Software Requirements Specification

for

<No Ticket for Me>

Version <1.1> approved

Prepared by <Brian Andrade , John Lopez, Gabriel Galindo, Anthony Sanchez, Braedon Edison, Angel Trujillo, Haonan Ma, Chiemela Eziechile>

<California State University , LA>

<11/22/24>

**Table of Contents**

Table of Contents................................................................................................................. pg #

Revision History................................................................................................................... pg #

1. Introduction................................................................................................................ pg #

1.1. Purpose........................................................................................................... pg #

1.2. Intended Audience and Reading Suggestions................................................ pg #

1.3. Product Scope................................................................................................ pg #

1.4. Definitions, Acronyms, and Abbreviations .................................................. pg #

1.5. References...................................................................................................... pg #

2. Overall Description.................................................................................................... pg #

2.1. System Analysis…......................................................................................... pg #

2.2. Product Perspective........................................................................................... pg #

2.3. Product Functions........................................................................................... pg #

2.4. User Classes and Characteristics.................................................................... pg #

2.5. Operating Environment.................................................................................. pg #

2.6. Design and Implementation Constraints........................................................ pg #

2.7. User Documentation...................................................................................... pg #

2.8. Assumptions and Dependencies.................................................................... pg #

2.9. Apportioning of Requirements...................................................................... pg #

3. External Interface Requirements............................................................................... pg #

3.1. User Interfaces............................................................................................... pg #

3.2. Hardware Interfaces....................................................................................... pg #

3.3. Software Interfaces........................................................................................ pg #

3.4. Communications Interfaces........................................................................... pg #

4. Requirements Specification....................................................................................... pg #

4.1. Functional Requirements............................................................................... pg #

4.2. External Interface Requirements................................................................... pg #

4.3. Logical Database Requirements.................................................................... pg #

4.4. Design Constraints......................................................................................... pg #

5. Other Nonfunctional Requirements........................................................................... pg #

5.1. Performance Requirements............................................................................ pg #

5.2. Safety Requirements...................................................................................... pg #

5.3. Security Requirements................................................................................... pg #

5.4. Software Quality Attributes........................................................................... pg #

5.5. Business Rules............................................................................................... pg #

6. Legal and Ethical Considerations.….......................................................................... pg #

Appendix A: Glossary........................................................................................................ pg #

Appendix B: Analysis Models........................................................................................... pg #

Appendix C: To Be Determined List................................................................................. pg #

Revision History

| Name | Date | Reason For Changes | Version |
| --- | --- | --- | --- |
| Brian | 10/10/24 | Creating chapters 1.0 - 1.5 | 1.0 |
|  | 11/22/24 | Update for new criteria. | 1.1 |
| Haonan Ma | 11/22/24 | Add 2.4, 3.3, 4.1, and 4.1.3 routes and functionality details from the GitHub repository. | 1.1 |
| John Lopez | 11/22/24 | Add and Remove portions of 4, 5, 6 and Appendix according to what we have for the program. |  |

<Add rows as necessary when the document is revised. This document should be consistently updated and maintained throughout your project. If ANY requirements are changed, added, removed, etc., immediately revise your document.>

**1.**  **Introduction**

This Software Requirements Specification (SRS) document provides a comprehensive overview of the "No Ticket 4 Me" Parking Registration System. The primary purpose of this document is to outline **what** the system is intended to achieve, defining its features, functions, and user interactions. The system is designed to streamline the parking registration process, allowing users to register their vehicles and avoid parking violations through an automated system that manages parking permits and registration status. This document will detail the scope, functionalities, and requirements of the system, without delving into the technical specifics of implementation, which will be addressed in the Software Design Document (SDD).

**1.1**  **Purpose**

The purpose of this document is to specify the software requirements for the "No Ticket 4 Me" Parking Registration System. It outlines the functional and non-functional requirements necessary for the development of the system, providing a detailed description of **what** the software must accomplish to meet user and system needs. This document is essential for ensuring that all stakeholders—developers, testers, and end-users—have a clear understanding of the system's intended functionality.

This version, **Release 1.0**, covers the full scope of the system’s software requirements. It addresses all aspects of the "No Ticket 4 Me" system, from user interactions and data management to security and performance requirements. No partial or incomplete features are documented here, ensuring a complete and cohesive overview of the entire system's functionality. Future revisions may extend or modify these requirements based on evolving needs or enhancements.

**1.2**  **Intended Audience and Reading Suggestions**

This Software Requirements Specification (SRS) is intended for various stakeholders involved in the development, management, and use of the "No Ticket 4 Me" Parking Registration System. These include:

* **Developers**: To understand the functional requirements, system behavior, and constraints to guide the implementation of the software.
* **Project Managers**: To track the scope of the project, ensure that all requirements are covered, and plan resources and timelines effectively.
* **Testers**: To design test cases based on the detailed functional and non-functional requirements, ensuring that the system meets the specified criteria.
* **Marketing Staff**: To gain a high-level understanding of the system’s capabilities and how it can be positioned to meet user needs.
* **End-Users**: To understand the key functions and features of the system, particularly if any feedback or input is required during development.
* **Documentation Writers**: To gather information for creating user manuals, support materials, and product documentation.

### **Reading Suggestions**

* **Developers** should focus on the **Functional Requirements** and **System Features** sections to understand what needs to be built. Additionally, the **Data Management** and **System Constraints** sections are crucial for technical design.
* **Project Managers** may find the **Overall Description** and **Scope** sections most relevant for understanding the project’s purpose and ensuring that all requirements are accounted for.
* **Testers** should concentrate on the **Functional Requirements** section, where the specific features and expected system behavior are detailed. This will help in designing comprehensive test cases.
* **Marketing Staff** may primarily focus on the **Introduction** and **System Overview** sections to understand the high-level functionality and user benefits.
* **End-Users** will benefit from reading the **Use Cases** and **User Interfaces** sections to understand how they will interact with the system.
* **Documentation Writers** should use the **Functional Requirements** and **User Interfaces** sections to create user guides and manuals that are aligned with the system’s actual features.

This SRS is organized in a logical flow, starting with a high-level overview and gradually diving into more detailed requirements and system specifications, making it accessible to all types of readers.

4o

**1.3**  **Product Scope**

The software product to be produced is the **"No Ticket 4 Me" Parking Registration System**. This system is designed to streamline the process of vehicle registration for parking and ensure that users can register their vehicles quickly and efficiently to avoid parking violations and fines.

### **What the Software Will Do**

The **"No Ticket 4 Me"** system will allow users to:

* Register their vehicles online for parking permits in designated areas.
* Update and manage their vehicle registration information.
* Receive notifications about parking violations or permit expirations.
* Track real-time parking availability in registered areas (if supported by the parking facility).

The system will also enable parking authorities to:

* Verify registered vehicles for parking compliance.
* Issue electronic citations for unregistered or unauthorized vehicles.
* Manage parking data and generate reports on parking usage and violations.

### **What the Software Will Not Do**

The **"No Ticket 4 Me"** system does not provide enforcement mechanisms, such as physical ticketing or towing, nor does it handle payments for parking fines directly. It also does not track vehicle location outside the parking facilities it governs.

### **How the Software Will Be Used**

Once released, the **"No Ticket 4 Me"** system will be used by both drivers and parking authorities. Users will interact with the system via a web or mobile interface to register their vehicles and manage their permits. Parking authorities will use the administrative dashboard to manage parking rules and verify compliance.

### **Benefits of the Software**

The primary benefits of the **"No Ticket 4 Me"** system include:

* **User Convenience**: Drivers can avoid tickets by easily registering their vehicles and keeping their parking permits up-to-date.
* **Efficiency for Authorities**: Parking officials can quickly identify unregistered vehicles and issue electronic citations, reducing manual work.
* **Data Management**: Parking data will be centralized, making it easier for authorities to monitor and manage parking operations and ensure parking compliance.

### **Objectives and Goals**

The main goal of the **"No Ticket 4 Me"** system is to reduce the occurrence of parking violations by offering an easy-to-use registration platform. It aims to simplify the parking permit process, improve compliance, and make parking management more efficient for authorities, while enhancing the overall user experience for drivers.

**1.4**  **Definitions, Acronyms, and Abbreviations**

This section provides definitions for key terms, acronyms, and abbreviations used throughout the **"No Ticket 4 Me"** Software Requirements Specification (SRS) document. Understanding these terms is essential for a clear interpretation of the document and the system being developed.

### **Key Terms**

* **Parking Permit**: An authorization allowing a vehicle to park in a designated area for a specified period.
* **User**: Refers to any individual using the system, including drivers who register their vehicles and parking authorities managing the registration process.
* **Parking Authority**: The administrative body responsible for managing parking rules, permits, and issuing citations.
* **Citation**: An electronic notice of violation issued to a vehicle that is not properly registered or is violating parking rules.
* **System Administrator**: A user with elevated privileges responsible for managing system configurations, user roles, and data integrity.

### **Acronyms and Abbreviations**

* **SRS**: Software Requirements Specification
  + The document outlining the detailed requirements of the software system.
* **UI**: User Interface
  + The visual and interactive components through which users interact with the system.
* **API**: Application Programming Interface
  + A set of functions that allow external systems or components to interact with the system.
* **DBMS**: Database Management System
  + The software used to manage and store the system’s parking registration data.
* **SMS**: Short Message Service
  + A communication protocol used to send text messages to users for notifications, such as registration confirmations or citations.
* **SSL**: Secure Sockets Layer
  + A standard security protocol for establishing encrypted links between a server and a client, used to ensure data confidentiality.

### **Note:**

If additional definitions or technical jargon arise during the course of development, they will be documented in the **Appendix** section at the end of this document for further clarification.

**1.5**  **References**

This section lists all documents and web addresses referenced in the **"No Ticket 4 Me"** Software Requirements Specification (SRS). Each reference includes enough detail to allow readers to access or obtain the documents. The references serve as foundational materials for understanding industry standards, design principles, and security measures that inform the requirements of this system.

### **Documents**

1. **IEEE Standard for Software Requirements Specifications (IEEE 830-1998)**
   1. **Author**: IEEE Software Engineering Standards Committee
   2. **Version**: IEEE 830-1998
   3. **Date**: 1998
   4. **Source**: IEEE Standards Association

[https://standards.ieee.org](https://standards.ieee.org/)

1. **National Parking Association: Parking Management Guidelines**
   1. **Author**: National Parking Association
   2. **Version**: 3rd Edition
   3. **Date**: February 2023
   4. **Source**: National Parking Association

[https://weareparking.org](https://weareparking.org/)

1. **W3C User Interface Design Guidelines for Web Applications**
   1. **Author**: World Wide Web Consortium (W3C)
   2. **Version**: W3C Recommendation
   3. **Date**: January 2021
   4. **Source**: World Wide Web Consortium

<https://www.w3.org/TR/>

1. **General Data Protection Regulation (GDPR) Compliance Guidelines**
   1. **Author**: European Union
   2. **Version**: 1.0
   3. **Date**: May 2018
   4. **Source**: European Commission

[https://gdpr.eu](https://gdpr.eu/)

1. **OWASP API Security Best Practices**
   1. **Author**: Open Web Application Security Project (OWASP)
   2. **Version**: 2022 Edition
   3. **Date**: April 2022
   4. **Source**: OWASP

[https://owasp.org](https://owasp.org/)

### **Web References**

1. **SSL.com Secure Sockets Layer (SSL) Certificate Guidelines**
   1. **URL**: <https://www.ssl.com/faqs/>
   2. **Author**: SSL.com
   3. **Version**: 2.5
   4. **Date**: August 2023
2. **Smashing Magazine: Mobile UI/UX Design Patterns**
   1. **URL**: <https://www.smashingmagazine.com/>
   2. **Author**: Smashing Magazine
   3. **Version**: Regularly Updated
   4. **Date**: Accessed September 2024

These references are integral for defining the usability, security, and data protection standards that guide the design and implementation of the **"No Ticket 4 Me"** system.

**2. Overall Description**

The software will be is website application for the purpose of purchasing a parking pass at CalState LA for various time frames, including 1-day, 1-month, and 1-semester, with a discount applied for those operating a 2-wheel vehicle. The website will also be able to store the inputted data of the user, including vehicle and personal information. Lastly the website will be capable of connecting to the California Department of Motor Vehicles to allow the user to pay off any outstanding tickets the user may have

**2.1 System Analysis**

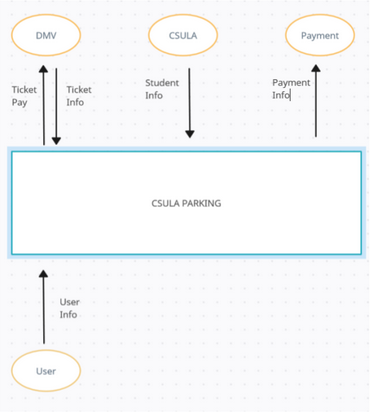
The purpose for the creation of this software is to improve the user experience and increase accessibility for both PC and mobile users.

1. Save user data retroactively with accounts linked to individual user ID’s saved across platforms and create a website equally accessible on mobile devices as well as on PC.
2. Major issue was implementing payment processors to allow purchases
3. Plans for overcoming the hurdles in order are establishing and maintaining communications with theusing easy user-friendly payment processors such as Stripe, and using a responsive CSS framework, bootstrap, to easily create an application capable of meeting both viewing requirements.

**2.2 Product Perspective**

Our software will be its own independent website, separate from the schools portal to allow ease of access and an online method of payment separate from kiosk.

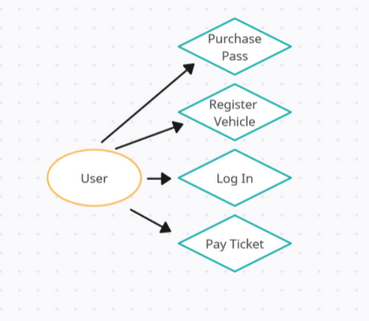
Our software is a direct upgrade from the schools previous parking website. It is similar in the sense that it is able to do the same things as before, however we plan to add improvement to the UI to increase accessibility for desktop and mobile users.

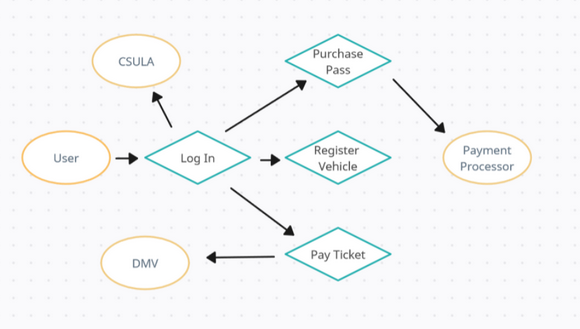




**2.3 Product Functions**

The software will allow for three major and primary functions, paying unpaid parking tickets, registering a 2 or 4 wheeled vehicle with the school, and purchasing digital parking passes.





**2.4 User Classes and Characteristics**

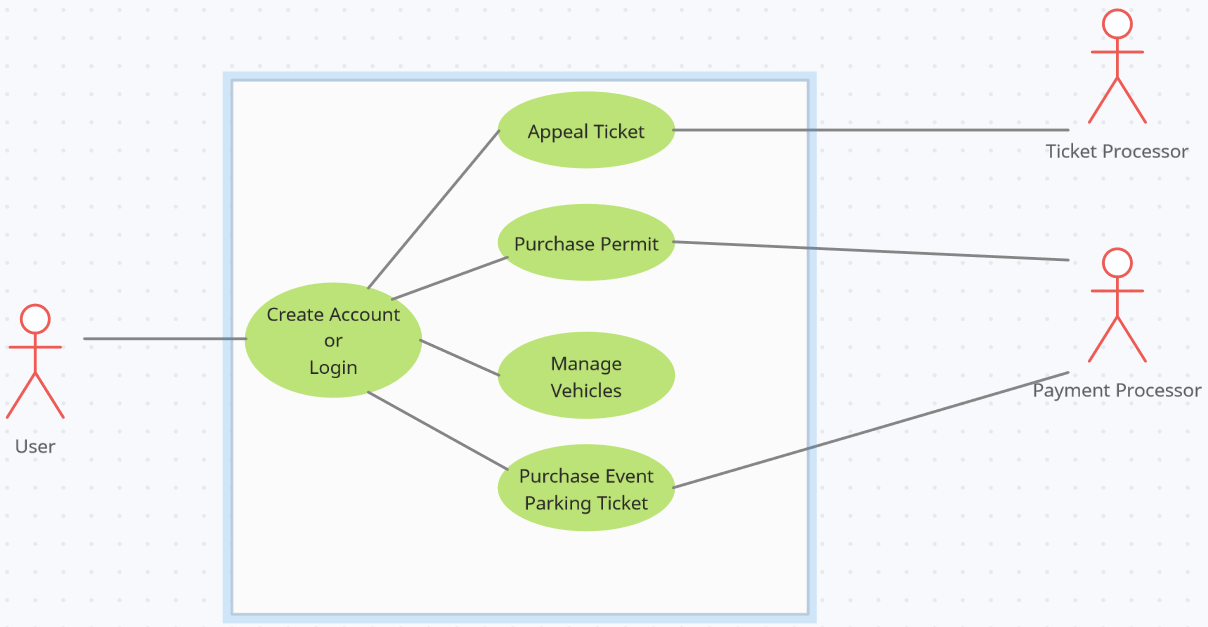
We expect 3 types of users to interact with our website. The primary group will be students who attend CalState LA, the 2nd group we expect are faculty who work at CalState LA, in both scenarios the user will be more inclined to purchase a semester pass. The last group we expect are miscellaneous visitors who will only attend once or twice, and in this scenario will more than likely only purchase a single day pass.

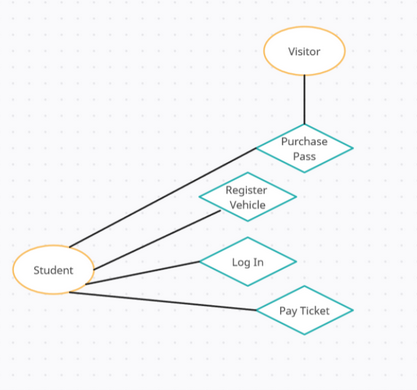
#### Regular Users

* Use cases:
  + Register and log in to the system.
  + Manage vehicles via /manage\_vehicle.
  + Appeal tickets via /appeal\_ticket.
  + Purchase parking tickets via /purchase\_ticket.

#### Administrators

* Elevated privileges include:
  + Managing events via /admin/events.
  + Viewing and editing user and vehicle details via /admin/get\_user and /admin/get\_vehicle.





**2.5 Operating Environment**

The software will run on a combination of multiple key softwares on a windows 10 machine, with the intended use environment being any and all machines capable of connecting to the wider internet.

* Python
* PostgreSQL
* HTML
* JavaScript
* CSS

**2.6 Design and Implementation Constraints**

Primary reasons for development constraints that we may experience are as followed:

* Hardware limitations: Lack of appropriate development needs.
* Time constraints: Lack of available time to work on project.
* Compliance constraints: Lack of communication with needed 3rd party software.
* Software Constraints: Lack of expertise and or familiarity with tools leading to increased development time.

**2.7 User Documentation**

Any user documentation to be uploaded will be appended to the end of the document if applicable.

**2.8 Assumptions and Dependencies**

Possible design constraints that may affect the requirements listed in this document include but are not limited to:

* California DMV repository disruptions
* Stripe payment API disruptions
* Google sign-in API disruption

**2.9 Apportioning of Requirements**

List any requirements that might be delayed until future versions of the system.

**3. External Interface Requirements**

**3.1**  **User Interfaces**

The parking system will be accessible through a easy to navigate UI that will have everything they need a few taps away. Users will be able to quickly register their cars and pay for any tickets all in one place rather than having to click around a bunch of different links The interface will be designed to:

* Work seamlessly with both mouse/keyboard and touchscreens, making it easy for anyone to use, whether on a computer or a phone. Preferably phone as most people will access this as they are looking for a place to park for the first time.
* Include standard buttons like "Help," "Submit," and "Cancel" to keep things straightforward across all pages.
* Comply with ADA guidelines, ensuring it's accessible for everyone, including screen reader support, high-contrast options, and full keyboard navigation.

**3.2**  **Hardware Interfaces**

The system will not need many hardware connections however a possible connection could be a GPS so that the app can quickly show which parking lot they are in and if they are allowed to park there.

**3.3**  **Software Interfaces**

#### 1. Stripe Payments API

* **Version**: 3.2
* **Source**: Stripe Payments, Inc. ([API Documentation](https://docs.stripe.com/api))
* **Integration Details**:
  + The parking system will use the Stripe Payments API to handle all payment transactions securely. The API will be accessed via HTTPS, and all payment details will be encrypted using TLS.
  + **Data exchange format**: JSON over HTTPS.
  + The system will send payment requests with user and account data, including the transaction amount and user ID.
  + The API requires API key authentication to validate requests.

#### 2. MySQL Database (via SQLAlchemy ORM)

* **Version**: 8.0
* **Source**: Oracle ([MySQL Documentation](https://www.mysql.com/))
* **Integration Details**:
  + The parking system will use SQLAlchemy ORM to interact with the MySQL database and abstract SQL queries.
  + Tables:
    - User: Stores user information such as email, password (hashed), account status, and admin roles.
    - Transaction: Tracks user transactions, including payment details and timestamps.
    - Event: Manages event parking details.
  + Communication with the database is through secure authentication and encrypted connections.
  + **Data Relationships**:
    - Users can have multiple vehicles (one-to-many).
    - Vehicles can have various transactions (one-to-many).

#### 3. Twilio REST API

* **Version**: Latest
* **Source**: Twilio (API Documentation)
* **Integration Details**:
  + The parking system integrates with Twilio for SMS-based notifications.
  + Notifications include ticket purchase confirmations, appeals status updates, and permit expiration reminders.
  + **Data exchange format**: JSON over HTTPS.
  + Authentication via API keys configured in environment variables.

#### 4. Google Sign-In API

* **Version**: 4.1
* **Source**: Google Developers ([API Documentation](https://developers.google.com/identity/sign-in/web))
* **Integration Details**:
  + The system will provide a Google Sign-In option for users, allowing login with their Google accounts.
  + **Data exchange format**: JSON Web Tokens (JWT) over HTTPS.
  + User information (e.g., name, email) will be securely stored upon successful sign-in.
  + Tokens will be validated against Google's authentication servers.

#### 5. Flask Framework

* **Version**: Latest stable release
* **Source**: Flask (Documentation)
* **Integration Details**:
  + Serves as the application’s core back-end framework.
  + Manages routing, session handling, and integration with other APIs.
  + Provides the HTTP endpoints (e.g., /purchase\_ticket, /manage\_vehicle) defined in the app.py file.

**3.4**  **Communications Interfaces**

**Web Browser Access**

* The system will be web-based, accessible via modern browsers. It will use HTTPS as the primary protocol to encrypt all communications.
* Message formatting will be JSON for API communications and HTML/CSS for web content presentation.
* All web transactions, such as user login, vehicle registration, and payment processing, will be secured using TLS to protect user information.

**Email Notifications**

* The system will send automated email notifications for when certain actions are taken such as payments or account confirmation etc.
* The email service will use the SMTP standard with TLS encryption.
* Email content will be formatted in both HTML (for enhanced styling) and plain text (for accessibility and compatibility) to accommodate different email clients.
* Data synchronization: The email system will work in real-time, triggered by key user actions to send messages promptly.

**Communication Standards and Security**

* The parking system will adhere to the following communication standards:
  + **HTTPS**: Used for all web and API communications.
  + **TLS:** Encrypts data in transit to protect sensitive information
  + **SMTP with TLS**: Ensure security of messages.
* The system will implement encryption mechanisms such as AES for various data stored in the database.

**4. Requirements Specification**

This section outlines the detailed software requirements necessary for the **"No Ticket 4 Me"** Parking Registration System. These requirements will guide developers in implementing the system’s functionality to meet the needs of users and stakeholders. Each requirement is designed to be traceable, verifiable, and specific, ensuring the system's functionality aligns with project goals.

### **Functional Requirements**

The functional requirements define **what** the system must do. These include inputs, outputs, processes, and interactions between users and the system.

1. **User Registration and Authentication**
   * **[R1]**: The system shall allow users to create an account by providing their name, email, phone number, and password.
   * **[R2]**: The system shall validate the user’s email address during registration.
   * **[R3]**: The system shall allow users to log in using their email address and password.
   * **[R4]**: The system shall enforce password security, requiring at least 8 characters, one uppercase letter, one number, and one special character.
2. **Vehicle Management**
   * **[R5]**: The system shall allow users to add, update, or delete vehicle information, including the vehicle's make, model, year, and license plate number.
   * **[R6]**: The system shall validate vehicle license plate formats based on the user’s region.
   * **[R7]**: The system shall allow users to register multiple vehicles under one account.
3. **Permit Application and Renewal**
   * **[R8]**: The system shall allow users to apply for a parking permit by selecting a permit type (e.g., monthly, daily).
   * **[R9]**: The system shall calculate the cost of the permit based on the user’s vehicle and permit type.
   * **[\*R10]**: The system shall allow users to renew their permits before expiration and send automated reminders 7 days before the expiration date.
4. **Violation Management**
   * **[R11]**: The system shall allow parking authorities to issue parking violations, including details such as location, violation type, and penalty amount.
   * **[R12]**: The system shall notify users of issued violations via email and in their account dashboard.
   * **[R13]**: The system shall allow users to pay penalties directly through the platform using secure payment options.(Stripe)
5. **Notifications and Alerts**
   * **[R16]**: The system shall send notifications to users for permit renewals, violations, and payment reminders via email.
6. **Reporting for Authorities**
   * **[R18]**: The system shall provide parking authorities with the ability to generate reports on the number of registered vehicles, active permits, and violations within a given time period.
   * **[R19]**: The system shall allow parking authorities to export data in CSV or PDF formats.

### **Non-Functional Requirements**

Non-functional requirements define the system's quality attributes, such as performance, security, and usability.

1. **Security**
   * **[R20]**: The AWS system uses AES-256 encryption for all data transmitted between users and the system to protect sensitive information.
   * **[R21]**: The system shall enforce multi-factor authentication for administrators and parking authorities with their special key only associated to that Admin User.
2. **Performance**
   * **[R23]**: The system shall support up to 1,000 concurrent users without performance degradation.
   * **[R24]**: The system shall process user actions (e.g., logins, registrations, payments) in under 2 seconds for 95% of transactions.
3. **Usability**
   * **[R25]**: The system shall provide a mobile responsive user interface that works across all modern browsers and devices.
4. **Availability**
   * **[R27]**: The system shall maintain 99.9% uptime, with planned maintenance occurring during off-peak hours.

### **Conclusion**

This section has outlined the key functional and non-functional requirements for the **"No Ticket 4 Me"** Parking Registration System. These requirements provide the foundation for designing and implementing the system to meet user needs, ensuring a reliable, secure, and user friendly experience. All requirements are specific, measurable, and traceable, ensuring that developers and testers can verify that the system performs as expected.

**4.1 Functional Requirements**

This section outlines the necessary software requirements for the **"No Ticket 4 Me"** Parking Registration System, providing developers with detailed specifications for inputs, outputs, and system actions. The goal is to ensure the system is designed accurately to meet the functional needs of both users and stakeholders. Each requirement is formulated to be traceable, verifiable, and specific, allowing for systematic design, development, and testing.

The requirements are categorized into **functional** and **non-functional** requirements:

1. **Functional Requirements**: Define specific tasks and behaviors that the system must perform, such as user interactions, vehicle registration, permit management, violation handling, and reporting.
2. **Non-functional Requirements**: Address system quality attributes such as security, performance, scalability, and user experience, ensuring the system operates effectively under various conditions.

By adhering to these requirements, the development process will align with the intended system functionality, ensuring that it delivers value to the users and meets the operational expectations of parking authorities and administrators.

### **4.1 Functional Requirements**

The functional requirements define the core actions of the **"No Ticket 4 Me"** Parking Registration System in terms of input handling, output generation, and data processing. Each requirement is expressed as **"The system shall..."** to indicate mandatory functionality that must be implemented. These requirements ensure that the system performs specific tasks needed to fulfill user and stakeholder expectations.

### **Core Functional Requirements**

#### User Registration and Authentication

* [R1]: The system shall allow users to sign up via the /sign-in route by providing their name, email, and password.
* [R2]: The system shall allow users to log in via the /login route with validated credentials.
* [R3]: The system shall allow users to log out via the /logout route.

#### Vehicle Management

* [R4]: The system shall allow users to manage their vehicles via the /manage\_vehicle route, including adding, updating, or deleting vehicles.
* [R5]: The system shall validate license plate numbers before adding a vehicle.
* [R6]: The system shall allow administrators to edit or retrieve vehicle information via /admin/get\_vehicle and /admin/edit\_vehicle.

#### Ticket Management

* [R7]: The system shall allow users to purchase parking tickets via the /purchase\_ticket route.
* [R8]: The system shall allow users to appeal parking tickets via the /appeal\_ticket route.

#### Event Parking

* [R9]: The system shall allow users to view and register for event-specific parking via the /event-parking route.
* [R10]: The system shall enable administrators to manage event details through the /admin/events route.

#### Account Management

* [R11]: The system shall provide a dashboard at /account for users to view and manage their account details.
* [R12]: Users shall be able to view all past transactions via /view\_transactions.

### **Input, Output, and Data Processing**

* **Input**: User-provided data such as personal details, vehicle information, permit applications, and payment information.
* **Output**: System responses, including permit approvals, violation notices, payment confirmations, and notifications.
* **Data Processing**: Validation, storage, retrieval, and updating of user, vehicle, and violation information.

Each of these functional requirements is designed to ensure the system provides the necessary capabilities to users and parking authorities, streamlining the overall parking management process.

#### 4.1.1 User Registration

4.1.1.1 The system shall allow users to create an account by providing personal information such as name, email, phone number, and vehicle information.

* The system shall validate the email address and phone number before creating the account.
* The system shall require a unique username for each user account.
* The system shall hash and store the user’s password securely.
* The system shall send a confirmation email to the user upon successful registration.

#### 4.1.2 Vehicle Registration

4.1.2.1 The system shall allow users to register their vehicles by entering vehicle details (license plate, make, model).

* The system shall validate the format of the license plate.
* The system shall allow users to register multiple vehicles under one account.

#### 4.1.3 Parking Permit Application

4.1.3.1 The system shall allow users to apply for parking permits based on the registered vehicle via the /purchase-ticket route.

* The system shall offer different permit types (e.g., daily, monthly, annual).
* The system shall display the price of each permit before purchase through the user interface.
* The system shall validate payment details using an external payment API (e.g., Stripe Payments) and securely process the transaction.
* The system shall issue a confirmation notification (via email and SMS) once the permit is successfully purchased.
* The system shall allow users to view purchased permits and their details on the /account dashboard.

#### 4.1.4 Notifications for Permit Expiration

4.1.4.1 The system shall automatically send notifications to users (via email) a configurable number of days before their permit expires.

* The system shall automatically send notifications to users (via email and SMS) a configurable number of days before their permit expires.
* By default, the system shall send a reminder 7 days before the expiration date and allow users to adjust this setting via notification preferences.
* Notifications shall include the permit expiration date, renewal link, and instructions for extending the permit.

#### 4.1.5 Administrative Dashboard

4.1.5.1 The system shall provide an administrative dashboard for parking authorities to manage user accounts and permits.

* The system shall allow administrators to issue citations for parking violations.
* The system shall allow administrators to generate reports on parking permits and usage.
* The system shall enforce role-based access control, ensuring only authorized users can access the dashboard. (Admins)

**4.2 External Interface Requirements**

This section specifies the inputs, outputs, and communication between the software system and external systems or users.

#### 4.2.1 Input: User Registration Form

* **Description**: Form for users to input personal and vehicle details.
* **Source**: User interface
* **Valid Range**: Name must be alphanumeric; email must follow email format standards.
* **Timing**: Instant validation upon form submission.
* **Data format**: The JSON payload was sent to the backend.

#### 4.2.2 Output: Email Notification

* **Description**: Email sent to users after successful registration or purchase of a parking permit.
* **Destination**: User’s email inbox.
* **Format**: HTML and plain text versions.
* **Timing**: Within seconds after the operation.
* **Content**: Includes the user’s name, confirmation details, and permit expiration date.

#### 4.2.3 External System: Stripe Payments API

* **Description**: External API used for handling payments.
* **Data format**: JSON request and response.
* **Method**: HTTPS POST requests.
* **Response**: Payment confirmation or error.

**4.3 Logical Database Requirements**

The system uses a relational database to manage user, vehicle, permit, and notification data. The following sections define the logical structure of the database.

#### 4.3.1 Data Entities

* **Users**: Stores user information such as username, email, and hashed passwords.
* **Vehicles**: Stores vehicle details, linked to users by a foreign key.
* **Permits**: Stores permit information, including type, price, and expiration date.
* **Notifications**: Logs all notifications sent to users, including type (email) and date.

#### 4.3.2 Data Relationships

* A **User** can have multiple **Vehicles** (one-to-many relationship).
* A **Vehicle** can have multiple **Permits** (one-to-many relationship).
* A **User** can receive multiple **Notifications** (one-to-many relationship).

#### 4.3.3 Data Retention and Constraints

* **User Data**: Retained indefinitely unless requested for deletion by the user (GDPR compliance).
* **Constraints**:
  + Primary keys for unique identification of records.
  + Identification to enforce relationships between users, vehicles, and permits.
  + Indexing on frequently searched fields like email and license plate for performance optimization.

**4.4 Design Constraints**

The following design constraints influence the development and implementation of the system:

#### 4.4.1 Regulatory Compliance

* The system shall comply with the C**alifornia Consumer Privacy Act (CCPA)** for users in California, USA.
* The system shall comply with the **ADA** (Americans with Disabilities Act) to ensure accessible user interfaces.

#### 4.4.2 Hardware Limitations

* The mobile version of the system shall be optimized for smartphones and tablets.

#### 4.4.3 Performance Constraints

* The system shall process parking permit purchases within 3 seconds on average.

#### 4.4.4 External API Integration

* The system shall rely on external APIs such as the Stripe Payments API for handling financial transactions

### 

### **5. Other Nonfunctional Requirements**

#### 5.1 Performance Requirements

* The system must support 50 simultaneous users during normal operations and should be scalable to 100 users during peak load.(1000 max using AWS Services free trial)
* The system must process 95% of transactions (e.g., parking permit purchases, user logins) in less than 2 seconds under normal conditions.
* During peak times, the system must process 80% of transactions within 3 seconds and 99% within 5 seconds.

#### 5.2 Safety Requirements

* The system must ensure no loss or corruption of data during data entry, transmission, or storage.

#### 5.3 Security Requirements

* All user data must be encrypted using bcrypt and sensitive data transmissions should be secured using AWS Encryption
* Users must authenticate using strong passwords and passwords must be hashed and stored securely.

#### 5.4 Software Quality Attributes

* Availability: The system must maintain an uptime of 99.9%, with a maximum downtime of 9 hours per year.
* Maintainability: The system will be designed with modular code, enabling easy updates and bug fixes. Maintenance windows will be scheduled for non-peak hours.
* Usability: The user interface must be intuitive and easy to navigate, requiring no more than 3 clicks for users to perform key tasks like purchasing a parking permit.
* Reliability: The system must ensure a 99% success rate in form submissions
* Portability: The software should be easily portable across Windows, Mac and the front end must be compatible with modern web browsers such as chrome, safari and firefox

#### 5.5 Business Rules

* Admins will have the ability to view, edit, and delete user and vehicle information, but regular users will only be able to modify their own data.
* Parking permits will only be valid for registered vehicles and the system will verify the license plate number before issuing the permit.
* Payment for parking permits can only be processed once the system verifies that the user has provided complete and valid vehicle information.
* User sessions must be logged out when users manually log out or after a session expires due to inactivity

**6. Legal and Ethical Considerations**

* Vehicular Information:
  + The biggest legal issues that would be involved in the project would be security concerns. Customers would want to know that their information as the process involves them entering their vehicle information.
  + Customers would also wonder what we do with that information
* Data/Privacy:
  + Customers would be concerned over whether or not the product is collecting their information and what it may do with that
* ADA:
  + The product must comply with the Americans with Disabilities Act
    - ex: alternate text for images
* CCPA:
  + Allow users to ask for their information to be deleted

**Appendix A: Glossary**

**API (Application Programming Interface):** A set of functions and protocols that allows different software applications to communicate with each other. In this project, APIs are used for external services such as Stripe Payments and Google Sign-In.

**\*\*\*bcrypt:** A password hashing function used to securely store passwords. The system uses bcrypt to ensure passwords are stored in a secure format.

**CSS (Cascading Style Sheets):** A stylesheet language used for describing the presentation of a web page. CSS is used to style the user interface of the parking registration system.

**DBMS (Database Management System):** Software that allows the creation, management, and manipulation of databases. MySQL is the DBMS used in this system.

**HTML (HyperText Markup Language):** The standard markup language for creating web pages. HTML is used to structure the content on the parking system's web interface.

**MySQL:** A popular open-source relational database management system used to store and manage user, vehicle, and parking data.

**PostgreSQL:** A powerful, open-source relational database system. It may be used in future iterations of the project for better scalability.

**SQL (Structured Query Language):** A standard language for managing and querying databases. SQL is used to interact with the MySQL database.

**\*\*\*Stripe Payments API:** An external API used for handling payment transactions in the parking system.

**UI (User Interface):** The visual part of a software application through which users interact with the system. The parking system's UI includes web pages for registering vehicles, paying for parking, and managing permits.

**User Session:** A period during which a user interacts with the system. Sessions are used to maintain state and track authenticated users.

**Vehicle Registration:** The process by which users register their vehicle details, such as license plate and make, within the system.