Air Ticket Reservation System

1.0 Problem statement

The Air Ticket Reservation System is basically a system that helps customers access the various operations such as air ticket search, booking and cancellation after getting themselves registered.

The following section will cover aspects related to Air Ticket Reservation System.

- Login
- Customer Registration
- Ticket Booking
- Cancel Booking

Scope of the System

The scope of the system is explained through its modules as follows

- User Registration and Login - The existing users can login to the application directly and the new users should click the **New User? Create Account** link to register themselves.

  When the user clicks on the **New User? Create Account** link, it should re-direct to the registration page, where the user needs to fill in some of the basic attributes/fields mentioned below

  - Email Address
  - Password
  - Confirm Password

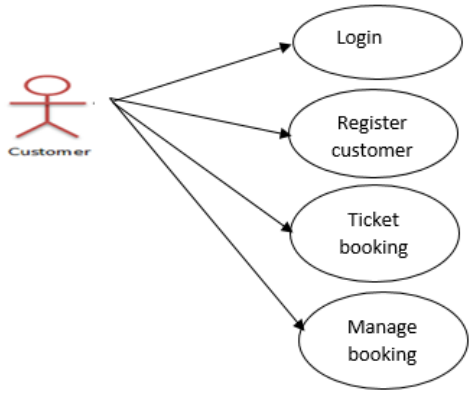
  If the email ID given by the user already exists in the database, the following message is displayed: “User is already registered. Please login to the application.”

- Ticket Booking – A user should be able to search for flights and book tickets. The flight details should be fetched using the service.

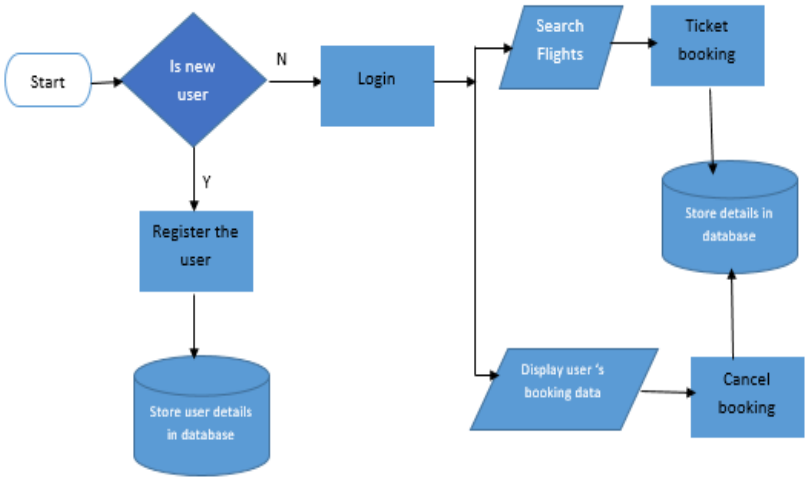
  Capture fields like Destination From, Destination To, Departure Date, Time, No of Passengers, Class

- Search and Cancel Booking – User should be able to view and cancel his/her flight bookings.
  The user should be able to logout when required.

2.0 Use Case Diagram



Flow Diagram



3.0 Project Development Guidelines

The project to be developed based on the below design considerations

Backend	• Use Rest APIs (Springboot/ASP.Net Core WebAPI to develop the services

Development	• Use Java/C# latest features • Use ORM with database • Use Swagger to invoke APIs • Implement API Versioning • Implement security to allow/disallow CRUD operations • Message input/output format should be in JSON (Read the values from the property/input files, wherever applicable). Input/output format can be designed as per the discretion of the participant. • Any error message or exception should be logged and should be user-readable (not technical) • Database connections and web service URLs should be configurable • Implement Unit Test Project for testing the API • Follow Coding Standards
Frontend Development	• Use Angular/React to develop the UI • Implement Forms, databinding, validations • Implement Routing and navigations • Use JavaScript to enhance functionalities • Implement External and Custom JavaScript files • Implement Typescript for Functions, Operators. • Any error message or exception should be logged and should be user-readable (and not technical) • Follow coding standards • Follow Standard project structure

4.0 Good to have implementation features

- Generate a SonarQube report and fix the required vulnerability
- Use the Moq framework as applicable
- Create a Docker image for the frontend and backend of the application
- Implement OAuth Security
- Implement Logging
- Implement design patterns
- Use JWT for authentication in SpringBoot/WebApi. A Token must be generated using JWT. Tokens must expire after a definite time interval, and authorization must be handled accordingly based on token expiry
- Deploy the docker image in AWS EC2 or Azure VM
- Build the application using the AWS/Azure CI/CD pipeline. Trigger a CI/CD pipeline when code is checked-in to GIT. The check-in process should trigger unit tests with mocked dependencies
- Use AWS RDS or Azure SQL DB to store the data