MUSIC RECOMMENDATION APPLICATION

PROJECT WORK PHASE-II

PPW481

Submitted by

PRAKHAR SINGH

SIDDHANT RAJHANS

In partial fulfillment of the requirements for
the award of the degree of

BACHELOR OF TECHNOLOGY

Computer Science and Engineering

(CSE Department)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Even Semester, 2022-23

CERTIFICATE

We hereby certify that the work which is being presented in PROJECT REPORT entitled **MUSIC RECOMMENDATION APPLICATION** in partial fulfillment of the requirement for the award of the **Bachelor of Technology in Computer Science & Engineering** and submitted to the Department of Computer Science & Engineering of Himalayan School of Science & Technology, Swami Rama Himalayan University, Jolly grant, Dehradun (U.K) is an authentic record of our own work carried out during a period from January 2023 to June 2023 under the supervision of **Mr. GAURAV SHARMA**, Assistant Professor of Computer Science & Engineering, Himalayan School of Science & Technology, Swami Rama Himalayan University, Jolly grant, Dehradun (U.K).

The matter presented in this report has not been submitted by us for the award of any other degree elsewhere.

PRAKHAR SINGH

SIDDHANT RAJHANS

This is to certify that the above statement is covered to the best of my knowledge.

SIDDHANT RAJHANS

ABSTRACT

The proliferation of digital music platforms and the vast music library available to users have made music recommendation systems indispensable for enhancing user experience and facilitating music discovery. This final year project presents a **MUSIC RECOMMENDATION APPLICATION (MRA)** which provides personalized and accurate music recommendations The recommendation system handles the abundance of information by filtering the most crucial information based on the information provided by a user and other criteria that take into account the user's choice and interest. It determines whether a user and an item are compatible and then assumes that they are similar in order to make recommendations. The MRA employs a **hybrid approach, integrating a popularity-based recommendation system with item similarity-based content filtering techniques**. This hybrid approach aims to provide personalized recommendations while taking into account the overall popularity of items. The final recommendation model is included into an approachable mobile application, enabling simple access to the music recommendation system. Users can find and explore music that appeals to their specific tastes thanks to the recommendation system's smooth integration into a mobile application, which improves accessibility. The effectiveness of the music recommendation app is demonstrated by its capacity to provide high-quality, personalised music recommendations that increase user engagement and pleasure. This project contributes to the field of music recommendation systems by developing an efficient and user-centric application.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 Introduction to Recommendation Systems

A recommender system, or a recommendation system, is a subclass of information filtering system that seeks to predict the “rating” or “preference” a user would give to an item. They are primarily used in commercial applications. Examples of such applications include recommending products on Amazon, music on Spotify, and stories on Medium. The famous The Netflix Prize is also a competition in the context of recommendation systems.

From 2006 to 2009, Netflix sponsored a competition, offering a grand prize of $1,000,000 to the team that could take an offered dataset of over 100 million movie ratings and return recommendations that were 10% more accurate than those offered by the company's existing recommender system. This competition energized the search for new and more accurate algorithms. On 21 September 2009, the grand prize of US$1,000,000 was given to the BellKor's Pragmatic Chaos team using tiebreaking rules [1].

Recommendation systems are information filtering system that aids users in predicting rating or preference of an item under users’ consideration. The systems offer users alternate selections without having to work out all the details by themselves. As overwhelming information explosion renders searching, extraction, analysis, and processing hideous and formidably time-consuming operations, recommender systems became a favourable decision tool or assistant to offload such undesirable tasks [2].

1.2 Project Category

The project falls under the category of "Application or System Development." The objective of the project is to develop a Music Recommendation Application, which involves designing and implementing a software application to provide music recommendations to users.

1.3 Objective

The objective of this project is to develop a Music Recommender Application that enhances the music discovery process for users. By combining popularity-based recommendation with item similarity-based content filtering, the project aims to deliver personalized music recommendations that align with users' individual preferences while also introducing them to popular and trending music. The application will be user-centric, offering a seamless and intuitive interface to enhance user engagement and satisfaction. The project aims to utilize these recommendation models to suggest songs to users based on their preferences, helping users discover new songs and enhancing their music listening experience.

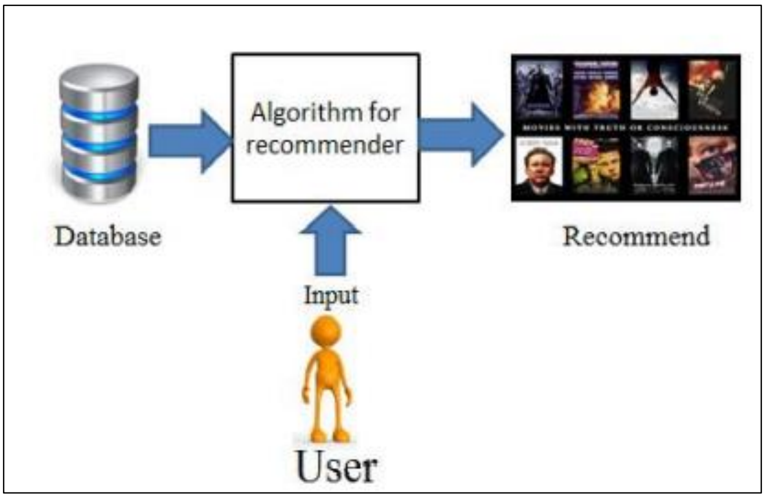


Fig 1.1 General Recommendation system [3]

1.4 Problem Formulation

The problem addressed by this project is the overwhelming abundance of music available to users, which often leads to difficulties in discovering music that aligns with their personal tastes and preferences. Existing music recommendation systems often rely solely on popularity-based recommendations or fail to provide highly personalized suggestions.

1.5 Identification / Reorganization of Need

There is a need for a music recommendation application that combines popularity-based recommendations with item similarity-based content filtering to deliver accurate and personalized music recommendations to users. The Music Recommendation Application aims to provide a solution that enables users to easily discover and explore new music that aligns with their individual preferences. The project seeks to develop an application that offers a seamless and personalized music discovery experience, enhancing user satisfaction and engagement.

1.6 Existing Systems

There are several existing music recommendation systems that address the problem of personalized music recommendations. Some popular examples include Spotify, Apple Music, and Pandora.

1.7 Proposed System

The proposed **Music Recommendation Application** (MRA) is a sophisticated and user-centric solution designed to address the challenges of music discovery and personalization. By combining popularity-based recommendations with item similarity-based content filtering, the MRA offers a unique and balanced approach to suggest music to users. The system analyses user preferences and behaviour to provide tailored recommendations that align with individual tastes, allowing users to explore new music while also introducing them to popular and trending tracks.

CHAPTER 2: LITERATURE REVIEW

2.1 Popularity Based Recommendation System

Popularity-based filtering is a type of recommendation system that suggests items to users based on the popularity of the item. These systems are relatively simple to implement, as they only require a count of the number of times an item has been viewed or purchased by all users. These systems do not take into account the user's preferences or interests and simply recommend the most popular items to all users. Popularity-based filtering is easy to implement, but it can have a number of drawbacks. The most popular items may not always be the best fit for a particular user. And it can lead to a "cold start problem" for new items, as they will not have been purchased or viewed enough times to be considered popular. Popularity-based systems do not take into account demographic factors such as age or location of the user which might change the preferences for the items. These systems also might not be good for niche items or items that are not popular but might be of interest to a specific user. In certain contexts, such as music or video streaming service where the popularity of a certain item is well-known, it can be a good way to get suggestions for the user.

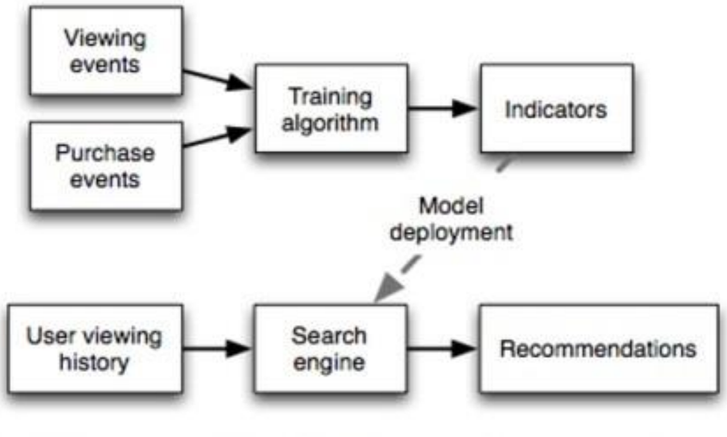


Fig.2.1 Architecture of Popularity Based Recommendation System [4]

2.2 Content-Based Filtering

Content-based filtering is a method of recommending items to users based on their past preferences and the characteristics of items they have previously consumed. The idea behind content-based filtering is to recommend items that are similar to items that a user has liked in the past. The approach typically starts by representing each item in the system as a set of features, or attributes. These features could be things like the genre of a movie, the actors in a movie, or the topics covered in a news article. Once the items have been represented in this way, the system can use them to determine which items are similar to each other. The system creates a profile for each user based on their past preferences. This profile can be represented as a vector of weights, where the weight of a feature represents the user's preference for that feature. Once a user's profile has been created, the system can use it to find other items that are similar to the items that the user has liked in the past.

Item similarity content-based filtering is a technique used in recommendation systems to personalize suggestions. It works by analyzing item features, such as descriptions or tags, to find similarities between items. First, relevant features are extracted from the items. Then, similarity measures like cosine similarity or Jaccard similarity are used to calculate the similarity between items. Recommendations are generated by identifying items similar to those the user has interacted with. These candidates are ranked based on relevance or user feedback, and the top items are recommended. This approach is advantageous when data is limited and can provide personalized suggestions.

2.3 Hybrid Recommendation System

A hybrid recommender system is one that combines the advantages of different types of recommendation algorithms to provide more accurate and diverse recommendations. The hybrid system can leverage the strengths of both methods to overcome their respective weaknesses. For example, collaborative filtering can handle the cold-start problem, by finding similar users, while popularity-based filtering can handle the problem of data sparsity, by recommending popular items. A hybrid recommendation system that combines popularity and item similarity content-based filtering operates by leveraging the strengths of both approaches to provide more accurate and diverse recommendations. The popularity component takes into account the overall popularity of items among users, considering factors such as purchase history, ratings, and reviews. This approach ensures that popular and widely accepted items are recommended to users, catering to their general preferences. On the other hand, the item similarity content-based filtering analyzes the characteristics and attributes of items, identifying similarities based on content features such as genre, keywords, or tags. By considering the similarity between items, the system suggests items that are closely related to those previously liked or rated positively by the user. This approach enables personalized recommendations that align with specific user preferences. By combining these two approaches, the hybrid system achieves a balanced recommendation strategy that provides both popular and relevant item suggestions, enhancing user satisfaction and discovery of new items.

CHAPTER 3: REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION

3.1 Feasibility Study

The feasibility study conducted for this project indicates a high level of feasibility and potential success. The implementation of the Music Recommendation Application (MRA) is feasible using readily accessible mobile application development frameworks and tools. From an economic perspective, the project shows promise in attracting a large user base and can generate revenue through monetization strategies. Socially and ethically, the project aims to enhance user experience. Overall, the feasibility study indicates that the project is viable and has the potential to deliver a successful and valuable Music Recommendation Application for enhancing the music discovery experience.

3.2 Software Requirement Specification

- Python: Version 3.7 or later [5]
- Flask: Version 2.0.1 or later [6]
- Flutter 3.10.2: - Tools: - Dart 3.0.2: - DevTools 2.23.1[7]
- IDE: - Visual Studio Code: Version 1.79 [8]

3.3 Expected Hurdles

While this project has promising potential, there are several hurdles that may be encountered during its development and implementation. Some of the expected hurdles may include data availability, scalability and performance, privacy and security concerns, user feedback and evaluation, continuous system adaptation etc.

3.4 SDLC Model to be used

Initially in the project, we chose the Waterfall Software Development Life Cycle (SDLC) model to establish a structured approach for planning, requirements gathering, and design. However, as the project progressed into development and testing, a transition to the Spiral model was made. This shift was motivated by the project's hybrid approach, which requires flexibility and iterative refinement. The Spiral model's incremental development, prototyping, and continuous feedback align well with the evolving nature of the project, ensuring user satisfaction and adaptability throughout the development process.

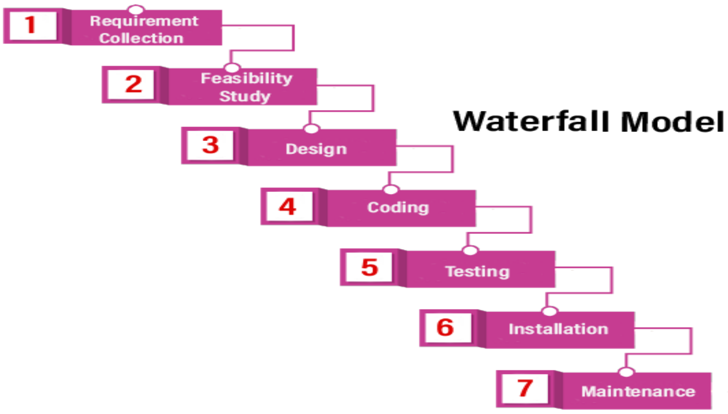


Fig 3.1 Waterfall Model [9] (Source: https://www.javatpoint.com)

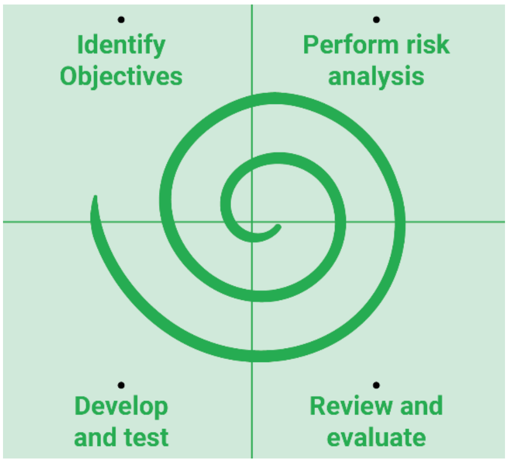


Fig 3.2 Spiral Model [10] (Source: https://walkingtree.tech)

CHAPTER 4: SYSTEM DESIGN

4.1 Design approach

The design approach of the Music Recommendation Application project is primarily Object-oriented, which emphasizes modular design, code reusability, and the organization of functionality into objects and classes. This approach allows for easier management of relationships, promotes extensibility, and enhances code maintainability.

4.2 System design using Data flow Diagram

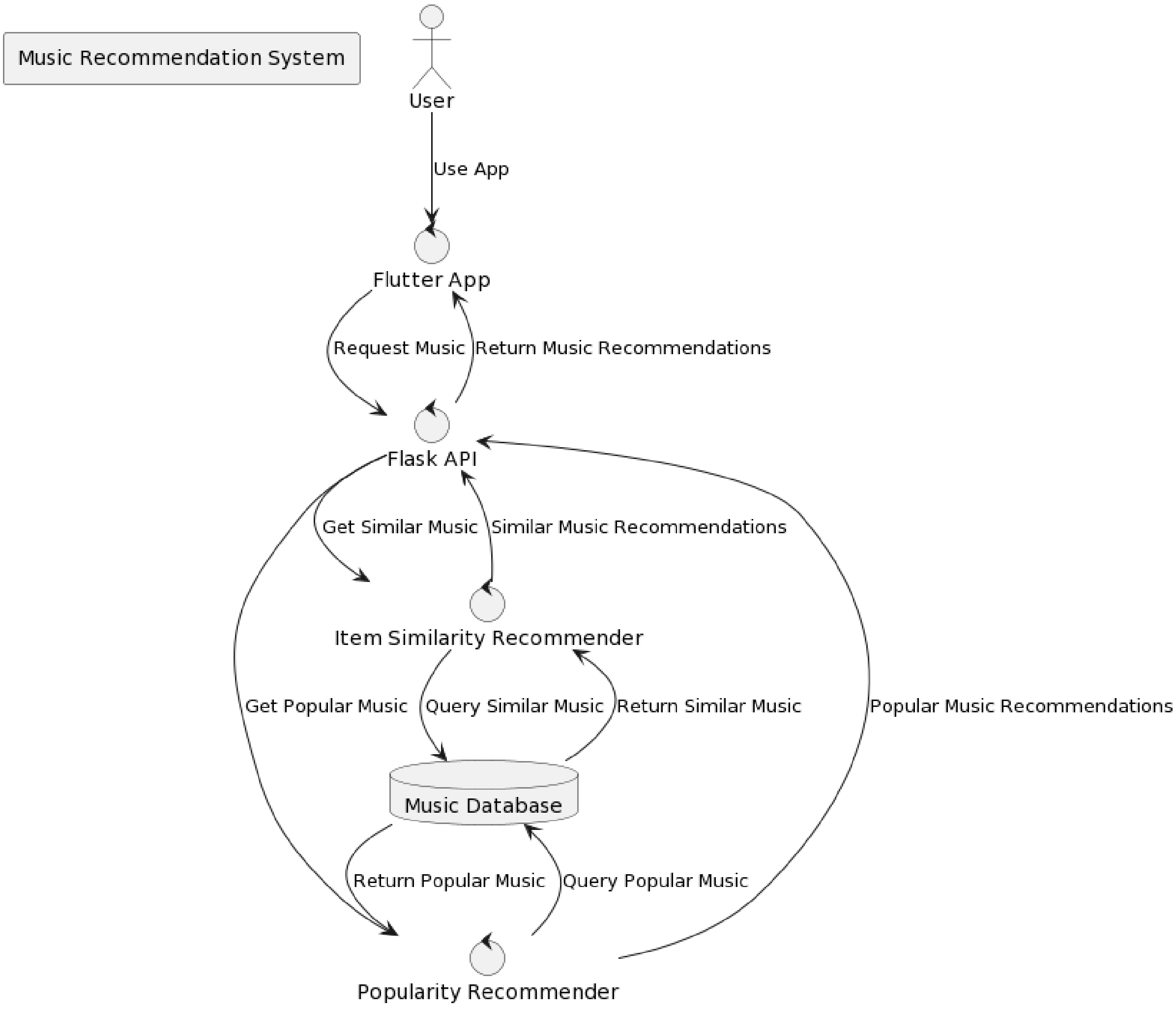


Fig 4.1 Data Flow Diagram [11]

CHAPTER 5: IMPLEMENTATION & TESTING

5.1 Introduction to Languages, IDE's, Tools and Technologies used for Implementation

5.1.1 Python

Python is a high-level, interpreted programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991. Python emphasizes code readability and has a design philosophy that emphasizes clear and concise syntax, making it easy to understand and write code. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python has gained popularity due to its versatility and extensive standard library, which provides a wide range of pre-built modules and functions for various tasks. It has a strong developer community and a vast ecosystem of third-party libraries and frameworks, making it suitable for diverse applications such as web development, scientific computing, data analysis, machine learning, and automation.

5.1.2 Flask

Flask is a micro web framework written in Python. Flask is a web framework that allows developers to build lightweight web applications quickly and easily with Flask Libraries. It was developed by Armin Ronacher, leader of the International Group of Python Enthusiasts (POCCO). It is basically based on the WSGI toolkit and Jinja2 templating engine. Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. [12][13]

5.1.3 Flutter

Flutter is a simple and high-performance framework based on Dart language, provides high performance by rendering the UI directly in the operating system's canvas rather than through native framework. Flutter also offers many ready to use widgets (UI) to create a modern application. These widgets are optimized for mobile environment and designing the application using widgets is as simple as designing HTML. [14]

5.1.4 Dart

Dart is an open-source general-purpose programming language. It is originally developed by Google. Dart is an object-oriented language with C-style syntax. It supports programming concepts like interfaces, classes, unlike other programming languages Dart doesn't support arrays. Dart collections can be used to replicate data structures such as arrays, generics, and optional typing. [15]

5.1.5 Firebase

Firebase is a set of backend cloud computing services and application development platforms provided by Google. It hosts databases, services, authentication, and integration for a variety of applications, including Android, iOS, JavaScript, Node.js, Java, Unity, PHP, and C++. [16]

5.1.6 Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET). [17]

5.2 Coding Standards of Languages used

The coding standards for the languages used in the project, specifically Python and Dart (used with Flutter), play a vital role in ensuring code readability, maintainability, and consistency across the project. While coding standards can vary based on preferences and guidelines, here are some commonly followed practices for each language:

5.2.1 Python

- PEP 8: Following the Python Enhancement Proposal (PEP) 8 style guide, which outlines conventions for code layout, naming conventions, and code organization.
- Indentation: Using four spaces for indentation to enhance code readability and adhering to consistent indentation throughout the codebase.
- Variable and Function Naming: Employing descriptive and meaningful names for variables, functions, and classes following lowercase_with_underscores for variable and function names and CamelCase for class names.
- Comments: Including clear and concise comments to explain the purpose, functionality, and any complex sections of code.
- Modularization: Encouraging modular design and breaking down code into reusable functions or classes to promote code reusability and maintainability.
- Error Handling: Implementing appropriate exception handling mechanisms to handle potential errors or exceptions and ensure robustness in the code.
- Documentation: Providing docstrings for classes, functions, and modules to document their purpose, parameters, return values, and usage.

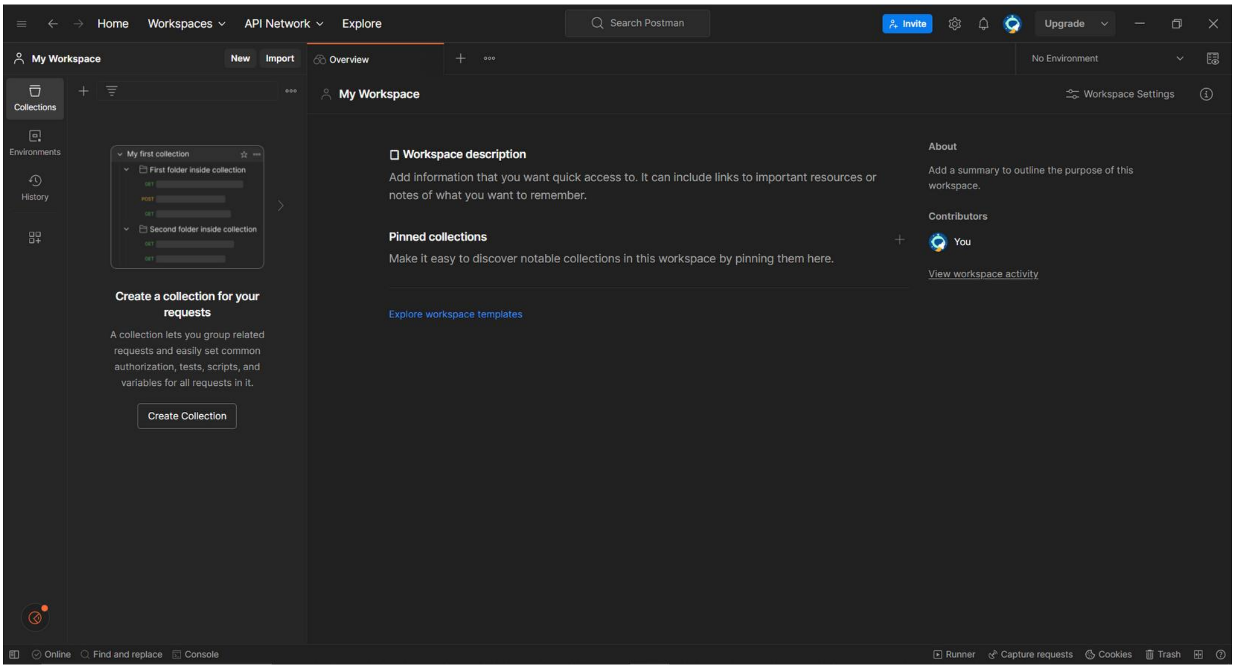
5.2.2 Dart (Flutter)

- Effective Dart: Following the guidelines provided in the Effective Dart style guide, which offers recommendations for code formatting, naming conventions, and best practices for writing clean and efficient Dart code.
- Indentation: Using two spaces for indentation to maintain consistent code formatting.
- Variable and Function Naming: Utilizing descriptive names using lowercase_with_underscores for variables and camelCase for functions and methods.
- Widget Composition: Breaking down complex UI components into smaller, reusable widgets to promote code reusability and maintainability.
- Null Safety: Adhering to null safety principles introduced in Dart 2.12 or later, using null safety operators and proper handling of nullable and non-nullable variables.
- Asynchronous Programming: Utilizing asynchronous programming features like async/await and Future objects for handling asynchronous operations and promoting responsive user interfaces.
- Documentation: Adding documentation using comments or annotations to provide insights into the purpose, usage, and parameters of functions, methods, and classes.

5.3 Testing the API

In this project for testing the API, we have used the Postman tool. Postman is a popular API development and testing tool that provides a user-friendly interface for interacting with APIs. To test a Flask API using Postman, start by launching the Postman application and creating a new request. Specify the HTTP method (such as GET, POST, PUT, or DELETE) and the API endpoint URL we want to test. If required, include any necessary headers, parameters, or request body in the request configuration. Then, click the "Send" button to make the request to the Flask API. Postman will display the response received, including the status code, headers, and body.

We can also inspect the response details, such as JSON or XML data, by using Postman's built-in viewer. This allows us to verify that the Flask API is functioning correctly and returning the expected results. Postman's intuitive interface and powerful features make it an excellent tool for testing and debugging Flask APIs, ensuring their reliability and performance. By leveraging the capabilities of Postman, we can thoroughly test our Flask API and ensure its reliability, security, and performance.



Workspace description

Add information that you want quick access to. It can include links to important resources or notes of what you want to remember.

Pinned collections

Make it easy to discover notable collections in this workspace by pinning them here.

Explore workspace templates

Create a collection for your requests

A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.

About

Add a summary to outline the purpose of this workspace.

Contributors

You

View workspace activity

Fig 5.1 Postman Tool for testing Flask API [18]

CHAPTER 6: RESULTS AND DISCUSSIONS

6.1 User Interface Representation

We have created the music recommendation application using dart language and Flutter framework.

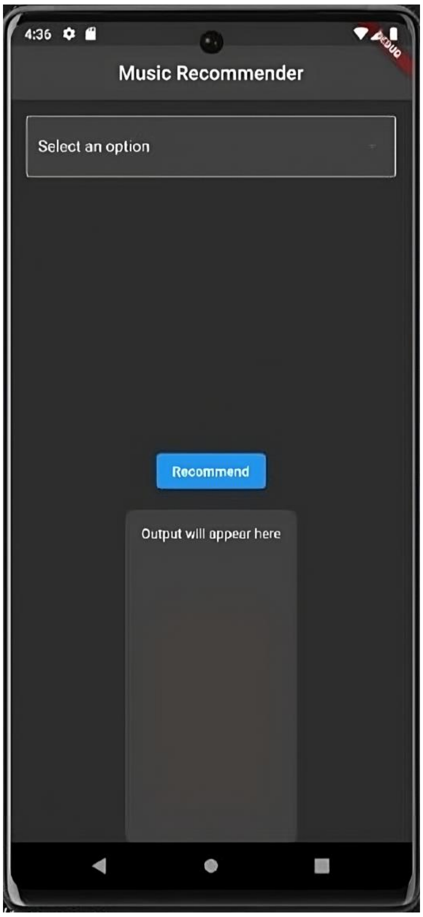
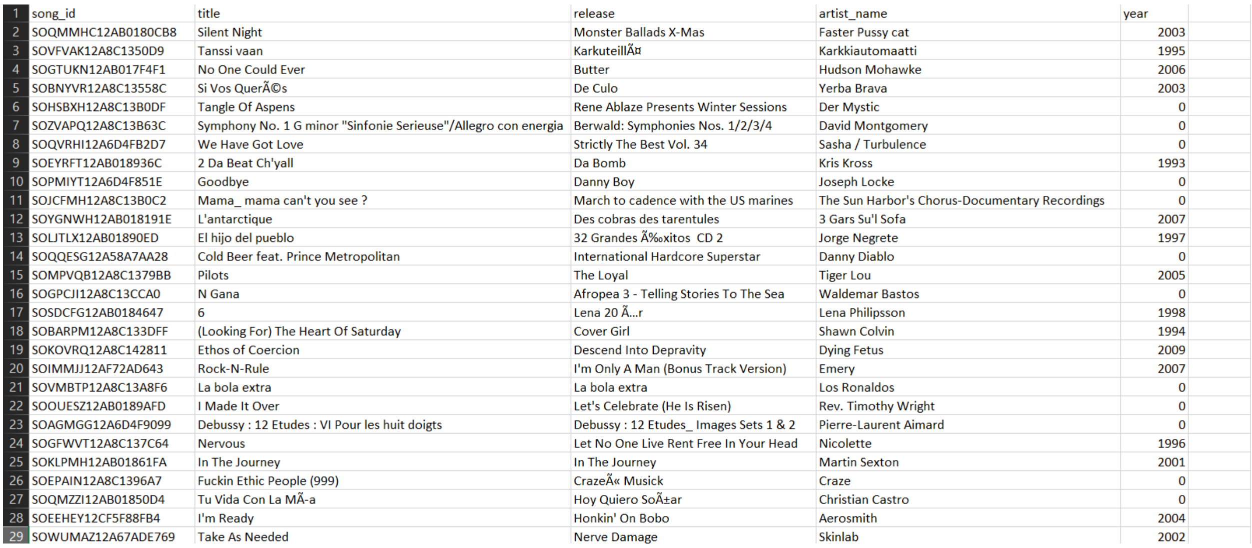


Fig 6.1 User Interface Representation

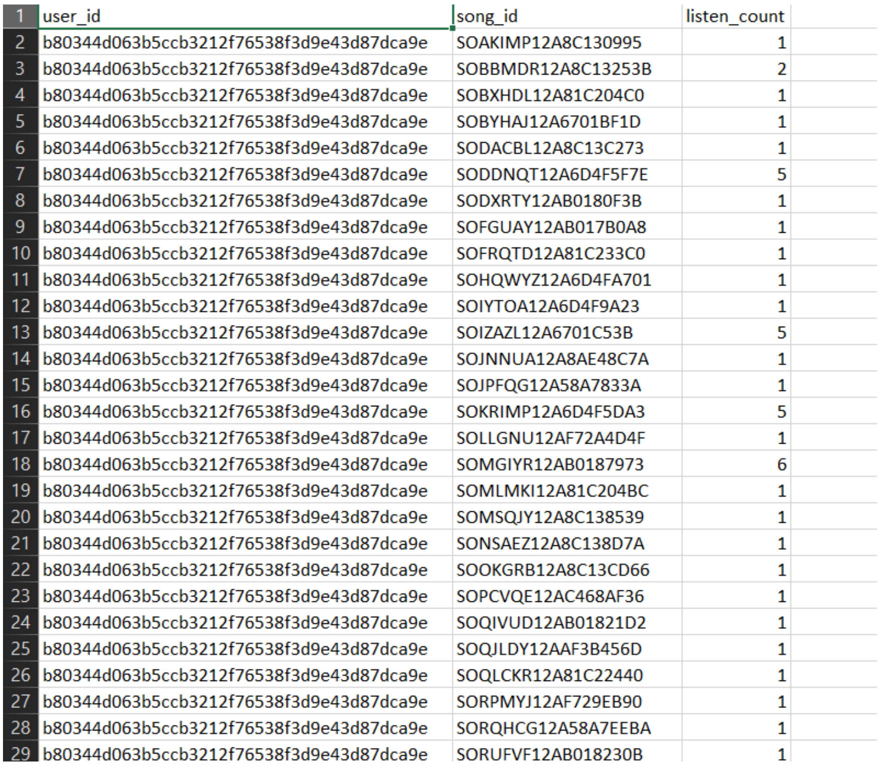
6.2 Brief Description of Various Modules of the system with Snapshots

The dataset we have used is million songs dataset subset and it contained H5 files which we then pre-processed into 2 .csv files namely songdata.csv and triplets.csv. Songdata.csv contains attributes like song id, title, release, artist name and year whereas triplets.csv contains user id, song id and listen count.



1	song_id	title	release	artist_name	year
2	SOQMMHC12AB0180CB8	Silent Night	Monster Ballads X-Mas	Faster Pussy cat	2003
3	SOVFVAK12A8C1350D9	Tanssi vaan	KarkuteillÃ¤	Karkkiautomaatti	1995
4	SOGTUKN12AB017F4F1	No One Could Ever	Butter	Hudson Mohawke	2006
5	SOBNYVR12A8C13558C	Si Vos QuerÃ©s	De Culo	Yerba Brava	2003
6	SOHSBXH12A8C13B0DF	Tangle Of Aspens	Rene Ablaze Presents Winter Sessions	Der Mystic	0
7	SOZVAPQ12A8C13B63C	Symphony No. 1 G minor "Sinfonie Serieuse"/Allegro con energia	Berwald: Symphonies Nos. 1/2/3/4	David Montgomery	0
8	SOQVRHI12A6D4FB2D7	We Have Got Love	Strictly The Best Vol. 34	Sasha / Turbulence	0
9	SOEYRFT12AB018936C	2 Da Beat Ch'yall	Da Bomb	Kris Kross	1993
10	SOPMIYT12A6D4F851E	Goodbye	Danny Boy	Joseph Locke	0
11	SOJCFMH12A8C13B0C2	Mama_ mama can't you see ?	March to cadence with the US marines	The Sun Harbor's Chorus-Documentary Recordings	0
12	SOYGNWH12AB018191E	L'antarctique	Des cobras des tarentules	3 Gars Su'l Sofa	2007
13	SOLJTLX12AB01890ED	El hijo del pueblo	32 Grandes Ã‰xitos CD 2	Jorge Negrete	1997
14	SOQQESG12A58A7AA28	Cold Beer feat. Prince Metropolitan	International Hardcore Superstar	Danny Diablo	0
15	SOMPVQB12A8C1379BB	Pilots	The Loyal	Tiger Lou	2005
16	SOGPCJI12A8C13CCA0	N Gana	Afropea 3 - Telling Stories To The Sea	Waldemar Bastos	0
17	SOSDCFG12AB0184647	6	Lena 20 Ã…r	Lena Philipsson	1998
18	SOBARPM12A8C133DFF	(Looking For) The Heart Of Saturday	Cover Girl	Shawn Colvin	1994
19	SOKOVRQ12A8C142811	Ethos of Coercion	Descend Into Depravity	Dying Fetus	2009
20	SOIMMJJ12AF72AD643	Rock-N-Rule	I'm Only A Man (Bonus Track Version)	Emery	2007
21	SOVMBTP12A8C13A8F6	La bola extra	La bola extra	Los Ronaldos	0
22	SOOUESZ12AB0189AFD	I Made It Over	Let's Celebrate (He Is Risen)	Rev. Timothy Wright	0
23	SOAGMGG12A6D4F9099	Debussy : 12 Etudes : VI Pour les huit doigts	Debussy : 12 Etudes_ Images Sets 1 & 2	Pierre-Laurent Aimard	0
24	SOGFWVT12A8C137C64	Nervous	Let No One Live Rent Free In Your Head	Nicolette	1996
25	SOKLPMH12AB01861FA	In The Journey	In The Journey	Martin Sexton	2001
26	SOEPAIN12A8C1396A7	Fuckin Ethic People (999)	CrazeÃ« Musick	Craze	0
27	SOQMZZI12AB01850D4	Tu Vida Con La MÃ-a	Hoy Quiero SoÃ±ar	Christian Castro	0
28	SOEEHEY12CF5F88FB4	I'm Ready	Honkin' On Bobo	Aerosmith	2004
29	SOWUMAZ12A67ADE769	Take As Needed	Nerve Damage	Skinlab	2002

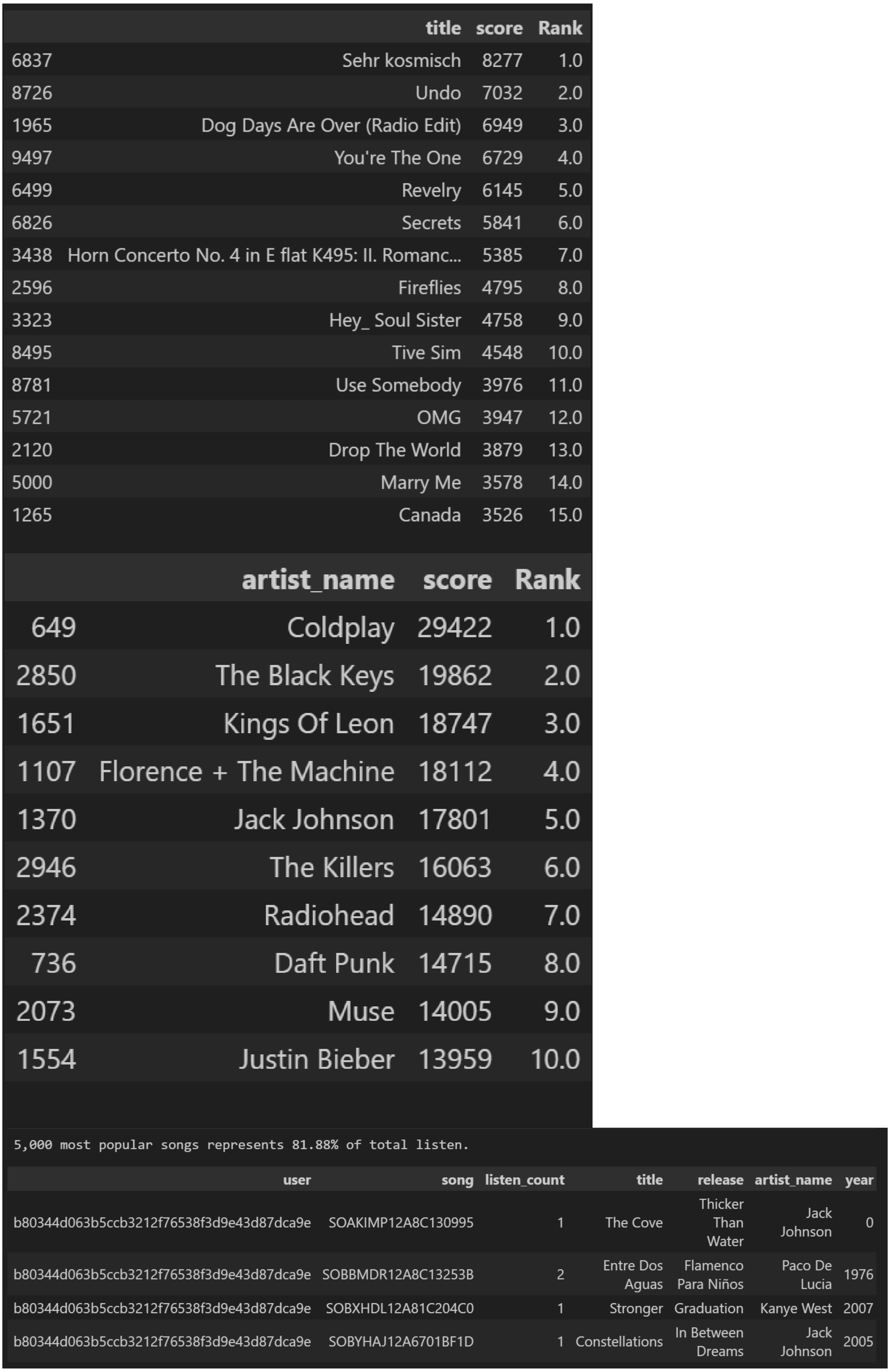
Fig 6.2 Song data CSV file



1	user_id	song_id	listen_count
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1
5	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1
6	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODACBL12A8C13C273	1
7	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODDNQT12A6D4F5F7E	5
8	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODXRTY12AB0180F3B	1
9	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOFGUAY12AB017B0A8	1
10	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOFRQTD12A81C233C0	1
11	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOHQWYZ12A6D4FA701	1
12	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOIYTOA12A6D4F9A23	1
13	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOIZAZL12A6701C53B	5
14	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOJNNUA12A8AE48C7A	1
15	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOJPFQG12A58A7833A	1
16	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOKRIMP12A6D4F5DA3	5
17	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOLLGNU12AF72A4D4F	1
18	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOMGIYR12AB0187973	6
19	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOMLMKI12A81C204BC	1
20	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOMSQJY12A8C138539	1
21	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SONSAEZ12A8C138D7A	1
22	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOOKGRB12A8C13CD66	1
23	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOPCVQE12AC468AF36	1
24	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOQIVUD12AB01821D2	1
25	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOQJLDY12AAF3B456D	1
26	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOQLCKR12A81C22440	1
27	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SORPMYJ12AF729EB90	1
28	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SORQHCG12A58A7EEBA	1
29	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SORUFVF12AB018230B	1

Fig 6.3 Triplets CSV file

After all the pre-processing was done, the hybrid model is created with popularity recommendation system and item similarity-based recommendation system which shows the recommendations as shown in the following figures.



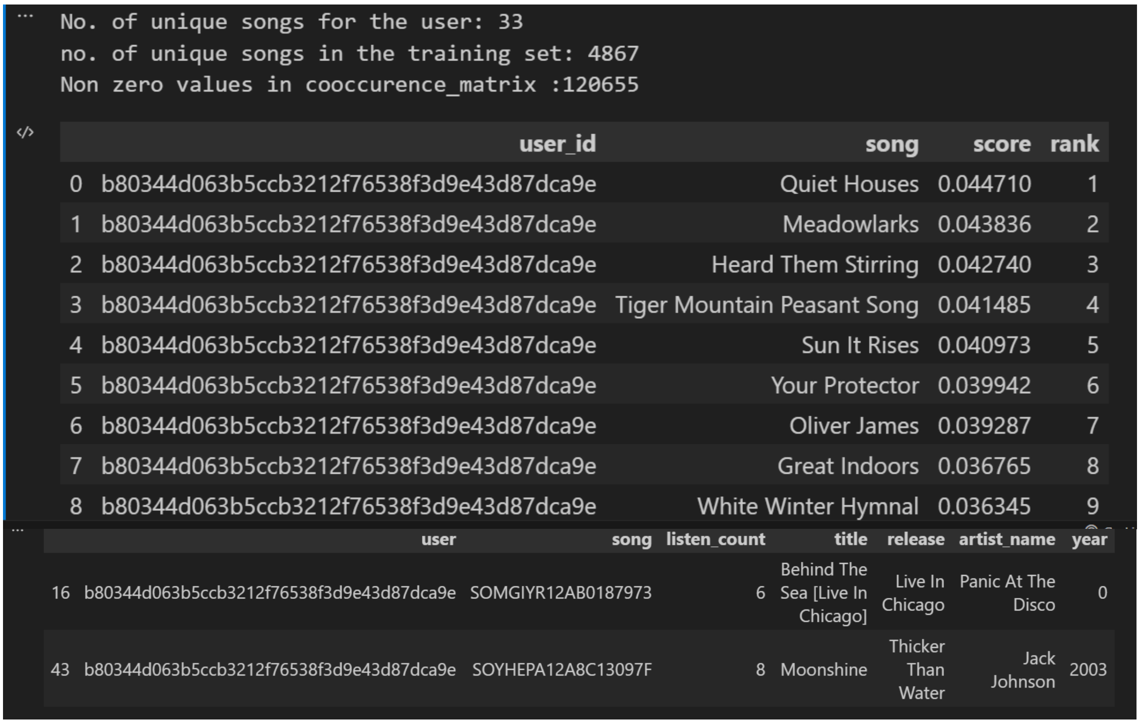
	title	score	Rank
6837	Sehr kosmisch	8277	1.0
8726	Undo	7032	2.0
1965	Dog Days Are Over (Radio Edit)	6949	3.0
9497	You're The One	6729	4.0
6499	Revelry	6145	5.0
6826	Secrets	5841	6.0
3438	Horn Concerto No. 4 in E flat K495: II. Romanc...	5385	7.0
2596	Fireflies	4795	8.0
3323	Hey_ Soul Sister	4758	9.0
8495	Tive Sim	4548	10.0
8781	Use Somebody	3976	11.0
5721	OMG	3947	12.0
2120	Drop The World	3879	13.0
5000	Marry Me	3578	14.0
1265	Canada	3526	15.0

	artist_name	score	Rank
649	Coldplay	29422	1.0
2850	The Black Keys	19862	2.0
1651	Kings Of Leon	18747	3.0
1107	Florence + The Machine	18112	4.0
1370	Jack Johnson	17801	5.0
2946	The Killers	16063	6.0
2374	Radiohead	14890	7.0
736	Daft Punk	14715	8.0
2073	Muse	14005	9.0
1554	Justin Bieber	13959	10.0

```
5,000 most popular songs represents 81.88% of total listen.
```

user	song	listen_count	title	release	artist_name	year
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1	The Cove	Thicker Than Water	Jack Johnson	0
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2	Entre Dos Aguas	Flamenco Para Niños	Paco De Lucia	1976
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1	Stronger	Graduation	Kanye West	2007
b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1	Constellations	In Between Dreams	Jack Johnson	2005

Fig 6.4 Popularity Based Recommendations



```
No. of unique songs for the user: 33
no. of unique songs in the training set: 4867
Non zero values in cooccurence_matrix :120655
```

	user_id	song	score	rank
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Quiet Houses	0.044710	1
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Meadowlarks	0.043836	2
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Heard Them Stirring	0.042740	3
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Tiger Mountain Peasant Song	0.041485	4
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Sun It Rises	0.040973	5
5	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Your Protector	0.039942	6
6	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Oliver James	0.039287	7
7	b80344d063b5ccb3212f76538f3d9e43d87dca9e	Great Indoors	0.036765	8
8	b80344d063b5ccb3212f76538f3d9e43d87dca9e	White Winter Hymnal	0.036345	9

	user	song	listen_count	title	release	artist_name	year
16	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOMGIYR12AB0187973	6	Behind The Sea [Live In Chicago]	Live In Chicago	Panic At The Disco	0
43	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOYHEPA12A8C13097F	8	Moonshine	Thicker Than Water	Jack Johnson	2003

Fig 6.5 Item-Similarity Based Recommendations

6.3 Results

The project was successful in creating a functional and user-friendly music recommendation system. The application was implemented using a combination of popularity-based recommendation and item similarity-based content filtering techniques. The integration of popularity-based recommendation algorithms ensured that users were exposed to popular and trending music within their preferred genres. Additionally, the item similarity-based content filtering approach enhanced the recommendation accuracy by suggesting songs similar to those the user has already liked or listened to. The Flutter framework was used to create the application's user interface, which offered an intuitive and aesthetically pleasing experience. The intuitive design and interactive features of the application contributed to a positive user experience. Overall, the project successfully developed an efficient and user-centric music recommendation system, showcasing the potential of combining popularity-based and item similarity-based approaches.

6.4 Discussions

- The project has several noteworthy aspects and implications. Firstly, the hybrid approach combining popularity-based recommendation and item similarity-based content filtering techniques resulted in more diverse and personalized recommendations. By considering both popular trends and individual preferences, the system catered to a wider range of user preferences, ensuring a balanced and engaging music discovery experience.
- The application's recommendation engine was implemented using Python, utilizing various libraries and algorithms for data processing and machine learning. This demonstrated the versatility of Python in handling complex data and applying advanced recommendation algorithms. The integration of Python-based recommendation models with the Flutter-based mobile application showcased the interoperability of different technologies and frameworks.
- The project also faced certain challenges during the development process. The integration of multiple data sources and ensuring consistency posed initial difficulties. Additionally, fine-tuning the recommendation algorithms to strike the right balance between popularity and personalization required careful analysis and iterative refinement.
- Future enhancements for the Music Recommendation Application could include incorporating user feedback mechanisms to further enhance the recommendation accuracy and personalization. Integration with external music streaming platforms and social media sharing features could also be explored to enrich the user experience and foster engagement.

CHAPTER 7: CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

A user-friendly and individualised music recommendation system has been created as part of the Music Recommendation Application project. By combining popularity-based recommendation and item similarity-based content filtering techniques, the application provides users with an effective tool for discovering new music and getting personalized recommendations. The project's outcomes show how the system can produce pertinent recommendations and offer a seamless user experience. Throughout the project, various challenges were encountered and addressed, such as integrating diverse data sources, fine-tuning recommendation algorithms etc. The adoption of Python for the recommendation engine and Flutter for the mobile application showcased the versatility and interoperability of different technologies. In conclusion, the Music Recommender Application project has successfully developed a functional and user-centric music recommendation system. The project's results highlight its effectiveness in generating personalized recommendations and providing an intuitive user experience. With future enhancements and expansions, the project can continue to evolve, offering an even more comprehensive and engaging music discovery platform.

7.2 Future Scope

The project for the Music Recommender Application has great potential for growth and expansion. Future areas of focus include:

- Enhanced Recommendation Algorithms: The recommendation engine can be further improved by incorporating advanced machine learning techniques, such as deep learning models or hybrid recommendation approaches, to enhance recommendation accuracy and personalization.
- User Feedback: Implementing mechanisms to collect user feedback, ratings, and preferences can enhance the recommendation system's performance. User interactions and social connections can also be leveraged to provide more accurate and diverse recommendations.
- Integration with Streaming Platforms: Integrating the application with popular music streaming platforms, such as Spotify or Apple Music, can offer users a seamless music playback experience and access to a vast library of songs.
- Social Sharing and Community Features: Enabling social sharing functionalities within the application, allowing users to share their favourite songs, playlists, and recommendations with friends and followers, can enhance user engagement and create a sense of community.
- Personalized Music Events and Concerts: Expanding the application to include personalized recommendations for music events, concerts, or live performances based on user preferences and location can provide a holistic music experience.
- Continuous Improvement and Bug Fixes: Regular updates, bug fixes, and performance optimizations are essential to ensure the application remains relevant, stable, and responsive to evolving user needs and technological advancements.

REFERENCES

[1] Lohr, Steve (22 September 2009). "A $1 Million Research Bargain for Netflix, and Maybe a Model for Others". The New York Times.

[2] Phorasim P, Yu L. Movies recommendation system using collaborative filtering and k-means. International Journal of Advanced Computer Research. 2017 Mar 1;7(29):52.

[3] https://www.semanticscholar.org/paper/Movies-recommendation-system-using-collaborative-Phorasim-Yu/

[4] https://developer.hpe.com/blog/an-inside-look-at-the-components-of-a-recommendation-engine/

[5] https://www.python.org/

[6] https://flask.palletsprojects.com/en/2.3.x/

[7] https://github.com/flutter/flutter.git

[8] https://code.visualstudio.com/download

[9] https://www.javatpoint.com/waterfall-model

[10] https://walkingtree.tech/ace-software-development-know-works/

[11] https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa70000

[12] https://en.wikipedia.org/wiki/Flask_(web_framework)

[13] https://www.geeksforgeeks.org/flask-tutorial/

[14] https://www.tutorialspoint.com/flutter/flutter_introduction.htm

[15] https://www.tutorialspoint.com/flutter/flutter_introduction_to_dart_programming.htm

[16] https://en.wikipedia.org/wiki/Firebase

[17] https://code.visualstudio.com/docs

[18] https://www.postman.com/downloads/