Software Requirements Specifications

Campus Parking Management System

(CPMS)

Version: 1.0

Date: / /2025

Release By: Group \_\_

Document Control

|  |  |
| --- | --- |
| Document Name | Campus Parking Management System |
| Reference Number | CPMS\_SRS |
| Version | 1.0 |
| Project Code | CPMS\_V1 |
| Status | In-use |
| Date Released |  |

|  |  |  |
| --- | --- | --- |
| Name | Position | Signature |
| Prepared By:  Mr. Analyst | \_\_\_\_\_\_ Analyst  Group \_\_ |  |
| Reviewed By:  Mr. Reviewer | Lead Developer  Group \_\_ |  |
| Approved By:  Mr. Client | Project Manager  Group \_\_ |  |

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Release Date | Section | Amendments |
| 1.0 | / /2025 | All | Original Document |

Contents

[1. Introduction 5](#_Toc198672102)

[1.1 Purpose 5](#_Toc198672103)

[1.2 Scope 5](#_Toc198672104)

[1.3 Product Overview 5](#_Toc198672105)

[1.3.1 Product Perspective 5](#_Toc198672106)

[1.3.2 Product Functions 6](#_Toc198672107)

[1.3.3 User Characteristics 6](#_Toc198672108)

[1.3.4 Limitations 6](#_Toc198672109)

[1.4 Definitions 6](#_Toc198672110)

[2. References 7](#_Toc198672111)

[3. Requirements 7](#_Toc198672112)

[3.1 Functions 7](#_Toc198672113)

[3.2 Performance Requirements 8](#_Toc198672114)

[3.3 Usability Requirements 8](#_Toc198672115)

[3.4 Interface Requirements 8](#_Toc198672116)

[3.4.1 System Interfaces: 9](#_Toc198672117)

[3.4.2 User Interfaces: 9](#_Toc198672118)

[3.4.3 Hardware Interfaces: 9](#_Toc198672119)

[3.4.4 Software Interfaces: 9](#_Toc198672120)

[3.4.5 Communications Interfaces: 9](#_Toc198672121)

[3.5 Logical Database Requirements 9](#_Toc198672122)

[3.6 Design Constraints 10](#_Toc198672123)

[3.7 Software System Attributes 10](#_Toc198672124)

[3.8 Supporting Information 10](#_Toc198672125)

[4. Verification 11](#_Toc198672126)

[4.1 Verification Approach 11](#_Toc198672127)

[4.2 Verification Criteria 11](#_Toc198672128)

[5. Appendices 12](#_Toc198672129)

[5.1 Assumptions and Dependencies 12](#_Toc198672130)

[5.2 Acronyms and Abbreviations 12](#_Toc198672131)

# 1. Introduction

## 1.1 Purpose

## The purpose of the “Campus Ride-Sharing Platform with Parking System Integration”(CRSP) is to integrate a new ride-sharing application with the campus parking management system and digital ID verification. This platform is specifically for university community members, it aims to reduce parking demand and traffic congestion around campus.

## This document follows the ISO/IEC/IEEE 29148:2018 standards for Software Requirements Specifications. It is a foundation for system design, development, testing, and validation.

## This document is intended for:

## System developer

## Test engineers

## Project manager

## University IT administrators

## Client

## 1.2 Scope

The Campus Ride-Sharing Platform with Parking System(CRSP) Integration is a mobile-based software application and specifically for university community members, including students, faculty, and staff. The platform aims to facilitate carpooling arrangements and integrate with the campus parking management system to reduce traffic congestion and parking demand.

This system will allow users to:

* Coordinate ridesharing with verified university members.
* Set ride preferences.
* View and book available parking spaces in real-time.
* Receive ride and parking notifications via mobile and campus email.
* Authenticate their ID through the university’s Digital ID system.

The system will integrate with existing services including the **Parking Management System**, **Digital ID System**, **Campus Email System**, and the **University Network**.

This system does not include:

* Non-university users.
* External ride-hailing services. (Uber, Grab)

## 1.3 Product Overview

The Campus Ride-Sharing Platform with Parking System Integration(CRSP) is a mobile-based application designed to integrate ridesharing and parking coordination within a university environment. The product supports carpooling among university community members, allowing users to manage rides, set preferences, view available parking, and manage bookings in real-time.

### 1.3.1 Product Perspective

The Campus Ride-Sharing Platform with Parking System Integration (CRSP) integrates with the following systems:

* **Parking Management System:** Provides real-time parking availability data to the platform for display and parking booking by users.
* **Digital ID System:** Authenticates users and verifies user roles.
* **Campus Email System:** Sending and receiving of notifications, alerts, and confirmations related to rides and parking events.

Primary users:

* **University Community Members:** The main users of the system, They interact with the platform to manage rides, view parking availability, and receive notifications.
* **Admin:** An authorized person responsible for managing platform configurations, approving ride and parking data, and ensuring the platform operates within university policy.

A diagram of a system

AI-generated content may be incorrect.

Figure 1.0 CRSP Context Diagram

### 1.3.2 Product Functions

The following table (Table 1.0) shows the list of primary functions to be implemented in CRSP.

|  |  |  |
| --- | --- | --- |
| Function | Description | Accessible Role |
| Rides-Sharing Management | * Allows users to create, join, or manage carpool rides. * Enables setting ride preferences. * Review drivers. | University Community Members, Admin |
| Parking Management | * Displays real-time parking availability. * Enable users to book and manage parking reservations. | University Community Members, Admin |
| Notifications and Alerts | * Sends email and push notifications to users. | University Community Members, Admin |
| User Authentication | * Verifies users using the university’s Digital ID System to ensure access is limited | Admin |
| System Monitoring and Administration | * Allows admins to monitor ride and parking data. | Admin |

Table1.0 CRSP Product Function Table

### 1.3.3 User Characteristics

The following table (Table 1.1) shows the intended user groups and their expected required knowledge in CRSP.

|  |  |  |
| --- | --- | --- |
| User | Description | Required Knowledge |
| University community members | Person related with the university, including students, faculty, and staff, who are eligible to access and use campus-related services. | Basic mobile app usage, university email access, parking and ride-sharing policies rules, university policies. |
| Admin | Authorized person responsible for managing user accounts, managing system functionality, backend settings and monitoring parking data. | Understanding of system backend, parking and ride-sharing rules, university policies, data handling, user role management, basic computer skills. |

Table1.1 CRSP User Characteristics Table

### 1.3.4 Limitations

The Campus Ride-Sharing Platform with Parking System Integration has the following limitations that may affect its functionality or performance:

* Platform Accessibility:
  + The system is designed for mobile platforms only. It is not accessible via desktop browsers or other non-mobile devices.
* User Scope Restriction:
  + Non- university community members are not supported.
* Network Dependency:
  + The application requires a stable internet connection. Network latency may cause delays in real-time parking availability.

## 1.4 Definitions

**Application**: A campus ride-sharing platform that used to facilitate carpooling and parking coordination among university community members.

**University community members**: Person related with the university, including students, faculty, and staff, who are eligible to access and use campus-related services.

**Student**: A university member who undergraduate or postgraduate studies. A potential ride requester or driver in the system. Also represent the primary users of the ride-sharing and parking platform.

**Faculty**: Academic staff employed by a university. Such as, Lecturers and tutors.

**Staff**: Non-academic university employees, such as administrative or cleaning workers.

**Admin**: Authorized person responsible for managing user accounts, managing system functionality, backend settings and monitoring parking data.

**Carpool**: A ridesharing activity where one or more users share a vehicle.

**Parking Booking**: A system function allowing users to reserve available parking spaces based on real-time data.

# 2. References

IEEE. (2018). ISO/IEC/IEEE 29148:2018 Systems and software engineering—Life cycle processes—Requirements engineering. https://www.iso.org/standard/72089.html

Pohl, K. (2010). Requirements engineering: Fundamentals, principles, and techniques. Springer.(Not Sure)

# 3. Requirements

## 3.1 Functions

A diagram of parking system

AI-generated content may be incorrect.

Figure 2.0 CRSP Use Case Diagram

### 3.1.1Manage Rides:

|  |  |  |
| --- | --- | --- |
| Use Case | Manage Rides | |
| Purpose | University community members may manage their own rides. | |
| Actor | University community members | |
| Trigger | User click manage rides button | |
| Precondition | User at the main page. | |
| Main Flow | Step | Description |
| 1 | System will display various button on the screen. Which are, “Join Rides”, “Create Rides”, “Set Preference” and “Review Drivers”. |
| 2 | Users click one of the buttons. |
| 3 | System direct to corresponding page. |

Table 2.0 Use Case Specification – Manage Rides

A diagram of a button

AI-generated content may be incorrect.

Figure 2.1 Activity Diagram – Manage Rides

### 3.1.2 Join Rides

|  |  |  |
| --- | --- | --- |
| Use Case | Join Rides | |
| Purpose | University community members may join others created rides. | |
| Actor | University community members | |
| Trigger | User clicked the “Join Rides” button. | |
| Precondition | User at the Manage Rides page. | |
|  | Step | Description |
| Main Flow | 1 | User is required to enter a destination. |
| 2 | System displays available rides. |
| 3 | Send the information of selected ride. |
| 4 | System direct back to “Manage Rides” page |
| Alternate Flow – Location unavailable | 1-1 | Display a message about location unavailable. |
| Alternate Flow – Rides unavailable | 2-1 | Display a message about rides unavailable. |
| Alternate Flow – Cancel join rides | 3-1 | System direct back to “Manage Rides” page |

Table 2.1 Use Case Specification – Join Rides

A diagram of a flowchart

AI-generated content may be incorrect.

Figure 2.2 Activity Diagram – Join Rides

### 3.1.3 Create Rides

|  |  |  |
| --- | --- | --- |
| Use Case | Create Rides | |
| Purpose | University community members may create a ride for others to join the rides. | |
| Actor | University community members | |
| Trigger | User clicked the “Create Rides” button. | |
| Precondition | User at the Manage Rides page. | |
|  | Step | Description |
| Main Flow | 1 | User is required to enter a location. |
| 2 | User is required to enter a destination. |
| 3 | User is required to select a time. |
| 4 | Store the rides information into system. |
| 5 | System direct back to “Manage Rides” page |
| Alternate Flow – Location unavailable | 1-1 | Display a message about location unavailable. |
| Alternate Flow – Destination unavailable | 2-1 | Display a message about destination unavailable. |
| Alternate Flow – Cancel create rides | 4-1 | System direct back to “Manage Rides” page |

Table 2.2 Use Case Specification – Create Rides

A diagram of a process

AI-generated content may be incorrect.

Figure 2.3 Activity Diagram – Create Rides

### 3.1.4 Set Preference

|  |  |  |
| --- | --- | --- |
| Use Case | Set Preference | |
| Purpose | University community members may customize their preferences such as upload picture, edit names and edit self-description. | |
| Actor | University community members | |
| Trigger | User clicked the “Edit Preference” button. | |
| Precondition | User at the main page. | |
|  | Step | Description |
| Main Flow | 1 | System will display a profile picture and a text box with the user’s current name and user’s current self-description. |
|  | 2 | User may upload a new profile picture |
|  | 3 | User may edit their name. |
|  | 4 | User may edit their self-description |
|  | 5 | Store the user preference information into system |
|  | 6 | System direct back to “Main” page |
| Alternate Flow – Cancel Editing | 5-1 | User cancels the action. System discard changes and direct back to “Main page” |
| Alternate Flow – Upload failed | 2-1 | Display a message if the uploaded picture format is unsupported or upload fails. |
| Alternate Flow – Name field empty | 3-1 | Display a message if the name field is blank. |

Table 2.3 Use Case Specification – Set Preference

A diagram of a software flowchart

AI-generated content may be incorrect.

Figure 2.4 Activity Diagram – Set Preference

### 3.1.5 Review Drivers

|  |  |  |
| --- | --- | --- |
| Use Case | Review Drivers | |
| Purpose |  | |
| Actor | University community members | |
| Trigger |  | |
| Precondition | User at the Manage Rides page. | |
|  | Step | Description |
| Main Flow | 1 |  |
| Alternate Flow - |  |  |

Table 2.4 Use Case Specification – Review Drivers

Figure 2.5 Activity Diagram – Review Drivers

### 3.1.6 View Parking Status

|  |  |  |
| --- | --- | --- |
| Use Case | View Parking Status | |
| Purpose |  | |
| Actor | University community members | |
| Trigger |  | |
| Precondition |  | |
|  | Step | Description |
| Main Flow | 1 |  |
| Alternate Flow - |  |  |

Table 2.5 Use Case Specification – View Parking Status

Figure 2.6 Activity Diagram – View Parking Status

### 3.1.7 Book Parking

|  |  |  |
| --- | --- | --- |
| Use Case | Book Parking | |
| Purpose | University community members may book a parking to join the rides. | |
| Actor | University community members | |
| Trigger | User clicked “Book Now” button while viewing current parking status | |
| Precondition | User views current parking status | |
|  | Step | Description |
| Main Flow | 1 | Redirects to booking form |
|  | 2 | User selects a location and confirms booking |
|  | 3 | System sends the booking request to database to validates |
|  | 4 | User proceeds to the booking payment |
|  | 5 | System sends booking success message |
|  | 6 | System direct back to “Manage Rides” page |
| Alternate Flow - | 2-1 | User selected location is not available and display message about location is not available |
|  | 3-1 | Display the message about validation failed (all parking is occupied) |
|  | 4-1 | Display the message about payment is failed |

Table 2.6 Use Case Specification – Book Parking

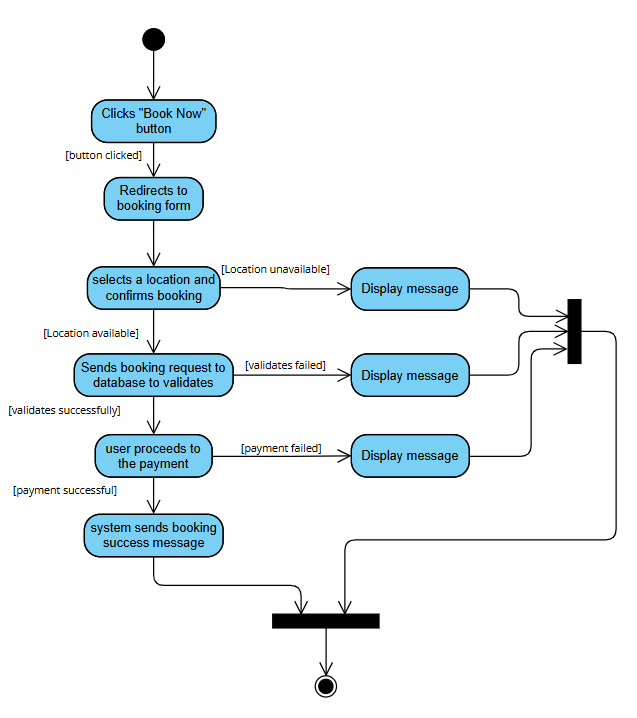


Figure 2.7 Activity Diagram – Book Parking

### 3.1.8 Approve Rides

|  |  |  |
| --- | --- | --- |
| Use Case | Approve Rides | |
| Purpose |  | |
| Actor | Admin | |
| Trigger |  | |
| Precondition |  | |
|  | Step | Description |
| Main Flow | 1 |  |
| Alternate Flow - |  |  |

Table 2.7 Use Case Specification – Approve Rides

Figure 2.8 Activity Diagram – Approve Rides

### 3.1.9 Manage Users

|  |  |  |
| --- | --- | --- |
| Use Case | Manage Users | |
| Purpose |  | |
| Actor | Admin | |
| Trigger |  | |
| Precondition |  | |
|  | Step | Description |
| Main Flow | 1 |  |
| Alternate Flow - |  |  |

Table 2.8 Use Case Specification – Manage Users

Figure 2.9 Activity Diagram – Manage Users

### 3.1.10 Monitoring Parking

|  |  |  |
| --- | --- | --- |
| Use Case | Monitoring Parking | |
| Purpose | Monitor real-time parking space availability and system usage | |
| Actor | Admin | |
| Trigger | Admin clicks a button like “Monitor Parking” on the admin dashboard | |
| Precondition | Admin is logged into the system | |
|  | Step | Description |
| Main Flow | 1 | Request parking data from database when “Monitor Parking” button is clicked |
|  | 2 | Get current parking status from database |
|  | 3 | Return latest parking data from database to admin dashboard |
|  | 4 | Display data on the admin dashboard |
|  | 5 | Show real-time parking info |
| Alternate Flow - | 1.1 | Request failed during data fetching and display message about request failed |
|  | 2.1 | Failed to get data from database and display message about fetching failed |

Table 2.9 Use Case Specification – Monitoring Parking

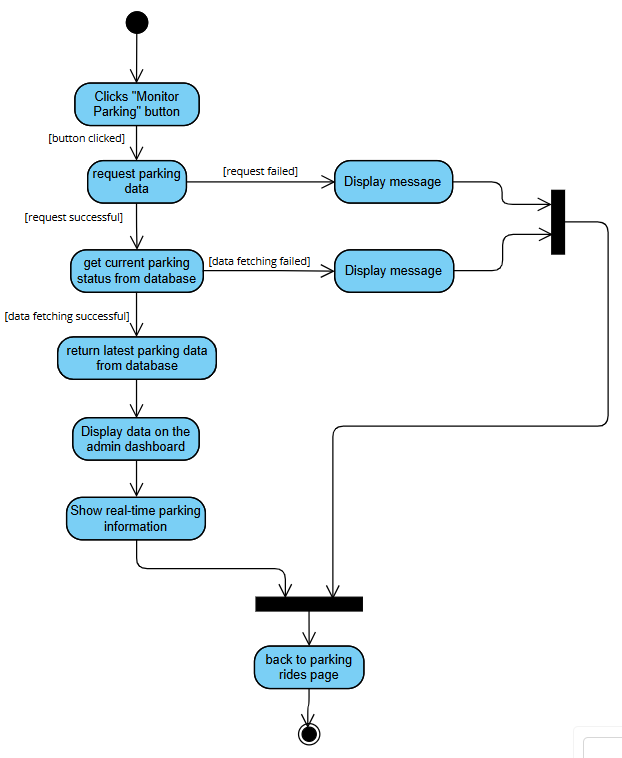


Figure 2.10 Activity Diagram – Monitoring Parking

## 3.2 Performance Requirements

(Mapped to 9.6.14 Performance Requirements)

Specify performance requirements, both static and dynamic, including response times, throughput, and scalability. These should be measurable with clear, quantitative targets.

Example:

The system shall respond to user queries within 2 seconds under a normal load.

## 3.3 Usability Requirements

(Mapped to 9.6.13 Usability Requirements)

Specify the usability objectives, including ease of use, learnability, efficiency, and user

satisfaction. These should be quantifiable and aligned with user needs.

Example:

The interface shall allow users to perform primary tasks within 3 clicks.

## 3.4 Interface Requirements

(Mapped to 9.6.11 External Interfaces and 9.6.4 System Interfaces, User Interfaces, Hardware

Interfaces, Software Interfaces, Communications Interfaces)

Specify all system interfaces, including external systems, user interfaces, hardware, and

communications.

### 3.4.1 System Interfaces:

Interfaces with external systems or hardware.

Example: The system will integrate with the university’s authentication system (LDAP).

### 3.4.2 User Interfaces:

Describe the layout and interaction elements, e.g., navigation,

buttons, data entry fields.

Example: The web interface will use a responsive layout with a fixed top navigation bar

for easy access to key features.

### 3.4.3 Hardware Interfaces:

Specify hardware connections, devices, and communication protocols.

Example: The system shall support USB-connected fingerprint readers for user

authentication.

### 3.4.4 Software Interfaces:

Describe interactions with other software or APIs.

Example: The system will interact with a third-party cloud service for file storage (e.g.,

Amazon S3).

### 3.4.5 Communications Interfaces:

Specify protocols, message formats, and network requirements.

Example: The system will use HTTPS for secure communication between client and

server.

## 3.5 Logical Database Requirements

(Mapped to 9.6.15 Logical Database Requirements)

Describe key data entities, relationships, and constraints. This could include an EntityRelationship (ER) diagram or class diagram.

Example:

The “Application” entity has attributes such as applicationID, title, and submissionDate, and it is

related to the “Reviewer” entity.

## 3.6 Design Constraints

(Mapped to 9.6.16 Design Constraints)

List any restrictions or limitations imposed on the design of the software, whether they are from

external standards, regulations, or technical limitations.

Examples:

The user interface must comply with the university’s branding guidelines.

## 3.7 Software System Attributes

(Mapped to 9.6.18 Software System Attributes)

Specify the required attributes of the software product, which affect its quality and

performance:

• Reliability: The system should be able to recover from a crash within 1 minute.

• Availability: The system should be available 99.9% of the time during working hours

(Monday through Friday, 8 AM to 6 PM).

• Security: The system should use role-based access control (RBAC) and encryption for all

sensitive user data.

• Maintainability: The system should follow best coding practices and be modular to

facilitate updates.

• Portability: The software should be able to run on both Linux and Windows servers

without additional configuration.

## 3.8 Supporting Information

(Mapped to 9.6.20 Supporting Information)

Any additional supporting information, including:

a) sample input/output formats, descriptions of cost analysis studies or results of questionnaires

or any other elicitation techniques;

b) supporting or background information that can help the readers of the SRS;

c) a description of the problems to be solved by the software; and

d) special packaging instructions for the code and the media to meet security, export, initial

loading or other requirements.

The SRS should explicitly state whether or not these information items are to be considered part

of the requirements.

Example:

Sample input/output formats for key system functions (e.g., CSV format for data export).

# 4. Verification

## 4.1 Verification Approach

(Mapped to 9.6.19 Verification)

Specify how the system will be verified, including methods, responsible parties, timing, and

locations.

Example:

• How: Functional testing, unit testing, and system integration testing will be used to

verify system performance.

• Who: Verification will be conducted by the product team and quality assurance (QA)

department.

• When: Verification will occur at key milestones in the development cycle (e.g., after

each sprint).

• Where: Verification activities will take place in the QA testing environment.

## 4.2 Verification Criteria

Define the criteria against which the software will be verified. These should align with the

functional and quality requirements.

Example:

The response time for a search query should be less than 3 seconds under normal load.

# 5. Appendices

## 5.1 Assumptions and Dependencies

(Mapped to 9.6.8 Assumptions and Dependencies)

List any assumptions and dependencies that impact the software development process or its requirements.

Example:

The system depends on the availability of the university's student database for user authentication.

## 5.2 Acronyms and Abbreviations

CRSP – Campus Ride-Sharing Platform with Parking System Integration

SRS – System Requirement Specification

IEEE – Institute of Electrical and Electronics Engineers

API - Application Programming Interface