Software Design Specifications

Airline Reservation System

Version: [4.0]

Project Code	SDA001
Supervisor	Ms. Syeda Rubab Jaffar
Co Supervisor	N/A
Project Team	Jahanzaib Hussain Mirza– 23k-3011 Qasim Ali– 23k-3002 Maaz Khan– 23k-3036
Submission Date	6-May-25

Document History

Version	Name of Person	Date	Description of change
1.0	Jahanzaib Mirza	2025-05-01	Created base document using SDS template, added title page and TOC
1.5	Muhammad Maaz Khan	2025-05-02	Wrote introduction, design goals, and scope of system
2.0	Qasim Ali	2025-05-03	Developed architecture design section with system models
2.5	Jahanzaib Mirza	2025-05-04	Added data design and database schema diagrams
3.0	Muhammad Maaz Khan	2025-05-05	Completed interface and component design sections
3.5	Jahanzaib Mirza	2025-05-06	Finalized System Behavior section; created Sequence and Activity Diagrams
4.0	Qasim Ali	2025-05-07	Performed final edits, formatting, and document validation for submission

Distribution List

Name	Role
Ms. Syeda Rubab Jaffar	Supervisor

Document Sign-Off

Version	Sign-off Authority	Project Role	Signature	Sign-off Date

Document Information

Category	Information
Customer	FAST-NUCES
Project	Airline Reservation System
Document	Software Design Specification
Document Version	1.0
Status	Draft
Author(s)	Jahanzaib Mirza & Qasim Ali
Approver(s)	Ms. Syeda Rubab Jaffar
Issue Date	6-May-25
Document Location	
Distribution	Advisor Project Coordinator’s Office (through Advisor)

Definition of Terms, Acronyms and Abbreviations

Term	Description
API	Application Programming Interface – allows software components to interact
RESTful	Representational State Transfer – architectural style for stateless APIs
SQL	Structured Query Language – used to manage data in relational databases
JWT	JSON Web Token – a compact, secure method for transmitting user authentication
HTTP	Hypertext Transfer Protocol – protocol for communication between client/server
UI	User Interface – the visual elements the user interacts with
IEEE	An international organization that sets standards for electrical, electronics, and computing technologies, including software engineering best practices.
GDPR	A data protection law in the European Union that regulates how personal data is collected, processed, and stored, with a focus on user privacy and consent.
PCI DSS	A global standard designed to ensure secure handling of credit/debit card

	information by organizations that process, store, or transmit card data.
JDBC	An API in Java that allows applications to connect and interact with relational databases using SQL queries.
JPA	A Java specification that simplifies database access by allowing developers to map Java objects to database tables (ORM – Object Relational Mapping).

Table of Contents

1 Introduction

1.1 Purpose of Document

This Software Design Specification (SDS) describes the design and architecture of the Airline Reservation System. It serves as a comprehensive guide for the development team to implement the system and for quality assurance teams to validate its functionality. The document defines the modules, components, data flow, and interactions between subsystems while adhering to modern software engineering principles

1.2 Intended Audience

This document is intended for the following stakeholders:

- **Developers**: To understand system components, architecture, and inter-component communication.
- **Project Supervisors & Advisors**: To monitor project compliance with design goals and academic requirements.
- **Testers**: To validate the design against functional and non-functional requirements.
- **Clients (Airline companies)**: For conceptual understanding of the system's design and capability.
- **Students and Reviewers**: As a reference for system architecture and software design techniques.

1.3 Document Convention

This document follows these conventions:

- **Font**: Arial, 12pt
- **Diagrams**: Labeled and referenced using figure numbers
- **Code snippets**: When present, formatted using monospace font
- **References**: Cited in IEEE format

1.4 Project Overview

The Airline Reservation System is a robust, web-based platform that offers a full suite of tools for booking and managing airline operations. It automates common airline functions such as flight creation, booking, seat allocation, fare calculation, and invoice generation. It also provides an administrative backend with business insights, user management, and real-time flight status updates.

The goal is to replace traditional, manual reservation systems with a modern, scalable platform that improves operational efficiency, ensures data accuracy, and delivers an enhanced customer experience.

1.5 Scope

This system will cover:

- User account creation and authentication
- Real-time flight listings with search, filter, and seat selection features
- Dynamic fare calculation based on class, availability, and distance
- Payment gateway integration for bookings and refunds
- An administrative dashboard with analytics for occupancy and revenue

Not included:

- Integration with external flight systems (e.g., real-time airport data feeds)
- Mobile application version
- International compliance layers (e.g., GDPR, PCI DSS) beyond basic web security

2 Design Considerations

This section highlights the critical design assumptions, dependencies, risks, and volatile areas that could impact the system's structure, implementation, and performance. It aims to set the groundwork for a robust and adaptable system architecture.

2.1 Assumptions and Dependencies

 The platform will be deployed and accessed via modern browsers (Chrome, Firefox).

 React.js and Spring Boot must be properly configured for seamless front-end and back-end communication.

 The system will run on a local or university-hosted server.

 The payment system will simulate real payments using mock APIs.

2.2 Risks and Volatile Areas

 Security Risks: Handling user data and payment information requires proper encryption and token management (e.g., JWT for sessions).

 Third-party dependencies: Reliance on external libraries or APIs may cause compatibility issues or introduce bugs.

 Scalability Concerns: Initial design assumes a small user base (academic use); scaling to production-level traffic would require additional architecture planning.

 Team Coordination: Frontend and backend modules are developed in parallel, increasing the risk of integration mismatches.

- **Timeline Constraints**: Due to academic semester limits, certain features like mobile responsiveness or cross-browser testing may be deprioritized.

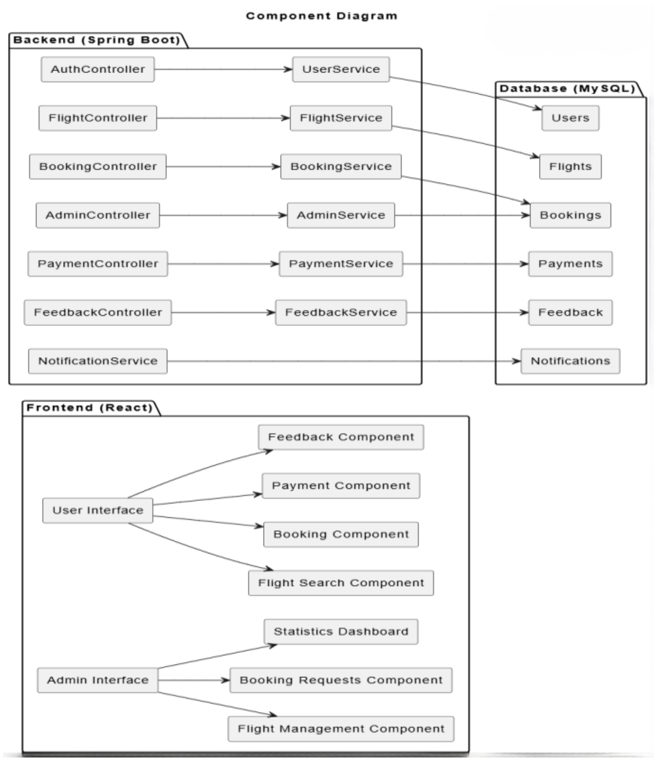
3 System Architecture

This section outlines the overall structure of the Vehicle Rental System, highlighting how system components interact, how responsibilities are distributed, and how the layers support modular development and scalability.

3.1 System Level Architecture

The Airline Reservation System is designed using a **modular, service-oriented architecture (SOA)** to promote scalability, flexibility, and ease of maintenance. The system is logically divided into three main layers:

- **Client Layer (Front-End)**: Responsible for all user interactions. Implemented using React.js, it provides dynamic and responsive interfaces for both end-users (passengers) and administrators.
- **Business Logic Layer (Back-End)**: This layer encapsulates all the core functionality such as user authentication, flight scheduling, booking logic, and report generation. Spring Boot is used here for RESTful services and business processing.
- **Data Layer (Database)**: SQL-based relational database designed to store user information, flight details, reservation records, payment logs, and more. It emphasizes referential integrity, indexing for performance, and normalization.

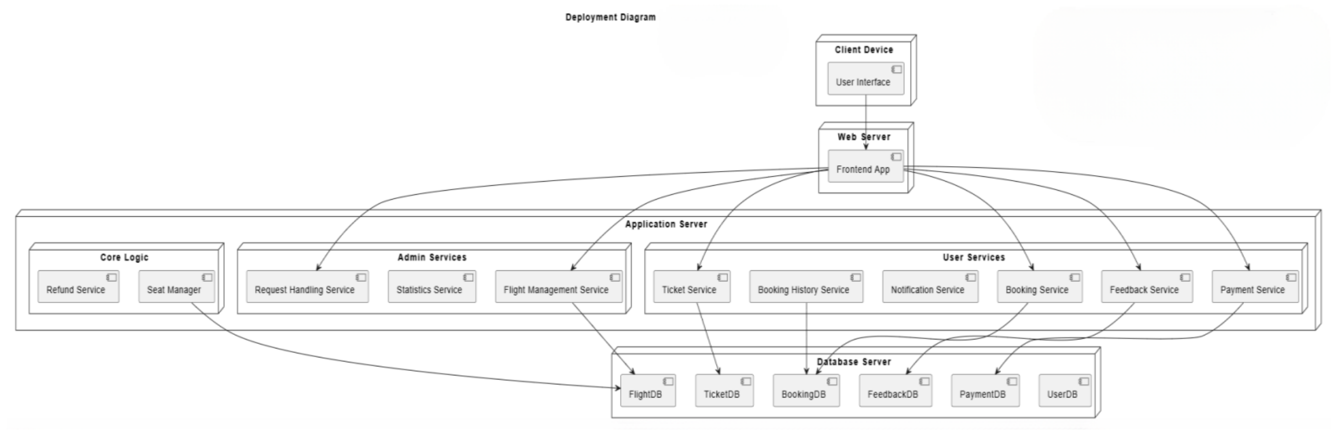


3.2 Software Architecture

- The system follows a **Layered (N-Tier) Architecture**:

- **Presentation Layer**: Manages user input/output, form validations, and API requests to the backend. React.js components are structured based on user roles (passenger, admin).

- **Application Layer**: Handles logic such as seat availability checks, fare calculations, and validation before interacting with the database. This ensures the separation of business rules from presentation.
- **Persistence Layer**: Manages interaction with the SQL database using JPA/Hibernate or JDBC. Includes Data Access Objects (DAOs) for modularity.
- **Security Layer**: Ensures secure communication using HTTPS, token-based authentication (e.g., JWT), role-based access control, and basic encryption for sensitive data.



*

4 Design Strategy

- The design is guided by key software engineering principles:

4.1 Modularity and Separation of Concerns

- Each component is encapsulated to handle a specific concern. For example, authentication logic is managed separately from booking or payment logic. This increases maintainability and debugging efficiency.

4.2 Extensibility and Scalability

- The modular design supports adding new features such as loyalty programs, multilingual support, or third-party integrations with minimal disruption to existing components.

4.3 Usability and User-Centric Design

- The system is built with a focus on simplicity, accessibility, and responsiveness. React's component-based structure allows for dynamic updates and mobile-friendly design. Color schemes, layout, and navigational elements follow UI/UX best practices.

4.4 Data Security and Privacy

- Security is implemented at multiple levels:

- Token-based user authentication (JWT)
- Input sanitization to prevent injection attacks
- HTTPS for data transmission
- Secure password hashing (e.g., bcrypt or SHA-256)

4.5 Reusability and Maintainability

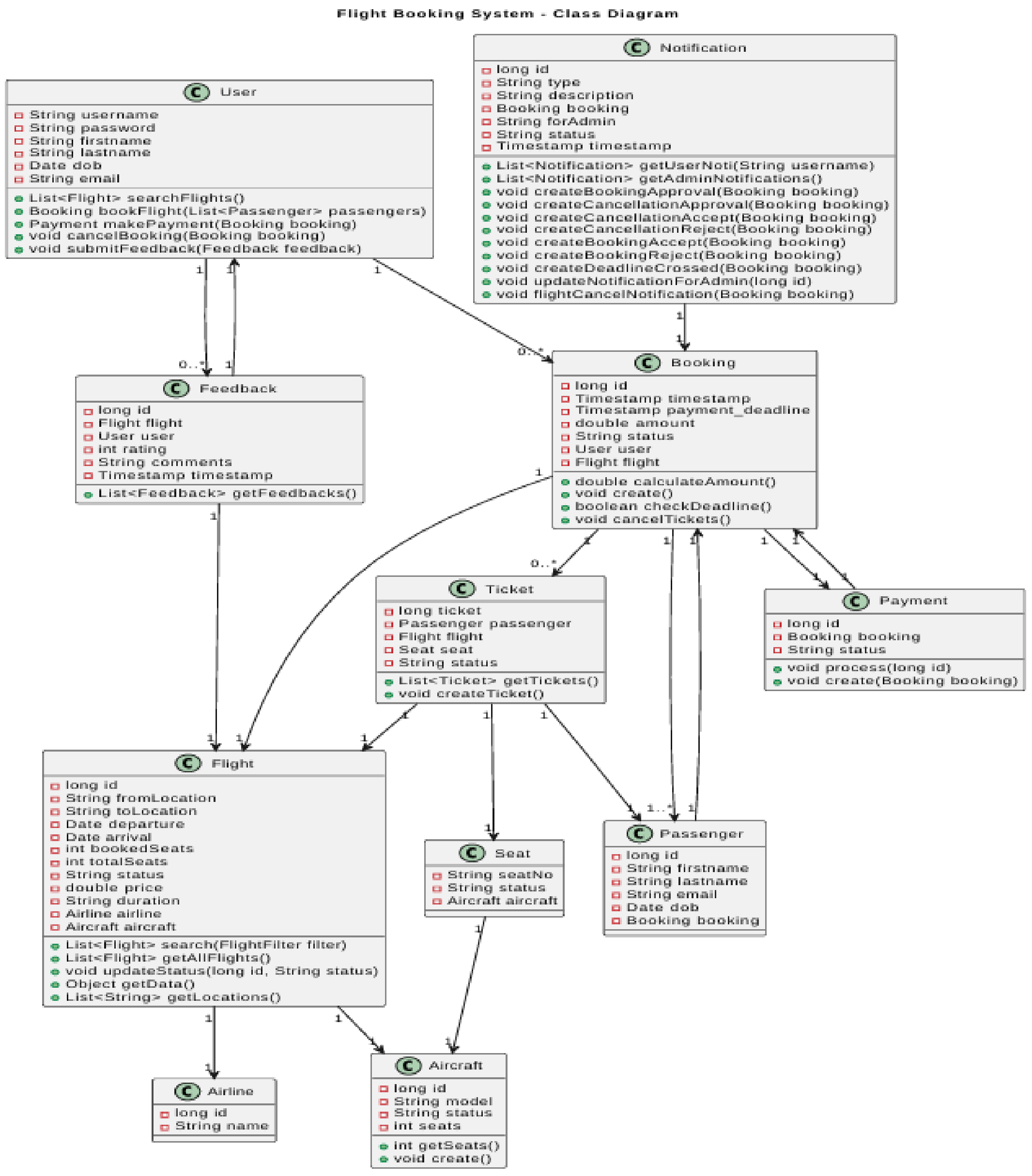
- All services (e.g., BookingService, FlightService, PaymentService) are designed as independent modules following the **Single Responsibility Principle (SRP)**. This enables developers to reuse components across different parts of the system or future applications.

4.6 Testing Strategy

- The system is designed with **testability** in mind:
  - Unit tests for business logic using JUnit
  - Front-end component testing using Jest
  - API testing using Postman or Swagger

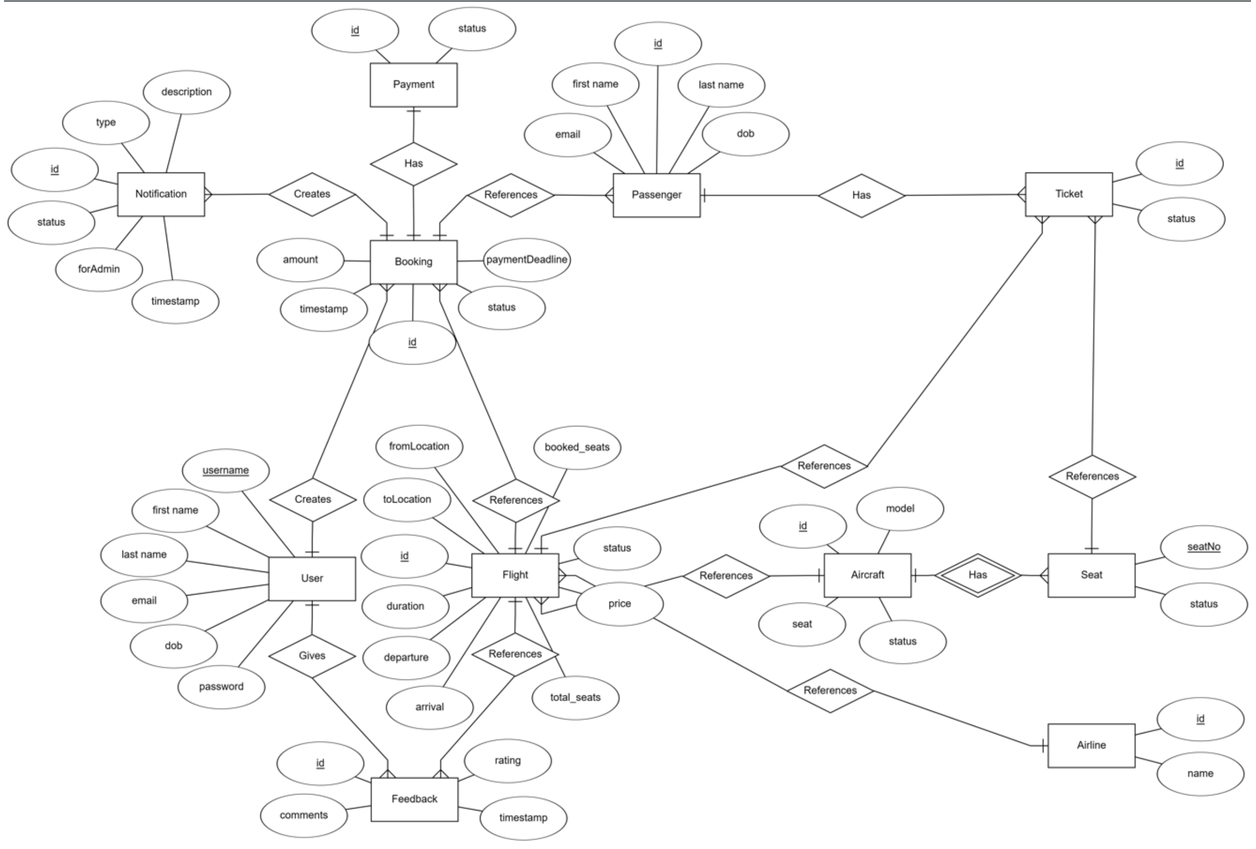
5 Detailed System Design

Class Diagram



5.1 Database Design

5.1.1 ER Diagram



Entity Descriptions

- **User**: Represents system users who manage or interact with the system. Contains personal information like name, email, date of birth, and login credentials.
- **Passenger**: Represents customers booking and using flights. Includes personal details necessary for ticket issuance and communication.
- **Flight**: Central entity representing scheduled airline journeys. Stores flight ID, departure and arrival information, duration, price, status, and seat allocation.
- **Aircraft**: Refers to the physical airplane assigned to a flight. Includes the aircraft model, total seats, and status.
- **Airline**: Represents the airline operating the aircraft. Includes an identifier and name.
- **Seat**: Defines individual seats within an aircraft. Attributes include seat number and status (e.g., available, reserved).
- **Ticket**: Contains booking confirmation details for a passenger, including seat assignment and status.
- **Booking**: Records the transaction between a passenger and a flight. Captures booking ID, status, amount, timestamp, and payment deadline.
- **Payment**: Logs payment details associated with a booking, including amount, status, and timestamps.
- **Notification**: Used to notify users or administrators about important system events. Contains type, description, status, and timestamp.
- **Feedback**: Allows passengers to provide ratings and comments about flights. Captures rating, comments, and timestamp.

Relationship Overview

- A **User** can create multiple **Flights** and can give **Feedback**.
- A **Passenger** can have multiple **Bookings** and **Tickets**.
- A **Booking** is associated with one **Flight** and one **Passenger**, and may generate **Payment** and **Notification** records.
- Each **Flight** is linked to exactly one **Aircraft**, and an **Aircraft** references a single **Airline**.
- A **Ticket** references a **Seat**, which is tied to an **Aircraft**.
- **Seat** details and availability are derived from their associated **Aircraft**.
- **Feedback** is submitted for a specific **Flight** by a **User**.
- **Notifications** can be created for any changes or alerts, especially tied to **Bookings**.
- A **Payment** record is generated for each **Booking**, with details about the transaction.

5.1.2 Data Dictionary

5.1.2.1 Data 1

< Data 1>	
Name	User
Alias	System User
Where-used/how-used	Stores user login credentials and roles for system access, enabling authentication and determining access privileges.
Content description	User = username + first_name + last_name + email + password_hash +dob.

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
username	*Unique identifier for the user*	*VARCHAR*	*255*	*NO*		*PK*
first_name	*First name of user*	*VARCHAR*	*255*	*NO*		
last_name	*Last name of user*	*VARCHAR*	*255*	*NO*		
email	*Email*	*VARCHAR*	*255*	*NO*		
password	*Password for account*	*VARCHAR*	*255*	*NO*		
dob	*Date of birth of user*	*DATETIME*	*6*	*NO*		

5.1.2.2 Data 2

< Data 2>	
Name	Passenger

Alias	Traveler
Where-used/how-used	Stores passenger information and is used for seat reservations and ticket generation.
Content description	Passenger = id + first_name + last_name + dob + email + booking_id

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
passenger_id	*Unique identifier for customer*	*BIGINT*		*NO*	*auto-increment*	*PK*
booking_id	*Reference to Booking table*	*BIGINT*		*NO*		*FK*
first_name	*First name of passenger*	*VARCHAR*	*255*	*NO*		
last_name	*Last name of passenger*	*VARCHAR*	*255*	*NO*		
email	*email*	*VARCHAR*	*255*	*NO*		
dob	*Date of birth of passenger*	*DATETIME*	*6*	*NO*		

5.1.2.3 Data 3

< Data 3>

Name	Aircrafts
Alias	Plane, AirVehicle
Where-used/how-used	Stores details of aircraft used for flights. Used in flight scheduling, seat availability, and capacity planning.
Content description	Aircraft = id + model + seats + status

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
id	*Unique identifier for aircraft*	*BIGINT*		*NO*	*auto-increment*	*PK*
model	*Model of aircraft*	*VARCHAR*	*255*	*NO*		
seats	*Seat capacity of aircraft*	*INTEGER*		*NO*		
status	*Availability status of aircraft (“Unassigned”/ “Assigned”)*	*VARCHAR*	*255*	*NO*	*“Unassigned”*	

5.1.2.4 Data 4

< Data 4>	
Name	Airline
Alias	Airline companies
Where-used/how-used	Stores information about airlines operating flights. Used to associate flights with airline operators and display branding information.
Content description	Airline = id + name

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
id	*Unique identifier for airline*	*BIGINT*		*NO*	*auto-increment*	*PK*
Name	*Airline name*	*VARCHAR*	*255*	*NO*		

5.1.2.5 Data 5

< Data 5>	
Name	Flight
Alias	Flight details
Where-used/how-used	Stores flight details. Used to manage airline operations, assign aircraft, and allow passengers to book specific flights.
Content description	Flight = id + airline_id + aircraft_id + from_location + to_location + departure + arrival + duration + status + booked_seats + total_seats + price

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
id	*Unique identifier for flight*	*BIGINT*		*NO*	*auto-increment*	*PK*
airline_id	*Reference to airline*	*BIGINT*		*NO*		*FK*
aircraft_id	*Reference to aircraft*	*BIGINT*		*NO*		*FK*
departure	*Date and time of flights departure*	*DATETIME*	*6*	*NO*		
arrival	*Date and time for flights*	*DATETIME*	*6*	*NO*		

	arrival					
status	*Flight status (“Scheduled”/ “Delayed”/ “Departed”/ “Cancelled”/ “Landed”)*	*VARCHAR*	*255*	*NO*	*“Scheduled”*	
price	*Booking price for a flight*	*DOUBLE*		*NO*		
total_seats	*Total seats in a flight*	*INTEGER*		*NO*		
booked_seats	*Total of booked and reserved seats*	*INTEGER*		*NO*	*0*	
duration	*Duration of flight*	*VARCHAR*	*255*	*NO*		
to_location	*Destination of flight*	*VARCHAR*	*255*	*NO*		
from_location	*Source of flight*	*VARCHAR*	*255*	*NO*		

5.1.2.6 Data 6

< Data 6>						
Name	Payment					
Alias	Booking Payment					
Where-used/how-used	Records payment status of bookings.					
Content description	Payment = id + booking_id + status					
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
id	*Unique ID for payment*	*BIGINT*		*NO*	*auto-increment*	*PK*
booking_id	*Reference to booking*	*BIGINT*		*NO*		*FK*
status	*Status (“Pending”/ “Successful”/ “Confirmed”/ “Refunded”/ “Not required”)*	*VARCHAR*	*255*	*NO*	*“Pending”*	

5.1.2.7 Data 7

< Data 7>	
Name	Feedbacks

Alias	Reviews for flights
Where-used/how-used	Stores reviews submitted by passengers about their flights. Used by the admin to evaluate and filter meaningful feedback, focusing on both positive and negative experiences.
Content description	Feedback = id + comments + rating + timestamp + flight_id + username

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
id	*Unique ID for feedback*	*BIGINT*		*NO*	*auto-increment*	*PK*
flight_id	*Reference to flight*	*BIGINT*		*NO*		*FK*
comments	*Feedback from user*	*VARCHAR*	*255*	*NO*		
rating	*Numeric rating for sentiment analysis. (Negative = -1, Neutral = 0, Positive = 1)*	*INTEGER*		*NO*		
timestamp	*Date and time of feedback submission*	*DATETIME*	*6*	*NO*	*CURRENT_DATE*	
username	*Username of user who submitted the feedback*	*VARCHAR*	*255*	*NO*		*FK*

5.1.2.8 Data 8

< Data 8>	
Name	Booking
Alias	Flight Booking
Where-used/how-used	Stores booking details for passengers. Used to track passenger reservations, payment status, and associated flight details.
Content description	Booking = id + amount + payment_deadline + timestamp + status + flight_id + username

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
id	*Unique ID for booking*	*BIGINT*		*NO*	*auto-increment*	*PK*
flight_id	*Reference to flight*	*BIGINT*		*NO*		*FK*

payment_deadline	*Date and time of payment deadline*	*DATETIME*	*6*	*NO*	*CURRENT_TIME + 1 DAY*	
timestamp	*Date and time of booking*	*DATETIME*	*6*	*NO*	*CURRENT_TIME*	
amount	*Amount to be paid*	*DOUBLE*		*NO*		
status	*Booking status (“Pending”/ “Waiting for approval”/ “Cancelled”/ “Approved”)*	*VARCHAR*	*255*	*NO*	*“Pending”*	
username	*Username of user who applied the booking.*	*VARCHAR*	*255*	*NO*		*FK*

5.1.2.9 Data 9

< Data 9>	
Name	Ticket
Alias	Flight Ticket
Where-used/how-used	Stores ticket details associated with bookings. Used to generate and track individual passenger tickets, including their status and specific travel details.
Content description	Ticket = ticket + status + flight_id + passenger_id + seat_no

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
ticket	*Unique identifier for ticket*	*BIGINT*		*NO*	*auto-increment*	*PK*
flight_id	*Reference to flight*	*BIGINT*		*NO*		*FK*
passenger_id	*Reference to passenger*	*BIGINT*		*NO*		*FK*
seat_no	*Reference to seat assigned to passenger*	*VARCHAR*	*255*	*NO*		*FK*
status	*Validity status (“Valid”/ “Invalid”)*	*VARCHAR*	*255*	*NO*	*“Valid”*	

5.1.2.10 Data 10

< Data 10>	
Name	Notification

Alias	Notifications
Where-used/how-used	Stores notification events triggered by flight status changes, booking updates, or special announcements. Enables alerting passengers and admins via in-app messages.
Content description	Notification = id + description + for_admin + status + timestamp + type + booking_id

Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
id	*Unique identifier for ticket*	*BIGINT*		*NO*	*auto-increment*	*PK*
description	*Notification message*	*VARCHAR*	*255*	*NO*		
for_admin	*Indicates if notification is for admin or not. ("Yes"/ "NO")*	*VARCHAR*	*255*	*NO*		
timestamp	*Date and time of notification generated*	*DATETIME*	*6*	*NO*		
status	*Visibility status ("Treated"/ "Untreated")*	*VARCHAR*	*255*	*NO*	*"Untreated"*	
type	*Status just for the UI of notifications ("Red"/ "Blue")*	*VARCHAR*	*255*	*NO*		
Booking_id	*Reference to booking*	*BIGINT*		*NO*		*FK*

5.1.2.11 Data 11

< Data 11>	
Name	Seat
Alias	Aircraft Seat
Where-used/how-used	Stores seat layout and assignment details for aircraft. Used to manage available, reserved, or occupied seats.
Content description	Seat = seat_no + status + aircraft_id

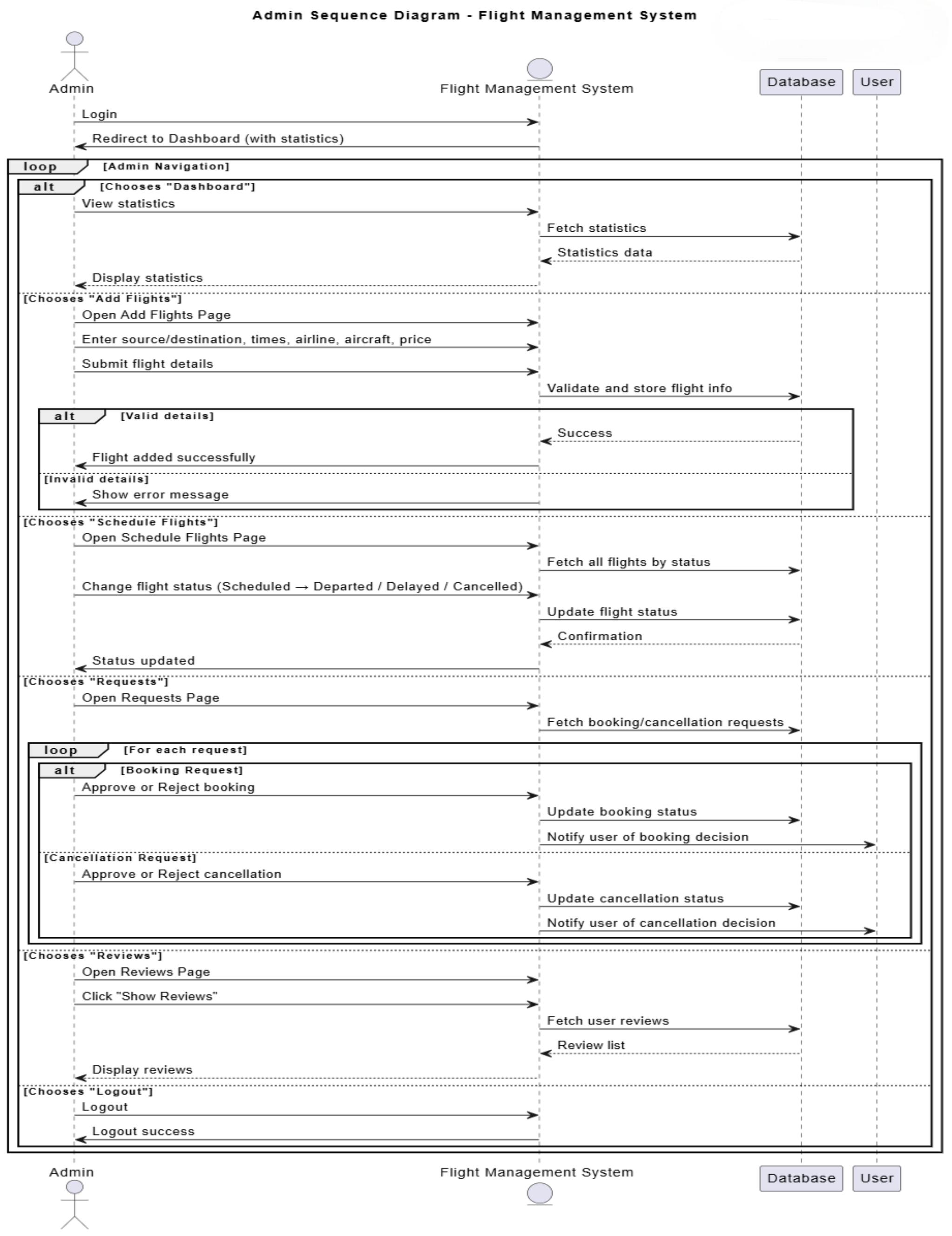
Column Name	**Description**	**Type**	**Length**	**Null able**	**Default Value**	**Key Type**
seat_no	*Unique identifier for seat*	*VARCHAR*	*255*	*NO*		*PK*

aircraft_id	*Reference to aircraft*	*BIGINT*		*NO*		*FK*
status	*Availability status (“Available”/ “Reserved”/ “Booked”)*	*VARCHAR*	*255*	*NO*	*“Available”*	

5.2 Application Design

5.2.1 Sequence Diagram

5.2.1.1 Admin Sequence Diagram



1. Admin Sequence Diagram

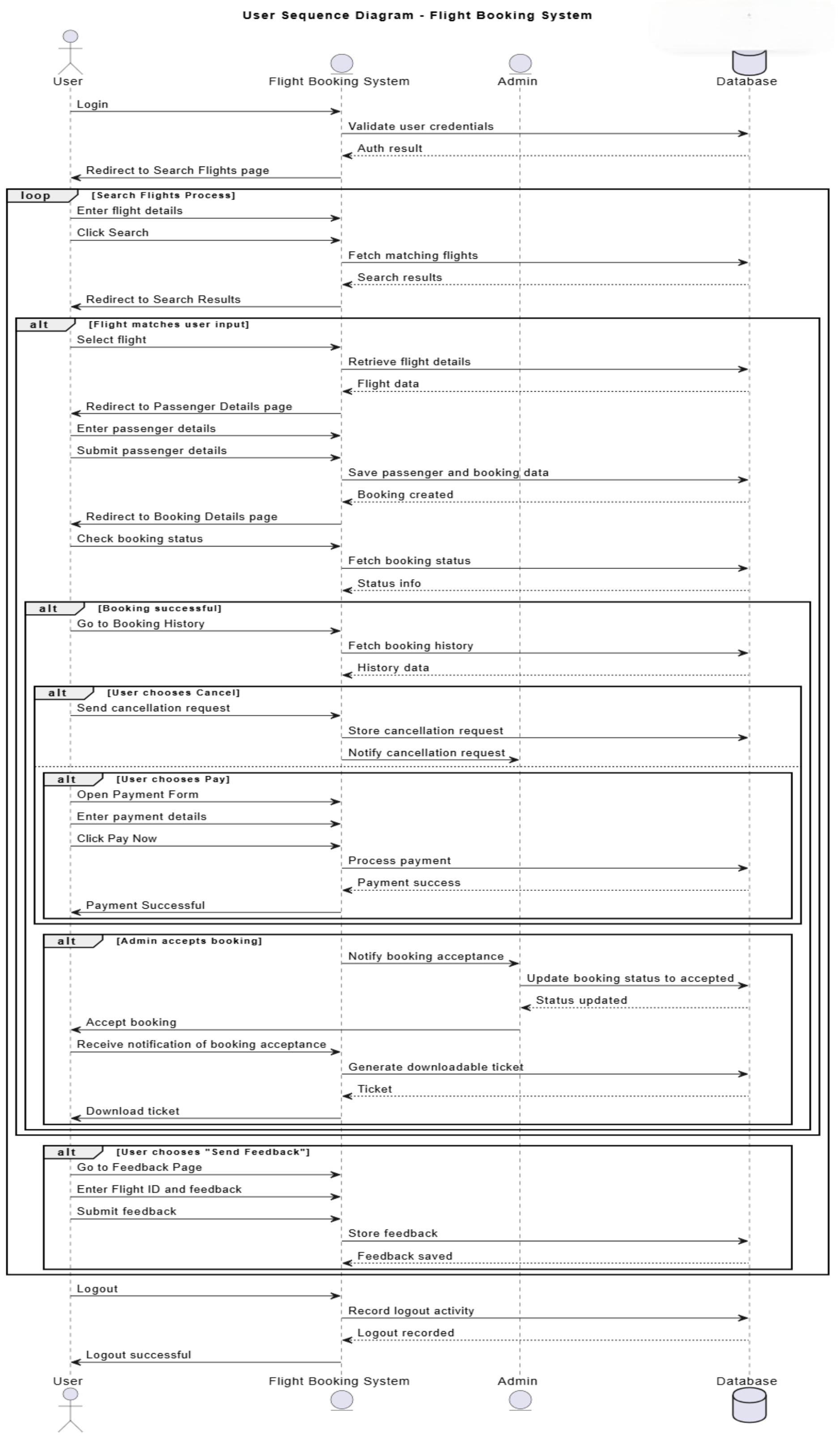
The Admin Sequence Diagram illustrates how an administrator interacts with the Flight Management System. It includes essential administrative functionalities for managing flights, bookings, reviews, and system data.

Process Overview:

1. **Login**: Admin logs in and is redirected to a dashboard that displays system statistics.
2. **Admin Navigation (Loop)**: After login, the Admin can repeatedly perform the following operations:
    - **Dashboard**: View statistical data fetched from the database.
    - **Add Flights**: Enter flight details. The system validates and stores them. Admin receives feedback based on success or error.
    - **Schedule Flights**: View and update flight statuses (Scheduled, Departed, Delayed, Cancelled).
    - **Manage Requests**: Handle booking and cancellation requests by approving or rejecting them. Users are notified accordingly.
    - **View Reviews**: Access and display user-submitted reviews.
3. **Logout**: Admin logs out and receives confirmation of successful logout.

This diagram highlights the system's support for efficient and controlled administrative operations through well-structured interactions.

5.2.1.2 User Sequence Diagram



2. User Sequence Diagram

The User Sequence Diagram captures the entire flow of a user interacting with the Flight Booking System—from flight search to booking, payment, cancellation, and feedback.

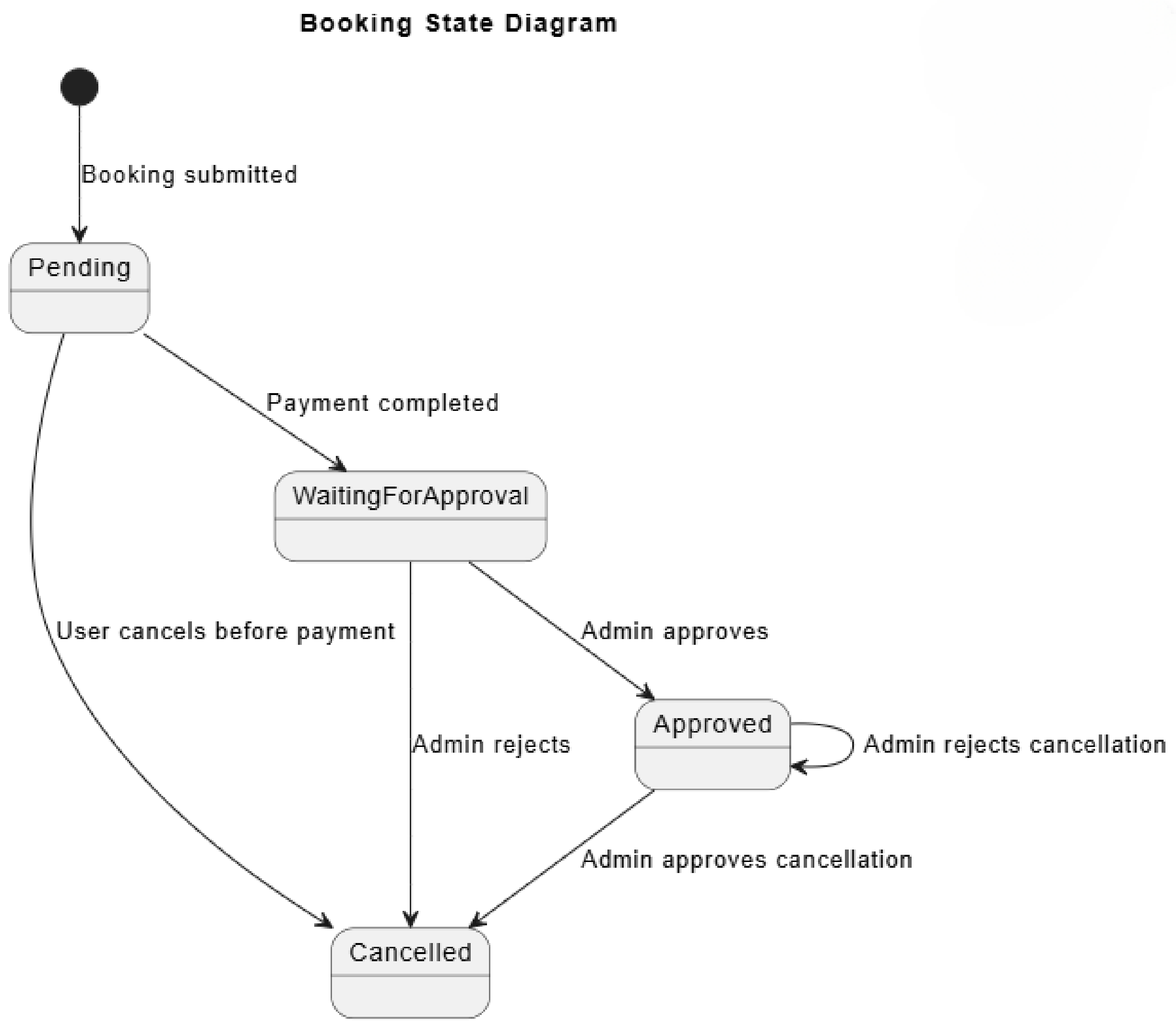
Process Overview:

1. **Login**: User logs in. Credentials are verified, and the user is redirected to the flight search interface.
2. **Search Flights (Loop)**: User enters search criteria. The system fetches and displays matching flights.
3. **Select Flight and Enter Details**:
   - User selects a flight and is redirected to the passenger details form.
   - User enters and submits details. The system creates a booking and redirects the user to booking details.
4. **Check Booking Status**: The system retrieves and shows the current booking status.
5. **Booking History**: If a booking is successful, the user can access their history.
6. **Cancellation (Alt)**: User can send a cancellation request. The system stores and notifies the admin.
7. **Payment (Alt)**: User can proceed with payment. Upon successful transaction, a confirmation is shown.
8. **Admin Accepts Booking (Alt)**:
   - Admin accepts the booking.
   - System updates status and generates a downloadable ticket for the user.
9. **Feedback (Alt)**: User submits feedback with flight ID. The system saves it.
10. **Logout**: User logs out. The activity is recorded, and confirmation is provided.

This sequence supports a seamless end-to-end booking process, ensuring all necessary steps—from search to post-flight feedback—are systematically handled.

5.2.2 State Diagrams

5.2.2.1 Booking States



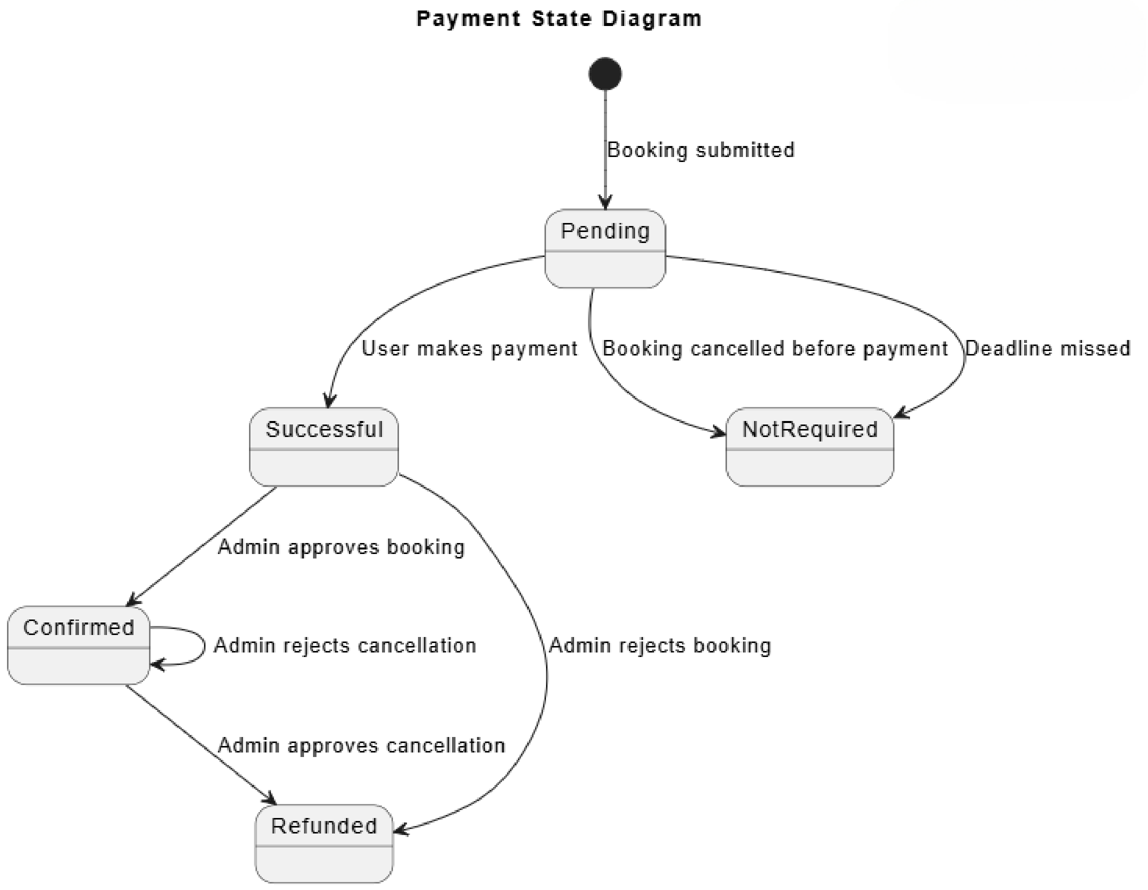
[*] → Pending
The booking is created when the user submits passenger info.
Pending → Cancelled
Happens if:

- The user cancels before payment
- Or the booking times out (e.g., payment deadline is missed)

Pending → Waiting For Approval
Payment is completed successfully.
Pending → Cancelled
User cancels the booking before completing payment.
Waiting For Approval → Approved
Admin reviews and approves the booking.

Waiting For Approval → Cancelled
Admin rejects the approval request; booking is canceled.
Approved → Cancelled
Admin approves the cancellation.
Approved → Approved
Admin rejects the cancellation request — booking remains valid.

5.2.2.2 Payment States



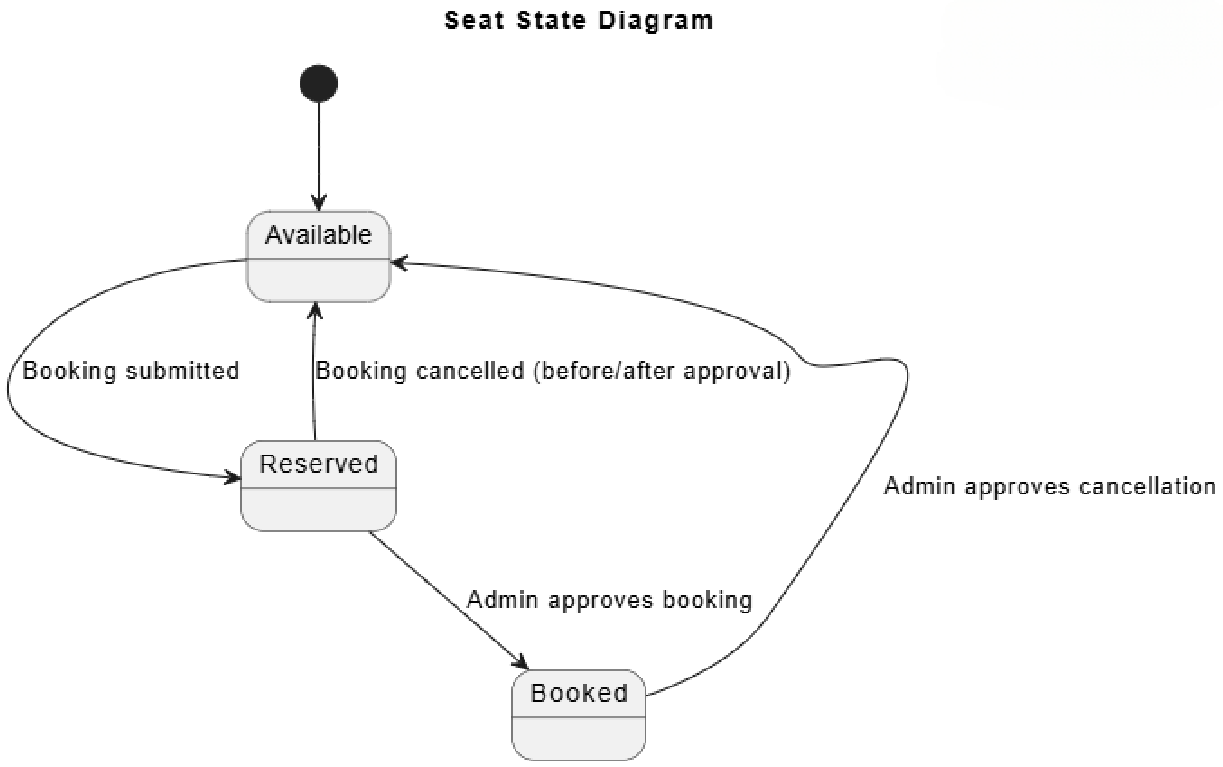
[*] → Pending
A payment record is created when a booking is submitted.
Pending → Successful
The user completes the payment.
Pending → Not Required
Payment is no longer needed because:

- Booking was cancelled before payment
- Payment deadline expired

Successful → Confirmed
Admin approves the booking after payment — confirming it.
Successful → Refunded
Admin rejects the booking — money is refunded.

Confirmed → Refunded
Admin approves a cancellation request after approval.
Confirmed → Confirmed
Admin rejects the cancellation request — no change.

5.2.2.3 Seat States

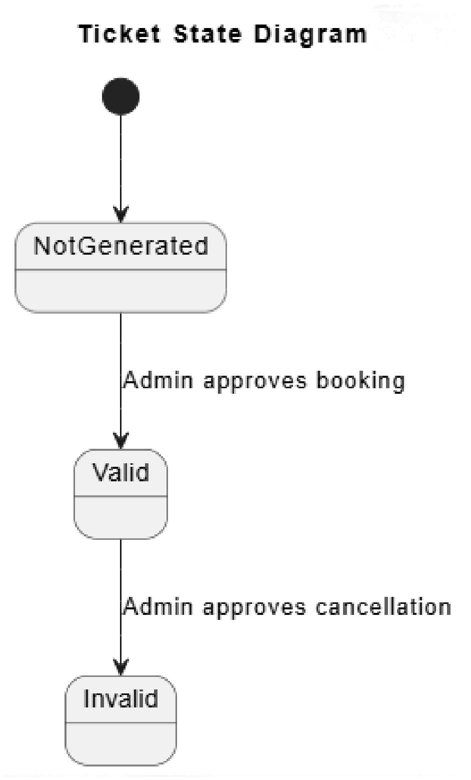


[*] → Available
Default state of a seat when no one has booked it.
Available → Reserved
When a user submits a booking, corresponding seats are reserved.
Reserved → Available
Reservation is undone if:

- *Booking is canceled before approval*
- *Admin rejects booking or cancels afterward*

Reserved → Booked
Booking is approved by admin — reserved seats are now fully booked.
Booked → Available
Admin approves a cancellation request — booked seats are released back.

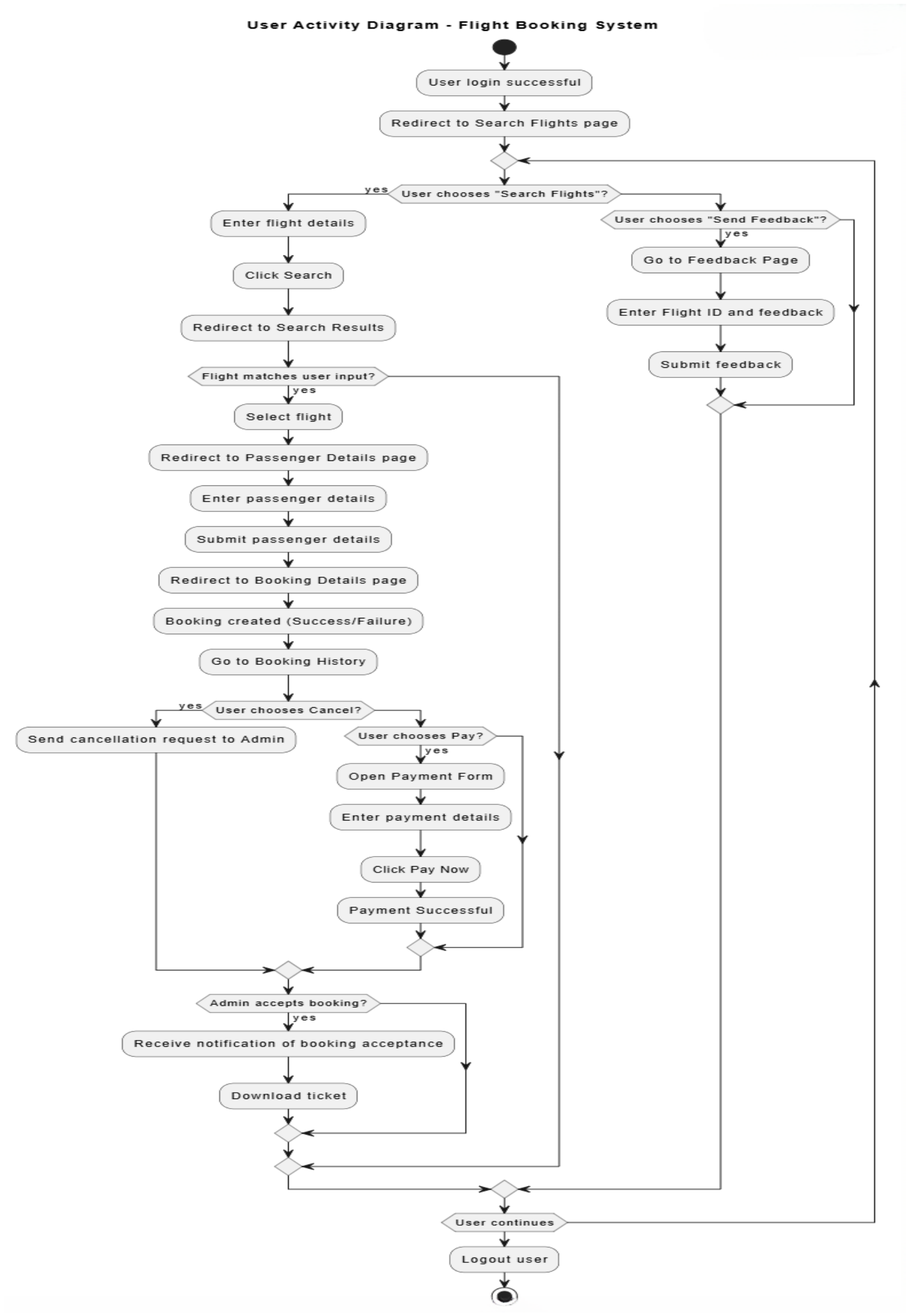
5.2.2.4 Ticket States



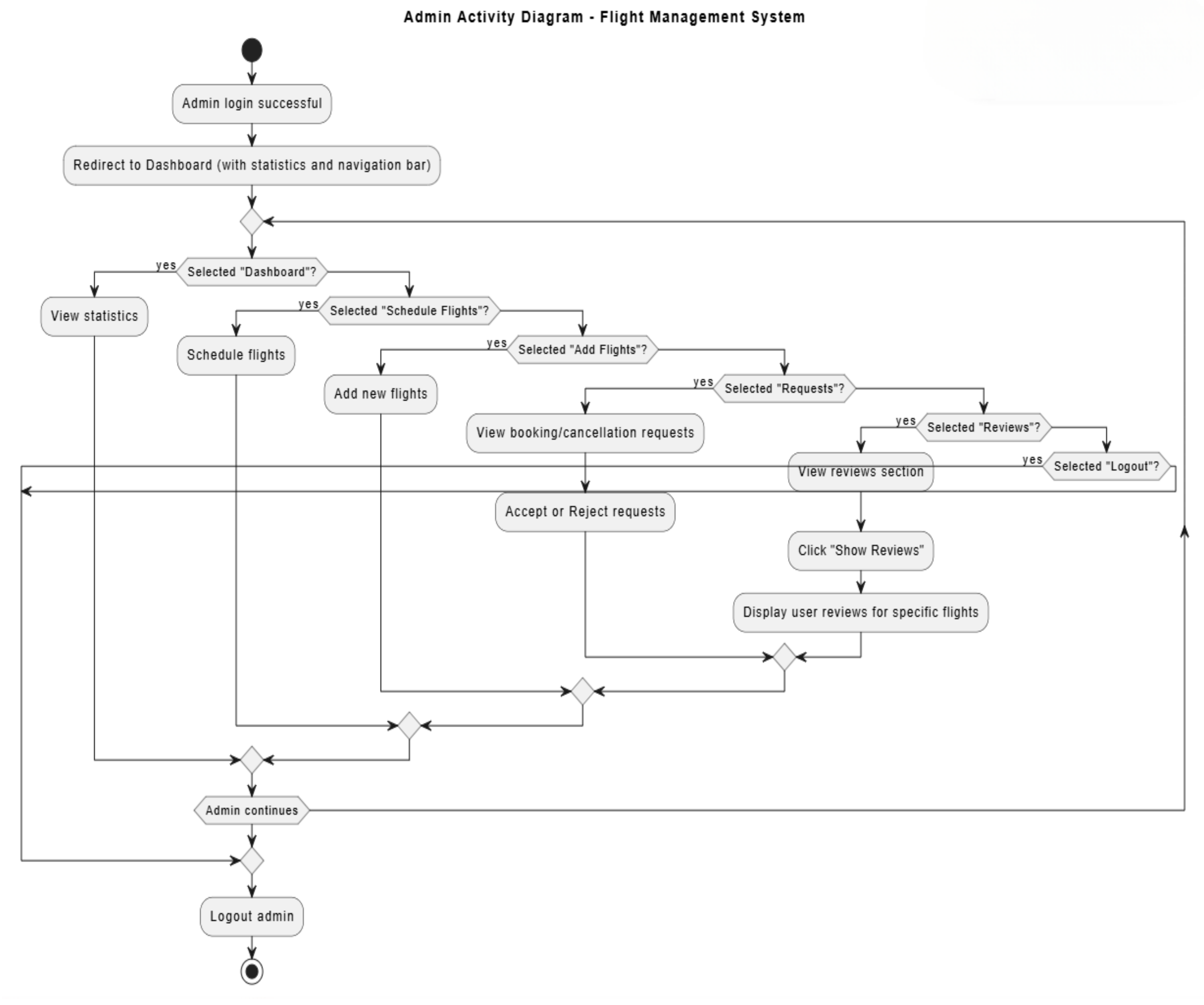
[*] → Not Generated
Initially, no ticket exists before booking approval.
Not Generated → Valid
Ticket is generated and marked valid when admin approves the booking.
Valid → Invalid
If admin later approves a cancellation request, the issued ticket is invalidated.

5.2.3 Activity Diagrams

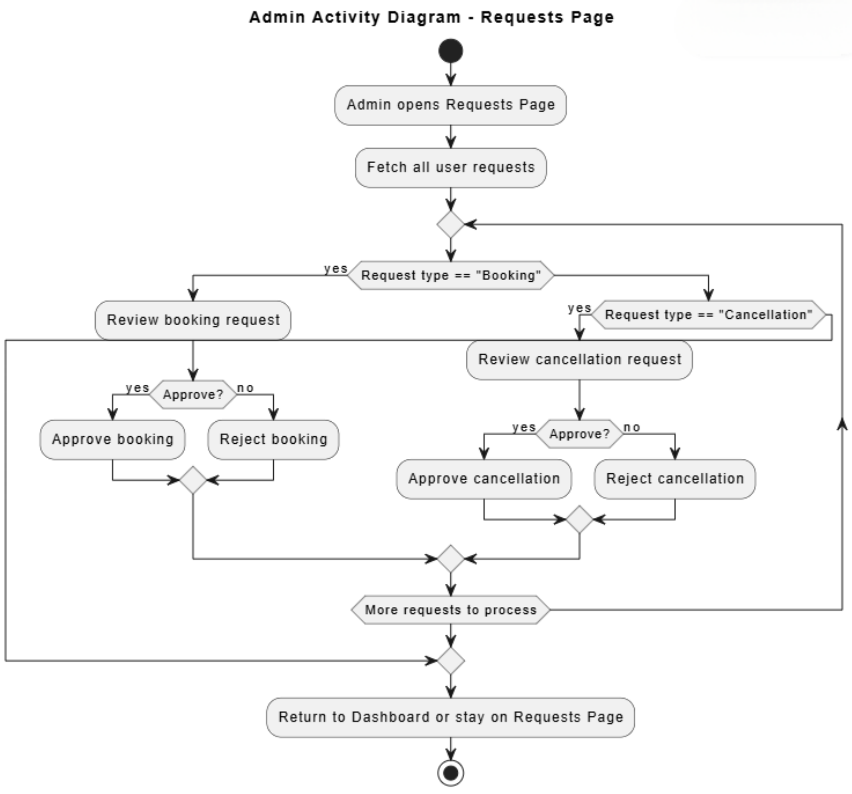
5.2.3.1 User Activity


User Activity Diagram - Flight Booking System
User login successful
Redirect to Search Flights page
User chooses "Search Flights"?
yes
Enter flight details
Click Search
Redirect to Search Results
Flight matches user input?
yes
Select flight
Redirect to Passenger Details page
Enter passenger details
Submit passenger details
Redirect to Booking Details page
Booking created (Success/Failure)
Go to Booking History
User chooses Cancel?
yes
Send cancellation request to Admin
User chooses Pay?
yes
Open Payment Form
Enter payment details
Click Pay Now
Payment Successful
Admin accepts booking?
yes
Receive notification of booking acceptance
Download ticket
User chooses "Send Feedback"?
yes
Go to Feedback Page
Enter Flight ID and feedback
Submit feedback
User continues
Logout user

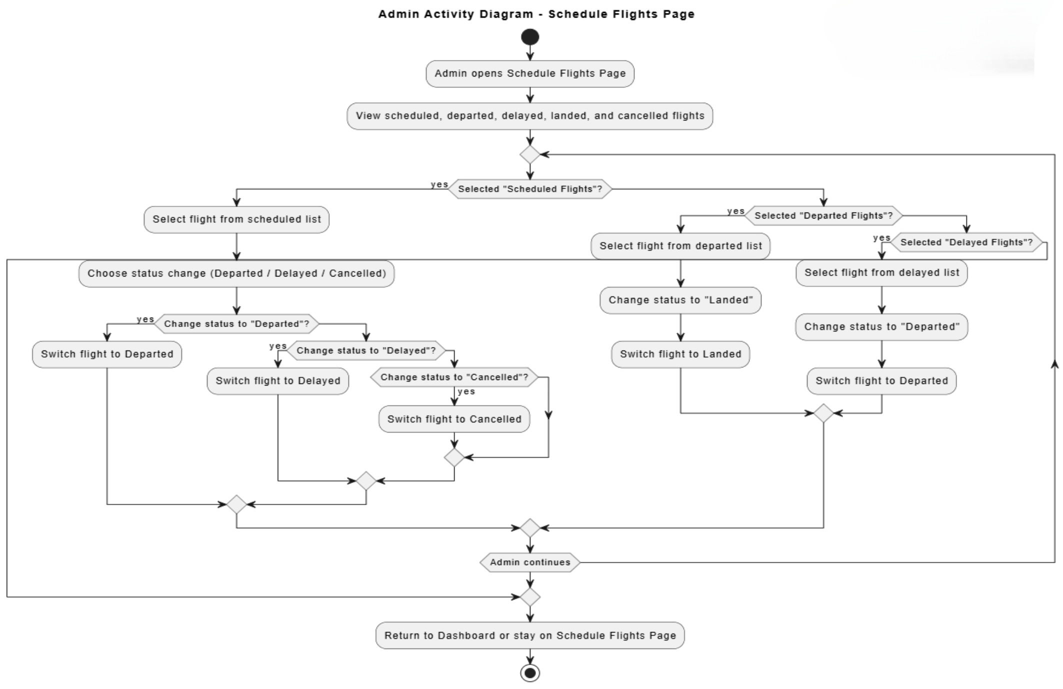
5.2.3.2 Admin Activity



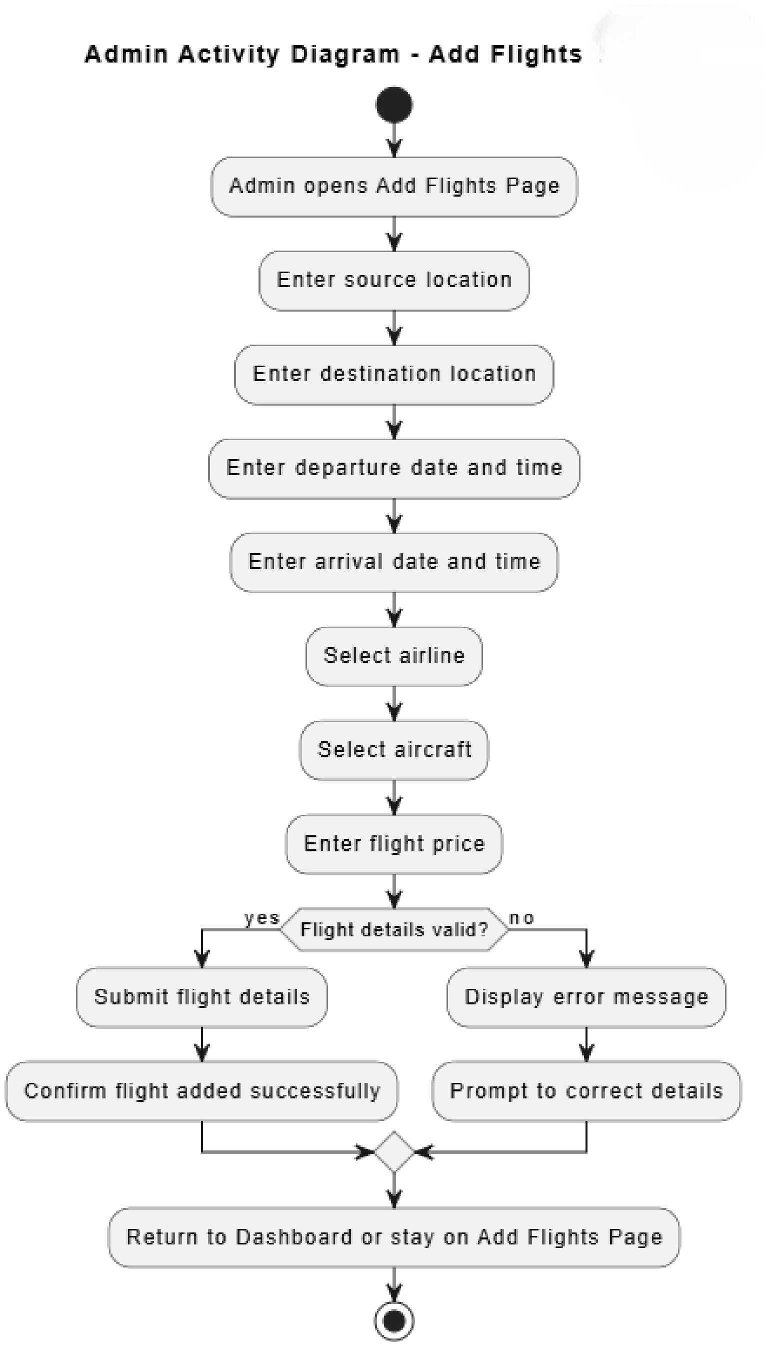
5.2.3.2.1


Admin Activity Diagram - Requests Page
Admin opens Requests Page
Fetch all user requests
yes
Request type == "Booking"
Review booking request
yes
Request type == "Cancellation"
Review cancellation request
yes
Approve?
no
Approve booking
Reject booking
yes
Approve?
no
Approve cancellation
Reject cancellation
More requests to process
Return to Dashboard or stay on Requests Page

5.2.3.2.2


Admin Activity Diagram - Schedule Flights Page
Admin opens Schedule Flights Page
View scheduled, departed, delayed, landed, and cancelled flights
yes
Selected "Scheduled Flights"?
Select flight from scheduled list
yes
Selected "Departed Flights"?
Select flight from departed list
yes
Selected "Delayed Flights"?
Choose status change (Departed / Delayed / Cancelled)
Select flight from delayed list
Change status to "Landed"
Change status to "Departed"
yes
Change status to "Departed"?
Switch flight to Departed
yes
Change status to "Delayed"?
Switch flight to Delayed
Change status to "Cancelled"?
yes
Switch flight to Cancelled
Switch flight to Landed
Switch flight to Departed
Admin continues
Return to Dashboard or stay on Schedule Flights Page

5.2.3.2.3


Admin Activity Diagram - Add Flights
Admin opens Add Flights Page
Enter source location
Enter destination location
Enter departure date and time
Enter arrival date and time
Select airline
Select aircraft
Enter flight price
Flight details valid?
yes
no
Submit flight details
Display error message
Confirm flight added successfully
Prompt to correct details
Return to Dashboard or stay on Add Flights Page

6 References

[This section should provide a complete list of all documents referenced at specific point in time. Each document should be identified by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained (This section is like the bibliography in a published book)].

7 Appendices

[Include supporting detail that would be too distracting to include in the main body of the document.]