CSC 431

Spotify Playlist Finder & Social Media

System Architecture Specification (SAS)

Team 17

Allegra Papera	Project Management, Product Designer
Grant Yaniv	Full Stack Developer
Yasir Nemat	Full Stack Developer

Version History

Version	Date	Author(s)	Change Comments
1.0	4/1/2021	Papera, A., Yaniv, G., Nemat, Y.	Filled out the system overview.
2.0	4/13/2021	Papera, A., Yaniv, G., Nemat, Y.	Fixed TA suggestions and added system, sequence, and class diagrams

Table of Contents

Table of Tables

Table of Figures

31.

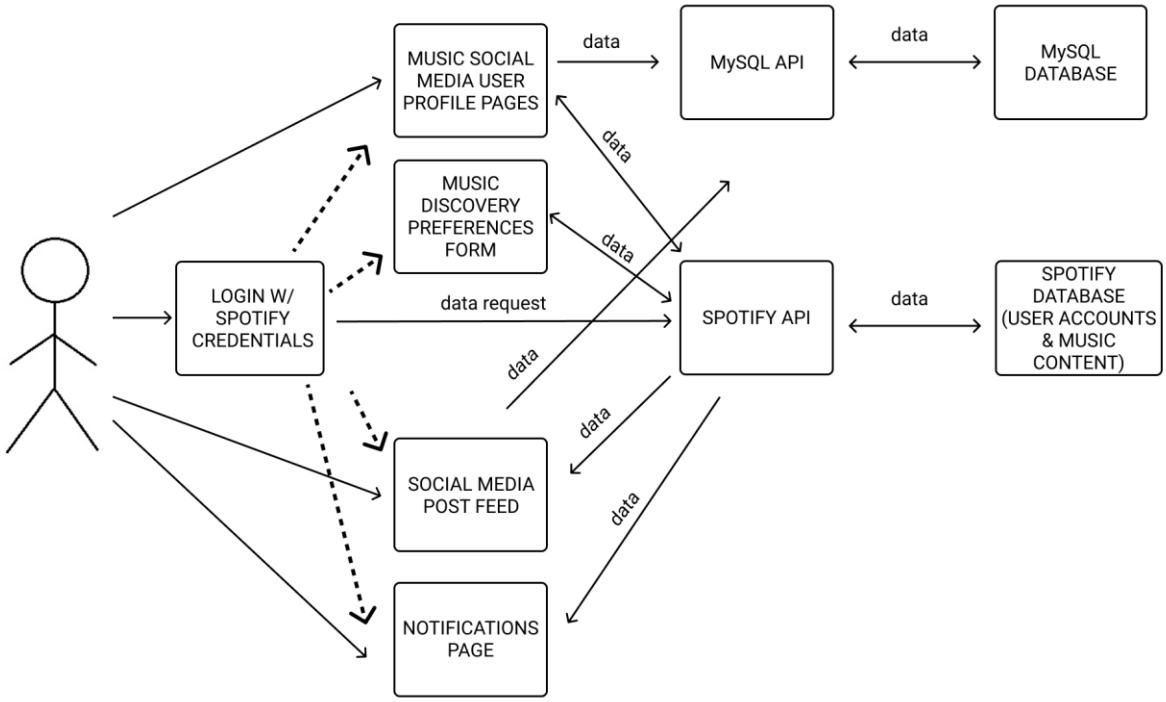
32. 1 System Analysis

32.1 1.1 System Overview

The application behaves as follows: (1) The user logs into our application by logging in with their Spotify account credentials, giving our application access to Spotify's database as well as the individual's Spotify account data (2) The user fills out their Music Discovery Preferences form, then our app will use that information to retrieve a sample of data from Spotify about which playlists fit the user's preference criteria. Our application will then randomly select certain playlists from this sample, and follow those playlists on the user's Spotify account. (3) The user will be notified with the Notifications page about the newly followed playlists, as well as any activity such as comments on their posts. (4) The Music Social Media User Profile Pages and Social Media Post Feed will also be updated with information about the playlists that were followed by our app on the user's Spotify account. (5) Posts added to the Music Social Media User Profile Pages and Social Media Post Feed will be passed to our application's MySQL Database via a MySQL API.

The system will be compromised of the following functional requirements: Spotify Login, Music Discovery Preferences, Music Social Media User Profile Pages, Social Media Post Feed, and Notifications Page. The Spotify Login will allow our application to access user data from Spotify's database via the Spotify API. The Music Discovery Preferences feature will interact with Spotify's music database (via the Spotify API) as it will be used to generate a sample of playlists retrieved from Spotify's database (via the Spotify API) that match the user's preferences for music discovery. The Music Social Media User Profile Pages will also retrieve and display user profile data from Spotify's database (via the Spotify API), in addition to our app's customizable profile features. The Social Media Post Feed and Notifications Page will not be directly connected to Spotify's database or its API, but rather they will reflect our app's actions triggered by our Music Discovery Preferences feature (which does interact directly with the Spotify database via the Spotify API) as well as any in-app behaviors such as writing text posts, album reviews, etc. In-app updates to the Social Media Post Feed and Music Social Media User Profile Pages will be passed to our application's MySQL Database via a MySQL API for storage.

1.2 System Diagram



32.2 1.3 Actor Identification

The human actors included in our Spotify Playlist Finder & Social Media platform is the user (who must login with their Spotify account credentials), who uses the application primarily to interact with their Spotify accounts (requesting new music to listen to via the Music Discovery Preferences Form) and to make social media posts related to their music listening activity on Spotify. The system will support interaction with non-human actors, the Spotify API and the MySQL API to allow for interaction between the application and the Spotify User Account Database and our MySQL Database containing data related to our app's Social Media Feed aspect.

32.3 1.4 Design Rationale

2.3.1 1.4.1 Architectural Style

The application is designed using the Pipes and Filters architectural style. Pipes and Filters has components called filters which perform transformations on data and process the input they receive, as well as pipes which serve as connectors for the stream of data being transformed. Specifically, filters will receive input about the user's Music Discovery Preferences, and it will transform that data into a selection of playlists that could match the user's preferences. A smaller random selection of those playlists that are not currently followed by the user will be chosen to be followed on the user's Spotify account. The Spotify API will serve as both a Pipe and a Filter, as it will be used to inform Spotify's DB of the user's preferences, and it will also be used to sample the playlists that match those preferences.

2.3.2 1.4.2 Design Pattern(s)

Based on this type of application and system we will be creating we believe that the Façade design pattern would be most applicable because it will give us the unified interface we are looking for on this project.

2.3.3 1.4.3 Framework

We will be using the Spotify API to be able to use the functionality of some Spotify features, such as their playlist catalog and user profiles, in order to lay the foundation of our application. We do not need to create the Spotify API, we just need to take advantage the existing Spotify API's functionalities. We will be able to make Spotify API calls using JavaScript code.
We will also be using the Node.js framework, as suggested by the Spotify API Developer Guidebook, to handle the back-end and server-side of this application since our front-end logic will be created with JavaScript.

33. 2 Functional Design

33.1 2.1 Sequence Diagram for General Use Case

This diagram shows the general sequence of usage for our application, outlining how the user begins using the application, what they can do with it, and how they terminate use.

- Once the app is opened, the login screen is displayed.
- The user submits login information, which is verified and loads user data if success, or denied and displays a failure message if it fails
- The profile page and social feed is then shown.
- The user may then create a social media post or fill out the playlist form.
- If the user creates a social media post, it is stored in the MySQL database.
- The post is then sent to the client and visible on the social media feed.
- Once a post is visible, a notification is sent to the user's notification page.
- A Playlist form is sent to the user to be filled out.
- Once the user fills out the playlist form, our algorithm sifts through the form and API music data to determine what types of music should be applicable.

- The Spotify database is queried to determine suitable databases according to the form data.
- The playlists are followed for the user and a notification is sent to the user’s notification page.
- When the app is closed manually or unused for long enough, it is closed.

34. 3 Structural Design