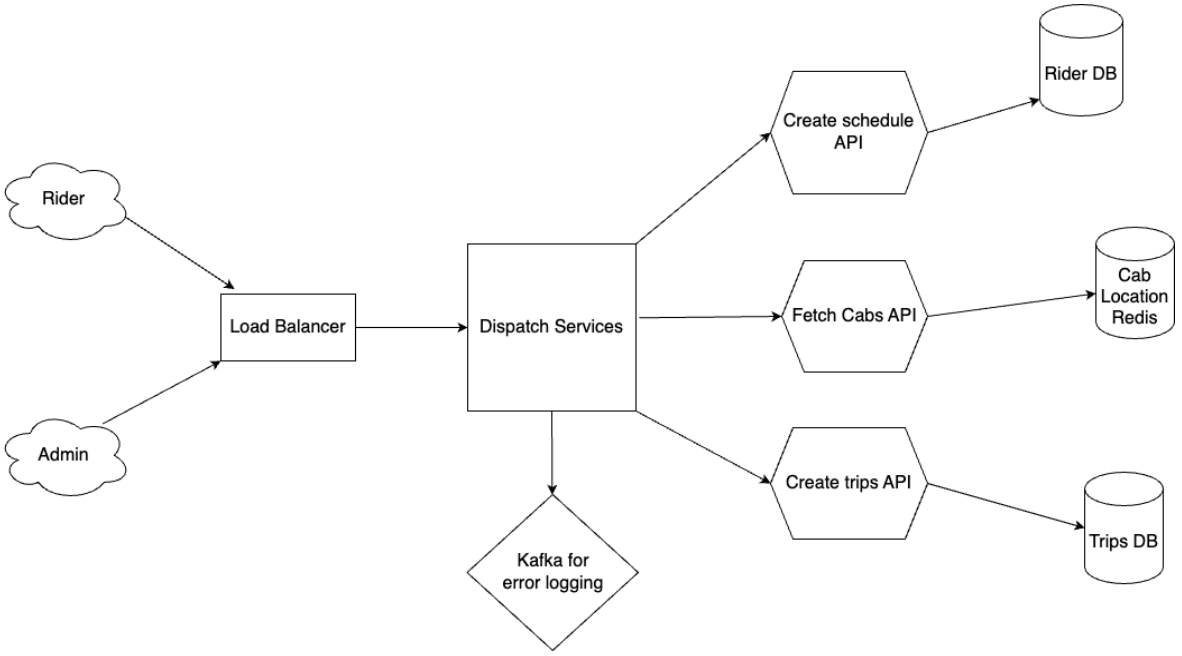
Case Study : Smart Cab Allocation System for Efficient Trip Planning

This report provides a comprehensive overview of the workflow, focusing on key features, system components, and design considerations for building a smart cab allocation system.

The Smart Cab Allocation System comprises three main modules: Cab Allocation Algorithm, Real-Time Location Tracking, and User Interface. The algorithm module (Create trips API) optimises cab allocation based on proximity and trip efficiency, while the tracking module (Fetch cabs API) ensures accurate real-time location data. The user interface provides a seamless experience for both administrators and employees, displaying relevant cab information.
The design of the application is as follows:



Frontend User Interface (UI) - Next.js:
1. Employee App:

- Login/Authentication: Secure login mechanism for employees
- Home Screen: Display nearby cabs, estimated arrival time, and other relevant information.
- Booking Interface: Allow employees to request a cab with details like destination, time, etc.
- Trip History: Show past trips, invoices, and ratings.

2. Admin Dashboard:

- Login/Authentication: Secure login for administrators.
- Cab Management: View, add, or modify cab details.
- Employee Management: Track employee trips, manage accounts, and access reports.
- Analytics: Monitor cab utilisation, popular routes, and performance metrics.

Backend Server - :
APIs:

- Authentication API: Handles user authentication for both employees and administrators - can be made secure using OAuth.

- Cab Management API: Manages cab details, availability, and status.
- Booking API: Handles trip requests, assigns cabs, and calculates fares.
- Reporting API: Generates reports and analytics for administrators.
- Geolocation API: Tracks real-time cab locations using GPS.

Database:

- Store user data (employees and admins).
- Cab information (location, availability, status).
- Trip history, including start and end locations, timestamps.

Cab Allocation Algorithm:

- A location-based algorithm to suggest the best cab for a trip.
- Prioritise available cabs based on proximity to the trip start location.
- **Advanced** - Taking factors like traffic conditions, historical trip data into account.

1. Admin's Cab Allocation Optimization Algorithm:

The cab allocation algorithm is similar to the formulation for the mTSP or m Travelling salesman problem, which can be solved using an integer linear programming problem solver.

This algorithm assumes that the schedules and locations of the employees are previously known and based on that, an optimal set of trips are planned on the admin's end.

Problem Formulation:

Consider a graph G = (V, A), where V is the set of n nodes (employees), and A is the set of edges. Associated with each edge $(i, j) \in A$ is a cost (or distance) c_{ij}. We assume that the cab depot is node 1 and there are m cabs at the depot. We define a binary variable x_{ij} for each edge $(i, j) \in A$; x_{ij} takes the value 1 if edge (i,j) is included in a tour and x_{ij} takes the value 0 otherwise. For the subtour elimination constraints, we define an integer variable u_i to denote the position of node i in a tour, and we define a value p to be the maximum number of nodes that can be visited by any cab.

Objective : Minimise $\sum_{(i,j) \in A} c_{ij} x_{ij}$

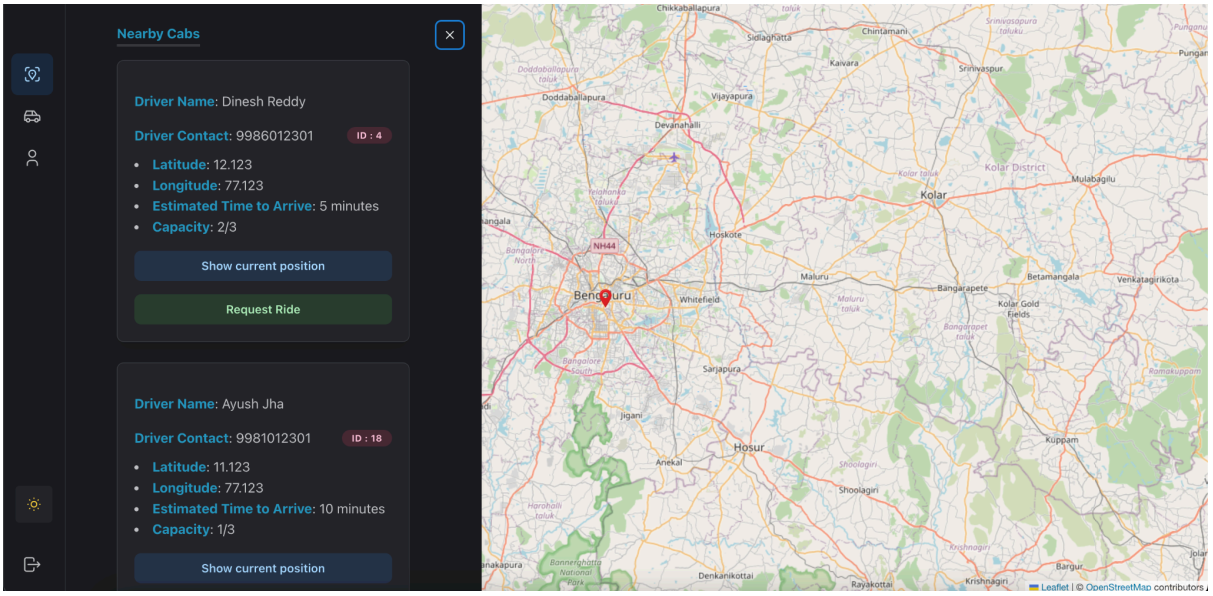
Constraints :

- Ensure that exactly m salesmen depart from node 1. $\sum_{j \in V:\,(1,j) \in A} x_{1j} = m$
- Ensure that exactly m salesmen return to node 1. $\sum_{j \in V:\,(j,1) \in A} x_{j1} = m$
- Ensure that exactly one tour enters each node. $\sum_{i \in V:\,(i,j) \in A} x_{ij} = 1, \forall j \in V$
- Ensure that exactly one tour exits each node . $\sum_{j \in V:\,(i,j) \in A} x_{ij} = 1, \forall i \in V$
- Include subtour elimination constraints $u_i - u_j + p.\,x_{ij} \leq p - 1, \forall 2 \leq i \neq j \leq n$

2. Employee's Cab Search Optimization:

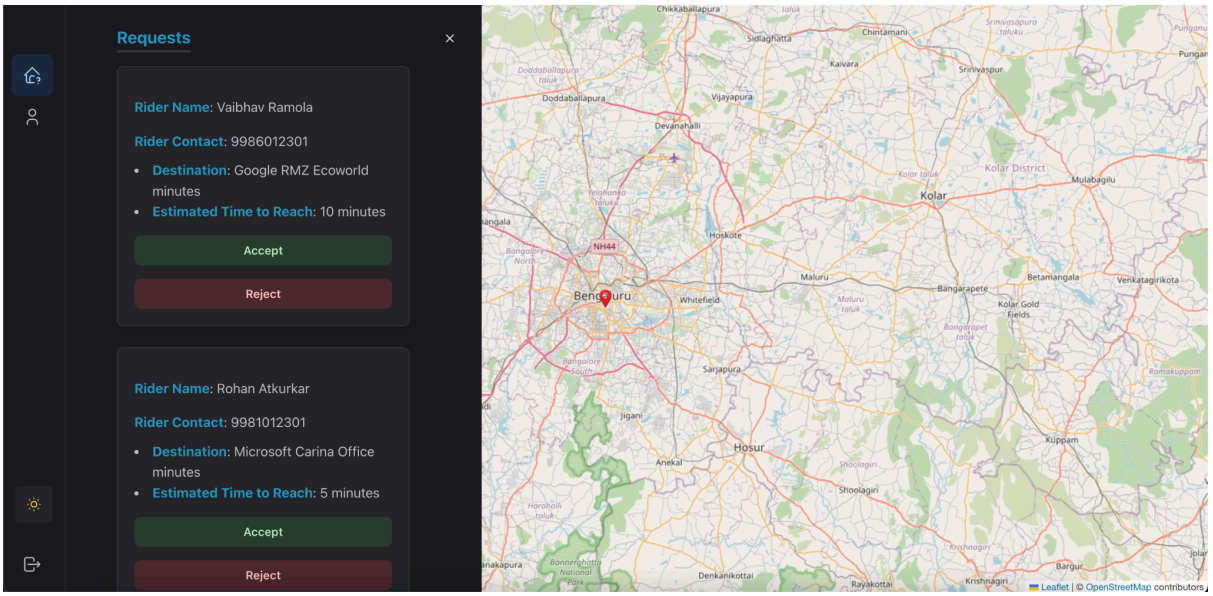
The riders can see the nearby drivers on the home screen and can view their locations, cab details. With real-time data, riders can request the ride from the interface. Riders can also see trip history and their profiles.

Rider's Dashboard -

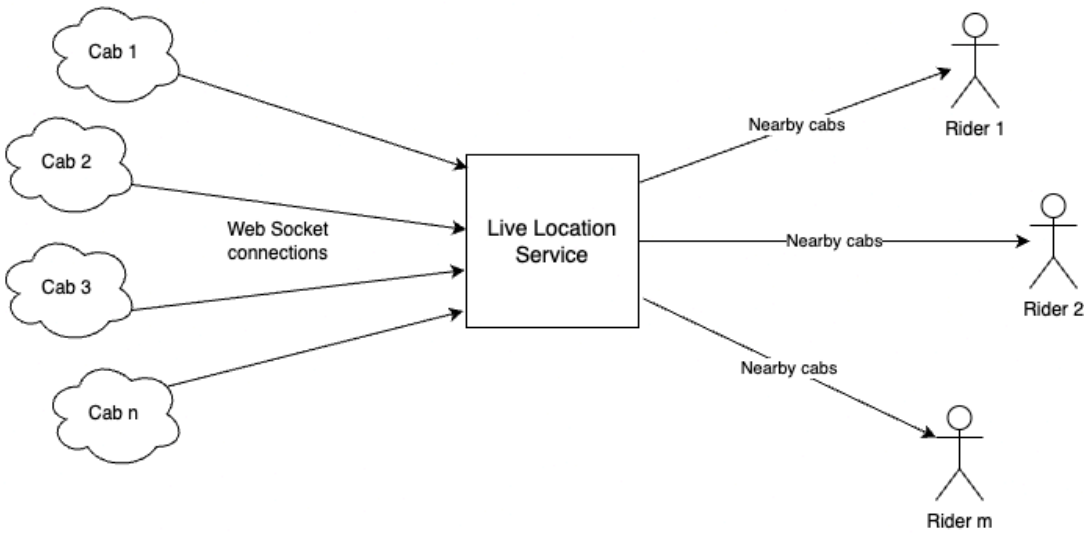


Driver's Dashboard -

Drivers can see the requested rides and can choose to accept or reject it. For the scheduled trips, the driver would get a pre-decided route which would be most efficient.



3. Real-Time Location Data Integration:



For real-time location data of cab drivers, we can use web socket connections to share the location info to the servers. For each rider, the nearby cabs can then be found using a live location service and creating a distance filter which will look for nearby cabs from the rider's location.