



East West University

Project Report

ON

Airline Management System (*SkyHigh Airlines*)

Submitted in fulfillment of the requirement for the Semester Spring 2025

COURSE CODE: CSE412; **SECTION:** 02

Of

BACHELOR OF SCIENCE

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

1. Seaum Insaniat Swapnil (2021-2-60-016)
2. Jerin Anan Proma (2022-1-60-132)
3. Shanta Islam (2022-1-60-288)
4. Nusrat Jahan Oishi (2022-2-60-033)




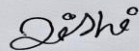
Submitted to:

Yasin Sazid

Lecturer

Dept. of Computer Science & Engineering

- **Declaration Page: Signature of all team members**

Student ID	Name	Signature
2021-2-60-016	Seaum Insaniat Swapnil	
2022-1-60-132	Jerin Anan Proma	
2022-1-60-288	Shanta Islam	
2022-2-60-033	Nusrat Jahan Oishi	

● Table of Contents

Chapter	Title	Page No.
Chapter 1: Introduction	Project Overview	1
	Objectives and Scope	2
	Stakeholders	3
	Technology Stack	4
Chapter 2: Software Requirements Specification & Analysis	Stakeholder Needs & Analysis	5
	List of Requirements	5
	Quality Function Deployment	8
	Requirements Modeling	10
Chapter 3: Software Design	Architectural Design	19
	Component-Level Design	21
	User Interface Design	34
	Link of Figma	38
Chapter 4: Implementation	Code Structure Overview	39
	GitHub Repository URL	40

Chapter	Title	Page No.
Chapter 5: Software Testing	White Box Testing	41
	Black Box Testing	46
	Bug Detection and Solution	48
Chapter 6: Deployment	Deployment Platform	55
	Deployment Process	55
	Website URL	56
Chapter 7: Conclusion	Learnings	57
	Limitations	58
	Future Plan	58

● List of Figures

Figure No	Figure Name	Page No
2.1	<i>Use Case Diagram for Passenger</i>	10
2.2	<i>Use Case Diagram for Admin</i>	11
2.3	<i>Use Case Diagram for Customer Support & Feedback</i>	11
2.4	<i>Use Case Diagram for Payment Processing</i>	12
2.5	<i>Activity Diagram for Login & Register</i>	13
2.6	<i>Activity Diagram for Flight booking</i>	14
2.7	<i>Activity Diagram for Payment</i>	15
2.4	<i>UI Sketch of HOMe Page</i>	16
3.1	<i>Architectural context diagram</i>	19
3.2	<i>Top-Level Component Diagram</i>	19
3.3	<i>Instantiation of Each Component with Component Elaboration</i>	20
3.4	<i>Authentication Component</i>	21
3.5	<i>Search & Recommendation Component</i>	22
3.6	<i>Booking Component</i>	23
3.7	<i>Payment Component</i>	24
3.8	<i>About Component</i>	25
3.9	<i>User Dashboard Component</i>	26
3.10	<i>Flight Management Component</i>	27
3.11	<i>Booking Management Component</i>	28
3.12	<i>Feedback Management Component</i>	29

Figure No	Figure Name	Page No
3.13	<i>Analytical Dashboard Component</i>	30
3.14	<i>Class Diagram</i>	31
3.15	<i>ER Diagram 1</i>	32
3.16	<i>ER Diagram 2</i>	33
3.17	<i>Navigation FLOW</i>	37-38
4.1	<i>Code Structure Overview</i>	39-40
5.1	<i>Control flow graph of registration class</i>	41
5.2	<i>Control flow graph of login class</i>	42
5.3	<i>Control flow graph of flight_booking class</i>	44
5.4	<i>Bug-1</i>	48
5.5	<i>Retesting-1</i>	49
5.6	<i>Bug-2</i>	50
5.7	<i>Retesting -2</i>	51
5.8	<i>Bug-3</i>	51
5.9	<i>Retesting-3</i>	52
5.10	<i>Bug-4</i>	53
5.11	<i>Retesting-4</i>	54
5.12	<i>Bug-5</i>	54

● List of Tables

Table No	Table Name	Page no
1	QFD Matrix	9
2	Test case for registration class	41-42
3	Test case for login class	43
4	Test case for flight booking class	45
5	Test case for number of passenger validation	46
6	Test case for payment amount validation	47
7	Test case for name length of hotel/flight	47-48
8	Bug Description	50

Chapter 1. Introduction

- **Project Overview**

The Airline Management System- “**SkyHigh Airlines**” is a comprehensive and user-friendly web-based application designed to streamline the flight and hotel booking experience for passengers while providing vigorous administrative tools for system management. The platform features an intuitive and visually appealing home page that serves as the gateway to two main user panels: Passenger Panel and Admin Panel.

Passengers can search and book flights by selecting source, destination, and travel date. Additionally, they have the option to book hotels at their selected destination. Upon successful payment through one of the multiple integrated payment gateways, passengers receive a digital e-ticket for both flights and hotels. The system requires secure user registration and login to proceed with bookings and access personalized features.

Passengers can:

- View flight status by date or flight number.
- Access a FAQ section and interact with a dummy chatbot for basic queries.
- View, modify, and manage their bookings from their profile.
- Provide feedback on their experience.

Admins have access to all passenger features, along with advanced management capabilities through a dedicated Admin Dashboard. This dashboard provides real-time analytics, including total users, bookings, flights, and revenue metrics. Admins can:

- View recent bookings with status indicators (e.g., confirmed, canceled).
- Add, update, or delete flights, hotels, users, and bookings.
- Change the status of bookings as needed.
- Access a detailed analytics page for monitoring system performance and trends.

This project effectively demonstrates the integration of front-end and back-end technologies to build a scalable, secure, and efficient airline and hotel management platform with clear roles, functionalities, and user experience

● **Objectives and Scope**

Objectives:

The primary objectives of the Airline Management System are:

1. To simplify the flight and hotel booking process by providing a user-friendly online platform for passengers.
2. To ensure secure access through proper user authentication (login/register system).
3. To provide real-time flight information including flight status by date or flight number.
4. To enable passengers to manage their bookings, including viewing and modifying flight and hotel reservations, and instantly receiving e-tickets.
5. To offer multiple payment gateway options for smooth and flexible transactions.
6. To allow passengers to submit feedback for improving service quality.
7. To provide an administrative panel for system managers to control flights, hotels, bookings, and users.
8. To present analytics and performance insights to admins through a dashboard, supporting better decision-making.
9. To simulate customer support with a dummy chatbot and FAQ section for assisting users.

Scope:

In Scope:

The scope of the Airline Management System includes:

- A dynamic and responsive home page for users to navigate the platform easily.
- A secure login and registration system for both passengers and admins.
- Flight booking functionality, including search by source, destination, and date.
- Hotel booking option for selected destinations.
- E-ticket generation upon successful booking and payment.
- Access to multiple payment gateways for transaction flexibility.

- A user profile section for managing personal bookings and providing feedback.
- Flight status checking feature by date or flight number.
- A FAQ page and dummy chatbot support for passenger queries.
- An Admin Dashboard with:
 - Overview of users, bookings, flights, and revenue.
 - Recent bookings management with status updates.
 - Management of flight, hotel, user, and booking data (CRUD operations).
 - An analytics page showing key metrics and trends.

Out of Scope:

- Real-time flight status notifications and alerts.
- Customer support via live chat, phone, and email.
- Mobile-friendly interface
- Automated customer notifications for flight changes.

• **Stakeholders**

1. Passengers (End Users)

- Primary users of the system who book flights and hotels.
- Interact with the system for travel planning and booking management.

2. Administrators

- Manage all aspects of the platform including users, bookings, flights, and hotels.
- Monitor system performance and take administrative actions through the dashboard.

3. System Developers (Our Team)

- Responsible for designing, developing, and maintaining the system.
- Ensure security, functionality, and user experience.

4. Potential Future Users (Travel Agencies, Airlines)

- Could use or integrate the system as a base for real-world travel booking solutions.

- **Technology Stack**

Frontend:

- **HTML** – For structuring the web pages.
- **CSS** – For styling and layout design.
- **JavaScript** – For interactive functionality and dynamic content.
- **Bootstrap** – For responsive design and pre-built UI components.

Backend:

- **PHP** – For server-side scripting and handling business logic.

Database:

- **MySQL** – For storing and managing user, flight, hotel, booking, and payment data.

Other Tools & Technologies:

- **AJAX** – For asynchronous data loading without refreshing the page.
- **jQuery** – For simplified JavaScript operations and DOM manipulation.
- **GitHub** – For version control and collaborative development.
- **XAMPP** – For local development and testing environment.
- **Figma** – For UI/UX design and prototyping.
- **InfinityFree** – For deploying the project online.
- **FileZilla** – For uploading files to the deployment server via FTP.

Chapter 2. Software Requirements Specification & Analysis

2.1 Stakeholder Needs & Analysis

- **Primary Stakeholders:**

- **Passengers:** Individuals who book flights and use the platform for flight-related services.
- **Airline Administrators:** Staff responsible for managing flight schedules, bookings, and passenger records.

- **Secondary Stakeholders:**

- **Travel Agencies:** Businesses that may use the platform to book flights for customers.
- **Payment Service Providers:** Entities handling secure financial transactions.
- **Developers & System Administrators:** Responsible for maintaining and upgrading the system.
- **Customer Support Teams:** Assisting passengers with queries, issues, and requests.

- **Methods for Requirement Elicitation (Surveys, interviews)**

- **Interviews:** Conducted with potential users (passengers and admins) to understand expectations.
- **Competitive Analysis:** Researched existing airline booking systems to identify strengths and gaps.

2.2 List of Requirements

- **Functional Requirements (FRs):**

1. Users should be able to register and log in securely.
2. The system shall support flight search with filters such as date, time, class, destination, and airline.
3. The system shall display real-time seat availability for selected flights.

4. The system shall allow users to book flights and receive e-tickets instantly upon payment.
5. The system should support multiple secure payment options.
6. The system shall allow users to cancel or modify bookings and handle refunds as per the policy.
7. Passengers should receive notifications for booking confirmations.
8. The system shall provide extra baggage and meal preference options during the booking process.
9. The system should include a loyalty program for frequent flyers.
10. The system shall allow users to view travel history and manage their bookings.
11. The system should support the use of offers, discounts, and promo codes during booking.
12. The system shall provide a Flight Status Dashboard displaying real-time updates (on-time, delayed, canceled) and send alerts to users.
13. The system shall support multi-city flight bookings.
14. The system shall allow users to submit feedback and rate their experience.
15. The system shall provide information about baggage policies and airport facilities.
16. The system shall allow hotel bookings to be made along with or independent of flight reservations.
17. The admin panel shall allow administrators to add, update, and cancel flights.
18. The admin panel shall enable management of customer support tickets.
19. The admin panel shall display customer feedback and reviews for action.
20. The admin panel shall provide analytics on Bookings, Payments, Flight status, Revenue breakdown and payment methods, Customer support (e.g., query distribution, resolution time)

● **Non-Functional Requirements (NFRs):**

1. The system shall run on the web.
2. The passwords shall be encrypted by the system.

3. It shall work fluently in case of many users and data.
4. The website shall have a user-friendly interface that is easy to navigate.
5. The system shall provide regular backups to prevent data loss.
6. It shall maintain the transactions flawlessly so that no error occurs.
7. The website shall respond to user requests within a specified time to get a smooth experience.

- **Extraordinary Requirements (Wow Factors):**

1. The system should include a Price Prediction Tool that uses machine learning to suggest the cheapest times to book flights.
2. The system should provide Personalized Travel Suggestions based on a user's past behavior and preferences.

2.3 Quality Function Deployment

- **Customer Requirements (CRs) List**

1. Quick and hassle-free booking
2. Transparent pricing with no hidden charges
3. Flexible and secure payment options
4. Real-time flight updates and notifications
5. Easy cancellation and refund process
6. Seat selection and add-on services (meals, baggage)
7. Reliable and responsive customer support
8. Discounts and loyalty rewards
9. Personalized travel suggestions
10. Price prediction insights

- **Engineering (Technical) Requirements (TRs) List**

1. Secure authentication system
2. Efficient database management

3. Payment gateway integration
4. Notification and alert system
5. Seat selection and add-on services logic
6. Customer support ticketing system
Dynamic pricing and discount engine
7. Loyalty rewards engine
8. Machine learning for price prediction
9. User behavior analytics for personalization

- **Quality Function Deployment Matrix (House of Quality)**

Table 1: QFD Matrix

User Requirement	Performance	Security	Usability	Reliability	Scalability	Speed	Customer Support	Aesthetic Design	Notifications	Loyalty Program	Discount Offers	Price Prediction Tool
Register and log in securely	Medium	Strong	Medium	Medium	Medium	Medium	Medium	Medium	Weak	Weak	Weak	Weak
Search flights with filters	Strong	Medium	Strong	Strong	Medium	Medium	Medium	Medium	Weak	Weak	Weak	Weak
Display real-time seat availability	Strong	Medium	Strong	Strong	Medium	Strong	Medium	Medium	Medium	Weak	Medium	Medium
Book flights and receive e-tickets	Strong	Strong	Strong	Strong	Medium	Strong	Medium	Medium	Strong	Weak	Medium	Weak
Secure multiple payment options	Medium	Strong	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Weak	Medium	Weak
Cancel/modify bookings & refund policy	Medium	Strong	Medium	Strong	Medium	Medium	Medium	Medium	Strong	Weak	Weak	Weak
Notifications for booking confirmations	Medium	Medium	Strong	Strong	Medium	Strong	Strong	Medium	Strong	Weak	Medium	Medium
Seat selection & add-on services	Medium	Medium	Strong	Medium	Medium	Medium	Medium	Strong	Medium	Medium	Medium	Weak
Loyalty program for frequent flyers	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Strong	Medium	Weak
Discounts & promo codes	Medium	Medium	Strong	Medium	Medium	Medium	Weak	Strong	Medium	Medium	Strong	Weak
Add/update/cancel flights (Admin)	Medium	Strong	Medium	Strong	Strong	Medium	Medium	Weak	Weak	Weak	Weak	Weak
Hotel booking integration	Medium	Medium	Strong	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Weak
Real-time flight status & alerts	Medium	Medium	Medium	Strong	Medium	Strong	Medium	Medium	Strong	Weak	Weak	Weak
Display baggage & airport info	Medium	Medium	Strong	Medium	Medium	Medium	Weak	Medium	Medium	Weak	Weak	Weak
Feedback & rating system	Medium	Medium	Strong	Strong	Medium	Medium	Strong	Medium	Medium	Weak	Weak	Weak
Personalized travel suggestions	Medium	Medium	Strong	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Weak
Price prediction for cheapest times	Medium	Medium	Medium	Medium	Medium	Medium	Weak	Medium	Medium	Medium	Medium	Strong

2.4 Requirements Modeling

● Use Case Diagrams

Actors:

- Passenger (End User)
- Admin (Airline Staff)

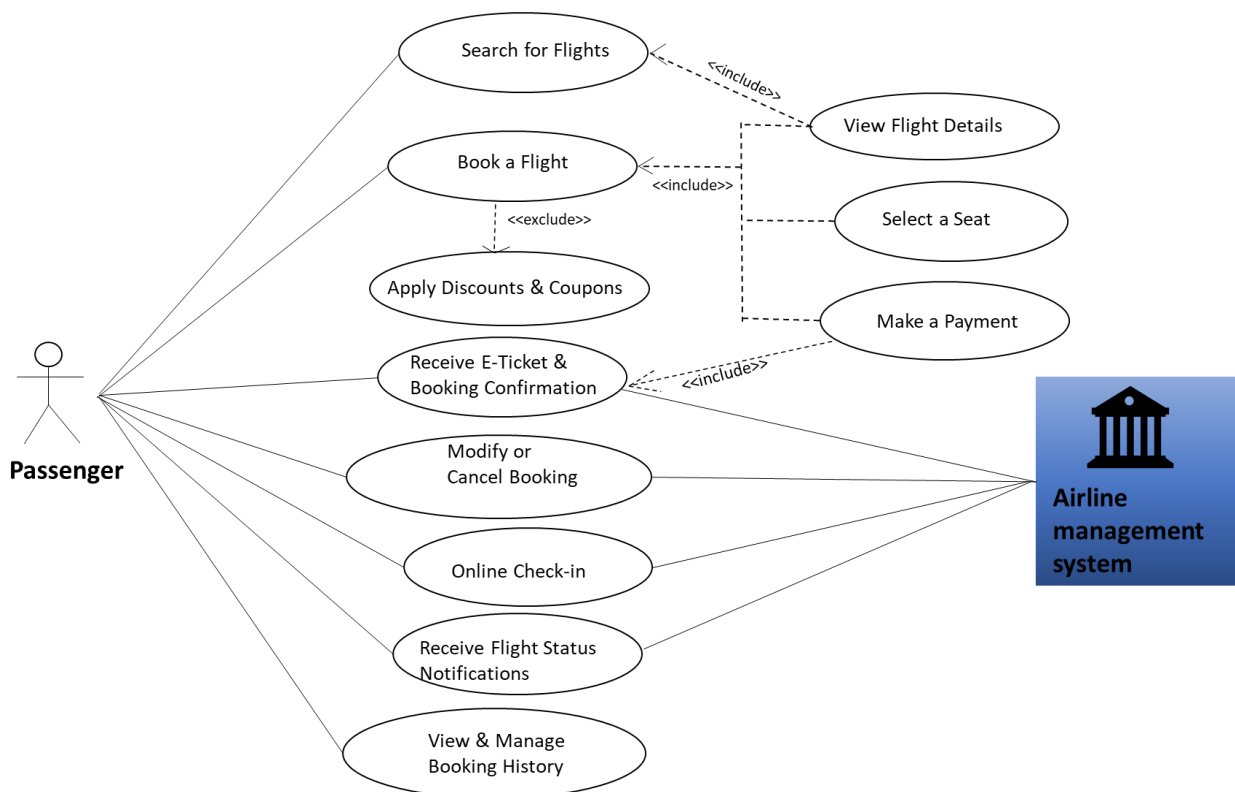


Figure 2.1: Use Case Diagram for Passenger

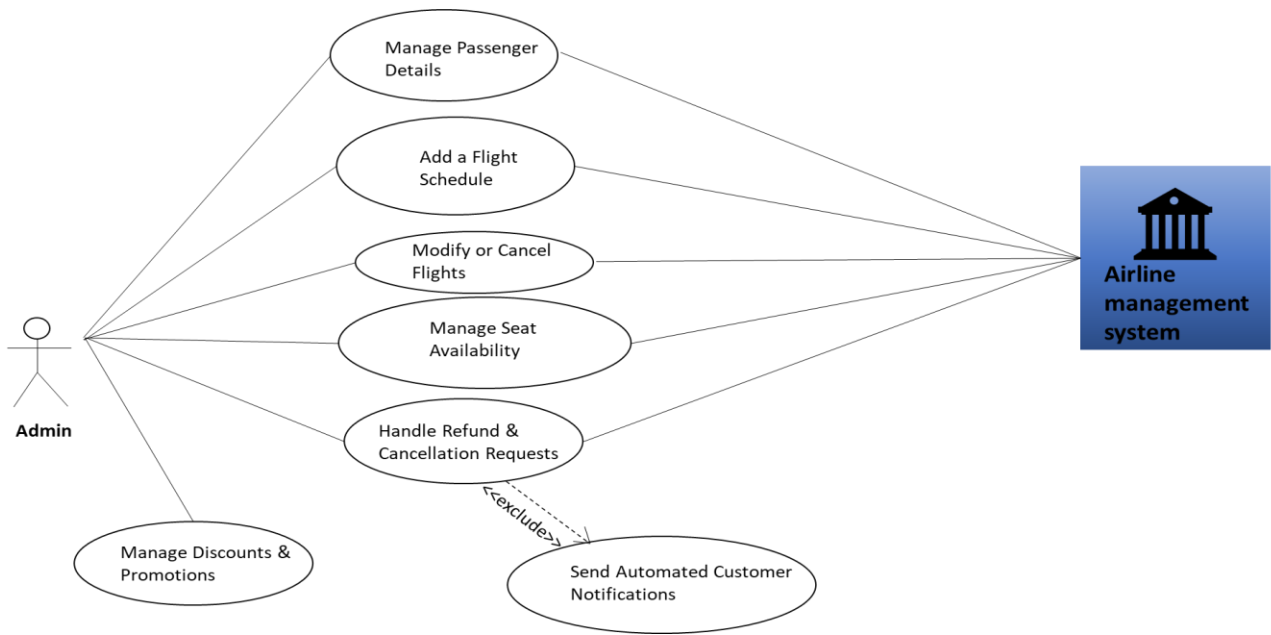


Figure 2.2: Use Case Diagram for Admin

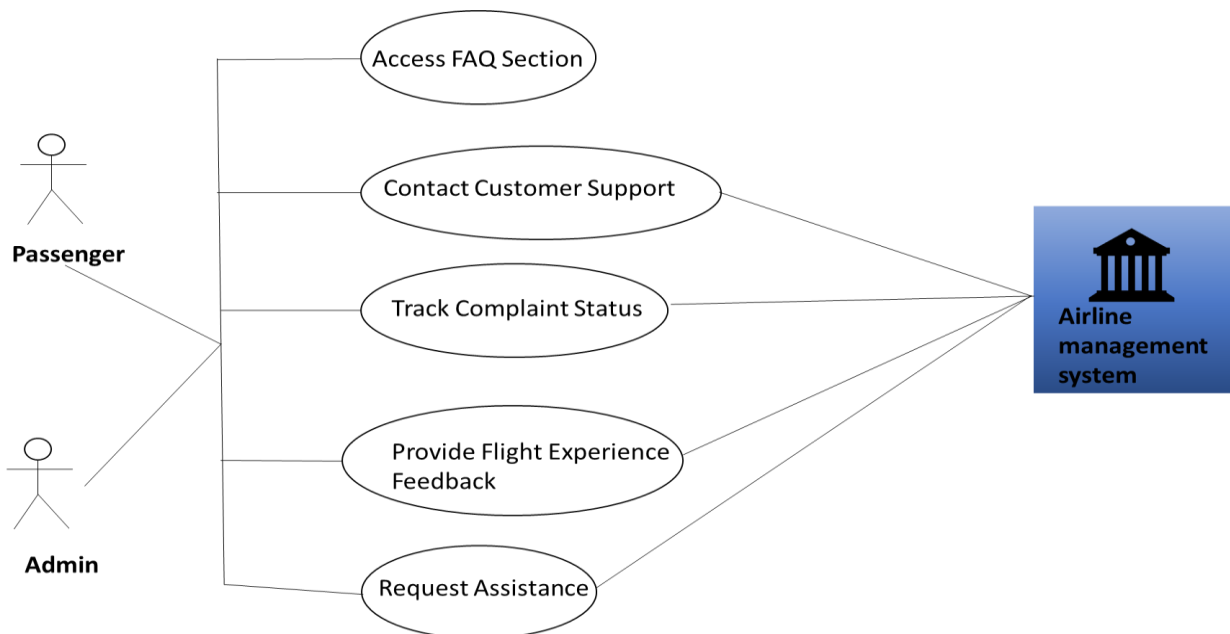


Figure 2.3: Use Case Diagram for Customer Support & Feedback

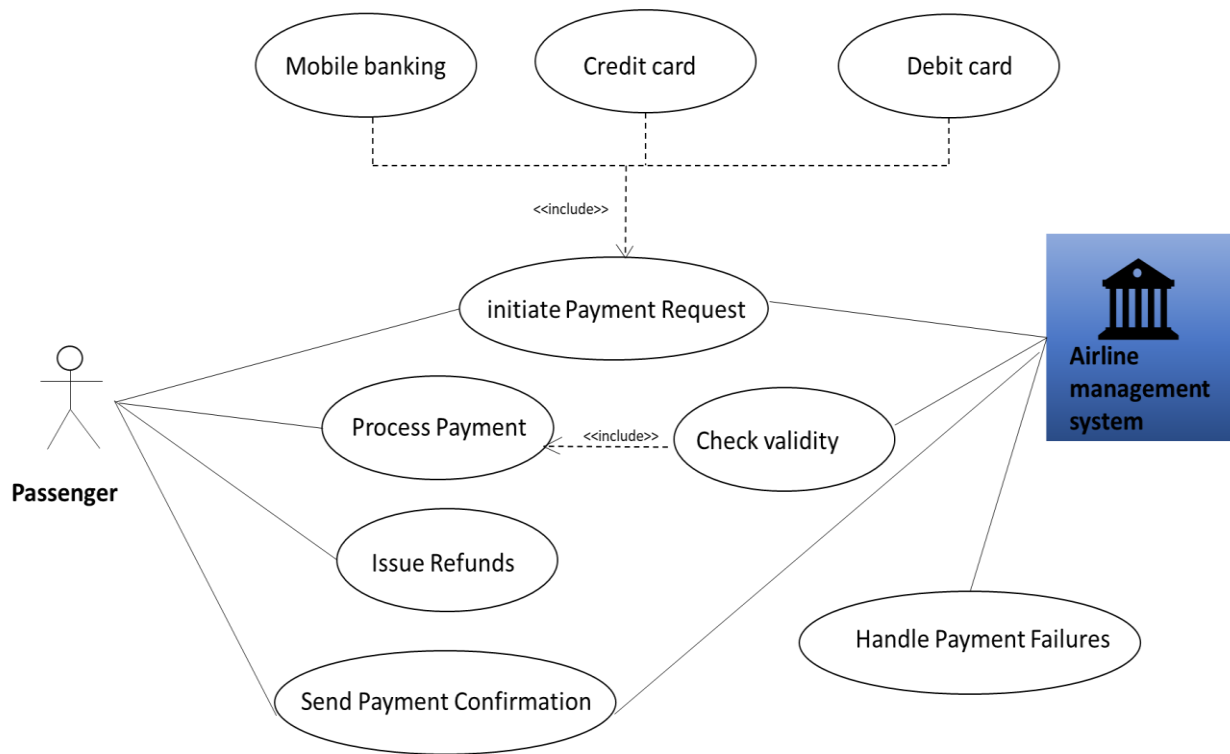


Figure 2.4: Use Case Diagram for Payment Processing

- Activity diagrams

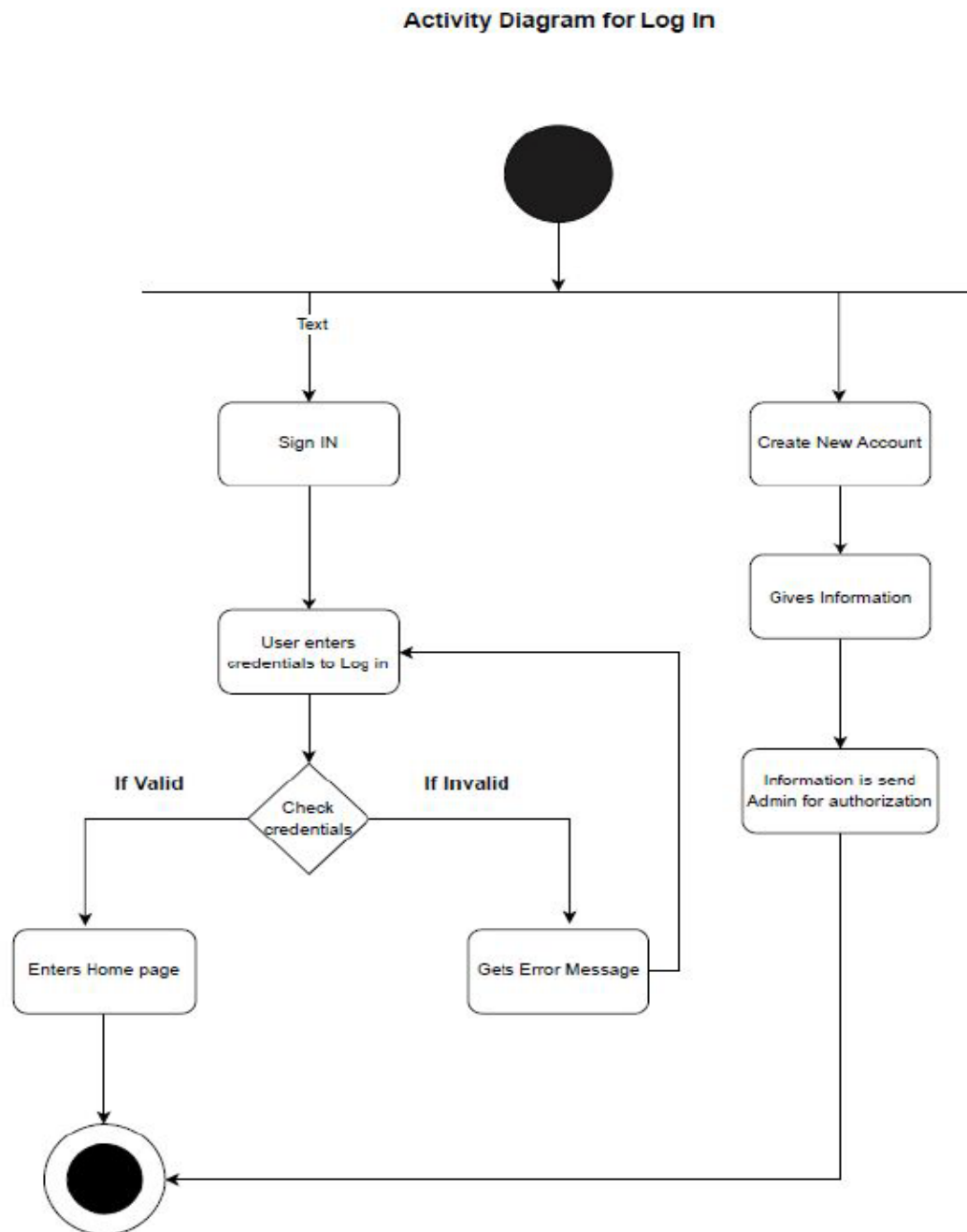


Figure 2.5: Activity Diagram for Login & Register

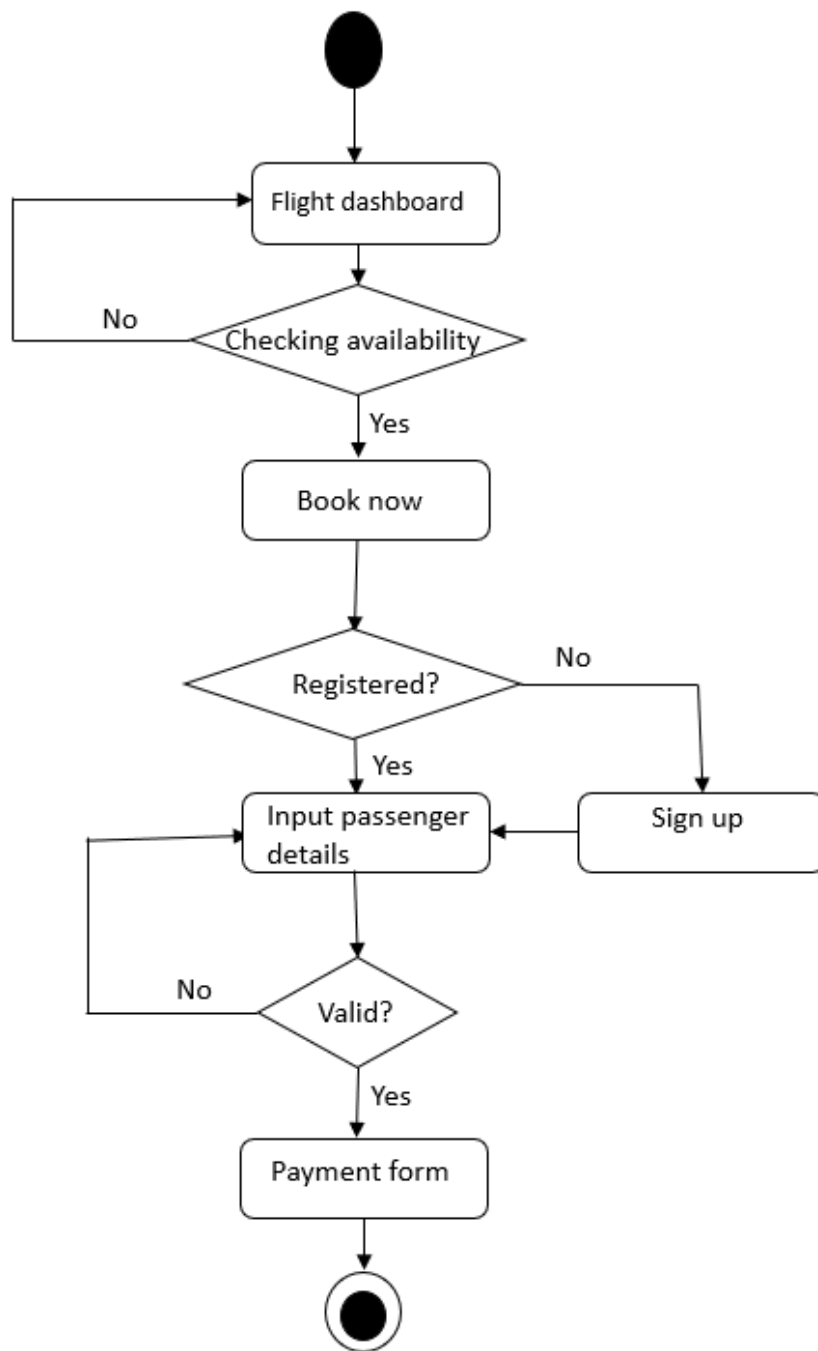


Figure 2.7: Activity Diagram for flight booking

Activity Diagram for Pay Bill

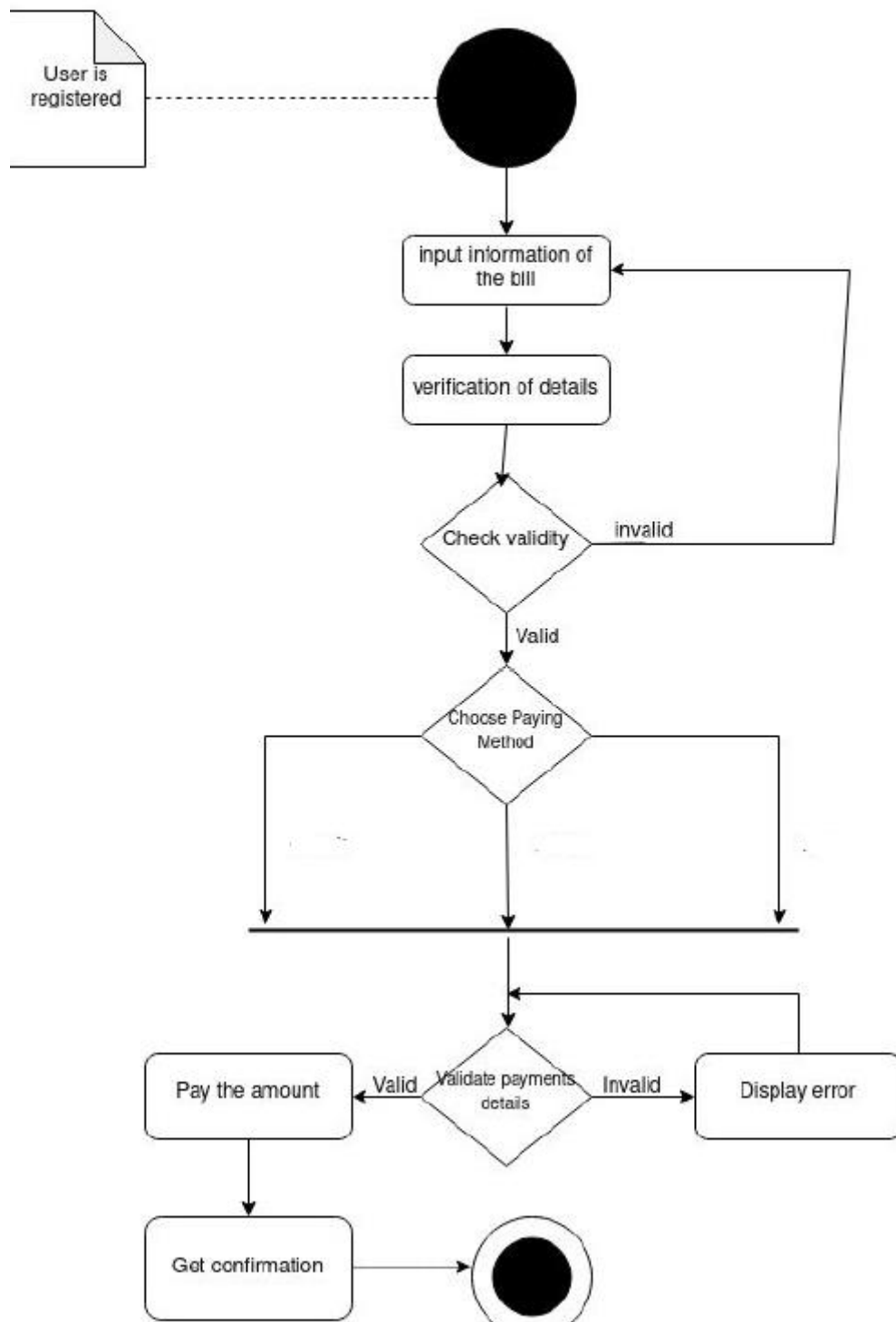


Figure 2.6: Activity Diagram for Payment

- UI Sketches

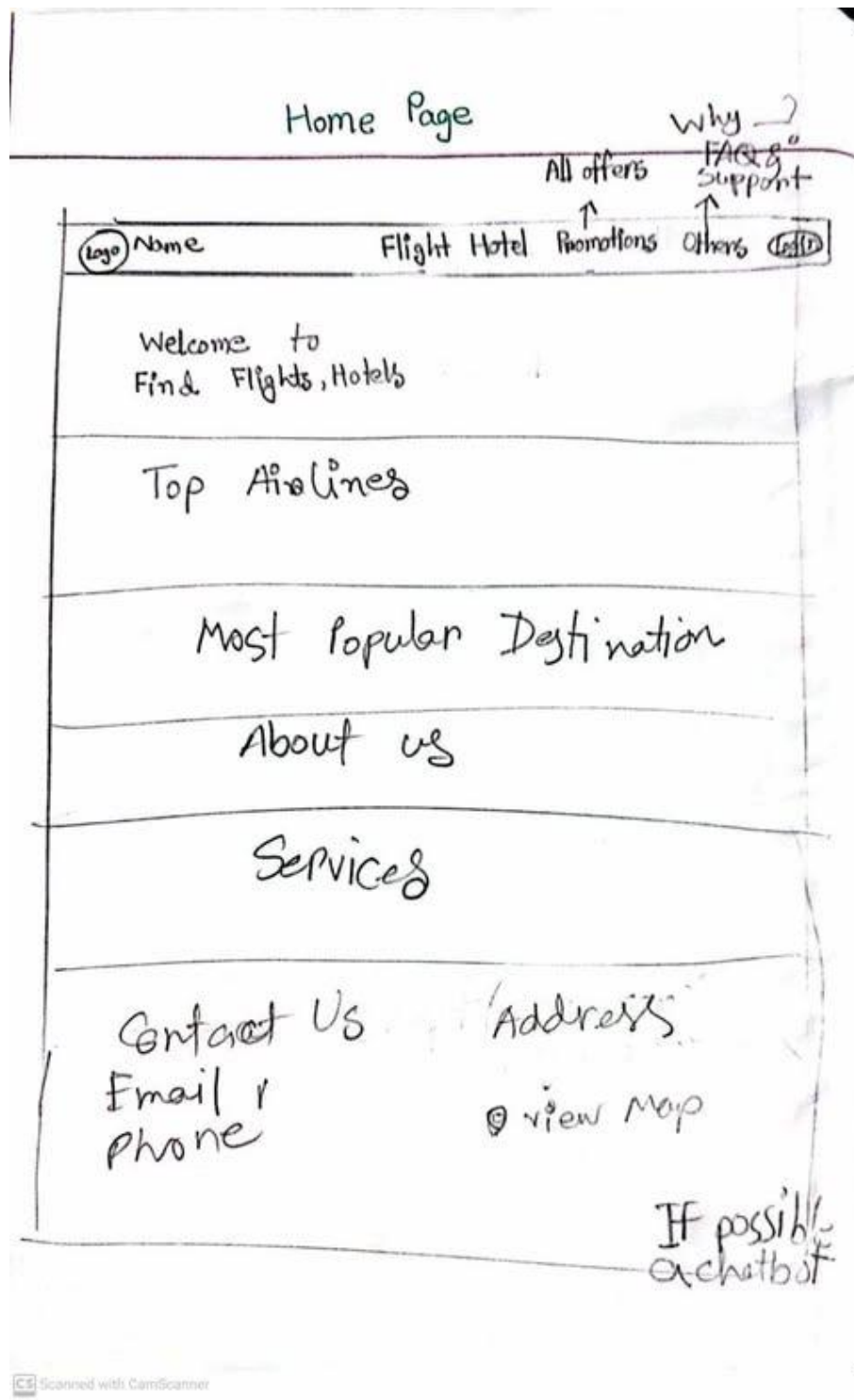


Figure 2.4: UI Sketch of HOMe Page

For Passengers

Search Flights

Search Flights

Flight Booking Options

Economy

Extra Baggage (kg)

No Food

Single Route

Enter Promo Code

Apply Promo Code

Cancel Flight

Modify Flight

Deployed by **Noob Coders**

AirTicket

Home Flights Help Center Admin Panel Passenger History Login

Book Your Flight Now
Find the best deals on airline tickets

Search Flights

Search Flights

Flight Booking Options

Economy

Extra Baggage (kg)

No Food

Single Route

Enter Promo Code

Apply Promo Code

Cancel Flight


Modify Flight

AirTicket

Home Flights Help Center Admin Panel

Book Your Flight Now
Find the best deals on airline tickets

FOR ADMIN

 AirTicket Admin

[Dashboard](#) [Manage Flights](#) [Manage Users](#) [Logout](#)

Admin Dashboard

Total Flights
120

Available Tickets
3000

Ticket Demand
High

Employees Working Today
50

Average Customer Rating
4.5/5

Manage Flights

Add Flight

Flight No	From	To	Departure Time	Seats Available	Price (\$)	Actions	Notice
AI-202	New York	London	10:30 AM	150	500	<div>Cancel Modify</div> <div>Adjust Price</div>	<div>Enter notice</div> <div>Send Notice</div>

Deployed by Noob Coders

Chapter 3. Software Design

3.1 Architectural Design

- **Architectural Context Diagram**

Architectural context diagram

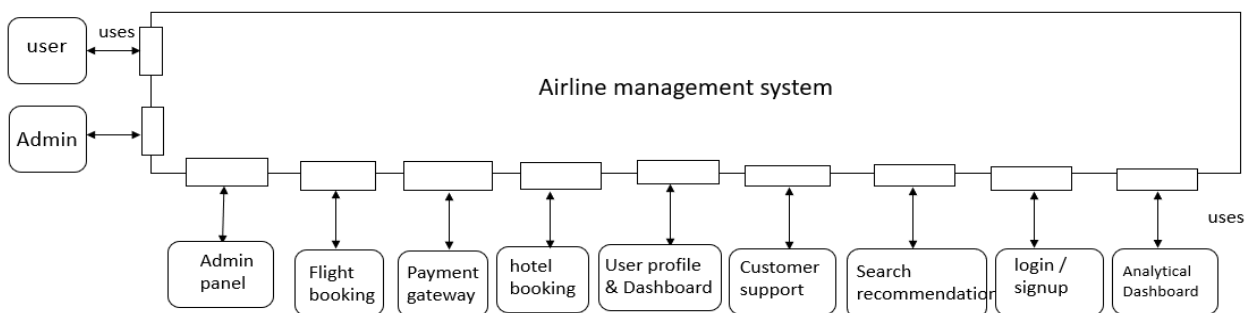


Figure 3.1: Architectural Context Diagram

- **Top-Level Component Diagram**

Top-level component diagram

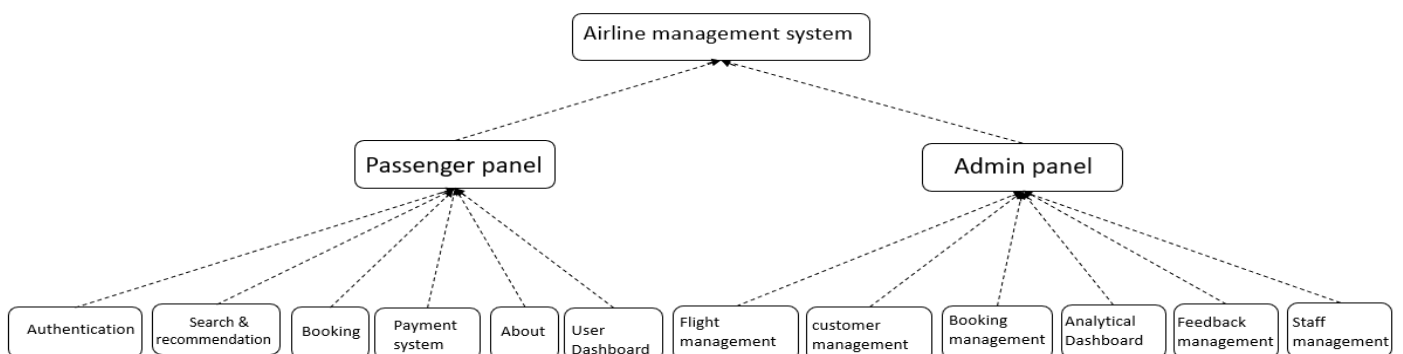


Figure 3.2 :Top-Level Component Diagram

- **Instantiation of Each Component with Component Elaboration**

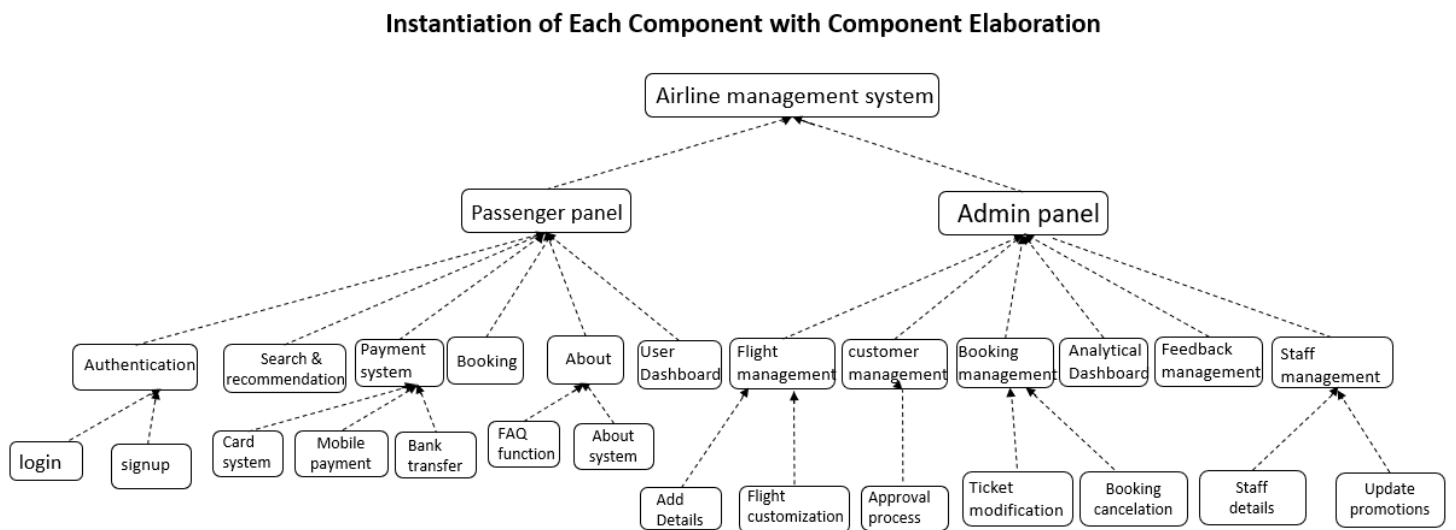


Figure 3.3 :Instantiation of Each Component with Component Elaboration

3.2 Component-Level Design

- Elaboration of Design Components

Authentication Component

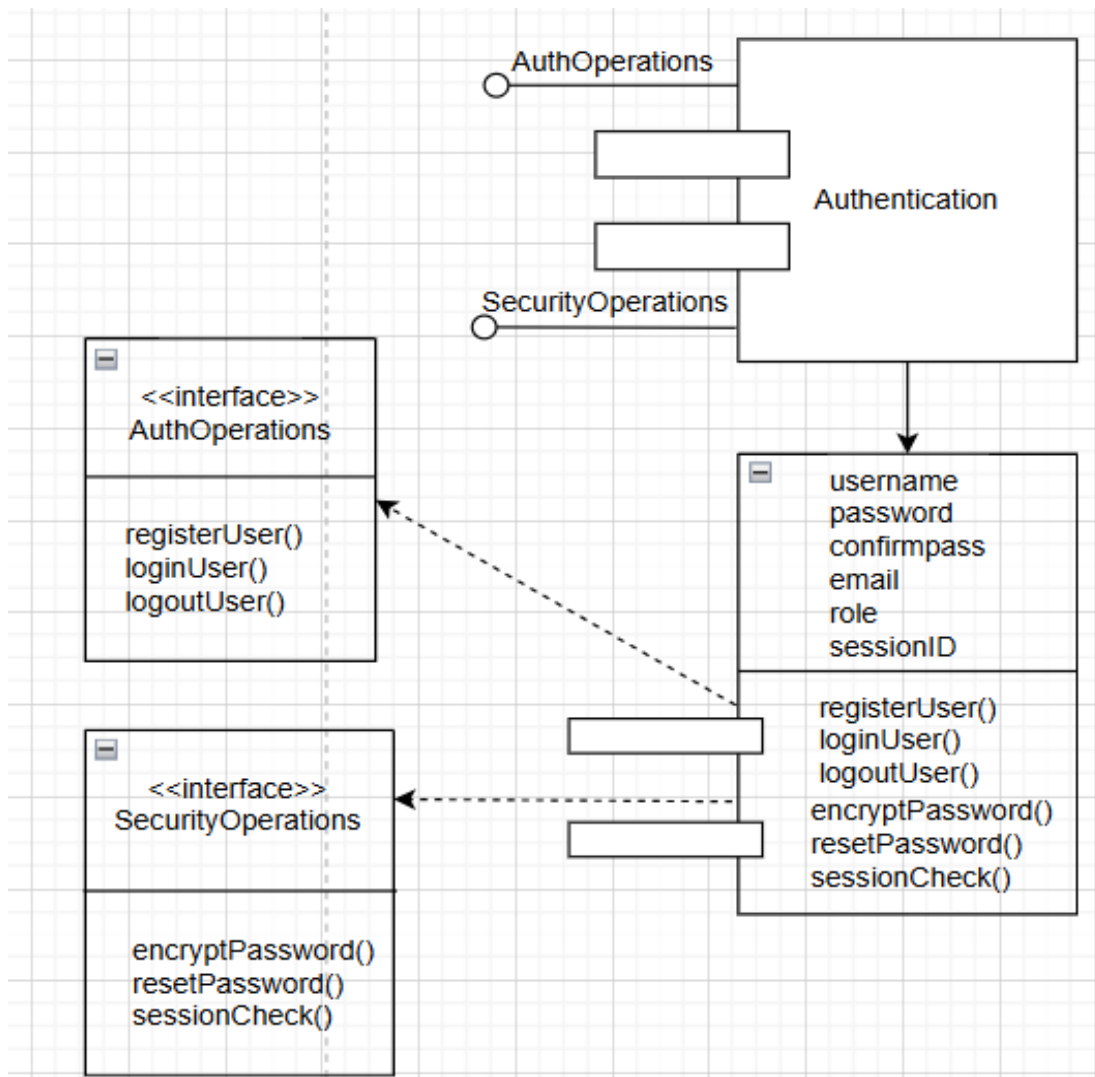


Figure 3.4: Authentication Component

Search & Recommendation Component

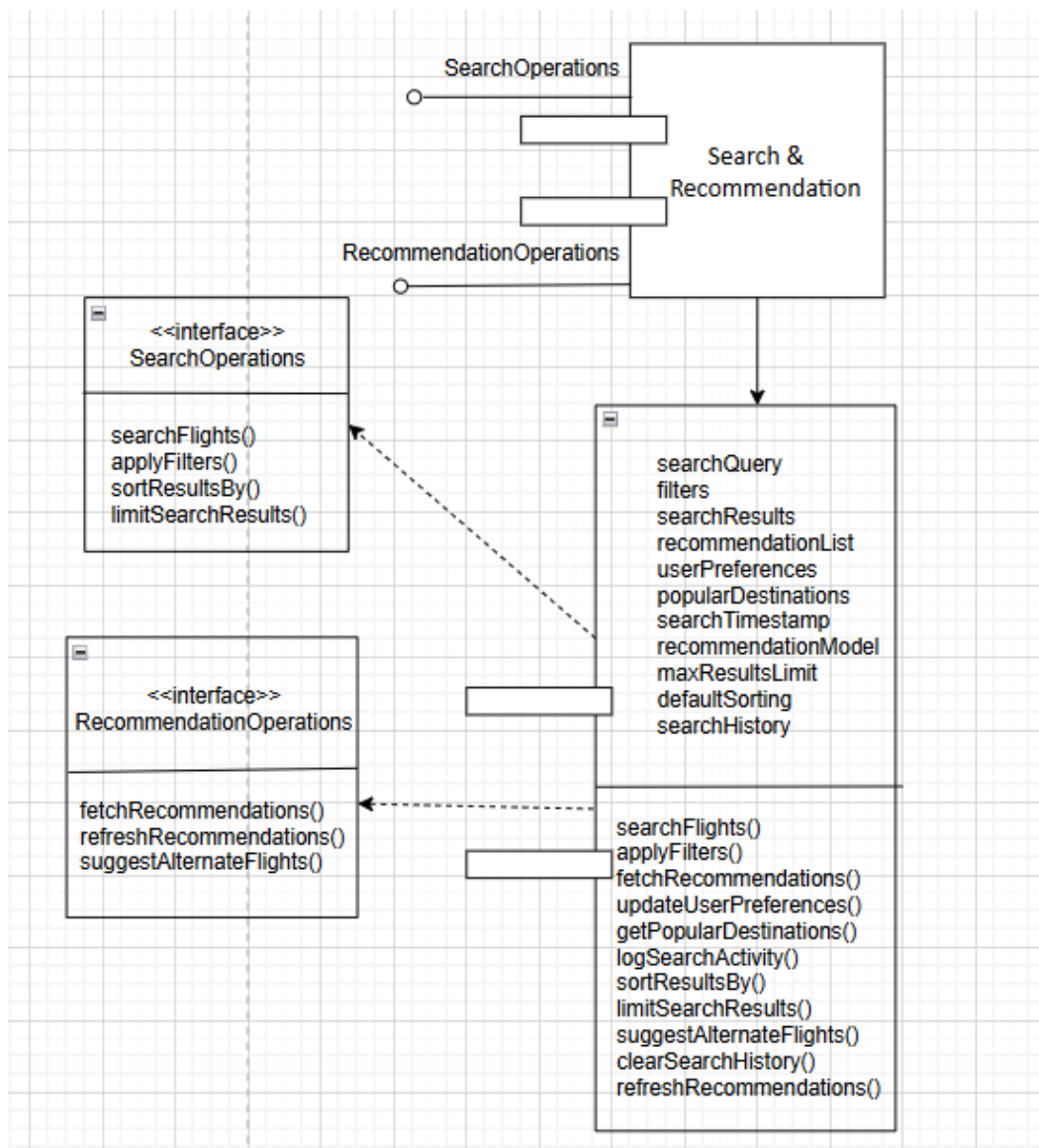


Figure 3.5: Search & Recommendation Component

Booking Component

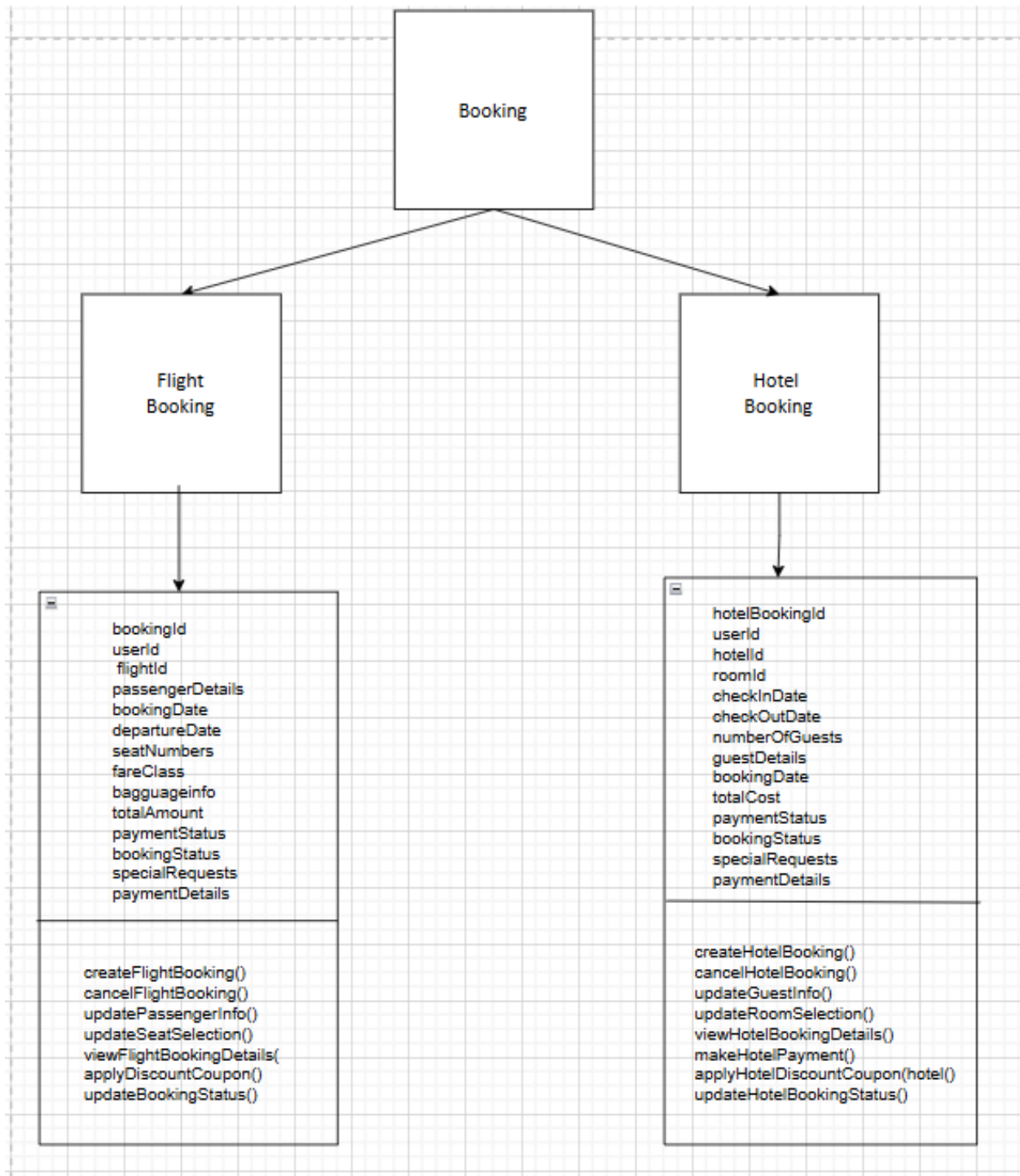


Figure 3.6: Booking Component

Payment Component

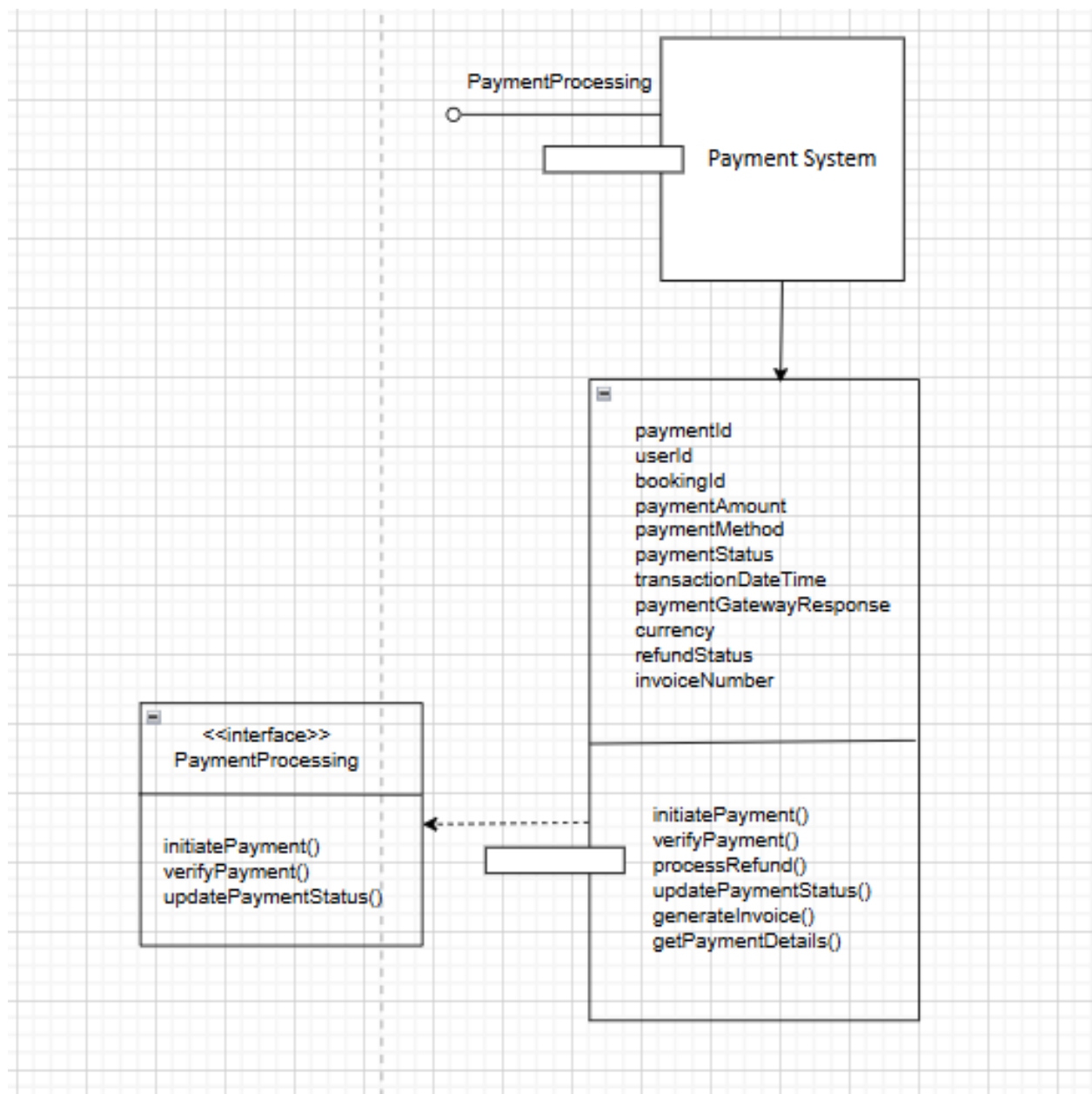


Figure 3.7: Payment Component

About Component

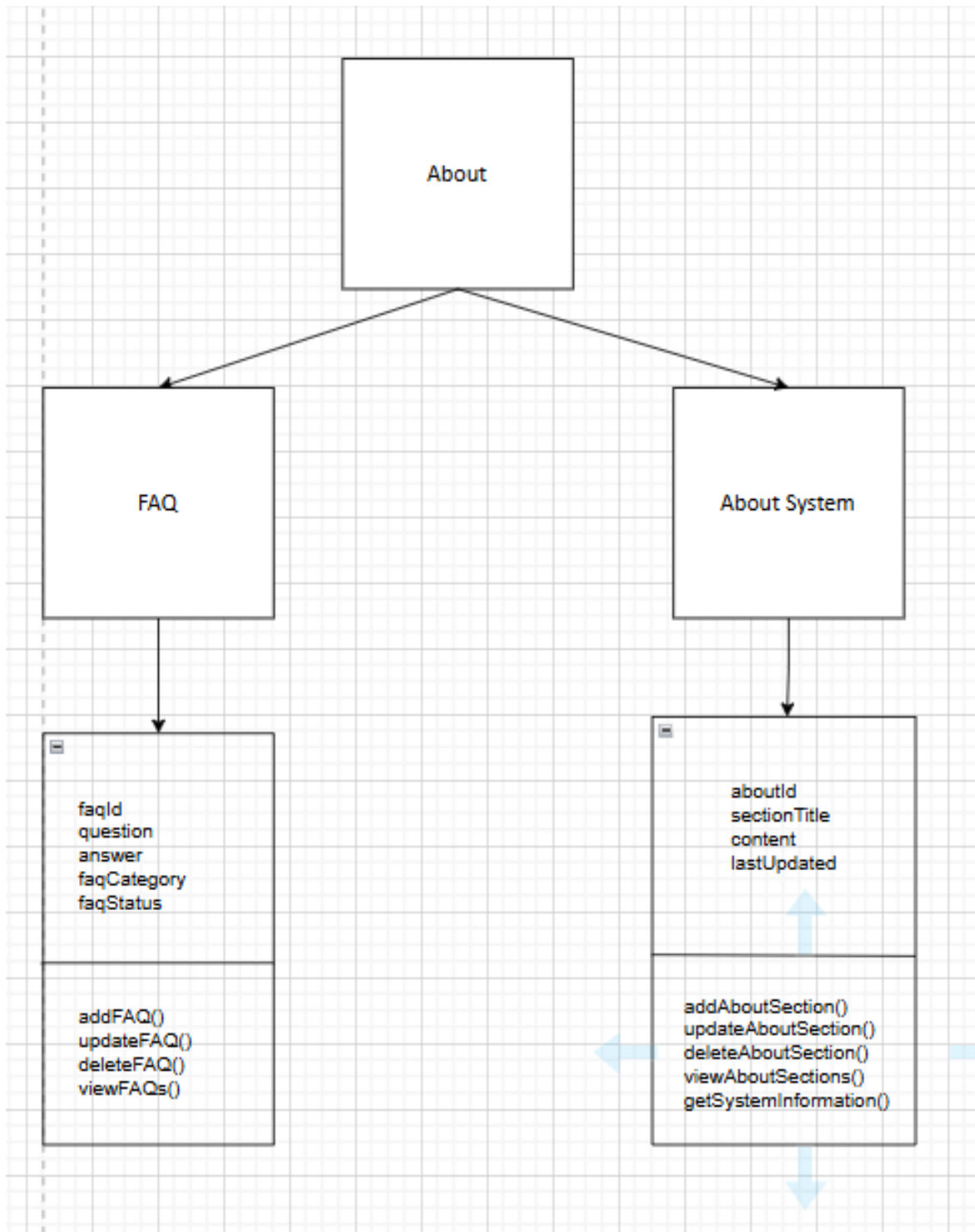


Figure 3.8: About Component

User Dashboard Component

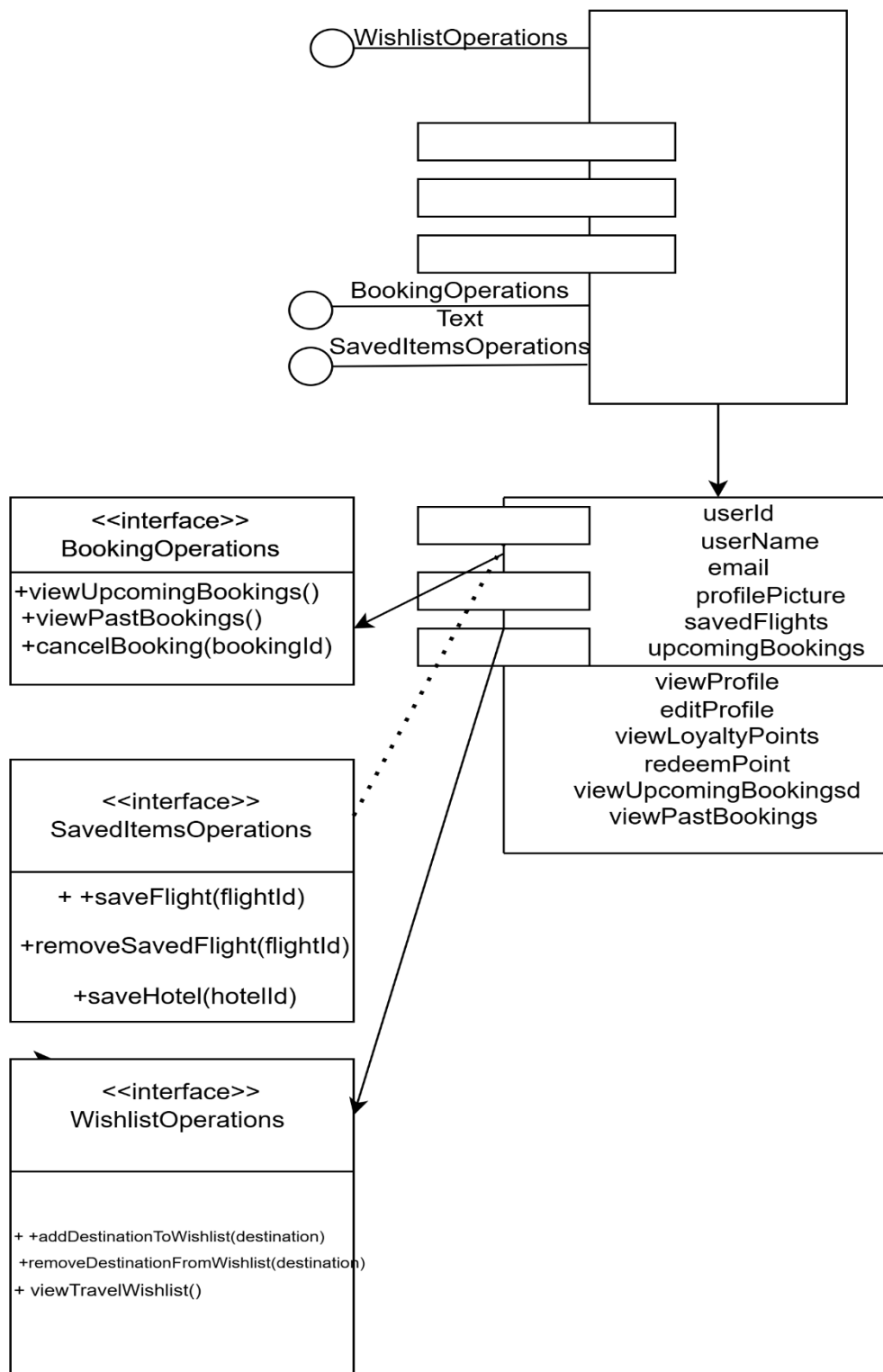


Figure 3.9: User Dashboard Component

Flight Management Component

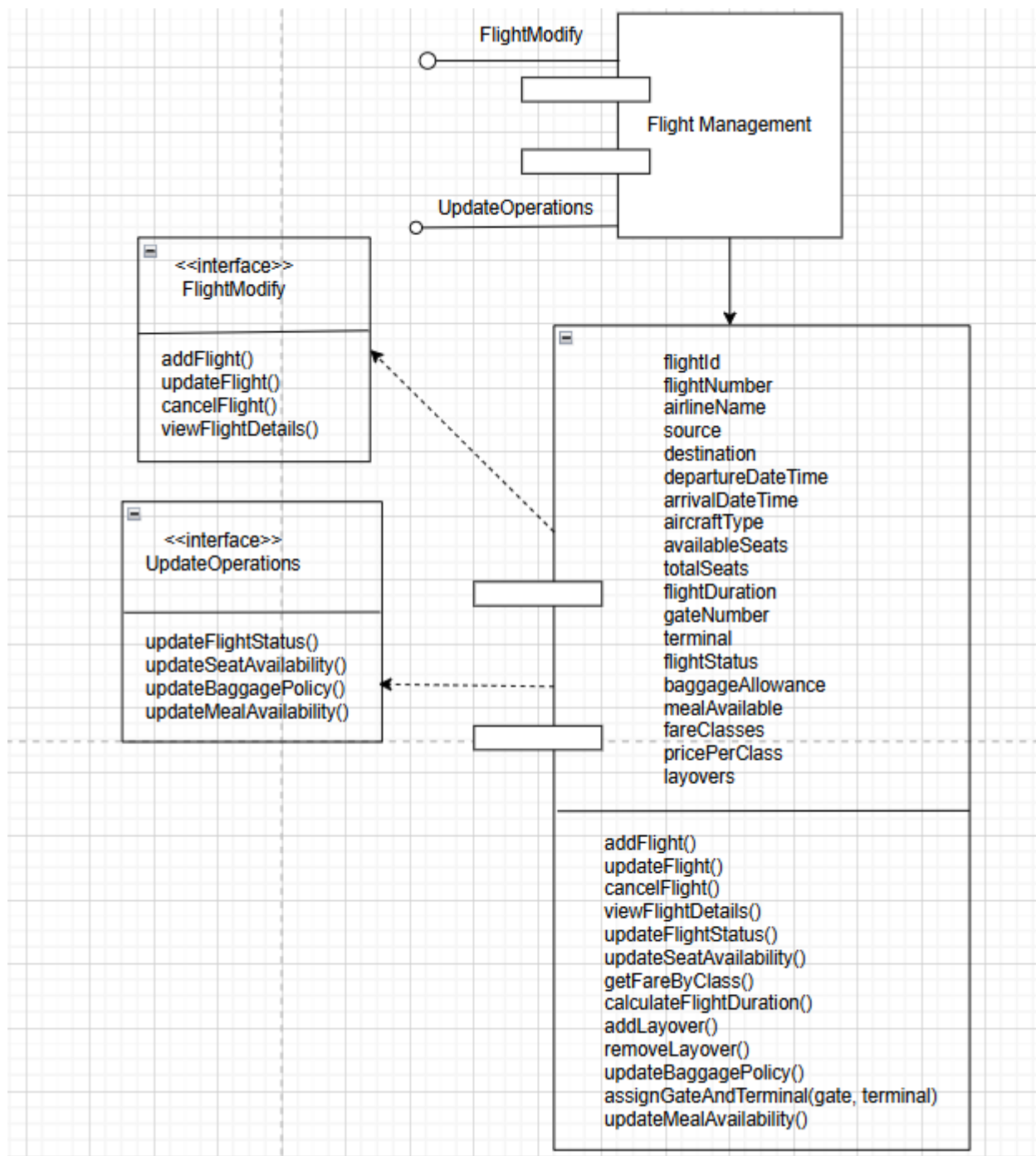


Figure 3.10: Flight Management Component

Booking Management Component

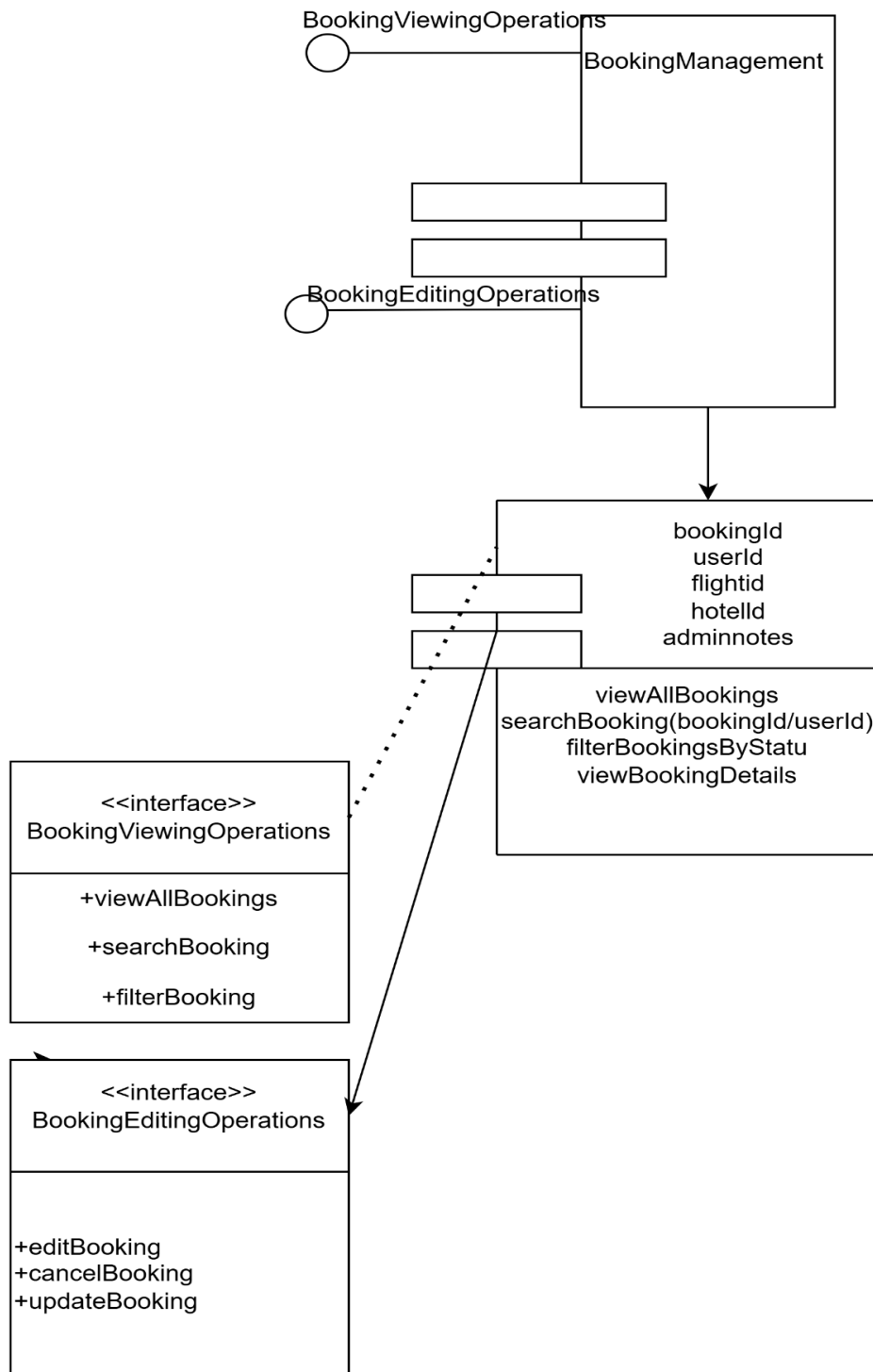


Figure 3.11: Booking Management Component

Feedback Management Component

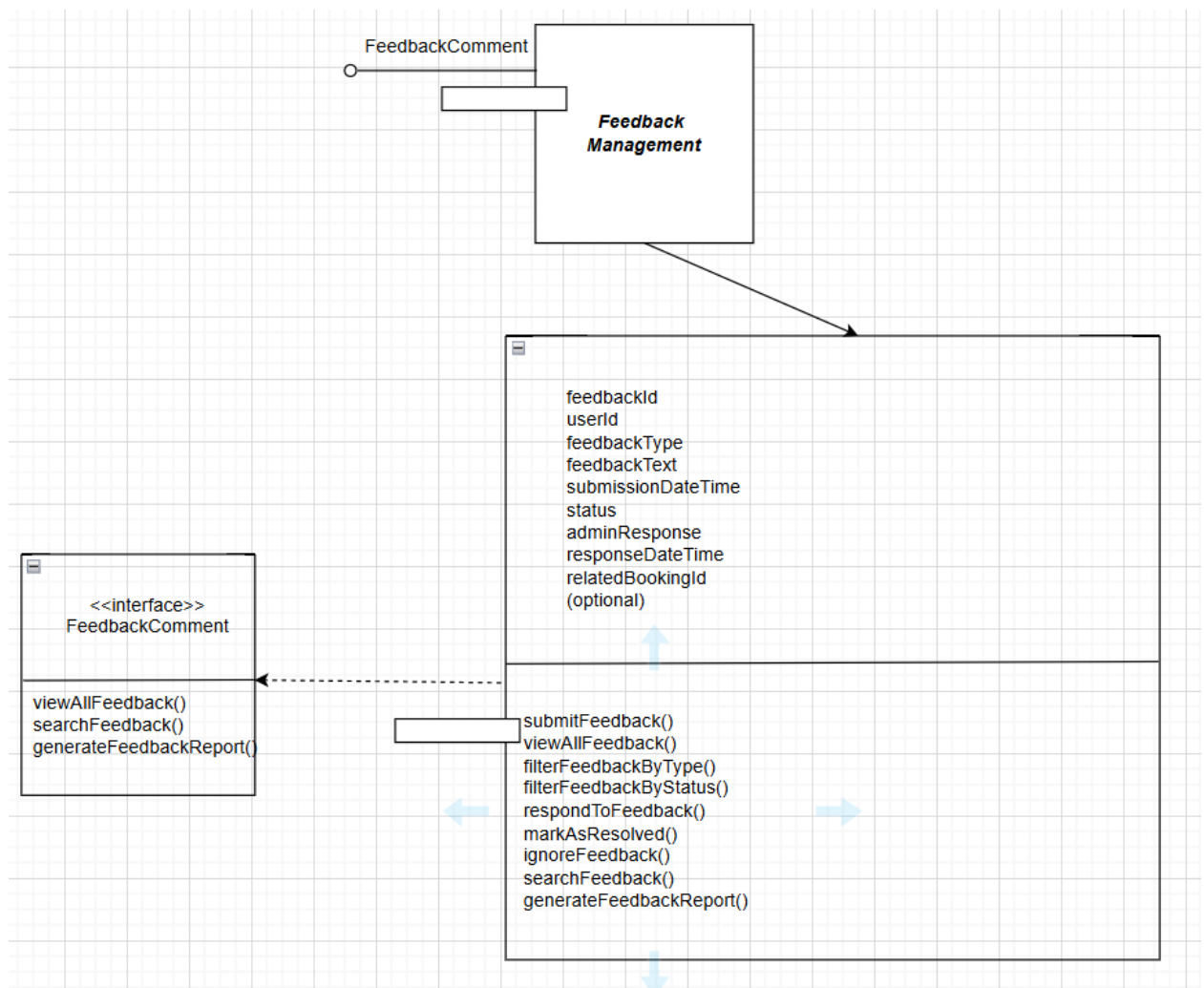


Figure 3.12: Feedback Management Component

10. Analytical Dashboard Component

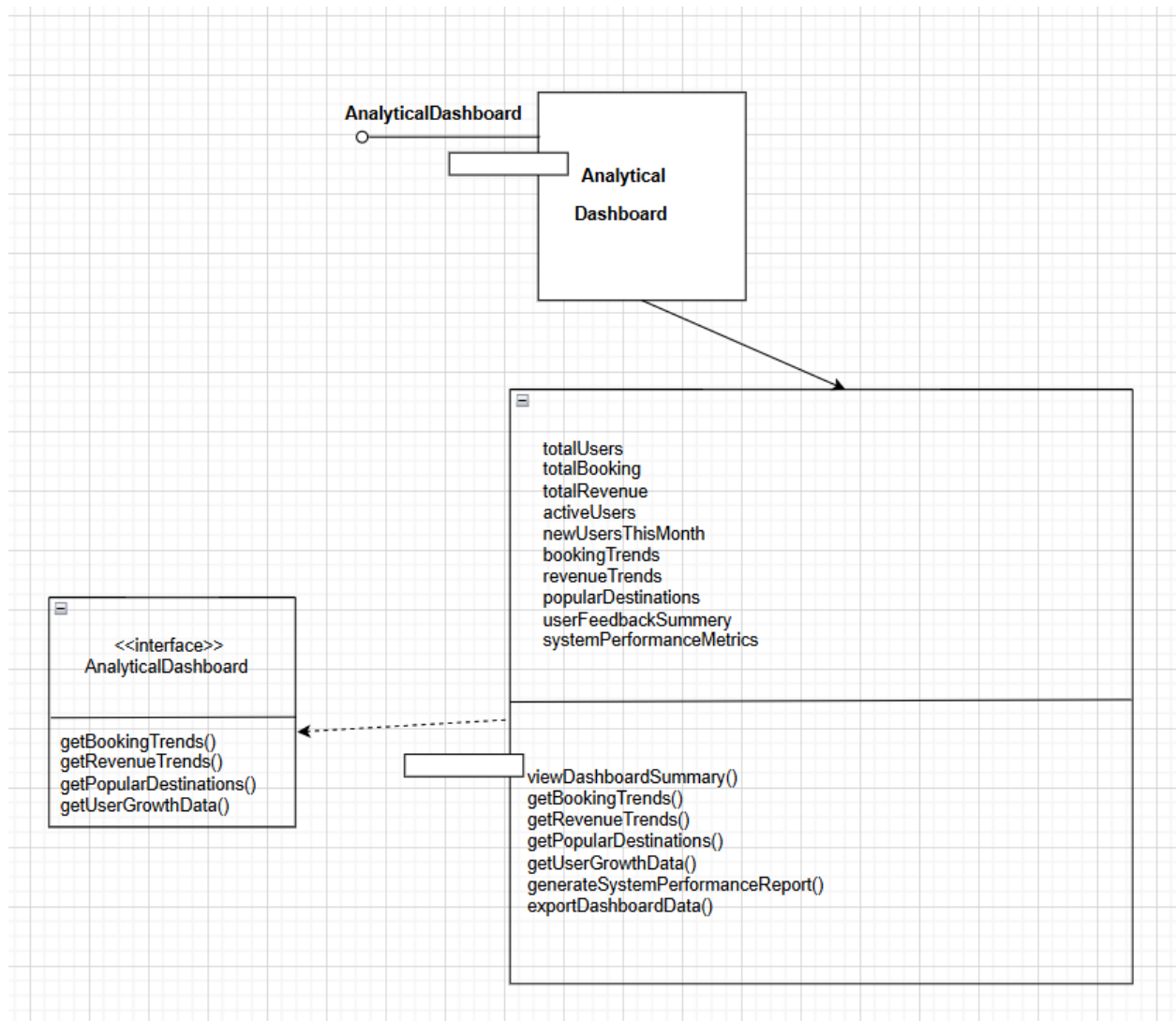


Figure 3.13: Analytical Dashboard Component

• Class Diagram

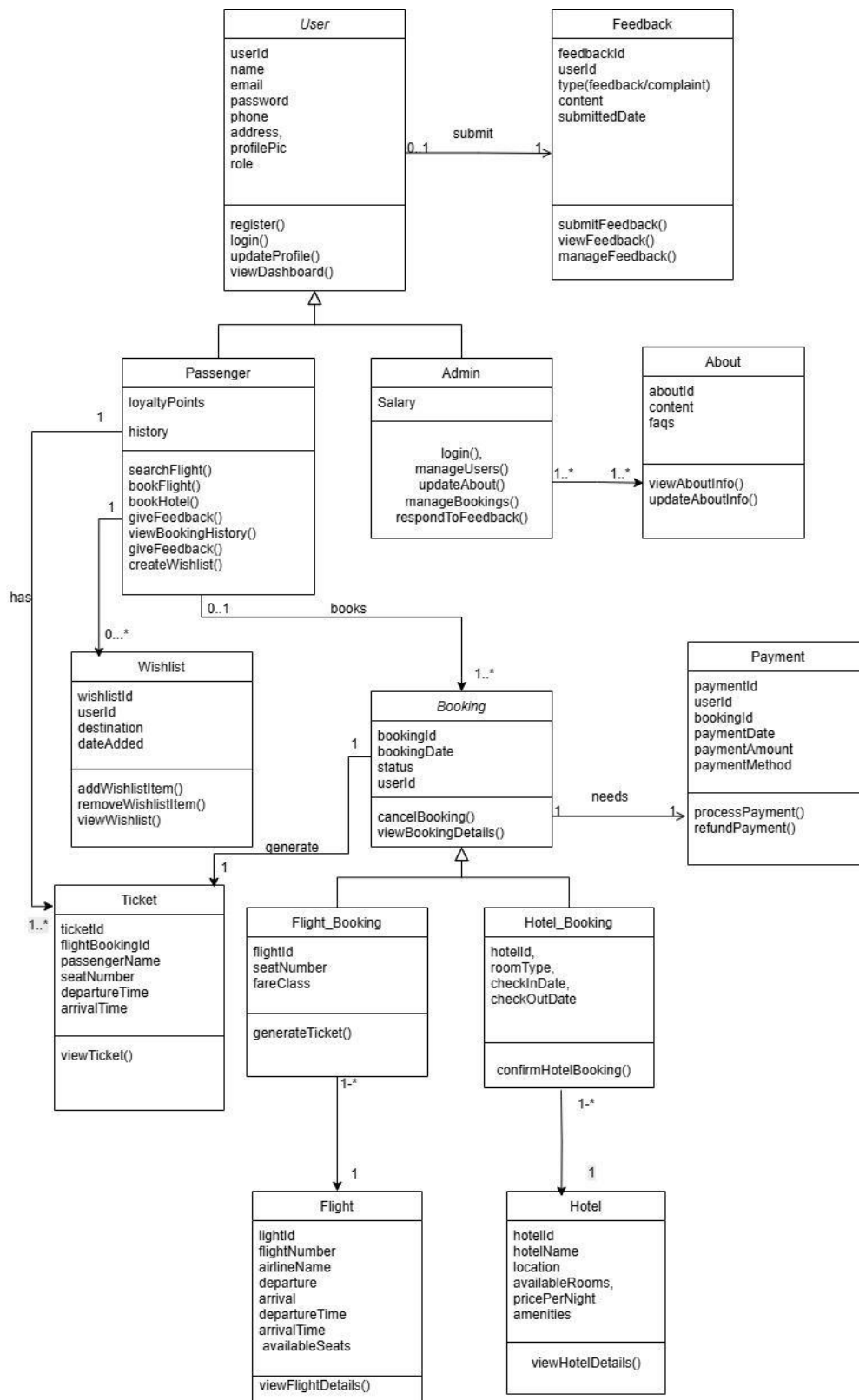


Figure 3.14: Class Diagram

● Database Design

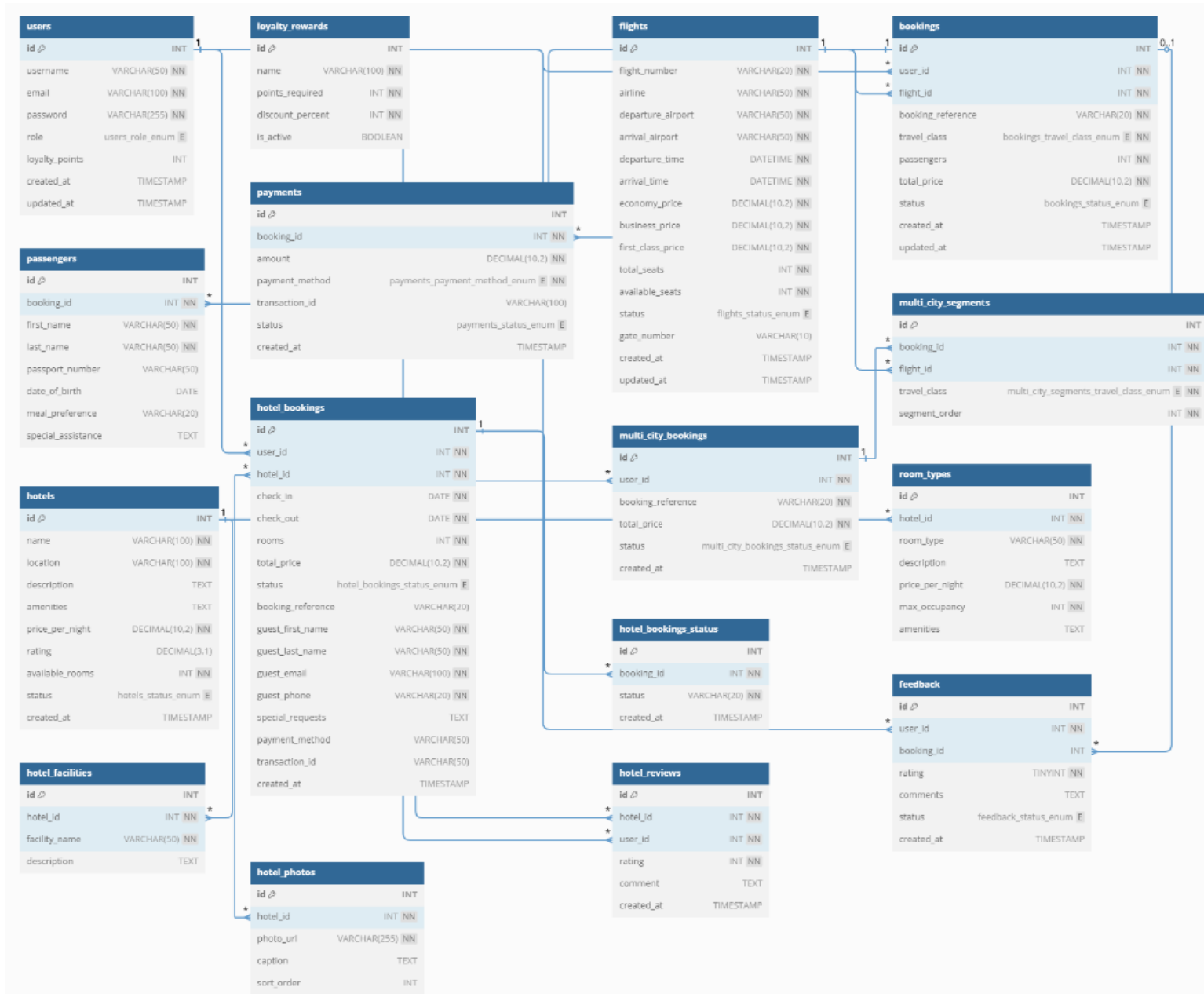


Figure 3.15: ER Diagram 1

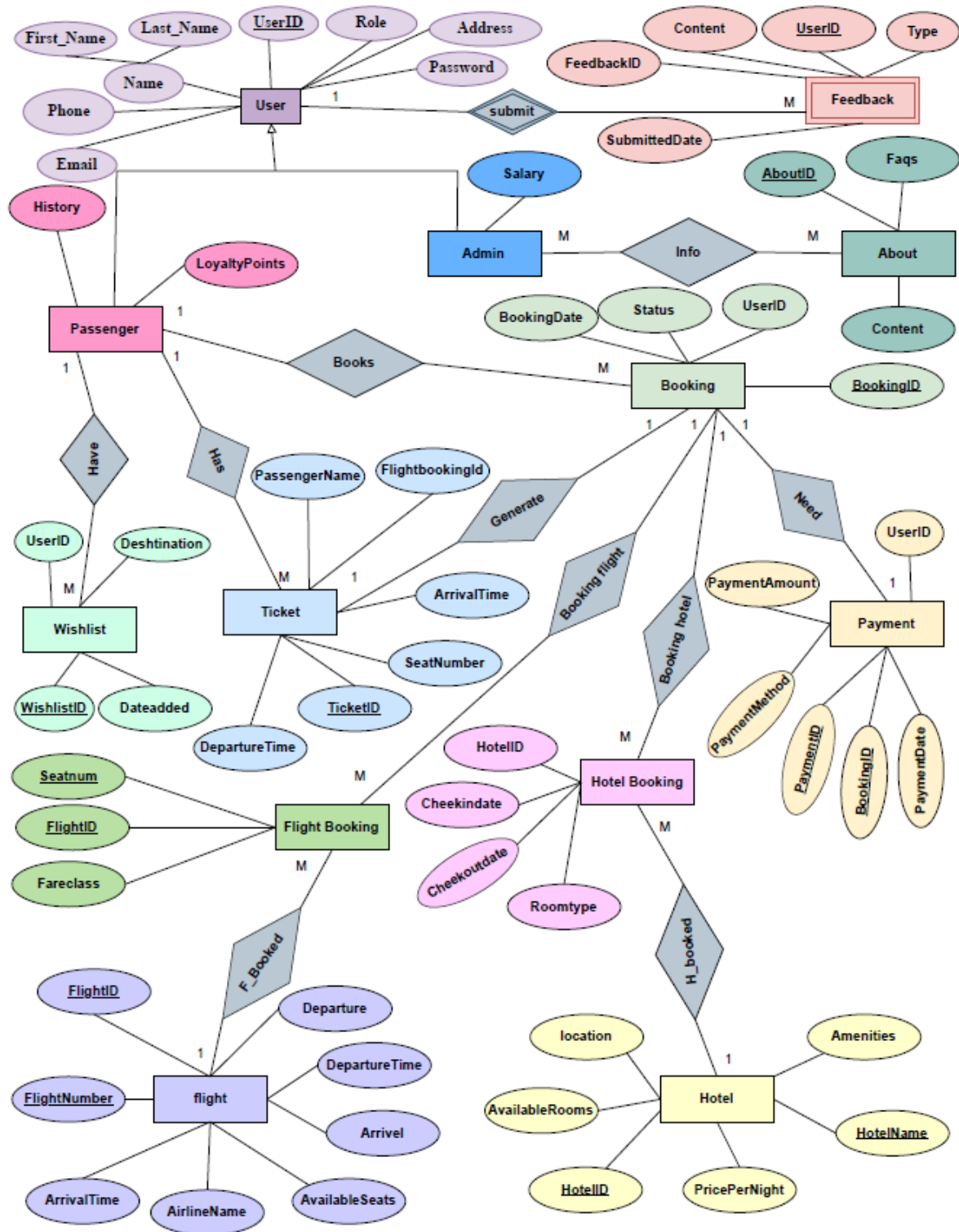
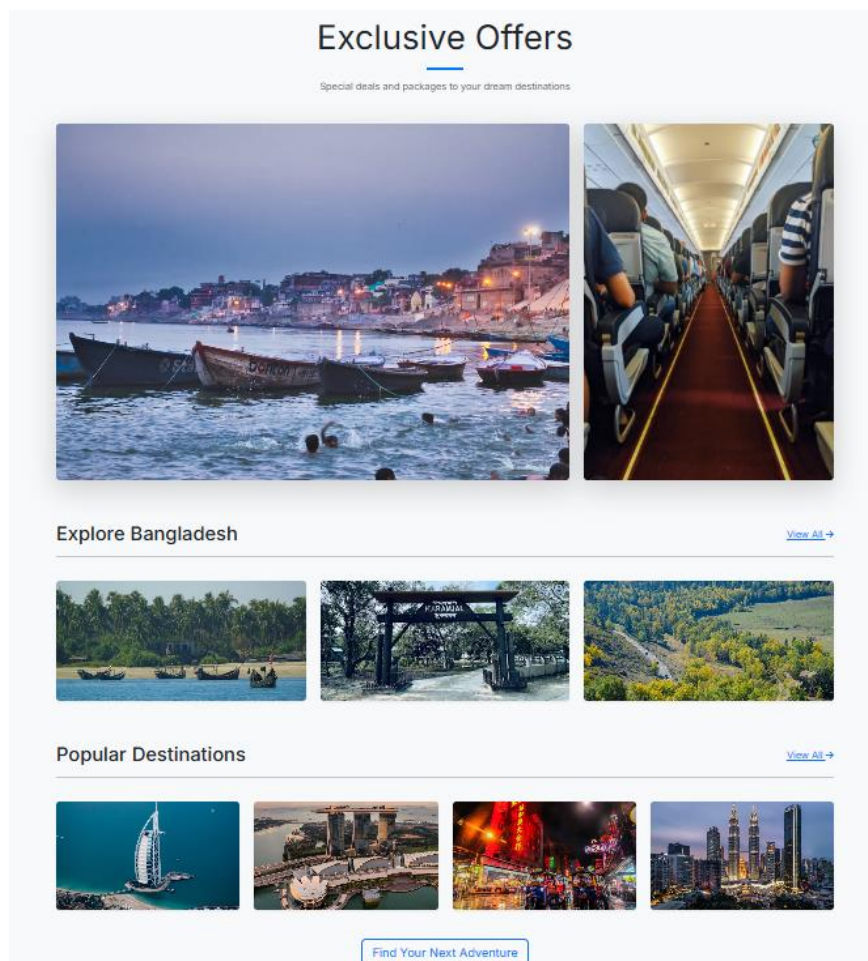
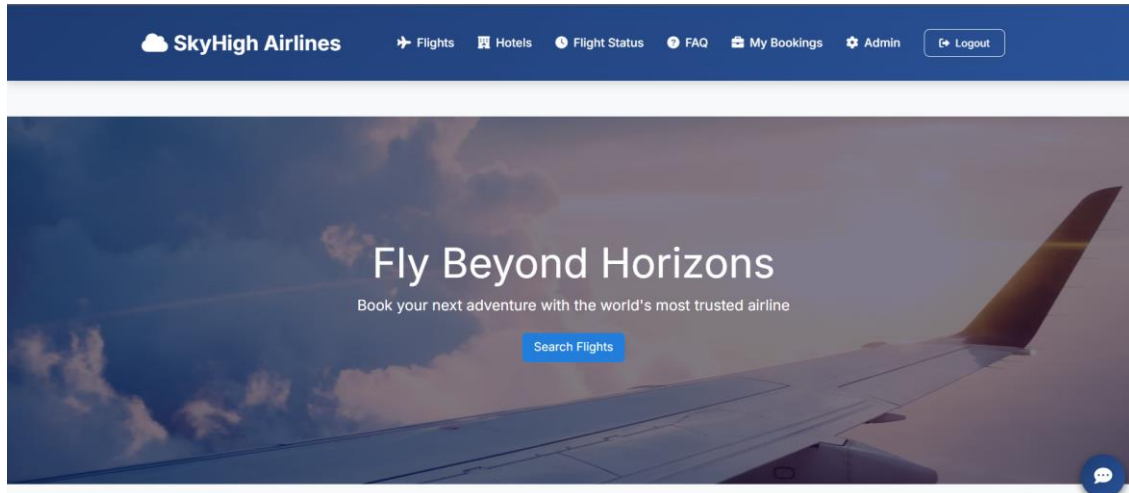


Figure 3.16: ER Diagram 2

3.3 User Interface Design

- **UI Wireframes/Mockups/Prototypes**

Home Page



Why Choose SkyHigh Airlines

Experience the difference with our premium service



★ 4.8/5

Over 2 million satisfied customers have rated our services as exceptional, making us the most trusted airline worldwide.



150+ Destinations

Fly to over 150 destinations across 6 continents with our extensive global network of premium routes.



24/7 Support

Our dedicated customer support team is available round the clock to assist you with any inquiries or requirements.

Join SkyHigh Rewards

Earn miles with every flight and enjoy exclusive benefits including priority boarding, lounge access, and free upgrades.

[Join Now](#)



What Our Passengers Say

★★★★★

"The service on my flight to Tokyo was exceptional. The cabin crew went above and beyond to make me comfortable."

Michael R. - Executive Traveler

★★★★★

"The in-flight entertainment and meal options were incredible. Made my 12-hour flight feel like a breeze!"


Sarah L. - Family Traveler

★★★★★

"The SkyHigh rewards program has given me incredible value. The points accumulate quickly and the redemption options are fantastic."

David T. - Frequent Flyer

Login Page



[✈ Flights](#)[🏨 Hotels](#)[🕒 Flight Status](#)[📄 FAQ](#)[👤 Login](#)

Please Sign in

You need to sign in first to continue

Username or Email

Password

[Sign in](#)

[Register](#)

User Dashboard

Welcome, yasin!

Upcoming Trips	Past Trips	Cancelled
0	0	1

Your Flight Bookings

Booking Ref	Flight	Route	Departure	Status	Actions
X9TRD458WP	DH106	DAC to CTG	May 23, 2025 10:45	Cancelled	View Ticket

Your Hotel Bookings

Booking ID	Hotel Name	Location	Check-in	Check-out	Rooms	Status	Actions
4	Sayeman	coxs bazar	May 20, 2025	May 21, 2025	1	Confirmed	Details

Admin Dashboard






Admin Dashboard

Users	Bookings	Flights	Revenue
8	6	10	\$240.00

Recent Bookings

#X9TRD458WP Flight: DH106 User: yasin	Cancelled
#1N3IC63NNJ Flight: DH107 User: jerin27	Confirmed
#OPKSYH6EUG Flight: DH106 User: jerin	Pending
#J1BIIHIVFU Flight: DH107 User: Shanta	Pending
#FW8NN0TQ3L Flight: DH107 User: jerin	Confirmed

Quick Actions

 Manage Flights
 Manage Bookings
 Manage Users
 Manage Hotels
 View Analytics

- Navigation Flow

For All Visitors (Unauthenticated Users):

pgsql

Home Page

- └─> Flight [Search](#) (Form)
 - └─> [Search](#) Results
 - └─> Flight Details (optional)
- └─> [Login](#)
- └─> Register
- └─> About / Contact (optional)

For Logged-in Users:

pgsql

User Dashboard

- └─> [View Profile](#) / [Update Info](#)
- └─> Book a Flight
 - └─> [Search](#) Flights
 - └─> [View](#) Details
 - └─> Confirm Booking
- └─> My Bookings
 - └─> [View](#) Booking Details
 - └─> Cancel Booking
 - └─> Download Ticket (optional)
- └─> Hotel Booking (if included)
 - └─> [Search](#) Hotels
 - └─> [View](#) Hotel Details
 - └─> Book / Cancel Hotel
- └─> Loyalty & Rewards
 - └─> [View](#) Points, Earn/Use Rewards
- └─> Logout

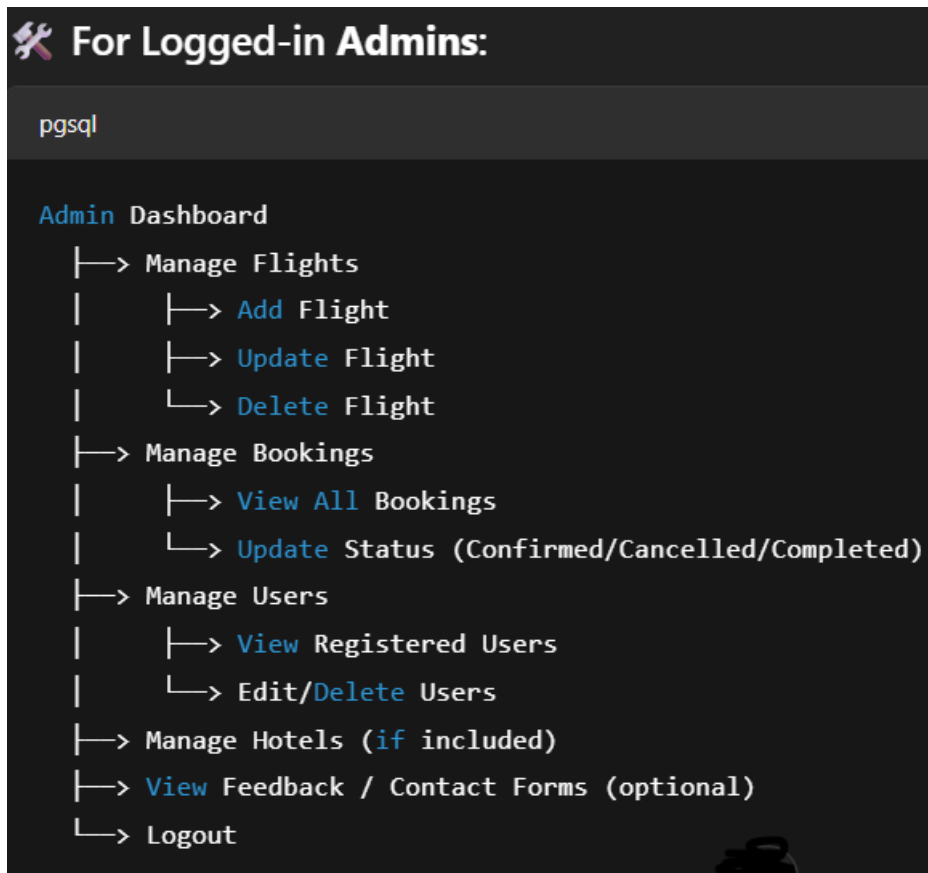


Figure 3.17: Navigation FLOW

- [Link of Figma](#)

<https://www.figma.com/design/amRKxtyWTdNuuOfuk4PBHN/Untitled?node-id=0-1&p=f&t=p8DfatxIAq3DTjjF-0>

Chapter 4. Implementation

- Code Structure Overview

```
/airline-system/           # Root directory of the project
|
├─ /admin/                 # Admin panel functionalities
|   ├── analytics.php      # View analytics dashboard (revenue, trends, stats)
|   ├── bookings.php       # Manage flight and hotel bookings
|   ├── dashboard.php      # Admin overview panel
|   ├── delete_flight.php  # Delete flight records
|   ├── delete_hotel.php   # Delete hotel records
|   ├── feedback.php       # View user feedback
|   ├── flight_status.php  # Check and manage flight status
|   ├── flights.php        # Add/edit flights
|   ├── hotels.php         # Add/edit hotels
|   ├── promo_codes.php    # Manage promo code offers
|   └─ users.php           # View and manage users
|
├─ /api/                   # Backend APIs
|   └─ check_promo.php     # Check validity of promo codes
|
├─ /assets/               # Static frontend assets
|   ├── /css/
|   |   └─ style.css      # Custom styling
|   ├── /js/
|   |   └─ main.js        # JavaScript functionality (AJAX, UI logic)
|   └─ /fonts/, /icons/  # Fonts and icons used in UI
|
├─ /auth/                 # User authentication system
|   ├── login.php         # Login page and logic
|   ├── register.php      # Registration form and processing
|   └─ logout.php         # Logout script
|
├─ /booking/              # Booking management logic
|   ├── cancel.php        # Cancel a booking
|   ├── confirm.php       # Confirm booking and generate ticket
|   ├── create.php        # Create a new hotel booking
|   ├── payment_.php      # Simulated payment gateway logic
|   └─ ticket.php         # Generate and view e-ticket
|
├─ /faq/                  # Frequently asked questions page
|   └─ faq.php
|
├─ /feedback/             # Passenger feedback submission
|   └─ submit.php
|
├─ /flights/              # Flight-related features
|   ├── multi_city.php    # Search and manage multi-city flights
|   ├── price_prediction.php # Predict price trends (if implemented)
|   └─ search.php         # Flight search functionality
|
```

```

├─ /hotels/                                # Hotel booking functionalities
│   ├── book.php                          # Book hotel room
│   ├── booking_details.php               # View details of bookings
│   ├── cancel.php                        # Cancel hotel booking
│   ├── cancel_booking.php                # Process cancellation
│   ├── confirm_booking.php               # Confirm hotel booking
│   ├── create.php                        # Add new hotel booking
│   ├── payment.php                       # Handle hotel payment
│   └─ search.php                         # Search hotels by destination
│
├─ /includes/                             # Common reusable components and config
│   ├── airports.php                     # Airport data list
│   ├── config.php                       # Configuration variables
│   ├── db_connect.php                   # Database connection logic
│   ├── header.php                       # HTML header template
│   ├── footer.php                       # HTML footer template
│   ├── auth.php                         # Session and access control
│   ├── functions.php                    # Helper functions
│   └─ hotels.sql                        # Sample SQL file for hotel data
│
├─ /user/                                 # Passenger user dashboard
│   ├── dashboard.php                    # User dashboard overview
│   ├── loyalty.php                      # View loyalty and reward points
│   └─ profile.php                       # Profile and booking management
│
├─ database.sql                           # Complete SQL dump of the database
├─ index.php                             # Landing (home) page
└─ status.php                            # Flight status checker (public)

```

Figure 4.1: Code Structure Overview

- **GitHub Repository URL**

<https://github.com/Jerinanan27/CSE412-Project.git>

Chapter 5. Software Testing

5.1 White Box Testing

- **Unit Testing on registration.php class**

Control flow graph

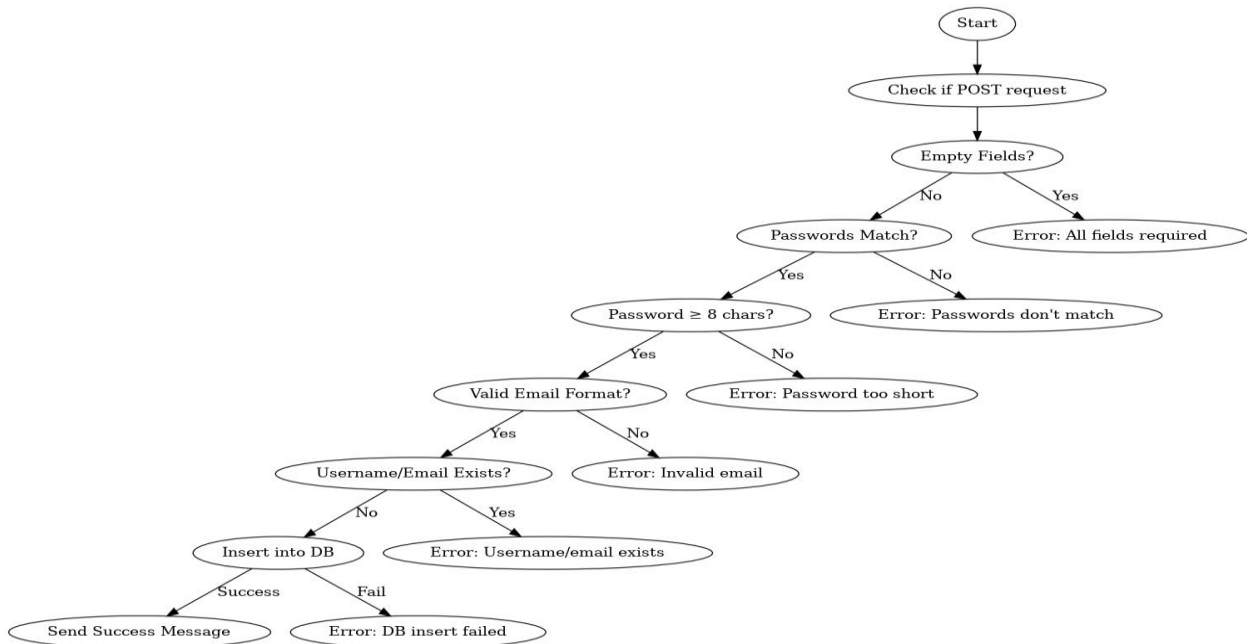


Figure 5.1: Control flow graph of registration class

Cyclomatic Complexity = 6 (6 decision points = 6 paths to test)

Table 2: Test case for registration class

Test Case ID	Input Conditions	Expected Output	Pass/fail
TC_REG_01	Empty username/email/password	"All fields are required"	pass
TC_REG_02	Password != Confirm	"Passwords do not match"	pass
TC_REG_03	Password < 8 chars	"Password must be at least 8 characters"	pass
Test Case ID	Input Conditions	Expected Output	Pass/fail

TC_REG_04	Invalid email	"Invalid email format"	pass
TC_REG_05	Existing username/email	"Username or email already exists"	pass
TC_REG_06	Valid input, unique user	"Registration successful!"	pass

- **Unit Testing on login class**

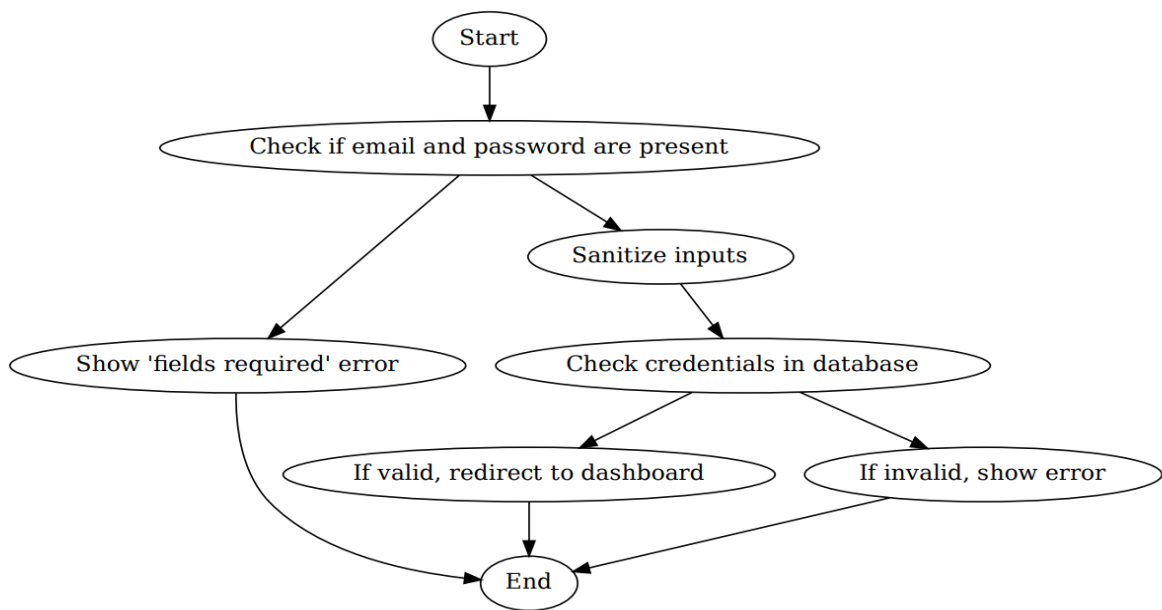


Figure 5.2: Control flow graph of login class

```

nodes = {
    "L1": "Start",
    "L2": "Check if email and password are present",
    "L3": "Show 'fields required' error",
    "L4": "Sanitize inputs",
    "L5": "Check credentials in database",
    "L6": "If valid, redirect to dashboard",
    "L7": "If invalid, show error",

```

"L8": "End"

}

Cyclomatic Complexity

- **Formula:** $M = E - N + 2P = 9 - 8 + 2*1 = 3$ linearly independent paths exist.
- Independent Paths (Basis Paths)
 1. **Path 1:** L1 → L2 → L3 → L8 (*Fields are missing*)
 2. **Path 2:** L1 → L2 → L4 → L5 → L6 → L8 (*Valid credentials — login success*)
 3. **Path 3:** L1 → L2 → L4 → L5 → L7 → L8 (*Invalid credentials — login fails*)

Table 3: Test case for login class

Test Case ID	Path Covered	Test Scenario	Input	Expected Output	Pass/fail
TC-L1	L1 → L2 → L3 → L8	Submit empty form	Email = " " Password = " "	Show “Fields required” error	pass
TC-L2	L1 → L2 → L4 → L5 → L6 → L8	Valid credentials	Email = "shanta@gmail.com" Password = "*****"(correct)	Redirect to dashboard	pass
TC-L3	L1 → L2 → L4 → L5 → L7 → L8	Invalid credentials	Email = "shanta@gmail.com" Password = "*****"(wrong)	Show “Invalid credentials” error	pass

- **Unit testing for flight booking:**

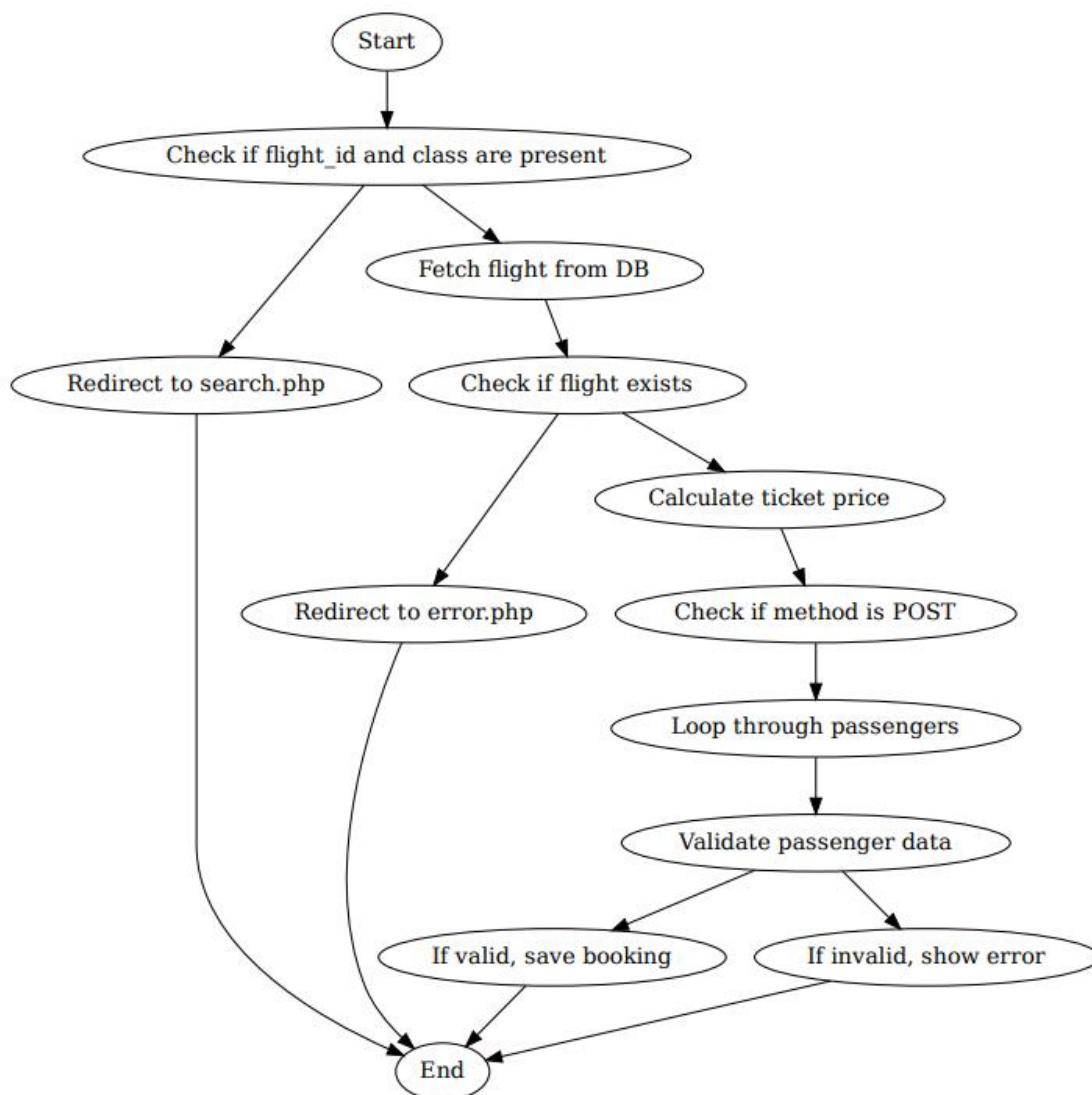


Figure 5.3: Control flow graph of flight_booking class

Control flow graph for flight booking:

```

nodes_booking = {
    "B1": "Start",
    "B2": "Check if flight_id and class are present",
    "B3": "Redirect to search.php",
    "B4": "Fetch flight from DB",
    "B5": "Check if flight exists",

```

```

"B6": "Redirect to error.php",
"B7": "Calculate ticket price",
"B8": "Check if method is POST",
"B9": "Loop through passengers",
"B10": "Validate passenger data",
"B11": "If valid, save booking",
"B12": "If invalid, show error",
"B13": "End"
}

```

Table 4: Test case for flight booking class

Test Case ID	Path Covered	Scenario	Input	Expected Output	pass/fail
TC-B1	B1 → B2 → B3 → B13	Missing flight ID or class	no flight_id or class in query params	Redirected to search.php	pass
TC-B2	B1 → B2 → B4 → B5 → B6 → B13	Invalid flight ID	flight_id=99999 class=economy (non-existent flight ID)	Redirected to error.php	pass
TC-B3	B1 → B2 → B4 → B5 → B7 → B8 → B9 → B10 → B11 → B13	Valid booking	Valid flight_id and class, POST method with valid passenger data (e.g., name, age, passport)	Booking saved, confirmation shown	pass
TC-B4	B1 → B2 → B4 → B5 → B7 → B8 → B9 → B10 → B12 → B13	Invalid passenger data	Valid flight_id, POST method with empty passenger name or invalid age	Error shown near invalid input fields	pass

5.2 Black Box Testing

Component: Booking Process (testing using boundary value analysis)

Number of Passengers (Adult)

Assumptions: Min: 1 adult Max: 9 passengers per booking

Table 5: Test case for number of passenger validation

Test Case ID	Component	Class	Input (No. of Adults)	Expected Output	Actual Output	Pass/Fail
TC-BK-BVA-01	Booking	Valid Min	1	Booking proceeds	As expected	Pass
TC-BK-BVA-02	Booking	Below Min	0	Error: At least 1 adult required	Notification of fill up	Pass
TC-BK-BVA-03	Booking	Valid Max	9	Booking proceeds	As expected	Pass
TC-BK-BVA-04	Booking	Above Max	10	Error: Too many passengers	Maximum passengers per booking	Pass
TC-BK-BVA-05	Booking	Just Below Max	8	Booking proceeds	As expected	Pass

Component: Payment Amount

Assumptions: Minimum booking amount = \$1 Maximum = \$10,000

Table 6: Test case for payment amount validation

Test Case ID	Component	Class	Input Amount (\$)	Expected Output	Actual Output	Pass/Fail
TC-PY-BVA-01	Payment	Valid Min	\$1	Payment proceeds	As expected	Pass
TC-PY-BVA-02	Payment	Below Min	\$0	Error: Invalid amount	As expected	Pass
TC-PY-BVA-03	Payment	Valid Max	\$10,000	Payment proceeds	As expected	Pass
TC-PY-BVA-04	Payment	Above Max	\$10,001	Error: Exceeds max limit	As expected	Pass
TC-PY-BVA-05	Payment	Just Below Max	\$9,999	Payment proceeds	As expected	Pass

Component admin features: New Hotel/Flight – Name Length

Assumption: Hotel/Flight name must be between 3 to 100 characters.

Table 7: Test case for name length of hotel/flight

Test Case ID	Component	Class	Input (Characters)	Expected Output	Actual Output	Pass/Fail
TC-AD-BVA-01	Add Hotel/Flight	Valid Min	3	Name accepted	As expected	Pass
TC-AD-BVA-02	Add Hotel/Flight	Below Min	2	Error: Name too short	As expected	Pass
TC-AD-BVA-03	Add Hotel/Flight	Valid Max	100	Name accepted	As expected	Pass

Test Case ID	Component	Class	Input (Characters)	Expected Output	Actual Output	Pass/Fail
TC-AD-BVA-04	Add Hotel/Flight	Above Max	101	Error: Name too long	As expected	Pass
TC-AD-BVA-05	Add Hotel/Flight	Just Below Max	99	Name accepted	As expected	Pass

5.3 Bug Detection and Solution

Bug 1: A deprecation warning appears during flight search

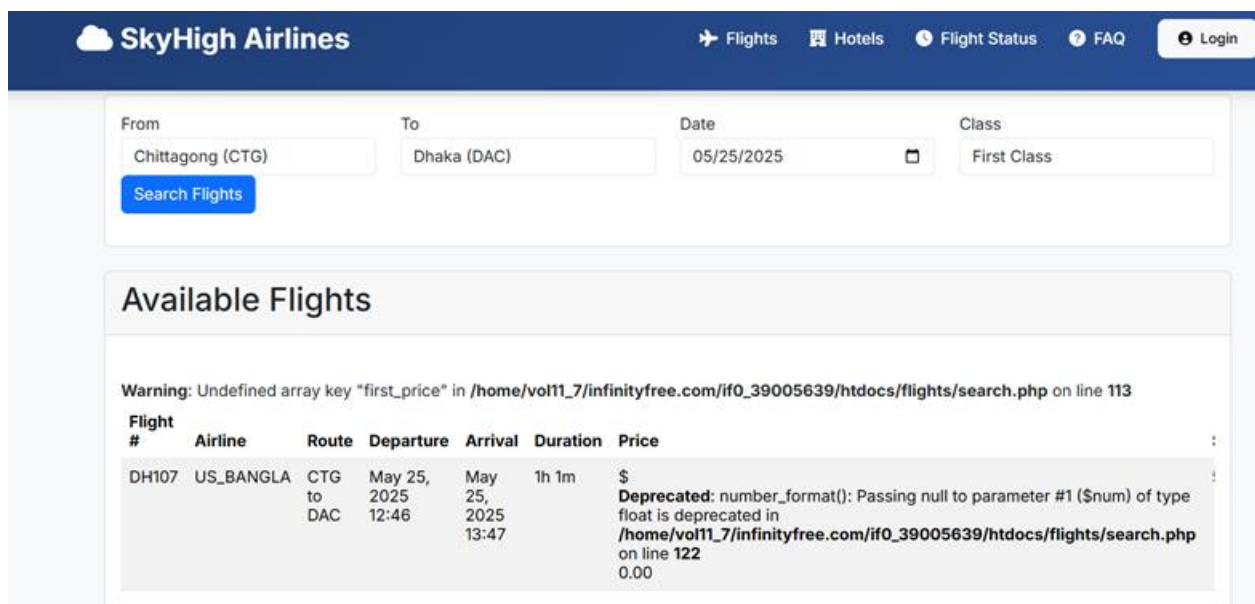


figure 5.4: Bug-1

Identification:

This occurs because the \$price variable, which is retrieved using-

```
$price = $flight[$class . '_price'];
```

may be null if the corresponding price (economy_price, business_price, or first_price) is not set in the database for a particular flight. Passing null to number_format() is deprecated in newer PHP versions.

Solution:

Used the null coalescing operator (??) to provide a default value (e.g., 0.0) in case the price is not set.

Updated Code:

```
$price = $flight[$class . '_price'] ?? 0.0;
```

This ensured \$price is always a float value, avoiding the deprecated behavior and ensuring the number_format() function works correctly:

```
<td>${?= number_format($price, 2) ?}</td>
```

Retesting:

Available Flights									
Flight #	Airline	Route	Departure	Arrival	Duration	Price	Seats	Status	Action
DH106	NOVO_AIR	DAC to CTG	May 23, 2025 10:45	May 24, 2025 01:45	15h 0m	\$0.00	58	Scheduled	Book Now
SH2025	USBangla	DAC to CTG	May 23, 2025 16:10	May 23, 2025 18:10	2h 0m	\$0.00	120	Scheduled	Book Now

figure 5.5: Retesting-1

Bug 2: The 'status' column was not seen

Recent Bookings	Recent Bookings
<div>#J1BIIHIVFU Flight: DH107 User: Shanta</div> <div>#FW8NN0TQ3L Flight: DH107 User: jerin</div> <div>#H1TOR5LBW1 Flight: DH106 User: user1</div>	<div>#J1BIIHIVFU Flight: DH107 User: Shanta</div> <div>#FW8NN0TQ3L Flight: DH107 User: jerin</div> <div>#H1TOR5LBW1 Flight: DH106 User: user1</div>

Bookings								
Reference	User	Flight	Class	Passengers	Price	Status	Created	Actions
J1BIIHIVFU	Shanta	DH107	Business	1	\$70.00	Pending	2025-05-19 13:04	Pending ▾
FW8NN0TQ3L	jerin	DH107	Business	2	\$140.00	Confirmed	2025-05-19 12:38	Confirmed ▾
H1TOR5LBW1	user1	DH106	Economy	1	\$50.00	Confirmed	2025-05-19 11:58	Confirmed ▾

figure 5.6: Bug-2

Identification:

Table 8: Bug Description

Bug	Description
1. Missing or ineffective CSS classes	The HTML used class-based styling (badge-success, etc.) but no matching CSS existed.
2. Poor contrast for some statuses	badge-warning (yellow) had white text, which was invisible or hard to read on a white background.
3. Dependency on external CSS frameworks	Using classes assumes Bootstrap or similar is included, which may not be the case in your pure HTML/CSS setup.

Solution:

Used **inline styles** to remove dependency on external CSS and ensure visibility on all backgrounds.

```
<span style="
    display: inline-block;
    padding: 5px 10px;
    border-radius: 12px;
    font-size: 0.9em;
    color: white;
    background-color:
        <?php
            echo $booking['status'] == 'confirmed' ? '#28a745' :    // Green
              ($booking['status'] == 'cancelled' ? '#dc3545' :    // Red
                ($booking['status'] == 'completed' ? '#17a2b8' :  // Blue
                  '#ffc107'));                                     // Yellow (default)
        ?>
    <?php
        // Make text black if background is yellow (warning)
        if (!in_array($booking['status'], ['confirmed', 'cancelled', 'completed'])) {
            echo 'color: black;';
        }
    ?>
">
    <?php echo ucfirst($booking['status']); ?>
</span>
```

Retesting:

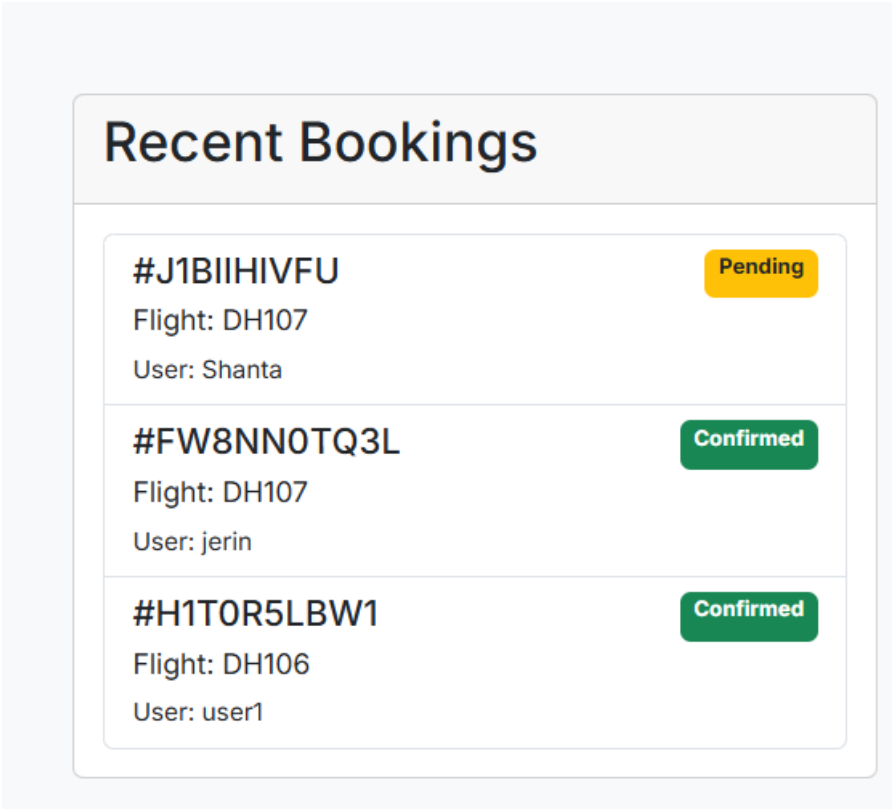


figure 5.7: Retesting -2

Bug-3: Past Dates Were Selectable

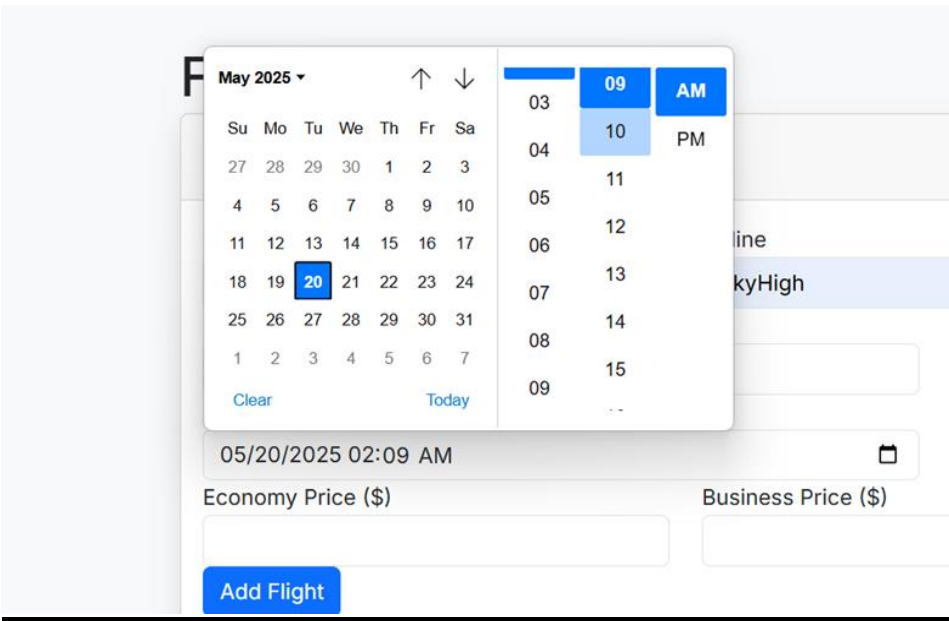


figure 5.8: Bug -3

Identification:

The `departure_time` and `arrival_time` fields used `<input type="datetime-local">` without restricting the user from selecting past dates/times.

So admins could set a flight's departure or arrival time in the past, which is invalid and could corrupt schedule integrity.

Solution:

We added the `min` attribute to the `departure_time` input and dynamically set it to the current time using PHP and JavaScript.

```
php
<input type="datetime-local" name="departure_time" id="departure_time" class="form-control"
      min="<?= date('Y-m-d\TH:i') ?>" required>

js
const now = new Date().toISOString().slice(0, 16);
departureInput.min = now;
```

Retesting:

Departure Time

mm/dd/yyyy --:-- --

May 2025

Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Clear Today

03	33	AM
04	34	PM
05	35	
06	36	
07	37	
08	38	
09	39	

Search

figure 5.9: Retesting -3

Bug 4: Arrival Time Could Be Earlier Than Departure Time

The screenshot shows the 'Flight Management' section of the SkyHigh Airlines website. The 'Add New Flight' form includes fields for Flight Number (SH2023), Airline (NOVO_Air), Departure Airport (New York (JFK)), and Departure Time (05/21/2025 04:05 AM). A date picker is open, showing the month of May 2025. The date 12 is selected, and the time 05:05 AM is chosen. Below the date picker, there are input fields for Economy Price (\$), Business Price (\$), and First Class Price (\$). An 'Add Flight' button is at the bottom left of the form.

figure 5.10: Bug -4

Identification:

The system allowed users to enter an arrival_time that was earlier than the selected departure_time. Logically invalid flight schedule could be submitted, which breaks the core functionality of flight management.

Solution:

We implemented JavaScript that:

- Dynamically sets the minimum allowable arrival time to the selected departure time.
- Validates on form submission to prevent bypassing via browser manipulation.

```
departureInput.addEventListener('change', () => {
  arrivalInput.min = departureInput.value;
  if (arrivalInput.value < departureInput.value) {
    arrivalInput.value = departureInput.value; // auto-correct
  }
});

document.querySelector('form').addEventListener('submit', function (e) {
  if (arrivalInput.value < departureInput.value) {
    e.preventDefault();
    alert('Arrival time must be after departure time.');
```

Retesting:

Departure Time

05/22/2025 02:40 AM

📅

Arrival Time

05/22/2025 02:40 AM

📅

Economy Price (\$)

Business Price (\$)

Add Flight

Existing Flights

Flight #	Airline	Route	Departure
----------	---------	-------	-----------

May 2025

↑

↓

Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Clear

Today

03	40	AM
04	41	PM
05	42	
06	43	
07	44	
08	45	
09	46	

figure 5.11: Retesting -4

Bug-5:

SkyHigh Airlines

FlightsHotelsFlight StatusFAQMy BookingsLogout

000

Your Flight Bookings

Booking Ref	Flight	Route	Departure	Status	Actions
J1BIIHIVFU	DH107	CTG to DAC	May 25, 2025 12:46	Pending	View Ticket

Your Hotel Bookings

Booking ID	Hotel Name	Location	Check-in	Check-out	Rooms	Status	Actions
2	Grand Central Hotel	New York City, USA	May 19, 2025	May 20, 2025	1	Confirmed	Details

SkyHigh Airlines

FlightsHotelsFlight StatusFAQMy BookingsLogout

Error loading booking details: SQLSTATE[HY000]: General error: 1267 Illegal mix of collations (latin1_swedish_ci,IMPLICIT) and (utf8mb4_general_ci,COERCIBLE) for operation 'ifnull'

Back to Dashboard

ABOUT SKYHIGH AIRLINES

BOOK

HELP

NEWSLETTER

figure 5.12: Bug -5

Couldn't Fix it.

Chapter 6. Deployment

- **Deployment Platform**

The Airline Management System was deployed using the following tools and services:

- **InfinityFree:** A free web hosting platform used to host the application online. It supports PHP and MySQL, making it suitable for this project. It provides features like a custom domain, file storage, and database hosting without any cost, which is ideal for student and demo projects.
- **FileZilla:** A free and open-source FTP (File Transfer Protocol) client used to upload project files from the local system to the InfinityFree hosting server. It provides a simple drag-and-drop interface and secure file transfers over FTP or SFTP.

- **Deployment Process**

The deployment process involved the following steps:

1. **Local Development & Testing:**

- The system was first developed and tested locally using **XAMPP**, which includes Apache and MySQL for running the PHP code and database locally.
- All functionalities, including booking, admin panel, and user authentication, were tested to ensure error-free operation.

2. **Project Directory Preparation:**

- The entire project folder was organized with appropriate directory structure.
- Database file (database.sql) was exported from phpMyAdmin in XAMPP.

3. **Hosting Account Setup on InfinityFree:**

- An account was created on <https://infinityfree.net>.
- A free subdomain and hosting account were set up.
- A MySQL database was created via the cPanel provided by InfinityFree.

4. Uploading Files Using FileZilla:

- FTP credentials (host, username, password) were obtained from InfinityFree.
- **FileZilla** was used to connect to the web server and upload all files and folders (excluding database.sql) into the /htdocs directory.

5. Database Import:

- Using the **phpMyAdmin** interface provided by InfinityFree, the database.sql file was imported to create necessary tables and insert initial data.
- The db_connect.php file was updated with the new database credentials provided by InfinityFree.

6. Testing Online Version:

- After uploading and configuring everything, the website was tested on the live URL to ensure that all features—like login, booking, flight status check, and admin functionalities—worked as expected.

7. Final Adjustments:

- Any broken links, permission issues, or database connection errors were resolved.
- Styling and responsiveness were verified to ensure the application worked across different devices.

- **Website URL**

<https://skyhigh.kesug.com>

Chapter 7. Conclusion

The Airline Management System project successfully achieved its core objectives of providing a functional, interactive, and user-friendly web application for both passengers and administrators. The system allows users to search and book flights and hotels, view flight statuses, manage bookings, and securely register/login. On the other hand, administrators can manage flights, hotels, users, and view system analytics via a dedicated dashboard. The deployment of the system on a live server using InfinityFree demonstrated practical knowledge of full-stack web development, hosting, and project deployment.

- **Learnings**

During the development of this project, several key skills and concepts were learned and applied:

- **Frontend Development:** Building responsive and visually appealing interfaces using HTML, CSS, JavaScript, and Bootstrap.
- **Backend Development:** Writing server-side logic using PHP and integrating it with the MySQL database.
- **AJAX & jQuery:** Implementing asynchronous requests for smoother user experience without full-page reloads.
- **Authentication & Security:** Creating a secure login and registration system with session management.
- **Database Management:** Designing and querying relational databases using MySQL.
- **Version Control & Collaboration:** Using GitHub for version control and collaboration.
- **Deployment & Hosting:** Gaining hands-on experience with web deployment using InfinityFree and FileZilla.
- **UI/UX Design:** Prototyping interfaces using Figma to improve usability.

- **Limitations**

Despite its functionality, the system has a few limitations:

- **No Real Payment Gateway:** Payment integration is simulated and not connected to real banking systems.
- **Basic Chatbot:** The chatbot is dummy-based and does not use AI or dynamic responses.
- **Limited Error Handling:** Some server-side validations and user feedback messages could be more robust.
- **No Email/SMS Integration:** Users do not receive email or SMS confirmations after booking.
- **Scalability:** The system is not optimized for large-scale commercial deployment.

- **Future Plan**

To enhance and expand the system in future iterations, the following improvements are planned:

- **Integrate Real Payment Gateways** (e.g., Stripe, PayPal) for secure and reliable transactions.
- **AI-Based Chatbot:** Replace the dummy chatbot with an AI-driven assistant using NLP for better user support.
- **Mobile App Version:** Develop a mobile application version for easier access and improved user experience.
- **Email & SMS Notifications:** Implement automatic notifications for bookings and cancellations.
- **Advanced Analytics:** Enhance the admin dashboard with graphical insights and trend analysis using charts.
- **Role-Based Access Control:** Improve security by assigning granular roles and permissions.