RouteLLM: Building Multi-Point Navigation Mobile Application with Large Language Models

CAPSTONE PROJECT

Project Report

Sai Shreesh Josyula
235813880
2024/08/10

Table of Contents

Abstract

Route LLM app is a pioneering solution that combines the most advanced AI with the power of geospatial data for a navigation experience that's safe and healthy. The app attempts to empower users with real-time route recommendations driven by environment and safety criteria, particularly pollution levels and crime rates. Further, with the integration of data from various third-party APIs and AI models in use, Route LLM is able to dynamically reassess multi-route options available for use by a user in a manner that puts users' well-being first.

The application architecture is based on React Native for a native cross-platform mobile experience, along with a strong backend server for data processing and API integration. Firebase will be used for user authentication and real-time database management to ensure secure yet efficient handling of the users' data and preferences. The app's UI has been designed with the aim to provide ease of use and accessibility to all by intuitive navigation and real-time updates.

Probably one of the biggest features of Route LLM is its in-built chatbox, powered by ChatGPT. It uniquely helps users get personalized guidance and real-time responses related to queries. It guides route choices during a journey and solves doubts by talking back through its chatbox feature.

Some of the key features of Route LLM include real-time GPS tracking, route matching algorithms allowing both speed limits and user preferences, and a gamified leaderboard that encourages users to follow safer and more environmentally friendly routes. The pollution and crime data are downloaded from reliable databases and further treated by specially developed algorithms classifying routes by these factors on their seriousness. This is then presented to the user in a simplified and actionable format, whereby they can avoid high-risk areas.

The ultimate goal of the Route LLM is to make its users' daily commutes and travels easier by saving time and allowing paths that reduce exposure to bad environmental conditions and unsafe neighborhoods. With this added layer of granularity and tailoring, along with a chatbox powered by AI for immediate assistance, it marks itself as a highly distinctive navigation tool—one that really cares about the health and safety of its end-users.

1 Introduction to Project

1.1 Overview

In today's urban environments, navigating safely and efficiently has become increasingly important, not only in terms of time and distance but also considering factors like environmental health and safety. Traditional navigation applications often prioritize optimizing routes based solely on these criteria, missing out on more comprehensive considerations such as air quality and neighborhood safety. To address this, Route LLM: A Navigation App that Classifies Routes Based on Pollution and Crime has been developed, focusing on providing users with enhanced route options that take these critical factors into account.

The backbone of the Route LLM application is, quite literally, the mobile application. It was developed in React Native to assure consistency and responsiveness of user experiences at least on Android and be easily upgradable to iOS. The application is also user-friendly, guides real-time navigation, and classifies routes while providing instantaneous feedback on the user's chosen path. The app includes a ChatGPT-powered chatbox that offers personalized support to users by answering their queries and hence helping them in making better decisions while on the move.

To support these features, the app connects to a robust backend server that I have configured and hosted. This server processes data related to pollution and crime, originally developed by another team and subsequently modified and optimized by me to better fit the application’s needs. These modifications ensure that the data is processed efficiently, providing accurate and timely information to users about the safest and healthiest routes available.

Route LLM is not just a navigation tool, it is a dynamic, intelligent application that adapts to the user’s needs and preferences, offering a more informed and safe travel experience. By enhancing existing data models and integrating them into a powerful mobile platform, Route LLM stands out as an advanced navigation solution that goes beyond traditional route planning.

1.2 Existing System

Many people use traditional navigation systems like Google Maps, Apple Maps, and Waze to provide information about real-time traffic and compute the best routes. However, current platforms are oriented to efficiency—for instance, reducing travel time and length without considering environmental and safety issues like pollution levels and crime rates.

While there are separate apps for air quality and crime data, users typically need to switch between multiple platforms to gather this information, making the process less seamless. Moreover, safety features in these apps are often reactive, providing alerts only after a risk is encountered, rather than guiding users away from potential hazards proactively.

Route LLM addresses these limitations by integrating real-time pollution and crime data directly into the navigation process. This provides users with not just the quickest routes, but also the safest and healthiest options. By enhancing and hosting existing data processing methods on a dedicated server, Route LLM offers a more comprehensive and user-friendly navigation experience within a single app.

1.3 Objectives of Project

The overriding goal of Route LLM is to offer users a navigation solution by which safety and health concerns could be combined into everyday travel. In particular, it considers the integration of real-time pollution and crime data. There is no route optimization by this tool but increased awareness of the levels of pollutants along their chosen path by color-coded indicators

Specific objectives of the project include:

- **Enhance User Safety:** Enhanced user safety by providing real-time crime levels along a user's route to reduce exposure to possible risk
- **Promote Healthier Travel:** Healthier traveling by taking advantage of real-time pollution data and representing it through color indicators for the user to gain an overview of air quality levels on their route.
- **Seamless User Experience:** Seamlessly developed user experience within a user-friendly mobile app in React Native, ensuring the smooth functioning of a mobile app on iOS and Android.
- **Personalized Assistance:** Developed a chatbox with ChatGPT that will provide real-time and personalized support to the user, display weather detail, and give customized replies to the user.

- **Efficient Data Processing:** Modify the existing methods of processing pollution and crime data and then host it on a dedicated server for the proper and timely delivery of information to the user.
- **Comprehensive Navigation:** Integrate conventional navigation with advanced safety and health indicators for much more aware traveling.

By achieving these objectives, Route LLM will be able to provide users with essential information about their surroundings in order to make safer, healthier traveling decisions without having to change their route.

2 System Architecture and Components

2.1 Mobile App

The Route LLM mobile application is the core system component, designed to provide a seamless and responsive user experience in navigation. The application, written in React Native, would support cross-platform and thus work on Android. At an architectural level, it is modular, hence providing an efficient way of managing and integrating a various functionalities such as real-time GPS tracking, pollution and crime data visualization, and user interaction through the ChatGPT-powered chatbox.

Key features of the mobile app include:

- **Real-Time Navigation:** The app provides live GPS tracking that helps in accurate and up-to-date navigation. The current location of a user is always tracked to provide real-time guidance and feedback.
- **Pollution and Crime Indicators:** Instead of offering alternative routes, the app visualizes pollution levels along the selected route using color-coded indicators. These indicators provide users with immediate insight into the air quality, allowing them to assess the healthiness of their journey at a glance. Crime data is similarly integrated, with areas of higher risk flagged on the map.
- **User Interface (UI):** The UI of the app is designed to be user-friendly and intuitive, focusing on easy navigation and rapid access to crucial information. Another key component of the user experience is the map interface, which includes overlays showing real-time data for indicators such as pollution and crimes. Other facets of the app's theme are that it will dynamically change based on the system's theme settings Light or Dark mode to give the best and most consistent user experience.

- **ChatGPT-Powered Chatbox:** In-app chatbox, powered by ChatGPT, that would provide personalized assistance. This feature shall assist users in raising questions and getting real-time answers on everything from explanations over pollution and crime indicators to general navigation help
- **Backend Communication and API Service:** The app communicates with a backend server that processes pollution and crime data. This server hosts a custom-built engine, which was developed by modifying the original Python code. Hosted on Render, this engine functions as an API service that provides the app with a JSON file. This data is fetched and updated in real-time, ensuring that users receive the most current information. The backend also handles user authentication via Firebase, securing user data and preferences.

The modular design of the Route LLM app allows for scalability, making it possible to introduce new features and improvements over time. The emphasis on real-time data visualization, coupled with the ChatGPT chatbox, dynamic theming, and robust backend integration, positions the app as a comprehensive tool for urban navigation, prioritizing both efficiency and user well-being.

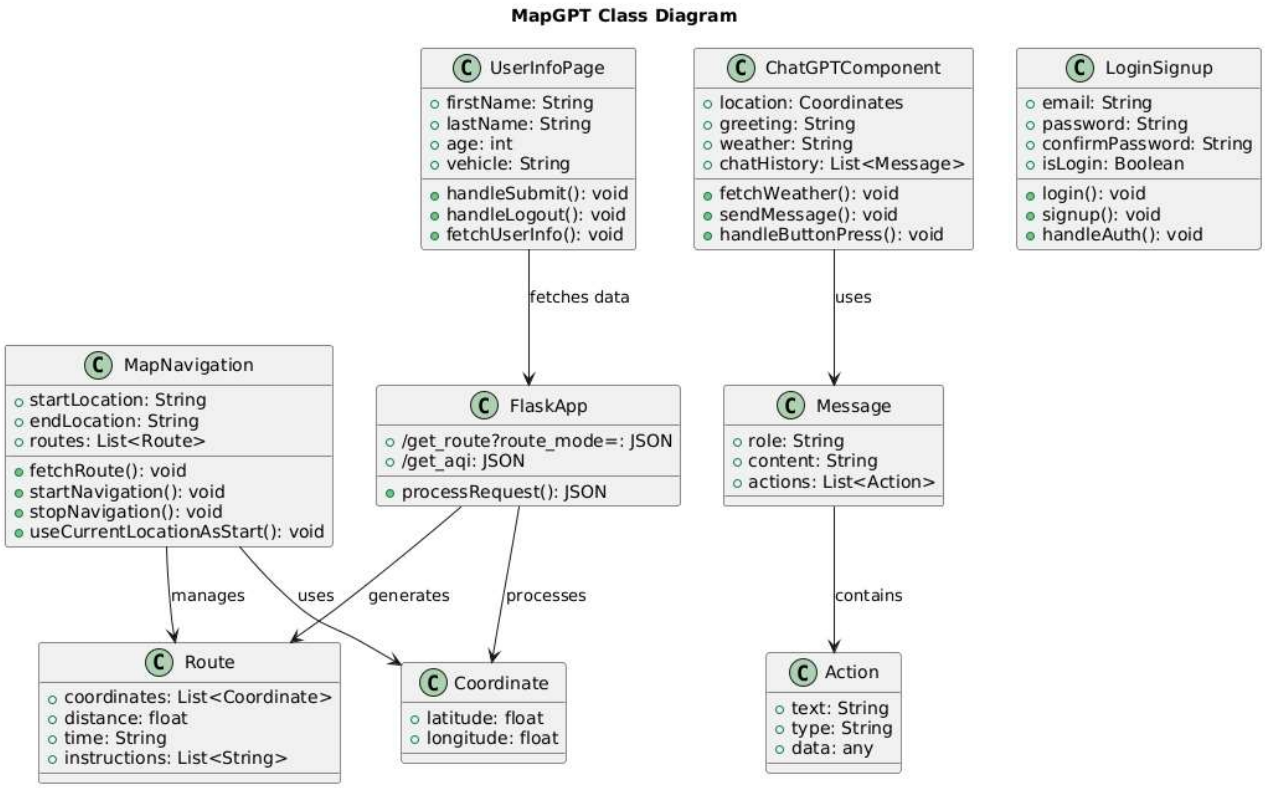


Figure 1. Route LLM Class Diagram

2.2 Backend Server

The backend server of Route LLM is a critical component that powers the app's advanced features, ensuring real-time data processing, secure user authentication, and personalized user experiences. The server is designed to handle various data inputs and outputs efficiently, providing the necessary support for the mobile app's operations.

Key elements of the backend server include:

1. **Mapbox Integration:** The backend leverages Mapbox for mapping functionalities and location suggestions. Mapbox provides precise coordinates for user-selected locations, enabling accurate map rendering and location-based services within the app. This integration ensures that users receive reliable and detailed map data in real-time.
2. **GraphHopper Integration:** To calculate and provide routes between coordinates, the backend uses GraphHopper. This routing engine processes the coordinates obtained from Mapbox and calculates the optimal route between them. The integration of GraphHopper ensures that users are guided along the most efficient paths, considering various routing factors.
3. **Firebase Authentication:** To securely handle user authentication, the backend makes advantage of Firebase Authentication. This service handles user sign-ins and ensures that user data remains secure throughout the app's usage. Firebase Authentication is critical in maintaining user privacy and enabling personalized experiences by identifying users accurately.
4. **Firebase Firestore**: Firebase Firestore is used to store user personal information, which is essential for providing personalized experiences through the ChatGPT-powered chatbox. This database securely stores details such as user preferences and interaction history, allowing the chatbox to tailor its responses and suggestions to each user's specific needs and past interactions.
5. **Weather API Integration:** The backend also integrates with a weather API to fetch real-time weather data. This data is used by the ChatGPT chatbox to greet users with weather updates, adding a personalized and contextually relevant touch to the interaction. By incorporating current weather conditions, the app enhances user engagement and delivers a more comprehensive service.
6. **API Service for Pollution and Crime Data:** The backend hosts a custom-built engine, hosted on Render, which acts as an API service. This engine provides the mobile app with a JSON file containing 10 coordinates, each with associated AQI (Air Quality Index) for pollution and CSI (Crime Severity Index) for crime and color coding, along with crime data. This real-time data integration allows the app to display relevant indicators on the map, helping users make informed decisions about their routes.

The backend server's architecture is designed for scalability and reliability, ensuring that the Route LLM app can efficiently handle user demands and deliver a seamless experience. By integrating various APIs and services, the backend provides the necessary support for the app's navigation, personalization, and data visualization features, making Route LLM a powerful tool for modern urban navigation.

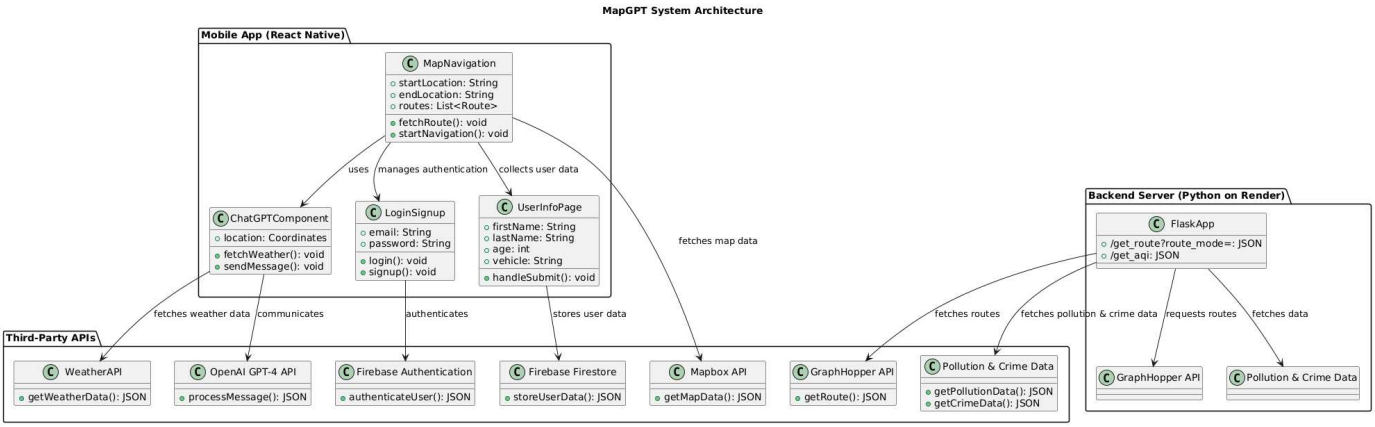


Figure 2. Route LLM System Architecture

2.3 Database

The Route LLM application mainly utilizes Firebase Firestore as its database solution, which is a critical factor in user data storage and management. Basically, Firebase Firestore is a NoSQL cloud database that provides scalable, real-time data storage. This enables the secure management of information on users and the delivery of experiences that are personalized within the app.

Key aspects of the database implementation include:

- **User Information Storage:** It uses the Firebase Firestore database for age, email, first name, last name, and vehicle type. Such data is quite critical in personalization within the app since it will ensure that the ChatGPT-driven chatbox adapts its interaction according to several details that are user-specific.
- **NoSQL Technology:** Firebase Firestore makes use of the NoSQL technology that enables flexible, hierarchical data storage. This structure is perfect for storing different user data in a form that can scale easily and at will, adapting to the needs of the app. The NoSQL model allows fast queries and real-time data updates necessary for a responsive user experience.
- **Real-Time Data Updates:** Firebase Firestore supports real-time data syncing, ensuring that any type of update made to user information shows up live in the app. This feature is

necessary for holding consistency amongst the profiles and for ChatGPT's chatbox to provide answers that are correct and updated.

- **Secure Data Management:** Firebase Firestore has inbuilt strong security features, into which it integrates authentication from Firebase Authentication. This helps keep users' data intact and lets only authenticated users view or modify sensitive information.
- **Scalability:** Firebase Firestore is a cloud-based solution, offers strong scalability, and can handle the rising load of users, still guaranteeing excellent performance. This can become most desirable for applications like Route LLM, which needs reliable and efficient data management as the app user base grows.

Besides user data storage, all the remaining information relating to the app is stored in Firebase Firestore. This information includes users' preference and interaction history. The data is therefore very important in tailoring each user's experience and thus changes depending on individual users.

2.4 Third-Party APIs

The Route LLM app integrates with a number of third-party APIs, most of which are targeted at improving the functionality of this application to make it more comprehensive in navigation. These APIs assist in providing real-time data and characteristics that make the application more user-friendly, accurate, and interactive.

Key third-party APIs integrated into the *Route LLM* app include:

1. **Mapbox API**:
    - **Functionality**: The Mapbox API is used for map rendering and providing location suggestions. It offers precise geographic data, allowing users to view their location and navigate the map seamlessly. The Mapbox API also provides coordinates for selected locations, which are essential for route planning and displaying relevant map data.
    - **Integration**: Within *Route LLM*, Mapbox handles all mapping-related functionalities, ensuring that users receive accurate, real-time geographic data. This integration is fundamental to the app's core functionality of visualizing routes and associated environmental data.
2. **GraphHopper API**:
    - **Functionality**: The GraphHopper API is utilized for calculating routes between the coordinates provided by Mapbox. It determines the most efficient path based on various routing parameters, ensuring that users are guided along the best possible route.

- **Integration**: *Route LLM* uses GraphHopper to process route calculations, which are then displayed on the map interface. This allows the app to provide accurate routing information, taking into account the user's current location and destination.

3. **Firebase Authentication API**:
   - **Functionality**: Firebase Authentication manages user sign-in and authentication processes. This API is very critical to the security of the app, so that only users who have authorized may gain access to a number of personalized features and stored data.
   - **Integration**: The app makes use of Firebase Authentication for user login and registration, ensuring highly protected data from every user and a unique, secure profile of each inside the app.
4. **Firebase Firestore API**:
   - **Functionality**: Store and manage user data, such as personal information and preferences, in Firebase Firestore. This API can be used by the app to fetch and real-time update data to keep users' profiles up to date.
   - **Integration**: In the App, Firebase Firestore is integrated for storing user information securely; this information is then utilized by the ChatGPT chatbox to give personalized interactions and suggestions.
5. **Weather API**:
   - **Functionality**: The weather API retrieves real-time weather data according to a user's current location from OpenWeatherMap. The application returns this data to users in the form of weather updates, hence making it more relevant and useful.
   - **Integration**: This weather data fetched via API is used by the ChatGPT chatbox itself, which greets the user with the current weather conditions at the front, providing a personalized and thus context-aware user experience.
6. **Custom API Service for Pollution and Crime Data**:
   - **Functionality**: A custom API service hosted on Render provides real-time data for the route chosen regarding pollution (AQI) and crime (CSI). It returns a JSON file with 10 coordinates, AQI and CSI values, and color-coding.
   - **Integration**: Since this API service drives the indicators for pollution and crime that appear on the map so that users are able to see in real time how safe and healthy their chosen route is, it is quite important.

These third-party APIs form an integral part of the functionality of Route LLM, realizing real-time, customized navigation experiences while taking priority to user safety and well-being. Each API contributes to the architecture that makes Route LLM a solid and versatile tool for modern urban navigation

3 Implementation

3.1 Introduction to React-Native-Typescript

Route LLM is built with React Native, a framework for cross-platform mobile development on both iOS and Android. Complementing React Native, the application uses TypeScript for enhancing the quality and maintainability of the code.

Key Benefits:

1. **Type Safety**: Static typing in TypeScript catches most of the errors at compile-time and avoids many runtime type errors.
2. **Improved Code Quality**: It forces cleaner, better-structured code with better documentation and makes a project easier to understand and maintain.
3. **Enhanced Development Experience**: Integration with modern IDEs provides features like intelligent code completion and refactoring tools, all of which make the development process easier.
4. **Seamless Integration**: TypeScript works very well with React Native libraries and components, helping to ensure uniformity in the development experience.

TypeScript in Route LLM, played an instrumental role in managing data flows and integrating third-party APIs to keep the app scalable and maintainable for further development.

3.2 Setting up the Mobile App

Setting up the *Route LLM* mobile app focused on the Android platform, ensuring a smooth and efficient development process tailored specifically for Android devices.

1. **Project Initialization**: The development began with initializing the React Native project configured with TypeScript. This setup included structuring the project, installing necessary dependencies, and configuring TypeScript for type safety and better code management.

2. **Android-Specific Configuration**:
    - **Platform Configuration**: The app was configured specifically for Android, addressing settings such as permissions, dependencies, and build configurations. This ensures optimal performance on Android devices.
    - **Development Environment**: Essential tools like a code editor (such as Visual Studio Code), Node.js, and Android Studio with an Android emulator were set up. This environment was crucial for a smooth development workflow and effective debugging.

3. **Third-Party API Integration**:
    - **Mapbox and GraphHopper**: Mapbox was integrated for mapping and location services, and GraphHopper for route calculations. These services were configured to work seamlessly within the app, providing real-time location and routing data specifically for Android.
    - **Firebase Services**: Firebase Authentication and Firestore were set up to handle user authentication and data storage, respectively. This involved configuring Firebase within the app and ensuring secure and efficient data management on the Android platform.

4. **User Interface Setup**: The UI was designed using React Native components, with a focus on creating an intuitive and user-friendly experience on Android devices. The dynamic theme was also configured to adapt to the system's light or dark mode settings.

5. **Testing and Debugging**: This app was lengthily tested on Android by emulators and physical device testing to ensure smooth navigation, real-time data update, and proper functioning of all the services integrated and APIs on the Android platform.

By following these steps, the *Route LLM* app was successfully set up for Android, ready for further development and feature implementation. This setup laid a strong foundation for delivering a reliable and responsive navigation experience to users on Android devices.

3.3 Integrating Firebase

Firebase integration is a key aspect of the *Route LLM* app, enabling secure user authentication and personalized data management.

Key Integration Steps:

1. **Firebase Authentication**:
    - **Setup and Configuration**: Integrated Firebase Authentication to manage user sign-ins, account creation, and security. This ensures that only authenticated users can access personalized features.
    - **Security**: Ensured secure management of user data, protecting it from unauthorized access.
2. **Firebase Firestore**:
    - **Data Storage**: Used Firebase Firestore to store user information such as age, email, and preferences. This data personalizes interactions within the ChatGPT chatbox.
    - **Real-Time Updates**: Implemented Firestore's real-time syncing to keep user profiles up-to-date across the app.
3. **Configuration and Testing**:

- **Connection**: Configured the app to securely connect with Firebase services.
- **Testing**: Thoroughly tested Firebase features, including authentication and data storage, to ensure smooth operation.

4. **Personalization**:
   - **ChatGPT Chatbox**: Leveraged Firestore data to personalize the chatbox experience, greeting users by name and tailoring responses based on stored preferences.

This integration of Firebase ensures a secure, scalable, and personalized user experience within the *Route LLM* app.

3.4 Data Integration and Processing

The *Route LLM* app integrates multiple data sources and services to deliver a seamless and personalized user experience. Here's a detailed breakdown of how these components are implemented and what data is stored and processed:

1. **Mapbox Integration**:
   - **Map Rendering and Location Suggestions**: The app uses Mapbox to render the map and provide real-time location suggestions. When users input start and end locations, the app fetches suggestions from Mapbox's geocoding API. The selected location data includes coordinates (latitude and longitude) and place names, which are stored temporarily in the app's state for immediate use in route calculations and map display.
   - **Dynamic Map Styling**: The map's appearance is adjusted dynamically according to the app's mode (e.g., navigation or browsing) and the system theme (light or dark). This ensures that the map is visually consistent and user-friendly across different scenarios.

2. **GraphHopper Integration**:
   - **Route Calculation**: Once the user selects start and end points, the app sends these coordinates to the GraphHopper API, which calculates the optimal route. The API provides comprehensive route information, such as:
     - **Coordinates**: The entire path is broken down into individual coordinate points.
     - **Distance**: The total distance of the route, typically in kilometers.
     - **Estimated Time**: The estimated time of travel by the chosen mode of transport.
     - **Instructions**: Turn-by-turn navigation instructions, such as turn left, continue straight.
   - This information is retained in the application state and guides changes to the map

view to display the route and render real-time navigation instructions.

3. **Custom API for Pollution and Crime Data**:
    - **Data Retrieval**: The app uses a custom API hosted on Render to retrieve pollution (AQI) and crime data (CSI) for the selected route. This API returns:
        - **Coordinates**: Specific segments of the route where data is relevant.
        - **AQI Values**: Air Quality Index values indicating the level of pollution.
        - **Color Codes**: Colors that represent different levels of pollution or crime risk, which are mapped to the route segments.
    - **Data Processing and Visualization**: The app processes this data to visually highlight affected route segments. For example, routes passing through high-pollution areas might be shown in red. This processed data is stored in the app's state and used to update the map in real-time.

4. **Firebase Integration**:
    - **User Authentication**: Firebase Authentication is used to manage user accounts and sessions. The authentication process involves storing:
        - **User Credentials**: Email and password for login, securely managed by Firebase.
        - **Session Data**: Details about the user's session are stored locally using AsyncStorage, allowing for persistent login across app restarts.
    - **Firestore Data Storage**: Firebase Firestore stores user-specific information, which includes:
        - **Personal Information**: First name, last name, age, vehicle type, and email address. This data is collected via the user info page and is used to personalize the ChatGPT chatbox and other app features.
        - **User Preferences**: Any preferences set by the user, such as their preferred route settings or interaction history with the app.
    - This data is securely stored in Firestore and is accessible across user sessions to ensure a personalized and consistent user experience.

5. **Weather API Integration**:
    - **Contextual Data**: The app fetches real-time weather data based on the user's current location using a weather API. The retrieved data includes:
        - **Temperature**: Either the current Fahrenheit or Celsius temperature.
        - **Weather Conditions**: Weather data contains information such as "Sunny," "Cloudy," or "Rainy."
    - **Data Usage:** This weather information is used by the ChatGPT chatbox to create contextual salutations and information display for users and further personalize the application.

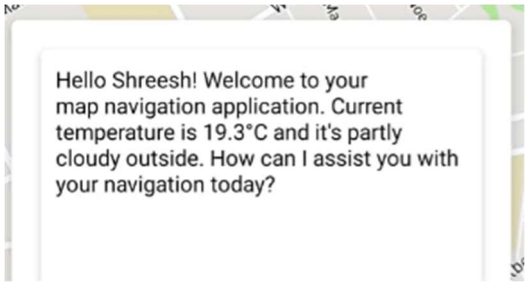


Figure 3. GPT chatbox greeting

6. **ChatGPT Integration**:
    - **Personalized Assistance**: The ChatGPT chatbox uses data from Firestore and the weather API to personalize its interactions. For instance:
        - **Greeting Users**: The chatbox greets users by their first name and includes real-time weather information.
        - **User Interactions**: It stores conversation history locally within the app's state during a session, allowing the conversation to remain contextually relevant.
        - **Knowledge Base Integration**: The chatbox uses a local JSON-based knowledge base to supplement its responses with additional relevant information, tailored to the user's queries and context.
    - The processed data from user interactions is stored temporarily within the app, ensuring a smooth conversational flow.

7. **User Interaction and Data Management**:
    - **Login and Signup Process**: The login and signup functionalities use Firebase Authentication to store and verify user credentials. Successful authentication triggers the retrieval or storage of user data in Firestore.
    - **User Profile Management**: Through the user info page, users can manage their personal information, such as first name, last name, age, and vehicle type. This information is crucial for the app's personalization features and is securely stored in Firestore.

By integrating these components, *Route LLM* ensures that all data—whether it be user information, route data, or environmental data—is handled securely and efficiently, providing a personalized and informed navigation experience.

3.5 User Interface Development

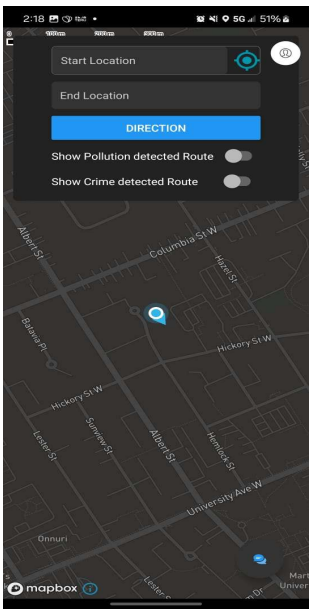
UI development of the Route LLM application was the most important part of the entire project to ensure there would be a smooth, intuitive, beautiful user interface.The UI was designed to cater to

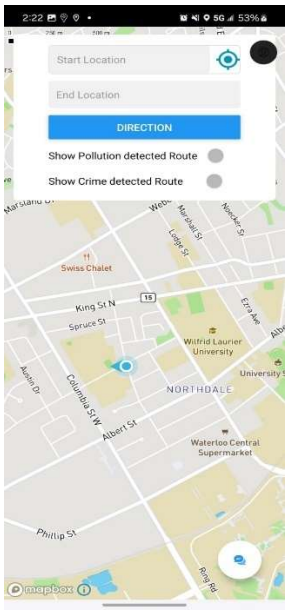
different user needs, from basic navigation to interactive AI-driven assistance, while also supporting both light and dark themes for better usability across different environments.

3.5.1 Home Screen Design

The home screen is the central hub of the Route LLM application, providing users with quick access to the core functionalities, such as setting the start and end locations, initiating directions, and toggling the display of pollution and crime-detected routes.

- **Design Elements:**
    - **Input Fields:** The start and destination input fields are right at the top of the screen, so a user can fill these values quickly. These are fitted with very clear placeholders and intuitive icons for easy comprehension.
    - **Direction Button:** There is a blue-colored "Direction" button centered to turn on the route search. The styling of the button differs to indicate clearly to the user that this is the main action on the screen.
    - **Toggles for Pollution and Crime Routes:** Flowing down from the direction button are two toggles. These allow the user to turn the display of pollution and crime-detected routes on and off. These toggles should be easy to get at and clearly indicate what they do.
    - **Home Screen Light Mode and Dark Mode**:



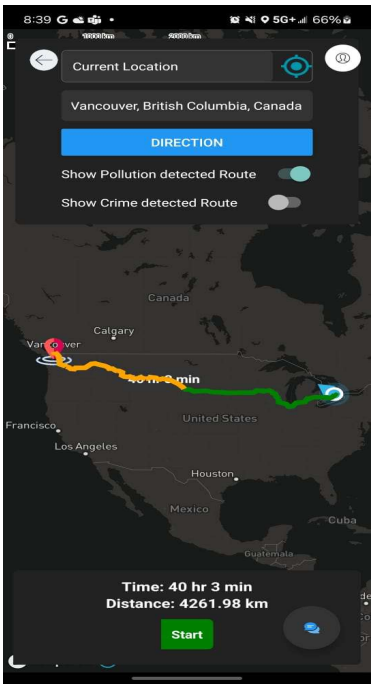


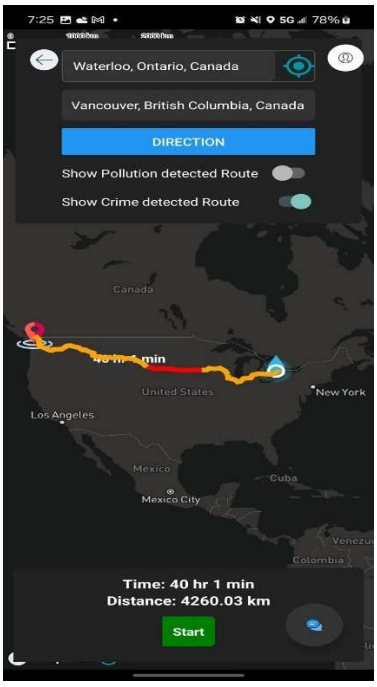
3.5.2 Navigation Screen Design

The navigation screen enables real-time visualization, where a user can view their current location and the route chosen. This screen combines data from Mapbox and OpenStreepMap (OSM) to guarantee accuracy and up-to-date information.

- **Design Elements:**
  - **Map Display:** The map display is the core of the navigation screen, giving a very clear and fine-grained depiction of the route. This indicates, at a glance, the location and route of the user on the map.
  - **Route Information:** The time and distance on the route are at the bottom of the screen so that the user can easily and quickly access such important information while navigating.
  - **Pollution and Crime Detection Overlays:** The features showing pollution and crime detection add an extra layer onto the map. Routes are colored according to environmental and safety data: green corresponds to a safe area, orange to a moderate, and red corresponds to high-risk areas. This feature provides real-time visual feedback to users about the route conditions based on the overlays.
  - **Light Mode and Dark Mode Navigation:**

- **Pollution and Crime Route Example:**





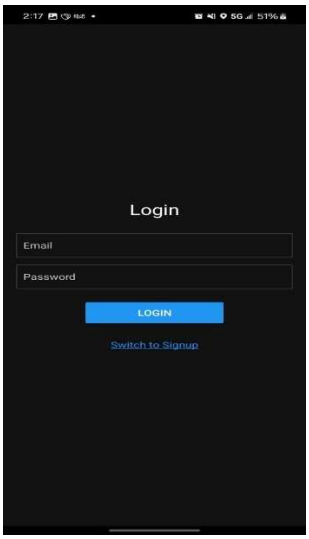
3.5.3 Login and Signup Screen Design

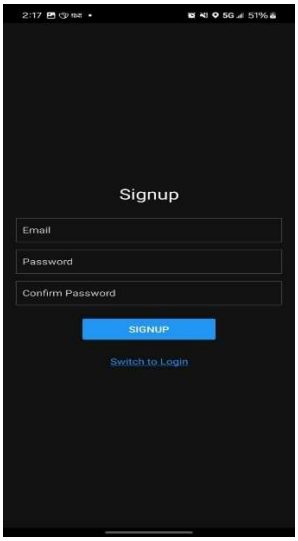
The login and signup screens are essential for user access and account management. The design prioritises clarity and simplicity, maintaining a unified appearance and experience on both screens.ti

- **Design Elements:**
  - **Input Fields:** The input fields for email and password are clearly labeled, with placeholders guiding the user on what to enter. The fields are easily accessible and follow a minimalistic design approach.
  - **Switch Option:** Users can easily switch between the login and signup screens using a link at the bottom of the screen, ensuring smooth transitions between the two functionalities.

- **Button Design:** The buttons for logging in and signing up are centrally placed and clearly labeled, with a blue color to indicate the primary action.

Login Screen and Signup Screen (Dark Mode):

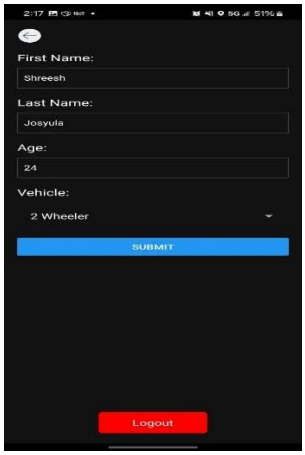




3.5.4 User Information Screen Design

After login/sign-up, the user will be asked for more information, including first name, last name, age, and preferred vehicle type. All this information enhances personalization of the user experience.

- **Design Elements:**
  - **Input Fields:** The fields are clearly labeled, and users can easily enter their personal information. The design follows the same minimalist approach as other parts of the app, maintaining consistency in user experience.
  - **Logout Button**: At the bottom of the display, there is a logout button that users can use to close the session. It is colored red to make its meaning obvious.

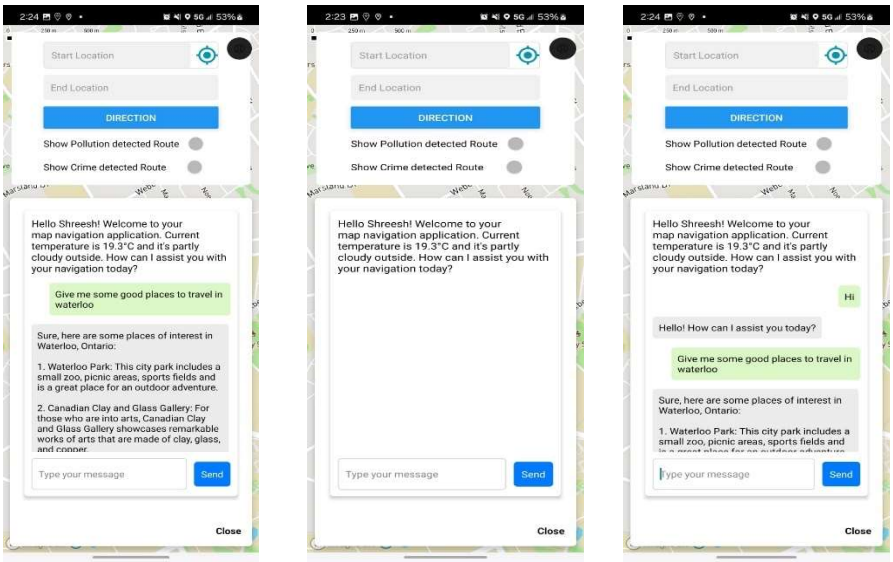


3.5.5 Chat Interface Design

One of the main features of Route LLM is a chat interface that includes ChatGPT, making the application capable of AI-driven assistance to users regarding navigation and location-based queries. The design of the chat interface will be user-friendly and engaging.

- **Design Elements:**
  - **Chat Box:** It is designed just like any other messaging interface, making it self-explanatory to be used. A text input field at the bottom of the screen allows the user to write their message with a "Send" button beside it.
  - **Message Display:** This is where messages by the user and ChatGPT are displayed in conversational form, clearly distinguishing between user input and AI responses. The color differentiation between the user and the AI makes the messages different in color.
  - **Contextual Responses:** The chat interface allows ChatGPT to provide information relevant to the context of a user's input, therefore making the interface more interactive and person-centered.

    Chatbox with ChatGPT Greeting and Suggestions:



3.6 Evaluation

Testing of the Route LLM application focused on checking the features implemented to work correctly, responses from the incorporated API were accurate, the user interface was responsive and usable. Since the application depends much on real-time data received from various called APIs, the evaluation process had to assess the response times, accuracy, and relevance of data provided by these APIs in normal conditions of use.

Functional Testing

- **Navigation Accuracy:** The navigation routes generated by the application were cross-verified with the actual routes provided by the GraphHopper API and OpenStreetMap (OSM). The application successfully generated the shortest routes and accurately displayed them on the map. The app was capable of handling multiple types of routes, including those detecting pollution levels and crime rates. The integration of pollution and crime data allowed the app to provide users with valuable information, enabling them to select safer or cleaner routes based on real-time environmental and safety conditions.

- **Pollution and Crime Data Fetching:** The application was tested to ensure that it could correctly fetch and display pollution and crime data for different routes. The server-side code

was modified to support two different endpoints (/route_mode=pollutionroute and /route_mode=crimeroute), allowing the same server to provide distinct responses based on the requested route mode. This functionality was evaluated by simulating various route requests and verifying that the application correctly displayed the pollution and crime levels on the map, providing accurate classifications that users could rely on when planning their journeys.

Diagram: Route Classification and Data Fetching Process

To provide a clearer understanding of the process involved in fetching and classifying routes based on pollution and crime data, the following diagram illustrates the flow of data and decisions made by the application:

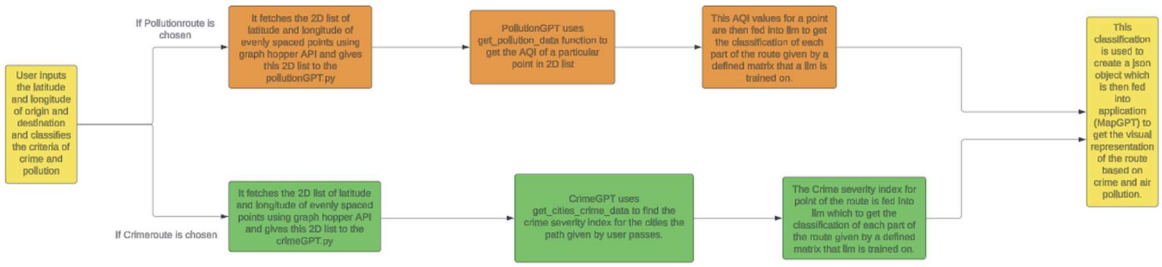


Figure 4. Route classification and data fetching for Pollution and Crime

- **ChatGPT Responses:** The ChatGPT integration was evaluated by testing various user queries related to navigation, location-based suggestions, and weather updates. The responses were relevant and correctly personalized based on user data fetched from Firebase, such as the user's first name. For instance, when queried about places to visit in Waterloo, the responses included accurate and meaningful suggestions that matched the user's location.

- **API Integration:** The application integrates multiple APIs, including OpenAI's GPT-4 for conversation, WeatherAPI for weather updates, and Firebase for user data storage. During testing, the APIs responded within acceptable time frames, and the data received was accurate and up-to-date. However, the application did not undergo load testing or stress testing, so it is unclear how it would perform under heavy user loads or in scenarios requiring high concurrency.

User Interface Evaluation

The user interface was tested across different devices and screen sizes to ensure consistency in user

experience. Both light and dark themes were implemented successfully, allowing users to switch between themes based on their system settings. The design is responsive, with all buttons and input fields functioning as expected. Additionally, the user interface was evaluated for its ability to handle the display of complex data, such as pollution and crime levels on the map, ensuring that the information was presented clearly and intuitively to users.

3.7 Deployment to Render and Python Code Modifications

In this section, I will discuss how I deployed the Python engine for the Route LLM application to Render, a cloud platform for hosting applications, and the specific changes I made to the Python code to facilitate this deployment.

Deployment to Render

The deployment process involved setting up the Python application as a web service on Render, allowing it to serve as an API that the Route LLM mobile application could interact with. Here's how the deployment was carried out:

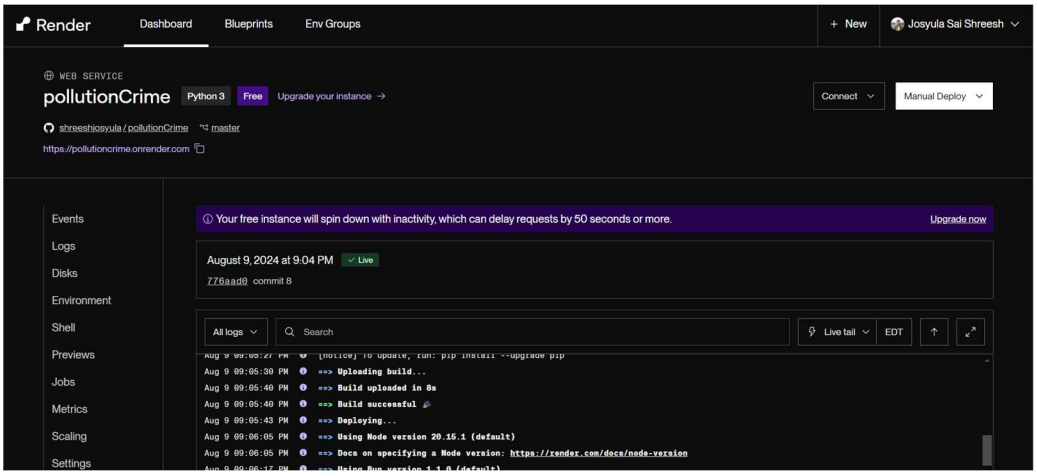


Figure 5. Hosted Server for Pollution and Crime GPT

1. **Setting Up the Render Service**:
    - First, I created an account on Render and set up a new web service. Render provides a straightforward interface for deploying web services, which made the setup process seamless.
    - I selected the repository containing my Python code from GitHub and connected it to Render. This allowed Render to automatically deploy the code whenever I pushed updates to the repository.
2. **Configuring the Environment**:
    - To ensure that the application could run smoothly on Render, I specified the Python version and any dependencies required in the requirements.txt file. This file was included in the root of the project and contained all the necessary packages, such as Flask for the web framework and additional libraries required for processing.
    - Environment variables, such as API keys for external services, were configured within Render's dashboard. This ensured that sensitive information was not hardcoded into the application and could be managed securely.
3. **Setting Up the Web Service**:
    - The Python application was modified to run as a Flask web service, which listens for HTTP requests and returns JSON responses. This was necessary for the mobile application to make API requests to the deployed service.
    - Render automatically handles the deployment and hosting, so after setting up the service, it was available at a public URL. The mobile app could then interact with this URL to fetch data.
4. **Continuous Deployment**:
    - With the Render service connected to GitHub, any changes pushed to the repository triggered an automatic deployment. This continuous deployment setup ensured that the latest version of the Python engine was always live and accessible to the Route LLM app.

Python Code Modifications

To adapt the Python code for deployment on Render, several key changes were made:

1. **Flask Integration**:
    - The original Python code, which might have been written for local execution, was modified to run as a Flask application. I added routes that correspond to different API endpoints that the mobile app would call.
    - For example, if the Python engine calculates air quality data, I created a /get_aqi endpoint that the mobile app could request to receive this data in JSON format.
2. **Environment Variables**:

- Hardcoded API keys and other sensitive information were replaced with environment variables. This change ensured that these details could be managed securely on Render without being exposed in the codebase.
- To get these environment variables and include them into the program, I utilised Python's os module.
-

3. **Dynamic Route Handling**:
   - A significant modification was made to the code to enable the same server to provide different responses based on the route mode specified in the API call. This was achieved by introducing a query parameter, route_mode, which could be set to either pollutionroute or crimeroute.
   - Depending on the value of this parameter, the server processes the request differently and returns the corresponding data. For instance, when the API is called with /route_mode=pollutionroute, it returns a route optimized to avoid high pollution areas. Similarly, when called with /route_mode=crimeroute, it returns a route that minimizes exposure to high-crime areas.
   - This dynamic handling made the service versatile, allowing it to serve multiple use cases without needing separate endpoints for each.
4. **Error Handling and Logging**:
   - Since the application would be running in a production environment, I added more robust error handling to the code. This included catching exceptions that might occur during API calls or data processing and returning appropriate HTTP status codes and error messages to the client.
   - Logging was also enhanced to track errors and monitor the application's performance on Render.
5. **Optimization for Web Deployment**:
   - The code was reviewed and optimized for better performance in a web service environment. This included optimizing data processing tasks to minimize response times and ensure that the API could handle multiple requests efficiently.

3.8 Future Work

Although the current form of the Route LLM application already offers a robust and functional solution for route planning that takes into consideration safety and environmental sensitiveness, there are a number of future improvements and added features that could be made to improve user experience and the general utility of the application. More importantly, the base code has been future-proofed to allow many of these enhancements with just a few lines of change in the existing code.

1. **Integration of a Voice Assistant and GPT-4 Features**:
   - One potential enhancement is the integration of a voice assistant feature, enabling

users to interact with the application through voice commands. This would make the application more accessible, especially while driving, where hands-free operation is crucial.
    - Additionally, leveraging advanced capabilities of GPT-4, such as the GPT-4-0 feature, could provide more intelligent and context-aware assistance. This could include offering more personalized recommendations, answering complex queries about routes, or even engaging in more natural and conversational interactions with the user.

2. **Modification of the Python Code to Fetch Multiple Routes**:
    - Another future enhancement could involve modifying the Python engine to fetch and process multiple route options in parallel. Currently, the application provides a single route optimized based on the selected criteria (pollution or crime index).
    - By adjusting the Python code, the application could generate several route options and allow users to compare these routes directly. This could involve fetching multiple JSON files from the server, each representing a different route with varying levels of pollution or crime exposure, and then displaying these options to the user.

3. **User-Driven Route Selection Based on Pollution and Crime Indices**:
    - Another improvement in the future could be in the route selection decision-making process: allow users to make a choice based on detailed pollution and crime indices.
    - It could be designed to provide a quick summary of each route in terms of its pollution levels, crime rates, and other factors involved in each journey. The user would then have the opportunity to choose which route best suits his or her priorities: whether safety, environment, or time.
    - An innovative feature that would empower users to make better decisions while enhancing the overall customization and user-centric design of the application.

Given the modular and scalable design of the Route LLM application's codebase, these future works can be implemented with relative ease. The architecture is future-proof, meaning that these additional features can be integrated with minimal disruption to the existing functionality, ensuring the application remains adaptable to evolving user needs and technological advancements.

4 Conclusion

Another milestone in smart navigation is reached with the development of Route LLM, putting into one's hands a solution much richer than classic route planning. Such real-time data as that on environmental conditions pollution (AQI), crime data (CSI) are integrated in this program, making it an exclusive and invaluable tool for users who want to enhance their safety and general awareness when on the move. This application merges data with conventional tools of navigation to guide users not only along the fastest routes but also along the safest and greenest ones.

Furthermore, an easy and gorgeously simple user interface has been designed that is context- and requirement-sensitive to users at large in light or dark mode. Thus, further integration of ChatGPT helps Route LLM not to fall behind other competitive applications, particularly on navigation. ChatGPT adds customized, AI-driven support that can respond to customer inquiries and place recommendations based on locations, improving the overall user experience. The addition thus turns the app into a more engaging platform that becomes versatile, much more than a mere navigational tool and one-stop travel guide.

Another critical component of this project was the deployment of the Python backend onto Render, making sure that the service scales and handles real-time queries. The Python code has been modified to handle route mode like crime and pollution dynamically, showing the flexibility of the application for further development. Due to the well thought-out, future-oriented architecture of the codebase, little modification should be needed to add new features, such as a voice assistant or the ability to retrieve and compare different routes.

With these upcoming features, Route LLM will turn into an even more formidable tool in the future. The status of this application as one of the leading intelligent solutions in navigation will be enhanced with features such as route selection according to comprehensive pollution and crime indices, coupled with sophisticated AI capabilities like a voice assistant. Thus, the base that this project provides assures further development of Route LLM and adaptation to the ever-changing needs of its users.

While the target vision is ambitious, Route LLM offers much more than a navigation app; it's an ingenious and rather far-looking solution to put convenience, users' safety, and the environment at the forefront. Given its integrations of multiple data sources, ultramodern AI capabilities applied, and ease of use, Route LLM will be one very stand-alone application in the field of navigation. In this way, while it grows in functionality and further develops its enhancements, the app could become a yardstick for any new breed of intelligent and AI-driven navigational tools in improving the very way users engage with their surroundings and travel through the larger world.

References

- React Native Documentation. - https://reactnative.dev/docs/getting-started
- Mapbox GL JS Documentation. - https://docs.mapbox.com/mapbox-gl-js/
- GraphHopper API Documentation - https://docs.graphhopper.com/
- Firebase Documentation. - https://firebase.google.com/docs
- OpenStreetMap Wiki - https://wiki.openstreetmap.org/wiki/Main_Page
- React Navigation Documentation. - https://reactnavigation.org/docs/getting-started
- React Native Maps Documentation. - https://github.com/react-native-maps/react-native-maps
- Overpass API Documentation. - https://wiki.openstreetmap.org/wiki/Overpass_API
- OpenWeatherMap API Documentation - https://openweathermap.org/current
- Mapbox Directions API Documentation. - https://docs.mapbox.com/api/navigation/directions/
- Mapbox Map Matching API Documentation - https://docs.mapbox.com/api/navigation/map-matching/
- React Native Firebase Documentation. (n.d.). Retrieved from https://rnfirebase.io/
- OpenAI GPT- 4-o - https://platform.openai.com/docs/overview
- Render - https://docs.render.com/
- Flask - https://flask.palletsprojects.com/en/3.0.x/