Parking Lot System

Emmanuel Mendoza

5/4/2025

Jafrina Jabin

INFO-C451: System Implementation

Table of Contents

Contents

[Problem Statement 3](#_Toc193227279)

[System Requirements 4](#_Toc193227280)

[Functional Requirements 5](#_Toc193227281)

[User Interface Specifications 8](#_Toc193227282)

[Project Plan 19](#_Toc193227283)

# Problem Statement

In urban and high-traffic areas, parking has become a constant challenge for both drivers and parking lot operators. As vehicles increase in number, the demand for organized and efficient parking has grown. Currently, many parking lots rely on manual operations, leading to inefficiencies such as long wait times, difficulty in locating available spots, and challenges in payment processing. Additionally, security concerns and the lack of real-time parking availability contribute to customer dissatisfaction.

Our proposed Parking Lot Management System aims to address these challenges by providing an automated, user-friendly, and efficient solution for managing parking spaces. The system will streamline parking spot allocation, provide real-time availability updates, automate ticketing and payment processing, and ensure secure operations.

**Objectives of the System**

The primary objective of this system is to enhance parking efficiency, reduce operational costs, and improve customer satisfaction. Specifically, the system will:

* Optimize parking space utilization by dynamically updating availability in real-time.
* Reduce entry and exit congestion through automated ticket issuance and payment processing.
* Enhance security and monitoring by integrating with existing CCTV and security systems.
* Improve customer experience by providing a seamless and automated parking process.
* Support multiple payment methods (cash, credit/debit card, and digital payments).
* Enable parking lot operators to manage operations efficiently with an intuitive dashboard and reporting tools.

# System Requirements

The Parking Lot Management System will support the following functionalities:

* Real-time Parking Spot Availability Display
* Automated Entry Ticket Generation
* Payment Processing (Automated and Manual Options)
* Parking Spot Reservation System
* Overcapacity Handling & Notifications
* Security and Integration with CCTV Cameras
* User Role Management (Customers, Operators, Security Personnel, and Administrators)
* Comprehensive Reporting and Analytics for Operators

**Typical Customers**

The proposed system will cater to different types of users:

Parking Lot Managers: Oversee parking lot operations, monitor availability, and generate revenue reports.

Drivers/Vehicle Owners: Use the system to find parking, pay fees, and exit seamlessly.

Parking Assistants & Security Personnel: Help customers and ensure smooth parking operations.

Business Owners (Malls, Offices, Stadiums, etc.): Use the system to manage parking for their visitors and employees.

# Functional Requirements

|  |  |  |
| --- | --- | --- |
| No. | Priority Weight | Description |
| REQ-1 | High | The system should allow users (parking attendants, and administrators) to securely log in using their credentials (Username, Password, Two-Factor Authentication). |
| REQ-2 | High | Customers should be able to check real-time parking availability through the system before entering the lot. |
| REQ-3 | High | The system should issue a parking ticket automatically upon a vehicle’s entry. |
| REQ-4 | High | The system should track and display available parking spots categorized by type (compact, motorcycle, emergency). |
| REQ-5 | High | The system should prevent entry when the parking lot has reached full capacity and display a notification at the entrance. |
| REQ-6 | Medium | Customers should be able to pay for parking via an automated exit panel or through a parking attendant (in the event the system is down). |
| REQ-7 | Medium | The system should allow payments through multiple methods, including credit/debit cards, mobile payments, and cash. |
| REQ-8 | Medium | The system should generate a receipt after payment, either as a printed or digital receipt. |
| REQ-9 | Low | The system should generate and store reports on parking usage, revenue, and occupancy trends for administrators. |
| REQ-10 | High | The system should provide security personnel access to vehicle logs and entry/exit timestamps. |

Non-Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Category | Priority Weight | Description |
| NFR-1 | Security | High | The system should provide secure access control with role-based permissions for customers, attendants, and administrators. |
| NFR-2 | Usability | High | The system interface should be user-friendly and intuitive, allowing for seamless navigation. |
| NFR-3 | Usability | Medium | The system should be accessible via mobile devices to allow customers to check parking availability on the go. |
| NFR-4 | Reliability | High | The system should have a high up-time to ensure uninterrupted service. |
| NFR-5 | Reliability | High | Automated backups should be performed daily to prevent data loss. |
| NFR-6 | Performance | High | The system should process ticket issuance, payments, and access control within minimal response time. |
| NFR-7 | Performance | Medium | The system should support concurrent usage of up to 500 users without performance degradation. |
| NFR-8 | Supportability | Medium | The system should provide error logs and diagnostic reports for administrators to troubleshoot issues. |
| NFR-9 | Supportability | Medium | The system should be scalable to accommodate future parking expansions. |
| NFR-10 | Supportability | Low | The system should be compatible with third-party parking management integrations. |

User Interface Requirements

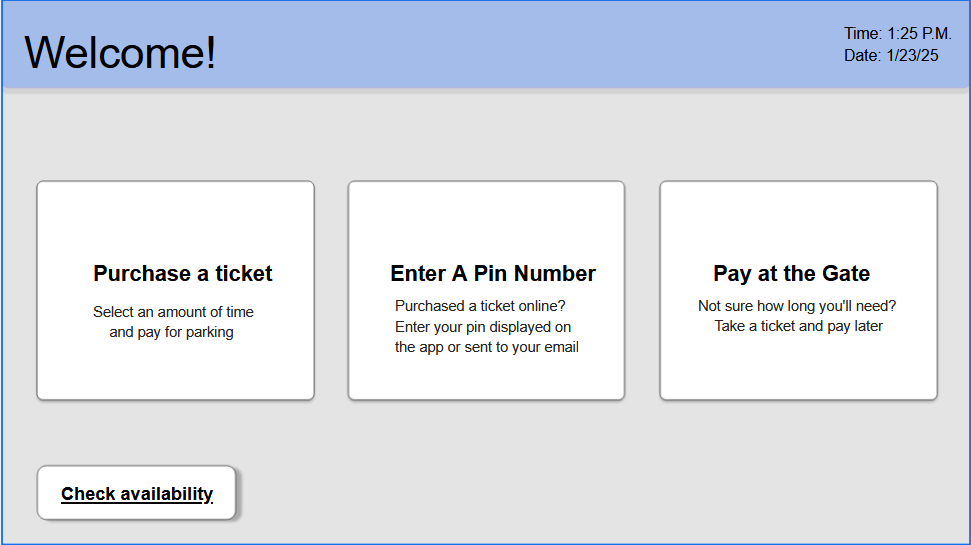
|  |  |  |
| --- | --- | --- |
| No. | Priority Weight | Description |
| UI-1 | High | The system should have a clear and intuitive dashboard displaying available parking spaces, ticket status, and payment options. |
| UI-2 | High | The login page should be simple with clear instructions for account recovery, login, and authentication. |
| UI-3 | High | The system should have a responsive design that works across different screen sizes, including desktops, tablets, and mobile devices. |
| UI-4 | High | Navigation menus should be clear and consistent across all pages, allowing users to easily access ticket status, payment options, and parking history. |
| UI-5 | Medium | The real-time parking availability page should display a map for customers to see their parking spot. |
| UI-6 | Medium | Notifications and alerts should inform users about the available parking spots, payment reminders, and entry/exit logs. |
| UI-7 | Medium | The system should display parking time and charges in a clear and readable format for customers. |
| UI-8 | Low | The color scheme and typography should be accessible, especially for visually impaired users. |
| UI-9 | Low | The system should allow users to customize notification preferences (email, SMS, app alerts). |
| UI-10 | Low | A help button should be implemented to allow users the option to speak with a parking assistant to aid with customer inquiries. |

# User Interface Specifications

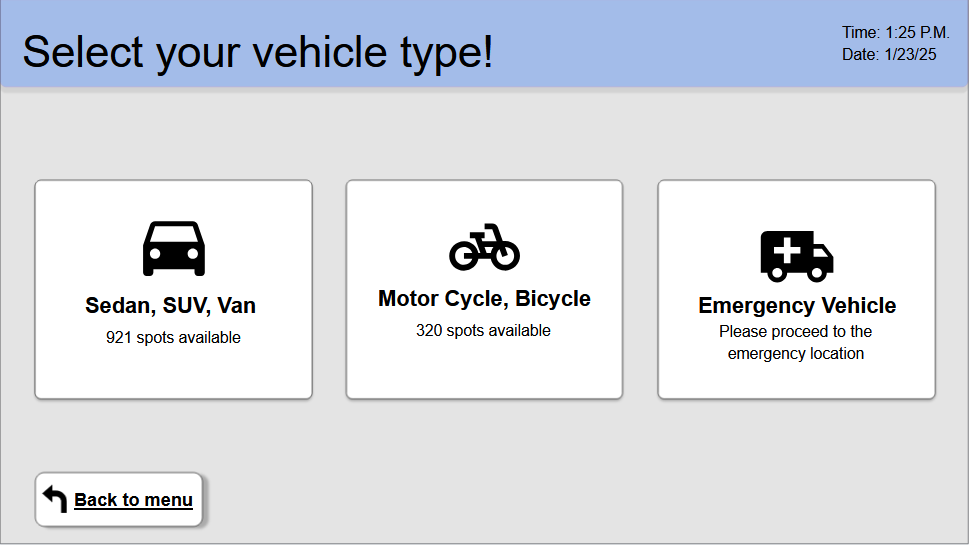
Preliminary Design

1. Ticket Issuance

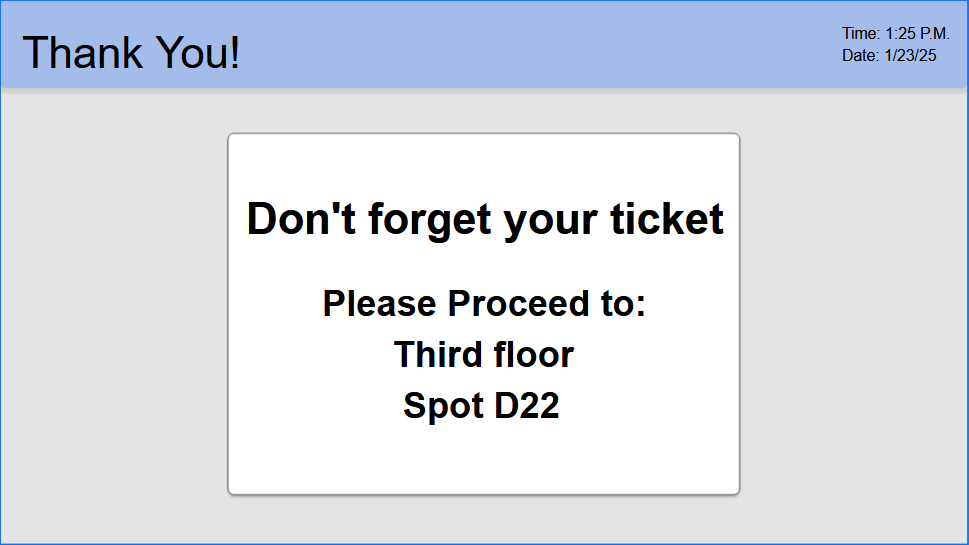
As the customer pulls into the parking lot, they will arrive near the terminal in which they will interact with the system to receive their parking ticket. There they can choose to take a ticket and be billed after exiting or to enter a pin if a ticket was purchased online or via app.



From here, the customer will select their vehicle type and the availability of parking spaces.

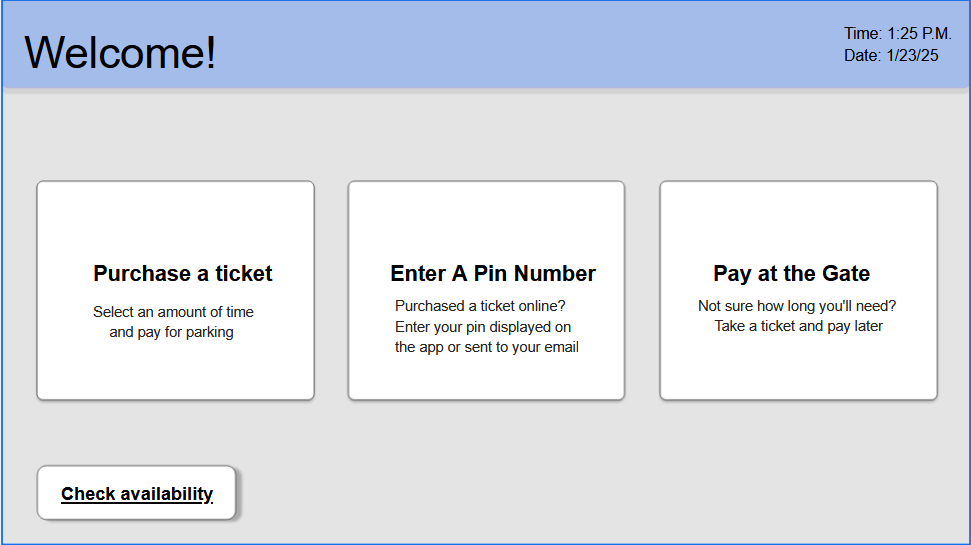


After selecting, the terminal will dispense a ticket with the designated parking space for the user and display the location on screen before returning to the main menu.

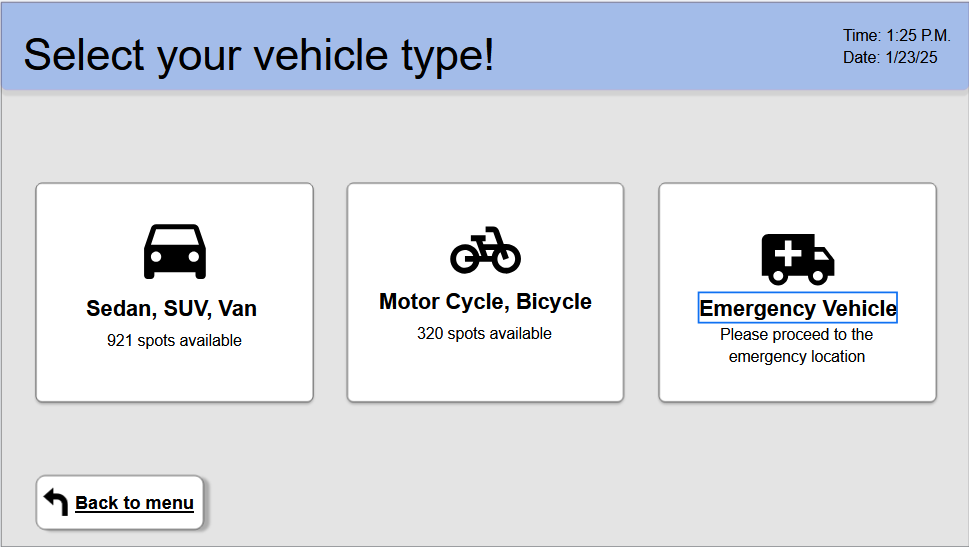


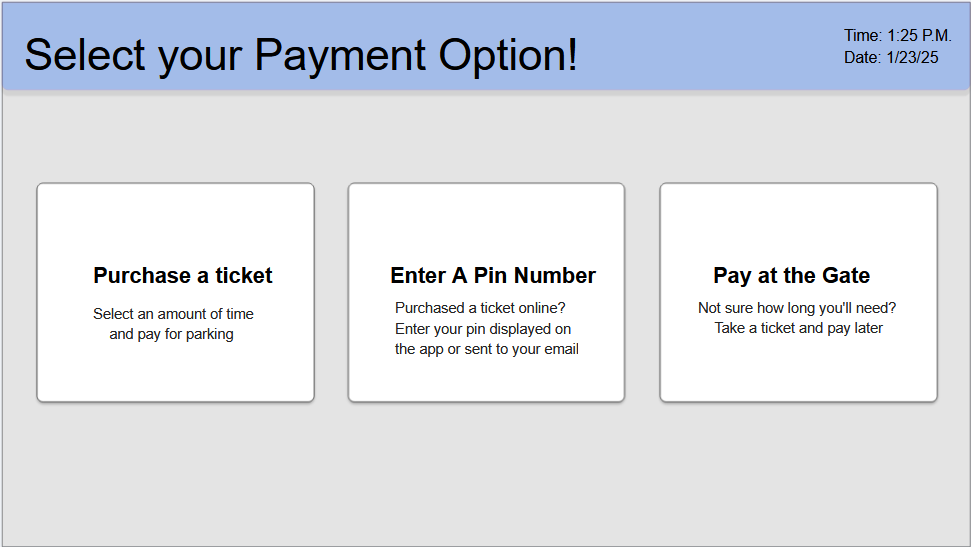
1. Real-Time spot availability display

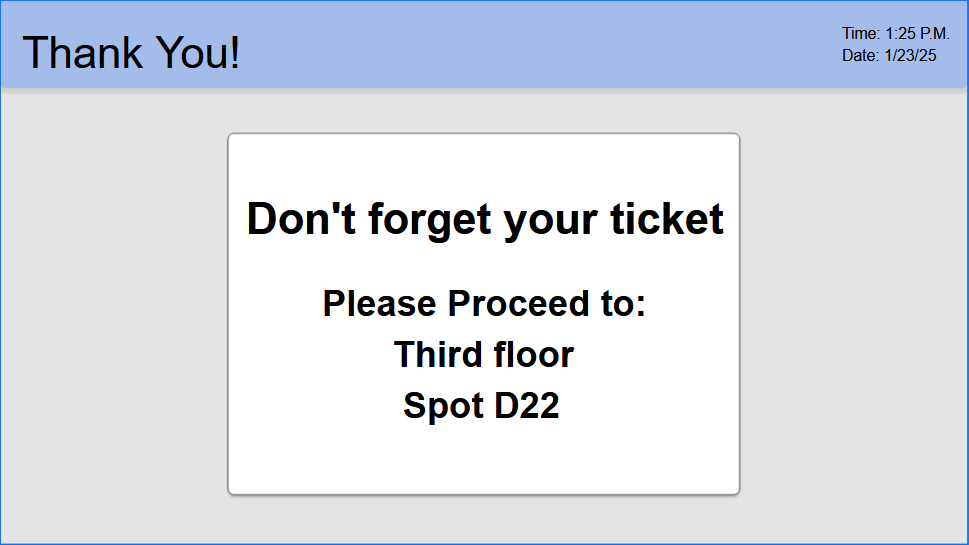
To check availability, the customer will tap the button on the main menu “Check availability” which is displayed on the lower left corner of the screen. This will redirect to the real-time availability screen.



Tapping on the vehicle will then display the purchase screen, locking in their vehicle selection for a seamless checkout.



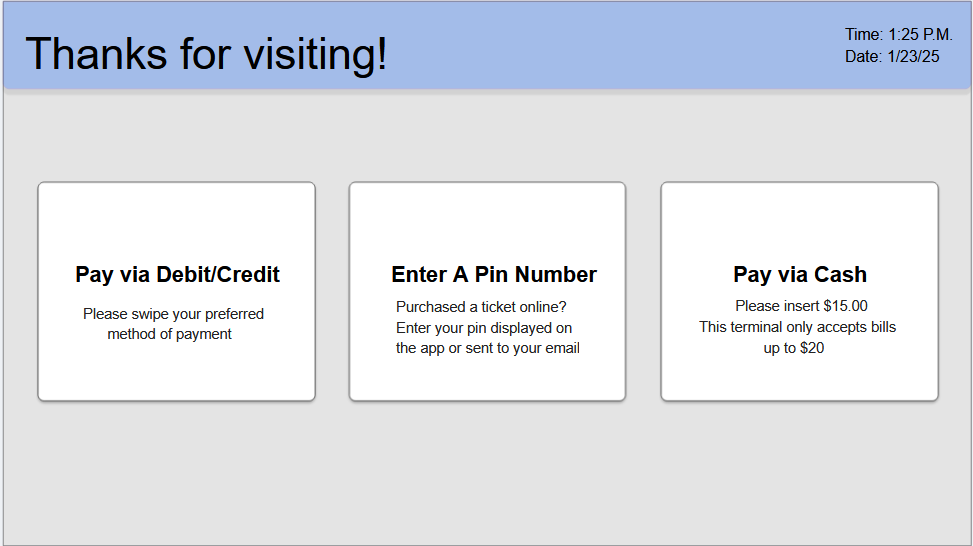




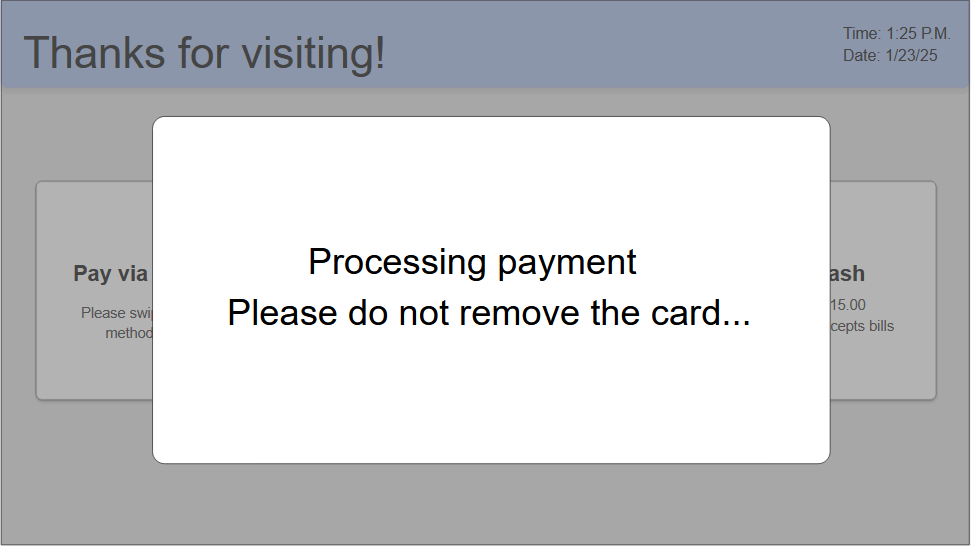
Note: The operation is relatively the same but allows the customer a chance to see any slots available in a different order. In this specific case, the customer chose the following path: Check availability, Sedan, Pay at the gate.

1. Payment Processing

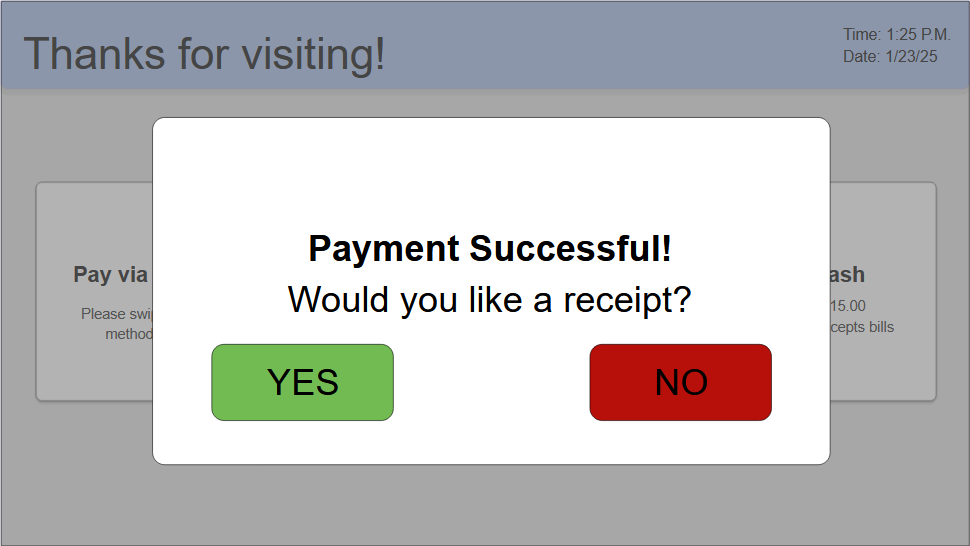
Using the “Pay at the gate option”, the customer will be prompted to insert cash or swipe a credit/debit card. If a customer purchased their ticket and paid previously but extended their time, the system will charge an additional fee based on 15-minute increments for the additional time used.



In this example, the debit card was used to pay for the ticket.



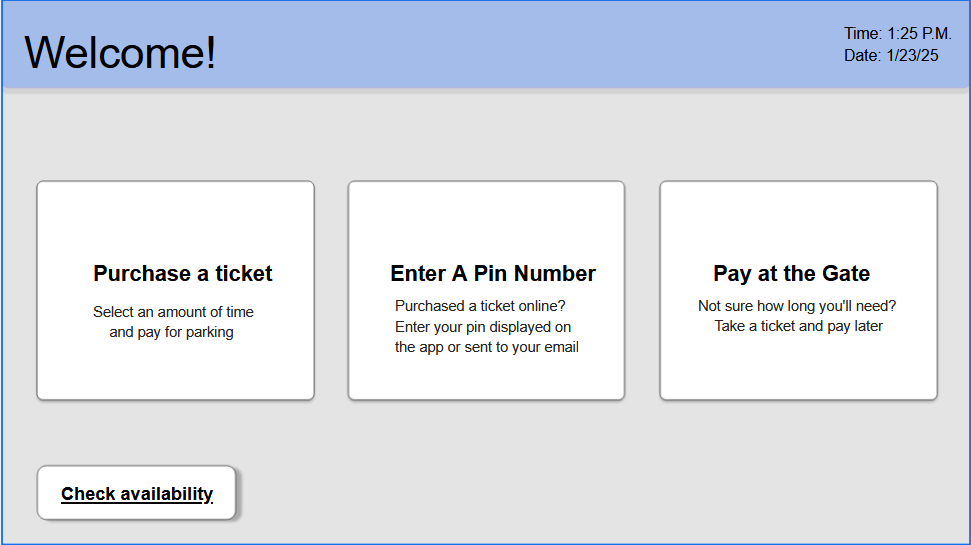
After the payment is processed, the system will notify the customer that the payment was successful and give the option to receive a receipt.

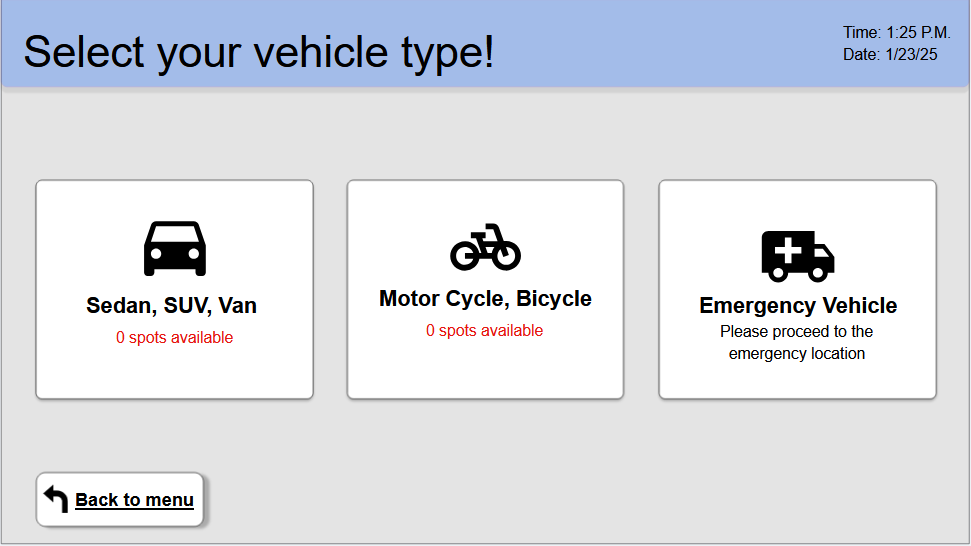


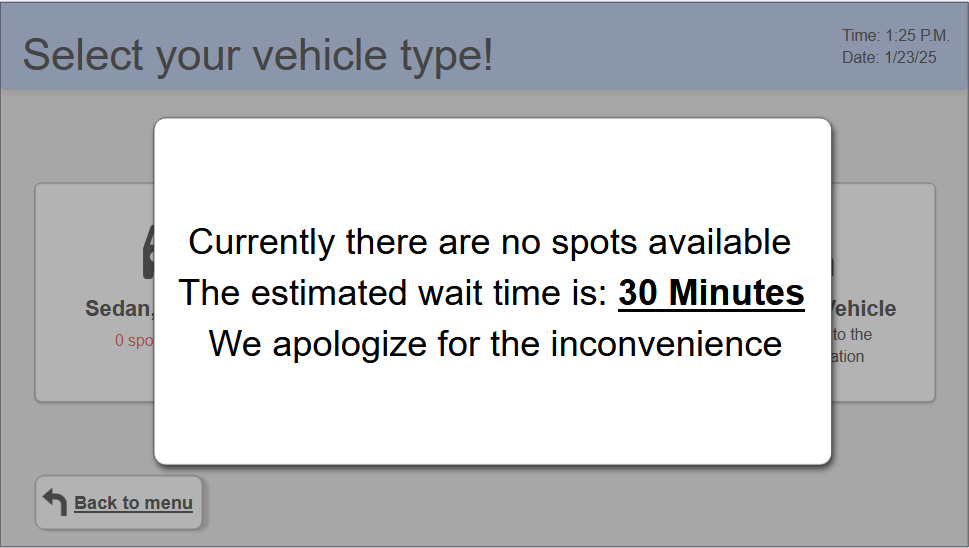
Tapping yes will dispense a receipt with the total amount, rate for time, date, time, and payment method used before returning to the main menu. Selecting no will simply return to the main menu.

1. System Overcapacity display

While checking availability and selecting a vehicle option that does not have any spots available, the system will output an overcapacity message with an estimated time until a new spot is available.

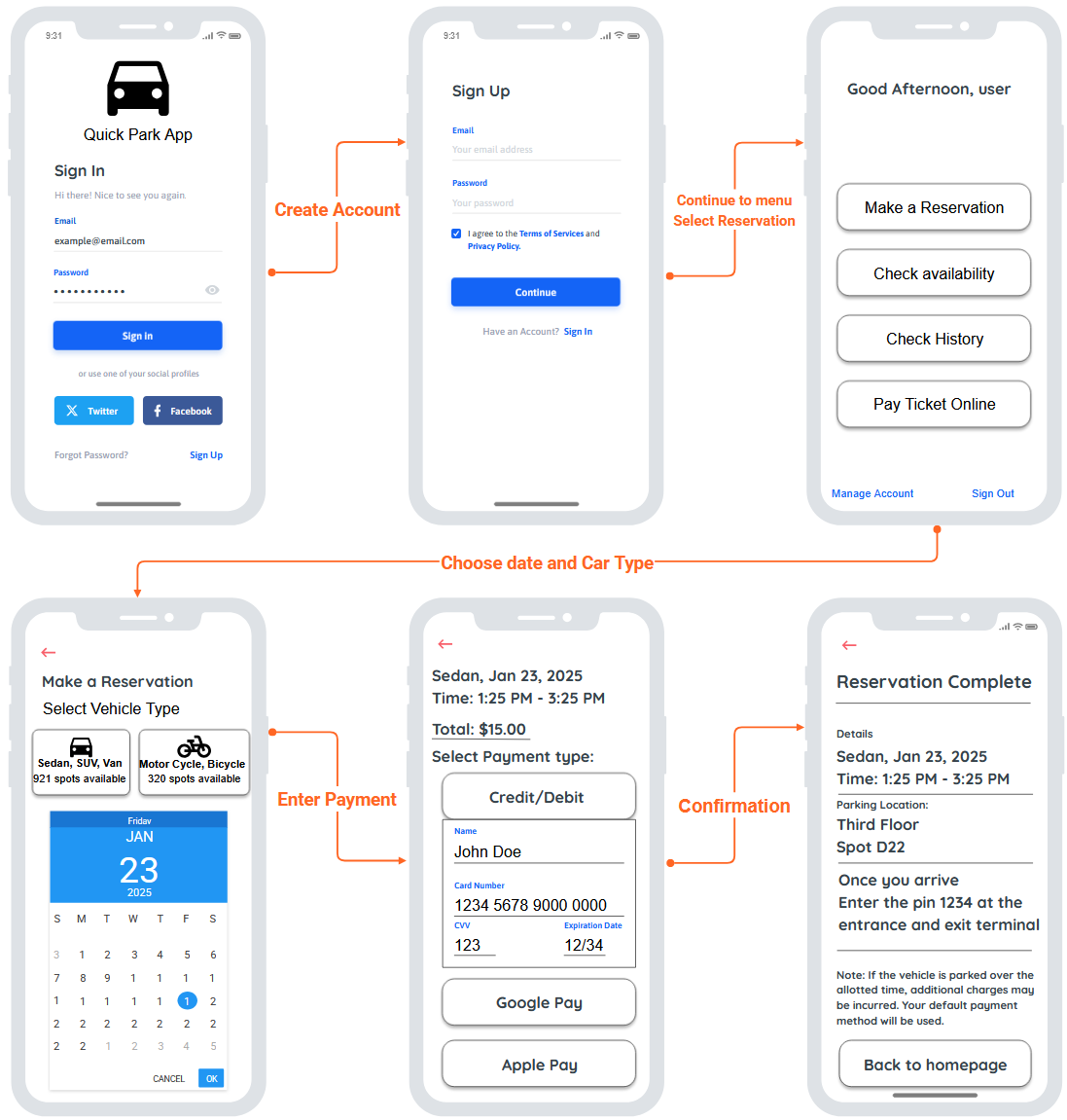






1. Parking Spot Reservation (Mobile App)

Downloading the Quick Parking App, A user can sign up and make a reservation online to expedite the process. After signing in/creating an account, the user will be prompted with options. The one for this example will be to make a reservation. After selecting their vehicle type and date, it will display the spots available before proceeding to the payments screen. After validation and processing payment, it will display a success screen with the reservation details, parking spot location, directions, pin number to enter the lot, and a notte that informs users that charges can be incurred if the vehicle stays over the paid time limit.



User Effort Estimation:

|  |  |  |  |
| --- | --- | --- | --- |
| Usage Scenario | Navigation | Clicks | Keystrokes |
| Ticket Issuance | Select Payment, Select Vehicle, Accept ticket, park at assigned location. | 2 | 0 |
| Real-time Spot Availability Display | Select Availability, Select vehicle | 1 | 0 |
| Payment Processing | Drive to exit terminal, Select payment option (Swipe card), take receipt | 2 | 0 |
| System Overcapacity Handling | Select availability, Select Vehicle type, display message | 1 | 0 |
| Create Online Reservation | Download the app, create an account, Select “make a reservation”, choose vehicle type and date, enter payment details, receive confirmation. | 11 | >100 |

# Project Plan

Development Approach

**Software & Technologies**

* Front-end: React.js (for a user-friendly web interface)
* Back-end: Node.js with Express.js (for handling API requests and business logic)
* Database: MySQL (for efficient data management and transaction processing)
* Security Integration: CCTV API for real-time monitoring and security alerts
* Payment Processing: Stripe API and PayPal API for digital transactions
* Cloud Hosting: AWS or Firebase for scalability and high availability

**Hardware & Network Requirements**

* Servers: Cloud-based or on-premises servers to store data and run applications.
* Parking Sensors & Display Boards: IoT-enabled sensors to detect parking availability.
* Automated Entry/Exit Gates: Integrated with the system to validate tickets and control access.
* Network Connectivity: Stable internet/Wi-Fi for real-time updates and cloud synchronization.

# Implementation Plan

Developmental Environmental Setup

**IDEs:** Spyder (Python + Tkinter support)

**Version Control:** Git hosted on GitHub (pull requests, code reviews)

**Languages & Frameworks:**

* Python 3.10 with Tkinter (desktop GUI)
* SSMS (database scripts)

**Justification:**

* Python/Tkinter is easy to maintain and cross-platform.
* SQL Server (SSMS) scales for >1M records and supports indexing.

Data Model Implementation

**ERD Overview (no need for picture since they aren’t technically connected):**

* **Car**(ID PK, Make, Model, Color)
* **Cycle**(ID PK, Make, Model, Color)
* **Fines**(FineID PK, Make, Model, FineAmount, FineTime)

**Relationships:**

* No direct FK between Car/Cycle and Fines (fines stored by Make/Model).

**Normalization & Indexing:**

* All tables in 3NF (no repeating groups).
* Index on FineTime for fast lookups of recent fines.
* Clustered PK on identity columns.

Testing Strategy

* **Approach:**
  + **Unit Tests:** Mock DB, test insert\_car, fetch\_car\_count, payment-logic functions.  
    **Integration Tests:** Simulate full purchase and security-fine workflows.
  + **System Tests:** End-to-end on a test machine with SQL Server.
  + Full thought out Unit and Integration tests are in the file structure.

Error Handling and Debugging

* **Approach:**
  + Wrap all DB calls in try/except, log exceptions
* Code Snippet below:

A screen shot of a computer program

AI-generated content may be incorrect.