**Software Implementation and Testing Document**

**For**

**Group 22**

Version 3.0

**Authors**:

Rebecca Anestad

Daniel Bryant

Cullen Hemsouvanh

Lance McMahon

Yassar Rich

# Programming Languages (5 points)

JavaScript - used because it is the standard for web development. Easy development and popularity over alternatives were our primary reason for using JavaScript.

HTML - used for website configuration. Used HTML because we took into consideration what we wanted our project to do, and using HTML was the only feasible way to do it.

CSS - used to style the website. Used CSS because we didn’t want to use XML and it is one of the most user friendly styling languages for web pages.

TailWind - used to style the website - used almost everywhere we implement design features. Used TailWind in some places instead of CSS because some members found it easier. It also sped up web development with pre-built CSS classes for development.

React - used to code the framework of the website because we thought React would be the easiest for each team member to learn if they didn’t already know it

Next.JS - tooling behind employment and routes. It also makes the web application more manageable to alter and understand what is happening.

# Platforms, APIs, Databases, and other technologies used (5 points)

Vercel - Building and Deploying the website

Database - We will be using SQL, to manage the data on our site. MongoDB might be used later in the project

Spotify Web API - allows us to retrieve content metadata with Spotify’s streaming service. Needed in order to login and retrieve data based on the signed in user.

# Execution-based Functional Testing (10 points)

We were able to verify the Spotify login works as expected on each of our machines, as well as the API request to Spotify’s database to retrieve the user’s top Spotify Artists. We also tested the website to see if the token was going through for authorization. Tested login flow to make sure state management was being handled appropriately.

# Execution-based Non-Functional Testing (10 points)

For execution-based non-functional testing, we tested the website’s performance, ensuring the load times of pages are reasonably fast (< 2 seconds). The current project structure is suited for large scalability.

# Non-Execution-based Testing (10 points)

Our group would meet once or twice a week to discuss the progress we have on our code as well as review what is next to implement. We also met on an individual basis to walk through the code as needed.