**Green SP-27 Spotify Mobile App**

**CS 4850-01, Section 01, Fall 2024**

**Sept 1, 2024**

|  |  |
| --- | --- |
| Marc Gagnon  Developer | Muhammad Saad  Documentation |
| Brian Okonji  Developer | Jace Windbigler  Team Lead |

**Team Members:**

|  |  |  |
| --- | --- | --- |
| Name | Role | Cell Phone / Alt Email |
| Muhammad Saad | Documentation | 678-629-6373  [Muhdsaad2003@gmail.com](mailto:Muhdsaad2003@gmail.com) |
| Marc Gagnon | Developer | 678-316-0327  [Chewie83@gmail.com](mailto:Chewie83@gmail.com) |
| Jace Windbigler | Tester | 912-227-8887  [Jacewindbigler@gmail.com](mailto:Jacewindbigler@gmail.com) |
| Brian Okonji | Developer | 404-518-7861  [Brianokonji9@gmail.com](mailto:Brianokonji9@gmail.com) |
| Sharon Perry | Project Owner or Advisor | 770.329.3895  [Sperry46@kennesaw.edu](mailto:Sperry46@kennesaw.edu) |

Spotify Mobile App

SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

CS 4850 Senior Project

Fall 2024

Professor Perry

9/01/2024

SP-27 Green Spotify

Table of Contents

[1.0 Introduction 2](#_Toc818956999)

[1.1 Overview 2](#_Toc933933294)

[1.2 Project Goals 2](#_Toc1549673112)

[1.3 Definitions and Acronyms 2](#_Toc947372498)

[1.4 Assumptions 2](#_Toc465410183)

[2.0 Design Constraints 3](#_Toc1325573474)

[2.1 Environment 3](#_Toc1927245799)

[2.2 User Characteristics 3](#_Toc1431411479)

[2.3 System 3](#_Toc1111428961)

[3.0 Functional Requirements 3](#_Toc819310274)

[4.0 Non-Functional Requirements 5](#_Toc385623287)

[4.1 Security 5](#_Toc405341990)

[4.2 Capacity 5](#_Toc1133923737)

[4.3 Usability 5](#_Toc585357156)

[4.4 Other 5](#_Toc26904995)

[5.0 External Interface Requirements 5](#_Toc692818419)

[5.1 User Interface Requirements 5](#_Toc1832786703)

[5.2 Software Interface Requirements 5](#_Toc385952064)

# Introduction

## 1.1 Overview

The project’s purpose is to develop a custom mobile application using the Spotify API that enhances the features of Spotify’s native application. The app will be developed in React Native framework with full compatibility for iOS and Android devices. The key functionality we will focus on is to generate, edit, and control in several ways the use of playlists, recommended songs list, and queues through sorting songs by the song’s tempo. The design process of the screen will include designing a prototype using Figma which will give us an interface that closely relates to what we want the final product to look like. At the end of the project, we aim to have a functional customized Spotify mobile application that will further enhance the user experience with features that were not offered before.

## 1.2 Project Goals

* Functional prototype design with new functionalities of the Spotify app that allows the user to generate, edit and manage playlists and queues based on the songs tempo or activity doing.
* Our project will have cross platform compatibility which is why we’re using react native to build the app.

## 1.3 Definitions and Acronyms

* API: Used to access and manipulate Spotify’s music data
* SDK: Deeper with Spotify’s services to be able to play the songs
* IDE: IntelliJ for software Development
* React Native: Framework to be used to build apps that are cross-platform compatible
* Figma: Used to create prototype that closely resembles our final product design
* VCS: Git will be used to manage and track code development progress

## 1.4 Assumptions

* Spotify API: Should be stable and reliable throughout our development process
* User Authentication: Spotify App should be able to Authenticate users after getting redirected for authentication
* Cross Platform Compatibility: Should be compatible with both iOS and Android devices

# 2.0 Design Constraints

* API Limitation: Any changes to the API by Spotify will disrupt our App performance
* Rate Limit: Spotify imposes a rate limit, so we must make sure to not make too many API requests in a brief period.
* Testing constraints: Testing and debugging issues that may arise will be continuous, which may be time-consuming.

## 2.1 Environment

* Tech Stack: The development environment will include IntelliJ as our IDE and Figma as our design tools. Android Studio to run the code and compile the code to see what the app looks like.
* API Dependent: The app relies heavily on the Spotify API for its core functionalities.

## 2.2 User Characteristics

The application will be used by Spotify Users. The users of the application will be redirected to the Spotify login page to sign in and give the application authorization to use the user’s information from Spotify.

## 2.3 System

* Performance: The app must be optimized for performance especially since it handles real-time data updates and constant sorting of playlist
* Security: The app needs to be secured but since we are directing users to Spotify for authentication that should help.

# 3.0 Functional Requirements

3.1 Login

3.1.1 Login page re-directs to Spotify login for Authorization.

3.2 Authentication

3.2.1 Authorization Key is sent back to app to save for future API calls.

3.2.2 Renewal Key is sent back to app to save for future API calls.

3.2.3 Expiration time for Authorization Key is set to 24 hours.

3.2.4 Redirects to home page of the app.

3.3 Display Home Page

3.3.1 Settings button is displayed on the homepage.

3.3.2 Must have a link to Audio player page.

3.3.3 My Playlists listed on homepage.

3.3.4 Create Playlist link on homepage.

3.3.5 Search bar on homepage.

3.6 Settings

3.4.1 App Information

3.4.2 Logout button displayed.

3.5 Player Page

3.5.1 Must show current Band/Artist

3.5.2 Must show current Album Cover

3.5.3 Must show current Album Name

3.5.4 Must show current Song Tempo

3.5.5 Must show current Song

3.5.6 Must have Play/Pause Button

3.5.7 Must have Skip Song Button

3.5.8 Must have Previous Song Button

3.5.9 Must show Progress Bar

3.5.10 Link to control Queue/Playlist Tempo Page

3.5.11 Buttons to increase or decrease the tempo of the current track queue.

3.6 My Playlists

3.6.1 Show list of User’s Playlists

3.6.2 Link to Create New Playlist

3.6.3 Link to Create Tempo Playlist

3.7 Playlist Page

3.7.1 Must be able to display/edit Playlist name

3.7.2 Must be able to display/edit Playlist description

3.7.3 Must be able to show/edit the List of songs

3.7.6 Must have a Sort by tempo button.

3.7.7 Must have a Play button to play the playlist in the player.

3.7.8 Must have a link Control Queue/Playlist Tempo Page

3.7.9 Must have Save Playlist button.

3.8 Create Tempo Playlist Page

3.8.1 Must be able to Set Tempo

3.8.2 Be able to create Recommended Tempo Playlist from Current Song

3.8.3 Be able to set a Max Tempo

3.8.4 Be able to set a Minimum Tempo

3.8.5 Must be able to add a Tempo Interval

3.8.6 Must be able to set the Length of a Tempo Interval

3.9 Track Class

3.9.1 Class must hold track information for Artist, Artist Spotify ID, Track Title, Track Spotify ID, Album, and Tempo

3.10 Tempo Tree

3.10.1 The Tree’s Nodes must hold a Track Object based on Tempo

3.10.2 The Left Child will hold a Track with a tempo slower than the parent node’s track.

3.10.3 The Middle Child will hold a Track with the same tempo as the parent node’s track.

3.10.4 The Right Child will hold a Track with a faster tempo than the parent node’s track.

3.10.5 When a track is added to a queue or played, the node is replaced by the Middle Child, and the Left and Right Child become children of the new parent.

3.10.6 If the Middle Child is Null, more tracks can be added with an API call using the current node’s tempo as a parameter.

# 4.0 Non-Functional Requirements

## 4.1 Security

* System must protect authorization details
* The system will protect API keys from leaking.

## 4.2 Capacity

* System should be bottlenecked through the Spotify servers, since we are using APIs to handle our app, most of the load handling will be dealt with by Spotify since our program will be using API calls to design our program., therefore user capacity is heavily dependent on Spotify and ios
* data storage capacity is also handled by Spotify, as caches and other data will be tied into users Spotify accounts. This covers user information, playlist capacity, and streaming capacity.

## 4.3 Usability

* As a user the interface must be easy and understandable.
* Users should be able to switch between our application and the Spotify application smoothly to easily interact with the generated playlist if needed – changes should reflect both sides so users can accurately make queues, playlist, etc.
* Users should be able to use touch screen devices to interact with the app, with functions being easily accessible and organized

## 4.4 Other

* Application should be aesthetically pleasing.

# 5.0 External Interface Requirements

## 5.1 User Interface Requirements

* Users must have access to IOS or Android, and a way to input on screen such as a touch screen.

## 5.2 Software Interface Requirements

* IOS 14 and Above
* Android OS 5.0 or above