Task 5 – Software Requirements Specification (SRS)



**Project**

CSE6224 Software Requirement Engineering   
Term 2510

|  |  |
| --- | --- |
| **Tutorial Section** | TT2L |
| **Group** | Group E |
| **Project Title** | Campus Ride-Sharing Platform with  Parking System Integration |

|  |  |
| --- | --- |
| **Name** | **Student ID** |
| Kelven Yee Kai Wen | 1211111244 |
| Koh Xuan Lin | 1211109618 |
| Ow Ka Sheng | 1211108820 |
| Shazreen Binti Sheridan | 243UC247P3 |

**Table of Contents**

[**1.0** **Introduction** 1](#_Toc198926891)

[**1.1** **Purpose** 1](#_Toc198926892)

[**1.2** **Scope** 2](#_Toc198926893)

[**1.2.1** **Functional Scope** 2](#_Toc198926894)

[**1.2.2** **Non-Functional Scope** 4](#_Toc198926895)

[**1.2.3** **Stakeholder Identification & Analysis** 4](#_Toc198926896)

[**1.3** **Product Overview** 5](#_Toc198926897)

[**1.3.1** **Product Perspective** 5](#_Toc198926898)

[**1.3.2** **Product Functions** 8](#_Toc198926899)

[**1.3.3** **User Characteristics** 12](#_Toc198926900)

[**1.3.4** **Limitations** 14](#_Toc198926901)

[**1.4** **Definition** 16](#_Toc198926902)

[**2.0** **References** 17](#_Toc198926903)

[**3.0** **Requirements** 18](#_Toc198926904)

[**3.1** **Functions** 18](#_Toc198926905)

[**3.1.1** **Login** 19](#_Toc198926906)

[**3.1.2** **Register** 20](#_Toc198926907)

[**3.1.3** **Manage Profile** 21](#_Toc198926908)

[**3.1.4** **Offer Ride** 22](#_Toc198926909)

[**3.1.5** **Request Ride** 23](#_Toc198926910)

[**3.1.6** **Modify Ride** 24](#_Toc198926911)

[**3.1.7** **Chat with Driver** 25](#_Toc198926912)

[**3.1.8** **Rating and Feedback** 26](#_Toc198926913)

[**3.1.9** **View Parking** 27](#_Toc198926914)

[**3.1.10** **View Notification** 28](#_Toc198926915)

[**3.1.11** **Redeem Rewards** 29](#_Toc198926916)

[**3.1.12** **SOS Message** 30](#_Toc198926917)

[**3.1.13** **Generate Report** 31](#_Toc198926918)

[**3.1.14** **Monitor Metrics** 32](#_Toc198926919)

[**3.2** **Performance Requirements** 34](#_Toc198926920)

[**3.3** **Usability Requirements** 38](#_Toc198926921)

[**3.4** **Interface Requirements** 43](#_Toc198926922)

[**3.4.1** **System Interfaces** 43](#_Toc198926923)

[**3.4.2** **User Interfaces** 44](#_Toc198926924)

[**3.4.3** **Hardware Interfaces** 44](#_Toc198926925)

[**3.4.4** **Software Interfaces** 45](#_Toc198926926)

[**3.4.5** **Communications Interfaces** 45](#_Toc198926927)

[**3.4.6** **Memory Constraints** 46](#_Toc198926928)

[**3.4.7** **Operations** 47](#_Toc198926929)

[**3.4.8** **Site Adaptation Requirements** 48](#_Toc198926930)

[**3.4.9** **Interfaces with Services** 49](#_Toc198926931)

[**3.5** **Logical Database Requirements** 50](#_Toc198926932)

[**3.5.1** **Data Dictionary** 52](#_Toc198926933)

[**3.6** **Design Constraints** 57](#_Toc198926934)

[**3.7** **Software System Attributes** 59](#_Toc198926935)

[**3.8** **Supporting Information** 63](#_Toc198926936)

[**4.0** **Verification** 66](#_Toc198926937)

[**4.1** **Verification Approach** 66](#_Toc198926938)

[**4.2** **Verification Criteria** 68](#_Toc198926939)

[**5.0** **Appendices** 70](#_Toc198926943)

[**5.1** **Assumptions and Dependencies** 70](#_Toc198926944)

[**5.2** **Acronyms and Abbreviations** 71](#_Toc198926945)

[**5.3** **Glossary** 72](#_Toc198926946)

# **Introduction**

## **Purpose**

The purpose of this document is to outline the specific requirements needed to develop the Campus Ride-Sharing Platform with Parking System Integration, designed for use by students, staff, and faculty of Multimedia University (MMU). This platform aims to encourage eco-friendly commuting, reduce campus congestion, and optimize parking space usage by allowing verified users to share rides and view real-time parking availability.

The development team, project manager, quality assurance team, university stakeholders, and IT support personnel involved in the system's integration and deployment are the target audience for this document.

## **Scope**

### **Functional Scope**

**User Onboarding & Digital ID Verification**

* MMU single sign-on (SSO) ensures only current students, staff, or faculty can register and log in.
* User profiles include name, university ID, photo, contact details, and vehicle information (make, model, license plate).

**Ride Offer & Request**

* Drivers publish ride offers with origin, destination, date/time, available seats, and vehicle type (e.g., sedan, SUV, van).
* Passengers search for or filter rides by schedule, proximity, or preferred vehicle type, then request to join.
* Matching is manual: drivers approve or decline incoming requests through the app.

**Parking Availability & Rewards**

* Real-time campus parking occupancy map, highlighting “carpool-only” zones and general lots.
* Reward Points System:
  + **Earn**: Drivers and riders earn points per trip (e.g., 10 points for drivers, 5 points for riders).
  + **Redeem**: Points can be redeemed for priority parking reservations in high demand lots, campus bookstore vouchers, or meal discounts at campus cafeterias.
  + **Leaderboard**: Top carpoolers receive badges and extra rewards to encourage regular participation.

**Communication & Notifications**

* Push notifications for ride confirmations, cancellations, parking spot reservations, and incentive milestones.
* In-app chat between matched drivers and riders for coordinating exact pickup/drop-off locations.

**Ride History**

* Users view a chronological log of past rides, seats offered/used, parking spots reserved, and rewards earned.

**Safety & Emergency Features**

* Users can add emergency contact information.
* Includes an SOS button that alerts selected contacts and shares ride details for safety.

**User Ratings & Feedback System**

* After each ride, drivers and passengers can rate and comment on each other.

**Admin Dashboard & Reporting**

* Web-based dashboard for administrators to view key KPIs immediately (total rides, active carpools, parking occupancy trends, reward redemptions).
* Custom report generation and export (CSV/PDF) on ride usage, parking utilization, reward redemption, and user growth over any selected period.

### **Non-Functional Scope**

**Performance & Reliability**

* Support an initial user base of up to 500 simultaneous active sessions in Phase 1.
* Basic fallback if parking data is delayed.

**Security & Privacy**

* All user data protected; only MMU-affiliated users can access the system.
* Data encrypted in transit (TLS) and at rest.

**Usability**

* Mobile-first design targets iOS and Android.
* Simple, intuitive screens (3-tap workflows) for creating/joining rides.

### **Stakeholder Identification & Analysis**

|  |  |
| --- | --- |
| **Stakeholder** | **Role & Needs** |
| Students | * Affordable, eco‑friendly commute * Easy scheduling |
| Lecturers & Staff | * Reliable rides * Secure, university-verified access |
| Parking Admin Team | * Accurate occupancy data * Define carpool zones |
| IT & Security Office Staff | * Ensure compliance with university policies * Approve SSO |
| Project Team | * Plan, build, test, and deploy the platform |
| Student Representative Council (SRC) | * Facilitate partnerships with external businesses to support incentive programs |

## **Product Overview**

### **Product Perspective**

The Campus Ride-Sharing Platform with Parking System Integration is an integrated module in MMU's online environment, providing ride-sharing coordination and parking management. It connects students, faculty, and employees with university infrastructure for secure, efficient, and environmentally friendly trips.

This platform communicates with several MMU infrastructure's central components, enabling seamless data exchanges between security, transport logistics, and parking management. It communicates with the MMU SSO Authentication to ensure user entry authentication, employs the Campus Parking Database for real-time tracking, and employs a Carpool Matching Engine to process ride requests and approval. Additionally, a notification system provides the alerts on ride confirmations, parking spots, incentive notifications, and emergency alerts.

Part of MMU's overall drive for increased mobility on campus, the platform supports safe and verified ride-sharing, better use of parking space, and sustainable behavior encouragement through reward-based incentives. Fully integrated with MMU's IT infrastructure, security controls, and parking facilities management, the platform offers a convenient commuting experience in compliance with the university policy.

#### **System Interfaces**

* User authentication via MMU SSO for validated logins.
* Campus Parking Database API for live parking spot availability.
* Ride-matching engine for processing carpool requests and approvals.

#### **User Interfaces**

* Mobile UI optimized for iOS & Android with an interactive dashboard.
* Three-step workflows for ride matching, approvals, and parking lookup.
* In-app messaging & notifications to improve coordination.

#### **Hardware Interfaces**

* GPS tracking for ride location validation.
* Mobile device sensor compatibility (Wi-Fi, GPS, push notifications).
* Campus parking control integration to enforce carpool zones.

#### **Software Interfaces**

* Push notification services linked to MMU’s existing IT infrastructure.
* Database integration for user profile management, ride history, and incentive tracking.
* API-based connectivity with MMU's parking and security systems.

#### **Communications Interfaces**

* Secure HTTPS protocol for encrypted data transmissions.
* Campus-wide notification integration for ride status alerts and rewards.

#### **Memory Constraints**

* Lightweight mobile storage usage for cached ride and parking data.
* Optimized low-bandwidth transactions to reduce overhead.

#### **Operations**

* User-initiated ride matching and approval system.
* Automated ride confirmations & parking availability updates.
* Scheduled leaderboard tracking to promote high-participation incentives.

#### **Site Adaptation Requirements**

* Campus-wide maps integration displaying active parking zones.
* Compliance with MMU branding and security policies.

#### **Interfaces with Services**

* Cloud-based ride management & authentication for scalability.
* Potential third-party integrations for expanding ride networks beyond MMU.

### **Product Functions**

The Campus Ride-Sharing Platform with Parking System Integration shall provide the following primary functions:

**User Account Management**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Accessible Role** |
| FR-1 | Allow users to register using their MMU credentials | Driver, Passenger |
| FR-2 | Enable users to create and manage their profile information | Driver, Passenger |
| FR-3 | Support user preference settings for ride matching | Driver, Passenger |

**Ride Offering and Requesting**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Accessible Role** |
| FR-4 | Allow users to offer rides by specifying origin, destination, time, and available seats | Driver, Passenger |
| FR-5 | Enable users to request rides by specifying pickup location, destination, and time | Driver, Passenger |
| FR-6 | Support recurring ride scheduling for regular commutes | Driver, Passenger |
| FR-7 | Provide ride modification and cancellation capabilities | Driver, Passenger |

**Ride Matching and Coordination**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Accessible Role** |
| FR-8 | Allow users to manually browse, select, and confirm ride matches based on preferences. | Driver, Passenger |
| FR-9 | Facilitate in-app communication between drivers and passengers | Driver, Passenger |

**Parking System Integration**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Accessible Role** |
| FR-10 | Display real-time parking availability across campus zones | Driver, Passenger |
| FR-11 | Reserve priority parking spots for verified carpools | Driver, Passenger |
| FR-12 | Provide navigation to available parking areas | Driver, Passenger |

**Safety and Security**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Accessible Role** |
| FR-13 | Verify user identity through MMU SSO authentication | Driver, Passenger |
| FR-14 | Provide SOS emergency alert functionality | Driver, Passenger |
| FR-15 | Enable ride tracking for designated emergency contacts | Driver, Passenger |
| FR-16 | Support rider rating and review system | Driver, Passenger |
| FR-17 | Enable voice recording during rides for safety purposes | Driver, Passenger |

**Reward Management**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Accessible Role** |
| FR-18 | Award eco-points for successful carpooling participation | Driver, Passenger |
| FR-19 | Provide a leaderboard of top carpoolers | Driver, Passenger |
| FR-20 | Enable redemption of rewards (parking credits, campus vouchers) | Driver, Passenger |

**Reporting and Analytics**

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Accessible Role** |
| FR-21 | Generate personal ride history and statistics | Admin |
| FR-22 | Provide system usage reports for administrators | Admin |
| FR-23 | Calculate environmental impact metrics | Admin |
| FR-24 | Support customizable data export functionality | Admin |

Figure 1.0 shows the context of Campus Ride-Sharing Platform with Parking System Integration.

A diagram of a system

AI-generated content may be incorrect.

**Figure 1.0 Campus Ride-Sharing Platform with Parking System Integration Context Diagram**

The users authenticate login via the MMU SSO Authentication System, book rides, send SOS messages, and get parking lot availability and ride matching updates. The Reward System tracks complete rides and redemption alerts, and the Chat System is used for one-to-one chat between the passenger and the driver. In case of emergencies, the Emergency Contact System processes SOS messages and sends messages with user and location status. Ride-sharing coordination is facilitated through the Carpool Matching Engine, responsible for handling manual match requests and confirmations. To keep users informed in a timely fashion, the notification system provides notice regarding ride events as well as parking spaces. The Campus Parking Database provides real-time feedback to queries about parking, again increasing accessibility speed. All these elements balance increasing campus transport efficiency with a safe and user-friendly experience.

### **User Characteristics**

The system will serve the following user groups, each with specific characteristics and expectations:

|  |  |  |
| --- | --- | --- |
| **Role** | **Description** | **Characteristic** |
| Student | Primary user group representing approximately 70% of the user base | * Typically, aged 18-25 with high technological proficiency * Have varied and sometimes irregular schedules * Usually operate on limited budgets, making cost-sharing appealing * Often live in clusters near campus or in designated student housing * Primary motivation: cost savings and convenience |
| Lecturer & Staff | Representing approximately 15% of the user base | * Regular working hours (typically 8:00 AM - 5:00 PM) * Consistent commuting patterns * Primary motivation: convenience and potential for social connections |

|  |  |  |
| --- | --- | --- |
| System Administrator | Small group responsible for system maintenance and monitoring | * Require comprehensive understanding of all system features * Need access to administrative functions and reports * Technical expertise allowing system configuration and troubleshooting * Primary focus: system efficiency, security, and user satisfaction |
| All User |  | * Must possess basic mobile device proficiency * Require MMU digital credentials for authentication * Need reliable internet access for real-time features * Should understand basic navigation concepts * Will need clear guidelines on emergency procedures |

The system shall accommodate these diverse user groups by providing intuitive interfaces, clear instructions, and tailored messaging appropriate to each user's role and technical proficiency level.

### **Limitations**

The Campus Ride-Sharing Platform with Parking System Integration operates under the following constraints and limitations:

**Technical Limitations**

* The system operates only within the geographical boundaries of MMU campuses and immediate surroundings (within 10km radius).
* Real-time parking data accuracy depends on the reliability of MMU's existing parking sensors and infrastructure.
* GPS accuracy is limited to approximately 5-10 meters, which may affect precise pickup coordination.
* The platform requires internet connectivity for core functionalities; offline mode supports only limited features.
* Mobile application performance may vary across different device specifications and operating system versions.

**Operational Limitations**

* The system can support a maximum of 1,000 concurrent active ride sessions.
* Ride matching will operate only during campus operational hours plus an additional buffer of 2 hours before and after (5:00 AM - 11:00 PM).
* Emergency SOS features require campus security personnel availability, which may fluctuate.
* System maintenance windows will be scheduled weekly, during which certain features may be unavailable.
* User verification is contingent upon the reliability of MMU's SSO authentication system.

**Regulatory Limitations**

* The system does not provide commercial ride-sharing services and cannot be used for profit-generating activities.
* The platform is not a substitute for public transportation or commercial ride-hailing services.
* Insurance coverage for ride-sharing activities is not provided by the system or the university.
* Data retention policies comply with Malaysian personal data protection regulations, limiting historical data availability.
* The system does not enforce legal agreements between riders beyond the user terms and conditions.

**Business Limitations**

* Initial rollout will be limited to the main campus, with phased expansion to satellite campuses.
* The reward system operates within the constraints of the university's allocated budget for sustainability initiatives.
* Integration with third-party services is subject to existing university contracts and procurement procedures.
* System customization capabilities are constrained by the development team's resources and timeline.
* Priority parking spot allocation is subject to availability and university parking management policies.

## **Definition**

|  |  |
| --- | --- |
| **Carpool Matching Engine** | The system component responsible for pairing ride requests and ride offerings based on route and timing compatibility |
| **Campus Parking System** | A real-time system that tracks available parking spaces and occupancy rates across different campus zone |
| **Eco-Points** | Reward points earned through carpool participation that can be redeemed for rewards such as priority parking or vouchers |
| **Emergency Contact System** | A module designed to notify pre-configured contacts and campus security in case of ride-sharing emergencies |
| **KPIs** | Key Performance Indicators for tracking ride efficiency. |
| **Push Notifications** | Automated alerts sent to users via the mobile application to inform them of ride matches, confirmations, parking updates, or emergency messages |
| **REST API** | an [application programming interface (API)](https://www.ibm.com/topics/api) that conforms to the design principles of the representational state transfer (REST) architectural style, a style used to connect distributed hypermedia systems |
| **SOS** | Emergency notification feature for safety alerts. |
| **SSO** | Single Sign-on authentication for users. |

# **References**

IEEE. (2018). *ISO/IEC/IEEE 29148:2018 Systems and software engineering—Life cycle processes— Requirements engineering.* [https://www.iso.org/standard/72089.html](https://www.iso.org/standard/72089.html%20)

*MMU Sustainability Policies*. (n.d.). Retrieved May 17, 2025, from <https://www.mmu.edu.my/wp-content/uploads/2025/01/MMU-Sustainability-Policy-new-2025.pdf>

*BlaBlaCar*. Retrieved May 23, 2025, from https://www.blablacar.co.uk/

*Kummute*. Retrieved May 23, 2025, from https://www.kummute.com.my/

# **Requirements**

## **Functions**

This section details the functional requirements of the Campus Ride-Sharing Platform with Parking System Integration through use cases and their specifications. The functionality is organized based on the primary user roles: Student, Faculty/Staff, and Administrator. Figure 2.0 shows the overall use case of the Campus Ride-Sharing Platform with Parking System Integration:

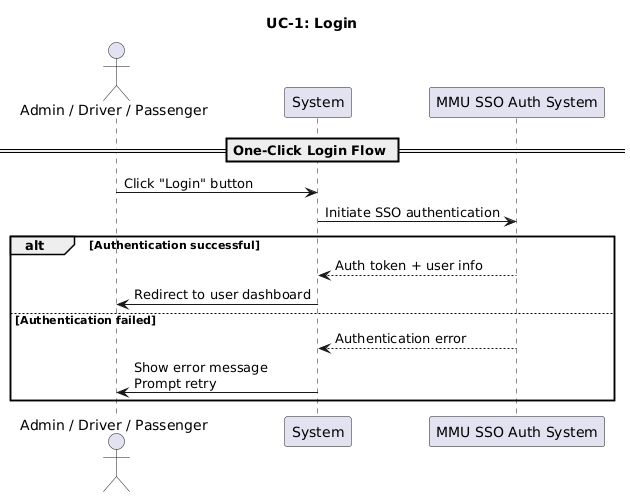
**A diagram of a system

AI-generated content may be incorrect.**

**Figure 2.0 Campus Ride-Sharing Platform with Parking System Integration Overall Use Case**

### **Login**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-1 | **Use Case** | Login |
| **Description** | Allows users to securely access the platform using their MMU credentials. Ensures authorized access to features based on user role | | |
| **Actors** | Admin, Driver, Passenger, MMU SSO Authentication System | | |
| **Preconditions** | User must be registered | | |
| **Postconditions** | User gains access to the platform | | |
| **Basic Flow** | 1. Users click login button. 2. The system verifies credentials via MMU SSO Authentication System. 3. If valid, the system grants access to the dashboard. 4. If invalid, the system displays an error message and prompts retry. | | |



### **Register**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-2 | **Use Case** | Register |
| **Description** | Enables new users to create accounts using their MMU digital ID for identity verification | | |
| **Actors** | Driver, Passenger | | |
| **Preconditions** | User has valid MMU credentials | | |
| **Postconditions** | Users can create accounts | | |
| **Basic Flow** | 1. User enters MMU digital ID details. 2. The system verifies credentials via MMU SSO. 3. User fills in profile information (name, car details, preferences). 4. System stores detail and confirms registration. 5. User receives confirmation email or notification | | |



### **Manage Profile**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-3 | **Use Case** | Manage Profile |
| **Description** | Allows users to update personal details, preferences, and vehicle information in their profile | | |
| **Actors** | Driver, Passenger | | |
| **Preconditions** | User must be authenticated | | |
| **Postconditions** | Profile details are successfully updated | | |
| **Basic Flow** | 1. Users access the profile settings page. 2. Users modify personal details, preferences, or vehicle information. 3. System validates and updates changes. 4. System stores updated details in the database. 5. User receives confirmation message | | |



### **Offer Ride**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-4 | **Use Case** | Offer Ride |
| **Description** | Enables drivers to create a ride listing, specifying route, time, and seat availability | | |
| **Actors** | Driver | | |
| **Preconditions** | Users must be authenticated and have a registered vehicle | | |
| **Postconditions** | Ride listing is created and available for passenger matching | | |
| **Basic Flow** | 1. The driver selects the “Offer Ride" option. 2. Driver enters the ride details (origin, destination, time, seats available). 3. The system verifies details and matches with potential passengers. 4. System stores ride listing in the database. 5. System sends notifications to matched passengers | | |



### **Request Ride**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-5 | **Use Case** | Request Ride |
| **Description** | Allows passengers to find and request available rides based on their preferred pickup location and destination | | |
| **Actors** | Passenger | | |
| **Preconditions** | User must be authenticated | | |
| **Postconditions** | Ride Request is sent and matched with an available driver | | |
| **Basic Flow** | 1. Passenger selects the “Request Ride" option. 2. Passenger enters pickup location, destination, and preferred time. 3. The system verifies details and matches requests with available drivers. 4. The system confirms match and notifies passengers. 5. Passenger receives ride confirmation details | | |



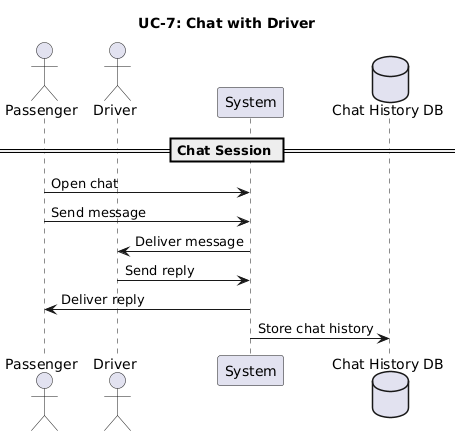
### **Modify Ride**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-6 | **Use Case** | Modify Ride |
| **Description** | Let users update ride details such as destination, timing, or passenger count | | |
| **Actors** | Driver, Passenger | | |
| **Preconditions** | User must have an active ride listing or request | | |
| **Postconditions** | Ride details are successfully modified | | |
| **Basic Flow** | 1. User accesses ride modification settings. 2. User updates ride details (timing, destination, passenger count). 3. The system verifies modifications and updates ride listing. 4. System notifies affected users of changes | | |



### **Chat with Driver**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-7 | **Use Case** | Chat with Driver |
| **Description** | Provides an in-app messaging feature for drivers and passengers to coordinate rides effectively | | |
| **Actors** | Passenger | | |
| **Preconditions** | User must have a matched ride session | | |
| **Postconditions** | Messages exchanged between rider and driver are stored and delivered | | |
| **Basic Flow** | 1. Passenger selects chat options within ride details. 2. Passenger sends messages. 3. The system delivers messages to the driver. 4. The driver responds and the system delivers a reply. 5. The system retains chat history for future reference. | | |



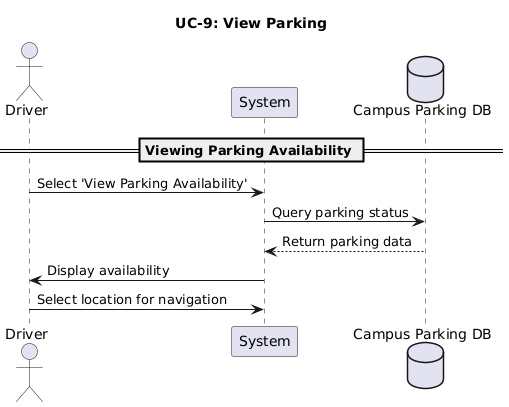
### **Rating and Feedback**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-8 | **Use Case** | Rating and Feedback |
| **Description** | Allows users to provide reviews and rate their ride-sharing experience | | |
| **Actors** | Driver, Passenger | | |
| **Preconditions** | User must have completed a ride | | |
| **Postconditions** | Feedback is stored and reflected in user ratings | | |
| **Basic Flow** | 1. User accesses the “Rate Ride" option after trip completion. 2. User selects rating and enters optional comments. 3. The system stores feedback and updates driver/passenger profile. 4. The system calculates an average rating score and displays it in the system. | | |



### **View Parking**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-9 | **Use Case** | View Parking |
| **Description** | Displays real-time campus parking availability for drivers | | |
| **Actors** | Driver, Passenger, Campus Parking Database, Third-Party Mapping Service | | |
| **Preconditions** | User must be authenticated | | |
| **Postconditions** | Parking availability is displayed with real-time updates | | |
| **Basic Flow** | 1. Driver selects "View Parking Availability." 2. System requests real-time parking data from Campus Parking Database. 3. Campus Parking Database returns available parking spots and location status. 4. System displays parking options to the driver. 5. Driver selects a parking location. 6. System interacts with Third-Party Mapping Service for navigation. | | |



### **View Notification**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-10 | **Use Case** | View Notification |
| **Description** | Shows important alerts regarding ride status, parking updates, and system notifications | | |
| **Actors** | Driver, Passenger | | |
| **Preconditions** | User must be authenticated | | |
| **Postconditions** | Notifications are viewed and acknowledged | | |
| **Basic Flow** | 1. User accesses the notification center. 2. The system retrieves relevant notifications (ride matches, confirmations, alerts). 3. User views detail and acknowledge messages. | | |



### **Redeem Rewards**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-11 | **Use Case** | Redeem Rewards |
| **Description** | Enables users to exchange accumulated eco-points for rewards like priority parking or vouchers | | |
| **Actors** | Driver, Passenger | | |
| **Preconditions** | User must have accumulated eco-points or ride-sharing rewards | | |
| **Postconditions** | Rewards are redeemed and reflected in the user account | | |
| **Basic Flow** | 1. User selects the “Redeem Rewards" option. 2. User browses available rewards. 3. The user selects a reward and confirms redemption. 4. System deducts eco-points and updates balance. 5. The system provides confirmation and details on reward collection | | |

A diagram of a reward system

AI-generated content may be incorrect.

### **SOS Message**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-12 | **Use Case** | SOS Message |
| **Description** | Provides emergency alert functionality for users in distress during rides | | |
| **Actors** | Driver, Passenger, Emergency Contact System | | |
| **Preconditions** | User must be in an active ride session | | |
| **Postconditions** | Emergency alerts are sent to campus security and designated contacts | | |
| **Basic Flow** | 1. User presses the “SOS Alert” button. 2. The system verifies emergency status and location. 3. System sends alerts to Emergency Contact System, notifying security personnel and designated contacts. 4. Security personnel respond accordingly. | | |



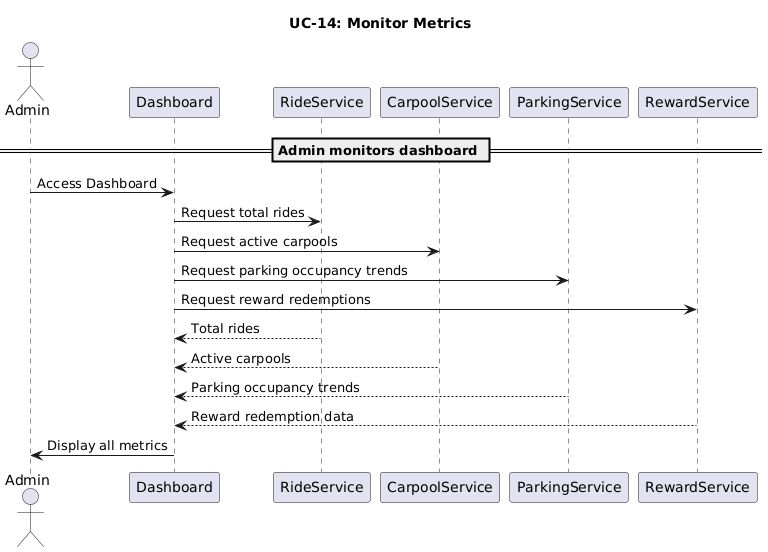
### **Generate Report**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-13 | **Use Case** | Generate Report |
| **Description** | Allows administrators to generate reports on ride-sharing activities, system usage, and environmental impact | | |
| **Actors** | Admin | | |
| **Preconditions** | Administrator must have access to reporting functions | | |
| **Postconditions** | System-generated reports are available for review and export | | |
| **Basic Flow** | 1. Admin accesses reporting dashboard. 2. Admin selects report criteria (date range, user activity, ride statistics). 3. The system retrieves data and generates structured reports. 4. Admin views report details and downloads/export data. | | |



### **Monitor Metrics**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC-14 | **Use Case** | Monitor Metrics |
| **Description** | Grants administrators control over system settings, permissions, and configurations | | |
| **Actors** | Admin | | |
| **Preconditions** | Admin must be authenticated with appropriate privileges | | |
| **Postconditions** | Key performance indicators (KPIs) are displayed successfully on the dashboard | | |
| **Basic Flow** | 1. Admin logs into the web-based dashboard. 2. Admin navigates to the KPI overview section. 3. The system retrieves and displays real-time data, including:  * Total rides * Active carpools * Parking occupancy trends * Reward redemptions  1. Admin reviews the KPIs to monitor system performance. 2. The system ensures data is up-to-date and securely presented. | | |



## **Performance Requirements**

The Campus Ride-Sharing Platform with Parking System Integration must meet the following performance requirements to ensure user satisfaction and system reliability:

**Response Time Requirements**

|  |  |
| --- | --- |
| **Requirement ID** | PR-1 |
| **Requirement** | User Interface Responsiveness |
| **Description** | * The mobile application shall load the main dashboard within 3 seconds of launch under normal network conditions. * Menu transitions shall occur within 0.5 seconds of user selection. * Form submissions shall be processed within 2 seconds, with feedback provided to the user. |

|  |  |
| --- | --- |
| **Requirement ID** | PR-2 |
| **Requirement** | Parking System Integration |
| **Description** | * Real-time parking availability data shall be updated at minimum every 2 minutes. * Parking availability queries shall return results within 3 seconds. * Parking space reservation confirmations shall be processed within 10 seconds. |

**Throughput Requirements**

|  |  |
| --- | --- |
| **Requirement ID** | PR-3 |
| **Requirement** | Concurrent Users |
| **Description** | * The system shall support a minimum of 500 concurrent users during normal operations. * During peak periods (8:00-10:00 AM and 4:00-6:00 PM), the system shall support up to 1,000 concurrent users. * Performance degradation shall not exceed 25% during peak usage periods. |

|  |  |
| --- | --- |
| **Requirement ID** | PR-4 |
| **Requirement** | Transaction Volume |
| **Description** | * The system shall process up to 100 ride requests per minute during peak periods. * The system should handle up to 50 ride matches per minute during peak periods. * The system should support up to 200 parking availability queries per minute. |

**Scalability Requirement**

|  |  |
| --- | --- |
| **Requirement ID** | PR-5 |
| **Requirement** | User Base Expansion |
| **Description** | * The system shall be scaled to accommodate a 50% increase in user base without performance degradation. * Database architecture shall support efficient expansion to handle increased data volume. * The system shall maintain performance metrics when expanded to additional campus locations. |

|  |  |
| --- | --- |
| **Requirement ID** | PR-6 |
| **Requirement** | Feature Expansion |
| **Description** | * Architecture shall support the addition of new features without requiring major redesign. * API endpoints shall be designed to accommodate additional functionality through version control. |

**Capacity Requirements**

|  |  |
| --- | --- |
| **Requirement ID** | PR-7 |
| **Requirement** | Data Storage |
| **Description** | * The system shall store ride history data for a minimum of 12 months. * User profiles and preferences shall be maintained indefinitely (until account deletion). * The database should be designed to efficiently handle up to 10,000 active users. |

|  |  |
| --- | --- |
| **Requirement ID** | PR-8 |
| **Requirement** | Network Bandwidth |
| **Description** | * Mobile data usage shall not exceed 5MB per hour during active use. * Backend systems shall support up to 50Mbps of data transfer during peak periods. |

**Reliability Requirements**

|  |  |
| --- | --- |
| **Requirement ID** | PR-9 |
| **Requirement** | Uptime |
| **Description** | * The system will maintain 99.5% uptime during academic semesters. * Scheduled maintenance shall occur during off-peak hours (typically 2:00-4:00 AM). * The maximum allowed unplanned downtime shall not exceed 1 hour per month. |

|  |  |
| --- | --- |
| **Requirement ID** | PR-10 |
| **Requirement** | Fault Tolerance |
| **Description** | * The system shall recover from crashes within 2 minutes without data loss. * Ride matching data shall be preserved in case of system failure.   User sessions shall be automatically restored after network interruptions. |

## **Usability Requirements**

The Campus Ride-Sharing Platform with Parking System Integration must meet the following usability requirements to ensure positive user experience for all target user groups:

**Learnability Requirements**

|  |  |
| --- | --- |
| **Requirement ID** | UR-1 |
| **Requirement** | Intuitive Interface |
| **Description** | * First-time users shall be able to complete the registration process without assistance within 5 minutes. * 90% of new users shall be able to successfully offer or request a ride within their first three attempts. * The system should provide an interactive tutorial for first-time users that can be completed in under 3 minutes. |

|  |  |
| --- | --- |
| **Requirement ID** | UR-2 |
| **Requirement** | Help and Documentation |
| **Description** | Context-sensitive help should be available for all major functions.   * The helpful documentation shall be searchable with relevant results appearing within 2 seconds. * Video tutorials shall be available for complex operations, with each tutorial lasting no longer than 2 minutes. |

**Efficiency Requirements**

|  |  |
| --- | --- |
| **Requirement ID** | UR-3 |
| **Requirement** | Task Completion |
| **Description** | * Regular users shall be able to complete a ride request in less than 30 seconds. * Regular users shall be able to offer a ride in less than 45 seconds. * Checking parking availability shall be achievable in less than 15 seconds from any screen. |

|  |  |
| --- | --- |
| **Requirement ID** | UR-4 |
| **Requirement** | Navigation Efficiency |
| **Description** | * Primary functions shall be accessible within 2 taps/clicks from the main dashboard. * Users shall be able to switch between primary functions without returning to the home screen. * The most recently used functions should be prominently displayed for quick access. |

**Satisfaction Requirements**

|  |  |
| --- | --- |
| **Requirement ID** | UR-5 |
| **Requirement** | User Satisfaction Metrics |
| **Description** | * The system shall achieve a minimum satisfaction rating of 4.0 out of 5.0 in user surveys. * The system shall maintain an app store rating of at least 4.2 out of 5.0. * Post-use surveys show that at least 85% of users would recommend the platform to others. |

|  |  |
| --- | --- |
| **Requirement ID** | UR-6 |
| **Requirement** | Visual Design |
| **Description** | * The interface shall comply with MMU branding guidelines for color schemes and typography. * The design shall be visually consistent across all screens and functions. * Animation and transitions shall be smooth and enhance rather than distract from the user experience. |

**Accessibility Requirements**

|  |  |
| --- | --- |
| **Requirement ID** | UR-7 |
| **Requirement** | Inclusive Design |
| **Description** | * The system shall comply with WCAG 2.1 Level AA accessibility standards. * Text elements have adjustable size options to accommodate users with visual impairments. * Color schemes should accommodate color-blind users with appropriate contrast ratios. |

|  |  |
| --- | --- |
| **Requirement ID** | UR-8 |
| **Requirement** | Device Compatibility |
| **Description** | * The mobile application should function properly on devices running iOS 13+ and Android 8.0+. * The interface shall be responsive and fully functional on screens from 4.7" to 10" diagonal. * Touch targets shall be at least 9mm in diameter for accessibility on all supported devices. |

**Error Prevention and Recovery**

|  |  |
| --- | --- |
| **Requirement ID** | UR-9 |
| **Requirement** | Error Prevention |
| **Description** | * Input fields shall validate data in real-time before submission. * Confirmation dialogs shall be presented for irreversible actions. * The system shall provide clear, non-technical error messages when issues occur. |

|  |  |
| --- | --- |
| **Requirement ID** | UR-10 |
| **Requirement** | Recovery Options |
| **Description** | * Users shall be able to cancel or modify ride requests up to 15 minutes before the scheduled departure. * The system shall provide a "reset to defaults" option for all customizable settings. * Form data shall be preserved if the application is unintentionally closed during input. |

**User Groups Accommodation**

|  |  |
| --- | --- |
| **Requirement ID** | UR-11 |
| **Requirement** | Student-Specific Feature |
| **Description** | * The interface shall highlight cost-saving benefits prominently for student users. * Quick access to popular campus destinations shall be available for frequent routes. * Budget tracking features shall help students monitor transportation costs. |

|  |  |
| --- | --- |
| **Requirement ID** | UR-12 |
| **Requirement** | Faculty/Staff Features |
| **Description** | * Schedule integration shall allow syncing with faculty/staff calendars for regular commutes. * Professional networking options shall be available for faculty/staff carpooling groups. * Priority notification settings shall be available for time-sensitive commuting needs. |

## **Interface Requirements**

This section defines all interfaces involved in the **Campus Ride-Sharing Platform with Parking System Integration,** including system interactions, user interface design, hardware connections, software dependencies, communication protocols, memory constraints, operational considerations, site adaptations, and service integrations.

### **System Interfaces**

|  |  |
| --- | --- |
| **Name** | University Authentication System (MMU SSO) |
| **Purpose** | Validates users via MMU credentials, ensuring only authorized students, staff, and faculty access the platform. |
| **Source of Input** | User-provided MMU login credentials. |
| **Destination of Output** | Verified user authentication result. |
| **Valid Range/Accuracy** | Strict access control ensuring 100% verified user identity. |
| **Units of Measure** | Authentication response time (ms). |
| **Timing** | Immediate verification upon login. |
| **Relationships** | Integrates with user management and security policies. |
| **Data Format** | Encrypted login data (TLS-secured JSON payload). |
| **Command Format** | API request to MMU SSO endpoint. |
| **Included Information** | User ID, session token. |

### **User Interfaces**

|  |  |
| --- | --- |
| **Name** | Mobile Application UI (iOS & Android) |
| **Purpose** | Provides an intuitive interface for ride management and parking integration. |
| **Source of Input** | User interactions via touchscreen. |
| **Destination of Output** | Ride requests, parking searches, notifications. |
| **Valid Range/Accuracy** | Optimized for mobile usability (responsive layout). |
| **Units of Measure** | Screen resolution, interaction speed (ms). |
| **Timing** | Instantaneous feedback (< 0.5s). |
| **Relationships** | Connects to carpool engine, parking database, and notification system. |
| **Data Format** | UI elements (HTML/XML for Android/iOS rendering). |
| **Command Format** | Touch inputs, gesture events. |
| **Included Information** | User selections, ride data, and parking preferences. |

### **Hardware Interfaces**

|  |  |
| --- | --- |
| **Name** | GPS & Sensor Interfaces |
| **Purpose** | Tracks ride location for safety and trip coordination. |
| **Source of Input** | Mobile device GPS and motion sensors. |
| **Destination of Output** | Real-time ride tracking updates. |
| **Valid Range/Accuracy** | ±5-10m location accuracy. |
| **Units of Measure** | Latitude/longitude, movement detection. |
| **Timing** | Refresh every 15s. |
| **Relationships** | Links to ride history, emergency alert system. |
| **Data Format** | GPS data (decimal degrees). |
| **Command Format** | Location update requests. |
| **Included Information** | User coordinates, movement speed. |

### **Software Interfaces**

|  |  |
| --- | --- |
| **Name** | Campus Parking Database API |
| **Purpose** | Fetches real-time parking availability. |
| **Source of Input** | Database query for parking spots. |
| **Destination of Output** | Parking availability response. |
| **Valid Range/Accuracy** | Live updates every 2 minutes. |
| **Units of Measure** | Occupancy percentage. |
| **Timing** | API query response within 3s. |
| **Relationships** | Links with carpool matching incentives. |
| **Data Format** | JSON response (spot ID, status). |
| **Command Format** | RESTful API request. |
| **Included Information** | Parking zone ID, availability status. |

### **Communications Interfaces**

|  |  |
| --- | --- |
| **Name** | Secure Data Transmission (HTTPS) |
| **Purpose** | Encrypts user communication, ride confirmations, and parking alerts. |
| **Source of Input** | Mobile app request. |
| **Destination of Output** | Server-side response. |
| **Valid Range/Accuracy** | AES-encrypted messages ensuring secure transactions. |
| **Units of Measure** | Data packet size (bytes). |
| **Timing** | Network latency-dependent, expected <100ms response. |
| **Relationships** | Integrated with notifications and system security. |
| **Data Format** | TLS-secured API messages. |
| **Command Format** | HTTPS request-response. |
| **Included Information** | Ride status, parking updates, emergency alerts. |

### **Memory Constraints**

|  |  |
| --- | --- |
| **Name** | Local Storage and Cache Management |
| **Purpose** | Minimizes data storage requirements by caching ride history and parking availability locally. |
| **Source of Input** | Ride requests, parking lookups, user preferences. |
| **Destination of Output** | Cached data for faster access. |
| **Valid Range/Accuracy** | Cached ride history retained for 12 months; parking availability refreshes every 2 minutes. |
| **Units of Measure** | Data size in MB. |
| **Timing** | Real-time cache updates, periodic cleanup of expired data. |
| **Relationships** | Links to ride history, parking database, user profile settings. |
| **Data Format** | JSON, indexed database. |
| **Command Format** | Local storage read/write operations. |
| **Included Information** | Cached ride requests and responses, temporarily stored parking availability, session-based user preferences. |

### **Operations**

|  |  |
| --- | --- |
| **Name** | Ride Matching and Parking Coordination |
| **Purpose** | Handles user-initiated ride sharing and parking spot identification. |
| **Source of Input** | User ride requests, parking availability queries. |
| **Destination of Output** | Matched ride details, available parking spaces. |
| **Valid Range/Accuracy** | 95% accuracy in matching based on user preferences. |
| **Units of Measure** | Number of requests per minute. |
| **Timing** | Ride matching completed within 5 seconds; parking updates every 2 minutes. |
| **Relationships** | Connects with user profile, parking system, notification engine. |
| **Data Format** | JSON responses. |
| **Command Format** | API requests for ride searching and parking lookup. |
| **Included Information** | Ride request details (origin, destination, seat count, timestamps), approved ride matches, parking allocation status. |

### **Site Adaptation Requirements**

|  |  |
| --- | --- |
| **Name** | Campus Parking Map and Access Controls |
| **Purpose** | Adjusts functionality based on MMU-specific parking zones and security policies. |
| **Source of Input** | Campus infrastructure data. |
| **Destination of Output** | Location-specific parking availability and restrictions. |
| **Valid Range/Accuracy** | Zone-based parking enforcement, mapped entry validation. |
| **Units of Measure** | Number of designated carpool zones. |
| **Timing** | Updates occur as per administrative configurations. |
| **Relationships** | Connects with parking system, user authentication, incentive tracking. |
| **Data Format** | GIS-enabled parking data. |
| **Command Format** | Database queries for real-time parking assignments. |
| **Included Information** | Campus map with designated carpool zones, branding elements for MMU policy compliance, site-specific parking constraints. |

### **Interfaces with Services**

|  |  |
| --- | --- |
| **Name** | Cloud-Based Ride Management & Security Services |
| **Purpose** | Supports authentication, ride storage, and notifications through cloud integration. |
| **Source of Input** | MMU SSO authentication, ride data submission. |
| **Destination of Output** | Secure storage of ride history, user profiles, reward calculations. |
| **Valid Range/Accuracy** | Secure encrypted transactions (TLS protocol). |
| **Units of Measure** | Transaction speed in ms. |
| **Timing** | Authentication <2s; ride storage <5s. |
| **Relationships** | Connected to MMU IT infrastructure, notification system, ride database. |
| **Data Format** | Secure REST API responses (JSON). |
| **Command Format** | HTTPS-secured API requests. |
| **Included Information** | Secure authentication tokens for MMU SSO, ride history logs stored in cloud databases, push notification preferences. |

## **Logical Database Requirements**

The Campus Ride-Sharing Platform with Parking System Integration comprises entities such as User, Passenger, Driver, Ride Offer, and Ride Request. These entities are interconnected through relationships that enable authentication, ride matching, and seamless user interactions.

A diagram of a computer

AI-generated content may be incorrect.

* The "User" entity has attributes such as userID, name, email, phone, and role, and it is related to the "Passenger”, "Driver", "Ride History", "MMU SSO Authentication", "Reward Points" and "Emergency Contact" entity.
* The "Driver" entity has attributes such as driverID, userID, vehicleInfo, rating and licensePlate, and it is related to the "User" and "Ride Offer" entity.
* The "Passenger" entity has attributes such as passengerID, userID, pickupLocation, and rating, and it is related to the "User" and "Ride Request" entity.
* The "Ride Offer" entity has attributes such as rideID, driverID, origin, destination, departureTime, availableSeats, and vechicleType, and it is related to the "Ride Request", "Ride History", "Campus Parking Database" and "Driver" entity.
* The "Ride Request" entity has attributes such as requestID, passengerID, rideID, and status, and it is related to the "Passenger" and "Ride Offer" entities.
* The "Ride History" entity has attributes such as historyID, userID, rideID, status, and timestamp, and it is related to the "User" and “Ride Offer" entity.
* The "MMU SSO Authentication" entity has attributes such as authenticationID, userID, method, status, timestamp, and it is related to the "User" entity.
* The "Reward Points" entity has attributes such as rewardID, userID, pointsEarned, redeemedRewards, and timestamp, and it is related to the "User" entity.
* The "Emergency Contact" entity has attributes such as contactID, userID, name, phone, and relationship, and it is related to the "User" entity.
* The "Campus Parking Database" entity has attributes such as parkingID, location, totalSlots, availableSlots and lastUpdated, and it is related to the "User" and “Ride Offer" entity.
* The "Chat" entity has attributes such as chatID, userID, message, and timestamp, and it is related to the "User" entity.
* The "Notification" entity has attributes such as notificationID, userID, message, and timestamp, and it is related to the "User" entity.

### **Data Dictionary**

The data dictionary defines the structure of the database tables, describing each field's data type, length, constraints, and purpose within the system.

**User Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| userID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the user |
| name | Short Text | 25 |  | Not Null | Full name of the user |
| email | Short Text | 25 |  | Not Null | Email address of the user |
| phone | Short Text | 25 |  | Not Null | User’s phone number |
| role | Short Text | 25 |  | Not Null | Role of the user (passenger, driver) |

**Passenger Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| passengerID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the passenger |
| userID | Short Text | 10 | FK | Not Null | Foreign key referencing User table |
| pickupLocation | Short Text | 50 |  | Not Null | Passenger’s pickup location |
| rating | Decimal | 3, 2 |  | Not Null | Passenger’s rating |

**Driver Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| driverID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the driver |
| userID | Short Text | 10 | FK | Not Null | Foreign key referencing User table |
| vehicleInfo | Short Text | 50 |  | Not Null | Driver’s vehicle details |
| rating | Decimal | 3, 2 |  | Not Null | Driver’s rating |
| licensePlate | Short Text | 25 |  | Not Null | Vehicle’s license plate number |

**Ride Offer Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| rideID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the ride offer |
| driverID | Short Text | 10 | FK | Not Null | Foreign key referencing Driver table |
| origin | Short Text | 50 |  | Not Null | Starting location |
| destination | Short Text | 50 |  | Not Null | End location |
| departureTime | Date/Time | 10 |  | Not Null | Departure time |
| availableSeats | Integer | 10 |  | Not Null | Number of available seats |
| vehicleType | Short Text | 25 |  | Not Null | Type of vehicle |

**Ride Request Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| requestID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the ride request |
| passengerID | Short Text | 10 | FK | Not Null | Foreign key referencing Passenger table |
| rideID | Short Text | 10 | FK | Not Null | Foreign key referencing Ride Offer table |
| status | Short Text | 25 |  | Not Null | Status of the ride request |

**Ride History Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| historyID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the ride history |
| userID | Short Text | 10 | FK | Not Null | Foreign key referencing User table |
| rideID | Short Text | 10 | FK | Not Null | Foreign key referencing Ride Offer table |
| status | Short Text | 25 |  | Not Null | Status of the ride history |
| timestamp | Date/Time | 10 |  | Not Null | Date and time of the ride |

**Campus Parking Database Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| parkingID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the campus parking system |
| location | Short Text | 50 |  | Not Null | Physical location |
| totalSlots | Integer | 10 |  | Not Null | Total parking slots |
| availableSlots | Integer | 10 |  | Not Null | Available parking slots |
| lastUpdated | Date/Time | 10 |  | Not Null | Timestamp of the last update |

**MMU SSO Authentication Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| authenticationID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the SSO authentication system |
| userID | Short Text | 10 | FK | Not Null | Foreign key referencing User table |
| method | Short Text | 25 |  | Not Null | Authentication method |
| status | Short Text | 15 |  | Not Null | Authentication status |
| timestamp | Date/Time | 10 |  | Not Null | Timestamp of authentication |

**Emergency Contact Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| contactID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the emergency contacts system |
| userID | Short Text | 10 | FK | Not Null | Foreign key referencing User table |
| name | Short Text | 25 |  | Not Null | Contact person’s name |
| phone | Short Text | 25 |  | Not Null | Contact phone number |
| role | Short Text | 25 |  | Not Null | Relationship to the user |

**Chat Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| chatID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the chat |
| userID | Short Text | 10 | FK | Not Null | Foreign key referencing User table |
| message | Short Text | 50 |  | Not Null | Message of the chat |
| timestamp | Date/Time | 10 |  | Not Null | Date and time of the chat |

**Notification Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Length** | **PK/FK** | **Null/Not Null** | **Description** |
| notificationID | Short Text | 10 | PK | Not Null | Primary key, unique identifier for the notification |
| userID | Short Text | 10 | FK | Not Null | Foreign key referencing User table |
| message | Short Text | 50 |  | Not Null | Message of the notification |
| timestamp | Date/Time | 10 |  | Not Null | Date and time of the notification |

## **Design Constraints**

The design constraints of the Campus Ride-Sharing Platform with Parking System Integration ensure a secure, efficient, and university-compliant ride-sharing experience while maintaining operational integrity.

**Compliance with MMU Policies**

* The user interface must adhere to Multimedia University (MMU) branding guidelines, including official colours, logos, and design elements.
* User authentication must follow MMU Single Sign-On (SSO) standards to verify only registered students, staff, and faculty.
* All user data must comply with MMU’s privacy and IT security policies to protect personal information.

**Technical Limitations**

* The system must operate within MMU’s digital infrastructure, meaning third-party integrations are restricted unless explicitly approved by the university.
* The mobile application must be compatible with iOS and Android devices, but initial deployment will support only the latest stable versions of each OS (minimum iOS 13, Android 8.0).
* The GPS accuracy is limited to 5–10 meters, affecting precise pickup location tracking.

**Regulatory Constraints**

* The platform does not support financial transactions (e.g., payment processing, fare splitting) due to university regulations.
* Ride-sharing activities must comply with Malaysian transportation policies, meaning only non-commercial rides are permitted.

**Scalability & Performance Constraints**

* The system must support up to 500 concurrent users in Phase 1, scaling up to 1,000 users in later releases.
* The ride-matching engine must process requests within 5 seconds to ensure quick response times.
* Parking occupancy updates every 2 minutes but refresh rates may depend on MMU’s existing infrastructure.

**Security & Data Protection**

* Access control must be role-based, restricting administrative functions to designated university personnel.
* Ride history and reward data must follow data retention policies, limiting storage to 12 months.

## **Software System Attributes**

This section specifies the required software quality attributes of the Campus Ride-Sharing Platform:

**Reliability**

|  |  |  |
| --- | --- | --- |
| **ID** | **Attribute** | **Measurement** |
| SSA-1 | The system shall achieve 99/5% uptime during university operational hours | Measured through system monitoring logs and reported as monthly uptime percentage |
| SSA-2 | The system shall have a Mean Time Between Failures (MBTF) of at least 720 hours | Measured through system monitoring and calculated monthly |
| SSA-3 | The system shall have a Mean Time To Repair (MTTR) of less than 2 hours for critical functions | Measured from incident logs and reported monthly |
| SSA-4 | The system shall maintain data integrity with zero corruption incidents | Verified through database consistency checks and audit logs |
| SSA-5 | The system shall implement automatic recovery from common error conditions | Validated through fault injection testing |

**Availability**

|  |  |  |
| --- | --- | --- |
| **ID** | **Attribute** | **Measurement** |
| SSA-6 | The system shall be available 24/7 with scheduled maintenance windows communicated 48 hours in advance | Tracked through system monitoring and maintenance logs |
| SSA-7 | The system shall implement redundancy for critical components to avoid single points of failure | Verified through architecture review and failover testing |
| SSA-8 | The system shall degrade gracefully under high load conditions, maintaining core functionality | Validated through load testing and performance monitoring |
| SSA-9 | The system shall remain operational during university network fluctuations | Verified through resilience testing during simulated network issues |
| SSA-10 | The system shall implement automatic failovers for critical services within 30 seconds | Measured during disaster recovery testing |

**Security**

|  |  |  |
| --- | --- | --- |
| **ID** | **Attribute** | **Measurement** |
| SSA-11 | The system shall prevent unauthorized access to personal and sensitive data | Verified through security testing and penetration tests |
| SSA-12 | The system shall encrypt all sensitive data both in transit and at rest | Verified through code review and security scanning |
| SSA-13 | The system shall maintain comprehensive audit logs of all security-relevant events | Inspected through log review and completeness verification |
| SSA-14 | The system shall resist common attack vectors (SQL injection, XSS, CSRF) | Validated through security testing and vulnerability scanning |
| SSA-15 | The system shall implement proper session management with secure timeout handling | Verified through security review and penetration testing |
| SSA-16 | The system shall restrict access based on user roles and permissions | Validated through access control testing |

**Maintainability**

|  |  |  |
| --- | --- | --- |
| **ID** | **Attribute** | **Measurement** |
| SSA-17 | The system shall be designed with modular architecture to facilitate component updates | Assessed through architecture review and coupling metrics |
| SSA-18 | The system shall implement comprehensive logging for troubleshooting | Verified through log coverage review and debug testing |
| SSA-19 | The system shall be accompanied by complete technical documentation | Evaluated through documentation review against prescribed standards |
| SSA-20 | The system shall support configuration changes without requiring code modification | Validated through configuration testing |
| SSA-21 | The system shall achieve a maintainability index score of at least 80 | Measured through static code analysis tools |

**Portability**

|  |  |  |
| --- | --- | --- |
| **ID** | **Attribute** | **Measurement** |
| SSA-22 | The web interface shall function correctly on major browsers (Chrome, Firefox, Safari, Edge) | Verified through cross-browser testing |
| SSA-23 | The mobile application shall function on both Android and iOS platforms | Validated through cross-platform testing |
| SSA-24 | The system shall utilize containerization for deployment flexibility | Verified through deployment testing in multiple environments |
| SSA-25 | The system shall implement responsive design for various screen sizes | Validated through multi-device testing |
| SSA-26 | The system database shall be compatible with both Oracle and PostgreSQL | Verified through database adapter testing |

**Scalability**

|  |  |  |
| --- | --- | --- |
| **ID** | **Attribute** | **Measurement** |
| SSA-27 | The system shall support linear scaling to accommodate up to 20,000 registered users | Validated through load testing and performance monitoring |
| SSA-28 | The system shall handle peak loads of 1,000 concurrent users without performance degradation | Measured through stress testing and response time monitoring |
| SSA-29 | The system shall support horizontal scaling of application servers | Verified through cluster deployment testing |
| SSA-30 | The system shall implement database sharding for future growth | Validated through database performance testing |
| SSA-31 | The system shall maintain response time under load through proper caching strategies | Measured through performance testing under increasing load |

**Usability**

|  |  |  |
| --- | --- | --- |
| **ID** | **Attribute** | **Measurement** |
| SSA-32 | The system shall achieve a System Usability Scale (SUS) score of at least 80 | Measured through user testing and surveys |
| SSA-33 | First-time users shall complete critical tasks successfully without assistance | Validated through usability testing with new users |
| SSA-34 | The system shall conform to university accessibility standards | Verified through accessibility testing and compliance review |
| SSA-35 | The system shall achieve a task completion rate of at least 95% for common functions | Measured through usability testing |
| SSA-36 | Users shall express satisfaction of at least 4 out of 5 in usability surveys | Measured through post-implementation user surveys |

## **Supporting Information**

This section provides supplemental materials, insights, and artifacts that support the requirements outlined in this SRS. These elements are not mandatory requirements unless otherwise stated but serve as context or guidance for the development and implementation teams.

**Sample Input/Output Information**

**Ride Request Input**

{

"userID": "1211111244",

"pickupLocation": "MMU Hostel A",

"destination": "MMU Faculty of Engineering",

"preferredTime": "08:30 AM"

}

**Ride Match Output**

{

"rideID": "RIDE9832",

"driverName": "Ali Bin Ahmad",

"vehicle": "Perodua Myvi (WXD2345)",

"pickupTime": "08:25 AM",

"estimatedArrival": "08:45 AM",

"matchConfidence": "92%"

}

**CSV Export Format for Admin Reports:**

userID,role,totalRides,ecoPoints,avgRating

1211109618,Driver,23,150,4.6

1211108820,Passenger,17,98,4.8

**Supporting & Background Information**

**Survey Results Summary:**

80% of students are open to carpooling if incentives are offered.

65% of staff expressed interest in parking space pre-reservations.

90% of respondents cited ride-sharing as a way to reduce stress and fuel costs.

Cost-Benefit Analysis Snapshot:

Initial Development Cost: RM 75,000 (internal dev team)

Estimated Annual Savings: RM 30,000 from reduced fuel usage and congestion costs.

ROI Timeline: 3 years through operational efficiency, reduced parking infrastructure strain, and eco-initiatives.

**Elicitation Techniques Used:**

Online questionnaires distributed to MMU students, staff, and faculty.

Stakeholder interviews with campus security and transport administrators.

Comparative analysis of existing ride-sharing platforms like GrabHitch, Waze Carpool.

**Description of the Problem to Be Solved**

MMU currently faces:

* Overutilized parking zones, especially during peak hours.
* Inefficient use of commuting resources, with many single-passenger vehicles.
* No centralized platform for secure, institution-backed carpooling coordination.
* Lack of visibility into real-time parking data, causing delays and frustration.

The proposed system solves these issues by offering:

* Verified, secure carpool matching via MMU SSO.
* Real-time parking availability data.
* Eco-incentives to promote sustainable transportation.
* Safety features (e.g., SOS alerts and ride tracking).

**Special Packaging Instructions**

* The application shall be packaged in the following formats:
  + Android APK and iOS IPA for mobile distribution via internal app stores or test environments.
  + Deployment Archive (ZIP) containing Docker containers for cloud/server deployment.
  + Database Seed File (SQL) with test users, vehicles, and parking data for QA environments.
* All code and packages must:
  + Comply with MMU IT Security Policies.
  + Be digitally signed using MMU’s DevOps certificates before deployment.
  + Follow versioning standards (e.g., v1.0.0-beta, v1.1.0-stable).
  + Include user manuals and developer documentation as PDFs in the /docs folder of the deployment archive.

# **Verification**

## **Verification Approach**

The verification approach defines how the requirements specified in this document will be validated to ensure the system meets stakeholder needs:

**Verification Methods**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Application** |
| Inspection | Systematic examination of requirements through reviews, walkthroughs, and audits | Used for verifying documentation completeness, consistency, and adherence to standards |
| Analysis | Evaluation of requirements through models, simulations, or theoretical analysis | Used for verifying system architecture, data models, and logical requirements |
| Demonstration | Showing that requirements are satisfied through prototype operation | Used for verifying user interface requirements and workflow functionality |
| Testing | Systematic operation of the system under control conditions to evaluate behavior | Used for verifying functional and performance requirements |
| Certification | Verification by an independent authority | Used for verifying compliance with external standards and regulations |

**Verification Process**

1. **Requirements Review**: All requirements will undergo peer review by stakeholders to ensure clarity, testability, and relevance
2. **Traceability Matrix**: A traceability matrix will be developed to link each requirement to its corresponding verification method and test case
3. **Prototyping and User Testing**: Key functional and usability requirements will be validated through prototype development and user testing
4. **Functional Testing**: Comprehensive test cases will be developed to verify functional requirements through black-box and white-box testing approaches
5. **Performance Testing**: Load testing, stress testing, and capacity testing will be conducted to verify performance requirements
6. **Security Assessment**: Security requirements will be verified through vulnerability scanning, penetration testing, and code review
7. **User Acceptance Testing**: End-users will validate that the system meets their needs through structured acceptance testing
8. **Compliance Verification**: External audits will verify compliance with applicable regulations and standards

## **Verification Criteria**



Each requirement category will be verified according to the following criteria:

**Functional Requirements**

* + Each function operates as specified
  + Functions handle both valid and invalid inputs appropriately
  + Functions properly integrate with other system components
  + User workflows complete successfully
  + All business rules are enforced correctly

**Performance Requirements**

* + Response times meet specified targets under normal and peak loads
  + System handles concurrent user limits
  + Resource utilization remains within specified boundaries
  + System remains stable during extended operation periods
  + Recovery procedures function as expected after failures

**Usability Requirements**

* + User tasks are completed within specified time frames
  + Error rates fall below acceptable thresholds
  + User satisfaction meets or exceeds target metrics
  + Learning curve metrics match or exceeds expectations
  + Accessibility compliance is verified

**Interface Requirements**

* + All specified interfaces function correctly
  + Data exchanges with external systems operate as expected
  + User interfaces display correctly across specified devices
  + Eror handling protocols function properly at interface boundaries
  + Interface performance meets specified targets

**Security Requirements**

* + Authentication mechanisms function as specified
  + Authorization controls properly restrict access
  + Data encryption functions correctly
  + Audit logging captures required information
  + System resists penetration testing attempts

# **Appendices**

## **Assumptions and Dependencies**

This section outlines the assumptions and dependencies that influence the Campus Ride-Sharing Platform with Parking System Integration.

**Assumptions**

* The MMU Single Sign-On (SSO) system will remain operational and accessible.
* Campus Wi-Fi and cellular coverage will provide reliable connectivity across parking zones.
* University-affiliated users (students, faculty, and staff) will have smartphones capable of running the app.
* The carpool reward system will remain viable under university policies.
* Parking occupancy data updates will be accurate and timely.

**Dependencies**

* **MMU IT Security Policies**: Affect user authentication and data management.
* **Campus Parking System**: Must provide real-time availability and enforce priority zones.
* **Notification Services**: Cloud-based push notifications depend on third-party providers.

## **Acronyms and Abbreviations**

|  |  |
| --- | --- |
| **AES** | Advanced Encryption Standard |
| **API** | Application Programming Interface |
| **CSV** | Comma-Separated Values |
| **GIS** | Geographic Information System |
| **GPS** | Global Positioning System |
| **HTML** | HyperText Markup Language |
| **HTTPS** | Hypertext Transfer Protocol Secure |
| **ID** | Identification |
| **IEEE** | Institute of Electrical and Electronics Engineers |
| **JSON** | JavaScript Object Notation |
| **KPIs** | Key Performance Indicators |
| **MMU** | Multimedia University |
| **PDF** | Portable Document Format |
| **REST API** | Representational State Transfer API |
| **SOS** | Safe Operating Stop |
| **SSO** | Single Sign- On |
| **SRC** | Student Representative Council |
| **SRS** | System Requirements Specification |
| **TLS** | Transport Layer Security |
| **UI** | User Interface |
| **XML** | Extensible Markup Language |

## **Glossary**

|  |  |
| --- | --- |
| **API (Application Programming Interface)** | A set of protocols that allows different software components to communicate effectively. |
| **Authentication** | The process of verifying a user's identity using **MMU SSO** credentials. |
| **Carpool Matching Engine** | The module responsible for pairing drivers and passengers based on route compatibility. |
| **Database** | A structured repository for storing and managing user profiles, ride history, and parking occupancy data. |
| **Eco-Points** | Reward points earned by users for ride-sharing participation, redeemable for rewards like priority parking or vouchers. |
| **Emergency Alert (SOS)** | A feature that allows users to notify campus security and selected emergency contacts in critical situations. |
| **GPS (Global Positioning System)** | A technology used for location tracking and ride validation. |
| **HTTPS (Hypertext Transfer Protocol Secure)** | A secure communication protocol ensuring encrypted data transmission. |
| **MMU SSO (Multimedia University Single Sign-On)** | A system used for authenticating users using university-issued credentials. |
| **Notification System** | A module responsible for sending alerts related to ride confirmations, parking availability, rewards, and emergencies. |
| **Passenger** | A registered user who requests a ride through the platform. |
| **Priority Parking** | Reserved parking spots allocated for verified carpools as an incentive for shared commuting. |
| **Push Notifications** | Instant messages sent to users via the platform’s mobile application to provide timely updates. |
| **Ride History** | A record of past rides taken or offered by a user, including details of trip durations and participants. |
| **Ride Offer** | A submission by a driver specifying available seats, origin, destination, and trip timing. |
| **Ride Request** | A submission by a passenger specifying pickup location, destination, and preferred time. |
| **Security Protocols** | Measures implemented to ensure data encryption, user authentication, and privacy compliance. |
| **User Interface (UI)** | The visual elements of the platform that facilitate interaction with ride-matching and parking lookup features. |
| **Verification Criteria** | Standards used to confirm the system meets specified functional, security, and usability requirements. |