IPARK

Software Requirements Analysis and Design Assignment
COMP 3059 – Capstone Project I
Version 1.0

Capstone Project I - Team #10

04 November 2023

Instructor: Anjana Shah

Revision History

Date	Description	Author	Comments
04/11/2023	Version 1.0	Mohammadali Talaei Negin Heidari Sheida Moazeni Mahshad Eilanlou Mahyar Ghasemi Khah	First Revision

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
MT	Mohammadali Talaei	Technical Leads or Architects	04/11/2023
MG	Mahyar Ghasemi Khah	Technical Leads or Architects	04/11/2023
NH	Negin Heidari	Operations/DevOps Lead	04/11/2023
SM	Sheida Moazeni	Business Analysts	04/11/2023
ME	Mahshad Eilanlou	Quality Assurance (QA) Engineer	04/11/2023
	Anjana Shah	Instructor, Capstone Project 1	

Table of Contents

REVISION HISTORY ..... **II**

DOCUMENT APPROVAL ..... **II**

1. INTRODUCTION **4**

1.1 Purpose 4
1.2 Scope 4
1.3 Definitions, Acronyms, and Abbreviations 6
1.4 References 7
1.5 Overview 7

2. GENERAL DESCRIPTION **9**

2.1 Product Perspective 9
2.2 Product Functions 10
2.3 User Characteristics 11
2.4 General Constraints 11
2.5 Assumptions and Dependencies 12

3. SPECIFIC REQUIREMENTS **13**

3.1 External Interface Requirements 13
3.1.1 User Interfaces *13*
3.1.2 Hardware Interfaces *13*
3.1.3 Software Interfaces *13*
3.1.4 Communications Interfaces *13*
3.2 Functional Requirements 14
3.2.1 User Account Management *14*
3.2.2 Parking Space Reservation *14*
3.3 Use Cases 15
3.3.1 Use Case #1 ..... *15*
3.3.2 Use Case #37 ..... *20*
3.4 Classes / Objects 20
3.4.1 ParkingLot *20*
3.4.2 Reservation *21*
3.4.3 User *22*
3.4.4 ParkingSpace *23*
3.5 Non-Functional Requirements 24
3.5.1 Performance *24*
3.5.2 Reliability *24*
3.5.3 Availability *24*
3.5.4 Security *24*
3.5.5 Maintainability *24*
3.5.6 Portability *24*
3.6 Inverse Requirements 24
3.7 Design Constraints 25
3.8 Logical Database Requirements 26
3.9 Other Requirements 27

4. ANALYSIS MODELS **28**

4.1 Sequence Diagrams 28
4.3 Data Flow Diagrams (DFD) 29
4.2 State-Transition Diagrams (STD) 29

5. CHANGE MANAGEMENT PROCESS **30**

1. Introduction

Welcome to the Software Requirements Specification (SRS) for IPark, the innovative application poised to revolutionize the parking industry. This document aims to meticulously capture and outline the functional and nonfunctional requirements of IPark, a platform designed to streamline parking management and enhance user experience by leveraging cutting-edge technology. The IPark application stands at the forefront of addressing the challenge of expensive and bulky hardware associated with conventional parking solutions, offering a digital, user-centric alternative. Herein, we present the blueprint that will guide the development, deployment, and maintenance of the IPark system, ensuring it meets the exacting demands of modern urban environments and delivers on its promise to simplify parking for drivers and operators alike.

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed overview of the IPark application, an intelligent parking solution aimed at reducing the reliance on physical hardware in traditional parking systems. This document outlines the functional and nonfunctional requirements, constraints, and user interfaces of the application. It serves as a formal agreement between the stakeholders and the development team and will be used as a reference point throughout the development lifecycle.

1.2 Scope

The IPark system comprises two primary software products: the IPark Mobile Application and the IPark Management Dashboard.

1.2.1. Software Products

IPark Mobile Application: A user-centric application designed to allow drivers to effortlessly locate, reserve, and pay for parking spaces through their mobile devices. This application includes parking availability information, navigation features, and secure payment processing.

IPark Management Dashboard: A back-end system that enables parking facility owners to manage parking spaces, analyze utilization data. This dashboard offers reporting tools, usage analytics, and administrative functionalities to streamline parking management.

1.2.2. Product Functions

The IPark Mobile Application will:

- Provide users with information about available parking spots.

- Enable reservation and advance payment for parking spaces.
- Offer navigation assistance integration to the reserved parking spot.
- NOT provide ride-sharing or car-rental services.

The IPark Management Dashboard will:

- Allow administrators to manage parking inventory.
- Generate reports and usage analytics.
- NOT manage non-parking related venues or events.

1.2.3. Application of the Software

Benefits, Objectives, and Goals:

- The primary goal of the IPark system is to optimize urban parking spaces by providing parameter-driven information and user-definable reservation capabilities with immediate transaction processing.
- An objective is to reduce traffic congestion by enabling quicker parking solutions, aiming for reduction in average parking time.
- A significant benefit is the reduction in environmental impact due to decreased idle time while searching for parking, aligning with green city initiatives.

Consistency with Higher-level Specifications:

- The outlined scope for the IPark system is in harmony with the vision articulated in the System Requirements Specification (SRS). It adheres to the overarching strategy to minimize hardware dependency and maximizes digital integration, aligning with modern smart city frameworks.
- The IPark system will support the goals of stakeholders by providing an adaptable, efficient, and user-friendly parking solution. It focuses on leveraging technology to reduce operational costs and improve the end-user experience.
- The scope of the IPark system is to deliver a seamless parking solution that meets the needs of today's urban environments. By integrating intelligent software with existing parking infrastructures, IPark aims to reduce the footprint of traditional parking systems and introduce efficiency and convenience for users and operators.

1.3 Definitions, Acronyms, and Abbreviations

Definitions

IPark Platform	Intelligent parking, the smart parking management system aimed at increasing profits for parking space owners by streamlining operations and enhancing user experience.
24-hour Advanced Reservation	Feature allowing users to book parking spots up to 24 hours in advance via the IPark application.
Offline Mode	Functionality within IPark enabling users to access essential information, such as receipt of payment, even in limited or no internet connectivity scenarios.
Robust Notification System	System within IPark providing timely notifications for reservations, check-ins, check-outs.

Acronyms, and Abbreviations

SRS	Software Requirements Specification
UI	User Interface
UX	User Experience
DB	Database

1.4 References

Title	Date	Organization	Source
Software Requirements Specification (SRS)	N/A	SLCC (Salt Lake Community College)	https://slcc.pressbooks.pub/technicalwritingatslcc/chapter/software-requirements-specification-srs/
Your Guide to Writing a Software Requirements Specification (SRS) Document	N/A	Relevant Software	https://relevant.software/blog/software-requirements-specification-srs-document/
Software Requirement Specification (SRS) Format	N/A	GeeksforGeeks	https://www.geeksforgeeks.org/software-requirement-specification-srs-format/

1.5 Overview

SRS for the IPark system is organized into several sections, each serving a specific purpose in outlining the software requirements and functionalities. This subsection provides a brief overview of what the rest of the SRS contains and how it is organized.

1.5.1. What the rest of the SRS contains:

The SRS document is structured into several sections, each of which serves a specific purpose in outlining the software requirements and functionalities of the IPark project.

These sections include:

1.5.1.1. Project Summary: This section provides a brief but comprehensive overview of the IPark project, its core mission, and key features.

1.5.1.2. Data visualization Information: Here, you will find detailed information on the advanced data visualization capabilities of IPark, which empower parking space owners with insights into peak occupancy times, revenue tracking, and strategic intelligence.

1.5.1.3. Feedback and Rating Systems: This section explains the feedback mechanisms incorporated into IPark, allowing users to rate their experiences and detailing how this feedback system aids in improving the service quality.

1.5.1.4. 24-Hour Advanced Reservation: This section outlines the 24-hour advanced reservation system, enhancing user convenience and planning.

1.5.1.5. Offline Mode: Details about the offline mode, which ensures users have access to essential information even with limited or no internet connectivity, are provided in this section.

1.5.1.6. Convenient Payment Method: This section elaborates on the convenient payment method within IPark, which eliminates the need for cash payments and enhances the efficiency and security of the parking process.

1.5.1.7. Reservation Management: Here, you can find information on how IPark allows users to modify or cancel their bookings, with specific limitations and a separate reservation management system.

1.5.1.8. Application-based Check-In Check-Out System: This section highlights how IPark simplifies the parking process through a mobile application, reducing costs associated with traditional hardware systems.

1.5.1.9. Notification System: The final section outlines the robust notification system in IPark, including reminders and prompts that ensure efficient space utilization and a smooth parking experience for users.

1.5.2. Explain how the SRS is organized:

The SRS is organized in a logical sequence that ensures clarity and ease of reference. It begins with the "Project Summary," which provides a high-level overview of the IPark project, and then progresses through the various functional components of the system, delving into their specifics and features.

The structure of the SRS is as follows:

1.5.2.1. Project Summary: An introductory section providing a concise overview of the IPark project.

1.5.2.2. Data visualization Information: Detailed information on the data visualization capabilities of IPark.

1.5.2.3. Feedback and Rating Systems: Explanation of the user feedback and rating system.

1.5.2.4. 24-Hour Advanced Reservation: Information about the reservation system and its benefits.

1.5.2.5. Offline Mode: Details regarding the offline functionality of IPark.

1.5.2.6. Convenient Payment Method: Information about the payment method used within IPark.

1.5.2.7. Reservation Management: Information about the reservation system and its management.

1.5.2.8. Application-based Check-In Check-Out System: Explanation of the application-based check-in and check-out process.

1.5.2.9. Notification System: Information about the notification system and its various functions.

2. General Description

This section provides an overview of the general factors that influence the IPark project and its associated requirements. It serves to contextualize the specific requirements outlined in the following sections, making them more understandable within the broader project context.
The IPark project aims to revolutionize the parking sector by introducing a smart management system that enhances convenience and efficiency for both parking space owners and users. The primary goal of this system is to optimize urban parking spaces by providing parameter-driven, information and user-definable reservation capabilities with convenient transaction processing.

2.1 Product Perspective

2.1.1. Integration in Existing Ecosystems

- IPark is designed to seamlessly integrate into existing parking infrastructures without the need for additional hardware. This approach positions IPark as a flexible and cost-effective alternative to traditional parking systems, which often require significant investment in sensors, cameras, and gate mechanisms.

2.1.2. Competitive Landscape

- Compared to conventional parking solutions that rely heavily on physical hardware, IPark sets itself apart by utilizing a software-first approach, offering an innovative solution to users and operators. While other apps in the market may offer similar digital conveniences, IPark's minimal reliance on hardware and its sophisticated software capabilities, such as manual license plate entry, provide a unique market proposition.

2.1.3. User-Centric Design

- The application is developed with a strong focus on user experience, aiming to reduce the time and effort required to find and pay for parking. This user-centric design philosophy ensures that IPark addresses the pain points of modern drivers, positioning it as a more attractive option for users who are accustomed to digital services in other areas of their lives.

2.1.4. Scalability and Flexibility

- IPark is built to be scalable, catering to various sizes and types of parking facilities, from small private lots to large municipal parking structures. Its software-based nature allows for rapid updates and feature enhancements without the need to alter physical infrastructure.

2.1.5. Environmental Impact

- By reducing the need for paper tickets and streamlining the parking process, IPark contributes to environmental sustainability efforts. It also helps reduce traffic congestion by minimizing the time drivers spend searching for parking, thereby lowering vehicle emissions.

2.1.6. Future Developments

- IPark is poised for future integration with smart city initiatives and can be adapted to support autonomous vehicles and other advancements in transportation technology. Its flexible framework is designed to accommodate evolving user needs and technological trends.

2.2 Product Functions

2.2.1. Spot Parking Availability

Functionality: The app uses a software-based management system where parking operators update the status of parking spots.

2.2.2. Digital Payment System

Functionality: Integrates with payment gateways to allow secure transactions. Users can pay with credit cards, e-wallets, or app-specific credits.

2.2.3. Reservation System

Functionality: Users select a parking spot and reserve it for a specific time slot. The system updates availability and confirms reservations instantly.

2.2.4. Automatic Billing

Functionality: The app automatically calculates the cost based on the parked time and processes the payment when the car leaves the spot. It sends digital receipts post-payment.

2.2.5. User Account Management

Functionality: Allows users to sign up, log in, and manage personal data, vehicles, payment methods, and view parking history.

2.2.6. Reporting and Analytics
Functionality: Collects data on parking lot usage and financial transactions, presenting it in an accessible format for operators.

2.2.7. Notifications and Alerts
Functionality: Sends out timely push notifications or SMS to inform users about their current parking status and other relevant updates.

2.2.8. Feedback System
Functionality: Users can rate their parking experience and provide comments, which are aggregated for quality control and service enhancement.

2.2.8. Check-In/Check-Out
Functionality: Users manually enter the spot number and choose their car.

2.3 User Characteristics

This subsection of the SRS outlines the general characteristics of the users of the IPark system, which include:

2.3.1. End Users: These are visitors who access the platform to find and use parking facilities. They may have varying levels of technological proficiency and diverse parking needs, such as short-term or long-term parking.

2.3.2. Parking Owners: Parking owners are the primary stakeholders in the parking management system. They are individuals or organizations that own and operate parking facilities, such as parking lots.

2.3.3. IPark Team Members: The IPark team members are the developers and individuals responsible for creating, maintaining, and operating the parking management system. They include software engineers, project managers, quality assurance testers, and other technical and non-technical staff involved in system development and support.

2.4 General Constraints

This subsection of the SRS provides a broad overview of the limitations and restrictions that will shape the design of the IPark system. These constraints include:

- Academic Project Limitations: As this project is a capstone project, there are specific guidelines, requirements, and constraints imposed by the academic

institution. These may include deadlines, project scope, and approval processes that need to be confirmed with the project supervisor or professor. These academic constraints will influence the development timeline and scope of the IPark system.

- Legal and Government Regulations: To ensure compliance with legal and governmental regulations, the IPark system must adhere to rules and laws that govern aspects like data privacy, payment processing, and public safety. These regulations may impose limitations on data handling, payment methods, and accessibility, affecting the system's design and operation.

2.5 Assumptions and Dependencies

The development and functionality of the IPark system rely on a set of assumptions and dependencies which are critical for its successful implementation. These are not constraints but factors that could affect the requirements outlined in the Software Requirements Specification (SRS).

2.5.1. Assumptions:

- Smartphone Penetration: It is assumed that the majority of the end-users will have access to smartphones capable of running the IPark Mobile Application.
- Network Connectivity: Constant and consistent internet connectivity is assumed to be available in urban areas where the IPark system will be deployed, facilitating data communication.
- Payment Gateway Integration: The availability of third-party payment gateway services for secure transaction processing is assumed.
- GPS Reliability: The accuracy of GPS services, essential for the navigation and location-based features of the app, is assumed to be high in the target urban areas.
- Stakeholder Collaboration: The willingness of parking facility owners and city authorities to adopt the IPark Management Dashboard and integrate it with their current systems is assumed.

2.5.2. Dependencies:

- Operating System Availability: The IPark Mobile Application's development is dependent on the availability of Android and iOS operating systems for mobile devices.
- Regulatory Compliance: Compliance with local and national regulations concerning digital payments, user privacy, and data security is a dependency that could affect requirements.
- Third-party Services: The system's functionality depends on the reliability and availability of third-party services, such as mapping services for navigation and financial services for payment processing.

- Technological Advances: The IPark system's requirements may change in response to advancements in related technologies, such as enhanced GPS precision, augmented reality for navigation, or the advent of smart cars.

3. Specific Requirements (IMPORTANT)

This section details the functional, performance, interface, and other system requirements that will guide the design, implementation, and testing phases of the IPark project. Each requirement is presented to be correct, traceable, unambiguous, verifiable, prioritized, complete, consistent, and uniquely identifiable.

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.1.1 Mobile Application Interface

The mobile application shall provide intuitive navigation for reserving parking spots, managing user accounts, and processing payments.

3.1.1.2 Web Interface

The web platform shall offer the same functionality as the mobile application and shall be accessible on all major browsers.

3.1.2 Hardware Interfaces

We intentionally removed the Hardware Interfaces from our project to prioritize cost-efficiency. Our project's primary advantage is to eliminate the additional expenses associated with hardware, enabling parking owners to maximize their revenue. Instead of incurring costs for hardware kiosks and maintenance, our solution offers a more cost-effective and streamlined approach.

3.1.3 Software Interfaces

3.1.3.1 Payment Gateway Integration

The system shall support integration with multiple payment gateways to process transactions securely. It shall interface with external payment services via secure API calls.

3.1.3.2 Mapping Service Integration

The system shall integrate with external mapping services to display parking locations and availability information.

3.1.4 Communications Interfaces

3.1.4.1 Data Exchange Protocols

The system shall use HTTPS for secure data transfer.

The system shall use appropriate RESTful API standards for communication with third-party services.

3.1.4.2 Notification Services

The system shall send notifications to users regarding reservation statuses, depending on user preferences.

3.2 Functional Requirements

This section details the functional requirements that describe the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are detailed for each feature that the IPark system is to support.

3.2.1 User Account Management

3.2.1.1 Introduction

The system shall provide users with the capability to create, manage, and delete their accounts within the IPark platform.

3.2.1.2 Inputs

- User personal information (e.g., name, email address, password)
- Payment information for transactions

3.2.1.3 Processing

- Verification of email address and payment information
- Secure storage of user information
- Authentication of user credentials upon login

3.2.1.4 Outputs

- Confirmation of account creation, update, or deletion
- User profile view and edit capabilities

3.2.1.5 Error Handling

- Incorrect input data will prompt the user with an error message and the opportunity to correct it.
- The system will handle account creation conflicts (e.g., email already in use) by notifying the user and suggesting alternatives.

3.2.2 Parking Space Reservation

3.2.2.1 Introduction

The system shall allow users to reserve parking spaces in advance through the IPark platform, providing spot availability check.

3.2.2.2 Inputs

- User selection of desired parking location, date, and time
- User vehicle information if necessary

3.2.2.3 Processing

- Checking the availability of the parking space
- Holding the parking spot for a specified time during the reservation process
- Processing the payment for the reservation

3.2.2.4 Outputs

- Confirmation of the reservation with details (location, time slot, cost)
- Digital receipt of the transaction

3.2.2.5 Error Handling

- In case of unavailability, the system will provide alternative suggestions.
- Payment failures will result in an error message with options to retry or change the payment method.

…

3.3 Use Cases

3.3.1 User Registration

Actor: New User
Description: Allows a new user to create an account on the IPark platform, providing them access to the application's features.

3.3.2 User Login

Actor: Registered User
Description: Enables a registered user to access their account by entering credentials, which the system validates.

3.3.3 Search for Parking spot

Actor: Registered User
Description: Permits a user to search for available parking spots based on location, time, and other preferences.

3.3.4 Reserve a Parking Spot

Actor: Registered User
Description: Allows a user to reserve an available parking spot and receive a reservation confirmation.

3.3.5 Cancel Reservation

Actor: Registered User

Description: Provides the user with an option to cancel a previously made reservation.

3.3.6 Modify Reservation

Actor: Registered User
Description: Enables a user to make changes to an existing reservation's details such as time or date.

3.3.7 Rate and Review

Actor: Registered User
Description: Allows users to rate and review a parking spot or experience, contributing to community feedback.

3.3.8 View Reservation Details

Actor: Registered User
Description: Users can view details of their active or upcoming reservations.

3.3.9 View Reservation History

Actor: Registered User
Description: Allows users to view their past parking spot reservations.

3.3.10 Payment

Actor: Registered User
Description: Facilitates users to make payments for their parking reservations using integrated payment methods.

3.3.11 View Parking Spot Availability

Actor: Registered User
Description: Users can view availability information of parking spots before making a reservation.

3.3.12 Notification Reminder

Actor: Registered User
Description: The system sends reminder notifications to users about their upcoming reservations.

3.3.13 Feedback Submission

Actor: Registered User
Description: Allows users to submit feedback about the app, which can be used to improve the service.

3.3.14 Receive Confirmation

Actor: Registered User
Description: After completing actions like reservations or payments, users receive confirmations.

3.3.15 MFA

Actor: Registered User

Description: Provides an additional security layer for user login through multi-factor authentication.

3.3.16 Check-in

Actor: Registered User
Description: Allows users to check in upon arrival at their reserved parking spot.

3.3.17 Check-out

Actor: Registered User
Description: Enables users to check out from a parking spot, finalizing the parking session.

3.3.18 Integration with Navigation Apps

Actor: Registered User
Description: The system integrates with navigation apps to guide users to their reserved parking spot.

3.3.19 User verification

Actor: Registered User

Description: The system verifies user identity for certain transactions or actions, adding a layer of security.

3.3.20 Spot Owner Registration

Actor: Parking Owner

Description: The parking owner wants to register as a spot owner on the platform. They provide required information, including their personal details, parking lot details, and verification documents. Once registered, the owner can list their parking lot and start managing it.

3.3.21 Add Parking Lot Listing

Actor: Parking Owner

Description: After successful registration, the owner can add a parking lot listing. They provide detailed information about the parking lot, including its location, capacity, pricing, reservation policies, security features, and high-quality photos. The listing is made available to potential customers.

3.3.22 Remove Parking Lot Listing

Actor: Parking Owner

Description: The owner wants to remove a parking lot listing. This might be necessary if the parking lot is no longer available for reservations, temporarily closed for maintenance, or if the owner decides to take it off the platform.

3.3.23 Owner Management

Actor: Parking Owner

Description: The owner can manage their account and parking lot listings. This includes updating personal information, modifying parking lot details, and managing access for employees or co-owners.

3.3.24 View Reservation History

Actor: Parking Owner

Description: The owner can access a reservation history log, which provides a detailed record of all past reservations, including user information, reservation dates and times, and payment details.

3.3.25 Pricing Management

Actor: Parking Owner

Description: The owner can manage pricing for their parking lot. This includes setting standard rates, creating special pricing for events, and adjusting rates during peak hours to optimize revenue.

3.3.27 Revenue Tracking

Actor: Parking Owner

Description: The owner can track revenue generated by their parking lot. They have access to reports, occupancy rates, and financial performance over specific time periods.

3.3.31 Log in to admin dashboard

Actor: Administrator

Description: The administrator logs in to the admin dashboard using their credentials. After successful login, they gain access to the admin interface with privileges to perform various administrative tasks.

3.3.32 View owner's listings

Actor: Administrator

Description: The administrator has the ability to view all parking lot listings created by parking owners. They can access detailed information about each listing, including location, capacity, pricing, and reservation policies. This helps them monitor the listings and ensure they comply with platform guidelines.

3.3.33 Error monitoring

Actor: Administrator

Description: The administrator monitors the system for errors and issues. They can access error logs and reports, which provide details about any system errors, user-reported issues, or technical glitches. The administrator takes necessary actions to address and resolve these errors to maintain the system's functionality and reliability.

3.3.34 Send Notifications

Actor: System

Description: The system is responsible for sending various types of notifications to users and parking owners. This includes reservation reminders, check-in prompts, reservation confirmations, and other important messages to ensure a smooth parking experience.

3.3.35 Payment receipt generator

Actor: System

Description: The system generates payment receipts for users after successful payments for parking reservations. The payment receipt includes details such as the amount paid, transaction ID, reservation details, and a timestamp. The receipt is sent to the user's email or made available for download.

3.3.36 Generate Reports For Owner

Actor: System

Description: The system generates reports for parking owners, providing insights into their parking lot's performance. These reports include information on occupancy, revenue, and other key metrics. Owners can access these reports to make data-driven decisions and optimize their operations.

3.3.37 Save credit card information

Actor: System

Description: The system securely stores and manages users' credit card information for the purpose of processing payments for parking reservations. The system ensures that this sensitive data is encrypted and compliant with security standards to protect user privacy.

…

3.4 Classes / Objects

3.4.1 ParkingLot

3.4.1.1 Attributes:

- `ParkingID`: The ID of the parking lot.
- `Capacity`: The total number of parking spaces available in the lot.
- `ReservationAreaLots`: The total number of reservation parking spaces.
- `CurrentOccupancy`: The number of parking spaces currently occupied.
- `PricingModel`: The pricing strategy for the parking lot (e.g., hourly, daily).
- `ReservationPolicy`: The policy for accepting and managing reservations.
- `SecurityLevel`: The level of security measures implemented in the parking lot.

3.4.1.2 Functions:

- `SetPricing(model)`: Enables the owner to configure the pricing model for the parking lot, specifying rates and payment options.
- `ViewOccupancyReport()`: Generates a report on current occupancy and usage statistics, offering insights into peak hours and popular spaces.
- `ViewFinancialReport(startDate, endDate)`: Generates financial reports for a specified time period, showing revenue and income.

Reference to functional requirements and/or use cases:

- Functional Requirement FR4: "Parking Lot Management" requires the "ParkingLot" class to manage occupancy and pricing, particularly through the "ManageOccupancy" and "SetPricing" functions.
- Use Case UC4: "View Occupancy Report" utilizes the "ViewOccupancyReport" function to provide occupancy data to parking owners.
- Use Case UC5: "Generate Financial Report" is fulfilled by the "ViewFinancialReport" function, helping owners assess the financial performance of their parking lots.

3.4.2 Reservation

3.4.2.1 Attributes:

- `ReservationID`: A unique identifier for each reservation.
- `ParkingSpaceID`: The ID of the reserved parking space.
- `UserID`: The ID of the user making the reservation.
- `ReservationTime`: The date and time of the reservation.
- `Status`: The status of the reservation (e.g., pending, confirmed).
- `PaymentStatus`: The payment status for the reservation, indicating whether the fee has been paid.

3.4.2.2 Functions:

- `CreateReservation(parkingSpaceID, userID, reservationTime)`: Allows the parking owner to create reservations, which may involve checking the availability of spaces and updating the reservation status.
- `ConfirmReservation(reservationID)`: Confirms a pending reservation and updates its status, ensuring that a reserved space is no longer available for others.
- `CancelReservation(reservationID)`: Enables the owner to cancel a reservation, making the space available for other users and potentially refunding fees.
- `ViewReservationDetails(reservationID)`: Retrieves detailed information about a specific reservation, offering insights into who made the reservation, its status, and payment information.
- `UpdatePaymentStatus(reservationID, paymentStatus)`: Updates the payment status of a reservation, indicating whether the user has completed the payment process.

Reference to functional requirements and/or use cases:

- Functional Requirement FR6: "Reservation Management" relies on the "Reservation" class for creating, confirming, and canceling reservations, which is achieved through the "CreateReservation," "ConfirmReservation," and "CancelReservation" functions.
- Functional Requirement FR7: "Payment Processing" uses the "UpdatePaymentStatus" function to manage payment-related aspects of reservations.
- Use Case UC6: "Create Reservation" is related to the "CreateReservation" function, allowing parking owners to facilitate reservations for their lots.
- Use Case UC7: "View Reservation Details" involves the "ViewReservationDetails" function to provide owners with detailed information about reservations.

3.4.3 User

3.4.3.1 Attributes:

- `UserID`: A unique identifier for each user.
- `Username`: The username chosen by the user for authentication.
- `Password`: The user's hashed password for secure authentication.
- `Profile`: User's profile information, including contact details and preferences.
- `ReservationHistory`: A list of past parking reservations made by the user.

3.4.3.2 Functions:

- `Authenticate(username, password)`: Authenticates the user by comparing the provided username and password with the stored credentials, granting access to the application.
- `UpdateProfile(profileInfo)`: Allows the user to update their profile information, including contact details and preferences.
- `ViewReservationHistory()`: Retrieves the user's reservation history, showing past bookings and related information.
- `MakeReservation(parkingSpaceID, reservationTime)`: Enables the user to make parking reservations, including checking space availability, selecting a time, and confirming the reservation.

Reference to functional requirements and/or use cases:
- Functional Requirement FR8: "User Authentication" is fulfilled by the "User" class's "Authenticate" function, allowing users to securely access their accounts.
- Functional Requirement FR9: "User Profile Management" is supported by the "UpdateProfile" function of the "User" class, enabling users to manage their profile information.
- Use Case UC8: "User Registration" involves the creation of a new "User" object, and the "Authenticate" function is utilized for user login.
- Use Case UC9: "User Login" is reliant on the "User" object for authentication, and "MakeReservation" allows users to create reservations.

3.4.4 ParkingSpace

3.4.4.1 Attributes:

- `SpaceID`: A unique identifier for each parking space.
- `ParkingID`: The ID of the parking lot that this space belongs to (foreign key).
- `Location`: The geographical location of the parking space.

- `Availability`: A boolean attribute indicating whether the parking space is available or occupied.
- `Rate`: The parking fee rate for this space.
- `Size`: The size or type of the parking space (e.g., compact, standard, disabled).
- `ReservationStatus`: Indicates if the parking space is reserved and for whom (if applicable).
- `Occupant`: The user who currently occupies the parking space (if occupied).

3.4.4.2 Functions:

- `ManageOccupancy(spaceID, action)`: Allows the parking owner to update the occupancy status of individual parking spaces, indicating if they are occupied or available.
- `ReserveSpace(userID)`: Allows a user to reserve the parking space, marking it as reserved and associating it with the user's ID.
- `ReleaseSpace()`: Marks the parking space as available when the user checks out, resetting its availability status.
- `CalculateFee(duration)`: Calculates the parking fee based on the duration of use, taking into account the parking rate.
- `UpdateAvailability(status)`: Updates the availability status of the parking space, indicating whether it is occupied or available.
- `GetLocation()`: Returns the geographical location of the parking space, helping users find their reserved spots.
- `GetSize()`: Returns the size/type of the parking space
, allowing users to select spaces that match their vehicle size.
- `IsOccupied()`: Checks if the parking space is currently occupied, helping users and parking owners monitor space usage.

Reference to functional requirements and/or use cases:

- Functional Requirement FR10: "Parking Reservation" involves the use of the "ParkingSpace" class to reserve a parking space, release it, and calculate fees.
- Use Case UC10: "User Check-Out" is dependent on the "ParkingSpace" class to release the parking space and calculate the fee, and "ReserveSpace" is utilized for making reservations.

3.5 Non-Functional Requirements

The following non-functional requirements define the system attributes such as performance, reliability, and other key metrics that the IPark application should adhere to.

3.5.1 Performance

- The system shall handle at least 1,000 simultaneous user connections.
- The response time for searching and displaying available parking spots shall not exceed 2 seconds 95% of the time under normal conditions.

- The transaction processing for reservations and payments shall complete within 3 seconds for 90% of the transactions.

3.5.2 Reliability

- The system shall achieve a mean time between failures (MTBF) of more than 10,000 hours.
- The system should ensure transaction integrity and provide a rollback mechanism in case of process failure.

3.5.3 Availability

- The IPark platform shall be available for use 24 hours a day, 7 days a week.
- The maximum allowable system downtime shall not exceed 2 hours per month.

3.5.4 Security

- All user data must be encrypted using industry-standard encryption protocols during transmission and at rest.
- The system shall implement multi-factor authentication to ensure secure user access.
- Regular security audits should be conducted every quarter to ensure ongoing system integrity.

3.5.5 Maintainability

- The system should be designed to allow updates and maintenance without more than 30 minutes of downtime per update.
- All code shall be documented to facilitate maintenance and future updates.

3.5.6 Portability

- The IPark application should be compatible with the two latest major versions of iOS and Android operating systems.
- The web version of the system must be compatible with the latest versions of Chrome, Firefox, Safari, and Edge browsers.

3.6 Inverse Requirements

- The IPark platform is not required to support payment methods outside of credit/debit cards, and approved digital wallet services.
- The system is not intended to manage or track parking availability for non-commercial or non-registered residential parking spaces.
- IPark does not provide insurance for damages or theft that occur in parking spots booked through the application.
- The application is not required to offer offline functionality;The application shall provide offline support for specific scenarios, such as downloading receipts and booking confirmations. However, the application requires an active internet connection to access

and perform other functions, including but not limited to real-time data synchronization, user authentication, and feature updates.
- IPark will not enforce parking regulations or issue penalties for parking violations; such enforcement is outside the system's scope and is the responsibility of the respective parking authority or property owner.
- The system does not include the functionality for users to offer their private parking spots for rent through the IPark platform.

3.7 Design Constraints

Compliance with Industry Standards:
- The application must adhere to data security standards such as PCI DSS for payment processing.
- Must follow accessibility guidelines (e.g., WCAG) to ensure the app is usable by all customers, including those with disabilities.

2. Company Policies:
- IPark must align with the company's privacy policy for handling and storing user data.
- The application's design and data handling practices must comply with the company's ethical guidelines.

3. Technological Limitations:
- The app should be optimized for performance across all supported devices, considering variations in operating systems, screen sizes, and processing power.
- Given the reliance on manual input for vehicle identification, the app must have robust error-checking to prevent mistakes in license plate entries.

4. Hardware Limitations:
- As the system will not use physical hardware for spot detection or vehicle recognition, the app must be designed to function effectively with manual input and updates from parking attendants or operators.

5. Legal and Regulatory Constraints:
- Must comply with local privacy laws in handling personal and payment information.
- The application will need to follow any local government regulations concerning parking management and digital transactions.

6. Network and Connectivity Requirements:
- The application must be able to operate in areas with limited connectivity, which may necessitate offline capabilities or low-bandwidth operation modes.

7. User Interface Design Constraints:
- The interface should be intuitive and not require extensive technical knowledge, appealing to a broad user base.
- Multilingual support may be required to cater to diverse user demographics.

8. Scalability Constraints:
- The system architecture must be designed to handle scaling, both in terms of growing user numbers and geographic expansion.

9. Environmental Considerations:

- The design process should consider the environmental impact, promoting digital over physical solutions where possible to minimize waste.

10. Maintenance and Support:
    - The application should be designed for easy maintenance and updates, with a clear plan for ongoing technical support.

3.8 Logical Database Requirements

3.8.1. Data Formats:
- The database must support various data formats including strings for text information, integers for numerical values, timestamps for tracking parking duration, and geolocation data types for parking locations.

3.8.2. Data Retention:
- The application will need policies for data retention in compliance with legal standards, ensuring data is kept only as long as necessary and purged appropriately.
- Backup and recovery procedures must be in place to prevent data loss.

3.8.3. Data Integrity:
- Implement mechanisms to maintain data accuracy and consistency throughout its lifecycle. This includes constraints, validation rules, and referential integrity.
- Data duplication should be minimized to prevent inconsistencies.

3.8.4. Data Security:
- The database must have robust security measures including encryption, access controls, and secure connections to protect sensitive user data.
- Comply with relevant data protection regulations to safeguard personal and payment information.

3.8.5. Data Accessibility:
- Provide interfaces for the application to access and manipulate the data efficiently.
- The database should support concurrent access by multiple users while maintaining data consistency.

3.8.6. Reporting and Analytics:
- The database should be structured to support complex queries for analytics and reporting purposes without impacting performance.
- Support for aggregate functions and the ability to join multiple tables will be necessary for generating insights and reports.

3.8.7. Transaction Management:
- The database must support transactional integrity, ensuring that all payment and reservation transactions are processed reliably and without error.
- Implement an atomic, consistent, isolated, and durable (ACID) transaction model to ensure reliability in the event of system failures.

3.8.8. Data Archiving:
- Establish a system for archiving old data that is no longer needed for immediate access but must be retained for historical analysis or regulatory compliance.

3.8.9. Data Migration:

- Design the database to allow for easy data migration in case of system upgrades or changes in database technologies.

3.9 Other Requirements

3.9.1. Security Requirements

- Access Control: Define user roles and permissions to access different parts of the system.
- Data Encryption: Ensure sensitive data, such as payment information and user details, are encrypted to prevent unauthorized access.
- Security Protocols: Implement measures to protect against cyber threats, ensuring the system's security against hacking attempts, data breaches, and other security risks.

3.9.2. UI and UX Requirements

- Intuitive UI/UX: Develop an easy-to-navigate and user-friendly interface for both parking owners and drivers.
- Responsive Design: Ensure the application is compatible across various devices and screen sizes for a seamless user experience.
- Accessibility: Ensure the application is accessible to users with disabilities, adhering to accessibility standards.

3.9.3. Integration and Compatibility Requirements

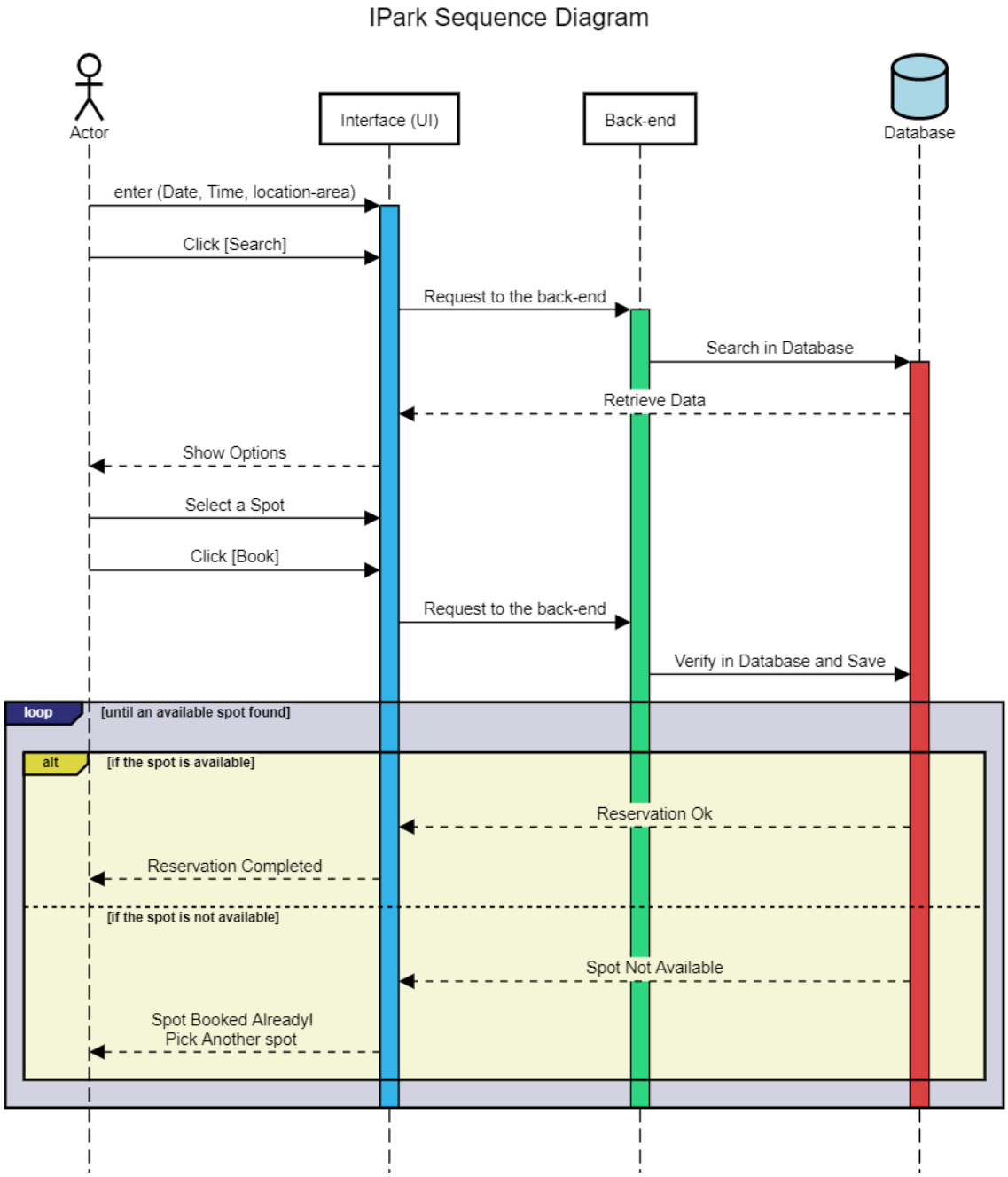
- Third-Party Integrations: Ensure compatibility and smooth integration with external services, such as payment gateways and mapping/navigation services.
- Mobile Device Compatibility: Ensure the application functions well across different mobile devices and operating systems.
- API Development: Provide APIs for potential integration with other systems and applications.

3.9.4. Notification System Requirements

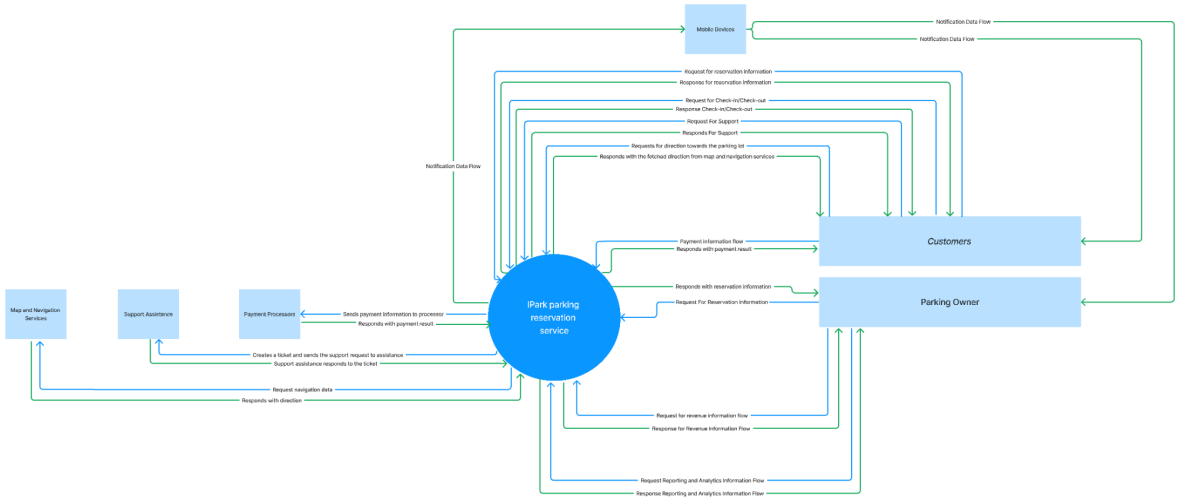
- Data Protection and Privacy: Comply with data protection regulations, ensuring user data privacy and security.
- Local Regulations: Adhere to local parking regulations and laws where the service operates.

4. Analysis Models

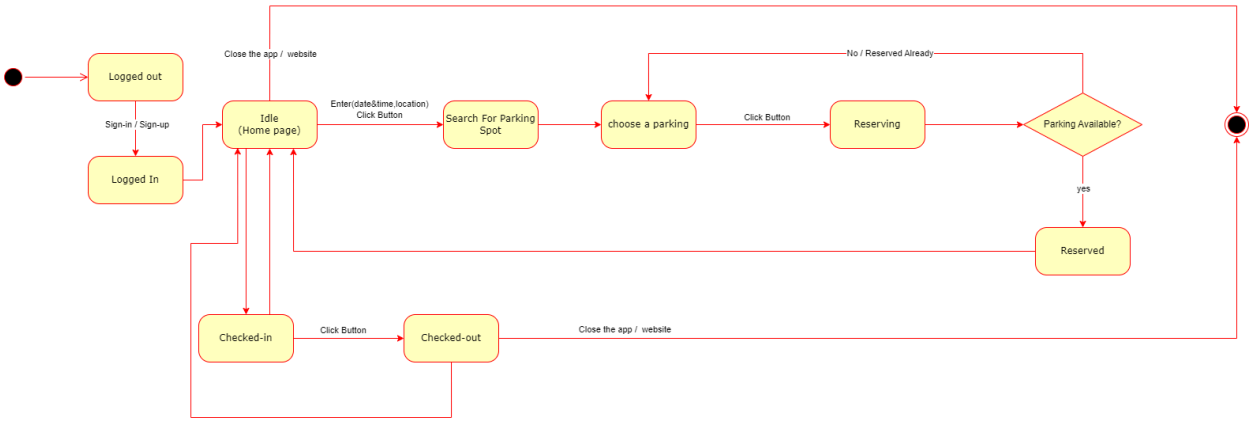
4.1 Sequence Diagrams



4.3 Data Flow Diagrams (DFD)



4.2 State-Transition Diagrams (STD)



5. Change Management Process

5.1. Change Identification:

- Any stakeholder, including developers, or end-users, can identify the need for a change in the SRS.

5.2. Submission of Change Request:

- Stakeholders must submit change requests through a formal channel, such as a dedicated project management system or via email to the developers.
- Each change request should include a detailed description of the proposed change, the rationale behind it, and the expected impact on the project.

5.3. Initial Review:

- Developers conduct a preliminary review to assess the validity and necessity of the change.
- If the request is incomplete or unclear, it is sent back to the submitter for clarification.

5.4. Impact Analysis:

- A cross-functional team comprising developers evaluate the implications of the proposed change. This includes analyzing the impact on the project timeline, costs, resources, and technical feasibility.

5.5. Approval Process:

- The developers review the change request and either approves, rejects, or requests further analysis.

5.6. Documentation:

- Upon approval, the SRS document is updated to reflect the change. The update includes a revision history noting the change's nature, the date of the change, and the name of the person who authorized it.

5.7. Communication:

- All developers are informed of the approved changes, and the updated SRS is distributed to the project team.
- Relevant training or briefings are provided if the change affects existing processes or requires new ones.

5.8. Implementation:

- The development team implements the changes as per the updated SRS. The project plan and other project documentation are also updated to align with the new requirements.

5.9. Verification:

- developers conduct a verification process to ensure that the implemented changes meet the updated requirements as specified in the SRS.

5.10. Continuous Monitoring:

- The developers monitor the effects of the changes on project progress and performance. Continuous monitoring helps to identify any further adjustments that may be required.