Ali# 2

Use cases for a parking system

Master of Information and Communications Technology

Database Design and Administration

Hamdi Ali

University of Denver University College

Date: 09/20/2025

Faculty: Nirav Shah, M.S , MBA

Director: Cathie Wilson, M.S

Dean: Michael J. McGuire, MLS

**Use case diagram:**

**Actors:**

Customer: registers, owns permits, parks cars, and pays bills.

Parking Office: manages registrations, billing, and permits.

Parking Lot System: scans permits on entry/exit, applies charges.

System Admin: maintains pricing structures, discounts, and rules.

**Main Use Cases:**

Register customer

Purchase permit

Renew/cancel permit

Scan permit on entry

Scan permit on Exit

Calculate fee (daily/hourly)

Apply discount (20% for compact cars)

Charge customer account

Generate monthly bill

Report lost permit

**System boundary:**

All parking system use cases are contained within the "Parking System" box. The actors remain outside, connected to relevant use cases.

**Relationships:**

The purchase permit includes the charge to the customer account.

The scan permit on exit includes the calculated fee.

The calculated fee includes applying a discount (20% for compact cars).

The scan permit upon entry includes a charge to the customer's account.

A diagram of a parking system

AI-generated content may be incorrect.

**Individual use case documentation**

**Use case ID:** UC-002

**Name:** Purchase parking permit

**Short description:**

The customer purchases a parking permit for his or her vehicle. Permits are linked to a particular car, and customers can own multiple permits. The system charges different fees for compact and SUV vehicles.

**Goal:** Allow customers to park in university lots with valid parking permits.

**Preconditions:**

The customer must be registered in the parking system.

The customer's account must be active and in good standing.

**Success end condition:**

The permit has been successfully created, linked to the customer's vehicle, and activated in the system.

**Failure end condition:**

The permit cannot be created due to invalid information, payment failures, or system errors.

**Stakeholders:**

**Primary actor:** Customer

**Secondary actors:** Parking office and Parking lot system

**Trigger:** The customer initiates the purchase of the permit.

**Normal flow:**

The customer logs into the parking system.

The system prompts for vehicle details (license plate, make, model, and size).

The customer enters vehicle information.

The system validates vehicle information.

The system calculates the permit fee (discount if you drive a compact car).

The customer confirms the purchase.

The system charges the customer's account.

The system issues a digital or physical permit linked to the vehicle.

**Alternative flows:**

**Invalid information:** The system requests correction before moving forward.

**Payment failure:** The transaction has been canceled; the customer has been notified.

**Duplicate vehicle entry:** The system alerts the customer and prevents the purchase.

**Includes:**

UC-001 Register customer (if not already registered).

UC-003 Charge the customer account.

**Frequency of use:**

Whenever a new vehicle is purchased by a customer, or once per semester.

**Constraints & special requirements:**

There can only be one vehicle per permit.

The compact car discount must always be applied automatically.

The permit is valid for all parking lots at the university.

**Assumptions:**

The customer provides accurate information about the type of car (compacts or SUVs).

The billing for parking is handled outside of the Parking System on a monthly basis.

**Notes/Issues:**

It is possible that direct online payments will be included in future integrations.

A future version of the system may phase out paper permits.

**Reflection:**

This assignment gave me a deeper understanding of how use cases work as the foundation of object-oriented systems. In the beginning, it seemed straightforward to identify the most obvious actors, such as the customer and the parking office. Roles such as these interact with the system naturally by registering accounts, buying permits, and handling billing. However, as I worked through the diagram, I realized that the most challenging part was distinguishing between actions that appear similar but serve different purposes. As an example, scanning a permit on entry and scanning a permit on exit involve the same physical tag, yet the processing rules are entirely different. As a result of entry scanning, card validation and account charging may be triggered, while exit scanning requires fee calculation and possibly discount application. When defining use cases, it is important to recognize these subtle differences.

The most helpful step was to walk through the customer's journey step-by-step before transforming it into a formal diagram in plain language. Imagining myself as a customer, the sequence became clear: registration, permit purchase, driving to the lot, scanning to enter, scanning to exit, and being charged or billed. As a result of framing the system in this way, it was much easier to define the actors and use cases without overcomplicating the design. In addition, referring back to the UML examples provided in the course readings gave me reassurance that I was structuring the diagram correctly. When I saw how include relationships were used in professional examples, I was able to justify linking charging and discount processes under broader actions like "Purchase Permit" or "Scan on Exit."

My biggest regret is that I didn't consider the importance of documenting assumptions alongside the diagram earlier. For instance, I assumed the system would accept a customer's declaration of their vehicle type when calculating a compact car discount. In reality, the university might require proof or supporting documentation to prevent discounts from being misused. Assumptions like these help stakeholders understand where simplifications were made and where real-world processes differ. In addition to improving communication, it also creates a record that can be revisited if the design changes.

Another design decision I contemplated was whether the parking lot system should be modeled as an internal actor or as an external actor. As a result, I chose to represent it as an external actor since hardware devices such as permit scanners are physical components that interact with software, rather than software components themselves. This approach makes the diagram clearer and gives a better idea of how the system would operate in practice. I have also added an admin actor to account for ongoing pricing rule management. The pricing structure and discount policies are not static, so someone must maintain and update them. By including the Admin actor, this ongoing responsibility is explicitly represented.

Overall, completing this assignment reinforced the idea that use cases are more than just academic exercises; they are essential tools for bridging the gap between user needs and system design. As well as capturing the interactions that matter most to stakeholders, they also guide technical decisions about system architecture. Using actors and their interactions as the basis, I developed a foundation that will be extremely valuable as we move forward with class diagrams, sequence diagrams, and ultimately the implementation of the system. I have learned that a strong use case design organizes requirements and creates a shared language between developers, administrators, and end users.

**References:**

Object Management Group. (2017). Unified Modeling Language (UML) specification, version 2.5.1. OMG. <https://www.omg.org/spec/UML/2.5.1>

GeeksforGeeks. “Use Case Diagram – Unified Modeling Language (UML).” GeeksforGeeks. Last updated July 23, 2025. Accessed September 20, 2025. <https://www.geeksforgeeks.org/system-design/use-case-diagram/>

Visual Paradigm. “What Is Use Case Diagram?” Visual Paradigm Guide. Accessed September 20, 2025. <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>

Wrike. “What Is a Use Case and How to Write One.” Wrike Blog. Published October 24, 2024. Accessed September 20, 2025. <https://www.wrike.com/blog/what-is-a-use-case/>

UML-Diagrams.org. “UML Use Case Diagrams.” uml-diagrams.org. Accessed September 20, 2025. <https://www.uml-diagrams.org/use-case-diagrams.html>