Spotify Playlist Analytics

Requirements Specifications

Team 7

Benjamin P, Brevin S, Jiawei L, and Leonard B

Course: CptS 322 - Software Engineering Principles I

Instructor: Sakire Arslan Ay

TABLE OF CONTENTS

I. Introduction

Our application generates and visualizes data queried from a user's Spotify playlist. It describes how happy, sad, energetic, slow, acoustic, electronics, etc. a playlist is. We intend to get this data from Spotify's web API.

The first section of this document acts as an introduction to our product, as well as a log of all changes made since its creation. It's a helpful guide to what's to come. The second part is a step-by-step explanation of the who, the why, and the what of this application. We'll go over the customers and users, their user cases, and the non-functional requirements of what's to come. After that, we'll talk about the front end and show mock-ups of the user interface. Finally, we'll go over our references and the things we looked to in the writing of this document.

Document Revision History

Rev 1.0 2019-10-07 Initial Version

II. Requirements Specification

The application is a front end for some of the data Spotify hosts on their Web API. The graphs generated from the data acts an easy visualization for end users so they can properly understand the mood, genre, and style of their playlists.

The user must first create an account, inputting a username and password. The next step requires the user to link their Spotify account. We'll generate API keys, and then store the username/password/key combination in a SQL database. The user can then sign in at any time and view graphs generated by their playlists.

II.1. Customer, Users, and Stakeholders

1. **Customer** - The customers of this application could be music provider companies like Spotify, Apple music, Amazon music, TIDAL. These companies can use this application to analyze user's music preferences so they can have a better understanding about their user's music preferences and promote similar songs. The customers could also be songwriters and music publishing companies. They can use this application to understand the characteristics of popular songs in the market and make songs which will be popular. Another customer could be the average person, looking to improve the content of their playlists.
2. **Stakeholder** - The potential stakeholders could be music publishing companies, song writers, data analyze companies. Music publishing companies can use this application to learn the market's music flavor and sign the proper singers/songwriters. Song writers/singers can use this application to adjust their music style and music type to get popular songs. Data analyze companies can use this application to do research. This also could be a built-in feature which can attract potential new users.
3. **User** - The user of this application could be all the music service subscribers and the people who want to know their music flavors. Both the music service subscribers and the people who want to know their music flavors can use this to learn their music preferences to find new songs.

Another user could be the average person, looking to improve the content of their playlists.

II.2. Use Cases

1. **Team member –** is responsible for completing tasks, update the task status and make sure that task is on time.

Name	Team member
Users	Participating Team Members. Namely Benjamin, Brevin, Leonard, and Jiawei
Rationale	Team member completes the task on time.
Triggers	Team member gets task assignments and information about the product
Preconditions	Team member gets the invitation
Actions	Team member completes the task and update the status of their work Team member sends messages to communicate with others
Postconditions	Team member has access to the system
Acceptance Tests	Team members are responsible for all the acceptance tests and make sure the project can run smoothly.
Iteration	All iterations

2. **Regular users-** experience the product, submit bugs and make useful suggestions

Name	Regular user
Users	All
Rationale	Experience the new application and submit the bugs
Triggers	The product releases to the public
Preconditions	The product is completed and passes the internal test
Actions	Regular users register their accounts Regular users sign into the application and get tutorial about the app Regular users experience the application and submit bugs
Postconditions	Regular users have access to the application
Acceptance Tests	N/A

Iteration	After all iterations

II.3. Non-Functional Requirements

1. [Maintainability]: [The system/application should be easy to maintain. The system should also have the ability of self-diagnose when no staff presents]

2. [Stability]: [The system/application must have the ability to run 7/24 hours without having break-down. It must be stable.]

3. [Capability]: [The system must handle properly when 5000 users are using at the same time. The website must respond to users' requests in a very short time.]

4. [Security]: [The system must have a good security mechanism to survive from normal attacks. It also needs to ensure secure transactions]

5. [Recoverability]: [The system/application should be able to go back to the most recent opened page when an error occurs and be able to let user do the same request again.]

6. [Scalability]: [The system/application should be friendly to new features and staff can expand the new feature quickly and perfectly]

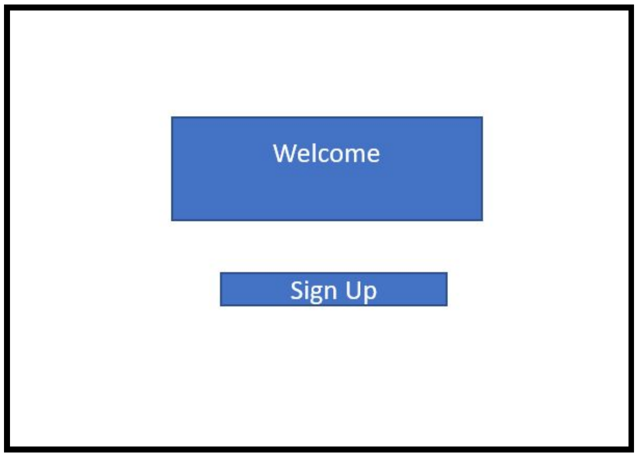
7. [Environmental]: [The system/application should be able to run on different platforms like running on phones, tablets, computers]

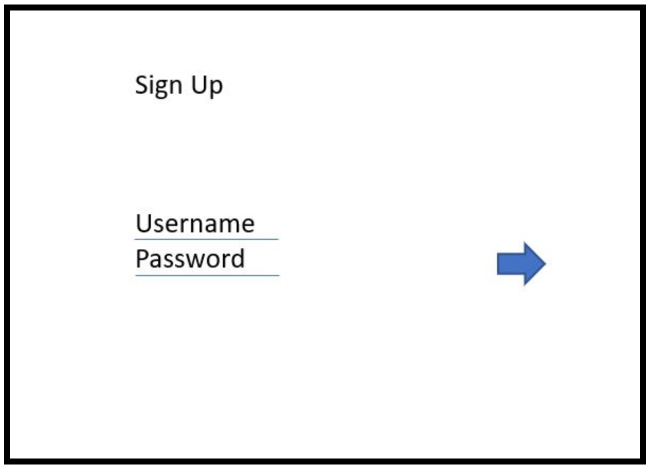
8. [Data Integrity]: [The system/application should use the same data format to avoid too much information being changed.]

9.[Usability]: [The system/application should be user-friendly. It should be simple to use and have a nice tutorial when user first use it.]

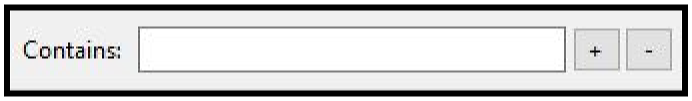
III. User Interface Requirements

The default page will bring the user to a page where they can sign-in or sign-up. This page will have a dark theme with white headers and grey common text. This page will be very minimal. Should a user choose to sign-up, it will bring them to a page where they are prompted for a username & password.



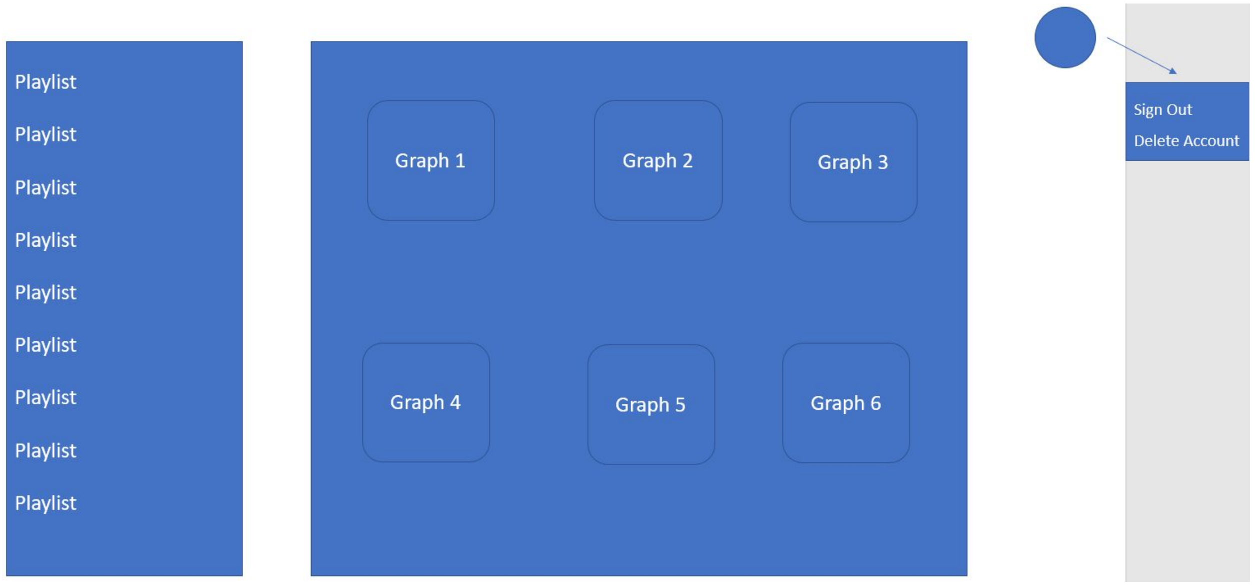


Once the user is signed-in, they will be prompted with a welcome screen where it will welcome their <username>. The interface will relatively simple by having a textbox that will accept the user's Spotify playlist URI and if they want to submit multiple playlists we have an "Add" Button that will pull up another textbox.



This is an example of the "Add" button. Instead of an addition symbol we will actually have it say "Add". Then another textbox will appear under the other textbox in front of "Contains:". The "Contains:" will say "Playlist URI:". The subtraction box will remove a textbox from the UI.

After the user enters their playlist(s) the page will go through an authorization waiting window for a brief moment when it will bring you to an analytic page where this page's purpose is to show all of the graphs and information that we were able to scrape from the Spotify API. The page layout will consist of two different columns. A smaller left column taking up 25% of page width, and a larger second column taking up roughly 60-70% of the page. In the left column we will display the full lists of playlists whether it be 1 or more. Each playlist will have a series of graphs and data corresponding to it displayed in the column on the right. A third very small column may be displayed on the right with a small size of 5-10% with this showing an integer value (playlist length). The example page;



This analytic page will also have a drop down in the top right corner to logout or remove the account you have with the songs you have on it. Overall the page style will contain a black background with white and grey texts. White upon headers and grey for normal div texts. The page will be designed to dynamically fit the number of playlists provided. The graphs will represent the different data we were able to discover. If we find it applicable we may generate text at the bottom underneath the six graphs.

IV. References

"Web API." *Spotify for Developers*, developer.spotify.com/documentation/web-api/.