NTNU
Kunnskap for en bedre verden

Department of Information Security
and Communication Technology

TTM4115 - Design of Communicating Systems

System Specification 3

Authors:
Ana Barrera Novas
Bendik Vereide
Einar Therkelsen
Ingeborg Lundamo Lien
Jan Havlas
Sverre Nystad

System code can be found at:
https://github.com/SverreNystad/communication-systems.git

Table of Contents

List of Figures

List of Tables

1 Requirements

1.1 Problem and Background

As cities continue to grow, the demand for efficient, sustainable, and accessible transportation is more pressing than ever. E-scooters have become a popular and eco-friendly mobility option, initially designed for personal use but later expanded into shared services. Since their introduction in 2018, these shared e-scooter systems have provided a convenient alternative to traditional transport (Robinson. Melia, 2018). However, their widespread use has also introduced several urban challenges that must be addressed. One of the primary issues is visual pollution and safety hazards caused by improperly parked scooters. In many cities, scooters are left in disorderly piles, blocking sidewalks, bike lanes, and wheelchair-accessible paths. This creates safety risks and contributes to frustration among pedestrians and local authorities. Additionally, vandalism has become a recurring problem, with damaged scooters often discarded irresponsibly. Another major concern is the uneven distribution of e-scooters, as they tend to be concentrated in high-traffic areas, making them less available in other parts of the city. Effective fleet redistribution strategies are needed to ensure fair access for all users. Operational inefficiencies such as theft, poor maintenance, and lack of system optimization further reduce service reliability and impact the sustainability of e-scooter businesses.

This has become quite popular, but it has also brought some challenges. The biggest issues are visual pollution and accessibility. Visual pollution happens when e-scooters are left in inconvenient places. In cities where they are widely used, they are often scattered everywhere, piled up in city squares, blocking wheelchair ramps, or lying on sidewalks and bike lanes. This creates safety risks and frustration among residents. In some cities, vandalism has also become a problem, with people damaging or even throwing e-scooters into rivers.

Accessibility is another issue, as e-scooters tend to be concentrated in busy areas, making them less available in the outskirts of the city. It is important to find a good balance between serving people in crowded areas and ensuring access for those in less central locations.

1.2 Vision

To address these challenges, we propose the development of an improved e-scooter system that enhances accessibility while reducing urban clutter. Our system will provide both users and administrators with a user-friendly platform, including a mobile application for seamless interaction. The system will feature secure booking and payment processing, and smart parking incentives that encourage responsible scooter placement.

To improve availability, scooters will be redistributed during low-traffic hours based on past usage patterns, weather conditions, and demand trends. Administrators will be able to monitor, manage, and disable scooters when necessary, with all data handled securely in the cloud to ensure smooth operation and prevent misuse.

2 System Objectives

The system aims to achieve the following objectives, each with measurable success criteria:

- Ensure secure payment processing by preventing transaction failures.
- Provide a smooth and efficient user experience with an intuitive interface.
- Increase scooter availability by minimizing inactive scooters.
- Reduce clutter in public spaces by encouraging responsible parking.

- Provide the user with easy access to the scooter information required

These objectives will be verified through data-driven metrics, including transaction error rates, scooter loss reports, user satisfaction scores, inactivity percentages, and complaint levels from pedestrians and city officials.

3 Stakeholders

Stakeholders	**Role**	**Values**	**Interests**	**Constraints**
End User - rider	Individuals using the scooters	Convenience, affordability, accessibility	Easy booking, availability, seamless payment	Limited battery life, parking restrictions, weather conditions, high prices
Scooter operators	Managing and operating the scooter fleet, running the business	Maintenance, easy access, availability, profitability, system efficiency	Scooters are easy to find, cost reduction, growth	Theft, vandalism, rebalancing challenges, local regulations, market competition, operational costs
Developers	Design, creation, and maintenance of software	Code quality, security, system stability	Well-structured, scalable, and maintainable system	Technical limitations, evolving user needs
Customer support	Handling user inquiries and issues	Responsiveness, problem-solving	Efficient tools to manage user complaints and technical issues	High volume of requests, communication barriers
City authorities	Regulating transportation and public access	Safety, order, environmental impact	Reducing clutter, enforcing regulations, ensuring public safety	Bureaucracy, balancing multiple stakeholders' interests

Table 1: Stakeholders

4 Use Cases

4.1 Use Case Sea-Level Overview

At first, to approach the use cases from the feedback given to the System Spec V1, we decided to define what we thought were the Minimum Viable Products (MVPs) and leave the rest of the features for future implementation in case we have the time to do it. We wrote the MVPs in red and left the rest in black. But in the end we implemented every use case.

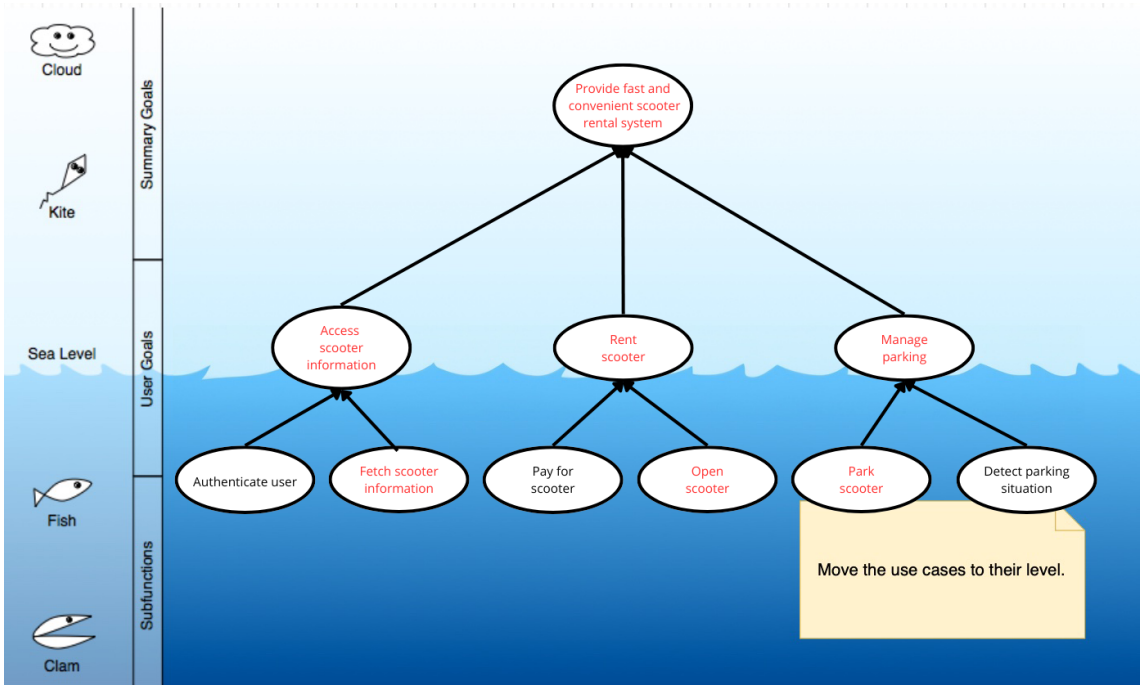

Figure 1: Use Case Sea-Level Hierarchy

4.2 Use Cases in Table Form

The use cases described below are the sea-level ones, already modified from the feedback of System Specification V1 and V2:

ID and Name:	**UC-1 Access Scooter Information**
Created By:	Team 05
Date Created:	11/03/2025
Primary Actor:	End User
Secondary Actors:	Scooter operators, Maintenance team, City authorities
Description:	The user accesses the application as a user or as an admin and can access the information of the scooter
Trigger:	The user accesses the application
Preconditions:	N/A
Postconditions:	POST-1 Information is displayed
Normal Flow:	**1.0 Authenticated Scooter Information** 1. The user accesses the "Log in" interface 2. The user authenticates themselves with their account 3. The system displays options of use cases to the user/admin 4. The user selects to request scooter information 5. The system displays scooter information
Alternative Flows:	N/A
Exceptions	**1.2 Failed Authentication** 1. User accesses the "Log in" interface 2. User inserts invalid credentials 3. System displays an error
Priority	High
Frequency of use	Approximately 5 times per week per application user
Business Rules	N/A
Other Information	N/A
Assumptions	N/A

Table 2: Use Case 1

ID and Name:	**UC-2 Rent Scooter**
Created By:	Team 05
Date Created:	11/03/2025
Primary Actor:	End User
Secondary Actors:	Scooter operators, Maintenance team
Description:	The user performs the payment for a ride and unlocks a scooter with its id.
Trigger:	The user purchases a ride
Preconditions:	PRE-1 The user is authenticated PRE-2 The scooter is available and locked
Postconditions:	POST-2 The selected scooter is unlocked and unavailable for other users
Normal Flow:	**2.0 Rent Scooter** 1. The user selects the scooter and performs the payment 2. The system performs the payment and confirms it 3. The system unlocks the scooter and notifies the user
Alternative Flows:	N/A
Exceptions	**2.1 Failed Payment** 1. The user selects the scooter and performs the payment 2. The system rejects the payment and notifies the user
Priority	High
Frequency of use	Approximately 5 times per week per application user
Business Rules	N/A
Other Information	N/A
Assumptions	N/A

Table 3: Use Case 2

ID and Name:	**UC-3 Manage parking**
Created By:	Team 05
Date Created:	11/03/2025
Primary Actor:	End User
Secondary Actors:	Scooter operators, Maintenance team
Description:	The user wants to finish the ride and locks the scooter if it is correctly parked
Trigger:	The user selects to park the scooter
Preconditions:	PRE-2 The user is authenticated PRE-3 The scooter is unlocked
Postconditions:	POST-3 The selected scooter is locked and available for renting again
Normal Flow:	**3.0 Park Scooter** 1. The user stops the scooter 2. The users selects the "parking" option 3. The system detects correct parking 4. The system locks the scooter and notifies the user
Alternative Flows:	N/A
Exceptions	**3.1 Improperly Park Scooter** 1. The user stops the scooter 2. The user selects the "parking" option 3. The system detects the incorrect parking 4. The system notifies the user to reposition the scooter
Priority	High
Frequency of use	Approximately 5 times per week per application user
Business Rules	N/A
Other Information	N/A
Assumptions	N/A

Table 4: Use Case 3

5 Architecture

5.1 Deployment Diagram

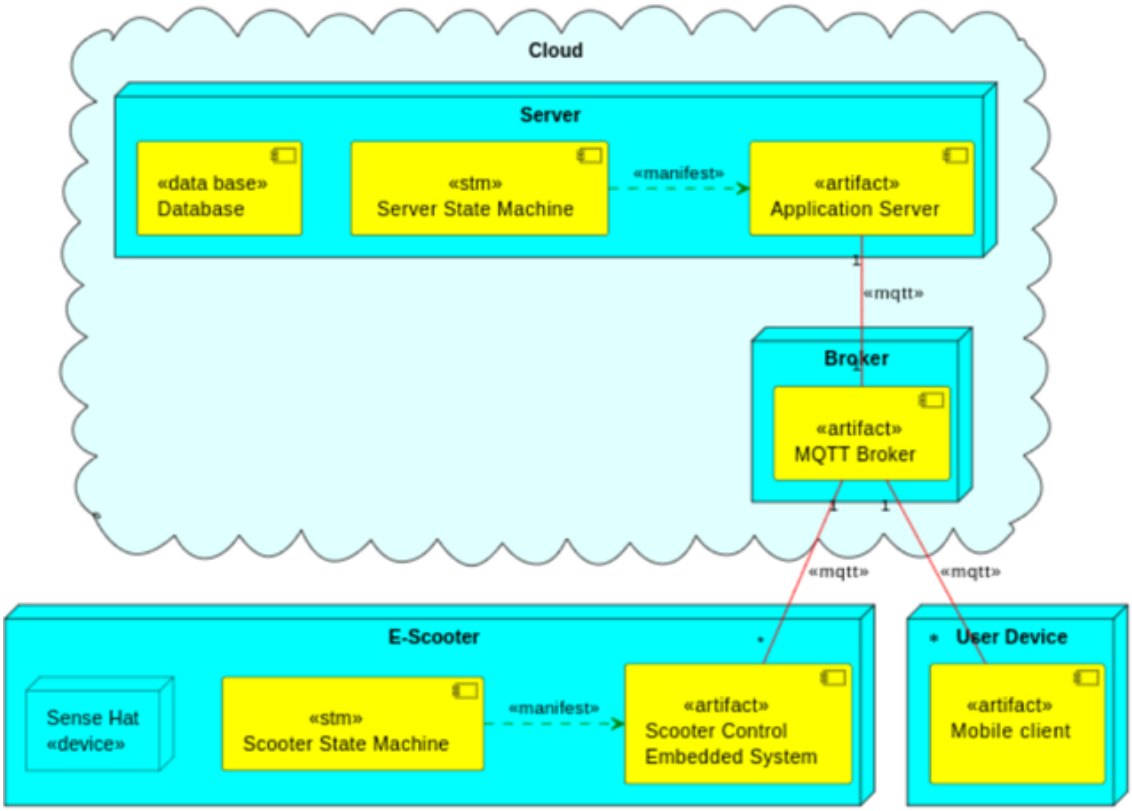

Figure 2: Deployment Diagram

5.2 Sequence Diagram

This section presents the key sequence diagrams for the e-scooter system, covering the retrieval of scooter information, scooter rental, and parking/locking management. Each diagram highlights the core interactions and alternative flows. All of the following systems communicate using MQTT. This makes the components decoupled.

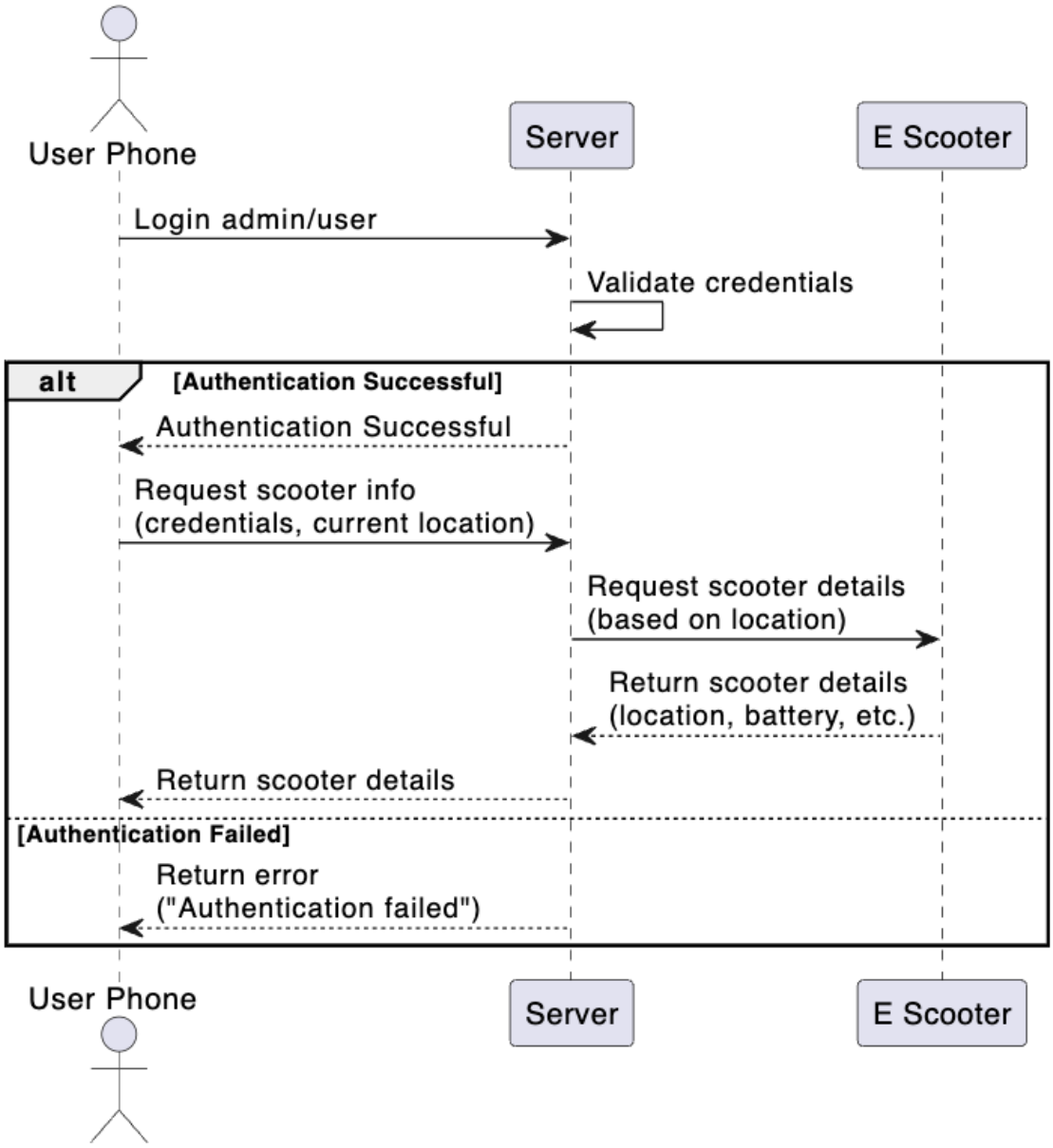

Figure 3: Accessing scooter information process

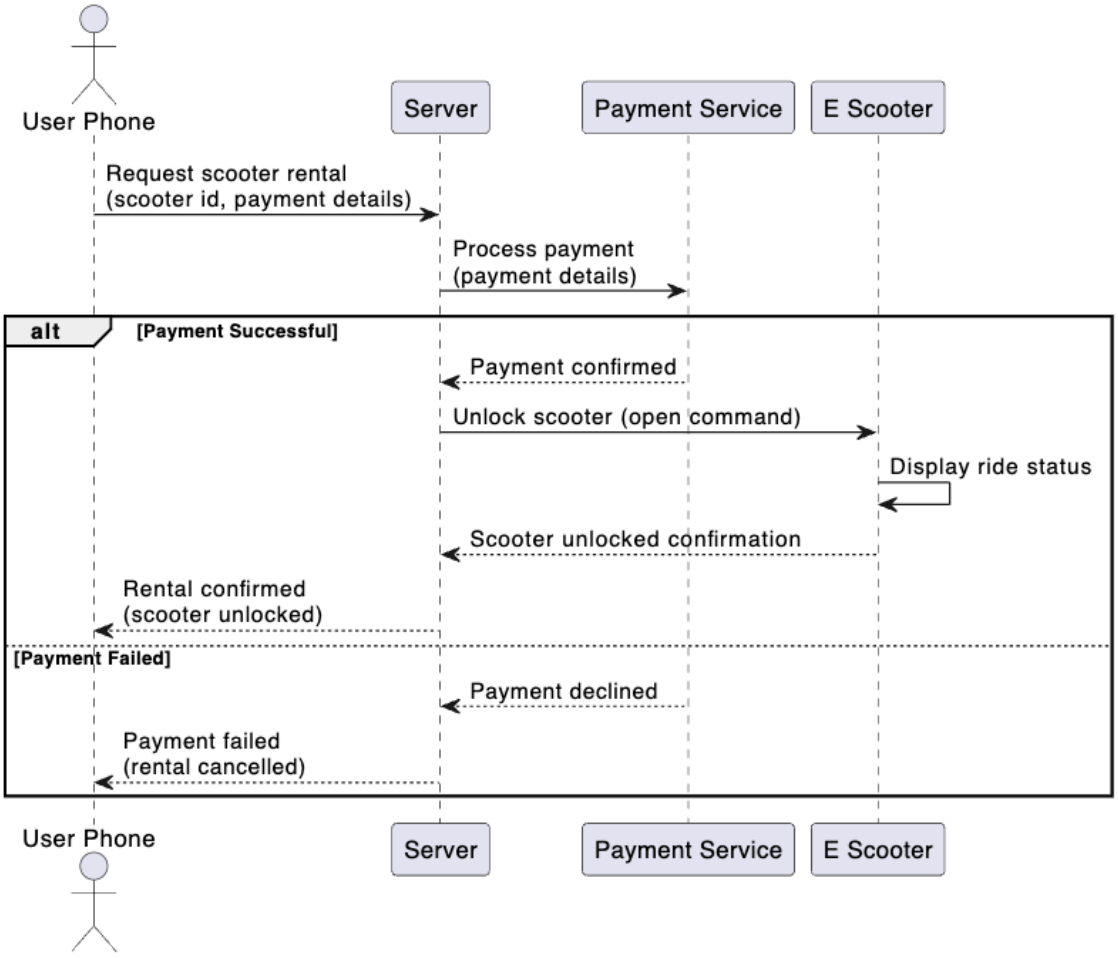

Figure 4: Scooter Rental Process

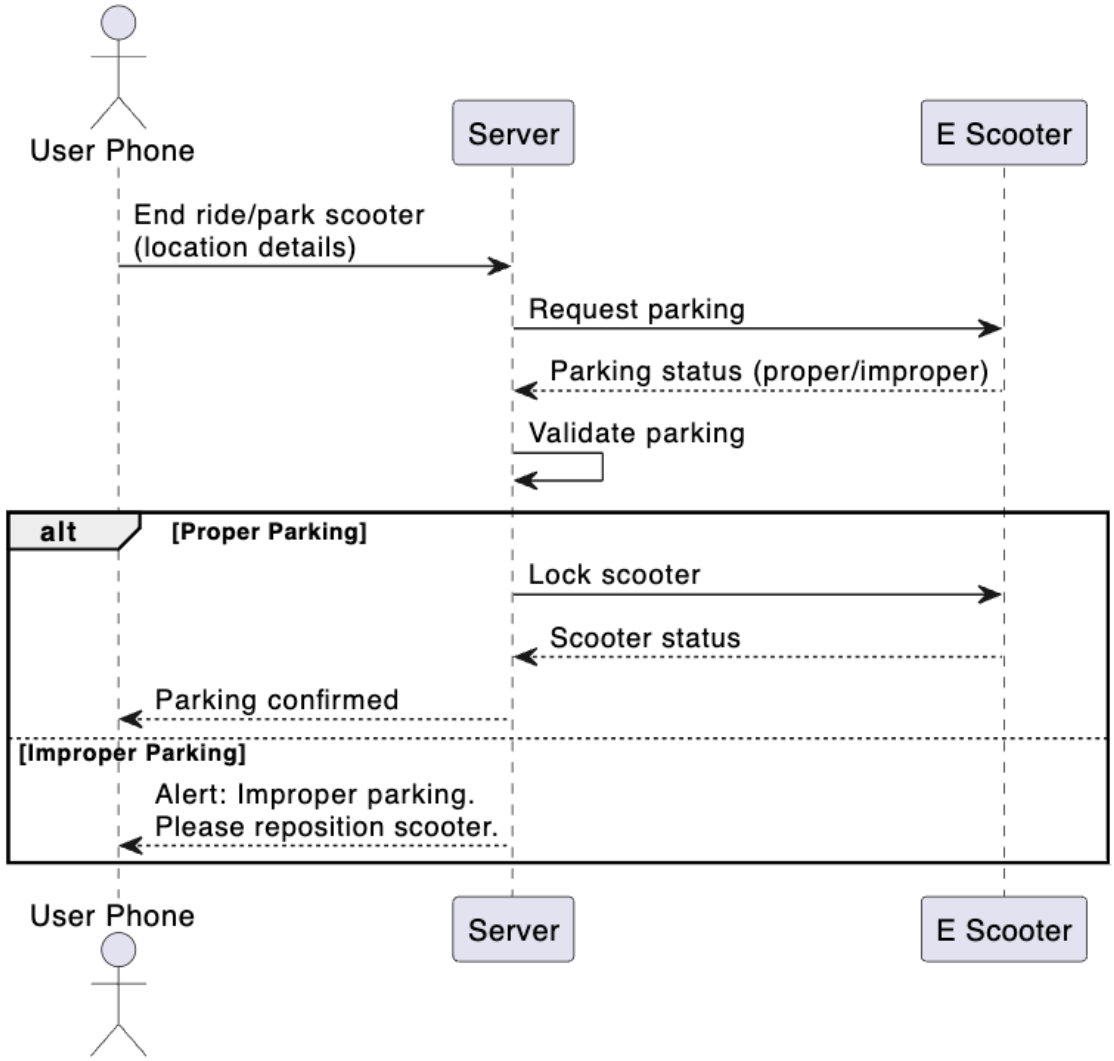

Figure 5: Parking and Locking Process

5.3 State Machines

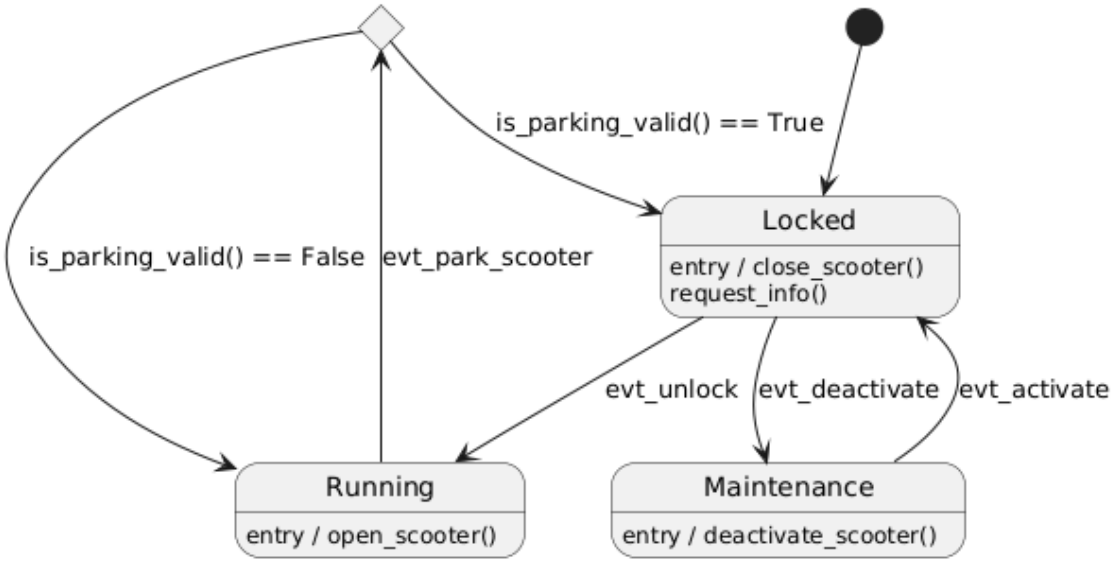

Figure 6: Scooter state machine

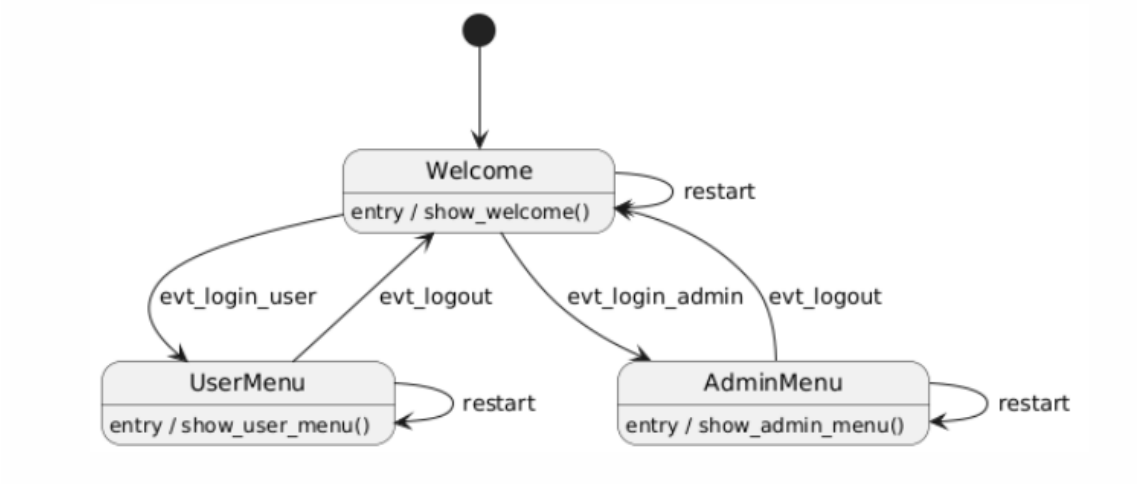

Figure 7: App state machine

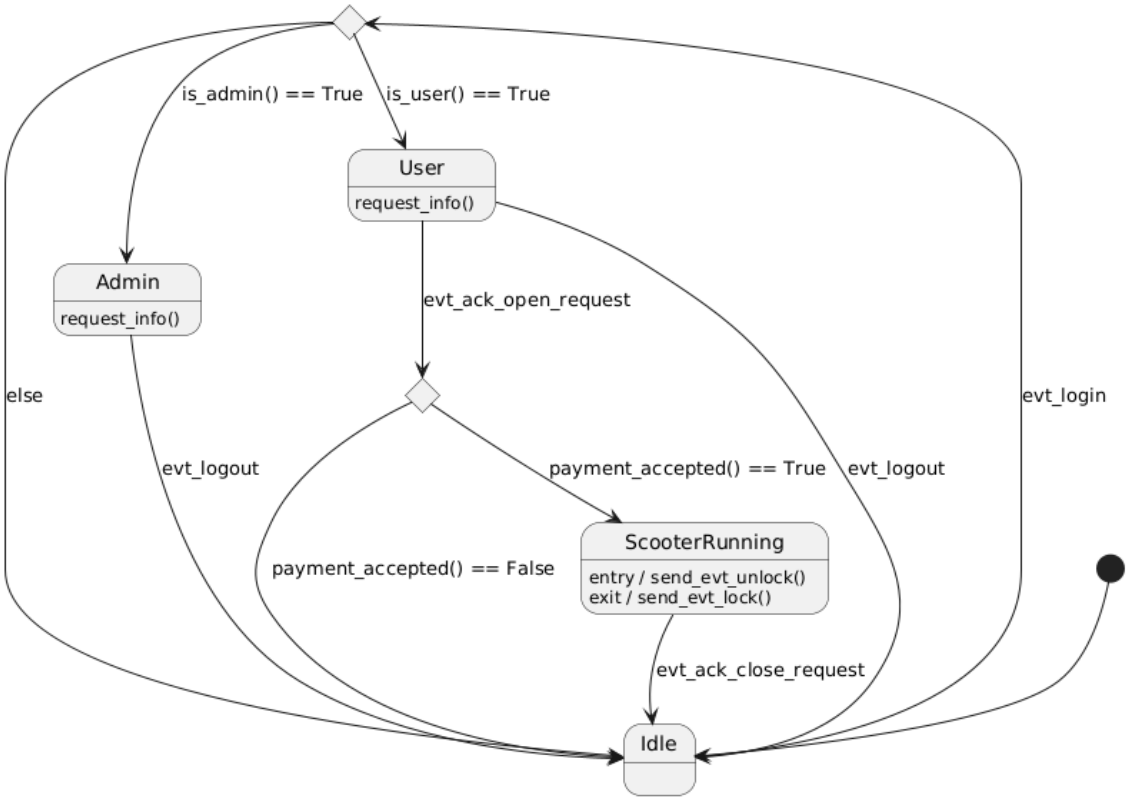

Figure 8: Server state machine

6 Implementation Comments

In our prototype, we successfully implemented the full set of use cases (scooter information retrieval, rental/payment flow, parking validation, and administrative controls) using a clear, modular architecture following explicit state machines. Early on, we attempted to follow the initial specification literally, intertwining UI logic, business rules, and messaging behavior. While that approach worked, it was slow and unsteady, as not all the group members had full understanding on the flow and code.

To regain clarity and maintainability, we refactored around improved state machines for each component (App UI, Server, and Scooter Service). By capturing each actor's lifecycle in its own machine—with well-defined states, transitions, entry/exit actions, and guards—we achieved a high level of decoupling:

- The UI simply sends commands over MQTT.
- The server mediates authentication, (simple) payment checks, and relays acknowledgments.
- The scooter service handles sensor-based parking validation, maintenance mode, and Sense HAT display feedback.

This separation made it straightforward to implement every use case end-to-end, to trace message flows in our sequence diagrams, and to simulate failures like random payment rejection without polluting core logic.

In addition, we containerized each component using Docker, parameterizing the MQTT broker's IP address and port via environment variables to support flexible deployment. For the message broker, we used Mosquito, that we configured. We also adopted strict dependency versioning (in our `requirements.txt` and Dockerfiles) to guarantee reproducible builds and simplify future deployments.

We are most proud of our robust state-machine-driven behavior, that ensures extense coverage and eases future extensibility; our clean MQTT integration with the state machines, messages and topics, replaced magic strings for event names and payload keys with Python `StrEnum` types and `@dataclass` models, improving code readability and reducing runtime errors; and our Sense Hat working properly and with all functionality at the end, after many configuration problems.

Overall, through many iterations and refactors from the original specification and refocusing on the improved state-machine design we delivered a maintainable, testable implementation that realizes every use case on a simple and well-organized implementation.