SONGS RECOMMENDATION SYSTEM

PROJECT OVERVIEW

This project aims to develop an efficient song recommendation system by employing content-based filtering with the K-Nearest Neighbors algorithm. The system will analyze the intricate characteristics of songs and suggest similar songs to users based on their unique preferences and tastes.

OTHER NOTES

Ideally, we would leverage a hybrid recommendation system that combines the strengths of both content-based and collaborative filtering approaches. However, if we encounter time-related constraints, we will prioritize the implementation of the content-based system as we deem it most suitable to ensure timely delivery without overlooking accurate functionnality.
The song recommendation system takes into account a comprehensive set of features, including but not limited to genre and artist, as described in detail in the dataset section, to provide tailored song recommendations to the user.

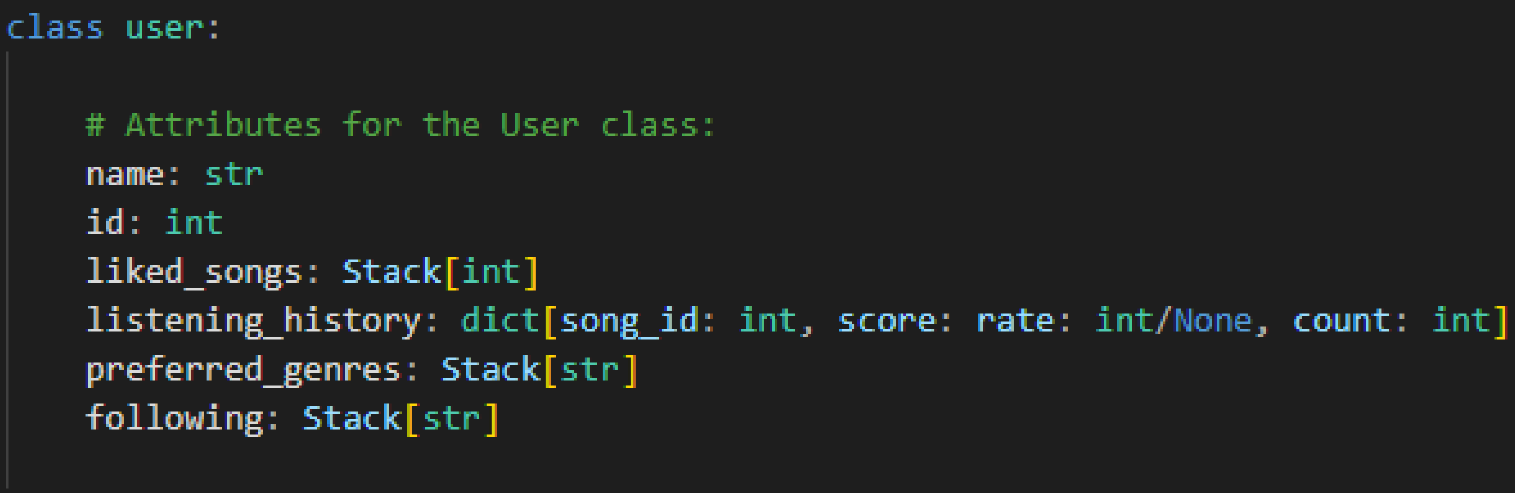
DATASET

- Source: Created by extracting and cleaning data from a dataset originally published at [Kaggle:Top Spotify songs from 2010-2019 - BY YEAR]. link: https://github.com/M0-Amine/songs_recommendation_system/blob/main/top50MusicFrom2010-2019.csv

- Description: The dataset for implementing the song recommendation system is stored in a JSON file and consists of the top Spotify music tracks from 2010 to 2019. It comprises 600 rows of data and includes 14 columns of features, including essential ones such as the song's title, artist, genre, and year of release, but also other relevant quantitative (represented as floating-point numbers) features such as beats per minute, energy, danceability, loudness, liveness, valence, length, acousticness, speechiness, popularity.

DATA STRUCTURES DESIGN

- User Class:



```
class user:

    # Attributes for the User class:
    name: str
    id: int
    liked_songs: Stack[int]
    listening_history: dict[song_id: int, score: rate: int/None, count: int]
    preferred_genres: Stack[str]
    following: Stack[str]
    # Methods for the User class:
    def listen_to(id: int) -> None:
        """ adds song object to listening_history and increment the
    listening count of the song by one"""

    def add_to_liked_songs(id: int) -> None:
        """ adds song to liked_songs """

    def remove_song(id: int) -> None:
        """ remove song from liked_songs """

    def history_recommendation(id: int) -> Stack:
        """ returns a stack of recommended songs based on listening_history """

    def liked_recommendation(id: int) -> Stack:
        """ returns a stack of recommended songs based on liked_songs """

    def rate_song(id: int, rate: int)  -> None:
        """ assigns rate to the value of the id key in listening_history """
```

- Song Class:

```
class Song:

    # Attributes for the Song class:
    title: str
    id: int
    genre: str
    artist: str
    Fi: float          #(Fi: i-th quantitative feature)
    magnitude: float

    # Methods for the Song class:
    def get_similar_songs() -> Stack:
        """ returns a list of similar songs based on the features
        specified previously """
```

FINAL OUTCOME

The final outcome of the system is a stack or playlist that contains all the newly suggested songs determined by the K-Nearest Neighbors algorithm.

- minimum result: suggesting a few songs similar to those added to the 'liked songs' playlist.
- ideal result: making whole playlists for the user based on his listening history, and potentially introduce new tracks from genres that the user may not typically explore.