Software Engineering
Software Requirements Specification (SRS) Document

[SpartanJam]

2/13/2024

[1.1]

By: [Cade Flinchum, Sebastian Del Campo, Nehabahen Chauhan, Kanika Sun**]**

[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 our project is to allow more user interaction within the application. Unlike other music apps that only allow sharing things using third party apps, this will allow users to share music, share ideas, and other things with one another in the same application.

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 our project and the requirements we would need for SpartanJam (SJ). There are sections that describe what our project would include, the users, and what it would do. The requirements describe how we would do it, what we would use, our accomplishments and the things we struggled with.

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.]

The intended audience for the application is people who are attending the University of North Carolina at Greensboro, this is also intended for an audience who enjoy listening to music on the go, and for those who are interested in creating music and trying to look for a platform to post their songs.

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 allow user-user interaction within the same platform and allow for them to listen to music they like and share them with others. It is our goal to allow for the same platform communication. In hindsight, it would kinda be like a mini social media app for listeners.

The benefits of the project include:

- Listeners are able to share likable music to other users in a quick manner.
- Artists posting music and getting feedback from users and other artists.
- Reducing the amount of time that users have to go from website to website to hear about certain things pertaining to music.

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]

This is all of our first time using APIs so it's going to be a little bit difficult for all of us to learn from the beginning. Another technological constraint that might be difficult to implement is being able to create a website scratch from Java and also integrating it with internet capabilities.

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.

References:

2. General Description

2.1. 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 individuals and artists to create an account and as well as some sort of admin being able to manage those accounts. Users should be able to log in and listen to music, share music with other users, create forums and comment on theirs or other forums. Artists should be able to listen to music, share music, add their own music and comment on forums as well. The admin would be monitoring forum comments, as well as approving artist music.

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.

2.2. User Class and Characteristics

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

Casual users should just be able to know what music they want to listen to. They should be able to know how to use a regular music website. Artists should know how to upload music. Admins should understand what is appropriate for the music application.

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.

2.3. Operating Environment

[Specification of the environment the software is being designed to operate in.]

The application is designed to operate on the web across various devices that are able to run a web browser.

The application is designed to operate on the web across many different devices.

2.4. Constraints

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

Some possible constraints that might pose a challenge to the development of the software could be testing API might be unavailable from time to time if Spotify APIs Server were to go down. Another possible constraint might also include the necessity of access to the internet as this is an online platform if there is no stable connection to the site the program will not function.

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.

2.5. 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 the Spring Web in order to execute the web application which will be developed via NetBeans, it will also be dependent on the Spotify API as it will be receiving and gaining information for music and audio from their server.

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.]

- The system will allow the user to create an account or log in, depending on what type of user they are (Listener, Artist, Admin)
- The system will allow users to listen to music and search for songs.
- The system will allow users to share music with others on the same website.
- The system will allow users to create and comment on forums.
- The system will allow artists to upload music.
- The system will allow an admin to monitor artist music uploads and approve of them. They will also be able to monitor forums by deleting.

- 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

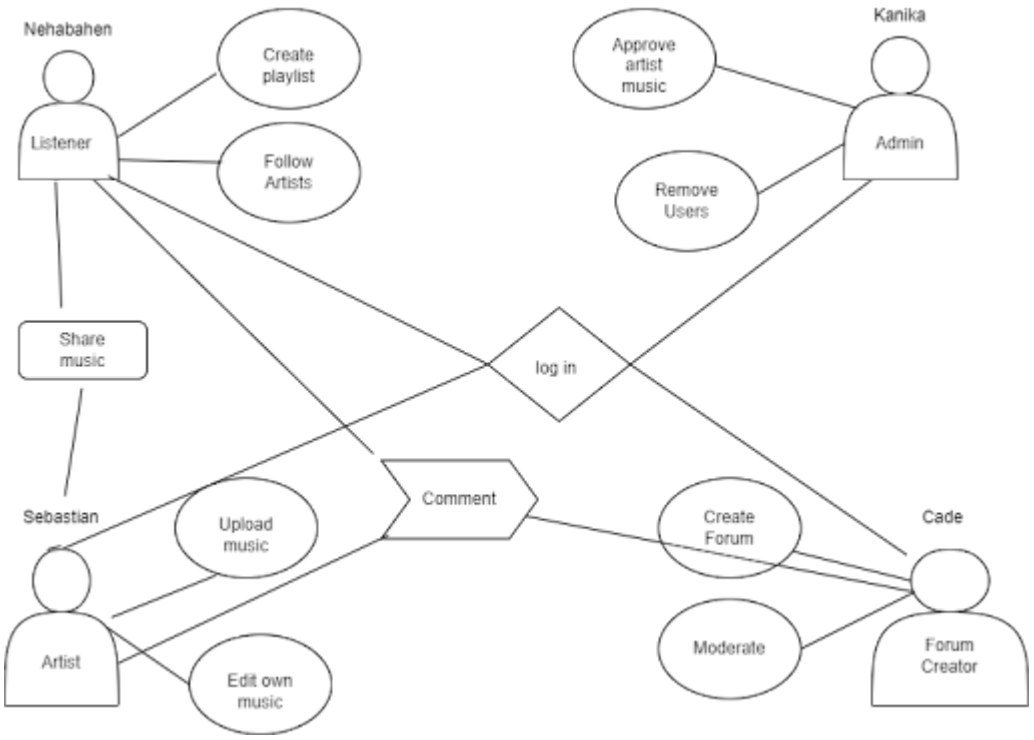
[Some functions that are used to support the primary requirements.]

- Passwords should only be accessible by individual account users.
- Only individual users should be able to see the music and artists they follow.
- Music sharing should only be kept between the sharer and receiver.
- Artists should be the only ones able to upload music to their account.
- Admins are the only ones able to monitor artist music.

- 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.

3.3. Use-Case Model

3.3.1. Use-Case Model Diagram



3.3.2. Use-Case Model Descriptions

3.3.2.1. Actor: Admin (Kanika)

- **Approve Artist Music**: Admins are able to approve artist music to see if its appropriate before allowing it to be uploaded onto the site.
- **Remove Users**: Admins are able to remove users off the site if there is a probable cause.

3.3.2.2. Actor: Form Creator (Cade)

- **Moderate**: The forum creator can delete messages and time people out for periods of time.
- **Create Forum**: The forum creator should be able to create forums.

3.3.2.3. Actor: Listener (Nehabahen)

- **Create Playlist**: Allows listener to create a playlist of songs that they like
- **Follow Artists:** Allows listener to search for and follow artists that they are interested in

3.3.2.4. Actor: Artist (Sebastian)

- **Upload Music**: Artists are able to upload their music onto the platform for other listeners to listen.
- **Edit Music**: Artists are able to edit their music by taking it down or reuploading the music.
- **Share Music:** Artist are able to share musics by getting a generated link which they can share to other platforms.
- **Login:** Artists will be able to have a login screen in which they can input their personal username and password which will access to their specific account.

3.3.3. Use-Case Model Scenarios

3.3.3.1. Actor: Listener (Nehabahen)

- **Use-Case Name**: Creating Playlists
    - **Initial Assumption**: Listener will be able to create his/her own favorite playlist.
    - **Normal**: Listener will create his/her own account and will interact with other features like shar, comment like the music.
    - **What Can Go Wrong:** When a user logs out, the playlist can be deleted.
    - **Other Activities**: add or delete the paylist
    - **System State on Completion**:The edit button should appear on the right of music name
- **Use-Case Name**: Follow Artists
    - **Initial Assumption**: Listeners are able to follow artists
    - **Normal**: Listeners should be able to search and follow artists they find interest in
    - **What Can Go Wrong**: The followed artist doesn't appear in their following.
    - **Other Activities**: n/a
    - **System State on Completion**: The following button should appear on the client side view.

3.3.3.2. Actor: Forum Manager (Cade)

- **Use-Case Name**: Moderate
    - **Initial Assumption**: The forum manager has access to the webapp and has the forum manager role.
    - **Normal**: The forum manager will have a special moderator view that will allow them to perform their duties.
    - **What Can Go Wrong**: n/a
    - **Other Activities**: The forum manager should be able to delete messages and time users out for a period of time.
    - **System State on Completion**: The forum manager should have deleted a message or timed a user out.
- **Use-Case Name**: Create Forum
    - **Initial Assumption**: The forum manager has access to the webapp and has the forum manager role.
    - **Normal**: The forum manager will be able to create a forum with specific details (for example, a name for the forum).
    - **What Can Go Wrong**: The forum manager could create two forums with the same name.
    - **Other Activities**: n/a
    - **System State on Completion**: The new forum is created and accessible to all users.

3.3.3.3. Actor: Artist (Sebastian Del Campo)

- **Use-Case Name**: Upload Music

        - **Initial Assumption**: Artists will upload an audio file of their choice which will be reviewed by the moderator and if approved will be shared on the forum.
        - **Normal**: The artist will upload an audio file into a submission box which will then be sent to the moderator for review before being uploaded onto the website.
        - **What Can Go Wrong**: A possible security threat is if a malicious file gets sent, the file might be corrupted or can't be opened on the moderator's side.
        - **Other Activities**: Create a specific page that contains an audio file along with comments and possible connections to forums.
        - **System State on Completion**: The audio file will be denied by a moderator and will need to fix the audio and resend it for approval or can be approved and get a specific link to a page that contains their audio file along with audio player options, sections to comment, and possible links to forums.
    - **Use-Case Name**: Edit Music
        - **Initial Assumption**: Artists can go into their specific audio file page and edit various data of the music. This will also need approval from moderation.
        - **Normal**: The artist can click a button to edit a specific audio file page which will open a new page where they can change the data of the audio file. After edits are made it will be sent to moderators to approve changes.
        - **What Can Go Wrong**: Possibility to upload malicious data, edits may produce bugs, and edits may be lost if gone unsaved.
        - **Other Activities**: The audio file page will be updated with edit if approved by the moderator
        - **System State on Completion**: The audio file will be denied by a moderator and will need to fix the audio and resend it for approval or can be approved and audio file page will be updated with new changes.
    - **Use-Case Name**: Share Music
        - **Initial Assumption**: Artists can get various methods to share their audio files with other users whether on the websites or on outside platforms
        - **Normal**: The artists get a link or data that allows them to share their audio page to other users on the SpartanJam platform or on outside platforms such as social media.
        - **What Can Go Wrong**: The link may be broken, and the link may be denied/marked as spam on websites if unrecognized.
        - **Other Activities**: A section within the audio file page will appear giving out various options to share the audio file across SpartanJam or on other platforms
        - **System State on Completion**: The artists will have a special link/data that will allow them to share their audio file to different users across the internet.
    - **Use-Case Name**: Login

- **Initial Assumption**: Create a security measure for people to access a certain account.
- **Normal**: Login can be created by first signing up where you will enter a username and then be prompted to enter a password to access an account
- **What Can Go Wrong**: If someone else has access to the user login information is that they will be able to use the account without the user permission, login may also be forgotten permanently leading to an abandoned account.
- **Other Activities**: Provide a method to register a specific username and password to create a specific account (Forum Creator, Artist, Listener, Admin)
- **System State on Completion**: You are able to access an account associated with a username that was inputted.

3.3.3.4. Actor: Admin (Kanika)

- **Use-Case Name**: Approve Artist Music
  - **Initial Assumption**: Admin can approve artist music.
  - **Normal**: When an artist uploads new music, has to go through a process where the admin approves of music.
  - **What Can Go Wrong**: The music doesn't reach the admin for approval.
  - **Other Activities**:
  - **System State on Completion**: Artist music uploads.
- **Use-Case Name**: Remove Users
  - **Initial Assumption**: Admin is able to remove users or artists on the platform.
  - **Normal**: Admin has the ability to delete a user.
  - **What Can Go Wrong**: Deletion of the user doesn't occur.
  - **Other Activities**:
  - **System State on Completion**: User removed.

4. Technical Requirements

4.1. Interface Requirements

4.1.1. 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.]

Users should be directed to a login page upon reaching the website. After logging in or creating an account, the user would have multiple tabs on screen available to them such as Search, Home, Profile, Forum, etc. Clicking on the search tab should give the user the option to search up artists or music they find interest in and visit the pages for them. The homepage should provide the music they saved and artists they follow. The Profile tab should give users the option to display things on their profile such as a bio or the ability to log out. The forum tab would have a collection of forums from the community that the user should be able to read and comment onto.

4.1.2. 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 should be able to run on computers and laptops that have access to the internet and can interact with websites.

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.

4.1.3. 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.

It must be able to connect to the internet. The communication protocol must be able to connect to the Spotify API.

4.1.4. 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.

- Spring Boot will be used in order to develop the website

- Spotify API will implement the Spotify Software which will be a software component towards the project

5. 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.]

5.1. 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.

- NFR0(R): The local copy of the Spotify Music API may consume less than 15 MB of memory
- NFR1(R): The system (including local copy of Spotify Music API) may consume less than 40 MB of memory
- NFR2(R): The novice regular user will be able to create an account and listen to music in less than 15 minutes.
- NFR3(R): The expert regular user will be able to create an account and listen to music in less than 5 minutes.
- NFR4(R): The novice artist will be able to create and post music onto the platform in less than 30 minutes.
- NFR5(R): The expert artist will be able to create and post music onto the platform in less than 10 minutes.

5.2. 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.]

- Security Login for Website maintenance
- Cybersecurity to prevent from cyber attacks (ie ddos attack)
- Moderation Team to prevent unwanted/malicious files from entering into website.

5.3. 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.

- NFR6(R): The system will only be used by authorized users
- NFR7(R): Any materials use in the project will be adhered to copyright laws

5.4. Software Quality Attributes

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

5.4.1. Availability

[Details]

5.4.2. Correctness

[Details]

Peer reviews and team members reviewing each others implementations will help with maintaining correctness in our software.

5.4.3. Maintainability

[Details]

5.4.4. Reusability

[Details]

5.4.5. Portability

[Details]

5.5. Process Requirements

5.5.1. Development Process Used

[Software Process Model]

5.5.2. Time Constraints

5.5.3. Cost and Delivery Date

5.6. Other Requirements

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