PURBANCHAL UNIVERSITY

HIMALAYAN WHITEHOUSE INTERNATIONAL COLLEGE

PUTALISADAK, KATHMANDU

Project-IV Report

On

"FLIGHT MANAGEMENT SYSTEM"

Submitted By:

Chirag Sedhai

Rahul Shrestha

Sunil Kumar Yadav

Submitted To:

THE DEPARTMENT OF SCIENCE & TECHNOLOGY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR IN INFORMATION AND TECHNOLOGY

December,2025

Kathmandu, Nepal

CERTIFICATE OF APPROVAL

The Project entitled “Flight Management System” submitted by **Chirag Sedhai, Rahul Shrestha and Sunil Kumar Yadav** in partial fulfillment of the requirement for the degree of “Bachelor of Information and Technology” has been accepted as a bona fide record of work carried out by them in the department.

Er. Aashish Lamsal

Supervisor

Er. Bimal Sharma

Head of Department

Bachelor of Information and Technology

External

Examiner

ACKNOWLEDGEMENT

We are very grateful of our supervisor for his constant encouragement, cooperation and guiding us through this project. We would like to thank our lecturer, friends as well as the seniors who helped us a lot.

The desired success obtained during wouldn't have been attained without the facilities and other guideline provided by our Head of Department **Er. Bimal Sharma**.

At last, we would like to thank to all the individual who directly or indirectly helped us in the completion of this project.

Thanking you,
Chirag Sedhai
Rahul Shrestha
Sunil Kumar Yadav

ABSTRACT

The Flight Management System is a Java-based software application designed to automate and manage essential flight operations such as flight scheduling, passenger booking, ticket management, and administrative control. Developed using object-oriented programming principles and integrated with a database for secure data storage and retrieval, the system ensures accuracy, efficiency, and reliability in handling flight-related information. It provides role-based access for administrators and users, reducing manual workload and minimizing human errors while improving overall operational efficiency. By leveraging Java's platform independence, robustness, and security features, the system offers a scalable and maintainable solution suitable for academic demonstration and practical implementation, with scope for future enhancements such as real-time flight tracking, online payment integration, and advanced reporting.

Keywords

1. Flight Management System
2. Flight Reservation
3. Customer Management
4. Ticket Booking
5. MySQL Database
6. Admin Module
7. User Authentication

TABLE OF CONTEXT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBEREVIATIONS

DFD	Data Flow Diagram
ER	Entity Relationship
FMS	Flight Management System
GUI	Graphical User Interface

CHAPTER 1
INTRODUCTION

1.1 Background

In recent years, software-based management systems have significantly improved the efficiency and accuracy of aviation operations. A Flight Management System integrates multiple activities such as scheduling, booking, and record maintenance into a single centralized platform. Java is widely used for such applications due to its platform independence, strong security features, and support for object-oriented programming. These characteristics allow developers to build scalable, modular, and maintainable systems. Implementing a Flight Management System using Java not only enhances technical understanding but also demonstrates how real-world problems in aviation can be addressed through structured software solution.

1.2 Problem Statement

Manual and semi-automated flight management processes are inefficient in handling large and dynamic datasets. These systems are prone to human errors, data duplication, slow processing, and poor security. There is a need for a reliable and automated system that can manage flight operations effectively, ensure data integrity, and provide quick access to accurate information for both administrators and users.

1.3 Objective

The main objective of this project is to develop a Flight Management System using Java that automates flight scheduling, passenger management and booking operations. The system aims to reduce manual workload, improve data accuracy, and enhance overall operational efficiency. Additionally, the project seeks to apply object-oriented programming principles and database integration to create a secure, modular, and esay-to-maintain application suitable for academic and practical use.

1.4 Scopes and limitations

Some scopes and limitations of our purposed system are listed below:

Scopes:

- Management of flight schedules and flight details
- Passenger information storage and management
- Booking and cancellation of flights
- Role-based access for administrators and users
- Secure database integration for data storage
- User-friendly interface for smooth interaction

Limitations:

- Does not support real-time flight tracking
- Limited scalability for large airline operations
- No online payment gateway integration
- Lacks advanced security features such as multi-factor authentication
- No integration with external airline or airport systems
- Performance depends on system hardware and database configuration

CHAPTER 2
LITRATURE REVIEW

2.1 Study of the Existing System

Previous studies and existing flight management systems reveal that traditional approaches largely relied on manual record-keeping or basic computerized applications to manage flight schedules, passenger details, and bookings. These systems were often standalone, lacked proper integration with databases, and required significant human intervention. Even in semiautomated systems, data redundancy, slow processing, limited user access control, and poor error handling were common issues. Some existing software solutions focus only on specific functionalities such as ticket booking or flight scheduling, rather than providing a fully integrated management platform. Additionally, many legacy systems are difficult to scale or modify due to rigid design structures and lack of modularity.

2.2 What's New in Our Project?

The proposed **Flight Management System** using Java introduces a more structured and efficient approach by applying object-oriented programming principles and integrating a centralized database for secure and consistent data management. Unlike existing systems, this project provides role-based access for administrators and users, improving security and usability. The system combines multiple flight-related operations such as flight management, passenger records, and booking processes into a single, cohesive platform. Its modular design enhances maintainability and allows future extensions, such as real-time flight updates, reporting modules, and payment integration. This project not only improves operational efficiency but also serves as a scalable and extensible model that addresses the limitations observed in earlier flight management systems.

CHAPTER 3
SYSTEM ANALYSIS

3.1 System Analysis

System analysis is the process of studying a system or a specific aspect of a system to understand its components, functions, interactions and requirements. There are various types of system analysis such as requirement analysis, feasibility analysis, system design analysis, risk analysis and others.

3.1.1 Requirement Analysis

Requirement analysis defines the functional and non-functional needs of the Flight Management System to ensure effective and reliable operation.

A. Functional requirement

It described the specific functions that the system must perform.

- User authentication: the system should allow users to securely log in using their username and password.
- Grocery Shop Management: the system should provide functionality for adding, updating, editing and deleting employee records.

B. Non-Functional Requirements

Non-functional requirements describe how the system should perform. The system should provide fast and accurate data processing, ensure data security and integrity through database control, and offer a user-friendly interface for easy navigation. It must be reliable with minimal system downtime, maintainable through modular Java-based design, and portable across different platforms due to Java's platform independence.

3.1.2 Feasibility Analysis

A feasibility study is a systematic analysis to determine the practicality and potential success of a proposed project or venture.

1. **Technical Feasibility:** Technical feasibility is concerned with the availability of hardware and software required for the development of the system. After the study we came to conclusion that we can proceed further with the tools and development environment chosen by us.
2. **Operational Feasibility:** Operational feasibility is all about problems that may arise during operations. There are two aspects related with the issue. What is the probability that the

solution developed may not be put to use or may not work? Though, there is very least possibility of management being averse to the solution, there is significant probability that the end users may not be interested in using the solution due to lack of training, insight, etc.

3. **Time Feasibility:** Time feasibility refers to the assessment of whether a proposed project can be completed within a reasonable timeframe. After the study of the tasks involved in completing the project including requirement gathering, frontend development, backend development, coding, testing and others, we have concluded that the project timeline appears feasible. By carefully analysing tasks, resource availability & potential risks or delays, we can assess the feasibility of meeting the projects deadlines and make adjustments to the schedule as needed to ensure timely completion.
4. **Cost Feasibility:** The cost feasibility analysis for the grocery shop management system involves evaluating expenses related to development, maintenance, and operational aspects. This includes estimating costs for software development, infrastructure, ongoing maintenance, integration, and any other associated expenses. After the study, it was determined that the employee management system implemented in Java is cost-feasible, with expenses well within the allocated budget.
5. **Legal Feasibility:** Legal feasibility entails ensuring compliance with relevant laws and regulations governing data privacy, intellectual property, and employment practices. Specifically, for the employee management system, it involves adherence to data protection laws ensuring proper handling and security of employee data. The system was found to be legally feasible since only the authorized admin is allowed to use the system which will further enhance data privacy.

CHAPTER 4
SYSTEM DESIGN

4.1 SDLC Model

The Software Development Life Cycle (SDLC) model is a structured approach used by software development teams to design, develop, test, deploy, and maintain software systems. It encompasses a series of phases or stages, each with specific activities and deliverables. Common SDLC models include Waterfall, prototype & spiral model with its own set of principles, methodologies, and best practices tailored to different project requirements and organizational needs.

The phases in the Software Development Life Cycle (SDLC) typically include:

1. **Planning**: This phase involves defining the project scope, objectives, timelines, and resources required. It may also include feasibility studies and risk assessments to ensure the project's viability.
2. **Requirement Analysis**: During this phase, the development team gathers and analyses requirements from stakeholders, such as users, customers, and business owners. The goal is to understand the needs and expectations of the software system to be developed.
3. **Design**: In this phase, the system architecture and design specifications are created based on the requirements gathered. This includes defining the software components, data structures, interfaces, and algorithms to be used in the system.
4. **Implementation**: Also known as coding or development, this phase involves translating the design specifications into actual code. Developers write, compile, and test the code to ensure it meets the requirements and design standards.
5. **Testing**: Once the code is developed, it undergoes rigorous testing to identify and fix defects or bugs. Testing includes various techniques such as unit testing, integration testing, system testing, and UAT.
6. **Deployment**: After successful testing, the software is deployed or released to the production environment. This phase involves installing the software on users' machines or servers and configuring it for use.

7. **Maintenance**: The final phase involves maintaining and supporting the software post-deployment. This includes addressing any issues or bugs reported by users, making enhancements or updates as needed, and ensuring the software remains functional and efficient over time.

These phases may vary slightly depending on the specific SDLC model or methodology being used, but they generally encompass the key activities involved in software development projects.

4.2 Selected Model

Our project aims to develop a **Flight Management System** using the Java programming language. After carefully evaluating various Software Development Life Cycle (SDLC) models, we concluded that the **Spiral Model** is the most suitable for our project. The Spiral Model is an iterative and risk-driven approach to software development that combines elements of the Waterfall Model and Prototyping. Development proceeds in a series of repeated cycles called spirals, where each cycle involves planning, risk analysis, design, implementation, and testing. Unlike linear models, the Spiral Model allows for continuous refinement of the system through repeated evaluation and user feedback. At the end of each spiral, the developed features are reviewed, and risks are identified and addressed before moving to the next iteration. This model emphasizes early detection and mitigation of risks, making it suitable for projects where requirements may evolve or where system reliability is critical, such as flight operations.

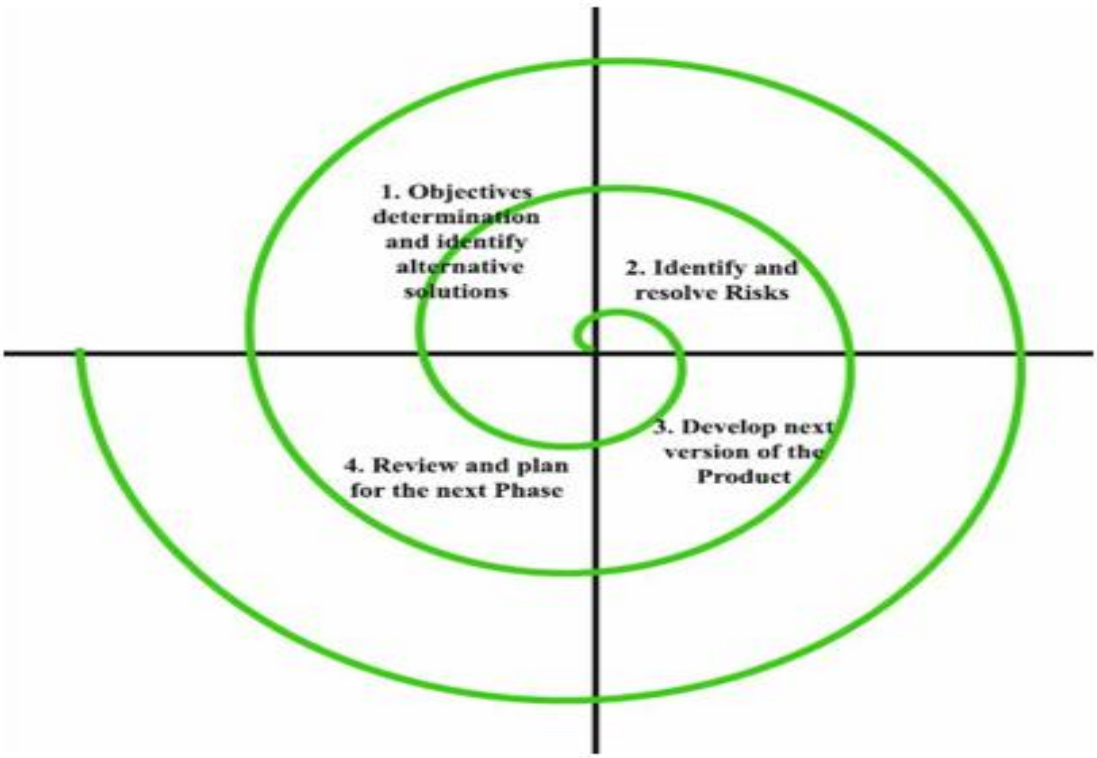

Figure 4.2: Spiral Model

The Spiral Model was selected for the development of the Flight Management System due to its iterative and flexible nature. This model allows the project to progress in repeated cycles of planning, design, implementation, and testing. Risk analysis is performed in each iteration to identify and reduce potential issues related to functionality, performance, and security. Continuous evaluation helps improve the system at every stage of development, ensuring accurate flight management, booking operations, and data integrity. Each cycle produces a more refined and reliable version of the system. Proper documentation is maintained throughout all phases of the project. Therefore, the Spiral Model is an effective and appropriate choice for developing a robust and efficient Flight Management System.

4.3 Context Diagram

A context diagram is a high-level visual representation that illustrates the scope and boundaries of a system or process within its environment. It provides an overview of the interactions between the system being analysed and its external entities, such as users, other systems, or external stakeholders. Context diagrams help stakeholders understand the context in which the system operates and facilitate discussions about its requirements, interfaces, and dependencies.

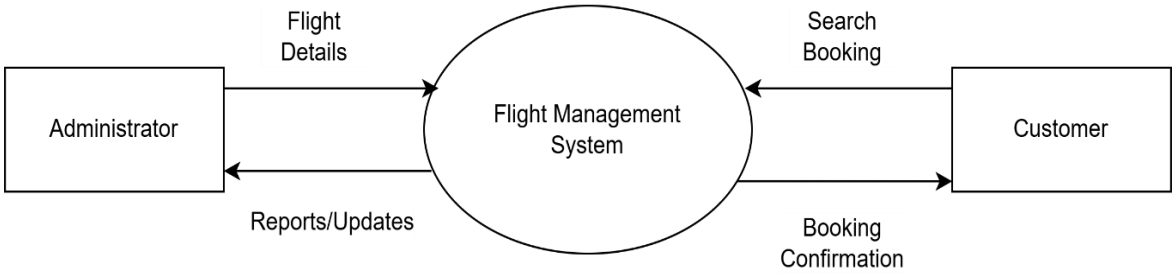

Figure 4.3: Context Diagram

4.4 DFD

DFD stands for Data Flow Diagram. It's a graphical representation that illustrates how data flows through a system or process. DFDs consist of processes, data stores, data flows, and external entities. Processes represent activities or transformations that occur within the system, data stores depict where data is stored, data flows show the movement of data between processes and data stores, and external entities represent sources or destinations of data outside the system. The level 1 DFD of our proposed system is as shown below:

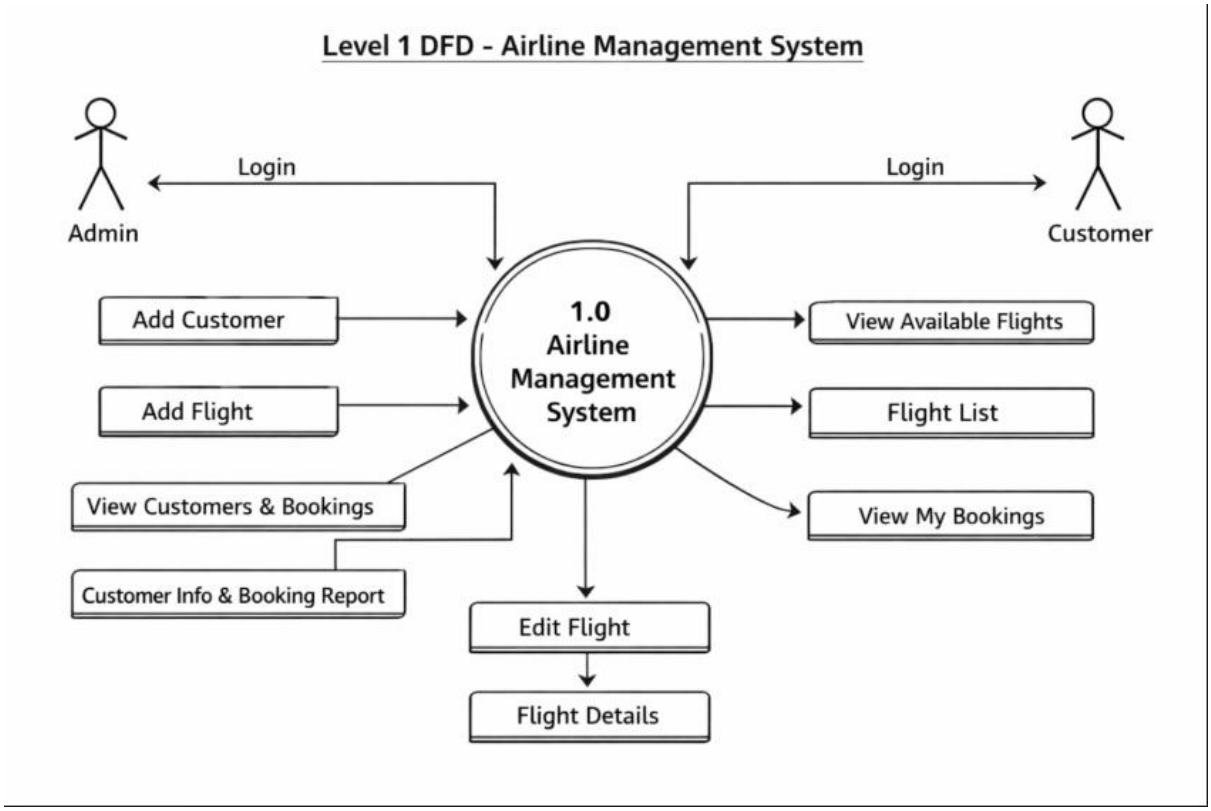

Figure 4.4: Data Flow Diagram

4.5 ER-Diagram

An ER diagram is a visual representation that depicts the relationships among entities within a database. ER diagrams help in understanding the structure of a database and are commonly used during the database design phase to model the relationships between different entities and their attributes. The ER diagram of our purposed system which will be further modified according to our requirements.

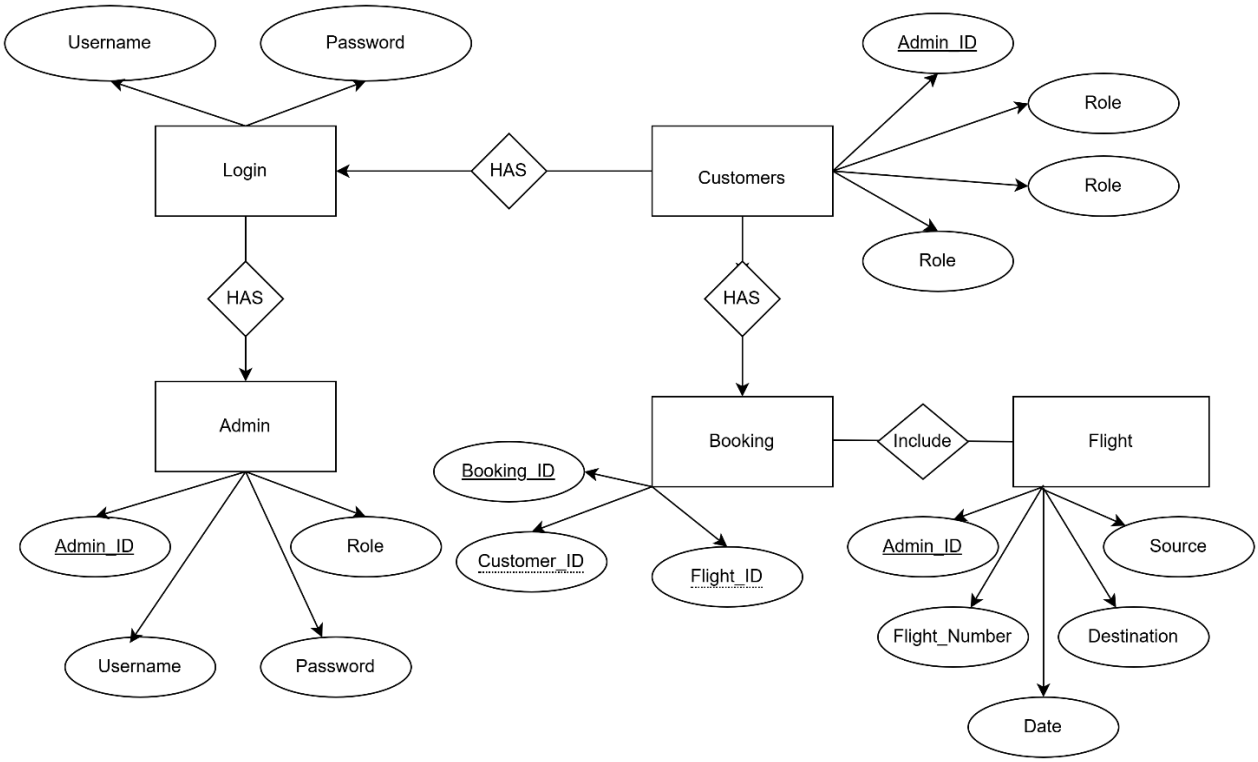

Figure 4.5: ER diagram

4.6 Use Case Diagram

A use case diagram is a type of behavioural diagram in Unified Modelling Language (UML) that illustrates the interactions between actors (users or external systems) and a system to accomplish specific goals or tasks. It shows the functionality of a system from the perspective of its users and helps to understand how users interact with the system.

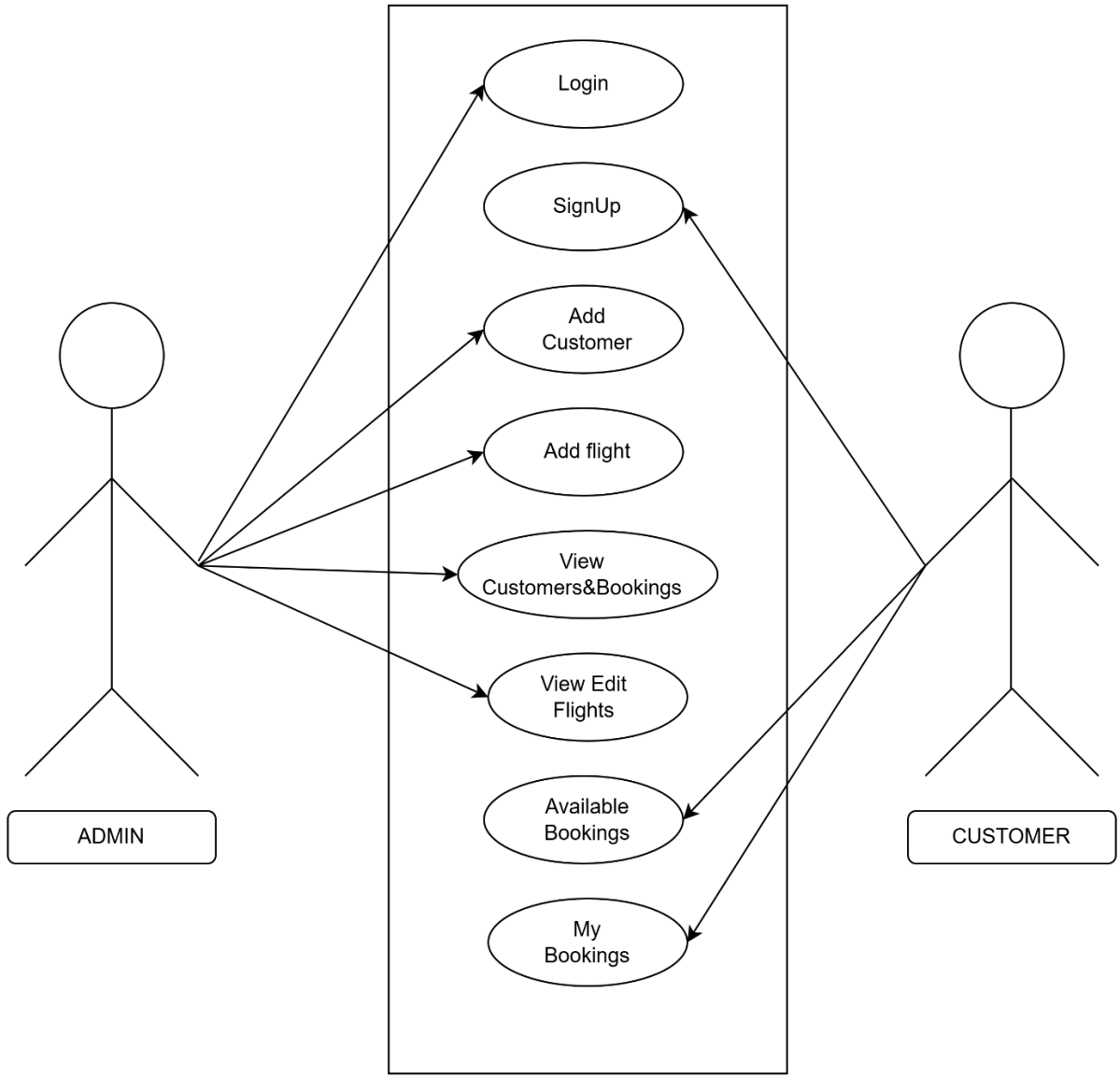

Figure 4.6: Use Case Diagram

4.7 Class Diagram

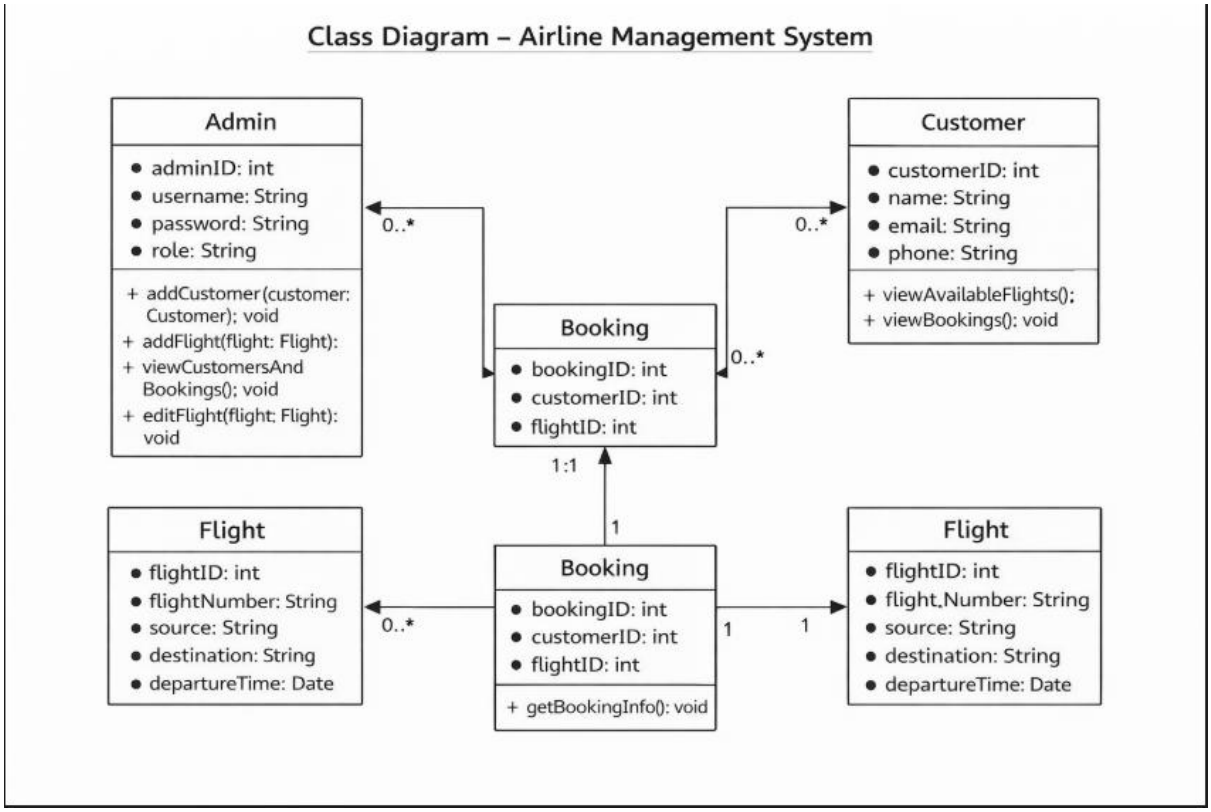

Figure 4.7: Class diagram

CHAPTER 5
IMPLEMENT & TESTING

5.1 Tools used

Table 1: Tools used

Apache Netbeans IDE 20	IDE used for GUI design, coding & debugging, and also supports multiple programming languages.
SQL YOG	GUI tool for managing MYSQL database and relational database

5.2 Gantt chart

We have outlined the timeline for the implementation of the GSMS below using a Gantt chart. This chart illustrates the major tasks, their dependencies and the estimated duration for each task.

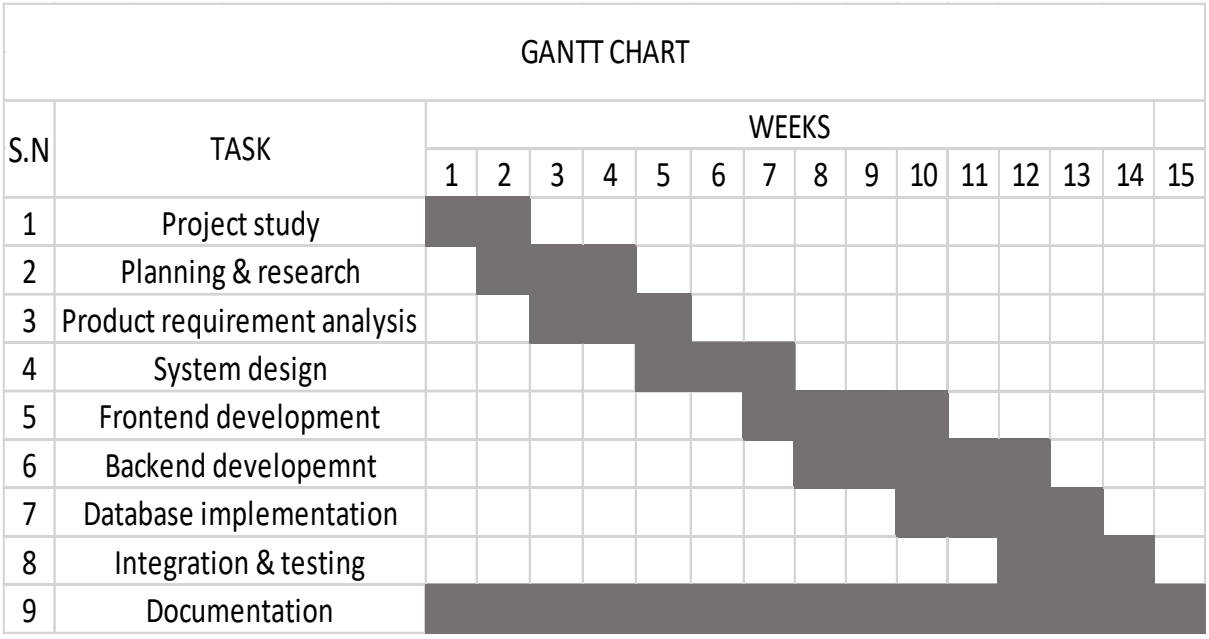

Figure 6: Gantt chart

5.3 Testing

Software testing is a process of analysing an application's functionality as per the requirements. If we want to ensure that our software is bug-free or stable, we must perform the various types of software testing because testing is the only method that makes our application bug free. Here's a brief overview of various types of software testing:

A. **Functional Testing:** In functional testing, all the components are tested by giving the value, defining the output, and validating the actual output with the expected value. Functional testing. In functional testing, all the components are tested by giving the value, defining the output, and validating the actual output with the expected value.
    i. Unit Testing: Unit testing is the first level of functional testing in order to test any software. In this, the test engineer will test the module of an application independently or test all the module functionality is called unit testing.
    ii. Integration Testing: Once we are successfully implementing the unit testing, we will go integration testing. It is the second level of functional testing, where we test the data flow between dependent modules or interface between two features is called integration testing.
    iii. System Testing: Whenever we are done with the unit and integration testing, we can proceed with the system testing. In system testing, the test environment is parallel to the production environment. In this type of testing, we will undergo each attribute of the software and test if the end feature works according to the business requirement. And analysis the software product as a complete system.
B. **Non-Functional Testing:** Non-functional testing is a type of software testing that focuses on the attributes of a system that do not relate to specific behaviours or functions. Instead, it assesses qualities such as performance, reliability, scalability, usability, security, and compatibility. Non-functional testing helps evaluate how well a system meets its requirements in terms of these attributes and ensures that it performs satisfactorily under various conditions beyond functional correctness.

For the completion of our mini project, we will be using functional testing method as it ensures that software meets specified requirements by testing individual functions or features, detecting bugs early, and ensuring user satisfaction and overall quality.

CHAPTER 6
CONCLUSION & DISCUSSION

6.1 Conclusion

The Flight Management System is a comprehensive solution designed to streamline and automate the process of flight booking, customer management, and flight scheduling. The system provides an intuitive interface for both administrators and customers, enabling efficient management of flights, bookings, and user data. By integrating Java with database management, the system ensures secure storage and retrieval of information, reduces manual errors, and enhances overall operational efficiency. The project demonstrates the practical application of object-oriented programming, database connectivity, and GUI development in a real-world scenario.

6.2 Future enhancement

Looking ahead, there are several potential enhancements that could further improve the **Flight Management System:**

- Online Payment Integration
- Mobile Application Support
- Real-time Flight Status Updates
- Seat Selection Feature
- Multi-language Support
- Advanced Reporting

These enhancements would not only improve the functionality and user experience but also position the system for scalability and future growth.

REFERENCES

1. Barua & Kaiser — *Enhancing resilience and scalability in travel booking systems using microservices architectures* — Discusses how scalable system design improves flight reservation systems' performance and fault tolerance in modern applications.
2. [*Airline Reservation System* — ResearchGate paper detailing airline reservation system modules, booking and cancellation processes.
3. *Design and Implementation of an Online Airline Reservation Information System* — Typical undergraduate project report outlining system use, troubleshooting and database design (PHP/Java & MySQL example).

ANNEX

Figure 7: Login

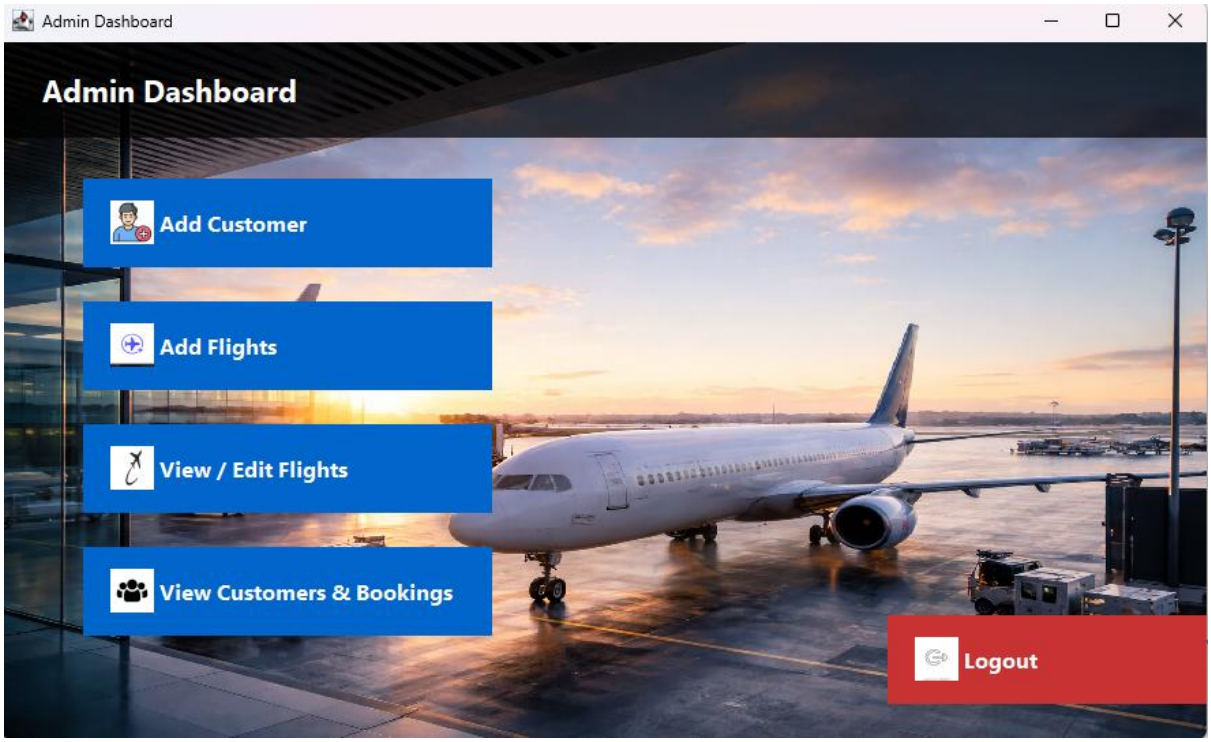

Figure 8: Admin page

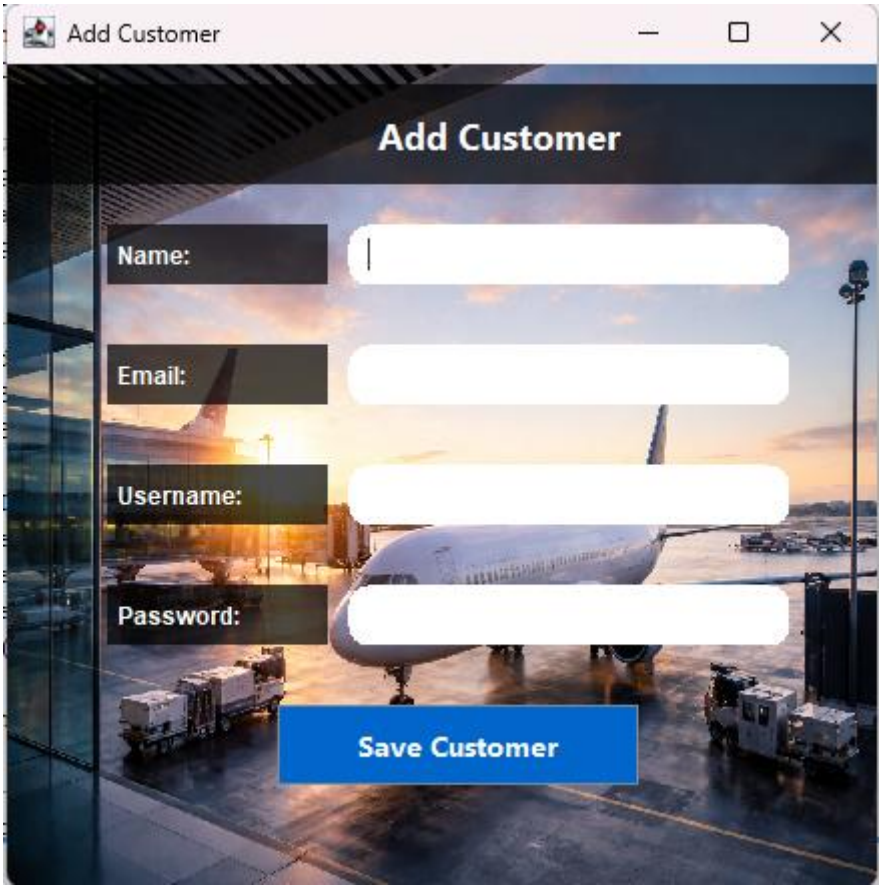

Figure 9: Add customer

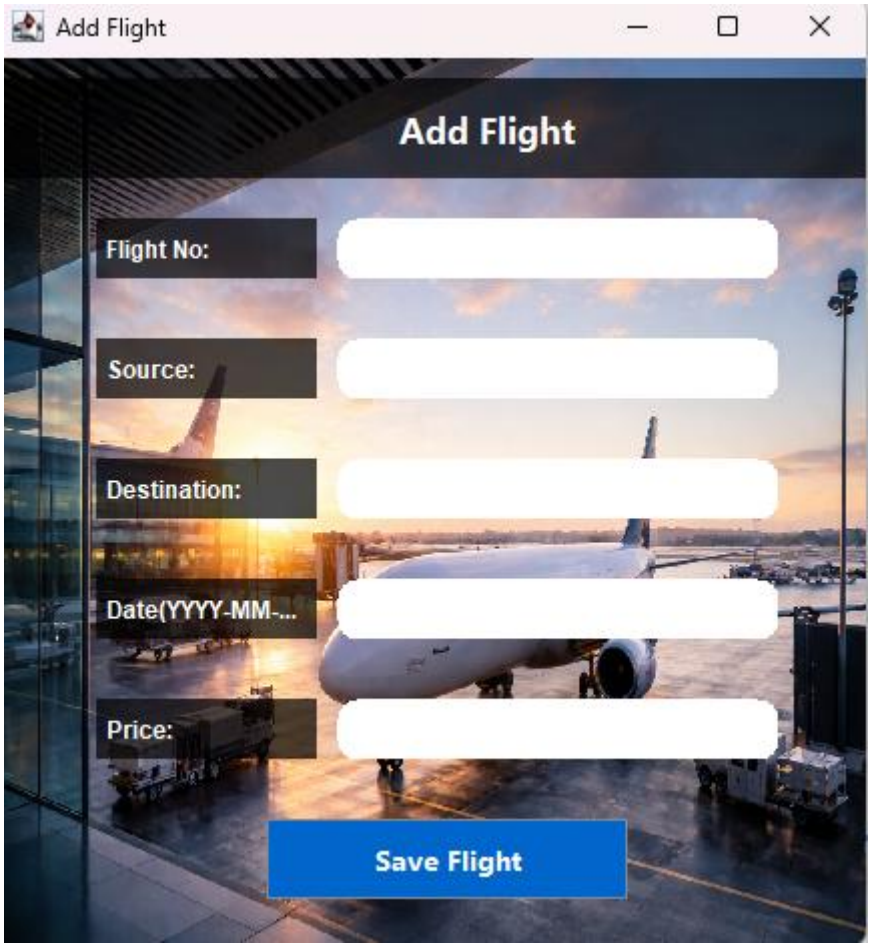

Figure 10: Add flight

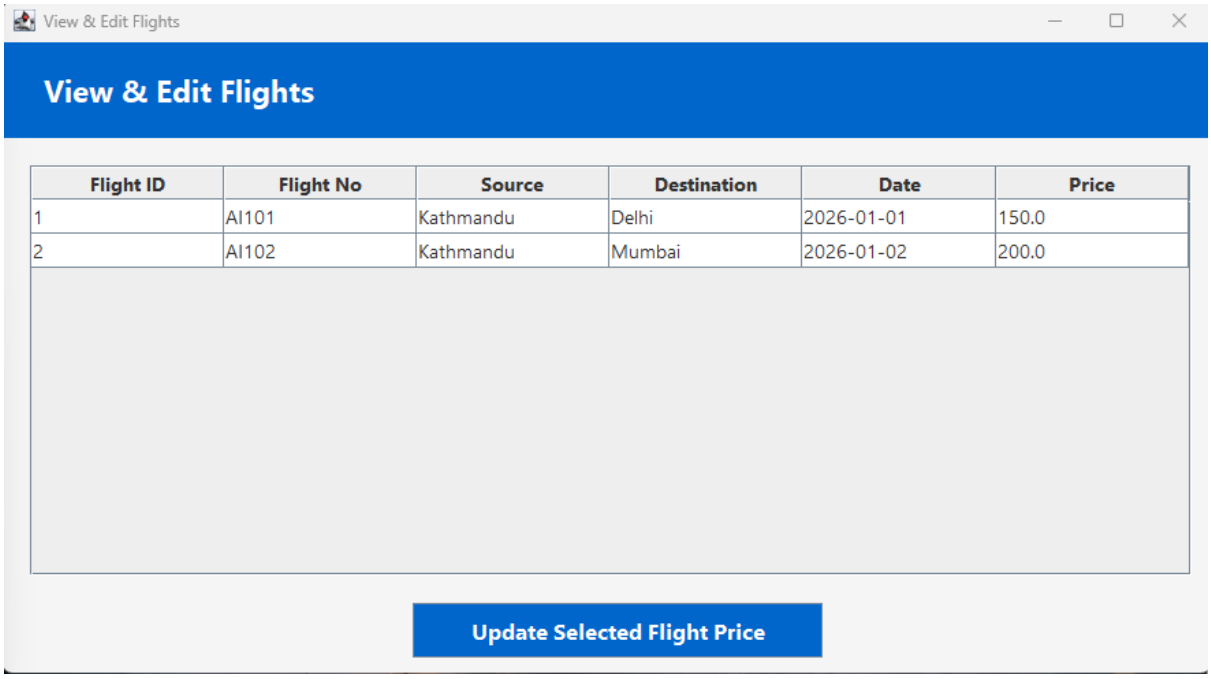

Figure 11: View & Edit Flights

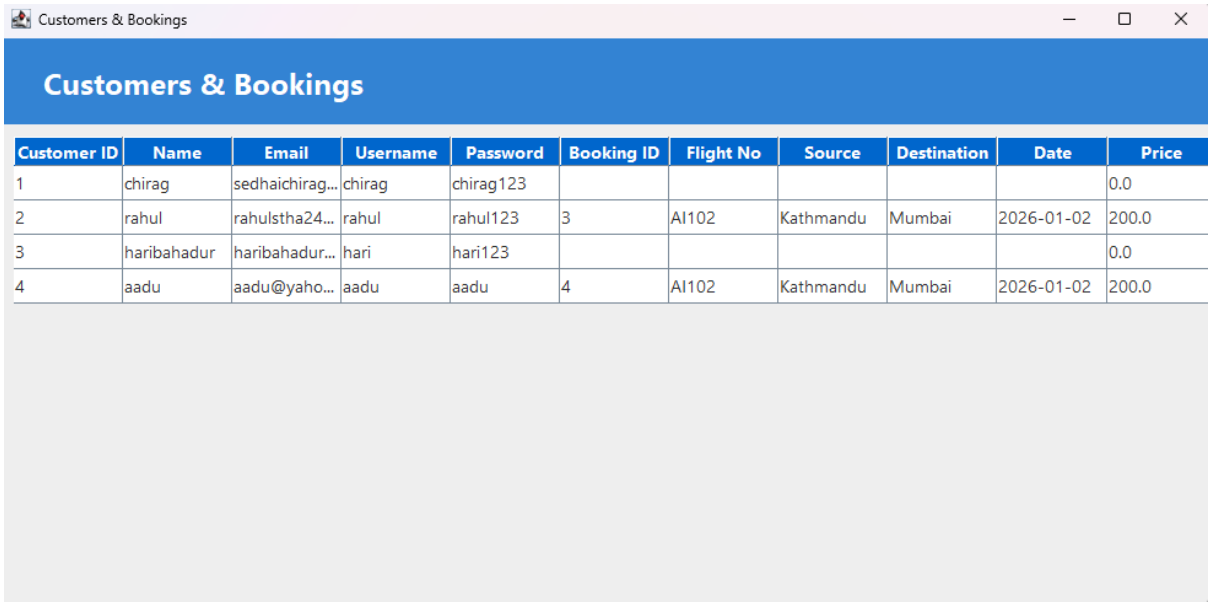

Figure 12: View Customer & Bookings

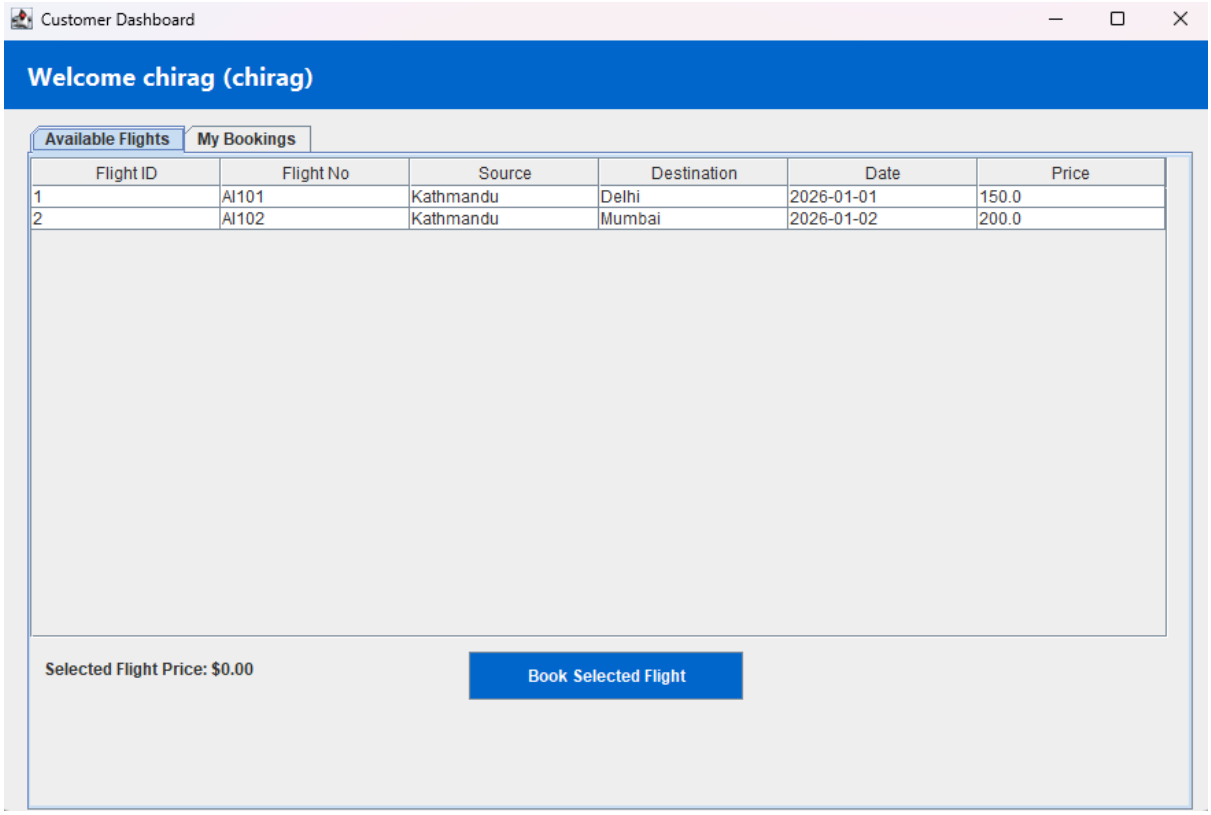

Figure 13: Customer Page