**Airline Reservation System**

**(Object-Oriented Programming in C++)**

****

**Submitted By**  
Hayat Nabi ( 09-131242-097)

Abdullah Dilawar (09-131242-005)

**Class: BSE-2A**

**Supervisor**  
Engr. Faisal Zia Sir

**Department**  
Department of Software Engineering  
Bahria University, H11 Campus, Islamabad

**Date:**  
29-May-2025

**Table of Contents**

*Declaration*………………………………………………………………………………………

*Abstract*…………………………………………………………………………………………..

*Chapter 1: Introduction*……………………………………………………………………….....

*Chapter 2: Methodology*…………………………………………………………………………

* *2.1 System Design*……………………………………………………….
* *2.2 Implementation*………………………………………………………
* *2.3 Data-File Handling*………………………………………………….

*Chapter 3: Testing*……………………………………………………………………………….

*Chapter 4: Project Output*………………………………………………………………………

*Conclusion*……………………………………………………………………………………....

*Appendix A*………………………………………………………………………………………

**Declaration**

*I certify that this project titled “Airline Reservation System Using Object-Oriented Programming in C++” is my own work. It has not been presented elsewhere for assessment. All external sources are properly acknowledged.*

**Abstract**

*This project demonstrates the development of an Airline Reservation System implemented in C++ using Object-Oriented Programming principles. The system allows users to manage flight details, passengers, and bookings with role-based access (Admin and Passenger). Core OOP concepts like encapsulation, inheritance, and polymorphism have been applied to build a modular and maintainable codebase. The system supports persistent data storage via file handling to maintain flight, user, and booking records across sessions. Input validation and error handling ensure robustness and user-friendly interactions. This project serves as a practical application of OOP concepts in building real-world reservation systems, highlighting the integration of file I/O, class design, and user management.*

**Key Words**

*Airline Reservation, Object-Oriented Programming, Flight Management, Booking System, File Handling, C++ Programming, Data Persistence.*

**CHAPTER 1: INTRODUCTION**

*The Airline Reservation System is a software application developed to manage flight information, user accounts, and ticket bookings. Airlines require such systems to maintain flight schedules, seat availability, and passenger bookings efficiently. This system provides two primary user roles: Admin and Passenger. Admins can add, view, and remove flights, while passengers can register, log in, search flights, book or cancel tickets, and view booking history.*

*The project is implemented using C++ and follows Object-Oriented Programming (OOP) principles to model entities like Users, Flights, and Bookings. OOP enables code reusability, modularity, and ease of maintenance. The system uses file operations to persist data, allowing continuity between program sessions. Exception handling and input validation mechanisms improve robustness, ensuring the system behaves correctly under various inputs.*

*This project offers hands-on experience in designing an end-to-end airline reservation solution, applying software engineering concepts such as inheritance, encapsulation, polymorphism, and file management.*

**CHAPTER 2: METHODOLOGY**

**2.1 System Design**

*The system is structured around core classes:*

* **User:** *Abstract base class for Admin and Passenger.*
* **Flight:** *Represents flight details including number, origin, destination, schedule, price, and seats.*
* **Booking:** *Manages passenger bookings, seat assignments, and cancellation status.*
* **Airline Reservation System:** *The controller class handling user interactions, data loading/saving, and business logic.*

*Relationships are established via inheritance (User → Admin, Passenger), and aggregation (Airline Reservation System contains collections of Users, Flights, Bookings). The system uses console-based menus for user interaction, supporting multiple functionalities per role.*

**2.2 Implementation**

*The program is implemented in C++ using:*

* *Classes with private data members and public member functions.*
* *Constructors and methods for data manipulation and validation.*
* *Polymorphism to manage different User types and login processes.*
* *File handling via ifstream and ofstream to persist data for Admins, Passengers, Flights, and Bookings in separate text files.*
* *Input validation helper functions (getInt, getDouble) to ensure correct user input.*
* *Colored console output to enhance user interface experience.*

*The main interaction loop allows role selection and calls relevant menus based on login status.*

**2.3 Data-File Handling**

*Data persistence is achieved through text files:*

* ***admins.txt*** *stores admin usernames and passwords.*
* ***passengers.txt*** *stores passenger credentials.*
* ***flights.txt*** *stores flight information in CSV format.*
* ***bookings.txt*** *stores booking records with cancellation flags.*

*On program startup, these files are read to load existing data into memory. Upon changes like adding flights or booking tickets, files are updated immediately to ensure data consistency. First-time setup creates an Admin account if no admin data is found.*

**CHAPTER 3: TESTING**

*The system was tested under multiple scenarios to verify functionality and robustness:*

* **Admin Login/Logout:** *Verified correct authentication and access to admin menus.*
* **Flight Management:** *Added flights, viewed flight lists, and removed flights; ensured invalid inputs were handled gracefully.*
* **Passenger Registration and Login:** *Tested duplicate username handling and login failure cases.*
* **Flight Search:** *Tested searching flights by origin, destination, and date with partial and full matches.*
* **Ticket Booking:** *Tested seat availability validation and booking confirmation.*
* **Booking Cancellation:** *Tested cancellation workflow, including duplicate cancellations and invalid booking IDs.*
* **Data Persistence:** *Verified that data saved correctly to files and loaded properly on program restart.*
* **Input Validation:** *Checked numeric inputs and ensured improper values were rejected with prompts for correction.*

*Edge cases, such as booking seats out of range or removing non-existent flights, were tested to ensure the system's stability.*

**CHAPTER 4: PROJECT OUTPUT**

*The system provides a user-friendly command-line interface with colored output highlighting various messages. Admins manage flight details through dedicated menus, while passengers can register, login, search flights, book seats, view booking history, and cancel tickets.*

*Sample outputs include:*

* *Flight listing table with formatted columns.*
* *Booking status display showing active or cancelled bookings with color indicators.*
* *Confirmation messages upon successful operations.*
* *Error messages guiding users to correct input errors.*

*The system's console interface presents clear prompts and feedback, enhancing user experience. Data is maintained persistently across sessions via file storage, demonstrated by reloading data on program restart.*

**Conclusion**

*The Airline Reservation System project successfully implements core airline booking functionalities using C++ and OOP. It demonstrates practical application of software design principles such as encapsulation, inheritance, and polymorphism, alongside effective file handling for data persistence.*

*Through this project, challenges like input validation, user authentication, and managing multiple related entities were addressed. The system is robust, modular, and extensible for future enhancements, such as graphical user interface integration or networked multi-user support.*

*Overall, this project offers a comprehensive learning experience bridging theoretical OOP concepts with real-world software development.*

**APPENDIX A**

*Include key source code listings or pseudocode snippets illustrating important classes and functions, such as:*

* *Code snippets showing file loading/saving methods.*
* *Sample input validation functions.*
* *Error Handling*