Trimester March/April, 2025

CSE6224 SOFTWARE REQUIREMENTS ENGINEERING

Project Part 1

Topic: Campus Ride-Sharing Platform with Parking System Integration

Software Requirements Specification

Name	Student ID	Course
Chee Rui	1211112287	Bachelor of Computer Science
Teh Li Wei	1211109581	Bachelor of Computer Science
Sow Chien Yee	1211210800	Bachelor of Computer Science
Lai Zi Xuan	1211109451	Bachelor of Computer Science

Table of Contents

1 Introduction

1.1 Problem Statement

The Multimedia University (MMU) Cyberjaya campus often faces issues with limited parking availability and the lack of coordinated transportation options for students or staff. Parking spots are hard to find due to absence of a real-time monitoring system or poor carpool coordination. Additionally there is no centralized platform that enables trusted ride-sharing among campus members while ensuring the security and legitimacy of users through digital ID verification. As a result, campus members waste time on searching for parking, contribute to environmental pollution and experience inefficient travel around campus.

1.2 Vision

To create a secure, user-friendly, and efficient campus ride-sharing platform integrated with a real-time parking management system. It aims to provide a secure, user-friendly, efficient and useful system that enables students or staff to coordinate carpools, promotes sustainable transportation and helps in reducing parking demands. By integrating with the university's digital ID verification system and real-time parking status checking, we aim to build a trusted, eco-friendly, and smarter mobility experience across campus.

1.3 Scope

The system will:

- Allow university members to verify their identity using their digital student or staff ID during registration.
- Enable users to offer and request rides based on time, destination, and availability.
- Automatically match riders and drivers using customizable filters.
- Display real-time parking availability across campus.
- Indicate which parking spaces are currently occupied by verified users.
- Obtain users' locations (with permission) for more accurate matching and parking coordination.
- Allow users to claim and unclaim parking spaces to maintain parking legitimacy.

The system will not:

- Provide rides to individuals outside the university community.
- Handle financial transactions or facilitate payments for rides. (have to decide? possible?) *(I mean its possible, but like student carpooling around campus should be fixed, its not a big place after all)*

1.4 Purpose

The purpose of the Campus Ride-Sharing and Parking Management System is to address the transportation and parking challenges at MMU Cyberjaya by providing a secure, integrated system for campus ride-sharing and real-time parking management. It aims to facilitate trusted carpool coordination through integration with student ID , reduce time spent searching for parking by showing available parking spots, lower environmental impact, and promote more efficient and sustainable campus mobility for students and staff.

On the other hand, this document defines the software requirements for the Campus Ride-Sharing and Parking Management System at Multimedia University Cyberjaya. It's purpose is to provide a clear and detailed description of the system's functionalities, constraints, and goals of the system. It serves as a reference for the development team, project stakeholders, and university administration to ensure the system is designed to meet user needs and institutional goals. This document also provides the foundation for future system design, development, and validation. It aims to ensure all parties share a common understanding of the system's expected behaviour, features, limitations and evaluation.

1.5 Goals

- Reduce campus parking congestion through coordinated ride-sharing.
- Provide secure and exclusive access to university members by using digital ID authentication.
- Offer real-time parking availability data to optimize parking usage.
- Encourage environmentally friendly commuting habits.

1.6 References

Below are the references used in our project:

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

2. Institute of Electrical and Electronics Engineers. (1998). *IEEE recommended practice for software requirements specifications (IEEE Std 830-1998)*. IEEE. Retrieved from https://www.cse.msu.edu/~cse435/Handouts/SRSExample-webapp.doc

3. University of Texas at Dallas. (n.d.). *Software Requirements Specification*. Retrieved from https://www.utdallas.edu/~chung/RE/Presentations07S/Team_1_Doc/Documents/SRS4.0.doc

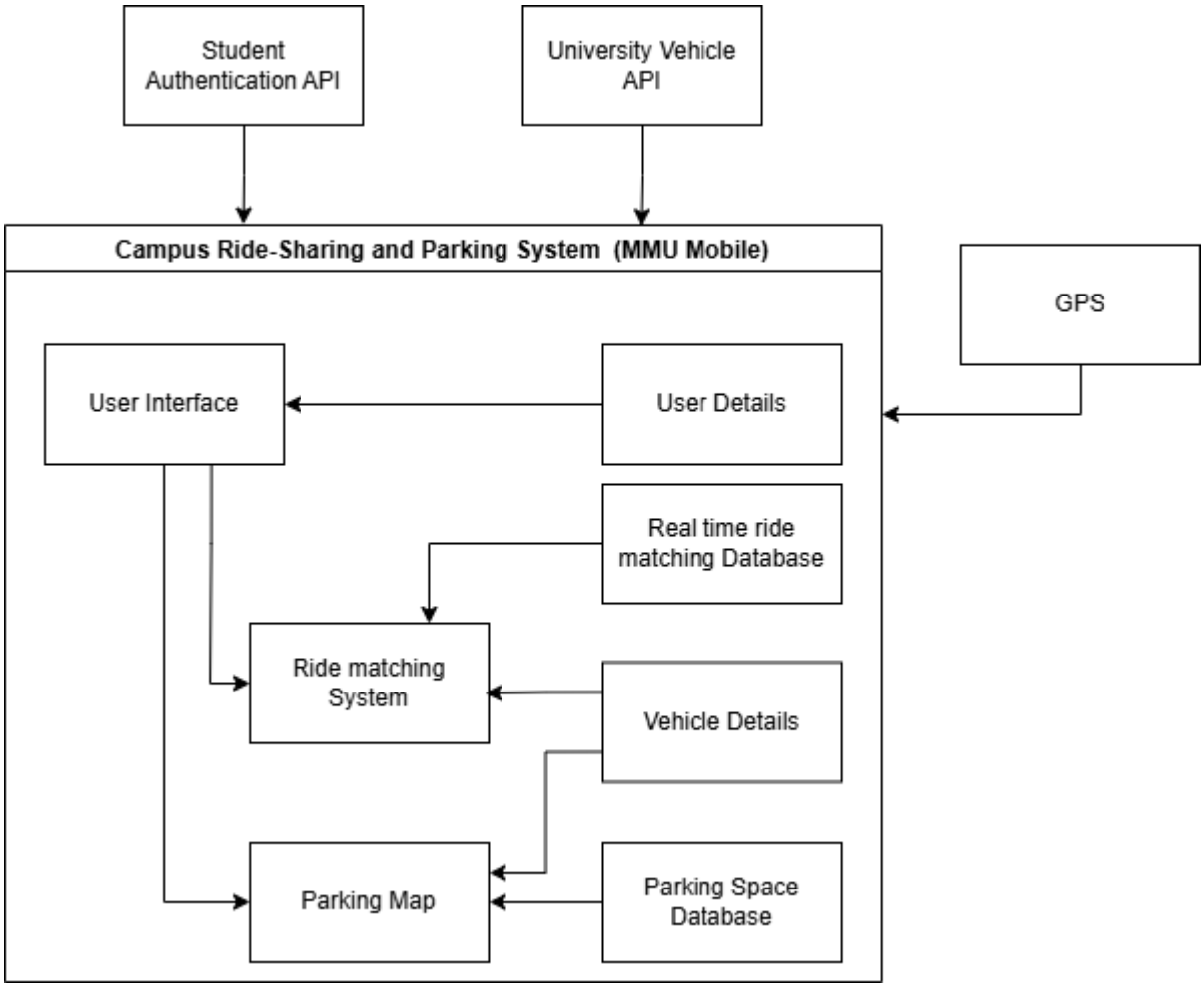
2 Product Overview

The Campus Ride-Sharing and Parking System is a module integrated within the existing MMU Mobile campus app. It provides additional functions to the app by coordinating secure, safe ride-sharing and providing real time parking for end users.

This integration supports reducing parking problems and promotes shared mobility while complementing other digital campus services. The system communicates with the campus app through APIs, sharing user data with the campus parking system, student authentication services with student id, and real-time mapping tools. It synchronizes with backend services for ride-matching and parking space management. These interfaces ensure seamless functionality within the larger campus infrastructure.

Accessible through mobile platforms, it offers a centralized solution to reduce parking congestion, lower carbon emissions, and enhance overall campus mobility.

2.1 Product Perspective



The block diagram above illustrates the campus ride-sharing and parking application. It extends the existing MMU Mobile App by integrating user data for secure login and accessing APIs for verified vehicle details and owner information. The app also interfaces with GPS to track user location and provide navigation to the destination. Integrated user and vehicle data are used to display information without creating additional databases. Data flows between the system, APIs, and databases to deliver a seamless user experience. These connections delivers efficient coordination between parking and carpooling services.

2.2 Product Functions

User Capabilities

- Log in using Student ID and password
- View and interact with the campus map (zoom, navigate)
- Check real-time parking space availability and associated vehicle plate numbers
- Report illegitimate parking
- Claim a parking space
- Release a parking space after using
- Book a ride to another faculty with available student/staff drivers

Admin Capabilities

- Perform all standard user functions
- Log in using Admin ID and password
- Access and manage reported parking cases
- View detailed information of car owners and related parking incidents

2.3 User Characteristics

User Group	Description	Relevant Characteristics Affecting Usability
MMU Students	Foundation and undergraduate students who will use the system for parking and carpooling.	- Moderate to high digital literacy - Familiar with MMU app ecosystem - Prefer fast, intuitive interfaces - Prefer Shortcuts
MMU Staff	Academic and administrative staff using the system occasionally.	- Varying levels of technical skill - Require simple and clear navigation - May Prefer larger text - Not overly complicated interactions
Administrators	Security guards responsible for handling parking issues.	- Require simple and straightforward navigation - Clean layout focused on their specific tasks

The targeted user groups include both end users and administrators. Since the users are primarily adults and teenagers, mobile device access is not an issue. However, differences in age and technical familiarity mean that a non-intuitive interface may cause difficulties for some individuals. Therefore, simplicity and clarity should be prioritized in the UI/UX design.

2.4 Limitations

The following table outlines the limitations that may impact the design, development, and deployment of the Campus Ride-Sharing Platform with Parking System Integration:

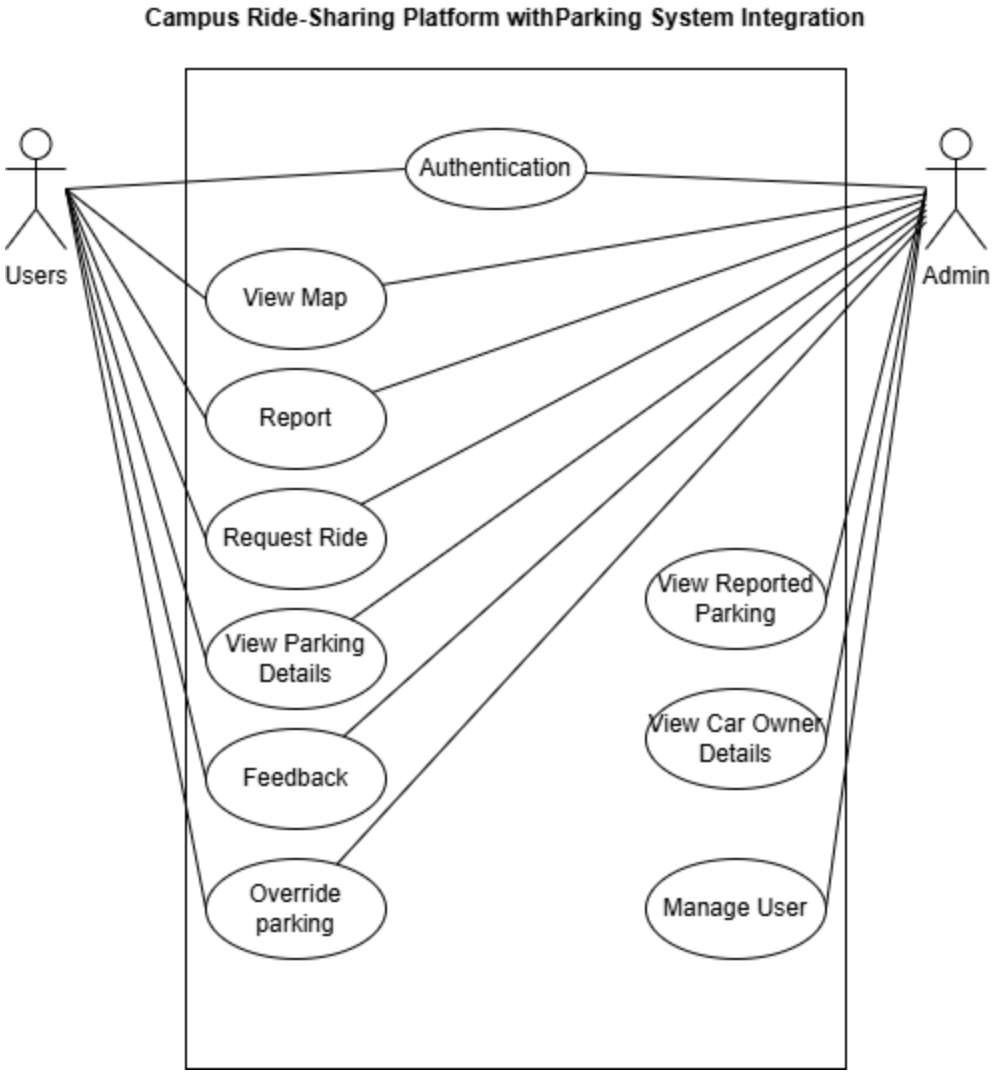
Limitation Category	**Description**
Regulatory Requirements and Policies	Only registered university students and staff can use the platform. User data must comply with the university's privacy policies and external data protection regulations (e.g., PDPA).
Hardware Limitations	Real-time parking information depends on the university's parking management infrastructure, which may have latency, limited coverage, or data inaccuracies.
Interfaces to Other Applications	Integration with campus digital ID verification and parking management systems is required. External system limitations (e.g., API restrictions) may affect platform functionality.
Parallel Operation	The platform must operate alongside existing manual parking procedures during the transition phase.
Audit Functions	User login and ride-sharing activities must be logged, but real-time auditing is not mandatory.
Control Functions	The platform provides parking recommendations but does not directly control external devices like parking gates.
Higher-Order Language Requirements	No restrictions on programming languages; system interfaces should follow common standards (e.g., REST APIs).
Signal Handshake Protocols	Standard HTTPS communication and OAuth authentication are sufficient; no special signal protocols are required.

Quality Requirements	The platform must ensure at least 99% uptime during semesters. Parking data refresh intervals should not exceed 2 minutes.
Criticality of the Application	The system is important for convenience but is not safety-critical; failures should degrade gracefully without endangering users.
Safety and Security Considerations	Authentication must be secure. Personal data must be encrypted both during transmission and at rest.
Physical/Mental Considerations	The user interface should include accessibility features (e.g., support simple navigation) to assist users with disabilities.
Limitations Sourced from Other Systems	Accuracy and timeliness of parking availability and ID verification depend on external university systems, which may introduce occasional delays or errors.

3 Requirements

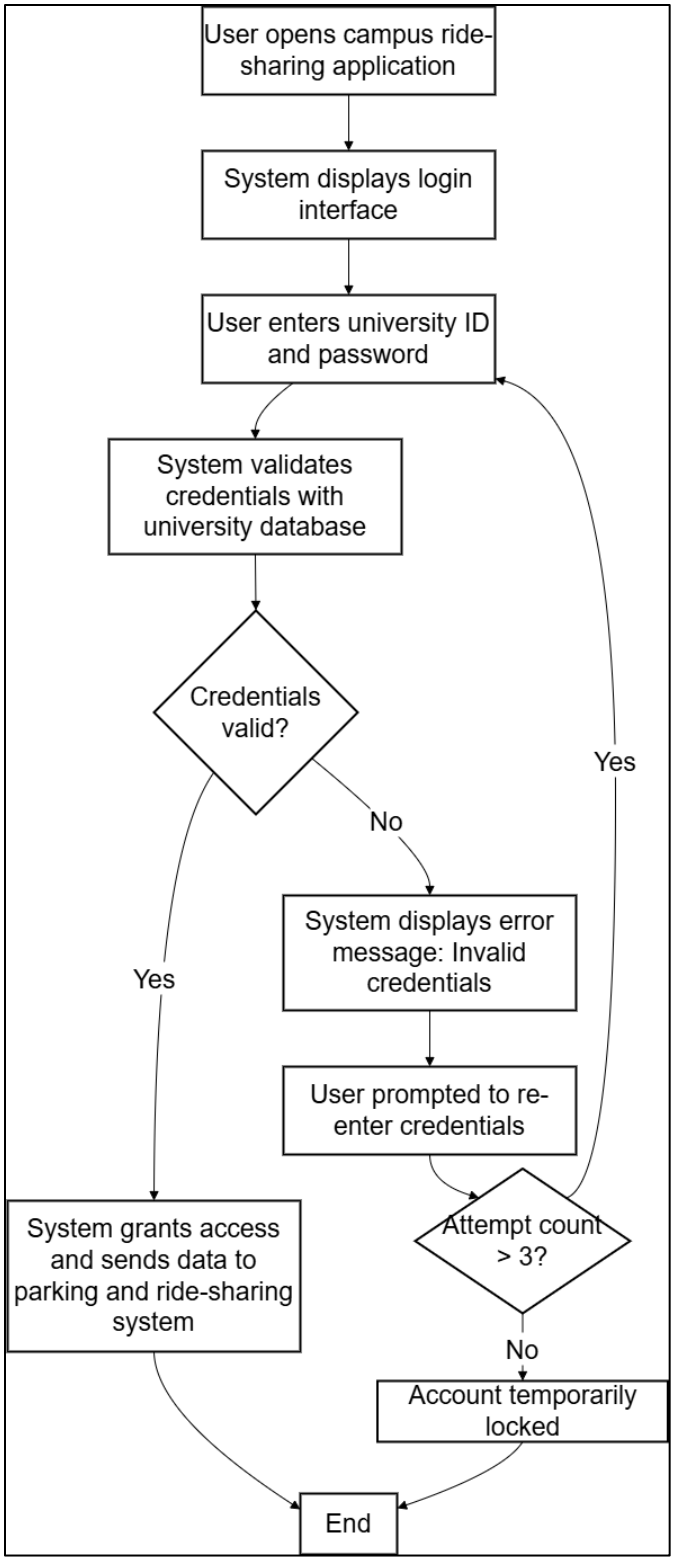
This section outlines the system requirements for the MMU Campus Ride-Sharing Platform with Parking System. It defines both functional and non-functional requirements, user expectations, and system constraints. These requirements guide the development process and ensure the system meets the needs of students, staff, and administrators.

3.1 Functions



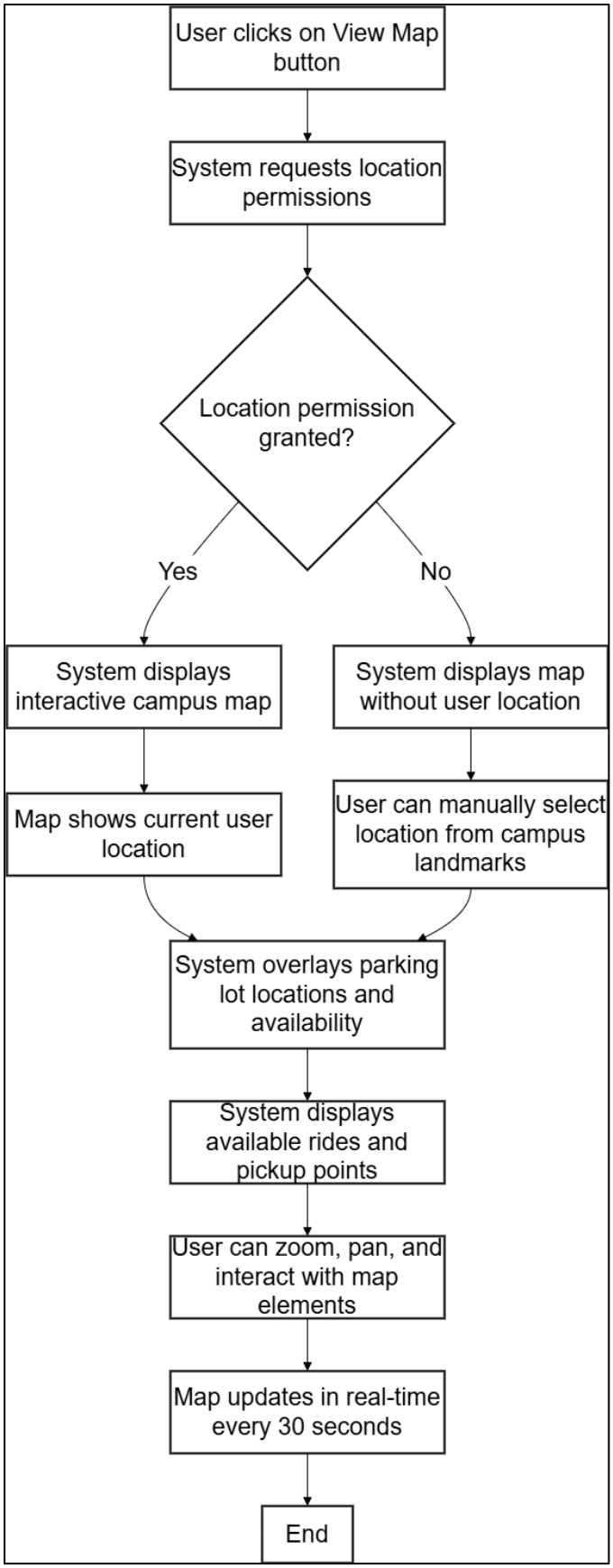
Detailed Use Case Descriptions

Use Case ID	UC001	**Version**	1.0
Feature	User Authentication		
Purpose	To allow users to securely log into the system using university credentials.		
Actor	Users, Admin		
Trigger	User attempts to access the platform		
Precondition	User has valid university account credentials		
Scenario Name	User Auth		
Main Flow	1. User opens the MMU mobile application 2. System displays login interface 3. User enters university ID and password 4. System validates credentials with university database 5. System grants access and redirects to main dashboard 6. User clicks to enter the parking and ride-sharing system. 7. User Data is sent to the parking and ride-sharing system		
Alternate Flow - Invalid Credentials	1. System displays error message "Invalid credentials" 2. User is prompted to re-enter credentials 3. After 3 failed attempts, account is temporarily locked		
Rules	• Only verified university students and staff can access the system • Session expires after 24 hours of inactivity		
Author	Software Requirements Engineering Team		



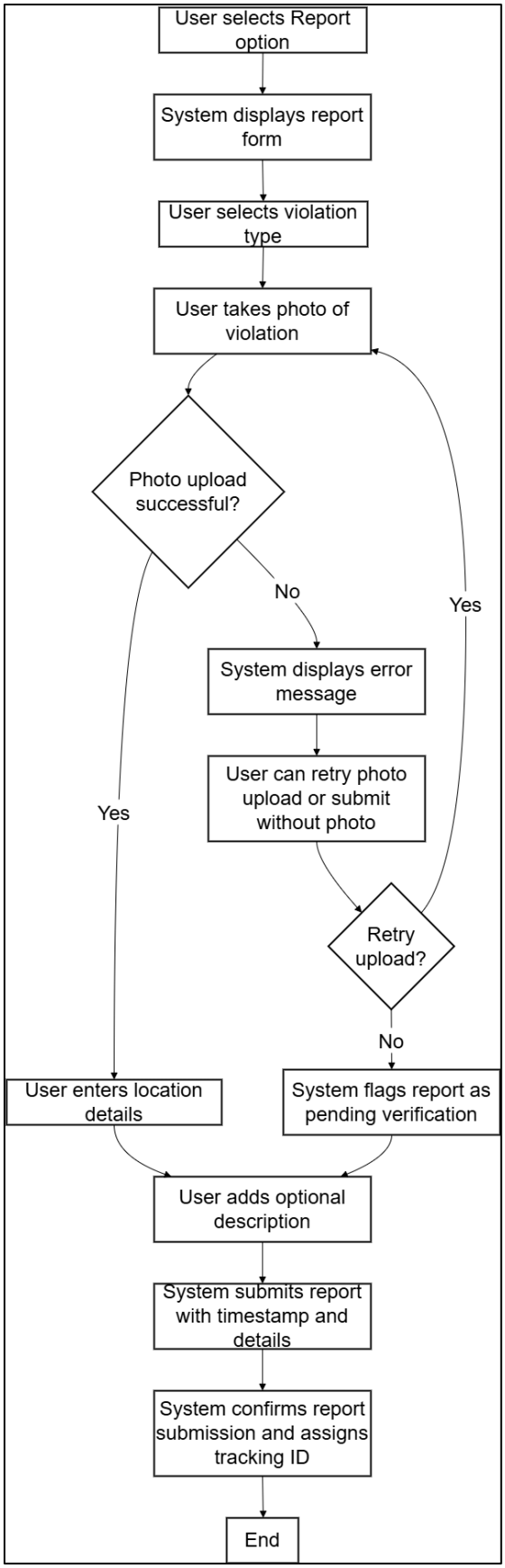
This diagram shows the login process for the MMU Parking and Ride-Sharing System. Users enter their university credentials and password, which are checked by the existing MMU system. If valid, access is granted; if not, an error is shown and the account may be locked after three failed attempts. After logging in, user data is passed to the Parking and Ride-Sharing System for display and use.

Use Case ID	UC002	**Version**	1.0
Feature	View Map		
Purpose	To display real-time campus map with parking and ride information		
Actor	Users, Admin		
Trigger	User selects "View Map" option		
Precondition	User is authenticated and has location permissions enabled		
Scenario Name	Display Campus Map		
Main Flow	1. User clicks on "View Map" button 2. System requests location permissions (if not granted) 3. System displays interactive campus map 4. Map shows current user location 5. System overlays parking lot locations and availability 6. System displays available rides and pickup points 7. User can zoom, pan, and interact with map elements		
Alternate Flow - Location Permission Denied	1. System displays map without user location 2. User can manually select location from campus landmarks		
Rules	• Map updates in real-time every 30 seconds • Location data is only used for functionality, not stored permanently		
Author	Software Requirements Engineering Team		



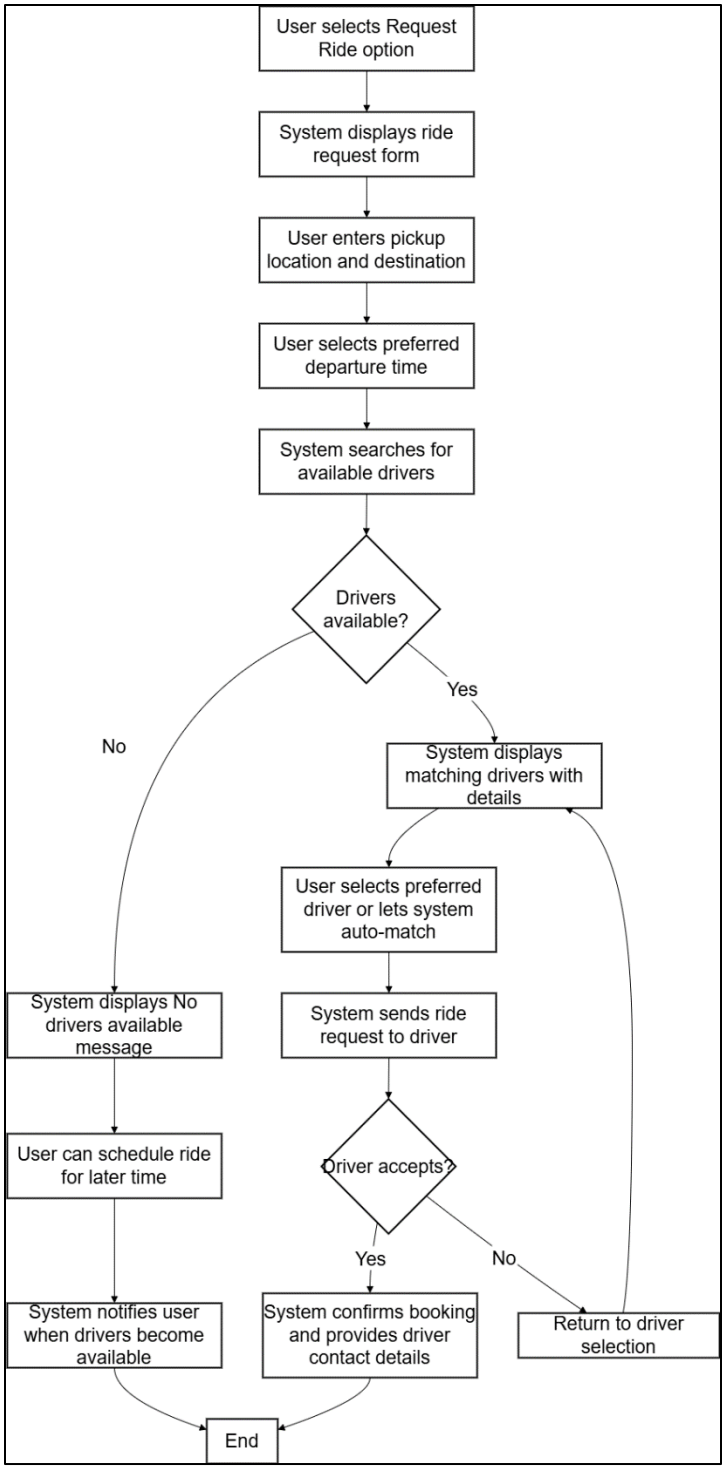
This diagram shows how after the user accesses the campus map, parking spaces and ride information are shown. If location permission is given, the user's location is shown on the map. If not, the user can still view the map and choose a location manually.

Use Case ID	UC003	**Version**	1.0
Feature	Report		
Purpose	To allow users to report parking violations with photo evidence		
Actor	User, Admin		
Trigger	User witnesses parking violation		
Precondition	User is authenticated and has camera permissions		
Scenario Name	Submit Violate Parking Report		
Main Flow	1. User selects "Report" option 2. System displays report form 3. User selects violation type (illegal parking, blocking access, etc.) 4. User takes photo of violation 5. User enters location details 6. User adds optional description 7. System submits report with timestamp and details 8. System confirms report submission and assigns report ID		
Alternate Flow - Photo Upload Failed	1. System displays error message 2. User can retry photo upload or submit without photo 3. System flags report as "pending verification"		
Rules	• Reports must include location information • Photo evidence is strongly recommended but not mandatory • Reports are reviewed within 24 hours		
Author	Software Requirements Engineering Team		



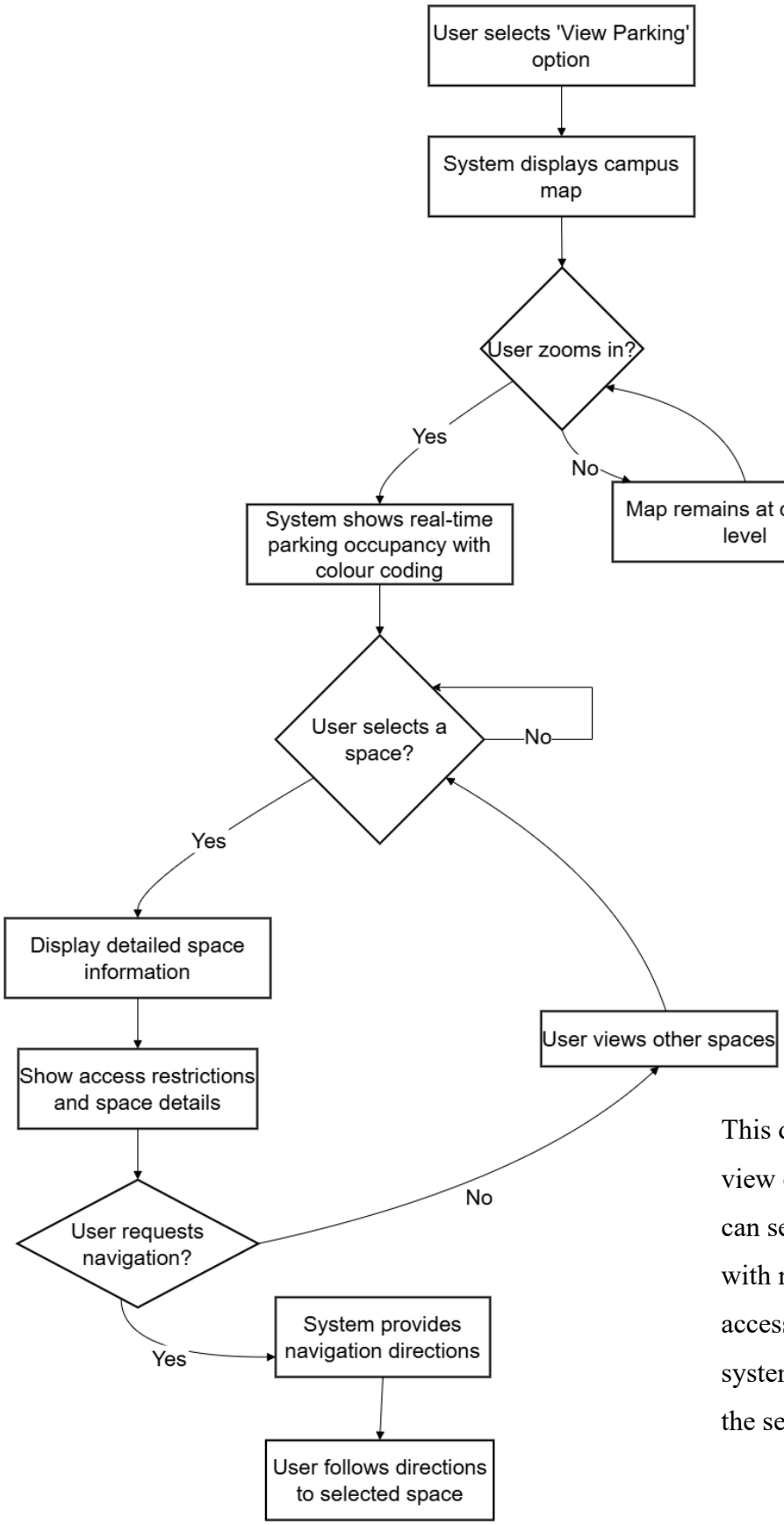
This diagram shows how users can report parking violations by filling out a form, taking a photo, and providing location details. If the photo upload fails, users can retry or submit without it. Each report is confirmed with a report ID and reviewed within 24 hours.

Use Case ID	UC004	**Version**	1.0
Feature	Request Ride		
Purpose	To allow users without vehicles to request rides from drivers		
Actor	Users, Admin		
Trigger	User needs transportation		
Precondition	User is authenticated and has completed profile		
Scenario Name	Book a Ride		
Main Flow	1. User selects "Request Ride" option 2. System displays ride request form 3. User enters pickup location and destination 4. User selects preferred departure time 5. System searches for available drivers 6. System displays matching drivers with details (photo, car info, rating) 7. User selects preferred driver or lets system auto-match 8. System sends ride request to driver 9. Driver accepts request 10. System confirms booking and provides driver contact details and car details		
Alternate Flow – **1) No Available Drivers**	1. System displays "No drivers available" message 2. User can schedule ride for later time 3. System notifies user when drivers become available		
2) Driver denies request	1. System displays “Driver declined, try again” message 2. Systems direct user back to available driver page		
Rules	• Users can only have one active ride request at a time • Cancellation allowed up to 15 minutes before pickup time		
Author	Software Requirements Engineering Team		



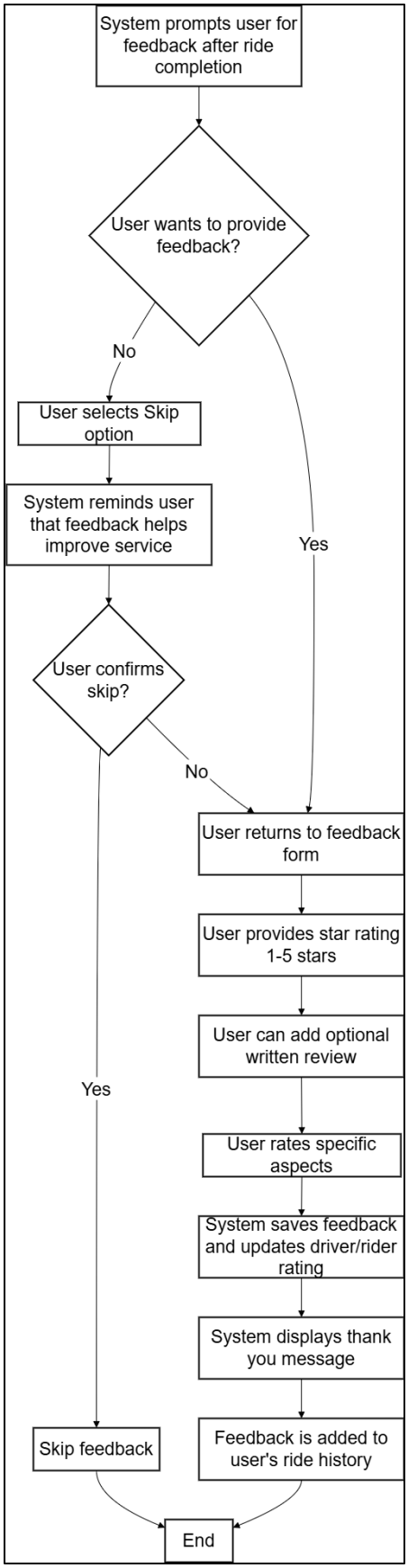
This diagram shows how a user can request a ride by entering pickup location and destination details. The system matches the user with available drivers. If no driver is found or a request is declined, the user is notified and can try again. A confirmed ride shows the driver's name, car details and contact details.

Use Case ID	UC005	**Version**	1.0
Feature	View Parking Details		
Purpose	To show real-time parking availability and details for campus parking space		
Actor	Users, Admin		
Trigger	User needs parking information		
Precondition	User is authenticated		
Scenario Name	Check Parking Availability		
Main Flow	1. User selects "View Parking" option 2. System displays campus map. 3. When zoomed in, system shows real-time parking space occupancy with colour coding (green/red) 4. User can view detailed space information including access restrictions 5. System provides navigation directions to selected space.		
Alternate Flow - Parking Suggestions	1. User enters destination building 2. System suggests best parking space based on proximity 3. System ranks suggestions by availability and walking distance		
Rules	• Parking data updates every 5 minutes • Colour coding: Green (available), Red (Unavailable) • Historical data shows peak usage times		
Author	Software Requirements Engineering Team		



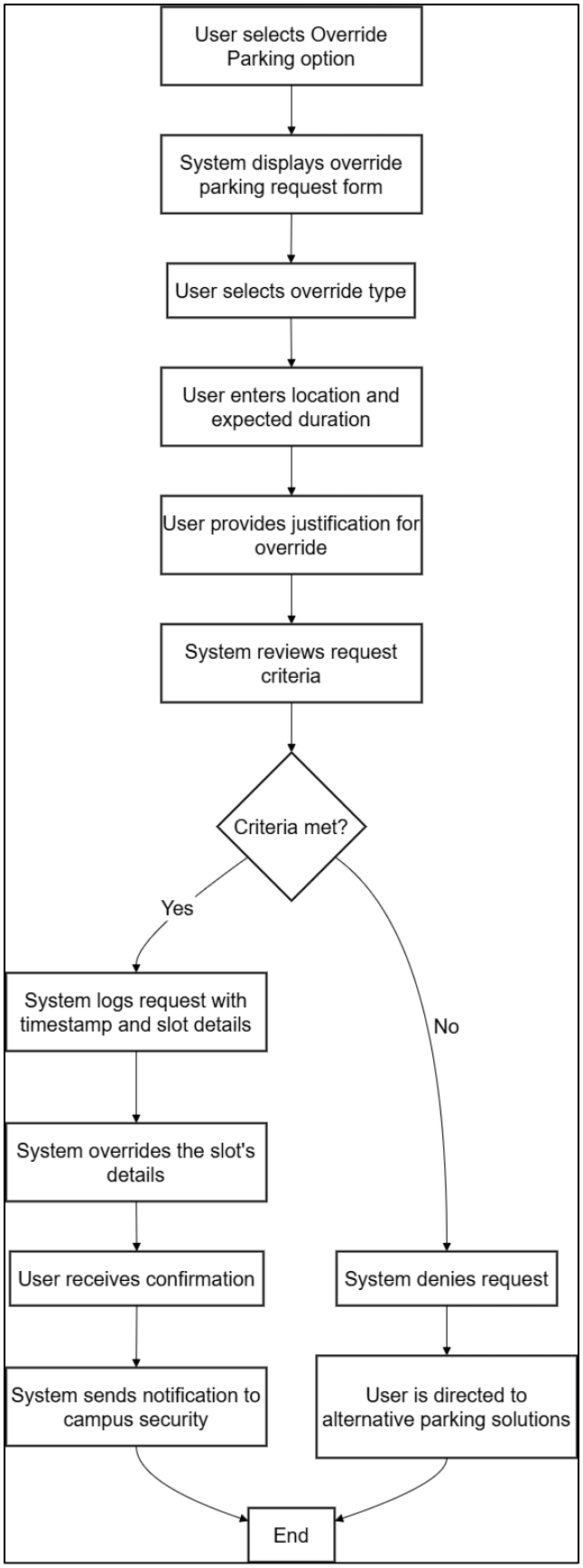
This diagram shows how users can view campus parking details. Users can see a map with parking spaces with real-time availability and access detailed lot information. The system also provides navigation to the selected lot.

Use Case ID	UC006	**Version**	1.0
Feature	Feedback		
Purpose	To allow users to rate and provide feedback on ride experiences		
Actor	Users, Admin		
Trigger	Ride completion		
Precondition	User has completed a ride (as driver or passenger)		
Scenario Name	Submit Ride Feedback		
Main Flow	1. System prompts user for feedback after ride completion 2. User provides star rating (1-5 stars) 3. User can add optional written review 4. User rates specific aspects (punctuality, cleanliness, communication) 5. System saves feedback and updates driver/rider rating 6. System displays thank you message 7. Feedback is added to user's ride history		
Alternate Flow – Skip Feedback	1. User selects "Skip" option 2. System reminds user that feedback helps improve service 3. User confirms skip or returns to feedback form		
Rules	• Feedback can be submitted within 24 hours of ride completion • Star ratings are mandatory, written reviews are optional • Inappropriate content is filtered and flagged for review		
Author	Software Requirements Engineering Team		



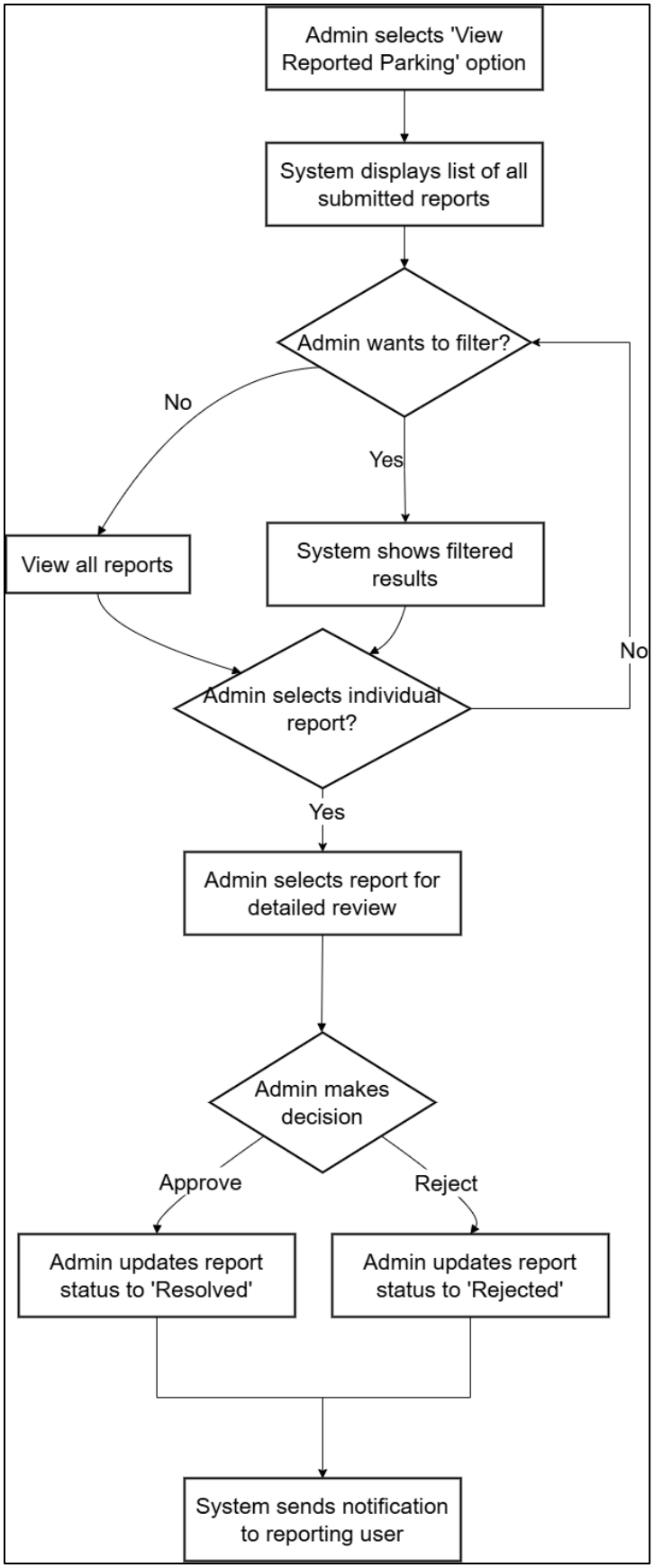
This diagram illustrates the feedback process after a ride. Users rate the ride with stars, optionally write a review, and assess key aspects. The system stores the feedback, updates ratings, and confirms submission.

Use Case ID	UC007	**Version**	1.0
Feature	Override Parking		
Purpose	To allow overriding parking slot when it is empty or driver already came out from the parking but forgot to unclaimed.		
Actor	Users, Admin		
Trigger	When car owner forgot to unclaim parking slot		
Precondition	User is authenticated and the parking slot is confirmed empty or available to park.		
Scenario Name	Overriding parking slot		
Main Flow	1. User selects "Override Parking" option 2. System displays override parking request form 3. User selects override type (forgot to unclaim / …) 4. User enters location and expected duration 5. User provides justification for override (photos of parking slot) 6. System logs request with timestamp and slot details 7. System overriding the slot's details (claimed by new driver) 8. User receives confirmation 9. System sends notification to campus security		
Alternate Flow – Override Denied	1. System reviews request criteria 2. If criteria not met, system denies request 3. User is directed to alternative parking solutions		
Rules	• Overrides requests are reviewed by campus security • Abuse of system results in account suspension		
Author	Software Requirements Engineering Team		



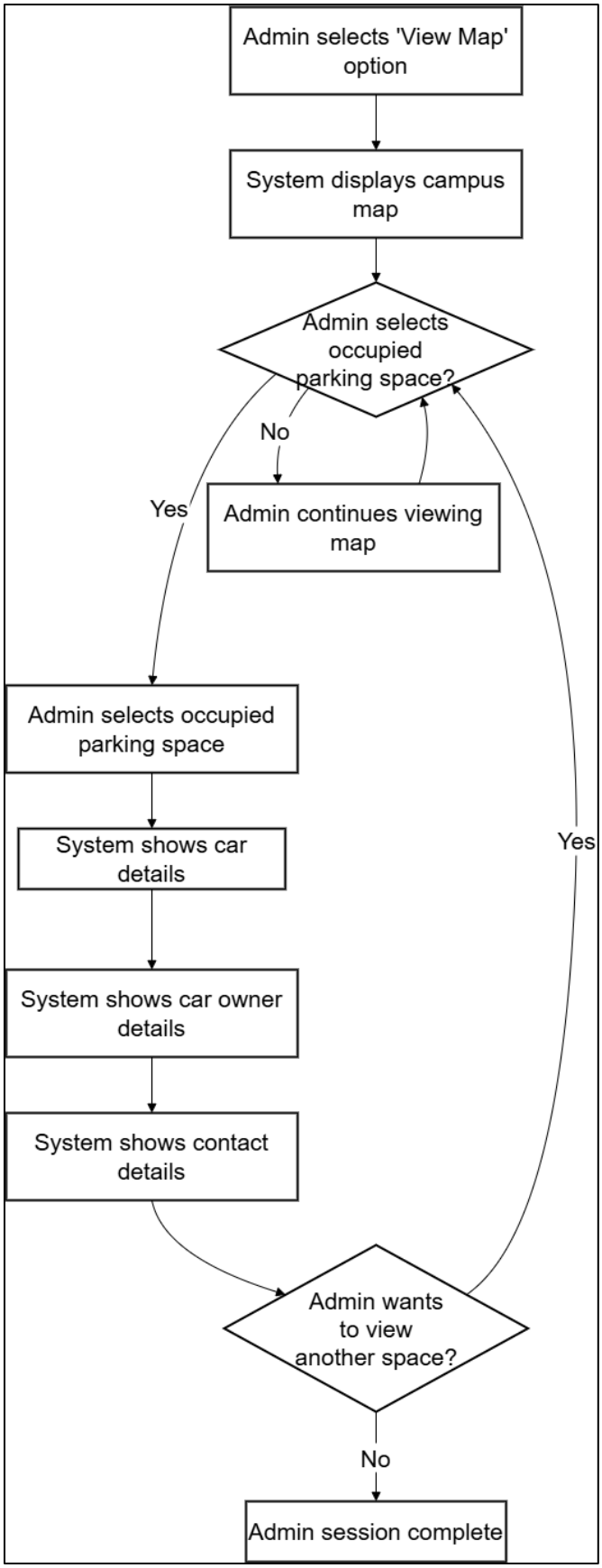
This diagram shows the steps for overriding a parking claim. The user selects the override option, fills in the form with details and justification, and submits it. The system updates the parking slot, confirms the action, and notifies campus security.

Use Case ID	UC008	**Version**	1.0
Feature	View Reported Parking		
Purpose	To allow administrators to review and manage parking violation reports		
Actor	Admin		
Trigger	Admin needs to review submitted reports		
Precondition	Admin is authenticated with administrative privileges		
Scenario Name	Review Parking Reports		
Main Flow	1. Admin selects "View Reported Parking" option 2. System displays list of all submitted reports 3. Reports are categorized by status (pending, reviewed, resolved) 4. Admin can filter by date, location, or violation type 5. Admin selects individual report for detailed review 6. System displays report details, photos, and user information 7. Admin can approve or reject 8. Admin updates report status and adds notes 9. System sends notification to reporting user about decision		
Alternate Flow – Escalate Report	1. Admin identifies serious violation 2. Admin escalates to campus security 3. System creates incident ticket and sends alerts		
Rules	• All reports must be reviewed within 48 hours • Admin actions are logged for audit trail		
Author	Software Requirements Engineering Team		



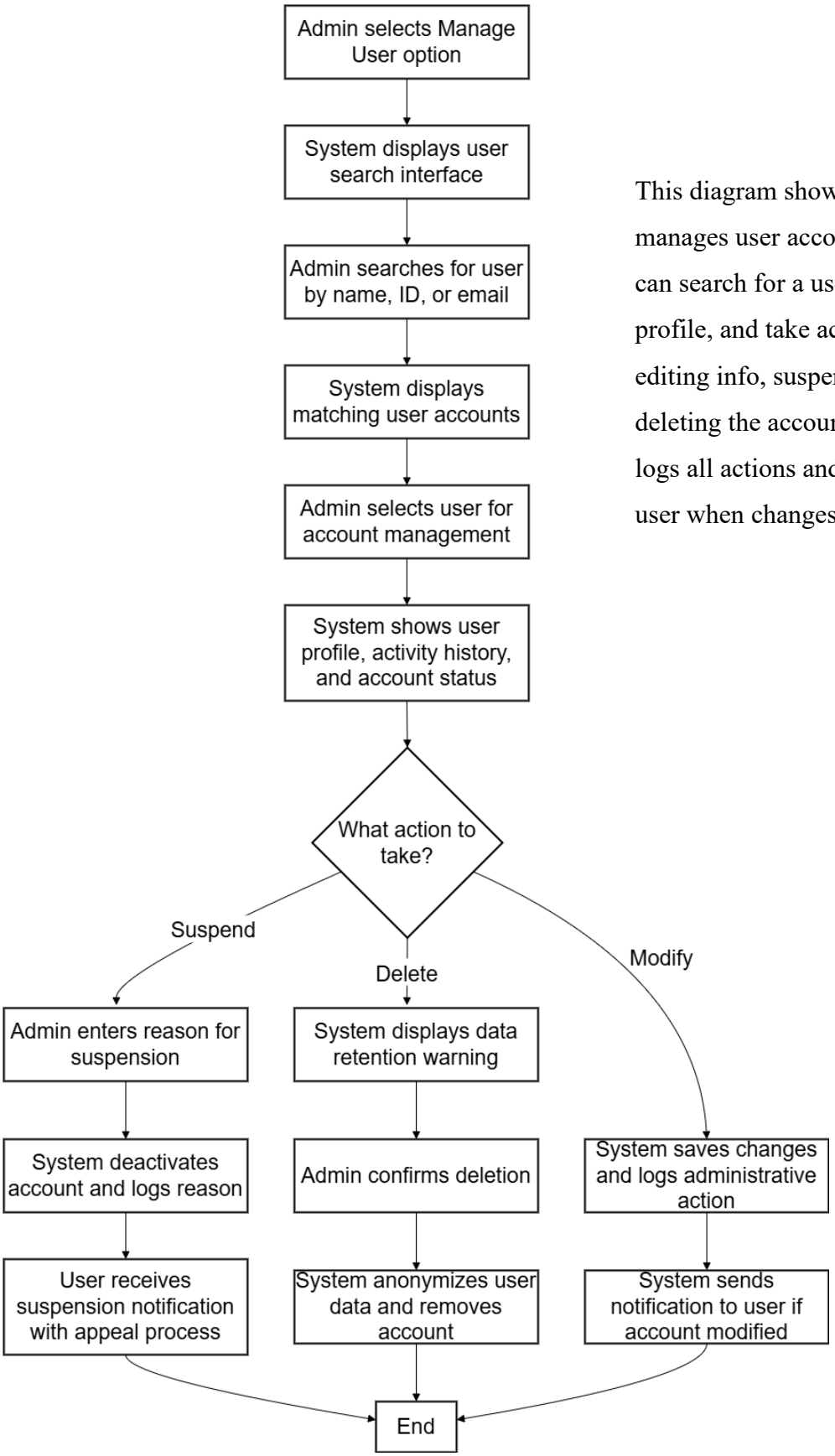
This diagram outlines how admins can review parking violation reports. Admins can filter and view detailed reports, make a decision, and update the report status. The system will then notifies the user of the outcome.

Use Case ID	UC009	**Version**	1.0
Feature	View Car Owner Details		
Purpose	To allow administrators to view and manage registered vehicle information		
Actor	Admin		
Trigger	Admin needs to verify vehicle to resolve parking dispute		
Precondition	Admin is authenticated with administrative privileges		
Scenario Name	Access Vehicle Information		
Main Flow	1. Admin selects "View Map" option 2. System displays campus map. 3. Admin selects occupied parking space. 4. System shows car details, car owner details and contact details.		
Rules	• Access requires valid administrative reason • All admin queries are logged with timestamp and justification		
Author	Software Requirements Engineering Team		



This diagram shows how an admin can use the campus map to check parking car details. By selecting a space, the system displays the vehicle and owner details.

Use Case ID	UC010	**Version**	1.0
Feature	Manage User		
Purpose	To allow administrators to manage user accounts and permissions		
Actor	Admin		
Trigger	Admin needs to modify user account or resolve user issues		
Precondition	Admin is authenticated with administrative privileges		
Scenario Name	Modify User Account		
Main Flow	1. Admin selects "Manage User" option 2. System displays user search interface 3. Admin searches for user by name, ID, or email 4. System displays matching user accounts 5. Admin selects user for account management 6. System shows user profile, activity history, and account status 7. Admin can modify : account status, permissions, profile information 8. Admin can warning, block or unlock accounts 9. System saves changes and logs administrative action 10. System sends notification to user if account modified		
Alternate Flow – 1)Account Suspension	1. Admin selects "Suspend Account" option 2. Admin enters reason for suspension 3. System deactivates account and logs reason 4. User receives suspension notification with appeal process		
2) Account Deletion	1. Admin initiates account deletion process 2. System displays data retention warning 3. Admin confirms deletion 4. System anonymizes user data and removes account		
Rules	□ Account modifications require documented justification □ User data must comply with privacy regulations □ Audit trail maintained for all administrative actions		
Author	Software Requirements Engineering Team		



This diagram shows how the admin manages user accounts. The admin can search for a user, view their profile, and take actions like editing info, suspending, or deleting the account. The system logs all actions and notifies the user when changes are made.

3.2 Performance Requirements

Requirement ID	Description	Priority
REQ_P001	The average response time of searching for a ride is 3 milliseconds	HIGH
REQ_P002	The system shall support at least 500 concurrent users without significant performance degradation.	HIGH
REQ_P003	The average response time of searching for an available parking slot is less than 5 milliseconds	HIGH
REQ_P004	The average process time of updating the parking slot availability is less than 10 milliseconds	HIGH
REQ_P005	The average time for the process of user log in operation is less than 5 milliseconds	HIGH

The requirements above defines how well the system must perform under various conditions and constraints. The app must perform efficiently under both regular and peak usage conditions. It should provide real-time updates for ride-matching and parking availability, responding to user actions within 2 seconds and maintaining at least 99.9% uptime. The system backend should support high concurrency, handling over 1,000 API requests per minute without lag.

3.3 Usability Requirements

Requirement ID	Description	Priority
REQ_U001	The app should use clear, campus-branded visuals and icons for navigation and function clarity.	HIGH
REQ_U002	The interface shall be accessible and usable on common screen sizes (phones and tablets) with touch-optimized controls.	HIGH
REQ_U003	Feedback (e.g., confirmation or error messages) shall be provided to the user within 1 second of completing any major action.	MEDIUM

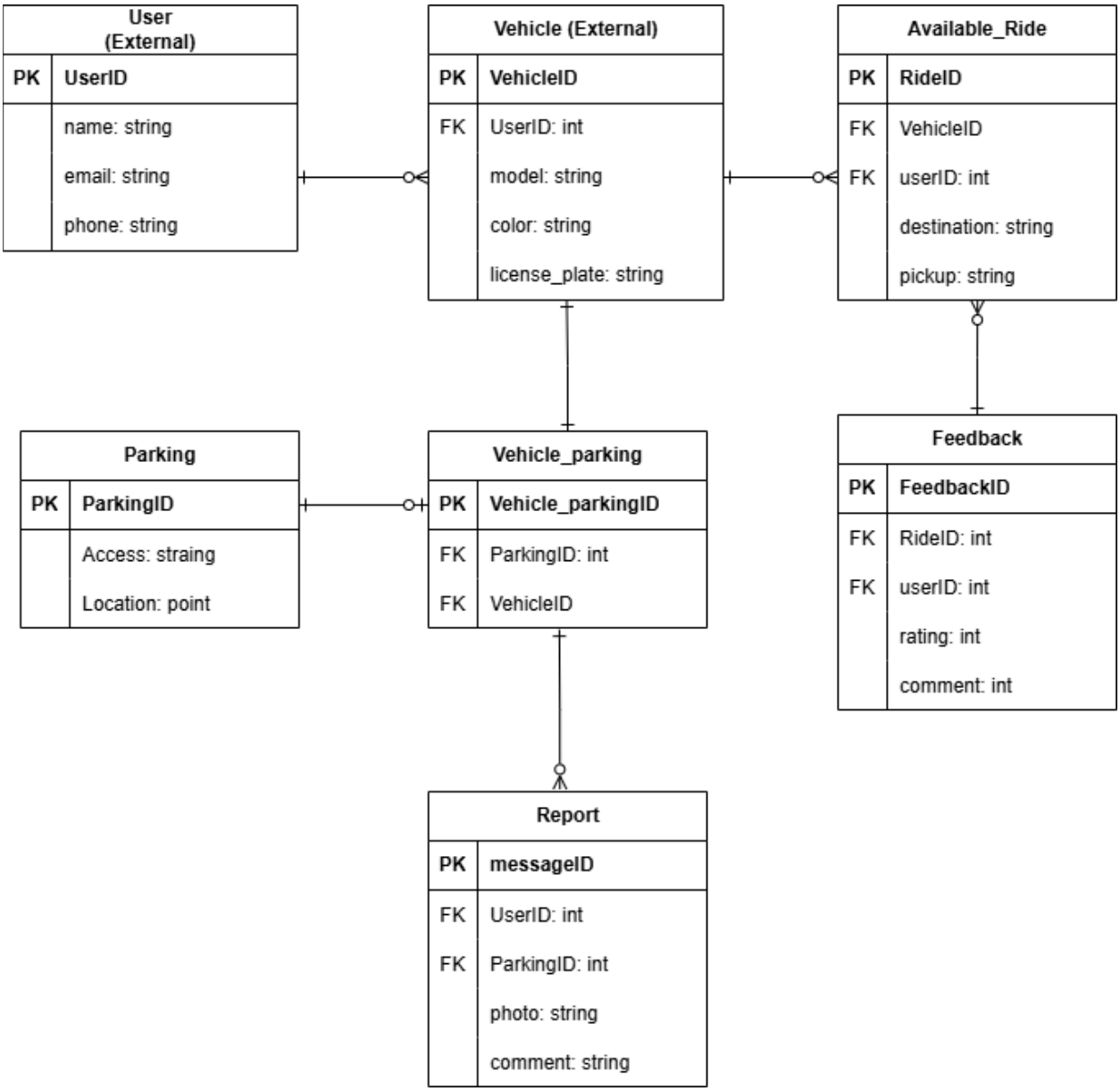
The usability requirements show how easy it is for users to interact with the app. The application must be user-friendly, enabling first-time users to complete essential tasks like booking a ride or reserving a parking spot within 5 minutes. It should offer a clean, campus-branded interface compatible with mobile devices and accessible for users with disabilities. Clear visual cues and immediate feedback must guide users through all actions, enhancing ease of use and satisfaction.

3.4 Interface Requirements

Requirement ID	Description	Priority
REQ_I001	Users shall log in via the campus Single Sign-On (SSO) system.	HIGH
REQ_I002	The app shall interface with GPS sensors in smartphones to determine user location.	HIGH
REQ_I003	It shall interact with campus APIs for student data, parking rules, and access control.	MEDIUM
REQ_I004	The app shall integrate with mapping APIs (e.g., Google Maps) for route planning and navigation.	HIGH
REQ_I005	The system shall support push notifications for ride updates, reminders, and alerts.	MEDIUM
REQ_I006	All data transmissions between the app and external systems shall use secure HTTPS protocols.	HIGH
REQ_I007	The system shall retrieve user and vehicle data from the university database through secured APIs.	HIGH
REQ_I008	The system shall access the smartphone camera for uploading violation photo evidence.	MEDIUM
REQ_I009	The app shall display a mobile-friendly UI with bottom navigation and touch-friendly buttons.	HIGH
REQ_I010	The system shall support Android and iOS mobile platforms with standard GPS and camera access.	HIGH
REQ_I011	The app shall refresh parking space data every 30 seconds using secure polling.	MEDIUM
REQ_I012	The system shall use Firebase or a similar service for backend data storage and notification delivery.	MEDIUM

These requirements specify how the app interacts with users, hardware and other systems. The app should integrate seamlessly with campus systems and external services. Users will interact through a mobile interface connected to GPS, parking sensors, and mapping APIs, while authentication will be handled via campus SSO. All data exchange should be secure, using HTTPS, and the system should support real-time communication through push notifications and cloud-based APIs.

3.5 Logical Database Requirements



ERD Diagram of Campus Ride-Sharing Platform with Parking System

Requirement ID	Requirement Description	Priority
REQ_D001	Data such as UserID, User name and contact number will be retrieved from an API integration from NICE MMU, to prevent storing redundant data.	HIGH
REQ_D002	Each vehicle must be linked to a specific user and must include a license plate number, model, and color.	HIGH
REQ_D003	The system must store parking space data, including location and vehicle if occupied.	HIGH
REQ_D004	The system must store ride booking details such as requester ID, driver ID, pick-up/drop-off location and ride status.	MEDIUM
REQ_D005	The system must allow logging of parking violations, including the reporter's ID, reported parking space, timestamp, and violation type.	HIGH
REQ_D006	Admin users must be able to retrieve all reported violations and the corresponding vehicle/user information.	HIGH
REQ_D007	The system shall use campus Single Sign-On (SSO) for authentication and shall not store or process user passwords. User roles shall be managed for access control.	MEDIUM
REQ_D008	The system shall support querying real-time parking availability, excluding already occupied spaces.	LOW
REQ_D009	The system must support storing map data for pathing and display of entrances, exits, and parking zones.	LOW

3.6 Design Constraints

Requirement ID	Requirement Description	Priority
REQ_DC001	The user interface must comply with MMU's official branding guidelines, including colour schemes and logo usage.	HIGH
REQ_DC002	The system must integrate with the existing MMU Mobile App ecosystem and shall not function as a standalone application.	HIGH
REQ_DC003	The system shall design around MMU's Single Sign-On (SSO) service for all authentication processes.	HIGH
REQ_DC004	The system must comply with the Personal Data Protection Act (PDPA) and avoid storing sensitive information such as plaintext passwords.	HIGH
REQ_DC005	The application shall be compatible with both Android and iOS platforms, supporting the latest three versions of each OS.	MEDIUM
REQ_DC006	Real-time features such as map and parking availability updates shall occur at intervals not exceeding 30 seconds.	MEDIUM
REQ_DC007	The user interface must have another interface colour palette suitable for dark mode, while adhering to MMU's colour scheme as close as possible	LOW

The table above outlines the Design Constraints for the system. These constraints specify technical, legal, and visual requirements that must be followed during development. They include integration with MMU systems, compliance with branding and data protection regulations, platform compatibility, update intervals for real-time features, and support for dark mode.

3.7 Software System Attributes

Attribute	Requirement Description
Reliability	The system shall recover from unexpected crashes or failures within 1 minute to minimize user disruption.
Availability	The system shall maintain 99.9% uptime, especially during campus operational hours (Monday to Friday, 8 AM – 6 PM).
Security	The system shall use MMU Single Sign-On (SSO) for secure login. All sensitive data, such as personal user information and vehicle data, shall be encrypted in transit.
Maintainability	The system shall be developed using modular code structure and clear documentation to allow easy maintenance, updates, and debugging.
Portability	The mobile interface shall support Android and iOS platforms (on the latest 3 versions) while the backend server should be able to run on windows server.
Scalability	The system shall support scaling to accommodate future increases in users and ride requests without major architecture changes.
Performance	Real-time data (such as parking availability and map updates) shall be refreshed at intervals not exceeding 30 seconds.
Usability	The system shall offer an intuitive user interface with accessible features suitable for students, staff, and admin users of all age groups.

This section outlines the non-functional quality requirements of the system, detailing the expected standards for reliability, availability, security, maintainability, portability, scalability, performance, and usability to ensure a robust and user-friendly experience.

3.8 Supporting Information

This section provides additional details that support understanding and development of the system:

A) Sample Input/Output Format:

To communicate between the frontend interfaces and backend servers, our system use internal API request/response patterns.

Example (for submitting a parking violation report):

Request (JSON):

```
{
  "userID": "U123",
  "violationType": "Illegal Parking",
  "photo": "base64string...", #To be located in folder
  "parkingID": "P3321"
}
```

Response (JSON):

```
{
  "status": "success",
  "reportID": "R456"
}
```

B) Supporting/Background Information:

The application is part of the MMU Mobile ecosystem and is integrated via MMU's Single Sign-On (SSO) and student data APIs. No sensitive information such as passwords will be stored locally, but User information and vehicle information will be

C) Problem Description:

The system aims to improve campus transportation and parking efficiency through real-time data, ride-sharing, and user-reporting features. It also facilitates administrative oversight for parking violations and ride activity.

D) Packaging/Security Considerations:

The application must comply with MMU's deployment procedures. Code and assets shall be packaged securely in a private Github and server to prevent unauthorized access. All data transmission must occur over secure HTTPS connections. Internal APIs will be used for all client-server interactions, even within the same system, to ensure modularity and scalability.

4 Verification

4.1 Verification Approach

To ensure that the system meets all functional and non-functional requirements, the following verification strategy will be applied:

After the implementation phase, the system will undergo the following types of testing:

- **Unit Testing** – to verify individual components such as login, map display, and report submission.
- **Integration Testing** – to ensure proper communication between modules (e.g., user data with parking availability).
- **System Testing** – to validate the entire application behaves as expected under realistic conditions.
- **User Acceptance Testing** – to confirm the system meets user needs through feedback from real users.
- **Security and Performance Testing** – to ensure data protection, SSO compliance, and system responsiveness.

The test will be conducted by the respective team.

- **Unit and integration tests** will be carried out by the development team.
- **System testing and user acceptance testing** will be conducted by the quality assurance team and selected end users.
- **Security and compliance checks** will involve the IT security and stakeholders from MMU's department of IT.

The test will be conducted during the selected time.

- **Unit Testing**: After development of each module.
- **Integration Testing**: At the end of each sprint.
- **System and User acceptance Testing**: At the end of development cycle before deployment.
- **Security & Performance Testing**: During final testing phase before release.

The test will be conducted under the selected scenario

- All verification activities will take place in a realistic environment that mirrors what will happen during the daily usage of the MMU mobile application

4.2 Verification Criteria

Criteria ID	Verification Criteria
VC001	The system shall respond to map-related requests (e.g., viewing parking availability) within 3 seconds under normal load and refresh shall happen every 30 seconds.
VC002	Login through the MMU Single Sign-On (SSO) must complete in under 5 seconds.
VC003	Users must be able to submit parking violation reports with photo and description successfully in 98% of test cases.
VC004	Ride requests must match users with available drivers in under 10 seconds or notify users if none are available.
VC005	Admin should be able to view car details when clicking on occupied parking spaces 98% of test cases under 0.5 seconds.
VC006	The system shall maintain 99.9% uptime, especially during campus operational hours (Monday to Friday, 8 AM – 6 PM).
VC007	All personal and sensitive data must be encrypted during transmission (verified via HTTPS and secure logs).
VC008	The user interface shall display properly and be functional on both Android and iOS.

5 Appendix

5.1 Assumptions and dependencies

Below are the expected assumptions and dependencies for our system to be working properly and as documented. Any exceptions would bring issues and must be addressed as soon as possible.

ID	Assumption / Dependency	Description
AD-001	Campus SSO Availability	It is assumed that MMU's Single Sign-On (SSO) service is operational and accessible for user authentication and MMU's IT department allows the transfer of user data
AD-002	External API Stability	The system depends on third-party APIs provided by MMU thus it is assumed to be stable and properly documented for implementation.
AD-003	Mobile Device Permissions	Users will grant necessary permissions (e.g., location, camera) so that the system's core features can function.
AD-004	Network Connectivity	The system assumes users always have an active internet connection during usage.
AD-005	Up-to-date Device OS	The mobile app assumes users are on supported versions (latest 3 versions) of Android/iOS.
AD-006	Admin Availability	System administrators are available to review reports, handle reports and understand the basic controls.
AD-007	Device Hardware Support	Devices have necessary hardware such as GPS and camera enabled.
AD-008	MMU Branding Guidelines	The user interface design assumes MMU branding guidelines remain unchanged during development.

5.2 Acronyms and abbreviations

Abbreviations / Acronyms	Meaning
MMU	Multimedia University, specifically Cyberjaya Campus
NICE	Multimedia University's Network & Intelligent Campus Ecosystem
SSO	Single Sign-On service, login doesn’t need to happen in the system as long as it happened in the app