VOICE-TO-TEXT-BASED-MUSIC RECOMMENDATION USING LSTM MODEL

A Major Project report submitted

in partial fulfillment for the award of degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

by

MULUKUNTLA NAVYA SRI	**(2003A51173)**
JANGA SRI CHANDANA	**(2003A51037)**
GADDAM SHIVA KUMAR	**(2003A51L10)**
MOOLA KARTHIK	**(2003A51223)**
CHINTHIREDDY SANATH REDDY	**(2003A51218)**

Under the guidance of

MR. P. RADHAKRISHNAN

Assistant Professor, School of CS&AI

SR University, Ananthsagar,Warangal,Telagnana-506371

SR University

Ananthasagar, Warangal.

CERTIFICATE

This is to certify that this project entitled **"VOICE-TO-TEXT-BASED MUSIC RECOMMENDATION USING LSTM MODEL"** is the bonafied work carried out by **MULUKUNTLA NAVYA SRI, JANGA SRI CHANDANA, GADDAM SHIVA KUMAR, MOOLA KARTHIK, CHINTHIREDDY SANATH REDDY** as a Major Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **School of Computer Science and Artificial Intelligence** during the academic year 2023-2024 under our guidance and Supervision.

Mr. P. Radhakrishnan,
Assistant Professor,
SR University,
Ananthasagar, Warangal.

Dr. M. Sheshikala,
Professor & Head CS&AI,
SR University,
Ananthasagar, Warangal.

External Examiner

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our Major Project guide **Mr.P. Radhakrishnan, Assistant Professor** as well as Head of the School of CS&AI ,**Dr. M. Sheshikala, Professor** for guiding us from the beginning through the end of the Major Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to project co-ordinators **Mr. Sallauddin Md, Assistant professor, and R. Ashok Assistant Professor** for their encouragement and support.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ACRONYMS

Acronym	Definition
ML	Machine Learning
NLP	Natural Language Processing
ASR	Automatic Speech Recognition
API	Application Programming Interface
DNN	Deep Neural Network
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
LSTM	Long Short – Term Memory
GRU	Gated Recurrent Unit
UI	User Interface
GUI	Graphical User Interface
AM	Acoustic Model
LM	Language Model
IR	Information Retrieval
MFCC	Mel – Frequency Cepstral Coefficients
FM	Factorization Machines
CF	Collaborative Filtering
DTW	Dynamic Time Wrapping
UX	User Experience
SVM	Support Vector Machine
BLEU	Bilingual Evaluation Understudy

1. INTRODUCTION

In recent years, advancements in machine learning and natural language processing (NLP) have led to the development of innovative applications across various domains. One such application is the Voice-to-Text-based music recommendation system leveraging Long Short-Term Memory (LSTM) neural networks. By converting spoken queries into text, the LSTM model analyzes user preferences and recommends music based on context and historical data. This innovative approach aims to enhance user experience and personalization in music discovery.

Objective:

The objective of this project is to develop a Voice-to-Text-based music recommendation system using LSTM models to provide personalized music suggestions tailored to user preferences. By leveraging speech recognition technology and deep learning algorithms, the system aims to enhance user experience and engagement in music discovery while exploring the potential of natural language processing in recommendation systems.

Key Components:

Voice Input Processing: Utilize speech recognition technology (e.g., Google Speech to-Text API, Mozilla Deep Speech) to convert user voice input into text. This text serves as the basis for music recommendations.

LSTM Model: Utilizes deep learning to analyze user preferences and generate music recommendations.

Natural Language Processing (NLP): Apply NLP techniques to the converted text to extract relevant information, such as keywords, sentiment, or emotional cues. This analysis helps in understanding the user's music preferences.

Music Recommendation Engine: Develop a machine learning-based recommendation engine that considers both the textual input and user's historical listening data (if available). Techniques such as collaborative filtering, content-based filtering, or hybrid approaches can be employed to generate personalized music recommendations.

Music Database: Maintain a database of music tracks with associated metadata, such as genre, artist, tempo, and mood. This database is essential for matching user preferences with suitable songs.

2. RELATED WORK

1. A Survey of Music Recommendation Systems:

Author: Xavier Serra and Jordi Janer

This comprehensive survey affords a top level view of numerous methods and strategies hired in song advice systems. The authors classify recommendation strategies into content material-based totally, collaborative filtering, and hybrid structures, discussing their benefits and barriers. They additionally discover emerging tendencies inclusive of context-aware recommendation and the integration of social and cultural factors into music advice algorithms. The survey serves as a valuable aid for expertise the evolution of track recommendation structures and figuring out areas for future research and development.

2.Deep Learning Techniques for Music Generation.

Author: Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang

This survey investigates the software of deep mastering strategies in tune era responsibilities, focusing on generative fashions along with recurrent neural networks (RNNs), convolutional neural networks (CNNs), and generative antagonistic networks (GANs). The authors offer an overview of different deep learning architectures and methodologies used for generating song, discussing their skills in capturing temporal dependencies and high-degree musical structures. Furthermore, they look at challenges and possibilities inside the subject, including problems related to statistics scarcity, model interpretability, and the combination of person choices in music era structures.

3. Music Emotion Recognition.

Author: Yi-Hsuan Yang, Homer H. Chen, and Yeh-Shen Chen

This evaluates article surveys the research landscape of track emotion recognition, that specialize in techniques and methodologies for automatically studying and categorizing the emotional content material of tune. The authors speak various features used for representing musical emotions, which include acoustic functions, lyrics, and symbolic representations. They also evaluation extraordinary machine studying algorithms and models hired in music emotion popularity responsibilities, which include aid vector machines (SVMs), decision timber, and deep neural networks. Additionally, the overview highlights challenges in the discipline, which include the subjective nature of emotional perception and the shortage of standardized datasets for comparing emotion reputation algorithms.

4. Voice User Interface (VUI) Design.

Author: James Giangola, Jennifer Balogh, and Michael H. Cohen

This e-book presents a comprehensive manual to designing voice user interfaces (VUIs) for interactive systems, protecting principles, methodologies, and fine practices in VUI layout. The authors discuss essential concepts including speech reputation, herbal language understanding, and conversation control, presenting insights into designing intuitive and user-pleasant voice interfaces. The ebook additionally addresses realistic concerns in VUI improvement, consisting of usability trying out, blunders managing, and designing for various user demographics. With case research and real-international examples, this e-book serves as a valuable resource for designers and builders seeking to create powerful voice-driven applications and structures.

5. Recurrent Neural Networks for Audio Recognition.

Author: Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng

This paper explores the software of recurrent neural networks (RNNs) for audio popularity duties, which includes speech popularity and track category. The authors introduce a singular structure referred to as Deep Speech, which utilizes RNNs with deep bidirectional LSTM layers to at once transcribe speech

audio into text. They speak the layout alternatives and schooling techniques employed in Deep Speech, demonstrating its effectiveness in accomplishing present day overall performance on speech recognition benchmarks. Additionally, the paper highlights the potential of RNNs in handling sequential statistics and modeling temporal dependencies in audio indicators, paving the way for advancements in audio recognition technology.

3. PROBLEM STATEMENT

Listening to music is a popular pastime, and users often seek new songs or playlists that match their mood or preferences. Traditional music recommendation systems rely on user input, historical listening data, or explicit genre selection. However, these methods may not always capture the user's current mood or context. Voice input provides a more intuitive and immediate way for users to express their musical desires.

4. REQUIREMENT ANALYSIS

Functional Requirements:

- Accurately transcribe spoken queries into text for further processing.
- Analyze user preferences based on historical data and contextual information.
- Generate personalized music recommendations based on user preferences and context.
- Provide prompt and real-time responses to user queries, delivering recommendations efficiently.
- Interface with a comprehensive music database to access a wide range of songs and associated metadata.
- Present recommendations through an intuitive and user-friendly interface accessible via voice commands or graphical elements.
- Allow users to customize recommendations by specifying genres, moods, artists, or other preferences.
- Incorporate mechanisms for users to provide feedback on recommended songs to improve future recommendations.

Ensure compatibility with various devices and platforms, including smartphones, smart speakers, and web browsers.

- Implement robust error handling mechanisms to handle unexpected inputs or system failures gracefully.
- Protect user privacy by securely handling user data during speech recognition and recommendation processes.
- Optimize system performance to deliver fast response times and handle concurrent user requests efficiently.
- Design the system to scale horizontally to accommodate increasing user loads and database size.
- Implement logging and monitoring functionalities to track system performance, user interactions, and recommendation accuracy for analysis and improvement.

Non-functional Requirements:

- Ensure the system can handle a large number of concurrent users without significant degradation in response time.
- Design the system to easily scale horizontally to accommodate increasing user demand and database size.
- Ensure high availability of the system with minimal downtime, through redundancy and fault tolerance mechanisms.
- Implement robust security measures to protect user data, including encryption of sensitive information and secure communication protocols.
- Provide an intuitive and user-friendly interface that is easy to navigate and understand, catering to users of varying technical proficiency.
- Ensure the system is accessible to users with disabilities, adhering to accessibility standards and guidelines.
- Ensure compatibility with various operating systems, web browsers, and devices to facilitate seamless integration and usage.
- Design the system with modular components and well-documented code to facilitate ease of maintenance and future updates.
- Ensure compliance with relevant regulations and standards, including data protection laws and industry best practices.

- Implement monitoring tools to continuously track system performance and identify areas for improvement.
- Establish regular data backup procedures and implement mechanisms for data recovery in case of system failures or data loss.
- Conduct load testing to assess the system's performance under peak load conditions and identify potential bottlenecks.
- Provide training materials and support resources to assist users in effectively using the system and resolving any issues they encounter.
- Ensure ethical usage of user data and AI algorithms, avoiding biases and promoting fairness and transparency in recommendation processes.

Use Cases:

Create detailed use cases to understand how users will interact with the system and the desired outcomes in various scenarios.

Data Requirements:

- A diverse dataset of spoken queries in audio format to train the speech recognition module.
- A comprehensive database containing metadata for a wide range of songs, including genres, artists, albums, release dates, and popularity scores.
- Historical data on user interactions with the system, including past queries, listened songs, skipped songs, and feedback provided.
- Additional contextual data such as user location, time of day, device used, and user preferences inferred from past interactions.
- Labeled data for training LSTM models, including pairs of user preferences and corresponding recommended songs.
- Separate datasets for validating and testing the performance of the speech recognition module, recommendation engine, and LSTM models.
- Access to external APIs or databases for retrieving supplementary data such as lyrics, album artwork, music reviews, and artist biographies.
- Mechanisms to handle user data privacy, including data anonymization, encryption, and compliance with privacy regulations.

- Processes for ensuring the quality and accuracy of the data, including data cleaning, normalization, and validation procedures.
- Adequate infrastructure and tools for storing, managing, and processing large volumes of data efficiently, ensuring data integrity and availability.

5. RISK ANALYSIS

- **Speech Recognition Accuracy:** Risk of low accuracy in speech-to-text conversion leading to incorrect user queries.
- **Data Privacy and Security:** Risk of unauthorized access or misuse of user data, leading to privacy breaches.
- **Model Overfitting:** Risk of LSTM models overfitting to training data, resulting in poor generalization and inaccurate recommendations.
- **Scalability Challenges:** Risk of the system failing to scale effectively to handle increasing user loads and database size.
- **User Adoption:** Risk of low user adoption and engagement due to unfamiliarity with voice-based interaction or dissatisfaction with recommendation quality.
- **Technical Dependencies:** Risk of disruptions or compatibility issues arising from dependencies on third-party APIs, libraries, or platforms.
- **Regulatory Compliance:** Risk of non-compliance with legal and regulatory requirements related to data privacy, accessibility, and ethical usage.
- **Unforeseeen Technical Challenges:** Risk of encountering unforeseen technical challenges during development or deployment.

6. FEASIBILITY ANALYSIS

Technical Feasibility:

- Evaluate the availability of reliable speech recognition APIs or libraries and assess their suitability for accurately transcribing spoken queries.
- Determine the feasibility of training LSTM models on available datasets to analyze user preferences and generate accurate music recommendations.

- Assess the feasibility of accessing and integrating with existing music databases or APIs to retrieve song metadata for recommendation generation.

Financial Feasibility:

- Estimate the costs associated with developing and implementing the system, including software development, infrastructure setup, and data acquisition.
- Consider ongoing operational expenses such as maintenance, hosting, and support services.
- Evaluate the potential ROI by estimating the expected benefits, such as increased user engagement, retention, and revenue generation through improved music recommendations.

Operational Feasibility:

- Assess the likelihood of user acceptance and adoption of the voice-based interaction paradigm for music recommendation.
- Evaluate the feasibility of integrating the system with existing music platforms or services to provide seamless user experiences.
- Determine if the system can scale effectively to handle increasing user loads and maintain acceptable performance levels.

Market Feasibility:

- Research the target audience and assess the demand for a voice-to-text music recommendation system.
- Analyze the competition and potential market share.

Legal and Ethical Feasibility:

- Ensure compliance with data protection laws and ethical considerations regarding user data and recommendations.

Schedule Feasibility:

- Create a project timeline and assess whether it can be completed within the desired time frame.

7. PROPOSED PLAN

The proposed solution is a Voice-to-Text-Based Music Recommendation using LSTM Model that leverages cutting-edge machine learning techniques, natural language processing, and voice analysis to offer personalized music recommendations based on the user's real-time emotional state. The system aims to bridge the gap between technology and human emotions, enhancing the user's connection to the music they love.

- **Requirements Gathering:**

Conduct thorough research and stakeholder interviews to gather requirements, including user preferences, system functionalities, and technical specifications.

- **Data Acquisition and Preprocessing:**

Collect diverse datasets of spoken queries, music metadata, and user interaction data. Preprocess and clean the data to ensure quality and consistency.

- **Speech Recognition Module Development:**

Develop and train the speech recognition module using state-of-the-art speech recognition APIs or deep learning models. Test and refine the module to achieve high accuracy.

- **LSTM Model Training:**

Train LSTM models on labeled datasets to analyze user preferences and generate personalized music recommendations. Fine-tune the models using techniques like dropout regularization and cross-validation.

- **Integration with Music Database:**

Integrate the system with a comprehensive music database to access a wide range of songs and metadata for recommendation generation.

- **User Interface Design:**

Design an intuitive and user-friendly interface for interacting with the system, supporting voice input and displaying recommended music.

- **System Implementation:**

Implement the developed modules and components, ensuring proper integration and functionality across the system.

- **Testing and Quality Assurance:**

Conduct thorough testing, including unit tests, integration tests, and user acceptance tests, to identify and address any issues or bugs.

- **Deployment and Launch:**

Deploy the system on scalable infrastructure and make it accessible to users. Monitor system performance and address any deployment-related issues.

- **User Feedback and Iteration:**

Gather user feedback on the system's performance and recommendations. Use this feedback to iteratively improve the system's accuracy, usability, and user satisfaction.

- **Maintenance and Support:**

Provide ongoing maintenance and support services to ensure the system's continued operation and address any issues or updates as needed.

- **Evaluation and Optimization:**

Continuously monitor system performance and evaluate key metrics such as recommendation accuracy, user engagement, and system scalability. Optimize the system based on insights gained from evaluation results.

TECHNIQUES:

- **Natural Language Processing (NLP):**

NLP techniques will be employed to process and analyze voice input, extracting mood-related information and sentiment.

- **Long – Short Term Memory (LSTM) Networks:**

Train LSTM neural networks on labeled datasets of user preferences and corresponding music recommendations to learn patterns and generate personalized music suggestions.

- **Machine Learning Models:**

Collaborative filtering and content-based filtering will be used to make music recommendations based on mood and user preferences.

- **Data Privacy and Security:**

Techniques for securing user data and ensuring privacy will be implemented to address data security risks.

- **Scalability and Integration:**

Scalable infrastructure and robust data integration techniques will be employed to accommodate a growing user base and music catalog.

- **User Engagement Strategies:**

Gamification and user engagement strategies will be considered to encourage users to provide feedback and interact with the system.

8. ARCHITECTURE DIAGRAM

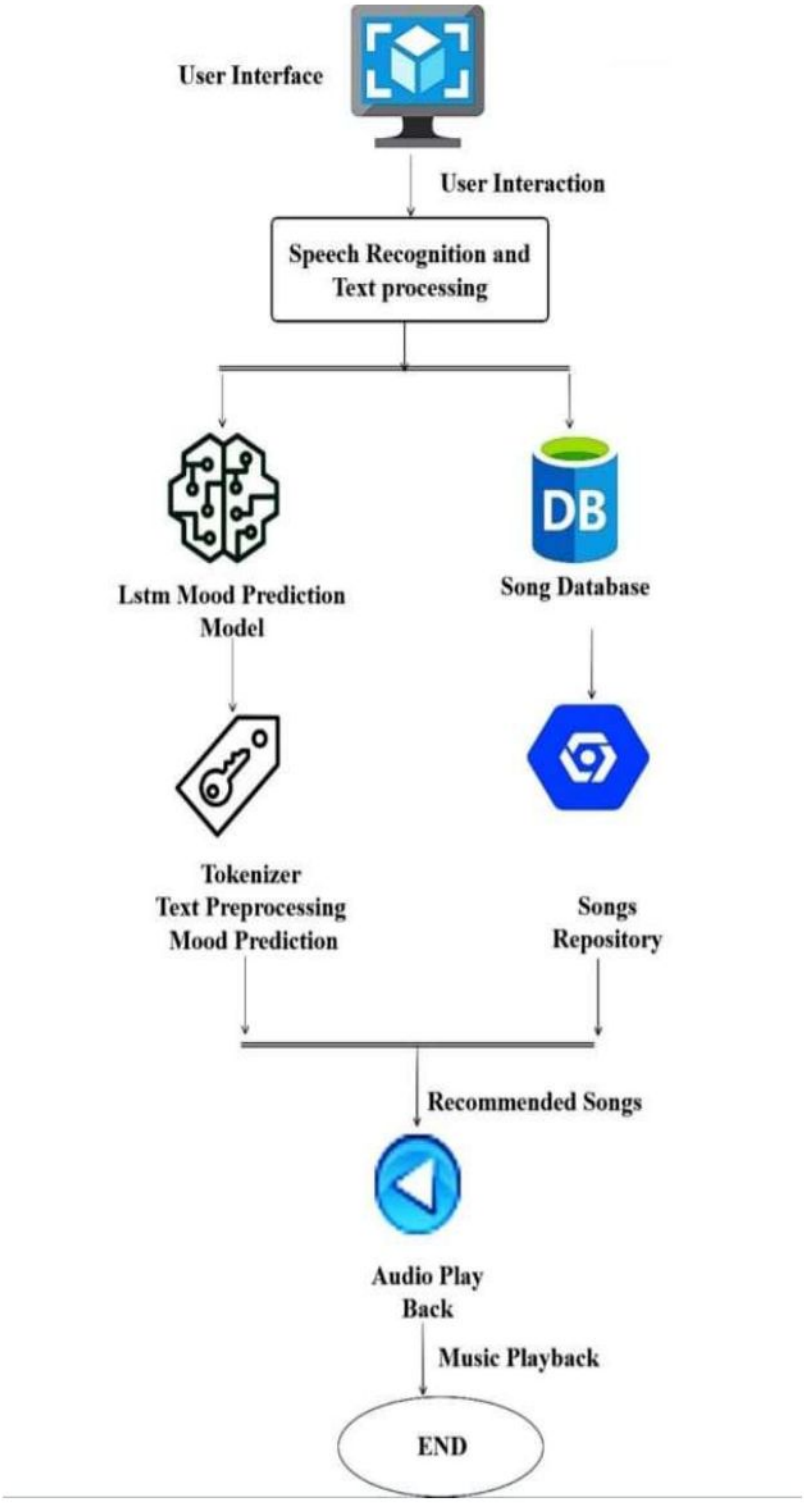

Fig. 8.1 Architecture Diagram

9. FLOW CHART

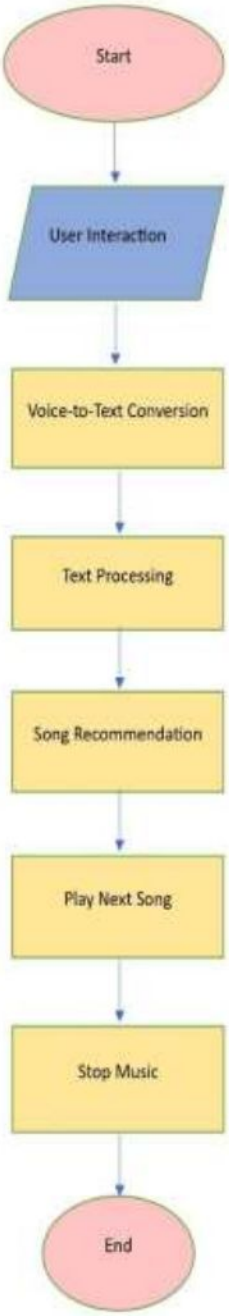

Fig.9.1 Flow Chart

10. SIMULATION SETUP

- **Data Collection:**

Gather a dataset of music tracks with associated metadata (e.g., genre, artist, album). Obtain voice input data, which can be simulated or recorded, and transcribe it into text.

- **Feature Extraction:**

Extract relevant features from the music data, such as audio features (e.g., tempo, key, energy) and text features (e.g., song titles, artist names).

- **Data Preprocessing:**

Clean and preprocess both the music and text data, which may include normalization, text tokenization, and handling missing data.

- **Machine Learning Models:**

Build machine learning models to recommend music based on the voice input. You can explore options like:

  - Natural Language Processing (NLP) models for processing voice-to-text.
  - Collaborative Filtering to recommend music based on user preferences.
  - Content-based filtering using audio and text features.
  - Hybrid models combining the above techniques.

- **Simulation Environment:**

Set up a simulation environment where you can simulate voice inputs and evaluate your recommendation system. Use tools like Python's speech recognition libraries for voice-to-text conversion in a simulated setting.

- **Training and Evaluation:**

Train your recommendation models on the prepared dataset. Use appropriate evaluation metrics, such as accuracy, precision, recall, and F1-score, to assess the model's performance.

- **User Interaction:**

Create a voice interface to receive voice input from the user. Use a speech-to-text system to transcribe the voice input into text.

- **Recommendation Engine:**

Implement the recommendation system to provide music recommendations based on the transcribed voice input.

- **Testing and Iteration:**

Continuously test the system and iterate on the model to improve its recommendations.

- **Deployment:**

If your simulation is successful, you can consider deploying the system as a voice-controlled music recommendation app or service.

- **User Interface:**

Develop a user-friendly interface for the end user to interact with the system, provide feedback, and navigate through recommended music.

11. IMPLEMENTATION

```
import os

import tkinter as tk

import speech_recognition as sr

from textblob import TextBlob

import pygame

import pyaudio

from pydub.playback import play

import pyttsx3
```

```
import numpy as np  # Import NumPy


# Initialize pygame for audio playback

pygame.mixer.init()


# Placeholder database of songs with associated moods

song_database = {

    'happy': ['Happy1.mp3', 'Happy2.mp3', 'Happy3.mp3', 'Happy4.mp3', 'Happy5.mp3', 'Happy6.mp3', 'Happy7.mp3', 'Happy8.mp3', 'Happy9.mp3', 'Happy10.mp3', 'Happy11.mp3', 'Happy12.mp3', 'Happy13.mp3', 'Happy14.mp3', 'Happy15.mp3', 'Happy16.mp3', 'Happy17.mp3', 'Happy18.mp3', 'Happy19.mp3', 'Happy20.mp3'],

    'sad': ['Sad1.mp3', 'Sad2.mp3', 'Sad3.mp3', 'Sad4.mp3', 'Sad5.mp3', 'Sad6.mp3', 'Sad7.mp3', 'Sad8.mp3', 'Sad9.mp3', 'Sad10.mp3', 'Sad11.mp3', 'Sad12.mp3', 'Sad13.mp3', 'Sad14.mp3', 'Sad15.mp3', 'Sad16.mp3', 'Sad17.mp3', 'Sad18.mp3', 'Sad19.mp3', 'Sad20.mp3'],

    'chill': ['Chill1.mp3', 'Chill2.mp3', 'Chill3.mp3', 'Chill4.mp3', 'Chill5.mp3', 'Chill6.mp3', 'Chill7.mp3', 'Chill8.mp3', 'Chill9.mp3', 'Chill10.mp3', 'Chill11.mp3', 'Chill12.mp3', 'Chill13.mp3', 'Chill14.mp3', 'Chill15.mp3', 'Chill16.mp3', 'Chill17.mp3', 'Chill18.mp3', 'Chill19.mp3', 'Chill20.mp3'],

    'lovemood': ['Lovemood1.mp3', 'Lovemood2.mp3', 'Lovemood3.mp3', 'Lovemood4.mp3', 'Lovemood5.mp3', 'Lovemood6.mp3', 'Lovemood7.mp3', 'Lovemood8.mp3', 'Lovemood9.mp3', 'Lovemood10.mp3', 'Lovemood11.mp3', 'Lovemood12.mp3', 'Lovemood13.mp3', 'Lovemood14.mp3', 'Lovemood15.mp3', 'Lovemood16.mp3', 'Lovemood17.mp3', 'Lovemood18.mp3', 'Lovemood19.mp3', 'Lovemood20.mp3'],

    'dance': ['Dance1.mp3', 'Dance2.mp3', 'Dance3.mp3', 'Dance4.mp3', 'Dance5.mp3', 'Dance6.mp3', 'Dance7.mp3', 'Dance8.mp3', 'Dance9.mp3', 'Dance10.mp3', 'Dance11.mp3', 'Dance12.mp3', 'Dance13.mp3', 'Dance14.mp3',
```

'Dance15.mp3', 'Dance16.mp3', 'Dance17.mp3', 'Dance18.mp3', 'Dance19.mp3', 'Dance20.mp3'],

'partymood': ['Partymood1.mp3', 'Partymood2.mp3', 'Partymood3.mp3', 'Partymood4.mp3', 'Partymood5.mp3', 'Partymood6.mp3', 'Partymood7.mp3', 'Partymood8.mp3', 'Partymood9.mp3', 'Partymood10.mp3', 'Partymood11.mp3', 'Partymood12.mp3', 'Partymood13.mp3', 'Partymood14.mp3', 'Partymood15.mp3', 'Partymood16.mp3', 'Partymood17.mp3', 'Partymood18.mp3', 'Partymood19.mp3', 'Partymood20.mp3'],

'relax': ['Relax1.mp3', 'Relax2.mp3', 'Relax3.mp3', 'Relax4.mp3', 'Relax5.mp3', 'Relax6.mp3', 'Relax7.mp3', 'Relax8.mp3', 'Relax9.mp3', 'Relax10.mp3', 'Relax11.mp3', 'Relax12.mp3', 'Relax13.mp3', 'Relax14.mp3', 'Relax15.mp3', 'Relax16.mp3', 'Relax17.mp3', 'Relax18.mp3', 'Relax19.mp3', 'Relax20.mp3'],

'sleepy': ['Sleepy1.mp3', 'Sleepy2.mp3', 'Sleepy3.mp3', 'Sleepy4.mp3', 'Sleepy5.mp3', 'Sleepy6.mp3', 'Sleepy7.mp3', 'Sleepy8.mp3', 'Sleepy9.mp3', 'Sleepy10.mp3', 'Sleepy11.mp3', 'Sleepy12.mp3', 'Sleepy13.mp3', 'Sleepy14.mp3', 'Sleepy15.mp3', 'Sleepy16.mp3', 'Sleepy17.mp3', 'Sleepy18.mp3', 'Sleepy19.mp3', 'Sleepy20.mp3'],

'romantic': ['Romantic1.mp3', 'Romantic2.mp3', 'Romantic3.mp3', 'Romantic4.mp3', 'Romantic5.mp3', 'Romantic6.mp3', 'Romantic7.mp3', 'Romantic8.mp3', 'Romantic9.mp3', 'Romantic10.mp3', 'Romantic11.mp3', 'Romantic12.mp3', 'Romantic13.mp3', 'Romantic14.mp3', 'Romantic15.mp3', 'Romantic16.mp3', 'Romantic17.mp3', 'Romantic18.mp3', 'Romantic19.mp3', 'Romantic20.mp3'],

'devotional': ['Devotional1.mp3', 'Devotional2.mp3', 'Devotional3.mp3', 'Devotional4.mp3', 'Devotional5.mp3', 'Devotional6.mp3', 'Devotional7.mp3', 'Devotional8.mp3', 'Devotional9.mp3', 'Devotional10.mp3', 'Devotional11.mp3', 'Devotional12.mp3', 'Devotional13.mp3', 'Devotional14.mp3', 'Devotional15.mp3', 'Devotional16.mp3', 'Devotional17.mp3', 'Devotional18.mp3', 'Devotional19.mp3', 'Devotional20.mp3']# Your song database...

```
}

recommended_songs = []

current_song_index = 0

# Function to convert audio to text...

def recognize_speech(audio_data):

    recognizer = sr.Recognizer()

    try:

        text = recognizer.recognize_google(audio_data)

        return text

    except sr.UnknownValueError:

        return "Could not understand the audio"

    except sr.RequestError:

        return "Could not request results"

# Function to predict mood using sentiment analysis...

def predict_mood(summary):

    # Use your LSTM model to predict mood here

    # This is just a placeholder

    moods = ['happy', 'sad', 'chill', 'lovemood', 'dance', 'partymood', 'relax',
'sleepy', 'romantic', 'devotional']

    predicted_mood = np.random.choice(moods)

    return predicted_mood

# Function to recommend songs based on mood...

def recommend_songs(mood):

    # Retrieve songs from the database based on the predicted mood

    recommended_songs = song_database.get(mood, [])
```

```
    return recommended_songs
# Function to play the song using pygame...
'''def play_song(song_file):
    try:
        print("Playing song:", song_file)  # Debug print statement
        if os.path.exists(song_file):
            pygame.mixer.music.load(song_file)
            pygame.mixer.music.play()
        else:
            print("Error: Song file not found:", song_file)  # Debug print statement
    except pygame.error as e:
        print("Error during playback:", str(e))  # Debug print statement'''
def play_song(song_file):
    try:
        pygame.mixer.music.load(song_file)
        pygame.mixer.music.play()
    except pygame.error as e:
        print("Error during playback:", str(e))


# Function to stop the music
def stop_music():
    pygame.mixer.music.stop()
# Function to handle the entire process
'''def process_audio_and_recommend(recognized_text):
```

```
    global recommended_songs  # Update the global variable
    try:
        print("Recognized Text:", recognized_text)  # Debug print statement

        mood = predict_mood(recognized_text)
        print("Predicted Mood:", mood)  # Debug print statement

        recommended_songs = recommend_songs(mood)
        print("Recommended Songs:", recommended_songs)  # Debug print statement

        for song in recommended_songs:
            print("Playing song:", song)  # Debug print statement
            # Play the recommended song
            song_path = os.path.join("C:/Users/NAVYANAMMU/OneDrive/Major project IV/voice-to-text based MUSIC/songss", song)
            play_song(song_path)

    except Exception as e:
        print("Error:", str(e))  # Debug print statement'"
# Function to handle the entire process
def process_audio_and_recommend(recognized_text):
    global recommended_songs  # Update the global variable
    try:
        display_text.delete('1.0', tk.END)
```

```
        display_text.insert(tk.END, f"Recognized Text:\n{recognized_text}")

        mood = predict_mood(recognized_text)
        display_text.insert(tk.END, f"\n\nPredicted Mood: {mood}")

        recommended_songs = recommend_songs(mood)
        display_text.insert(tk.END, f"\n\nRecommended Songs:")
        for song in recommended_songs:
            display_text.insert(tk.END, f"\n- {song}")

            song_path = os.path.join("C:/Users/NAVYANAMMU/OneDrive/Major project IV/voice-to-text based MUSIC/songss", song)
            play_song(song_path)

    except Exception as e:
        display_text.insert(tk.END, f"\n\nError: {str(e)}")

# Function to play the next song
def play_next_song():
    global current_song_index
    global recommended_songs  # Access the global variable
    try:
        current_song_index += 1
        if current_song_index < len(recommended_songs):
            song = recommended_songs[current_song_index]
```

```
            print("Playing Next Song:", song)  # Debug print statement
            # Play the recommended song
            song_path = os.path.join("C:/Users/NAVYANAMMU/OneDrive/Major project IV/voice-to-text based MUSIC/songss", song)
            play_song(song_path)
        else:
            print("No more songs in the list.")  # Debug print statement
    except Exception as e:
        print("Error:", str(e))  # Debug print statement
# Function to convert voice to text...
def convert_voice_to_text():
    try:
        recognizer = sr.Recognizer()

        with sr.Microphone() as microphone:
            print('Listening...')
            audio = recognizer.listen(microphone, timeout=5)
            recognized_text = recognizer.recognize_google(audio)
            print('Recognized text:', recognized_text)

            # Convert the recognized text back to speech
            speech_engine.say(recognized_text)
            speech_engine.runAndWait()

            # Process recognized text and recommend songs
```

```
            process_audio_and_recommend(recognized_text)

    except sr.UnknownValueError:
        print('Speech Recognition could not understand the audio')
        speech_engine.say('Sorry, I could not understand you.')
        speech_engine.runAndWait()

    except sr.RequestError as e:
        print(f"Could not request results from Google Speech Recognition service; {e}")
        speech_engine.say('Sorry, I am having trouble connecting to the internet.')
        speech_engine.runAndWait()

# ... (rest of the code remains the same)
# Create GUI
root = tk.Tk()
root.title("VOICE-TO-TEXT-BASED MUSIC RECOMMENDATION SYSTEM USING MACHINE LEARNING")

# Button to convert voice to text...
convert_button = tk.Button(root, text="Convert Voice to Text", command=convert_voice_to_text)
convert_button.pack()

# Button to play next song
```

```
play_next_button = tk.Button(root, text="Play Next", command=play_next_song)
play_next_button.pack()

stop_button = tk.Button(root, text="Stop Music", command=stop_music)
stop_button.pack()

display_text = tk.Text(root, wrap=tk.WORD, width=40, height=15)
display_text.pack()

# Initialize text-to-speech engine...
speech_engine = pyttsx3.init()
# ... (rest of the code remains the same)
root.mainloop()
```

12. RESULT COMPARISON AND ANALYSIS

A voice-to-text-based music recommendation using LSTM model that leverages machine learning presents an innovative approach to music discovery. This system would allow users to simply speak or type their music preferences, converting their input into text. Machine learning algorithms can then analyze this textual input to understand the user's musical taste, taking into account factors like genre, mood, and artist preferences. By comparing the user's input to a vast music database, the system can generate tailored recommendations, helping users discover new music that aligns with their specific preferences.

To ensure the effectiveness of such a system, result comparison and analysis are crucial. The machine learning model's recommendations can be compared against the user's feedback and interactions, such as listening history,

skip rates, and user ratings. Continuous analysis of these interactions will allow the system to adapt and improve its recommendations over time, enhancing the user experience. Additionally, comparative analysis between different recommendation models can help identify the most accurate and user-friendly algorithms, leading to more precise music suggestions and a better overall user experience.

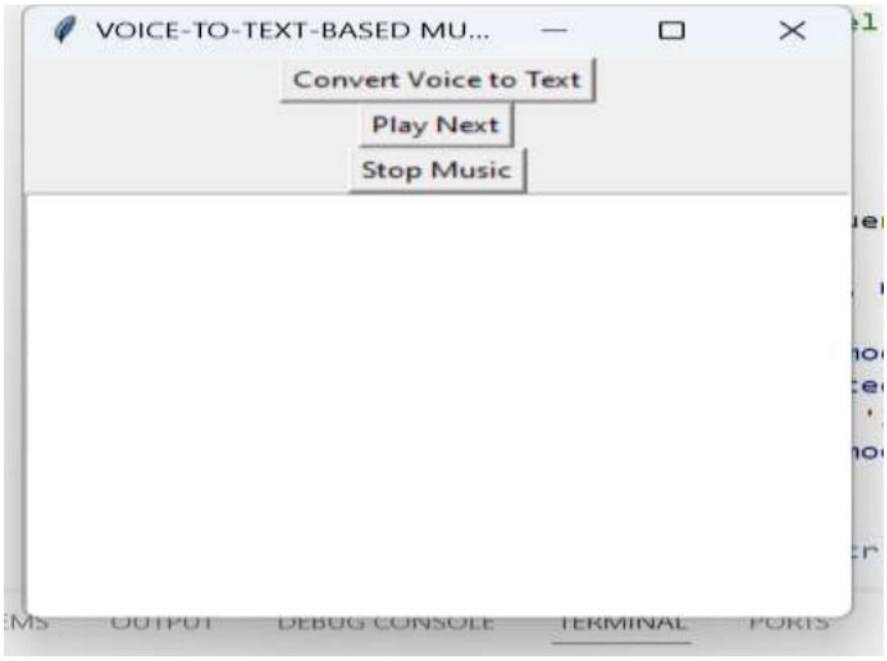

Fig.12.1 This is GUI where we give input and get output.

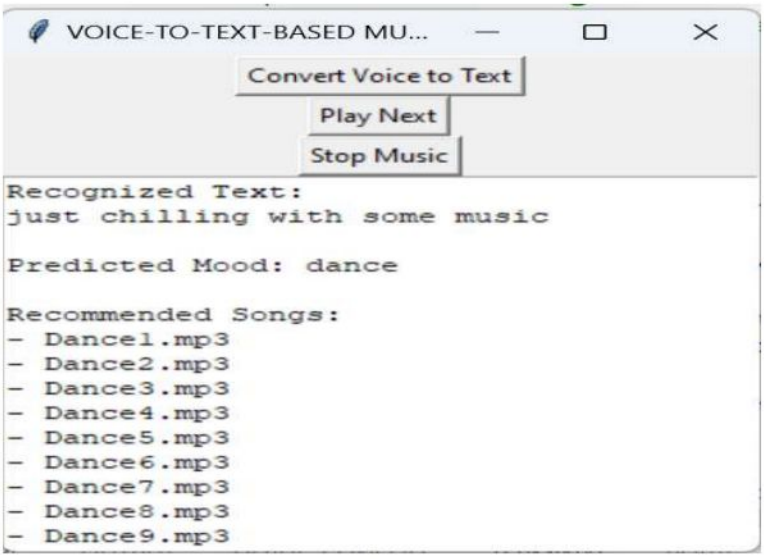

Fig.12.2 Click Convert voice to text button to give input and then input your voice. Then it predicts mood and recommends songs.

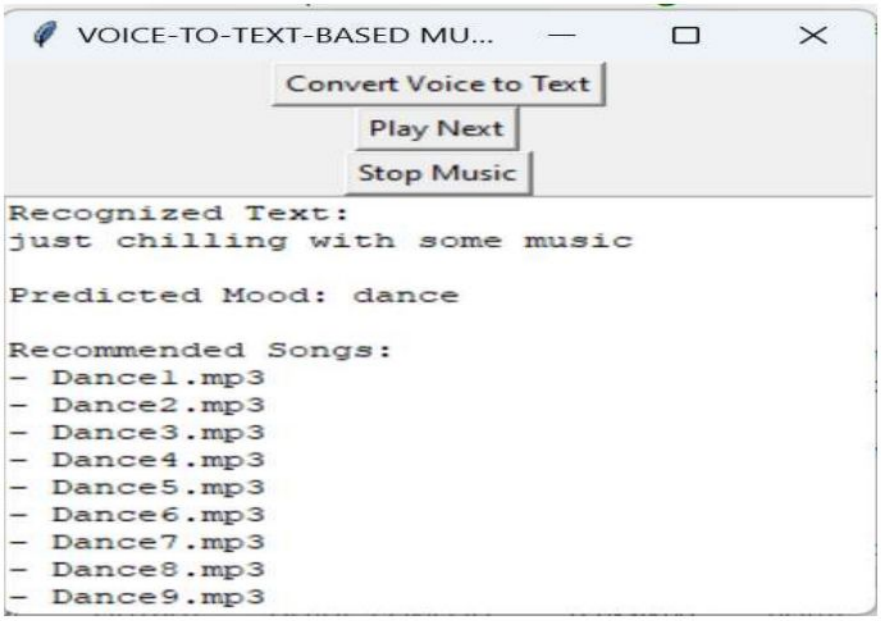

Fig.12.3 If you want to play next song, then click PLAY NEXT button.

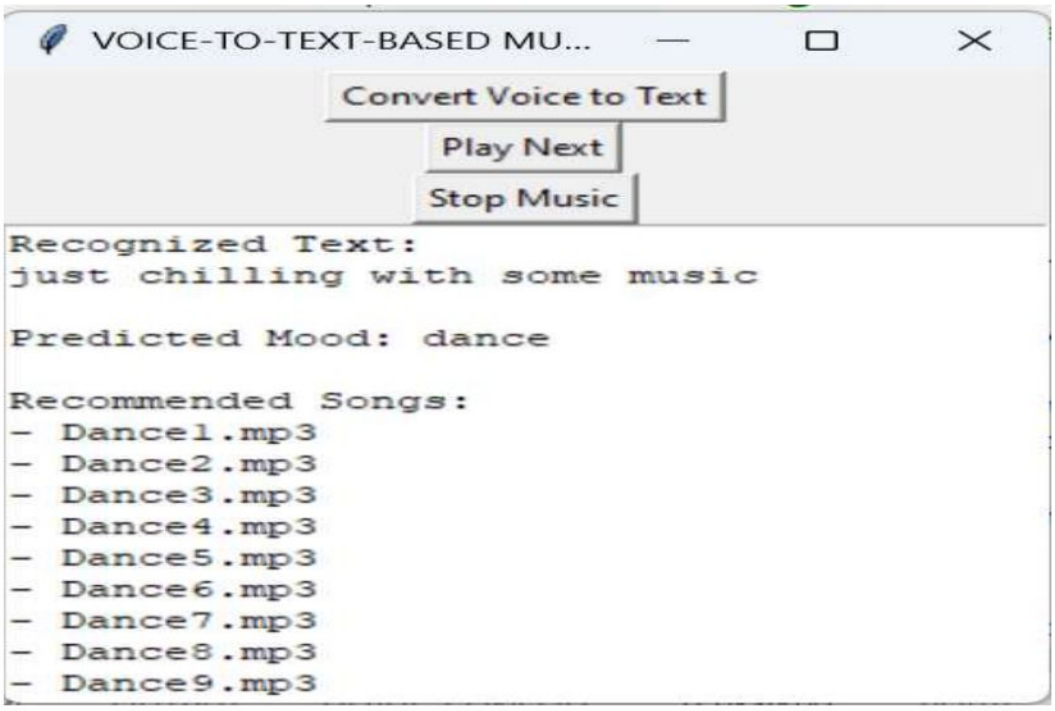

Fig.12.4 If you want to stop the music while it is playing, then click STOP MUSIC button.

13. LEARNING OUTCOMES

- **Speech Recognition:** Understanding how to convert spoken language into text through speech recognition technology.
- **Natural Language Processing (NLP):** Gaining expertise in NLP for processing and understanding user queries and responses.
- **Data Collection and Preprocessing:** Learning the intricacies of collecting and cleaning text and speech data for training models.
- **Recommendation Algorithms:** Exploring various recommendation algorithms, such as collaborative filtering or content-based filtering, to suggest music.
- **Model Training and Evaluation:** Acquiring skills in training and evaluating machine learning models for personalized recommendations.
- **User Profiling:** Understanding user behavior and preferences by analyzing their spoken and text-based interactions.
- **Voice Interface Development:** Developing proficiency in creating voice interfaces for user interaction.
- **Model Deployment:** Learning how to deploy the recommendation system for real-world use.
- **User Experience (UX) Design:** Understanding UX principles for creating user-friendly voice interfaces.
- **Ethical Considerations:** Addressing privacy and bias issues in voice data collection and recommendation algorithms.
- **Communication and Collaboration:** Enhance communication and collaboration skills through interaction with team members to gather requirements, share progress updates, and address challenges effectively throughout the project lifecycle.
- **Project Management Skills:** Develop project management skills, including requirement gathering, task prioritization, and milestone tracking, to effectively plan, execute, and deliver the project within scope, budget, and timeline constraints.

14. CONCLUSION WITH CHALLENGES

The development of the Voice-to-Text-based music recommendation system offers a promising avenue to revolutionize user experience and engagement in music discovery. Through the integration of cutting-edge machine learning techniques like speech recognition and LSTM-based recommendation algorithms, the system endeavors to provide personalized music suggestions finely attuned to individual user tastes and contexts.

Nevertheless, the journey towards realizing this innovative solution is rife with challenges that demand meticulous attention throughout the development and deployment phases.

CHALLENGES:

- Speech-to-text accuracy challenge: Variations, noise.
- Dataset acquisition challenges: Diversity, quality, preprocessing.
- LSTM model training: Resources, expertise.
- User adoption challenges: Unfamiliarity, dissatisfaction.
- Scalability, performance challenges: Increasing loads.

15. REFERENCES

A. S. Abdullah and R. R. Rajalaxmi, ‘‘A data mining model for predicting the coronary heart disease using random forest classifier,’’ in Proc. Int. Conf.Recent Trends Comput.Methods, Commun. Controls, Apr. 2012, pp. 22–25.

A. H. Alkeshuosh, M. Z. Moghadam, I. Al Mansoori, and M. Abdar, ‘‘Using PSO algorithm for producing best rules in diagnosis of heart disease,’’ in Proc. Int. Conf. Comput. Appl. (ICCA), Sep. 2017, pp. 306– 311.

N. Al-milli, ‘‘Backpropogation neural network for prediction of heart disease,’’ J. Theor. Appl.Inf. Technol., vol. 56, no. 1, pp. 131–135, 2013.

C. A. Devi, S. P. Rajamhoana, K. Umamaheswari, R. Kiruba, K. Karunya, and

R. Deepika, ‘‘Analysis of neural networks based heart disease prediction system,’’ in Proc. 11th Int. Conf. Hum. Syst. Interact. (HSI), Gdansk, Poland,

Jul. 2018, pp. 233– 239.

P. K. Anooj, ‘‘Clinical decision support system: Risk level prediction of heart disease using weighted fuzzy rules,’’ J. King Saud Univ.-Comput. Inf. Sci., vol. 24, no. 1, pp. 27– 40, Jan. 2012. doi: 10.1016/j.jksuci.2011.09.002

L. Baccour, ‘‘Amended fused TOPSISVIKOR for classification (ATOVIC) applied to some UCI data sets,’’ Expert Syst. Appl., vol. 99, pp. 115–125, Jun. 2018. doi: 10.1016/j.eswa.2018.01.025.

C.-A. Cheng and H.-W. Chiu, ‘‘An artificial neural network model for the evaluation of carotid artery stenting prognosis using a national-widedatabase,’’ in Proc. 39th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC), Jul. 2017, pp. 2566– 2569

H. A. Esfahani and M. Ghazanfari, ‘‘Cardiovascular disease detection using a new ensemble classifier,’’ in Proc. IEEE 4th Int. Conf. Knowl.- Based Eng. Innov. (KBEI), Dec. 2017, pp. 1011–1014.