**SMART COMMUTE PLANNER**

1. OVERVIEW: **Smart Commute Planner** is a travel planning application designed to make your journeys more convenient. It offers a largely no-touch experience, meaning that once set up, the app continuously monitors and provides the best travel options without needing constant input. It helps you plan your travel based on your preferences, including the best timings, routes, and real-time traffic updates.
2. USE-CASES:
   1. Plan your travel between a source and destination location.
   2. Input your source and destination into the application, and it will track the optimal route for reaching your destination.
   3. Set the time you intend to leave for your destination, and the application will suggest the best time to depart to avoid traffic, ensuring a smoother journey.
3. FEATURES:
   1. **No-Touch Operation:** After an initial login/registration and input of your source, destination, and departure time, the app works in the background. You don't need to interact with it regularly; it will notify you based on your set preferences.
   2. **Notification Preferences:** The app can send notifications through default channel, i.e., App Notification and through various channels—Mobile, WhatsApp, or Email—based on your chosen preference.
   3. **Real-Time Updates:** As traffic conditions change, the app will notify you during your journey if a faster route becomes available or if the predicted arrival time changes. The app will provide the new route and updated estimated time of arrival in the notification.
   4. **Multiple Notification Channels:** You can opt to receive notifications through one or multiple channels, ensuring you stay informed in the way that suits you best.
   5. **Recurring Feature:** Users can select specific days of the week to receive notifications. This allows users to disable notifications on weekends or other specific days. Users have the option to manually select days or opt for notifications on all days.

**Same feature when we use the Alarm application in our mobile.**

**ROADMAP**

1. Project Planning:
   1. Requirement Gathering:
      1. User Stories: This helps us to understand the application from the perspective of the end-users.
         1. User Registration & Login:
            1. *As a user, I want to register and login to the application so that I can save my travel preferences and receive notifications.*
            2. *As a user, when I login to the application, I can update my Profile details like name, email address or password.*
            3. *As a user, when I login to the application, I can update the notification preferences as well.*
         2. Set Travel Preferences:
            1. *As a user, I want to set my source and destination locations so that the app can track the best route for me.*
            2. *As a user, I want to set the time I plan to leave so that the app can notify me when to depart to avoid traffic.*
            3. *As a user, I want to set the recurring days of my travel, so that I don’t have to come everyday to set the travel.*
            4. *As a user, I should have the option to select the mode of transportation (car, bus, bike).*
         3. Receive Notifications:
            1. *As a user, I want to receive notifications through my preferred channel (WhatsApp, SMS & Email) (App Notification (default)), so that I can stay informed about my travel.*
            2. *As a user, I want to receive real-time updates on traffic and route changes during my journey so that I can arrive on time.*
         4. View Travel History:
            1. *As a user, I want to view my past travel plans and routes so that I can analyze my travel patterns.*
         5. Adjust Preferences:
            1. *As a user, I want to update my travel preferences at any time so that I can adapt to changing schedules or routes.*
      2. Functional Requirements: These are the specific functions the application must perform to satisfy the user stories:
         1. User Management:
            1. User registration, login and authentication.
            2. Profile Management (update personal information, notification preferences).
         2. Travel Planning:
            1. Allow users to input and save source and destination locations.
            2. Option to save frequent locations (e.g., Home, Work).
            3. Choose recurring days in between the weeks.
            4. Options to set the time of travel.
         3. Route Calculation:
            1. Calculate the best route based on the current traffic conditions.
            2. Predict the optimal departure time based on historical traffic data.
         4. Notifications:
            1. Send notifications via App Notification, SMS, WhatsApp, or Email.
            2. Support real-time notifications via the same channel for traffic changes and route updates.
         5. Machine Learning Integration:
            1. Predict traffic conditions based on time of day, weather, and historical data.
            2. Optimize route suggestions dynamically during the journey.
         6. Travel History:
            1. Store and display travel plans and routes.
            2. Provide insights or suggestions based on travel history.
      3. Non-Functional Requirements: These requirements focus on the quality attributes of the system.
         1. Performance:
            1. The app should provide real-time notifications with minimal latency.
            2. Route calculations should be completed within a few seconds.
         2. Scalability:
            1. The system should be able to handle a growing number of users and requests.
         3. Security:
            1. Secure user data with encryption (e.g., passwords, personal information).
            2. Ensure secure communication channels for notifications.
         4. Usability:
            1. The user interface should be intuitive and easy to navigate.
            2. Provide clear instructions and feedback to users.
         5. Reliability:
            1. The app should be available 99.9% of the time.
            2. Implement fault tolerance for key services (e.g., notifications).
      4. Technical Requirements:
         1. Frontend:
            1. **React.js** for building the user interface.
            2. Redux or **Context API** for state management.
         2. Backend:
            1. **Node.js** with **Express.js** for building RESTful APIs.
            2. Integration with third-party services like Twilio (for SMS, WhatsApp) and NodeMailer/SendGrid (for email).
         3. Database:
            1. **MongoDB** for storing user data, travel plans, and historical data.
         4. Machine Learning:
            1. **Python** for building traffic prediction models. (e.g., using libraries like TensorFlow, Scikit-learn)
            2. Integration of the ML model with the Node.js backend.
         5. Deployment:
            1. Cloud services like **AWS**, Heroku, Azure or GCP for hosting.
            2. CI/CD Pipeline for continuous integration and deployment.
   2. Requirement Prioritization:
      1. Priority 1: Must-Have
         1. User Management:
            1. User registration, login, and authentication.
            2. Profile management (update personal information, delete account).
         2. Travel Planning:
            1. Allow users to input and save source and destination locations.
            2. Set departure time for a specific travel plan.
            3. Options to save frequent locations, like Home, Work, Other etc.
            4. Allow users to set the recurring days for the same travel plan[Store the data as an array].
            5. Implement the CRUD operation, which allows to create, read, update and delete the travel plans.
         3. Travel History:
            1. After user creates a travel plan, it should display with all the details of the travel.
         4. Route Calculation:
            1. Calculate the best route based on current traffic conditions.
            2. Predict the optimal departure time based on historical traffic data and the current traffic conditions.
         5. Notifications:
            1. Send notifications via App Notifications for departure and real-time updates.
            2. Support real-time notifications for traffic change and route updates.
         6. Frontend:
            1. User interface for registration, login, and travel plan input.
            2. Integration with the backend to display calculated routes and notifications.
         7. Backend:
            1. RESTful APIs for user management, travel planning, and notifications.
            2. Integration with third-party services for notifications.
         8. Database:
            1. MongoDB schema for users, travel plans and preferences.
      2. Priority 2: Should-Have
         1. User Management:
            1. Implement the feature of Forgot password. Take the help with the security questions.
            2. Implement the password validation before deleting the user account.
         2. Machine Learning Integration:
            1. Initial implementation of traffic prediction models.
            2. Dynamic route optimization based on real-time traffic updates.
         3. Notifications:
            1. Expand notification channels to include SMS, WhatsApp or Email.
            2. Option to add and update the notification preferences in the Profile Management section.
         4. Frontend:
            1. Enhanced user interface with additional features like travel history, frequent locations, and details route information.
      3. Priority 3: Nice-to-Have
         1. Usability Enhancements:
            1. Advanced UI/UX improvements for a more polished experience.
            2. User onboarding and help guides.
         2. Machine Learning Enhancements:
            1. Advanced prediction models that consider more variables like weather, events, etc.
            2. Continuous learning to improve accuracy over time.
         3. Scalability and Performance Optimization:
            1. Optimizing the backend and database for handling large-scale traffic.
            2. Load balancing and failover mechanisms.
         4. Advanced Notifications:
            1. More customizable notification preferences (e.g., different types of alerts for different situations).
         5. Additional Features:
            1. Multi-language support.
            2. Integration with calendar apps for automatic travel planning.
   3. Implementation Plan:
      1. Project Setup (Week 1)
         1. Version Control:
            1. Set up Git/GitHub repository.
            2. Define branching strategy (e.g., feature branches, development branch).
         2. MERN Stack Setup:
            1. Initialize the project structure.
            2. Setup Node.js with Express.js for the backend.
            3. Setup React.js for the frontend.
            4. Setup MongoDB for the database.
         3. Basic CI/CD Pipeline:
            1. Setup CI with automated testing.
            2. Deploy a basic version of the app to a cloud service (**Current: Render**).
      2. Core User Management (Week 2-3)
         1. User Registration & Login:
            1. Implement authentication using JWT (JSON Web Tokens).
            2. Create API endpoints for registration, login and profile management.
            3. Basic frontend forms for registration and login.
         2. Profile Management:
            1. Allow users to update personal information or delete accounts.
      3. Travel Planning & Route Calculation (Week 4-5)
         1. Travel Planning:
            1. Develop forms to input source and destination locations.
            2. Implement backend login for storing travel plans.
         2. Route Calculation:
            1. Integration a basic route calculation service (e.g., Google Maps API).
            2. Calculate optimal routes based on current traffic conditions.
            3. Implement departure time prediction based on historical data.
            4. Display route information on the frontend.
      4. Notifications (Week 6-7)
         1. Notification Service Setup:
            1. Integration Twilio for SMS notifications.
            2. Integration SendGrid/NodeMailer for Email notifications.
         2. Notification Logic:
            1. Implement backend logic to send notifications based on travel plans.
            2. Develop a mechanism to trigger real-time updates and notifications.
         3. Frontend Integration:
            1. Allow users to set and update notification preferences.
            2. Display notification history on the frontend.
      5. Machine Learning Integration (Week 8-10)
         1. Traffic Prediction Model:
            1. Develop a basic traffic prediction model using Python.
            2. Train the model using historical traffic data.
         2. Model Integration:
            1. Integration the ML model with the backend to provide real-time predictions.
            2. Update route calculations logic to consider ML predictions.
         3. Testing and Optimization:
            1. Test the accuracy of predictions and optimize the model as needed.
      6. Travel History & Additional Features (Week 11-12)
         1. Travel History:
            1. Implement backend logic to store and retrieve past travel plans.
            2. Create a frontend interface for users to view their travel history.
         2. Frequent Locations:
            1. Allow users to save frequent locations (e.g., Home, Work).
            2. Integrate these locations into travel planning.
         3. WhatsApp Notifications:
            1. Expand notification channels to include WhatsApp.
         4. Advanced Notifications:
            1. Enhance notification settings with more customization options.
      7. Testing, Debugging, and Optimization (Week 13-14)
         1. Unit and Integration Testing:
            1. Write and run tests for all critical components (frontend and backend).
            2. Conduct end-to-end testing for the full user journey.
         2. Performance Optimization:
            1. Optimize database queries, API responses, and frontend performance.
         3. Security Review:
            1. Conduct a security audit to ensure data protection and secure communications.
      8. Final Deployment & Launch (Week 15)
         1. Final Deployment:
            1. Deploy the fully functional application to the cloud.
            2. Set up monitoring and alerting for the live application.
         2. Launch:
            1. Announce the launch to initial users.
            2. Provide a feedback mechanism for continuous improvement.

**DEVELOPMENT PHASE**

1. Project Setup (Week 1)
   1. Version Control Setup - **Done**
      1. Create a GitHub Repository:
         1. Set up a new repository on GitHub.

**https://github.com/somjeet2000/smart-commute-planner-client.git**

**https://github.com/somjeet2000/smart-commute-planner-server.git**

* + - 1. Define the repository structure, including branches (e.g., main, development, feature/branch-name).

Branch Created – **main(default), development**

* + 1. Set Up Git Ignore:
       1. Include a .gitignore file to exclude unnecessary files from version control.
    2. Initial Commit:
       1. Make an initial commit with the basic project structure and setup files.
  1. MERN Stack Setup - **Done**
     1. Backend Setup (**Done)**:
        1. Initialize a new Node.js project in the backend directory.
        2. Install Express.js and other necessary dependencies.
        3. Set up the basic server structure with a sample API.
     2. Frontend Setup (**Done)**:
        1. Create a new React.js project in the frontend directory.
        2. Install necessary dependencies.
        3. Set up a basic React component structure.
     3. Database Setup (**Done)**:
        1. Set up MongoDB locally or on a cloud service (e.g., MongoDB Atlas).
        2. Create a basic schema/model for users in the backend.
        3. Ensure the backend can connect to the MongoDB instance.
  2. Basic CI/CD Pipeline
     1. Continuous Integration Setup (**Done)**:
        1. Configure **Jenkins** to run automated tests on push requests (**Done)**.
        2. Write simple tests for both the frontend and backend (Jest) - **Done**.
     2. Basic Deployment:
        1. Deploy a basic version of the application to **Render**. - **Done**

**Render URL: https://smart-commute-planner-v1.onrender.com**

* + - 1. Set up environment variables securely using the service's built-in features. - **Done**
  1. Documentation:
     1. README.md: - **Done**
        1. Update the README.md file with project details, setup instructions, and contribution guidelines.

1. Core User Management (Week 2-4):
   1. Objective: Implement core user management features including user registration, login and profile management.
   2. User Registration & Login:
      1. Backend Development:
         1. Set Up Authentication:
            1. Implement user registration endpoints (`**POST api/v1/auth/create\_user**`).
            2. Implement user login endpoints (`**POST /api/v1/auth/login\_user**`).
            3. Use JSON Web Tokens (JWT) for authentication.
         2. Password Handling:
            1. Use `**bcrypt**` for hashing passwords.
            2. Validate passwords securely.
         3. Database Integration:
            1. Design and implement the user schema in MongoDB (e.g., username, email, password etc.).
      2. Frontend Development:
         1. Registration Form:
            1. Create a registration form with fields for Username, email and password.
            2. Implement form validation and error handling.
         2. Login Form:
            1. Create a login form with fields for email and password.
            2. Implement form validation and error handling.
         3. Authentication State:
            1. Manage authentication state with **Context API**.
      3. Testing:
         1. Unit Testing:
            1. Write unit tests for registration and login endpoints.
         2. Integration Testing:
            1. Test registration and login flows in the frontend.
         3. End-to-End Testing:
            1. Validate end-to-end functionality of user registration and login.
   3. Profile Management:
      1. Backend Development:
         1. Profile Update Endpoint:
            1. Implement endpoint for updating user profile (`**PUT /api/v1/auth/update\_user**`).
            2. Allow updates to personal information (e.g., name, email).
         2. Password Update Endpoint:
            1. Implement endpoint for updating user password (**`PUT /api/v1/auth/update\_user\_password`**).
            2. Validate current password before updating.
         3. Delete Account Endpoint:
            1. Implement endpoint for deleting the user account (`**DELETE /api/v1/auth/delete\_user**`).
      2. Frontend Development:
         1. Profile Management UI:
            1. Create a profile management page where users can update personal information.
            2. Implement forms for updating personal details and changing passwords.
            3. Create a **Delete Account** button to delete the user account. Implement the alert message “Are you sure want to delete the account?”
         2. User Alert:
            1. Provide clear alert messages on successful updates or errors.
      3. Testing:
         1. Unit Testing:
            1. Write unit tests for profile management endpoints.
         2. Integration Testing:
            1. Test profile update and password change functionality in the frontend.
         3. End-to-End Testing:
            1. Validate the complete flow of updating profile information and password.
   4. Documentation & Deployment:
      1. API Documentation:
         1. Document the API endpoints for user registration, login and profile management using tools like Postman.
      2. Deployment:
         1. Deploy backend changes to the **Render**.
         2. Deploy frontend changes to the **Netlify/Render**.