**Software Requirements**

**Specification**

**for**

**ParkSense**

**Version 1.0 approved**

**Prepared by Landicho, John Jorel**

**Libudan, Rchie**

**Sayson, Timothy Jay**

**Tauyan, Angela Mae**

**ImPossible**

**November 27, 2024**

***Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.***

# Table of Contents

[Table of Contents 2](#_Toc184901269)

[1. Introduction 1](#_Toc184901270)

[1.1 Purpose 1](#_Toc184901271)

[1.2 Document Conventions 1](#_Toc184901272)

[1.3 Intended Audience and Reading Suggestions 1](#_Toc184901273)

[1.4 Product Scope 2](#_Toc184901274)

[1.5 References 3](#_Toc184901275)

[2. Overall Description 4](#_Toc184901276)

[2.1 Product Perspective 4](#_Toc184901277)

[2.2 Product Functions 5](#_Toc184901278)

[2.3 User Classes and Characteristics 5](#_Toc184901279)

[2.4 Operating Environment 6](#_Toc184901280)

[2.5 Design and Implementation Constraints 7](#_Toc184901281)

[2.6 User Documentation 7](#_Toc184901282)

[2.7 Assumptions and Dependencies 8](#_Toc184901283)

[3. External Interface Requirements 9](#_Toc184901284)

[3.1 User Interfaces 9](#_Toc184901285)

[3.2 Hardware Interfaces 12](#_Toc184901286)

[3.3 Software Interfaces 12](#_Toc184901287)

[3.4 Communications Interfaces 13](#_Toc184901288)

[4. System Features 14](#_Toc184901289)

[4.1 Parking Slot Monitoring System 14](#_Toc184901290)

[4.2 Parking Analytics Dashboard 14](#_Toc184901291)

[4.3 Parking Analytics Dashboard 15](#_Toc184901292)

[5. Other Nonfunctional Requirements 16](#_Toc184901293)

[5.1 Performance Requirements 16](#_Toc184901294)

[5.2 Safety Requirements 17](#_Toc184901295)

[5.3 Security Requirements 17](#_Toc184901296)

[5.4 Software Quality Attributes 17](#_Toc184901297)

[5.5 Business Rules 17](#_Toc184901298)

[6. Other Requirements 18](#_Toc184901299)

[Appendix A: Glossary 18](#_Toc184901300)

[Appendix B: Analysis Models 19](#_Toc184901301)

[Appendix C: To Be Determined List 35](#_Toc184901302)

# Introduction

## Purpose

The purpose of this document is to specify the requirements and preview some elements of the analysis model of ParkSense.

ParkSense is a management system that can efficiently keep track and display in real time the availability of parking spaces in Asia Pacific College. Combining entry and exit data in a centralized Vehicle Count Database provides accurate accounting of parking spaces and helps optimize value the space in our parking. It gives quick updates through an information display for a smooth, seamless, and hassle-free parking experience. It also analyzes vehicle data for useful vehicle management and decision-making purposes.

## Document Conventions

This Software Requirement Specification (SRS) follows specific convention for utmost consistency and clarity ensuring readers to get a clear understanding. The formats used are stated below:

1. **Fonts and Formatting:**

* Font: This SRS document will use Times New Roman as font style with 12 font size.
* Heading: All major sections are bold to be able to differentiate sections from subsections. Numbers and letters will be used to denote subsections.

1. **Highlighting and Emphasis**

* Bold: Used for requirements, section headings, and emphasis on critical information or specific terms that need to stand out.

1. **Version Control**

* Revision History: Any future changes and revision will be stated in the revision history in this document.

## Intended Audience and Reading Suggestions

**Developers**: Developers are responsible for designing, coding, and maintaining the ParkSense system to ensure effective implementation and reliable functionality.

**Suggested Reading:**

* **Section 1.1 Purpose:** Understand the objectives behind the ParkSense system.
* **Section 1.4 Product Scope:** Gain an overview of the boundaries and extent of the system.
* **Section 2.1 Product Perspective:** Learn how the system integrates with existing environments.
* **Section 3 Interface Requirements:** Dive into the technical specifications for system interaction.
* **Section 4 System Features:** Explore the capabilities and functionalities of the system.

**Users**

**Admin and Office Staff**

Admins and office staff will use the system to manage and oversee parking operations, gaining insights into how ParkSense supports effective administration.

**Suggested Reading:**

* **Section 1.1 Purpose:** Overview of the system's goals.
* **Section 2.1 Product Perspective:** How the system fits into organizational workflows.
* **Section 2.3 User Classes:** Understand different user roles and access levels.
* **Section 4 System Features:** Discover tools and functionalities available to staff.

1. **Parking Lot Owners and Applicants**  
   These users will interact with the system to streamline applications, monitor requirements, and manage parking slots.

**Suggested Reading:**

* **Section 1.1 Purpose:** Understand the overall goals of ParkSense.
* **Section 2.2 Product Functions:** Learn how the system addresses specific needs.
* **Section 2.3 User Classes:** Gain clarity on user roles and interactions.

**Tester**  
Testers play a critical role in identifying and validating the system’s functional and non-functional requirements during quality assurance.

**Suggested Reading:**

* **Section 2.5 Constraints:** Understand limitations and challenges.
* **Section 3 Interface Requirements:** Assess technical requirements for system interfaces.
* **Section 4 System Features:** Verify all functionalities are implemented as expected.

## Product Scope

ParkSense, a comprehensive parking management system, maintains track of all available parking spaces in real-time. Real-time updating of a centralized Vehicle Count Database occurs when automobiles enter and exit via appropriate sensors. The created data is then shown on information panels, allowing individuals to conveniently keep track of available spaces.

Most notably, it offers integrated solutions that connect vehicle sensors, databases, anddisplay interfaces to provide smooth parking management. Along with these capabilities, an analytical report would generate reports that would assist BMO staff in identifying trends, potential high usage times, and making informed decisions.

The optimization of parking operations will allow for real-time tracking of vehicle movements and updates. Such a solution will improve the overall parking experience by giving dependable information and insights for day-to-day business operations and long-term planning.

## References

Asia Pacific College Data Privacy: <https://www.apc.edu.ph/privacy-policy/>

MSYADD Document: [ImPossible\_SSYADD\_Finals\_Paper.pdf](https://asiapacificcollege.sharepoint.com/:b:/r/sites/ImPossible/Shared%20Documents/General/MSYADD1%20Finals/ImPossible_SSYADD_Finals_Paper.pdf?csf=1&web=1&e=FE7JKa)

# Overall Description

## Product Perspective

The ParkSense integrated parking management system promises to track vehicles and improve parking management operations. A centralized vehicle count database is employed, with sensors monitoring vehicle entry and exit, as well as real-time parking spot availability displays, allowing customers to easily discover free slots. As a result, ParkSense steps in to provide accurate and up-to-date information, as well as analytical insights, to help with parking management decision-making.

The system consists of two key components: the Sensor-Database interface and the Display Management module. The Display Management module displays information from the database on information screens. As a result, in most cases, consumers receive updates around the clock.

The system also includes an Analysis and Reporting component, which collects and processes data within the database. It would generate parking-related trend reports and analyze their usage to gain insight on how BMO Staff would use them to optimize parking operations.

ParkSense's integrated components enable effective and dependable parking management, as well as increased user satisfaction and operational efficiency.

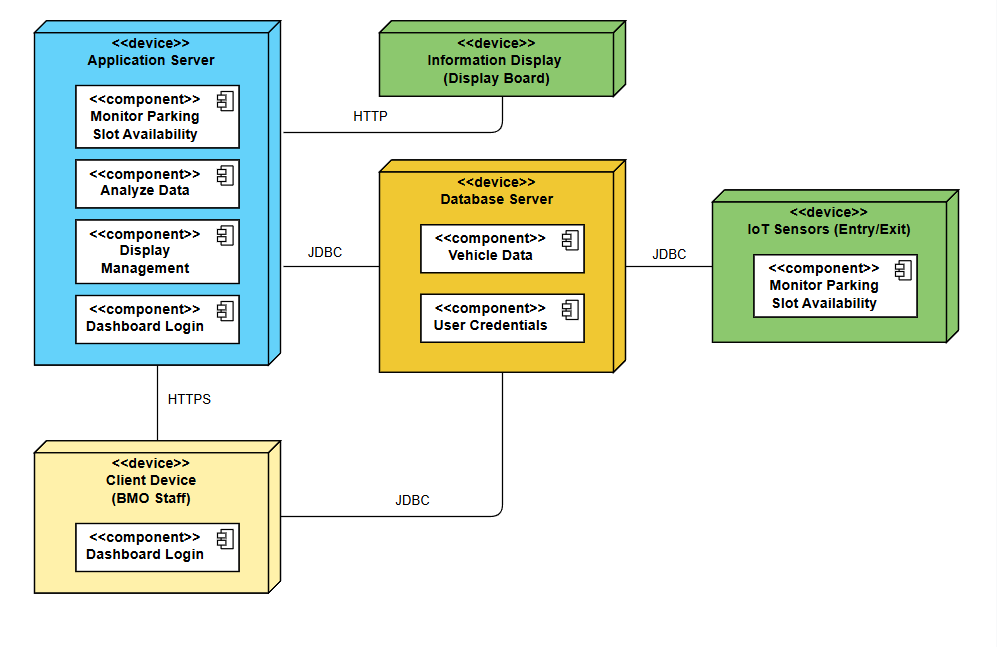


Figure : Deployment Diagram

## Product Functions

**Dashboard Login**

Allows the user, BMO Staff, to access the system with their login credentials, enabling access to the system functionalities.

**Monitor Slot Availability**

Tracks and updates the availability of the slots by tracking the vehicle entries and exits through the help of the sensors.

**Analyze Data**

Processes vehicles count data and generate insights to assist the BMO staff in decision-making and reporting about the basement parking.

**Display Management**

Provides real-time status of the basement parking which updates the information display for vehicle users to view slot availability.

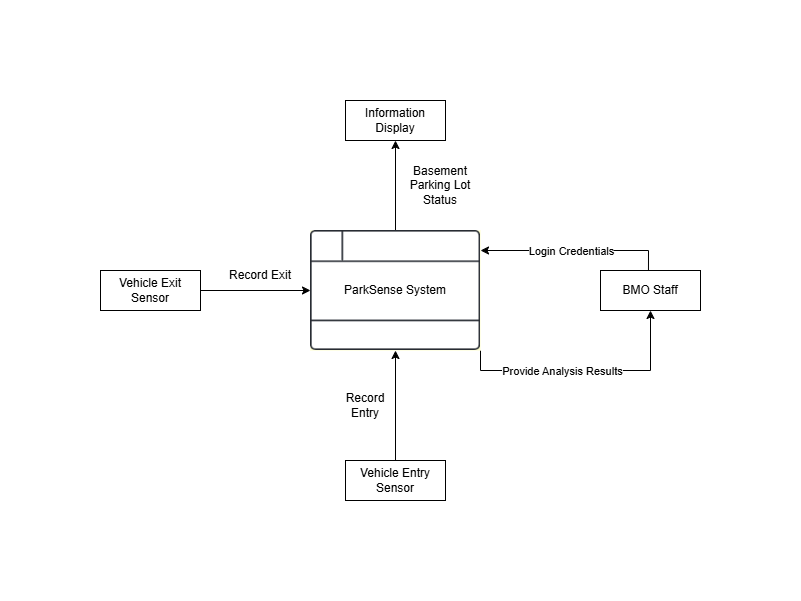


Figure : Level 0 Dataflow Diagram (Context Diagram)

## User Classes and Characteristics

For conventional reasons we name each of the user classes-actors with this format:

**Physical Actors:**

* **BMO Admin**: The BMO Admin has full control over the system, including managing user accounts, viewing reports, and utilizing system tools. They can generate analytics reports and access maintenance tools to ensure the proper functioning of the parking management system.
* **BMO Staffs**: BMO Staff are responsible for monitoring parking status and ensuring smooth parking operations. They log into the system to track real-time parking availability and verify vehicle entries and exits, which are detected using Arduino sensors.

**System Actors:**

* **Arduino Sensors**: The purpose of these tiny sensors is to identify when a car enters or exits. They are also essential for tracking parking spot occupancy and updating the ParkSense system's status since these sensors provide accurate parking occupancy data, which makes the entire system very efficient.

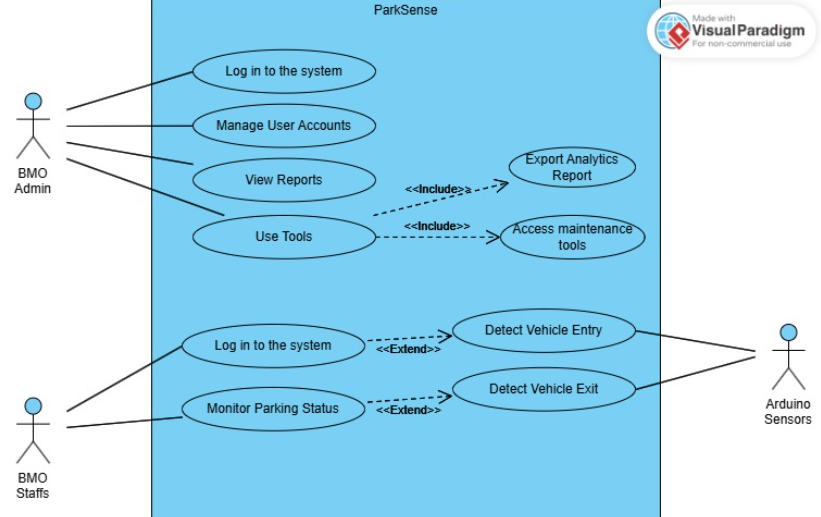


Figure : Use Case Diagram

## Operating Environment

**Hardware Requirements:**

* **Sensors**: Describe the type (e.g., Arduino or other compatible parking sensors) used to detect vehicle entry and exit.
* **Display Boards**: Hardware for real-time slot availability updates.
* **Server Requirements**: Minimum hardware specifications for hosting the system (e.g., CPU, RAM, storage).
* **Network Infrastructure**: Routers, switches, and any necessary cabling for connectivity.

**Software Requirements:**

* **Operating System**: Specify compatible OS versions (e.g., Windows Server, Linux-based distributions).
* **Backend**: Django Framework
* **Database**: MongoDB or other compatible systems
* **Web Server**: Apache or Nginx
* **Client Software**: Browser-based access for users (e.g., Google Chrome, Mozilla Firefox).

**Network Requirements:**

* **Protocols:** Mention TCP/IP for communication.
* **Connectivity:** Internet or intranet connectivity for real-time updates.

## Design and Implementation Constraints

**Corporate** **or** **Regulatory** **Policies**:

* Compliance with Asia Pacific College’s IT policies, including data privacy and security.
* Adherence to local parking and facility management regulations.

**Hardware Limitations:**

* **Sensor Responsiveness**: Arduino or similar sensors must detect vehicle entry and exit within 2 seconds to ensure real-time updates.
* **Display Board Limitations**: Monitors must be capable of operating in various environmental conditions (e.g., temperature, humidity).
* **Server Capacity**: The system must function efficiently with the expected peak load of vehicles during campus hours.

**Interfaces to Other Applications:**

* Integration with external analytics tools for generating detailed parking usage reports.
* Possible compatibility with mobile apps or dashboards for extended functionality in the future.

**Specific Technologies, Tools, and Databases:**

* The system is built using:
* **Backend Framework**: Django
* **Database**: MongoDB for development; scalable options for production deployment.
* **Frontend**: HTML, CSS, JavaScript for display interfaces.
* Limited to open-source tools to reduce costs.

**Language Requirements:**

* System must be implemented in Python for compatibility with Django.

**Security Considerations:**

* The system must ensure secure storage of parking data, such as encrypting data at rest and during transmission.
* Access to the dashboard is restricted to authorized personnel (e.g., BMO staff) using role-based authentication.

## User Documentation

This section shows the documentation that will be delivered with the software. It will help the user to further understand, navigate, and use the system’s features.

**User Manuals**

* This contains all the guides on how to use the system. This will be delivered in PDF format.

**System Diagrams**

* This document will contain all the diagrams involved in the development of the system. This document will be in PDF format.

## Assumptions and Dependencies

ParkSense has been developed using Django, a high-level Python-based web framework renowned for its simplicity and security in building web applications. Django was chosen for its built-in functionalities such as user authentication, URL routing, and ORM for database operations, making it an excellent choice for a scalable and maintainable parking management system. This document assumes readers are primarily focused on understanding the system's functionality and user interactions rather than its underlying technical implementation. Detailed documentation for developers, including coding standards and database configurations, is available separately.

**Assumptions:**

* The system will be deployed in an environment where Python, Django, and other dependencies are installed and properly configured.
* All users accessing the system have devices with modern browsers capable of running JavaScript.
* Internet connectivity is required for certain features, such as user authentication, real-time parking updates, and notification services.
* Administrators are responsible for managing user roles and permissions, ensuring that only authorized personnel have access to sensitive functions.

**Dependencies:**

* The system relies on a MongoDB database for storing user information, parking slot data, and transaction records.
* Sensors and IoT devices for parking space detection must be properly installed and operational.

These assumptions and dependencies are critical for the successful deployment and operation of ParkSense, ensuring a seamless user experience and robust system performance.

# External Interface Requirements

## User Interfaces

A screenshot of a parking meter

Description automatically generated

Figure 4 Prototype: Tracker Display

A screen shot of a computer

Description automatically generated

Figure 5 Prototype: Dashboard

A diagram of a basement

Description automatically generated

Figure 6 Prototype: Basement 1 Display

A diagram of a basement

Description automatically generated with medium confidence

Figure 7 Prototype: Basement 2 Display

A screenshot of a video game

Description automatically generated

Figure 8 Prototype: Basement 1 Slots Taken

A graph of a car

Description automatically generated with medium confidence

Figure 9 Prototype: Analytics

## Hardware Interfaces

The ParkSense system must interact with the following hardware components to function reliably:

Client Devices

* + - 1. Desktop Computers or Laptops
* Purpose: Allows car drivers to monitor the real-time availability of parking slots and system notifications from anywhere.
* Specification:
  + Operating System: Android or iOS.
  + Screen Size: At least 5 inches diagonal for good user experience.
  + Connectivity: Active internet connection through Wi-Fi or cellular data.
  + Peripherals: Touchscreen interface for ease of use.
    - 1. Mobiles Devices (Smartphones and Tablets)
* Purpose: Allows car drivers to monitor the real-time availability of parking slots and system notifications from anywhere.
* Specification:
  + Operating System: Android or iOS
  + Screen Size: At least 5 inches diagonal for good user experience.
  + Connectivity: Active internet connection Wi-Fi or cellular data.
  + Peripherals: Touchscreen interface

Networking Hardware

Router and Switches

* Purpose: This facilitates safe and reliable communication between sensors, client devices, and the ParkSense database.
* Specification:
  + Support for IPv4
  + Minimum of 1 Gbps for smooth data transfer.

Arduino Sensors

* Purpose: Determine entry and exit times of vehicles. Real time system update.
* Specification:
  + Type Proximity and motion detectors.
  + Wired or wireless communication with the server.
  + Power Requirements: It works on low power consumption, which can be used for extended periods.

## Software Interfaces

ParkSense will interact with various systems to ensure seamless monitoring and management of parking spaces. Its primary integrations include hardware sensors for real-time parking slot availability updates and backend systems for data processing and analytics. Communication will be facilitated using RESTful APIs. Detailed descriptions of these connections are provided below:

* **Databases:**
  + **MongoDB.** This will be used to store all system data, including user details, parking slot statuses, booking records, and system logs.
* **Operating Systems:**
  + **Windows Server.** This will host the web application, providing reliable support for HTTPS, ODBC, and SSH for server administration.
* **Tools and Libraries:**
  + **Frontend Framework:** React.js. This will be used to develop interactive and responsive user interfaces.
  + **Backend Framework:** Django REST Framework. This will handle server-side logic, process RESTful API requests and responses, and communicate with the database.
  + **Authentication:** Django Allauth. This will be utilized to manage user authentication, role assignments, and access control.
* **API Protocols:**
  + **RESTful API.** This will establish communication between the frontend, backend, and external services like sensor APIs or third-party analytics.

## Communications Interfaces

* **Web Browser Interface:**
  + The system will support modern web browsers to provide users access to the ParkSense platform.
  + Communication between the user’s browser and the server will use HTTP/HTTPS protocols for security and compatibility.
  + All client-side requests will be formatted as JSON to ensure standardization and ease of parsing.
* **Network Server Communications Protocols:**
  + Communication between the frontend and backend will utilize RESTful APIs over HTTPS for secure data exchanges.
  + The server will adhere to HTTP/1.1 or HTTP/2.0 standards for request/response communication.
  + API requests, such as for parking slot availability or user data, will include authentication tokens in HTTP headers.
* **Communication Standards:**
  + HTTPS for all data transmissions to protect sensitive information.
  + JSON formatting for API communication to ensure consistency.
  + UTF-8 character encoding for all transmitted data.
* **Synchronization Mechanisms:**
  + Synchronization between server systems and local parking sensors will occur every 10 seconds. This ensures that the parking slot statuses are updated in near real-time across all connected devices and the central database.

# System Features

## Parking Slot Monitoring System

### Description and Priority

* **Description**: Parking slot availability is in real time as managed by sensors. It keeps refreshing the display board at the entry so that drivers come to know how many slots are vacant, thus reducing searching time.
* **Priority**: High

### Stimulus/Response Sequences

* **Stimulus**: A car pulls into a parking space.
  + **Response**: The sensor detects the location of the car while the system marks this slot occupied.
* **Stimulus**: A vehicle exits the parking space.
  + **Response**: The sensor detects when the vehicle exits, and the system marks the slot as vacant.
* **Stimulus**: A user sees the display board.
  + **Response**: The system indicates whether there are currently available slots.

### Functional Requirements

* REQ-1: The system shall update the parking slot status within 2 seconds of vehicle detection.
* REQ-2: The Display Board should be readable at 10 meters and update on line.
* REQ-3: The system must accommodate a maximum of 200 concurrent slots.
* REQ-4: Anomalies in sensor data should activate notifications to the maintenance personnel.

## Parking Analytics Dashboard

### Description and Priority

* **Description**: It enables the BMO workforce to view both historical and live parking data to optimize resource allocation and operations.
* **Priority**: Medium

### Stimulus/Response Sequences

* **Stimulus**: Employees click on "View Reports" on the dashboard.
  + **Response**: This system displays the information through graphs and charts.
* **Stimulus**: Staff requests for the export report.
  + **Response**: The system produces the report in the specified format, such as PDF or Excel.

### Functional Requirements

* REQ-1: Reports should have occupancy trends at different hour, day, and month levels.
* REQ-2: Reports shall be able to export from the dashboard to PDF and Excel.
* REQ-3: Graphs and Chart Structures: Bar charts, line graphs, and pie charts.
* REQ-4: Only authorized users can view the dashboard.

## Parking Analytics Dashboard

### Description and Priority

* **Description**: This feature identifies vehicle entry and exit by using Arduino sensors for the record of occupancy in parking.
* **Priority**: High

### Stimulus/Response Sequences

* **Stimulus**: A car is approaching the parking gate.
  + **Response**: It increments the count of used slots.
* **Stimulus**: A car leaves the parking gate.
  + **Response**: It decrements the count of used slots.

### Functional Requirements

* REQ-1: Sensors must detect vehicles with 95% accuracy.
* REQ-2: The system should synchronize entry/exit data in the database within 2 seconds.
* REQ-3: Failure alarms should notify the maintenance staff within 1 minute.

# Other Nonfunctional Requirements

## Performance Requirements

The ParkSense system must meet the following performance requirements to ensure a smooth and efficient user experience under various conditions:

1. **Real-Time Updates:**
   1. The system must update parking slot availability data within **2 seconds** of a sensor status change. This ensures that users receive fast feedback on parking availability.
   2. Synchronization between sensors and the central database should occur every **10 seconds** to maintain consistency across all connected devices.
2. **User Response Time:**
   1. User interactions, such as logging in and checking availability, must receive a response within **1 second** for 95% of all requests.
   2. For complex queries, such as generating analytics reports, responses must be delivered within **5 seconds** to maintain user engagement.
3. **Data Handling and Processing:**
   1. The system must process and store new entries, sensor data, and cancellations within **3 seconds** of submission.
   2. Batch processing tasks, such as nightly data summaries or archiving, should complete within **30 minutes**, even when handling large datasets.
4. **Uptime and Availability:**
   1. The system must have an uptime of **99.9%**, allowing only minimal downtime for maintenance or updates.

These performance benchmarks are designed to ensure that ParkSense operates efficiently in real-world scenarios while delivering a reliable and user-friendly experience. Developers should prioritize scalability and responsiveness when designing and implementing system features.

## Safety Requirements

The system must ensure that no data is permanently deleted to maintain the integrity and auditability of the system. Instead of true deletion, data such as parking slot statuses, booking records, and user activities will be marked as "cancelled" or "archived" when necessary. Historical logs of all system actions, including updates, cancellations, and user activities, must be preserved to facilitate audits, troubleshooting, and accountability. This ensures a transparent record of all operations within the system.

## Security Requirements

ParkSense manages sensitive operations such as user data, parking reservations, and payment processing. All connections will only be permitted from authenticated users through secure channels like HTTPS. Role-based access controls will be implemented using Django Allauth to restrict access to specific features or data based on user roles (e.g., guests, admins). User authentication tokens will be validated for every API request to enhance security.

## Software Quality Attributes

* **Reliability:** The system must be robust and capable of handling errors gracefully. For example, the system should ensure that parking slot statuses and booking operations are failure-free. If there is a network disruption or sensor malfunction, users should receive clear and actionable feedback.
* **Portability:** While the primary client machines run on Windows, the system should be designed with cross-platform compatibility, ensuring it can run on mobile devices such as Android tablets. This flexibility allows future transitions to mobile-based clients.
* **Usability:** The system must have an intuitive and user-friendly interface, catering to non-technical staff or users. Training requirements for using ParkSense should be minimal, and the transition from legacy parking systems should be smooth and frictionless.
* **Maintainability:** The software must adhere to best coding practices and align with the technological stack chosen by the development team (e.g., Django and React). This ensures a seamless handover process to maintenance teams and supports long-term scalability and updates.

## Business Rules

One **PARKING SLOT SENSOR** monitors One **PARKING SLOT**; One **PARKING SLOT** is monitored by One **PARKING SLOT SENSOR**.

One **USER** receives One **PARKING SLOT ALLOCATION**; One **PARKING SLOT ALLOCATION** is received by One **USER**.

One **PARKING REPORT** is generated by One **BMO STAFF MEMBER**; One **BMO STAFF MEMBER** generates One **PARKING REPORT**.

One **MONITOR DISPLAY** shows One **BASEMENT LEVEL STATUS**; One **BASEMENT LEVEL STATUS** is shown on One **MONITOR DISPLAY**.

One **BMO STAFF MEMBER** performs One **DATA ANALYSIS**; One **DATA ANALYSIS** is performed by One **BMO STAFF MEMBER**.

One **DATABASE** stores One **PARKING RECORD**; One **PARKING RECORD** is stored in One **DATABASE**.

One **SYSTEM CHECK** is conducted by One **BMO STAFF MEMBER**; One **BMO STAFF MEMBER** conducts One **SYSTEM CHECK**.

One **SENSOR** updates One **AVAILABILITY STATUS** in real-time; One **AVAILABILITY STATUS** is updated by One **SENSOR**.

# Other Requirements

Upon implementation, the ParkSense system must achieve the following target metrics to demonstrate its effectiveness in parking availability monitoring:

1. Reduce the average time users spend identifying available parking slots by **40%** within the first 3 months of operation.
2. Ensure a **99.9% accuracy** in displaying real-time parking slot availability based on sensor data.
3. Ensure that sensor data synchronization with the central server is completed within **10 seconds** of a status change.
4. Attain a user satisfaction rate of **90% or higher** in surveys conducted during the first 6 months of deployment, based on the system's accuracy, responsiveness, and ease of understanding.

# Appendix A: Glossary

**ParkSense:** The parking basement monitoring system designed to display real-time parking slot availability for users in an intuitive and efficient manner.

**Dashboard:** A user interface that provides administrators with a clear overview of the system's status, including metrics like total parking slots, occupancy rates, and availability trends.

**Parking Slot Availability:** The primary feature of ParkSense, which shows the status of each parking slot (occupied or available) in real-time using sensor data.

**Sensors:** Hardware components installed in parking slots that detect whether a space is occupied or vacant. These sensors communicate data to the ParkSense system for accurate availability reporting.

**Django:** The backend framework used to build the ParkSense system, managing server-side logic, user authentication, and the integration of sensor data.

**React.js:** The frontend framework used for creating the user interface of ParkSense, ensuring a responsive and user-friendly experience.

**RBAC (Role-Based Access Control):** A security mechanism in ParkSense to define user permissions. For instance, administrators can view detailed analytics, while guests can only view parking slot availability.

**HTTPS (Hypertext Transfer Protocol Secure):** A secure protocol used to encrypt data transmission between users and the ParkSense system to ensure the safety of information.

**Administrator:** A system user with access to manage the system's settings, view analytics, and oversee parking slot monitoring operations.

**Guest:** A system user, such as a driver, who interacts with the system to view parking slot availability and other general information.

**Occupancy Trends:** Analytical insights displayed on the dashboard, showing patterns of parking lot usage over time to aid in operational planning.

**Synchronization:** The process by which sensor data is updated in the ParkSense system to ensure real-time accuracy in displaying parking slot availability.

# Appendix B: Analysis Models

**Dataflow Diagram**

### **Level 0**

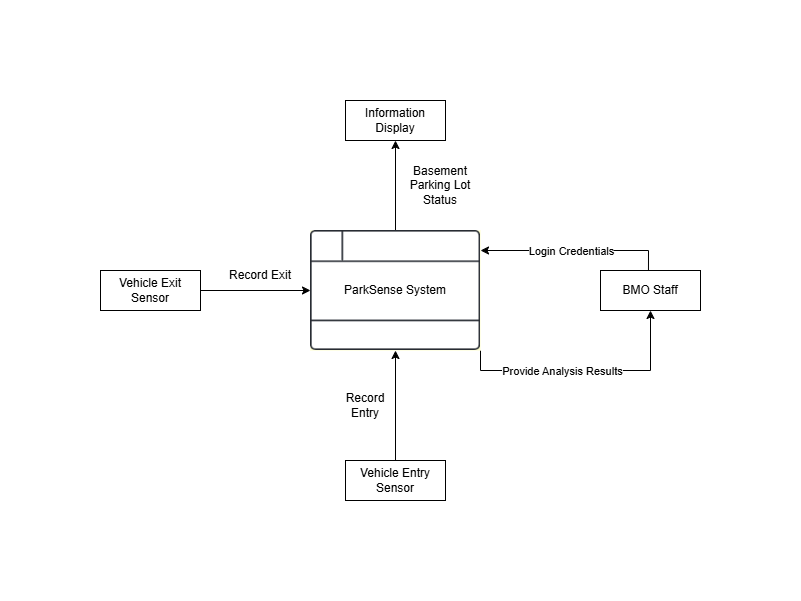


Figure 10: Level 0 Dataflow Diagram

### 

### **Level 1**

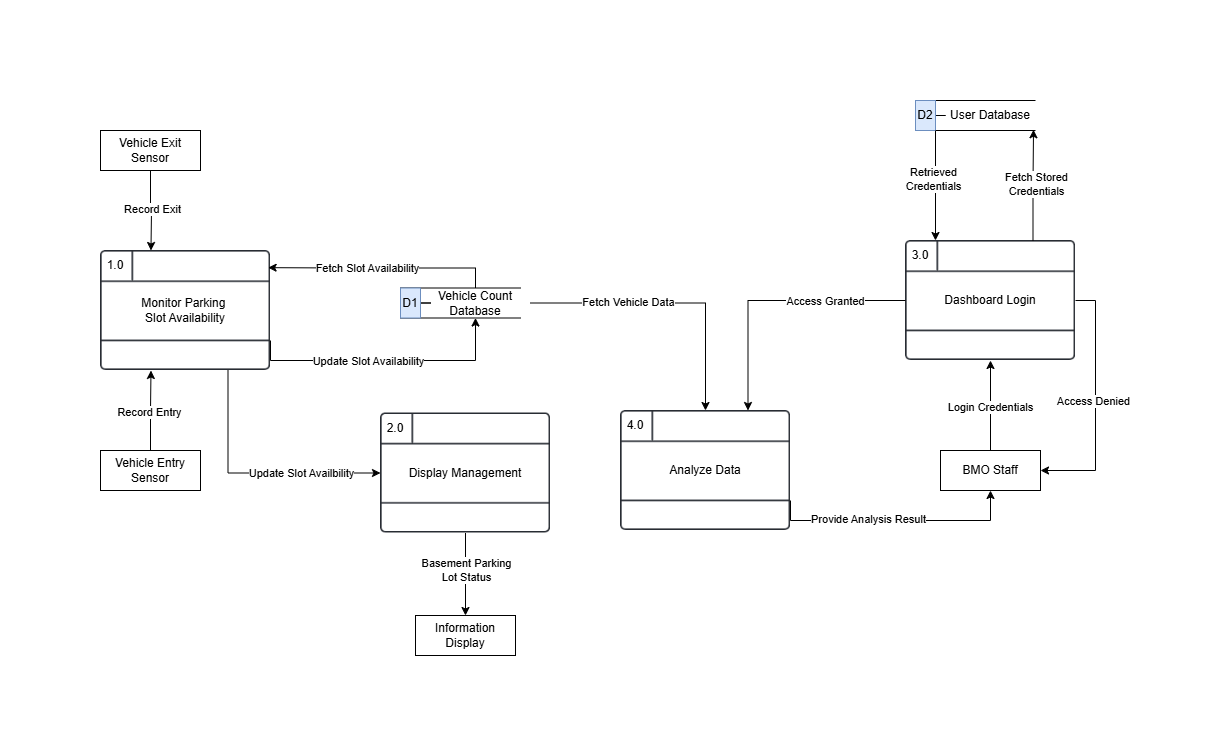


Figure 11: Level 1 Dataflow Diagram

### Level 2

#### **Monitor Parking Slot Availability**

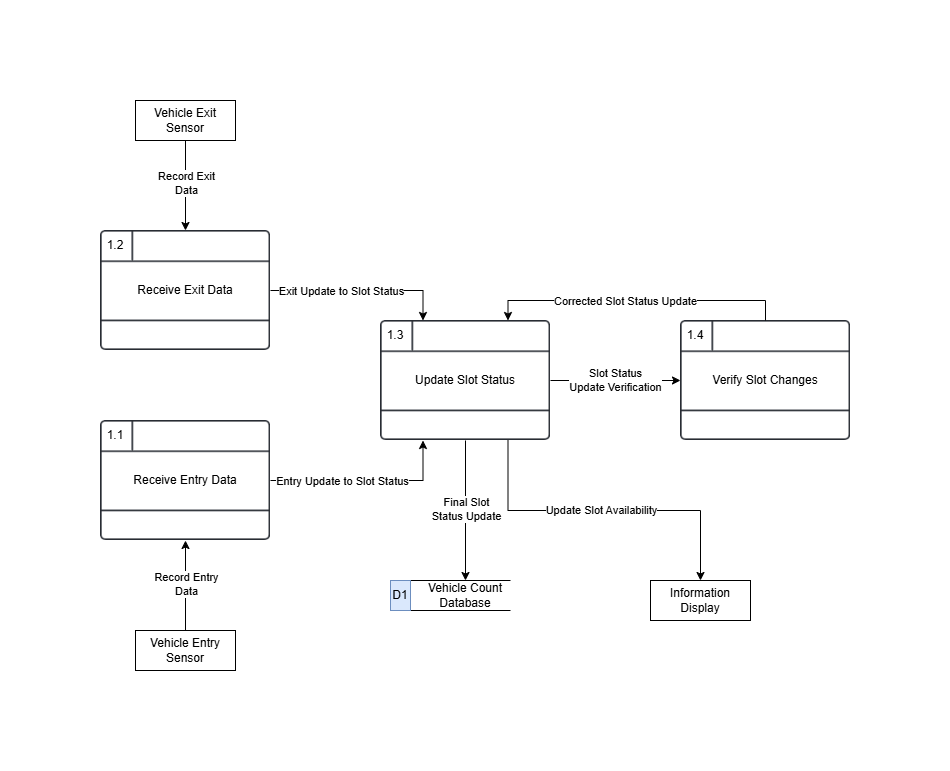


Figure 12: Level 2 Dataflow Diagram

#### **Dashboard Login**

A diagram of a computer

Description automatically generated

Figure 13: Level 2 Dataflow Diagram

#### **Analyze Data**

A diagram of a computer

Description automatically generated

Figure 14: Level 2 Dataflow Diagram

## 

**Use Case Diagram**

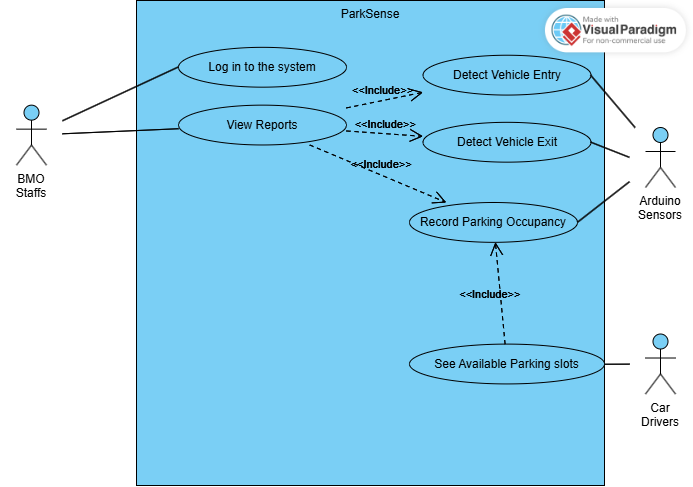


Figure 15: Use Case Diagram

**Fully Dressed Use Cases**

### **Login into the System**

|  |  |
| --- | --- |
| Use Case ID | PAS\_UC\_1 |
| Use Case Name | **Login in to the system** |
| Created By | John Jorel Landicho |
| Date Created | 9/16/2024 |
| Description | This use case describes the process of logging into the ParkSense system, allowing authorized users (BMO Staff) to access specific system functions after successful authentication. |
| Primary Actor | BMO Staff |
| Preconditions | 1. The Park Sense system is operational. 2. The user has a valid username and password. |
| Postconditions | **Success**: The user is logged into the system and can perform their assigned tasks.    **Failure**:The system denies access, and no tasks can be performed until a successful login. |
| Main Success Scenario (Basic Flow) | 1. The actor (BMO Staff) opens the ParkSense system login screen. 2. The system displays fields for the username and password. 3. The actor inputs their username and password. 4. The system checks the entered credentials against its database. 5. The system confirms that the credentials are correct and logs the actor into the system. 6. The actor is redirected to the appropriate dashboard based on their role (e.g., BMO Staff is directed to the dashboard for analyzing data;). |
| Extensions (Alternate Flows) | 2a. **Invalid Credentials:**   * **Trigger:** The user enters an incorrect username or password. * **Action:** The system detects that the credentials do not match the database records. * **Response:** The system displays an error message ("Invalid username or password") and prompts the user to retry. |
| Special Requirements | 1. Passwords should follow security protocols (e.g., minimum length, complexity requirements). 2. The login interface should be simple and user-friendly. |

Table : Fully Dressed Use Case 1

### **View Reports**

|  |  |
| --- | --- |
| Use Case ID | PAS\_UC\_2 |
| Use Case Name | **View Reports** |
| Created By | **ImPossible**: John Jorel Landicho, Rchie Libudan, Angela Mae Tauyan, Timothy Jay Sayson |
| Date Created | 11/01/2024 |
| Description | This use case allows BMO Staff to view reports based on parking data collected by the system. |
| Primary Actor | BMO Staff |
| Include Use Case | 1. Detect Vehicle Entry 2. Detect Vehicle Exit 3. Record Parking Space Occupancy |
| Preconditions | 1. The Park Sense system is operational and has logged sufficient data. |
| Postconditions | **Success:** Accurate and comprehensive reports are generated.    **Failure:** Reports are missing key information or contain errors. |
| Main Success Scenario (Basic Flow) | 1. BMO Staff selects the "View Reports" option. 2. The system retrieves historical occupancy and usage data. 3. The system displays the data in a report format. |
| Extensions (Alternate Flows) | 2a. **Report Contains Errors or Missing Data:**   * **Trigger:** The system notifies the BMO Staff that there is no data available. * **Action:** The BMO Staff adjusts the report parameters or requests system maintenance. * **Response:** A corrected report is generated. |
| Special Requirements | 1. The reporting tool should support different formats (PDF, Excel). 2. Reports should include visual aids like charts and graphs for easy interpretation. |

Table : Fully Dressed Use Case 2

### **Detect Vehicle Entry**

|  |  |
| --- | --- |
| Use Case ID | PAS\_UC\_3 |
| Use Case Name | **Detect Vehicle Entry** |
| Created By | **ImPossible**: John Jorel Landicho, Rchie Libudan, Angela Mae Tauyan, Timothy Jay Sayson |
| Date Created | 11/01/2024 |
| Description | Detects when a vehicle enters the parking facility and updates occupancy data. |
| Primary Actor | Arduino Sensors |
| Preconditions | 1. Arduino sensors are functional and calibrated. |
| Postconditions | **Success:** The system records a new vehicle entry and updates parking occupancy.    **Failure:** The system fails to record  new vehicle entry and not update the parking occupancy. |
| Main Success Scenario (Basic Flow) | 1. A vehicle passes through the entry point. 2. The Arduino sensor detects the vehicle's entry. 3. The system increments the count of occupied parking slots. |
| Extensions (Alternate Flows) | 2a. **If the sensor fails to detect a vehicle entering:**   * **Trigger:** The system prompts BMO Staff to check the entry status. * **Action:** The BMO Staff check the actual device and do some maintenance. * **Response:** Updates the occupied parking slots. |
| Special Requirements | 1. The system must update occupancy within 2 seconds of detection. |

Table : Fully Dressed Use Case 3

### **Detect Vehicle Exit**

|  |  |
| --- | --- |
| Use Case ID | PAS\_UC\_4 |
| Use Case Name | **Detect Vehicle Exit** |
| Created By | **ImPossible**: John Jorel Landicho, Rchie Libudan, Angela Mae Tauyan, Timothy Jay Sayson |
| Date Created | 11/01/2024 |
| Description | Detects when a vehicle exits the parking facility and updates occupancy data. |
| Primary Actor | Arduino Sensors |
| Preconditions | 1. Arduino sensors are functional and calibrated. |
| Postconditions | **Success:** The system records the vehicle exit and updates parking occupancy.    **Failure:** The system fails to record  vehicle exit and not update the parking occupancy. |
| Main Success Scenario (Basic Flow) | 1. A vehicle passes through the exit point. 2. The Arduino sensor detects the vehicle's exit. 3. The system decrements the count of occupied parking slots. |
| Extensions (Alternate Flows) | 2a. **If the sensor fails to detect a vehicle exiting:**   * **Trigger:** The system prompts BMO Staff to check the exit status. * **Action:** The BMO Staff check the actual device and do some maintenance. * **Response:** Updates the occupied parking slots. |
| Special Requirements | 1. The system should reflect the updated count within 2 seconds. |

Table : Fully Dressed Use Case 4

### **Record Parking Occupancy**

|  |  |
| --- | --- |
| Use Case ID | PAS\_UC\_5 |
| Use Case Name | **Record Parking Occupancy** |
| Created By | **ImPossible**: John Jorel Landicho, Rchie Libudan, Angela Mae Tauyan, Timothy Jay Sayson |
| Date Created | 11/01/2024 |
| Description | Records current parking occupancy based on vehicle entries and exits. |
| Primary Actor | Arduino Sensors |
| Preconditions | 1. ParkSense system is active and logging entry/exit data. |
| Postconditions | **Success:** The parking occupancy is accurately recorded and available for reporting.    **Failure:** The system fails to accurately record the parking occupancy. |
| Main Success Scenario (Basic Flow) | 1. The system tracks each vehicle entry and exit. 2. The occupancy data is updated in real-time. 3. The system logs the occupancy status for historical reporting. |
| Extensions (Alternate Flows) | 1a. **If there is an inconsistency in the entry/exit count:**   * **Trigger:** The system alerts BMO Staff for manual verification. * **Action:** BMO Staff logs into the system and reviews the data. * **Response:** BMO Staff performs a manual override to correct the occupancy count if necessary. |
| Special Requirements | 1. Data must be stored in a secure database accessible only by authorized personnel. |

Table : Fully Dressed Use Case 5

### **See Available Parking Slot**

|  |  |
| --- | --- |
| Use Case ID | PAS\_UC\_6 |
| Use Case Name | **See Available Parking slots** |
| Created By | **ImPossible**: John Jorel Landicho, Rchie Libudan, Angela Mae Tauyan, Timothy Jay Sayson |
| Date Created | 11/01/2024 |
| Description | Allows Car Drivers to view available parking slots displayed at the entrance. |
| Primary Actor | Car Drivers |
| Include Use Case | 1. Record Parking Occupancy |
| Preconditions | 1. ParkSense system is operational and updated in real-time. |
| Postconditions | **Success:** Car Drivers are informed of the number of available slots.    **Failure:** The system fails to accurately record the parking occupancy, leading to inconsistent slot availability information for Car Drivers |
| Main Success Scenario (Basic Flow) | 1. The system calculates the available parking slots based on occupancy data. 2. The system updates the display board with the current count of available slots. |
| Extensions (Alternate Flows) | 1a. **If the display board is malfunctioning:**   * **Trigger:** The system detects an issue with the display board. * **Action:** The BMO Staff performs diagnostic checks or repairs to restore functionality. * **Response:** Display board is working and updated. |
| Special Requirements | 1. Display visibility must meet local safety standards and be readable from at least 10 meters away. |

Table : Fully Dressed Use Case 6

## 

**Activity Diagrams with Swimlane**

### **Log in into the system**

A diagram of a system

Description automatically generated

Figure 16 Activity Diagram 1

### **View Reports**

A diagram of a data flow

Description automatically generated

Figure 17: Activity Diagram 2

### 

### **Detect Vehicle Entry**

A diagram of a vehicle entry

Description automatically generated

Figure 18: Activity Diagram 3

### 

### **Detect Vehicle Exit**

A diagram of a vehicle

Description automatically generated

Figure 19: Activity Diagram 4

### 

### **Record Parking Occupancy**

A diagram of a system

Description automatically generated

Figure 20: Activity Diagram 5

### 

### **See Available Parking Slot**

A diagram of a computer program

Description automatically generated

Figure 21: Activity Diagram 6

## 

**Database Design**

A screenshot of a computer

Description automatically generated

Figure 22: Entity Relationship Diagram

# Appendix C: To Be Determined List

* **TBD-1: Sensor Specifications:** Specific type and model of Arduino-compatible sensors to be used for vehicle detection. Calibration specification for sensors under various environmental conditions.
* **TBD-2: Display Board Specifications:** Specification of display boards Vendor Screen size Resolution Weather resistance Power Specifications and Installation configurations for outdoor deployment.
* **TBD-3: Database Technology:** Whether to deploy MongoDB or PostgreSQL. Strategy for backup and recovery of the database system.
* **TBD-4:** **Analytics Tools Integration:** The integration of specific external analytical tools aims to enrich reporting functionality with regard to parking utilization.
* **TBD-5: Safety Protocols:** Final selection of encryption methods for data both in rest and in transit. Defined roles and permissions associated with Role-Based Access Control (RBAC).
* **TBD-6: Hardware Interface Requirements:** Use server information that contains CPU, memory, and storage. Brands of networking hardware (Router and Switch) and configurations.
* **TBD-7: User and Client Feedback Mechanism:** Ways in assessing if the project will have a positive impact and improvement.