Smart Music Playlist Generator

March 2025

CSCI 3329-01

1. Proposal Head

- Project Title: “VibeGen”
- Team Number & Team Members:
    - Team Number: 14
    - Team Members: Jose Ramos, Kayla Coria, Carlos Garza

2. Project Introduction

- Project Description: This project will develop a CLI-based Music Playlist Generator that uses several python OOP concepts, allowing users to create personalized playlists on Spotify.
- Background & Motivation:
    - Background: Similar programs in the market include Spotify’s feature to create a radio station and personalized playlists based on the specific song you liked. However, these features lack flexibility, as they primarily focus on a song’s genre without deeper customization.”
    - Motivation: Manually creating a music playlist whether it be for a specific occasion, or mood can be time consuming on Spotify. This program aims to fix this issue with a user-friendly playlist generator that links to your Spotify account and creates a playlist. Our goal is to understand python OOP concepts to maximise performance and code readability to develop our overall programming skills and knowledge of these concepts.
    - Personalized music experience - the usual algorithm used in popular apps can be quite generic and almost identical, VibeGen has the capability to be unique with its customizable experience for users based on their moods, genres selection and previous music background
    - Similar to the operating system Linux this open source utility a lightweight CLI-based for developers or users that might prefer terminal-based interactions over graphical visualizations
    - The automatization can quickly create a playlist based off of the user’s chosen parameters such as their mood, listening habits without actually searching for songs
    -
- Objective:
    - Our primary objective is to apply key OOP concepts in a practical project, reinforcing our understanding through implementation.
    - The program should facilitate an already known concept when relating to generative music playlists.

    - Ensure maintainability and scalability as it can be a well documented program that can be extended and implement new features as desired to improve its reliability.

3. Functionalities & Features

- Main Functionalities:
    - Menu system that allows users to log in or exit.
    - Input display for username and password.
    - Options for generative music playlist or to view existing playlists.
    - If generating a new playlist it should ask for the amount it should include and the mood.
    - A filtering section that will make sure the music generated is related or directly represented by the mood or genre typed.
    - To save all songs as a new playlist. There is also the capability of deleting the generated and asking for a new list of songs.
    - If it's saving a new playlist, ask for a string input to name the playlist.
    - A way to turn back to the main menu to log out and exit.
- Interface Example:
    - Log in or exit: It implements the usage of a class that holds a display function with the main menu options that receive the integer that represents the choices. Each option will be appointed to a number that when the input is received the program will be able to recognize what the user wants.
    - Username and Password: Spotify API runs off OAuth which requires the user's spotify credentials to log in. This project ensures authentication runs smoothly through encapsulation features for data hiding and user credential safety.
    - Menu: Once access has been granted the next menu will appear, this includes the option to create a new playlist or view existing playlists, delete playlist, and logout/exit
        - Create playlist - Users can prompt the number of songs, select playlist type (mood-based, genre, artist).
        - Playlist management - view saved playlist (album), delete playlist (album/song), save new playlist, return to main menu.
    - Filtering: The feature will receive a string input where the user can include the genre, artist, mood, or vibe they wish for the songs. With this AI will be able to recommend specific songs that fit. If some of the songs are not fit for what was asked of the program the user can generate more.
    - Save or delete:
        - When a new playlist is generated the user will have the option to save the whole playlist or delete and regenerate a new list of songs.
        - An additional feature is the capability of choosing how many songs and what songs to save.

        - If a new playlist is saved the user can give it a name so it's recognizable when the program is used again and they see existing playlists.
- Program Structure: The program will follow a module based structure to improve code organization. We will have user management modules to manage user authentication, and an API integration module to communicate with the API's used.
    - **Class Hierarchy:** Base Class - We will create a playlist class defining common functionality of a playlist such as adding, removing and displaying songs.
    - Core Classes - UserPlaylists to represent user's created playlists
    - Other Essential Classes: SpotifyClient which handles authentication and API requests.
    - Login Class will manage user authentication and session handling.
    - AIProcessor will be used to process user's input for playlist generation.

  This object-oriented structure ensures that the project aligns with our goal of learning OOP concepts.
- AI power song recommendation - the usage of AI models to help filtering user's input by creating keywords to use as parameters for calling Spotify API which returns a list of recommended songs based on the keywords. For example, if the user prompts: "I want a chill study playlist with Lo-Fi beats", the AI model will return keywords such as: "chill", "Lo-Fi", "study".

4. OOP Implementation

- OOP concepts Used:
    - Inheritance: Create a base class Playlist with common functionality of a playlist where we will extend it with child classes such as UserPlaylist (for user-created playlists), or GeneratedPlaylist(for AI recommended playlists).
    - Encapsulation: Create a SpotifyClient to handle authentication and API requests also a basic Login class which will be utilizing private attributes to protect users tokens. Instead of direct access to sensitive data, getter and setter methods will be used.
    - Polymorphism: The Playlist class will have methods like display_playlist(), which will be overridden in different playlist types to format output differently.

5. Libraries & Tools

- Python Libraries
- Generative AI Utilization: Use Generative AI OpenAI API to create keywords based on the input of the user's prompt. These keywords will be used to communicate with the Spotify API to receive songs.
- Spotipy - Lightweight Python library for the Spotify Web API.
- SQLite - storing users preference and playlist history
- Typer - to create intuitive CLI commands

6. Timeline

- Project Timeline:
  - Week 1: Set up Classes and Spotify API access keys as well as gather documentation for libraries and the API that we will be using.
  - Week 2: Create CLI interface and functionality of different classes.
  - Week 3: Setup Authentication, and API handling through Spotify Class using documentation to make playlists with user's credentials.
  - Week 4: Create an AI keyword generator to assist in playlist generation.
  - Week 5: Connect CLI interface with backend, and ensure OOP concepts are being used effectively throughout the project.

7. References

- Reference Materials:
  - Spotify API Documentation: https://developer.spotify.com/documentation/web-api
  - Spotipy Python Library Documentation: https://spotipy.readthedocs.io/en/2.9.0/
  - OpenAI API Documentation: https://platform.openai.com/docs/models
  - Similar Project: https://medium.com/@ccpythonprogramming/how-to-create-a-personalized-spotify-playlist-generator-with-python-98bfa590aee9