



BHARATI VIDYAPEETH'S COLLEGE OF ENGINEERING FOR WOMEN,
KATRAJ-DHANKWADI, PUNE-43

MINI-PROJECT REPORT

ON

“Music Recommendation System”

SUBMITTED BY

MISS. Anushruti Adhikari (T190344202)

Faculty: Prof. J.D.Jadhav

DEPARTMENT OF COMPUTER ENGINEERING

ACADEMIC YEAR -**2023-24**



BHARATI VIDYAPEETH'S COLLEGE OF ENGINEERING FOR WOMEN,
KATRAJ-DHANKWADI, PUNE-43

CERTIFICATE

This is to certify that,

MISS. Anushruti Adhikari (T190344202)

Have successfully completed the Mini-Project titled

“Music Recommendation System”

During the academic year 2023-24 towards fulfilment of the

Third Year in Computer Engineering under Savitribai Phule Pune University.

Prof. J.D.Jadhav
(Internal Guide)

Prof. D.D.Pukale
(HOD Comp Engg.)

ABSTRACT

The Music Recommendation System represents a significant advancement in the realm of personalized music discovery, utilizing cutting-edge data science methodologies and natural language processing (NLP) techniques. In an era inundated with an overwhelming abundance of musical content, the need for tailored recommendations tailored to individual tastes has never been more pronounced. This project addresses this need by leveraging a comprehensive dataset sourced from Spotify, a leading music streaming platform, containing essential metadata and lyrics for a vast array of songs.

Through meticulous data preprocessing procedures, including cleansing, tokenization, and feature engineering, the system transforms raw data into a format conducive to analysis. Key steps include the removal of redundant information, handling of missing values, and the application of NLP techniques such as tokenization and stemming to distill the essence of song lyrics. The resulting dataset is then subjected to TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, a method that quantifies the importance of words in a document relative to a corpus, thereby enabling the system to derive semantic similarities between songs.

Central to the functionality of the Music Recommendation System is the calculation of cosine similarity scores between song vectors, a measure that quantifies the degree of similarity between two songs based on their lyrical content. This similarity metric forms the foundation of the recommendation engine, allowing users to input a song of their choice and receive a curated list of songs deemed most similar by the system. Moreover, the recommendation engine has been integrated into a user-friendly web interface using Streamlit, facilitating seamless interaction and exploration for users.

This project not only showcases the technical prowess of data science and machine learning but also underscores their practical applications in enhancing user experiences in the digital music landscape. By harnessing the power of NLP and advanced algorithms, the Music Recommendation System represents a significant step towards democratizing music discovery, empowering users to explore and appreciate diverse musical genres tailored to their unique preferences.

INDEX

Sr No	Contents	Page No
1	Introduction	1
2	Problem Statement	2
3	Objectives	3
4	Motivation and rationale of the study	4
5	Methodological Details	5
6	Implementation code	11
7	Results	17
8	Future Scope	20
9	Conclusion	22
10	Reference	23

CHAPTER 1: INTRODUCTION

The advent of digital streaming platforms has revolutionized the music industry, granting users unparalleled access to an extensive catalog of songs from around the globe. However, with this abundance of choice comes the daunting challenge of navigating through a vast sea of music to discover new favorites. Recognizing this dilemma, the Music Recommendation System emerges as a beacon of innovation, offering users a personalized gateway to musical exploration.

The primary objective of the Music Recommendation System is to alleviate the burden of choice by leveraging the power of data science and natural language processing (NLP) to deliver tailored recommendations based on individual preferences. At its core, the system analyzes song metadata and lyrics to discern underlying patterns and similarities, facilitating the identification of songs that resonate with the user's musical tastes.

In today's interconnected world, where streaming platforms abound and musical diversity knows no bounds, the need for intelligent recommendation systems has never been more pressing. Whether it's uncovering hidden gems within a preferred genre or venturing into uncharted musical territories, the Music Recommendation System stands ready to guide users on a journey of sonic discovery.

This report delves into the intricacies of the Music Recommendation System, elucidating its underlying methodologies, implementation details, and potential impact on the user experience. By exploring the system's data preprocessing techniques, recommendation algorithms, and user interface design, we aim to provide a comprehensive understanding of its functionality and significance in the ever-evolving landscape of digital music consumption. Through this exploration, we endeavor to showcase the transformative potential of data science in revolutionizing the way we interact with and explore music in the digital age.

CHAPTER 2: PROBLEM STATEMENT

Develop a music recommendation model using the streamlit library in python.

Refer to Spotify Million Playlist Dataset: <https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset>

CHAPTER 3: OBJECTIVES

- 1. Develop a Music Recommendation System:** Create a robust recommendation system capable of providing personalized song suggestions to users based on their individual musical preferences and tastes.
- 2. Utilize Spotify Data:** Leverage the Spotify API and datasets containing information about songs, artists, and user interactions to gather comprehensive data for training the recommendation algorithm.
- 3. Implement Machine Learning Algorithms:** Apply machine learning techniques to analyze the collected data and develop algorithms capable of identifying patterns in user preferences and generating accurate recommendations.
- 4. Enhance User Experience:** Design an intuitive user interface using Streamlit to facilitate seamless interaction, allowing users to input preferences and receive personalized recommendations effortlessly.
- 5. Evaluate System Performance:** Conduct rigorous testing and evaluation to assess the efficacy and accuracy of the recommendation system in providing relevant and engaging song suggestions to users.
- 6. Explore Future Enhancements:** Identify potential avenues for improvement and future enhancements, such as incorporating additional data sources, refining recommendation algorithms, and adapting to evolving user preferences and behaviors.

CHAPTER 4: Motivation and Rationale of the study

The development of the Music Recommendation System stems from a confluence of factors and motivations driven by the evolving landscape of music consumption and the increasing demand for personalized experiences. The following key motivations and rationales underpin the inception and execution of this study:

1. Information Overload: In today's digital age, music consumers are inundated with an overwhelming abundance of choices, making it increasingly challenging to navigate and discover new music that aligns with their tastes and preferences. The Music Recommendation System seeks to address this issue by leveraging advanced algorithms to curate personalized recommendations tailored to individual users, thereby alleviating the burden of choice and streamlining the music discovery process.

2. Enhanced User Experience: Music consumption is not merely a passive activity but an immersive and interactive experience that deeply resonates with individuals on an emotional and personal level. By providing users with relevant and engaging recommendations that reflect their musical preferences, the Music Recommendation System aims to enhance the overall user experience, fostering a deeper connection and engagement with the music content.

3. Exploration and Discovery: While users may have established preferences and favorite artists, there is inherent value in exploring new and diverse musical genres and discovering hidden gems that may lie beyond one's immediate scope of familiarity. The Music Recommendation System encourages exploration and discovery by introducing users to a wide range of musical content that they may not have encountered through traditional means, thereby broadening their musical horizons and enriching their listening experiences.

4. Community and Connection: Music has the power to unite people across geographical boundaries and cultural divides, fostering a sense of community and connection among individuals with shared musical interests. By facilitating the sharing of music recommendations and discoveries, the Music Recommendation System serves as a catalyst for building and nurturing communities of music enthusiasts, promoting dialogue, and fostering a sense of belonging and camaraderie.

In summary, the development of the Music Recommendation System is motivated by a desire to address the challenges posed by information overload, enhance the user experience, promote exploration and discovery, foster community and connection, and leverage technological advancements to create a more personalized and engaging music discovery platform. By aligning with these motivations and rationales, the study aims to make a meaningful contribution to the field of music recommendation and enhance the overall quality of the music listening experience for users worldwide.

CHAPTER 5: METHODOLOGICAL DETAILS

The Music Recommendation System epitomizes a groundbreaking endeavor at the forefront of digital music discovery, meticulously engineered to cater to the diverse tastes and preferences of users navigating the expansive realm of musical content. At its inception, the project initiates a comprehensive data collection process from Spotify, a premier platform renowned for its extensive music library. This dataset is meticulously curated to encompass a wealth of song metadata, including attributes such as song titles, artists, albums, release years, and popularity rankings, alongside the complete lyrics of each song. Subsequently, the data undergoes an exhaustive preprocessing pipeline, wherein it is subjected to stringent quality control measures to ensure integrity and reliability. This involves thorough data cleaning to rectify discrepancies, handling missing values with precision, and standardizing text data through tokenization, stemming, and normalization techniques. Furthermore, leveraging the transformative power of TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, the lyrical content of each song is transformed into numerical feature vectors, enabling the quantification of semantic similarities between songs through cosine similarity calculations.

The heart of the Music Recommendation System lies in its recommendation algorithm, meticulously crafted to decipher the intricate nuances of musical compositions and discern patterns that resonate with users' musical preferences. Drawing upon the computed cosine similarity scores, the algorithm employs advanced techniques to identify songs with comparable lyrical themes and content, thereby facilitating the generation of personalized recommendations tailored to individual users. Moreover, the integration of cutting-edge technologies such as machine learning and natural language processing empowers the system to adapt and evolve over time, continuously refining its recommendations based on user feedback and interactions.

In tandem with its robust recommendation engine, the Music Recommendation System boasts an intuitive and user-centric interface developed using Streamlit, a Python library for building interactive web applications. Through this interface, users are afforded a seamless and immersive music discovery experience, wherein they can effortlessly input their favorite songs and explore curated recommendations that align closely with their musical preferences. Additionally, the interface facilitates dynamic interaction and engagement, empowering users to provide feedback, refine their preferences, and delve deeper into the diverse tapestry of musical genres and artists.

In essence, the Music Recommendation System transcends the traditional paradigms of music discovery, ushering in a new era of personalized exploration and immersion. Through its fusion of cutting-edge technology, sophisticated algorithms, and intuitive design, the system empowers users to embark on a journey of musical discovery tailored to their unique tastes and interests, fostering a deeper connection with the boundless wonders of music in all its forms.

Methodology to develop the project:**1. Data Collection:****Source:**

The data for the Music Recommendation System is sourced from Spotify, one of the largest and most popular music streaming platforms globally. Spotify offers a vast repository of songs spanning various genres, artists, and albums, making it an ideal source for building a comprehensive music recommendation system.

Dataset:

The dataset comprises a diverse range of song metadata and lyrics, providing valuable insights into each song's characteristics and content. This includes attributes such as song titles, artists, albums, release years, and popularity rankings, which are essential for understanding the context and relevance of each song. Additionally, the dataset includes the complete lyrics of each song, allowing for a deeper analysis of the lyrical content and themes.

Collection Process:

Data collection is facilitated through the Spotify API (Application Programming Interface), which allows developers to programmatically access Spotify's extensive music database. Using the Spotify API, the project queries relevant endpoints to retrieve song metadata and lyrics for inclusion in the dataset. This process involves formulating specific requests to retrieve information such as song titles, artists, albums, and release years, as well as accessing additional endpoints or third-party APIs to obtain the complete lyrics for each song.

2. Data Preprocessing:**Cleaning:**

The Data Preprocessing stage begins with cleaning the dataset to ensure its integrity and consistency. This involves identifying and handling any anomalies, discrepancies, or irregularities in the data that may affect its quality or reliability. Common tasks in the cleaning process include:

- Removing duplicate entries: Eliminating duplicate rows or records to avoid redundancy and ensure each data point is unique.
- Handling missing values: Addressing any missing or null values in the dataset by either imputing them with appropriate values or removing them altogether.
- Rectifying inconsistencies: Resolving inconsistencies or errors in the data, such as typos, formatting issues, or incorrect values, to maintain data accuracy.

Tokenization:

Tokenization is a crucial step in preprocessing textual data, such as song lyrics, which involves breaking down the text into individual words or tokens. This process facilitates further analysis by

enabling the system to interpret and process the textual data more effectively. Tokenization may also involve removing punctuation marks, special characters, and stopwords (commonly occurring words with little semantic significance) to streamline the text for analysis.

Stemming and Normalization:

Stemming and normalization are techniques used to standardize and simplify textual data. Stemming involves reducing words to their root or base form, known as the "stem," by removing prefixes and suffixes. This helps reduce the dimensionality of the text data and ensures consistency in vocabulary usage. Normalization, on the other hand, involves converting all text to a consistent format, such as lowercase, to eliminate variations in case and ensure uniformity in data representation. By applying stemming and normalization techniques, the system can standardize the vocabulary and reduce the complexity of the text data, making it more suitable for analysis.

Overall, the Data Preprocessing stage plays a crucial role in preparing the dataset for analysis and model training. By cleaning, tokenizing, and standardizing the data, the preprocessing pipeline ensures data quality and consistency, laying the groundwork for more effective and accurate analysis in subsequent stages of the project.

3. Feature Engineering:

Feature engineering is a critical stage in the development of machine learning models, involving the creation and selection of relevant features or variables from raw data to improve model performance. In the context of the Music Recommendation System, feature engineering primarily focuses on transforming textual data, such as song lyrics, into numerical feature representations that can be effectively utilized by machine learning algorithms. The key techniques employed in feature engineering include:

TF-IDF Vectorization:

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is a popular technique used to convert textual data into numerical feature vectors. It works by assigning weights to words based on their frequency within a document (term frequency) and their importance relative to the entire corpus of documents (inverse document frequency). Specifically, TF-IDF calculates a weight for each word in the lyrics of a song, where words that occur frequently within the lyrics but infrequently across all songs in the dataset are assigned higher weights, indicating their significance in representing the content of the song.

Numerical Feature Representation:

Once the lyrics of each song have been transformed into TF-IDF weighted vectors, they serve as numerical feature representations of the lyrical content. These feature vectors capture the semantic similarities and distinctions between songs based on the words they contain and their respective importance within the lyrics. By utilizing numerical feature representations derived from TF-IDF vectorization, the Music Recommendation System can quantify the lyrical similarity between songs and generate personalized recommendations tailored to individual users' preferences.

Overall, feature engineering plays a crucial role in enhancing the effectiveness and interpretability of machine learning models by transforming raw data into meaningful and informative features. In the context of the Music Recommendation System, TF-IDF vectorization enables the system to leverage the textual content of song lyrics to generate accurate and personalized recommendations, thereby enhancing the music discovery experience for users.

4. Recommendation Algorithm:

The Recommendation Algorithm employed in the Music Recommendation System leverages cosine similarity and nearest neighbor search techniques to generate personalized song recommendations based on the lyrical content of songs. Here's how it works:

Cosine Similarity:

Cosine similarity is a measure used to quantify the similarity between two non-zero vectors in a multi-dimensional space. In the context of the Music Recommendation System, the TF-IDF weighted vectors derived from the lyrics of songs serve as the vectors. Cosine similarity computes the cosine of the angle between two vectors, resulting in a value between -1 and 1. A cosine similarity score of 1 indicates that the two vectors are identical, while a score of -1 indicates complete dissimilarity.

Nearest Neighbor Search:

Upon receiving a user input song, the system calculates the cosine similarity scores between the TF-IDF weighted vector representation of the input song and all other songs in the dataset. These cosine similarity scores quantify the lyrical similarity between the input song and each song in the dataset. The system then identifies the nearest neighbors based on these similarity scores, selecting the songs that exhibit the highest degree of similarity to the input song.

Generating Recommendations:

Once the nearest neighbors are identified, the system recommends songs that are deemed to be the most similar to the input song in terms of lyrical content. These recommended songs are selected based on their cosine similarity scores, with higher scores indicating a greater degree of similarity to the input song. By employing cosine similarity and nearest neighbor search techniques, the Recommendation Algorithm facilitates the generation of personalized song recommendations tailored to individual users' preferences, thereby enhancing the music discovery experience.

In summary, the Recommendation Algorithm plays a pivotal role in the Music Recommendation System by leveraging cosine similarity and nearest neighbor search techniques to generate accurate and personalized song recommendations based on the lyrical content of songs. This algorithm enables the system to identify songs with similar lyrical themes and content, thereby facilitating an immersive and engaging music discovery experience for users.

5. User Interface Design:

The User Interface (UI) Design of the Music Recommendation System aims to provide users with an intuitive and interactive platform for exploring and interacting with personalized song recommendations. Here's an overview of its key components and functionalities:

Streamlit Integration:

The Music Recommendation System utilizes Streamlit, a Python library for building interactive web applications, to develop its user interface. Streamlit offers a simple and straightforward approach to creating web-based applications, enabling developers to design and deploy interactive UIs with ease. By leveraging Streamlit, the Music Recommendation System can provide users with a seamless and immersive music discovery experience directly within their web browser.

Input Interface:

The UI features an input interface where users can interact with the system by providing input in the form of their favorite songs or artists. This input serves as the basis for generating personalized song recommendations tailored to the user's preferences. The input interface may include dropdown menus, text input fields, or other interactive elements that allow users to specify their preferences and refine their search criteria.

Recommendation Display:

Once the user inputs their preferences, the UI displays personalized song recommendations generated by the Recommendation Algorithm. These recommendations are presented in a visually appealing format, accompanied by relevant metadata such as song titles, artists, albums, and popularity rankings. Users can explore the recommended songs and click on individual recommendations to access additional details or listen to song previews.

Interactive Elements:

The UI may incorporate various interactive elements to enhance user engagement and facilitate seamless navigation. This includes buttons for triggering recommendation generation, dropdown menus for selecting input options, sliders for adjusting preferences, and checkboxes for filtering results. These interactive elements empower users to customize their music discovery experience and interact with the system in a dynamic and intuitive manner.

Feedback Mechanism:

To foster user engagement and improve recommendation accuracy, the UI may include mechanisms for collecting feedback from users. This could involve providing options for users to rate recommended songs, indicate preferences, or provide comments and suggestions. By incorporating a feedback mechanism, the system can continuously refine its recommendations based on user interactions and preferences.

Overall, the User Interface Design of the Music Recommendation System is designed to provide users with a seamless and enjoyable music discovery experience. By leveraging Streamlit and incorporating intuitive interactive elements, the UI empowers users to explore personalized song

recommendations, discover new music, and engage with the system in a dynamic and interactive manner.

In summary, the methodology of the Music Recommendation System encompasses a multifaceted approach, beginning with the collection and preprocessing of data sourced from Spotify. Through meticulous cleaning, tokenization, and feature engineering, the system transforms raw data into actionable insights, laying the groundwork for a robust recommendation engine. Leveraging cosine similarity and nearest neighbor search techniques, the recommendation algorithm generates personalized song recommendations based on the lyrical content of songs. Integration with Streamlit facilitates a seamless user interface, empowering users to explore and interact with recommendations intuitively. Through continuous evaluation and refinement, the methodology ensures the system's effectiveness in delivering personalized music discovery experiences tailored to individual preferences. Overall, the methodology represents a harmonious blend of advanced technologies and user-centric design principles, aimed at enriching the music exploration journey for users worldwide.

CHAPTER 6: IMPLEMENTATION CODE

i. mini.ipynb

```

In [45]: import pandas as pd

In [46]: df = pd.read_csv("D:\Anushruti_Data\Anushruti\College Materials\Bhartiya Vidyapeeth's College Of Engine
Out[47]:
  artist      song      link      text
0  ABBA  Ahe's My Kind Of Girl  /a/abba/ahes+my+kind+of+girl_20598417.html  Look at her face, it's a wonderful face \nA...
1  ABBA    Andante, Andante  /a/abba/andante+andante_20002708.html  Take it easy with me, please \nTouch me gen...
2  ABBA  As Good As New  /a/abba/as+good+as+new_20003033.html  I'll never know why I had to go \nWhy I had...
3  ABBA      Bang  /a/abba/bang_20598415.html  Making somebody happy is a question of give an...
4  ABBA  Bang-A-Boomerang  /a/abba/bang+a+boomerang_20002668.html  Making somebody happy is a question of give an...

In [48]: df.tail(5)
Out[48]:
  artist      song      link      text

```

```

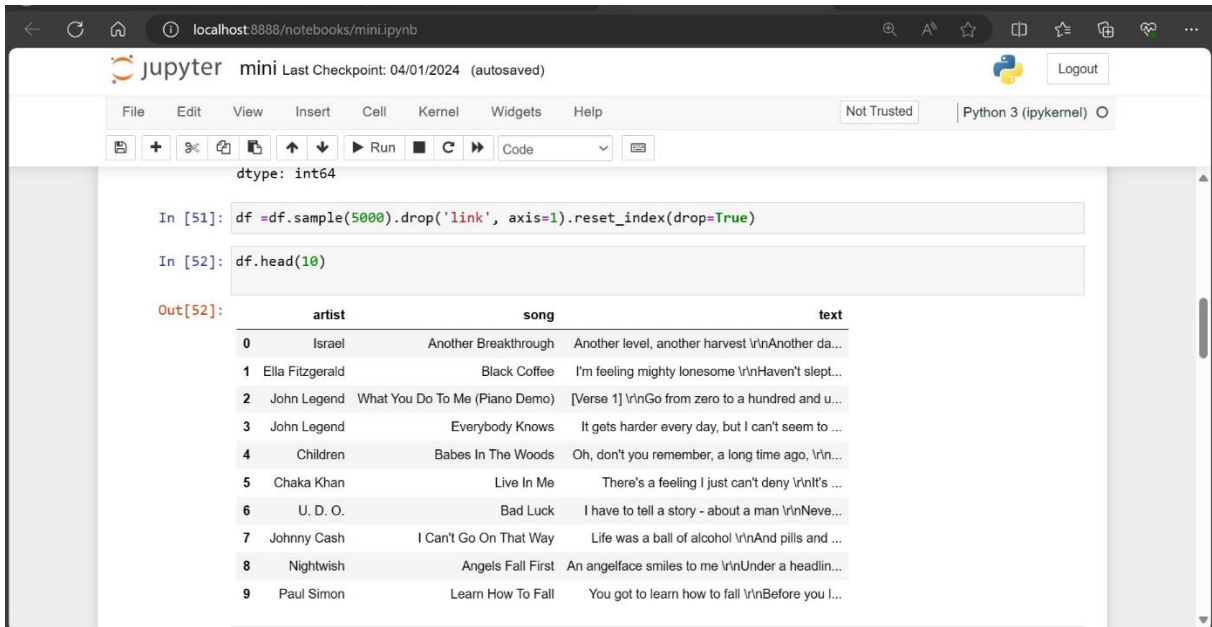
Out[48]:
  artist      song      link      text
57645  Ziggy Marley  Good Old Days  /z/ziggy+marley/good+old+days_10198588.html  Irie days come on play \nLet the angels fly...
57646  Ziggy Marley  Hand To Mouth  /z/ziggy+marley/hand+to+mouth_20531167.html  Power to the workers \nMore power \nPowe...
57647  Zwan  Come With Me  /z/zwan/come+with+me_20148981.html  all you need \nis something i'll believe \...
57648  Zwan  Desire  /z/zwan/desire_20148986.html  northern star \nam i frightened \nwhere ...
57649  Zwan  Heartsong  /z/zwan/heartsong_20148991.html  come in \nmake yourself at home \ni'm a ...

In [49]: df.shape
Out[49]: (57650, 4)

In [50]: df.isnull().sum()
Out[50]:
artist    0
song      0
link      0
text      0
dtype: int64

```

Music Recommendation System



A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/mini.ipynb'. The Jupyter logo and 'mini' are in the top left, with 'Last Checkpoint: 04/01/2024 (autosaved)' and a 'Logout' button in the top right. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The status bar shows 'Not Trusted' and 'Python 3 (ipykernel)'. The code area contains two cells: In [51] with `df = df.sample(5000).drop('link', axis=1).reset_index(drop=True)` and In [52] with `df.head(10)`. The output area shows a table with 10 rows and 3 columns: artist, song, and text.

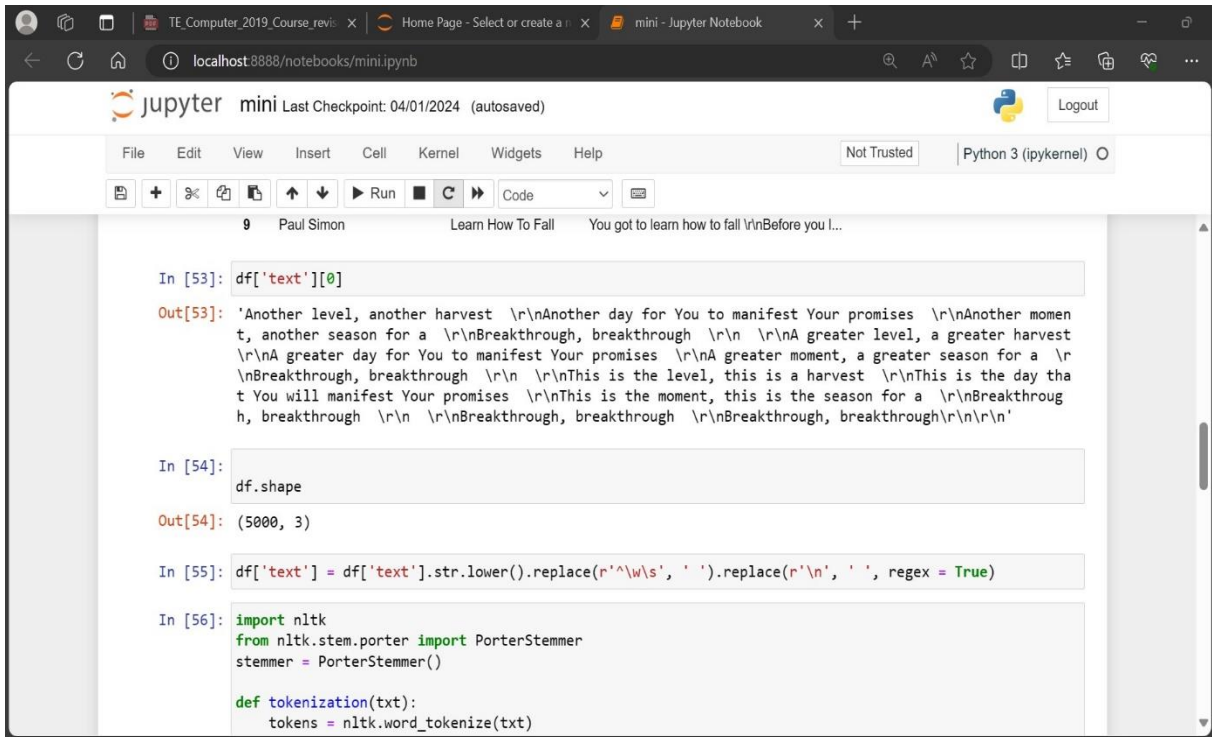
```
dtype: int64
```

```
In [51]: df = df.sample(5000).drop('link', axis=1).reset_index(drop=True)
```

```
In [52]: df.head(10)
```

```
Out[52]:
```

	artist	song	text
0	Israel	Another Breakthrough	Another level, another harvest \r\nAnother da...
1	Ella Fitzgerald	Black Coffee	I'm feeling mighty lonesome \r\nHaven't slept...
2	John Legend	What You Do To Me (Piano Demo)	[Verse 1] \r\nGo from zero to a hundred and u...
3	John Legend	Everybody Knows	It gets harder every day, but I can't seem to ...
4	Children	Babes In The Woods	Oh, don't you remember, a long time ago, \r\n...
5	Chaka Khan	Live In Me	There's a feeling I just can't deny \r\nIt's ...
6	U. D. O.	Bad Luck	I have to tell a story - about a man \r\nNeve...
7	Johnny Cash	I Can't Go On That Way	Life was a ball of alcohol \r\nAnd pills and ...
8	Nightwish	Angels Fall First	An angelface smiles to me \r\nUnder a headlin...
9	Paul Simon	Learn How To Fall	You got to learn how to fall \r\nBefore you l...



A screenshot of a Jupyter Notebook interface, continuing from the previous one. The browser address bar shows 'localhost:8888/notebooks/mini.ipynb'. The Jupyter logo and 'mini' are in the top left, with 'Last Checkpoint: 04/01/2024 (autosaved)' and a 'Logout' button in the top right. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The status bar shows 'Not Trusted' and 'Python 3 (ipykernel)'. The code area contains four cells: In [53] with `df['text'][0]`, In [54] with `df.shape`, In [55] with `df['text'] = df['text'].str.lower().replace(r'^\w\s', ' ').replace(r'\n', ' ', regex = True)`, and In [56] with `import nltk`, `from nltk.stem.porter import PorterStemmer`, `stemmer = PorterStemmer()`, and a function definition `def tokenization(txt):` followed by `tokens = nltk.word_tokenize(txt)`. The output area shows the result of In [53] as a long string of text with line breaks.

```
In [53]: df['text'][0]
```

```
Out[53]: 'Another level, another harvest \r\nAnother day for You to manifest Your promises \r\nAnother momen
```

```
In [54]: df.shape
```

```
Out[54]: (5000, 3)
```

```
In [55]: df['text'] = df['text'].str.lower().replace(r'^\w\s', ' ').replace(r'\n', ' ', regex = True)
```

```
In [56]: import nltk
```

```
from nltk.stem.porter import PorterStemmer
```

```
stemmer = PorterStemmer()
```

```
def tokenization(txt):
```

```
    tokens = nltk.word_tokenize(txt)
```


Music Recommendation System

```
localhost:8888/notebooks/mini.ipynb
jupyter mini Last Checkpoint: 04/01/2024 (autosaved)
Python 3 (ipykernel)

def tokenization(txt):
    tokens = nltk.word_tokenize(txt)
    stemming = [stemmer.stem(w) for w in tokens]
    return " ".join(stemming)

In [57]: df['text'] = df['text'].apply(lambda x: tokenization(x))

In [58]: from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

In [59]: tfidfvector = TfidfVectorizer(analyzer='word', stop_words='english')
matrix = tfidfvector.fit_transform(df['text'])

In [60]: print(matrix.toarray())

[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
```

```
localhost:8888/notebooks/mini.ipynb
jupyter mini Last Checkpoint: 04/01/2024 (autosaved)
Python 3 (ipykernel)

[0. 0. 0. ... 0. 0. 0.]]

In [61]: similarity = cosine_similarity(matrix)
similarity[0]

Out[61]: array([1.          , 0.01247876, 0.          , ..., 0.00355533, 0.          ,
                0.          ])

In [62]: df[df['song'] == 'Crying Over You']

Out[62]:
  artist  song  text
1198  UB40  Crying Over You  cri over you in the morn cri over you in the e...

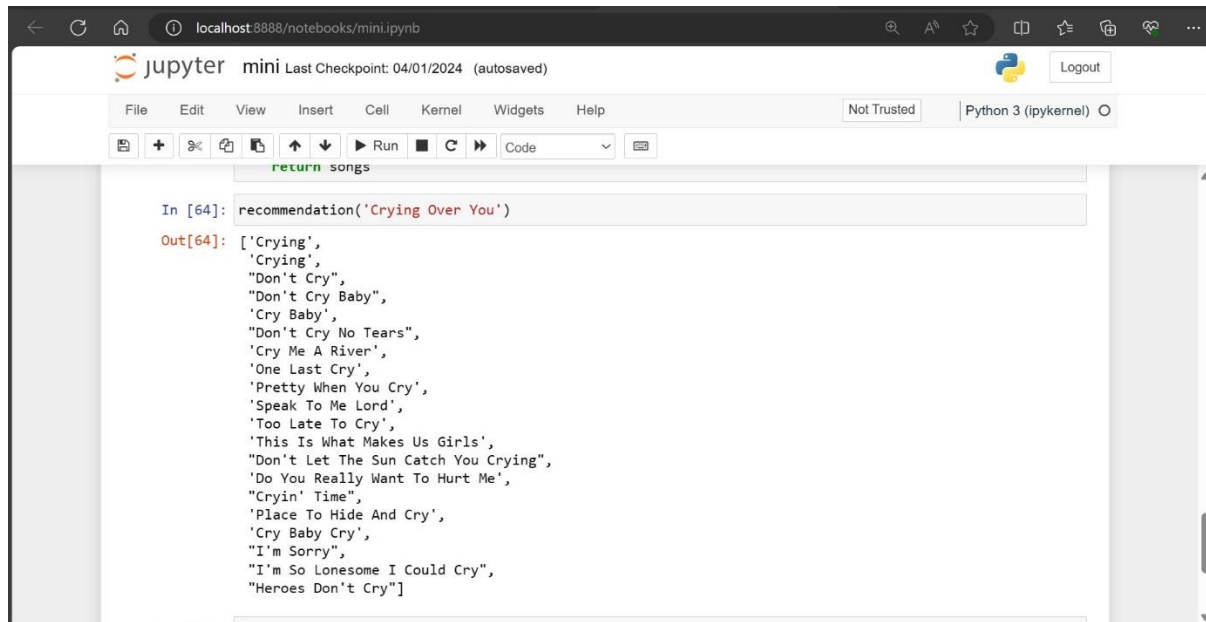
In [63]: def recommendation(song_df):
    idx = df[df['song'] == song_df].index[0]
    distances = sorted(list(enumerate(similarity[idx])), reverse=True, key=lambda x: x[1])

    songs = []
    for m_id in distances[1:21]:
        songs.append(df.iloc[m_id[0]].song)

    return songs

In [64]: recommendation('Crying Over You')
```

Music Recommendation System

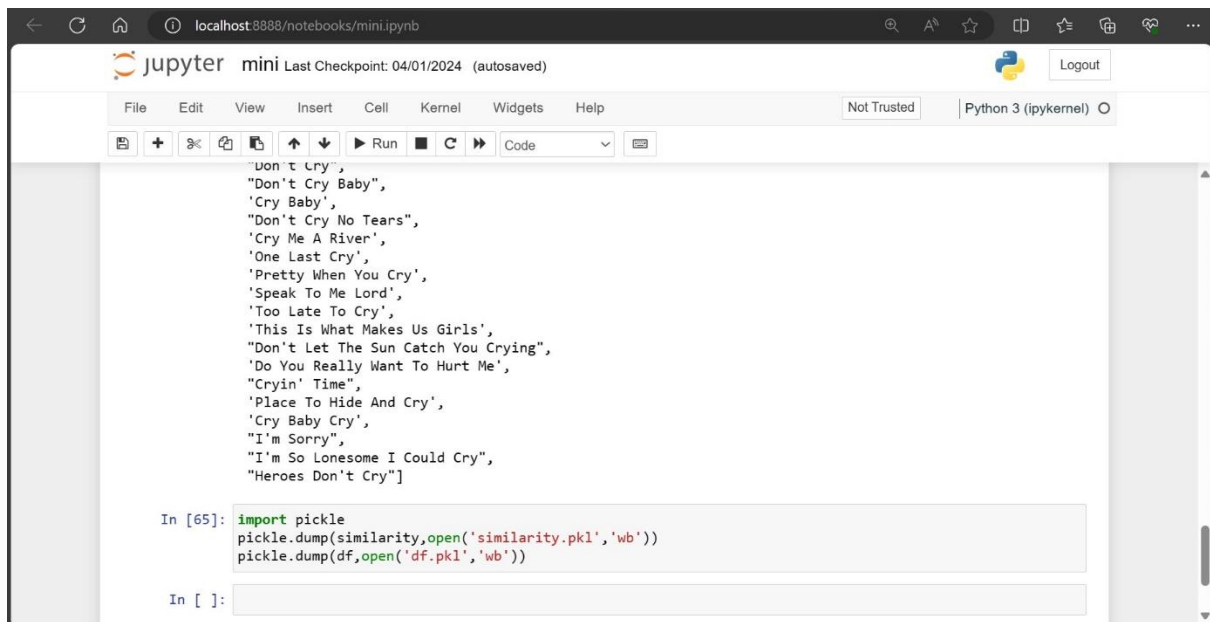


A screenshot of a Jupyter Notebook interface. The browser address bar shows 'localhost:8888/notebooks/mini.ipynb'. The Jupyter logo and 'mini' are in the top left, with 'Last Checkpoint: 04/01/2024 (autosaved)' and a 'Logout' button in the top right. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The status bar shows 'Not Trusted' and 'Python 3 (ipykernel)'. The toolbar contains icons for file operations and execution. The code cell shows a function call: `In [64]: recommendation('Crying Over You')`. The output cell shows a list of 20 song titles: `Out[64]: ['Crying', 'Crying', 'Don't Cry', 'Don't Cry Baby', 'Cry Baby', 'Don't Cry No Tears', 'Cry Me A River', 'One Last Cry', 'Pretty When You Cry', 'Speak To Me Lord', 'Too Late To Cry', 'This Is What Makes Us Girls', 'Don't Let The Sun Catch You Crying', 'Do You Really Want To Hurt Me', 'Cryin' Time', 'Place To Hide And Cry', 'Cry Baby Cry', 'I'm Sorry', 'I'm So Lonesome I Could Cry', 'Heroes Don't Cry']`.

```
return songs

In [64]: recommendation('Crying Over You')

Out[64]: ['Crying',
'Crying',
'Don't Cry',
'Don't Cry Baby',
'Cry Baby',
'Don't Cry No Tears',
'Cry Me A River',
'One Last Cry',
'Pretty When You Cry',
'Speak To Me Lord',
'Too Late To Cry',
'This Is What Makes Us Girls',
'Don't Let The Sun Catch You Crying',
'Do You Really Want To Hurt Me',
'Cryin' Time',
'Place To Hide And Cry',
'Cry Baby Cry',
'I'm Sorry',
'I'm So Lonesome I Could Cry',
'Heroes Don't Cry']
```



A screenshot of a Jupyter Notebook interface, similar to the one above. The browser address bar shows 'localhost:8888/notebooks/mini.ipynb'. The Jupyter logo and 'mini' are in the top left, with 'Last Checkpoint: 04/01/2024 (autosaved)' and a 'Logout' button in the top right. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The status bar shows 'Not Trusted' and 'Python 3 (ipykernel)'. The toolbar contains icons for file operations and execution. The code cell shows the following code: `In [65]: import pickle
pickle.dump(similarity,open('similarity.pkl','wb'))
pickle.dump(df,open('df.pkl','wb'))`. The output cell is empty. The code cell above it shows a list of 20 song titles: `'Don't Cry', 'Don't Cry Baby', 'Cry Baby', 'Don't Cry No Tears', 'Cry Me A River', 'One Last Cry', 'Pretty When You Cry', 'Speak To Me Lord', 'Too Late To Cry', 'This Is What Makes Us Girls', 'Don't Let The Sun Catch You Crying', 'Do You Really Want To Hurt Me', 'Cryin' Time', 'Place To Hide And Cry', 'Cry Baby Cry', 'I'm Sorry', 'I'm So Lonesome I Could Cry', 'Heroes Don't Cry']`.

```
'Don't Cry',
'Don't Cry Baby',
'Cry Baby',
'Don't Cry No Tears',
'Cry Me A River',
'One Last Cry',
'Pretty When You Cry',
'Speak To Me Lord',
'Too Late To Cry',
'This Is What Makes Us Girls',
'Don't Let The Sun Catch You Crying',
'Do You Really Want To Hurt Me',
'Cryin' Time',
'Place To Hide And Cry',
'Cry Baby Cry',
'I'm Sorry',
'I'm So Lonesome I Could Cry',
'Heroes Don't Cry']

In [65]: import pickle
pickle.dump(similarity,open('similarity.pkl','wb'))
pickle.dump(df,open('df.pkl','wb'))

In [ ]:
```

ii. app.py

```

1 import pickle
2 import streamlit as st
3 import spotify
4 from spotify.oauth2 import SpotifyClientCredentials
5
6 CLIENT_ID = "6ee5b83acbd44f0f8b02b32b5bbe9d79"
7 CLIENT_SECRET = "aafa9987a8dc4b4f90e55ecf4ddc597a"
8
9 # Initialize the Spotify client
10 client_credentials_manager = SpotifyClientCredentials(client_id=CLIENT_ID, client_secret=CLIENT_SECRET)
11 sp = spotify.Spotify(client_credentials_manager=client_credentials_manager)
12
13 usage
14 def get_song_album_cover_url(song_name, artist_name):
15     search_query = f"track:{song_name} artist:{artist_name}"
16     results = sp.search(q=search_query, type="track")
17
18     if results and results["tracks"]["items"]:
19         track = results["tracks"]["items"][0]
20         album_cover_url = track["album"]["images"][0]["url"]
21         print(album_cover_url)
22         return album_cover_url
23     else:
24         return "https://i.postimg.cc/0QNxYz4V/social.png"

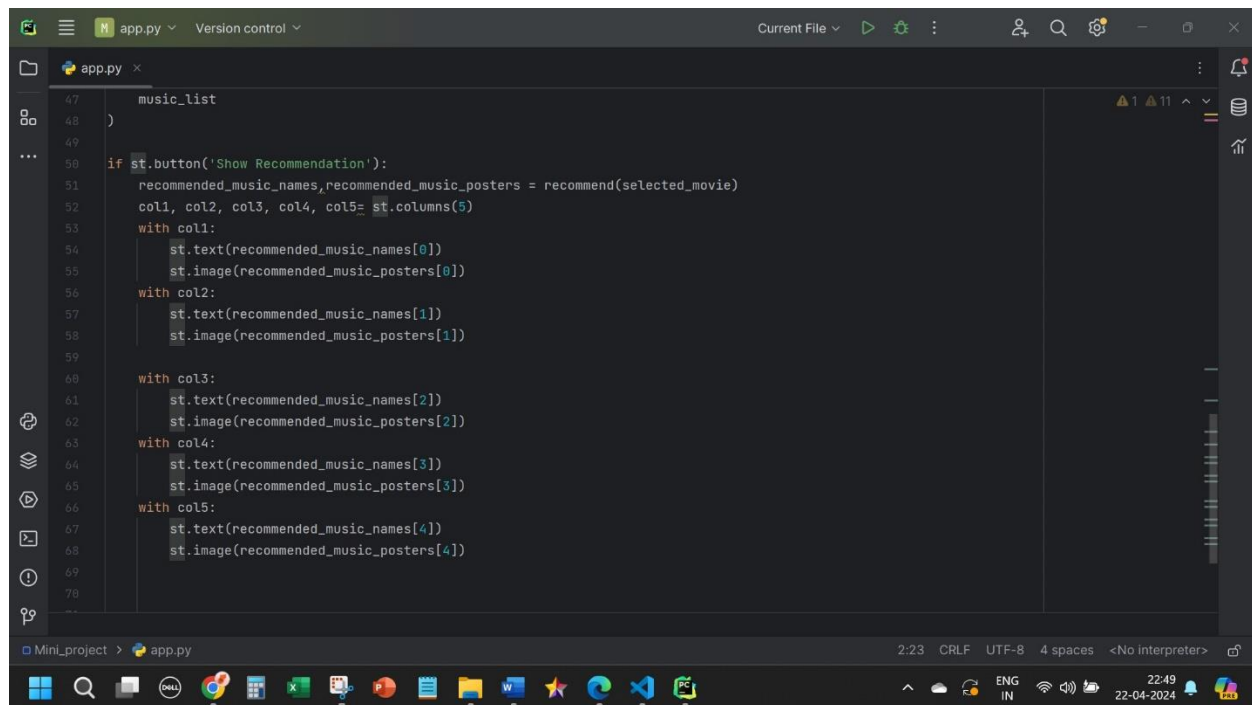
```

```

24 usage
25 def recommend(song):
26     index = music[music['song'] == song].index[0]
27     distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])
28     recommended_music_names = []
29     recommended_music_posters = []
30     for i in distances[1:6]:
31         # fetch the movie poster
32         artist = music.iloc[i[0]].artist
33         print(artist)
34         print(music.iloc[i[0]].song)
35         recommended_music_posters.append(get_song_album_cover_url(music.iloc[i[0]].song, artist))
36         recommended_music_names.append(music.iloc[i[0]].song)
37
38     return recommended_music_names, recommended_music_posters
39
40 st.header('Music Recommender System')
41 music = pickle.load(open('df.pkl', 'rb'))
42 similarity = pickle.load(open('similarity.pkl', 'rb'))
43
44 music_list = music['song'].values
45 selected_movie = st.selectbox(
46     "Type or select a song from the dropdown",

```

Music Recommendation System



The screenshot shows a code editor window with a dark theme. The file is named 'app.py' and is part of a project named 'Mini_project'. The code is written in Python and implements a music recommendation system. It features a 'Show Recommendation' button that, when clicked, calls a 'recommend' function to retrieve recommended music names and posters. These are then displayed in a grid layout with 5 columns. The code uses Tkinter for the GUI and a 'recommend' function (not shown) for the logic. The status bar at the bottom indicates the current file is 'app.py', the encoding is 'UTF-8', and the interpreter is '<No interpreter>'. The system clock shows 22:49 on 22-04-2024.

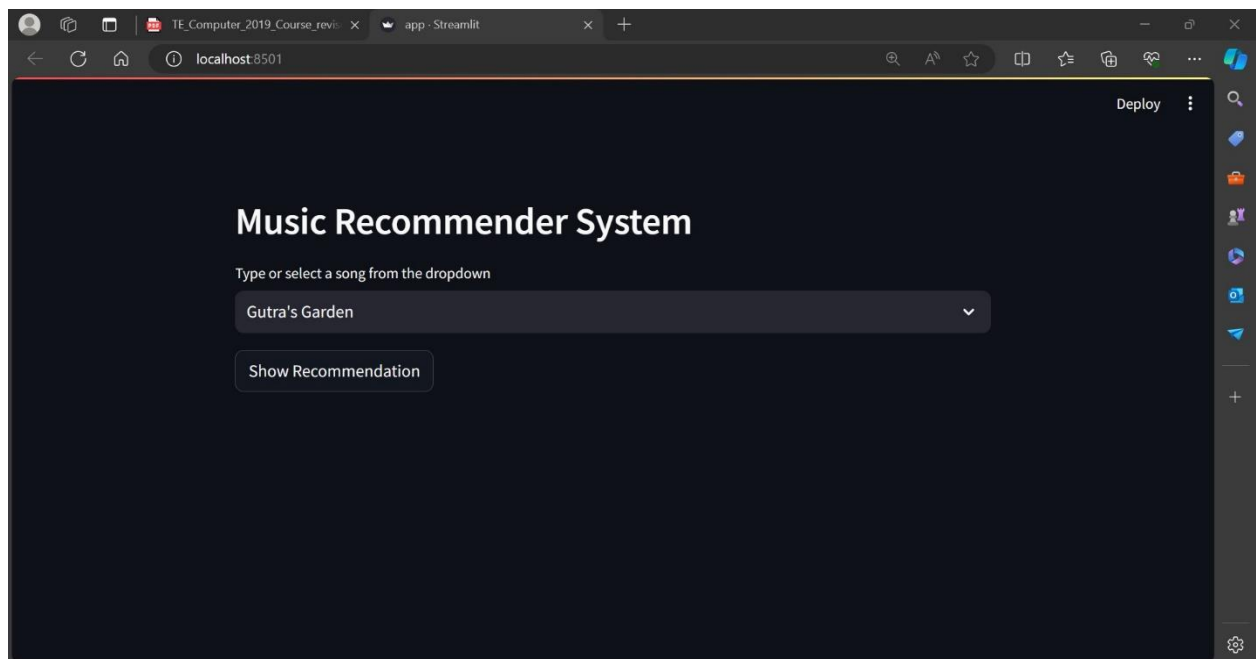
```
47     music_list
48 )
49
50 if st.button('Show Recommendation'):
51     recommended_music_names,recommended_music_posters = recommend(selected_movie)
52     col1, col2, col3, col4, col5= st.columns(5)
53     with col1:
54         st.text(recommended_music_names[0])
55         st.image(recommended_music_posters[0])
56     with col2:
57         st.text(recommended_music_names[1])
58         st.image(recommended_music_posters[1])
59
60     with col3:
61         st.text(recommended_music_names[2])
62         st.image(recommended_music_posters[2])
63     with col4:
64         st.text(recommended_music_names[3])
65         st.image(recommended_music_posters[3])
66     with col5:
67         st.text(recommended_music_names[4])
68         st.image(recommended_music_posters[4])
69
70
```

CHAPTER 7: RESULT

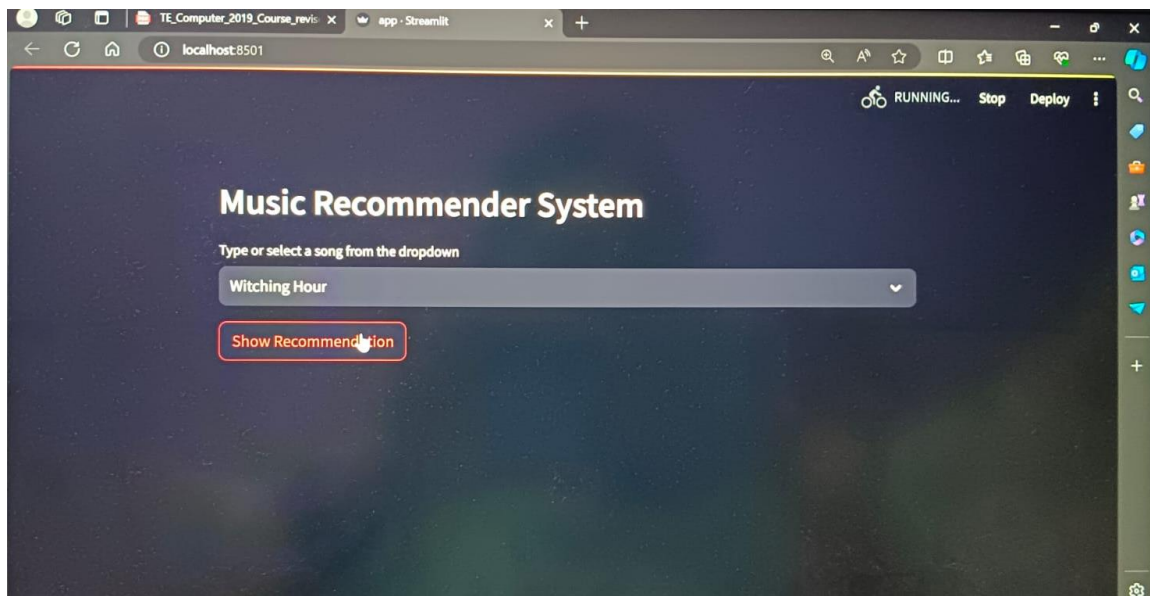
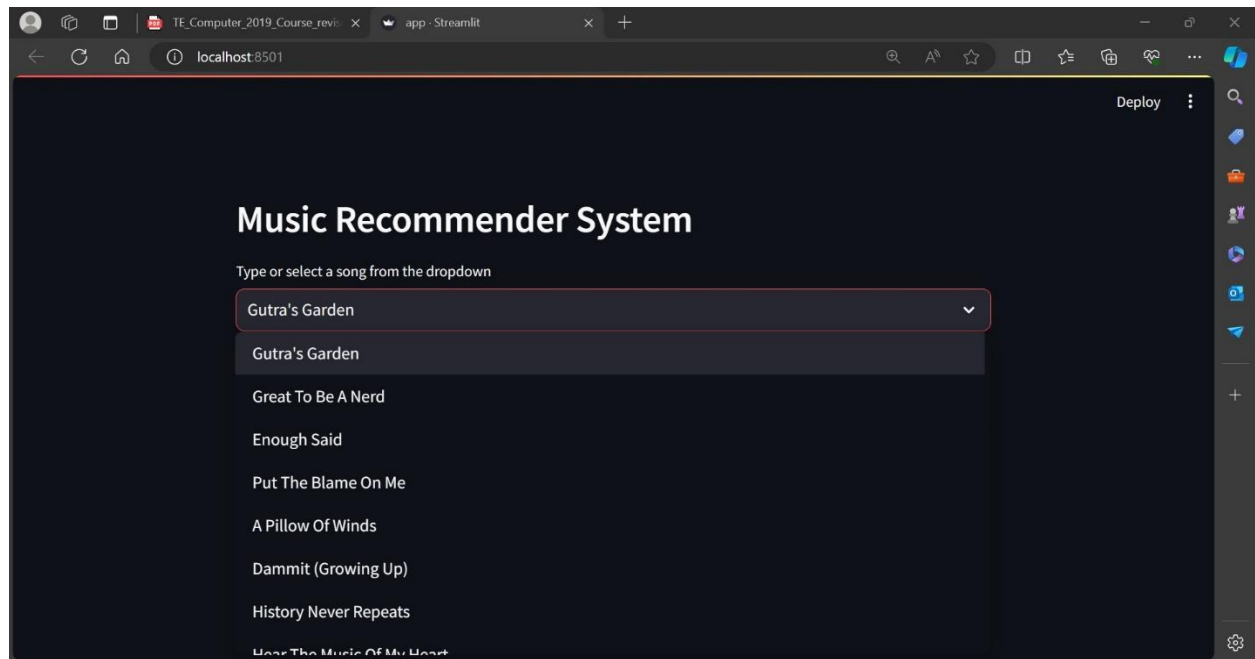
```
Terminal: Local x + v
\Mini_project> streamlit run app.py

You can now view your Streamlit app in your browser.

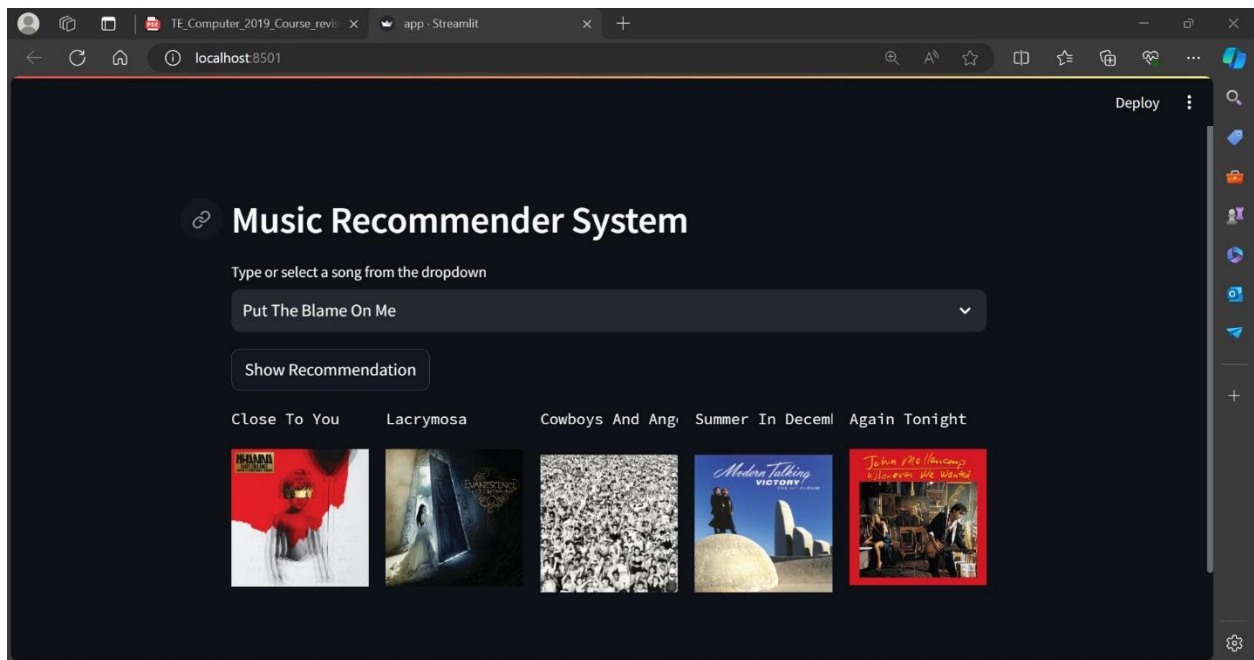
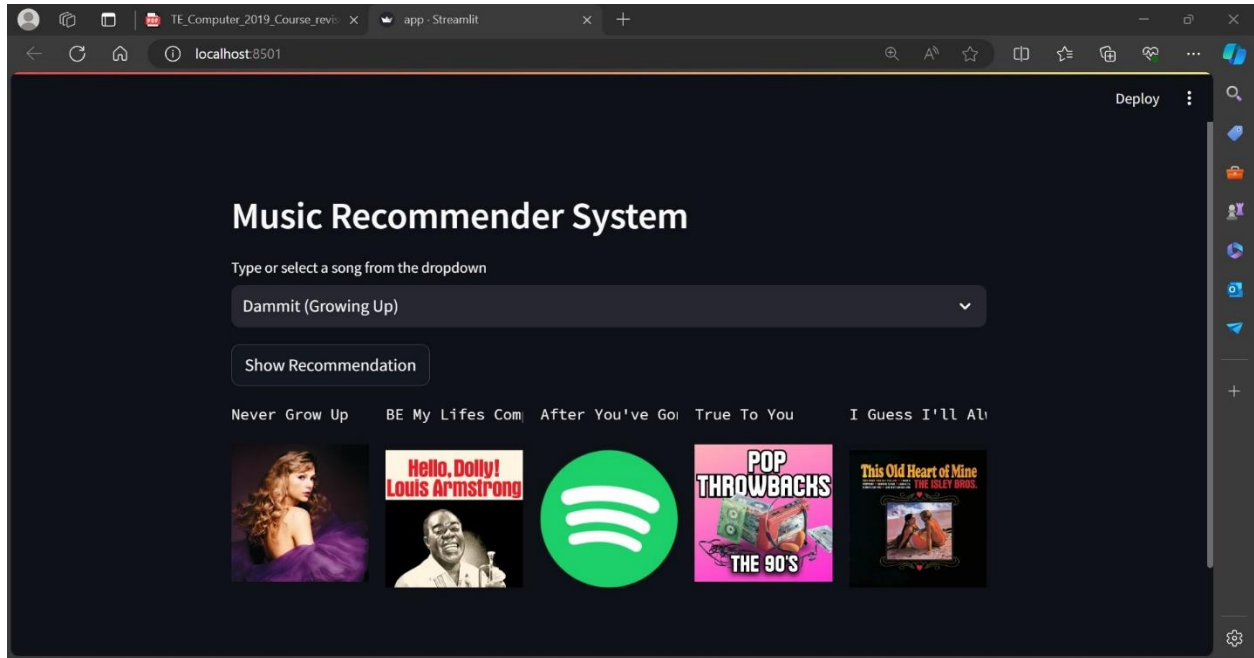
Local URL: http://localhost:8501
Network URL: http://192.168.1.7:8501
```



Music Recommendation System



Music Recommendation System



CHAPTER 8: Future Scope

The future scope of the Music Recommendation System is promising and encompasses several avenues for further development and enhancement. Some potential areas for future exploration include:

1. Integration of Additional Data Sources: While the current system relies on data sourced exclusively from Spotify, integrating data from other music streaming platforms or external sources could enrich the recommendation engine's capabilities. Incorporating data from social media platforms, music blogs, or user-generated playlists could provide supplementary insights into users' musical preferences and enhance the accuracy of recommendations.

2. Enhanced Recommendation Algorithms: Future iterations of the recommendation algorithm could incorporate advanced machine learning techniques, such as deep learning or collaborative filtering, to further improve recommendation accuracy and personalization. These algorithms could leverage additional features beyond song lyrics, such as user listening history, user demographics, or contextual information, to generate more contextually relevant recommendations.

3. Semantic Analysis and Contextual Understanding: Integrating natural language processing (NLP) techniques for semantic analysis and contextual understanding of song lyrics could enable the system to capture deeper insights into the lyrical content and thematic elements of songs. This could involve sentiment analysis, topic modeling, or emotion recognition to identify subtle nuances and themes within the lyrics, leading to more nuanced and emotionally resonant recommendations.

4. Interactive User Feedback Mechanisms: Implementing more sophisticated user feedback mechanisms, such as sentiment analysis of user comments or implicit feedback analysis based on user interactions, could enhance the system's ability to adapt and refine recommendations over time. Additionally, incorporating real-time user feedback and collaborative filtering techniques could foster a more interactive and dynamic music discovery experience.

5. Multimodal Recommendation Systems: Exploring multimodal recommendation systems that combine textual analysis of song lyrics with audio features, album artwork analysis, or user-generated content could provide a more holistic understanding of users' musical preferences. By leveraging multiple modalities of data, these systems could offer more diverse and engaging recommendations that appeal to a broader range of user preferences.

6. Personalized User Profiles and Context-aware Recommendations: Developing personalized user profiles that capture individual listening habits, preferences, and contextual factors could enable the system to deliver context-aware recommendations tailored to specific situations, moods, or listening contexts. Incorporating contextual information such as time of day, location, or social context could enhance the relevance and timeliness of recommendations, leading to a more dynamic and adaptive music discovery experience.

Overall, the future scope of the Music Recommendation System is vast and encompasses a wide range of possibilities for innovation and advancement. By embracing emerging technologies, refining recommendation algorithms, and prioritizing user-centric design principles, the system has the potential to evolve into a sophisticated and indispensable tool for music enthusiasts worldwide.

CHAPTER 9: Conclusion

In conclusion, the Music Recommendation System represents a significant advancement in the field of music discovery, leveraging cutting-edge technologies and user-centric design principles to deliver personalized and immersive music exploration experiences. Through meticulous data collection, preprocessing, and feature engineering, the system transforms raw data into actionable insights, laying the foundation for a robust recommendation engine. By harnessing cosine similarity and nearest neighbor search techniques, the system generates accurate and personalized song recommendations based on the lyrical content of songs, enriching users' music discovery journeys.

The integration of Streamlit provides users with an intuitive and interactive interface, enabling seamless navigation and exploration of recommendations. Moving forward, the future scope of the Music Recommendation System is vast, with opportunities for further development and enhancement in areas such as advanced recommendation algorithms, semantic analysis of song lyrics, and personalized user profiles. By embracing these opportunities and prioritizing innovation, the system has the potential to redefine how users engage with and discover music, fostering deeper connections and enriching experiences in the digital music landscape.

Ultimately, the Music Recommendation System stands as a testament to the transformative power of technology in enhancing human experiences, bridging the gap between users and the vast universe of musical content. As it continues to evolve and adapt to the ever-changing preferences and needs of users, the system remains poised to revolutionize the way we explore, discover, and connect through music.

CHAPTER 10: REFERENCE

<https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset/code>

<https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset/discussion>

<https://www.kaggle.com/datasets/notshrirang/spotify-million-song-dataset/suggestions?status=pending&yourSuggestions=true>

<https://www.youtube.com/watch?v=jm9JamrbSv8&t=1559s>