SMART RIDE APP

Group Members

1. Vijayaraghavan Sudesh Kumar - 8868141
2. Kennedy Mbano - 8826852
3. Derryck Duwuona - 8862396
4. Christina Tresa Abraham - 8875231

PROG8750 and Capstone (Web Development)

Semester 4

Submitted to

Pankaj Bains

Table of Contents

Introduction

The "***Smart Ride App***" represents an ambitious initiative aimed at transforming the ride-sharing landscape by introducing a sophisticated and user-centric mobile application. This proposal outlines the project's core functionalities, emphasizing its commitment to user-friendly interactions, robust security measures through ID verification, and an efficient platform for ride posting and searching.

The motivation behind the Smart Ride App project lies in addressing the evolving transportation needs of users. By providing a seamless and secure platform, the app seeks to offer users a hassle-free and dependable ride-sharing experience. The benefits of this project are outlined below:

User-Friendly Experience: The Smart Ride App prioritizes simplicity in its registration processes, ensuring that users can easily sign up and navigate the application, making the overall experience enjoyable.

Enhanced Safety Measures: The implementation of stringent ID verification procedures adds an extra layer of security, fostering a sense of trust within the Smart Ride App community. This measure ensures that users can confidently connect with verified individuals for rides.

Efficient Ride Management: The app streamlines the process of posting and searching for rides, allowing both drivers and riders to interact seamlessly. The negotiation feature for ride prices adds a dynamic element, empowering users to agree on fair terms for their journeys.

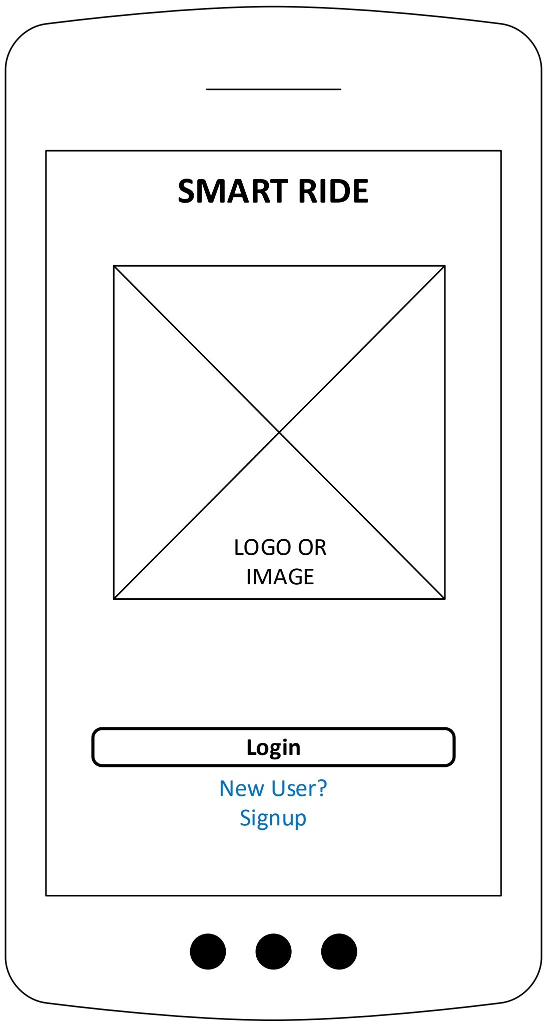
Connected Community: By fostering a reliable and connected community of drivers and riders, the Smart Ride App aims to create an environment where users can confidently participate in ride-sharing, knowing that their safety and convenience are paramount.

Design and Data Flow

This section illustrates the Design, Page flows and flow data charts for the features associated with the application.

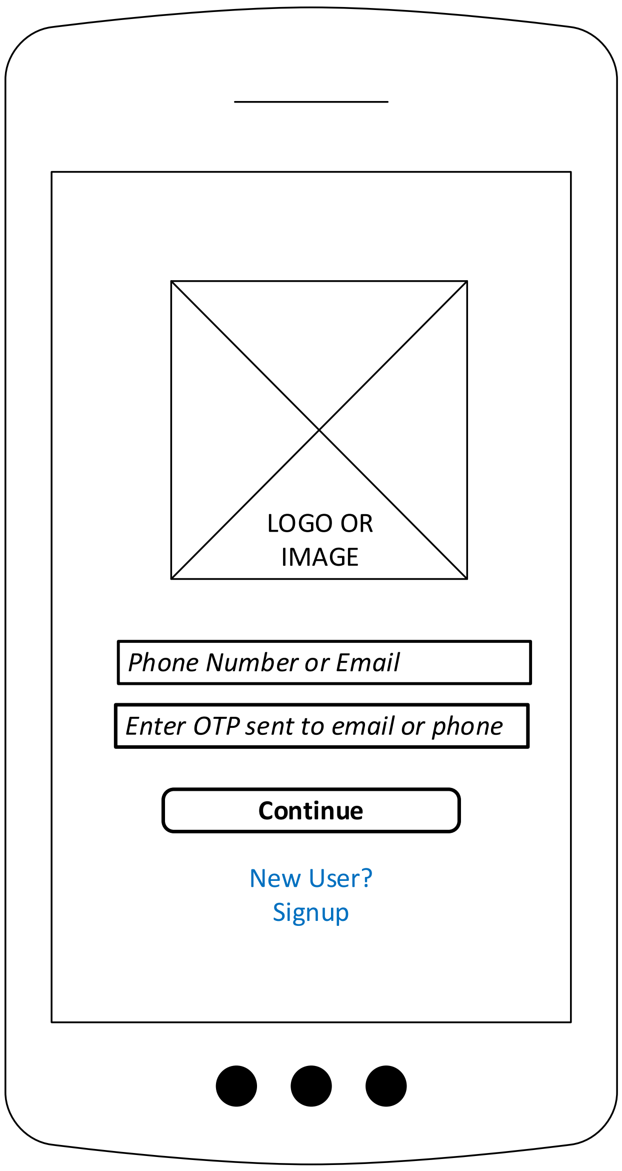
Page Flow

I. HOME SCREEN



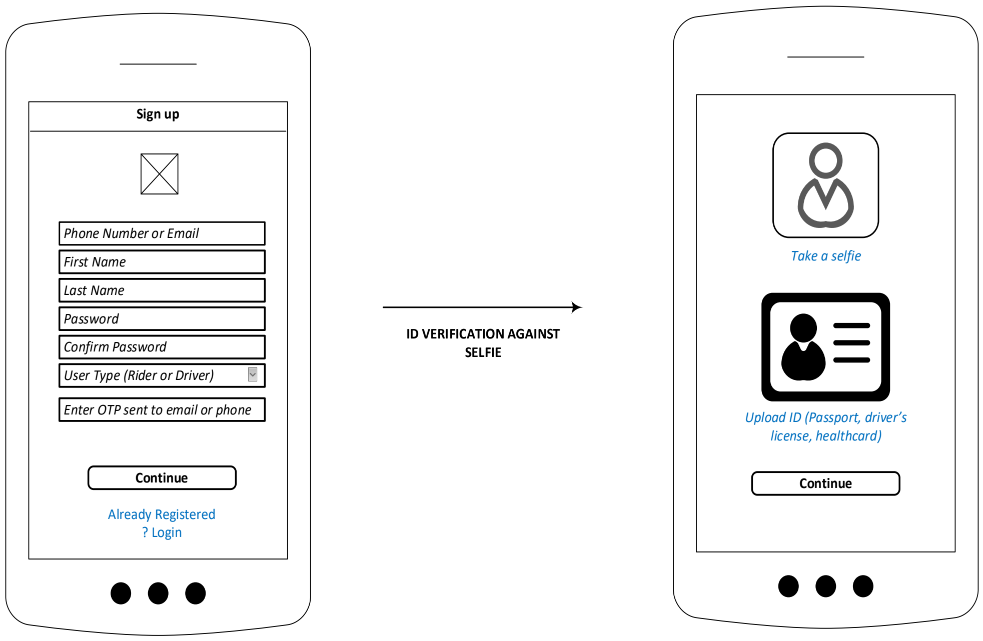
This will serve as the initial screen, functioning as the home interface, featuring buttons and links that guide users to either log in to the application or initiate the registration process as a new user.

II. LOGIN



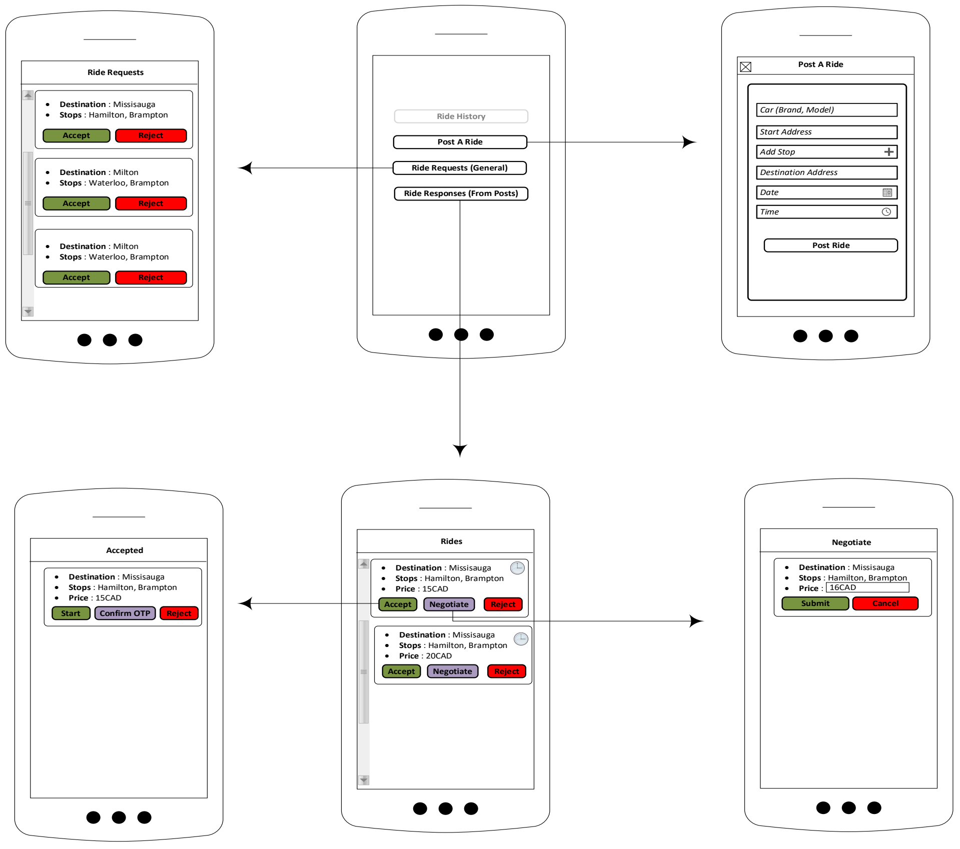
On the login page, users undergo authentication via OTP verification following the submission of their email/phone number and password.

III. SIGN UP


Sign up
Phone Number or Email
First Name
Last Name
Password
Confirm Password
User Type (Rider or Driver)
Enter OTP sent to email or phone
Continue
Already Registered ? Login
ID VERIFICATION AGAINST SELFIE
Take a selfie
Upload ID (Passport, driver's license, healthcard)
Continue

On the Signup page, users are prompted to input their details, as illustrated above, and indicate their role as either a rider or a driver. Following this, users are instructed to capture a selfie and upload any form of identification, such as a Driver's license, passport, health card, etc.

IV. Main Menu (Driver)



The Driver has 3 menu Options.

a. Post A Ride

b. Ride Requests (General)

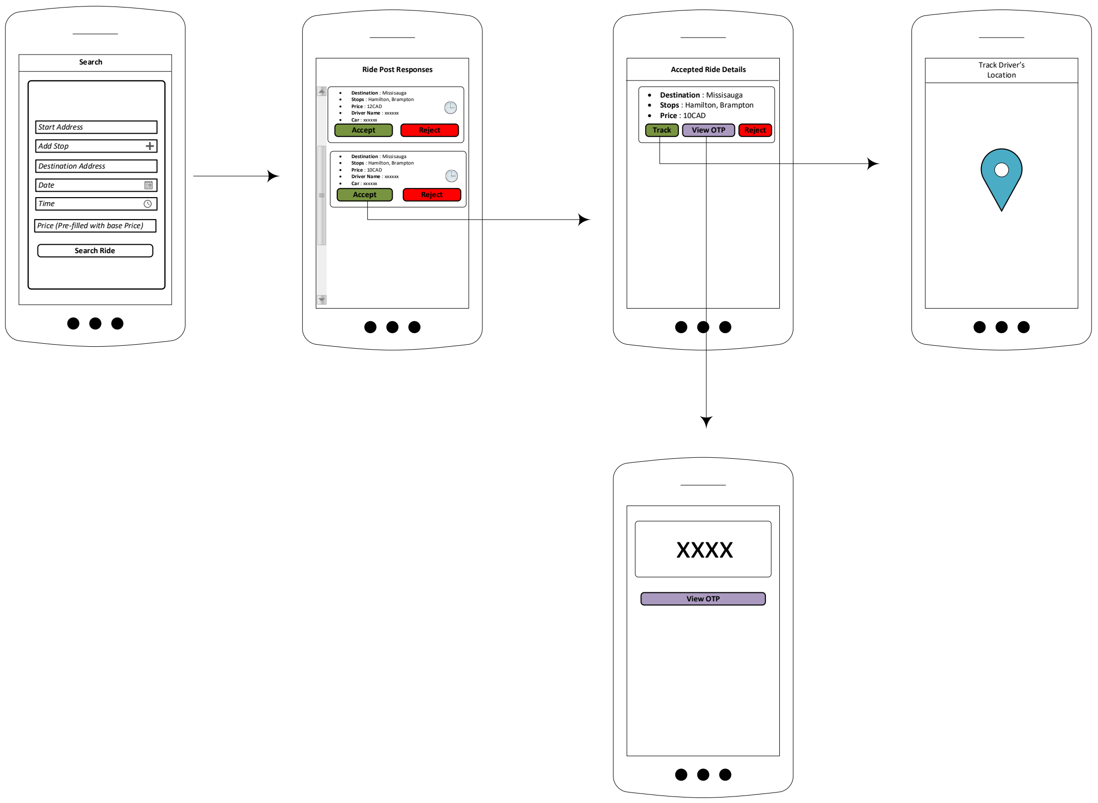
c. Ride Responses (From Posted Ride)

For the "***Post a Ride***" feature, drivers can publish details about their journey to a specific destination, indicating a base price open for negotiation. Riders in proximity receive notifications, giving them the option to either accept the ride or engage in a negotiation process.

Regarding the "***Ride Requests***" feature, the driver receives notifications of ride requests from nearby riders. The driver has the option to either accept the ride or decline the request based on their preference.

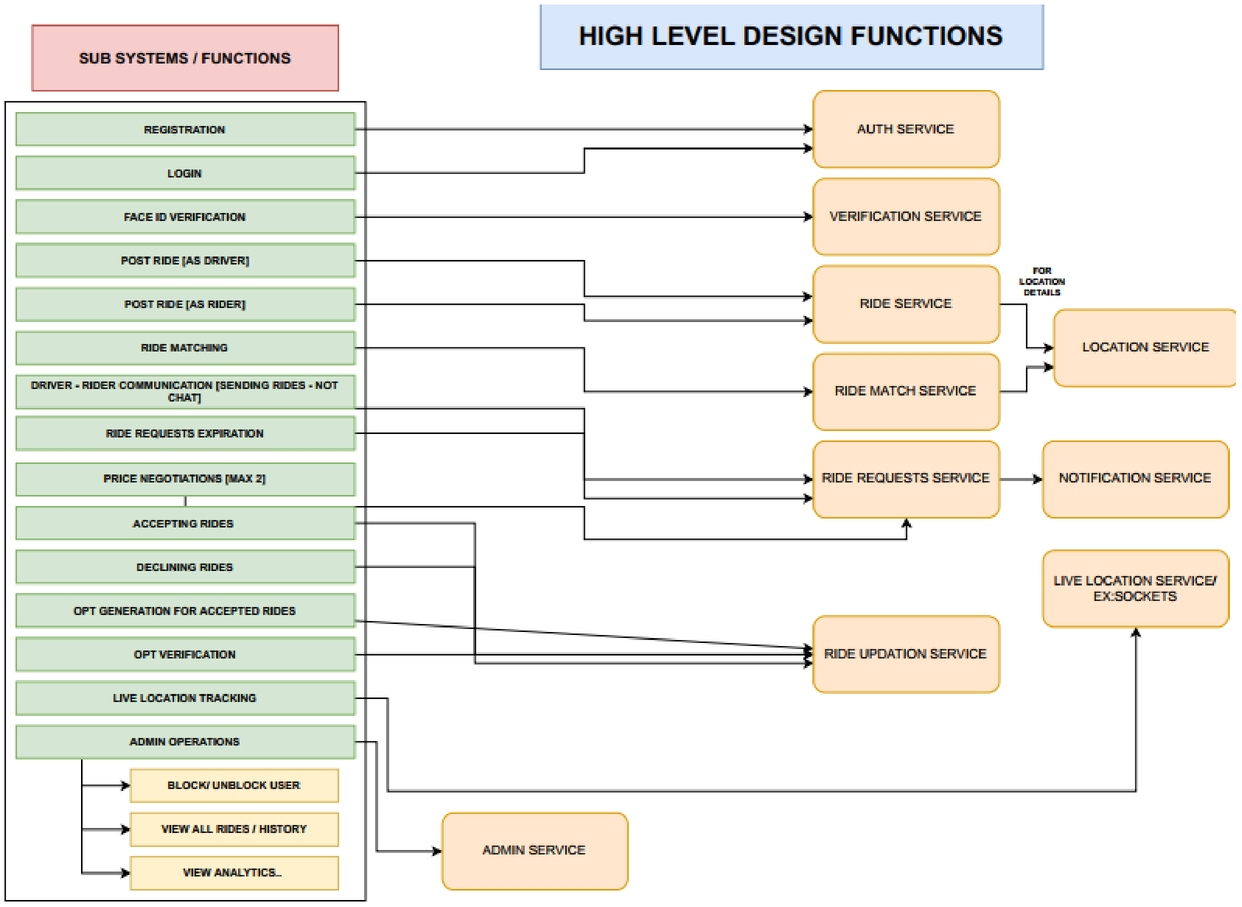
Concerning the "***Ride Responses***" feature, the driver receives real-time push notifications for posted rides (from the "Post a Ride" menu option). The driver can then choose to accept the ride, engage in a one-time price negotiation, or decline the ride.

V. Main Menu (Rider)



In the "Request a Ride" feature, the rider can provide specific details such as destination, date, negotiated price, and any necessary stops. The request undergoes evaluation for a match with available drivers, allowing the user to accept or decline the ride. Upon acceptance, the user can track the driver's location and confirm an OTP upon arrival to ensure verification.

High Level Component Design



The visual representation above depicts the architecture of the Smart-Ride Application. Further details regarding the specific roles of each service component are outlined below:

Auth Service: This service is responsible for user registration and login. It ensures that users are authenticated before they can access the application. This could involve checking the user's credentials against a database and generating a token that the user can use to authenticate subsequent requests.

Verification Service: This service manages Face ID Verification. It ensures that the user is who they claim to be by comparing their face with the one on file. This adds an extra layer of security to the application.

Ride Service: This service handles the posting of rides by drivers and riders. It could involve storing ride details in a database and making them available for matching.

Ride Match Service: This service is responsible for matching riders with drivers. It would involve running an algorithm that finds the best match based on factors like location, destination, and time.

Ride Requests Service: This service manages various aspects of ride requests. It handles communication between drivers and riders (not chat), expiration of ride requests, price negotiations (up to a maximum of 2 rounds), accepting rides, queuing rides, and OTP (One-Time Password) generation and verification for accepted rides.

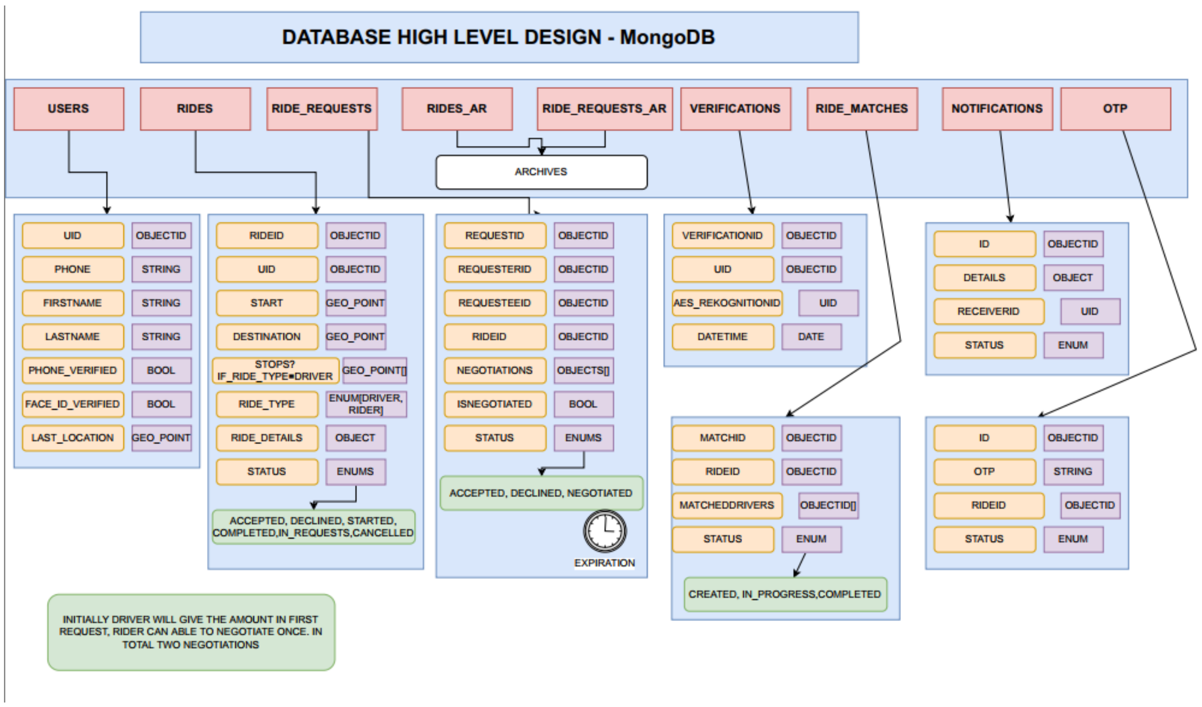
Notification Service: This service handles notifications related to the ride service. It could involve sending push notifications or emails to users about ride updates.

Live Location Service (***E.g: Sockets or SignalR***): This service tracks the live location of drivers and riders. It would involve using GPS data from the user's device and updating the location in real-time.

Admin Service: This service handles various admin operations. It allows admins to block or unblock users, view the history of all rides, and view analytics related to rides.

Each of these services is designed to be independent and modular, which means that they can be developed, updated, and scaled independently of each other. This architecture allows for flexibility and scalability, making it easier to maintain and expand the application over time. It also improves the reliability of the application, as issues in one service do not affect the others.

High Level Database Design



The design is structured into seven main collections:

USERS: This collection contains user information including unique ID (UID), phone number (PHONE), first name (FIRSTNAME), last name (LASTNAME), whether the phone number is verified (PHONE_VERIFIED) and face ID verified (FACE_ID_VERIFIED), and last location (LAST_LOCATION).

RIDES: This collection holds data related to rides such as UID of the driver or rider, start point and destination of the ride with geolocation coordinates embedded within it.

RIDE_REQUESTS: This collection stores information about ride requests including request ID (REQID), UID of requester and requested person along with status which can be accepted, declined or negotiated.

RIDES_AR and RIDE_REQUESTS_AR: These are archives for rides and ride requests respectively.

VERIFICATIONS: This collection contains verification details like UID of the person being verified along with AES recognition data and date/time of verification.

RIDE_MATCHES: This collection holds data on matched rides including match ID(MATCHID) , UIDs of riders involved in the match along with their status which can be created , in progress or completed

NOTIFICATIONS: This section includes notification details like notification ID(ID) , details(DETAILS) , receiver ID(RECEIVERID) along with their status(STATUS)

OTP: It includes OTP details like OTP id (ID) , OTP(OTP) , receiver id(RECEIVERID) along with their status(STATUS)

Each of these collections serves a specific purpose within the application, and together they form a comprehensive database system for managing the various aspects of a ride-sharing service. This design ensures that all necessary data is stored efficiently and can be accessed quickly, providing a smooth and seamless user experience.

Chosen Technology Stack

Flutter for Mobile Development:

Rationale: Flutter offers a single codebase for both Android and iOS platforms, reducing development time and ensuring consistency across devices. Its rich set of pre-designed widgets, hot reload feature, and excellent performance make it an ideal choice for creating a responsive and visually appealing mobile application.

React for Admin Web App:

Rationale: React's component-based architecture simplifies the development of dynamic and interactive user interfaces. It facilitates the creation of a responsive admin web app with real-time updates, providing administrators with an efficient tool for managing the Smart Ride App.

NestJS for Backend Services:

Rationale: NestJS, built on top of Node.js, combines the benefits of TypeScript with a modular and scalable architecture. Its use of decorators, dependency injection, and support for WebSocket communication align well with the requirements of real-time features, ensuring smooth communication between the frontend and backend. NestJS also integrates seamlessly with MongoDB.

MongoDB for the Database:

Rationale: MongoDB, a NoSQL database, is chosen for its flexibility and scalability. Its document-oriented nature allows for the efficient storage and retrieval of data related to users, rides, requests, and more. MongoDB's ability to handle large amounts of unstructured data aligns with the dynamic and diverse nature of the Smart Ride App.

Conclusion

In conclusion, the "Smart Ride App" represents a comprehensive solution to address the evolving transportation needs of users, offering a user-friendly experience, enhanced safety measures, efficient ride management, and a connected community. By prioritizing simplicity in registration processes, implementing stringent ID verification, and streamlining ride interactions, the Smart Ride App aims to revolutionize the ride-sharing landscape.

The high-level component design emphasizes modularity and independence, ensuring each service can be developed and scaled independently. This architectural approach enhances flexibility, scalability, and reliability, crucial for the long-term success of the application.

The chosen technology stack, including Flutter for mobile development, react for the admin web app, NestJS for backend services, and MongoDB for the database, is carefully selected to optimize development efficiency, real-time communication, and data storage scalability.

Looking ahead, the project completion timeline is estimated to be within 3 months and 2 weeks, considering the complexity of implementing features such as user authentication, ID verification, ride posting and matching, and real-time communication. This timeline is subject to adjustments based on the development progress and any unforeseen challenges encountered during the implementation phase.

We are committed to delivering a robust and user-centric Smart Ride App, and the GitHub link to our project will be provided for transparent collaboration and review. The link will serve as a platform for stakeholders to track our progress, provide feedback, and stay informed about the latest developments in the project.

We appreciate the opportunity to work on this transformative project and are excited about the positive impact the Smart Ride App can have on the ride-sharing community. With a focus on innovation, security, and user satisfaction, we look forward to bringing this ambitious initiative to fruition.

GitHub Link

https://github.com/KennedyGIT/Smart_Ride