Software Requirements Specification

for

Airlines Reservation System

Version 3.0

Prepared by

Debjit Basu	**12021002018001 (05)**
Saad Mohammad	**12021002018039 (06)**
Vidhi Mantri	**12021002018008 (09)**
Shreyansh Agarwal	**12021002018009 (10)**
Ved Anand	**12021002018015 (11)**
Ayanika Bera	**12021002018014 (12)**
Nikhil	**12021002018022 (15)**
Samridh Agrawal	**12021002018044 (23)**
Adesh Kumar Dubey	**12021002018033 (24)**

Instructor: **Prof. Dr. INDRAJIT DE**
Course: **SOFTWARE ENGINEERING LAB**
Lab Section: **A**
Submission Date: **2023-11-09**
Supervisor's Sign:

Contents

REVISIONS

Version	Primary Author(s)	Description of Version	Date Completed
1.0		This is the first draft of the software.	14/08/23
2.0		This is the second draft of the software	08/10/23
3.0		This is the third draft of the software	08/11/23

1 Introduction

1.1 Document Purpose

Airline Reservation System (ARS) is a system that allows airline to sell their inventory (seats). It contains information on schedules and fares and contains a database of reservations (or passenger name records) and of tickets issued. ARS eventually evolved into the Computer Reservation System (CRS). A Computer Reservation System is used for the reservations of a particular airline and interfaces with a Global Distribution System (GDS) which supports travel agencies and other distribution channels in making reservations for most major airlines in a single system. ARS aims to automate the flight operations and seat booking and confirmation system of an Airline company. The software is providing options for viewing different flights available within different timings for a specific day which provide customers within facility to be able to book tickets smoothly.

1.2 Product Scope

The Airlines Reservation System allows the airline passenger to search for flights that are available between the two travel cities, namely the "Departure City" and "Arrival City" for a particular departure and arrival dates. The system displays all the flight's details such as flight no., name, price and duration of journey, etc.
The Airline booking website is an application stored in the user server. The purpose of the website is to resolve the client to allow website users to perform the following functions:

- Automation of flight operations
- Automation of seat booking
- Confirmation System
- Cancellation
- Improved and optimized service

1.3 Intended Audience and Document Overview

The Airlines Reservation System documentation is intended for stakeholders across the aviation industry, including airline executives, project managers, IT professionals, software developers, and regulatory authorities. This document provides a comprehensive overview of the system's features, its underlying technologies, and the expected benefits for both airlines and their passengers.

1.4 Definitions, Acronyms and Abbreviations

- ARS: Airlines Reservation System
- API: Application Programming Interface
- ATC: Air Traffic Control
- CRM: Customer Relationship Management
- DBMS: Database Management System
- GUI: Graphical User Interface
- IT: Information Technology
- OS: Operating System
- SRS: Software Requirements Specification

- UI: User Interface
- UML: Unified Modeling Language
- URL: Uniform Resource Locator

1.5 Document Conventions

The Airlines Reservation System is a web-based application that allows visitors check air ticket availability, buy air ticket, cancel air ticket and pay the fare for air ticket online. This system is established for all the home/office users after gaining access from the administrator.
Throughout this document, certain conventions are used to ensure clarity and consistency. Font styles such as italics are employed to emphasize specific terms or concepts. The document adheres to recognized industry standards, including IEEE guidelines for Software Requirements Specification, ensuring a professional and structured representation of the Airlines Reservation System.

1.6 References and Acknowledgments

The development of the Airlines Reservation System draws inspiration from various sources that contribute to the advancement of aviation technology and operations management. Key references include industry reports, academic research, and existing airline reservation systems that have paved the way for innovative solutions. Acknowledgments go to the contributors of these resources and the aviation community for their collective efforts in shaping the landscape of airline operations.

2 Overall Description

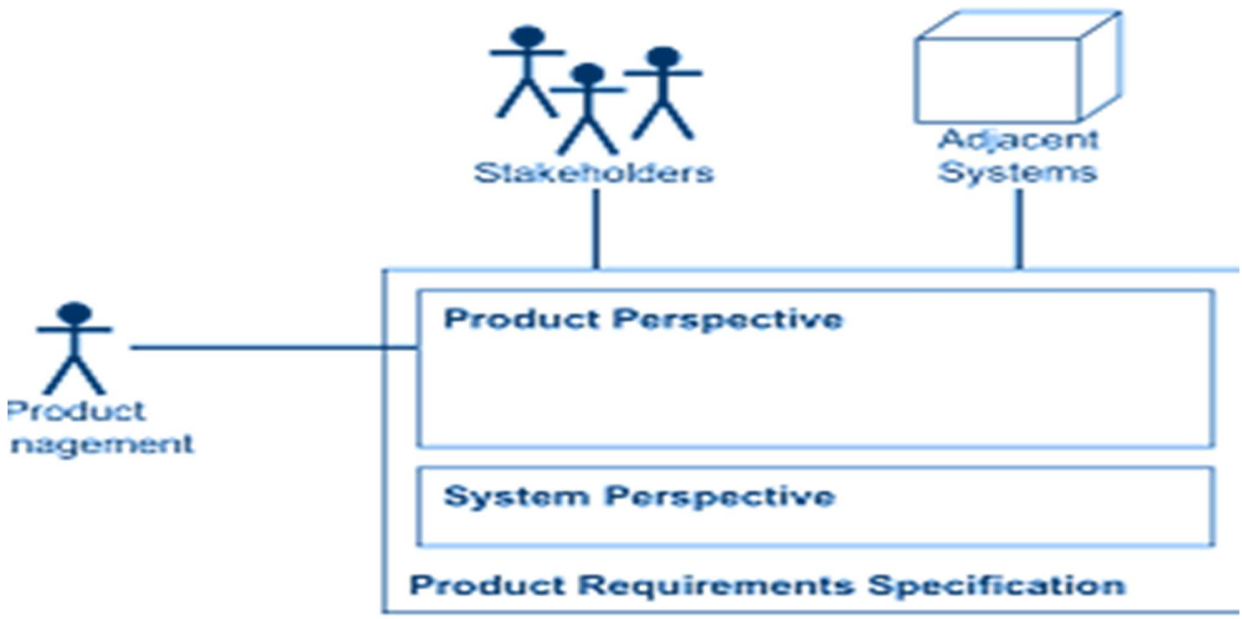
2.1 Product Perspective

Our Airlines Reservation System (ARS) is a comprehensive software solution designed to streamline and optimize various operations within reservation system. ARS is a new, self-contained product aimed at providing a centralized platform for managing airline passenger services and reservations.

The ARS is proposed with the following perspective:

- The computerization of the reservation system will reduce a lot of paper-work and hence load on airline admin and staff.
- The machine will perform all calculations. Hence chances of error are nearer to nil.
- The passenger, reservation, cancellation list can be easily retrieved and any required addition, deletion and updating can be performed easily and fast.
- Proper way of confirmation of bookings, etc.

The following diagram provides a high-level overview of how the Airlines Reservation System interacts with its environment



2.2 Product Functionality

Booking agents with varying levels of familiarity with computers will mostly use this system. With this in mind, an important feature of this software is that it will be relatively simple to use. The Scope of this product encompasses:

- SEARCH: This function allows the booking agent to search for airplanes and ticket's availability between two cities, i.e. Departure city and Arrival city, the date of departure, preferred time and number of passengers.
- SELECTION: This function allows a particular airplane to be selected from the displayed list. All details such as:
    - Airplane number
    - Date, time and place of departure
    - Date, time and place of arrival
    - Fare per head, etc.
- REVIEW: If seats are available, then system prompts for booking. All the information including total fare with taxes and flight details are reviewed.
- TRAVELER INFO.: The details of all passengers who are supposed to travel including with their respective name, address, contact number, Email etc.
- PAYMENT: It asks the agent to enter the various credit details of the person making the reservation, i.e. Credit card type, Credit card number, Expiration date of the card, Name on the card, CVV, etc.
- CANCELLATION: The system allows the passenger to cancel a reservation and register the information regarding his/her ticket. It includes Confirmation no, name, date of journey, fare deducted.

2.3 Operating Environment

The designed system is thought to be a website and will be available via any Web Browser application. It will not be dependent on the technical capabilities or operating system of user's device.

2.4 Design and Implementation Constraints

Flight dates and hours should be displayed according to respective city's time zones and saving time settings for each country should be considered. Additionally, info about any changes made in the database should be displayed with no delay and to do these all we need some constraints.

So, our system design and implementation are subject to certain constraints that shape our development approach. These constraints include:

- Technology Stack: The System is Web-Based, so it will run on a web browser, i.e. IE, Chrome, Firefox etc.
- Hardware Limitations: The system should be designed to work efficiently under any OS with internet functionality.

- Integration with External Services: AMS will integrate with external services for services like real-time flight tracking and payment processing.
- Security Considerations: Robust security measures must be implemented to safeguard sensitive passenger data and ensure compliance with industry regulations.
- Communication Protocols: The system must support secure communication between various system components and external services.

2.5 User Documentation

The instructions on how to book a flight will be provided on the website itself for the Inexperienced users.

2.6 Assumptions and Dependencies

The successful deployment and functioning of the Airlines Reservation System are based on several key assumptions:

- The availability of stable and high-speed internet connectivity for real-time data exchange.
- The timely availability of accurate flight data from external sources for proper info display.
- The availability of complete booking knowledge to the Booking agent with a proper username and passcode to access the System.
- The system will be deployed on reliable and scalable hosting infrastructure.
- The development team will have access to the required development tools and technologies.

3 Specific Requirements

3.1 External Interface Requirements

The external interface requirements of the Airlines Reservation System outline the interactions and connections between the software product and its users, hardware components, and other software systems. These interfaces play a vital role in ensuring seamless communication, efficient data exchange, and smooth functionality of the system. The external interfaces include user interfaces, hardware interfaces, and software interfaces.

3.1.1 User Interfaces

- Airlines Reservation System should have a friendly Customer User Interface.
- Airlines Reservation System should have an Administrator user Interface.
- Customer UI should have a Graphical User Interface (GUI).
- Administrator UI should also have a GUI.

3.1.2 Hardware Interfaces

No Hardware Interface is required for the Airline ticket Reservation System.

3.1.3 Software Interfaces

- ARS should be a web-based system.
- It should be possible to open the website on the computers with operating systems of Microsoft 7, Microsoft 8, Microsoft 10, Mac OS, Linux, Ubuntu.
- Oracle database 12C should be used to store the data about flight details.
- Oracle Database 12C should also be used to store the data of and about users.

3.1.4 Communication Interface

HTTP protocol should be used as an interface of communication between client and server sides. The system supports Google Chrome, Mozilla Firefox and other web browsers also.

.

3.2 System Features

ARS is a next-generation flight passenger service platform designed to provide airlines with greater business flexibility and operational efficiency. Born out of the need for a more dynamic PSS system in the aviation industry, this platform features a customer-centric design that can seamlessly manage changing business models – from low cost to hybrid operations. Our Rules Engine allows system configuration to reach your target, reducing vendor dependence while ensuring higher productivity and operational ease.

The six critical components of ARS – Reservation, Cancellation, Fares, Departure Control, and Central Customer Management – cover the entire spectrum of flight processes, working together to provide an unparalleled customer experience. Our Flight Reservation System assists in transactions like Air Ticket Booking, including booking, reserving, canceling, and rescheduling tickets. It minimizes repetitive work done by system administrators and reservation clerks, ensuring consistency across different access modes such as mobile, online, inquiries, and various locations.

3.2.1 Registration

User properties like Name, Address, Age will be required by the user to access the ARS website if they are using the website for the first time or they can login. If they register, they will get a username and will have to setup a password for future logins.

3.2.2 Searching

The user can search for airplanes and ticket's availability between two cities, i.e. departure city and arrival city, the date of departure, preferred time and number of passengers.

3.2.3 Booking

The users should be taken through he same steps by the system as they go through in conventional desk-reservation systems.
User can book a flight by selecting details like Airplane Number, Date of Departure and Arrival. The booking will be done only when the user completes the payment process.

3.2.4 Cancellation

Our Flight Reservation System Software maintains customer information in case of any emergencies, e.g. Flight cancellation due to inclement weather. The user can cancel their

reservation if cancellation is available according to the cancellation policy. If the cancellation is available, the user will be forwarded to the refund page where the user can enter card details and apply for refund.

3.2.5 Pricing and Discounts

The flight companies will use this profile to track user choice and travel patterns to serve them better, plan routes, for better marketing and effective scheduling of flights. It will help to raise the revenue of the flight company by various means: make aware of frequent travelers about various offers and discounts, reduce the number of vacant seats on a flight and maximize flight capacity utilization, and maintain the capability to adapt a flexible pricing policy. The fare of the tickets should be determined based on how early and before the date of departure, the customer buys the ticket.

3.2.6 Ancillary Services

The system caters to ancillary services, offering passengers a range of additional services beyond their flight bookings. From extra baggage options to in-flight meal preferences, passengers can customize their travel experience according to their preferences. This functionality contributes to increased revenue streams for airlines and enhances passenger satisfaction.

3.3 Functional Requirements

3.3.1 Performance requirements

• User Satisfaction: The system is such that it stands up to the user expectations. • Response Time: The response of all operations is good.

• Error Handling: Response to user errors and undesired situation has been taken care of to ensure that the system operates without halting.

• Safety and Robustness: The system is able to avoid or tackle disastrous action. In other words, it should be foul proof.

• Portable: The software should not be architecture specific. It should be easily transferable to other platforms if needed.

• User Friendliness: The system is easy to learn and understand. A native user can also use the system effectively, without any difficulties.

3.3.2 Hardware Requirements

For the hardware requirements like memory restrictions, cache size, the processor, RAM size etc... those are required for the software to run.

- MINIMUM Hardware Requirements:
  - Processor Pentium IV
  - Hard Disk Drive 100 GB
  - RAM 1 Gb

- PREFERED Hardware Requirements:

  - Processor Core i3
  - Hard Disk Drive 500 GB
  - RAM 4 GB

3.3.3 Software Requirements

Any window-based operating system with DOS support are primary requirements for software development. Windows 7 and up are required. The system must be connected vie LAN and connection to internet is mandatory.

3.4 Use Case Model

Actors:

- **Administrator**
- **Bank**
- **Passenger**

3.4.1 Use Case #1: User Registration

Purpose: New users can create an account in the system. **Flow:**

1. **Actor:** Passenger
2. Passenger selects "Register" option.
3. System displays registration form.
4. Passenger enters required information (name, email, password, etc.).
5. System validates information and creates a new user account.
6. System notifies the user about successful registration.
7. End of use case.

3.4.2 Use Case #2: User Login

Purpose: Registered users can log in to their accounts. **Flow:**

1. **Actor:** Passenger
2. Passenger selects "Login" option.
3. System prompts for email and password.
4. Passenger enters login credentials.
5. System verifies credentials and grants access.
6. End of use case.

3.4.3 Use Case #3: View Flight Timing and Locations

Purpose: Users can view available flight timings and locations. **Flow**:

1. **Actor**: Passenger
2. Passenger selects "Flight Timing and Locations" option.
3. System displays available flight timings and locations.
4. Passenger reviews options.
5. End of use case.

3.4.4 Use Case #4: Check Availability and Seat Selection

Purpose: Passengers can check flight availability and select seats. **Flow**:

1. **Actor**: Passenger
2. Passenger selects "Check Availability and Seat Selection" option.
3. System displays available flights and seat options.
4. Passenger chooses a flight and selects seats.
5. End of use case.

3.4.5 Use Case #5: Ticket Booking and Payment

Purpose: Passengers can book tickets and make payments. **Flow**:

1. **Actor**: Passenger
2. Passenger selects "Book Ticket" option.
3. System prompts for flight selection and passenger details.
4. Passenger enters required information.
5. System calculates the total fare and requests payment details.
6. Passenger provides payment information.
7. System processes the payment and confirms the booking.
8. End of use case.

3.4.6 Use Case #6: Validation, Ticket Generation, and Receipt

Purpose: Passengers' booking details are validated, tickets are generated, and receipts are provided. **Flow**:

1. **Actor**: Passenger
2. Passenger selects "Validate Booking and Get Ticket" option.
3. System validates booking details and payment.
4. System generates an electronic ticket.
5. System sends the ticket to the passenger and provides a receipt.
6. End of use case.

3.4.7 Use Case #7: Ticket Cancellation and Refund Process

Purpose: Passengers can cancel their booked tickets and initiate the refund process. **Flow**:

1. **Actor**: Passenger
2. Passenger selects "Cancel Ticket" option.
3. System prompts for booking reference or passenger details.
4. Passenger provides required information.
5. System validates the request and cancels the ticket.
6. Refund process is initiated, and the passenger is notified.
7. End of use case.

3.4.8 Use Case #8: Boarding Pass Retrieval

Purpose: Passengers can retrieve their boarding passes for a booked flight. **Flow**:

1. **Actor**: Passenger
2. Passenger selects "Retrieve Boarding Pass" option.
3. System prompts for booking reference or passenger details.

4. Passenger provides required information.
5. System retrieves the boarding pass and presents it to the passenger.
6. End of use case.

3.4.9 Use Case #9: Flight Schedule Update

Purpose: Admin can update the schedule of flights. **Flow**:

1. **Actor**: Admin
2. Admin selects "Update Flight Schedule" option.
3. System displays the current flight schedule.
4. Admin modifies flight timings or locations as needed.
5. System validates changes and updates the schedule.
6. System notifies affected passengers about schedule changes.
7. End of use case.

4 Other Non-functional Requirements

4.1 Performance Requirements

Performance requirements dictate the expected system behavior under various circumstances, ensuring that the system operates efficiently and responsively. These requirements are often closely tied to specific functional features and help guide design choices to achieve optimal performance. They also ensure that the system can meet the timing relationships in real-time scenarios.

- **Response Time:** The system should promptly process and confirm flight reservations, seat assignments, and upgrades within 5 seconds of user submission. Rapid response times minimize passenger wait times during booking and enhance user satisfaction.

- **Scalability:** The system must be capable of accommodating increasing numbers of concurrent users during peak booking periods, such as holiday seasons or special events. Horizontal scaling should be implemented to add resources and maintain responsiveness.

- **Flight Data Processing:** Flight updates, such as departure and arrival information, should be processed and reflected across the system within 1 minute of the information becoming available to prevent any discrepancies in flight statuses.

4.2 Safety and Security Requirements

Safety and security requirements are critical for the protection of both users and the system itself. These requirements mitigate potential risks and threats associated with the use of the system. They define safeguards, actions to prevent harm, and any mandatory actions needed to ensure safety and security compliance. Additionally, they outline access control mechanisms to safeguard sensitive passenger data and ensure data privacy
.

- **Access Control:** User authentication should follow aviation security standards. The system must automatically log out all customers after a period of inactivity. The system should not leave any cookies on the customer's computer containing the user's password. The system's back-end servers shall only be accessible to authenticated management

- **Data Encryption:** All passenger and flight data transmitted over the network should be encrypted to prevent unauthorized access or eavesdropping. Sensitive information, such as personal details and flight plans, must be encrypted both at rest and in transit.

- **Audit Trails:** Detailed audit logs of user activities, flight status changes, and system modifications should be maintained. These logs support traceability, security investigations, and regulatory compliance.

4.3 Software Quality Attributes

Software quality attributes, also known as non-functional requirements, define the overall qualities and characteristics that the system should possess to ensure its effectiveness, efficiency, and usability. These attributes guide system design and development to ensure that the end product meets the desired quality standards.

- Usability: The user interface must be intuitive and easy to navigate for everyone. Quick access to critical flight information and streamlined task execution are essential, especially in time-sensitive situations. The design should support Windows Server 2003, Linux 2.6.x, V10 UNIX and later.

- System Availability: The system should be operational 24/7, minimizing downtime for maintenance. Failover mechanisms and redundant servers must be in place to ensure uninterrupted access to critical functionalities.

- Data Consistency: The system should maintain accurate and consistent flight data across all modules. Flight details, passenger records, and availability information should be synchronized to avoid discrepancies.

- Robustness: The system design shall include recovery scenarios allowing the ability to restore a state no older than one business day old.

4.4 Client Questionnaire: ARS

- **By Airline Administrators and Management:**

  Q1) What are the core functionalities of the Airlines Reservation System as outlined in the SRS?

  Ans: The core functionalities of our Airlines Reservation System include:

  - User Registration and Login: Passengers can create accounts, log in, and manage their profiles.

  - Flight Search and Booking: Users can search for available flights, select seats, and complete the booking process efficiently.

- Secure Payment System: We support multiple payment methods, ensuring secure and smooth transactions for our users.
- Cancellation and Refund: Passengers can cancel bookings, and if applicable, initiate refund processes following the defined policies.
- Real-time Flight Information: Passengers can access real-time updates on flight availability, timings, and status.

Q2) How does the system ensure the security of passenger data and transactions?

Ans: The system employs advanced encryption techniques to secure all passenger data and payment transactions. We follow industry-standard security protocols and conduct regular security audits to safeguard user information.

Q3) Can you explain the process of adding new flight schedules and managing existing ones through the system?

Ans: Authorized administrators can log in, access the system's dashboard, and add new flight schedules or modify existing ones. The system allows for easy input of flight details, including departure and arrival times, locations, and available seats.

- **By Passengers:**

Q4) How user-friendly is the booking process? Can passengers easily search for flights, select seats, and complete the booking?

Ans: The booking process is designed to be intuitive and user-friendly. Passengers can easily search for flights based on their preferences, select seats from available options, and complete the booking with a few simple steps.

Q5) What payment methods are supported by the system? Is the payment process secure and efficient?

Ans: We support major credit and debit cards, as well as other secure online payment methods. Our payment process is encrypted and highly secure, ensuring the safety of all transactions.

Q6) How does the system handle cancellations and refunds, as mentioned in the SRS?

Ans: Passengers can initiate cancellations through their accounts. The system verifies eligibility based on the cancellation policy and processes refunds promptly if applicable. Users receive notifications about the refund status.

Q7) Is there a user-friendly interface for passengers to check flight availability, timings, and other details?

Ans: Yes, passengers can easily check flight availability, timings, and other details through our user-friendly interface. The system provides real-time updates, ensuring accurate and current information.

Q8) Can passengers receive real-time updates on flight status, including delays and cancellations?

Ans: Absolutely. Our system provides passengers with real-time updates on flight status, including delays, cancellations, and other relevant information. Passengers are promptly notified of any changes to their flights.

4.5 Business Rules

There are a number of factors in the client's environment that may restrict the choices of a designer. Such factors include standards that must be followed, resource limits, operating environment, reliability and security requirements and policies that may have an impact on the design of the system.

5 References

5.1 Books

- Essential of aviation management: a gold for aviation service businesses by Brian S. Flynn.
- Flight training manual
- Air regulations for PPA/ATPL (4th revised edition 2021) by the Civil Aviation Authority of the Philippines.
- Hard Landing: The Epic Contest for Power and Profits That Plunged the Airlines into Chaos by Thomas Petzinger Jr.

5.2 Websites

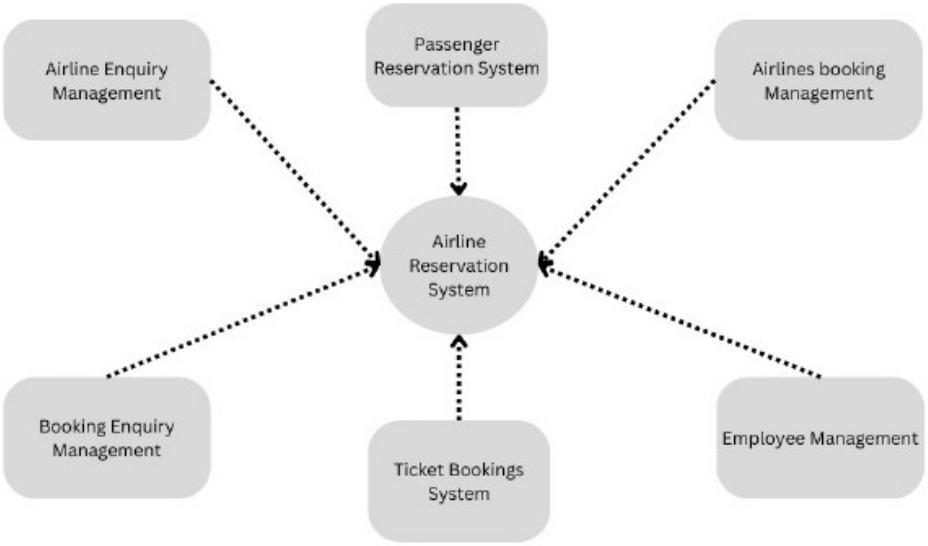
- Airline Reservation System SRS: https://www.studocu.com/row/document/iqra-university/bachelor-of-computer-science/srs-airline-management-system/8695767
- SRS for Airline Reservation System: https://www.scribd.com/doc/130966364/Airline-Reservation-System-SRS
- Amadeus Airline Booking Vulnerability Exposes Passenger Records: https://techcrunch.com/2019/01/15/amadeus-airline-booking-vulnerability-passenger-records/
- Aereos: https://aereos.com/
- SRS on Airline Reservation System: https://www.coursehero.com/sitemap/schools/3065-Utah-Valley-University/courses/4050407-AVSC3600/
- Airline Reservation System SRS: https://www.slideshare.net/arokhandelwal/airlinesnopsisfinal

Appendix A – Group Log

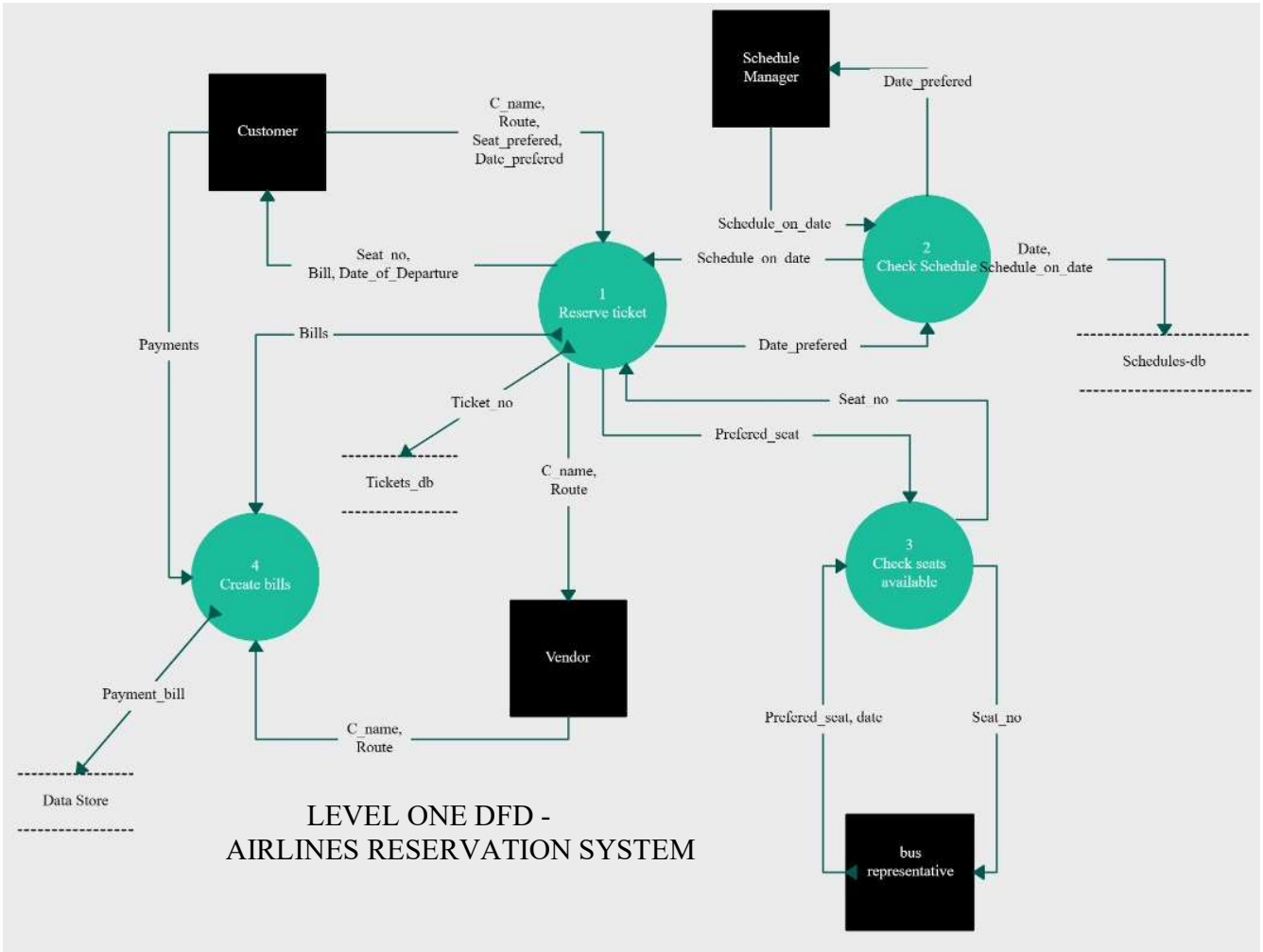
- *GROUP DISCUSSION: HELD ON:*
  - *26/07/2023*
  - *28/07/2023*
  - *04/08/2023*
  - *06/08/2023*
  - *14/08/2023*
  - *13/09/2023*
  - *19/09/2023*
  - *04/10/2023*
  - *10/10/2023*
  - *06/11/2023*
- *ALL RESEARCH DONE OVER AN ONLINE MEETING ON:*
  - *07/08/2023*
  - *08/08/2023*
  - *10/08/2023*
  - *14/08/2023*
  - *20/09/2023*
  - *08/10/2023*
  - *10/10/2023*
  - *08/11/2023*

Appendix B – Analysis Models

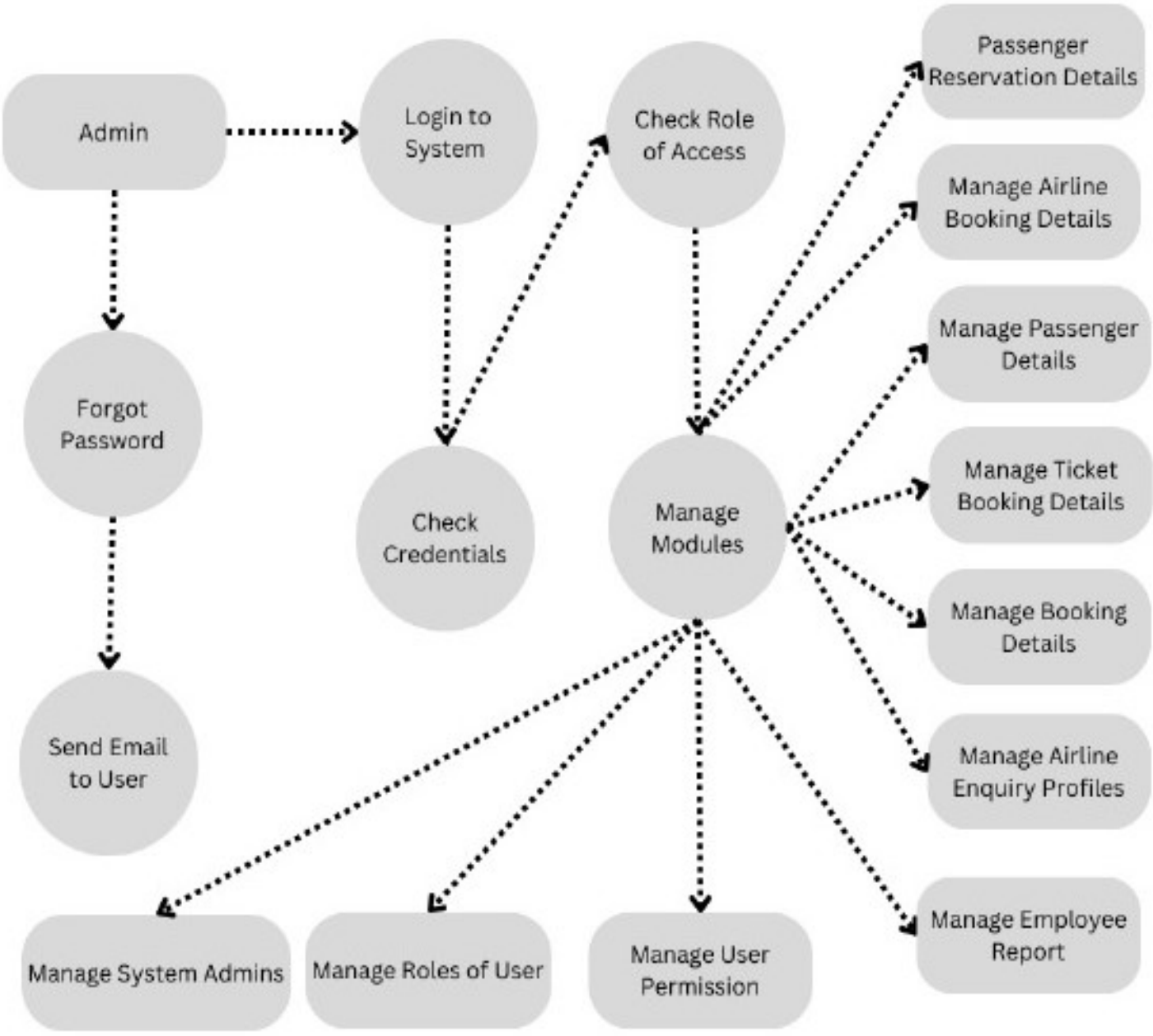
- **Data Flow Diagrams for ARS**
  - **Level 0 DFD**
  - **Level 1 DFD**
  - **Level 2 DFD**
- **Use Case Diagram for ARS**
- **Class Diagram for ARS**
- **Sequence Diagram for ARS**
- **Activity Diagram for ARS**
- **State Chart Diagram for ARS**
- **Jackson Structured Diagram for ARS**
- **Gantt Chart for ARS**
- **Network Diagram For ARS**



LEVEL ZERO DFD- AIRLINE RESERVATION SYSTEM

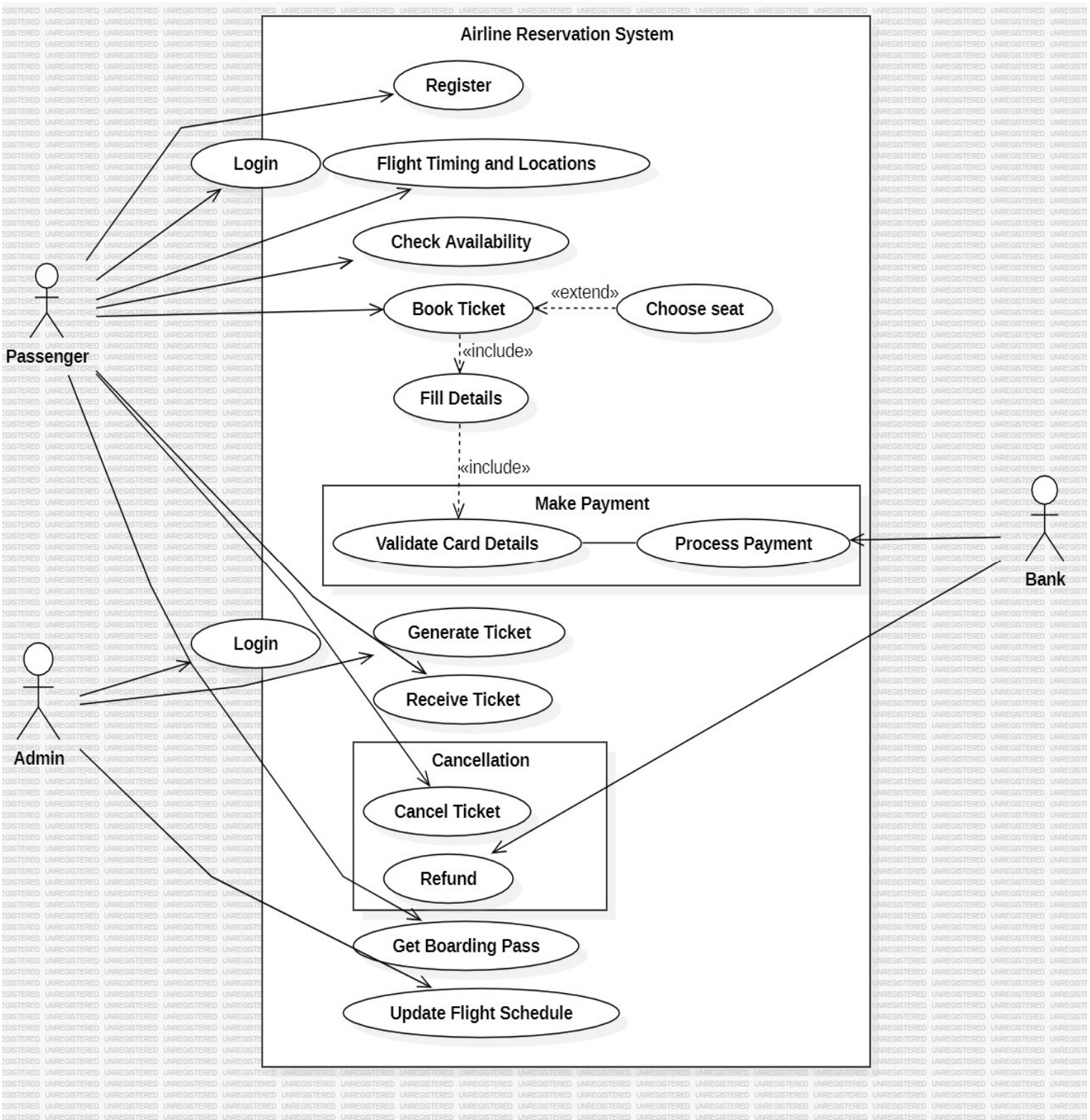


LEVEL ONE DFD - AIRLINES RESERVATION SYSTEM



Second Level DFD - Airline Reservation System

USE CASE DIAGRAM



Code for USE CASE DIAGRAM

```
@startuml
title AirlineReservationSystem

actor Passenger
actor Admin
actor Bank

Passenger --> (Register)
Passenger --> (Login)
Passenger --> (Flight Timing and Locations)
Passenger --> (Check Availability)
Passenger --> (Book Ticket)
Passenger --> (Receive Ticket)
Passenger --> (Get Boarding Passes)

Admin --> (Generate Ticket)
Admin --> (Update Flight Schedules)


(Choose Seat) --|> (Book Ticket) : extends

(Book Ticket) --> (Fill Details)
(Make Payment) --> (Generate Ticket)

rectangle "Cancellation" {
  (Cancel Ticket) --> (Refund)
}
Passenger --> (Cancel Ticket)

rectangle "Make Payment" {
  (Fill Details) --> (Validate Card Details)
  (Validate Card Details) --> (Process Payment)
}
Bank --> (Process Payment)
Bank --> (Refund)

@enduml
```

JAVA Code for CLASS DIAGRAM

Cancellation.java

```
import java.util.*;
public class Cancellation {

    private String passengerName;
    private int passengerID;
    private int phoneNumber;
    private int flightNumber;

    public Cancellation() {
        // Default constructor
    }
    public void setPassengerName(String name) {
        this.passengerName = name;
    }
    public void setPassengerID(int id) {
        this.passengerID = id;
    }
    public void setPhoneNumber(int phoneNo) {
        this.phoneNumber = phoneNo;
    }
    public void setFlightNumber(int flightNo) {
        this.flightNumber = flightNo;
    }
    public void addDetails() {
        // Implementation to add cancellation details
    }
    public void modifyDetails() {
        // Implementation to modify cancellation details
    }
    public void totalAmount() {
        // Implementation to calculate total cancellation amount
    }
}
```

AIRLINE ADMIN.java

```
import java.util.*;

public class AirlineAdmin {

    private String name;
    private int id;
    private String emailID;

    public AirlineAdmin() {
        // Default constructor
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void setEmailID(String emailID) {
        this.emailID = emailID;
    }

    public void updateInfo() {
        // Implementation to update administrator information
    }

    public void notifyPassengers() {
        // Implementation to notify passengers
    }

    public void updateStatus() {
        // Implementation to update status
    }
}
```

Booking System.java

```
import java.util.*;
public class BookingSystem {

    private String flightName;
    private String startingLocation;
    private String destination;
    private Date dateTime;
    private int numberOfSeats;

    public BookingSystem() {
        // Default constructor
    }
    public void chooseFlight() {
        // Implementation to choose a flight
    }
    public void selectTickets() {
        // Implementation to select tickets
    }
    public void makePayment() {
        // Implementation to make a payment
    }
    public void removeFlight() {
        // Implementation to remove a flight
    }
    public void modifySeats() {
        // Implementation to modify seats
    }
    public void cancel() {
        // Implementation to cancel the booking
    }
}
```

Refund.ja

```
import java.util.*;

public class Refund {

    public Refund() {
    }

    public float amount;
    public int accountNumber;

    public void sendOtp() {
        // TODO implement here
    }

    public void verifyOtp() {
        // TODO implement here
    }

    public void refundAmount() {
        // TODO implement here
    }
}
```

Ln 16, Col 31 100% Unix (LF) UTF-8

Payment.j

```
import java.util.*;

public class Payment {

    public Payment() {
    }

    public float amount;
    public String cardNumber;
    public int expiryDate;
    public int customerID;

    public void sendOtp() {
        // TODO implement here
    }

    public void verifyOtp() {
        // TODO implement here
    }

    public void makePayment() {
        // TODO implement here
    }

    public void giveReceipt() {
        // TODO implement here
    }
}
```

Ln 23, Col 6 100% Unix (LF) UTF-8

travellor.java

```
import java.util.*;

public class travellor extends Customer {
    public travellor() {
    }
}
```

Ln 5, Col 6 100% Unix (LF) UTF-8

airhostess.java

```
import java.util.*;

public class airhostess extends Customer {
    public airhostess() {
    }
}
```

Ln 1, Col 1 100% Unix (LF) UTF-8

Interface1.java

```
import java.util.*;

public interface Interface1 {

}
```

Ln 1, Col 1 100% Unix (LF) UTF-8

Customer.java

```
import java.util.*;
public class Customer {

    private String name;
    private String address;
    private int phoneNo;
    private int age;
    private String emailID;

    public Customer() {
        // Default constructor
    }
    public void register() {
        // Implementation to register a customer
    }
    public void login() {
        // Implementation for customer login
    }
    public void viewFlights() {
        // Implementation to view available flights
    }
    public void bookTickets() {
        // Implementation to book tickets
    }
    public void selectSeats() {
        // Implementation to select seats
    }
    public void makePayment() {
        // Implementation to make a payment
    }
    public void cancelTickets() {
        // Implementation to cancel tickets
    }
    public void logout() {
        // Implementation for customer logout
    }
}
```

Ln 33, Col 6 100% Unix (LF) UTF-8

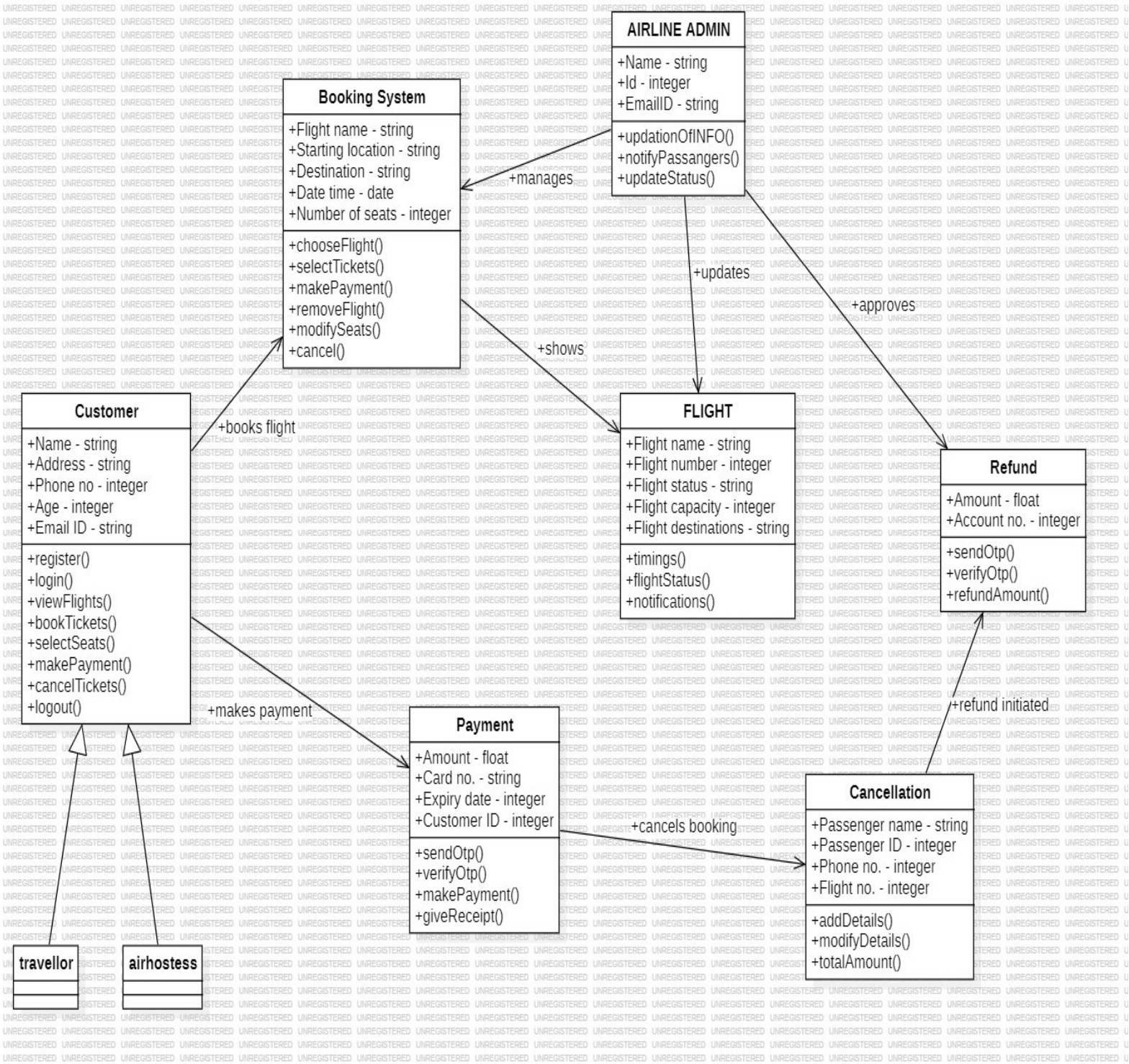
Flight.java

```
import java.util.*;
public class Flight {

    private String flightName;
    private int flightNumber;
    private String flightStatus;
    private int flightCapacity;
    private String flightDestinations;

    public Flight() {
        // Default constructor
    }
    public String getFlightName() {
        return flightName;
    }
    public void setFlightName(String flightName) {
        this.flightName = flightName;
    }
    public int getFlightNumber() {
        return flightNumber;
    }
    public void setFlightNumber(int flightNumber) {
        this.flightNumber = flightNumber;
    }
    public String getFlightStatus() {
        return flightStatus;
    }
    public void setFlightStatus(String flightStatus) {
        this.flightStatus = flightStatus;
    }
    public int getFlightCapacity() {
        return flightCapacity;
    }
    public void setFlightCapacity(int flightCapacity) {
        this.flightCapacity = flightCapacity;
    }
    public String getFlightDestinations() {
        return flightDestinations;
    }
    public void setFlightDestinations(String flightDestinations) {
        this.flightDestinations = flightDestinations;
    }
    public void timings() {
        // Implementation to display flight timings
    }
    public void flightStatus() {
        // Implementation to check flight status
    }
    public void notifications() {
        // Implementation to send notifications
    }
}
```

Class Diagram for Airlines Reservation



* Code for SEQUENCE DIAGRAM *

*01

```
@startuml
' Sequence Diagram for User Registration
actor Passenger
participant "ARS System" as System

Passenger -> System: Registration Details
System -> System: Validate and Create User Account
System -> Passenger: Notify Successful Registration

Passenger -> System: Login Credentials
System -> System: Verify Credentials
System --> Passenger: Grant Access/Deny Entry




' Sequence Diagram for Flight Search and Booking
actor Passenger
actor Bank
participant "ARS System" as System

Passenger -> System: Select Search Criteria
System -> System: Search for Available Flights
System --> Passenger: Display Available Flight Options
```

//www.plantuml.com/plantuml/dpng/jLNDRjiy4BphAOXSSlhmtNCee

*02

```
' Sequence Diagram for Flight Search and Booking
actor Passenger
actor Bank
participant "ARS System" as System

Passenger -> System: Select Search Criteria
System -> System: Search for Available Flights
System --> Passenger: Display Available Flight Options

Passenger -> System: Select Flight
Passenger -> System: Provide Passenger Details
Passenger -> Bank: Provide Payment Info
Bank --> Bank: Validate Payment
Bank --> System: Payment Authorization Response
System -> System: Process Payment and Confirm Booking
System --> Passenger: Notify Booking Confirmation




' Sequence Diagram for Ticket Cancellation and Refund Process
actor Passenger
actor Bank
```

//www.plantuml.com/plantuml/dpng/jLNDRjiy4BphAOXSSlhmtNCeejXj4

*03

```
' Sequence Diagram for Ticket Cancellation and Refund Process
actor Passenger
actor Bank
participant "ARS System" as System

Passenger -> System: Request Ticket Cancellation
System -> System: Verify Request and Cancel Ticket
System -> System: Initiate Refund Process

Passenger -> System: Provide Refund Details
System -> Bank: Initiate Refund Transaction
Bank --> Bank: Process Refund Transaction
Bank --> System: Refund Authorization Response
System --> Passenger: Notify Refund Status




' Sequence Diagram for Flight Schedule Update
actor Admin
```

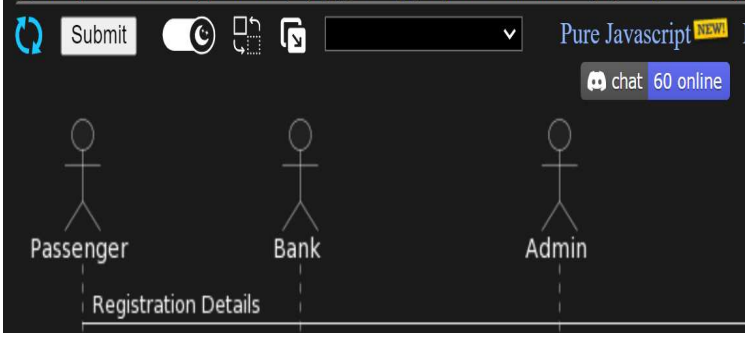
//www.plantuml.com/plantuml/dpng/jLNDRjiy4BphAOXSSlhmtNCeejXj4I

*04

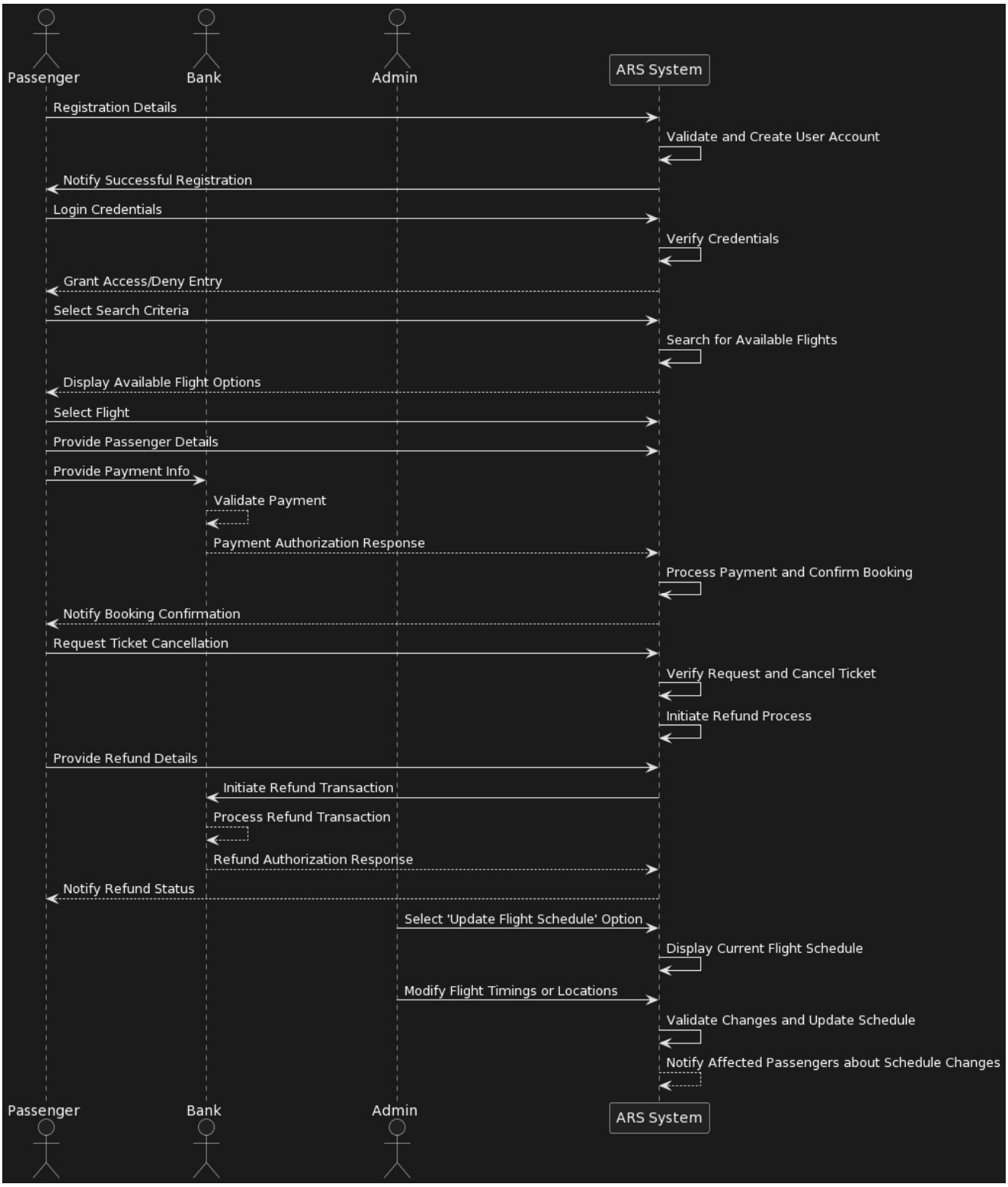
```
' Sequence Diagram for Flight Schedule Update
actor Admin
participant "ARS System" as System

Admin -> System: Select 'Update Flight Schedule' Option
System -> System: Display Current Flight Schedule
Admin -> System: Modify Flight Timings or Locations
System -> System: Validate Changes and Update Schedule
System --> System: Notify Affected Passengers about Schedule Changes
@enduml
```

//www.plantuml.com/plantuml/dpng/jLNDRjiy4BphAOXSSlhmtNCeejXj4I1j0olfVKEjv



* SEQUENCE DIAGRAM *

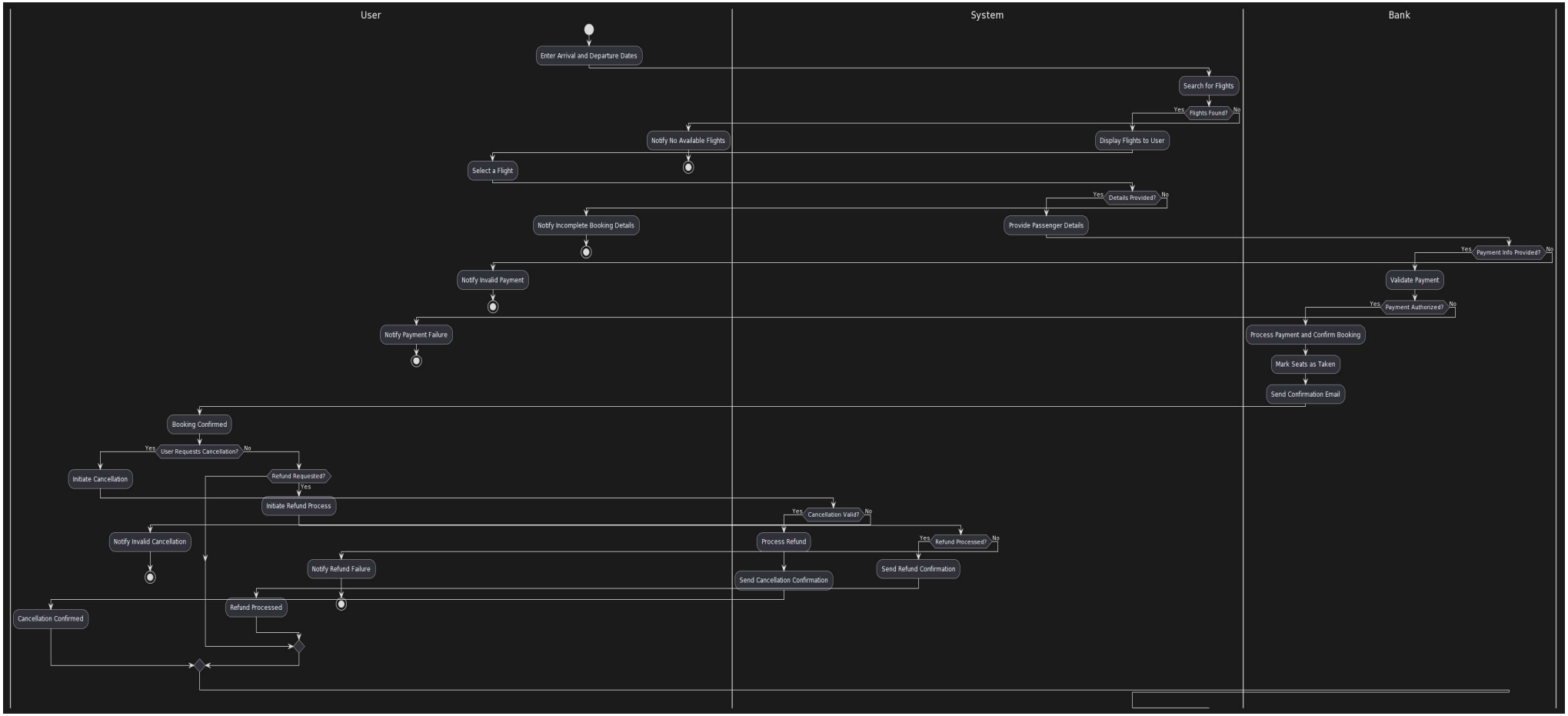


Code for ACTIVITY DIAGRAM

```
@startuml
' Activity Diagram for Airlines Reservation System

|User|
start
:Enter Arrival and Departure Dates;
|System|
:Search for Flights;
if (Flights Found?) then (Yes)
  :Display Flights to User;
  |User|
  :Select a Flight;
  |System|
  if (Details Provided?) then (Yes)
    :Provide Passenger Details;
    |Bank|
    if (Payment Info Provided?) then (Yes)
      :Validate Payment;
      if (Payment Authorized?) then (Yes)
        :Process Payment and Confirm Booking;
        :Mark Seats as Taken;
        :Send Confirmation Email;
        |User|
        :Booking Confirmed;
        if (User Requests Cancellation?) then (Yes)
          :Initiate Cancellation;
          |System|
          if (Cancellation Valid?) then (Yes)
            :Process Refund;
            :Send Cancellation Confirmation;
            |User|
            :Cancellation Confirmed;
          else (No)
            :Notify Invalid Cancellation;
            |User|
            stop
          endif
        else (No)
          if (Refund Requested?) then (Yes)
            :Initiate Refund Process;
            |System|
            if (Refund Processed?) then (Yes)
              :Send Refund Confirmation;
              |User|
              :Refund Processed;
            else (No)
              :Notify Refund Failure;
              |User|
              stop
            endif
          endif
        endif
      else (No)
        :Notify Payment Failure;
        |User|
        stop
      endif
    else (No)
      :Notify Invalid Payment;
      |User|
      stop
    endif
  else (No)
    :Notify Incomplete Booking Details;
    |User|
    stop
  endif
else (No)
  :Notify No Available Flights;
  |User|
  stop
endif
@enduml
```

ACTIVITY DIAGRAM

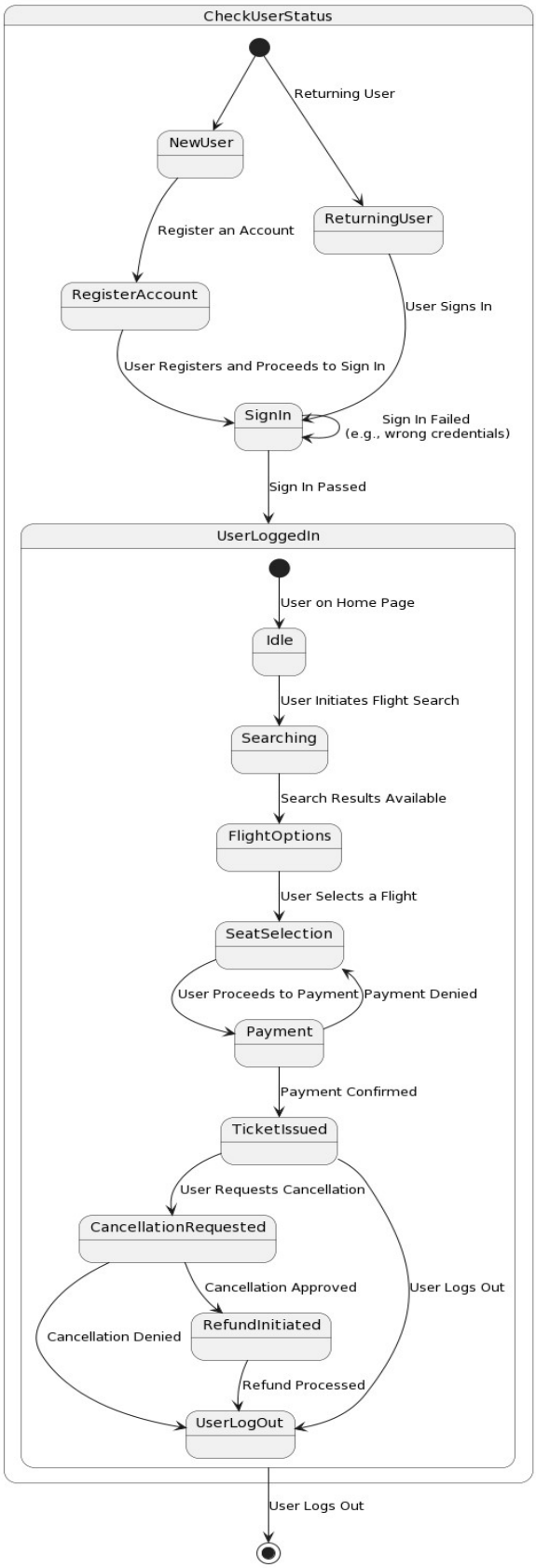


Code for STATE CHART DIAGRAM

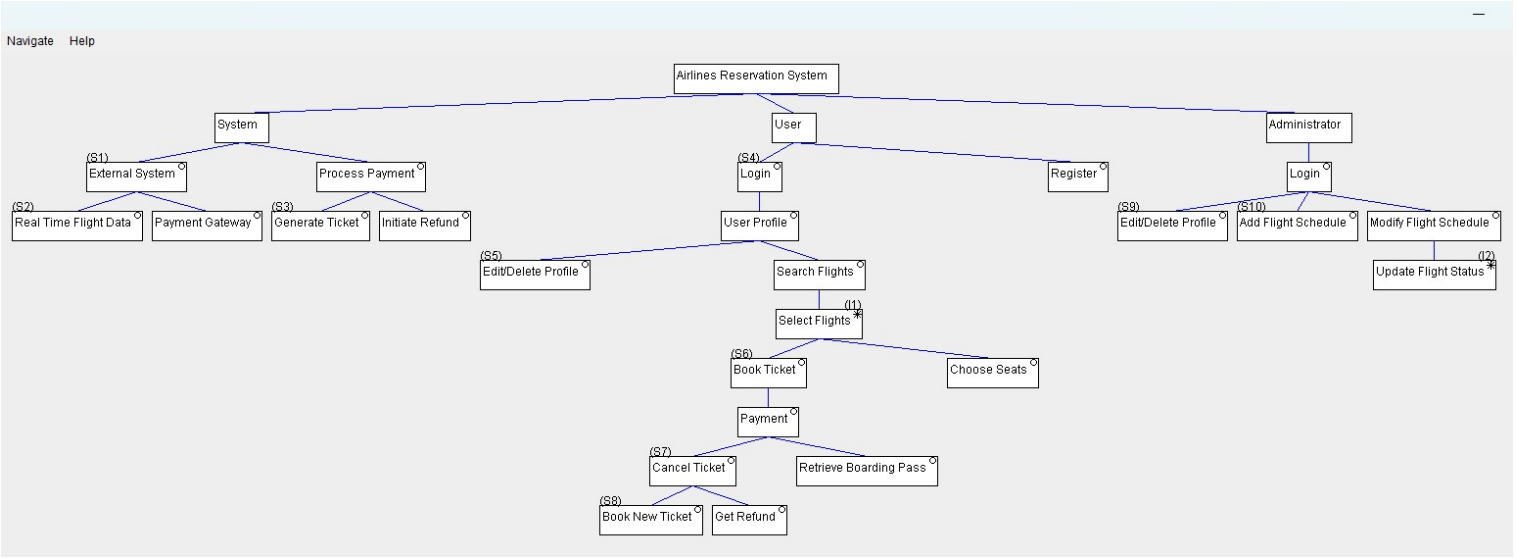
```
@startuml
state CheckUserStatus {
   [*] --> NewUser
   NewUser --> RegisterAccount : Register an Account
   RegisterAccount --> SignIn : User Registers and Proceeds to Sign In
   SignIn --> UserLoggedIn : Sign In Passed

   [*] --> ReturningUser : Returning User
   ReturningUser --> SignIn : User Signs In
   SignIn --> SignIn : Sign In Failed \n(e.g., wrong credentials)
}
state UserLoggedIn {
   [*] --> Idle : User on Home Page
   Idle --> Searching : User Initiates Flight Search
   Searching --> FlightOptions : Search Results Available
   FlightOptions --> SeatSelection : User Selects a Flight
   SeatSelection --> Payment : User Proceeds to Payment
   Payment --> TicketIssued : Payment Confirmed
   Payment --> SeatSelection : Payment Denied
   TicketIssued --> CancellationRequested : User Requests Cancellation
   TicketIssued --> UserLogOut : User Logs Out
   CancellationRequested --> RefundInitiated : Cancellation Approved
   RefundInitiated --> UserLogOut : Refund Processed
   CancellationRequested --> UserLogOut : Cancellation Denied
}
UserLoggedIn --> [*] : User Logs Out
@enduml
```

* STATE CHART DIAGRAM *



JACKSON STRUCTURED DIAGRAM



C-Code for JSD

```
int main()
{
  if (S1)
    if (S2)
      /* Real Time Flight Data */
    else
      /* Payment Gateway */
  else
    if (S3)
      /* Generate Ticket */
    else
      /* Initiate Refund */
  if (S4)
    if(1==1)
      if (S5)
        /* Edit/Delete Profile */
      else
        while (I1)
          if (S6)
            if(1==1)
              if (S7)
                if (S8)
                  /*  Book  New  Ticket
*/
                else
                  /* Get Refund */
              else
                /*   Retrieve   Boarding
Pass */
            else
              /* Choose Seats */
    else
      /* Register */
    if(1==1)
      if (S9)
        /* Edit/Delete Profile */
      else if (S10)
        /* Add Flight Schedule */
      else
        while (I2)
          /* Update Flight Status */
}
```

GANTT CHART

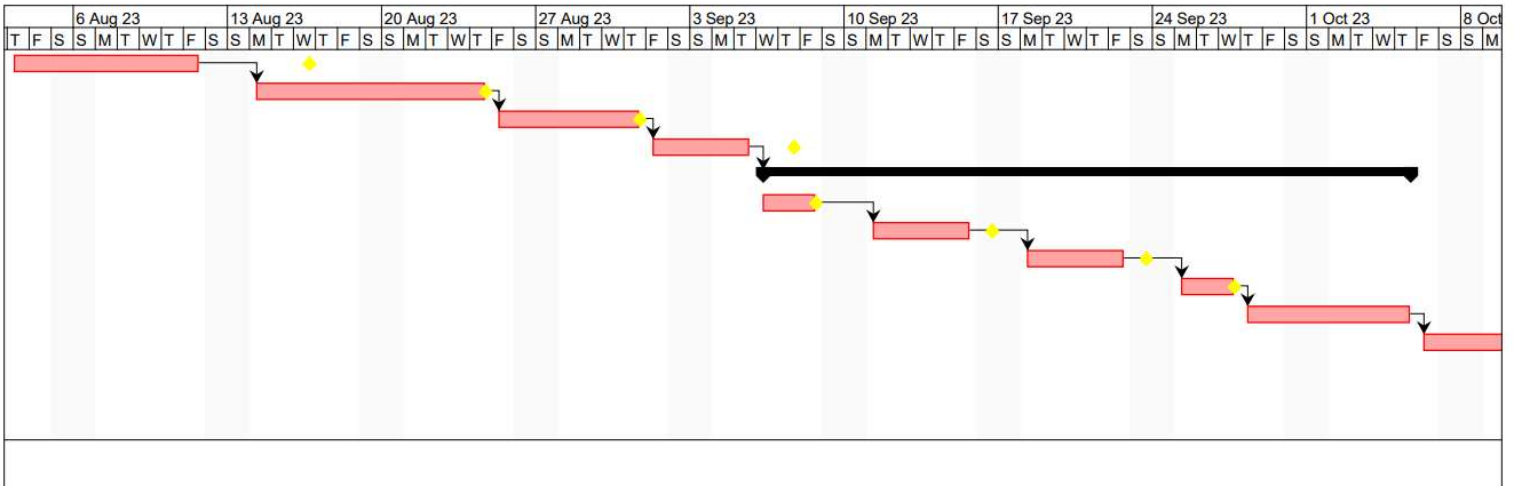
Airlines Reservation System - C:\Users\Ved\Downloads\Airlines Reservation System.pod

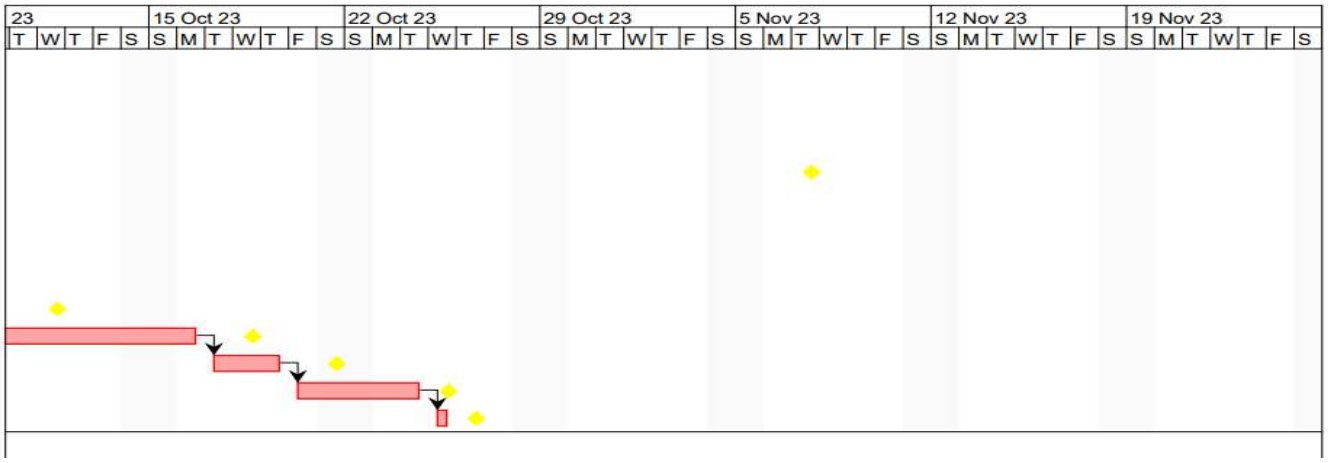
ProjectLibre

File Task Resource View

Save Open New Save as Close — File
Print Preview PDF — Print
Projects Information Calendar Projects Dialog Save Baseline Clear Baseline Update — Project

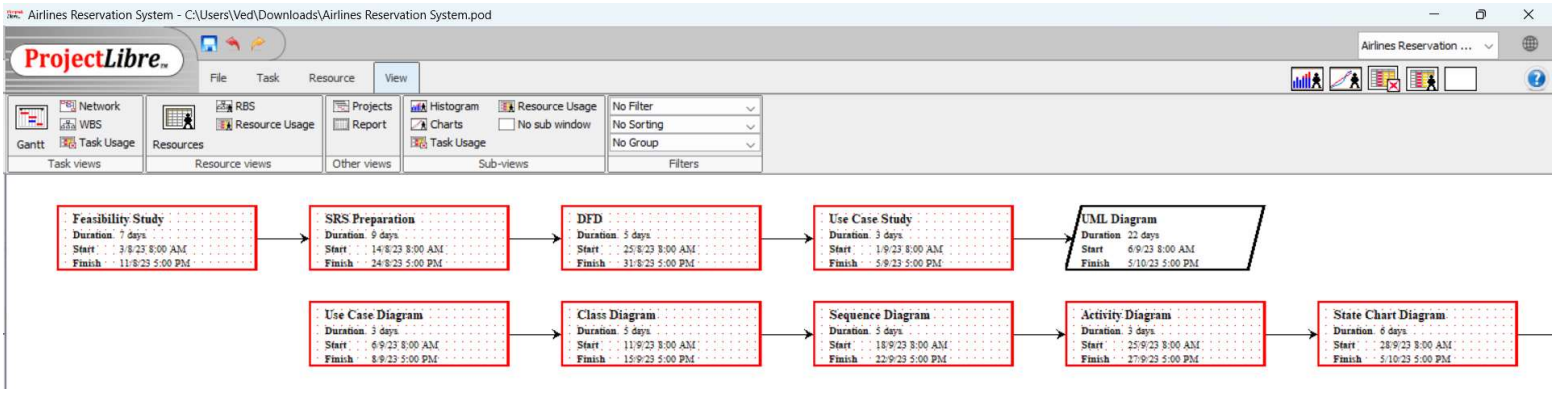
		Name	Duration	Deadline	Start	Finish	Predecessors
1		Feasibility Study	7 days	16/8/23 5:00 PM	3/8/23 8:00 AM	11/8/23 5:00 PM	
2		SRS Preparation	9 days	24/8/23 5:00 PM	14/8/23 8:00 AM	24/8/23 5:00 PM	1
3		DFD	5 days	31/8/23 5:00 PM	25/8/23 8:00 AM	31/8/23 5:00 PM	2
4		Use Case Study	3 days	7/9/23 5:00 PM	1/9/23 8:00 AM	5/9/23 5:00 PM	3
5		**UML Diagram**	22 days	**7/11/23 5:00 PM**	**6/9/23 8:00 AM**	**5/10/23 5:00 PM**	**4**
6		Use Case Diagram	3 days	8/9/23 5:00 PM	6/9/23 8:00 AM	8/9/23 5:00 PM	
7		Class Diagram	5 days	16/9/23 5:00 PM	11/9/23 8:00 AM	15/9/23 5:00 PM	6
8		Sequence Diagram	5 days	23/9/23 5:00 PM	18/9/23 8:00 AM	22/9/23 5:00 PM	7
9		Activity Diagram	3 days	27/9/23 5:00 PM	25/9/23 8:00 AM	27/9/23 5:00 PM	8
10		State Chart Diagram	6 days	11/10/23 5:00 PM	28/9/23 8:00 AM	5/10/23 5:00 PM	9
11		Jakson Structure Diagram	7 days	18/10/23 5:00 PM	6/10/23 8:00 AM	16/10/23 5:00 PM	10
12		UML Code Generation	3 days	21/10/23 5:00 PM	17/10/23 8:00 AM	19/10/23 5:00 PM	11
13		GNATT Chart	3 days	25/10/23 5:00 PM	20/10/23 8:00 AM	24/10/23 5:00 PM	12
14		Network Diagram	1 day?	26/10/23 5:00 PM	25/10/23 8:00 AM	25/10/23 5:00 PM	13





NETWORK DIAGRAM

L.H.S.



R.H.S.