Report on Airport Management System

Introduction:

The provided code implements a simple Airport Management System in C++. It allows passengers to book seats on flights and managers to view available flights and passenger details. Let's break down the structure and functionality of the system.

Classes:

Booking Exception:

A custom exception class inherited from std::exception, used for handling booking-related errors.

Payment:

An Enum class defines two payment methods: CASH and CARD.

Passenger:

Represents a passenger with attributes like name, ticket price, and payment method.

Passenger Node:

A node for creating a linked list of passengers on a flight.

Flight:

Represents a flight with attributes such as flight number, origin, destination, capacity, base ticket price, and a linked list of passengers.

Node:

A node for creating a binary search tree (BST) to manage flights.

Flight Manager:

Manages flights using a binary search tree structure. It allows adding flights, displaying available flights, booking seats, and displaying passengers for a particular flight.

Functionality:

Adding Flights: Flights can be added with details such as flight number, origin, destination, capacity, and base ticket price.

Booking Seats:

Passengers can book seats on available flights. The system checks for available seats and updates the passenger list accordingly.

Displaying Flights:

Managers can view a list of available flights showing flight number, origin, destination, available seats, and base ticket price.

Displaying Passengers:

Managers can view the list of passengers for a specific flight.

User Interface:

The system provides a simple command-line interface. Users are prompted to choose between two roles: Passenger or Manager. Depending on the role selected, users are guided through different functionalities.

Passenger: Enters flight details, passenger name, ticket price, and payment method to book a seat.

Manager:

Can choose to view available flights or display passengers on a specific flight. The manager can exit the manager options menu to return to the main menu.

Error Handling:

The system includes error handling for cases such as booking when no seats are available and when a flight is not found.

Conclusion:

The Airport Management System provides essential functionalities for managing flights and passenger bookings. It demonstrates object-oriented programming concepts like classes, inheritance, and dynamic memory management. However, there is room for improvement, such as implementing more features like seat allocation, cancellation, and data persistence. Additionally, enhancing the user interface and error messages could improve user experience and clarity. Overall, the system serves as a foundation that can be expanded and refined to meet specific requirements and scale for larger applications.

https://github.com/Dawood365/Dawood-/upload