Capital University of Science and Technology, Islamabad

Name: MOHAMMAD KHIZAR ALAM (BSE223202)

Section No: 3

Course: SOFTWARE ARCHITECTURE &DESIGN

FINAL REPORT

Submitted To: Ms. SABAHAT ASAD

Table of Contents

Ride Sharing Carpool

1.1 Description

The Ride Sharing Carpool Appconnects drivers and passengers traveling along similar routes. It allows users to either offer or request rides, making travel more affordable, efficient, and environmentally friendly. The app leverages GPS tracking to match riders and drivers based on proximity and destination, while providing secure payment options, real-time chat, and a robust rating system to ensure a reliable and safe experience for all users.

Main Functionalities

1. **User Registration and Profiles**
    - Sign up or log in via email, phone, or social media.
    - View and edit profile details, travel preferences, and user history.
2. **Driver and Rider Modes**
    - Allow users to switch roles between a driver (offering rides) and a rider (requesting rides).
3. **Location-Based Matching**
    - Real-time GPS integration for location tracking and mapping.
    - Ride-matching algorithm to connect users with similar routes and schedules.
4. **Ride Scheduling and Booking**
    - Schedule rides in advance or book immediately for near-instant matching.
    - Display estimated trip details, including distance, route, and fare.
5. **In-App Chat and Communication**
    - Real-time messaging between drivers and riders to coordinate pickup locations and other details.
6. **Payment Integration**
    - Secure payment gateway for in-app transactions.
    - Multiple payment methods, such as credit cards, digital wallets, and PayPal.
7. **Rating and Review System**
    - Post-ride ratings and reviews for drivers and riders to ensure quality and trustworthiness.

8. **Push Notifications**
    - Notifications for booking confirmations, ride status updates, and cancellations.
9. **Trip History and Receipts**
    - View past trips, including details, payment receipts, and ratings.

1.2 4+1 VIEW

1. Logical View

The Logical View focuses on the core functionality of the app and how its components interact with each other. Imagine the app divided into several main parts:

- User Management: This is where users sign up, create their profile, and log in. It also handles user preferences (like favourite routes or payment options).
- Ride Matching: This part is all about connecting riders with drivers. When a rider requests a ride, this module looks for available drivers nearby who are heading in the same direction.
- Booking: Once a match is made, the booking module handles the details—confirming the ride, setting pickup times, and generating ride IDs.
- Payment: After the ride is completed, this module makes sure that the payment is processed smoothly and securely through services like Stripe or PayPal.
- Chat: Riders and drivers can communicate via in-app chat to finalize details like exact pickup locations or timing.
- Notifications: Whether it's a ride request, ride acceptance, or a last-minute change, this module makes sure users are always in the loop with real-time push notifications.
- Ratings and Reviews: After the ride ends, users can rate each other based on their experience to maintain trust and quality within the platform.

These modules work together to create a seamless experience. For example, when a rider books a ride, the system checks available drivers through the Ride Matching module, confirms the booking, and sends notifications to both the rider and driver.

2. Development View

In the Development View, we break down how the app is structured in terms of code and organization.

- The Frontend is built with React Native, which allows you to deploy the app on both Android and iOS with a single codebase. It handles all the user interface and user interactions, like displaying ride requests, showing available drivers, and managing payments.

- On the Backend, we're using Node.js with Express, which acts as the server and handles all API requests between the frontend and the database. This layer manages things like booking a ride, matching riders with drivers, and storing user data.

  Inside the backend, we have a Controller Layer to handle incoming requests and responses, a Service Layer that contains business logic (like matching routes and calculating fares), and a Database Layer where all user data, ride history, and payment details are stored securely.

- External APIs like Google Maps for location tracking, Stripe for payments, and Firebase for notifications are integrated into the backend to handle specific tasks.

3. Process View

The Process View describes these interactions and ensures that all modules talk to each other in the right sequence to deliver a smooth user experience.

- When a rider requests a ride, the request is sent to the backend. The system uses the Ride Matching module to search for nearby drivers and returns a match.
- Once a driver accepts the ride, the system creates a booking and sends notifications to both the driver and rider through the Notifications Module.
- The Payment Module takes over when the ride is completed. It processes the payment through an integrated service like Stripe, and both the rider and driver are notified once the transaction is complete.
- In the background, the system is constantly running checks to ensure that everything works smoothly, from ensuring accurate location data from Google Maps to delivering notifications in real time via Firebase Cloud Messaging.

4. Physical View

The Physical View shows how everything is deployed on physical hardware and cloud infrastructure.

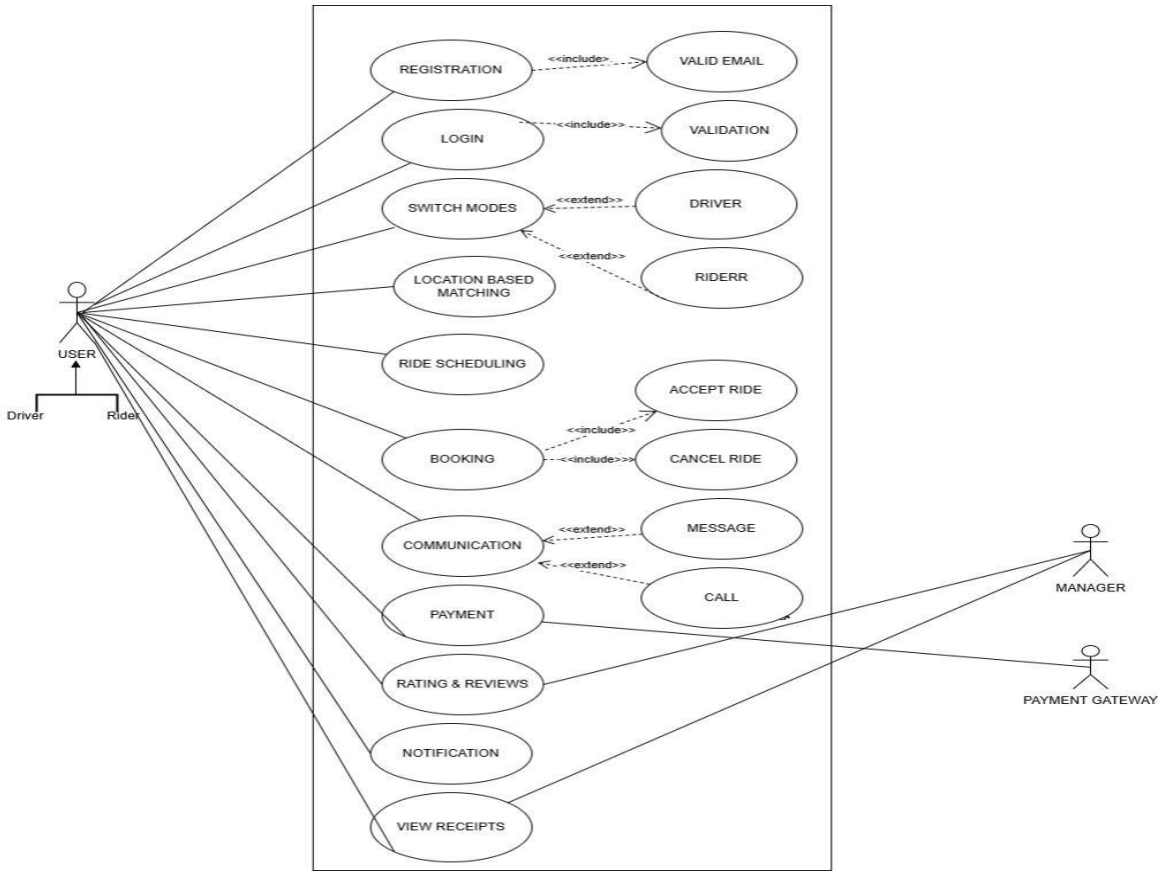
- The Frontend (the mobile app) is deployed on Android and iOS devices, which users can download from their respective app stores.
- The Backend is hosted on cloud service i.e Google Cloud, which handle all the server-side logic and database operations. This makes sure the app can scale as the number of users grows.
- The Database, MongoDB Atlas, is used to store user data and ride-related information in the cloud, which means it's accessible from anywhere and is easy to manage.
- External services, such as Google Maps for location tracking and Stripe for processing payments, are integrated to handle specific tasks but are hosted outside your app.

5. Scenarios (Use-Case View)

Scenario 1: Booking a Ride

1. User opens the app and logs in or signs up.
2. The Rider enters pickup and destination locations and requests a ride.
3. The Ride Matching module checks the availability of drivers nearby.
4. A driver is found and receives a ride request notification.
5. The driver accepts the ride, and both users receive confirmation via a push notification.
6. The ride is completed, and payment is processed through the Payment Module.
7. After the ride, the Rider and Driver rate each other, and feedback is stored for future reference.

USE CASE DIAGRAM



2.1 Fully Dressed Use case:

Q1 : Use Case: Registration

Field	Details
Use Case ID	UC-001
Use Case Name	Register on the Platform
Created By	Khizar Alam
Last Updated By	Abdul Rehman
Date Created	1-6-2024
Last Revision Date	8-12-2024
Actors	User
Description	This use case allows a customer to register on the platform.
Trigger	The user wants to register to access the platform.
Preconditions	The user has downloaded and installed the application.
Postconditions	The user successfully registers on the platform.

Normal Flow:

Step	Actor Action	System Response
1	Opens or launches the app.	Responds with an interface for registration.
2	Enters valid signup credentials (email and password).	Verifies the credentials and checks the validity of the email.
3	Successfully registers.	Confirms registration and grants access to the user.

Alternative Flows:

Scenario	System Response
Incorrect Password or Email	Displays an error message and prompts the user to re-enter correct information.

Exceptions:

Scenario	System Response
User already has an account	Directs the user to log in instead of registering.
Incorrect email or password repeatedly entered	Notifies the user to verify their information or recover the account.

2.2 Fully Dressed Use Case: Login

Field	Details
Use Case ID	UC-002
Use Case Name	Login to the Platform
Actors	User
Description	This use case allows the customer to log into the system to use its services.
Trigger	The user wants to access the platform after registration.
Preconditions	The user has successfully registered on the platform.
Postconditions	The user successfully logs into the platform.

Normal Flow:

Step	Actor Action	System Response
1	Opens the app and navigates to the login page.	Responds and asks for login credentials.
2	Enters valid credentials (username and password).	Verifies the provided information.
3	Logs in successfully.	Grants access and displays the home page.

Alternative Flows:

Scenario	System Response
Invalid Credentials	Displays an error message prompting for re-entry.
Unregistered User	Suggests the registration process.

Exceptions:

Scenario	System Response
Wrong Password	Alerts the user and allows password recovery.
Forgotten Password	Enables the password recovery option to reset credentials.

2.3 Fully Dressed Use Case: Rate and Review Drivers

Field	Details
Use Case ID	UC-003
Use Case Name	Rate and Review Drivers
Actors	User or Rider

Description	This use case enables users to check reviews and add their own ratings about drivers.
Trigger	The user wants to know the driver’s reputation or provide feedback after a ride.
Preconditions	The user has access to the review section.
Postconditions	The user successfully views or submits a review.

Normal Flow:

Step	Actor Action	System Response
1	Navigates to the review page.	Displays the list of existing reviews.
2	Clicks on the "Add Review" option.	Displays the input field for adding a review.
3	Submits the review.	Saves the review and updates the review list.

Alternative Flows:

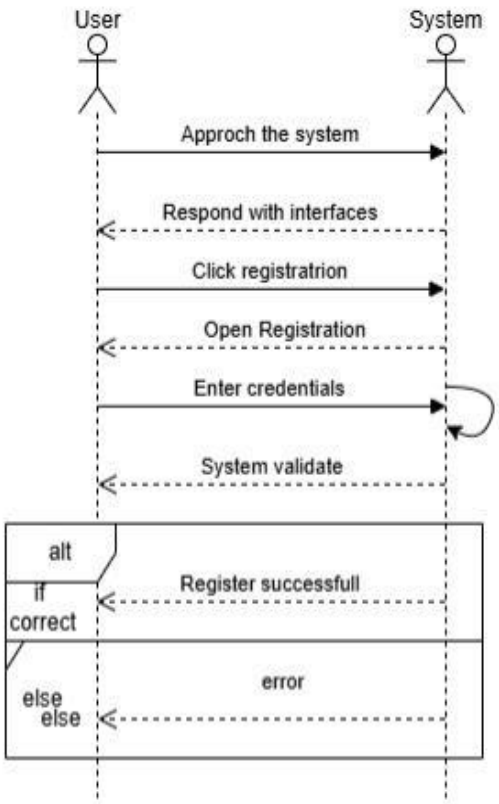
Scenario	System Response
Invalid Input	Displays an error message prompting correction.

Exceptions:

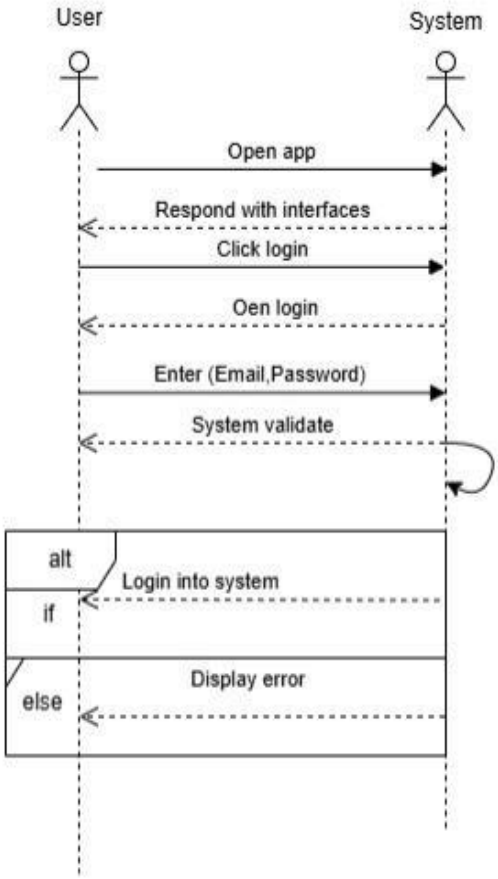
Scenario	System Response
System Error	Displays an error message and advises the user to try again later.

SSD Diagram:

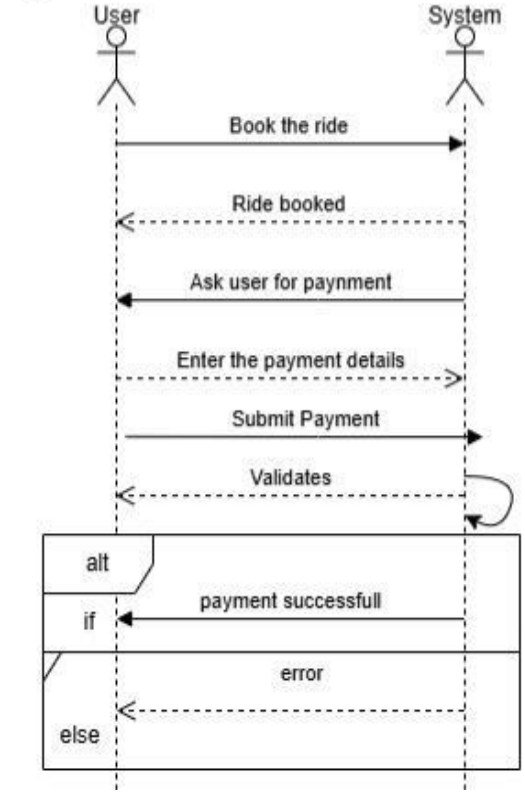
Registration



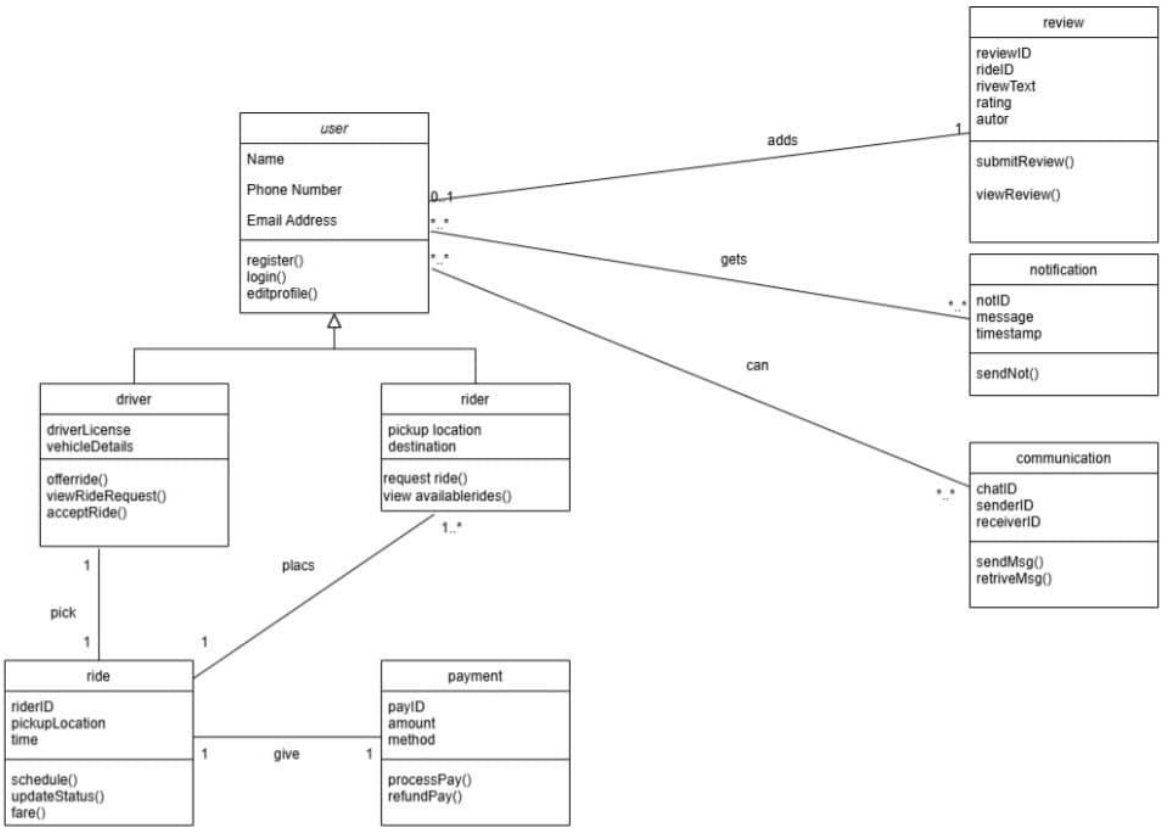
Log-In



Payment



Class Diagram:



Activity Diagrams: