**HW-6 (Architecture and Implementation)**

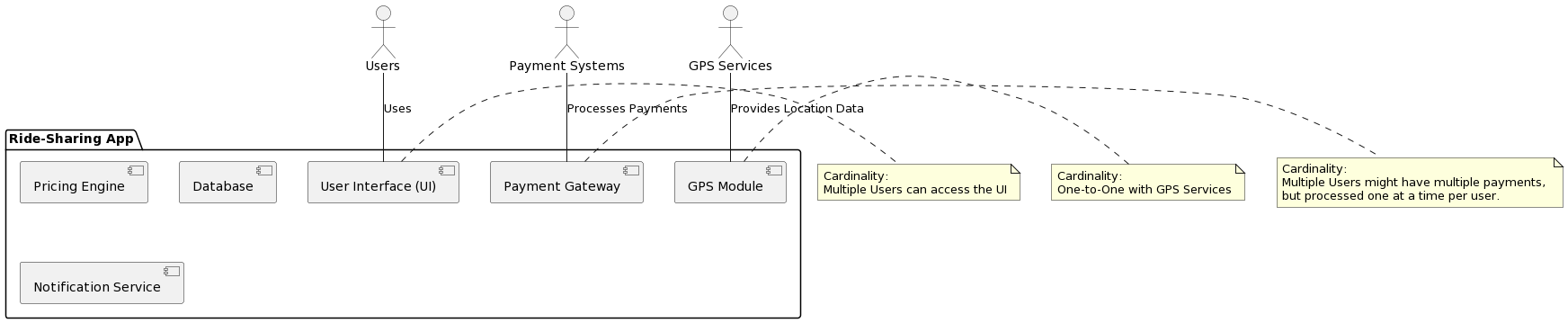
1. **Identify the application you are examining.**

**Ride-sharing application (e.g., Uber):** It connects riders who need transportation to drivers with their own cars. Features include ride requests, fare calculation, real-time tracking, and digital payments.

1. **What are its major components? (Refer slides 10 and 11 in implementation ppt)**

Breaking down a social media platform and identifying its major components:

1. **User Interface (UI):** This is the primary interaction point for both drivers and riders. The design is intuitive offering functionalities like map viewing, ride history, driver/rider profiles, and feedback mechanism.
2. **GPS Module:** Vital for such applications, this module interfaces with the GPS system of the user’s phone or device. For drivers, it provides route navigation. For riders, it showcases the driver’s live location and estimated time of arrival.
3. **Pricing Engine:** Based on various parameters like distance, time of the day, ongoing promotions, or even surge pricing due to high demand, this module calculates the ride’s fare dynamically.
4. **Database:** A robust database system is essential that would store diverse data types including user profiles, ongoing and historical rides, payment details, feedback, and any other important information.
5. **Notification Service:** This is responsible for all real-time communications. For example, when a driver accepts a ride, the rider gets a notification. If a ride is completed, a payment notification is sent.
6. **Payment Gateway:** Integrating with various financial institutions and digital wallets, this ensures secure and efficient processing of payments.
   1. **Create a system context diagram that describes the system.**

****

* 1. **Create use cases for each part of the system context.**
     1. **Rider:**

**User Interface:**

* **Register**: A new user creates an account by providing necessary details like name, phone number, email, and setting up a password.
* **Log In**: Returning users provide credentials to access their account and services.
* **Request Ride**: Users can enter their destination, choose a vehicle type, and request a ride.
* **Pay for Ride**: Once the ride is over, users can make a payment through various modes like credit card, digital wallets, or cash.
* **Rate Driver**:Post-ride, users have the option to rate the driver based on their experience.

**Pricing Engine:**

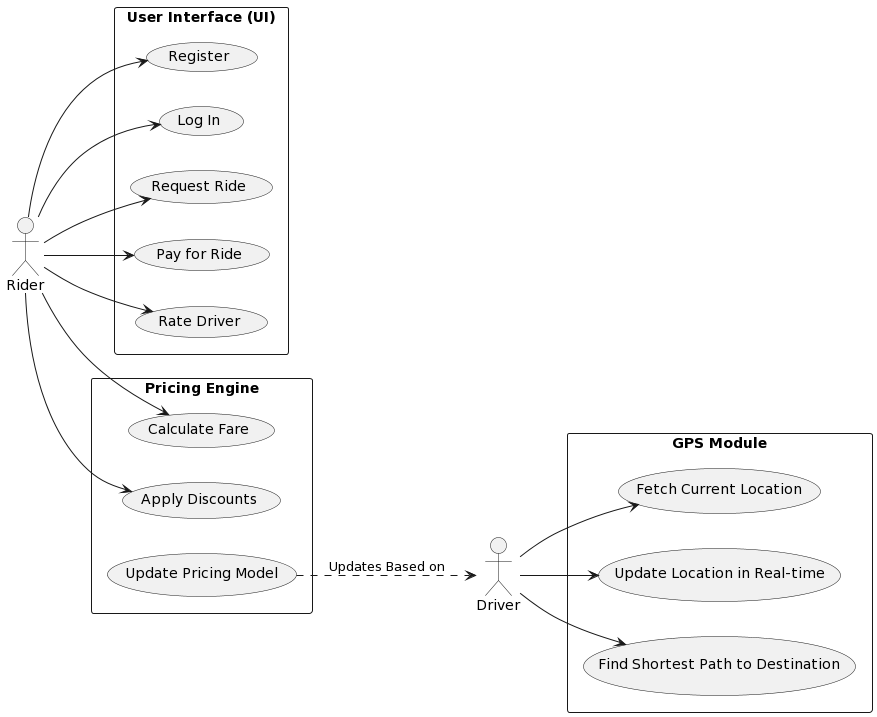
* **Calculate Fare:** Based on parameters like distance, time, and vehicle type, this function determines the fare.
* **Apply Discounts:** If there are ongoing promotions or user-specific discounts, this function applies them.
  + 1. **Driver:**

**GPS Module:**

* **Fetch Current Location:** Acquires the real-time location of the user or driver’s device.
* **Update Location in Real-time:** As the driver or rider moves, this function updates their location in intervals.
* **Find Shortest Path to Destination:** Suggests the quickest route to a destination based on current traffic and other conditions.

**Pricing Engine:**

* **Update Pricing Model:** Occasionally, the pricing algorithms might be updated or tweaked, based on the driver.

****

1. **Identify a design pattern that you think exists in the application.**
   1. **Explain what the design pattern is and how it is evident the design pattern exists OR why you would suggest using that design pattern in the application.**

**Observer Pattern:** It is a behavioural design pattern that defines a one-to-many dependency between objects. The essence of the Observer pattern is having an object (subject) maintain a list of dependents (observers) and notify them of any state changes.

The pattern is most often used in event-driven systems. The key components are the ‘Subject’ (which maintains the list of observers and sends notifications) and the ‘Observer’ (which does some action in response to a state change).

**Evident Existence:**

* **Ride Status Monitoring:** Whenever a ride status changes, say from ‘Waiting for a driver’ to ‘Ride Accepted’ or from ‘Ride in Progress’ to ‘Ride Completed’, the application should notify both the rider and the driver. This immediate update can be seen as the application having implemented the Observer Pattern where the ride’s status is the subject, and the driver and rider apps are the observers.
* **Dynamic Pricing Notification:** For instance, if the application implements dynamic pricing or surge pricing based on demand, riders could be informed in real-time when the surge goes up or down. Here, the pricing engine is the subject, and all active rider apps in a region are observers.

1. **Refer to slide 13 in the architecture ppt. Respond to 3 bullets explaining your thoughts.**

* **How will the system be distributed?**

**Distributed System**:

* + 1. **Global Reach:** By adopting a distributed system, a ride-sharing application can cater to users on a global scale. Spreading servers and data centres across different continents or regions not only provides local data but also ensures the system is always available irrespective of any regional server outages.
    2. **Data Redundancy and Reliability:** Storing user data across multiple sites ensures data redundancy. If one data centre faces an outage, the data is retrievable from another location, ensuring continuous availability and minimizing data loss.
    3. **Latency Reduction:** Local servers mean reduced data travel distances, resulting in faster response times for users. The closer a user is to a server, the quicker their requests can be processed.
* **What architectural styles are appropriate?**

**Microservices Architectural Style:** Each microservice can scale independently. If the Payment gateway witnesses heavy traffic, only that particular service can be scaled up without affecting or scaling the entire application. Microservices allow for continuous deployment. A new feature or an update to a particular service can be rolled out without the need for a complete application shutdown, leading to minimal downtime. If one service fails, it won’t crash the entire application. For instance, if there’s an issue with the Notification Service, users can still request rides or make payments. This isolated failure ensures enhanced system availability.

* **How will the architectural design be evaluated?**

**Stress Testing:** By pushing the system to its limits (in terms of user requests, data processing, etc.), stress tests provide insights into how the system behaves under extreme conditions. It helps in identifying the breakpoints and ensuring the system can handle peak loads.

**Modularity Assessment:** Evaluating the design for modularity means checking how easily features can added or removed. A modular design ensures that changes in one part won’t disrupt the entire system, paving the way for continuous improvement.

**Fault Tolerance:** This can involve testing how the system recovers from crashes or how it switches to a backup server during outages. The goal is to ensure the least disruption to the end-user during any unforeseen incidents.

1. **Refer to slide 21 in the architecture ppt.**
   1. **Explain which architecture pattern you believe is used.**

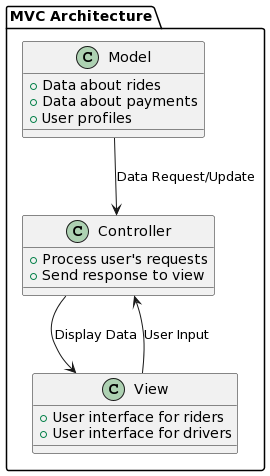
**MVC Architecture:** Model-View-Controller

**Model:** Represents the data structure and the business logic. In our context, all the data stored in databases about rides, payments, user profiles would form the model.

**View:** This is what the end-user interacts with. The user interfaces for both riders and drivers represents the view.

**Controller:** Acts as an intermediary that processes and relays the user’s requests to the model and send the appropriate response back to the view.

* 1. **Also provide a diagram for what you think that architecture pattern would look like.**

****