1. **Features to Implement**
    a. - User Authentication: Users should be able to register and log in using email/password. Implement authentication using JWT or Passport.js.
    b. - Route Planning and Display: Users should be able to enter their start and destination locations. Integrate Google Maps API (or OpenStreetMap, Mapbox) to display the route.
    c. - Live Traffic Updates (Optional): Display live traffic data for better route optimization.
    d. - Commute History Management: Store user commute history in MongoDB. Allow users to view past routes taken.
    e. - Fare Estimation (Optional): Estimate travel cost based on distance and transportation mode.
    f. - User Dashboard: Display commute statistics (e.g., total distance traveled, average commute time).
2. **Technical Requirements**
    a. Backend (Node.js & Express.js): RESTful API endpoints for user authentication, route planning, and history management.
    b. Frontend (Optional): Use HTML, CSS, JavaScript, and frontend frameworks (React.js/Vue.js) if implementing a UI.
    c. Google Maps API Integration: Use Google Maps API to fetch and display routes and locations.

d. Deployment: Deploy the application using Render/Heroku/Vercel for demonstration.

3. Project Setup Instruction.

a. Clone the Repository
    i. bash
    ii. git clone https://github.com/Rajatt-Chawla/daily_commute_portal.git
    iii. cd daily_commute_portal
b. Install Dependencies
c. Backend (Node.js, Python, Java, etc. – depending on your tech stack)
    i. Bas
    ii. # Example for Node.js:
        1. cd backend
        2. npm install
d. Frontend (React, Angular, Vue, etc. – as required)
    i. bash
    ii. # Example for React:
        1. cd ../frontend
        2. npm install
e. Set Environment Variables
f. Create a file like .env (or .env.local) in both frontend and backend (if required) to store any secret keys or environment-specific variables.
    i. Example:
    ii. env
        1. REACT_APP_API_URL=http://localhost:5000/api

2. DB_CONNECTION_STRING=your_database_connection_string

g. Database Setup
h. Run the Application
    i. Backend:
        1. bash
        2. npm run start
        3. or
        4. Bash
    ii. npm run dev
    iii. Frontend:
        1. bash
        2. npm start
    iv. The frontend should now be accessible in your browser at something like http://localhost:3000 (depending on your settings).
    v. Build for Production (Optional)
        1. For the frontend, you might run:
        2. bash
        3. npm run build
        4. Serve the build directory through your chosen server or cloud hosting.

4. API Endpoints and Functionalities

a. Below is a quick overview of key API endpoints. (Adjust based on your actual routes and controllers.)
b. User Authentication
    i. POST /api/auth/register
    ii. Description: Creates a new user account.
    iii. Request Body: { username, email, password }
    iv. Response: Success or error message.

v. POST /api/auth/login
vi. Description: Authenticates user.
vii. Request Body: { email, password }
viii. Response: Auth token or error.

c. Commute Scheduling
i. GET /api/commutes
ii. Description: Retrieves a list of available commutes/routes.
iii. Query Params: Optional filters (e.g., origin, destination).
iv. Response: List of commute objects.
v. POST /api/commutes
vi. Description: Creates a new commute listing (e.g., a carpool schedule).
vii. Request Body: Commute details (e.g., { origin, destination, time, seatsAvailable }).
viii. Response: Newly created commute object or error message.

d. Booking / Joining Commutes
i. POST /api/bookings
ii. Description: Book a seat in a listed commute.
iii. Request Body: { commuteId, userId }.
iv. Response: Booking confirmation or error.

5. Challenges Faced and Solutions Applied

a. Large File / Media Handling

    i. Challenge: Needed to store or serve large media files (like a background image) which exceeded GitHub's size limit.
    ii. Solution: Used Git LFS for large files or compressed the file size to stay within limits. Alternatively, used external storage (CDN / cloud storage).

b. Database Integration
    i. Challenge: Ensuring real-time or fast data retrieval for route listings.
    ii. Solution: Indexed frequently queried fields (e.g., origin, destination) in the database and implemented caching where necessary.

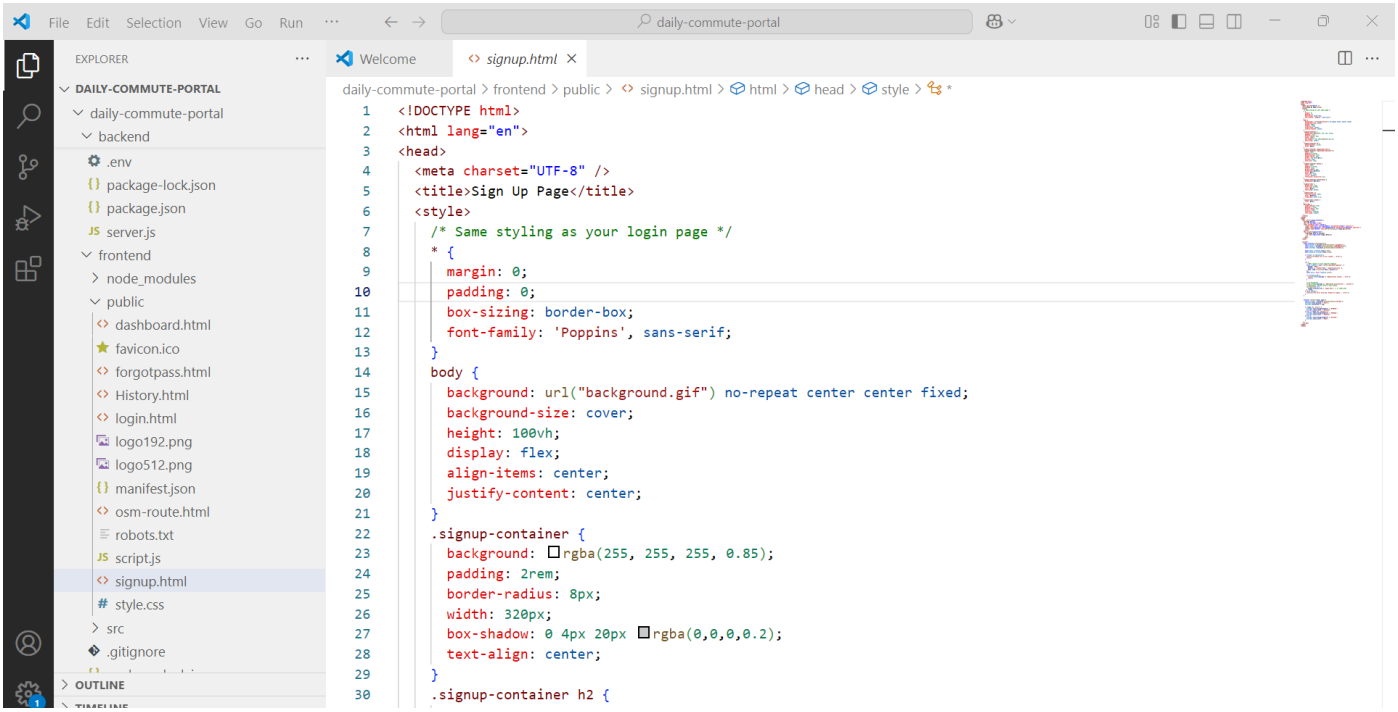
c. Authentication & Security
    i. Challenge: Protecting endpoints from unauthorized access.
    ii. Solution: Implemented JWT-based authentication (or session-based) and middleware checks for role-based access control if needed.

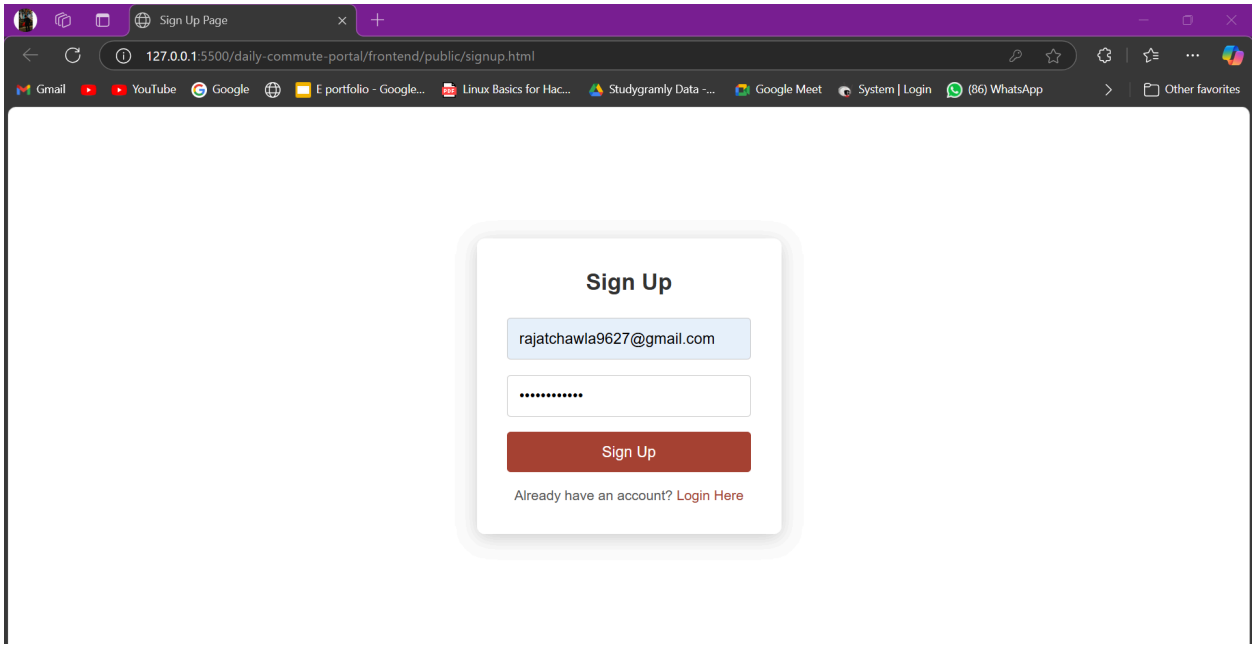
d. Handling Concurrent Bookings
    i. Challenge: Avoiding seat overbooking when multiple users simultaneously try to book.
    ii. Solution: Implemented transaction-based approach or used database-level locks/optimistic concurrency checks to ensure seat count updates were atomic.

e. Performance Optimization
    i. Challenge: Large data sets or multiple concurrent users slowing down the app.

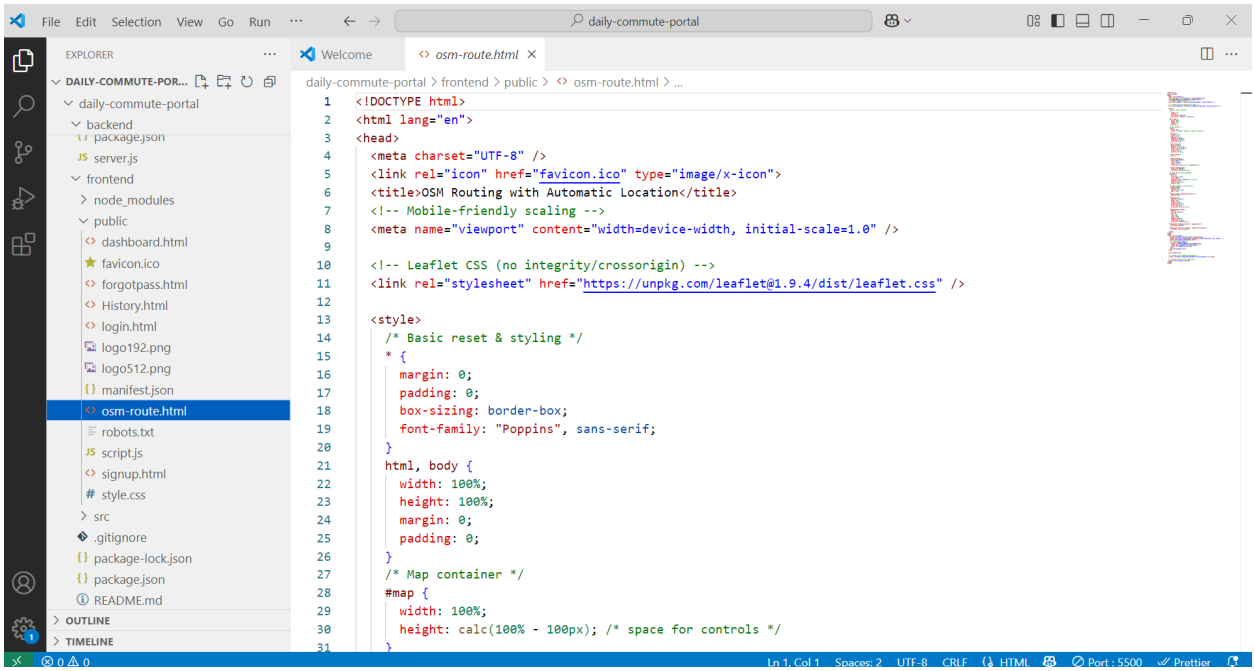
ii. Solution: Added server-side pagination, used efficient queries, and introduced caching (e.g., Redis) for frequently accessed data.

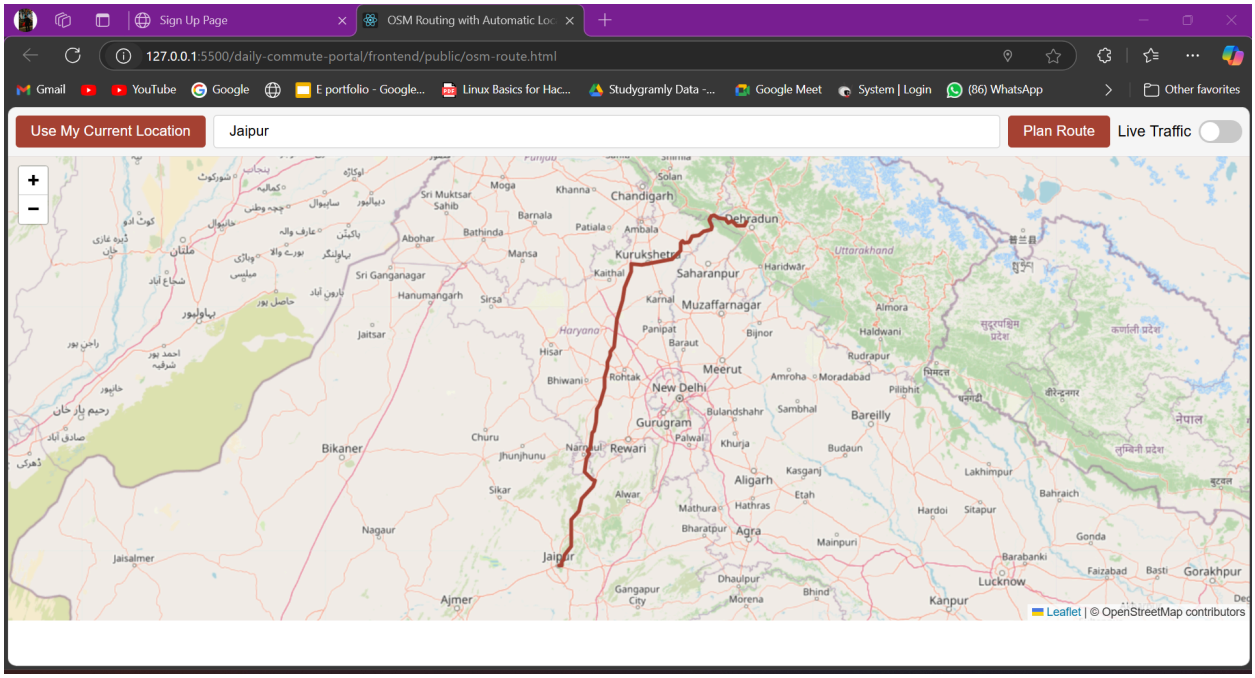
6. Sign Up page

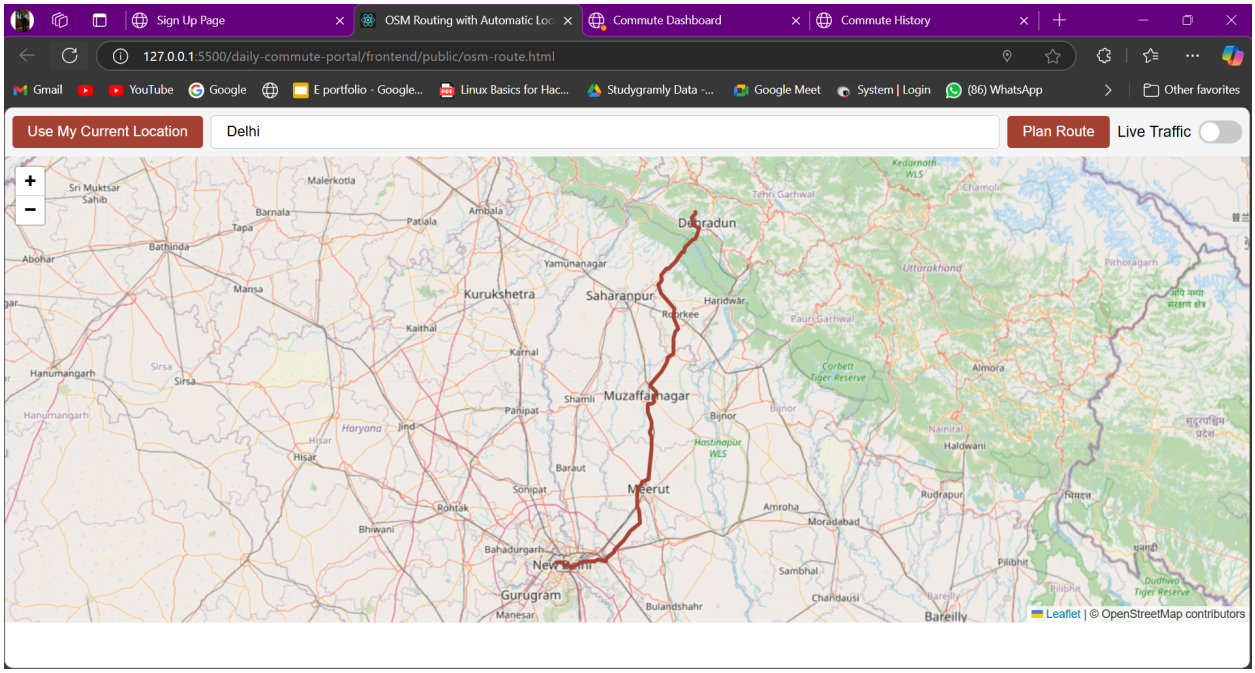




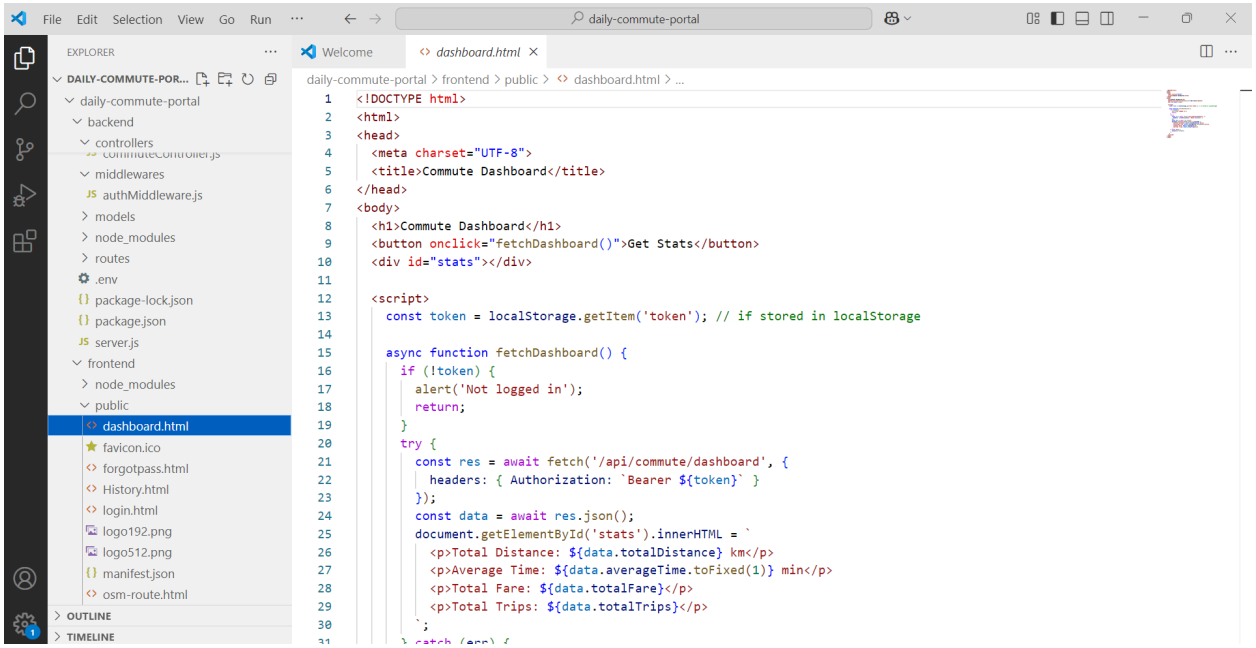
7. Route Planning



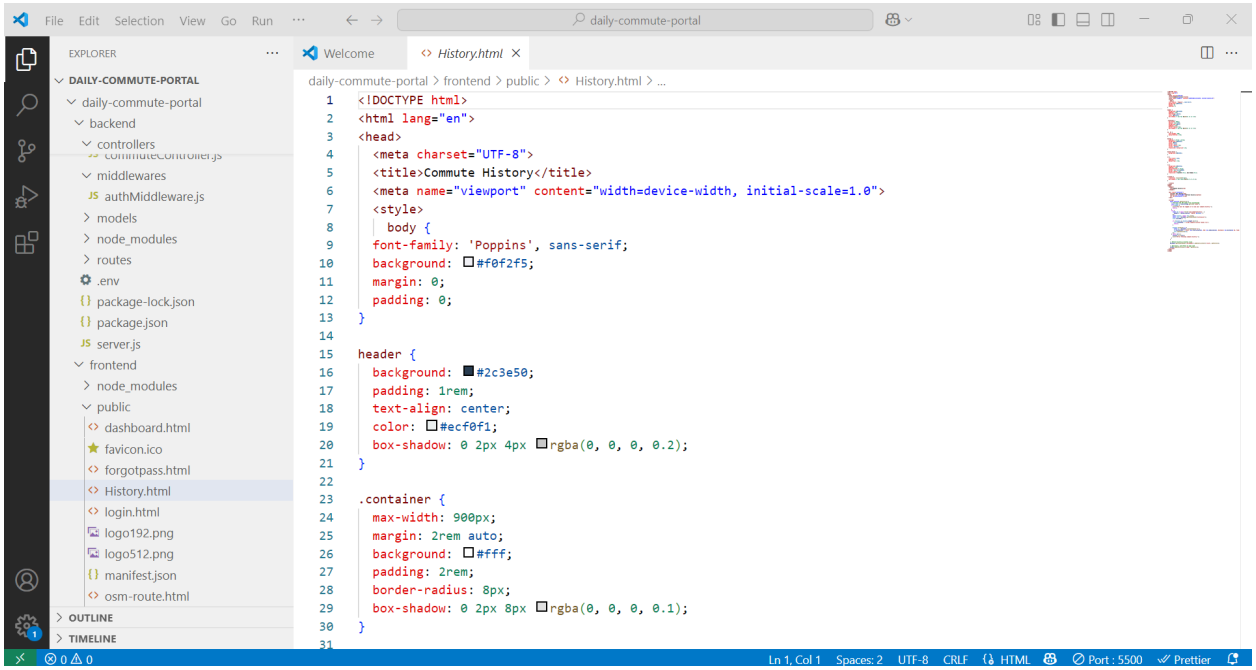

Sign Up Page
OSM Routing with Automatic Loc
127.0.0.1:5500/daily-commute-portal/frontend/public/osm-route.html
Gmail
YouTube
Google
E portfolio - Google...
Linux Basics for Hac...
Studygramly Data -...
Google Meet
System | Login
(86) WhatsApp
Other favorites
Use My Current Location
Jaipur
Plan Route
Live Traffic
Leaflet | © OpenStreetMap contributors


Sign Up Page
OSM Routing with Automatic Loc
Commute Dashboard
Commute History
127.0.0.1:5500/daily-commute-portal/frontend/public/osm-route.html
Gmail
YouTube
Google
E portfolio - Google...
Linux Basics for Hac...
Studygramly Data -...
Google Meet
System | Login
(86) WhatsApp
Other favorites
Use My Current Location
Delhi
Plan Route
Live Traffic
Leaflet | © OpenStreetMap contributors

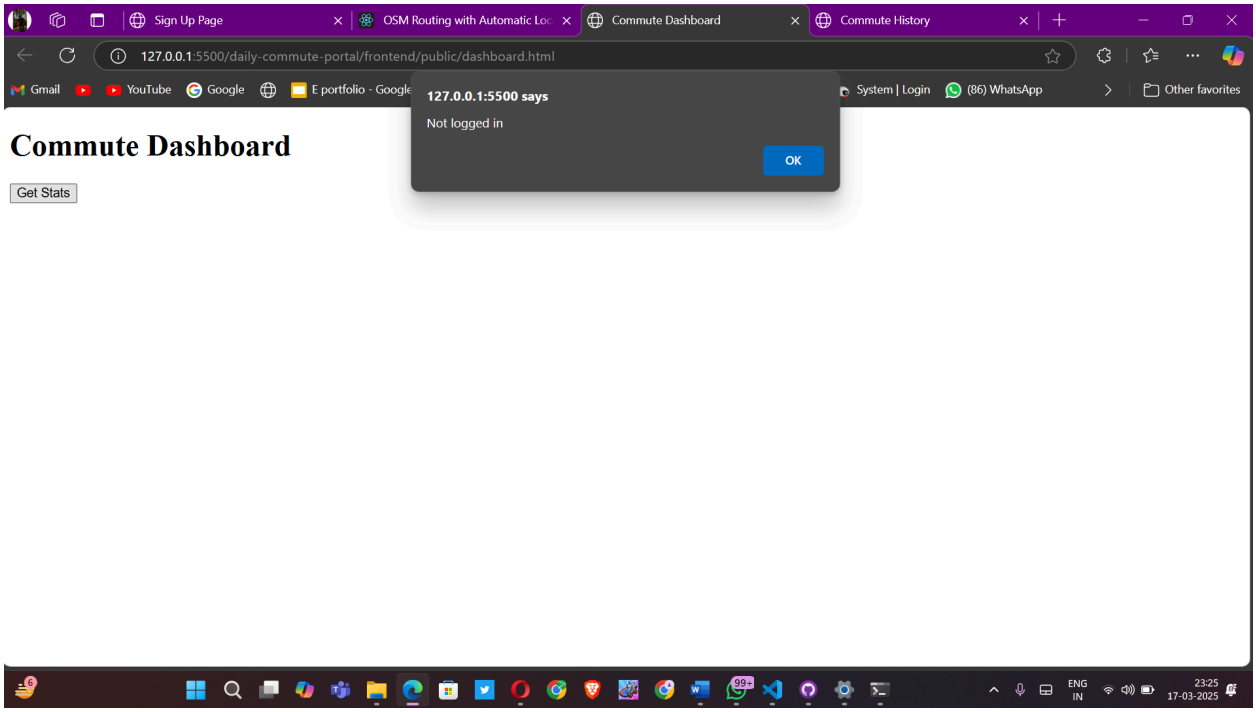
8.Commute History Dashboard

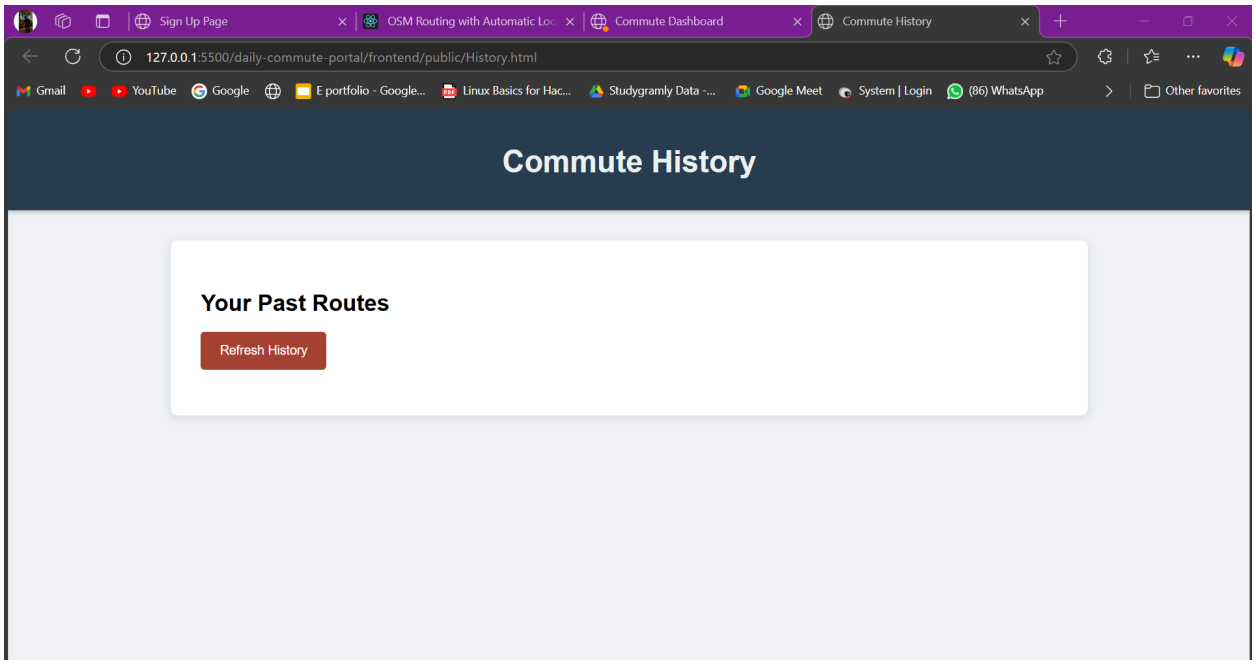


```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Commute Dashboard</title>
</head>
<body>
  <h1>Commute Dashboard</h1>
  <button onclick="fetchDashboard()">Get Stats</button>
  <div id="stats"></div>

  <script>
    const token = localStorage.getItem('token'); // if stored in localStorage

    async function fetchDashboard() {
      if (!token) {
        alert('Not logged in');
        return;
      }
      try {
        const res = await fetch('/api/commute/dashboard', {
          headers: { Authorization: `Bearer ${token}` }
        });
        const data = await res.json();
        document.getElementById('stats').innerHTML = `
          <p>Total Distance: ${data.totalDistance} km</p>
          <p>Average Time: ${data.averageTime.toFixed(1)} min</p>
          <p>Total Fare: ${data.totalFare}</p>
          <p>Total Trips: ${data.totalTrips}</p>
        `;
```



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Commute History</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    body {
  font-family: 'Poppins', sans-serif;
  background: #f0f2f5;
  margin: 0;
  padding: 0;
}

header {
  background: #2c3e50;
  padding: 1rem;
  text-align: center;
  color: #ecf0f1;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
}

.container {
  max-width: 900px;
  margin: 2rem auto;
  background: #fff;
  padding: 2rem;
  border-radius: 8px;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
}
```


Sign Up Page
OSM Routing with Automatic Loc
Commute Dashboard
Commute History
127.0.0.1:5500/daily-commute-portal/frontend/public/dashboard.html
Gmail
YouTube
Google
E portfolio - Google
System | Login
(86) WhatsApp
Other favorites
127.0.0.1:5500 says
Not logged in
OK
Commute Dashboard
Get Stats
ENG
IN
23:25
17-03-2025


Sign Up Page
OSM Routing with Automatic Loc
Commute Dashboard
Commute History
127.0.0.1:5500/daily-commute-portal/frontend/public/History.html
Gmail
YouTube
Google
E portfolio - Google...
Linux Basics for Hac...
Studygramly Data -...
Google Meet
System | Login
(86) WhatsApp
Other favorites
Commute History
Your Past Routes
Refresh History

9. User Dashboard