

**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 3**](#_8qhk8rbclkmo)

[1.1 Problem Statement 3](#_4v7znnun3zh4)

[1.2 Vision 3](#_o9f67vy27l3)

[1.3 Scope 4](#_wi4uzsbz4l3b)

[1.4 Purpose 4](#_5uy3u3vxj47b)

[1.5 Goals 4](#_68ac5kpjsfzr)

[1.6 References 5](#_fynhq0z0nam3)

[**2 Product Overview 5**](#_g156thw0ybsw)

[2.1 Product Perspective 5](#_clpaxk4g822q)

[2.2 Product Functions 6](#_hdiiialvyllf)

[2.3 Product Characteristics 8](#_fsoeyxwk6833)

[2.4 Limitations 9](#_9d3r0v7bqxr3)

[**3 Requirements 11**](#_odfxfrkvacet)

[3.1 Functions 11](#_jt8bb5fqpmi2)

[3.2 Performance Requirements 11](#_z6k751gx52st)

[3.3 Usability Requirements 11](#_nwvck47ltin8)

[3.4 Interface Requirements 11](#_5djepz9hbhoe)

[3.5 Logical Database Requirements 11](#_3eyo5qq74kla)

[3.6 Design Constraints 11](#_jp33usrxvqvi)

[3.7 Software System Attributes 11](#_325hrgwxbxeu)

[3.8 Supporting Information 11](#_75ck7psircmx)

[**4 Verification 11**](#_f237zn9f3707)

[4.1 Verification Approach 11](#_ltma1231bdog)

[4.2 Verification Criteria 11](#_xu7mpkwux0ej)

[**5 Appendix 11**](#_zenz8ijvhdw9)

[5.1 Assumptions and dependencies 11](#_9zq6gvnp7wol)

[5.2 Acronyms and abbreviations 12](#_en4rmp2cpbq9)

[5.3 Glossary 12](#_9qn5nlgdapc)

# 

# 1 Introduction

## 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.

## 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.

# 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

A screen shot of a black screen

AI-generated content may be incorrect.

The block diagram above shows the campus ride-sharing and Parking App is part of a campus mobile app that includes services like campus shuttles, public transport integration, and vehicle access control. It connects to the campus authentication system for secure login and to APIs for real-time parking data and ride-matching. The app also interfaces with GPS to track user location and space availability. Data flows between the app, cloud services, and external systems to provide a seamless user experience. These connections ensure efficient coordination between parking, carpooling, and campus-wide transport services.

The campus ride-sharing system with parking system interfaces with campus authentication system for secure user authentication and integrates with parking and mobility APIs for real-time data exchange. Its mobile friendly user interface allows users to book rides, view parking availability, and receive notifications. The app connects to hardware such as GPS modules and smartphones for accurate tracking and monitoring. It also works alongside software components like mapping APIs and the university’s backend systems. Communication relies on secure HTTPS protocols, with real-time updates delivered via pushing notifications. The app is optimized for typical mobile device memory limits and uses cloud storage to reduce local memory usage. Operationally, it includes an admin dashboard for monitoring and managing activity. Site adaptation allows for custom configurations based on specific campus policies, such as reserved zones or academic schedules. Finally, the app integrates with services like email, SMS, and potentially payment gateways to enhance functionality and user experience.

## 2.2 Product Functions

**User Capabilities:**

* Verify identity using StudentID and password
* Open and zoom into the school map
* View parking spaces and car plate details
* Report illegitimate parking
* Override or empty a parking space
* Book a ride to a faculty member

**Admin Capabilities:**

1. All User Functions
2. Log in using admin ID and password
3. View reported parking spaces
4. View car owner details

## 2.3 Product Characteristics

View ISO docs 9.6.6

## 

## 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

## 3.1 Functions

## 3.2 Performance Requirements

## 3.3 Usability Requirements

## 3.4 Interface Requirements

## 3.5 Logical Database Requirements

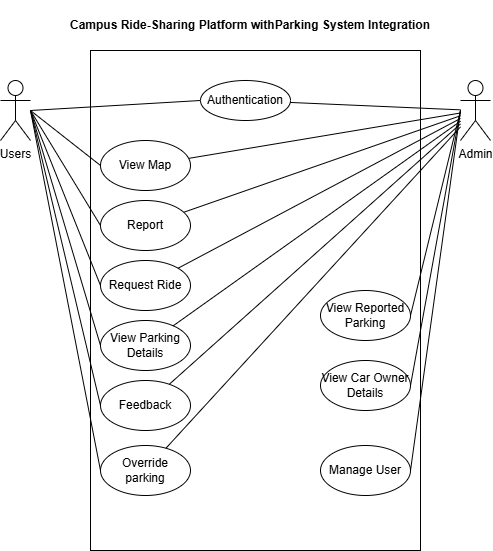
## 3.6 Design Constraints

## 3.7 Software System Attributes

## 3.8 Supporting Information

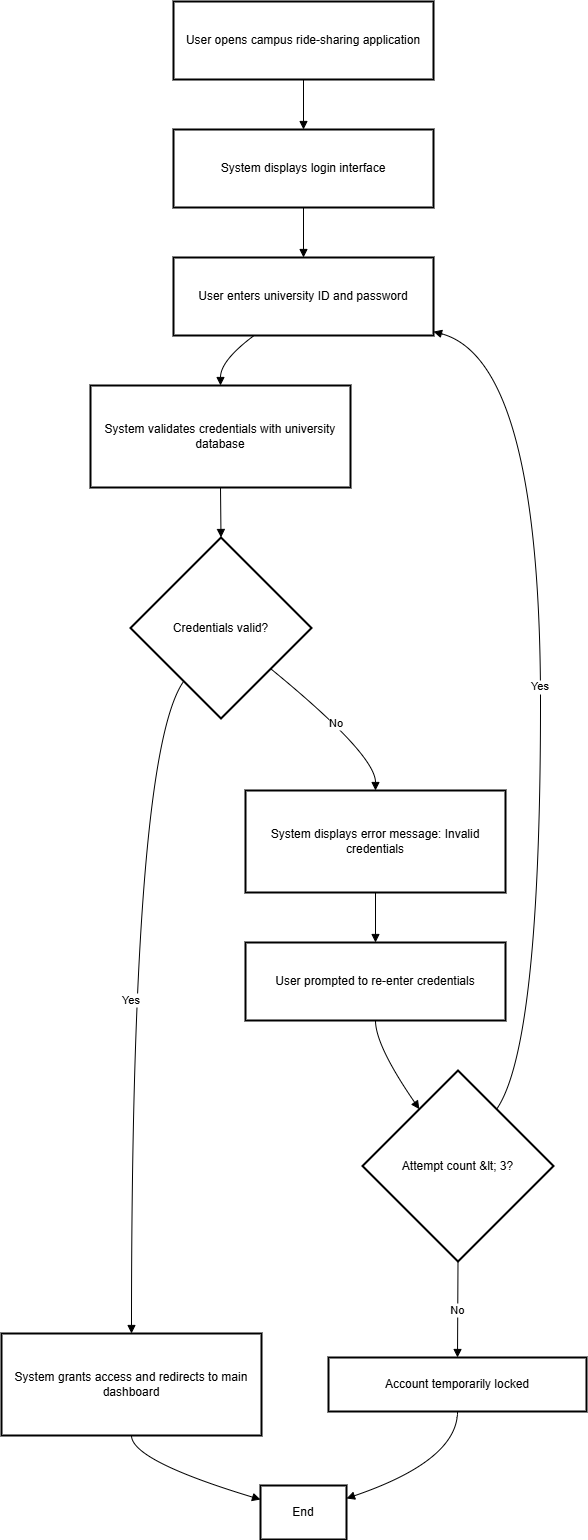
# 4 Use Case

4.1 Use Case Diagram

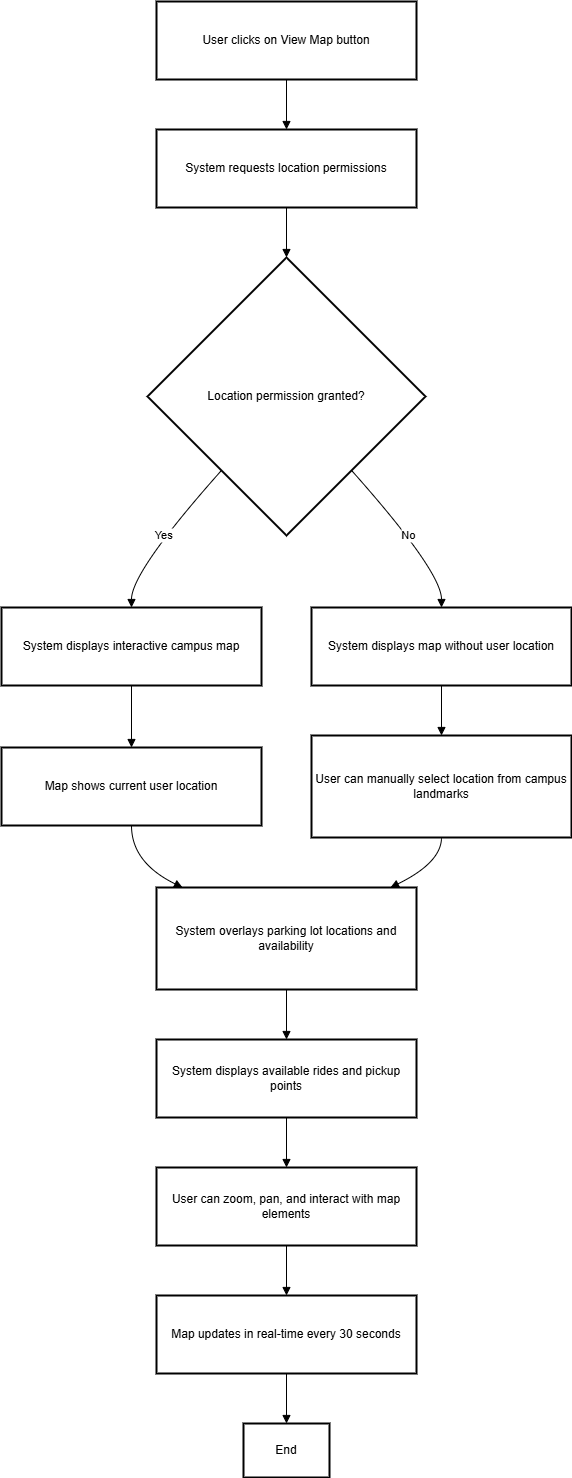


4.2 Detailed Use Case Descriptions

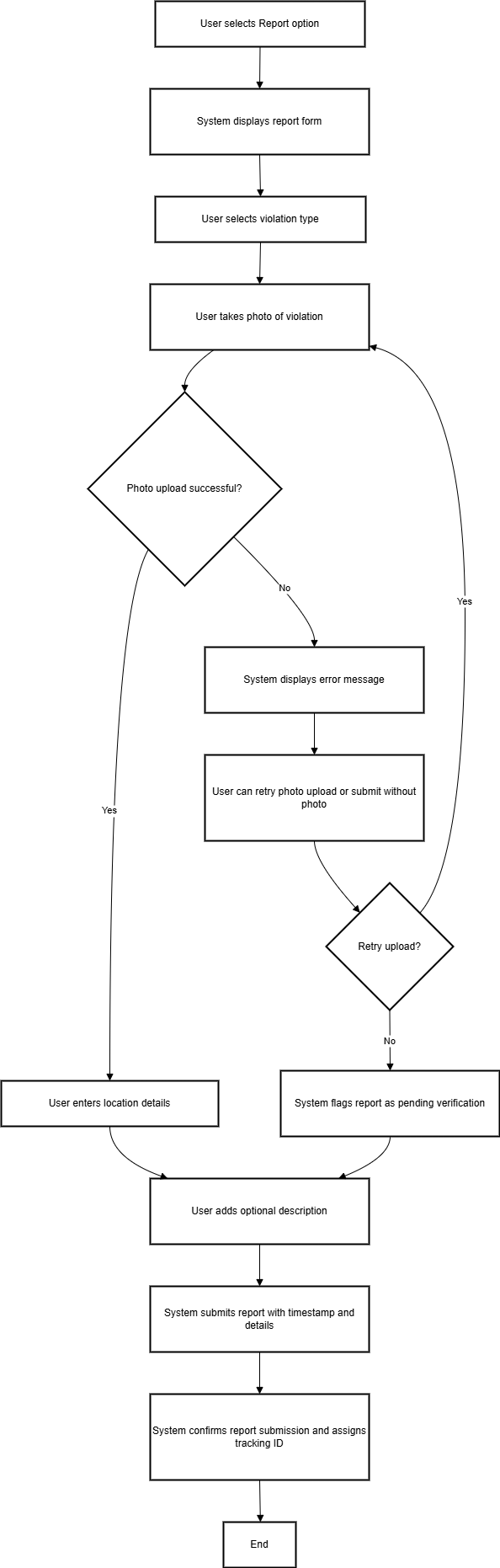
|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC001 | **Version** | 1.0 |
| **Feature** | User Authentication | | |
| **Purpose** | To allow users to securely log into the system using university credentials. | | |
| **Actor** | Users ( Driver / Rider), Admin | | |
| **Trigger** | User attempts to access the platform | | |
| **Precondition** | User has valid university account credentials | | |
| **Scenario Name** | User Auth | | |
| **Main Flow** | 1. User opens the campus ride-sharing 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 | | |
| **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 | | |



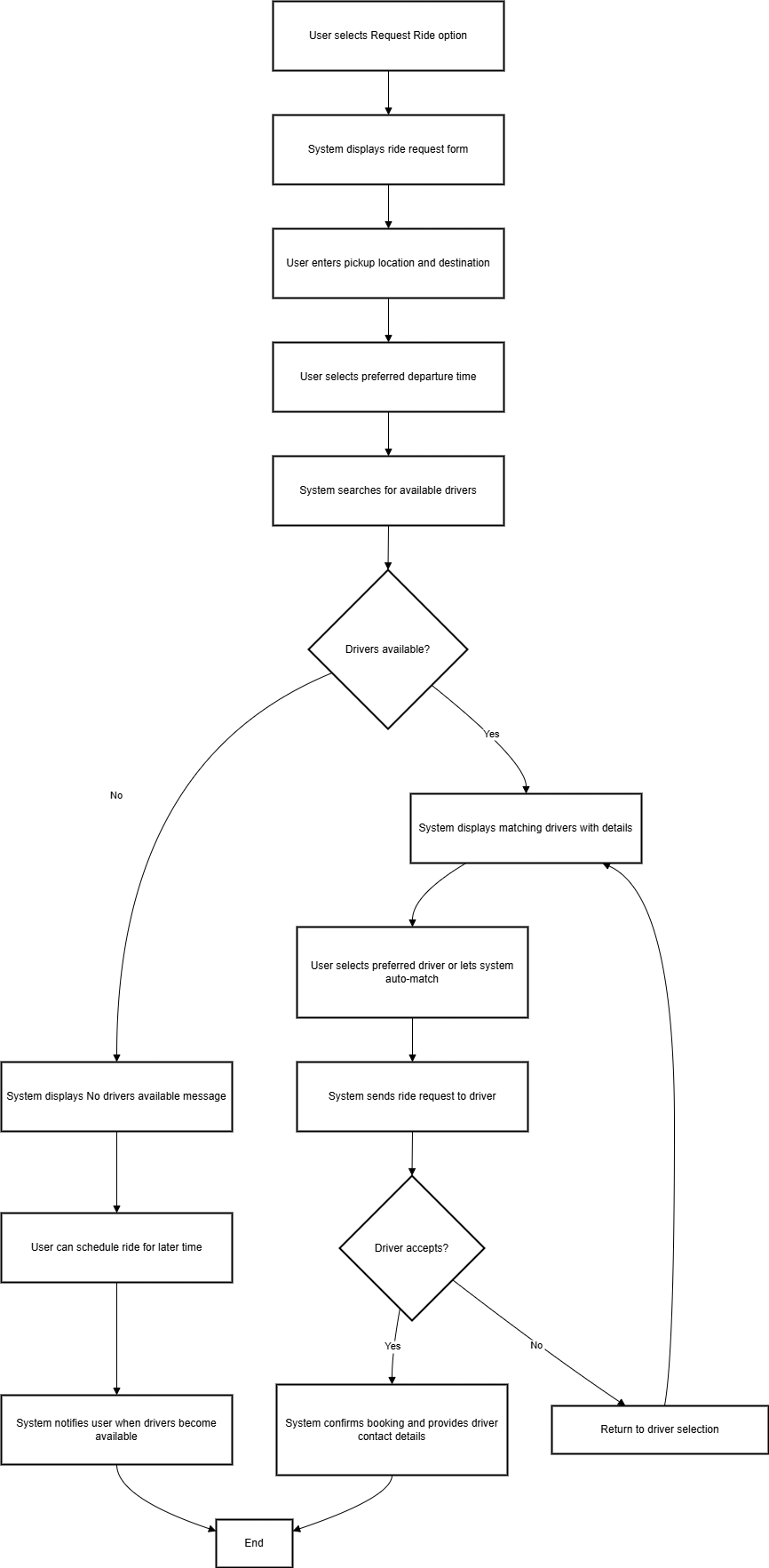
|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC002 | **Version** | 1.0 |
| **Feature** | View Map | | |
| **Purpose** | To display real-time campus map with parking and ride information | | |
| **Actor** | Users ( Driver / Rider), Admin | | |
| **Trigger** | User selects "View Map" option | | |
| **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 | | |



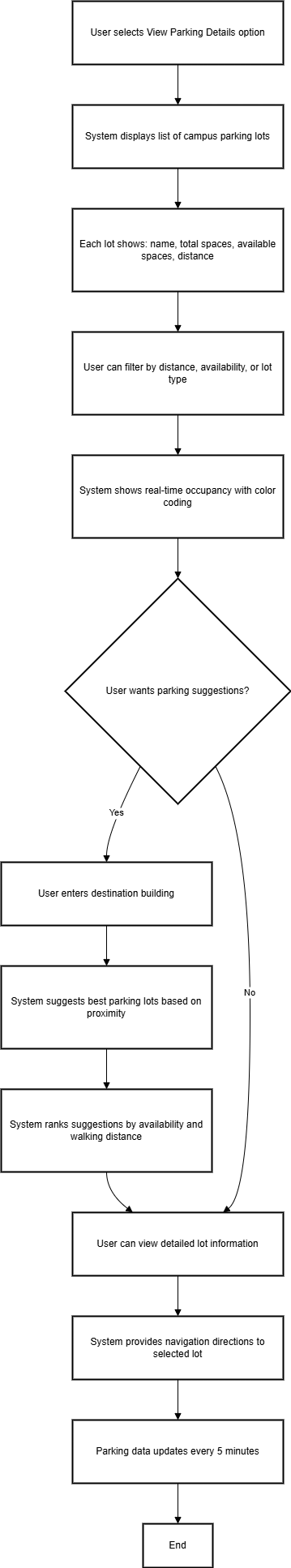
|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC003 | **Version** | 1.0 |
| **Feature** | Report | | |
| **Purpose** | To allow users to report parking violations with photo evidence | | |
| **Actor** | Users ( Driver / Rider), Admin | | |
| **Trigger** | User witnesses parking violation | | |
| **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 tracking 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 | | |



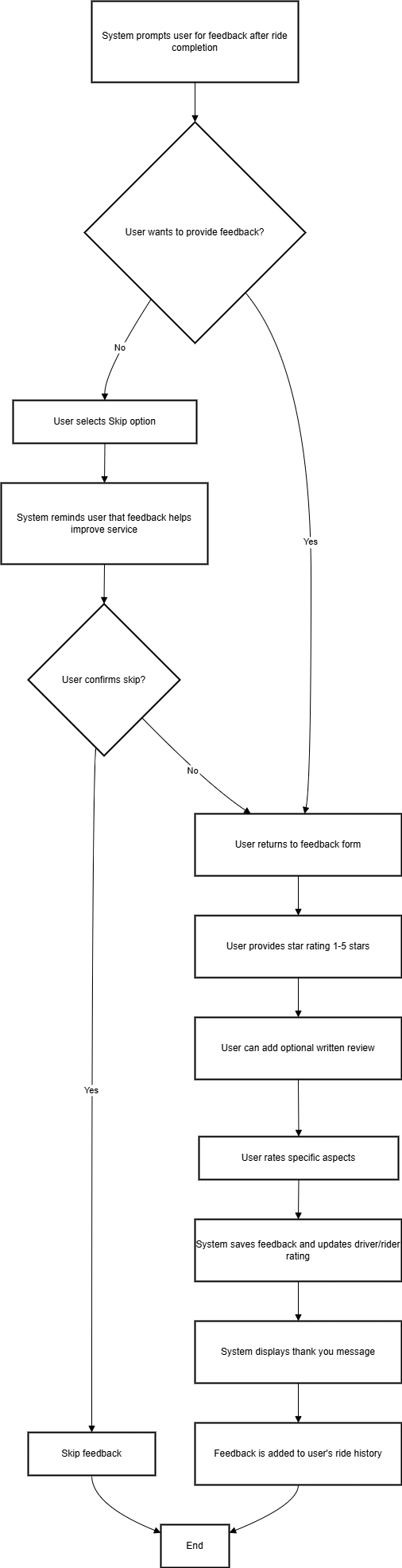
|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC004 | **Version** | 1.0 |
| **Feature** | Request Ride | | |
| **Purpose** | To allow users without vehicles to request rides from drivers | | |
| **Actor** | Users ( Driver / Rider), Admin | | |
| **Trigger** | User needs transportation | | |
| **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 | | |
| **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) Recurring Booking** | 1. User selects "Recurring Ride" option 2. User sets schedule (daily/weekly) 3. System automatically books rides based on schedule | | |
| **Rules** |  Ride requests can be made up to 24 hours in advance   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 | | |



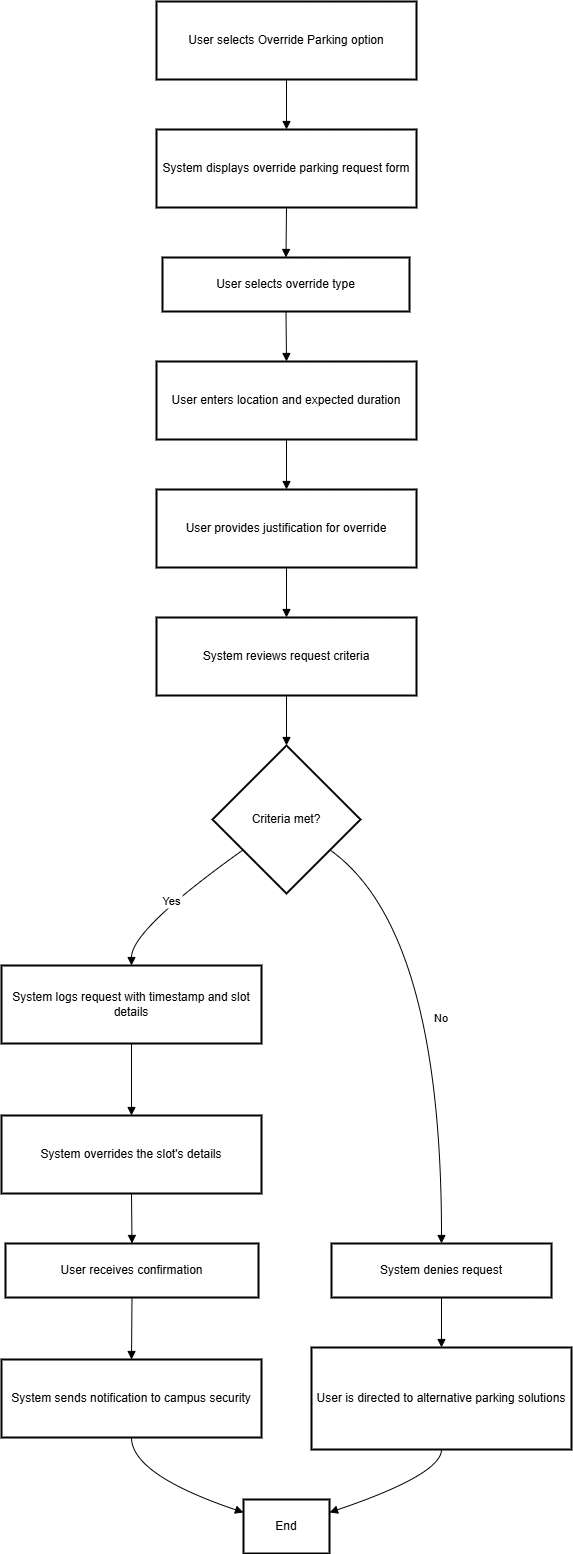
|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC005 | **Version** | 1.0 |
| **Feature** | View Parking Details | | |
| **Purpose** | To show real-time parking availability and details for campus lots | | |
| **Actor** | Users ( Driver / Rider), Admin | | |
| **Trigger** | User needs parking information | | |
| **Precondition** | User is authenticated | | |
| **Scenario Name** | Check Parking Availability | | |
| **Main Flow** | 1. User selects "View Parking Details" option 2. System displays list of campus parking lots 3. Each lot shows: name, total spaces, available spaces, distance from user 4. User can filter by distance, availability, or lot type 5. System shows real-time occupancy with colour coding (green/yellow/red) 6. User can view detailed lot information including access restrictions 7. System provides navigation directions to selected lot | | |
| **Alternate Flow - Parking Suggestions** | 1. User enters destination building 2. System suggests best parking lots based on proximity 3. System ranks suggestions by availability and walking distance | | |
| **Rules** |  Parking data updates every 5 minutes   Colour coding: Green (>20% available), Yellow (5-20%), Red (<5%)   Historical data shows peak usage times | | |
| **Author** | Software Requirements Engineering Team | | |



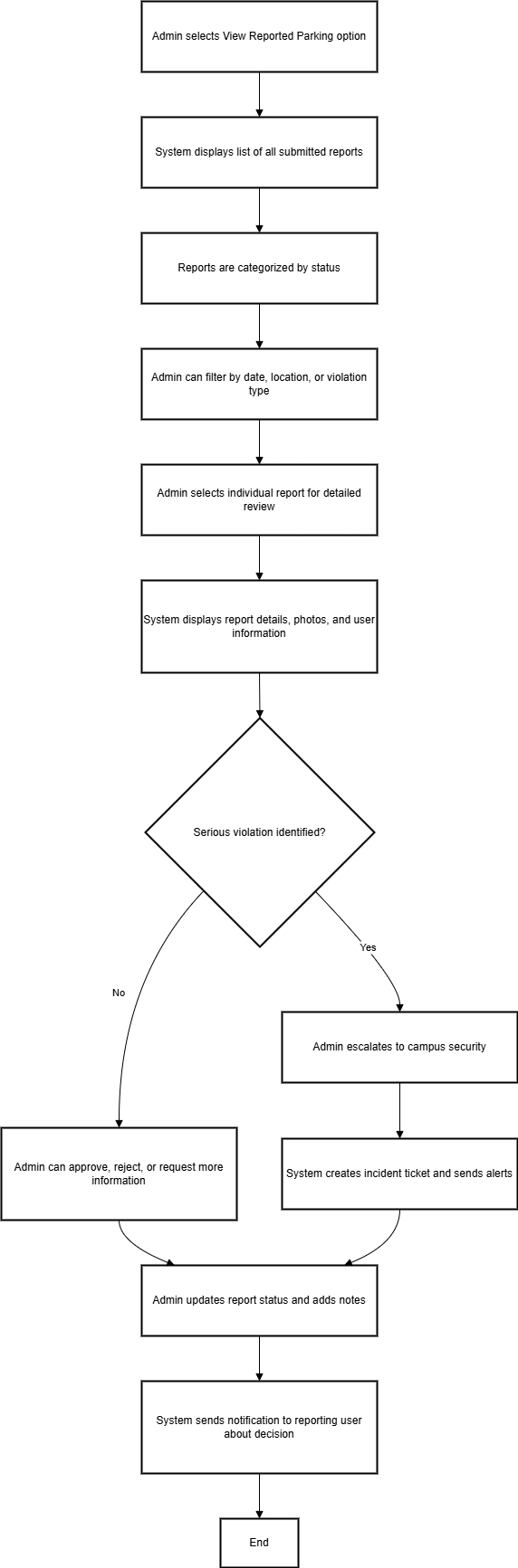
|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC006 | **Version** | 1.0 |
| **Feature** | Feedback | | |
| **Purpose** | To allow users to rate and provide feedback on ride experiences | | |
| **Actor** | Users ( Driver / Rider), Admin | | |
| **Trigger** | Ride completion | | |
| **Precondition** | 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 | | |



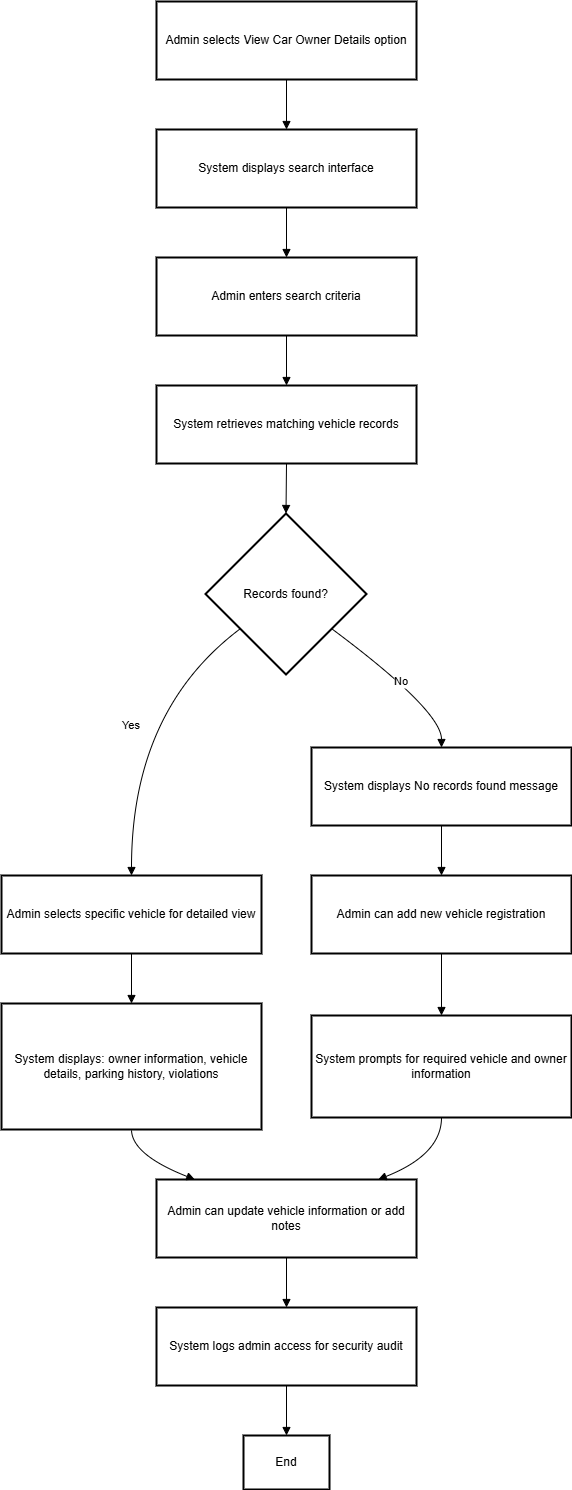
|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID** | UC007 | **Version** | 1.0 |
| **Feature** | Override Parking | | |
| **Purpose** | To allow overriding parking slot when it is empty or driver already came out from the parking but forgot to unclaimed. | | |
| **Actor** | Users ( Driver / Rider), 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** |  Override duration maximum 2 hours   Overrides requests are reviewed by campus security   Abuse of system results in account suspension | | |
| **Author** | Software Requirements Engineering Team | | |



|  |  |  |  |
| --- | --- | --- | --- |
| **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, reject, or request more information 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 | | |



|  |  |  |  |
| --- | --- | --- | --- |
| **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 or resolve parking dispute | | |
| **Precondition** | Admin is authenticated with administrative privileges | | |
| **Scenario Name** | Access Vehicle Information | | |
| **Main Flow** | 1. Admin selects "View Car Owner Details" option 2. System displays search interface 3. Admin enters search criteria (license plate, owner name, or student ID) 4. System retrieves matching vehicle records 5. Admin selects specific vehicle for detailed view 6. System displays: owner information, vehicle details, parking history, violations 7. Admin can update vehicle information or add notes 8. System logs admin access for security audit | | |
| **Alternate Flow – Vehicle Not Found** | 1. System displays "No records found" message 2. Admin can add new vehicle registration 3. System prompts for required vehicle and owner information | | |
| **Rules** |  Access requires valid administrative reason   All admin queries are logged with timestamp and justification | | |
| **Author** | Software Requirements Engineering Team | | |



|  |  |  |  |
| --- | --- | --- | --- |
| **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 | | |



# 6 Appendix

## 5.1 Assumptions and dependencies

## 5.2 Acronyms and abbreviations

## 5.3 Glossary