**NAAN MUDHALVAN PROJECT**

**MERN STACK BY MONGODB**

**Project Title: Flight Ticket Booking Application**

**Team Members:**

**VINOD A - 113321243060**

**YATHESH KUMAR P – 113321243061**

**YEDLA SASIDHAR – 113321243062**

**YOGESHWARAN S - 113321243063**

**DEPARTMENT OF**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE (FINAL YEAR)**

**VELAMMAL INSTITUTE OF TECHNOLOGY**

**PANCHETTI**

**CHENNAI – 601204**

**Index**

* **Project Overview**
* **Architecture**
* **Setup Instructions**
* **Folder Structure**
* **Running the Application**
* **API Documentation**
* **Authentication**
* **User Interface**
* **Testing**
* **Screenshots and Demo link**
* **Known Issues**
* **Future Enhancements**
* **Conclusion**

1. **Project Overview**

**Purpose:**  
The Flight Ticket Booking application streamlines the process of browsing flights, booking tickets, managing reservations, and tracking flight details for travelers. It aims to provide a user-friendly platform that simplifies booking, improves convenience, and enhances the overall travel experience.

**Features:**

* + **Flight Search and Filtering:** Search for flights by origin, destination, date, and class.
  + **Booking and Reservation Management:** Easy ticket booking and reservation history tracking.
  + **User Authentication:** Secure registration and login, with profile management.
  + **Seat Selection:** Interactive seat selection based on availability.
  + **Notifications and Alerts:** Real-time notifications for booking confirmations, flight status, and check-in reminders.
  + **Dark and Light Mode:** Accessibility-friendly theme toggle.

**2.Architecture**

#### **Frontend (React)**

* **Component-Based Structure**: Organized with reusable, modular components.
  + **Components**: Individual UI elements (e.g., FlightCard, SeatMap, Navbar).
  + **Pages**: Main views such as Home, Flight Search, Booking, User Profile, and Confirmation.
* **State Management**:
  + **Redux or Context API**: Centralized state management for user data, booking information, and flight details.
  + **Local Storage**: Stores session data temporarily to maintain user state across sessions.
* **Responsive Design**: Optimized with CSS Flexbox/Grid and media queries for mobile, tablet, and desktop screens.

#### **Backend (Node.js and Express.js)**

**RESTful API:**

* **User Routes:** Registration, login, profile, and booking history.
* **Flight Routes:** Search flights, filter by location and date, retrieve flight details.
* **Booking Routes:** Book flights, view reservations, and manage cancellations.

**Middleware:**

* **Authentication Middleware:** Secures routes, allowing only authorized users.
* **Error Handling Middleware:** Consistent error responses and debug information.

#### **Database (MongoDB)**

* **Schema Design**:
  + **User Schema**: Stores user information, including encrypted passwords, personal details, and booking history.
  + **Flight Schema**: Details flight information (flight number, departure, destination, schedule, seat map).
  + **Booking Schema**: Tracks bookings, including user ID, flight ID, seat number, and booking status.
* **Database Interactions**:
  + **Mongoose ORM**: Simplifies database interactions and schema validations.
  + **Indexes**: Optimized for fast queries by indexing fields such as flight numbers and user IDs.

**3. Setup Instructions**

**Prerequisites:**

* + **Node.js**: Server-side JavaScript runtime.
  + **MongoDB**: Database for storing user, flight, and booking data.
  + **Git**: For cloning the repository.

**Installation:**

* + **Clone Repository:** git clone[repo URL].
  + **Install Dependencies:** Run npm install in both /client and /server directories.
  + **Environment Variables:**
* Set up .env files in /client and /server.
* Configure variables such as MongoDB URI, JWT\_SECRET, and API keys for third-party services if applicable.

### 4.Folder Structure

**Client (React)**

* **/src**: Organized subfolders:
* **/components**: Reusable UI components (Navbar, FlightCard, SeatMap).
* **/pages**: Main application pages (Search, Booking, Profile).
* **/redux**: State management setup (actions, reducers).
* **/styles**: Custom CSS files for component styling.

**Server (Node.js and Express)**

* **/models**: Mongoose schemas (User.js, Flight.js, Booking.js).
* **/routes**: Organized API route handlers (userRoutes.js, flightRoutes.js, bookingRoutes.js).
* **/controllers**: Business logic (userController.js, flightController.js, bookingController.js).
* **/middleware**: Authentication and error-handling logic.
* **/config**: Configuration files, such as database connection and environment settings.

### 5.Running the Application

* **Frontend**: Run npm start in the /client directory.
* **Backend**: Run npm start in the /server directory.

### 6. API Documentation

* **Flight Search API**: GET /api/flights/search
* **Booking API**: POST /api/bookings
* **User Profile API**: GET /api/users/:userId/profile
* **Example**: Each endpoint includes the request method, necessary parameters, headers, and example JSON responses.

### 7. Authentication

* **JWT Authentication**: For secure login sessions, with tokens stored in HTTP-only cookies.
* **Authorization**: Role-based access control, ensuring secure access to user data and bookings.

### 8. User Interface

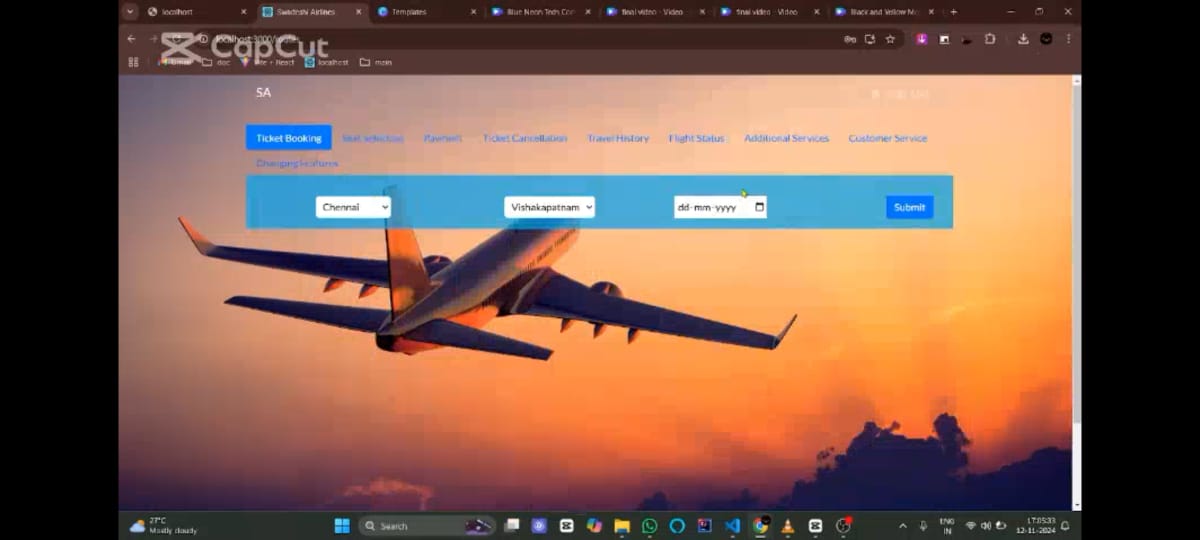
* **Screenshots and GIFs**: Show the main pages, such as the search results, booking page, and seat selection interface.

### 9. Testing

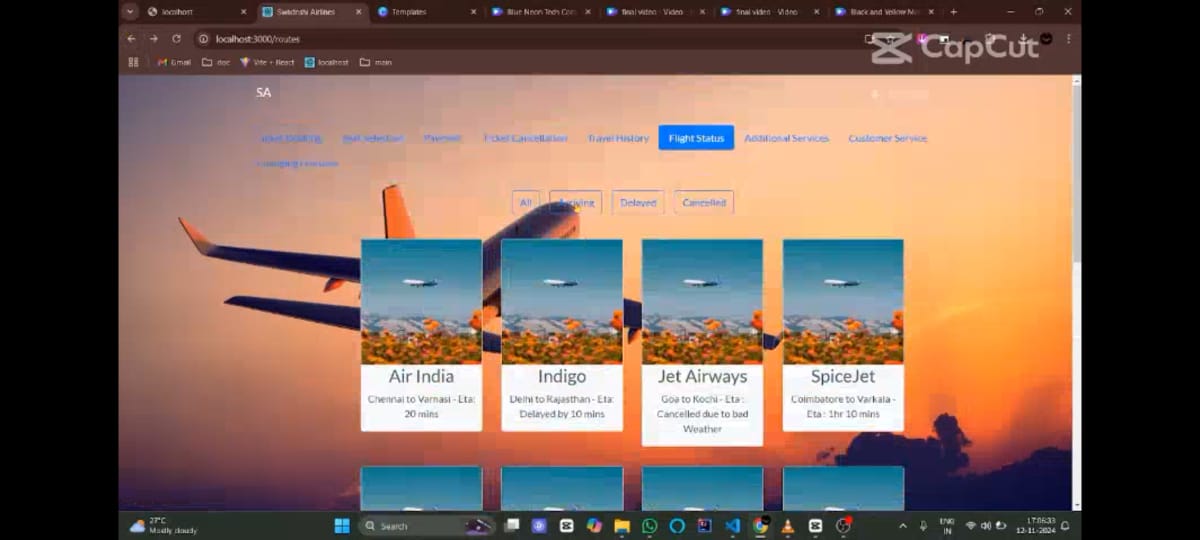
* **Frontend Testing**:
  + **React Testing Library**: Component testing to ensure proper UI rendering.
  + **Jest**: Testing Redux state management and UI functionality.
* **Backend Testing**:
  + **Mocha and Chai**: Test API endpoints and response handling.
  + **Supertest**: Testing HTTP requests to the API.

**10.Screenshots and Demo link**

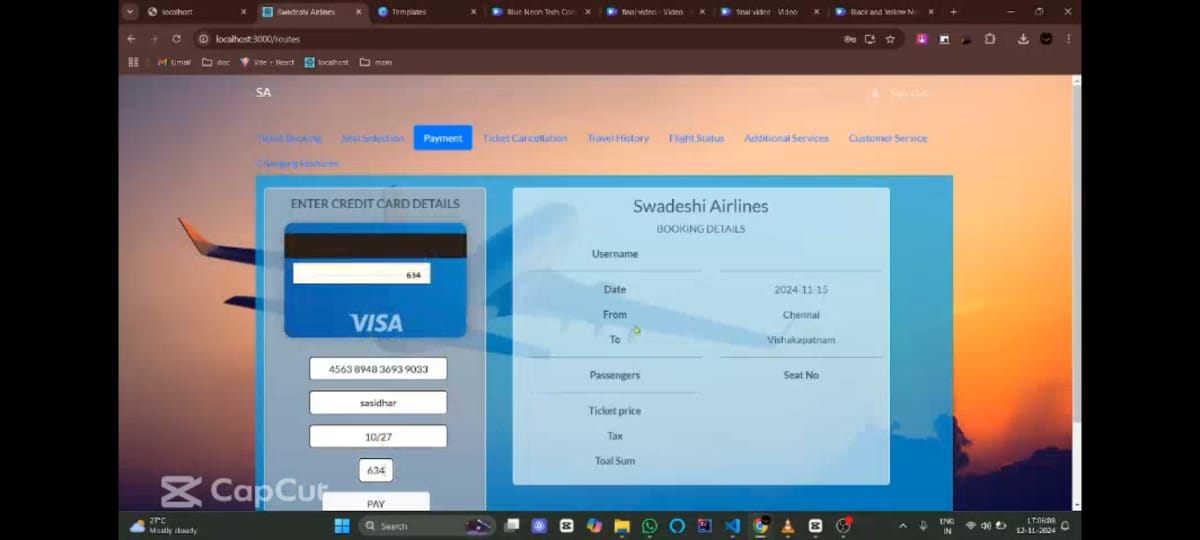
**Screenshots:**

**USER DASHBOARD **

**FLIGHT SEARCH RESULTS**

****

**BOOKING CONFIRMATION**



**DEMO LINK:**

[**Flight Ticket Booking Application.MP4**](file:///C:\Users\Dell\Downloads\demo%20video%20project.mp4)

### 11. Known Issues

* **Performance Optimizations**: Potential issues with large data loads for popular routes.
* **Responsive Design Bugs**: Minor layout issues on smaller screens.

### 12. Future Enhancements

* **Multi-city Booking**: Allow users to book complex routes with multiple destinations.
* **Frequent Flyer Program**: Reward users with points for each booking.
* **Price Comparison Feature**: Show prices from multiple airlines for better choices.
* **In-app Check-in**: Integration with airlines to enable check-in directly from the app.
* **Mobile App Version**: Dedicated app for mobile platforms to enhance the booking experience.

**13.Conclusions**

The Flight Ticket Booking application leverages the MERN stack to deliver a seamless, user-friendly platform for booking and managing flights. With a responsive React frontend, a secure Node.js and Express backend, and MongoDB for flexible data handling, the app ensures smooth searching, booking, and tracking of flights. Core features like JWT-based authentication, real-time updates, and a mobile-friendly UI enhance user convenience, while potential future features like multi-city booking and price comparison promise even greater functionality. This project effectively demonstrates how modern technology can simplify and enhance the travel booking experience.