## **Project Documentation: Flight Booking APP**

### **1. Introduction**

**Project Title:**

Flight Booking APP

**Team Members:**

* **Developer 1:** Suriya Prakash S
* **Developer 2:** Hariharan R S
* **Developer 3:** Hariharan S
* **Developer 4:** Seran E

### **2. Project Overview**

**Purpose:**

The Flight Booking APP is a digital platform designed to simplify the process of booking airline tickets. Our web app offers a seamless experience for travelers to explore, compare, and book flights based on their preferences. Whether for business or leisure, users can easily find and reserve flights that match their schedule and budget.

**Features:**

* **User Features:**
  + Search for flights based on destination, date, and preferences.
  + Book flights and manage reservations.
  + View ticket details and travel history.
  + Receive notifications and updates about booked flights.
* **Admin Features:**
  + Add, update, and manage flight schedules.
  + Monitor bookings and revenue reports.
  + Manage user accounts and permissions.

### **3. Architecture**

**Frontend:**

* **Technology:** React.js
* A dynamic, responsive interface allows users to search and book flights effortlessly.
* Pages include: Search, Booking, Payment, and User Dashboard.

**Backend:**

* **Technology:** Node.js and Express.js
* Handles flight searches, bookings, user authentication, payment integration, and other business logic.
* Implements a RESTful API for seamless frontend-backend communication.

**Database:**

* **Technology:** MongoDB
* Stores user profiles, flight details, bookings, payment records, and notifications.
* Key collections: Users, Flights, Bookings, and Transactions.

### **4. Setup Instructions**

**Prerequisites:**

* [Node.js](https://nodejs.org/): v14+
* [MongoDB](https://www.mongodb.com/): Community Edition or MongoDB Atlas
* npm or yarn as a package manager

**Installation:**

1. Clone the repository:

bash

Copy code

git clone [repository link]

1. Navigate to the project folder.
2. Install dependencies:
   1. For the backend:

bash

Copy code

cd server && npm install

* 1. For the frontend:

bash

Copy code

cd client && npm install

1. Configure environment variables:
   1. Create a .env file in the server directory with keys (e.g., database URI, JWT secret, payment gateway credentials).

### **5. Folder Structure**

**Client:**

* src/components: Contains reusable UI components.
* src/pages: Houses pages such as Search, Booking, Payment, and Dashboard.
* src/services: Manages API calls and handles communication with the backend.

**Server:**

* controllers/: Handles API request logic for flights, bookings, and user management.
* models/: Contains Mongoose schemas for MongoDB collections like Users, Flights, and Bookings.
* routes/: Defines API routes for user actions and admin management.
* middlewares/: Includes middleware for authentication and error handling.

### **6. Running the Application**

**Frontend:**

Navigate to the client directory and run:

bash

Copy code

npm start

**Backend:**

Navigate to the server directory and run:

bash

Copy code

npm start

### **7. API Documentation**

**User Routes:**

* GET /api/flights/search: Search for available flights.
* POST /api/bookings: Create a booking.
* GET /api/bookings/:id: Retrieve booking details.
* DELETE /api/bookings/:id: Cancel a booking.

**Admin Routes:**

* POST /api/admin/flights: Add a new flight.
* PUT /api/admin/flights/:id: Update flight details.
* DELETE /api/admin/flights/:id: Remove a flight.
* GET /api/admin/bookings: View all bookings.

### **8. Authentication**

**Authentication Mechanism:**

* Managed using **JSON Web Tokens (JWT)**. Users receive a JWT upon login, stored locally for session management.

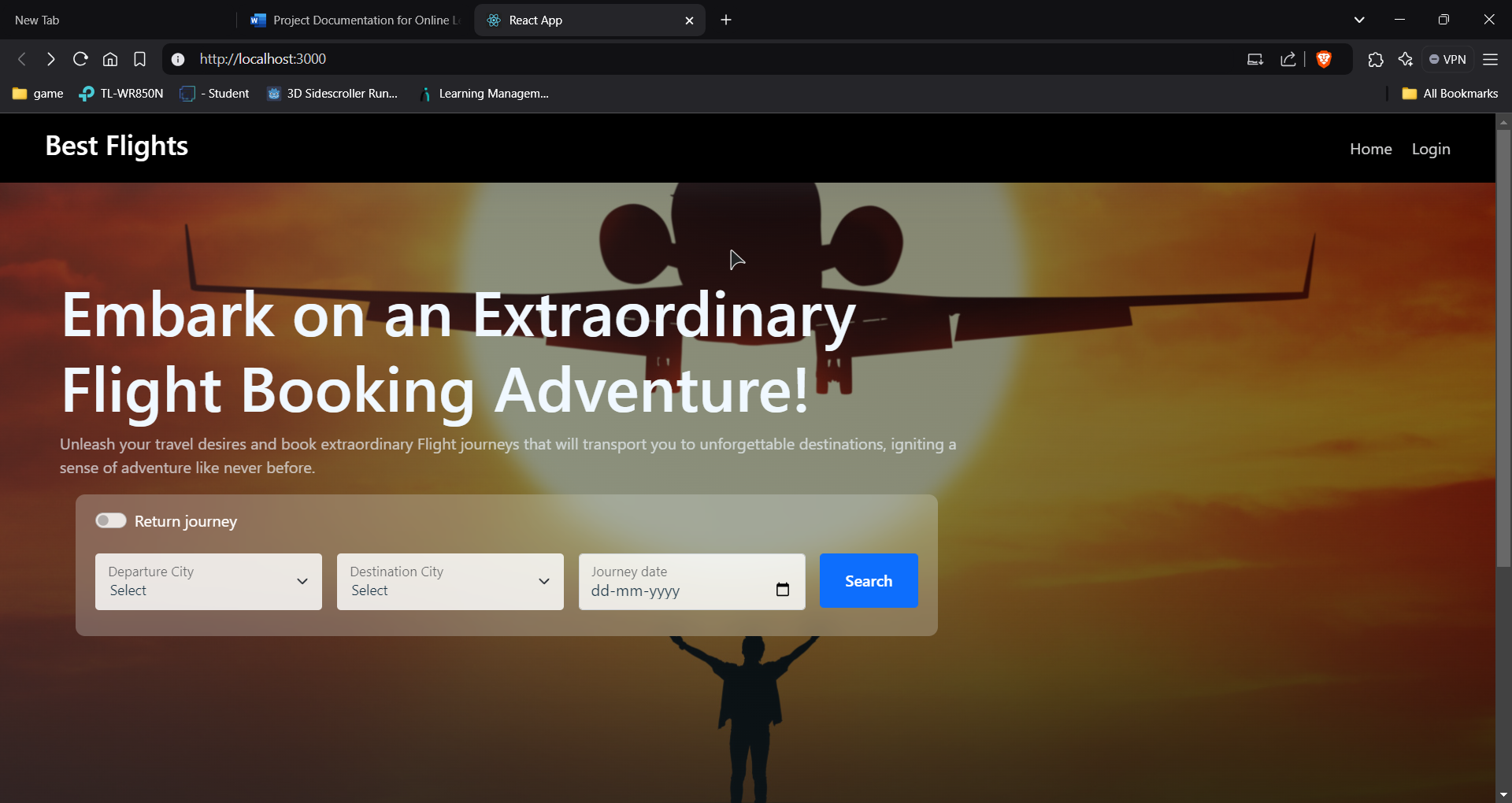
**Authorization:**

* User roles include **traveler** and **admin**, with specific permissions enforced via middleware:
  + **Traveler:** Book flights, view history.
  + **Admin:** Manage flights and view booking reports.

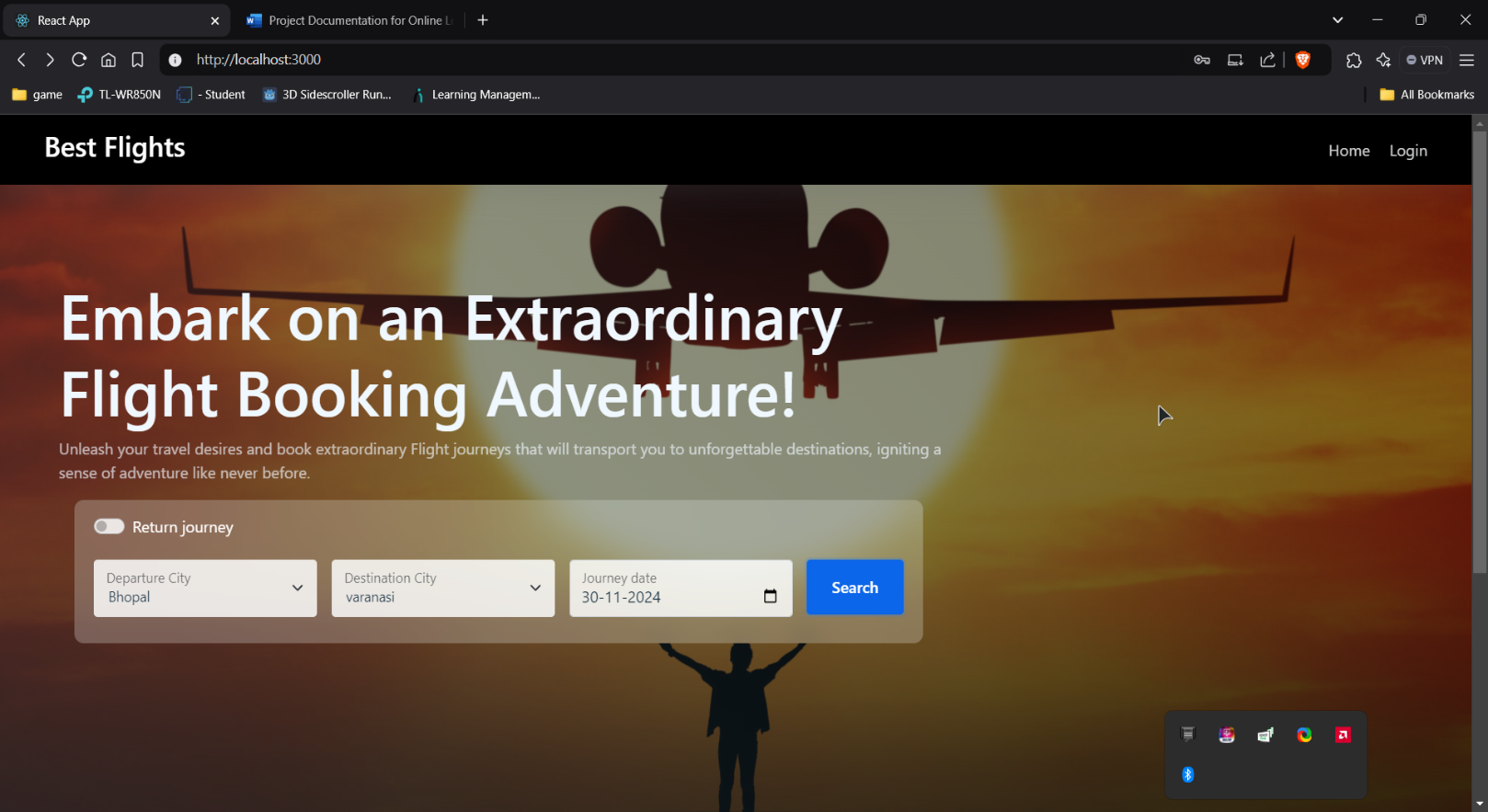
### **9. User Interface**

* **Screenshots**:

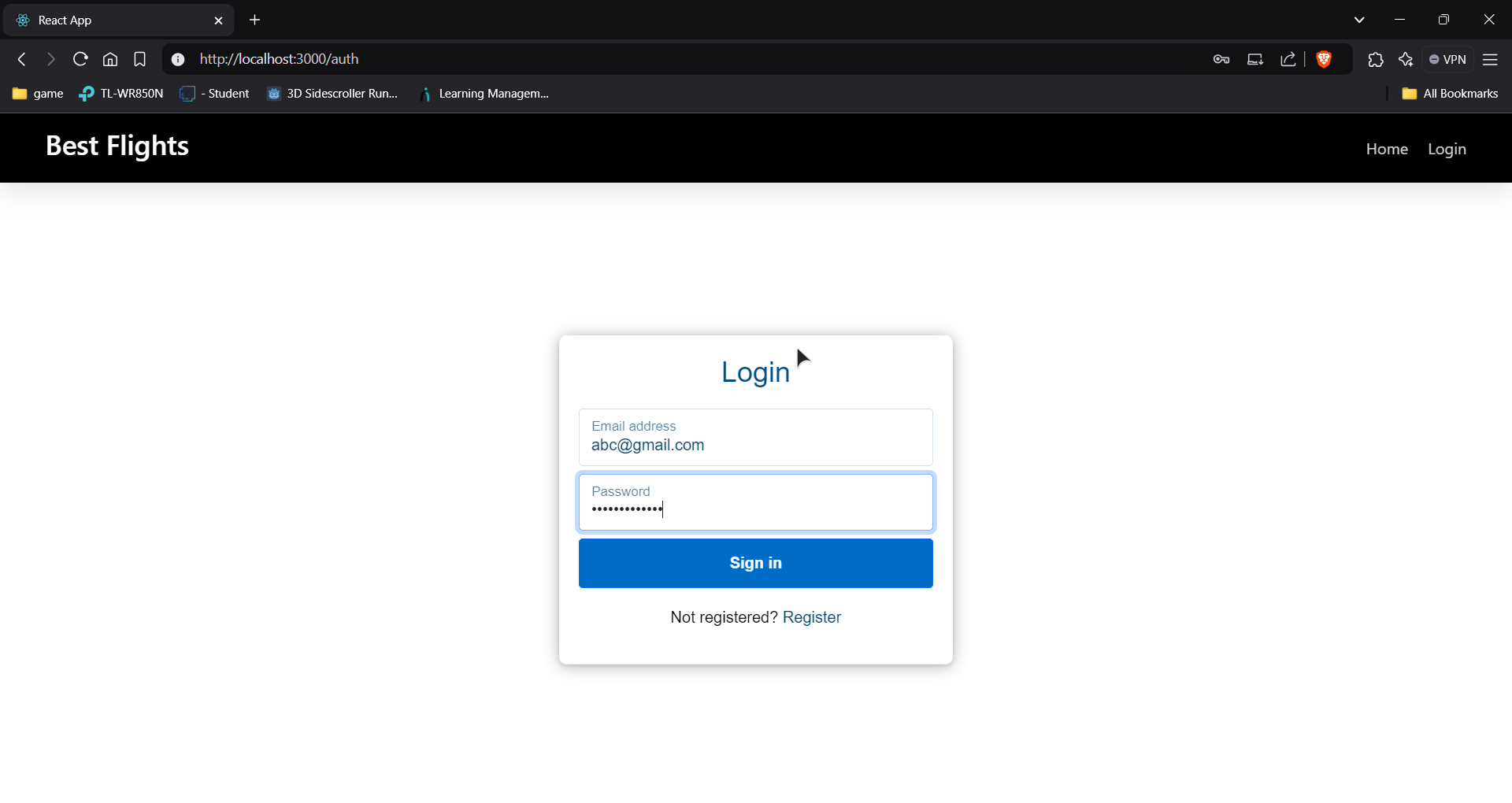
**Home Page**



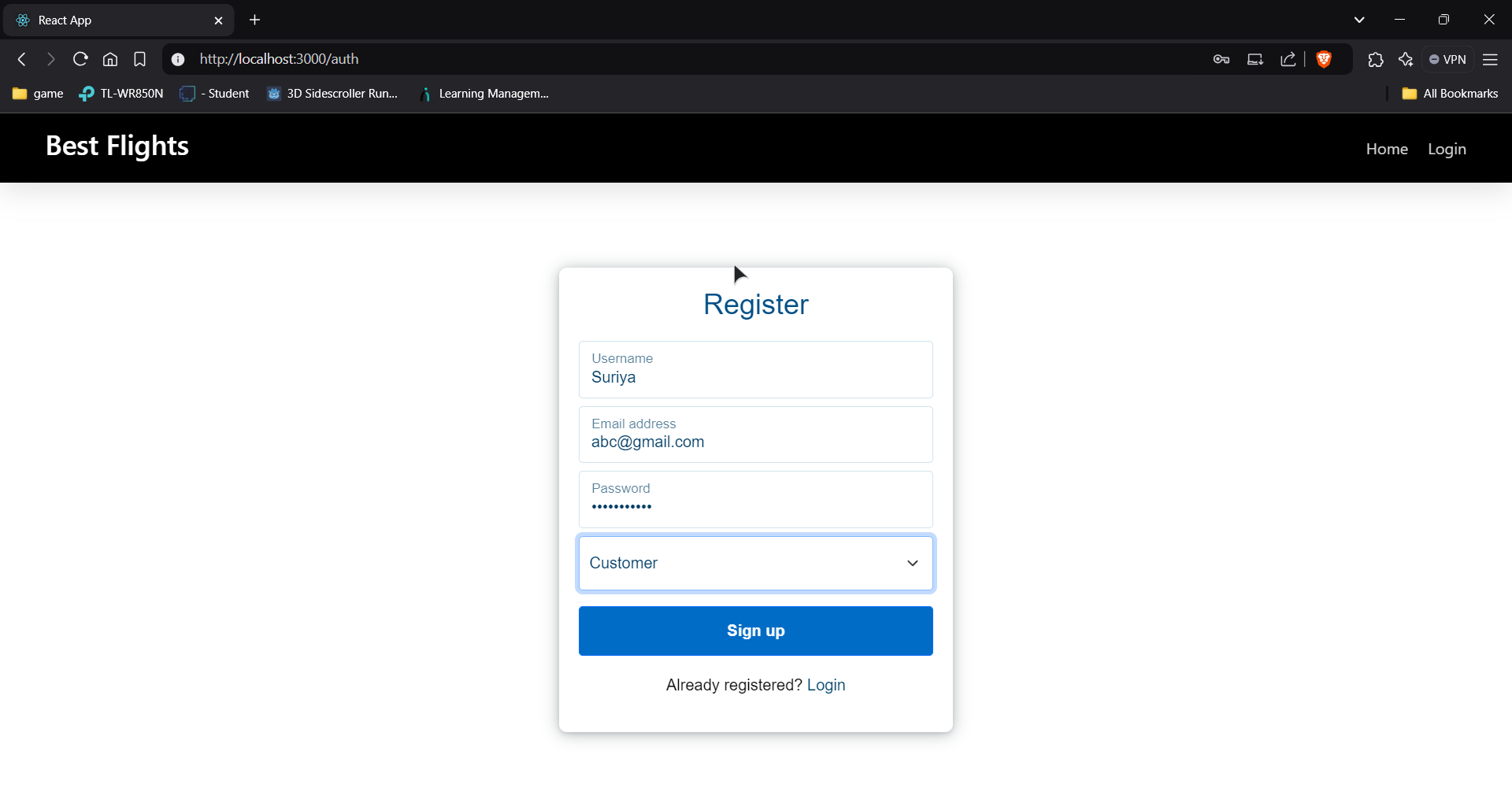
**Ticket Booking**



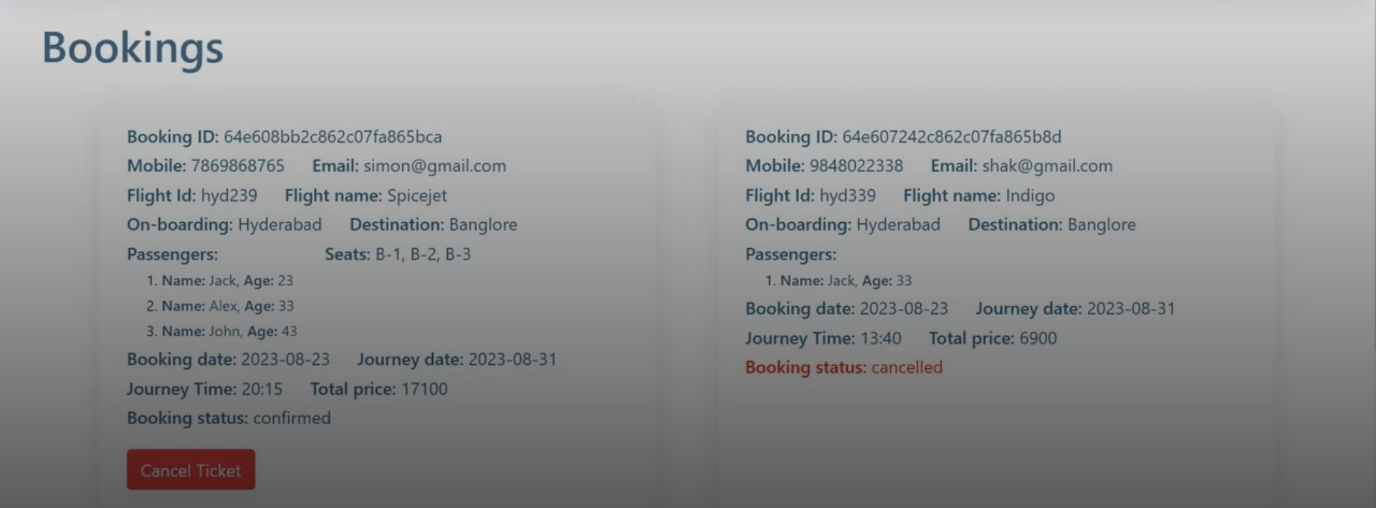
**Login Page**



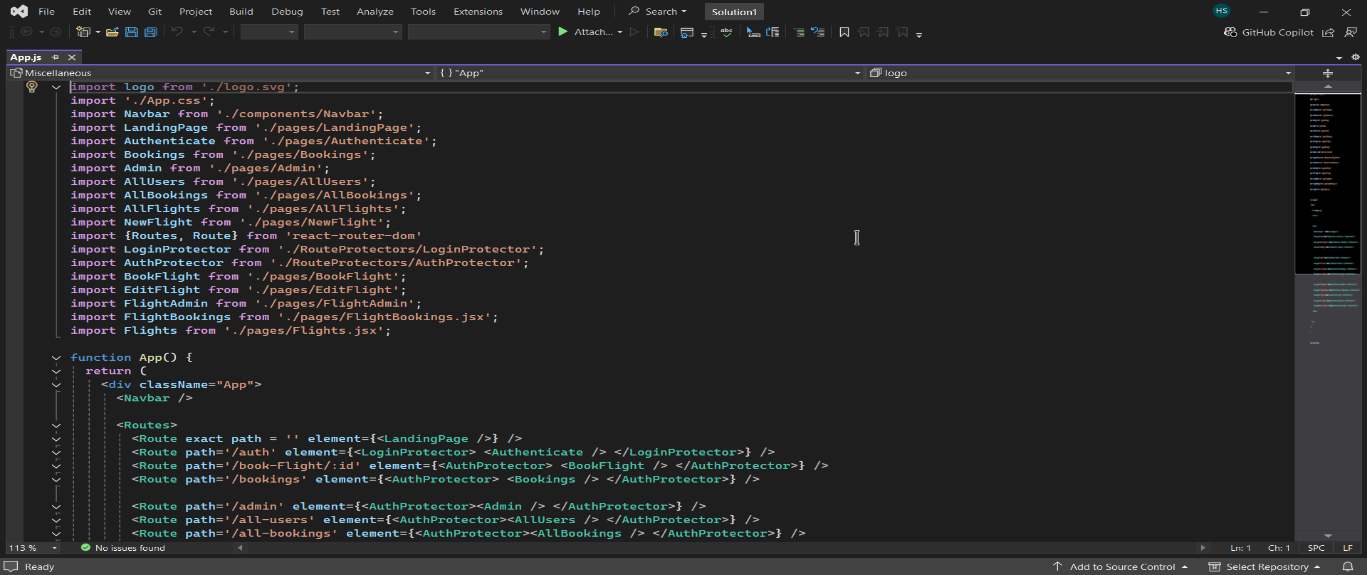
**Register Page**



**Bookings**



**Project Structure**



### **10. Testing**

**Testing Strategy:**

* **Unit Tests:**
* Validate individual functions and components using **Jest** for the backend and frontend logic.
* **Integration Tests:**

Test API endpoints using **Mocha** and **Chai** to ensure seamless backend and frontend communication.

* **UI Tests:**

Conduct end-to-end testing with **Cypress** to verify a smooth user experience across different devices and browsers.

### **11. Demo Link**

[[Demo Link](https://drive.google.com/file/d/18_HIZqmUV2PGj6c8rbfL01HLuYW4_FKY/view?usp=sharing)]

### **12. Known Issues**

**Issues:**

* Performance lags may occur when searching flights across extensive datasets.
* Mobile layout requires additional optimization for smaller screen resolutions.
* Notification system delays in updating real-time flight changes.

### **13. Future Enhancements**

**Planned Features:**

1. **Loyalty Program Integration:**

Offer reward points and benefits for frequent travelers.

1. **AI-Powered Flight Suggestions:**

Recommend flights based on user preferences, past bookings, and search history.

1. **Dynamic Pricing:**

Implement real-time pricing updates based on demand and availability.

1. **Payment Gateway Enhancements:**

Support for additional payment options like UPI, cryptocurrencies, and international payment methods.

1. **Multilingual Support:**

Enable app usage in multiple languages for a global audience.

1. **Offline Booking Mode:**

Allow users to save their search results and booking details for later submission when online