Team members:

1. **Kamal Abdullayev**
2. **Feray Gulu-zada**
3. **Nika Mgaloblishvili**

Task #1 (0.5 points): Please provide an updated description of the system you are modelling. Make sure that it matches the designed model as closely as possible and that all details (previously - classes, their attributes, relationships between classes etc., now – also actors, use-cases, operations etc.) are specified in both the model and the description.

The system being modeled is a ride-hailing platform, similar to Bolt, comprising users (customers and drivers), vehicles, orders (rides), transactions, and related elements.

The User class includes attributes such as ID, name, contact details, and account status. Users can either be Customers or Drivers, with drivers having specific attributes like earned money and work hours. Each driver must possess a License and own a Vehicle.

Vehicle types include Cars, Scooters, and Electric Cars, each with specific attributes like the number of seats or battery level. Orders are linked to customers, vehicles, and transactions, with order and transaction statuses defined by enumerations.

Payment is handled via Transactions, which are associated with a PaymentMethod (e.g., card or cash) or a subscription. Addresses are linked to users, and relationships between classes are defined by multiplicity. For example, one user can have many orders, but each order belongs to a single user.

Update:

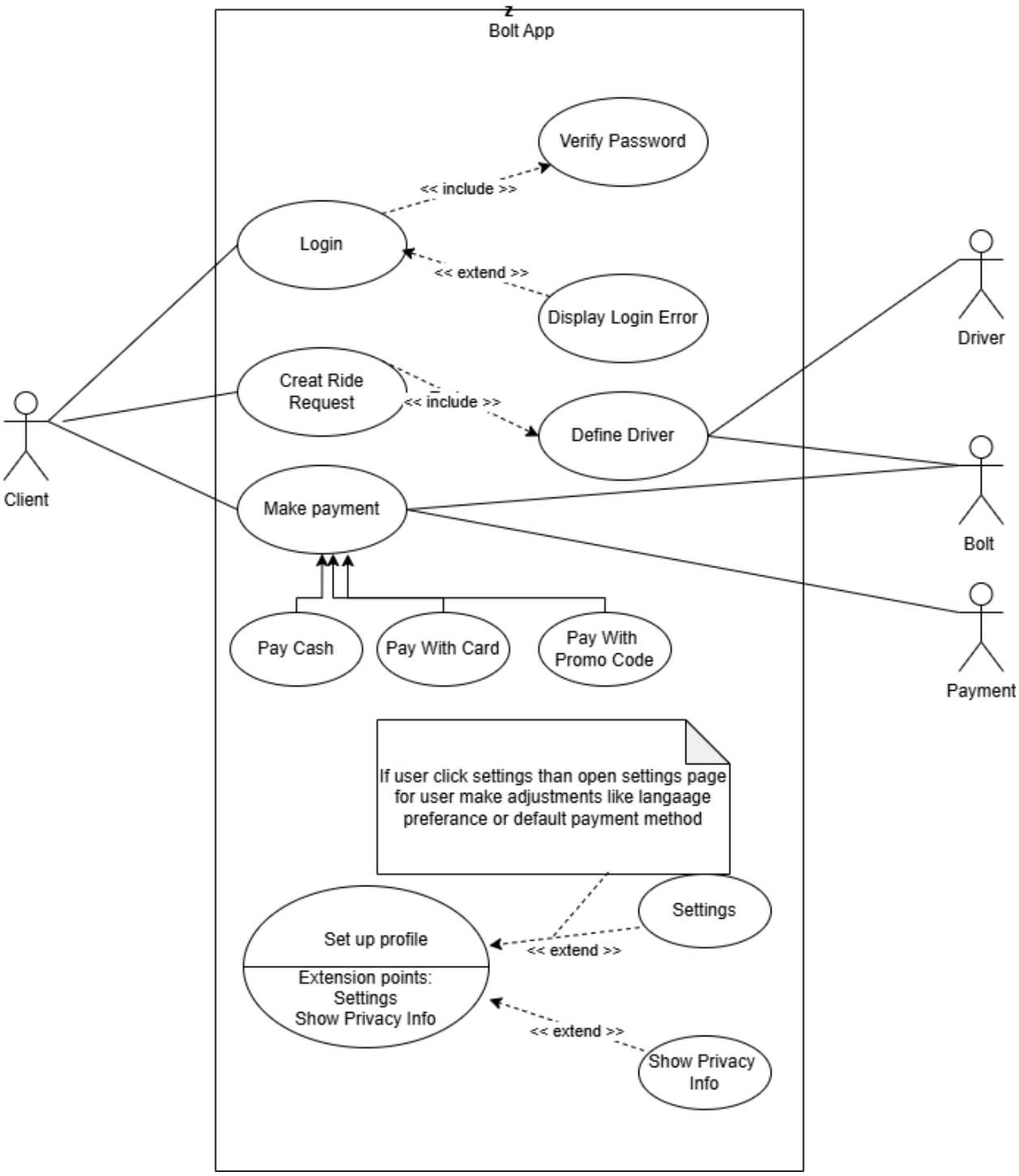
This time, we've included actors such as User (customer/client), Bolt, Payment, and Driver…

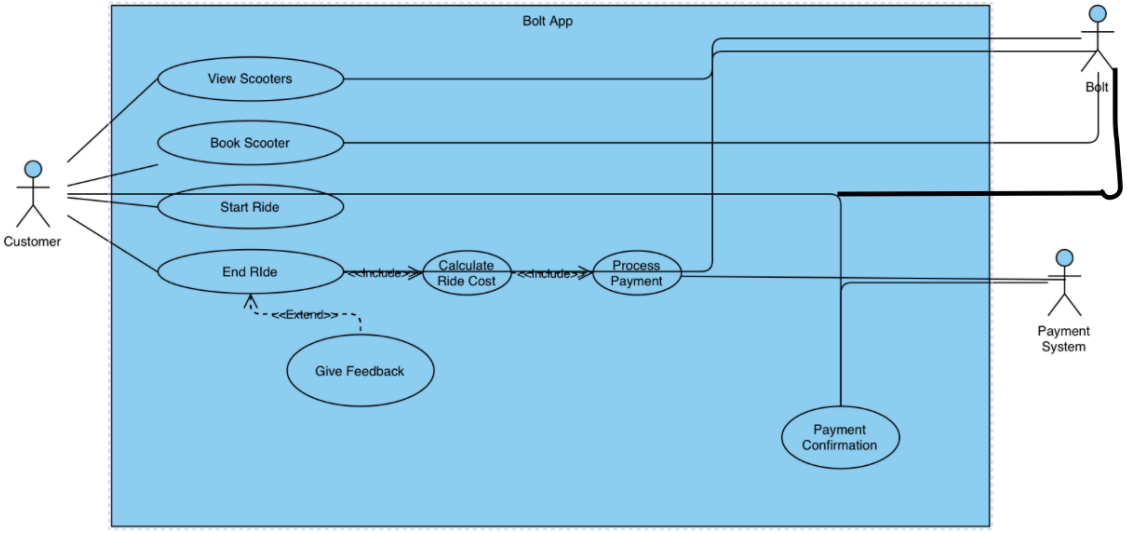
We've added various use cases based on the previous architecture, including actions like logging in, setting up a profile, ordering a ride, or hiring a scooter. Additionally, detailed sequence diagrams have been incorporated to illustrate workflows clearly. By adding both use case and sequence diagrams, the architecture has become more structured, clear, and easier to understand.

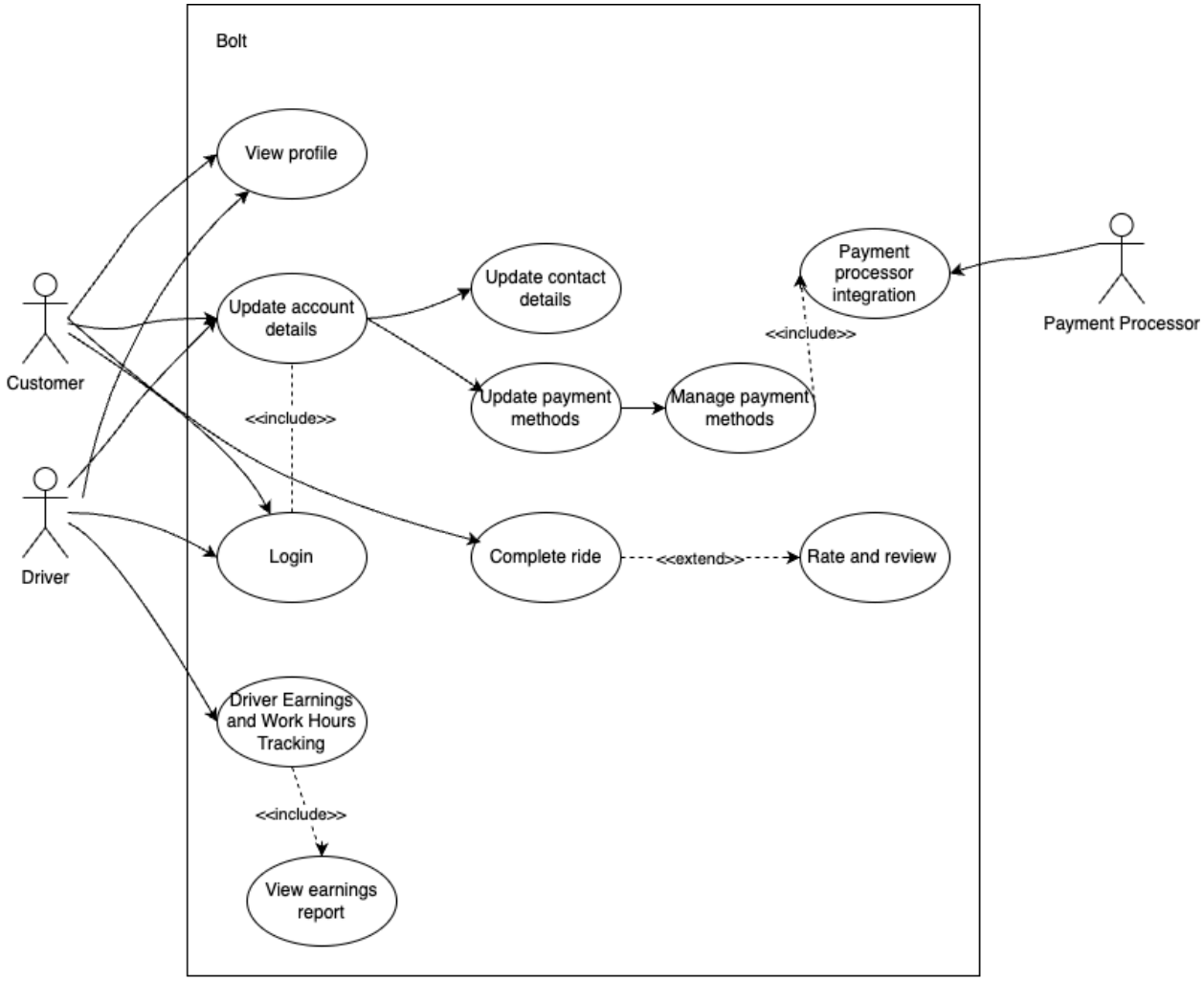
Task #2 (3 points): Provide a use-case model(s) for the system you described in Task #1.

Make sure it is as detailed as necessary (!).

Make sure that you model(s) has at least 10 use-cases, 4 actors, four association types, including generalization (of two types), "extend", "include" relationships, 3 use-case descriptions


Bolt App
Verify Password
<< include >>
Login
<< extend >>
Display Login Error
Creat Ride Request
<< include >>
Define Driver
Make payment
Client
Driver
Bolt
Payment
Pay Cash
Pay With Card
Pay With Promo Code
If user click settings than open settings page for user make adjustments like langaage preferance or default payment method
Settings
Set up profile
<< extend >>
Extension points:
Settings
Show Privacy Info
<< extend >>
Show Privacy Info


Bolt App
View Scooters
Book Scooter
Start Ride
End RIde
Calculate Ride Cost
Process Payment
Give Feedback
Payment Confirmation
<<Include>>
<<Include>>
<<Extend>>
Customer
Bolt
Payment System



End Ride Description

Actors: User, Bolt, Payment System

Trigger: User has completed their ride and chooses to end it via app interface

Pre-Conditions:

- User has started the ride and is using the scooter
- User is within the acceptable distance or area to end the ride
- The ride is active, and scooter is in use

Post-Conditions:

- The user's ride is marked as ended

- Ride Cost is calculated
- Payment is processed
- Payment confirmation is sent to User
- Ride details are saved in the User's ride history

Main Flow:

1. **User** opens the app and selects the option to end the ride.
2. **Bolt** confirms that the scooter is in a valid location to end the ride
3. **Bolt** triggers the **Calculate Ride Cost** process.
    - **Bolt** calculates the cost based on time.
4. **Bolt** triggers the **Process Payment** use case.
    - **Payment System** charges the **User** for the ride.
5. **Payment System** sends a **Payment Confirmation** to the **User**.
    - **User** receives confirmation of successful payment or failure.
6. **Bolt** updates the ride as completed.
7. **Bolt** sends a **Payment Confirmation** to **User** to finalize the ride status in the system.
8. **User** is prompted with the option to **Give Feedback** about the ride.

Alternative Flow:

A1: User is unable to end the ride due to location issues:

- **Step 2 Alternative**: If the **User** is not in a valid location to end the ride, **Bolt** notifies the **User** that they cannot end the ride and prompts them to move to a valid location.
- The **User** must move to the valid location, or the system may suggest another nearby location.

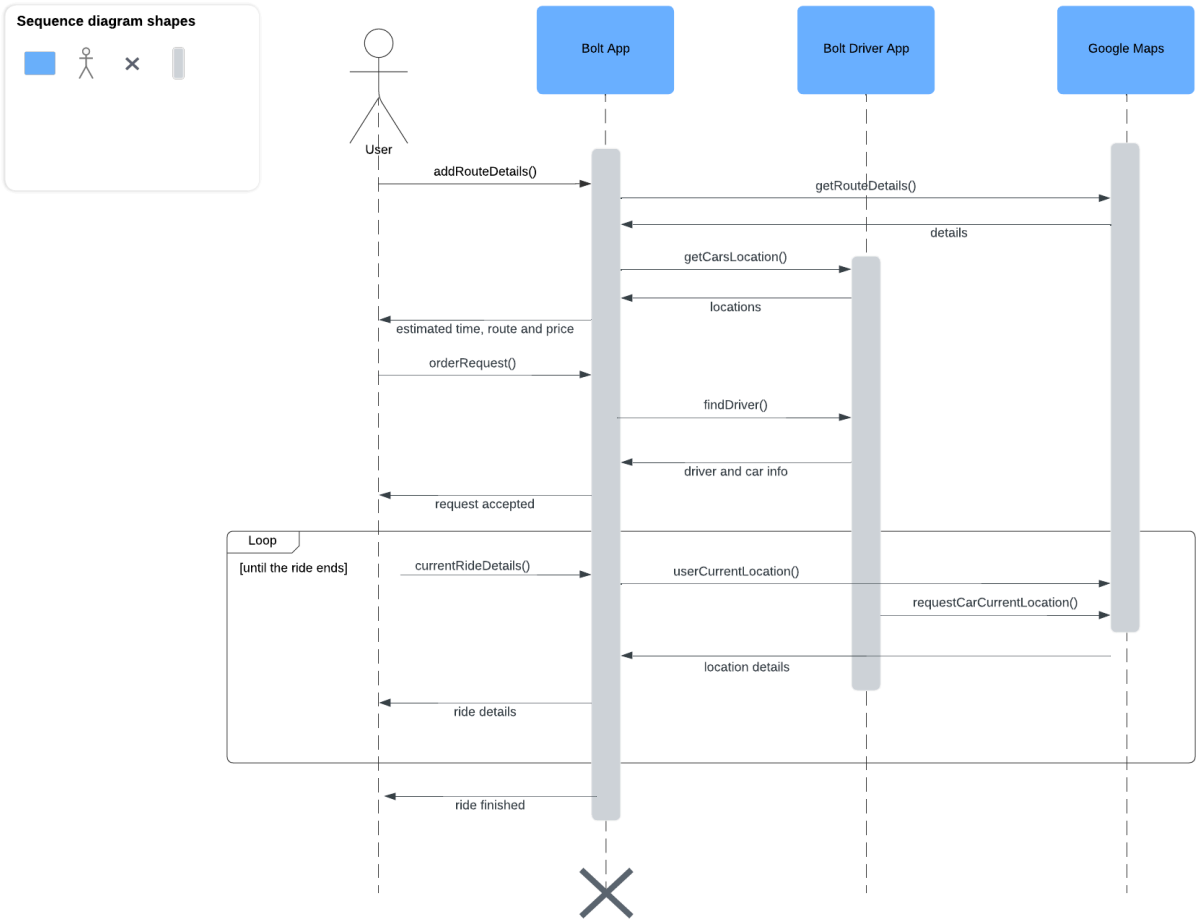
A2: Payment fails:

- **Step 4 Alternative**: If the **Payment System** fails to process the payment (e.g., insufficient funds or connection issues), **Bolt** notifies the **User** that the payment could not be completed.
- The **User** is asked to retry payment or use another payment method.
- **Bolt** may block the ride completion until the payment is successfully processed.

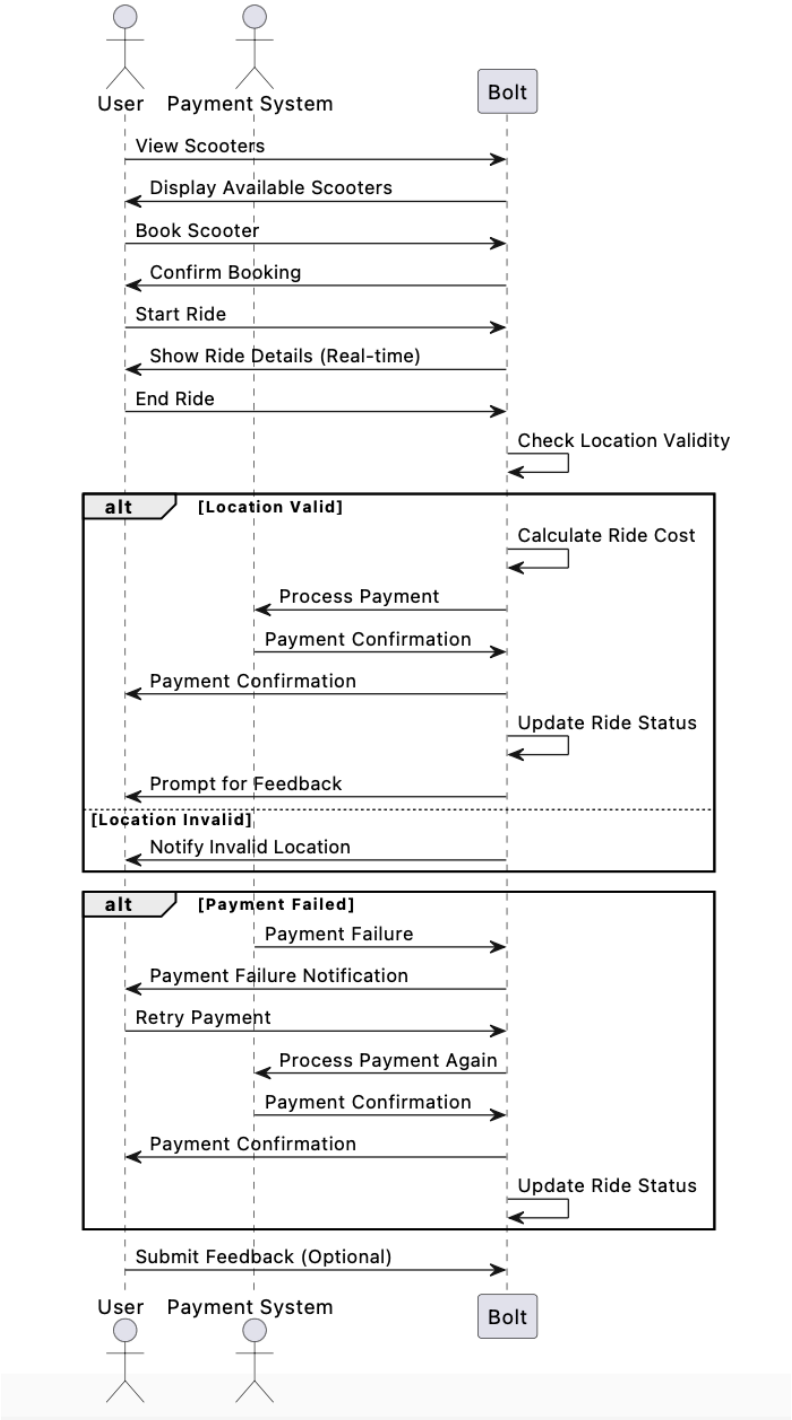
A3: User decides not to end the ride:

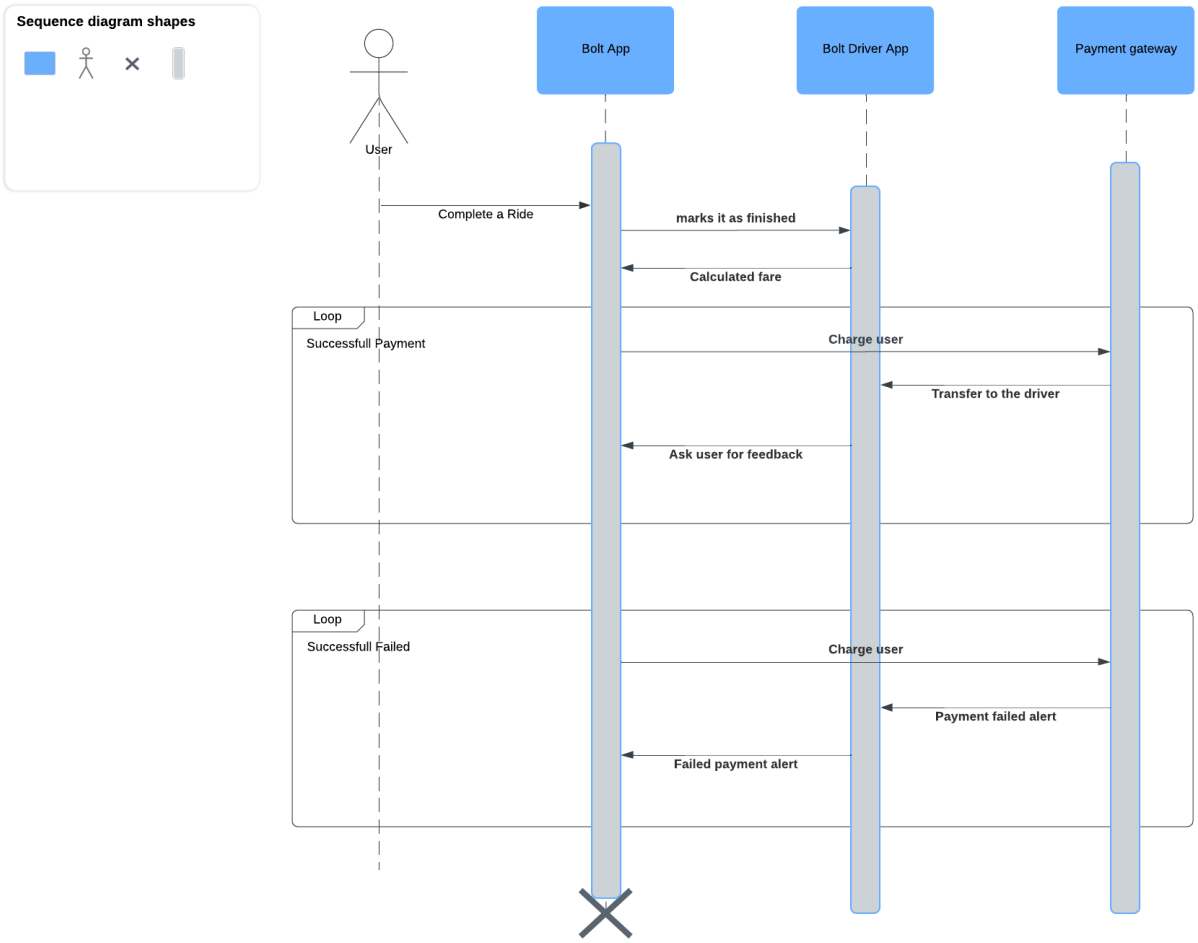
- **Step 1 Alternative**: If the **User** decides to cancel ending the ride during the process, they can select an option to continue the ride.
- **Bolt** cancels the end-ride operation and returns the **User** to the ride interface.

Task #3 (3 points): Provide at least three sequence diagrams for the system you described in Task #1. Make sure you use at least two frames.



Scooter Sequence Diagram





Task #4 (0.3 point): What is the environment, which you choose to design your use-case diagram and sequence diagram? What were the decisive factors to give the preference to this environment(s)?

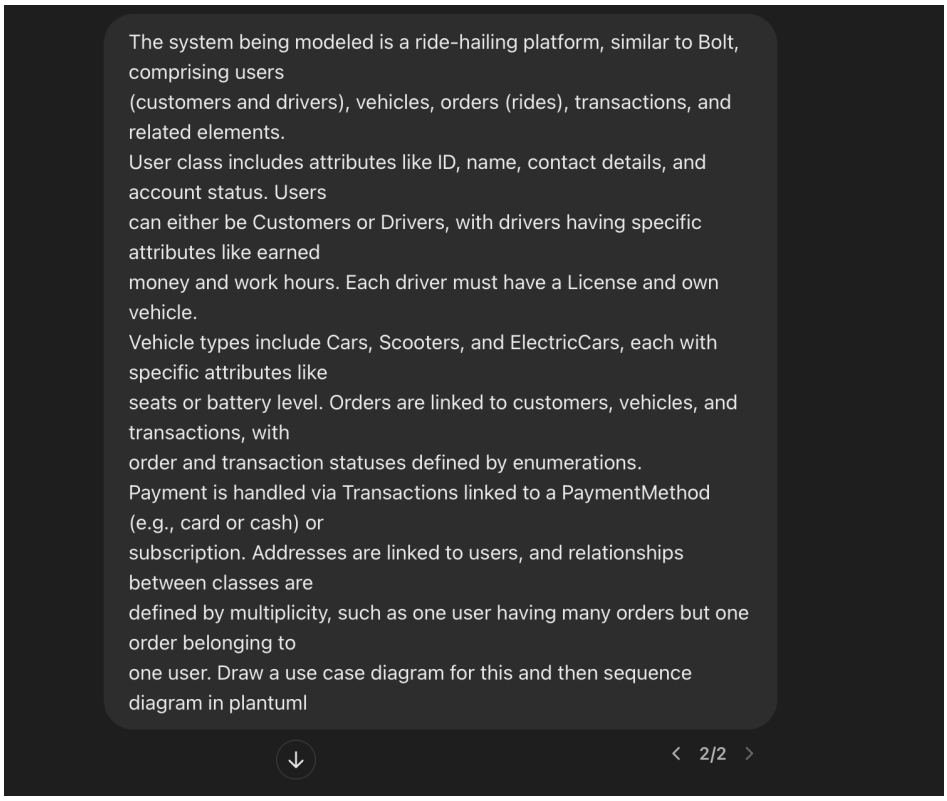
- Lucidchart
    - It provides basic schema for quick start
- DrawIO
    - It's easy to find the needed diagrams

Task #5 (0.2 point): Provide evidence that the work was conducted in a team (e.g., in the form of a link to the VCS; the screenshot of the above etc.).

- We used a shared word file and each contributed our own. We cannot provide any images at this time

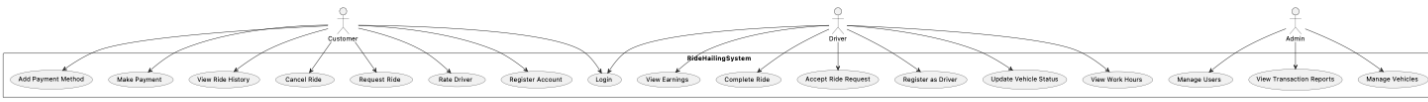
Task #6*(3 points): Explore and document Generative AI capabilities in modelling use-case model and sequence diagram model (in addition to Task#2 and #3, but not instead!). When selecting the tool, seek for specialized tools for the given purpose. Justify your choice.

Provide prompts you used to obtain the result.

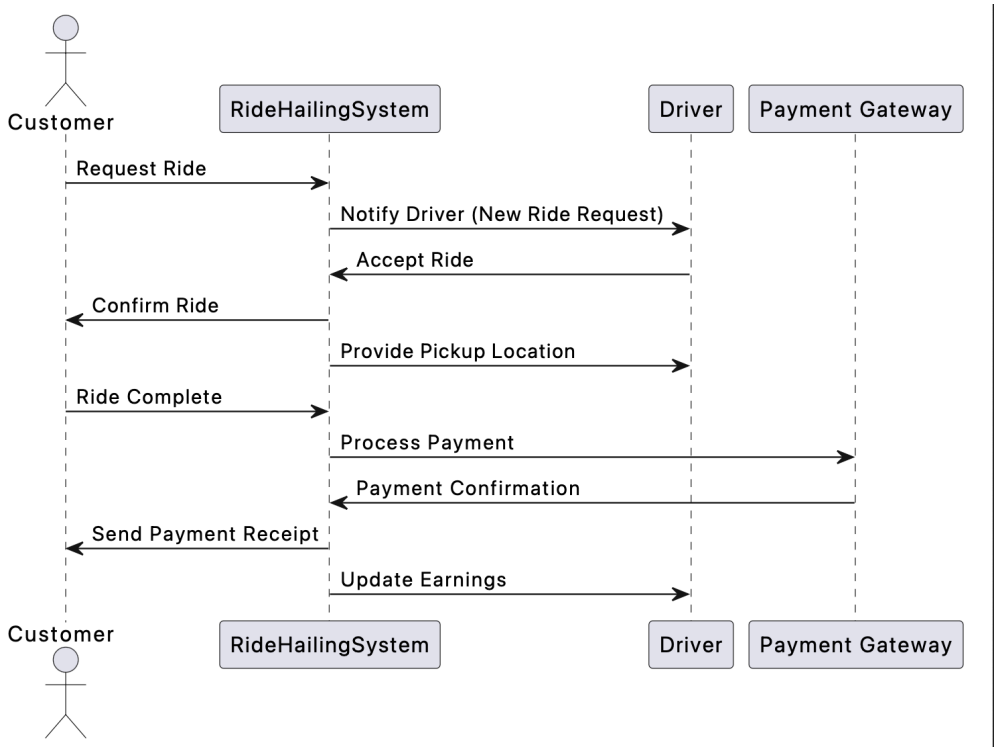


Compare the model produced by Generative AI tools with the one designed by you - assess both critically. Was it possible to achieve the same result that you obtained in Task #2 and #3? If not, what were the differences? What are strengths and weaknesses of GenAI tools in addressing this task?

This is ChatGPT's diagram:



It is absolutely not the same result. Instead it is very simple and doesn't cover most details. It's not necessarily wrong, just very simple, hence doesn't do its job as a use case diagram.



For the sequence diagram though, ChatGPT did a better job. Although more simplified than ours this could be counted as a normal sequence diagram.

So ChatGPT can draw good sequence diagrams, definitely not use case diagrams, only the very simple ones and struggles a lot with details. So we would say that ChatGPT is good for drawing simple diagrams, to speed up the drawing process, and if used with some human understanding of the process could help achieve faster results. But it's very weak with details.

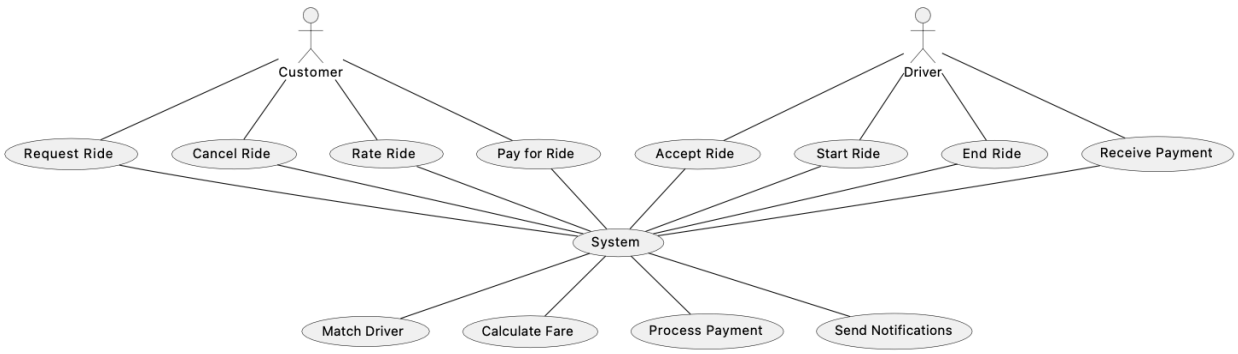
Had the use of GenAI tools allowed you to improve either the description (Task#1) or the model (Task#2)?

Not really, they make mistakes.

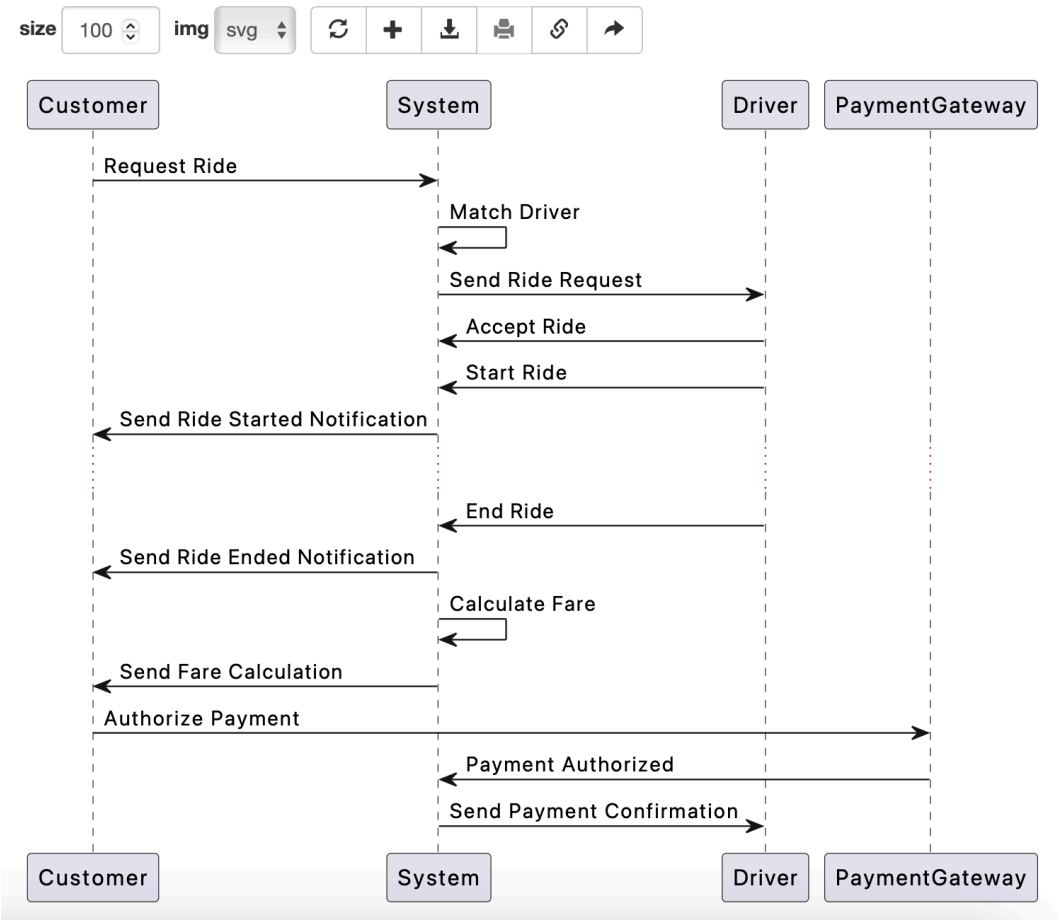
(*optional (+up to 5 bonus points)): if more than one tool has been tried and critically assessed

As a second tool we asked Gemini to do the same thing. Below are the prompts and their results:

The system being modeled is a ride-hailing platform, similar to Bolt, comprising users (customers and drivers), vehicles, orders (rides), transactions, and related elements. User class includes attributes like ID, name, contact details, and account status. Users can either be Customers or Drivers, with drivers having specific attributes like earned money and work hours. Each driver must have a License and own vehicle. Vehicle types include Cars, Scooters, and ElectricCars, each with specific attributes like seats or battery level. Orders are linked to customers, vehicles, and transactions, with order and transaction statuses defined by enumerations. draw a use case diagram for this and then sequence diagram in plantuml Payment is handled via Transactions linked to a PaymentMethod (e.g., card or cash) or subscription. Addresses are linked to users, and relationships between classes are defined by multiplicity, such as one user having many orders but one order belonging to one user.



Gemini's use case diagram is a little better than ChatGPTs but once again the same weakness is evident, it is very simple and avoids all details.



Just like in the previous example Gemini does a decent job creating a sequence diagram, but once again this is not as detailed as the one that could be created with the help of humans.

So all in all AI can be used as an assistant to humans while drawing these diagrams, and could speed up the process of drawing. But definitely not to replace the human force.