High | **Test Executed by:** Hasib |
| **Module Name:** User Registration | **Test Execution date:** 22/08/2025 |
| **Test Title:** Verify duplicate registration is not allowed |  |
| **Description:** Prevent duplicate username/email |  |
| **Precondition:** User already registered |  |
| **Dependencies:** Database unique constraint |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Open registration  2. Enter duplicate email  3. Submit | Email: [test@mail.com](mailto:test@mail.com) Username: test01 | Error message displayed | As expected | Pass | | |

**TEST CASE – TC\_05**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Hasib |
| **Test Case ID:** TC\_05 | **Test Designed date:** 10/07/2025 |
| **Test Priority (Low, Medium, High):** Medium | **Test Executed by:** Hasib |
| **Module Name:** Forgot Password | **Test Execution date:** 22/08/2025 |
| **Test Title:** Verify password recovery using username/email |  |
| **Description:** Test forgot password functionality |  |
| **Precondition:** User exists |  |
| **Dependencies:** Password recovery module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Click Forgot Password  2. Enter username  3. Submit | Username: test01 | Password recovery successful | As expected | Pass | | |

**TEST CASE – TC\_06**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Hasib |
| **Test Case ID:** TC\_06 | **Test Designed date:** 10/07/2025 |
| **Test Priority (Low, Medium, High):** Medium | **Test Executed by:** Hasib |
| **Module Name:** Flight Search | **Test Execution date:** 22/08/2025 |
| **Test Title:** Verify flight search by source and destination |  |
| **Description:** Search available flights |  |
| **Precondition:** User logged in |  |
| **Dependencies:** Flight database |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Login  2. Enter From & To  3. Click Search | From: Dhaka To: Chittagong | Flight list displayed | As expected | Pass | | |

**TEST CASE – TC\_07**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Jitu |
| **Test Case ID:** TC\_07 | **Test Designed date:** 10/07/2025 |
| **Test Priority (Low, Medium, High):** Low | **Test Executed by:** Jitu |
| **Module Name:** Flight Search | **Test Execution date:** 22/08/2025 |
| **Test Title:** Verify no flight available message |  |
| **Description:** Handle empty search result |  |
| **Precondition:** User logged in |  |
| **Dependencies:** Search module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Enter invalid route  2. Click Search | From: CityA To: CityB | “No flights found” message | As expected | Pass | | |

**TEST CASE – TC\_08**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Jitu |
| **Test Case ID:** TC\_08 | **Test Designed date:** 14/07/2025 |
| **Test Priority (Low, Medium, High):** High | **Test Executed by:** Jitu |
| **Module Name:** Ticket Booking | **Test Execution date:** 30/08/2025 |
| **Test Title:** Verify ticket booking with available seat |  |
| **Description:** Book ticket successfully |  |
| **Precondition:** Seats available |  |
| **Dependencies:** Booking module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Select flight  2. Enter passenger info  3. Confirm | Passenger: Rahim Seat: 1 | Booking confirmed | As expected | Pass | | |

**TEST CASE – TC\_09**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Jitu |
| **Test Case ID:** TC\_09 | **Test Designed date:** 14/07/2025 |
| **Test Priority (Low, Medium, High):** High | **Test Executed by:** Jitu |
| **Module Name:** Ticket Booking | **Test Execution date:** 30/08/2025 |
| **Test Title:** Verify booking blocked when no seats available |  |
| **Description:** Prevent overbooking |  |
| **Precondition:** Seats = 0 |  |
| **Dependencies:** Seat validation |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Select full flight  2. Book ticket | Seats: 0 | Booking denied | As expected | Pass | | |

**TEST CASE – TC\_10**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by** Jitu |
| **Test Case ID:** TC\_10 | **Test Designed date:** 14/07/2025 |
| **Test Priority (Low, Medium, High):** Medium | **Test Executed by:** Jitu |
| **Module Name:** Booking History | **Test Execution date:** 30/08/2025 |
| **Test Title:** Verify booking history display |  |
| **Description:** View previous bookings |  |
| **Precondition:** Booking exists |  |
| **Dependencies:** History module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Login  2. Open Booking History | User: user01 | Booking history displayed | As expected | Pass | | |

**TEST CASE – TC\_11**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Jitu |
| **Test Case ID:** TC\_11 | **Test Designed date:** 16/07/2025 |
| **Test Priority (Low, Medium, High):** High | **Test Executed by:** Jitu |
| **Module Name:** Agent Login Session | **Test Execution date:** 02/09/2025 |
| **Test Title:** Verify agent login with valid credentials |  |
| **Description:** Test agent login functionality |  |
| **Precondition:** Agent account exists and is active |  |
| **Dependencies:** Database connection available |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Open application  2. Enter agent username  3. Enter password  4. Click Login | Username: agent01 Password: Agent@123 | Agent dashboard displayed successfully | As expected | Pass | | |

**TEST CASE – TC\_12**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Jitu |
| **Test Case ID:** TC\_12 | **Test Designed date:** 16/07/2025 |
| **Test Priority (Low, Medium, High):** High | **Test Executed by:** Jitu |
| **Module Name:** Agent Ticket Booking | **Test Execution date:** 02/09/2025 |
| **Test Title:** Verify agent can book up to 10 tickets |  |
| **Description:** Test agent group booking limit |  |
| **Precondition:** Agent logged in and seats available |  |
| **Dependencies:** Booking module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Login as agent 2. Select flight 3. Enter 10 passengers 4. Confirm booking | Seats: 10 | Booking completed successfully | As expected | Pass | | |

**TEST CASE – TC\_13**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Jitu |
| **Test Case ID:** TC\_13 | **Test Designed date:** 16/07/2025 |
| **Test Priority (Low, Medium, High):** **High** | **Test Executed by:** Jitu |
| **Module Name:** Agent Ticket Booking | **Test Execution date:** 04/09/2025 |
| **Test Title:** Verify agent booking blocked for more than 10 tickets |  |
| **Description:** Prevent agent from booking more than 10 tickets |  |
| **Precondition:** Agent logged in |  |
| **Dependencies:** Booking validation logic |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Select flight 2. Enter 11 passengers 3. Confirm booking | Seats: 11 | Error message shown and booking denied | As expected | Pass | | |

**TEST CASE – TC\_14**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Jitu |
| **Test Case ID:** TC\_14 | **Test Designed date:** 18/07/2025 |
| **Test Priority (Low, Medium, High):** **Medium** | **Test Executed by:** Jitu |
| **Module Name:** Review System | **Test Execution date:** 12/09/2025 |
| **Test Title:** Verify customer can submit a travel review |  |
| **Description:** Test review submission functionality |  |
| **Precondition:** Customer logged in |  |
| **Dependencies:** Review module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Login as customer 2. Open review form 3. Write review 4. Submit | Review: “Good service” | Review saved successfully | As expected | Pass | | |

**TEST CASE – TC\_15**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Rasel |
| **Test Case ID:** TC\_15 | **Test Designed date:** 02/07/2025 |
| **Test Priority (Low, Medium, High):** **Medium** | **Test Executed by:** Rasel |
| **Module Name:** Complaint System | **Test Execution date:** 22/08/2025 |
| **Test Title:** Verify customer can submit complaint/report |  |
| **Description:** Test complaint submission |  |
| **Precondition:** Customer logged in |  |
| **Dependencies:** Report module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Open complaint form 2. Enter issue 3. Submit | Issue: “Flight delay” | Complaint recorded successfully | As expected | Pass | | |

**TEST CASE – TC\_16**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Rasel |
| **Test Case ID:** TC\_16 | **Test Designed date:** 02/07/2025 |
| **Test Priority (Low, Medium, High):** **High** | **Test Executed by:** Rasel |
| **Module Name:** Admin Login Session | **Test Execution date:** 22/08/2025 |
| **Test Title:** Verify admin login with valid credentials |  |
| **Description:** Test admin login functionality |  |
| **Precondition:** Admin account exists |  |
| **Dependencies:** Authentication module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Open login page 2. Enter admin username 3. Enter password 4. Click Login | Username: admin Password: Admin@123 | Admin dashboard opens | As expected | Pass | | |

**TEST CASE – TC\_17**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Rasel |
| **Test Case ID:** TC\_17 | **Test Designed date:** 02/07/2025 |
| **Test Priority (Low, Medium, High):** **High** | **Test Executed by:** Rasel |
| **Module Name:** Admin User Management | **Test Execution date:** 22/08/2025 |
| **Test Title:** Verify admin can add new agent |  |
| **Description:** Test agent creation by admin |  |
| **Precondition:** Admin logged in |  |
| **Dependencies:** User management module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Open user management 2. Click Add Agent 3. Enter details 4. Save | Username: agent02 Email: [agent02@mail.com](mailto:agent02@mail.com) | Agent account created | As expected | Pass | | |

**TEST CASE – TC\_18**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Rasel |
| **Test Case ID:** TC\_18 | **Test Designed date:** 14/07/2025 |
| **Test Priority (Low, Medium, High):** **Medium** | **Test Executed by:** Md. Rasel Hossain |
| **Module Name:** Admin User Management | **Test Execution date:** 12/09/2025 |
| **Test Title:** Verify admin can edit user information |  |
| **Description:** Test updating user data |  |
| **Precondition:** User exists |  |
| **Dependencies:** Database update |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Select user 2. Edit email 3. Save | Email: [new@mail.com](mailto:new@mail.com) | User information updated | As expected | Pass | | |

**TEST CASE – TC\_19**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Rasel |
| **Test Case ID:** TC\_19 | **Test Designed date:** 16/07/2025 |
| **Test Priority (Low, Medium, High):** **High** | **Test Executed by:** Rasel |
| **Module Name:** Admin Booking Management | **Test Execution date:** 12/09/2025 |
| **Test Title:** Verify admin can cancel booking |  |
| **Description:** Cancel any booking by admin |  |
| **Precondition:** Booking exists |  |
| **Dependencies:** Booking management module |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. View bookings 2. Select booking 3. Click Cancel | Booking ID: BK101 | Booking status changed to cancelled | As expected | Pass | | |

**TEST CASE – TC\_20**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Md. Rasel Hossain |
| **Test Case ID:** TC\_20 | **Test Designed date:** 22/07/2025 |
| **Test Priority (Low, Medium, High):** **Medium** | **Test Executed by:** Rasel |
| **Module Name:** Role-Based Access Control | **Test Execution date:** 12/09/2025 |
| **Test Title:** Verify customer cannot access admin panel |  |
| **Description:** Test role-based access restriction |  |
| **Precondition:** Customer logged in |  |
| **Dependencies:** Access control logic |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Login as customer 2. Try to open admin panel | Customer role | Access denied message shown | As expected | Pass | | |

**TEST CASE – TC\_21**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Rasel |
| **Test Case ID:** TC\_21 | **Test Designed date:** 22/07/2025 |
| **Test Priority (Low, Medium, High):** **Medium** | **Test Executed by:** Rasel |
| **Module Name:** Database Validation | **Test Execution date:** 23/09/2025 |
| **Test Title:** Verify booking data stored in database |  |
| **Description:** Check booking record persistence |  |
| **Precondition:** Booking completed |  |
| **Dependencies:** SQL Server database |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Complete booking 2. Check database record | Booking details | Booking data saved correctly | As expected | Pass | | |

**TEST CASE – TC\_22**

|  |  |
| --- | --- |
| **Project Name:** Travel & Ticket Management System | **Test Designed by:** Md. Rasel Hossain |
| **Test Case ID:** TC\_22 | **Test Designed date:** 25/07/2025 |
| **Test Priority (Low, Medium, High):** **Low** | **Test Executed by:** Md. Rasel Hossain |
| **Module Name:** Logout Session | **Test Execution date:** 19/09/2025 |
| **Test Title:** Verify logout ends user session |  |
| **Description:** Test logout functionality |  |
| **Precondition:** User logged in |  |
| **Dependencies:** Session management |  |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status** | | 1. Click Logout 2. Try to access dashboard | N/A | Redirected to login page | As expected | Pass | | |

# 6. CONCLUSION

* Write a conclusion within 180 to 200 words.

**Instructions:**

* Minimum of 3 members and Maximum of 5 members per group.
* Font: Times New Roman ; Size: 12; Justify the para [Ctrl + J].
* Delete the highlighted part after completing this project report.
* The completed report should be within the range of 40 to 50 pages.
* **Submission:** Bring a hard copy of this report [per group] on the project evaluation day. Also, you will need to upload the soft copy later.