Park Zone

Software Requirements Specifications

Group 7

Revision History

Date	Revision	Description	Author
09/19/2025	1.0	Initial Version, a combination of thoughts from Sophia and Mario. Revision by Jose.	Jose Ignacio Zaragoza
09/22/2025	1.1	More revision, comments in specific sections left for changes or removal.	Jose Ignacio Zaragoza
09/24/2025	1.2	Small changes to wording, unnecessary/out of scope sections removed	Jose Ignacio Zaragoza
09/25/2025	1.3	More formatting/wording fixes	Jose Ignacio Zaragoza

1. Purpose

ParkZone exists to eliminate the frustration of finding parking by connecting drivers with available parking spaces in real-time while maximizing revenue for parking garage operators through intelligent space management and automated billing systems.

1.1. Scope

This document defines the requirements for developing ParkZone, a comprehensive parking management system consisting of:

- **Server Application**: Backend systems managing databases, payments, reservations, and reporting
- **Client Application**: User-facing mobile/web interface for customers and garage operators
- **Hardware Integration**: Sensors, payment terminals, and access control systems

What's included:

- Real-time parking availability tracking
- Reservation and payment systems
- User profiles and rewards program
- Operator dashboards and reporting
- Multi-level garage support

What's excluded:

- Hardware manufacturing
- Physical installation services
- Integration with municipal parking systems (Phase 2)

1.2. Definitions, Acronyms, Abbreviations

- **GUI**: Graphical User Interface
- **API**: Application Programming Interface
- **SRS**: Software Requirements Specification
- **Overstay**: Parking beyond the paid/reserved time limit
- **Grace Period**: 5-minute buffer before penalties apply
- **Garage Operator**: Business owner managing parking facilities

- **End User**: Customer seeking parking
- **EV**: Electric Vehicle

1.3. References

- IEEE Std 830-1998 (Software Requirements Specifications)
- ParkZone Project Kickoff Presentation
- Team meeting notes and requirements gathering sessions
- UML Use Case and Class Diagrams.

1.4 Overview

The ParkZone application is designed for End User (driver/custormer) to reserve parking spaces using their license plate at select locations while maximizing revenue for parking Garage Operators (parking garage owner/manager). Parking availability is tracked using a list of available parking spots and is updated instantly, allowing users to reserve spaces and pay for parking. Parking garage owners benefit from the automated billing and reporting logs.

The Client Application will provide customers with a user-friendly interface to view all available spaces, allow for filtering by vehicle type (compact, large, motorcycle, or EV), make a reservation, and handle their payments securely. The Server Application will manage those reservations, process their payments, ensure the availability of parking spots is accurate, and generate a report for the parking garage owners. The Hardware Integration Layer will connect these physical payment terminals, access systems, and any sensors to accurately represent the parking garage's current state with the digital one.

Two key roles are necessary for this application's operation. The End User can search for any available spots, make reservations, pay, and earn reward points from frequent use. The Garage Operator will have the ability to oversee the parking space usage, adjust all rates, and review all reports generated.

A log will be generated for important events, such as reservations, cancellations, payments, overstays, and any operator actions. Reports that summarize revenue generated, occupancy, and violations are generated at a (daily, weekly, monthly?) basis. End Users will be notified for confirmation on their reservations, overstay alerts, or rewards earned, or any promotional offers, while the Garage Operator will be notified for any system issues.

2. Overall Description

// changes here in 2.1, 2.2?

2.1. Product Perspective

ParkZone is a new, standalone system that bridges the gap between parking supply and demand. The system operates as:

- **Customer-facing application** for finding and reserving spots
- **Operator management console** for garage owners
- **Backend infrastructure** connecting all components
- **Integration layer** for payment processors and hardware sensors

2.2. Product Architecture

ParkZone is a new, standalone system that bridges the gap between parking supply and demand. The system operates as:

- **Customer-facing application** for finding and reserving spots
- **Operator management console** for garage owners
- **Backend infrastructure** connecting all components
- **Integration layer** for payment processors and hardware sensors

2.3. Product Functionality/Features

Java Client-Server Architecture with Centralized Coordinator:

Server Application (Java) - Single-Threaded Coordinator:

- **ParkingServer** class manages ALL space updates in one thread
- Maintains master list of available/occupied spaces
- Processes all client requests sequentially (no race conditions)
- Broadcasts updates to all connected clients immediately
- File-based persistence for garage state

2.4. Constraints

Java Client-Server Architecture with Centralized Coordinator:

Server Application (Java) - Single-Threaded Coordinator:

- **ParkingServer** class manages ALL space updates in one thread
- Maintains master list of available/occupied spaces
- Processes all client requests sequentially (no race conditions)
- Broadcasts updates to all connected clients immediately
- File-based persistence for garage state

2.5. Assumptions and Dependencies

Assumptions:

- Parking garages have basic internet connectivity.
- Users have smartphones with GPS capabilities.
- Garage operators are willing to adopt new technology.
- Sufficient server infrastructure can be secured.

Dependencies:

- Third-party payment processor APIs (Stripe, PayPal).
- Cloud hosting services (AWS, Azure, or Google Cloud).
- GPS and mapping services.
- Hardware sensor integration capabilities.

3. Specific Requirements

3.1. Server-Side Core Systems

3.1.1. Database Management & Real-Time Tracking

- **FR-001**: System shall maintain a real-time inventory of all parking spaces and display “Full” at the entrance and display board.
- **FR-002**: System shall update space availability within 5 seconds of a status change
- **FR-003**: System shall store user profiles, transaction history, and garage data
- **FR-004**: System shall support multi-level garage configurations
- **FR-004**: System shall provide different parking spots for different vehicle types (compact, large, EV, motorcycle).

3.1.2. Payment Processing Backend

- **FR-005**: System shall process cash and card payments
- **FR-006**: System shall calculate fees based on duration, vehicle type, and pricing rules
- **FR-007**: System shall handle refunds for canceled reservations
- **FR-008**: System shall generate payment receipts and transaction logs
- **FR-009**: System shall have a per-hour pricing model, $5 for the first hour, $3 for the second and third, and $2 for the remaining hours.

3.1.3. Reservation Logic & Validation

- **FR-010**: System shall allow users to reserve spaces up to 24 hours in advance
- **FR-011**: System shall release reservations after a 5-minute grace period
- **FR-012**: System shall prevent double-booking of spaces
- **FR-013**: System shall send confirmation notifications for reservations
- **FR-014**: System shall support compact, large, EV, and motorcycles.

3.1.4. Report Generation Engine

- **FR-014**: System shall generate daily/weekly/monthly usage reports
- **FR-015**: System shall track revenue, occupancy rates, and user behavior
- **FR-016**: System shall identify overstay incidents and violations
- **FR-017**: System shall export reports in PDF and CSV formats

3.2. Client Application Interface

3.2.1 Parking Space Selector GUI

- **FR-018:** Interface shall display available spaces in real-time
- **FR-019:** Interface shall allow filtering by accessibility and EV charging
- **FR-020:** Interface shall show pricing information before selection
- **FR-021:** Interface shall provide a garage layout visualization

3.2.2 Payment Interface & Forms

- **FR-022:** Interface shall support multiple payment methods
- **FR-023:** Interface shall calculate total costs before payment
- **FR-024:** Interface shall store preferred payment methods securely
- **FR-025:** Interface shall handle payment failures gracefully

3.2.3 Reservation Booking Interface

- **FR-026**: Interface shall allow time-based reservations
- **FR-027**: Interface shall show reservation confirmations
- **FR-028**: Interface shall allow reservation modifications/cancellations
- **FR-029**: Interface shall display countdown timers for reserved spaces

3.2.4 User Dashboard & Profile Management

- **FR-030**: System shall maintain user profiles with vehicle information
- **FR-031**: System shall display parking history and receipts
- **FR-032**: System shall show rewards points and available discounts
- **FR-033**: System shall allow users to report issues and violations

3.3. System Integration Requirements

- **FR-034**: System shall use HTTPS for all client-server communication
- **FR-035**: System shall implement WebSocket connections for real-time updates
- **FR-036**: System shall provide RESTful APIs for all operations
- **FR-037**: System shall handle network interruptions gracefully

3.3.1 Server-Client Communication Protocols

- **FR-038**: System shall sync data across all client instances within 10 seconds
- **FR-039**: System shall maintain data consistency during high-traffic periods
- **FR-040**: System shall queue updates during temporary network outages

- **FR-041**: System shall resolve data conflicts using timestamp precedence

3.3.2 Real-Time Data Synchronization

- **FR-042**: System shall integrate with payment gateway APIs
- **FR-043**: System shall connect with garage sensor hardware
- **FR-044**: System shall interface with access control systems
- **FR-045**: System shall support third-party mapping services
- **FR-046**: System shall have floor display boards with the number of free spots by type.

3.3.3 External System Interfaces

- **FR-047**: System shall send push notifications for reservations
- **FR-048**: System shall alert users of overstay situations
- **FR-049**: System shall notify operators of system issues
- **FR-050**: System shall send promotional offers to frequent users
- **FR-051**: System shall provide charging and pay panels for EVs.

3.4 User Experience Features

3.4.2 Rewards & Gamification Elements

- **FR-052**: System shall award points based on usage duration
- **FR-053**: System shall allow points redemption for free parking
- **FR-054**: System shall provide loyalty tier benefits
- **FR-055**: System shall track and display usage statistics

3.4.2 Ticket Generation

- **FR-056**: System shall allow the use of credit/debit cards for payment at the time of reservation
- **FR-057**: System shall use the license plate entered during the time of reservation as part of the ticket ID
- **FR-058:** System shall use cameras to scan the license plate to ensure the spot is in use by the right car.

Notes:
Another parking space reservation application i've seen, shows lots, you cant reserve a specific spot at any lot, but you pay for it after your stay. You type in the amount of time you stayed

before you leave or when you you arrive. QR codes available physically in the lots for easy payments, the time slot input and your car’s plate number. Assuming they photograph your plate for security if you dont pay.

Implementing this basically, reformat again, reword based on new info tie in all members stuff, remove old info, recheck for repeats!

Reformatted and worded a lot of stuff, next is sophias

4. Non-Functional Requirements

4.1.1 Authentication and Authorization

- **NFR-001**: System shall require strong password policies (8+ characters, mixed case, numbers)
- **NFR-002**: System shall implement multi-factor authentication for operators
- **NFR-003**: System shall use role-based access control (customer, operator, admin)
- **NFR-004**: System shall automatically log out inactive sessions after 30 minutes

4.1.2 System Security

- **NFR-005**: System shall log all security-relevant events
- **NFR-006**: System shall implement rate limiting to prevent abuse
- **NFR-007**: System shall use HTTPS/TLS 1.3 for all communications
- **NFR-008**: System shall regularly update and patch security vulnerabilities

4.2. Environmental Requirements

4.2.1 Hardware Environment

- **NFR-009**: Server shall run on cloud infrastructure (AWS/Azure/GCP)
- **NFR-010**: System shall support both portrait and landscape orientations

4.2.2 Software Environment

- **NFR-011**: Server shall be compatible with Linux-based operating systems
- **NFR-012**: Database shall use PostgreSQL or equivalent relational database
- **NFR-013**: System shall support modern web browsers (Chrome 90+, Safari 14+, Firefox 88+)
- **NFR-014**: System shall be deployment-ready using containerization (Docker)

4.2.3 Integration Environment

- **NFR-015**: System shall integrate with existing garage management systems
- **NFR-016**: System shall support common sensor protocols (REST APIs, MQTT)
- **NFR-017**: System shall work with standard payment terminals
- **NFR-018**: System shall accommodate various garage layouts and configurations

4.2.4Parking Garage Layout

- **NFR-019**: Multiple floors should be available to customers.
- **NFR-020**: Multiple entry and exit points to avoid traffic jams in the parking garage.

4.3. Performance Requirements

4.3.1 Response Time

- **NFR-021**: System shall respond to user actions within 2 seconds under normal load
- **NFR-022**: Real-time updates shall propagate within 5 seconds
- **NFR-023**: Payment processing shall be completed within 10 seconds
- **NFR-024**: Report generation shall complete within 30 seconds for standard reports

4.3.2 Throughput and Scalability

- **NFR-025**: System shall support minimum 1,000 concurrent users
- **NFR-026**: System shall handle 10,000+ parking transactions per day
- **NFR-027**: Database shall support horizontal scaling for growth
- **NFR-028**: System shall maintain performance during peak usage (weekends, events)

4.3.4 Availability and Reliability

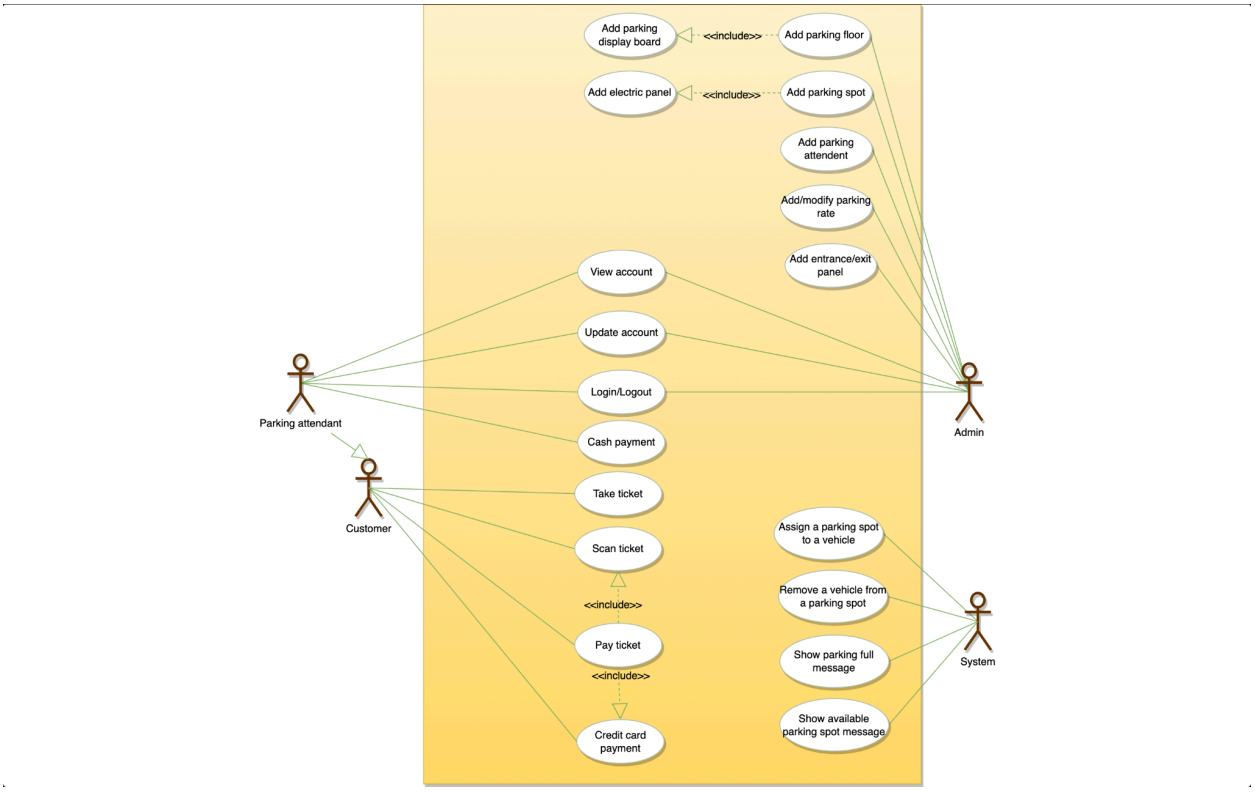
- **NFR-029**: System shall maintain 99.5% uptime during business hours
- **NFR-030**: System shall implement automatic failover for critical components
- **NFR-031**: System shall perform automated daily backups
- **NFR-032**: System shall recover from failures within 15 minutes

4.3.5 Resource Usage

- **NFR-033**: Server shall efficiently handle database queries without performance degradation
- **NFR-034**: System shall optimize bandwidth usage for real-time updates

5. Relevant Diagrams

Use case diagram:



ParkZone Main Actors:

- **Admin:** Mainly responsible for adding and modifying parking floors, parking spots, entrance, and exit panels, adding/removing parking attendants, etc.
- **End User:** All End Users can get a parking ticket and pay for it.
- **Parking attendant:** Parking attendants can do all the activities on the customer's behalf, and can take cash for ticket payment.
- **System:** To display messages on different info panels, as well as assign and remove a vehicle from a parking spot.

Use Cases for Parking Lot

Use Cases for Parking Lot:

- **Add/Remove/Edit parking floor:** To add, remove or modify a parking floor from the system. Each floor can have its own display board to show free parking spots.

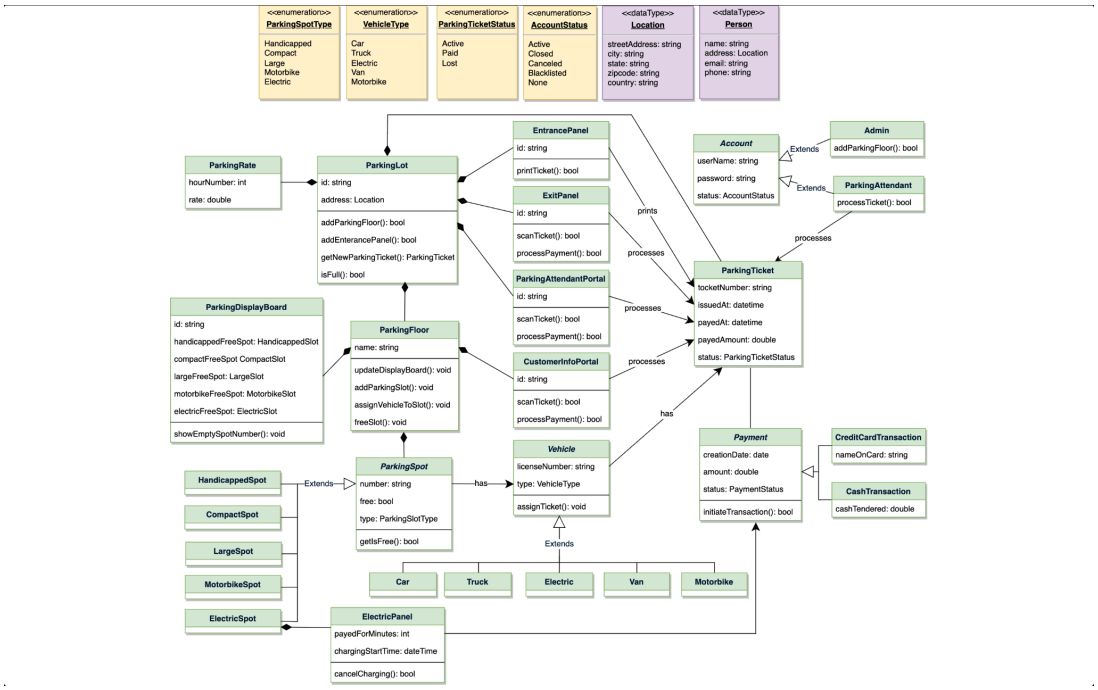
- **Add/Remove/Edit parking spot:** To add, remove or modify a parking spot on a parking floor.
- **Add/Remove a parking attendant:** To add or remove a parking attendant from the system.
- **Take ticket:** To provide customers with a new parking ticket when entering the parking lot.
- **Scan ticket:** To scan a ticket to find out the total charge.
- **Credit card payment:** To pay the ticket fee with a credit card.
- **Cash payment:** To pay the parking ticket with cash.
- **Add/Modify parking rate:** To allow admin to add or modify the hourly parking rate.

Class diagram (UML)

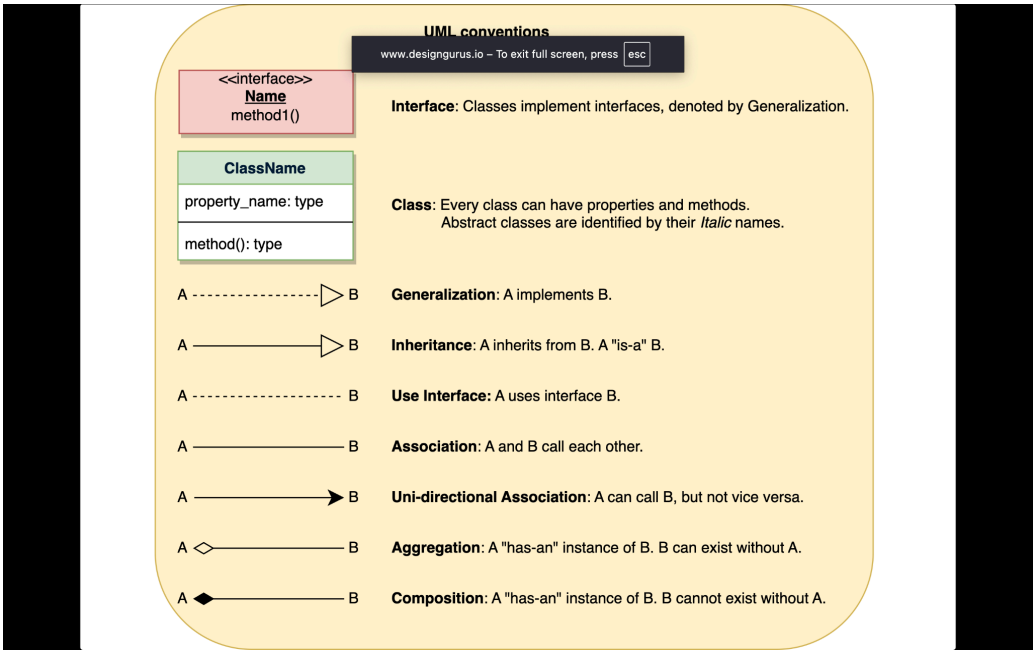
Main Classes for the Parking Lot System:

- **ParkingLot:** The central part of the organization for which this software has been designed. It has attributes like 'Name' to distinguish it from any other parking lots and 'Address' to define its location.
- **ParkingFloor:** The parking lot will have many parking floors.
- **ParkingSpot:** Each parking floor will have many parking spots. Our system will support different parking spots
    1. Handicapped
    2. Compact
    3. Large
    4. Motorcycle
    5. Electric.
- **Account:** We will have two types of accounts in the system: one for an Admin, and the other for a parking attendant.
- **Parking ticket:** This class will encapsulate a parking ticket. Customers will take a ticket when they enter the parking lot.
- **Vehicle:** Vehicles will be parked in the parking spots. Our system will support different types of vehicles
    1. Car
    2. Truck
    3. Electric
    4. Van
    5. Motorcycle.
- **EntrancePanel and ExitPanel:** EntrancePanel will print tickets, and ExitPanel will facilitate payment of the ticket fee.

- **Payment:** This class will be responsible for making payments. The system will support credit card and cash transactions.
- **ParkingRate:** This class will keep track of the hourly parking rates. It will specify a dollar amount for each hour. For example, for a two hour parking ticket, this class will define the cost for the first and the second hour.
- **ParkingDisplayBoard:** Each parking floor will have a display board to show available parking spots for each spot type. This class will be responsible for displaying the latest availability of free parking spots to the customers.
- **ParkingAttendantPortal:** This class will encapsulate all the operations that an attendant can perform, like scanning tickets and processing payments.
- **CustomerInfoPortal:** This class will encapsulate the info portal that customers use to pay for the parking ticket. Once paid, the info portal will update the ticket to keep track of the payment.
- **ElectricPanel:** Customers will use the electric panels to pay and charge their electric vehicles.



This one tells you what the arrows mean:



Customer paying for parking ticket: Any customer can perform this activity. Here are the set of steps: