A blue and black logo

Description automatically generated, Picture

**CSE 6224**

**SOFTWARE REQUIREMENTS ENG**

**System Requirements Specification (SRS)**

**Title:**

**Campus Ride-Sharing Platform with Parking System Integration**

**TT2L**

**GROUP E** 

**KELVEN YEE KAI WEN 1211111244**

**KOH XUAN LIN 1211109618**

**OW KA SHENG 1211108820**

**SHAZREEN BINTI SHERIDAN 243UC247P3**

**Table of Contents**

[**1.0** **Introduction** 3](#_Toc198495291)

[**1.1** **Purpose** 3](#_Toc198495292)

[**1.2** **Scope** 3](#_Toc198495293)

[**1.3** **Product Overview** 4](#_Toc198495294)

[**1.3.1** **Product Perspective** 4](#_Toc198495295)

[**1.3.2** **Product Functions** 7](#_Toc198495296)

[**1.3.3** **User Characteristics** 10](#_Toc198495297)

[**1.3.4** **Limitations** 12](#_Toc198495298)

[**1.4** **Definition** 14](#_Toc198495299)

[**2.0** **References** 15](#_Toc198495300)

[**3.0** **Requirements** 16](#_Toc198495301)

[**3.1** **Functions** 16](#_Toc198495302)

[**3.2** **Performance Requirements** 17](#_Toc198495303)

[**3.3** **Usability Requirements** 18](#_Toc198495304)

[**3.4** **Interface Requirements** 19](#_Toc198495305)

[**3.5** **Logical Database Requirements** 20](#_Toc198495306)

[**3.6** **Design Constraints** 21](#_Toc198495307)

[**3.7** **Software System Attributes** 22](#_Toc198495308)

[**3.8** **Supporting Information** 23](#_Toc198495309)

[**4.0** **Verification** 24](#_Toc198495310)

[**4.1** **Verification Approach** 24](#_Toc198495311)

[**4.2** **Verification Criteria** 25](#_Toc198495312)

[**5.0** **Appendices** 26](#_Toc198495316)

[**5.1** **Assumptions and Dependencies** 26](#_Toc198495317)

[**5.2** **Acronyms and Abbreviations** 27](#_Toc198495318)

[**5.3** **Glossary** 28](#_Toc198495319)

# **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**

The Campus Ride-Sharing Platform with Parking System shall facilitate mainly the following operations:

1. User registration and login via MMU digital ID verification.
2. Establishment and participation in ride-sharing programs for users going in comparable routes.
3. Real-time availability of parking spaces within MMU campus areas.
4. Notifications regarding ride requests, confirmations, and cancellations.
5. Ride history tracking and user feedback collection.
6. Earn and redeem carpool incentives (e.g., priority parking, vouchers).
7. Ensure safety through emergency alert features and identity verification via MMU SSO authentication.

## **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:**

* FR-1: Allow users to register using their MMU credentials
* FR-2: Enable users to create and manage their profile information
* FR-3: Support user preference settings for ride matching
* FR-4: Provide account deactivation and data management options

**Ride Offering and Requesting:**

* FR-5: Allow users to offer rides by specifying origin, destination, time, and available seats
* FR-6: Enable users to request rides by specifying pickup location, destination, and time
* FR-7: Support recurring ride scheduling for regular commutes
* FR-8: Provide ride modification and cancellation capabilities

**Ride Matching and Coordination**

* FR-9: Match ride requests with available offerings based on route compatibility
* FR-10: Calculate and display estimated arrival times for riders
* FR-11: Facilitate in-app communication between drivers and passengers
* FR-12: Support multi-stop ride coordination for optimal carpooling

**Parking System Integration**

* FR-13: Display real-time parking availability across campus zones
* FR-14: Reserve priority parking spots for verified carpools
* FR-15: Provide navigation to available parking areas
* FR-16: Track historical parking usage patterns

**Safety and Security**

* FR-17: Verify user identity through MMU SSO authentication
* FR-18: Provide SOS emergency alert functionality
* FR-19: Enable ride tracking for designated emergency contacts
* FR-20: Support rider rating and review system

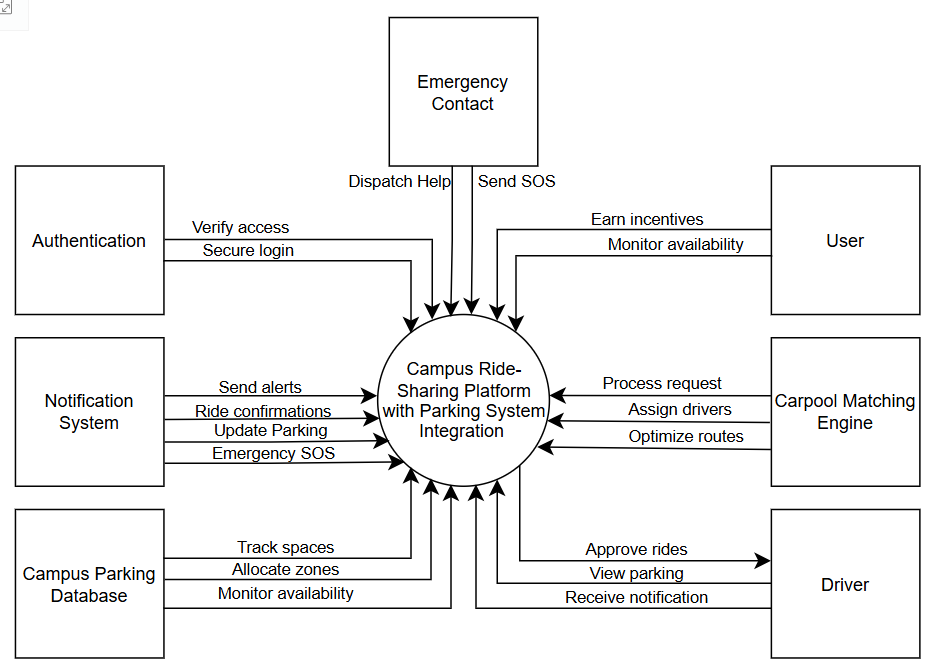
**Incentive Management**

* FR-21: Award eco-points for successful carpooling participation
* FR-22: Provide a leaderboard of top carpoolers
* FR-23: Enable redemption of rewards (parking credits, campus vouchers)
* FR-24: Track carbon emission reduction through ride sharing

**Reporting and Analytics**

* FR-25: Generate personal ride history and statistics
* FR-26: Provide system usage reports for administrators
* FR-27: Calculate environmental impact metrics
* FR-28: Support customizable data export functionality

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



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

The user initially logs in to the system and authenticates their identity using the MMU User Authentication. They can order a ride once successfully logged in, which is handled by the Carpool Matching Engine. When a match is found, the driver is notified of the ride information as well as information on available parking spaces. The system also sends out notifications, like ride confirmations and updates, through the Notification System to keep everyone informed. At the same time, the Campus Parking Database keeps track of parking spaces in real time to help with efficient allocation. Emergency Contact is available to provide quick responses in case of any emergencies to ensure safety throughout the ride.

### **User Characteristics**

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

**Students**

* 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

**Faculty**

* Representing approximately 15% of the user base
* More regular and predictable schedules aligned with class timetables
* Higher expectations for reliability and professionalism
* Often commute from more diverse locations
* Primary motivation: reduced parking stress and environmental considerations

**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 Administrators**

* 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 for system configuration and troubleshooting
* Primary focus: system efficiency, security, and user satisfaction

**All Users**

* 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 incentive 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**

|  |  |
| --- | --- |
| SSO | Single Sign-on authentication for users. |
| KPIs | Key Performance Indicators for tracking ride efficiency. |
| SOS | Emergency notification feature for safety alerts. |
| 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 |
|  |  |

# **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>

# **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.

**Use Case Diagram**

**A diagram of a diagram

AI-generated content may be incorrect.**

**Use Case Specification**

Use Case: Login

Actors: Admin, Driver, Passenger

Preconditions: User must be registered

Postconditions: User gains access to the platform

Basic Flow:

1. Users enter login credentials.
2. The system verifies credentials via MMU SSO authentication.
3. If valid, the system grants access to the dashboard.
4. If invalid, the system displays an error message and prompts retry.

Use Case: Register

Actors: Driver, Passenger

Preconditions: User has valid MMU credentials

Postconditions: User 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

Use Case: Manage 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

Use Case: Offer Ride

Actors: Driver

Preconditions: User 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

Use Case: Request Ride

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

Use Case: Modify Ride

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

Use Case: Chat with Driver

Actors: Driver, 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.

Use Case: Rating and Feedback

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.

Use Case: View Parking

Actors: Driver

Preconditions: User must be authenticated

Postconditions: Parking availability is displayed with real-time updates

Basic Flow:

1. User selects "View Parking Availability" option.
2. System queries about the Campus Parking Database.
3. The system displays available parking spots with locations and occupancy status.
4. User selects a parking location for navigation assistance.

Use Case: View Notification

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.

Use Case: Redeem Incentives

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 Incentives" 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

Use Case: SOS Message

Actors: Driver, Passenger

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 in the app.
2. The system verifies emergency status and location.
3. The system dispatches emergency notification to campus security.
4. System alerts designated emergency contacts with location details.
5. Security personnel respond accordingly.

Use Case: Generate Report

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.

Use Case: Manage System

Actors: Admin

Preconditions: Admin must be authenticated with appropriate privileges

Postconditions: System settings and configurations are modified successfully

Basic Flow:

1. Admin accesses system management interface.
2. Admin modifies settings (user permissions, notifications, feature updates).
3. System validates and applies changes.
4. System stores updated configurations securely.

## **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**

**User Interface Responsiveness**

* 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.

**Ride Matching Performance**

* The system shall complete ride matching operations within 5 seconds of request submission.
* Real-time location updates for active rides shall refresh at intervals of no more than 15 seconds.
* Route calculations shall be completed within 3 seconds of request.

**Parking System Integration**

* 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**

**Concurrent Users**

* 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.

**Transaction Volume**

* The system shall process up to 100 ride requests per minute during peak periods.
* The system shall handle up to 50 ride matches per minute during peak periods.
* The system shall support up to 200 parking availability queries per minute.

**Scalability Requirement**

**User Base Expansion**

* The system shall scale 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.

**Feature Expansion**

* 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**

**Data Storage**

* 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 shall be designed to efficiently handle up to 10,000 active users.

**Network Bandwidth**

* 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**

**Uptime**

* The system shall maintain 99.5% uptime during academic semesters.
* Scheduled maintenance shall occur during off-peak hours (typically 2:00-4:00 AM).
* Maximum allowed unplanned downtime shall not exceed 1 hour per month.

**Fault Tolerance**

* 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 a positive user experience for all target user groups:

**Learnability Requirements**

**Intuitive Interface**

* 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 shall provide an interactive tutorial for first-time users that can be completed in under 3 minutes.

**Help and Documentation**

Context-sensitive help shall be available for all major functions.

* The help 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**

**Task Completion**

* 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.

**Navigation Efficiency**

* 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 shall be prominently displayed for quick access.

**Satisfaction Requirements**

**User Satisfaction Metrics**

* 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 shall show that at least 85% of users would recommend the platform to others.

**Visual Design**

* 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**

**Inclusive Design**

* 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.

**Device Compatibility**

* 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**

**Error Prevention**

* 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.

**Recovery Options**

* 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**

**Student-Specific Feature**

* 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.

**Faculty/Staff Features**

* 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**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **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**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **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**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **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**

|  |  |
| --- | --- |
| **Feature** | Description |
| **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**

|  |  |
| --- | --- |
| **Feature** | Description |
| **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**

|  |  |
| --- | --- |
| **Feature** | Description |
| **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**

|  |  |
| --- | --- |
| **Feature** | Description |
| **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**

|  |  |
| --- | --- |
| **Feature** | Description |
| **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**

|  |  |
| --- | --- |
| **Feature** | Description |
| **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**

# (Mapped to 9.6.15 Logical Database Requirements)

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

# Example:

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

# related to the “Reviewer” entity

## **Design Constraints**

(Mapped to 9.6.16 Design Constraints)

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

external standards, regulations, or technical limitations.

Examples:

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

## **Software System Attributes**

# (Mapped to 9.6.18 Software System Attributes)

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

# performance:

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

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

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

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

# sensitive user data.

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

# facilitate updates.

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

# without additional configuration

## **Supporting Information**

# (Mapped to 9.6.20 Supporting Information)

# Any additional supporting information, including:

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

# or any other elicitation techniques;

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

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

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

# loading or other requirements.

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

# of the requirements.

# Example:

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

# **Verification**

## **Verification Approach**

**(Mapped to 9.6.19 Verification)**

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

**locations.**

**Example:**

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

**verify system performance.**

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

**department.**

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

**each sprint).**

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

## **Verification Criteria**



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

## functional and quality requirements.

## Example:

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

# **Appendices**

## **Assumptions and Dependencies**

**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**

|  |  |
| --- | --- |
| SRS | System Requirements Specification |
| MMU | Multimedia University |
| ID | Identification |
| SSO | Single Sign- On |
| KPIs | Key Performance Indicators |
| SOS | Safe Operating Stop |
| IEEE | Institute of Electrical and Electronics Engineers |

## **Glossary**

Explain the purpose:

Include a glossary if your project involves many domain-specific or technical terms. This section

is especially useful when your system is used by non-technical users, stakeholders, or clients. It

complements Section 1.4 (Definitions), but allows for a broader, more explanatory list of terms.