**Full Stack Development with MERN**

**API Development and Integration Report**

|  |  |
| --- | --- |
| Date | 10/07/2024 |
| Team ID | SWTID1720194751 |
| Project Name | Project – Flight Booking App(Fly High) |
| Maximum Marks | 6 |

**Project Title:** Flight Booking App(Fly High)  
**Date:** 10/07/2024  
**Prepared by:** SWTID1720194751

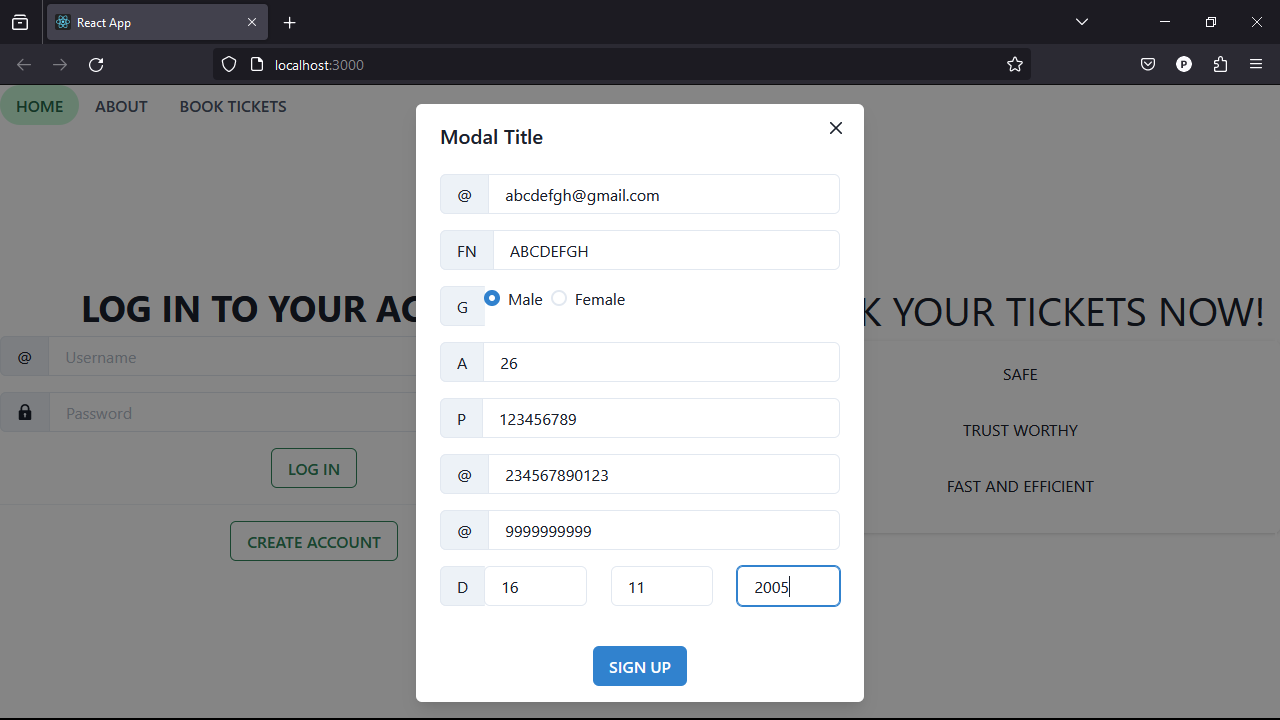
**Objective**  
The objective of this report is to document the API development progress and key aspects of the backend services implementation for the Flight Booking App project.

**Technologies Used**

* **Backend Framework:** Node.js with Express.js
* **Database:** MongoDB
* **Authentication:** None

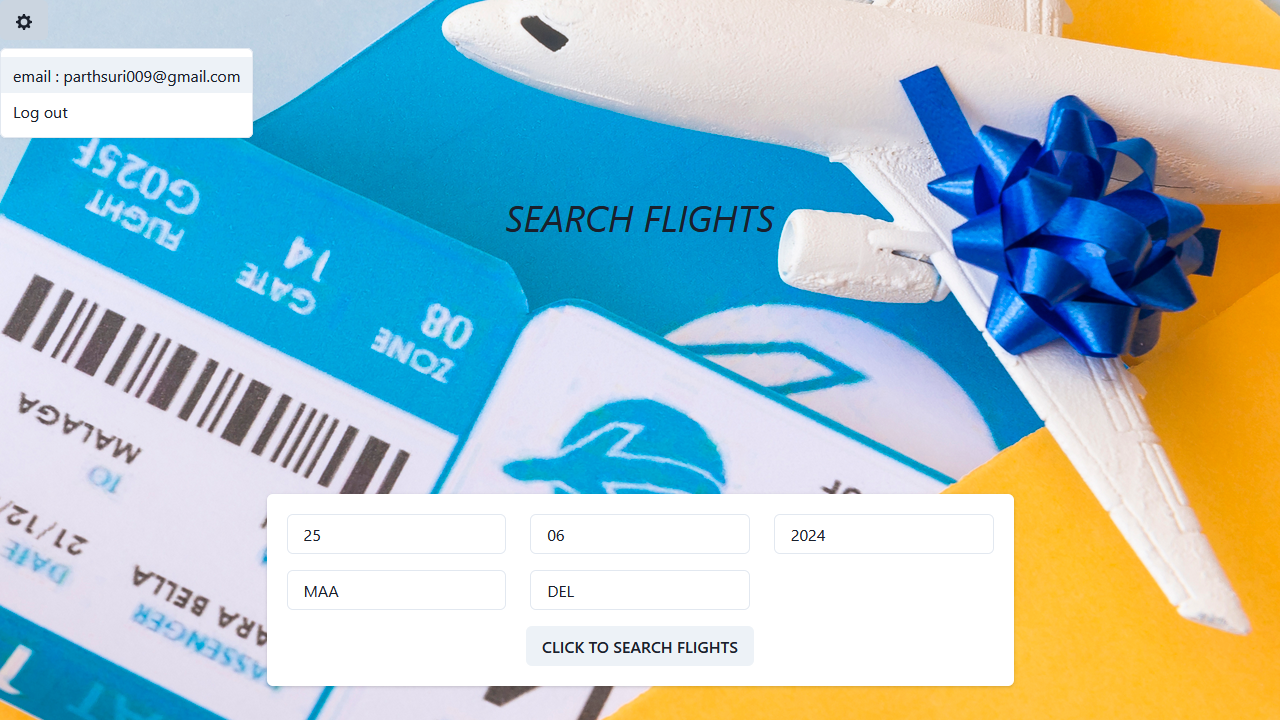
**Project Structure**



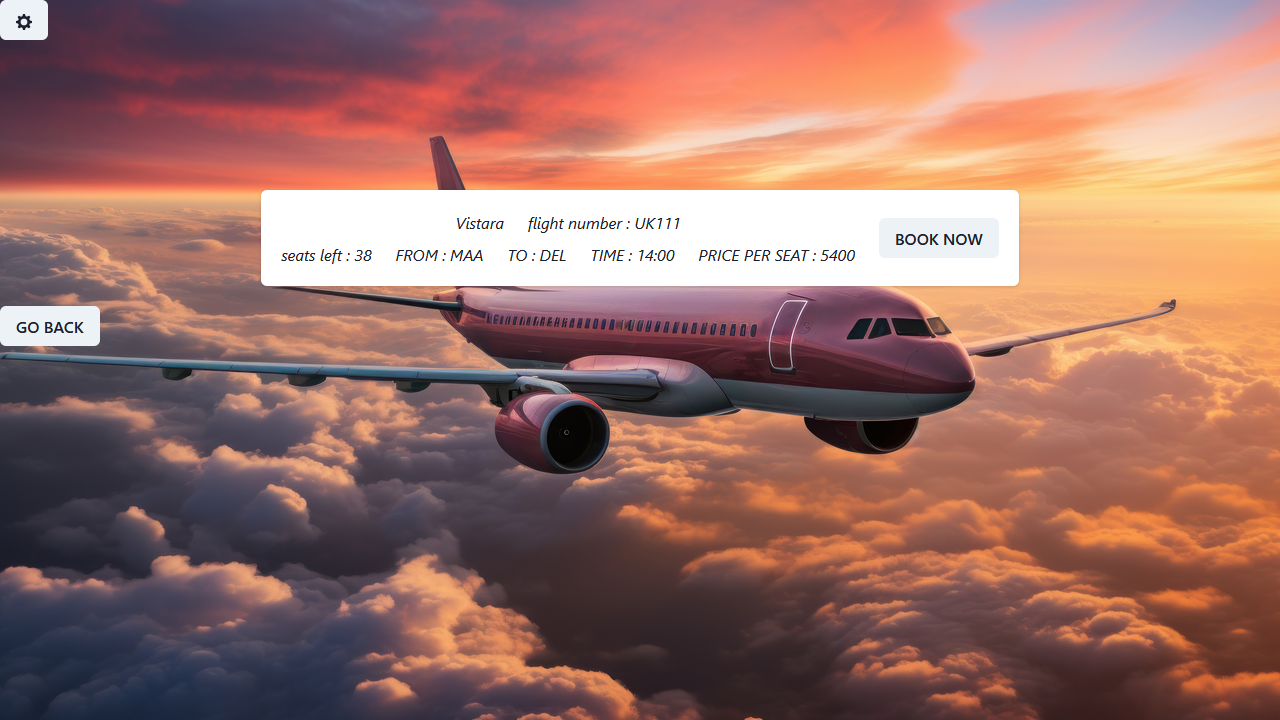
Front end part after this:-  
  
Creating an Account:-  


Search Flight page after logging in:-

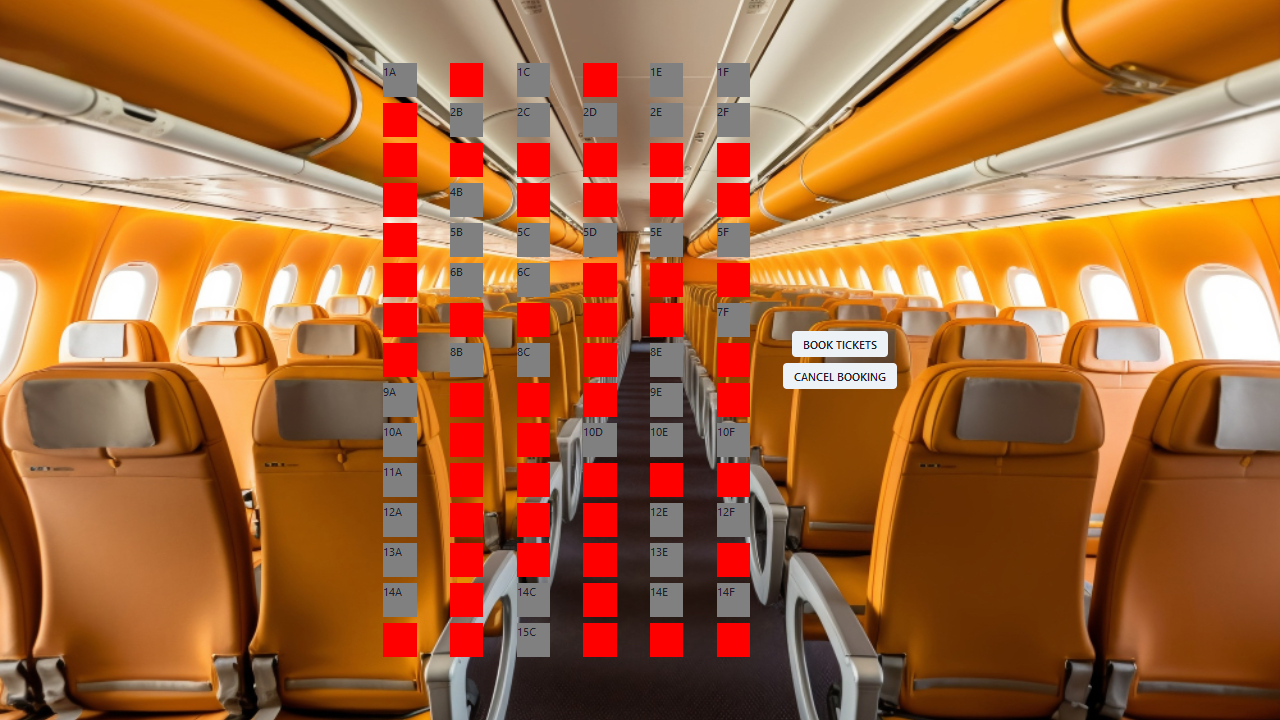


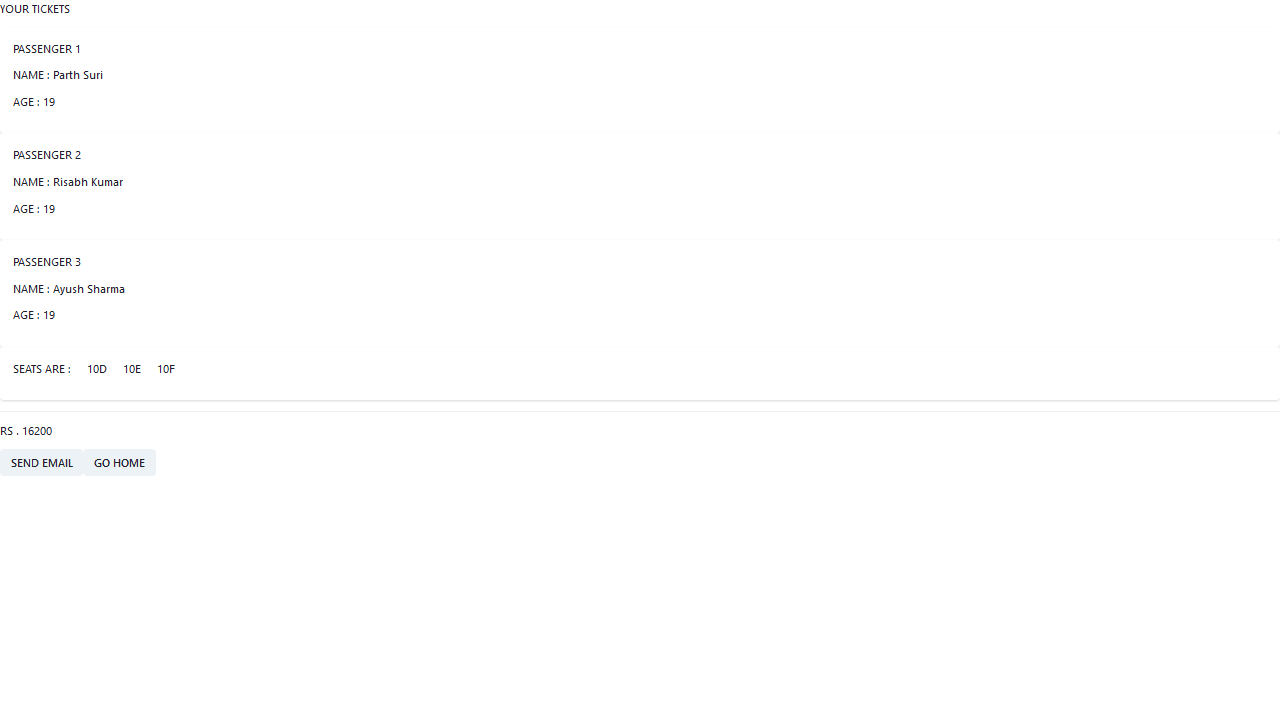
Searching for flight:-  
  


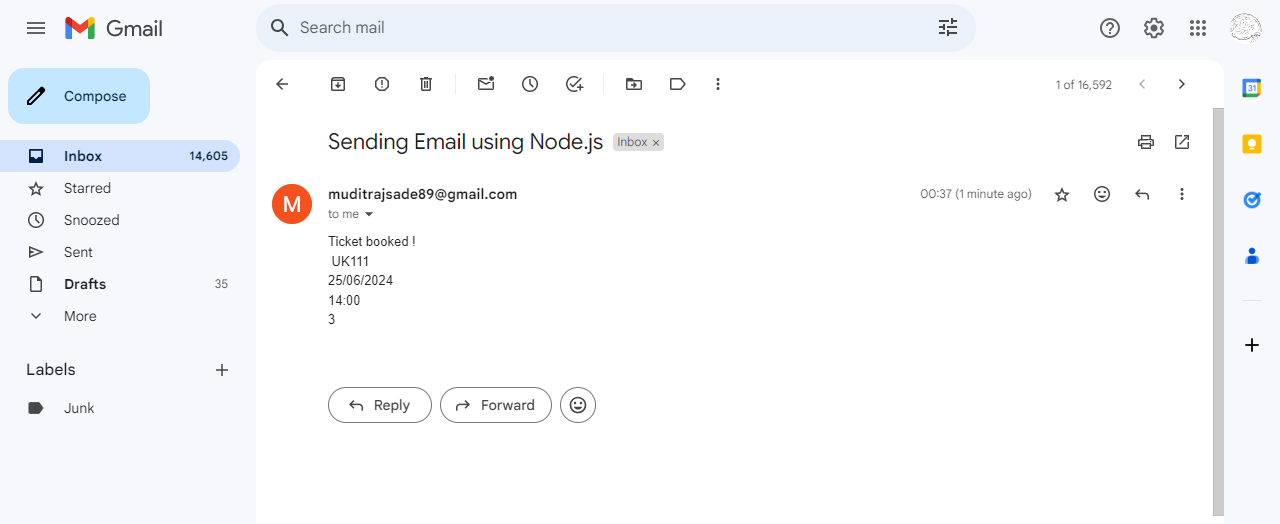
Available flights:-



Booking Page:-  
  


Seat Selection:-  
  
Red Indicates:- Not available  
  
Grey Indicates:- Can be booked  
  


Confirmation after booking:-  
  


Email Confirmation:-   
  


Backend developed by us is for the airlines who can add their flight details which can be visible to the customers after logging in.

**Key Directories and Files**

 **index.js**: Main server file containing the Express.js setup and API endpoints.

 **modules/**: Directory containing Mongoose models.

* user\_details.js: Mongoose schema for user details.
* travel\_flights.js: Presumed to contain the Mongoose schema for flight details.

**API Endpoints**  
A summary of the main API endpoints and their purposes:

 **User Management**

* POST /get\_user\_details: Adds a new user if the email doesn't exist.
* POST /log\_in\_check: Checks user login credentials.

 **Flight Management**

* POST /add\_flight: Adds a new flight with seat details.
* POST /update\_flight: Updates or deletes a flight based on seat availability.
* POST /get\_flight\_travels\_current: Retrieves flights based on specified criteria.
* POST /send\_mail: Sends an email confirmation for a booked flight.

**User Authentication**

* **POST /api/user/register** - Registers a new user.
* **POST /api/user/login** - Authenticates a user and returns a token.

**User Management**

** Users are managed via Mongoose models.**

** Endpoint /get\_user\_details handles user registration.**

** Endpoint /log\_in\_check handles user login.Workout Plans**

**Integration with Frontend**  
The backend communicates with the frontend via RESTful APIs. Key points of integration include:

**add\_flight.html**

* **Form to input flight details and generate seat layout.**
* **Uses JavaScript to handle form submission and seat selection.**
* **Submits flight details to the /add\_flight endpoint.**

**Error Handling and Validation**  
 Error **Handling Strategy**

* Error responses are sent with appropriate status codes (e.g., 100 for unsuccessful operations, 200 for success).
* Try-catch blocks are used to handle errors in async functions.

 **Validation Mechanisms**

* Basic validation is performed using conditional checks.
* For example, checking if a user already exists before saving a new user.

**Security Considerations**  
 Authentication

* Authentication is handled through email and password verification.
* Passwords are stored in plain text, which is a security concern.

 **Data Encryption**

* No explicit mention of data encryption.
* For secure systems, it’s recommended to encrypt sensitive data both at rest and in transit.

 **Other Security Measures**

* **CORS**: Configured to allow all origins.
* **Database Connection**: Secure connection string is used to connect to MongoDB.