# Full Stack Development with MERN

## Project Documentation

---

### 1. Introduction

**Project Title : Flight-finder--Navigating-your-Air-Travel-Option**

**Team Members:**

- **Akkisetty Mahima Jyothi** : Backend Development, Schemas & Controllers
- Maavuri Santhosh Kumar : Backend Development, Rotes & API Integration
- **M.Gowtham Durga Pravin** : Frontend Development, UI Design & Implementation
- **M.Durga Charan** : Frontend Development, Authentication, Cart & Order Logic

---

### 2. Project Overview

**Purpose**

The purpose of the **Flight Finder Navigation** project is to provide travelers with a user-friendly platform to search, compare, and book flights in real time. It also enables airline employees to manage flight schedules and customer bookings efficiently. By leveraging the MERN stack, the system aims to streamline air travel planning, enhance booking convenience, and support airline operations through centralized, role-based access.

**Features**

- **Real-time flight search** by source, destination, and date.
- **Filter flights** based on airline, price, and departure time.
- **User registration and secure login** using JWT authentication.
- **Booking management** with confirmation and history tracking.
- **Role-based dashboards** for customers, employees, and admins.
- **Flight schedule management** for airline staff through an intuitive interface.

---

### 3. Architecture

**Frontend**

The frontend is built using **React.js**. It includes:

- **React Router** for dynamic page routing.
- **Redux** (optional) for state management.
- Custom components for user interface design, including reusable components like the header, footer, cart, and product cards.

**Backend**

The backend uses **Node.js** and **Express.js** to handle:

- API routes for user authentication, cart management, and order processing.
- Role-based access control (user, restaurant, admin).
- JWT token authentication for secure communication.

**Database**

The database uses **MongoDB** for storing:

- **Users** (authentication data)
- **Navigation** (details, bookings)
- **Bookings** (booking status, payment tracking)

---

**4. Setup Instructions**

**Prerequisites**

- **Node.js**: Version 14.x or higher
- **MongoDB**: Installed locally or using MongoDB Atlas
- **npm**: Node package manager

**Installation**

1. **Clone the repository**:
   https://github.com/karthik19-cs/Flight-finder--Navigating-your-Air-Travel-Option
2. **Install Dependencies**:
   - For **Frontend** (React):
   - cd client
   - npm install
   - For **Backend** (Node.js, Express):
   - cd server
   - npm install
3. **Set up environment variables**:
   - Create .env files in both client and server directories:
     - **Frontend**: Set the API base URL (REACT_APP_API_URL)
     - **Backend**: Set MongoDB URI (MONGO_URI), JWT secret (JWT_SECRET)
4. **Start the Application**:
   - For **Frontend**:
   - cd client
   - npm start

- For **Backend**:
- cd server
- npm start

---

## 5. Folder Structure

**Client**

- client/: React.js frontend folder.
  - src/
    - components/: Reusable UI components (e.g., Header, Footer)
    - pages/: React components for different routes (e.g., Home, Cart, Orders)
    - redux/: Optional state management files (if Redux is used)

**Server**

- server/: Node.js backend folder.
  - controllers/: Logic to handle API requests.
  - models/: MongoDB schema models.
  - routes/: API routes (e.g., user, restaurant, order).
  - middleware/: Custom middleware (e.g., authentication, error handling).
  - config/: Environment variables and database connection.

---

## 6. Running the Application

To start the application locally, follow these commands:

- **Frontend**:
- cd client
- npm start
- **Backend**:
- cd server
- npm start

---

## 7. API Documentation

**User Endpoints**

- **POST /api /register**: Register a new user
- **POST /api /login**: Log in and receive JWT token

- **GET /api /profile**: Get user details (Protected route)

**Restaurant Endpoints**

- **POST /api/restaurants/login**: Restaurant login
- **GET /api/restaurants/:id/products**: Get products for a specific restaurant
- **POST /api/restaurants/:id/products**: Add a product

**Order Endpoints**

- **POST /api/orders**: Create a new order
- **GET /api/orders/:id**: Get order details

**Admin Endpoints**

- **POST /api/admin/promote/:id**: Promote restaurant to homepage

---

## 8. Authentication

Authentication in *Flight Navigation* is handled using **bcrypt** for secure password hashing and **React Context API** for client-side authentication state management.

- **Password Handling**:
  - User passwords are hashed using bcrypt before being stored in the database.
  - During login, the hashed password is compared with the entered password securely.
- **Session Management**:
  - Once a user logs in, their information (like user ID, role, etc.) is stored in React's **Auth Context**.
  - This context is used across the app to control access to protected routes and display appropriate UI components (e.g., user vs. restaurant header).
- **Role-Based Access**:
  - React Context tracks user roles (user, restaurant, admin) to ensure each role sees the correct dashboard and has access to relevant features.

---

## 9. User Interface

Showcasing key UI elements:

- **Homepage**: Displays restaurant categories and popular restaurants.
- **Cart Page**: Lists added items with the option to update and checkout.

---

## 10. Testing

**Testing Strategy**

- **Unit Tests**: Tests for backend API routes, models, and controllers.
- **Integration Tests**: Full-stack testing (frontend to backend communication).
- **Manual Testing**: Functional testing of user flows such as login, cart updates, and order placement.

**Tools Used**

- **Jest** for unit testing
- **Supertest** for API testing

---

---

**11. Known Issues**

- **Payment Integration**: Not implemented yet, placeholder functionality.
- **Mobile App**: Currently no mobile app version, only web-based.

---

**12. Future Enhancements**

- **Payment Gateway**: Integrate with a real payment provider (e.g., Stripe, Razorpay).
- **Mobile App**: Develop a mobile version using **React Native**.
- **Notifications**: Implement order status notifications for users and restaurants.
- **Analytics Dashboard**: Add real-time analytics for restaurants and admins.