Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

This paper documents our project, “Campus Parking Organizer,” a web application that allows users to see available parking spaces inside the campus and create reservations. It includes all the project specifications and requirements for developing the application.

1.2 Scope

Campus Parking Organizer effectively manages vehicles parked inside the campus, including those owned by students, faculty, and other employees. The app's main features include displaying the number of available parking lots, map navigation, and space reservation.

1.3 Definitions, Acronyms, and Abbreviations

- ❖ Campus Parking Organizer – a web application that tracks available parking spaces
- ❖ Display of Available Lots – displays the number of available parking lots
- ❖ Map Navigation – displays a map view of the parking places inside the campus
- ❖ Space Reservation – allows the user to reserve a parking space beforehand
- ❖ Admins – are the technical staff that manages the website
- ❖ Users – are the clients of the web application

1.4 Overview

This paper is divided into seven main categories: Introduction, Overall Description, Specific Requirements, External Interface Requirements, System Features, Other Non-Functional Requirements, and Appendices. This document explains how the Campus Parking Organizer application will work and what it needs before deployment.

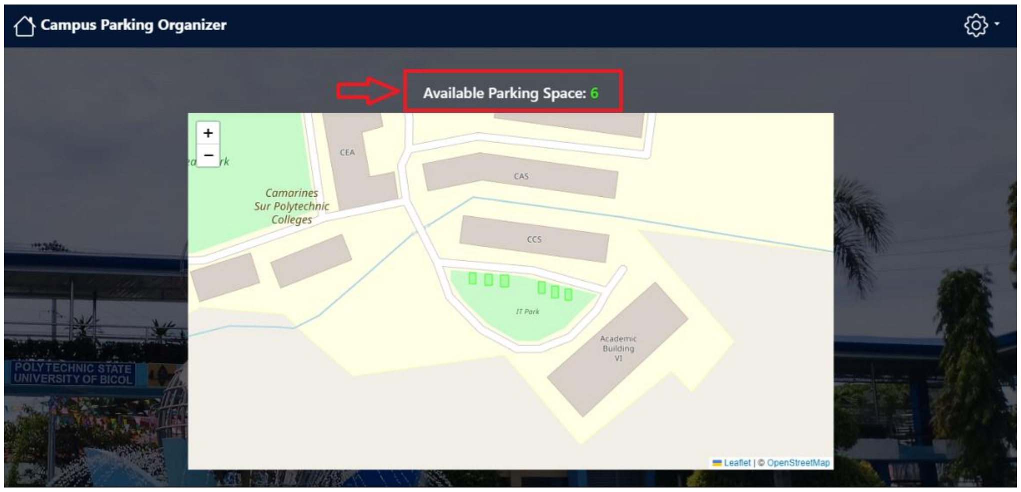
2. Overall Description

2.1 Product Perspective

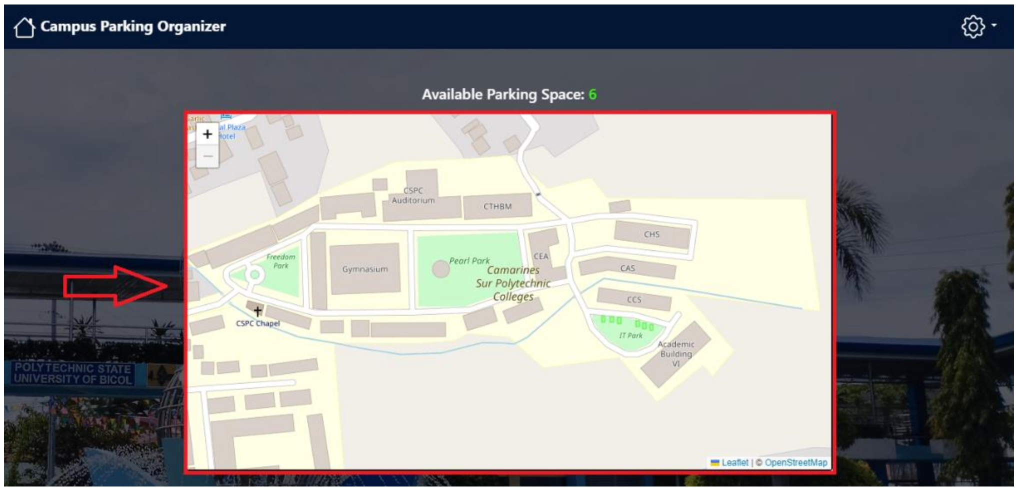
There are several projects like the proposed application in different places. Khanna and Anand (2016) present an IoT-based cloud-integrated smart parking system that monitors parking space availability and allows users to book parking spots. Another study by Sadhukhan (2017) proposed an IoT-based E-parking system prototype with real-time detection and automatic collection of parking fees. Both of these studies serve as the foundation of the current proposed application.

2.2 Product Features

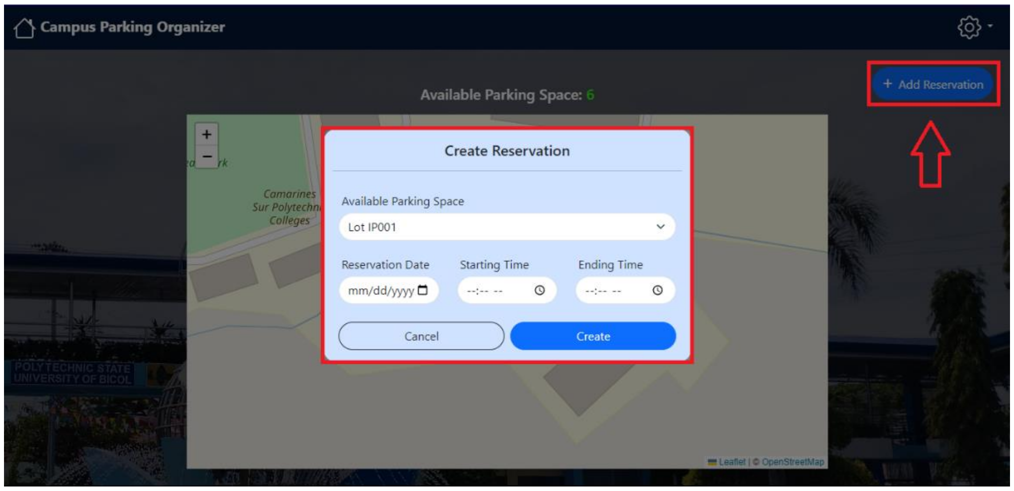
- Display of Available Lots – display the number of available parking spaces


Campus Parking Organizer
Available Parking Space: 6
CEA
CAS
CCS
Camarines Sur Polytechnic Colleges
IT Park
Academic Building VI
Leaflet | © OpenStreetMap

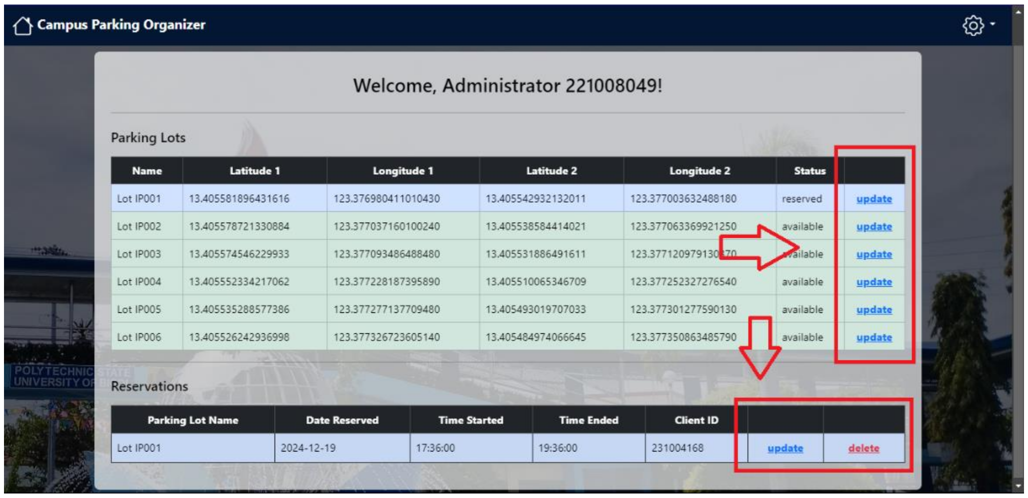
- Map Navigation – display a map view of the parking places inside the campus



- Space Reservation – allow logged in users to reserve a parking space



- ➢ <u>Booking and Parking Lot Management</u> – allow admins to modify or delete a booking and update the status of a parking space



Welcome, Administrator 221008049!

Parking Lots

Name	Latitude 1	Longitude 1	Latitude 2	Longitude 2	Status	
Lot IP001	13.405581896431616	123.376980411010430	13.405542932132011	123.377003632488180	reserved	update
Lot IP002	13.405578721330884	123.377037160100240	13.405538584414021	123.377063369921250	available	update
Lot IP003	13.405574546229933	123.377093486488480	13.405531886491611	123.377120979130[illegible]70	available	update
Lot IP004	13.405552334217062	123.377228187395890	13.405510065346709	123.377252327276540	available	update
Lot IP005	13.405535288577386	123.377277137709480	13.405493019707033	123.377301277590130	available	update
Lot IP006	13.405526242936998	123.377326723605140	13.405484974066645	123.377350863485790	available	update

Reservations

Parking Lot Name	Date Reserved	Time Started	Time Ended	Client ID		
Lot IP001	2024-12-19	17:36:00	19:36:00	231004168	update	delete

2.3 User Classes and Characteristics

- ➢ Students – are the current students of the campus
- ➢ Employees – are the employees of the campus
- ➢ Guests – are the guests from outside the campus (can't log in)
- ➢ Admin – are the technical staff that manage the website

2.4 Operating Environment

The software is a web application hence, it can run on any device with a browser (e.g. Google Chrome, Firefox, Brave). It requires an internet connection; the device must also be connected to the campus network. The software uses the Flask template, Leaflet.js for the map interface, and MySQL as the database.

2.5 Design and Implementation Constraints

The software requires an internet connection to run the map interface. The Map API interacts with Leaflet and OpenStreetMap to fetch map data, geolocation services, and other features. Additionally, the reservation feature is limited to users who are currently logged in.

2.6 Assumptions and Dependencies

The application depends on the Leaflet.js and OpenStreetMap services to display a map of the parking spaces. The users are also required to be connected to the campus network to access the web app. The application also uses a database that requires the Xampp server to host the MySQL database.

3. Specific Requirements

3.1 Functional Requirements

The Campus Parking Organizer must help manage parking effectively and allow the users to register an account and log in. Once the user logs in, they can see the original dashboard, but they are now able to make a booking. A user can only reserve a space that is available.

3.2 User Interfaces

The Campus Parking Organizer will include a standard map showing the availability of parking spaces. The user can see the number of available parking spaces above the map display. At the top-right corner of the webpage, the reservation button is located where logged-in users can make reservations

3.3 Hardware Interfaces

The software can interact with client devices through a browser and internet connection to the campus network. This allows the users to access the application on many devices, giving it much portability. The application also interacts with a device managed by administrators on the server side.

3.4 Software Interfaces

Flask will serve as the interface between the database backend of the application and the front-end application which the users interact with. The map interface is provided by Leaflet.js and OpenStreetMap making it available only when there is an internet connection. MySQL serves as the database where the application's data is stored.

3.5 Communication Interfaces

The web application can be accessed by the users connected to the campus network. The application gets the map and user data from the database and requires a MySQL server connection.

3.6 Performance Requirements

The current system cannot perform real-time updates, hence, the users connected to the server must refresh the webpage to see the latest updates. Users can perform multiple bookings if there are at least two available parking lots.

3.7 Security Requirements

Security is very important, so user authentication will have hashed passwords, and only registered users can make reservations. Sensitive information such as user transactions can only be accessed by administrators.

3.8 Reliability Requirements

The application will implement proper input validation and error handling to avoid unexpected errors or bugs. Regular database backups will also be performed to allow quick restoration in case of data loss.

3.9 Maintainability Requirements

The application will undergo regular maintenance at night when there are no users. Documentation will be included in the web application allowing the users to learn how to properly use the application.

3.10 Portability Requirements

The application can be accessed on any device with any modern web browser that supports HTML and HTTP such as Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari, and Opera.

4. External Interface Requirements

4.1 User Interfaces

The web application is composed of a navigation bar at the top with a link to the homepage at the left and a settings button at the right. The main content of the webpage displays the number of available parking spaces as well as their locations in an interactive map.

4.2 Hardware Interfaces

The application can be accessed on any device with any modern web browser that supports HTML and HTTP. The following are examples of such browsers:

- Google Chrome
- Mozilla Firefox
- Microsoft Edge
- Apple Safari
- Opera

4.3 Software Interfaces

Flask will serve as the interface between the database backend of the application and the front-end application which the users interact with. The map interface is provided by Leaflet.js and OpenStreetMap making it available only when there is an internet connection. MySQL serves as the database where the application's data is stored.

4.4 Communication Interfaces

The web application can be accessed by the users connected to the campus network. The application gets the map and user data from the database and requires a MySQL server connection.

5. System Features

5.1 Feature 1: Display of Available Lots

- **Description**: Allows the users to view the number of available parking spots as well as their locations
- **Functional Requirements**: This feature requires a database to store the data of all the parking spots including their states whether available or not

5.2 Feature 2: Map Navigation

- **Description**: This feature allows the user to navigate the parking lot locations using a map interface
- **Functional Requirements**: This feature requires Leaflet and OpenStreetMap services to display a map interface in the application

5.3 Feature 3: Space Reservation

- **Description:** This feature allows the users to ask for a reservation of a parking spot
- **Functional Requirements:** This feature requires user account management to record which user currently reserves a parking spot.

5.4 Feature 4: Booking and Parking Lot Management

- **Description:** allows admins to modify or delete a booking and update the status of a parking space
- **Functional Requirements:** This feature requires Role-Based Access management to identify the role of the logged-in user.

6. Other Non-functional Requirements

6.1 Performance

The current system cannot perform real-time updates, hence, the users connected to the server must refresh the webpage to see the latest updates. Users can perform multiple bookings if there are at least two available parking lots.

6.2 Security

The application will implement password encryption for sensitive information. It also has proper input validation to decrease the risks of data breaches. Sensitive information such as user transactions can only be accessed by administrators.

6.3 Reliability

In the occurrence of a major error that cannot be fixed within minutes, the application will be temporarily shut down to fix it and is expected to work the next day. The application also allows regular checkups during the night with fewer users.

6.4 Maintainability

The system can be easily updated as it runs on a web server and any updates implemented in the application can easily reach the users. The application documentation can also be included as another page inside the application so the users can easily access and read it.

6.5 Portability

The application is very portable as it can be accessed from different devices with a browser. The only restriction of access is the requirement of the device to connect to the campus network.

7. Appendices

7.1 Glossary

- **Admin –** are authorized users that manage the application from the server side
- **Application Programming Interface (API) -** according to IMB, it is a set of rules or protocols that allow programs to connect and exchange data, features, and functionality.
- **Bug -** an error or malfunction in the software
- **Campus Parking Organizer –** a web application that tracks available parking spaces
- **Encryption -** transforming information into a code that only authorized people can decode.
- **Input Validation -** ensuring the user input has a correct format

7.2 References

[1] IBM. (2024). “What is an API (application programming interface)?”. Retrieved from https://www.ibm.com/topics/api

[2] A. Khanna and R. Anand, "IoT-based Smart Parking System," *2016 International Conference on Internet of Things and Applications (IOTA)*, Pune, India, 2016, pp. 266-270, doi: 10.1109/IOTA.2016.7562735.

[3] P. Sadhukhan, "An IoT-based E-parking system for smart cities," *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Udupi, India, 2017, pp. 1062-1066, doi: 10.1109/ICACCI.2017.8125982.