A logo for a university

AI-generated content may be incorrect.

**CSE 6224 Software Requirements ENG**

Software Requirements Specification (SRS)

Project Title: Campus Ride-Sharing Platform with Parking System Integration

Tutorial Session: TT2L

Group Member:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Student ID | Email | Contact Number | Tutorial Session |
| LAU KAIXUAN | 1221103162 | 1221103162@student.mmu.edu.my | 01164442810 | TT2L |
| ALSAMMAN, LEEN | 1221308098 | 1221308098@student.mmu.edu.my |  | TT2L |
| MONISH A/L MOORTHY | 1211111608 | 1211111608@student.mmu.edu.my | 01154240624‬ | TT2L |
| THONG YUN PENG | 1211107295 | 1211107295@student.mmu.edu.my |  | TT2L |

Table of Contents

[1. Introduction 4](#_Toc198935923)

[1.1 Purpose 4](#_Toc198935924)

[1.2 Scope 5](#_Toc198935925)

[1.3 Product Overview 7](#_Toc198935926)

[1.3.1 Product Perspective 8](#_Toc198935927)

[1.3.2 Product Functions 8](#_Toc198935928)

[1.3.3 User Characteristics 12](#_Toc198935929)

[1.3.4 Limitations 13](#_Toc198935930)

[1.4 Definitions 14](#_Toc198935931)

[2. References 19](#_Toc198935932)

[3.0 Requirements 21](#_Toc198935933)

[3.1 Functions 21](#_Toc198935934)

[3.2 Performance Requirements 64](#_Toc198935935)

[**Response Time** 64](#_Toc198935936)

[**Throughput** 64](#_Toc198935937)

[**Scalability** 64](#_Toc198935938)

[**Availability & Reliability** 64](#_Toc198935939)

[**Concurrency & Capacity** 65](#_Toc198935940)

[**Data Volume** 65](#_Toc198935941)

[**Latency (Inter-Service)** 65](#_Toc198935942)

[**Resource Utilization** 65](#_Toc198935943)

[**Peak Load & Stress Conditions** 65](#_Toc198935944)

[3.3 Usability Requirements 67](#_Toc198935945)

[**Usability Objectives** 67](#_Toc198935946)

[3.4 Interface Requirements 70](#_Toc198935947)

[3.4.1 System Interfaces (External Systems) 70](#_Toc198935948)

[3.4.2 User Interfaces 71](#_Toc198935949)

[3.4.3 Hardware Interfaces 72](#_Toc198935950)

[3.4.4 Software Interfaces 72](#_Toc198935951)

[3.4.5 Communications Interfaces 73](#_Toc198935952)

[3.5 Logical Database Requirements 74](#_Toc198935953)

[3.6 Design Constraint 76](#_Toc198935954)

[3.7 Software System Attributes 79](#_Toc198935955)

[3.8 Supporting Information 81](#_Toc198935956)

[4. Verification Approach & Criteria 85](#_Toc198935957)

[4.1Verification Approach: 85](#_Toc198935958)

[4.2 Verification Criteria: 87](#_Toc198935959)

[5. Appendices 89](#_Toc198935960)

[5.1 Assumptions and Dependencies 89](#_Toc198935961)

[5.2 Acronyms and Abbreviations 92](#_Toc198935962)

# 1. Introduction

## 1.1 Purpose

The purpose of this project is to design and develop a campus ride-sharing and parking app meticulously tailored to tackle the persistent and multifaceted commuting and parking challenges encountered by students, faculty, and staff at the university. As the university’s population has steadily grown over recent years, its parking infrastructure—once sufficient—has struggled to keep pace, resulting in a marked increase in traffic congestion and a chronic shortage of available parking spaces. Students frequently find themselves arriving late to classes or missing critical deadlines, forced to spend inordinate amounts of time circling overcrowded lots in search of elusive parking spots. Faculty and staff are not immune to these difficulties, often beginning their workdays frazzled after battling the same logistical hurdles, which in turn erodes productivity and heightens stress levels across the campus community. These issues have compounded over time, transforming what should be a seamless aspect of campus life into a daily source of frustration and inefficiency.

To address these pressing concerns, the app introduces a suite of innovative, user-centric features designed to streamline the commuting experience. For instance, it will provide real-time updates on parking availability, leveraging data from sensors or user reports to display open spots across campus lots instantly. This functionality empowers users to bypass the time-consuming guesswork of parking searches, directing them straight to viable options and freeing up valuable time in their schedules. Complementing this, the ride-sharing component will serve as a game-changer by intelligently matching users with overlapping travel routes or schedules. By facilitating shared rides, the app reduces the number of individual vehicles converging on campus each day, directly alleviating the strain on parking facilities and easing the flow of traffic through busy university thoroughfares.

Beyond solving immediate logistical woes, the project carries profound implications for sustainability—a cornerstone of the university’s long-term vision. By encouraging ride-sharing and decreasing reliance on single-occupancy vehicles, the app stands to substantially lower greenhouse gas emissions and reduce the campus’s overall carbon footprint. Imagine a scenario where dozens, if not hundreds, of cars are replaced by shared rides each week—this shift not only mitigates air pollution but also aligns seamlessly with the university’s established commitment to environmental stewardship. In doing so, it positions the institution as a forward-thinking leader among peer universities, demonstrating a tangible model for how technology can harmonize convenience with ecological responsibility.

The app’s design goes further by prioritizing an exceptional user experience, ensuring it is both accessible and secure for its diverse audience. Features such as a secure login system will protect user data, while integrated emergency tools—such as a quick-contact button for campus safety services—will enhance peace of mind, particularly for those commuting during early morning or late-night hours. The interface itself will be intuitive and straightforward, crafted with input from students, faculty, and staff to accommodate varying levels of technological proficiency. Whether a first-year student navigating campus for the first time or a seasoned professor juggling a packed schedule, every user will find the app approachable and efficient, breaking down barriers to adoption and maximizing its reach across the community.

This initiative is more than a technical solution—it’s a catalyst for broader cultural and operational change on campus. By fostering a more connected and collaborative commuting network, the app will strengthen the sense of community among its users, encouraging interactions that might not otherwise occur. Picture students from different majors carpooling and exchanging ideas, or staff members building rapport outside the confines of office walls—these small connections can ripple outward, enriching the social fabric of the university. Moreover, the project supports the institution’s strategic objectives of innovation and engagement, showcasing how cutting-edge technology can be harnessed to solve real-world problems while enhancing the daily lives of its stakeholders.

Ultimately, this project aspires to transform the university into a more efficient, sustainable, and cohesive environment. It seeks to reclaim time lost to parking woes, reduce the environmental toll of commuting, and elevate the quality of life for every member of the campus community. Students will arrive at lectures refreshed rather than rushed, faculty will approach their teaching and research with fewer distractions, and staff will benefit from a smoother start to their workday. In the long term, the app’s success could inspire similar initiatives elsewhere, amplifying its impact beyond the university’s borders. This endeavor reflects a commitment not just to solving today’s challenges but to building a foundation for a more resilient and vibrant campus future.

## 1.2 Scope

The project's scope encompasses the comprehensive development of a dual-platform solution which is a mobile application for iOS and Android devices tailored for students and staff, and a corresponding web platform. This web platform will offer a user-facing portal for desktop access to core functionalities and a robust administrative backend for MMU administrators.

Central to the project is the implementation of core ride-sharing functionalities. This includes secure user registration using MMU credentials with potential for advanced digital ID verification and multi-factor authentication, alongside comprehensive profile management. The system will facilitate ride matching through geolocation for pick-up/drop-off points, suggest optimal routes considering real-time traffic, and provide live vehicle tracking with ETAs. Flexible scheduling options for both advance reservations and instant ride-finding, complete with easy cancellation or rescheduling, will be key. Communication will be enabled via in-app chat/call features and push notifications for ride updates and campus alerts. Furthermore, the scope includes automated, dynamic fare estimation, transparent cost-splitting, integration of diverse digital payment methods (including campus billing and e-wallets, with a potential cash toggle), detailed trip/payment histories, and a two-way rating and feedback system, including a suggestion function.

In parallel, the project will deliver integrated parking management functionalities. This involves displaying real-time parking lot occupancy by integrating with campus sensor or gate systems, and allowing users to reserve parking spots, potentially linked with ride-sharing. Automated campus gate entry/exit via RFID or license-plate recognition is a significant component. The system will also support digital parking permits, dynamic parking fees based on demand, and incentives for ride-sharing participants.

Safety and security are paramount, with the scope including in-app tools like an SOS button and trip sharing, alongside robust driver and vehicle verification processes such as profile displays with ratings, support for background screening, and cross-checking vehicles against approved campus lists.

For administrators, the web platform will provide a centralized dashboard for system oversight, detailed reporting and analytics on usage and trends, tools for managing user roles, policies, and pricing, and alerts for system maintenance needs. This also covers the management of digital permits and potential integration points for citation workflows.

Finally, the project is bound by critical non-functional requirements, ensuring high performance, availability, and reliability. A strong emphasis will be placed on usability through intuitive design, accessibility compliance across platforms, robust security measures to protect data, a consistent brand identity, and a responsive interface for all users.

## 1.3 Product Overview

This software product is a comprehensive campus ride-sharing and parking management application designed to enhance the commuting and parking experience for students, staff, and administrators at MMU. The application will provide core ride-sharing functionalities such as user registration and digital ID verification using campus credentials, real-time vehicle tracking, dynamic fare estimation and cost splitting, and various digital payment options. It will also integrate with the campus parking system to offer real-time parking availability, allow users to reserve specific parking spots, and automate gate access using technologies like RFID or license-plate recognition. For user safety and security, the application will include in-app safety tools like SOS buttons and trusted contacts, along with robust driver background checks and vehicle verification. Administrative functions, accessible via an admin dashboard, will allow oversight of ride-matching activity, parking utilization, permit issuance, and system health, with reporting and usage analytics. The application is being developed for both iOS and Android mobile platforms, with a web portal planned for desktop access and administrative tasks.

The application's functionality is interconnected with several external systems and context objects crucial for its operation. Students and staff (riders and drivers) are the end-users who will interact with the system. The application will interface with campus parking space infrastructure, utilizing data from smart parking IoT databases and facilities manager interviews for real-time slot reservations and fee information. For security, it will integrate with the campus vehicle registry to map license plates to users, adhering to campus security policies. User authentication and role management will be handled through a digital campus Single Sign-On (SSO) system, following best practices for trusted login. An in-app wallet will manage ride cost splitting and parking fees, aligning with the Finance Department's Service Level Agreement. Real-time traffic and ETA data will be sourced from campus maps and sensors via GIS layer specifications. The University Safety Office will receive SOS alerts and incident logs, following their Safety Standard Operating Procedures. Furthermore, the system will comply with Road Transport Regulations, specifically Malaysia's e-hailing regulations, for non-commercial carpooling. Competitive heuristic teardowns of benchmark applications like Grab and Kumpool are also being used to inform UX baselines and identify "delighter" features.

### 1.3.1 Product Perspective

The application's functionality is interconnected with several external systems and context objects crucial for its operation. Students and staff (riders and drivers) are the end-users who will interact with the system. The application will interface with campus parking space infrastructure, utilizing data from smart parking IoT databases and facilities manager interviews for real-time slot reservations and fee information. For security, it will integrate with the campus vehicle registry to map license plates to users, adhering to campus security policies. User authentication and role management will be handled through a digital campus Single Sign-On (SSO) system, following best practices for trusted login. An in-app wallet will manage ride cost splitting and parking fees, aligning with the Finance Department's Service Level Agreement. Real-time traffic and ETA data will be sourced from campus maps and sensors via GIS layer specifications. The University Safety Office will receive SOS alerts and incident logs, following their Safety Standard Operating Procedures. Furthermore, the system will comply with Road Transport Regulations, specifically Malaysia's e-hailing regulations, for non-commercial carpooling. Competitive heuristic teardowns of benchmark applications like Grab and Kumpool are also being used to inform UX baselines and identify "delighter" features.

Context Diagram:

A diagram of a company

AI-generated content may be incorrect.

### 1.3.2 Product Functions

**Core Ride-Sharing Functions:**

1. **User Registration and Digital ID verification:**
   * Allow students/faculty/staff to sign up using campus credentials.
   * Verify identity via government-ID/ID card selfie matching in real-time.
   * Support multi-factor authentication for additional security.
2. **User Profile Management:**
   * Enable users to update personal details and contact information.
   * Allow users to view their ride history.
   * Manage notification preferences
3. **Geolocation and Route Optimization:**
   * Use GPS for precise pick-up/drop-off location selection.
   * Provide optimal routes with real-time traffic updates.
4. **Real-time Vehicle Tracking:**
   * Offer live map tracking of the driver's location.
   * Provide estimated time of arrival (ETA).
5. **Fare Estimation and Cost Splitting:**
   * Generate dynamic estimated shared-ride costs.
   * Enable automatic cost splitting among riders.
6. **Payment Processing and Multiple Payment Options:**
   * Process payments via credit/debit cards.
   * Allow campus account billing.
   * Support mobile wallets.
   * Include a cash toggle option (for some users, if it does not complicate the system or raise safety concerns).
7. **Payment History:**
   * Provide detailed records of past payments, including driver, vehicle, route, and cost.
8. **Rating and Feedback System:**
   * Enable driver and rider rating functionality.
   * Allow reporting of incidents or issues.
9. **Ride Scheduling:**
   * Support pre-booking of rides.
   * Allow setting up recurring rides.
10. **Emergency (SOS) Button and Safety Features:**
    * Provide direct access to the University Safety Office in emergencies.
    * Enable trusted contacts functionality for live trip sharing.

**Parking Management Functions:**

1. **Real-time Parking Availability:**
   * Display available parking slots and types on the campus map.
2. **Parking Slot Reservation:**
   * Allow pre-booking of parking spots.
   * Enable reservation of specific parking bays.
3. **Automated Gate Access:**
   * Integrate with RFID or license plate recognition systems for automated entry/exit.
4. **Parking Fee Management:**
   * Facilitate in-app payment for parking fees.
   * Manage recurring parking passes.
5. **Parking Navigation:**
   * Provide in-app guidance to reserved parking spots.
6. **Parking History:**
   * Allow users to view past parking sessions and associated fees.

**Administrative Functions:**

1. **Admin Dashboard:**
   * Provide tools for user and driver management.
   * Display ride and parking analytics.
2. **Reporting and Analytics:**
   * Generate reports on usage trends and peak times.
   * Provide revenue reporting.
3. **Content Management:**
   * Manage in-app content and notifications.
4. **System Configuration:**
   * Allow configuration of fare rates and parking rules.
5. **User Support Management:**
   * Handle user inquiries and dispute resolution.

### 1.3.3 User Characteristics

The target users for the Campus Ride-Sharing Platform with Parking System Integration include University students, faculty/staff and campus administrator. Below are the characteristics which may influence the usability:

1. **Students:** These are a primary user group, heavily involved in providing survey responses and interview feedback during the elicitation phase.

* **Usability Influence:** They are expected to have basic computer and smartphone skills. The application should be intuitive and require no specific technical expertise to navigate and use.

1. **Faculty/Staff:** This group represents secondary users who also contributed through surveys and interviews.

* **Usability Influence:** They are generally expected to have basic to moderate technical familiarity, comfortable with standard applications but still benefiting from a straightforward user interface.

1. **Campus Administrators:** This group provided insights into administrative needs via interviews.

* **Usability Influence:** Administrators are expected to have basic computer skills, with a moderate level of technical familiarity required for using the administrative dashboard and its reporting features. The interface for administrative tasks should be clear and efficient for managing users, rides, and parking.

### 1.3.4 Limitations

The Campus Ride-Sharing Platform with Parking System Integration, while designed to be a comprehensive solution for campus transportation, operates within certain defined boundaries and dependencies that inherently limit its scope and functionality.

Limitations:

1. **Campus-Specific Scope:** The application's operational scope is limited to the MMU campus, as it is designed for students, staff, and administrators within that specific university ecosystem. It is not intended for broader public use beyond the campus.
2. **Non-Commercial Carpooling Focus:** The system is governed by "Road Transport Regulations" pertaining to "non-commercial carpooling." This means it is limited to facilitating non-commercial ride-sharing among campus members and is not designed to operate as a commercial e-hailing service.
3. **Reliance on External Systems:** The application's functionality is highly dependent on integrations with various external campus systems and third-party services. Its performance and availability are thus limited by the reliability, data accuracy, and operational status of these interconnected systems, including:

* Digital Campus Single Sign-On (SSO) for user authentication.
* Smart parking IoT databases for real-time parking data.
* The Campus Vehicle Registry for vehicle verification.
* The In-App Wallet and associated financial systems for payment processing.
* Campus Map & Sensors for traffic and ETA data.
* The University Safety Office for emergency response.

1. **Usability Constraints (Inferred from "Reverse" Requirements):** The system's design is constrained by user expectations regarding modern application usability, implying limitations on certain design choices:

* It is implicitly limited in that it cannot effectively function without GPS-based pick-up/drop-off, as manual entry would significantly frustrate users.
* It is limited in that it must provide payment history and detailed ride information; the absence of these features would severely erode user trust and convenience.

## 1.4 Definitions

To ensure a shared understanding among all stakeholders of the Campus Ride-Sharing Platform with Parking System Integration Parking application, the following key terms and concepts are defined:

* **Application (App):** Refers to the " Campus Ride-Sharing Platform with Parking System Integration," a software product designed for mobile (iOS and Android) and web platforms to facilitate ride-sharing and parking management within the university campus.
* **Campus Admins:** A key user group consisting of university staff responsible for administrative oversight of the application, including user/driver management, analytics, and system configuration.
* **Context Object:** An external entity or system that the " Campus Ride-Sharing Platform with Parking System Integration & Parking App" interacts with or is influenced by.
* **Cost Splitting:** The functionality within the app that automatically calculates and divides ride costs among participating riders.
* **Delighters (Attractive Requirements):** Features that are unexpected or innovative, which significantly boost user satisfaction when present but do not cause dissatisfaction if absent. They exceed basic user expectations. Examples include in-app chat, campus map with parking zones, and a web platform for users.
* **Digital Campus SSO (Single Sign-On):** An authentication system that provides trusted login and delivers user roles using existing MMU credentials.
* **Dissatisfiers (Must-be Requirements):** Essential or basic features that users expect as a minimum standard. Their absence causes dissatisfaction, but their presence does not significantly increase satisfaction beyond a neutral level. Examples include safety features (SOS button, trusted contacts), MMU ID login access, and real-time parking spot information.
* **Driver:** A user of the application who provides vehicle seats for ride-sharing and may require parking privileges on campus.
* **Elicitation Methods:** Techniques used to gather requirements, including Google Form Surveys, Interviews, and Observation.
* **ETA:** Estimated Time of Arrival, a real-time prediction for vehicle arrival, provided through geolocation and tracking features.
* **Functional Requirements:** Specific capabilities that the system must perform, categorized into core ride-sharing, parking system integration, safety/security, administrative, and feedback functions.
* **In-App Wallet:** A system within the application designed to manage and split ride costs and parking fees.
* **Kano Model:** A framework used to categorize user requirements into Dissatisfiers (Must-be), Satisfiers (One-dimensional), Delighters (Attractive), Indifferent, and Reverse based on how their presence or absence affects user satisfaction.
* **License Plate Recognition (LPR):** A technology used for automated gate access by identifying vehicle license plates.
* **Multi-factor Authentication (MFA):** An enhanced security measure requiring users to provide two or more verification factors to gain access.
* **Non-Functional Requirements:** Attributes of the system such as performance, usability, and interface design that define how well the system performs its functions. These are categorized as Must-be and One-dimensional.
* **Real-time Tracking:** The ability to display the live location of a vehicle or driver on a map, often with estimated time of arrival.
* **RFID:** Radio-Frequency Identification, a technology used for automated entry and exit gate control.
* **Rider:** A user of the application who requests or accepts rides.
* **Satisfiers (One-dimensional Requirements):** Features that increase satisfaction proportionally to their quality of implementation, and dissatisfaction proportionally to their poor implementation. Users actively evaluate these based on performance. Examples include fare estimation, live driver tracking, and advanced ride scheduling.
* **SLA (Service Level Agreement):** A contract or agreement defining the level of service expected from a service provider, relevant for the In-App Wallet.
* **SOS Button:** An in-app safety tool that provides direct access to emergency services or trusted contacts.
* **Stakeholders:** Individuals or groups with an interest in the project, including students, faculty/staff, and campus administrators.
* **User Groups:** Categories of individuals who will interact with the application, including Students, Faculty/Staff, and Campus Admins.
* **Vehicle Registry:** An external system or database used for mapping vehicle license plates to users for campus security purposes.
* **UX (User Experience):** The overall experience of a person using a product, which is informed by benchmark applications.
* **Carpool**: Shared ride by multiple users.
* **Ride**: Individual journey posted by a user.
* **MMU ID**: University-issued digital login credentials.

# 2. References

This section lists all sources cited or consulted during the preparation of this Software Requirements Specification (SRS), adhering to APA 7th edition format for consistency and credibility.

* **Project Documentation:**
  + [Group Name/ID, e.g., TT2L\_G03]. (2025). requirements.docx. [Internal Project Document].
  + [Group Name/ID, e.g., TT2L\_G03]. (2025). TT2L\_G03\_RequirementsCategorize.docx. [Internal Project Document].
  + [Group Name/ID, e.g., TT2L\_G03]. (2025). TT2L\_G03\_SRS.docx. [Internal Project Document].
* **Standards and Methodologies:**
  + IEEE. (2018). ISO/IEC/IEEE 29148:2018 Systems and software engineering—Life cycle processes—Requirements engineering. International Organization for Standardization. <https://www.iso.org/standard/72089.html>
  + Kano, N. (1984). Attractive quality and must-be quality. The Journal of the Japanese Society for Quality Control, 14(2), 39-48. (This is a foundational reference for the Kano Model, which was used for requirements categorization).
  + Pohl, K. (2010). Requirements engineering: Fundamentals, principles, and techniques. Springer.
* **Technical References (Implicitly Consulted based on detailed requirements):**
  + Android Developers. (n.d.). Android documentation. Google. <https://developer.android.com/docs>
  + Apple Developer. (n.d.). iOS developer documentation. Apple. <https://developer.apple.com/documentation/>
  + Open Web Application Security Project (OWASP). (n.d.). OWASP Top 10. <https://owasp.org/www-project-top-ten/>
  + Python Software Foundation. (n.d.). PEP 8 -- Style Guide for Python Code. <https://www.python.org/dev/peps/pep-0008/>
  + The Docker Team. (n.d.). Docker documentation. Docker. <https://docs.docker.com/>

# 3.0 Requirements

## Functions

**Student / Rider**

* **User Registration and Digital ID verification**
* **User Profile Management**
* **Geolocation and Route Optimization**
* **Real-time Vehicle Tracking**
* **Fare Estimation and Cost Splitting**
* **Payment Processing and Multiple Payment Options**
* **Payment History**
* **Rating and Feedback System**
* **Ride Scheduling**
* **Emergency (SOS) Button and Safety Features**
* **Parking Management**
* **Parking Navigation**
* **Parking History**

**Driver**

* **User Registration and Digital ID verification**
* **User Profile Management**
* **Geolocation and Route Optimization**
* **Real-time Vehicle Tracking**
* **Fare Estimation and Cost Splitting**
* **Payment Processing and Multiple Payment Options**
* **Payment History**
* **Rating and Feedback System**
* **Ride Scheduling**
* **Emergency (SOS) Button and Safety Features**
* **Parking Management**
* **Parking Navigation**
* **Parking History**

**Campus Admin**

* **Admin Dashboard**
* **Reporting and Analytics**
* **Content Management**
* **System Configuration**
* **User Support Management**
* **Policy and Permission Management**
* **Alerts and Maintenance**

Use Case Diagram:

A diagram of a system

AI-generated content may be incorrect.

**Use Case Specification**

**User Registration and Digital ID Verification**

**Use Case Name:** Register and Verify User Identity

**Actor(s):** Student / Staff Rider, Driver

**Description:** Allows students, faculty, and staff to sign up using their campus credentials and verify their identity through government-ID/ID card selfie matching, supporting multi-factor authentication for enhanced security.

**Preconditions:**

* + User has access to the app.
  + User possesses valid campus credentials.
  + User has a government-issued ID or campus ID card.

**Postconditions:**

* + User account is created and verified.
  + User can log in to the application.

**Basic Flow:**

* + User initiates registration process.
  + System prompts for campus credentials.
  + User enters campus credentials.
  + System verifies campus credentials against Digital Campus SSO.
  + System prompts for government-ID/ID card selfie matching.
  + User uploads ID and selfie.
  + System processes ID verification via External ID Verification Service.
  + System prompts for multi-factor authentication setup.
  + User completes MFA setup.
  + System confirms successful registration and verification.

**Alternative Flows:**

* + **Invalid Credentials:** System denies registration, prompts for re-entry or contact support.
  + **Verification Failure:** System denies registration, prompts for re-attempt or manual review.
  + **MFA Setup Skipped:** User proceeds without MFA, but is notified of security implications.

**Non-Functional Requirements:**

* + **Security:** High security for sensitive ID data (attractive feature).
  + **Performance:** Real-time ID verification (must-be for campus credentials, attractive for ID/selfie).
  + **Usability:** Intuitive registration flow.

**Activity Diagram:**

A diagram of a system

AI-generated content may be incorrect.

**User Profile Management**

**Use Case Name:** Manage User Profile

**Actor(s):** Student / Staff Rider, Driver

**Description:** Allows users to create, view, and update their personal information, contact details, preferences, ride history, and notification settings within the app.

**Preconditions:**

* + User is registered and logged in.

**Postconditions:**

* + User profile information is updated or viewed.

**Basic Flow:**

User navigates to profile section.

System displays current profile details (name, contact info, preferences, ride history, notification settings).

User selects an option to view or edit specific details.

User modifies editable fields (e.g., contact info, preferences).

User saves changes.

System updates the user's profile and confirms.

**Non-Functional Requirements:**

* + **Usability:** Easy navigation and clear options for profile management.
  + **Data Integrity:** Ensures accurate storage and retrieval of user data.

**Activity Diagram:**

A diagram of a system

AI-generated content may be incorrect.

**Geolocation and Route Optimization**

**Use Case Name:** Optimize Ride Route

**Actor(s):** Student / Staff Rider, Driver

**Description:** Uses GPS to enable selection of pick-up/drop-off points and calculates optimal routes based on real-time traffic data.

**Preconditions:**

* + User's device GPS is enabled.
  + User is logged in.

**Postconditions:**

* + Optimal route is displayed with estimated travel time.

**Basic Flow:**

* 1. User enters or selects pick-up and drop-off locations (or system auto-detects current location for pick-up).
  2. System accesses Campus Map & Sensors for real-time traffic data and location information.
  3. System calculates and displays the most optimal route.
  4. System provides estimated time of arrival (ETA).

**Non-Functional Requirements:**

* + **Accuracy:** Precise GPS location and route calculations (one-dimensional).
  + **Performance:** Real-time route optimization.
  + **Reliability:** Consistent access to map and traffic data.

**Activity Diagram:**

A diagram of a system

AI-generated content may be incorrect.

**Real-time Vehicle Tracking**

**Use Case Name:** Track Vehicle Location in Real-time

**Actor(s):** Student / Staff Rider, Driver

**Description:** Displays live map tracking of the driver's approach to the pick-up point (for riders) or the driver's own vehicle location (for drivers), including ETA.

**Preconditions:**

* + A ride is active or imminent.
  + User's device has internet connectivity.

**Postconditions:**

* + Vehicle location and ETA are visible on the map.

**Basic Flow:**

* 1. Ride begins or is accepted.
  2. System continuously updates vehicle's position on a live map using Campus Map & Sensors.
  3. System calculates and displays updated ETA to destination or pick-up point.

**Non-Functional Requirements:**

* + **Real-time:** Updates must be near-instantaneous (one-dimensional).
  + **Accuracy:** High precision in vehicle location.
  + **Reliability:** Stable connection for continuous tracking.

**Activity Diagram:**

A diagram of a vehicle

AI-generated content may be incorrect.

**Fare Estimation and Cost Splitting**

**Use Case Name:** Estimate Fare and Split Cost

**Actor(s):** Student / Staff Rider, Driver

**Description:** Calculates dynamic estimated shared-ride costs and automatically splits payments among riders.

**Preconditions:**

* + Ride details (pick-up, drop-off, number of riders) are confirmed.

**Postconditions:**

* + Estimated fare and individual split costs are displayed.

**Basic Flow:**

* 1. User (Student/Staff Rider) requests a ride with specified details (e.g., shared ride, number of passengers).
  2. System calculates dynamic estimated total fare based on route, distance, and number of riders.
  3. System automatically divides the total fare among participating riders.
  4. System displays estimated total fare and individual split costs to all relevant users.

**Non-Functional Requirements:**

* + **Accuracy:** Precise fare calculations (one-dimensional).
  + **Transparency:** Clear display of cost breakdown.

#### Activity Diagram:

A diagram of a system

AI-generated content may be incorrect.

**Payment Processing and Multiple Payment Options**

**Use Case Name:** Process Payments

**Actor(s):** Student / Staff Rider, Driver

**Description:** Processes payments via credit/debit, allows campus account billing, supports mobile wallets, and includes a cash toggle option.

**Preconditions:**

* + Ride or parking session is completed.
  + Payment amount is determined.

**Postconditions:**

* + Payment is successfully processed.

**Basic Flow:**

* 1. User (Student/Staff Rider) is prompted to pay after service completion.
  2. System displays available payment options (credit/debit, campus account billing, mobile wallets, cash toggle).
  3. User selects a payment method.
  4. **If digital payment:** User confirms payment via In-App Wallet or linked payment method.
  5. System processes payment.
  6. System confirms payment success.

**Alternative Flows:**

* + **Payment Failure:** System notifies user of failure, prompts for alternative payment or re-attempt.
  + **Cash Toggle Selected:** System notifies driver of cash payment, does not process digital payment.
* **Non-Functional Requirements:**
  + **Security:** Secure handling of payment information (must-be).
  + **Usability:** Multiple convenient payment options (attractive).
  + **Reliability:** Robust payment processing.

**Activity Diagram:**

A diagram of a system

AI-generated content may be incorrect.

**Payment History**

**Use Case Name:** View Payment History

**Actor(s):** Student / Staff Rider, Driver

**Description:** Allows users to view detailed records of past payments, including driver, vehicle, route, and cost details.

**Preconditions:**

* + User is logged in.
  + User has completed transactions.

**Postconditions:**

* + Detailed payment history is displayed.

**Basic Flow:**

* 1. User navigates to the "Payment History" section.
  2. System retrieves and displays a list of past transactions.
  3. User can select a specific transaction to view detailed information (driver, vehicle, route, cost).

**Non-Functional Requirements:**

* + **Accuracy:** Correct display of all transaction details.
  + **Usability:** Easy access and clear presentation of history (must-be).

#### Activity Diagram:

A diagram of a payment process

AI-generated content may be incorrect.

**Rating and Feedback System**

**Use Case Name:** Provide Rating and Feedback

**Actor(s):** Student / Staff Rider, Driver

**Description:** Enables users to provide ratings and reviews for drivers after trips and submit suggestions for app improvements.

**Preconditions:**

* + Ride is completed (for ratings/reviews).
  + User has an opinion or suggestion.

**Postconditions:**

* + Rating/review is submitted.
  + Suggestion is submitted.

**Basic Flow:**

* 1. **For Ratings/Reviews:** After a trip, system prompts user to rate the driver.
  2. User provides a rating (e.g., stars) and optional text review.
  3. **For Suggestions:** User navigates to a "Feedback" or "Suggestions" section.
  4. User enters their suggestion for app improvement.
  5. User submits the feedback/suggestion.
  6. System records the feedback/suggestion.

**Non-Functional Requirements:**

* + **Usability:** Easy and quick feedback submission.
  + **Data Collection:** Effective capture of user sentiments.

#### Activity Diagram:

A diagram of a process

AI-generated content may be incorrect.

**Ride Scheduling**

**Use Case Name:** Schedule Ride

**Actor(s):** Student / Staff Rider, Driver

**Description:** Allows advance booking for carpools and provides easy cancellation/reschedule options.

**Preconditions:**

* + User is logged in.

**Postconditions:**

* + Ride is scheduled, cancelled, or rescheduled.

**Basic Flow:**

* 1. User navigates to "Schedule Ride" feature.
  2. User inputs desired pick-up time, date, pick-up/drop-off locations, and other ride details (e.g., carpool).
  3. System displays available options and confirms the booking.
  4. User can later view scheduled rides.
  5. User can select a scheduled ride to cancel or reschedule.
  6. System processes the cancellation/reschedule and confirms.
* **Non-Functional Requirements:**
  + **Usability:** Flexible and straightforward scheduling/management.

#### Activity Diagram:

A diagram of a system

AI-generated content may be incorrect.

**Emergency (SOS) Button and Safety Features**

**Use Case Name:** Activate Emergency / Safety Features

**Actor(s):** Student / Staff Rider, Driver

**Description:** Provides direct access to the University Safety Office in emergencies, offers "Trusted Contacts" trip sharing, speed-alert notifications, and allows setting emergency contact numbers.

**Preconditions:**

* + User is logged in.
  + User has configured emergency contacts (for trip sharing).

**Postconditions:**

* + Emergency alert is sent to University Safety Office.
  + Trip details are shared with trusted contacts.
  + User is notified of speed alerts.

**Basic Flow:**

* 1. **SOS Button:** User presses an in-app SOS button.
  2. System immediately sends an alert with location details to the University Safety Office.
  3. **Trusted Contacts:** Before a trip, user opts to share trip with trusted contacts.
  4. System sends trip details and live tracking link to pre-configured trusted contacts.
  5. **Speed Alerts:** During a trip, if vehicle exceeds speed limit, system triggers speed-alert notification to the user.
  6. **Set Emergency Contacts:** User navigates to safety settings to add/edit emergency contact numbers.
* **Non-Functional Requirements:**
  + **Reliability:** Immediate and accurate emergency alerts (must-be).
  + **Security:** Privacy for trusted contacts (attractive).
  + **Accessibility:** Easily discoverable and usable SOS function.

#### Activity Diagram:

A screenshot of a chat

AI-generated content may be incorrect.

**Parking Management**

**Use Case Name:** Manage Parking

**Actor(s):** Student / Staff Rider, Driver

**Description:** Allows users to view real-time parking availability, reserve specific parking spots, utilize automated entry/exit gate control, and manage digital parking permits.

**Preconditions:**

* + User is logged in.
  + User has parking privileges/permits.

**Postconditions:**

* + Parking information is displayed.
  + Parking spot is reserved.
  + Gate access is granted/managed.
  + Digital permit is managed.

**Basic Flow:**

* 1. User navigates to the parking section.
  2. System displays real-time parking availability (slots, type) using Campus Parking Space IoT DB.
  3. User selects option to reserve a spot.
  4. User selects desired spot and confirms reservation.
  5. System registers the reservation and updates availability.
  6. **For Gate Control:** Upon approach to gate, system uses RFID/license plate recognition to automatically open/close gate.
  7. **For Permits:** User manages their digital parking permits (view, renew, etc.).
* **Non-Functional Requirements:**
  + **Real-time:** Accurate and immediate parking availability updates (one-dimensional).
  + **Integration:** Seamless integration with Campus Parking Space IoT DB and gate systems.
  + **Usability:** Intuitive interface for parking reservation and management.

#### Activity Diagram:

A diagram of a system

AI-generated content may be incorrect.

**Parking Navigation**

**Use Case Name:** Navigate to Parking Spot

**Actor(s):** Student / Staff Rider, Driver

**Description:** Provides in-app guidance to reserved parking spots.

**Preconditions:**

* + User has a reserved parking spot.
  + User's device GPS is enabled.

**Postconditions:**

* + User is guided to the reserved parking spot.

**Basic Flow:**

* 1. User views their reserved parking spot.
  2. User selects "Navigate" option.
  3. System initiates in-app turn-by-turn guidance to the reserved spot using Campus Map & Sensors.

**Non-Functional Requirements:**

* + **Accuracy:** Precise navigation guidance.
  + **Usability:** Clear and easy-to-follow directions.

**Activity Diagram:**

A diagram of a parking spot

AI-generated content may be incorrect.

**Parking History**

**Use Case Name:** View Parking History

**Actor(s):** Student / Staff Rider, Driver

**Description:** Allows users to view past parking sessions and associated fees.

**Preconditions:**

* + User is logged in.
  + User has completed parking sessions.

**Postconditions:**

* + Detailed parking history is displayed.

**Basic Flow:**

* 1. User navigates to the "Parking History" section.
  2. System retrieves and displays a list of past parking sessions, including duration and fees.

**Non-Functional Requirements:**

* + **Accuracy:** Correct display of all parking session details.
  + **Usability:** Easy access and clear presentation of history.

#### Activity Diagram:

A diagram of a parking system

AI-generated content may be incorrect.

**Admin Dashboard**

**Use Case Name:** Oversee System Operations via Admin Dashboard

**Actor(s):** Campus Admin

**Description:** Provides a centralized dashboard to oversee ride-matching activity, monitor parking utilization, oversee permit issuance, and monitor overall system health.

**Preconditions:**

* + Admin is logged in.

**Postconditions:**

* + Admin has a real-time overview of system operations.

**Basic Flow:**

* 1. Admin logs into the admin portal.
  2. System displays the Admin Dashboard with key performance indicators (KPIs) and operational summaries (e.g., active rides, parking occupancy, permit status, system health metrics).
  3. Admin reviews displayed information.
* **Non-Functional Requirements:**
  + **Real-time:** Dashboard data should be up-to-date.
  + **Usability:** Intuitive and comprehensive dashboard design.

#### Activity Diagram:

A diagram of a computer program

AI-generated content may be incorrect.

**Reporting and Analytics**

**Use Case Name:** Generate Reports and Analytics

**Actor(s):** Campus Admin

**Description:** Generates reports on ride-share adoption, parking lot occupancy trends, peak demand times, and revenue.

**Preconditions:**

* + Admin is logged in.

**Postconditions:**

* + Desired reports are generated and accessible.

**Basic Flow:**

* 1. Admin navigates to the "Reporting and Analytics" section.
  2. Admin selects desired report type and parameters (e.g., date range, specific metrics).
  3. System processes data and generates the report.
  4. Admin views or downloads the report.
* **Non-Functional Requirements:**
  + **Performance:** Efficient report generation.
  + **Accuracy:** Reliable data for analytics.
  + **Data Visualization:** Clear and insightful report presentation.

#### Activity Diagram:

A diagram of a system

AI-generated content may be incorrect.

**Content Management**

**Use Case Name:** Manage App Content

**Actor(s):** Campus Admin

**Description:** Allows admins to manage in-app content, including FAQs, announcements, and promotional materials, and control notifications.

**Preconditions:**

* + Admin is logged in.

**Postconditions:**

* + App content or notification settings are updated.

**Basic Flow:**

1. Admin navigates to "Content Management".
2. Admin selects content type (e.g., FAQ, announcement, notification settings).
3. Admin creates, edits, or deletes content.
4. Admin publishes or schedules content/notifications.
5. System updates content/notification settings.

**Non-Functional Requirements:**

* + **Usability:** Easy content creation and publishing.

#### Activity Diagram:

A diagram of a content management system

AI-generated content may be incorrect.

**System Configuration**

**Use Case Name:** Configure System Settings

**Actor(s):** Campus Admin

**Description:** Configures core system settings, including fare rates and parking rules.

**Preconditions:**

* + Admin is logged in.
  + Admin has appropriate permissions.

**Postconditions:**

* + System settings (fare rates, parking rules) are updated.

**Basic Flow:**

* 1. Admin navigates to "System Configuration".
  2. Admin selects settings to modify (e.g., fare rates, parking zone rules).
  3. Admin adjusts parameters.
  4. Admin saves changes.
  5. System applies new configuration.

**Non-Functional Requirements:**

* + **Security:** Only authorized admins can modify settings.
  + **Auditability:** Changes to configuration should be logged.

**Activity Diagram:**

A diagram of a system configuration

AI-generated content may be incorrect.

**User Support Management**

**Use Case Name:** Manage User Support

**Actor(s):** Campus Admin

**Description:** Manages user inquiries and facilitates dispute resolution.

**Preconditions:**

* + Admin is logged in.

**Postconditions:**

* + User inquiries are addressed.
  + Disputes are managed/resolved.

**Basic Flow:**

* 1. Admin navigates to "User Support" section.
  2. System displays a list of open inquiries or disputes.
  3. Admin selects an item to review.
  4. Admin communicates with user, investigates, and takes action to resolve.
  5. Admin updates status of inquiry/dispute.

**Non-Functional Requirements:**

* + **Usability:** Efficient workflow for managing support tickets.
  + **Data Integrity:** Accurate logging of support interactions.

#### Activity Diagram:

A diagram of a system

AI-generated content may be incorrect.

**Policy and Permission Management**

**Use Case Name:** Manage System Policies and Permissions

**Actor(s):** Campus Admin

**Description:** Configures user roles (student, staff, visitor), sets approval workflows for permits, and adjusts pricing rules.

**Preconditions:**

* + Admin is logged in.

**Postconditions:**

* + User roles, permit workflows, or pricing rules are updated.

**Basic Flow:**

* 1. Admin navigates to "Policy and Permission Management".
  2. Admin selects policy or permission type (e.g., user roles, permit approval, pricing rules).
  3. Admin modifies policies/permissions.
  4. Admin saves changes.
  5. System applies new policies/permissions.

**Non-Functional Requirements:**

* + **Security:** Robust access control for policy changes.
  + **Flexibility:** Ability to adapt policies as needed.

#### Activity Diagram:

A diagram of a system

AI-generated content may be incorrect.

**Alerts and Maintenance**

**Use Case Name:** Receive System Alerts and Manage Maintenance

**Actor(s):** Campus Admin

**Description:** Receives notifications for system issues such as low-battery sensors, gate malfunctions, or parking lot emergencies.

**Preconditions:**

* + Admin is logged in.
  + Monitoring systems are active.

**Postconditions:**

* + Admin is alerted to system issues.
  + Maintenance actions can be initiated.

**Basic Flow:**

* 1. System detects a system issue (e.g., sensor low battery, gate malfunction, parking emergency).
  2. System generates and sends an alert to the Admin.
  3. Admin receives and reviews the alert.
  4. Admin initiates appropriate action (e.g., dispatches maintenance).

**Non-Functional Requirements:**

* + **Real-time:** Immediate alert delivery.
  + **Reliability:** Consistent monitoring and alert generation.
  + **Maintainability:** Clear identification of issues to facilitate maintenance.

#### Activity Diagram:

A diagram of a system

AI-generated content may be incorrect.

## Performance Requirements

As an integral part of the Campus Ride-Sharing Platform with Parking System Integration, performance requirements are crucial to ensure a seamless, efficient, and reliable experience for all users—students, faculty, staff, and administrators. These requirements define the system's responsiveness, capacity, and ability to handle varying loads, directly impacting user satisfaction and operational efficiency. The targets set forth aim to guarantee quick interactions for critical functions, stable operation under peak demand, and future scalability to accommodate growth and evolving campus needs.

**Response Time**

* **Target:** ≤2 seconds for any UI-driven operation.
* **Target:** ≤500 ms for internal API calls.
* **Worst-case:** Under peak load, 95th-percentile response times must stay under 5 seconds to avoid user frustration.

**Throughput**

* **Initial launch target:** ≥200 requests/second sustained.
* **Two-year horizon target:** Scale to ≥1,000 requests/second without degradation in response times.
* **Measurement:** Monitor both peak and average throughput during business-hour load tests.

**Scalability**

* **Horizontal scaling target:** Support adding application instances behind a load balancer to maintain performance linearly up to anticipated peaks (e.g., 5,000 concurrent sessions).
* **Predictive scaling target:** Auto-scale when CPU > 60 % or request queue length exceeds threshold.

**Availability & Reliability**

* **Target:** ≥99.9 % ("three nines") per month (≤43 minutes downtime).
* **MTTR (Mean Time To Recovery) Target:** ≤15 minutes.
* **MTBF (Mean Time Between Failures) Target:** ≥30 days.

**Concurrency & Capacity**

* **Target:** Support 5,000 concurrent authenticated sessions.
* **Future Target:** Rising to 10,000 concurrent authenticated sessions within two years.

**Data Volume**

* **Trip records:** Handle storage and retrieval of up to 1 million trips per month.
* **Parking updates:** Process real-time parking-lot sensor feeds at 1 update/second per lot.

**Latency (Inter-Service)**

* **Target:** ≤100 ms per RPC (Remote Procedure Call).
* **Target:** ≤50 ms per database query under normal load.

**Resource Utilization**

* **CPU Usage:** Keep average CPU utilization per node below 70 % during business-hour peaks.
* **Memory Usage:** Average RAM usage < 75 % of capacity.
* **Memory Leaks:** Must not exceed 1 % of heap over 24 hours.
* **Disk I/O & Network:**
  + Ensure average disk write latency < 10 ms.
  + Network throughput per node ≥500 Mbps

**Peak Load & Stress Conditions**

* **Load-Testing Goals:**
  + Simulate **2$\times$ expected peak** for 30 minutes with no SLA breaches.
  + **Ramp tests:** Increase load by 10 % every 5 minutes until 200 % of target throughput.
* **Error Budget:** Allow no more than **0.1 %** of requests to fail during sustained peak.

## Usability Requirements

Usability objectives for the Campus Ride-Sharing Platform with Parking System Integration aim to ensure that the system is intuitive, efficient, and satisfying for all users. These objectives are aligned with the user needs identified in the project's requirements document, with quantifiable targets to measure their success.

**Usability Objectives**

**1. Ease of Use**

**Objective:** The platform should be straightforward and simple to navigate, requiring minimal effort from users.

* **Intuitive Interface:** The application shall have a clean and straightforward interface, allowing users to navigate effortlessly without prior training.
  + **Quantifiable Target:** 95% of first-time users shall successfully complete core tasks (e.g., booking a ride, checking parking availability) without accessing help documentation or requiring assistance during usability testing.
* **Consistent Design:** The application shall maintain uniformity in design elements across all screens to reduce the learning curve and enhance user familiarity.
  + **Quantifiable Target:** Post-deployment user surveys shall show an average rating of 4.5 out of 5 (on a 5-point Likert scale) regarding design consistency.

**2. Learnability**

**Objective:** New users should be able to quickly understand and effectively use the application's features.

* **Quick Onboarding:** New users should be able to register and start using the app within a few minutes, with clear guidance provided throughout the process.
  + **Quantifiable Target:** The average time for a new user to complete registration and successfully perform their first core action (e.g., search for a ride or check parking availability) shall be ≤3 minutes.
* **Logical Navigation Structure:** The application's navigation shall be organized in a logical manner, allowing users to remember how to access different features easily.
  + **Quantifiable Target:** During usability testing, returning users shall achieve a feature findability rate of 90% within 15 seconds for common functions.
* **Visual Cues:** The application shall use icons and visual indicators consistently to help users recall functions and actions.
  + **Quantifiable Target:** 90% of test participants shall correctly identify the function of key icons without additional labels during usability testing.

**3. Efficiency**

**Objective:** Users should be able to accomplish their goals quickly and with minimal effort.

* **Minimal Steps for Core Tasks:** Users should be able to perform primary actions, such as booking a ride or checking parking availability, in as few steps as possible.
  + **Quantifiable Target:** Booking a ride (from initiating search to confirmation) shall require ≤5 steps. Checking parking availability shall require ≤3 steps.
* **Fast Load Times:** The application should load quickly, with minimal delays, to enhance user satisfaction.
  + **Quantifiable Target:** Initial application load time shall be ≤3 seconds on standard mobile network conditions (e.g., 4G/Wi-Fi). Screen transition times shall be ≤1 second.
* **Real-Time Updates:** The application shall provide users with real-time information on ride statuses and parking availability to facilitate timely decisions.
  + **Quantifiable Target:** The displayed data for real-time updates (e.g., driver location on map, parking slot occupancy) shall have a latency of ≤2 seconds from the actual event.

**4. User Satisfaction**

**Objective:** Users should have a positive experience with the application and feel their needs are met.

* **Overall Satisfaction:**
  + **Quantifiable Target:** The application shall achieve a System Usability Scale (SUS) score of ≥75 in post-launch user surveys.
  + **Quantifiable Target:** The application shall achieve a Net Promoter Score (NPS) of ≥30 within six months of launch.
* **Effective Feedback Mechanisms:** The system shall collect rider and driver feedback after every trip to identify issues and drivers of satisfaction. It shall also allow users to give suggestions if there is anything that can make improvements.
  + **Quantifiable Target:** At least 70% of completed rides shall receive a rating or review from the user.
  + **Quantifiable Target:** A minimum of 5 significant user suggestions, identified through the suggestion function, shall be incorporated into the application per major release cycle (e.g., quarterly).

## Interface Requirements

The Campus Ride-Sharing Platform with Parking System Integration requires various interfaces to interact with external systems, users, hardware, and other software components to fulfill its functional and non-functional requirements.

### 3.4.1 System Interfaces (External Systems)

The platform will integrate with several external university and third-party systems to function effectively:

* **Campus Authentication System (SSO/LDAP):** For user registration and digital ID verification, allowing students/faculty/staff to sign up using their campus credentials.
* **External ID Verification Service:** For real-time government-ID/ID card selfie matching during user identity verification.
* **Mapping and Traffic Data Providers:** To calculate optimal routes based on real-time traffic data and enable geolocation for pick-up/drop-off points.
* **Campus Parking Sensors / IoT Database:** To integrate with campus sensors or gate systems to display live occupancy levels of each lot/garage.
* **Campus Gate Control Systems:** For automated entry/exit gate control using RFID or license-plate recognition at campus gates.
* **Payment Gateways:** To process payments via credit/debit cards.
* **Campus Account Billing System:** To allow campus account billing for payment processing.
* **Mobile Wallet Providers:** To support mobile wallets for payment processing.
* **University Safety Office Communication System:** To provide direct access to the University Safety Office via an SOS/emergency button.
* **Third-Party Background Check Service:** To enforce regular background screening for drivers.
* **Campus Citation Management and Enforcement Workflows:** To integrate with for digital parking permit issuance and validation, and enforcement.

### 3.4.2 User Interfaces

The platform will offer intuitive and consistent user interfaces across multiple devices:

* **Mobile App Interface:** Available for iOS and Android platforms. This interface will be able to run without bugs or display issues.
* **Web Portal Interface:** A web portal for desktop access and administrative tasks.
* **Layout and Interaction Elements:**
  + **Intuitive Interface:** The application will have a clean and straightforward interface, allowing users to navigate effortlessly without prior training.
  + **Persistent Menu:** A simple tab bar (e.g., Rides / Parking / Profile) or a hamburger menu will allow users to jump between main sections in one tap.
  + **Navigation Controls:** A clear "Back" control will always be shown when drilling into sub-screens to prevent users from feeling lost.
  + **Minimalist Layouts:** Only core functions will be displayed on each screen, with secondary options hidden under “More” or in settings. Generous white space will be used to group related elements and guide the eye.
  + **Touch-Friendly Controls:** All tappable buttons and icons will have adequate hit areas (at least 44x44 px) to prevent mis-taps on mobile. Standard components (native buttons, toggles, dropdowns) will be leveraged for instant user recognition.
  + **Immediate Feedback:** Loading indicators (spinners or skeleton screens) will be shown whenever data is loading to reassure users. Action confirmations (toasts or banners for success messages like "Ride booked!") and clear error alerts will be displayed near relevant fields.
  + **Accessibility:** Text alternatives will be provided for icons and images for screen reader compatibility. Text will meet a 4.5:1 contrast ratio, and users will be able to adjust font size within the app.
  + **Responsive Behavior:** Layouts will adapt at key breakpoints (phone, tablet, desktop) without hiding core features. Content will reflow gracefully in both portrait and landscape orientation.
  + **Consistent Branding:** A shared component library (colors, typography, icon set) will ensure every screen feels like part of the same system. The university’s logo and color palette will be incorporated in header or tab bar elements for instant recognition.

### 3.4.3 Hardware Interfaces

The system will interact with specific hardware components for parking and user verification:

* **Campus Parking Sensors:** For integration to display live occupancy levels of parking lots/garages.
* **RFID Readers:** Used at campus gates for automated entry/exit gate control based on reservation/permit status.
* **License Plate Recognition Systems:** Used at campus gates for automated entry/exit gate control based on reservation/permit status.
* **Gate Barrier Lifts:** Controlled by the system for automated entry/exit at campus gates.
* **User Device Camera:** Used for government-ID/ID card selfie matching during user registration and digital ID verification.
* **GPS Modules:** Utilized by user devices and potentially driver vehicles for geolocation and real-time vehicle tracking.

### 3.4.4 Software Interfaces

The platform will interact with various software components and APIs:

* **Digital Campus SSO API:** For integrating with the university's authentication system during user registration.
* **External ID Verification Service API:** For programmatic identity verification.
* **Mapping and Geocoding APIs:** For route optimization, real-time traffic data, and location services.
* **Campus Parking Space IoT Database/API:** For retrieving real-time parking occupancy levels from campus sensors.
* **Payment Gateway APIs:** For processing credit/debit payments, mobile wallet transactions, and campus account billing.
* **Push Notification Service APIs (e.g., FCM, APNs):** For sending push notifications for ride status updates and campus alerts.
* **Real-time Communication APIs:** For in-app chat/call features between riders and drivers.
* **Third-Party Background Check Service API:** For driver background screenings.
* **Campus Citation Management System API:** For digital parking permit enforcement and citation management workflows.
* **Internal API Calls:** The system itself will involve internal API calls between its various microservices or modules.

### 3.4.5 Communications Interfaces

The platform will employ specific protocols and formats for secure and efficient data exchange:

* **HTTPS:** For secure communication between client applications (mobile apps, web portal) and the server [example provided in prompt].
* **Real-time Communication Protocols:** Such as WebSockets or MQTT, for live map tracking, real-time parking availability updates, and real-time parking-lot sensor feeds, as well as in-app chat.
* **Push Notification Protocols:** Specific protocols used by services like Google Firebase Cloud Messaging (FCM) and Apple Push Notification service (APNs) for sending alerts and updates.
* **RPC (Remote Procedure Call) Protocols:** For inter-service communication within the platform, with a target latency of ≤100 ms per RPC.
* **Database Query Protocols:** For interactions with the database, with a target latency of ≤50 ms per database query under normal load.
* **Network Requirements:** The system requires network throughput per node of ≥500 Mbps.

## 3.5 Logical Database Requirements

The **Campus Ride-Sharing Platform with Parking System Integration** relies on a structured set of data entities to manage its core functionalities, including user management, ride services, and parking operations.

At the heart of the system is the **User** entity, representing individuals who interact with the platform, differentiated by roles such as Rider, Driver, or Admin. Each User is uniquely identified and holds attributes like name, email, campus ID, and authentication details. Drivers, as a specific type of User, can register one or more **Vehicles**, each having a unique license plate and specifying details like make, model, and capacity.

The ride-sharing functionality centers around the **Ride** entity, which captures the details of each journey. A Ride is initiated by a **User** (the Rider) and, once assigned, is performed by another **User** (the Driver) using one of their registered **Vehicles**. Rides track key information such as pick-up/drop-off locations, timestamps, status (e.g., requested, in progress, completed, cancelled), and estimated/actual fare. Each completed Ride is linked to a **Payment** entity, recording the amount, method, and transaction date. Users, specifically Riders, can provide a **Rating** for the Driver associated with a completed Ride, offering feedback on their experience.

The parking system integrates with **ParkingLot** entities, representing physical parking areas on campus, each with a total capacity and real-time occupancy. Within each ParkingLot, individual **ParkingSpace** entities are defined, indicating their availability status (available, occupied, reserved) and potentially linked to physical sensors. Users can create a **ParkingReservation** for a specific ParkingSpace for a defined period, and similar to rides, these reservations are tied to a **Payment**. Additionally, Users can hold a **DigitalPermit**, which tracks their authorized parking privileges, permit type, and validity.

Finally, the system leverages a **Notification** entity to send alerts and messages to specific Users, detailing updates on ride statuses, parking events, or general campus announcements. These entities are interconnected through various relationships: Users can request many Rides and make many Payments or Parking Reservations. A Ride can be performed by one Driver and results in one Payment. ParkingLots contain many ParkingSpaces, which can then be reserved by Users. These relationships ensure data integrity and enable comprehensive tracking of all platform activities.

**ER Diagram:**

A diagram of a network

AI-generated content may be incorrect.

## Design Constraint

This section details the various restrictions and limitations that influence the design and development of the Campus Ride-Sharing Platform with Parking System Integration. These constraints stem from external standards, regulatory requirements, and technical capabilities, all of which are critical for ensuring the platform's compliance, performance, and usability.

Restrictions and Limitations

**1. External Standards and Regulations**

* **University Branding Guidelines:**
  + The user interface must incorporate the university's logo and color palette in header or tab bar elements for instant recognition. This implies adherence to specific visual and identity guidelines.
  + A unified style guide (colors, typography, icon set) must be used to ensure every screen feels like part of the same system, aligning with campus identity.
* **Accessibility Standards:**
  + The user interface must provide labels for icons and images to ensure screen readers can announce them, complying with basic accessibility principles.
  + Text must meet a 4.5:1 contrast ratio, and users must be able to adjust font size within the app, aligning with visual accessibility guidelines.
* **Identity Verification Regulations:**
  + The digital ID verification process (government-ID/ID card selfie matching) must comply with relevant data privacy regulations and identity verification standards to ensure legal and secure handling of sensitive user information.
* **Driver Safety Regulations:**
  + The platform must enforce regular background screenings for drivers, implying compliance with transportation safety regulations and background check legal requirements.
* **Data Privacy Regulations:**
  + Handling of user data, especially sensitive information from ID verification, payment details, and location data, must comply with applicable data protection and privacy regulations (e.g., GDPR, local university policies, etc.).

**2. Technical Limitations**

* **Performance Targets:**
  + **Response Time:** All UI-driven operations must have a response time of ≤2 seconds, with internal API calls ≤500 ms. Under peak load, 95th-percentile response times must remain under 5 seconds.
  + **Throughput:** The system must sustain ≥200 requests/second initially, scaling to ≥1,000 requests/second within two years without degradation.
  + **Availability:** The system must maintain ≥99.9% availability per month, with a Mean Time To Recovery (MTTR) of ≤15 minutes and Mean Time Between Failures (MTBF) of ≥30 days.
  + **Concurrency:** The system must support 5,000 concurrent authenticated sessions, with a future target of 10,000 within two years.
  + **Latency (Inter-Service):** Inter-service communication (RPCs) must have ≤100 ms latency, and database queries ≤50 ms under normal load.
* **Resource Utilization:**
  + Average CPU utilization per node must remain below 70% during business-hour peaks.
  + Average RAM usage must stay below 75% of capacity.
  + Memory leaks must not exceed 1% of heap over 24 hours.
  + Average disk write latency must be < 10 ms, and network throughput per node ≥500 Mbps.
* **Hardware Integration Constraints:**
  + The system's design must accommodate integration with existing campus hardware, specifically RFID readers, license plate recognition systems, and gate barrier lifts for automated parking control, which might impose specific protocol or interface limitations.
  + Reliance on GPS modules and device cameras on user mobile phones limits data accuracy and availability based on device capabilities and user permissions.
* **Software Platform Constraints:**
  + The mobile applications must be developed for and compatible with both iOS and Android platforms, implying adherence to their respective ecosystem guidelines and technical specifications.
  + The web platform must utilize a responsive design that adapts effectively across various screen sizes (phone, tablet, desktop) and handles both portrait and landscape orientations gracefully.
* **Data Volume Management:**
  + The system must be designed to handle the storage and efficient retrieval of up to 1 million trip records per month and process real-time parking-lot sensor feeds at 1 update/second per lot, indicating significant data management requirements.

## 3.7 Software System Attributes

This section specifies the essential attributes that define the quality and performance of the **Campus Ride-Sharing Platform with Parking System Integration**. These attributes are critical for ensuring the system's resilience, security, usability, and long-term viability, addressing both functional and non-functional aspects of the software product.

Required Software Product Attributes

**1. Reliability**

Reliability refers to the system's ability to perform its required functions under stated conditions for a specified period, including its capacity for recovery.

* **Mean Time Between Failures (MTBF):** The system shall have an MTBF of ≥30 days.
* **Crash Recovery:** The system should be able to recover from a crash and resume full operation within **1 minute**.
* **Error Handling:** The system shall implement robust error handling mechanisms to gracefully manage unexpected inputs, system failures, and external service outages, minimizing impact on user experience.

**2. Availability**

Availability specifies the proportion of time the system is operational and accessible to users.

* **Target Availability:** The system shall be available ≥99.9% ("three nines") per month, which translates to a maximum of approximately 43 minutes of downtime per month.
* **Mean Time To Recovery (MTTR):** The system shall have an MTTR of ≤15 minutes.
* **Operational Hours:** The system should strive for this availability target continuously, with particular emphasis during working hours (Monday through Friday, 8 AM to 6 PM), which are critical for campus operations.

**3. Security**

Security encompasses the measures taken to protect the system and its data from unauthorized access, use, disclosure, disruption, modification, or destruction.

* **Data Encryption:** All sensitive data (e.g., Personally Identifiable Information (PII), payment information, user credentials) must be encrypted both at rest (using AES-256 or stronger) and in transit (using TLS 1.2+).
* **Access Control:** The system shall implement Role-Based Access Control (RBAC) to restrict features and data access based on defined user roles (Rider, Driver, Admin).
* **Authentication:** The system shall support Multi-Factor Authentication (MFA) for all user roles to enhance login security.
* **Input Validation:** All user inputs must be rigorously validated server-side to prevent common vulnerabilities such as SQL injection and Cross-Site Scripting (XSS) attacks.
* **Session Management:** Secure session management techniques (e.g., short-lived tokens, regular re-authentication) shall be implemented to protect user sessions.
* **Audit Trails:** The system shall maintain immutable audit logs for all critical system actions and data access events to support security monitoring and forensics.
* **Vulnerability Testing:** Regular penetration testing (quarterly) and vulnerability scans (monthly) shall be conducted to identify and remediate security weaknesses proactively.

**4. Maintainability**

Maintainability describes the ease with which the software can be modified, updated, debugged, and enhanced after deployment.

* **Modularity:** The system design shall be highly modular, with clearly defined interfaces between components to enable independent development, testing, and deployment.
* **Code Readability:** The codebase must adhere to established coding standards (e.g., PEP 8 for Python, or equivalent for other languages used) with comprehensive inline comments and external documentation.
* **Testability:** Components and modules should be designed for easy automated testing, including unit, integration, and end-to-end tests.
* **Extensibility:** The architecture shall allow for easy addition of new features or integration with new services without requiring major rework or impacting existing functionalities.
* **Deployment Simplicity:** Automated build and deployment pipelines (CI/CD) shall be established to enable frequent, low-risk releases and updates.
* **Documentation:** Comprehensive system documentation (including architecture diagrams, API specifications, and deployment guides) must be maintained and kept up-to-date.

**5. Portability**

Portability refers to the software's ability to function consistently across different operating systems, hardware environments, or cloud platforms with minimal modification.

* **Platform Independence:** The backend system should be designed to run seamlessly on common operating systems (e.g., Linux, Windows Server) and major cloud environments (e.g., AWS, Azure) without platform-specific dependencies.
* **Containerization:** Utilize containerization technologies (e.g., Docker, Kubernetes) to package the application and its dependencies, ensuring consistent environments across development, testing, and production stages.
* **Database Portability:** The database schema should be designed to be compatible with multiple relational database systems (e.g., PostgreSQL, MySQL) to facilitate potential migration or deployment flexibility.

## 3.8 Supporting Information

This section provides additional supporting information relevant to the Campus Ride-Sharing Platform with Parking System Integration. While these items offer valuable context and background, they are generally not considered formal functional or non-functional requirements but rather supplemental details to aid the understanding and implementation of the SRS.

Additional Supporting Information

**Sample Input/Output Formats, Elicitation Techniques & Results:**

* **Sample Output Formats:**
  + For Payment History, the system will show details such as the driver's name, vehicle type, vehicle plate number, payment time and date, payment method, and the location. While specific file formats for data export (e.g., CSV, JSON) are not explicitly defined as requirements in the provided documents, the user interface is expected to display this information in a clear, structured list or tabular format. This description of output content is **part of the functional requirements.**
* **Elicitation Techniques & Results:** The requirements for this platform were primarily gathered using the following elicitation methods:
  + **Google Form Survey:** Structured feedback was collected through paired questions designed to assess user reactions to the presence and absence of various features. For example, 18 out of 22 survey respondents strongly agreed or agreed on the importance of safety features, and 19 out of 22 deemed real-time parking spot information critical.
  + **Interviews:** In-depth discussions were conducted with stakeholders to uncover detailed needs and pain points. Interviewees emphasized the necessity of SOS buttons and live tracking and highlighted parking frustrations.
  + **Observation:** Real-time monitoring of commuting and parking behaviors was performed to validate reported issues.
  + The detailed results, such as the specific percentages of agreement or user quotes from these elicitation techniques, are presented within the "Categorization of Requirements" document to justify the Kano Model categorization of features (Must-be, One-dimensional, Attractive, Indifferent, Reverse). These elicitation techniques and their summarized results are **supporting background information** that helped shape the requirements but are not formal requirements themselves.
* **Cost Analysis Studies:** The provided documents do not include descriptions or results of any cost analysis studies.

**Supporting or Background Information:**

* **Project Vision and Purpose:** The **Campus Ride-Sharing Platform with Parking System Integration** is designed to modernize campus transportation by offering a seamless, efficient, and safe solution for ride-sharing and parking management. Its primary goal is to enhance overall campus mobility, reduce traffic congestion, and improve user experience for students, faculty, and staff.
* **User-Centric Design:** The system's design is heavily influenced by user feedback, as evidenced by the Kano Model categorization. This approach ensures that core functionalities meet basic expectations (Must-be features like safety tools, digital payments, and real-time parking info), while others offer proportional satisfaction based on performance (One-dimensional features like fare estimation and live tracking), and some provide delight as unexpected bonuses (Attractive features like in-app chat and dynamic pricing).
* **Stakeholder Needs:** The platform caters to the diverse needs of its user base, including riders, drivers, and campus administrators, ensuring comprehensive management and oversight.

This background information is **essential supporting documentation** to provide context for the requirements listed in the SRS.

**c) A Description of the Problems to be Solved by the Software:**

The Campus Ride-Sharing Platform with Parking System Integration is developed to address several prevalent challenges faced within the university environment:

* **Campus Mobility Inefficiency:** Students and staff often experience difficulties with existing transportation methods, leading to delays and inconvenience. The platform aims to provide an efficient, on-demand, and scheduled ride-sharing solution.
* **Parking Scarcity and Frustration:** Finding available parking spaces on campus is a significant source of frustration for users. The system resolves this by providing real-time parking availability, enabling parking reservations, and offering automated gate control.
* **Safety and Security Concerns:** Users, particularly those traveling late or alone, face safety concerns. The software addresses this through in-app safety tools like an SOS button, trusted contacts trip sharing, speed alerts, driver background checks, and vehicle verification. Users explicitly emphasized the necessity of these features to feel safe.
* **Complex and Inconvenient Payment Processes:** Manual or fragmented payment methods for transportation and parking can be cumbersome. The platform streamlines payments by integrating various digital options, including credit/debit, campus account billing, and mobile wallets, and provides transparent payment history. Users expect seamless payment options.
* **Lack of Real-time Information:** Users often lack immediate updates on ride status, driver location, or parking availability, leading to uncertainty. The system provides real-time vehicle tracking, ETAs, and live parking occupancy updates.

This description of problems to be solved forms the **foundational context for the functional requirements** of the software.

**d) Special Packaging Instructions for Code and Media:**

* **Containerization:** The software's components will be packaged using **containerization technologies (e.g., Docker)**. This is a crucial instruction to ensure consistent execution environments across development, testing, and production stages. This approach contributes to the software's portability and ease of deployment.
* **Automated Deployment:** The system will leverage **automated build and deployment pipelines (CI/CD)** for releasing code and updates. This ensures efficient and low-risk deployments, facilitating frequent updates and maintenance.
* **Security of Media/Code:** While not a specific packaging instruction, all code repositories and deployment media will be subject to the overarching security requirements of the system, including **encryption at rest and strict access controls** to protect against unauthorized access or tampering.

These packaging and deployment instructions are **considered part of the non-functional requirements**, specifically falling under Portability and Maintainability, as they dictate how the software will be delivered and managed. No explicit export restrictions for the code or media are specified in the provided documents.

# Verification Approach & Criteria

This section outlines the strategies and benchmarks used to verify that the Campus Ride sharing platform with Parking System Integration meets its intended requirements. Each verification phase is designed to validate system behavior, performance, usability, and compliance under real-world and stress conditions. The goal is to ensure the platform is robust, secure, user friendly, and aligned with MMU’s operational and regulatory standards.

## 4.1Verification Approach:

A multi-layered verification strategy is adopted to ensure that each module of the system functions correctly and aligns with user expectations and institutional policies. The verification phases include:

**Functional Testing**

This phase validates whether all individual components of the system perform their intended tasks as specified in the requirements document.

Core features such as ride booking, digital ID verification, parking reservation, payment processing, and admin dashboard functions will undergo rigorous black-box and integration testing.

Each feature will be tested across multiple user roles (Student, Staff, Admin) using test scripts based on actual use case scenarios derived from survey and interview feedback.

**Security Testing**

Comprehensive penetration testing and vulnerability scanning will be conducted to assess the system’s defences against common threats such as SQL injection, cross-site scripting (XSS), and unauthorized access attempts.

Particular focus will be placed on verifying the security of the login system (SSO), payment data handling, ID document uploads, and role-based access control.

Data encryption (both in-transit via HTTPS and at-rest in the database) will be tested to ensure compliance with Malaysia’s Personal Data Protection Act (PDPA) and university IT security policies.

**Performance and Load Testing**

Simulated load scenarios will be created to evaluate the system's ability to maintain responsiveness and stability under stress, especially during peak hours such as class changeovers.

Key performance metrics including system response time, server CPU and memory usage, and database transaction throughput will be measured.

Load balancing and auto-scaling capabilities will be tested to verify the system can dynamically handle high concurrency (up to 5,000 users) without service disruption.

**User Acceptance Testing (UAT)**

A selected group of real users including students, faculty, and administration staff will participate in UAT sessions to assess how well the system meets their needs and expectations.

Participants will perform a series of basic tasks ( booking a ride, reserving a parking spot, navigating the dashboard) while being observed for ease of use, efficiency, and satisfaction.

Post-test feedback will be gathered via questionnaires, one-on-one interviews, and usability scoring tools such as the System Usability Scale (SUS) and Net Promoter Score (NPS).

## 4.2 Verification Criteria:

The following criteria define the success benchmarks that the system must meet to be considered verified and ready for deployment:

|  |  |
| --- | --- |
| **Category** | **Verification Metric** |
| **Performance** | The application must respond to user inputs within 2 seconds and maintain 99.9% uptime monthly (≤ 43 min downtime). |
| **Functionality** | All functional requirements (ride matching, parking integration, ID verification, payments, alerts) must execute correctly without logical or runtime errors. |
| **Security** | Sensitive user data must be protected by encryption and secure authentication. No critical vulnerabilities should be found in security testing. |
| **Compliance** | The system must comply with PDPA guidelines and MMU's internal IT security, SSO, and data handling policies. |
| **Usability** | At least 80% of UAT participants should rate the app as user-friendly and efficient using a standardized usability questionnaire (e.g., SUS). |
| **Accessibility** | The interface must meet WCAG 2.1 Level AA accessibility guidelines, ensuring usability for users with visual, auditory, or motor impairments. |
| **Scalability** | The system should seamlessly auto-scale its resources (e.g., spin up new servers) when resource thresholds like 60% CPU usage are reached. |
| **Maintainability** | Admin dashboards must display accurate real-time system metrics and retain logs, reports, and audit trails for at least 30 days to support ongoing system monitoring and issue resolution. |

# Appendices

This section outlines the key assumptions made during the requirements definition and the dependencies that may impact the software development process or the system's operation. These items are not considered formal requirements themselves but are critical contextual factors that must be understood and managed.

## 5.1 Assumptions and Dependencies

**Assumptions:**

* **Stable Internet Connectivity:** It is assumed that users will have reliable internet access on their mobile devices and that the campus network infrastructure will provide stable connectivity for the web platform and backend systems.
* **Device Capability:** User mobile devices (smartphones/tablets) are assumed to have functional GPS capabilities and cameras necessary for features like geolocation, real-time vehicle tracking, and digital ID verification.
* **University Infrastructure Support:** It is assumed that the university's IT department will provide the necessary infrastructure, network access, and support for deploying and hosting the backend systems.
* **Third-Party Service Stability:** The reliability and continued operation of integrated third-party services (e.g., external ID verification, mapping services, payment gateways) are assumed. Any changes or outages from these services could impact system functionality.
* **Sufficient Resources:** It is assumed that adequate budget, human resources (development, QA, operations staff), and time will be allocated throughout the project lifecycle for development, testing, deployment, and ongoing maintenance.
* **Compliance with University Policies:** It is assumed that the system's design and operation will comply with all relevant university policies, including IT, data privacy, and transportation regulations.
* **User Adoption:** It is assumed that the campus community (students, faculty, staff) will adopt the new ride-sharing and parking system, leveraging its features for daily campus mobility.

**Dependencies:**

* **University Authentication System (LDAP/SSO):** The system's user registration and authentication heavily depend on integration with the university's existing campus credentials system. The availability and reliability of this system are critical.
* **External ID Verification Service:** Real-time digital ID verification via government-ID/ID card selfie matching depends on a third-party external ID verification service.
* **Mapping and Real-time Traffic Data Providers:** Geolocation, route optimization, and real-time vehicle tracking functionalities depend on access to and accuracy of external mapping services and real-time traffic data providers.
* **Campus Parking Sensors and Gate Systems:** Real-time parking availability and automated entry/exit gate control are dependent on successful integration with existing or new campus parking sensors and gate systems (including RFID or license-plate recognition hardware).
* **Payment Gateways and University Billing System:** Payment processing for credit/debit cards, mobile wallets, and campus account billing depends on integration with third-party payment gateways and the university's internal billing system.
* **University Safety Office Communication System:** The emergency (SOS) button feature depends on a predefined communication channel or API to directly alert the University Safety Office.
* **Third-Party Driver Background Check Service:** Driver safety features, specifically regular background screenings, depend on integration with a third-party background check service.
* **Campus Citation Management and Enforcement System:** Digital parking permit validation and enforcement workflows depend on successful integration with the university's existing citation management and enforcement systems.
* **Mobile Operating System Updates:** Continued functionality of the mobile applications depends on ongoing compatibility with updates to iOS and Android operating systems.
* **Cloud Service Providers:** The system's scalability, availability, and resource utilization targets depend on the capabilities and reliability of the chosen cloud service provider (e.g., AWS, Azure) and its underlying infrastructure.

## 5.2 Acronyms and Abbreviations

* **API:** Application Programming Interface
* **APNs:** Apple Push Notification service
* **AES-256:** Advanced Encryption Standard with 256-bit key
* **AWS:** Amazon Web Services
* **CI/CD:** Continuous Integration/Continuous Deployment
* **CPU:** Central Processing Unit
* **DB:** Database
* **ERD:** Entity-Relationship Diagram
* **ETA:** Estimated Time of Arrival
* **FCM:** Firebase Cloud Messaging
* **FK:** Foreign Key
* **GPS:** Global Positioning System
* **HTTP:** Hypertext Transfer Protocol
* **HTTPS:** Hypertext Transfer Protocol Secure
* **IoT:** Internet of Things
* **LDAP:** Lightweight Directory Access Protocol
* **LPR:** License Plate Recognition
* **Mbps:** Megabits per second
* **MFA:** Multi-Factor Authentication
* **MQTT:** Message Queuing Telemetry Transport
* **MTBF:** Mean Time Between Failures
* **MTTR:** Mean Time To Recovery
* **OS:** Operating System
* **PII:** Personally Identifiable Information
* **PK:** Primary Key
* **QA:** Quality Assurance
* **RAM:** Random Access Memory
* **RBAC:** Role-Based Access Control
* **RFID:** Radio-Frequency Identification
* **RPC:** Remote Procedure Call
* **SLA:** Service Level Agreement
* **SOS:** Save Our Souls (Emergency button context)
* **SRS:** Software Requirements Specification
* **SSO:** Single Sign-On
* **TLS 1.2+:** Transport Layer Security version 1.2 and above
* **UI:** User Interface
* **XSS:** Cross-Site Scripting