Software Design Specification

for

Space Jam Music App

**Version 1.0 approved**

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

**Team Space Jammers**

**October 17, 2018**

**Table of Contents**

**Table of Contents**

**Revision History**

**1. Introduction - Hunter/Alex**

1.1 Purpose

1.2 Scope

1.3 Document Overview

1.4 Definitions

1.5 References

**2. System Overview - Hunter**

**3. High Level Design - Cameron/Alex**

**4. Low Level Design -Cameron/Alex/Kawther**

**5. User Interface -Jacob/Nicole/Kawther**

**6. Requirements Matrix – Nicole**

**Appendix A: Glossary - Kawther**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Initial Draft | 10/17/18 | Initial Creation of Document | 0.0 |
|  |  |  |  |
|  |  |  |  |

# 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.

## 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 people can find new songs and people that have similar taste in music.

## Document Overview

This document discusses the functional design of Space Jam.

## References

No reference material was used for this document

# System Overview

Space Jam is a web application that allows the user to listen to music that is determined by the user location. This works through the mechanic of users placing a song from their library at a certain location for other users to listen to it, the catch is a user has to be at that location to place it. The songs that are placed will stay in that location for a certain amount of time. If another user listens to a placed song, they earn one Music Secret Stuff (MSS) as well as the user who placed the song. This incentivizes users to place songs in locations that have high traffic so themselves and other users will earn as many MSS as possible. A user can add songs to their library by either purchasing it outright with MSS or if they listen to a song a certain amount of times.

# High Level Design

3.1 Introduction

This section describes the high-level functionality of Space Jam Music Application. All general functions and modules in the application will be explained in this section as well as how they interact with each other.

3.2 Login

The login module handles the security of the Space Jam Music Application. Prior to each user session, users will be prompted to login to their Spotify account to use the Space Jam Music App. Once the user login credentials have been satisfied by Spotify, the user will be directed to the Space Jam home page.



3.3 Choose Location

This function will allow the user to select a location at which a song can be posted. Once a location has been selected, the location is saved and the system waits for the Select Music function to be completed. The Location module handles where the user is and what songs are accessible/available.



3.4 Search Music

This module will be responsible to searching the Spotify database and the user’s library for songs specified by the user. Song searches can be done by entering the desired song title, artist, or album which will go through the Spotify database. Searches can also be conducted through the user’s library which will only show the songs that user has purchased or collected on the SJM application.



3.5 Post Song

This module is responsible for getting the user’s specified location and selected song from the Choose Location module and the Search Music module. Once this information is retrieved, the song will be posted to the selected location.



3.6 Voting System

The voting system is responsible for tracking the number of likes and dislikes for each songs posted to a selected location. Once a posted song has timed out, the final voting score is sent to the postee’s MSS for point collection.



3.7 Capture/Purchase Music

The capture music module is responsible for adding liked music to the listener’s library. This allows users to add music to their posting library without spending their MSS on music. This module is responsible for allowing users to spend their MSS points on searched music. Once a song has been purchase by the user, the song will be available to the user for posting in their library.



# Low Level Design

**4.1 Introduction**

The application broken down at the low level is broken up into several Controllers and Services for each major function of the application. The Controllers serving as the functionality ann d interpolation providers for the User Interface and front-end of the application. The Services serving as the portion of the application that communicates outside of the application, for example to spotify or the google sheets database.

**4.2 Controllers**

**4.2.1 AppController**

Serves as the controller for determining if the the user is logged in and controlling whether the main view or the login view are visible. Communicates with the API and Authentication Services.

**4.2.1.1 checkUser - Function**

Function that checks if the user is currently logged in, if true do nothing, else redirect the user to the login screen and prompt for relogin. Communicates with the API to get the current users information.

**4.2.1.2 ‘on’ login - Function**

Function that listens for the login message to be emitted, if heard shows the main view and hides the login view.

**4.2.1.3 ‘on’ logout - Function**

Function that listens for the logout message to be emitted, if heard shows the login view and hides the main view.

**4.2.2 LoginController**

Serves as the controller for logging, listeners for the login button to be clicked. On clicked the function calls a function in the authentication service that starts the Spotify login.

**4.2.2.1 login - Function**

On login function activated fires the spotify api to start the authentication process which occurs under the control of spotify and occurs in a separate window, after login returns to the main view.

**4.2.3 PlayerController**

Serves as the controller for the main view application and controls the spotify music playing capabilities. Player Auth, API, Playback, and Play Queue services, which control the spotify actions from login to the song searching and song playing.

**4.2.3.1 ‘on’ login - Function**

On user login gets the users username and gets there location.

**4.2.3.2 Play - Function**

Starts the music playing process by connecting to the Play Queue Service.

**4.2.3.3 Logout - Function**

Function on activation sets the username and access token to empty and emits the logout message.

**4.2.4 DatabaseController**

Controls the database interaction between the user interface and the spotify API.

**4.2.4.1 checkUser - Function**

Function to when the user completes the login function, the function will check if the user already exists in the database, if the user does not, it will add the user and give the user the initial amount of MSS.

**4.2.4.2 getUser - Function**

Function that returns all of the current signed in users information for the user interface to display.

**4.2.4.3 getLocations - Function**

Function to get the list of locations with songs in them and songs in the location.

**4.2.4.4 setLocations - Function**

Function to receive the user selected location and song, and add to the location table.

**4.2.4.5 addSong - Function**

Function to add a new song to a users account when purchased or collected from a location.

**4.2.5 LocationController**

This controller controls the sending and receiving of the location of the user.

**4.2.5.1 getLocation - Function**

This function gets the current location of the user and checks using the Database controller if songs exist in the location, if does returns the songs to Play Queue Service and plays the songs.

**4.2.5.2 setLocation - Function**

This function gets the users current location and selected song and assigns it to the location using the Database controller.

**4.3 Services**

**4.3.1 API Service**

This service serves as the direct contact to the Spotify API.

**4.3.1.1 getMe - Function**

Function gets and returns the current logged in user account information and stores the username in local storage for the currentuser session.

**4.3.1.2 getTrack - Function**

Function gets and returns the track data for a specific song searched by name or trackid, and returns the trackdata.

**4.3.1.3 getTracks - Function**

Function gets and returns the track data for an array of songs by name or trackid, and returns an array object for the tracks.

**4.3.2 AuthService**

Service serves the login and authentication, on creation, which is when the login process begins, and open the login process for spotify.

**4.3.2.1 getLoginURL - Function**

Function gets and returns the URL for the client and creates a full url to give to the openlogin function.

**4.3.2.2 openLogin - Function**

Function opens a new to allow the user to create or login into a spotify account and upon successful login the user is returned to the application.

**4.3.2.3 getAccessToken - Function**

Function allows the program to get access to the locally stored access token for the user which allows the program to verify that the user has previously logged within the allotted time.

**4.3.2.4 setAccessToken - Function**

Function sets the access token received from the authentication service and sets a expiration time, and stores in local storage

**4.3.2.5 getUsername**

Function gets the locally stored username and returns it for the application to use.

**4.3.2.6 setUsername**

Function sets the username from the authentication service and stores locally.

**4.3.3 PlaybackService**

Serves as the main music playing service, creates the audio component and controls the playing, pausing, and volume.

**4.3.3.1 createAndPlayAudio - Function**

Function to create the audio component with the track URL, length, and volume information.

**4.3.3.2 getVolume - Function**

Gets the current volume amount and returns to for use in the user interface.

**4.3.3.3 setVolume - Function**

Sets the current volume amount received from the user interface

**4.3.3.4 startPlaying - Function**

Function starts the audio component to start playing the song, after going through the API component to get the song data.

**4.3.4 playQueueService**

Services serves as the communication for the playback services and creates a queue of songs and plays the queue

**4.3.4.1 Play - Function**

Function clears the queue and adds the song to the queue then plays the song by access the playback start playing function.

# User Interface

**5.1 Introduction**

The Space Jam Music Application has a very concise and easy to follow interface that has a slide-out menu to allow the user to focus on the music player, the location map, and music search features without the screen being too crowded with links to other features or other less important sections. The user can open the menu easily by clicking on the three lines at the top-left hand corner of the main screen. From there, the user can see their username and amount of points. They may also choose to log out, access their library of purchased or “captured” songs, and see nearby songs.

The first screen that the user will encounter will be the login screen. After selecting “Connect Spotify”, a second window will open that prompts the user to login to their Spotify account or to login to Spotify via Facebook. This will be verified using the Spotify API. Afterwards, the second open window will close and the user will be automatically brought to the application homepage, where they will see the music player, the location map, and music search features.

**5.2 User Interface Functional Overview**

|  |  |
| --- | --- |
|  | When the user accesses the website with a link or by typing in the websites address, they will first be brought to this screen. This welcomes the user to the website and asks them to login to their Spotify account. To do so, they will have to click the “Connect Spotify” button. Below the button, there is a disclaimer, letting the user know that by logging into this website with their Spotify account, they agree to Spotify’s “Terms & Conditions”. |

|  |
| --- |
|  |
| After logging in this is the first page the user sees. This segmented into three sections: the toolbar, map view, and music playbar. |

|  |  |
| --- | --- |
|  | This is the toolbar section. At the top of this section it displays the user’s username, music points, and allows the user to log out. Under that it displays the nearby songs in the area. Under nearby songs it displays: song title, artist, the distance away the song is, and the song’s album cover. The next part is the shop button, this brings up the shop where the user can purchase songs using their music points. The setting button brings up the settings and the help button displays a help box. |

|  |
| --- |
|  |
| On the shop page, the user can buy music for their library using their music points by clicking on the song of their choice. |

|  |
| --- |
|  |
| This is the music playbar. It displays the current song playing and its artist. It also displays the runtime of the song using the bar. The user can play/pause and vote for this song in this section. The music also displays who placed the song in the lower right. |

|  |
| --- |
|  |
| In the map view the user is able to see where they are and where songs are placed. Users can search for songs using the search icon. |

# Requirements Matrix

|  |  |
| --- | --- |
| 4.1.3 REQ - 1 |  |
| 4.2.3 REQ - 1 |  |
| 4.2.3 REQ - 2 |  |
| 4.3.3 REQ - 1 |  |
| 4.3.3 REQ - 2 |  |
| 4.3.3. REQ - 3 |  |
| 4.4.3 REQ - 1 |  |
| 4.4.3 REQ - 2 |  |
| 4.5.3 REQ - 1 |  |
| 4.5.3 REQ - 2 |  |
| 4.5.3 REQ - 3 |  |
| 4.6.3 REQ - 1 |  |
| 4.6.3 REQ - 2 |  |

**Appendix A: Glossary**

## Definitions and Acronyms

Poster: The owner of the posted song

Listener: Users within the designated radius of posted songs currently listening to a song

Spotify: An online music streaming service

Track: A piece of music

SJM: Space Jam Music

API: Application Program Interface

MSS: Music Secret Stuff

UI: User Interface

AUTH: Authentication

Capture: A user can capture a song by going to a location and listening to songs

MP: Music points

REQ: Requirements