

CRITIQUING-BASED MUSIC RECOMMENDATION CHATBOT

A PROJECT REPORT

Submitted by

JAGATHESWARAN S (422619104018)

SIVAPRIYA H (422619104041)

KEERTHIKA S (422619104701)

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



UNIVERSITY COLLEGE OF ENGINEERING, PANRUTI

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023



ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“CRITIQUING-BASED MUSIC RECOMMENDATION CHATBOT”** is the bonafide work of **“JAGATHESWARAN S (422619104018), SIVAPRIYA H (422619104041), KEERTHIKA S (422619104701)”** who carried out the project work under my supervision.

SIGNATURE

Dr. D. MURUGANANDAM,
M.Tech, Ph.D.,
Assistant Professor

HEAD OF THE DEPARTMENT
Computer Science & Engineering
University College of Engineering
Panruti – 607106

SIGNATURE

Dr. S. AYSHWARYA LAKSHMI,
B.Tech., M.Tech., Ph.D.,
Assistant Professor

SUPERVISOR
Computer Science & Engineering
University College of Engineering
Panruti – 607106

EXAMINATION HELD ON _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the work entitled “**CRITIQUING-BASED MUSIC RECOMMENDATION CHATBOT**” is submitted in partial fulfillment for the award of the degree in Bachelor of Engineering in Computer Science & Engineering. University College of Engineering, Panruti is a record of our own work carried out by us during the academic year 2022-2023. Under the supervision and guidance of **Dr.S.Ayshwarya Lakshmi B.Tech., M.Tech., Ph.D., Assistant Professor/CSE**, Department of Computer Science and Engineering, UNIVERSITY COLLEGE OF ENGINEERING PANRUTI. The extent and source of information are derived from the existing literature and have been indicated through dissertation at the appropriate places. The matter embodied in this work is original and has not been submitted for the award of any other degree or diploma, either in this or any other university.

| REGISTER NUMBER | NAME | SIGNATURE |
|-----------------|-----------------|-----------|
| 422619104018 | Jagatheswaran S | |
| 422619104041 | Sivapriya H | |
| 422619104701 | Keerthika S | |

I certify that the declaration made above by the candidate is true

SIGNATURE

**Dr. S. AYSHWARYA LAKSHMI,
B.Tech., M.Tech., Ph.D.,**

Assistant Professor

SUPERVISOR

Computer Science & Engineering
University College of Engineering
Panruti – 607106

ABSTRACT

Despite the vast amount of music available online, it can be overwhelming for users to find new songs that match their unique preferences. To address this issue, this project aims to develop a music recommendation chatbot that utilizes the Spotify API to provide personalized song suggestions. The chatbot will incorporate natural language processing techniques to understand the user's input and generate song recommendations based on their individual preferences. By interacting with the chatbot through a messaging interface, users can receive personalized recommendations and provide feedback on the suggested songs. The chatbot's use of the Spotify API will enable it to recommend songs from a broad range of genres, offering users a greater variety of music choices. This project showcases the potential of combining natural language processing with music data to create personalized music recommendations. By providing a seamless and interactive music recommendation experience, this chatbot aims to enhance the user's overall music discovery experience and offer a solution to the problem of generic music recommendations.

TABLE OF CONTENTS

| CHAPTER NO. | TITLE | PAGE NO. |
|------------------------|--|---------------------|
| | ABSTRACT | iv |
| | LIST OF TABLES | vii |
| | LIST OF FIGURES | viii |
| 1 | INTRODUCTION | 1 |
| | 1.1 Chatbot | 1 |
| | 1.2 Spotify API | 2 |
| | 1.3 Personalized Recommendations | 3 |
| | 1.4 Project Overview | 4 |
| | 1.5 Purpose | 5 |
| | 1.6 Problem Statement | 6 |
| 2 | LITERATURE SURVEY | 7 |
| | 2.1 Chat Bot Song Recommender System | 7 |
| | 2.2 Mu-Sync - A Music Recommendation Bot | 8 |
| | 2.3 Music Recommender System Using Chatbot | 9 |
| | 2.4 Emotion Based Music Playlist Recommendation System Using Interactive Chatbot | 10 |
| | 2.5 A Machine Learning Based Chatbot Song Recommender System | 11 |
| 3 | SYSTEM ANALYSIS | 13 |
| | 3.1 Existing System | 13 |
| | 3.2 Disadvantages | 14 |
| | 3.3 Proposed System | 15 |

| | | |
|----|---|----|
| | 3.4 Advantages | 16 |
| 4 | SYSTEM REQUIREMENTS | 18 |
| | 4.1 Hardware Requirements | 18 |
| | 4.2 Software Requirements | 18 |
| 5 | SOFTWARE DESCRIPTION | 19 |
| | 5.1 SOFTWARE: PYTHON | 19 |
| 6 | SYSTEM DESIGNING | 23 |
| | 6.1 System Architecture | 23 |
| | 6.2 Data Flow Diagram | 26 |
| | 6.3 Use Case Diagram | 27 |
| | 6.4 Class Diagram | 27 |
| 7 | SYSTEM IMPLEMENTATION | 28 |
| | 7.1 Spotify API | 28 |
| | 7.2 Chatbot | 29 |
| | 7.3 Recommendation | 30 |
| 8 | SYSTEM TESTING | 31 |
| | 8.1 Testing | 31 |
| | 8.2 Test Cases | 31 |
| 9 | SYSTEM STUDY | 33 |
| 10 | SCREENSHOTS | 46 |
| 11 | CONCULSION AND FUTURE ENCHANCEMENT | 54 |
| | REFERENCES | 55 |

LIST OF TABLES

| TABLE NO. | TITLE | PAGE NO. |
|----------------------|--|---------------------|
| 2.1 | Overview of Literature Survey Table | 12 |
| 3.1 | Comparison Scenario Table for Existing and Proposed System | 17 |
| 6.1 | Data Flow Symbols Table | 25 |
| 8.1 | Test-Case for App Running | 31 |
| 8.2 | Test-Case for Chatbot | 31 |
| 8.3 | Test-Case for Spotify API | 32 |

LIST OF FIGURES

| FIGURE NO. | TITLE | PAGE NO. |
|---------------|--|-------------|
| 6.1 | System Architecture | 24 |
| 6.2 | Dataflow Diagram | 26 |
| 6.3 | Use-Case Diagram | 27 |
| 6.4 | Class Diagram | 27 |
| 10.1 | Greetings | 46 |
| 10.2 | Artist or Song Query | 46 |
| 10.3 | User Responding to Query as Artist | 46 |
| 10.4 | User Responding to Query as Song | 47 |
| 10.5 | Natural Language Processing | 47 |
| 10.6 | Natural Language Processing | 47 |
| 10.7 | Artist Name | 48 |
| 10.8 | Artist Name Recognition | 48 |
| 10.9 | The Artist isn't Recognised by The User | 48 |
| 10.10 | Artist Top Tracks | 49 |
| 10.11 | User Selects A Song from The Artist's Top Tracks | 49 |
| 10.12 | Error Handling | 50 |
| 10.13 | Recommendation by Artist | 50 |
| 10.14 | Recommendation by Song | 51 |
| 10.15 | New Recommendation | 51 |
| 10.16 | Error Handling | 52 |
| 10.17 | End the Program | 52 |

CHAPTER 1

INTRODUCTION

1.1 CHATBOT

Chatbots are becoming increasingly popular as a means of automating interactions between humans and computers. A chatbot is a computer program designed to simulate conversation with human users, typically through messaging interfaces. Chatbots use Natural Language Processing (NLP) and Artificial Intelligence (AI) technologies to interpret and respond to user inputs, providing automated assistance or information.

They can be programmed to provide information, answer questions, make recommendations, or carry out simple tasks such as scheduling appointments or placing orders. The use of chatbots has grown rapidly in recent years, with many businesses adopting them as a means of providing customer service, streamlining processes, and improving engagement.

A chatbot can be designed to perform a wide range of tasks, from answering frequently asked questions to assisting with complex transactions. The key advantage of chatbots is their ability to provide quick and accurate responses to users, without the need for human intervention. This makes chatbots an ideal solution for handling high-volume tasks and improving efficiency.

There are several types of chatbots, including rule-based chatbots and AI-powered chatbots. Rule-based chatbots follow a set of pre-determined rules to provide responses, while AI-powered chatbots use machine learning algorithms to learn from user interactions and improve their responses over time.

One of the unique features of chatbots is their ability to provide 24/7 availability and immediate responses to users. This can result in improved customer satisfaction and increased efficiency for businesses, as they can handle large volumes of inquiries and requests without the need for additional staffing.

1.2 SPOTIFY API

Application Programming Interfaces (APIs) have revolutionized the way software developers build applications by allowing them to access data and functionality from other software systems. APIs provide a standardized way of interacting with external systems and can simplify the development process by allowing developers to reuse code and functionality. Spotify is a popular music streaming platform that offers access to millions of songs, podcasts, and other audio content. The Spotify API is an interface that allows developers to access Spotify's vast music library and metadata, as well as integrate Spotify features into their own applications.

With the Spotify API, developers can create applications that perform a wide range of functions, such as searching for tracks, artists, and albums, retrieving user playlists and profile information, accessing audio analysis data, and creating custom playlists.

The Spotify API uses RESTful principles and supports several programming languages, including Python, Java, JavaScript, and Ruby. Developers can interact with the API by sending HTTP requests to Spotify's servers and receiving JSON responses. The API also includes several authentication options to ensure that only authorized applications can access user data.

Developers can use the Spotify API to analyze user playlists and determine which songs are most popular, or to create custom playlists based on a user's listening history and preferences. Developers can use this data to create interactive music visualizations or to synchronize music with other multimedia content.

The API also offers real-time data on music trends and popularity, providing valuable insights for businesses and music industry professionals. With its rich data and powerful features, the Spotify API is a valuable resource for developers and businesses looking to innovate in the music streaming space.

1.3 PERSONALIZED RECOMMENDATIONS

With millions of songs available at our fingertips, it can be overwhelming to navigate the vast library of music and find the perfect track for any given moment. That's where personalized recommendations come in. By leveraging user data and advanced algorithms, music streaming services can provide tailored recommendations that match users' unique tastes and preferences. Personalized recommendations are powered by complex algorithms that analyze user data to identify patterns and preferences. The Spotify API provides developers with a wealth of data, including user listening history, playlist data, and audio analysis. Using this data, algorithms can generate personalized recommendations based on a user's listening habits, favourite genres, and musical preferences.

Personalized recommendations can enhance the listening experience by providing a seamless, tailored experience that matches their unique tastes and preferences. With personalized playlists and daily mixes, users can discover new music that they might not have otherwise found, while also enjoying their favourite tracks and artists. Another important technology behind personalized recommendations is Content-based filtering. It is a recommendation technique that uses the characteristics of items, such as songs, to recommend similar items to users. In the context of music recommendation, content-based filtering involves analyzing the audio features of songs to identify similarities and recommend similar songs to users based on their listening history.

The key idea behind content-based filtering is that if a user likes a particular song, they are likely to enjoy other songs that share similar characteristics. To implement content-based filtering for music recommendation, algorithms analyze audio features such as tempo, key, and mood to identify patterns and similarities between songs. These features are then used to generate recommendations for users based on their listening history and preferences.

1.4 PROJECT OVERVIEW

This Critiquing-Based Music Recommendation Chatbot is an innovative application that utilizes the power of natural language processing to provide personalized music recommendations to users. By integrating with the Spotify API, the chatbot is able to access a vast library of songs and artists allowing it to make highly accurate recommendations based on the user's preferences. With this project, users can discover new music, and receive personalized recommendations all through a simple and intuitive chatbot interface.

This project consists of two main parts: a set of functions that make use of the Spotify API to retrieve information about artists and songs, and a chatbot that uses these functions to provide music recommendations to users. The chatbot greets the user and asks whether they want recommendations based on an artist or a song. It then prompts the user to input the name of the artist or song they are interested in, and uses the appropriate function to retrieve information about the artist or track.

Next, the chatbot provides a list of the top tracks by the artist or tracks related to the song seed, and prompts the user to select their favourite. Based on this selection, the chatbot then calls either the recommend by artist or recommend by songs function to provide recommendations for tracks similar to the selected artist or song. The chatbot then displays the recommended tracks to the user and asks if they want to try another recommendation. If the user says yes, the process repeats. If the user says no, the chatbot bids farewell. Overall, this project demonstrates how to use the Spotify API to retrieve information about artists and tracks, and how to create a simple chatbot that makes use of this functionality to provide music recommendations based on user input.

1.5 PURPOSE

The purpose of the Critiquing-Based Music Recommendation Chatbot project is to provide a personalized music recommendation service to users based on their preferences and interests. The chatbot will use the Spotify API to retrieve information about artists and tracks, and then provide users with recommendations based on their feedback. One of the primary goals of this project is to help users discover new music that they may not have been exposed to before. With so much music available online, it can be overwhelming to find new artists and tracks to listen to. By using a chatbot that can personalize recommendations based on user feedback, users can save time and effort in finding new music that aligns with their tastes.

Another goal of the project is to provide a critiquing-based recommendation system. This means that the chatbot will not only provide recommendations based on user feedback, but will also ask for feedback on previous recommendations. By doing so, the chatbot can learn more about the user's tastes and preferences, and provide more accurate recommendations in the future. This critiquing-based approach can also help users to discover new music that they may not have initially considered based on their past listening habits.

Overall, this project aims to provide a personalized music recommendation service that can help users discover new music and expand their musical tastes. By using a critiquing-based recommendation system and an NLP system for natural language interaction, the chatbot will be able to provide accurate and relevant recommendations to users based on their preferences and feedback. The interface will make the service more accessible to a wider audience, and allow users to view their listening history and provide feedback on previous recommendations. This project has the potential to revolutionize the way people discover and listen to music, and make the process of finding new music more efficient and enjoyable.

1.6 PROBLEM STATEMENT

In recent years, music recommendation chatbots have become increasingly popular, as they provide users with an efficient way to discover new music based on their preferences. However, despite their convenience, these chatbots come with several limitations that can make it challenging for them to provide satisfactory music recommendations. One of the most significant drawbacks of music recommendation chatbots is their limited music database. With access to only a limited collection of songs, chatbots may find it challenging to cater to users with unique musical tastes. In addition, chatbots lack personalization and are often unable to access the user's listening history or individual preferences. This can lead to unsatisfactory recommendations that do not align with the user's interests. Another significant disadvantage of music recommendation chatbots is their inability to interpret emotions accurately. Technical errors can also hinder the ability of music recommendation chatbots to provide accurate and relevant music recommendations. Additionally, chatbots may rely too heavily on algorithms to suggest songs, which can lead to a lack of diversity in recommendations, leaving out songs that could be an excellent fit for the user. Other disadvantages of music recommendation chatbots include a lack of human touch, which can lead to bias in recommendations and insufficient user feedback. All of these challenges make it difficult for chatbots to provide satisfactory music recommendations to users. Nevertheless, researchers and developers are continually working to improve music recommendation chatbots by integrating more advanced algorithms and technologies, including machine learning, to enhance the chatbot's ability to provide personalized recommendations based on the user's preferences and mood. In conclusion, while music recommendation chatbots have several limitations, they remain a useful tool for discovering new music and expanding one's musical horizon.

CHAPTER 2

LITERATURE SURVEY

2.1 CHAT BOT SONG RECOMMENDER SYSTEM

AUTHOR: Prof. Suvarna Bahir, Amaan Shaikh, Bhushan Patil, Tejas Sonawane (2022)

Technology has a great impact on every part of our lives, which also includes our day-to-day habits. In the present scenario, fields like artificial intelligence and machine learning are booming. With the help of advancements in these fields, a large number of people are interacting with the system via chat bots and voice assistants. Considering the above factors, this project is aimed at implementing a machine learning-based chat bot song recommendation system that includes a chat bot to assist users and recommend songs using natural language processing. In this paper, we will discuss methodology, algorithms, and the flow of the system. The proposed system works by developing a personalised system where the user's current emotions are analysed with the help of the chat bot. The chat bot identifies the user's sentiment by having a chat conversation with the user. Based on the input provided by the user, the current motion or mood is analysed by the chat bot, and it will suggest a song to the user. The objective of our application is to identify the mood expressed by the user, and once the mood is identified, songs are played by the application. We have a methodology for building the chatbot song recommender system. To perform this, we first identified various approaches for building a chat bot known to date. Then we evaluated the algorithms considered useful in building our system in terms of their ability to work on the recommendation process of the system. This also gathered all the requirements needed for building our system and studied the overall process involved in the chat bot's working.

2.2 MU-SYNC - A MUSIC RECOMMENDATION BOT

AUTHOR: Nishtha Kapoor, Arushi Gupta, Gulshan Kumar, Dhruv Aggarwal (2021)

People interact with systems more and more through chatbots. These new modes of user interaction are aided in part by advancements in artificial intelligence and machine learning. technology. This project is aimed at implementing a music-based web application to assist the user and provide a more personal experience. This includes the chatbot, which will be trained on the dataset. This includes chatbots that combine multiple services and open-source tools to simulate a human conversation and recommend songs based on the tone of the conversation. One other thing that we can count on is that our mood will change. Believe it or not, the only thing keeping us sane during this time and the time we all left behind in lockdown was music. Hence, you can listen to music based on your mood. We have also observed that apart from music, texting is also a man's favourite task to carry on his day-to-day activities, and a man cannot live without his phone or texting. For this reason, we have created a chatbot that will analyse the user's mood and, based on that, will recommend songs. We have also used certain open source APIs for our project; these are IBM's emotions API and Last FM API. With the help of these two, our bot will first analyse the mood or emotions of the user judging by the responses given by him or her. We have used Python as our primary language because it hosts a wide array of open-source libraries that can be used by chatbots and are very practical. We have used Keras and Tensorflow for training our bot. Music recommendation system using chatbot (Mu-sync) recommends songs to the user based on their mood. With this experiment, we can conclude that if we add the music genre information in our Mu sync app, it will increase the quality of music recommendations and we can add more features into the app to make the overall performance better.

2.3 MUSIC RECOMMENDER SYSTEM USING CHATBOT

AUTHOR: Shivam Sakore, Pratik Jagdale, Mansi Borawake, Ankita Khandalkar (2021)

Communication is a daily part of life, but understanding feelings through music takes it to another level. People find music important for relaxation and feeling better. Humans can detect emotions, which is vital for successful communication. Chatbots help businesses scale their interactions with users and can be embedded in major chat apps such as Facebook Messenger, Slack, Telegram, and text messages. Chatbots improve user experience by facilitating interactions between users and services. This project aims to build a chatbot service that can casually interact with users and recommend songs based on the tone of their voice. Last.fm API will be used for song recommendations, a service similar to Spotify API. Additionally, IBM Tone Analyzer API will be used to analyze the emotional tone of conversations. API integration is important for chatbots today as they offer user-friendly features beyond data-driven conversation. Python provides a wide array of open-source libraries for building chatbots such as scikit-learn and TensorFlow. For small data sets and simpler analyses, Python's libraries are more practical. The figure shows how the project will look, with the frontend displaying the user interface and reflecting the conversation with the chatbot. The server side will mainly use two APIs to detect the emotion of the user's text: IBM Emotional API and Last.FM API for song recommendations. When a conversation is initiated, the IBM Emotional API will analyze the conversation's emotional content, and the Chatbot will reply. Using the Last.fm songs API, the app retrieves the top songs based on the perceived emotion. After listening to a song for some time, a similar song will be recommended to the user using the Last.fm API.

2.4 EMOTION BASED MUSIC PLAYLIST RECOMMENDATION SYSTEM USING INTERACTIVE CHATBOT

AUTHOR: Amrita Nair, Smriti Pillai, Ganga S Nair, Anjali T (2021)

Music is an integral part of our lives. Music is not judged by its quality but rather by its popularity, hindering quality music from underrated artists from reaching people. Therefore, due to current music trends, it's not always necessary for the user to find music they relate to. And as research shows that personalised music can have a positive impact on the human mind, it's important to have access to music based on our moods. As a result, users are exposed solely to mainstream music, and the recommendations on music streaming platforms are not very personalised. An emotion-based recommendation system permits users to listen to music based on their emotions. The proposed research work aims to develop a personalised system where the user's current emotions are analysed with the help of a chatbot. The chatbot identifies the user's sentiment by asking some general questions. Based on the input provided by the user, a score is generated for each response, which adds up to a final score. This score is then used to generate the playlist. The proposed recommendation system utilises the Spotify platform and API for playlist generation and recommendation. Human-Computer Interaction (HCI) is one of the most sought-after ways to help computers understand humans better. The most popular way of interacting is using chatbots, which are very engaging and user-friendly. Chatbots can be trained to understand the context of a user's response and respond accordingly, making the system efficient and giving the user a proper understanding. Spotify is one of the world's largest streaming platforms, with 345 million active users. It provides a web API with full access to music data. In the API, each song is associated with attributes like energy (which tells us about the energy of the song), valence (which tells us if the music is cheerful and happy), danceability, acousticness, etc. Such attributes help us better understand the overall mood of the song.

2.5 A MACHINE LEARNING BASED CHATBOT SONG RECOMMENDER SYSTEM

AUTHOR: Prof. N.S.Sharma, Amaan Shaikh, Bhushan Patil, Tejas Sonawane (2021)

Music plays an integral role in our lives, but with the rise of social media platforms like TikTok and Instagram, mainstream music dominates the charts and users are not exposed to personalized recommendations on streaming platforms. To address this, a proposed research work aims to develop a personalized system that analyzes the user's current emotion with the help of a chatbot, which identifies the user's sentiment by asking general questions. Based on the input provided by the user, the chatbot generates a playlist using APIs for playlist generation and recommendation. There are different approaches to classifying emotions, including knowledge-based, statistical, and hybrid techniques. However, there are difficulties in retrieving music information, such as querying by singing or genre classification. The most possible implementation is to produce music suggestions based on the content. A feeling descriptor is found to be useful in describing a music taxonomy for building an outstanding music recommendation system. The project aims to build an extensive chatbot service with which users can have casual conversations. Additionally, the chatbot will recommend songs based on the tone of the user's conversation. The song recommendation feature will utilize the Last.fm API, similar to the popular Spotify API. To analyze the tone or emotion of the conversation, the IBM Tone Analyzer API will be used. Collaborating with these types of APIs is critical as popular chatbots today supplement additional user-oriented features. Python is chosen to build the chatbot due to its wide array of open-source libraries for chatbots, including scikit-learn and TensorFlow. Python is also great for small data sets and more simple analyses, and its libraries are practical.

OVERVIEW OF LITERATURE SURVEY

Table 2.1 Overview of Literature Survey Table

| TITLE | AUTHOR | YEAR | MERITS | DEMERITS |
|--|--|-------------|--|---|
| Chatbot Song Recommender System | Prof. Suvarna Bahir, Amaan Shaikh, Bhushan Patil, Tejas Sonawane | 2022 | Personalized recommendation, Natural language processing, User-friendly. | Limited music selection, Inaccurate recommendations |
| Mu-Sync - A Music Recommendation Bot | Nishtha Kapoor, Arushi Gupta, Gulshan Kumar, Dhruv Aggarwal | 2021 | Personalized music recommendations, Integration with open-source tools | Limited music library, Limited accuracy, Outdated API |
| Music Recommender System Using Chatbot | Shivam Sakore, Pratik Jagdale, Mansi Borawake, Ankita Khandalkar | 2021 | Accessibility, Personalization, Automation. | Limited Interaction, Outdated API |
| Emotion Based Music Playlist Recommendation System Using Interactive Chatbot | Amrita Nair, Smriti Pillai, Ganga S Nair, Anjali T | 2021 | Personalized music recommendation, engaging and user-friendly, | The accuracy of emotion recognition, reliance on technology to determine emotions |
| A Machine Learning Based Chatbot Song Recommender System | Prof. N.S.Sharma, Amaan Shaikh, Bhushan Patil, Tejas Sonawane | 2021 | personalized recommendations to users based on their emotions, casual conversation experience. | Accuracy of emotion detection, Outdated API, costly and time-consuming |

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Music recommendation chatbots have been widely popular in recent years, offering users a way to discover new music based on their preferences. However, these chatbots come with several disadvantages that can make them less effective in providing satisfactory music recommendations. One of the most significant drawbacks of music recommendation chatbots is their limited music database. These chatbots may not have access to a vast collection of songs, making it challenging to provide suitable recommendations to users with unique musical tastes. Moreover, these chatbots often lack personalization, as they do not have access to the user's listening history or individual preferences. This can result in unsatisfactory recommendations that do not align with the user's interests. Another disadvantage of music recommendation chatbots is their inability to interpret emotions accurately. Music plays a vital role in shaping one's emotional state, and users often seek recommendations based on their mood. However, chatbots may not be able to interpret the emotional state of the user accurately, leading to inappropriate recommendations. Technical errors are another significant disadvantage of music recommendation chatbots. These chatbots can encounter errors that hinder their ability to provide accurate and relevant music recommendations. This can result in recommendations that are completely unrelated to the user's request or preferences. Additionally, music recommendation chatbots may rely too heavily on algorithms to suggest songs. This can lead to a lack of diversity in the recommendations and exclude songs that may be a good fit for the user. Furthermore, chatbots may be limited by language barriers, making it difficult to recommend music in languages other than the one they are programmed to understand.

3.2 DISADVANTAGES

1. Limited music database
2. Lack of personalization
3. Inability to interpret emotions accurately
4. Technical errors
5. Over-reliance on algorithms
6. Language barriers
7. Lack of human touch
8. Bias in recommendations
9. Insufficient user feedback

3.3 PROPOSED SYSTEM

The advent of music recommendation chatbots has revolutionized the music industry, enabling users to discover new music based on their preferences. However, to ensure effective music recommendations, developers must integrate chatbots with robust music streaming platforms. One such platform is the Spotify API, which can provide comprehensive access to millions of songs, albums, and playlists. Therefore, this project proposes a music recommendation chatbot system using the Spotify API to enhance the overall user experience. The proposed music recommendation chatbot system will utilize the Spotify API to access and analyze users' listening histories and preferences. With access to such information, the chatbot can offer more personalized recommendations, eliminating the limitations of the existing music recommendation chatbots. Moreover, the system will utilize recommend songs based on the user's preferences, musical taste, and mood. The algorithms will consider various factors such as tempo, genre, instrumentation, and lyrics to recommend songs that best suit the user's preferences. Additionally, the proposed music recommendation chatbot system will incorporate natural language processing (NLP) capabilities, enabling it to interpret users' requests accurately. This will eliminate the challenge of contextual understanding and ensure that the chatbot provides relevant recommendations based on the user's request. It will incorporate features such as the ability to share recommend songs based on user feedback. The proposed music recommendation chatbot system using the Spotify API will provide a comprehensive solution to the limitations of the existing music recommendation chatbots. The integration of the Spotify API will provide access to millions of songs, while NLP capabilities will offer personalized recommendations based on the user's preferences and context. Overall, the proposed system will enhance the user experience, providing a more satisfying and personalized music recommendation service.

3.4 ADVANTAGES

1. Comprehensive access to millions of songs, albums, and playlists through the Spotify API.
2. Personalized recommendations based on the user's listening history and preferences.
3. NLP capabilities to interpret users' requests accurately and provide relevant recommendations.
4. A human touch to the user experience through features such as the ability to share recommend songs based on user feedback.
5. Elimination of the limitations of existing music recommendation chatbots, such as limited music databases, lack of personalization, and insufficient user feedback.
6. Enhanced user experience, providing a more satisfying and personalized music recommendation service.
7. Potential for increased user engagement and community building, as users can share their music preferences and discover new music.

COMPARISON SCENARIO

Table 3.1 Comparison Scenario Table for Existing and Proposed System

| Features | Existing System | Proposed System |
|-----------------------------|--|--|
| Music Database | Limited access to songs, albums, and playlists | Comprehensive access to millions of songs, albums, and playlists through the Spotify API |
| Personalization | Limited access to user's listening history, preferences, and playlists | Personalized recommendations based on the user's listening history, preferences, and playlists |
| Emotional Interpretation | Limited accuracy in interpreting the user's emotional state | Improved accuracy of emotional interpretation, enabling the chatbot to recommend songs that align with the user's mood |
| Customization | Limited customization options | Customization options, allowing users to set preferences such as genre, tempo, and instrumentation |
| Natural Language Processing | Limited ability to interpret users' requests accurately | Natural Language Processing (NLP) capabilities, enabling the chatbot to interpret users' requests accurately and provide relevant recommendations |
| Human Touch | Limited human touch features | Incorporation of human touch features such as the ability to share playlists and recommend songs based on user feedback, creating a sense of community |
| User Feedback | Insufficient user feedback | Potential for increased user engagement and community building, as users can share their music preferences and discover new music from other users |
| Overall User Experience | Limited and unsatisfactory user experience | Enhanced user experience, providing a more satisfying and personalized music recommendation service |

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

Minimum Hardware Requirements:

- Processor: Dual-core CPU
- RAM: 2 GB
- Hard disk: 256 GB
- Compact Disk: Not required
- Keyboard: Standard keyboard
- Monitor: Standard monitor

Recommended Hardware Requirements:

- Processor: Quad-core CPU or better
- RAM: 2 GB or more
- Hard disk: 512GB or more
- Compact Disk: Not required
- Keyboard: Standard keyboard
- Monitor: High-resolution monitor (at least 1920x1080)

4.2 SOFTWARE REQUIREMENTS

- Operating system: Windows OS
- Software: Python
- Front End: command prompt

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 SOFTWARE: PYTHON

Python is a high-level, interpreted programming language that has gained widespread popularity in recent years. It was created in the late 1980s by Guido van Rossum, and its design philosophy emphasizes code readability and ease of use. Python is an open-source language, which means that its source code is freely available for anyone to use, modify, and distribute. One of the key strengths of Python is its simplicity. Its syntax is straightforward and easy to learn, making it an excellent choice for beginners. It is also a versatile language, with a wide range of applications in fields such as data science, web development, scientific computing, and artificial intelligence. Python is also known for its vast library of modules and packages, which make it easy to add functionality to your code without having to write everything from scratch. Another advantage of Python is its strong community support. There are many resources available online, including tutorials, forums, and documentation, which can help you to learn and develop your skills. Additionally, Python has a large and active user base, which means that there are always new libraries and tools being developed to help solve new problems. Python is also a highly portable language, meaning that code written in Python can be run on many different platforms, including Windows, Mac, and Linux. This makes it an ideal choice for projects that need to be run across multiple operating systems. In summary, Python is a versatile, easy-to-learn programming language that has gained widespread popularity due to its simplicity, vast library of modules, and strong community support. Its wide range of applications and portability make it an excellent choice for developers of all levels, from beginners to experts.

Feature of Python:

1. Easy to learn and read: Python has a simple syntax that is easy to learn and read, making it an excellent choice for beginners.
2. Interpreted language: Python is an interpreted language, which means that you can write and run code without needing to compile it first.
3. Dynamically typed: Python is a dynamically typed language, which means that you don't need to declare the data type of a variable before using it.
4. Large standard library: Python comes with a large standard library that includes modules for tasks such as file I/O, regular expressions, and networking.
5. Third-party libraries: There are thousands of third-party libraries available for Python that can be easily installed and used.
6. Cross-platform: Python is a cross-platform language, which means that code written on one operating system (such as Windows) can be easily run on another operating system (such as Linux or macOS).
7. Object-oriented: Python is an object-oriented language, which means that it supports concepts such as classes, objects, and inheritance.
8. High-level language: Python is a high-level language, which means that it abstracts away many of the low-level details of the computer.
9. Interactive shell: Python comes with an interactive shell, which allows you to write and run code interactively, making it an excellent choice for testing and prototyping.
10. Open-source: Python is an open-source language, which means that the source code is freely available and can be modified and distributed by anyone.

Uses of Python:

1. **Web Development:** Python is widely used for web development, with popular web frameworks such as Django and Flask built on top of it. It offers powerful tools for building web applications, managing databases, and handling web requests.
2. **Data Science:** Python has become a popular language for data science and machine learning, with libraries such as NumPy, Pandas, and TensorFlow providing powerful tools for data manipulation, analysis, and modeling.
3. **Automation:** Python is widely used for automation tasks, such as system administration, network programming, and testing. Its simple syntax and powerful libraries make it easy to automate repetitive tasks and streamline workflows.
4. **Scientific Computing:** Python is widely used in scientific computing, with libraries such as SciPy and Matplotlib providing powerful tools for numerical computation, visualization, and scientific analysis.
5. **Desktop GUI Development:** Python can be used for creating desktop GUI applications, with libraries such as PyQt and wxPython providing powerful tools for building cross-platform desktop applications.

Python Libraries

- **base64**: A Python module that provides functions to encode and decode binary data as Base64-encoded strings. Base64 encoding is often used in email protocols and in various data serialization formats.
- **requests**: A Python module that simplifies making HTTP requests in Python. It allows developers to send HTTP/1.1 requests extremely easily, and provides many useful features, such as handling cookies, authentication, and sessions.
- **json**: A Python module for encoding and decoding data in JSON format. JSON is a lightweight data interchange format that is widely used for web APIs, data storage, and configuration files.
- **re**: A Python module for working with regular expressions. Regular expressions are a powerful way to match and manipulate strings, and can be used for tasks such as data validation, text processing, and web scraping.
- **random**: A Python module that provides functions for generating random numbers and making random choices. It is often used in simulations, games, and cryptography.
- **termcolor**: A Python module that allows you to print colored text to the terminal. This can be useful for adding visual cues or highlighting important information in command-line interfaces.

CHAPTER 6

SYSTEM DESIGN

6.1 SYSTEM ARCHITECTURE

The architecture of a system reflects how the system is used and how it interacts with other systems and the outside world. It describes the interconnection of all the system's components and the data link between them. The architecture of a system reflects the way it is thought about in terms of its structure, functions, and relationships. In architecture, the term "system" usually refers to the architecture of the software itself, rather than the physical structure of the buildings or machinery. The architecture of a system reflects the way it is used, and therefore changes as the system is used.

This System performs various tasks related to music recommendations using the Spotify API. It authenticates with the Spotify API using a client ID and client secret, retrieves an access token, and makes requests to the API to search for artists and tracks, get top tracks for a given artist, and recommend music based on an artist or song seed. The code also includes a chatbot component that asks the user for their preferences on whether they want recommendations based on an artist or song, prompts the user to input an artist or song name, and recommends music based on the user's input. The system design follows a client-server model, with the client being the chatbot component and the server being the Spotify API.

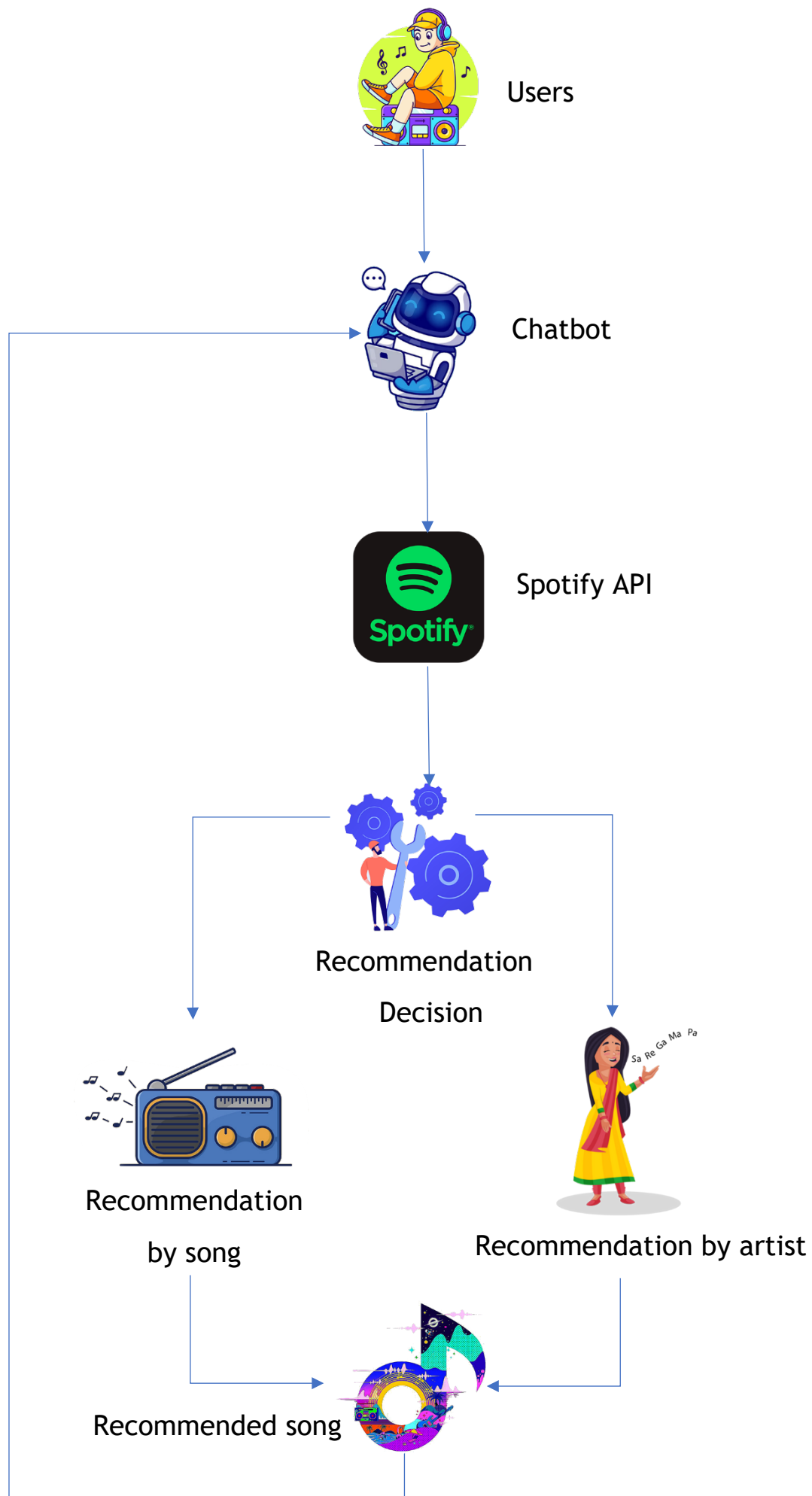

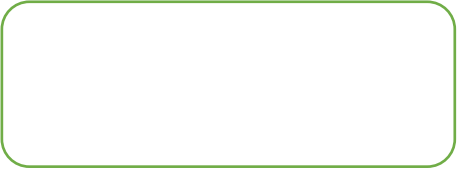
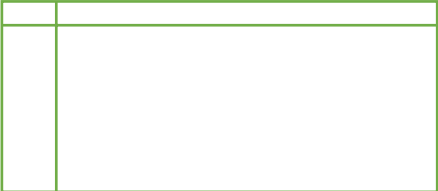



Fig 6.1 System
Architecture 24

6.2 DATA FLOW DIAGRAM

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

Table 6.1 Data flow Symbols table

| Symbol | Description |
|---|---|
|  | An entity. A source of data or a destination for data. |
|  | A process or task that is performed by the system. |
|  | A data store, a place where data is held between processes. |
|  | A data flow indicator. |

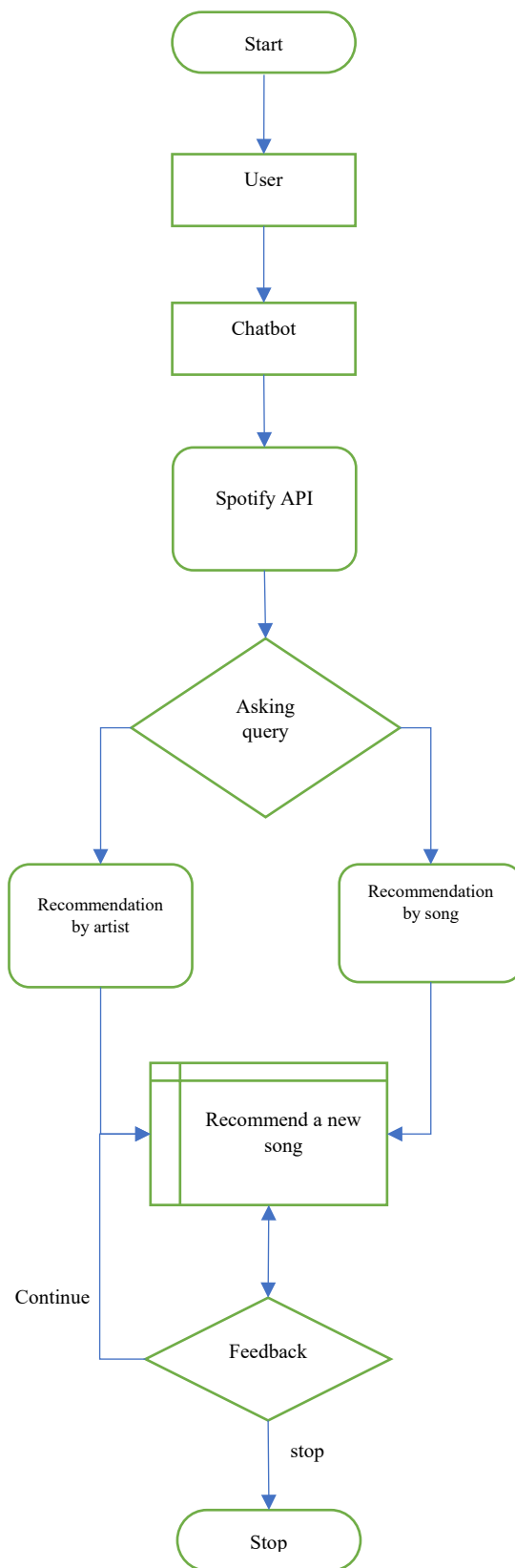


Fig 6.2 Dataflow Diagram

6.3 USE CASE DIAGRAM

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

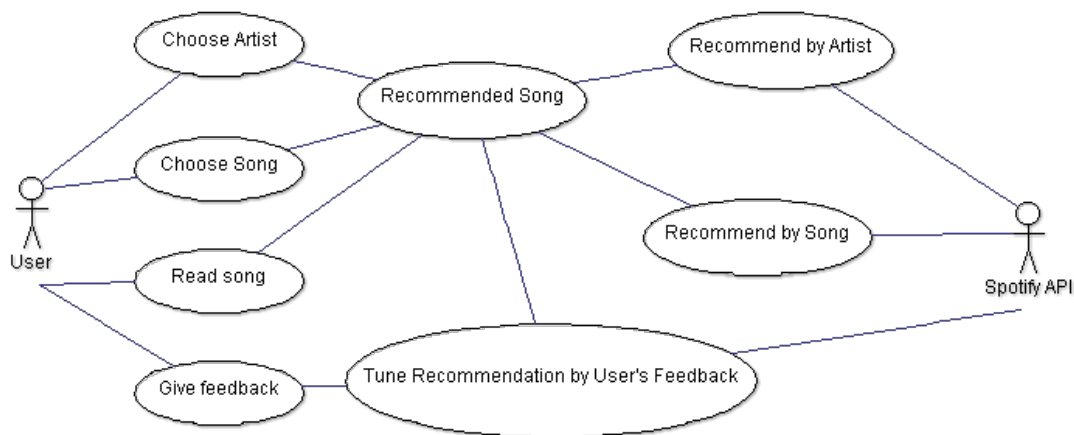


Fig 6.3 Use-Case Diagram

6.4 CLASS DIAGRAM

The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of the structure of the application, and for detailed modelling, translating the models into programming code. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

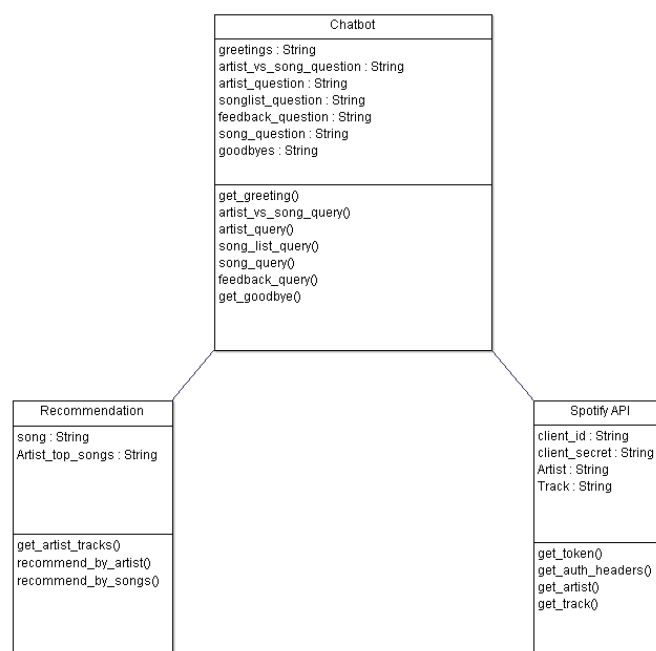


Fig 6.4 Class Diagram

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 SPOTIFY API

The first part of the code defines the client ID and secret required to authenticate and access the Spotify API. The `get_token()` function is used to retrieve an access token from the Spotify API, which is required to make authorized requests to the API. This function encodes the client ID and secret as base64 and sends a POST request to the Spotify API's token endpoint to retrieve the access token. The `get_auth_headers()` function is used to create the authorization headers required for making requests to the Spotify API with the access token retrieved by `get_token()`. The `get_artist()` function is used to search for an artist on the Spotify API using the artist's name. It sends a GET request to the Spotify API's search endpoint with the query parameters specifying the artist name and type as "artist". The function returns the first artist result from the response, or None if no results are found. The `get_artist_tracks()` function is used to retrieve the top tracks of an artist on the Spotify API. It sends a GET request to the artist's tracks endpoint with the artist ID and country as parameters. The function returns a list of the top tracks of the artist. The `recommend_by_artist()` function is used to retrieve music recommendations based on an artist and a seed song. It sends a GET request to the Spotify API's recommendations endpoint with the artist ID and seed song as parameters. The `recommend_by_songs()` function is used to retrieve music recommendations based on a seed song. It sends a GET request to the Spotify API's recommendations endpoint with the seed song as a parameter. The function returns a list of recommended tracks. The `get_track()` function is used to search for a specific track on the Spotify API using the track name. It sends a GET request to the Spotify API's search endpoint with the query parameters specifying the track name and type as "track". The function returns the first track result from the response, or None if no results are found.

7.2 CHATBOT

The second part of the code defines the chatbot's behaviour. It starts by randomly selecting a greeting message from a list of predefined messages. The chatbot then asks the user if they want music recommendations based on an artist or song they like. It randomly selects a question from a list of predefined questions to ask the user. If the user selects an artist, the chatbot asks the user to input the name of the artist they want to search for. It then calls the `get_artist()` function to search for the artist on the Spotify API. If the artist is found, the chatbot displays the artist's name and top tracks and asks the user to select a seed song for music recommendations. If the user selects a seed song, the chatbot asks the user to select the song they enjoyed the most from the list of top tracks displayed earlier. It then calls the `recommend_by_artist()` function to retrieve music recommendations based on the selected artist and seed song. If the user selects a song, the chatbot asks the user to input the name of the song they want to search for. It then calls the `get_track()` function to search for the song on the Spotify API. If the song is found, the chatbot displays the song's name and artist and asks the user to select the song as a seed song for music recommendations. If the user selects a seed song, the chatbot calls the `recommend_by_songs()` function to retrieve music recommendations based on the selected seed song. If the user wants to hear more recommendations, the chatbot asks if they want recommendations based on the same artist or song or if they want to search for a new artist or song. If the user wants to exit, the chatbot displays a goodbye message and ends the conversation. Throughout the conversation, the chatbot uses various error handling techniques to ensure that the program does not crash or behave unexpectedly. The chatbot also uses natural language processing techniques to understand and interpret the user's input. It uses regular expressions and string manipulation to extract relevant information from the user's input, such as the name of the artist or song they want to search for.

7.3 RECOMMENDATION

Spotify is a music streaming service that provides users with access to millions of songs, albums, and playlists from all over the world. One of the features that sets Spotify apart from other music streaming services is its "Get Recommendations" feature, which uses machine learning algorithms to suggest new songs, artists, and playlists based on a user's listening habits. The first step in generating recommendations is to create a unique "taste profile" for each user. This is done by analyzing the user's listening history and identifying patterns and preferences. The system then matches this taste profile to other users who have similar profiles, and looks for music that these users have listened to that the current user has not yet discovered. The model considers various factors, such as the popularity of the song, the similarity to other songs the user has liked, and the diversity of the recommendations. The system then generates a list of recommended songs and presents them to the user in the "Get Recommendations" section. As the user listens to more music and interacts with the recommendations, the system updates their taste profile and refines its recommendations accordingly. This creates a feedback loop where the more the user listens and interacts with the recommendations, the better the recommendations become. To make recommendations, Spotify uses a combination of collaborative filtering and content-based filtering. Collaborative filtering involves comparing a user's listening habits to those of other users with similar tastes, and then suggesting music that those users have enjoyed. By using a combination of collaborative filtering, content-based filtering, Spotify is able to provide highly personalized recommendations that are tailored to each user's individual tastes and interests. This makes it easier for users to discover new music and artists, and to build their own personalized playlists and music collections.

CHAPTER 8

SYSTEM TESTING

8.1 TESTING

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

8.2 Test Cases

Table 8.1 Test-Case for App Running

| ID | Test Cases | Input Data | Steps of execution | Expected Output | Result |
|----|--|--|---------------------|---------------------------------------|--------|
| 1 | User is able to access the application | Running code in terminal or command prompt | Run the python file | Greets user | Pass |
| 2 | All the UI should be working fine and display prompts properly | Running code in terminal or command prompt | Run the python file | Text should display in various colour | Pass |

Table 8.2 Test-Case for Chatbot

| ID | Test Cases | Input Data | Steps of execution | Expected Output | Result |
|----|--|-----------------|----------------------|---|--------|
| 1 | User is able to see greetings from chatbot | - | Run the python file | Chatbot should greet user | Pass |
| 2 | User is able to see Decision question about artist or song | - | Run the python file | Chatbot should prompt query to user | Pass |
| 3 | User is able to type artist name | Any Artist name | choosing Artist | Chatbot should prompt query to user | Pass |
| 4 | User is able to see artist name and confirm prompt | Yes/No | choosing Artist name | Chatbot should prompt artist name and query to user | Pass |

| ID | Test Cases | Input Data | Steps of execution | Expected Output | Result |
|----|--|---------------------------------|--------------------|--|--------|
| 5 | User is able to see Artist top tracks from chatbot | - | Choosing yes | Chatbot should show the artist top tracks | Pass |
| 6 | User is able to type number to choose song | Selecting numbers between 1 -10 | Automatic | Chatbot should prompt query to user | Pass |
| 7 | User is able to see select song and its preview | - | Automatic | Chatbot should Display song and it preview to user | Pass |
| 8 | User should see recommended song | - | Automatic | Chatbot should Display song and it preview to user | Pass |
| 9 | User should be able to answer the feedback | Yes/No | Automatic | Chatbot should prompt query to user | Pass |
| 10 | Chatbot must exit if user said stop | Stop | Automatic | Program should exit | Pass |

Table 8.3 Test-Case for Spotify API

| ID | Test Cases | Input Data | Steps of execution | Expected Output | Result |
|----|-------------------------|-----------------------------|---------------------------|--------------------------------|--------|
| 1 | Get token from Spotify | Client id and client secret | Requesting to Spotify API | Get token in json format | Pass |
| 2 | Get valid artist name | Artist name | Requesting to Spotify API | Get artist name in json format | Pass |
| 3 | Get artist's top track | Artist ID | Requesting to Spotify API | Get token in json format | Pass |
| 4 | Get track from Spotify | Song ID | Requesting to Spotify API | Get Song name in json format | Pass |
| 5 | Get recommend by artist | Artist ID, Song ID | Requesting to Spotify API | Get token in json format | Pass |
| 6 | Get recommend by song | Song ID | Requesting to Spotify API | Get token in json format | Pass |

CHAPTER 9

SYSTEM STUDY

SOURCE CODE

```
#Spotify API
import base64
from requests import post, get
import json

client_id = "3c807945a2cc4df0bfc4d2143a091a27"
client_secret = "e558e47f4912400db3038a1c67e81a80"

def get_token():
    auth_string = client_id + ":" + client_secret
    auth_bytes = auth_string.encode("utf-8")
    auth_base64 = str(base64.b64encode(auth_bytes), "utf-8")

    url = "https://accounts.spotify.com/api/token"
    headers = {
        "Authorization": "Basic " + auth_base64,
        "Content_type": "application/x-www-form-urlencoded"
    }
    data = {"grant_type": "client_credentials"}
    result = post(url, headers=headers, data=data)
    result_json = json.loads(result.content)
    token = result_json["access_token"]
    return token
```

```

def get_auth_headers(token):
    headers = {"Authorization": "Bearer " + token}
    return headers

def get_artist(token, artist_name):
    url = "https://api.spotify.com/v1/search"
    headers = get_auth_headers(token)
    query = f"?q={artist_name}&type=artist&limit=1"
    query_url = url + query
    result = get(query_url, headers=headers)
    result_json = json.loads(result.content)["artists"]["items"]
    if len(result_json) == 0:
        return None
    return result_json[0]

def get_artist_tracks(token, artist_id):
    url = f"https://api.spotify.com/v1/artists/{artist_id}/top-tracks?country=IN"
    headers = get_auth_headers(token)
    result = get(url, headers=headers)
    result_json = json.loads(result.content)["tracks"]
    return result_json

```

```

def recommend_by_artist(token,artist_seed, song_seed):
    url = "https://api.spotify.com/v1/recommendations"
    headers = get_auth_headers(token)
    params = {
        "seed_artist": artist_seed,
        "seed_tracks": song_seed,
        "market": "IN",
        "limit": 5
    }
    result = get(url, headers=headers, params=params)
    result_json = json.loads(result.content)["tracks"]
    return result_json

```

```

def recommend_by_songs(token, song_seed):
    url = "https://api.spotify.com/v1/recommendations"
    headers = get_auth_headers(token)
    params = {
        "seed_tracks": song_seed,
        "market": "IN",
        "limit": 5
    }
    result = get(url, headers=headers, params=params)
    result_json = json.loads(result.content)["tracks"]
    return result_json

```

```

def get_track(token,track_name):
    url = "https://api.spotify.com/v1/search"
    headers = get_auth_headers(token)
    query = f"?q={track_name}&type=track&market=IN&limit=1"
    query_url = url + query
    response = get(query_url, headers=headers)
    response_json = json.loads(response.content)['tracks']['items']
    if len(response_json) == 0:
        return None
    return response_json[0]

def empty_URL(preview_url):
    if preview_url is not None:
        return cprint(preview_url, "blue")
    else:
        return cprint("There is no music preview available for this song.", "blue")

#####CHATBOT#####

import re
import random
from termcolor import colored, cprint

greetings = ['Hello!',
             'Hi there!',
             'Hey!',
             'Greetings!',

```

'Nice to see you!']

artist_vs_song_question = ["Are you looking for music recommendations based on a certain artist or song?",

"Would you like me to suggest some music based on an artist or song you like?",

"Do you want me to recommend music based on a particular artist or song?",

"Are you interested in discovering new music based on an artist or song you enjoy?",

"Can I provide you with music recommendations based on a specific artist or song?",

"Do you want me to suggest some songs or artists based on your preferences?",

"Are you open to getting music suggestions based on an artist or song you like?",

"Would you like me to curate a playlist based on an artist or song that you enjoy?",

"Can I offer you some music recommendations based on a favorite artist or song?",

"Are you interested in exploring new music based on an artist or song you're familiar with?"]

artist_question = ["Please provide the name of the artist:",

"What is the name of the artist you would like to search?",

"Type in the artist's name:",

"Enter the name of the artist you're interested in:",

"What is the name of the artist you want to look up?",

"Please input the artist's name:",

"Who is the artist you'd like to find?",

"What's the name of the artist you're searching for?",

"Type the name of the artist you want to find:",

"Please provide the artist's name to begin the search:"]

songlist_question = ["Which song did you enjoy the most? Please enter it below (1-10):",

 "Input the title of the song you liked the most (1-10):",

 "Please type in the name of the song you enjoyed the most (1-10):",

 "Enter the title of the song that you liked the most (1-10):",

 "What is the name of the song that you enjoyed the most (1-10)?",

 "Please provide the title of the song you liked the most (1-10):",

 "Which song stood out to you the most? Please enter it below (1-10):",

 "Type the title of the song you enjoyed the most (1-10):",

 "Can you tell me the name of the song that you liked the most (1-10)?",

 "Which song was your favorite? Please enter the title (1-10):"]

feedback_question = ["Would you say you enjoyed the song? (Yes/No/Stop)",

 "Did the song appeal to you? (Yes/No/Stop)",

 "Are you liking the song? (Yes/No/Stop)",

 "Do you find the song enjoyable? (Yes/No/Stop)",

 "Would you like to hear more music like this? (Yes/No/Stop)",

 "Was the song to your liking? (Yes/No/Stop)",

 "Do you want to continue listening to the song? (Yes/No/Stop)",

 "Is the song to your taste? (Yes/No/Stop)",

 "Did you enjoy listening to the song? (Yes/No/Stop)",

 "Are you interested in listening to more songs like this? (Yes/No/Stop)"]

song_question = ["What type of song do you feel like listening to based on your current mood?",

 "Which genre of music suits your current mood?",

"What kind of music would you like to listen to right now, based on your mood?",

"Based on how you're feeling right now, what type of song would you like to hear?",

"Can you describe your mood so I can suggest a suitable song?",

"Tell me about your mood and I'll recommend a song that fits.",

"Depending on your mood, what genre of music would you like to listen to?",

"Which type of music would you prefer to listen to right now, given your mood?",

"Let me know your current mood, and I'll suggest a song that fits the bill.",

"What kind of music would you like to hear that would complement your current mood?"]

goodbyes = ['Goodbye!',

'See you later!',

'Bye!',

'Nice chatting with you!',

'Take care!']

```
def get_greeting():
```

```
    return random.choice(greetings)
```

```
def artist_vs_song_query():
```

```
    return random.choice(artist_vs_song_question)
```

```
def artist_query():
```

```
    return random.choice(artist_question)
```

```

def song_list_query():
    return random.choice(songlist_question)

def song_query():
    return random.choice(song_question)

def feedback_query():
    return random.choice(feedback_question)

def get_goodbye():
    return random.choice(goodbyes)

cprint("\n" + get_greeting() + "\n", "yellow")

stop_flag = True

decision = input(colored(artist_vs_song_query() + "\n", "light_green")).strip()

artist_pattern = re.compile(r'^s*(art(?:i|is|ist|ists)?\w*)s*[\w\s]*s*$',
re.IGNORECASE)

song_pattern = re.compile(r'^s*(son(?:g|gs|gg)?)\s*[\w\s]*s*$',
re.IGNORECASE)

while not (artist_pattern.match(decision) or song_pattern.match(decision)):
    print("Invalid input. Please enter either 'artist' or 'song'." + "\n")
    decision = input(colored(artist_vs_song_query() + "\n", "light_green")).strip()

token = get_token()

if artist_pattern.match(decision):

```



```

while True:
    artist_name = input(colored("\n" + artist_query() + "\n", "light_green"))
    search_artist = get_artist(token, artist_name)
    if search_artist:
        artist = search_artist["name"]
        cprint("\n" + artist + "\n", "cyan")
        confirm_artist = input(colored("Is this the artist you are looking for?
(Yes/No)\n", "light_green")).strip()
        if confirm_artist.lower() == "yes":
            artist_id = search_artist["id"]
            artist_tracks = get_artist_tracks(token, artist_id)
            cprint("\n" + f"These are all the {artist}'s top tracks:" + "\n", "cyan")
            for idx, song in enumerate(artist_tracks):
                cprint(f'{idx + 1}. {song['name']}', "yellow")
            break
        elif confirm_artist.lower() == "no":
            cprint("\nPlease provide the appropriate artist name.", "red")
    else:
        cprint("\n" + "Artist not found. Please try again.", "red")

```

```

while True:
    try:
        asking_song = int(input(colored("\n" + song_list_query(),
"light_green")))
        if 1 <= asking_song <= 10:
            break
    else:
        cprint("Please enter a number between 1 and 10.", "red")

```

```

except ValueError:

    cprint("Invalid input. Please enter a number between 1 and 10.", "red")
select_song = artist_tracks[asking_song - 1]
song_id = select_song["id"]
cprint("\nHere is the song that you chosen", "cyan")
cprint("\n" + select_song['name'], "yellow")
cprint(select_song['preview_url'], "blue")
recommended_list = recommend_by_artist(token, artist_id, song_id)
cprint("\nHere's a diverse song that you might enjoy and that showcases a
range of musical styles within this category.", "light_green")
recommend_song_name = recommended_list[0]['name']
recommend_song_url = recommended_list[0]['preview_url']
latest_recommend_song_id = recommended_list[0]['id']
cprint("\n" + recommend_song_name, "yellow")
cprint(recommend_song_url, "blue")

while True:

    feedback = input(colored("\n" + feedback_query() + "\n",
"light_green")).strip()

    if feedback.lower() == 'stop':

        cprint("\n" + get_goodbye(), "light_green")
        break

    elif feedback.lower() == 'yes':

        seed_song = recommend_by_artist(token, artist_id,
latest_recommend_song_id);

```

```

song_name = seed_song[0]['name']
song_preview = seed_song[0]['preview_url']
cprint("\n" + song_name, "yellow")
cprint(song_preview, "blue")

elif feedback.lower() == 'no':
    next_song = 0
    next_song += 1
    recommended_list_song = recommend_by_artist(token, artist_id,
song_id)[next_song]
    song_name = recommended_list_song['name']
    song_preview = recommended_list_song['preview_url']
    cprint("\n" + song_name, "yellow")
    cprint(song_preview, "blue")

else:
    cprint("\n" + "Sorry, I didn't understand that. Please choose 'yes', 'no', or
'stop'." + "\n", "red")

elif song_pattern.match(decision):

while stop_flag :
    user_song = input(colored("\n" + song_query() + "\n", "light_green"))
    song_recommend = get_track(token, user_song)

    if song_recommend is None:
        cprint("\nSorry, I couldn't find a song matching that name. Please try
again.\n", "red")

```

```

        continue

    else:

        song_id = song_recommend["id"]
        cprint("\nHere is the song that you chosen", "cyan")
        cprint("\n" + song_recommend['name'], "yellow")
        cprint(song_recommend['preview_url'], "blue")
        recommended_list = recommend_by_songs(token, song_id)

        cprint("\nHere's a diverse song that you might enjoy and that showcases
a range of musical styles within this category.", "light_green")

        latest_recommend_song_id = recommended_list[0]['id']
        cprint("\n" + recommended_list[0]['name'], "yellow")
        cprint(recommended_list[0]['preview_url'], "blue")

    while True:

        feedback = input(colored("\n" + feedback_query() + "\n",
"light_green")).strip()

        if feedback.lower() == 'stop':
            stop_flag = False
            cprint("\n" + get_goodbye(), "light_green")
            break

        elif feedback.lower() == 'yes':
            seed_song = recommend_by_songs(token,
latest_recommend_song_id);
            song_name = seed_song[0]['name']
            song_preview = seed_song[0]['preview_url']

```

```

        cprint("\n" + song_name,"yellow")
        cprint(song_preview, "blue")

    elif feedback.lower() == 'no':
        next_song = 0
        next_song += 1
        recommended_list_song = recommend_by_songs(token,
song_id)[next_song]
        song_name = recommended_list_song['name']
        song_preview = recommended_list_song['preview_url']
        cprint("\n" + song_name,"yellow")
        cprint(song_preview, "blue")

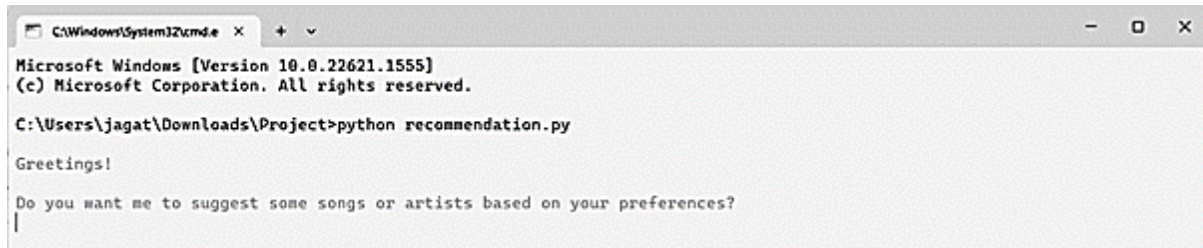
    else:
        cprint("Sorry, I didn't understand that. Please choose 'yes', 'no', or
'stop'.", "red")

```

CHAPTER 10

SCREENSHOTS

The chatbot greets the user with a random greeting message from the list of pre-defined greetings.



```
C:\Windows\System32\cmd.exe x + v
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

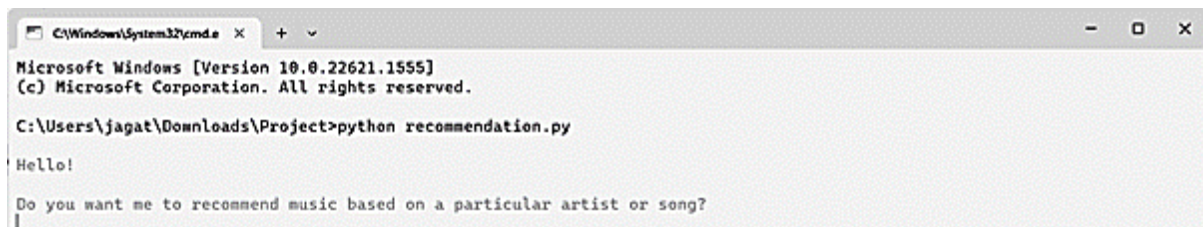
C:\Users\jagat\Downloads\Project>python recommendation.py

Greetings!

Do you want me to suggest some songs or artists based on your preferences?
|
```

Fig 10.1 Greetings

The chatbot asks the user whether they are looking for music recommendations based on a certain artist or song. It randomly selects one of the questions from the list.



```
C:\Windows\System32\cmd.exe x + v
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

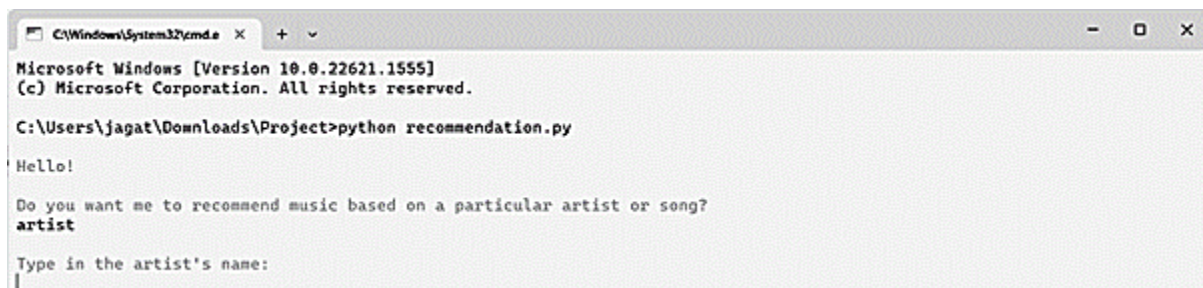
C:\Users\jagat\Downloads\Project>python recommendation.py

Hello!

Do you want me to recommend music based on a particular artist or song?
|
```

Fig 10.2 Artist or Song query

The user types in their response, which should be either "artist" or "song".



```
C:\Windows\System32\cmd.exe x + v
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jagat\Downloads\Project>python recommendation.py

Hello!

Do you want me to recommend music based on a particular artist or song?
artist

Type in the artist's name:
|
```

Fig 10.3 User Responding to Query as Artist

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jagat\Downloads\Project>python recommendation.py

Hey!

Are you interested in discovering new music based on an artist or song you enjoy?
song

Let me know your current mood, and I'll suggest a song that fits the bill.
```

Fig 10.4 User Responding to Query as Song

If the user's response is "artist", the chatbot asks the user to provide the name of the artist. It randomly selects one of the questions from the list of artist question. Chatbots can recognise even when users give incorrect input.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jagat\Downloads\Project>python recommendation.py

Hey!

Can I offer you some music recommendations based on a favorite artist or song?
arti

Please provide the name of the artist:
```

Fig 10.5 Natural Language Processing

Remove any leading or trailing whitespace from the user input. Remove any non-alphanumeric characters from the user input using regular expression.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

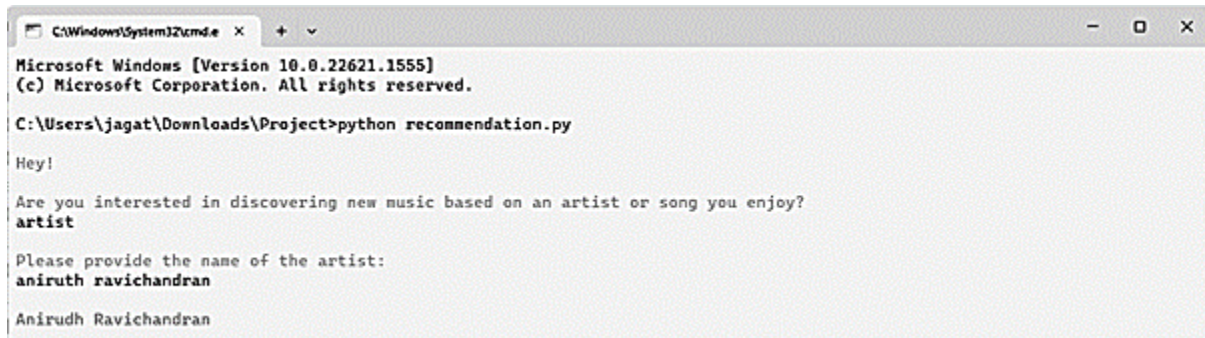
C:\Users\jagat\Downloads\Project>python recommendation.py

Nice to see you!

Are you open to getting music suggestions based on an artist or song you like?
artistt&
```

Fig 10.6 Natural Language Processing

The user types in the name of the artist.



```
C:\Windows\System32\cmd.exe x + v
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jagat\Downloads\Project>python recommendation.py

Hey!

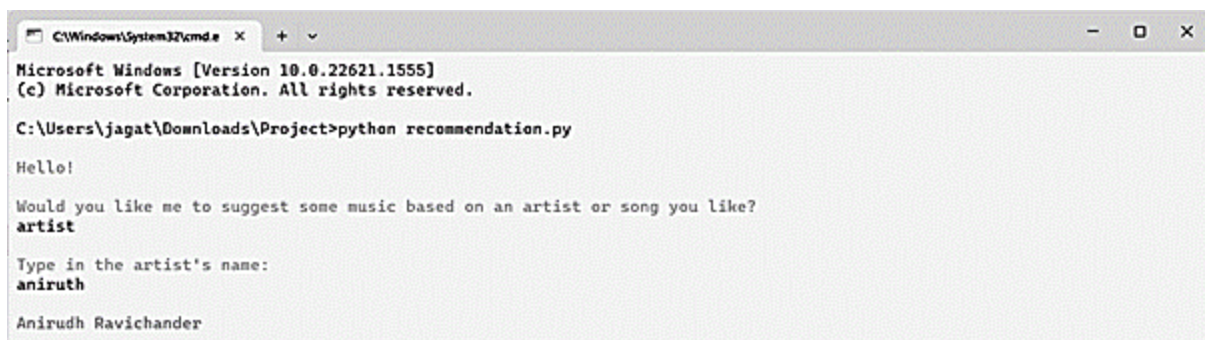
Are you interested in discovering new music based on an artist or song you enjoy?
artist

Please provide the name of the artist:
aniruth ravichandran

Anirudh Ravichandran
```

Fig 10.7 Artist name

Even if the user provides incorrect information about the artist's name, the chatbot will understand it.



```
C:\Windows\System32\cmd.exe x + v
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jagat\Downloads\Project>python recommendation.py

Hello!

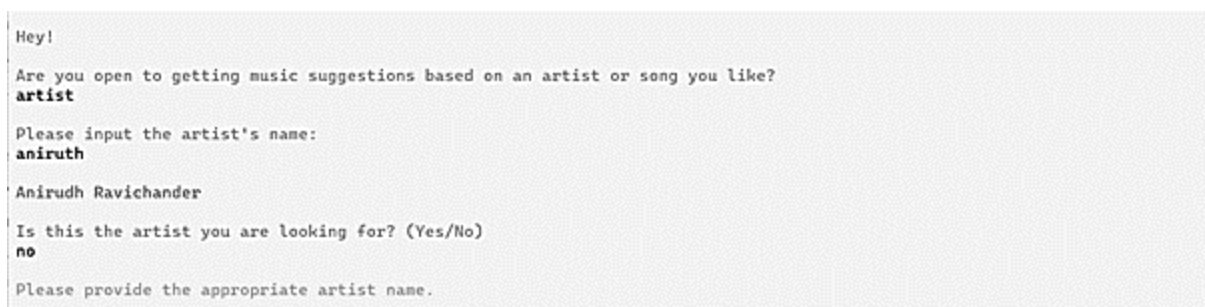
Would you like me to suggest some music based on an artist or song you like?
artist

Type in the artist's name:
aniruth

Anirudh Ravichander
```

Fig 10.8 Artist name recognition

If the name of the artist is not found in the Spotify database, the chatbot informs the user that it couldn't find the artist and asks them to try again.



```
Hey!

Are you open to getting music suggestions based on an artist or song you like?
artist

Please input the artist's name:
aniruth

Anirudh Ravichander

Is this the artist you are looking for? (Yes/No)
no

Please provide the appropriate artist name.
```

Fig 10.9 The artist isn't recognised by the user

If the name of the artist is found in the Spotify database, the chatbot returns the top tracks for the artist and asks the user to select one of them.

```
C:\Windows\System32\cmd.exe x + v
C:\Users\jagat\Downloads\Project>python recommendation.py
Hello!
Would you like me to suggest some music based on an artist or song you like?
artist
Type in the artist's name:
aniruth
Anirudh Ravichander
Is this the artist you are looking for? (Yes/No)
yes
These are all the Anirudh Ravichander's top tracks:
1. Jimikki Ponnu
2. Arabic Kuthu - Halamithi Habibo (From "Beast")
3. Thenmozhi (From "Thiruchitrabalam")
4. Megham Karukatha
5. Gundellonaa
6. Idhazhin Oram - The Innocence of Love
7. Dippam Dappam (From "Kaathuvaakula Rendu Kaadhal")
8. Mainaru Vetti Katti
9. Mayakkama Kalakkama (From "Thiruchitrabalam")
10. Nee Paartha Vizhigal - The Touch of Love
Please provide the title of the song you liked the most (1-10):|
```

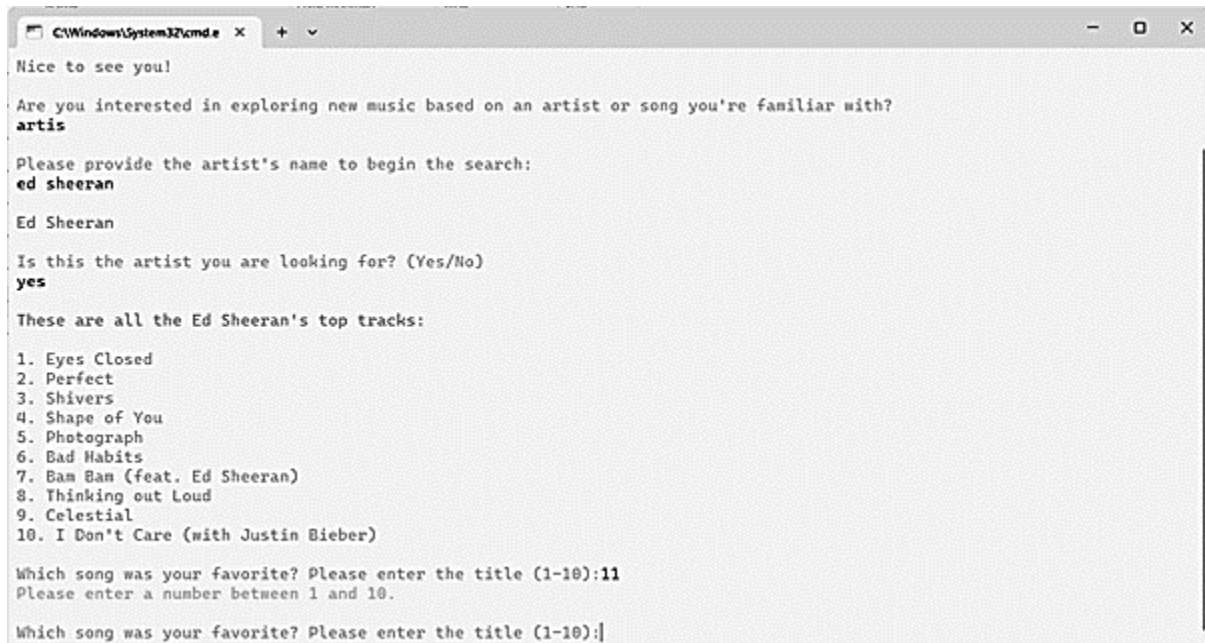
Fig 10.10 Artist top tracks

The user types in the number of the track they want to select.

```
C:\Windows\System32\cmd.exe x + v
Is this the artist you are looking for? (Yes/No)
yes
These are all the Ed Sheeran's top tracks:
1. Eyes Closed
2. Perfect
3. Shivers
4. Shape of You
5. Photograph
6. Bad Habits
7. Bam Bam (feat. Ed Sheeran)
8. Thinking out Loud
9. Celestial
10. I Don't Care (with Justin Bieber)
Which song was your favorite? Please enter the title (1-10):2
Here is the song that you chosen
Perfect
https://p.scdn.co/mp3-preview/4e30857a3c7da3f8891483643e310bb233afadd27?cid=3c807945a2cc4df0bfc4d2143a091a27
Here's a diverse song that you might enjoy and that showcases a range of musical styles within this category.
Cheerleader - Felix Jaehn Remix Radio Edit
https://p.scdn.co/mp3-preview/3b8b1f0961ad4b68d3e08de52489ee509062342d?cid=3c807945a2cc4df0bfc4d2143a091a27
Did the song appeal to you? (Yes/No/Stop)
yes
```

Fig 10.11 The user selects a song from the artist's top tracks

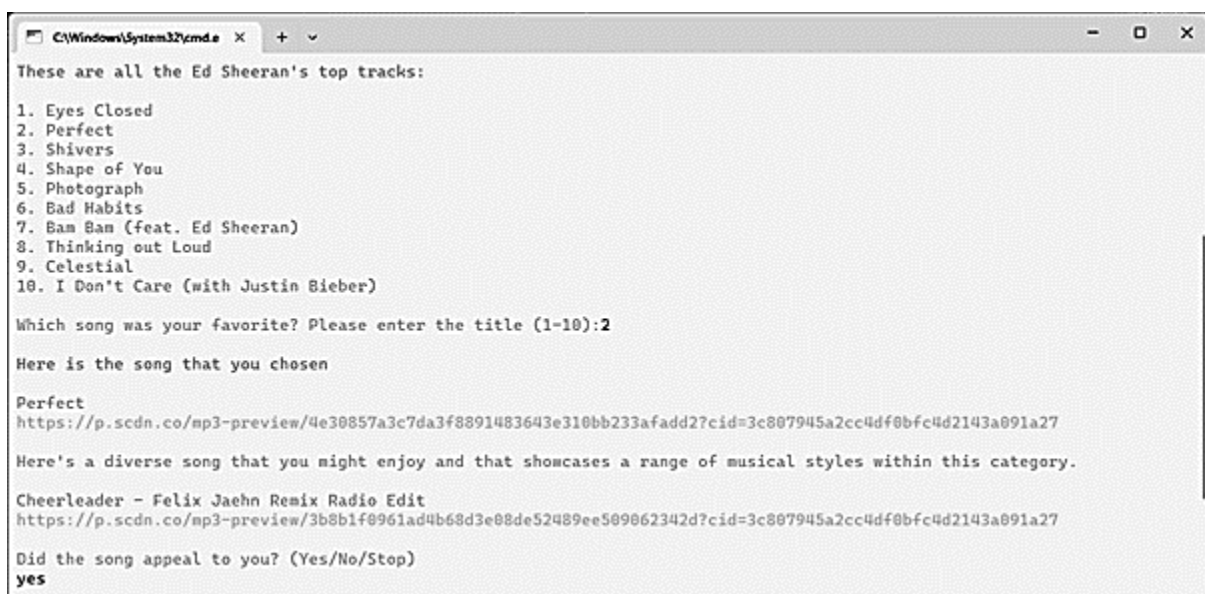
If the user enters an invalid input (i.e., a number that is not between 1 and 10), the chatbot informs the user that their input is invalid and asks them to try again.



```
C:\Windows\System32\cmd.exe x + v
Nice to see you!
Are you interested in exploring new music based on an artist or song you're familiar with?
artis
Please provide the artist's name to begin the search:
ed sheeran
Ed Sheeran
Is this the artist you are looking for? (Yes/No)
yes
These are all the Ed Sheeran's top tracks:
1. Eyes Closed
2. Perfect
3. Shivers
4. Shape of You
5. Photograph
6. Bad Habits
7. Bam Bam (feat. Ed Sheeran)
8. Thinking out Loud
9. Celestial
10. I Don't Care (with Justin Bieber)
Which song was your favorite? Please enter the title (1-10):11
Please enter a number between 1 and 10.
Which song was your favorite? Please enter the title (1-10):|
```

Fig 10.12 Error handling

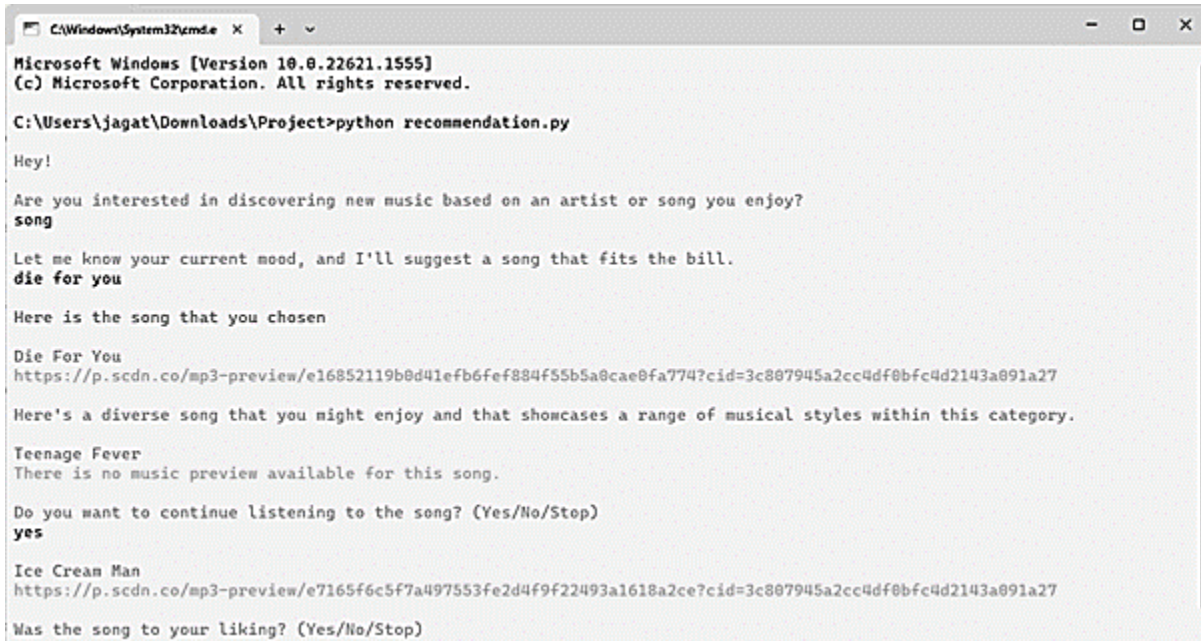
If the user enters a valid input, the chatbot recommends similar tracks based on the selected artist and track. It returns recommended track and asks the user if they want to hear another recommendation.



```
C:\Windows\System32\cmd.exe x + v
These are all the Ed Sheeran's top tracks:
1. Eyes Closed
2. Perfect
3. Shivers
4. Shape of You
5. Photograph
6. Bad Habits
7. Bam Bam (feat. Ed Sheeran)
8. Thinking out Loud
9. Celestial
10. I Don't Care (with Justin Bieber)
Which song was your favorite? Please enter the title (1-10):2
Here is the song that you chosen
Perfect
https://p.scdn.co/mp3-preview/4e30857a3c7da3f8891483643e310bb233afadd2?cid=3c807945a2cc4df0bfc4d2143a091a27
Here's a diverse song that you might enjoy and that showcases a range of musical styles within this category.
Cheerleader - Felix Jaehn Remix Radio Edit
https://p.scdn.co/mp3-preview/3b8b1f0961ad4b68d3e08de52489ee509062342d?cid=3c807945a2cc4df0bfc4d2143a091a27
Did the song appeal to you? (Yes/No/Stop)
yes
```

Fig 10.13 Recommendation by artist

If the user enters a valid input, the chatbot recommends similar tracks based on the selected song. It returns recommended track and asks the user if they want to hear another recommendation.



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jagat\Downloads\Project>python recommendation.py

Hey!

Are you interested in discovering new music based on an artist or song you enjoy?
song
Let me know your current mood, and I'll suggest a song that fits the bill.
die for you

Here is the song that you chosen

Die For You
https://p.scdn.co/mp3-preview/e16852119b0d41efb6fef884f55b5a0cae0fa774?cid=3c807945a2cc4df0bfc4d2143a091a27

Here's a diverse song that you might enjoy and that showcases a range of musical styles within this category.

Teenage Fever
There is no music preview available for this song.

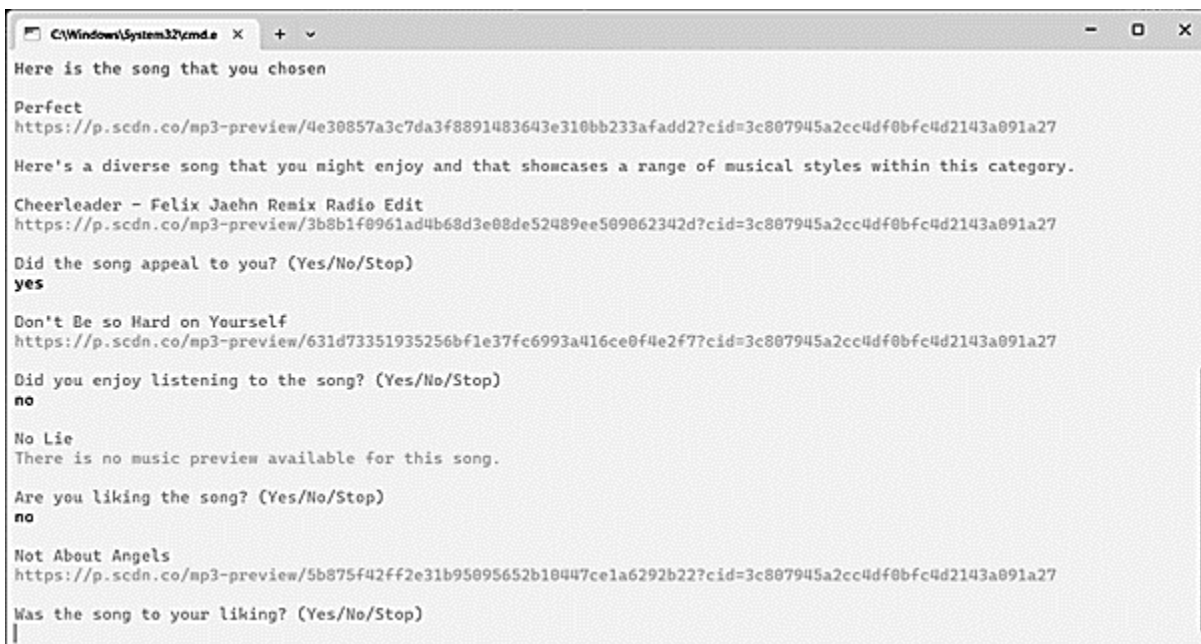
Do you want to continue listening to the song? (Yes/No/Stop)
yes

Ice Cream Man
https://p.scdn.co/mp3-preview/e7165f6c5f7a497553fe2d4f9f22493a1618a2ce?cid=3c807945a2cc4df0bfc4d2143a091a27

Was the song to your liking? (Yes/No/Stop)
```

Fig 10.14 Recommendation by song

If the user's response is "yes", the chatbot repeats steps with a new recommendation.



```
C:\Windows\System32\cmd.exe
Here is the song that you chosen

Perfect
https://p.scdn.co/mp3-preview/4e30857a3c7da3f8891483643e310bb233afadd2?cid=3c807945a2cc4df0bfc4d2143a091a27

Here's a diverse song that you might enjoy and that showcases a range of musical styles within this category.

Cheerleader - Felix Jaehn Remix Radio Edit
https://p.scdn.co/mp3-preview/3b8b1f0961ad4b68d3e08de52489ee509062342d?cid=3c807945a2cc4df0bfc4d2143a091a27

Did the song appeal to you? (Yes/No/Stop)
yes

Don't Be so Hard on Yourself
https://p.scdn.co/mp3-preview/631d73351935256bf1e37fc6993a416ce0f4e2f7?cid=3c807945a2cc4df0bfc4d2143a091a27

Did you enjoy listening to the song? (Yes/No/Stop)
no

No Lie
There is no music preview available for this song.

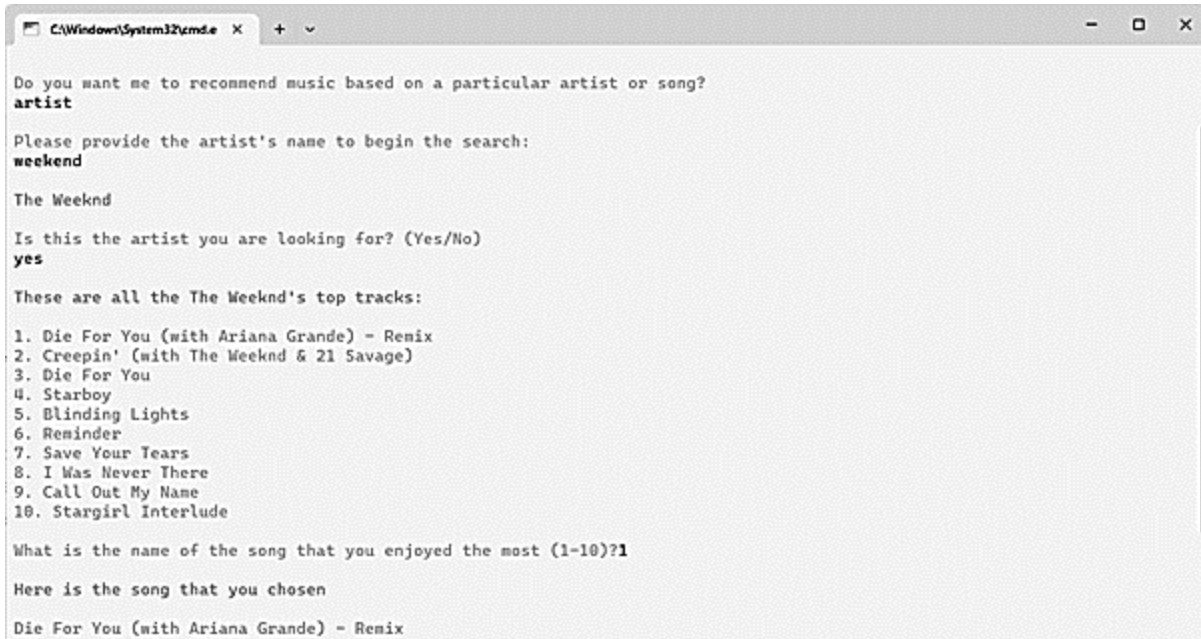
Are you liking the song? (Yes/No/Stop)
no

Not About Angels
https://p.scdn.co/mp3-preview/5b875f42ff2e31b95095652b10447ce1a6292b22?cid=3c807945a2cc4df0bfc4d2143a091a27

Was the song to your liking? (Yes/No/Stop)
```

Fig 10.15 New recommendation

The preview URL will not be available for every song, so the chatbot will say to the user that the preview URL isn't available for this song.



```
C:\Windows\System32\cmd.exe x + v

Do you want me to recommend music based on a particular artist or song?
artist

Please provide the artist's name to begin the search:
weekend

The Weeknd

Is this the artist you are looking for? (Yes/No)
yes

These are all the The Weeknd's top tracks:

1. Die For You (with Ariana Grande) - Remix
2. Creepin' (with The Weeknd & 21 Savage)
3. Die For You
4. Starboy
5. Blinding Lights
6. Reminder
7. Save Your Tears
8. I Was Never There
9. Call Out My Name
10. Stargirl Interlude

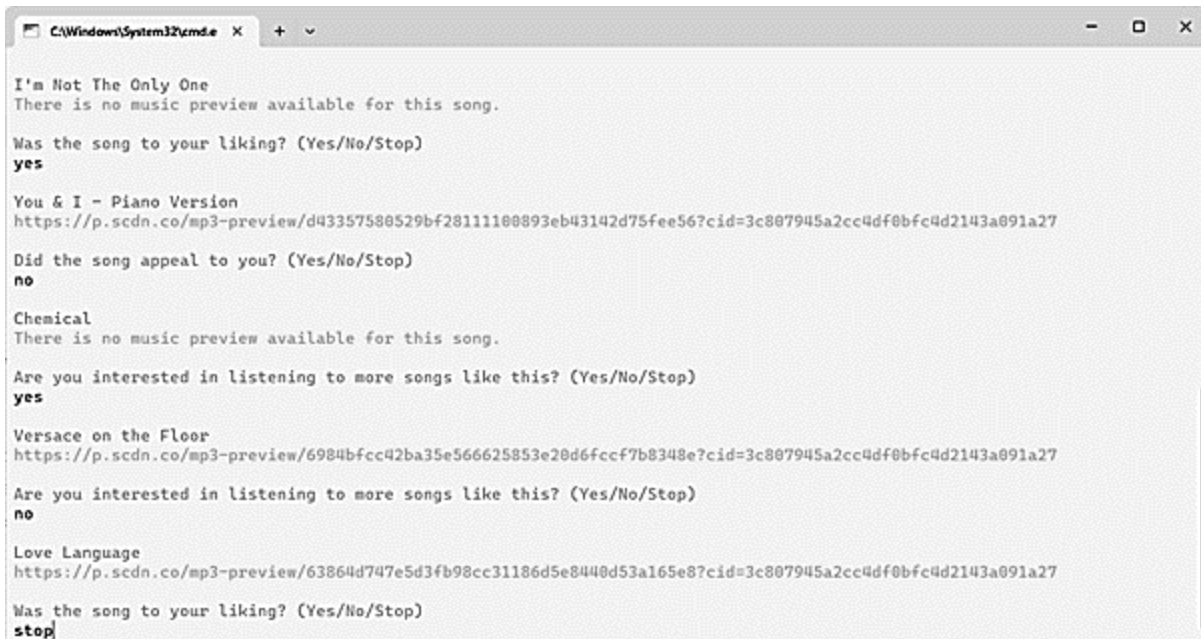
What is the name of the song that you enjoyed the most (1-10)?1

Here is the song that you chosen

Die For You (with Ariana Grande) - Remix
```

Fig 10.16 Error handling

If the user's response is "no", the chatbot repeats steps with a new recommendation. If the user's response is "stop", the chatbot says goodbye and ends the conversation.



```
C:\Windows\System32\cmd.exe x + v

I'm Not The Only One
There is no music preview available for this song.

Was the song to your liking? (Yes/No/Stop)
yes

You & I - Piano Version
https://p.scdn.co/mp3-preview/d43357580529bf28111100893eb43142d75fee56?cid=3c807945a2cc4df0bfc4d2143a091a27

Did the song appeal to you? (Yes/No/Stop)
no

Chemical
There is no music preview available for this song.

Are you interested in listening to more songs like this? (Yes/No/Stop)
yes

Versace on the Floor
https://p.scdn.co/mp3-preview/6984bfcc42ba35e566625853e20d6fccf7b8348e?cid=3c807945a2cc4df0bfc4d2143a091a27

Are you interested in listening to more songs like this? (Yes/No/Stop)
no

Love Language
https://p.scdn.co/mp3-preview/63864d747e5d3fb98cc31186d5e8440d53a165e8?cid=3c807945a2cc4df0bfc4d2143a091a27

Was the song to your liking? (Yes/No/Stop)
stop
```

Fig 10.17 End the program

CHAPTER 11

CONCLUSION AND FUTURE ENHANCEMENT

11.1 CONCLUSION

In conclusion, The Critiquing-Based Music Recommendation Chatbot is a promising development in the field of music recommendation technology. Its two-way recommendation system offers users the flexibility to search for recommendations either by artist or by song, and its feedback method allows users to review and refine their recommendations over time. This personalized approach to music recommendation is a significant improvement over more traditional approaches, which can be limiting and fail to consider the unique preferences and tastes of individual users.

The potential impact of the Critiquing-Based Music Recommendation Chatbot is significant, not only for individual users but also for the music industry as a whole. By facilitating the discovery of new and emerging artists, the chatbot has the potential to drive growth and innovation in the industry, providing a platform for talented musicians to showcase their work to a wider audience.

As the chatbot continues to be developed and refined, it is likely that it will become even more sophisticated and effective in its recommendations. By leveraging the power of machine learning and artificial intelligence, the chatbot may eventually be able to make predictions about users' musical tastes and preferences before they even know them themselves. This could revolutionize the way that people discover and enjoy music, and pave the way for a more personalized and engaging listening experience in the years to come. In short, the Critiquing-Based Music Recommendation Chatbot is an exciting development that has the potential to shape the future of music discovery and consumption.

11.2 FUTURE ENHANCEMENT

1. Error handling: The code currently assumes that all API requests will be successful. However, there may be scenarios where the API returns errors, or the code fails to process the API response. Implementing robust error handling can make the code more reliable and user-friendly.
2. Expand search capabilities: Currently, the code only searches for artists and tracks based on the exact name provided by the user. Expanding the search capabilities to include similar names or misspellings can provide more accurate results.
3. Implement more recommendation algorithms: The current code uses only two recommendation algorithms: based on an artist and based on a song. Implementing additional recommendation algorithms, such as based on genre, mood, or popularity, can provide a more diverse set of recommendations.
4. Improve user interface: Currently, the code uses a simple text-based interface that can be improved to provide a more user-friendly experience. Implementing a Graphical User Interface (GUI) can allow the users to interact with the code more easily and intuitively.
5. Use caching to improve performance: The code currently makes API requests every time a recommendation is requested. Implementing caching can allow the code to store the results of previous requests and reuse them when possible, which can improve performance and reduce the number of API requests made.

REFERENCES

- [1] Wanling Cai , Yucheng Jin , and Li Chen, “Task-Oriented User Evaluation on Critiquing-Based Recommendation Chatbots” vol. 52, no. 3, June 2022.
- [2] Prof. Suvarna Bahir, Amaan Shaikh, Bhushan Patil, Tejas Sonawane, “Chat Bot Song Recommender System” Volume:04/Issue:04/April-2022.
- [3] Nishtha Kapoor, Arushi Gupta, Gulshan Kumar, Dhruv Aggarwal, “MU-SYNC - A MUSIC RECOMMENDATION BOT” 8 pp. 19-21. 2022.
- [4] Shivam Sakore, Pratik Jagdale, Mansi Borawake, Ankita Khandalkar, “Music Recommender System Using Chatbot” Volume 9, Dec 2021.
- [5] Amrita Nair, Smriti Pillai, Ganga S Nair, Anjali T, “Emotion Based Music Playlist Recommendation System Using Interactive Chatbot” Volume: 06, 2021.
- [6] Prof. N.S.Sharma, Amaan Shaikh, Bhushan Patil, Tejas Sonawane, “A Machine Learning Based Chatbot Song Recommender System” Volume 8, 2021.
- [7] W. Cai, Y. Jin, and L. Chen, “Critiquing for music exploration in conversational recommender systems,” in Proc. 26th Int. Conf. Intell. User Interfaces, pp. 480–490, 2021.
- [8] Y. Jin, N. Tintarev, N. N. Htun, and K. Verbert, “Effects of personal characteristics in control-oriented user interfaces for music recommender systems,” User Model. User-Adapted Interact., vol. 30, no. 2, pp. 199–249, 2020.
- [9] W. Cai and L. Chen, “Predicting user intents and satisfaction with dialogue-based conversational recommendations,” in Proc. 28th ACM Conf. User Model., Adapt. Personalization, pp. 33–42, 2020.
- [10] Anand Neil Arnold, Vairamuthu S, “Music Recommendation using Collaborative Filtering and Deep Learning” Volume-8, 2019.