**Airport Management System – Project Report**

**Project Title: Airport Management System**

**Submitted By:**

**-** Mutahir Ahmed – 24K-0030

- Burair Hyder – 24K-0804

- Sameed Imran – 24K-1036

- Ammar Kamran – 24K-0732

**Course: OOP Lab**

**Instructor: [Sir Shafique Rehman]**

**Submission Date: [5/11/2025**]

# 1. Executive Summary

Project Overview:  
This project aimed to build a fully functional console-based Airport Management System using Object-Oriented Programming (OOP) in C++. The system enables the admin to manage flights, passengers, staff, and cargo efficiently. It also integrates feedback collection, analytics on popular destinations, and robust exception handling to manage real-time errors, particularly related to file input/output operations.

# 2. Introduction

Background:  
Airports deal with a vast number of operations including flight management, passenger registration, cargo handling, and staff coordination. Traditionally, these operations are handled by complex software systems. We decided to simulate such a system in C++ using OOP to explore real-world data management, inheritance structures, and user interaction in a console environment.  
  
Objectives of the Project:  
- Implement core airport operations using C++  
- Use OOP principles: encapsulation, inheritance, and polymorphism  
- Persist data using text files  
- Handle exceptions gracefully  
- Design an admin-only interface with authentication

# 3. System Description

Original Concept:  
The system simulates airport processes. Unlike traditional airport systems, this one is simplified and focused on learning and demonstrating OOP concepts.  
  
Innovations and Modifications:  
- Added flight comparison logic using operator overloading  
- Destination analytics (most popular route)  
- Feedback storage and display per passenger  
- File-based auto-saving and loading

# 4. OOP Approach and Methodology

OOP Concepts Used:  
- Encapsulation: Data members of classes are kept private  
- Inheritance: Passenger and Staff inherit from AirportUser  
- Polymorphism: display() method is virtual and overridden  
- Friend Classes: AirportSystem is a friend to other classes for direct access  
  
Methodology:  
- System initialized by reading from files  
- All data stored in vector containers  
- Data auto-saved upon exit via destructor  
  
Exception Handling:  
- Custom AirportExceptions class for descriptive errors  
- Catches file I/O failures like missing or corrupt files

# 5. System Mechanics

System Rules:  
- Only authenticated admins can use the system  
- Passenger and cargo are added per flight  
- Feedback is linked to passport number  
  
Turn-based Mechanics:  
- Admin is shown a menu to select actions  
- Based on input, control shifts to the relevant function  
  
End Condition:  
- System exits when the admin selects '0' from the main menu  
- Data is saved before exit

# 6. Implementation and Development

Development Process:  
- Requirements finalized  
- Data structures (classes) were defined  
- Functions implemented and tested incrementally  
- Final system connected with persistent storage  
  
Programming Language and Tools:  
- Language: C++  
- Libraries: <iostream>, <fstream>, <vector>, <string>, <exception>  
- Tools: GitHub (version control), Dev-C++ (compiler), Notepad++ (editing)  
  
Challenges Encountered:  
- Ensuring safe input handling with cin.ignore() and getline()  
- Managing file read/write consistency  
- Designing meaningful analytics for flights

# 7. Team Contributions

- Mutahir Ahmed: Designed class structures, implemented analytics and comparison logic  
- Burair Hyder: Developed passenger and cargo handling modules  
- Sameed Imran: Focused on feedback system and exception handling  
- Ammar Kamran: Worked on file I/O integration and login module

# 8. Results and Discussion

System Capabilities:  
- Successfully adds/removes passengers, cargo, staff, and flights  
- Delays flights and logs new departure time  
- Collects and views per-passenger feedback  
- Displays analytics and detailed reports  
  
Performance Metrics:  
- Compilation Time = 5.13 sec.  
- Handles corrupted file errors gracefully  
- Admin operations executed in <1 sec on average

# 9. References

- C++ Documentation: https://www.w3schools.com/cpp/  
- Stack Overflow Community  
- Lecture Notes and Class Slides  
- GeeksforGeeks: File handling and OOP in C++