SM2023 assignment #3

Task #1 (0.5 point): 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, actors, use-cases, sequence of operations etc., now - states, operations etc.) are specified in both the model and the description.

Expected length – min ½ - 1 A4 page.

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…

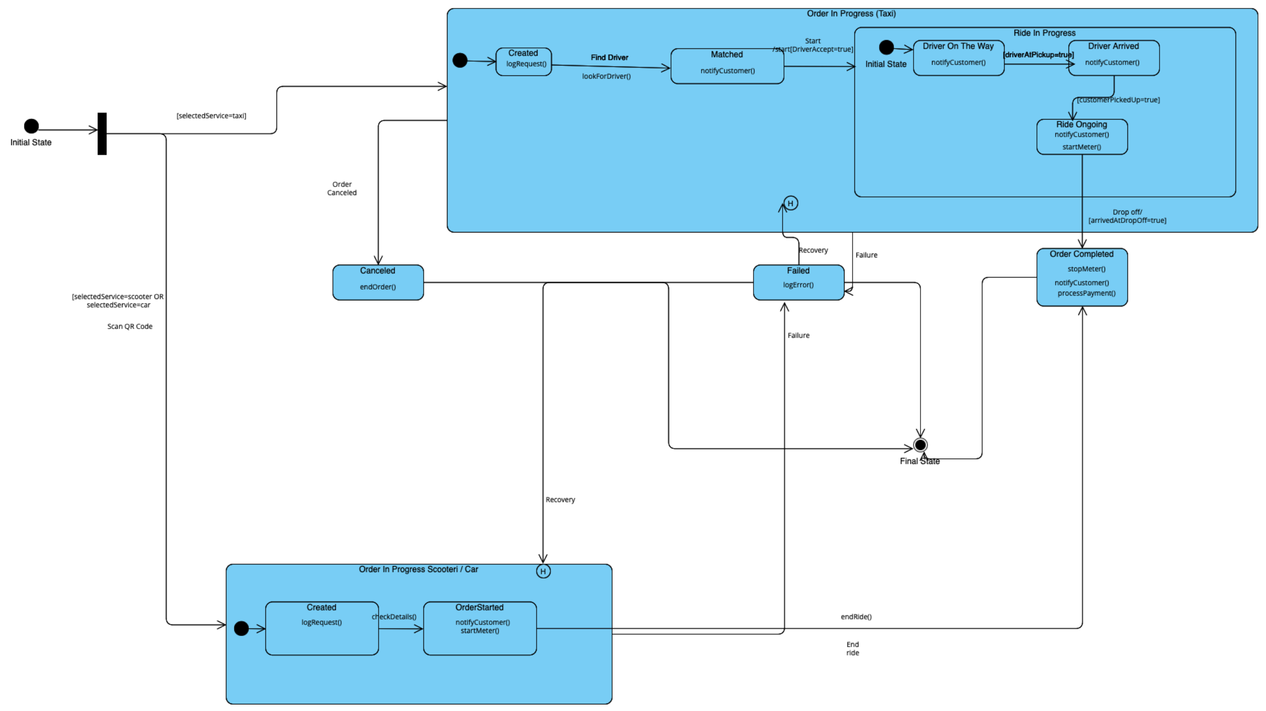
We've added various use cases based on the previous architecture, including actions like logging in, setting up a profile, ordering a ride, or hiring a scooter. Additionally, detailed sequence diagrams have been incorporated to illustrate workflows clearly. By adding both use case and sequence diagrams,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

Update 2:

This time we added a state chart to represent the application's lifecycle. This includes the car/scooter hiring process and the taxi ordering scenario, covering different outcomes such as success, cancellation, and failure. We also updated the class model diagram from Assignment 1 by adding some functions that were missing before to make it more complete.

Task #2 (3 points): Provide state chart for the system you described in Task #1. Make sure you use composite states (where relevant) at least once.

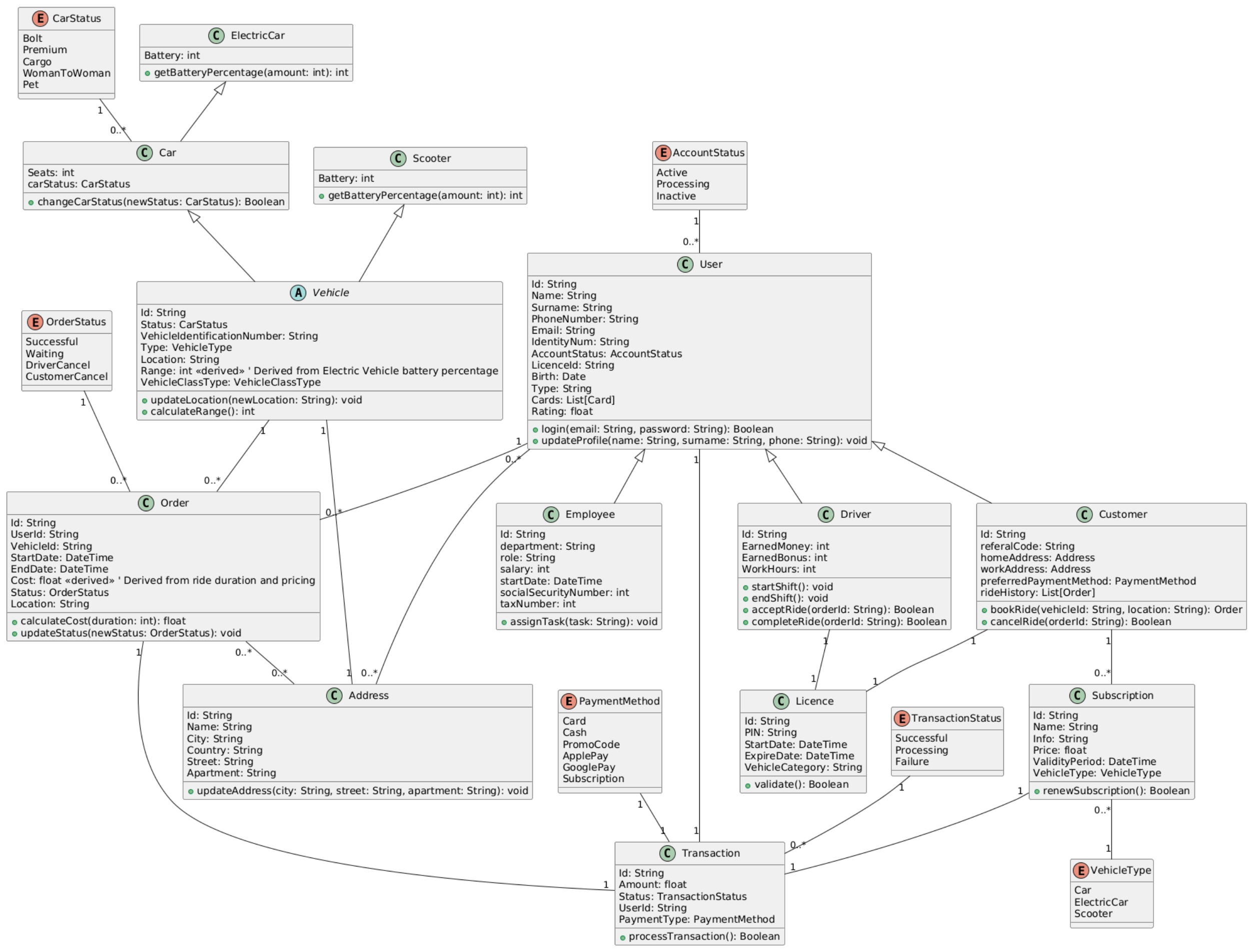
Note that the model should consist of at least at least 4 states, 7 events, guards, at least three types of actions/operations, composite states (at least once, where relevant), history pseudo-states (at least once, where relevant)



Task #3 (2 points): Provide application class model for the system you described in Task#1.

Add at least one sentence describing how does it differ from the model you developed as part of assignment#1 Task#1.

In the first task, we didn't add any methods to the model. This time, we decided to add methods to the classes for core functionalities (getBatteryPercentage, updateLocation, bookRide, processTransaction...)



Task #4 (1.2 point): What is the environment, which you choose to design your statechart and application class model? What were the decisive factors to give the preference to this environment(s)?

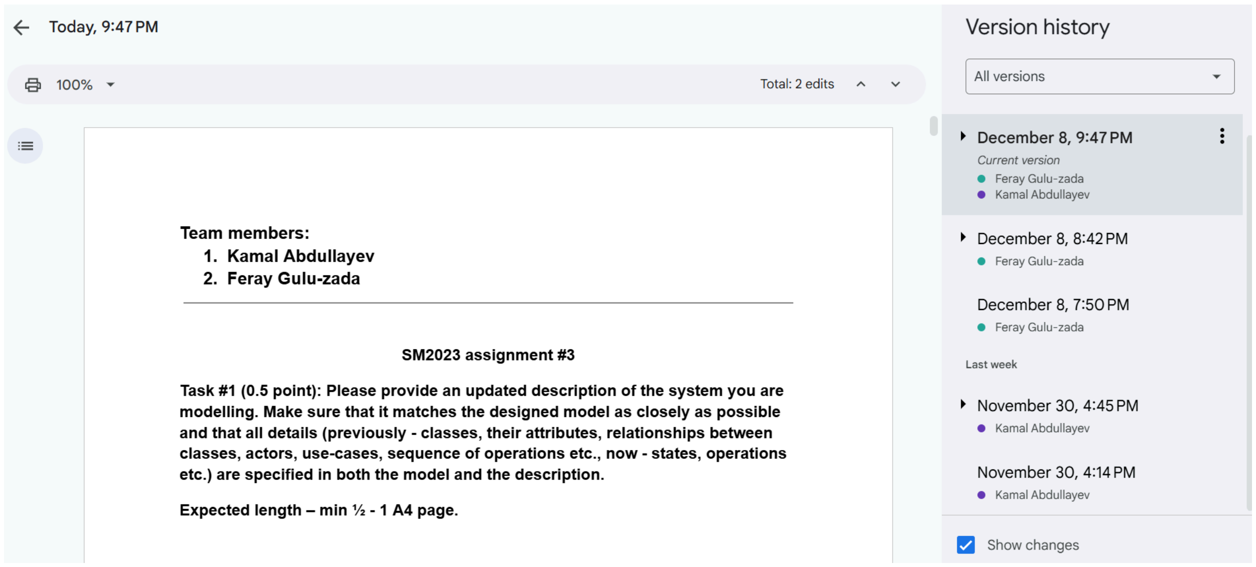
(1 point out of 1.5): Does it allow you to generate the code from the designed model? If yes, what are your observations regarding the generated code? Have you identified any weakness in it? Does the code match the designed model? If not, were the expectations-reality differences due to deficiencies in the original model that you were forced to correct?

The tools we used for design our statechart and application class model:

- PlantUML => Offers flexibility to make quick changes as needed. It also saves significant time by allowing us to write code instead of manually drawing diagrams
- Visual Paradigm => Features a simple user interface that streamlines the workflow and makes the process more efficient

Task #5 (0.3 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.).

The shared doc file activities:



Team members:
1. Kamal Abdullayev
2. Feray Gulu-zada

SM2023 assignment #3

Task #1 (0.5 point): 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, actors, use-cases, sequence of operations etc., now - states, operations etc.) are specified in both the model and the description.

Expected length – min ½ - 1 A4 page.

December 8, 9:47 PM
Current version
● Feray Gulu-zada
● Kamal Abdullayev

December 8, 8:42 PM
● Feray Gulu-zada

December 8, 7:50 PM
● Feray Gulu-zada

Last week

November 30, 4:45 PM
● Kamal Abdullayev

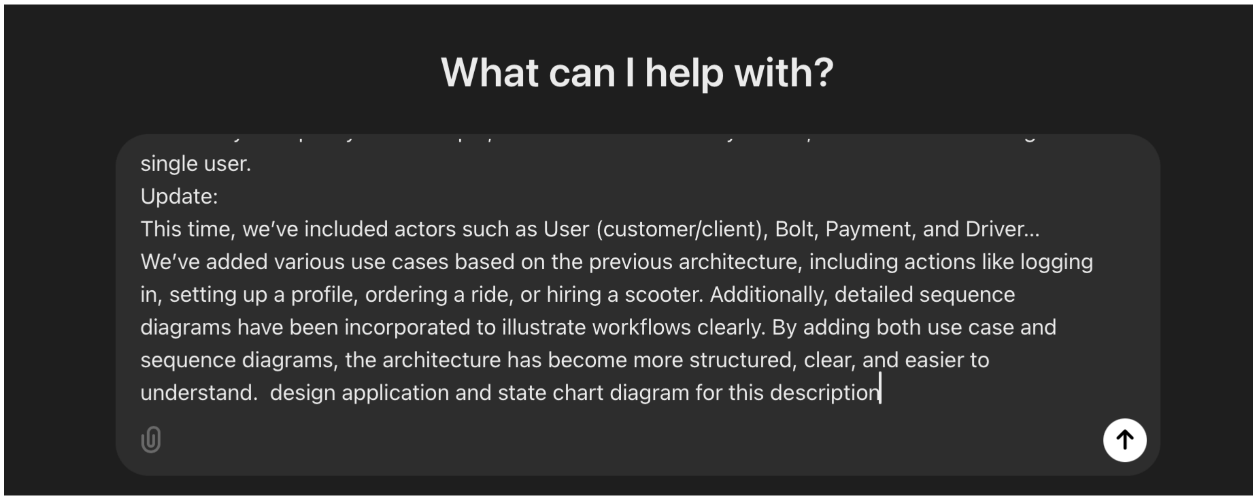
November 30, 4:14 PM
● Kamal Abdullayev

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.

For this task we tried ChatGPT (first we wanted to try special GPT for modeling but it required a paid subscription so we sticked to the regular one)

We asked it to design the Activity Model and State Chart Model for our system. Below is the prompt:



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. design application and state chart diagram for this description

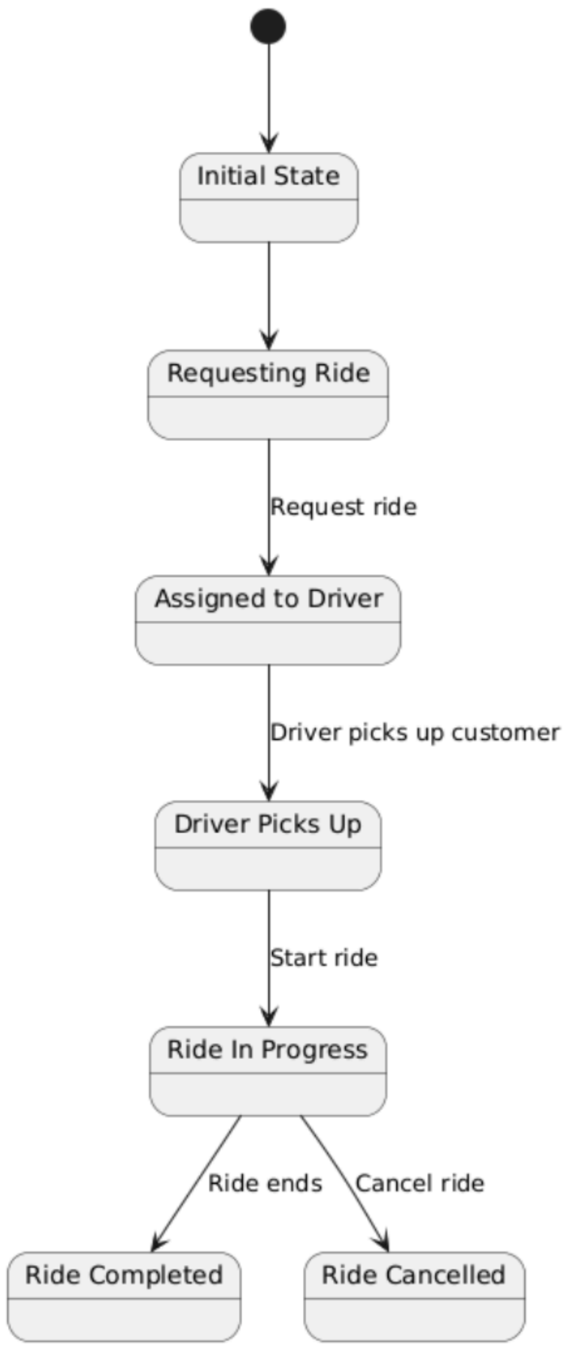
Here is the result: Application Model

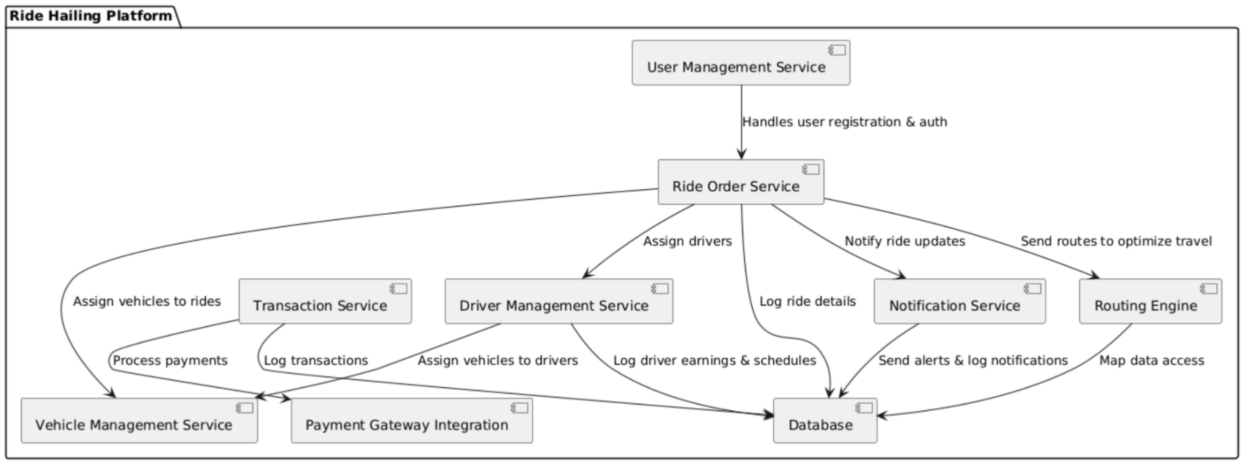
```
+--------------------------+
|     User Management     |
| Service (Login, Profile |
| management, and auth)  |
+------------+------------+
             |
+------------v------------+
| Ride Order Service       |
| (Ordering/assign rides)  |
+------------+------------+
             |
+------------v------------+
| Vehicle Management      |
| Service (Car, Scooter,  |
| battery tracking, etc.) |
+------------+------------+
             |
+------------v------------+
| Driver Management       |
| Service (Driver earnings |
| licensing and work hours)|
+------------+------------+
             |
+------------v------------+
| Transaction Service      |
| (Payments - Cash/Card/  |
| Subscription methods)   |
+------------+------------+
             |
+------------v------------+
| Notification Service    |
| (Real-time alerts)      |
+------------+------------+
```

State Chart:

```
      [Initial State]
            |
            v
  +------------------+
  | Requesting Ride  |
  +------------------+
            |
            v
  +------------------+
  | Assigned to Driver|
  +------------------+
            |
            v
  +------------------+
  | Driver Picks Up  |
  +------------------+
        |
        v
+------------------+
| Ride In Progress  |
+------------------+
        |
        v
+------------------+
| Ride Completed    |
+------------------+
        |
        v
+------------------+
| Ride Cancelled    |
+------------------+
```

For curiosity we also asked ChatGPT to write plantuml code, because we thought maybe it will generate better model with code


Initial State
Requesting Ride
Request ride
Assigned to Driver
Driver picks up customer
Driver Picks Up
Start ride
Ride In Progress
Ride ends
Cancel ride
Ride Completed
Ride Cancelled



Above are plantuml models

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?

As evident from the models, the ones created by us are much more detailed, and go to details that ChatGPT wouldn't think of going to. For example in state chart there are no guards, activities, composite state or history. The model is very surface level despite the fact that we provided a very detailed description of our model. So no, it was not possible to achieve the same result as us. We think the only strength of GenAI tools for tasks like this is when you know exactly what you wanna draw but just need some help to automate the process, or when you forgot something syntax related. But definitely not to build something from scratch. Otherwise it is not really useful. It doesn't understand how modeling works enough to help with tasks like these we think.

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

Not really

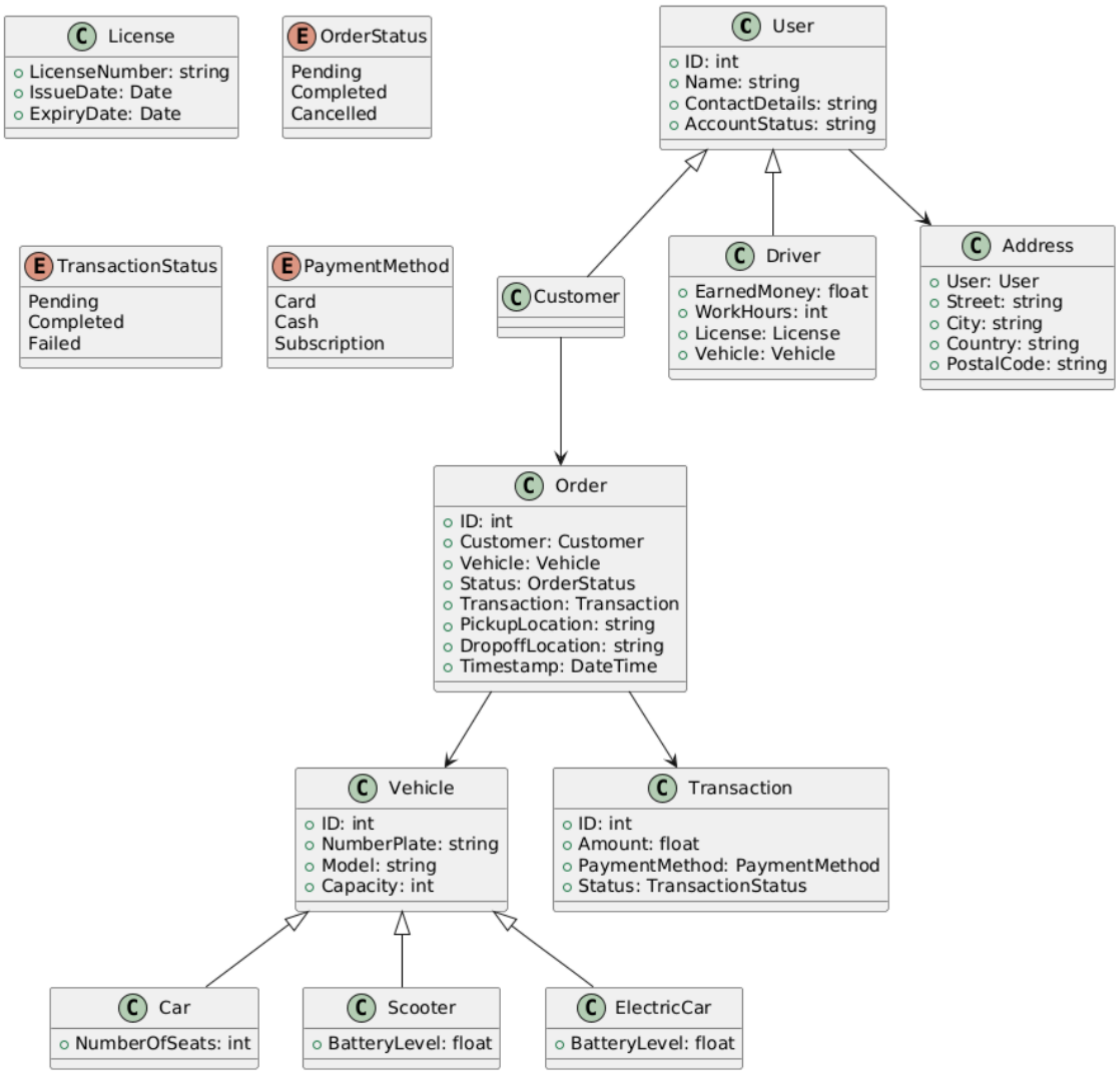
(*optional (+up to 5 bonus points)): if more than one tool has been tried and critically assessed

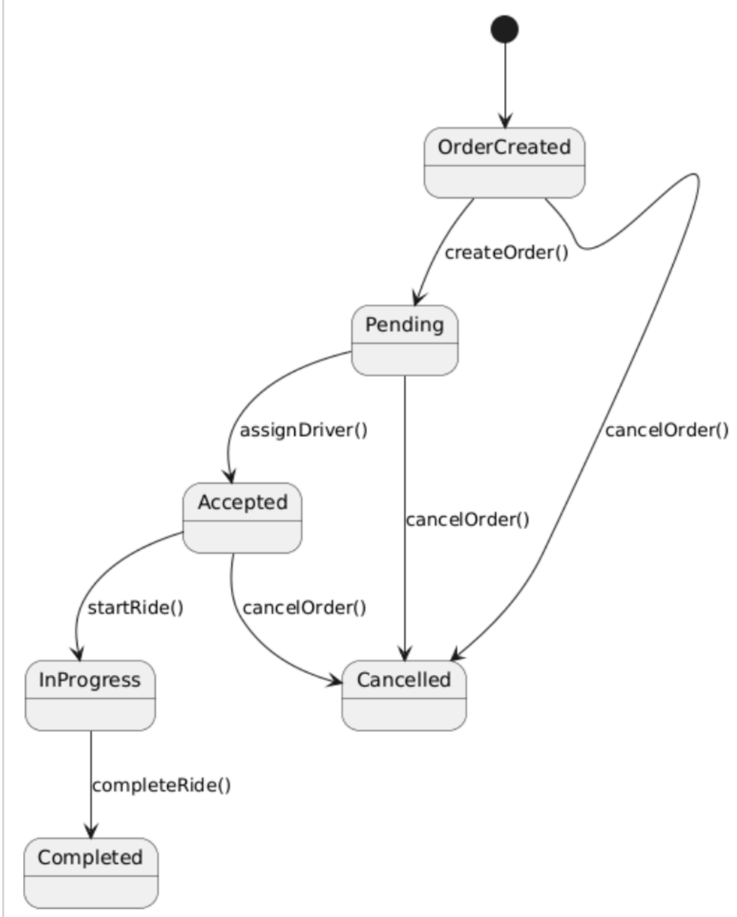
Copilot seems to be better in things related to code, so we decided to test it out as well. The prompt given to CoPilos is the same as the one given to ChatGPT.

Results are on the next page, and below are our thoughts:

Copilot did a much better job in drawing application diagram, we think it could be used as a great starting point but again needs much more details, hence the one created by us is still better. And for the State Chart it wasn't so good either because again misses quite a lot of details, and wouldn't even be a good starting point. Copilot only worked with plantuml, otherwise couldn't draw anything. But the Application Model was surprisingly fine as a starting point.

(NEXT PAGE)


License
LicenseNumber: string
IssueDate: Date
ExpiryDate: Date
OrderStatus
Pending
Completed
Cancelled
User
ID: int
Name: string
ContactDetails: string
AccountStatus: string
TransactionStatus
Pending
Completed
Failed
PaymentMethod
Card
Cash
Subscription
Customer
Driver
EarnedMoney: float
WorkHours: int
License: License
Vehicle: Vehicle
Address
User: User
Street: string
City: string
Country: string
PostalCode: string
Order
ID: int
Customer: Customer
Vehicle: Vehicle
Status: OrderStatus
Transaction: Transaction
PickupLocation: string
DropoffLocation: string
Timestamp: DateTime
Vehicle
ID: int
NumberPlate: string
Model: string
Capacity: int
Transaction
ID: int
Amount: float
PaymentMethod: PaymentMethod
Status: TransactionStatus
Car
NumberOfSeats: int
Scooter
BatteryLevel: float
ElectricCar
BatteryLevel: float


OrderCreated
createOrder()
cancelOrder()
Pending
assignDriver()
cancelOrder()
Accepted
startRide()
cancelOrder()
InProgress
Cancelled
completeRide()
Completed