Project –

Airline Reservation and Management System

Project Statement (Expect having this only to build the project):

The **Airline Reservation and Management System** is an advanced, console-based application developed in C++ that leverages Object-Oriented Programming (OOP) principles and Modern C++ features to simulate the comprehensive operations of an airline. Designed to cater to multiple user roles—including Administrators, Booking Agents, and Passengers—the system ensures secure and role-based access to its diverse functionalities. Administrators can manage user accounts, oversee flight schedules, and handle aircraft and crew assignments, while Booking Agents facilitate flight reservations, seat selections, and payment processing. Passengers benefit from an intuitive interface to search for flights, make bookings, select seats, and manage their personal profiles, including participation in loyalty programs.

Flight scheduling and management system, allowing for the addition, updating, and removal of flights with detailed information such as flight numbers, origins, destinations, departure and arrival times, and aircraft types. The system also manages the airline's fleet by tracking aircraft specifications, maintenance schedules, and availability, ensuring optimal operational efficiency. Crew assignment functionalities enable the allocation of pilots and flight attendants to specific flights while adhering to regulatory compliance regarding maximum flight hours.

The booking system is robust, offering search capabilities based on various criteria like date, destination, and price, along with seat selection through interactive seat maps. It supports reservation creation, modification, and cancellation, complete with simulated payment processing and refund handling. Additionally, the system maintains comprehensive passenger profiles, facilitating personalized experiences and tracking travel histories. The check-in module provides both online and airport-based check-in processes, including the generation of boarding passes and management of the boarding procedure.

Maintenance tracking is integral to the system, ensuring that all aircraft undergo scheduled maintenance and that detailed logs of maintenance activities are maintained for safety and regulatory compliance. Real-time flight status updates inform relevant users of changes such as delays or cancellations, while the reporting and analytics module generates insightful reports on flight performance, reservations, financial summaries, and user activities.

Data persistence is achieved through file-based databases like JSON or CSV.

Key Features:

1. **User Roles and Authentication:**
    - **Roles: Implement multiple user roles including Administrators, Booking Agents, and Passengers.**
    - **Authentication: Secure login system with role-based access control.**
    - **User Management: Administrators can create, update, and delete user accounts.**
2. **Flight Management:**
    - **Flight Scheduling: Add, update, and remove flights with details like flight number, origin, destination, departure and arrival times, aircraft type, and status.**
    - **Aircraft Management: Manage fleet information, including aircraft specifications, maintenance schedules, and availability.**
    - **Crew Assignment: Assign pilots and flight attendants to specific flights, ensuring compliance with regulations (e.g., maximum flight hours).**
3. **Booking System:**
    - **Search Flights: Allow passengers and booking agents to search for flights based on various criteria (e.g., date, destination, price).**
    - **Seat Selection: Implement seat maps for flights, enabling passengers to select preferred seats.**
    - **Booking Management: Create, modify, and cancel reservations. Handle booking confirmations and waitlists.**
    - **Payment Processing: Simulate payment transactions for bookings, including handling refunds for cancellations.**
4. **Passenger Management:**
    - **Passenger Profiles: Store and manage passenger information, including contact details, preferences, and travel history.**
    - **Loyalty Programs: Implement a basic loyalty system where passengers earn points for bookings and can redeem them for discounts.**
5. **Check-In System:**
    - **Online Check-In: Allow passengers to check in online, select seats, and obtain boarding passes.**

- **Airport Check-In: Simulate airport check-in counter operations for booking agents to assist passengers.**

6. **Flight Operations:**
    - **Real-Time Updates: Update flight statuses (e.g., on time, delayed, canceled) and notify relevant users.**
    - **Boarding Process: Manage boarding procedures, including scanning boarding passes and verifying passenger identities.**
7. **Maintenance Tracking:**
    - **Scheduled Maintenance: Track maintenance schedules for each aircraft, ensuring timely servicing and compliance with safety regulations.**
    - **Maintenance Logs: Maintain detailed logs of maintenance performed, parts replaced, and any issues encountered.**
8. **Reporting and Analytics:**
    - **Operational Reports: Generate reports on flight performance, reservation statistics, and financial summaries.**
    - **Maintenance Reports: Provide insights into maintenance activities and aircraft utilization.**
    - **User Activity Reports: Track user interactions and booking patterns for analysis.**
9. **Data Persistence:**
    - **Database Integration: Use file-based databases (e.g., JSON, CSV)**
10. **User Interface:**
    - **Console-Based UI: Design a user-friendly and intuitive text-based interface with clear menus and prompts.**
    - **Input Validation: Ensure all user inputs are validated to maintain data integrity and prevent errors.**

Technical Requirements:

- **OOP Principles:**
    - **Classes and Objects: Define classes such as User, Administrator, BookingAgent, Passenger, Flight, Aircraft, Reservation, Payment , and Maintenance.**

- **Inheritance and Polymorphism: Utilize inheritance for different user roles and account types. Employ polymorphism for handling various operations like booking and maintenance.**

- **Modern C++ Features:**
  - **Smart Pointers: Use std::unique_ptr and std::shared_ptr for dynamic memory management of objects like flights and reservations.**
  - **STL Containers: Employ containers like std::vector, std::map, std::unordered_map, and std::set to manage collections of entities.**
  - **Lambda Expressions: Implement custom sorting, filtering, and searching functionalities using lambda expressions with STL algorithms.**
  - **Filesystem Library: Utilize <filesystem> for managing file operations related to data persistence and backups (C++17 and above).**
  - **Multi-threading (Optional): Incorporate threads for handling concurrent operations such as multiple booking agents processing reservations simultaneously using <thread> and synchronization primitives like std::mutex.**

- **Error Handling:**
  - **Exceptions: Use try-catch blocks to manage exceptions and ensure the application handles errors gracefully without crashing.**
  - **Validation: Implement comprehensive input validation to prevent invalid data entries and operations.**

Suggested Timeline (5 Days):

Day 1: Planning and Core Setup

- **Requirements Analysis:**
  - **Define detailed requirements and functionalities.**
  - **Create class diagrams and establish relationships between classes.**
- **Project Setup:**
  - **Set up the project structure and environment.**
  - **Initialize version control (e.g., Git).**
- **Basic Class Implementations:**

- **Implement foundational classes like User, Flight, Aircraft, and Reservation with essential attributes and methods.**

Day 2: User Authentication and Role Management

- **Authentication System:**
  - **Develop secure login and registration functionalities.**
  - **Implement role-based access control to restrict functionalities based on user roles.**
- **User Management:**
  - **Enable administrators to manage user accounts.**
  - **Implement password hashing and secure storage mechanisms.**

Day 3: Flight and Aircraft Management

- **Flight Scheduling:**
  - **Create functionalities to add, update, and remove flights.**
  - **Implement flight search and filtering based on criteria.**
- **Aircraft Management:**
  - **Develop systems to manage aircraft details, maintenance schedules, and availability.**
- **Crew Assignment:**
  - **Implement mechanisms to assign and manage crew members for flights.**

Day 4: Booking and Reservation System

- **Reservation Processing:**
  - **Develop booking functionalities including seat selection and reservation creation.**
  - **Implement booking modification and cancellation features.**
- **Payment Simulation:**
  - **Simulate payment processing for reservations, handling transactions and refunds.**
- **Passenger Management:**
  - **Create and manage passenger profiles and loyalty programs.**

Day 5: Check-In, Maintenance, Reporting, and Refinement

- **Check-In System:**

- **Implement online and airport check-in processes.**
- **Generate and manage boarding passes.**

- **Maintenance Tracking:**
  - **Develop maintenance scheduling and logging functionalities.**
- **Reporting Module:**
  - **Create reporting features for operational insights, maintenance logs, and user activities.**
- **(Optional) User Interface Enhancements:**
  - **Refine the console-based UI for better user experience.**
  - **Ensure all input validations and error handling are robust.**
- **Testing and Debugging:**
  - **Conduct thorough testing of all modules.**
  - **Fix bugs and optimize code performance.**
- **Documentation:**
  - **Document code, functionalities, and user guides for the system.**

Optional Enhancements:

1. **Graphical User Interface (GUI):**
   - **Transition from a console-based interface to a GUI using libraries like Qt for a more interactive user experience.**
2. **Email Notifications:**
   - **Integrate email notifications to send booking confirmations, flight updates, and boarding pass details to passengers.**
3. **Advanced Reporting:**
   - **Implement advanced analytics and visualization tools to provide deeper insights into operations and performance.**
6. **Security Enhancements:**
   - **Implement advanced security measures such as encryption for sensitive data, secure communication protocols, and intrusion detection mechanisms.**

7. **Scalability and Performance Optimization:**
    - **Optimize the system to handle a larger number of concurrent users and transactions efficiently.**

Resources and Tools:

- **C++ Compiler:**
    - **Ensure you have a modern C++ compiler that supports C++17 or later (e.g., GCC, Clang, MSVC).**
- **IDE/Text Editor:**
    - **Visual Studio Code.**
- **Libraries:**
    - **JSON Handling: Utilize nlohmann/json for JSON serialization/deserialization.**
- **Version Control:**
    - **Use Git for version control to manage code changes and collaborate effectively.**
- **Documentation Tools:**
    - **Employ tools like Doxygen for generating code documentation.**

Thank You
Edges For Training Team