TuneMe – Music Recommendation and Search Engine

Project Members and Advisor

Member(s)

- Zaina Qasim
- qasimza@mail.uc.edu

Faculty Advisor

- Fred Annexstein
- fred.annexstein@uc.edu

Project Background and Goals

The most popular music listening, and recommendation apps are Apple Music and Spotify. The former uses listening history and similarity to other users' tastes to recommend music, and the latter analyzes a song's rhythm and structure. Both do not ask users to specify their definition of "similarity" - by artists, years, and/or genres, etc. This project aims to provide better suggestions by doing so.

Intellectual Merits

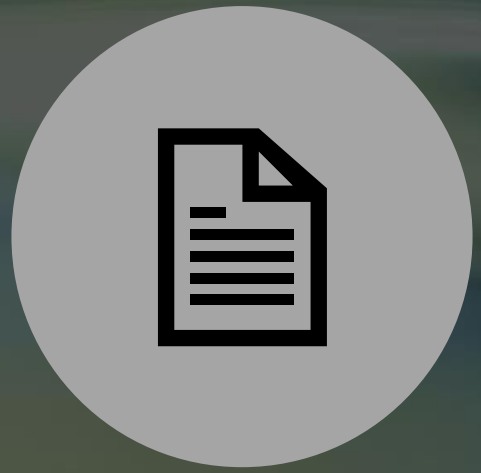
User Focused Design = Potential for Improving Music Discovery: TuneMe addresses a gap in existing music recommendation systems by allowing users to specify their definition of similarity. This means that the application has the potential to improve music discovery by providing users with more personalized and accurate recommendations.

Data Set with Lyric-Theme Tagging: A webscraper was used to generate a dataset comprising of roughly 20,000 data points with lyric to theme tagging which can be useful in searching for music based on themes. An ML model was also used to predict the themes of other songs that are not tagged yet.

Broader Impacts

- **Improved Access to Music:** By providing more accurate and personalized music recommendations, TuneMe can improve access to music for users who may have otherwise struggled to find new music that they enjoy. This has the potential to enhance people's overall enjoyment of life.
- **Promoting Cultural Diversity:** TuneMe is capable of finding music in languages other than English if the user opts to. T By recommending music in other languages, TuneMe can expose users to new cultures and promote cross-cultural understanding.

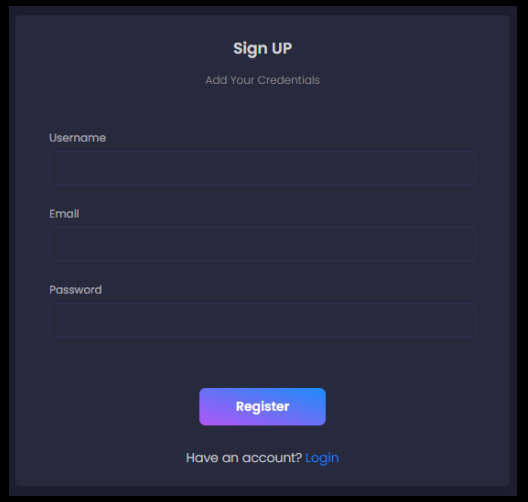
Design Specifications

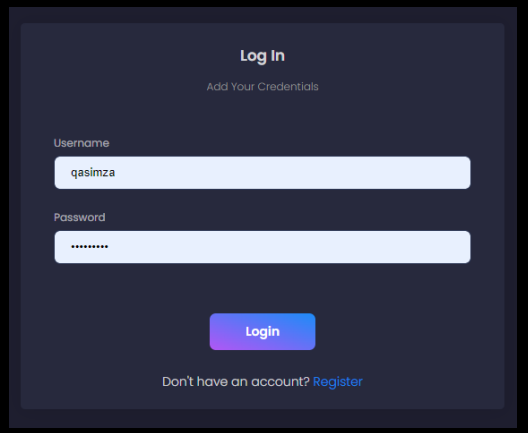


SYSTEM OVERVIEW

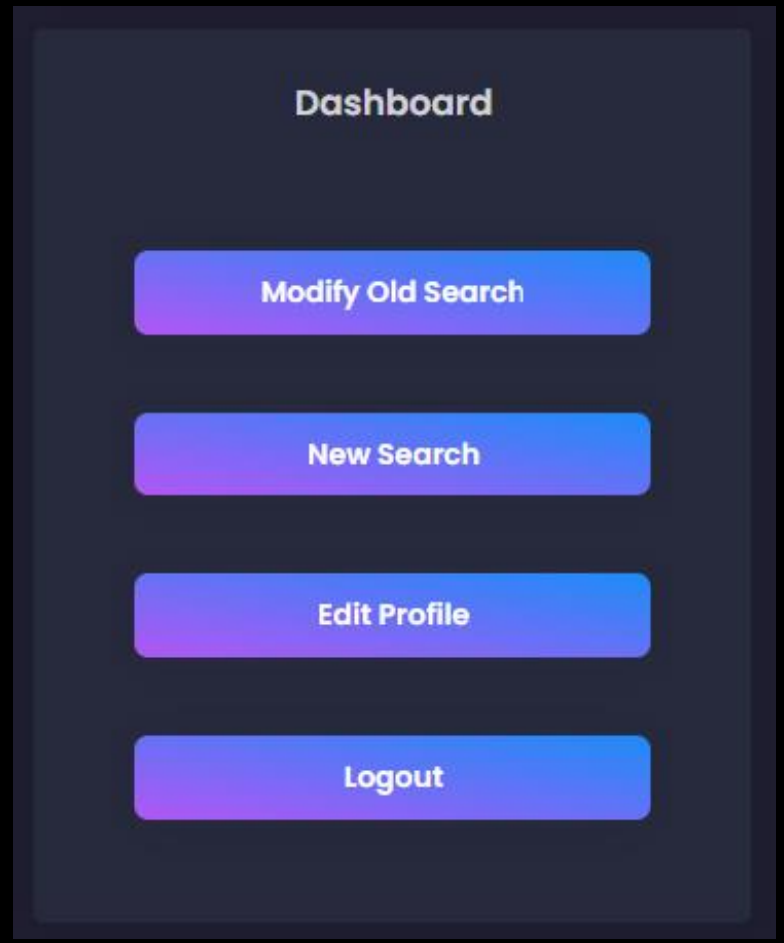
DESIGN DIAGRAMS

System Overview



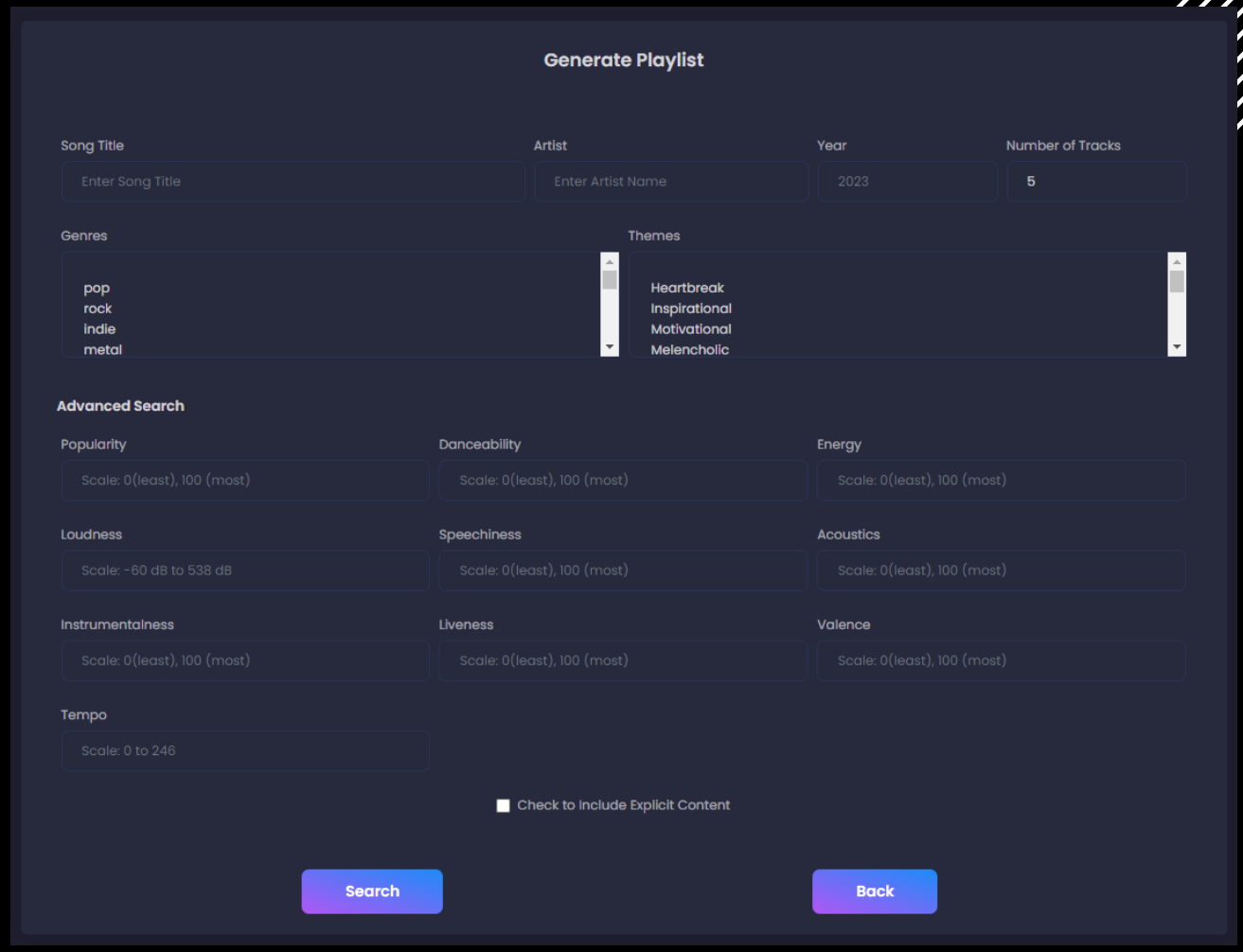


- **Intended Audience/User:**
  - Anyone who wants to find new music.
- **Key features and functionality (Signup, Login, Dashboard):**
  - Simple dashboard accessible after creating an account and logging in.
  - Dashboard links to new search page, old searches and a page to edit profile.



System Overview

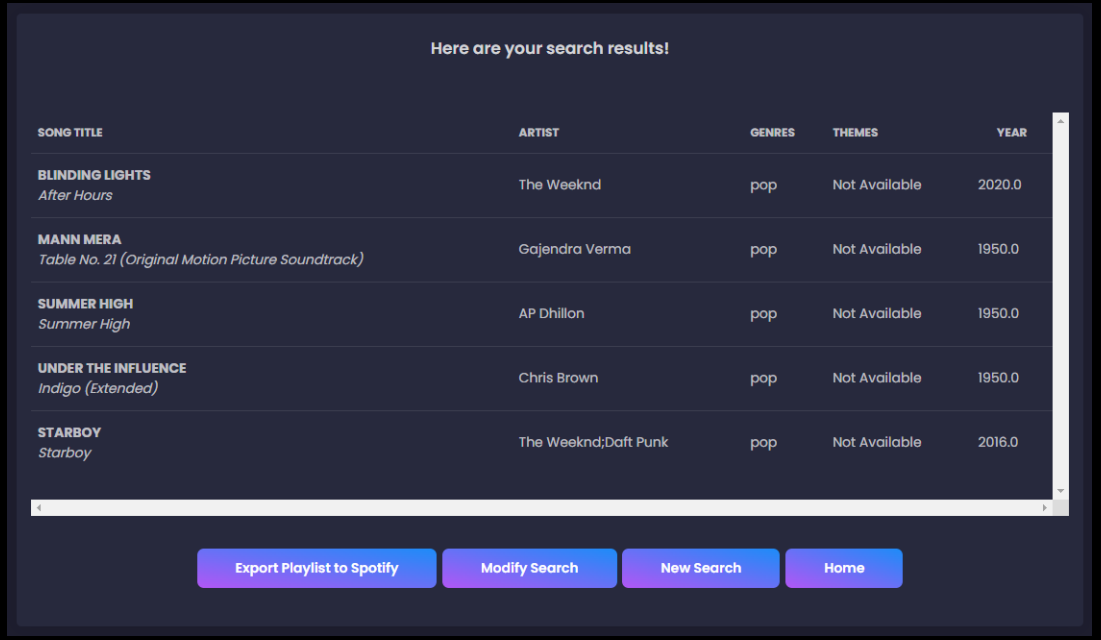
- **Key features and functionality (Search):**
  - Search page that allows users to find music by Song Title, Artist, Genre, Year or Theme (or a combination) with the number of suggestions needed.
  - 10 extra advanced parameters for fine tuning search results –Popularity, Danceability, Energy, Loudness, Speechiness, Acousicness, Instrumentalness, Liveness, Valence, Tempo.
  - Excludes explicit content by default although the user may opt to include it.



System Overview

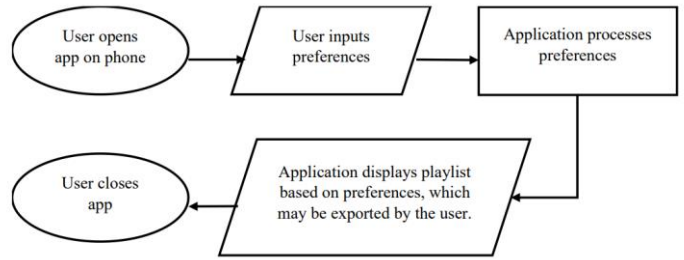
- **Key features and functionality (Results)**
  - Results page displays songs matching user's query.
  - Allows user to export playlist to Spotify, Modify the current search or perform a new one.

- **System's constraints:**
  - Inaccurate/Dissatisfactory results on some occasions.
  - No results for highly complex queries.
  - Results with no themes.

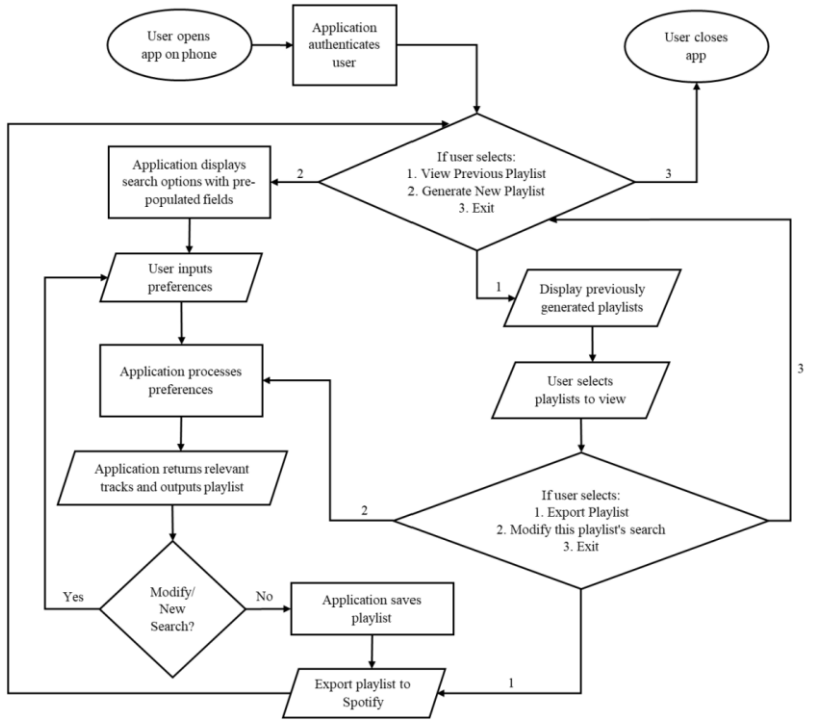


Design Diagrams

D0

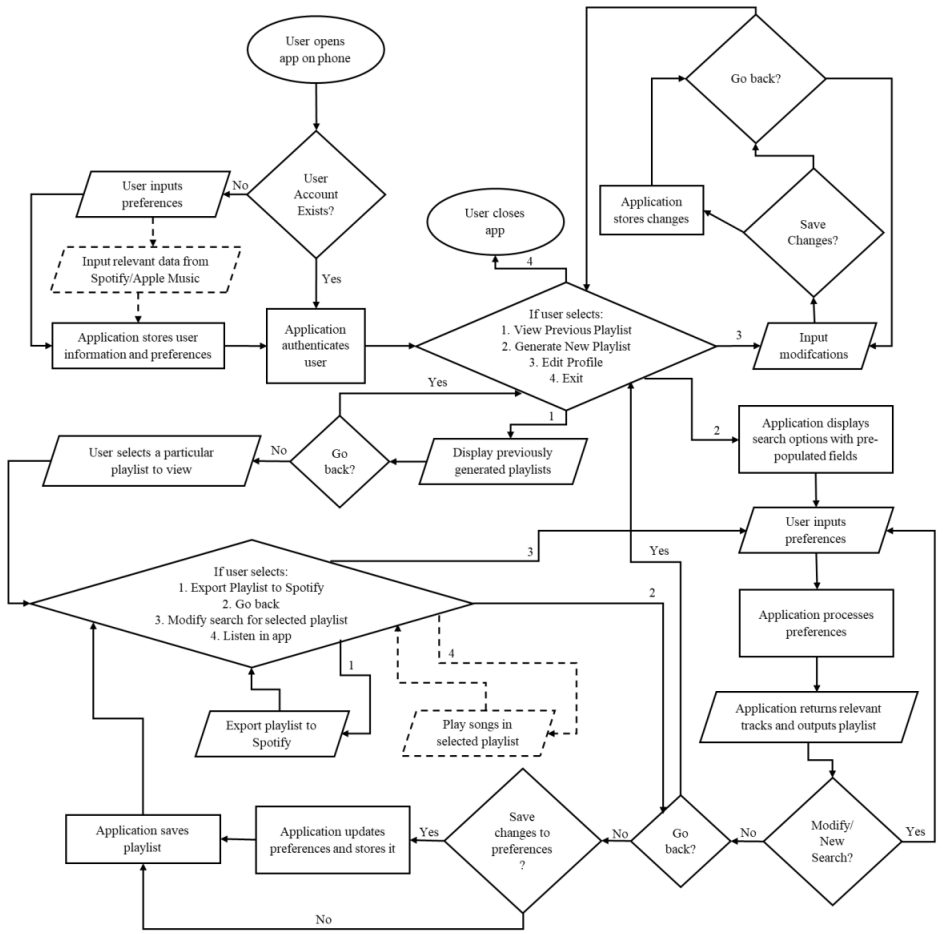


D1



Design Diagrams

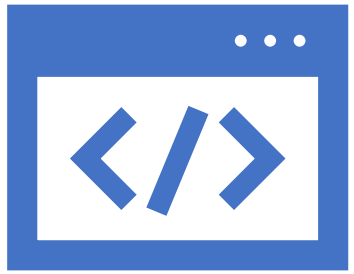
D2



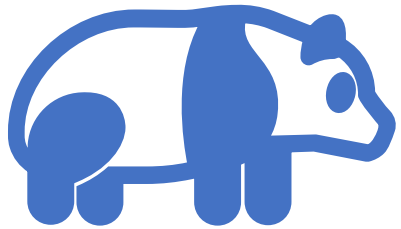
Symbols and Conventions

Shape	Description
Oval	Terminator, used for indicating the start/end of a process.
Rectangle	Used for indicating process steps
Parallelogram	Used for indicating inputs and outputs.
Diamond	Used for indicating decisions, alternative process routes.
Arrow	Used for indicating directional flow.
Solid Line	Used for indicating Requirements (refer Task List).
Dashed Line	Used for indicating Deliverables (refer Task List).

Technologies



Programming Languages – Python, HTML/CSS/JavaScript, SQLite



Frameworks and APIs – Flask, NumPy, Pandas, Matplotlib, scikit-learn, Bootstrap 4, jQuery

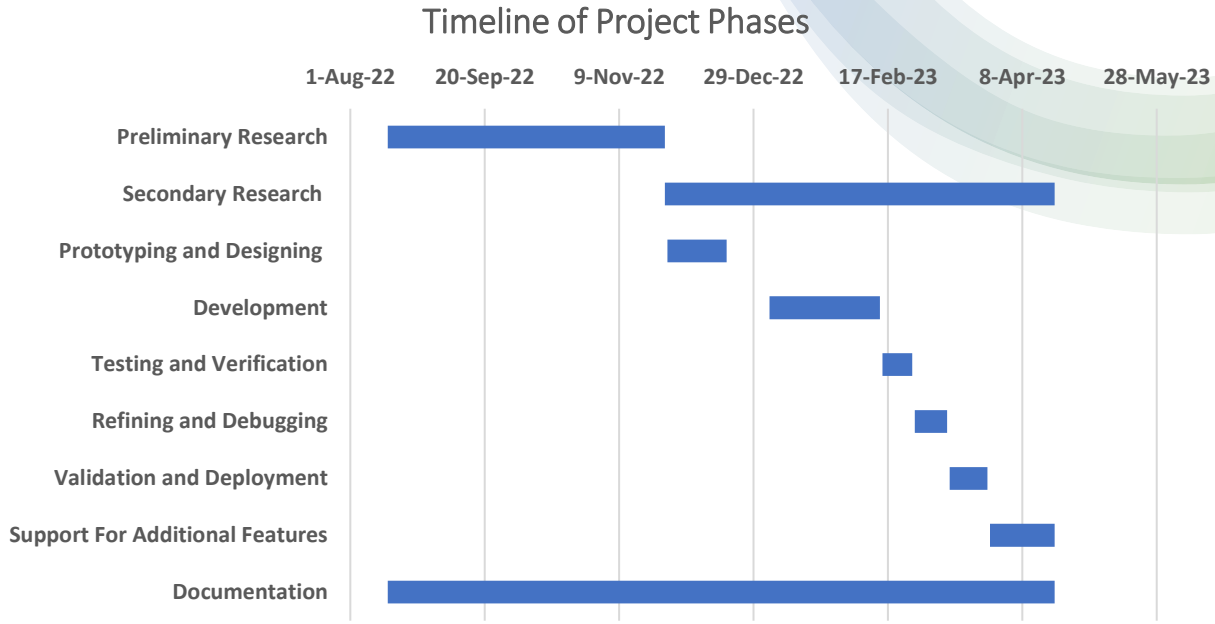
Milestones

Total Days

248

Total Hours

281



Phase	Days to complete	Hours of Effort
Preliminary Research	100	44
Prototyping and Designing	21	25
Development	34	62
Testing and Verification	10	23
Refining and Debugging	12	24
Validation and Deployment	20	20
Develop Support For Additional Features	19	38

Results

- Successful implementation of a Flask application.
- Effectively find music based on a variety of parameters, including artist, year, and song title.
- At least 10 songs/query on average.
- 70 % median accuracy.
- 0.092 seconds/query.

```
================================================================================
Test: Timing Test 1
--------------------------------------------------------------------------------
Start Time: 12:43:30.253485
End Time: 12:45:02.479920
Nunber of queries: 1000
Total Time: 92.226435
Time per query: 0.092226435
================================================================================
```

```
================================================================================
Test: Recommendation System Test 1
--------------------------------------------------------------------------------
Average number of results (rounded to 2 decimal places) 18.27
Median similarity Score (rounded to 2 decimal places): 88.56%
Test Status: PASSED
================================================================================
```

```
================================================================================
Test: Recommendation System Test 2
--------------------------------------------------------------------------------
Average number of results (rounded to 2 decimal places) 18.86
Median similarity Score (rounded to 2 decimal places): 91.04%
Test Status: PASSED
================================================================================
```

```
================================================================================
Test: Recommendation System Test 3
--------------------------------------------------------------------------------
Average number of results (rounded to 2 decimal places) 18.86
Median similarity Score (rounded to 2 decimal places): 91.04%
Test Status: PASSED
================================================================================
```

```
================================================================================
Test: Recommendation System Test 4
--------------------------------------------------------------------------------
Average number of results (rounded to 2 decimal places) 13.2
Median similarity Score (rounded to 2 decimal places): 72.37%
Test Status: PASSED
================================================================================
```

```
================================================================================
Test: Recommendation System Test 5
--------------------------------------------------------------------------------
Average number of results (rounded to 2 decimal places) 15.98
Median similarity Score (rounded to 2 decimal places): 71.09%
Test Status: PASSED
================================================================================
```

```
================================================================================
Test: Recommendation System Test 6
--------------------------------------------------------------------------------
Average number of results (rounded to 2 decimal places) 19.66
Median similarity Score (rounded to 2 decimal places): 62.76%
Test Status: PASSED
================================================================================
```

```
================================================================================
Test: Recommendation System Test 7
--------------------------------------------------------------------------------
Average number of results (rounded to 2 decimal places) 13.34
Median similarity Score (rounded to 2 decimal places): 70.45%
Test Status: PASSED
================================================================================
```

Challenges

Data Cleaning – Several datasets were sourced from Kaggle, a popular data science platform. While these datasets provided valuable information for the project, they also presented some challenges, particularly with regard to data cleaning.

Technical Knowledge and Time - One of the challenges encountered during the TuneMe project was the learning curve and time constraint associated with developing both a frontend and backend application. Initially, the decision was made to separate the frontend using the Ionic framework as an opportunity to learn Android development, while the backend was built using Flask. However, this approach proved to be complicated and time-consuming, leading to the decision to switch to a lightweight Flask application. A cross-disciplinary challenge encountered during the development of the TuneMe application was the need to learn musical terms and concepts in order to effectively categorize and recommend music to users.

Diversity and Cultural Challenges- Music is heavily influenced and differs by cultural attributes, such as language, and instrumentation. Many musical genres are unique to some geographical region or to an ethnic, religious, or linguistic group. Given the extensive nature of music, its it is not feasible to create a solution that is exhaustive and applicable to all kinds of music.

Thank you