Software Requirements Specification

for

Space Jam Music Application

**Version 1.0 approved**

**Prepared by Alex Shiffer, Hunter Gardner, Nicole Ruppert, Jacob, Kawther Aldashti, Cameron Stark**

**Team 5: Space Jammers**

**October 5, 2018**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 Login

4.2 Choose Location

4.3 Search/Select Music

4.4 Purchase Music

4.5 Voting System

4.6 Rewards System

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

This document contains information on the Space Jam web application. The intention of this document is to provide an initial vision for the reader of the intentions of making this web app. As well as giving the reader an idea of what this project is being developed into, it will give the team a list of features to be completed.

## Document Conventions

No specific conventions were used in the making of this document.

## Intended Audience and Reading Suggestions

This document is made up of the different front end, back end, hardware requirements and expectations for the the project. This is a resourceful document for the Space Jam Team to be able to track progress and for the Professor to see what the scope of the project.

## Product Scope

Space Jam is a web application that allows a user to engage in a community completely centered around sharing music. The user can post songs for a certain amount of time in a particular area for others to listen to if they are in that area. The user can also listen to music others have posted. Through a system of earnable credits, the user can buy songs to add to their collection for others to listen to. Through community interaction we want people to find new songs and new people that have similar taste to them.

## References

No References have been used so far.

# Overall Description

## Product Perspective

People often listen to music when walking from place to place or when in in a certain location. This product came from the inspiration of listening to music others have curated for a particular location. An example being most people pick uplifting music for the gym, so when a user is in the gym they can listen to music they have selected as well as songs that others have picked for that gym. Songs only stay for around 24 hours and the user only has a limited number of songs they can place per day. Not all songs are unlocked initially though. The progression system of “collecting” songs of inspiration from the popular mobile game Pokemon Go, where the user tries to collect many creatures. In Space Jam, the user can collect songs by listening to a song a certain number of times. Alternatively, the user can earn credits called Music Secret Stuff (MSS) by listening to songs, having their song listened to, and for their song being given a positive review by another user.

## Product Functions

* User can log in using Spotify.
* User can place a song on their location.
* User is able to listen to a song that is at their location.
* User can search songs to be added to their account.

## User Classes and Characteristics

*User*

Listens to music using the app and adds music for others to use.

## Operating Environment

Any up-to-date web browser that has access to the internet.

## Design and Implementation Constraints

* Most of development team is new to Javascript
* Completely dependent on getting the Spotify API

## User Documentation

UD-1: There will be a help button with tips/FAQ.

## Assumptions and Dependencies

AS -1: There will be multiple users that are placing songs and listening to songs placed by others.

DE-1: The Spotify API’s that are implemented do not change.

# External Interface Requirements

## User Interfaces

The User Interface connects to the controller of the site, which based on the button (function) clicked, the controller can either send information to the database [explained in section 3.3], send/receive information from the spotify API [explained in section 3.3], and/or send/receive information to the front-end of the site for the user to see.

**3.1.1 Spotify Login Button**

The user clicks the login button, and the program makes a request to the Spotify login API, launching a new window, with the Spotify login and/or account set up, if the user does not have a Spotify account. Once user gets logged in, they are returned to the Space Jam site, with all of their user information populating the site.

[Software process explained more in section 3.3]

**3.1.2 Music Player Functionality**

The user can view the current song playing, and the progression of the song playing.

**3.1.2.1 Play/Resume Button**

Upon user selection, the application will pause if the song was previously playing will resume or it will get the track information from Spotify and begin playing the song.

**3.1.2.2 Pause Button**

Upon user selection, the application will pause the song and store the timestamp for the song, so that it could be resumed

**3.1.2.3 Like/Dislike Button**

Upon user selection the program will take the action selected and apply it to the original poster of the songs account and increment the value for rating in their database

**3.1.3 Location Selection**

The user can select where they want to drop a song, this takes the selected location, song and user and sends it to the location database so that other users can listen to the song. The other side is a user can enter a location and the application detects that they are in a location with songs and plays the songs attached to that location

**3.1.4 Song Selection/Navigation**

The user can select a song by searching by artists, album, genre, song [Explained in 33], which is displayed to the user with additional options to that are controlled by the software and communicated to the database.

## Hardware Interfaces

**3.2.1 Audio Control**

When the user plays a song from their selection or enters a zone with songs present, the application will make a request to Spotify and will be returned with the song and begin playing the song from the beginning by making use of the devices speakers.

**3.2.2 Location Detection**

The program will gain access to the user devices IP address and then cross reference that with any known IP address in the database to determine if there are songs in that area.

## Software Interfaces

**3.3.1 Front-End**

The Front-End is built using the Bootstrap CSS, JS, and JQuery framework version 4.1.3, this allows the website to look modern and behave modern with very little effort on the developers part, and will allow the application to reach those on mobile devices since it can scale properly to smaller screens. The Front-End also makes use of Google’s AngularJS version 1.7.4, this framework allows for advanced front-end manipulation and interpolation, but also provides a diverse and powerful Back-End to the site [Explained in section 3.3.2].

**3.3.2 Back-End**

The Back-End will be built using AngularJS and using a Library called ALASql, version 0.4.11, that allows AngularJS to connect to a Database created from Excel Files, Google Sheets and other alternatives to an actual Database. This application will making use of the Google Sheets tooling so that they can serve as a database.

**3.3.2 Database**

The database will be making use of Google Sheets where each document will be acting as a “table” in a normal database and will be using ALASql to connect and perform any normal database operations.

**3.3.3 Music Database**

The music database will be Spotify, making use of there Spotify API and their Web Playback SDK, to allow the site to search and play songs. The database will store the songs name or the given Spotify ID for the song and when needed will send that to the Spotify API to return the selected song.

## Communications Interfaces

The Application will communicate with the Spotify API and Google Sheets over an HTTP connection making use of each their own REST API’s, where the application will send GET and POST connections to each. Encryption is handled by the spotify API and is not a concern for the application. All data will be returned and sent in JSON format for easy reading and posting.

# System Features

## Login

4.1.1 Description and Priority

The screen will prompt the user to login using their Spotify account. The Spotify API requires that users have a Spotify account to access the music database, and it is for this reason that this feature is of a high priority.

4.1.2 Stimulus/Response Sequences

Stimulus: User opens the application and selects the login button.

Response: Application requests user account information.

Stimulus: User enters valid account information.

Response: Application verifies with user database and allows the user access to the features of the application.

*If the user enters invalid information*

Stimulus: User enter invalid account information.

Response: Application declines access and prompts the user to login again. This repeats until the user has entered valid information.

4.1.3 Functional Requirements

REQ-1: Application shall prompt user for their Spotify login.

-Invalid information shall be caught and the application will prompt the user to enter the information again.

## Choose Location

4.2.1 Description and Priority

The application shall be able to read the user’s location and generate a map of locations that music can be played from. The user can then select a location to add music to. This feature is a high priority since the basis of the application is to play music in a certain location.

4.2.2 Stimulus/Response Sequences

Stimulus: User allows the application to access their location.  
Response: Application generates a map of location that the user can add music to.   
Stimulus: User selects a location to add music to.  
Response: Application confirms selection and moves to music selection.

4.2.3 Functional Requirements

REQ-1: The application shall be able to find locations near the user’s location that have music posted to them/can have music posted to them.

REQ-2: The application shall allow the user to select a location.

## Search/Select Music

4.3.1 Description and Priority

Using the Spotify API, the user shall be able to search and select music from a library of previously captured or purchased songs. Once a song is selected, it will be applied to the playlist in the user’s selected location. This is one of the main features of the application, therefore it of a high priority.

4.3.2 Stimulus/Response Sequences

Stimulus: User has successfully selected a location.  
Reponse: Application prompts the user to search for music OR to select music from the user’s library of captured/purchased music.   
Stimulus: User selects a song.   
Response: Application adds the selected song to the user’s selected location.

4.3.3 Functional Requirements

REQ-1: The application shall allow the user to search for music through Spotify.

REQ-2: The application shall allow the user to select music from their Spotify library of captured/purchased songs.

REQ-3: The application shall apply the selected music to the chosen location.

## Purchase Music

4.4.1 Description and Priority

Purchasing songs is one out of the two ways a user can acquire music to add to a location. Using point gained, the user can select a song of their choosing and purchase it for the appropriate amount of points. Because users can aquire song in other ways, this feature is of a medium priority.

4.4.2 Stimulus/Response Sequences

Stimulus: User requests to search for music.   
Response: Application prompts the user to search for music within the Spotify API.   
Stimulus: User selects a song for purchase.   
Response: Application deducts the appropriate amount of points from the user’s account and the song is added to the user’s library.

4.4.3 Functional Requirements

REQ-1: The application shall allow the user to use points to purchase music.

REQ-2: The application shall add the purchased song to the user’s library.

## Voting System

4.5.1 Description and Priority

In the music player, the user has the option to like a song or to dislike a song. A like or dislike directly affects the account of the original poster of the song. Because this is how user can gain points, this feature is of medium priority.

4.5.2 Stimulus/Response Sequences

Stimulus: User selects the like/dislike button while in the music player.   
Response: Application adds points to the original posters user account if the song is liked. If the song is disliked, the music player will skip to the next song in the playlist.

4.5.3 Functional Requirements

REQ-1: The application shall allow the user to like/dislike songs.

REQ-2: The application shall skip a song if the user dislikes it.

REQ-3: The application shall add points to the original posters account if the song they posted is liked.

## Rewards System

4.6.1 Description and Priority

When a user has listened to a specific song a certain amount of times, the song will become part of their permanent music library along with other purchased songs.

4.6.2 Stimulus/Response Sequences

Stimulus: User listens to a specific song a certain amount of times.  
Response: The song becomes “captured” and is added to the user’s library of music as a permanent song.

4.6.3 Functional Requirements

REQ-1: The application shall track how many times a user has listened to a specific song.

REQ-2: The application shall add the song to the user library if the user has listened to is a certain amount of times.

# Other Nonfunctional Requirements

## Performance Requirements

*The performance of Space Jam Music Application shall be able to maintain functionality while under a minimum load scenario. Space Jam Music Application shall be able to play songs posted to the current zone to all users using the application in that designated zone. Space Jam Music Application shall be able to handle multiple songs posted to the same selected zone.*

## Safety Requirements

*For the use of the Space Jam Music Application, no songs shall be able to be posted to any location that is hazardous or dangerous to users. Space Jam Music Application will notify users attempting to post songs in hazardous zones that their song cannot be posted in the selected zone due to safety.*

## Security Requirements

*This application will require users to login to have access to the application. Due to this application primarily running off of Spotify API, the users will need to have a Spotify account to use the Space Jam Music Application. Spotify requires its users to login to their accounts to have access to their content. The security requirements for Space Jam Music Application will be satisfied by the Spotify login credentials. User identity authentication will also be legitimized through the Spotify login process.*

## Software Quality Attributes

*Space Jam Music Application shall be available whenever a user opens the application unless Space Jam Music is under scheduled maintenance. Space Jam Music shall be correct in spelling, grammar, and ethics. Space Jam Music shall be maintainable for the developers by following the specified software structure. Space Jam Music shall be portable for users to use on their smartphone. The Space Jam Music Application shall be adaptable as to accommodate potential changes to any or all of the features on the application. The Space Jam Music Application shall be testable by providing clear, concise, and quantifiable pass/fail criteria. The Space Jam Music Application shall be made with usability that is easy for the user to understand modeling similar music playing applications.*

## Business Rules

*For Space Jam Music Application, a user shall be able to manage their user account at anytime. Users shall also have the ability to use their MSS points to purchase songs for their library. Users shall have the ability to choose an area to post a song to. Space Jammers Corporation (SJC) retains the right to delete user accounts for misuse. Space Jammers Corp. also retains the right to change the terms and conditions of this application. SJC is not responsible for injuries acquired through the use of this application. By using this application, users agree that SJC is not responsible for the actions of any user on the application. Furthermore, SJC does not retain or distribute any personal, confidential, private or otherwise sensitive information of any user.*

# Other Requirements

*There shall be a MP (Music Point) for each user whose songs get played more than once or for each time the user’s song is played in a selected zone. These points will be given for each user.*

**Appendix A: Glossary**

**Definition**

**User:** A person who uses a service to obtain benefits.

**Spotify:** An online music streaming service.

**Zone:** An area which the songs be played upon

**Front-end:** Is what the user sees, for example (GUI, buttons, transitions..)

**Abbreviation**

**API:** Application Program Interface.

**MP:** Music Points.

**MSS:** Music Secret Stars.

**REQ:** Requirement.

**SJC:** Space Jammers Corporation.

**GUI:** Graphical User Interface

**Appendix B: Analysis Models**

4

**Appendix C: To Be Determined List**

*To be determined..*