HIMANSHU HITCH

1. Introduction

1.1 Purpose

HIMANSHU HITCH aims to revolutionise transportation by providing a user-friendly platform for efficient ride-sharing, resulting in reduced travel costs and minimised environmental impact. Through advanced algorithms, the platform optimises routes and matches users with compatible travel plans, enhancing convenience and affordability. By encouraging collaboration among commuters, HIMANSHU HITCH not only reduces traffic congestion but also contributes to a more sustainable transportation system, leading to cleaner air and a healthier environment.

1.2 Scope

The HIMANSHU HITCH application offers a comprehensive suite of features to enhance the user experience throughout every stage of their journey. It begins with a seamless user registration process, allowing individuals to create personalised accounts with ease. Once registered, robust authentication mechanisms ensure the security and integrity of user accounts, safeguarding sensitive information. The platform empowers users to initiate and find rides effortlessly, providing intuitive interfaces to input journey details such as starting point, destination, and preferred time.

1.3 Overview

The HIMANSHU HITCH application, developed using the MERN stack, leverages MongoDB for flexible data storage and Express.js for efficient server-side logic. React.js powers the dynamic client-side interface, while Node.js facilitates fast and scalable server operations. Additionally, the integration of geolocation, mapping, and routing APIs enhances functionality, interactive mapping, and optimised route planning for a seamless ride-sharing experience

2. Database Models

2.1 User Model

The user model represents individuals who interact with the system. It contains the following fields:

- **Name**: The full name of the user.
- **Email**: The email address associated with the user's account, used for communication and authentication.
- **Phone number**: The contact number provided by the user.
- **Password**: A securely hashed password for authentication purposes.
- **User photo**: An optional field allowing users to upload a profile picture.
- **Trips history**: A record of past trips taken by the user, including details such as trip ID, source, destination, date, and completion status.
- **Active trip**: Information about any trip the user is currently on, including trip ID, driver details, source, destination, waypoints, and current status.
- **Driver status**: Indicates whether the user is registered as a driver, with possible statuses such as active, inactive, or pending approval.

2.2 Trip Model

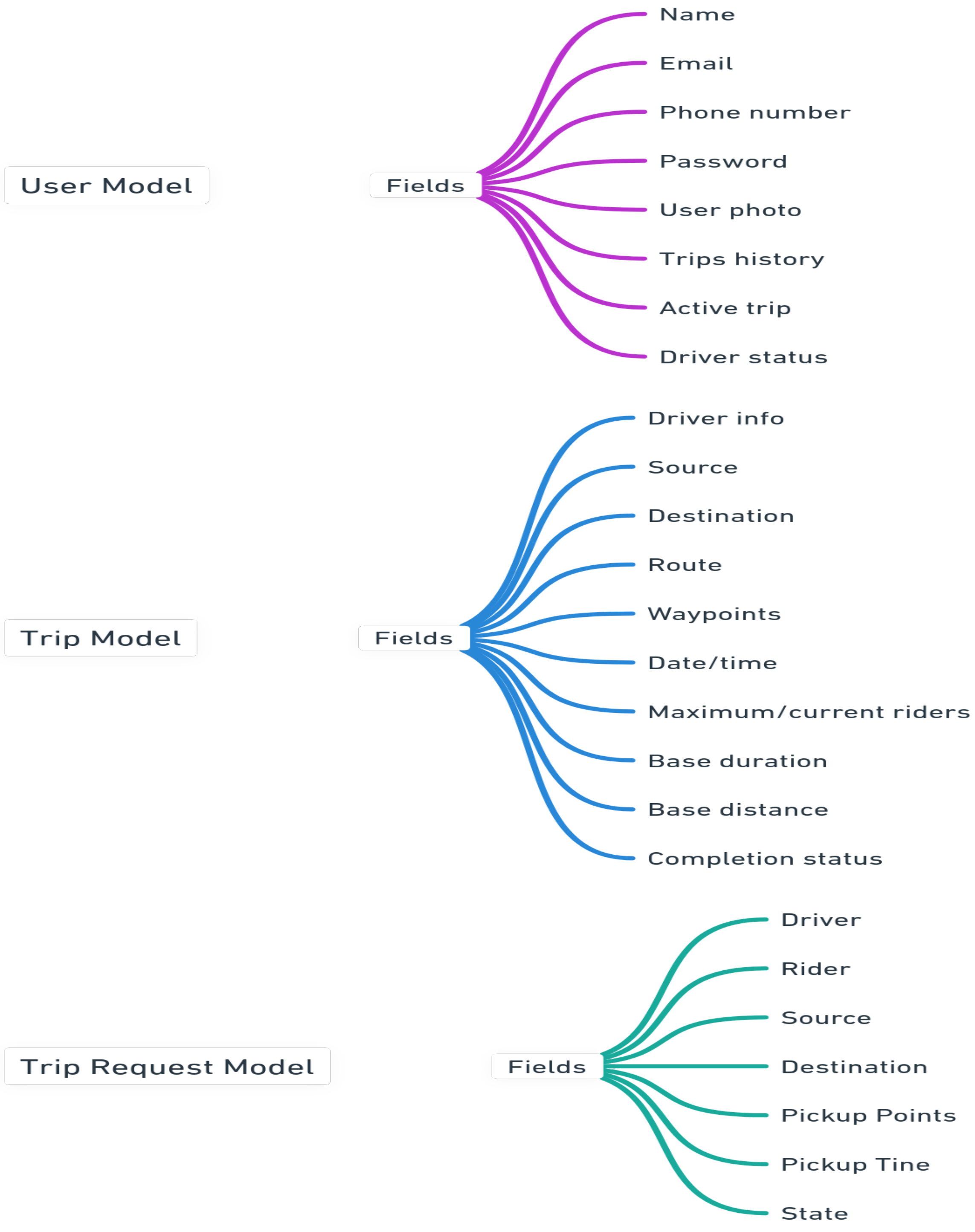
The trip model represents individual journeys within the system. It includes the following fields:

- **Driver:**The Object Id of the driver user.
- **Source**: The starting point of the trip, specified by coordinates.
- **Destination**: The final destination of the trip, specified by coordinates.
- **Route**: The planned route from the source to the destination represented as a polyline.
- **Waypoints**: Any intermediate points along the route, which may include stops or waypoints specified by the driver or system.
- **Date/time**: The scheduled date and time for the trip to begin.
- **Maximum/current riders**: Limits on the number of riders that can join the trip and the current count of riders already assigned to the trip.
- **Base duration**: The estimated duration of the trip without considering factors such as traffic or delays.
- **Base distance**: The estimated distance of the trip, calculated based on the planned route.
- **Completion status**: Indicates whether the trip has been completed, cancelled, or is still ongoing.

2.3 Trip Request Model

The trip request model represents requests made by users to join or create trips. It includes the following details:

- **Driver**: The MongoDB ObjectID of the driver associated with the trip request.
- **Source**: The coordinates specifying the starting location for the trip request.
- **Destination**: The coordinates specifying the intended destination for the trip request.
- **Pick-up Points**: Specific points where the rider requests to be picked up, if different from the source.
- **Rider**: The MongoDB ObjectID of the rider associated with the trip request.
- **Rider Name**: The name of the rider, if provided.
- **Driver Name**: The name of the driver associated with the trip request.
- **Trip**: The MongoDB ObjectID of the trip associated with the request.
- **Pick-up Time**: The requested time for pick-up.
- **State**: Indicates the current status of the trip request, which could be pending, accepted, rejected, or cancelled. Default status is "pending".


User Model
Fields
Name
Email
Phone number
Password
User photo
Trips history
Active trip
Driver status
Trip Model
Fields
Driver info
Source
Destination
Route
Waypoints
Date/time
Maximum/current riders
Base duration
Base distance
Completion status
Trip Request Model
Fields
Driver
Rider
Source
Destination
Pickup Points
Pickup Tine
State

3. Major Functions

3.1 User Authentication

- Secure sign-up, sign-in, and sign-out.
- Passwords are encrypted using unique salts.

3.2 Initiating a Drive

- Drivers schedule drives with source, destination, route, date, and maximum riders.
- Active trip is automatically assigned to the driver.

3.3 Requesting a Ride

- Riders search and view available trips based on pick-up/drop-off points, date, and time.
- Riders send ride requests to drivers.

3.4 Accepting/Rejecting Ride Requests

- Drivers view incoming ride requests and can accept or reject them.
- Trip becomes active for the rider upon acceptance.

3.5 Completing a Trip

- Drivers can mark a trip as completed, updating details and notifying riders.

3.6 Cancelling an Active Trip

- Users can cancel an active trip before the scheduled time.
- Trip details are updated accordingly.

3.7 Retrieving User and Trip Details

- Users can view their profile details, trip history, and active trip information.

3.8 Checking Driver Status

- Users can check if they have an active driver role.

3.9 User Sign Out

- Users can securely sign out, clearing authentication tokens.

4. API Routes

4.1 Authentication Routes

- POST /api/signup: User registration.
- POST /api/signin: User login.
- DELETE /api/delete: Delete user account.
- GET /api/signout: User sign-out.

4.2 Trip Routes

- POST /api/trip/drive: Initiate a drive.
- POST /api/trip/request: Request a ride.
- POST /api/trip/done: Complete a trip.
- GET /api/trip/history: Get user trip history.
- GET /api/trip/activetrip: Get user's active trip details.

4.3 User Routes

- GET /api/user/details: Get user details.
- GET /api/user/isDriver: Check if the user is a driver.

4.4 Driver Routes

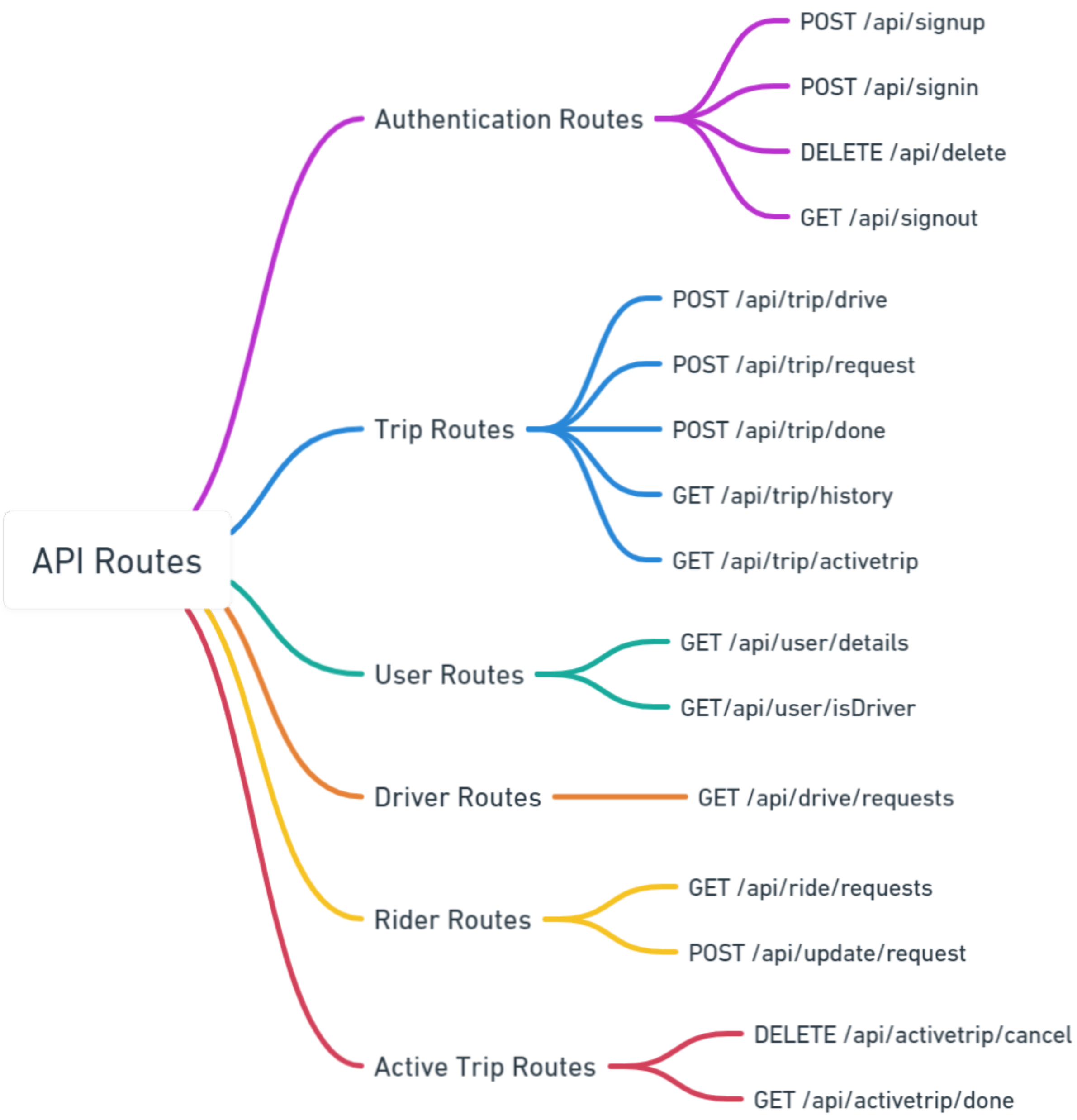
- GET /api/drive/requests: Get incoming ride requests for drivers.

4.5 Rider Routes

- GET /api/ride/requests: Get ride requests for riders.
- POST /api/update/request: Update the status of a ride request.

4.6 Active Trip Routes

- DELETE /api/activetrip/cancel: Cancel an active trip.
- GET /api/activetrip/done: Mark an active trip as done.


API Routes
Authentication Routes
POST /api/signup
POST /api/signin
DELETE /api/delete
GET /api/signout
Trip Routes
POST /api/trip/drive
POST /api/trip/request
POST /api/trip/done
GET /api/trip/history
GET /api/trip/activetrip
User Routes
GET /api/user/details
GET/api/user/isDriver
Driver Routes
GET /api/drive/requests
Rider Routes
GET /api/ride/requests
POST /api/update/request
Active Trip Routes
DELETE /api/activetrip/cancel
GET /api/activetrip/done

5. Technologies Used

5.1 Backend

- Node.js with Express for server-side development.
- MongoDB for the database.
- Mongoose for object modelling.

5.2 Frontend

- React for the user interface.
- Leaflet for map integration.

6. Additional Libraries and APIs

6.1 Geolocation

- Utilises the react-leaflet library for map integration.

6.2 Reverse Geocoding

- Implemented using the Nominatim API.

6.3 Routing

- Routing is performed using the OSRM API.

6.4 Geometrical Calculations

- Geometrical calculations are performed using the turf.js .

7.1 User Roles

7.1.1 Driver

Responsibilities:

- *Schedule Drives*: Drivers have the ability to plan and schedule their drives according to their availability and preferences. This includes specifying the date, time, and route details for upcoming trips.

- *Manage Incoming Ride Requests*: Drivers can view and manage incoming ride requests from riders. They have the authority to accept or reject ride requests based on factors such as proximity, time constraints, or personal preferences.

- *View and Manage Active Trip*: Drivers have access to information about their active trip. This includes real-time updates on the trip status, passenger information, and navigation details. They can also manage trip settings or make adjustments as needed.

7.1.2 Rider

Responsibilities:

- *Search and Request Rides*: Riders can search for available rides based on their desired route, time, and other preferences. They have the ability to request rides from available drivers, specifying pick-up and drop-off locations.

- *View and Manage Active Trips*: Riders have access to information about their active trips. This includes real-time updates on the trip status, driver information, and estimated time of arrival. They can also make adjustments to their trip settings or cancel rides if needed.

- *Access Trip History:* Riders can view their trip history, which includes details about past rides such as dates, times, routes, and driver feedback.

7.2 User Authentication

The application utilises JWT-based authentication for secure user access. JWT (JSON Web Tokens) are used to securely transmit information between parties as a JSON object. This mechanism helps to ensure that users are properly authenticated before accessing the system's features and data.

7.3 Trip Interaction

Users, both drivers and riders, can perform various interactions related to trips:

DRIVER

- *Schedule New Drives*: Drivers can schedule new drives by specifying the date, time, and route details for their upcoming trips.

- *Manage Incoming Ride Requests*: Drivers can view incoming ride requests from riders and manage them by accepting or rejecting requests based on their availability and preferences.

- *View Active Trip*: Drivers have access to information about their active trip, including real-time updates on the trip status, passenger information, and navigation details.

- *View Trip History*: Drivers can access their trip history, which includes details about past drives such as dates, times, routes, and passenger feedback.

RIDER

- *Search and Request Rides*: Riders can search for available rides based on their desired route and time preferences, and request rides from available drivers.

- *View Active Trip*: Riders have access to information about their active trips, including real-time updates on the trip status, driver information, and estimated time of arrival.

- *View Trip History*: Riders can access their trip history, which includes details about past rides such as dates, times, routes, and driver feedback.

8. Conclusion

The conclusion of the Carpooling Application emphasises its primary objective: to create a seamless experience for users seeking to share rides efficiently. This involves ensuring that the platform is both robust and responsive, catering to the needs of both drivers and riders alike.

The emphasis on seamlessness underscores the importance of a user-friendly interface and smooth interaction flow within the application. By providing a hassle-free experience, the application aims to encourage more users to adopt carpooling as a convenient and sustainable transportation option.

Additionally, the mention of a robust and responsive platform highlights the reliability and adaptability of the application. It indicates that the system is designed to handle various user interactions, manage data securely, and adapt to changing requirements or conditions effectively.

Overall, the conclusion encapsulates the core mission of the Carpooling Application: to offer a reliable, user-friendly platform that facilitates efficient ride-sharing, ultimately contributing to the promotion of sustainable transportation practices