Group I

Software Requirements Specifications

Campus Ride-Sharing Platform with Parking System Integration

(CRSP)

Version: 1.0

Date: 23/05/2025

Release By: Group I

**Document Control**

|  |  |
| --- | --- |
| Document Name | Campus Parking Management System |
| Reference Number | CRSP\_SRS |
| Version | 1.0 |
| Project Code | CRSP\_V1 |
| Status | In-use |
| Date Released | 23/05/2025 |

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

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Release Date | Section | Amendments |
| 1.0 | 23/05/2025 | All | Original Document |

**Table of Content**

[**1.0 Introduction** 1](#_Toc198923554)

[**1.1 Purpose** 1](#_Toc198923555)

[**1.2 Scope** 1](#_Toc198923556)

[**1.3 Product Overview** 2](#_Toc198923557)

[**1.3.1 Product Perspective** 2](#_Toc198923558)

[**1.3.2 Product Functions** 3](#_Toc198923559)

[**1.3.3 User Characteristics** 4](#_Toc198923560)

[**1.3.4 Limitations** 4](#_Toc198923561)

[**1.4 Definitions** 4](#_Toc198923562)

[**2.0 References** 6](#_Toc198923563)

[**3.0 Requirements** 7](#_Toc198923564)

[**3.1 Functions** 7](#_Toc198923565)

[**3.1.1Manage Rides** 8](#_Toc198923566)

[**3.1.2 Join Rides** 9](#_Toc198923567)

[**3.1.3 Create Rides** 10](#_Toc198923568)

[**3.1.4 Set Preference** 11](#_Toc198923569)

[**3.1.5 Review Drivers** 12](#_Toc198923570)

[**3.1.6 View Parking Status** 13](#_Toc198923571)

[**3.1.7 Book Parking** 14](#_Toc198923572)

[**3.1.8 Approve Rides** 15](#_Toc198923573)

[**3.1.9 Manage Users** 16](#_Toc198923574)

[**3.1.10 Monitoring Parking** 17](#_Toc198923575)

[**3.2** **Performance Requirements** 18](#_Toc198923576)

[**3.3 Usability Requirements** 18](#_Toc198923577)

[**3.4 Interface Requirements** 19](#_Toc198923578)

[**3.4.1 System Interfaces:** 19](#_Toc198923579)

[**3.4.2 User Interfaces:** 19](#_Toc198923580)

[**3.4.3 Hardware Interfaces:** 22](#_Toc198923581)

[**3.4.4 Software Interfaces:** 22](#_Toc198923582)

[**3.4.5 Communications Interfaces:** 22](#_Toc198923583)

[**3.5 Logical Database Requirements** 23](#_Toc198923584)

[**3.6 Design Constraints** 25](#_Toc198923585)

[**3.7 Software System Attributes** 26](#_Toc198923586)

[**3.8 Supporting Information** 27](#_Toc198923587)

[**4.0 Verification** 28](#_Toc198923588)

[**4.1 Verification Approach** 28](#_Toc198923589)

[**4.2 Verification Criteria** 28](#_Toc198923590)

[**5.0 Appendices** 29](#_Toc198923591)

[**5.1 Assumptions and Dependencies** 29](#_Toc198923592)

[**5.2 Acronyms and Abbreviations** 30](#_Toc198923593)

# 

# **1.0 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.

# **2.0 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.

# **3.0 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 | University community members may review drivers after completing a ride. | |
| Actor | University community members | |
| Trigger | User clicked the “Review Drivers” button. | |
| Precondition | User at the Manage Rides page. | |
|  | Step | Description |
| Main Flow | 1 | System displays ride history |
|  | 2 | User selects a completed ride |
|  | 3 | User clicks “Review Driver” |
|  | 4 | System display a review form |
|  | 5 | User fill in review form |
|  | 6 | Store the review information into system |
| Alternate Flow –  No completed ride | 1-1 | Display a message about no completed rides |

Table 2.4 Use Case Specification – Review Drivers

A diagram of a software system

AI-generated content may be incorrect.

Figure 2.5 Activity Diagram – Review Drivers

### **3.1.6 View Parking Status**

|  |  |  |
| --- | --- | --- |
| Use Case | View Parking Status | |
| Purpose | University community members may check the availability of parking spaces in real time | |
| Actor | University community members | |
| Trigger | User click “View Parking Status” button | |
| Precondition | User at the main page | |
|  | Step | Description |
| Main Flow | 1 | System display parking locations and availability status |
|  | 2 | User select a parking locations |
|  | 3 | System show detailed information about the selected parking location |
|  | 4 | User click “Book parking” button |
|  | 5 | System direct to “Book Parking” page |
| Alternate Flow –  Booking not available | 4-1 | Display a message if the selected parking location is full or not available for booking |
|  | 4-2 | System remains on the “View Parking Status” page |

Table 2.5 Use Case Specification – View Parking Status

A diagram of parking location

AI-generated content may be incorrect.

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

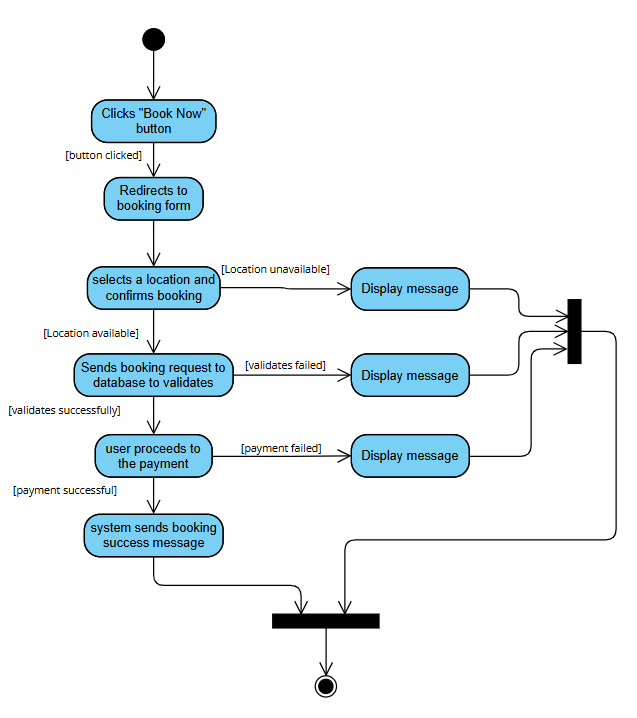


Figure 2.7 Activity Diagram – Book Parking

### **3.1.8 Approve Rides**

|  |  |  |
| --- | --- | --- |
| Use Case | Approve Rides | |
| Purpose | To allow administrators to approve or reject ride-sharing requests posted by users to ensure compliance with campus policies. | |
| Actor | Admin | |
| Trigger | A new ride request is submitted by a user. | |
| Precondition | The admin is logged into the system and has access to the ride approval module. | |
|  | Step | Description |
| Main Flow | 1 | Admin accesses the ride approval section from the dashboard. |
| 2 | System displays a list of pending ride requests. |
| 3 | Admin reviews ride details (e.g., driver, time, destination). |
| 4 | Admin selects a ride to approve or reject. |
| 5 | Admin submits the decision. |
| 6 | System updates ride status and notifies the user. |
| Alternate Flow - | 2.1 | Display message of no pending ride requests. |
| 4.1 | Admin flag the ride for revision instead of approving or rejecting. |

Table 2.7 Use Case Specification – Approve Rides

A diagram of a system

Description automatically generated

Figure 2.8 Activity Diagram – Approve Rides

### **3.1.9 Manage Users**

|  |  |  |
| --- | --- | --- |
| Use Case | Manage Users | |
| Purpose | To enable the administrator to manage user accounts, including activating, deactivating, or modifying user roles. | |
| Actor | Admin | |
| Trigger | Admin selects 'Manage Users' from the system dashboard. | |
| Precondition | The admin has administrative privileges and is logged into the system. | |
|  | Step | Description |
| Main Flow | 1 | Admin navigates to the user management section. |
|  | 2 | System displays all registered users. |
|  | 3 | Admin selects a user to view details. |
|  | 4 | Admin modifies user status or role (activate, deactivate, assign role). |
|  | 5 | Admin submits the changes. |
|  | 6 | System updates the user's profile and confirms the action. |
| Alternate Flow - | 2.1 | System display message of no user. |

Table 2.8 Use Case Specification – Manage Users

A diagram of a system

Description automatically generated

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

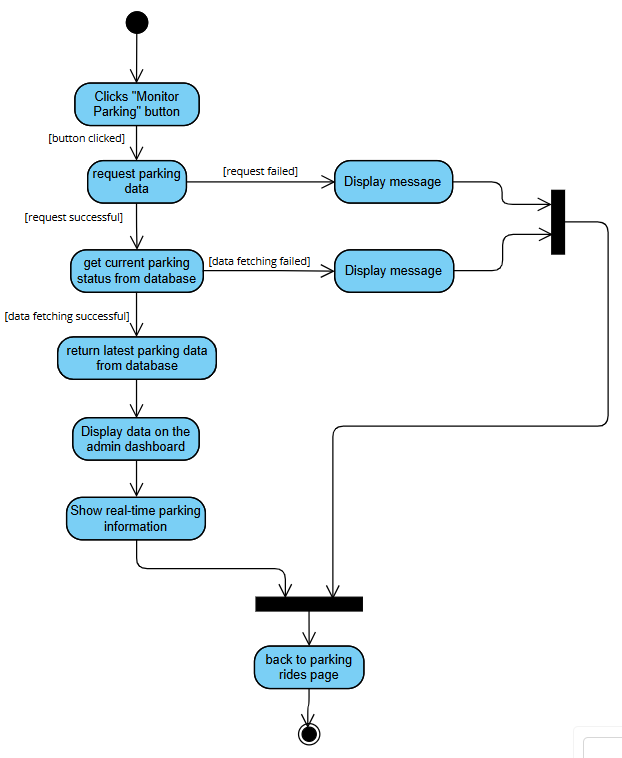


Figure 2.10 Activity Diagram – Monitoring Parking

## **Performance Requirements**

The CRSP shall meet the following performance requirements:

* Authentication and login processes shall complete within 3 seconds under normal network conditions.
* The system shall respond to user interactions within 2 seconds.
* The platform shall handle at least 100 ride-sharing transactions and 200 parking status queries per minute during peak hours.
* The system shall be scalable to support an increase of up to 200% in user load during semester start and end periods.
* Notifications sent via the Campus Email System shall be delivered within 1 minute after user create or join ride.
* Parking availability data from the Parking Management System shall be updated in the application every 20 seconds.

## **3.3 Usability Requirements**

The CRSP shall meet the following usability requirements:

* The user interface shall be intuitive enough that 90% of new users can successfully perform basic tasks (join rides, view parking status) without external help within their first 15 minutes of use.
* The “View Parking Status” actions shall be accessible within 2 clicks from the main screen.
* New users shall be able to learn to use the basic task (join rides, view parking status) with less than 15 minutes of guided interaction.
* A tutorial guide shall be provided for first-time users.
* Admins shall be able to review and approve ride or parking data in under 3 minutes per task.
* Frequent users shall be able to complete a task under 60 seconds.

## **3.4 Interface Requirements**

### **3.4.1 System Interfaces:**

The system will interface with the following external systems:

University Digital ID System: Used for user authentication and identity verification during login and registration.

Parking Management System: Used to retrieve real-time data about parking availability and to update user-specific parking assignments.

University Email System: Utilized to send notifications such as ride confirmations, schedule updates, and administrative messages.

### **3.4.2 User Interfaces:**

The platform will be available as a responsive web application and a mobile application. Key elements include:

Navigation: A top-fixed navigation bar for quick access to core modules (Home, My Rides, Book Parking, Admin Panel).

Forms: Login, registration, ride creation, and profile update forms with intuitive input validation.

Map Interface: Embedded map showing routes, meeting points, and parking zones with real-time updates.

Notifications: In-app pop-ups and email-based alerts for ride status, reminders, and system announcements.

Accessibility: Ensuring support for screen readers and keyboard navigation.

University Community Members:

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | REQ\_001 | Version | 1.0 |
| Item | Join Rides | | |
| Description | Allows users to search for and join available rides. | | |
| Purpose | Enable ride-sharing among campus community members. | | |
| Format | Mobile | Valid Range | Valid ride listings |
| Related I/O | Ride list input, confirmation output. | | |
| Author | Low Mun Kit | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | REQ\_002 | Version | 1.0 |
| Item | Set Preferences (User Profile) | | |
| Description | Allows users to customize profile settings (vehicle info, seat availability). | | |
| Purpose | Personalize ride-sharing preferences. | | |
| Format | Mobile | Valid Range | Text, dropdowns |
| Related I/O | Profile info input/output | | |
| Author | Low Mun Kit | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | REQ\_003 | Version | 1.0 |
| Item | Create Rides | | |
| Description | Enables users to post new ride listings. | | |
| Purpose | Facilitate carpool coordination. | | |
| Format | Form with date/time pickers and dropdowns | Valid Range | Logical inputs only |
| Related I/O | Ride form input, ride summary output | | |
| Author | Low Mun Kit | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | REQ\_004 | Version | 1.0 |
| Item | Review Drivers | | |
| Description | Allows passengers to leave feedback and ratings. | | |
| Purpose | Improve ride safety and quality. | | |
| Format | Rating stars + comments | Valid Range | 1–5 stars |
| Related I/O | Ride log input, rating display output | | |
| Author | Low Mun Kit | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | REQ\_005 | Version | 1.0 |
| Item | Manage Rides | | |
| Description | Users can edit or cancel their scheduled rides. | | |
| Purpose | Allow real-time ride management. | | |
| Format | Buttons and toggles | Valid Range | Scheduled rides only |
| Related I/O | Ride list input/output | | |
| Author | Low Mun Kit | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | REQ\_006 | Version | 1.0 |
| Item | View Parking Status | | |
| Description | Provides live parking availability data. | | |
| Purpose | Help users plan ahead before arriving on campus. | | |
| Format | Embedded map UI | Valid Range | Real-time status |
| Related I/O | Parking API feed | | |
| Author | Low Mun Kit | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | REQ\_007 | Version | 1.0 |
| Item | Book Parking | | |
| Description | Users can reserve parking slots based on ride status. | | |
| Purpose | Ensure parking is available for drivers. | | |
| Format | Form with calendar | Valid Range | Available time slots |
| Related I/O | Parking system input/output | | |
| Author | Low Mun Kit | | |

Admin:

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | REQ\_008 | Version | 1.0 |
| Item | Approve Rides | | |
| Description | Admin can review and approve/reject posted rides. | | |
| Purpose | Maintain control and ensure rule compliance. | | |
| Format | Admin dashboard | Valid Range | Approved/pending |
| Related I/O | Ride database input/output | | |
| Author | Low Mun Kit | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | REQ\_009 | Version | 1.0 |
| Item | Manage Users | | |
| Description | Admin can manage user accounts, block users, or reset credentials. | | |
| Purpose | Control system access and maintain data integrity. | | |
| Format | Admin dashboard | Valid Range | All active users |
| Related I/O | User database | | |
| Author | Low Mun Kit | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement ID | REQ\_010 | Version | 1.0 |
| Item | Monitor Parking | | |
| Description | Admin can track real-time parking statistics and user parking activities. | | |
| Purpose | Enable analytical and operational oversight. | | |
| Format | Live analytics UI | Valid Range | Parking activity |
| Related I/O | Parking logs, dashboard visuals | | |
| Author | Low Mun Kit | | |

### **3.4.3 Hardware Interfaces:**

The system will be deployed in an environment requiring minimal hardware dependency but will support:

Mobile Devices: GPS access for location tracking and geofencing.

Server Hardware: Backend services hosted on virtual or dedicated servers supporting at least 8 GB RAM, SSD storage, and quad-core CPUs.

### **3.4.4 Software Interfaces:**

The application will interact with several external software and APIs, including:

Map API (Google Maps API): For route optimization, meeting point marking, and visual navigation.

University Authentication API: For secure identity verification.

Email API (Microsoft Graph): For sending user emails and system alerts.

Parking API: To fetch and update live parking availability data.

Database System: Microsoft SQL Server for structured data storage and access control.

### **3.4.5 Communications Interfaces:**

The system will use secure and efficient communication protocols to support real-time operations:

HTTPS: All client-server communications will be encrypted using HTTPS over TLS 1.3.

WebSocket: Used for real-time updates of parking space availability and ride status changes.

RESTful APIs: Standardized interfaces will be used for all server-side requests and third-party system integrations.

SMTP: For sending email notifications and system alerts.

JSON: Default message format for API payloads; XML support will be considered based on legacy system needs.

## **3.5 Logical Database Requirements**

Entity Descriptions

User:

Represents a university community members and admin. Each user has a id, name and role.

* userID is the primary key

Ride:

Created by a user. Contains ride details such as starting location, destination, scheduled time.

* rideID is the primary key
* userID is the foreign key reference User table userID

Review

Records feedback by users. Contain comment.

* reviewID is the primary key
* rideID and userID are foreign key reference User and Ride table

ParkingSlot

Represents a praking slot on campus. Contains location and availability status.

* slotID is the primary key
* userID is the foreign key reference User table userID for the user who has reserved the slot.
* Availability is a Boolean indicating whether the slot is available or not.

RideMember

A junction table that links user and rides to represent user who have joined a specific ride.

* Composite primary key which is rideID and userID

Relationship description:

* User to RideMember to Ride: A many-to-many relationship between users and rides through RideMember.
* User to Review to Ride: A user can submit one review per ride
* User to Parking Slot: A user can book a parking slot

A diagram of a computer

AI-generated content may be incorrect.

Figure 2.11 Entity Relationship Diagram

## **3.6 Design Constraints**

**University Branding Compliance:**

* The user interface must align with the university's official branding guidelines, including colours, fonts, logos, and overall visual identity to ensure consistency and familiarity for users.

**Authentication Integration:**

* The platform must support integration with the university’s digital ID verification system for secure access by students, faculty, and staff.

**Platform Accessibility:**

* The system must comply with common web accessibility standards (e.g., WCAG 2.1) to ensure usability for users with disabilities.

**Technology Stack:**

* The design should be compatible with frameworks such as Django (Python), Java, and SQLite or similar relational databases to ensure ease of development, maintenance, and integration.

**Data Privacy Regulations:**

* The system must adhere to institutional and legal data protection regulations (e.g., GDPR-like rules for user consent and personal data handling) since it involves sensitive personal data and location tracking.

**Network Dependency:**

* Real-time parking data and ride-matching features must rely on stable internet connectivity. Offline functionality is limited and should be clearly documented.

**Deployment Environment:**

* The software must be deployable on university-managed Linux and Windows servers without requiring extensive custom configuration.

**Cross-Platform Support:**

* The application should be designed to operate on both desktop and mobile platforms, including web browsers and Android/iOS devices, without requiring separate codebases.

## **3.7 Software System Attributes**

* **Reliability**:  
  The system should deliver reliable updates of parking availability and ride-matching. In the event of a crash, it should recover in under 1 minute to not cause service disruption.
* **Availability**:  
  The platform should maintain at least 99.9% uptime between peak campus times (Monday to Friday, 7:00 AM – 10:00 PM), which encompasses students' and employees' primary use times.
* **Security**:  
  The system will implement role-based access control (RBAC) to restrict user permissions. All user sensitive data, personal data, and location history shall be encrypted using industry best practices protocols.
* **Maintainability**:  
  Software shall be coded with best coding practices using modular architecture. This will make it easier to do future upgrades, debugging, and scalability.
* **Portability**:  
  The application should run smoothly on both Linux and Windows server environments. No major reconfiguration should be required for deployment across different operating systems.

## **3.8 Supporting Information**

**a) Sample Input/Output Formats**:

* Input: User login credentials (e.g., campus email and password).
* Output: List of available ride-sharing options, parking lot availability in JSON or tabular format.
* Export Format: Data logs can be exported in CSV format for administrative review.

**b) Supporting/Background Information**:  
The project responds to growing parking space deficits and campus transportation inefficiency by encouraging ride-sharing and integrating digital ID verification for secure entry.

**c) Problem Description**:  
Parking and traffic limitations exist in campus populations. The goal is to provide a shared solution that enables easy sharing of rides and optimal parking space utilization.

**d) Special Packaging Instructions**:  
Deployment packages must include a README that includes setup instructions, secure API key configurations, and permissions. Deployment packages should be signed and verified prior to being installed on university servers to protect the installation package.

**Note**: These supporting information items are **not considered binding requirements** unless explicitly marked as such during the development phase.

# **4.0 Verification**

## **4.1 Verification Approach**

* How:
  + Unit testing for individual features such as join ride, parking booking and create ride.
  + Manual functional testing by group members to ensure the system behaves as expected.
  + User simulation by group members acting as end users.
* Who:
  + All verification will be done by the project group members.
* When:
  + After completing each major feature
  + Before submitting the project
* Where:
  + Testing will be conducted on group members personal machines.

## **4.2 Verification Criteria**

* All system functions must perform as described in the functional requirements
* User actions like creating a ride, booking a parking must complete without errors.
* System should display accurate parking availability.
* All data must be saved and retrieved correctly.

# **5.0 Appendices**

## **5.1 Assumptions and Dependencies**

The development and functionality of the Campus Ride-Sharing Platform are based on the following assumptions and external dependencies:

* **University Authentication Access**  
  The application is based on the availability and stability of the university’s digital ID authentication system for user (students, staff, and faculty) verification.
* **Parking Management API Integration**  
  It is assumed that the campus parking system has a stable and accessible API for retrieving real-time parking availability data.
* **User Internet Connectivity**  
  The app assumes that users have stable internet connectivity while using the application, particularly for features such as ride matching and real-time updates.
* **User Device Compatibility**  
  Users will likely access the system using modern web browsers or smartphones capable of running the web/mobile interface of the platform.
* **Map and Location Services**  
  The system depends on external mapping and location services (e.g., Google Maps API or equivalent) for route guidance, distance calculations, and location-based carpool matching.
* **Technology Stack Availability**  
  It is assumed that the required frameworks (e.g., Django, Python, SQLite, or equivalents) will be available and supported throughout development and deployment.
* **User Honesty and Cooperation**  
  The system assumes that users will provide accurate information when offering or requesting rides and will engage in ethical use of the platform.
* **University Policy Compliance**  
  It is assumed that necessary approvals and cooperation from university departments (e.g., IT services, transportation management) will be granted in accordance with campus policies.

## **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

RAM - Random Access Memory

SMTP - Simple Mail Transfer Protocol

SSD - Solid State Drives

XML - Extensible Markup Language

I/O – Input and output

GPS - Global Positioning System