Practice

Case Study

- **Uber**, founded in 2009, revolutionized the transportation industry by creating a technology-driven ride-hailing platform. The platform seamlessly connects riders and drivers, enabling the booking of rides via a mobile application.

- Uber’s platform involves several key actors like **Rider**, **Driver**, **Admin**, and an external **Payment Gateway**. Each of these actors plays a critical role in the system’s overall operation. The **Rider** is the customer who requests transportation, while the **Driver** fulfills this request by offering their vehicle and services. The **Admin** oversees the entire system, handling the management of users, resolving disputes, and ensuring that payments are correctly processed. Finally, the **Payment Gateway** handles the transactional aspect of Uber, ensuring secure and seamless payment processing.

- The interaction begins with **registration** and **login**, where riders create an account and provide personal details such as contact information and payment methods. Once authenticated, the rider **requests a ride**, wherein they input their pick-up and drop-off locations, select the ride type (UberX, UberPool, etc.), and submit the request to available drivers. The rider can then **track the ride** in real-time using GPS integration, receiving constant updates about the driver's location and estimated time of arrival. At the end of the journey, riders engage in **payment processing**, using stored payment methods that interact with the **Payment Gateway** for secure transactions. Post-ride, the rider has the option to **rate the driver**, providing feedback that contributes to Uber’s driver-rating system.

- Like riders, drivers must **register** and **log in** to the system. However, their registration process involves more scrutiny, requiring them to submit documents such as driver's licenses, vehicle registration, and insurance. Once logged in, the driver's primary task is to **accept ride requests** sent by the system based on proximity and availability. Upon accepting a ride, the driver navigates to the rider's location, using the app to **track the ride's progress** and ensure efficient routes. After completing the journey, drivers **mark the ride as complete**, which triggers the payment process. Similar to riders, drivers can also **rate the rider**, ensuring two-way feedback that maintains quality standards on both sides.

- The **Admin** is responsible for ensuring the smooth operation of the Uber platform. This role includes managing rider and driver accounts, handling driver documentation, and resolving disputes. Admins are also involved in overseeing the **payment management** process, ensuring that payments are appropriately divided between the driver and Uber, handling refunds, and addressing transaction failures. In addition, the Admin has oversight of the **support system**, where both riders and drivers can submit inquiries or complaints related to rides, payments, or technical issues. One notable feature is the **rating system**, which extends from the ride completion. After the trip, both rider and driver rate each other. This feedback is vital for maintaining service quality, influencing driver eligibility and potentially affecting a rider's future ride experience. The Admin can intervene if there are disputes related to these ratings or any inappropriate behavior reported by either party.

Steps to Create the Diagram:

- **Identify Actors:** List out all primary actors
- **Identify Use Cases:** Each actor should have specific actions they perform in the system.
- **Draw the System Boundary:** Place all the use cases within the system boundary.
- **Draw Use Case Relationships**:

Actors:

- Rider (**Primary**)
- Driver (**Primary**)
- Admin (**Supporting**) Admin doesn't directly interact with the system to achieve the core business goals (e.g., requesting or completing rides). Instead, the Admin performs background tasks that support and maintain the system's operations.
- Payment Gateway (**External)**

Why GPS is not a secondary system?

- The **GPS system** is not directly interacting with the system in the same way that users (like Riders, Drivers, or Admins) do. It functions more as a tool or a resource within the system, providing data to support use cases such as Request Ride or Track Ride.

- GPS provides location data in the background, but it does not initiate actions or make decisions like an external actor would.

- **GPS is a Supporting Technology**

Main Use Cases:

Rider	Driver	Admin	Payment Gateway
• Register • Login • Request Ride • Track Ride • Make Payment • Rate Ride • View Ride History • Contact Support	• Register • Login • Accept Ride • Complete Ride • Rate Rider • View Ride History • Contact Support	• Manage Users • Manage Drivers • Manage Payments • Handle Support Requests	• Process Payment

Accept ride

- Although it may seem like just tapping a button to accept a ride, the Accept Ride action is a critical decision point in the system. It transitions the ride request from a pending state to an active trip and is the driver's confirmation to participate in a ride. Without this action, the ride cannot proceed. The system depends on the driver's acceptance to assign a ride and proceed with the booking.

- **"Rate Driver"** use case is crucial to maintaining Uber’s two-way feedback system, which helps ensure service quality. Riders rate drivers after each trip, which directly impacts the driver’s reputation and future opportunities within the platform.

Complete Ride

- seem like a single action performed by the driver, it triggers several important system operations and business functions.
- initiates a series of important automated processes within Uber’s system:
  - **Fare Calculation**: Once the ride is completed, the system finalizes the fare, considering factors like time, distance, surge pricing, and tolls.
  - **Payment Processing**: The system triggers the payment process, charging the rider’s saved payment method and initiating the transfer of earnings to the driver.
  - **Receipt Generation**: Uber generates a detailed receipt for the rider, summarizing the cost and the trip details.
  - **Driver Availability Update**: The driver’s status is updated from "on a trip" to "available" again, allowing them to accept new ride requests.
  - **Feedback Loop**: Both rider and driver are prompted to rate each other, a crucial part of Uber's reputation system.

accessing past ride information

- The primary goal of the **Rider** or **Driver** in this use case is to **access past ride information** for a variety of reasons, such as:Reviewing previous trips for expense tracking.
- Checking the details of the routes, dates, or times of rides.
- Retrieving receipts for payments.
- Resolving disputes or issues related to a ride.

Why login might still appear in a use case diagram

- **Technical Necessity:** In most systems, logging in is necessary to access other core functions, such as booking a ride or accepting a ride request. Without login, users wouldn’t be able to perform their intended actions.
- **Supporting Functionality:** While login isn’t a business goal, it supports critical functionality (e.g., verifying user identity, associating past rides or payment methods).

1. Request Ride (Rider)

- **Description**: The rider selects their pick-up and drop-off locations, ride type, and submits a ride request.
- **Actor**: Rider

2. Accept Ride Request (Driver)

- **Description**: A driver receives a notification of a ride request and decides whether to accept or reject it.
- **Actor**: Driver

3. Track Ride (Rider & Driver)

- **Description**: Both the rider and driver use real-time GPS to track each other's location and the ride's progress.
- **Actors**: Rider, Driver

4. Make Payment (Rider)

- **Description**: After the ride is completed, the rider is charged, and payment is processed through the app's integrated payment system.
- **Actor**: Rider, Payment Gateway (External)

5. Complete Ride (Driver)
•**Description**: The driver marks the ride as completed once the rider is dropped off at their destination.
•**Actor**: Driver

6. Rate Ride (Rider & Driver)
•**Description**: After the ride, both rider and driver provide feedback by rating each other.
•**Actors**: Rider, Driver

7. View Ride History (Rider & Driver)
•**Description**: Users can view a list of their completed rides and details such as date, time, and cost.
•**Actors**: Rider, Driver

8.Manage Profile (Rider & Driver)
•**Description**: Users can update personal information such as payment methods (for riders) or vehicle information (for drivers).
•**Actors**: Rider, Driver

9. Support Request (Rider & Driver)
•**Description**: Users can request support for any issues related to rides, payments, or account problems.
•**Actors**: Rider, Driver, Admin

Fully Dressed Use Case: Request Ride (Rider)

1. **Use Case Name: Request Ride**

2. **Primary Actor: Rider**

3. **Scope: Uber**

4. **Level: user goal**

5. **Stakeholders and Interests:**
   - **Rider**: Wants to quickly request a ride and be matched with an available driver.
   - **Driver**: Wants to receive ride requests and complete rides to earn income.
   - **Admin (Uber)**: Ensures that the system operates efficiently, matching riders and drivers effectively.
   - **Payment Gateway**: Processes payments once the ride is completed.
   - **Government/Regulators**: Ensure the system complies with local transportation regulations, safety, and driver/rider accountability.

6. Preconditions:

- The rider must be logged in to the Uber app.
- The rider must have a valid payment method associated with their account.
- The rider’s GPS must be enabled to determine their current location.

7. Postconditions:

- The system has successfully matched the rider with a driver, and the ride request has been accepted.

8. Main Success Scenario (Basic Flow):

1. The rider wants to book the ride using Uber app
2. System identifies the rider's current location using GPS.
3. The rider inputs their desired drop-off location.
4. The system displays different ride options (e.g., UberX, UberPool, etc.).
5. The rider selects a ride type based on price and availability.
6. The system calculates and displays an estimated fare based on distance, traffic, and ride type.
7. The rider confirms the ride request by tapping the "Request Ride" button.
8. The system searches for available drivers in the nearby area.
9. An available driver accepts the ride request.
10. System provides the rider with driver and vehicle details (e.g., name, photo, car model, license plate).
11. The system displays the driver's current location and estimated time of arrival (ETA) for the rider.
12. The ride is officially matched, and the rider waits for the driver to arrive.

9. Extensions (Alternate Flows):

- **3a. No Available Drivers**:
  - 3a1. The system informs the rider that no drivers are available at the moment.
  - 3a2. The rider is given the option to try again or switch to a different ride type.
- **5a. Payment Method Failure**:
  - 5a1. The system detects that the rider's payment method is invalid or expired.
  - 5a2. The rider is prompted to update their payment method before proceeding with the request.
- **6a. Rider Cancels Ride**:
  - 6a1. Before the driver accepts, the rider chooses to cancel the ride request.
  - 6a2. The system cancels the request and informs the rider of the cancellation policy.
- **7a. Driver Declines Ride**:
  - 7a1. The system attempts to match the rider with a driver, but the driver declines.
  - 7a2. The system searches for another available driver.
  - 7a3. If no other drivers are available, the system informs the rider.

Special Requirements:

- The app must provide real-time tracking of driver location and ride status updates.
- The system must process ride requests in under 5 seconds to ensure a seamless user experience.
- GPS must accurately detect the rider's pick-up location to avoid confusion.
- Surge pricing must be clearly communicated to the rider if applicable.