Project 7

Project Name: Advanced Spotify Search

Team Members:

- Arjun Lakshmi Narasimhan
- Ben Wedeen
- Kyle Kells

Final State of Project

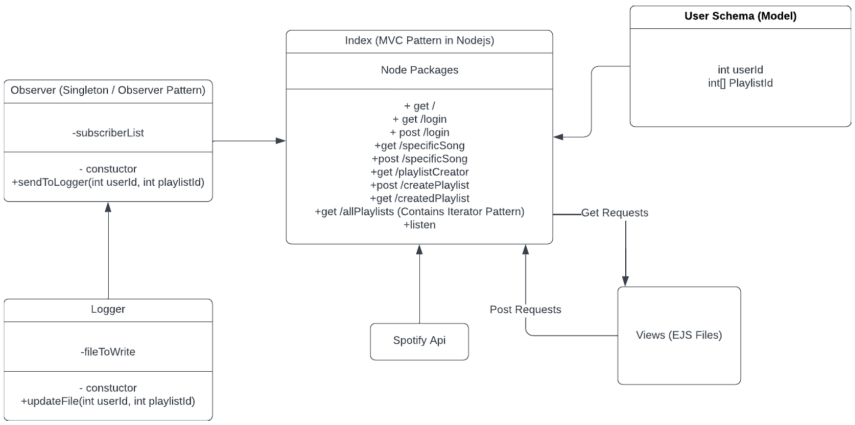
After Project 5, we knew we wanted to create a React based web application that would leverage Spotify's REST API to allow users to login into their Spotify account, acquire detailed information about music and create playlists of music with similar characteristics. We planned on creating a Logger object like in the music store assignment that outputs user activity to a text file.

By project 6, we had the architecture of our React web application established and had the login, navbar, and search features created. Our search page displayed a list of song results with album art and a list of song attributes fetched from the spotify API.

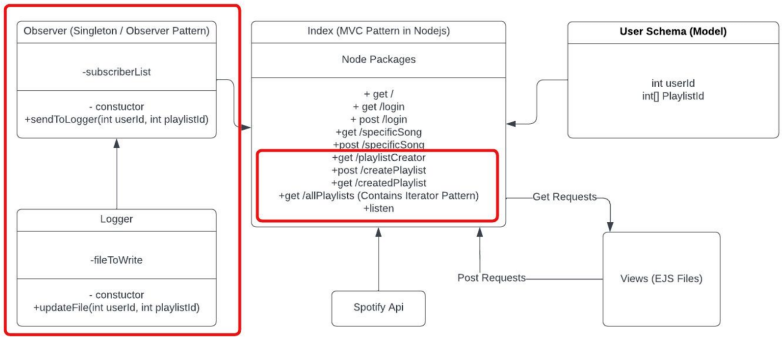
Since Project 6, we have created an algorithm to generate playlists from a desired range of song attributes alongside a slider-based UI for inputting the song attributes. In order to keep track of the activity of our users, we created an observer pattern that subscribes to the activity of the user and logs new users logins, returning user logins, and newly generated playlists into the console. Since there is only one user logged into a session of the application at a time, we implemented a singleton pattern on the object that acts as the user being observed. The text output based Logger object didn't seem as appropriate in a webapp when it came time to implement the Observer pattern so it was modified. Finally, we deployed the app to Heroku (https://advance-spotify-search.herokuapp.com/)

Final Class Diagram

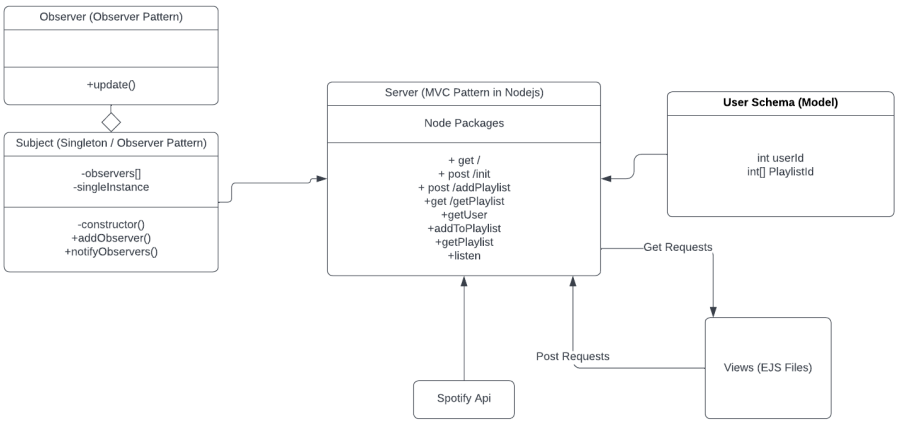
Project 5



Project 6



FInal



As demonstrated in the diagrams, our overall architecture was implemented as initially planned. Between Project 5 and 6 we created all the essential components of the Architecture, established a connection with the spotify API, and put some GET and POST requests to work in order to create our search page and Navbar. After project 6 we implemented the playlist creation and the observer/singleton pattern, making some deviations from the previous UML when it seemed necessary.

Third-Party Code vs Original Code

The majority of our code is integrated with third-party code, using Mongoose packages in the node framework for server process as well as Axios and Spotify packages in our React-based front end. Using these frameworks, we were able to get our desired results. Every promise (.then and .catch) and the usage of package methods are original code. Most of our code involves creating the UI using HTML, using effects and states which are part of the React framework, all promises done for all RESTFUL requests which are asynchronous by nature.

We used the API and package documentation for our app.

Spotify API - https://developer.spotify.com/documentation/web-api/reference/#/ (API)
Spotify Web API Node - https://www.npmjs.com/package/spotify-web-api-node (Handle get request and post request to api
Axios - https://www.npmjs.com/package/axios (Handle get requests and post request to server)
Mongoose - https://mongoosejs.com/docs/api.html (Link and Query MongoDB model)

OOAD Process

The greatest element of our OOAD was the creation of the MVC, with the Model being the database schema, the view being the react frontend, and the controller being the server app.js. Another big element of the design process was reading through the available documentation in order to learn how to use the abundance of packages that we were using. The Spotify API caught our eye while looking through APIs we could work with because it allowed us to access interesting data about music you can't normally access through Spotify. Our biggest problems stemmed from the fact that we were exploring new technologies that all of us were unfamiliar with. Getting all of the different technologies to work together in a full stack web app while implementing object-oriented design patterns felt like a bit of a juggling act. As a result, completing most of our tasks took longer than we originally predicted.