**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 DESISGN DOCUMENT (SDD)

CS 4850 Senior Project

Fall 2024

Professor Perry

09/01/2024

SP-27 Green Spotify

[1. Introduction and Overview 4](#_Toc235221615)

[2. Design Considerations 4](#_Toc677315911)

[2.1. Assumptions and Dependencies 4](#_Toc991255568)

[2.2. General Constraints 5](#_Toc817523897)

[2.3. Development Methods 5](#_Toc620984980)

[3. Architectural Strategies 5](#_Toc830279608)

[4. System Architecture 7](#_Toc1590466368)

[5. Detailed System Design 7](#_Toc192079905)

[5.1. Classification 7](#_Toc1211182626)

[5.2. Definition 8](#_Toc1771726586)

[5.3. Constraints 8](#_Toc1853590130)

[5.4. Resources 9](#_Toc1882546341)

[6. Glossary 9](#_Toc495885590)

[7. Bibliography 9](#_Toc385030362)

# Introduction and Overview

This document describes the ideas and design of our Spotify App with Tempo Control for Playlists. The document will provide the outline of the design considerations, constraints, methods, and architectural strategies used while creating the application. This document will also have a general description of the app's architecture design and a more in-depth description of each component.

# Design Considerations

## Assumptions and Dependencies

* Related software or hardware
  + Spotify API
  + Spotify iOS SDK
  + Spotify Android SDK
  + Spotify APP
* Operating systems
  + Our Spotify app will run on the iOS and Android operating systems
* End-user characteristics
  + The User will be redirected to the Spotify login page to sign in (Authorization Code Flow | Spotify for Developers 2024).
  + The app will request permission from the user to use the user’s information from Spotify (Authorization Code Flow | Spotify for Developers 2024).
* Possible and/or probable changes in functionality

## General Constraints

* The App must be able to run on IOS 14 and Above Android OS 5.0 or above
* App will primarily be used on mobile devices, Windows laptop, or mac
* Spotify imposes a rate limit, so we must make sure to not make too many API requests in a brief period
* App will Authenticate users by redirecting to Spotify
* Access to the modification would involve the user having an ios/android with the Spotify app already installed on their device.

## Development Methods

The development of this app involved Agile software development. In development of the app, a plan detailing all the steps of development will be written along with a timeline of the start and end of each step in the development. The app will then be designed and developed, which would lead to it being deployed for testing. This is one cycle, Agile is a development method with multiple cycles, more cycles are run to perfect the product. After the testing phase, review of the application is used to develop a new plan to improve the product.

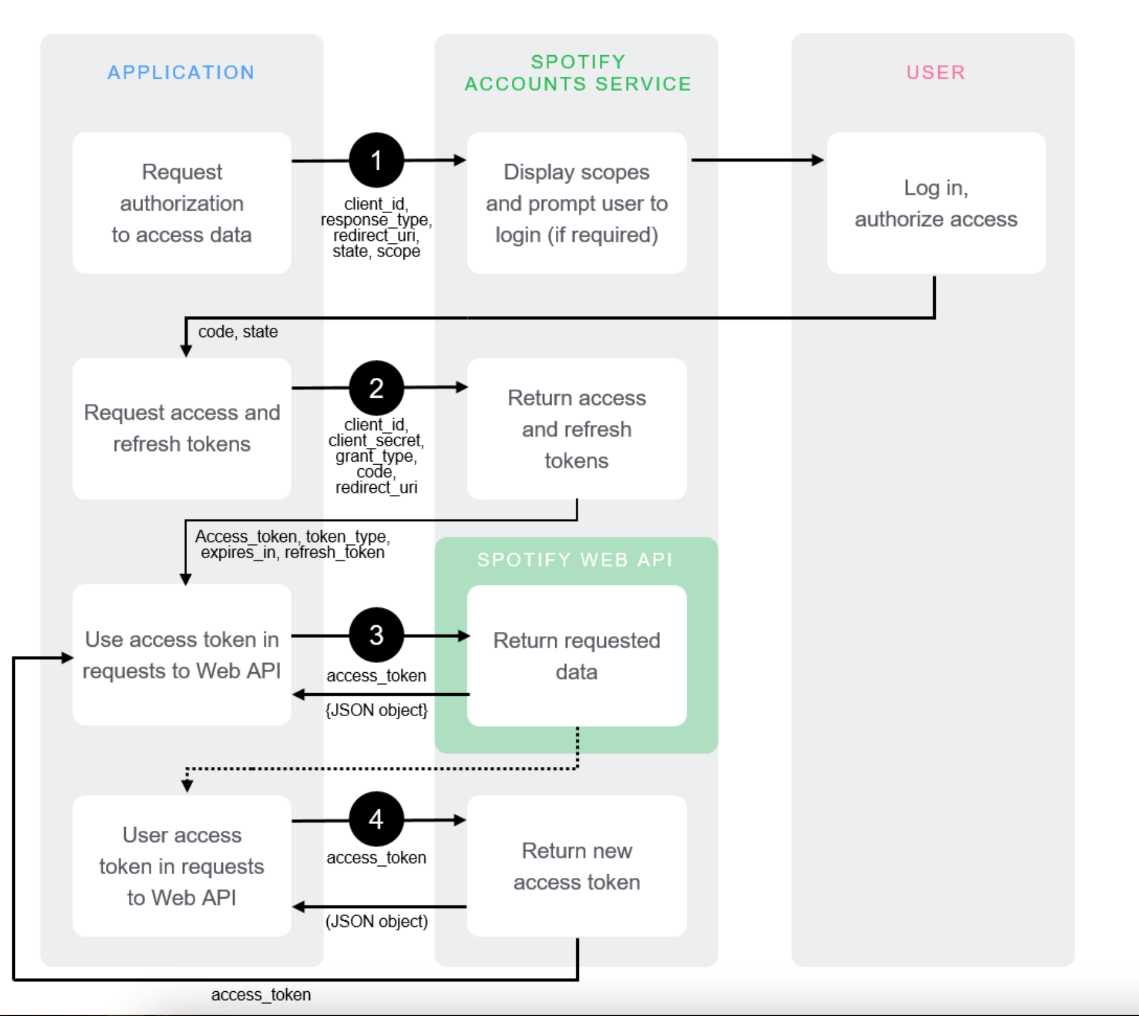
# Architectural Strategies

This section discusses the most important high-level architectural strategies and design decisions that have driven the overall organization of the system and its large-scale structures. The strategies provide insight into the important abstractions and mechanisms that a system's architecture employs to ensure that the design goals are successfully achieved. Each important strategy is covered, including the reasoning behind the decisions made, the trade-offs considered, and other alternatives appraised:

* **Programming Language and Framework**: React Native was chosen for its ability to develop cross-platform applications that run on both iOS and Android using a single codebase, reducing the time and cost required in developing the project. JavaScript is widely used, with great support and resources.
* **Reuse of Existing Software Components**: The system reuses Spotify's API for music data retrieval. Reusing well-established components like Spotify API allows the development team to focus on unique feature implementation rather than reinventing the wheel.
* **Distributed Data or Control Over a Network:** The application relies on the Spotify API for distributed data management, with all music-related data being fetched from Spotify’s servers rather than stored locally. Leveraging Spotify's existing infrastructure for data management allows the application to provide up-to-date music information without the overhead of managing its own distributed data system.
* The system does not have a database. All information needed is retrieved from Spotify using the Spotify APIs. Any creation of playlists by the user will be stored by

sending the playlist to Spotify using the APIs. Since all information can be stored within Spotify, no database is needed.

* The Spotify App must be on the same device as our application for our application to act as music Player. Our application will use either iOS Spotify SDK or the Android Spotify SDK to communicate with the Spotify app on the device to stream the music file. This is no other way to play tracks from Spotify without the SDKs and the Spotify app on the same device.
* The Application will follow Spotify’s OAuth protocol (Authorization Code Flow | Spotify for Developers 2024). The application will request authorization from Spotify using the application’s clientID from Spotify. The Spotify Account’s Service will request the user to login, and then state scopes and prompt the user to confirm authorization. Once authorized by the user, the application will reach out to the Spotify Account’s Service to retrieve the user’s access token, refresh token and refresh token time. The Access Token will be used to retrieve the user data from Spotify. The refresh token is used to retrieve a new access token for the user after the refresh token time has expired. Figure 3.1 shows the process of OAuth and using the Access Tokens. (Authorization Code Flow | Spotify for Developers 2024)

Fig 3.1 

# System Architecture

* Spotify APIs will be used to retrieve/send information to and from Spotify. This will allow us to use Spotify as the application’s “Database.”
* Spotify SDKs will be used to communicate with the Spotify App to stream tracks.
* Track Class will be used to hold individual Tracks information.
* Tempo Tree will be a tree holding Track objects and will have methods to sort tracks and send tracks into queue.
* Playlist Queue Class will be a queue holding the Track objects on the current playlist, as well as save the playlist for Spotify.

# Detailed System Design

## Classification

The kind of component, such as a subsystem, module, class, package, function, file, etc. ....

5.1.1 Track Class - The Track Class Object will hold the information of the track in variables. The variables will be Title, TrackID, Artist, ArtistID, Album, Length and Tempo.

5.1.2 Tempo Tree – The Tempo Tree will be a data structure tree to hold the tracks based on the individual track’s tempo. Each node will be a different track with three child nodes. The Left Child node will be for tracks will a tempo or tempo range smaller than the parent node. The Right Child node will be for tracks with a tempo or tempo range larger than the parent node. The Middle Child node will be for tracks with the same tempo or tempo range as the parent node. This will allow for quick searching of the tree for a specific tempo. If a song is added to the Playlist queue while making a Playlist, the Parent Node will be replaced by the Middle Child node, and the Middle Child will become the new Parent Node of the Left and Right Children of the former Parent Node. If the Middle Child is Null, then a GetRecommend API call can be made to Spotify using the Parent TrackID and Parent Tempo as parameters to get more songs for that tempo. New tracks can be added when needed. This will also allow for a dynamic tempo change for the user.

5.1.4 tempoEnqueuLow – Function under the Tempo Tree. tempoSortLow will send tracks from the Tempo Tree into the Track queue starting from the low end of the user’s set tempo range. The function will send the tracks to the Track queue in preorder, until the desired higher tempo range is met, or the number of songs is met.

5.1.5 tempoEnqueuHigh – Function under the Tempo Tree. tempoSortHigh will send tracks from the Tempo Tree into the Track queue starting from the high end of the user’s set tempo range. The function will send the tracks to the Track queue in a reverse preorder, until the desired higher tempo range is met, or the number of songs is met. Reverse preorder will start with the Right Child, then Middle Child, then Left Child.

5.1.6 Track Queue – Track queue holds the current playlist either pulled from Spotify playlist, the user’s Spotify playlist, recommend songs, or a created tempo playlist. The Track queue will be sorted by the desired tempo or tempo intervals as provided by the user.

## Definition

* Track Class:

**Purpose**: The Track class represents the single track of music in an application. It maintains all the pertinent information for a track: title, artist, album, length, and tempo so other parts can use this information.

**Semantic Meaning:** The Track class forms the core data responsible for music tracks in a structured and orderly manner. It is the base for playlist operations, sorting algorithms, and user interaction with music data.

* Tempo Tree:

**Purpose:** To arrange tracks in a hierarchical format according to the tempo of different tracks. This makes it fast and easy to sort through songs by BPM, creating a dynamic way of creating and controlling playlists.

**Semantic Meaning:** The Tempo Tree is a cornerstone in enabling tempo-based music organization. It underlies the app feature to make dynamic adjustments to playlists based on the user's activity or mood by arranging similar tempo tracks.

* Track Queue:

**Purpose:** A Track Queue orchestrates the order and track position of the tracks in a playlist. It is intended to facilitate the re-ordering of the tracks according to user preference, say the tempo, and then have the playlist presented in the desired order for playing.

**Semantic Meaning:** The Track Queue plays a very essential role in managing the flow of music playback. It acts as an operational interface between the user's playlist selections and the playback engine in the right sequence according to the settings of the user.

## Constraints

* Track Class:

**Timing Constraints**: Instantiation and access of the Track class must be done in real-time without significant latency in operations such as sorting and playlist generation.

* Tempo Tree:

**Performance Constraints**: The tree structure should be capable of facilitating the search and sort operations in a very efficient manner for insertions, deletions, and search operations

* Track Queue:

**Data Consistency**: The queue should be capable of maintaining data consistency all the time regarding the correct order of tracks as per the user's taste and should not lose consistency even when modifying the playlist dynamically.

* Security Constraint: Spotify OAuth expires every 24 hours.

## Resources

* Spotify API will be used to retrieve user data, playlist data and Track data from Spotify. The user must authorize the application to use the Spotify API during login.
* The Spotify iOS SDK will be used to stream tracks to the player. The application will communicate using the Spotify iOS SDK with the Spotify application on the same device to stream the tracks.
* The Spotify Android SDK will be used to stream tracks to the player. The application will communicate using the Spotify Android SDK with the Spotify application on the same device to stream the tracks.

# Glossary

--

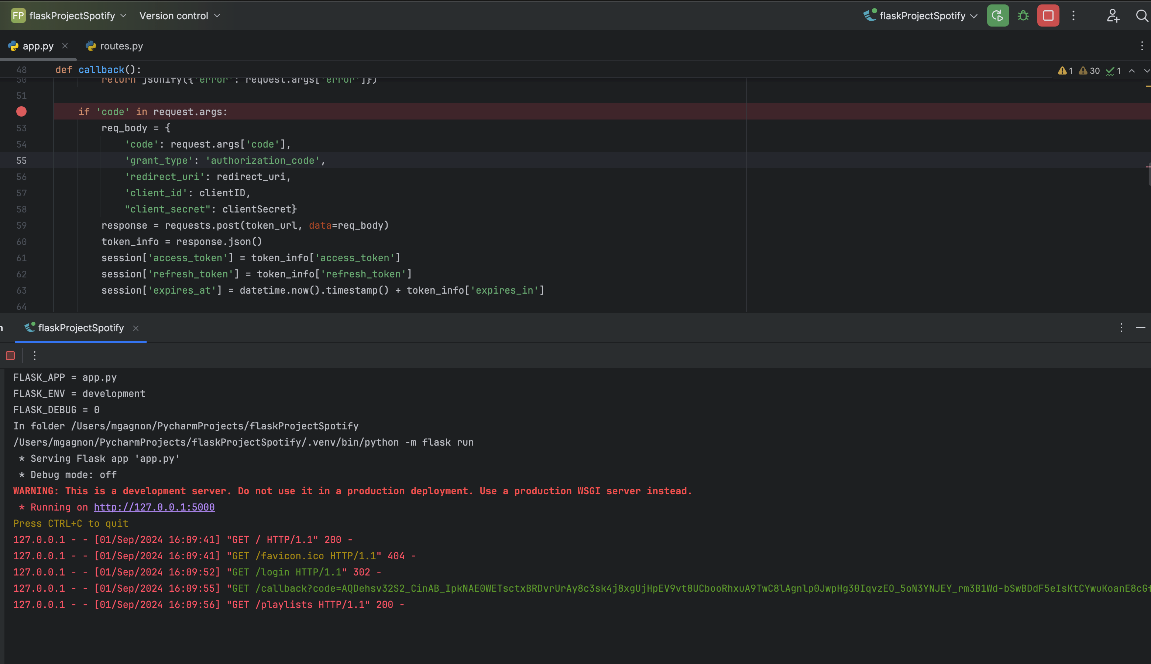
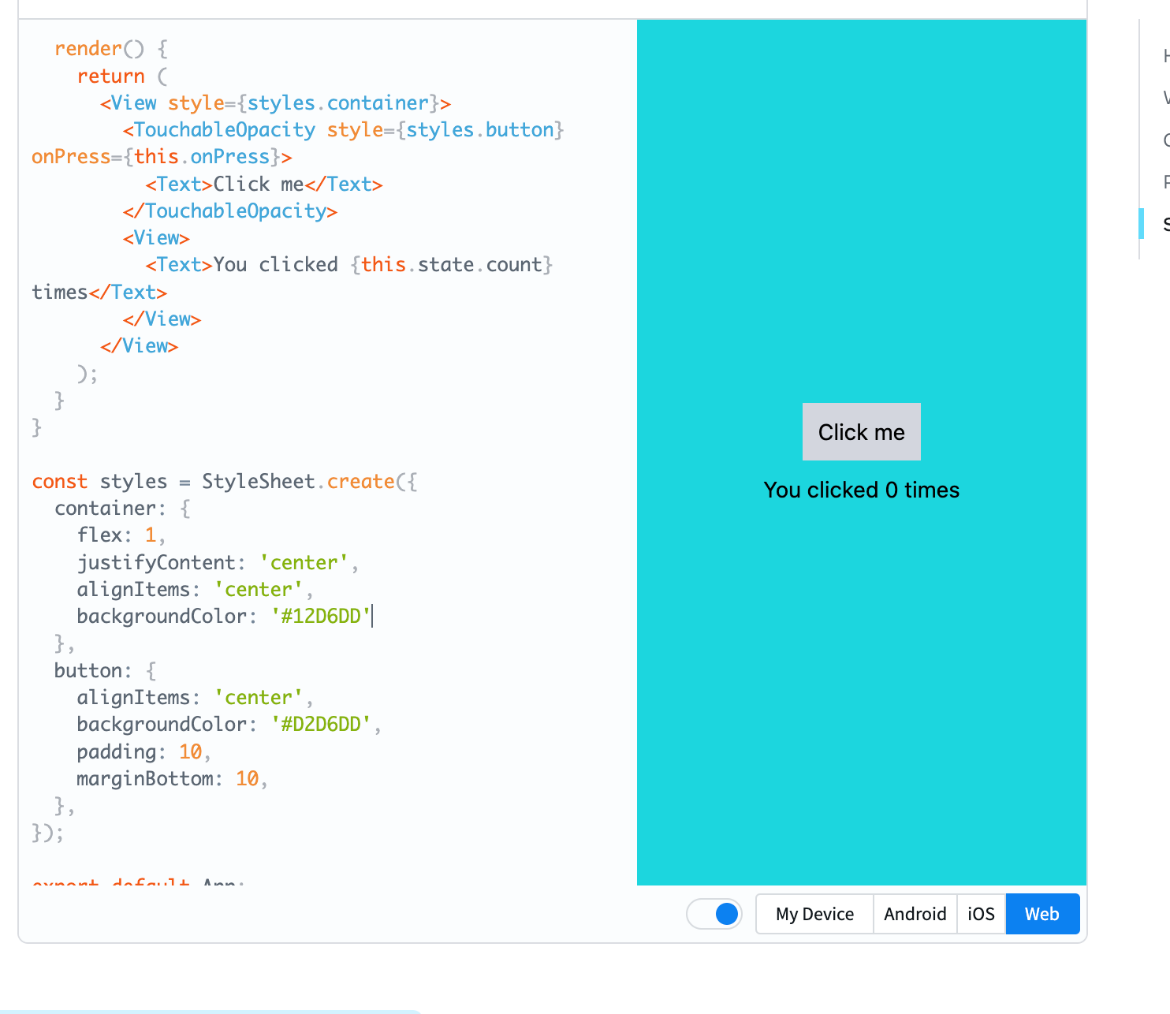
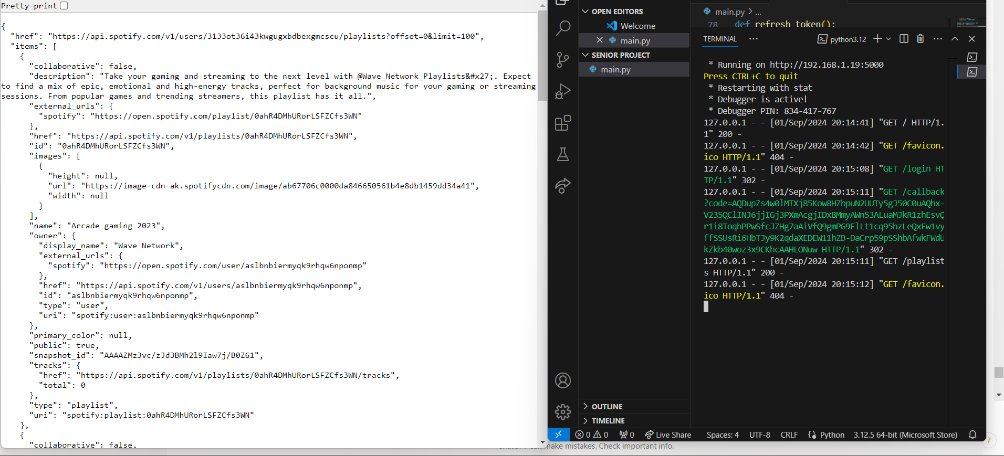
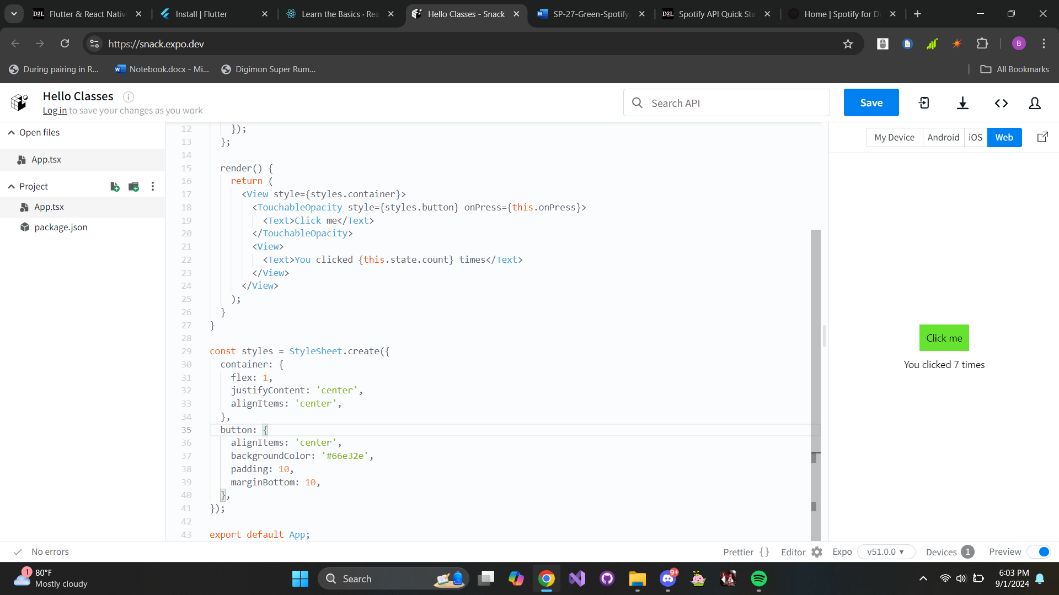
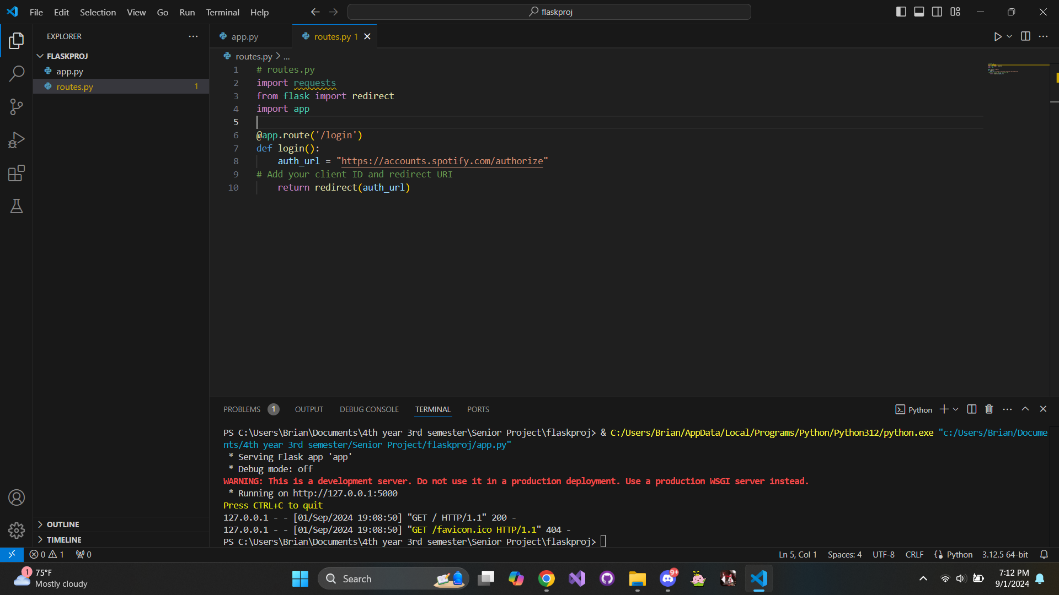
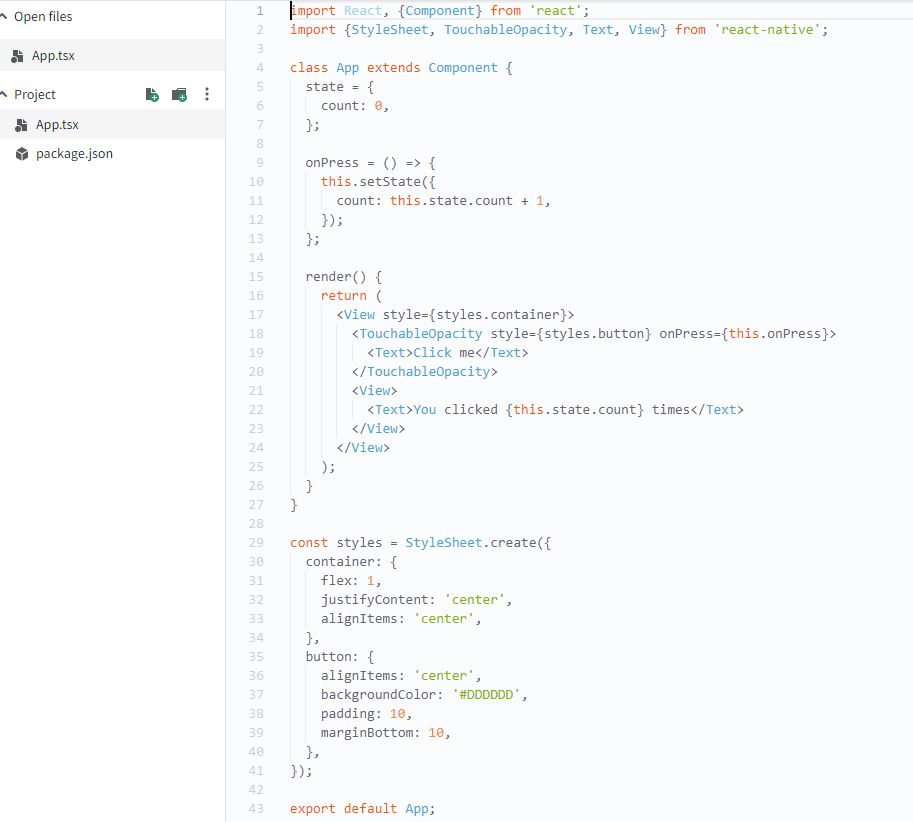
Definitions to be updated later.

# Bibliography

“Authorization Code Flow | Spotify for Developers.” *Developer.spotify.com*, developer.spotify.com/documentation/web-api/tutorials/code-flow.

# Appendix

8.1 Tutorials

* Marc Spotify Tutorial
  + 
* Marc React Tutorial
  + 
* Muhammad Spotify Tutorial
* 
* Muhammad React Native Tutorial
* 
* Brian React Native Tutorial
  + 
* Brian Spotify tutorial
  + 
* Jace React Native Tutorial
  + 
* Jace Spotify Tutorial
  + 