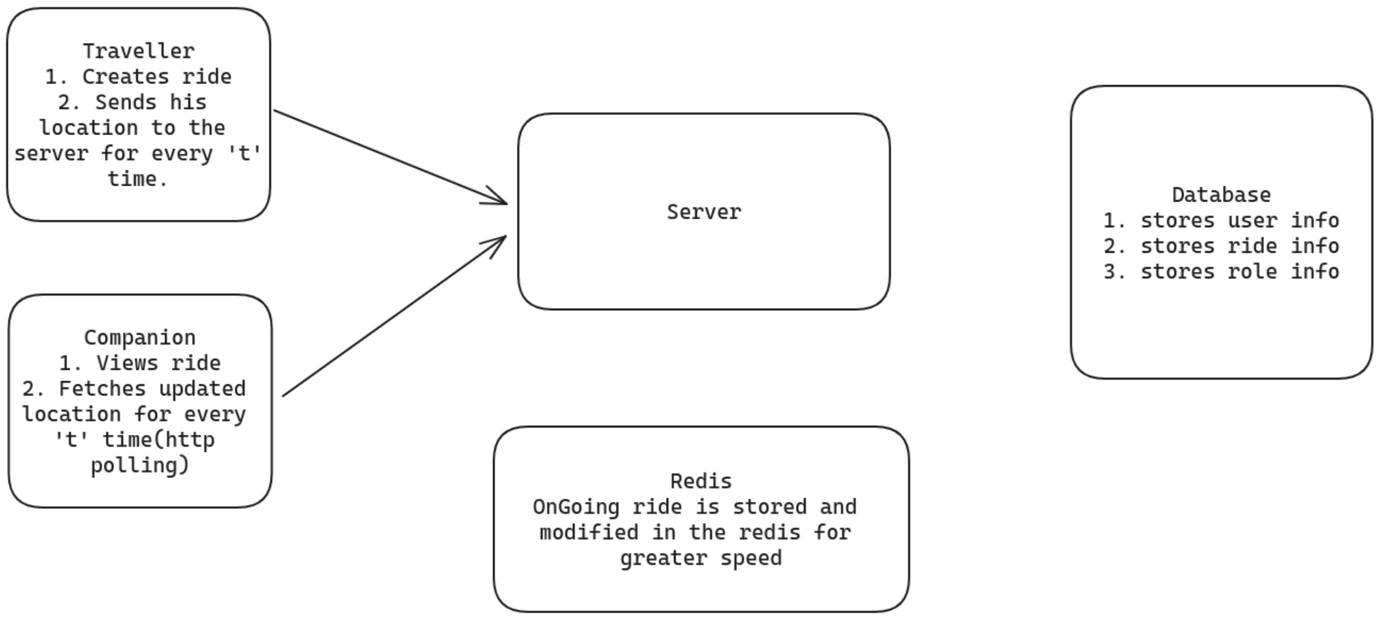
Case Study Report

By: D. Bharadwaja Rao
Email: cs20btech11012@iith.ac.in

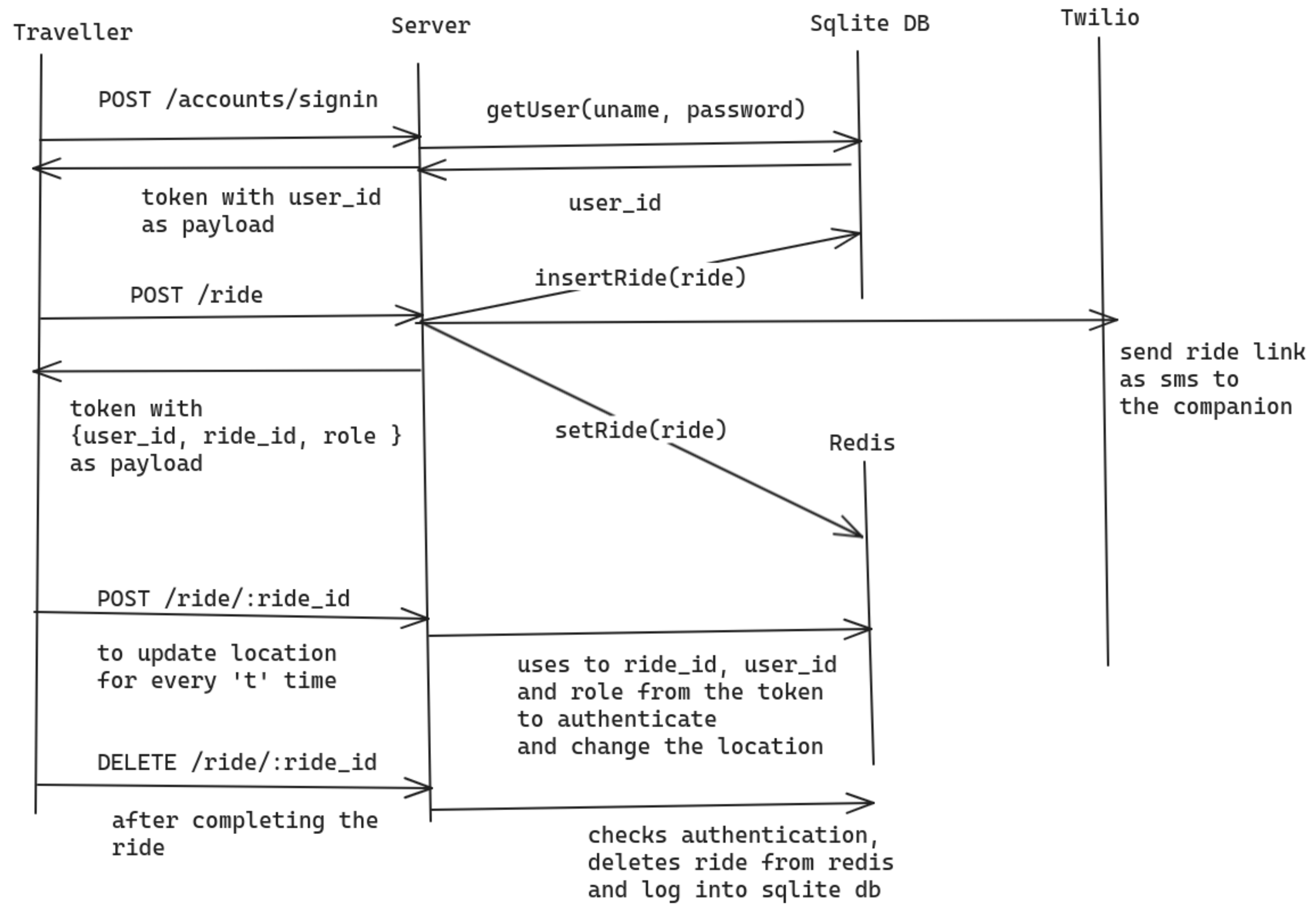
Overview of the case study:

Build a ride-sharing platform which is robust and catters to the needs of 3 distinct user roles which are traveler, companion and admin. The platform facilitates efficient ride-sharing, real-time tracking, and feedback mechanisms.
Traveler: can share link containing ride details, including Driver Name, Driver Phone Number, Cab Number along with live GPS tracking through SMS. The shared link expires automatically once the trip is complete.
Companion: Can view the live location of the traveler
Admin: Admins should have access to all the rides shared by the user and also the feedback shared by the user.
The implementation of the solution should consider various aspects like authentication for security, data reliability, code readability and maintainability, good coding practices etc.

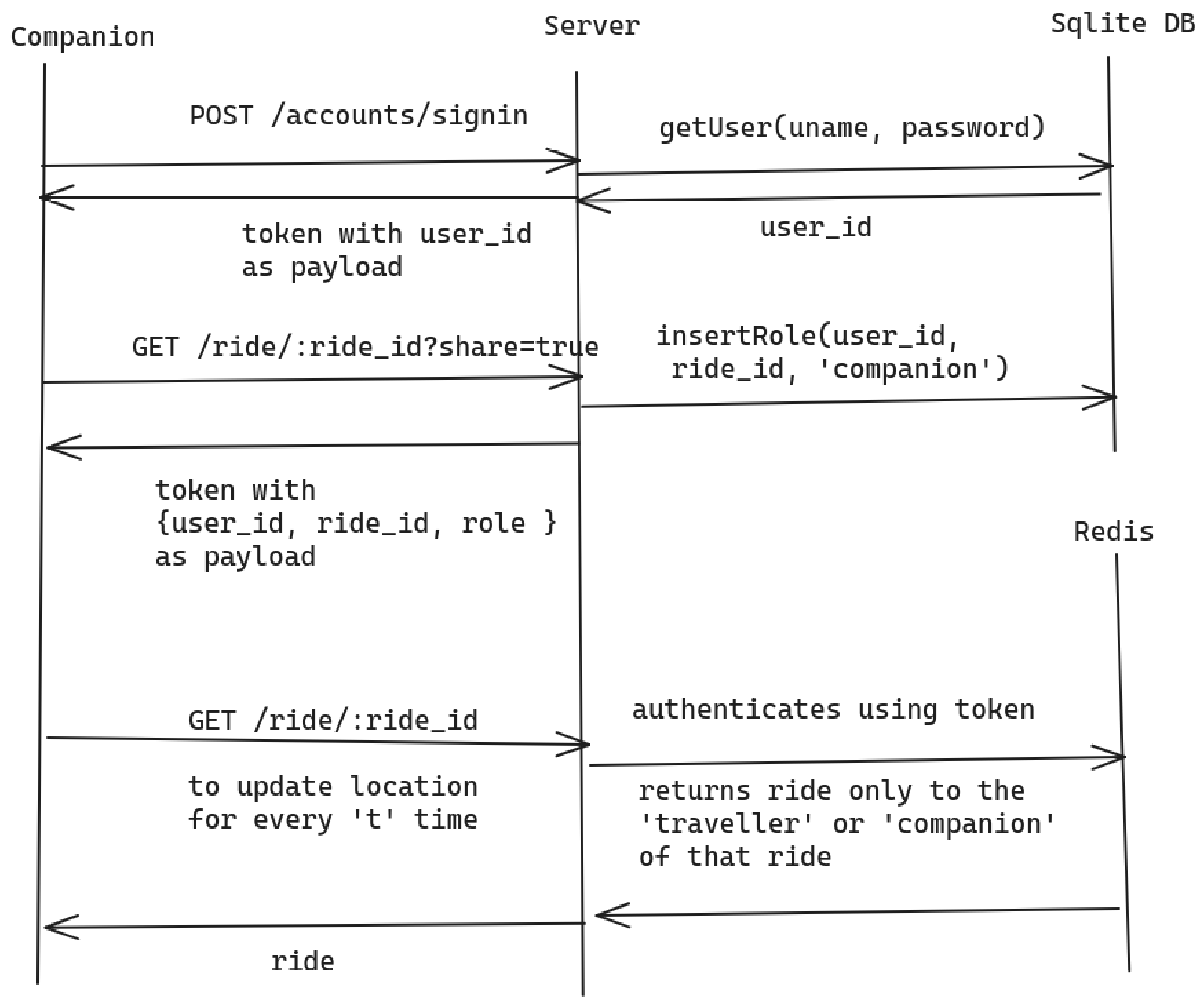
System Design and Architecture


Traveller
1. Creates ride
2. Sends his location to the server for every 't' time.
Server
Database
1. stores user info
2. stores ride info
3. stores role info
Companion
1. Views ride
2. Fetches updated location for every 't' time(http polling)
Redis
OnGoing ride is stored and modified in the redis for greater speed

High Level Implementation.



Control flow for Traveller


Companion
Server
Sqlite DB
POST /accounts/signin
getUser(uname, password)
token with user_id as payload
user_id
GET /ride/:ride_id?share=true
insertRole(user_id, ride_id, 'companion')
token with {user_id, ride_id, role } as payload
Redis
GET /ride/:ride_id
to update location for every 't' time
authenticates using token
returns ride only to the 'traveller' or 'companion' of that ride
ride

Control flow for companion

Technology Stack and Implementation Details

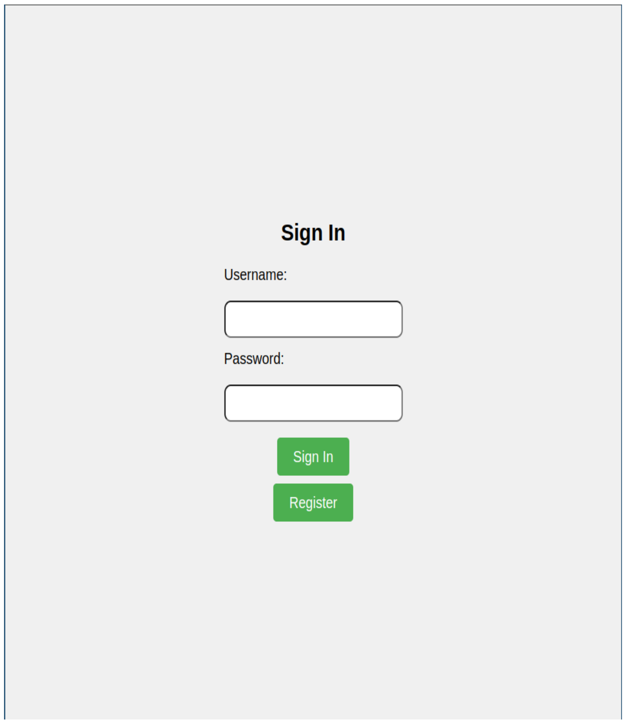
- **Frontend:** Used React to develop the user interface for the platform because of its component structure and state management.
- **Backend:** Used express JS to develop the APIs because of its simplicity, flexibility, and minimalistic approach leading to a lightweight and fast backend.
- **Messaging:** Twilio messaging services are used to send the messages.
- **DataBase:** sqlite3 is used for persisting data due to its lightweight nature, simplicity and easy to setup database(avoiding separate database server like postgres)
- **Cache:** Redis due to its fast in memory key value datastore nature.
- **Authentication:** Json Web Tokens are used for authentication due to their stateless nature and ease of integration.

- **Leaflet:** an open source map renderer is used to show the location of the traveler to the companion. https://leafletjs.com/

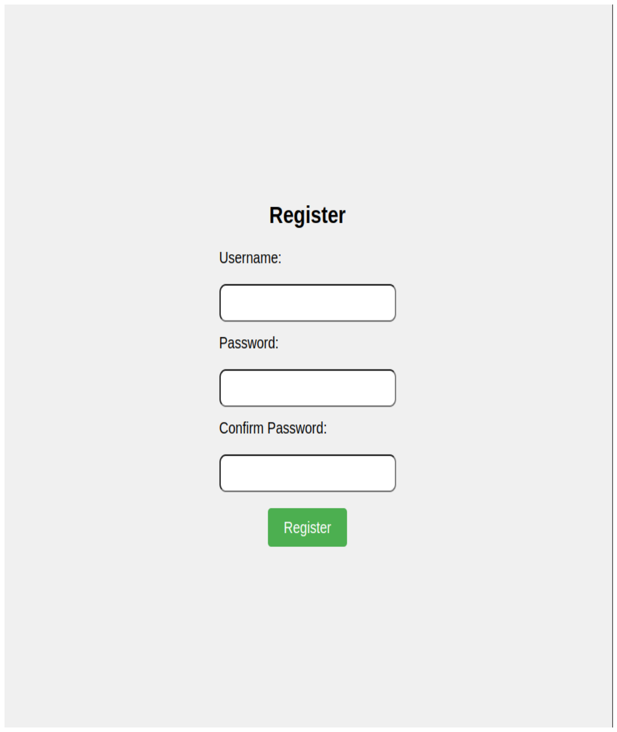
  The choice to use React and Express was influenced by past experience and the need for a quick development turnaround. React's familiarity allowed for rapid frontend development, while Express provided a lightweight and straightforward backend solution. The decision to avoid Django was driven by the desire to minimize boilerplate setup and configuration, making the stack more suitable for a project with time constraints and a specific scope.

Link to the demo video:
https://drive.google.com/file/d/15a2om9LCHa4Zy2kAIwr5xe5b95I0Racn/view?usp=sharing

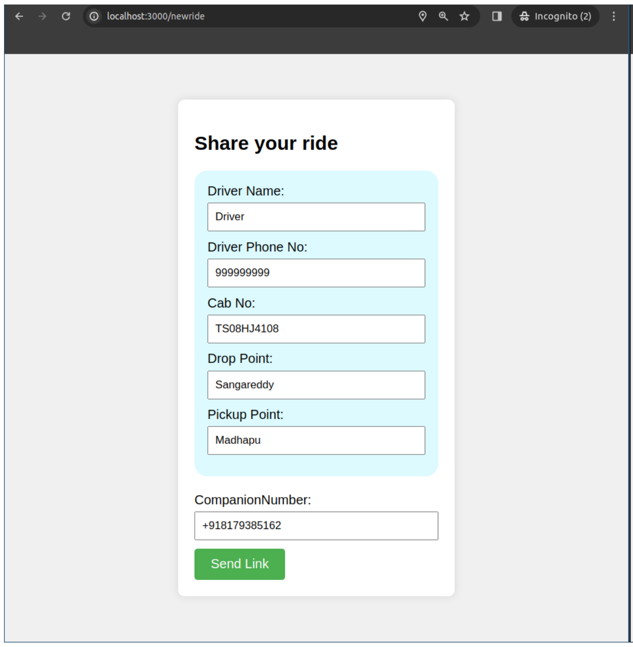
Screenshots of UI



Sigin page



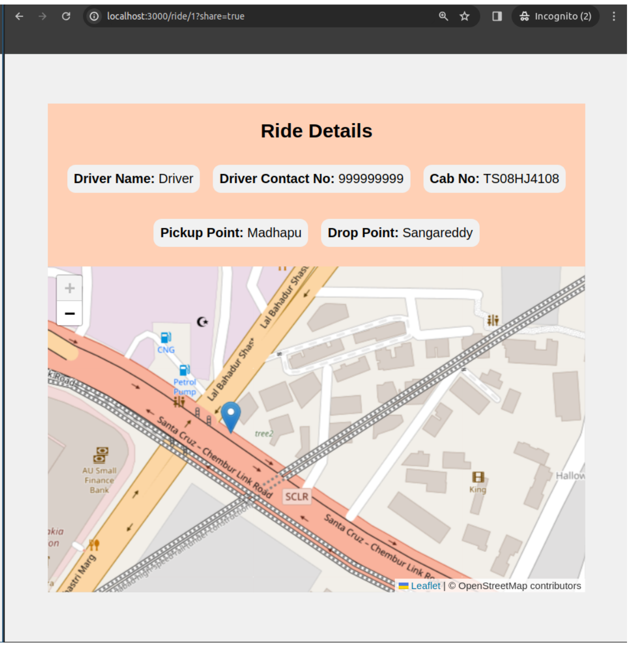
Register page



Share your ride

Driver Name:	Driver
Driver Phone No:	999999999
Cab No:	TS08HJ4108
Drop Point:	Sangareddy
Pickup Point:	Madhapu
CompanionNumber:	+918179385162

Send Link


localhost:3000/ride/1?share=true
Incognito (2)
Santa Cruz - Chembur Link Road
Lal Bahadur Shastri Marg
CNG
Petrol Pump
AU Small Finance Bank
SCLR
Leaflet | © OpenStreetMap contributors

Ride Details

Driver Name: Driver **Driver Contact No:** 999999999 **Cab No:** TS08HJ4108

Pickup Point: Madhapu **Drop Point:** Sangareddy

Traveler's view Companion view

Future plans and improvements

1. Use socket instead of http polling for companion while getting travelers' live location.
2. Use geolocation indexing (like uber h3 or google s2) to detect when the traveler reaches the geofence of the destination.
3. Sophisticated authentication system and including sign in with Google
4. Better Ride Id generation and companion authentication system for more security.
5. Adding Marker and RouteLayer for the leaflet map to get an appealing map.
6. Better error handling in frontend and overall improvement in the look of front end.