Software Engineering
Software Requirements Specification (SRS) Document

[Project Name]

[Date]

[Version]

By: [Team Member Names]

[Honor Code]

Table of Contents

1. Introduction

1.1.Purpose

[The goal of your project and the objectives it wishes to accomplish]

The goal of the Puzzle application is to enable rock climbers to effortlessly keep track of their climbing progress and to help them find the motivation they need to achieve their rock climbing goals by connecting them to the rock-climbing community through climbing gyms.

1.2.Document Conventions

[Full description of the main objectives of this document in the context of your project.
Here's how you should begin this section:
"The purpose of this Software Requirements Document (SRD) is to..."
"In it, we will . . ., . . ., and . . .."]

The purpose of this Software Requirements Document (SRD) is to describe the client-view and developer-view requirements for the Automated Police Ticketing System (APTS). Client-oriented requirements describe the system from the client's perspective. These requirements include a description of the different types of users served by the system. Developer-oriented requirements describe the system from a software developer's perspective. These requirements include a detailed description of functional, data, performance, and other important requirements.

1.3.Definitions, Acronyms, and Abbreviations

[Include any specialized terminology dictated by the application area or the product area.
For example:]

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the Restaurant Manager.
MySQL	Open-source relational database management system.
.HTML	Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create and run our application.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
Spring Web	Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system.
Thymeleaf	A modern server-side Java template engine for our web environment. This is one of the dependencies of our system.
NetBeans	An integrated development environment (IDE) for Java. This is where our system will be created.

API	Application Programming Interface. This will be used to implement a function within the software where the current date and time is displayed on the homepage.

1.4.Intended Audience

[Describe which part of the SRS document is intended for which reader. Include a list of all stakeholders of the project, developers, project managers, and users for better clarity.]

1.5.Project Scope

[Specify how the software goals align with the overall business goals and outline the benefits of the project to business.]

The goal of the software is to provide an easy-to-use interface for all customers, employees, and managers of a restaurant, as well as provide customers with flexibility to meet their needs. This aligns with the overall business goals of a restaurant as a restaurant requires fast and efficient service in order to fulfill the needs of its customers.

The benefits of the project to business include:

- Relieving stress and pressure from employees and managers as customers are given the opportunity to request services when needed.
- Increasing pleasure to customers as they are given more power when they want to order rather than having to wait for an employee to ask for their order.
- Reducing the amount of time that a customer needs to wait; therefore, increasing the amount of customers that are able to be served in the restaurant within a day.

1.6.Technology Challenges

[Any technological constraints that the project will be under. Any new technologies you may need to use]

1.7.References

[Mention books, articles, web sites, worksheets, people who are sources of information about the application domain, etc. Use proper and complete reference notation. Give links to documents as appropriate. You should use the APA Documentation model (Alred, 2003, p. 144).]

Alred, F., Brusaw, C., and Oliu, W. (2003). Handbook of Technical Writing (7th ed.). Boston: Bedford/St. Martin's.

2. General Description

2.1.Product Perspective

[Describe the context and origin of the product.]

Puzzle found its origins in a climber's desire for a simpler way to track their climbing progress. The idea was originated by a climber for climbers.

2.2.Product Features

[A high-level summary of the functions the software would perform and the features to be included.]

The product features include the ability for individual climbers and climbing gyms to create accounts and the ability for administrators to manipulate those accounts. Climbers can also add climbing routes to their

profiles, where they can track their climbing progress, with different tracking options based on climbing style. For gyms, the functionality also includes the ability to create climbing routes. They can also create events with a title and information. For administrators, the functionality also includes the possibility to view and delete accounts.

2.3.User Class and Characteristics

[A categorization and profiling of the users the software is intended for and their classification into different user classes]

Our website application does not expect our users to have any prior knowledge of a computer, apart from using a web browser, or knowledge of astronomy. Our website application has removed the need for them to have astronomy, math, or science knowledge and allows the user to focus on exploring the night sky.

2.4.Operating Environment

[Specification of the environment the software is being designed to operate in.]
The application is designed to operate on the web across many different devices.

2.5.Constraints

[Any limiting factors that would pose challenge to the development of the software. These include both design as well as implementation constraints.]

Due to the use of a 3d engine, we had to limit the web browsers supported. To limit user error when entering the user's address, we implemented a drop-down AJAX country, state, and city selection.

2.6.Assumptions and Dependencies

[A list of all assumptions that you have made regarding the software product and the environment along with any external dependencies which may affect the project]

The software will be dependent on Spring Web and Thymeleaf in order to create and execute the MVC architecture that will be developed within NetBeans. The application will also use the World Time API (http://worldtimeapi.org/) that will display the current date and time on the home dashboard for everyone to see.

3. Functional Requirements

[Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.]

3.1.Primary

[All the requirements within the system or sub-system in order to determine the output that the software is expected to give in relation to the given input. These consist of the design requirements, graphics requirements, operating system requirements and constraints if any.]

- FR0: The system will allow the user to lookup of vehicle owner information based on license plate number. This information will contain owner's permit number, assigned lot, and previous violations including tow history.
- FR1: The system will allow the user to enter a new vehicle into the vehicle violation database.

- FR2: The system will allow the user to issue a ticket. The ticket information will be issued in electronic and paper form.

3.2.Secondary

[Some functions that are used to support the primary requirements.]

- Password protection for information only accessible to employees, managers, and each individual table.
- Authorization scheme so that customers can only alter and see their orders and no other customers' orders.

4. Technical Requirements

1. Operating System and Compatibility

[The environments that will be needed to operate the system]

The application will be compatible with any operating system that is able to view and to interact with traditional web pages.

2. Interface Requirements

2.1.User Interfaces

[The logic behind the interactions between the users and the software. This includes the sample screen layout, buttons and functions that would appear on every screen, messages to be displayed on each screen and the style guides to be used.]

2.2.Hardware Interfaces

[All the hardware-software interactions with the list of supported devices on which the software is intended to run on, the network requirements along with the list of communication protocols to be used.]

The web application will run on any hardware device that has access to the internet, the ability to display webpages, and the ability to interact with web pages. This includes, but is not limited to, smartphones, tablets, desktop computers, and laptops.

2.3.Communications Interfaces

[Determination of all the communication standards to be utilized by the software as a part of the project]

It must be able to connect to the internet as well as the local database on phpMyAdmin. The communication protocol, HTTP, must be able to connect to the World Time API and return the current date and time.

2.4.Software Interfaces

[The interaction of the software to be developed with other software components such as frontend and the backend framework to the used, the database management system and libraries describing the need and the purpose behind each of them.]

We will use React and Spring Boot ThymeLeaf to help build the frontend, as well as JPA for the backend database functionality. We will also use Spring Boot with Java to connect the frontend to the backend.

2. Non-Functional Requirements

[Constraints on the services or functions offered by the system (e.g., timing constraints, constraints on the development process, standards, etc.). Often apply to the system as a whole rather than individual features or services.]

2.1.Performance Requirements

[The performance requirements need to be specified for all the functional requirements.]

- NFR0(R): The local copy of the vehicle violation database will consume less than 20 MB of memory
- NFR1(R): The system (including the local copy of the vehicle violation database) will consume less than 50MB of memory
- NFR2(R): The novice user will be able to create and print a ticket in less than 5 minutes.
- NFR3(R): The expert user will be able to create and print a ticket in less than 1 minute.

2.2.Safety Requirements

[List out any safeguards that need to be incorporated as a measure against any possible harm the use of the software application may cause.]

2.3.Security Requirements

[Privacy and data protection regulations that need to be adhered to while designing of the product.]

- NFR4(R): The system will only be usable by authorized users.

2.4.Software Quality Attributes

[Detailing on the additional qualities that need to be incorporated within the software like maintainability, adaptability, flexibility, usability, reliability, portability etc.]

2.4.1.Availability

[Details]

2.4.2.Correctness

[Details]

2.4.3.Maintainability

[Details]

2.4.4.Reusability

[Details]

2.4.5.Portability

[Details]

2.5.Process Requirements

2.5.1.Development Process Used

[Software Process Model]

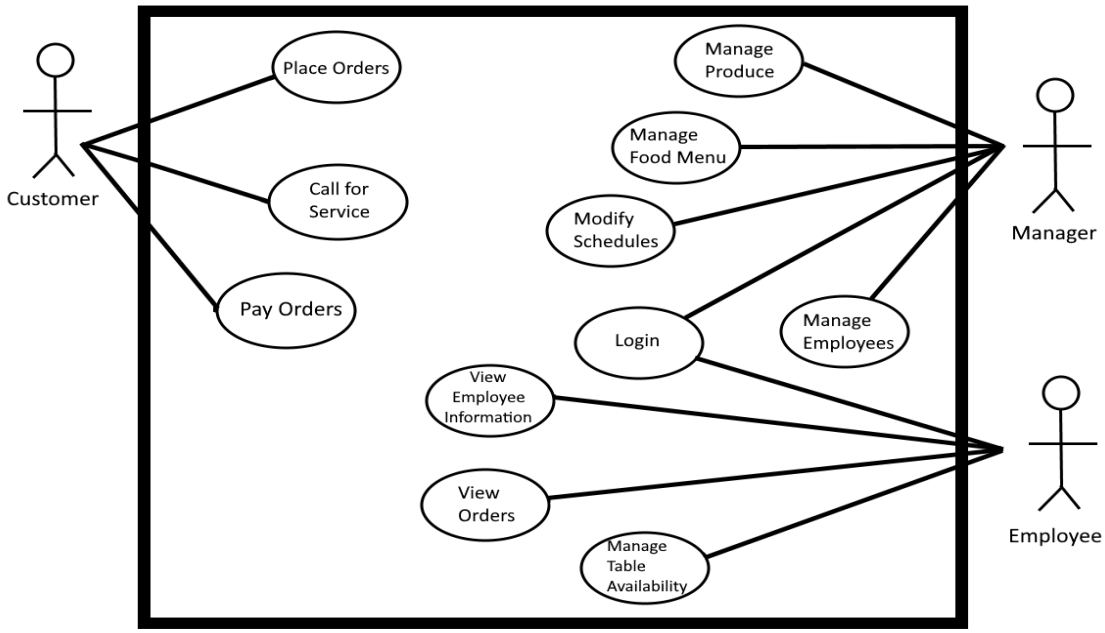
2.5.2.Time Constraints

2.5.3.Cost and Delivery Date

2.6.Other Requirements

TBD

2.7.Use-Case Model Diagram



2.8.Use-Case Model Descriptions

2.8.1.Actor: Actor Name (Responsible Team Member)

- **Use-Case Name**: [Brief Use-Case Description]
- **Use-Case Name**: [Brief Use-Case Description]

2.8.2. Actor: Actor Name (Responsible Team Member)

- **Use-Case Name**: [Brief Use-Case Description]
- **Use-Case Name**: [Brief Use-Case Description]

2.8.3.Actor: Actor Name (Responsible Team Member)

- **Use-Case Name**: [Brief Use-Case Description]
- **Use-Case Name**: [Brief Use-Case Description]

2.9.Use-Case Model Scenarios

2.9.1.Actor: Actor Name (Responsible Team Member)

- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**
    - **Other Activities**:
    - **System State on Completion**:
- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:
    - **System State on Completion**:

2.9.2.Actor: Actor Name (Responsible Team Member)

- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:
    - **System State on Completion**:
- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:
    - **System State on Completion**:

2.9.3.Actor: Actor Name (Responsible Team Member)

- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:
    - **System State on Completion**:
- **Use-Case Name**:
    - **Initial Assumption**:
    - **Normal**:
    - **What Can Go Wrong**:
    - **Other Activities**:

- **System State on Completion**:

6. Design Documents

6.1.Software Architecture

6.2.High-Level Database Schema

6.3.Software Design

6.3.1.State Machine Diagram: Actor Name (Responsible Team Member)

6.3.2.State Machine Diagram: Actor Name (Responsible Team Member)

6.3.3.State Machine Diagram: Actor Name (Responsible Team Member)

6.4.UML Class Diagram

7. Scenario

7.1.Brief Written Scenario with Screenshots