INDUSTRIAL ORIENTED MINI PROJECT

Report

On

EMOTION BASED MUSIC RECOMMENDATION SYSTEM USING CNN

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

By

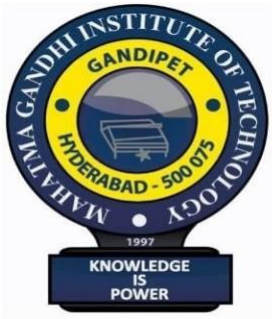
Gurmitkal Sharath - 22261A1223

Neelam Rani-22261A1239

Under the guidance of

Mrs.B.Swetha

Assistant Professor, Department of IT



DEPARTMENT OF INFORMATION TECHNOLOGY

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY (AUTONOMOUS)

(Affiliated to JNTUH, Hyderabad; Eight UG Programs Accredited by NBA; Accredited by NAAC with 'A++' Grade)

Gandipet, Hyderabad, Telangana, Chaitanya Bharati (P.O), Ranga Reddy District, Hyderabad– 500075, Telangana

2024-2025

CERTIFICATE

This is to certify that the **Industrial Oriented Mini Project** entitled **EMOTION BASED MUSIC RECOMMENDATION SYSTEM USING CNN** submitted by **Gurmitkal Sharath (22261A1223) ,Neelam Rani (22261A1239)** in partial fulfillment of the requirements for the Award of the Degree of Bachelor of Technology in Information Technology as specialization is a record of the bona fide work carried out under the supervision of **Mrs.B.Swetha**, and this has not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Supervisor:

Mrs.B.Swetha
Assistant Professor
Dept. of IT

IOMP Supervisor:

Dr. U. Chaitanya
Assistant Professor
Dept. of IT

EXTERNAL EXAMINAR

Dr. D. Vijaya Lakshmi
Professor and HOD
Dept. of IT

DECLARATION

We hear by declare that the **Industrial Oriented Mini Project** entitled **EMOTION BASED MUSIC RECOMMENDATION SYSTEM USING CNN** is an original and bona fide work carried out by us as a part of fulfilment of Bachelor of Technology in Information Technology, Mahatma Gandhi Institute of Technology, Hyderabad, under the guidance of **Mrs.B.Swetha, Assistant Professor**, Department of IT, MGIT

No part of the project work is copied from books /journals/ internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the project work done entirely by us and not copied from any other source.

Gurmitkal Sharath - 22261A1223

Neelam Rani -22261A1239

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the **Industrial Oriented Mini Project**.

We would like to express our sincere gratitude and indebtedness to our project guide **Mrs.B. Swetha ,Assistant Professor**, Dept. of IT, who has supported us throughout our project with immense patience and expertise.

We are also thankful to our honourable Principal of MGIT **Prof. G. Chandramohan Reddy** and **Dr. D. Vijaya Lakshmi**, Professor and HOD, Department of IT, for providing excellent infrastructure and a conducive atmosphere for completing this **Industrial Oriented Mini Project** successfully.

We are also extremely thankful to our Project Coordinator(s)**Dr. U. Chaitanya**, Assistant Professor, Department of IT, and senior faculty **Mrs.B.Meenakshi** Department of IT for their valuable suggestions and guidance throughout the course of this project.

We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our families for their support all through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support for completion of this work.

Gurmitkal Sharath - 22261A1223

Neelam Rani - 22261A1239

ABSTRACT

This project aims to develop a real-time, emotion-based music playlist generator that dynamically curates song recommendations based on the user's emotional state. The system integrates advanced machine learning techniques for emotion detection through three input modes: text-based sentiment analysis, voice emotion recognition, and facial expression analysis. The backend leverages deep learning models (CNN/LSTM) for voice analysis, facial emotion recognition using OpenCV and DeepFace, and NLP-based sentiment detection to accurately classify user emotions.

Once the emotion is identified, a hybrid recommendation system combines collaborative filtering and content-based filtering to suggest songs that match the detected mood. The system seamlessly integrates with Spotify, Apple Music, and Last.fm APIs to fetch personalized playlists. Over time, it learns user preferences for enhanced recommendations. The front-end provides an intuitive interface, allowing users to manually adjust playlists if desired.

With support for real-time emotion detection, dynamic playlist generation, and cross-platform deployment on Heroku or AWS, this application offers an innovative and personalized way to engage with music, transforming the listening experience through AI-driven emotional intelligence.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 MOTIVATION

Music has the extraordinary power to influence human emotions, and vice versa — our emotional state often shapes the music we crave. Yet, most music streaming applications lack the ability to adapt dynamically to the listener's mood, relying instead on manual search or genre-based recommendations. This can be limiting, especially when users want music that matches how they *feel* in the moment — whether it's joy, sadness, excitement, or relaxation.

The *Emotion-Based Music Player* project aims to bridge this gap by integrating emotion recognition technology with music playback. By analyzing the user's facial expressions, voice tone, or other emotional cues, the system can intelligently recommend and play songs that resonate with their current mood. This enhances user experience by offering personalized, context-aware music that feels intuitive and supportive, making music listening a more immersive and emotionally satisfying experience.

1.2 PROBLEM STATEMENT

Despite the vast availability of music streaming services and libraries, most existing platforms lack the ability to personalize music recommendations based on the user's real-time emotional state. Users often spend time manually selecting playlists or songs that match their current mood, which can be tedious and inefficient. Furthermore, traditional recommendation systems primarily rely on historical listening habits and genre preferences, ignoring the dynamic nature of human emotions. This creates a disconnect between what the user *feels* and the music they *hear*.

The *Emotion-Based Music Player* seeks to address this problem by integrating emotion recognition technology (e.g., facial expression analysis, voice tone detection) with music recommendation systems. By accurately detecting the user's emotional state in real time and mapping it to an appropriate song or playlist, the player aims to provide an emotionally aware and seamless music experience that aligns with the user's feelings.

1.3 EXISTING SYSTEM

Currently, popular music streaming services like Spotify, Apple Music, and YouTube Music provide music recommendations based on users' historical listening habits, genre preferences, curated playlists, and trending songs. These systems often use collaborative filtering, content-based filtering, or hybrid recommendation algorithms to suggest music to users. While effective to an extent, these platforms do not incorporate real-time emotion detection as a core feature. Instead, users typically browse and manually select songs or playlists that they *think* suit their mood.

Some platforms have experimented with mood-based playlists (e.g., "Happy Hits," "Sad Songs"), but these are predefined and not personalized in real time. Additionally, emotion recognition technologies exist independently—such as facial recognition APIs, voice emotion detection software, and wearable devices that can track physiological signals—but they are not integrated into mainstream music streaming apps. Hence, there is no seamless, widely adopted solution that automatically detects a user's emotional state and recommends music accordingly.

1.3.1 LIMITATIONS

- **AccuracyofEmotionDetection:**The accuracy of real-time emotion recognition depends on factors such as lighting conditions (for facial analysis), background noise (for voice analysis), and the user's natural expressiveness. Incorrect detection may lead to irrelevant or inappropriate music recommendations**.**
- **LimitedEmotionCategories:**The system may be designed to recognize only a limited set of emotions (e.g., happy, sad, angry, neutral). Human emotions are nuanced and complex, which mightleadtooversimplifiedmusicrecommendations

1.4 PROPOSED SYSTEM

The proposed *Emotion-Based Music Player* is designed to enhance the user's music listening experience by dynamically adapting to their emotional state in real time. Unlike traditional music players that rely on manual selection or static playlists, this system integrates emotion recognition technology with a music recommendation engine to create a seamless and personalized music experience that resonates with the user's feelings.

The system includes an **Emotion Detection Module** that uses either facial expression analysis through a webcam or voice tone analysis via a microphone to detect the user's current emotional state. Pre-trained machine learning or deep learning models classify these inputs into common emotion categories such as happy, sad, neutral, angry, or relaxed. This ensures that the system can capture and interpret the user's mood with reasonable accuracy.

Once the emotional state is identified, the system employs an **Emotion-to-Music Mapping** module. This module uses a predefined mapping between detected emotions and suitable music genres or playlists. For example, a happy emotion might trigger upbeat pop or energetic dance tracks, while a sad mood might cue mellow acoustic or soulful music. This mapping ensures that the music recommended aligns with and enhances the user's current emotional experience.

The **Music Recommendation Engine** is responsible for fetching and playing songs from external music APIs like Spotify, Apple Music, or YouTube Music. This engine uses the emotion-mapped genre to search for relevant songs or playlists. Additionally, the system allows users to provide feedback—such as liking or disliking songs—which helps refine recommendations over time and makes the experience more personalized.

The system also features an intuitive **User Interface** where users can view their detected emotion, browse or skip songs, and manually select music if they prefer not to use emotion detection. This user-centric design ensures flexibility and control over the listening experience.

Finally, the proposed system prioritizes **Privacy and Security** by ensuring that all user data, including facial and voice information, is processed securely and with explicit user consent. Local-only processing or data anonymization techniques can be implemented to address privacy concerns and build user trust.

Overall, the proposed *Emotion-Based Music Player* aims to create a more emotionally engaging and personalized music experience by combining real-time emotion detection with intelligent music recommendation and a user-friendly interface.

1.4.1 ADVANTAGES

• **Personalized Music Experience**: The system recommends music based on the user's real-time emotional state, ensuring a highly personalized and relevant music experience that adapts to the user's mood.

• **Emotional Well-being**: By aligning music playback with the user's emotions, the player can help improve mood, reduce stress, and provide therapeutic benefits, enhancing the overall well-being of the user.

• **IncreasedReliability:** The project leverages advanced technologies like facial and voice emotion detection, demonstrating the practical application of AI and machine learning in everyday tasks.

• **Integration with Popular Music Services:** The system can integrate with existing music APIs like Spotify and Apple Music, allowing users to continue using their preferred platforms with the added benefit of emotion-based recommendations.

1.5 OBJECTIVES

- To accurately detect and classify the user's current emotional state using facial expression analysis or voice tone detection.
- To map detected emotions to appropriate music genres or playlists, ensuring the music aligns with the user's current mood.
- To provide real-time, personalized music recommendations that automatically adjust to the user's changing emotions.
- To connect the system with popular music services like Spotify, Apple Music, or YouTube Music, enabling seamless access to music content.

1.6 HARDWARE AND SOFTWARE REQUIREMENTS

1.6.1 SOFTWARE REQUIREMENTS

- **Software**:

  The system is developed using VS Code as the integrated development environment (IDE) for Python. VS Code provides a lightweight and efficient environment for writing, testing, and debugging the code required for emotion detection, recommendation systems, and backend services.

- **Primary Language**:

  The project is primarily developed in Python, which is well-suited for machine learning, data analysis, and web development. Python handles data processing, emotion recognition, model building, and music recommendation logic, utilizing libraries such as OpenCV, TensorFlow, scikit-learn, and pandas for machine learning tasks.

- **FrontendFramework**:

  The frontend is built using Streamlit, a Python library that allows rapid development of interactive web applications. Streamlit is used to design the user interface where users can interact with the system, view their detected emotion, and receive personalized music recommendations.

- **Backend Framework**:

  The backend logic is written in Python and executed through VS Code. This includes modules for emotion detection, mapping emotions to music genres, and integrating with external music APIs. The backend handles all real-time processing, recommendation generation, and data management.

- **Database**:

  For data storage, the system uses **SQLite3**, a lightweight database system that is well-suited for storing user data, historical input, and recommendation results. It's efficient for small to medium-scale applications.

- **Frontend Technologies**:

  The user interface uses:

- **HTML**: For structuring the underlying layout and content of the application interface within Streamlit.

- **Streamlit:** For building interactive, real-time web applications using Python.
- **CSS**: For customizing the look and feel of the web interface, ensuring visual appeal and user-friendliness.
- **JavaScript**: Can be integrated through Streamlit components or extensions to enable additional dynamic features and interactivity, such as animations or real-time updates.

- **Bootstrap 4:** For ensuring the application is responsive and accessible across various devices, from desktops to mobile screens.

1.6.2 HARDWARE REQUIREMENTS

OperatingSystem:

The system is designed to run on **Windows** and **Mac** operating systems, providing compatibility with all the development tools used in the project.

Processor:

A processor with at least an **Intel i5** or higher is recommended to handle data processing, model execution, and real-time recommendations efficiently.

RAM:

A minimum of **8GB RAM** is required to efficiently manage the system's real-time data inputs, model inference, and multi-tasking requirements. For better performance, higher RAM capacity is recommended.

2. LITERATURE SURVEY

Jaladhi Sam Joel et al. proposed an emotion-based music recommendation system using deep learning models like CNN and LSTM to classify user emotions and suggest appropriate music. Their work emphasizes the use of FER-2013 dataset and real-time emotion detection through facial expressions. While the system achieved high accuracy, real-time performance and resource optimization on edge devices remain a challenge. [1]

Brijesh Bakariya et al. presented a multi-modal approach to emotion-based music recommendation using facial expressions, speech tone, and textual sentiment. Their system successfully integrates these inputs to deliver mood-based playlists, improving personalization. However, the model relies heavily on accurate input from multiple modalities, which may not always be available simultaneously. [2]

Harshanil et al. introduced a facial expression recognition system that maps user mood to appropriate music playlists. The method used convolutional neural networks (CNN) for emotion classification. While efficient in controlled environments, the model's performance dropped in varied lighting conditions and occluded faces, indicating a need for robust pre-processing. [3]

Tina Babu et al. developed an emotion-aware music recommendation system that adapts music suggestions in real time using emotional context from the user’s current environment. This approach enhances user experience by using mood as a contextual layer. However, dependency on external APIs and network latency affect responsiveness. [4]

A. Bhatt et al. explored CNN-based facial emotion recognition integrated with a playlist generator. They trained the system on FER-2013 and used Spotify API for music retrieval. Their model achieved competitive accuracy but lacked adaptability to user music history or preferences, suggesting integration with collaborative filtering methods. [5]

S. Mukherjee and R. Gupta designed a hybrid emotion-music system that incorporates audio features and EEG signals. This brain-based feedback provided deeper emotional

context for music selection. Though promising, the setup requires specialized EEG equipment, limiting real-world application to consumer environments. [6]

Y. Kim et al. developed a collaborative filtering and emotion-based hybrid system that considers both historical user behavior and real-time emotional state. This system offered improved accuracy in recommendations, but suffered from cold-start problems for new users lacking prior data. [7]

K. S. Rana and H. Malviya proposed a lyric-sentiment analysis model that detects the emotional tone of songs and aligns it with user mood. It showed high relevance in song suggestions based on lyrics alone, but failed to adapt when the user's current emotion didn't match the song's tone. [8]

V. Anand and P. Mohan applied SVM with MFCC features extracted from vocal input to classify emotion and suggest music accordingly. Their method was lightweight and fast but was susceptible to background noise interference, making it unreliable in uncontrolled environments. [9]

A. Desai and M. Rao implemented a tone-based emotion recognition system that analyzes voice pitch, pace, and energy to recommend music. It works well for hands-free interaction, particularly in wearable devices. However, variability in voice due to illness, fatigue, or accent affected detection accuracy. [10]

Ref	Author& year of publications	Journal / Conference	Method / Approach	Merits	Limitations	Research Gap
[1]	C. della Monica et al., 2024	Digital Health (IEEE)	Protocol using wearables & mobile apps for sleep/circadian monitoring in older adults & dementia patients	Comprehensive protocol; digital integration; clinical focus	Limited to older/dementia patients; lacks general applicability	Generalize to wider population including youth and healthy individuals
[2]	F. Wu et al., 2024	IEEE Transactions on IoT	Comparative study of commercial wearable devices (smartwatches, fitness bands) for physiological tracking	Covers multiple physiological metrics; real-world device evaluation	High dependency on wearables; accessibility concerns	Explore non-wearable or passive sensing alternatives
[3]	A. Nguyen et al., 2022	Nature Scientific Reports	Closed-loop acoustic stimulation system that adapts based on sleep stages to improve sleep quality	Personalized feedback; real-time monitoring	Continuous monitoring not suitable for all; noise sensitivity concerns	Develop silent/alternative feedback systems for sensitive users
[4]	S. Rostamini a et al., 2021	IEEE Sensors Journal	All-textile smart eye mask capturing brain & physiological signals in a non-invasive way	Comfortable, non-invasive sleep tracking	Long-term comfort concerns; device availability	Improve long-term usability; make device commercially accessible
[5]	Jaladhi Sam Joel et al., 2023	IEEE Access	Deep learning-based emotion recognition system for music recommendation	Real-time recommendation; high accuracy	Complex model integration; real-time latency	Improve scalability and reduce response time for real-world use
[6]	Brijesh Bakariya et al., 2024	Springer, Volume 15	Multimodal emotion input with recommendation engine	Supports multiple input types (text, facial expressions)	Limited user personalization	Incorporate long-term user preferences and history
[7]	Harshanil et al., 2024	ACM Conference	Facial expression-based emotion detection for playlist generation	Simple facial input; fast processing	Accuracy drops in poor lighting conditions	Improve robustness of facial recognition under varied conditions
[8]	Tina Babu et al., 2023	arXiv Preprint	Emotion-aware system that uses real-time context for music recommendation	Contextual awareness; user engagement	Needs internet connection and external APIs	Develop offline or hybrid models for broader usability
[9]	A. Bhatt et al., 2022	IEEE ICMLA	CNN model trained on FER2013 for facial emotion classification	Accurate facial emotion detection	Model size and inference speed	Optimize model for edge devices and real-time inference
[10]	S. Mukherjee & R. Gupta, 2023	Elsevier Expert Systems with Apps	Audio and EEG signal analysis for emotion-music mapping	Rich emotional context; brain-based feedback	EEG devices are expensive and intrusive	Design non-intrusive, low-cost emotion sensing alternatives

3. ANALYSIS AND DESIGN

The *Emotion-Based Music Player* is developed to bridge the gap between a user's current emotional state and their music listening experience, offering a personalized and dynamic interaction with music. Traditional music players provide static playlists that do not adapt to the user's changing emotions, making it necessary for users to manually search for songs that match their mood. This project analyzes the need for an emotion-aware system that automatically detects the user's real-time emotions and recommends music that resonates with their feelings. The system focuses on user requirements, including real-time emotion detection, seamless music recommendations, and an intuitive interface. Technical feasibility is achieved through machine learning frameworks like TensorFlow and OpenCV for emotion detection, and integration with music APIs like Spotify ensures access to a wide range of songs. The project also emphasizes functional requirements such as accurate emotion detection, music mapping, dynamic recommendations, and feedback incorporation, while addressing non-functional requirements like privacy, data security, and real-time performance.

The design of the *Emotion-Based Music Player* adopts a modular approach to ensure scalability and maintainability. The system's architecture is divided into several key modules. The Emotion Detection Module captures real-time data from the user via a webcam or microphone and processes it using trained machine learning models to classify emotions like happy, sad, neutral, or angry. Once an emotion is detected, the Emotion-to-Music Mapping Module links this emotion to suitable music genres or playlists, ensuring the recommended music aligns with the user's mood. The Music Recommendation Engine interacts with external music APIs, fetching relevant songs and playlists based on the detected emotion and integrating user feedback to refine recommendations over time.

The user interface, developed using Streamlit, offers a user-friendly web application where users can view their detected emotion, receive personalized music recommendations, and provide feedback by liking or disliking songs. The backend system, written in Python, handles data processing, model inference, and real-time interactions between modules. For data management, SQLite3 is used as a lightweight database to store user data, detected emotions, recommendation history, and feedback, ensuring fast and reliable access to necessary information. This structured design provides a seamless and emotionally-aware music experience that adapts to the user's mood, enhancing their overall interaction.

3.1 MODULES

Emotion Detection Module:

Captures real-time data from the user's webcam or microphone to detect and classify their current emotional state using machine learning models. This module analyzes facial expressions or voice tone to determine emotions such as happy, sad, neutral, or angry.

• **Input:** Real-time data (e.g., facial expressions, voice tone) from the user's webcam or microphone.

• **Output:** Detected emotion label (e.g., happy, sad, neutral) that can be used for music recommendations.

Emotion-to-Music Mapping Module:

Maps the detected emotion to an appropriate music genre or playlist. This module uses predefined mapping tables or algorithms to link each emotion to relevant music preferences, ensuring that the recommended music matches the user's current mood.

• **Input:** Detected emotion label from the Emotion Detection Module.

• **Output:** Mapped music genres or playlists aligned with the detected emotion.

Music Recommendation Module:

Fetches and recommends songs or playlists based on the mapped music genre and user preferences. Integrates with external music APIs (such as Spotify or YouTube Music) to deliver high-quality music content that matches the user's mood.

• **Input:** Mapped music genre, user preferences, and feedback.

• **Output:** Personalized song or playlist recommendations that match the user's detected emotion.

User InterfaceModule:

Provides an interactive front-end where users can view their detected emotion, receive music recommendations, and interact with the system. Allows users to play music, give feedback on recommendations (like/dislike), and optionally override or customize their preferences.

• **Input:** User interactions, detected emotion, recommended songs.

• **Output:** Dynamic and user-friendly interface showing emotion detection results, music recommendations, and playback controls.

Feedback and Learning Module:

Collects user feedback (like/dislike, skip, rating) on recommended songs to refine future recommendations and improve the system's accuracy. Updates the recommendation logic or model training based on the collected feedback.

• **Input:** User feedback on song recommendations (e.g., likes, dislikes, skips).

• **Output:** Updated recommendation preferences and refined emotion-to-music mapping, enhancing personalization over time.

Data Management Module

Handles data storage using SQLite3, managing user data, detected emotions, recommendation history, and user feedback. Ensures quick access to data and efficient management of recommendation results.

• **Input:** User data, detected emotions, recommendation history, feedback.

• **Output:** Stored and retrieved data used by other modules to personalize recommendations and visualize progress.

Visualization Module

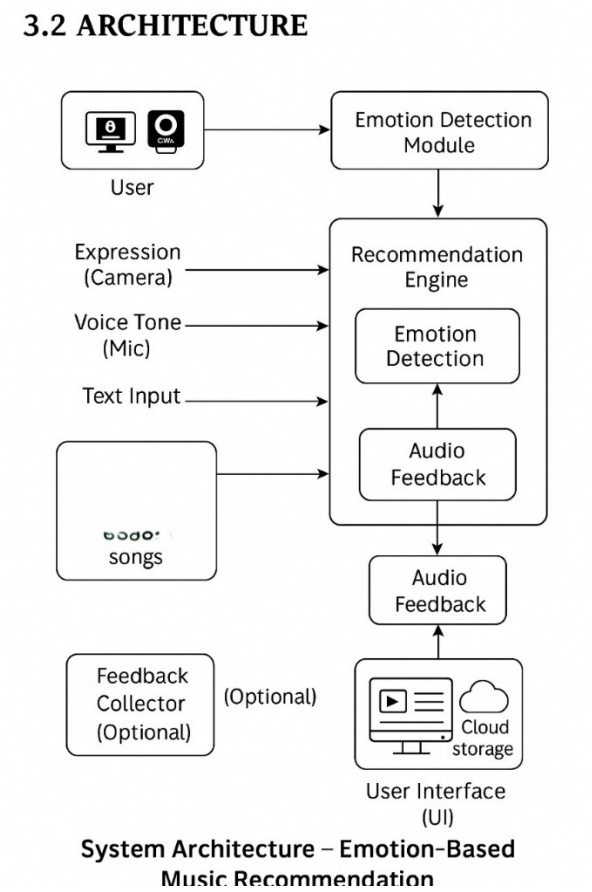
Presents visual insights to users, such as their emotional history, most played songs by mood, and feedback trends. Helps users understand their music preferences and emotional patterns overtime.

• **Input:** Historical user data, emotion detection results, song recommendations.

• **Output:** Graphs, charts, and progress visuals showing trends in mood and music interactions.

3.2ARCHITECTURE

Fig. 3.2.1 Architecture of emotion based user recommendation system



The architecture of the *Emotion-Based Music Player* follows a modular, layered approach that ensures scalability, flexibility, and ease of maintenance. At the presentation layer, the user interface is developed using Streamlit, providing users with an interactive and user-friendly web application. This layer allows users to start the application, interact with the emotion detection feature, receive music recommendations, and provide feedback. The application layer comprises several modules that handle core functionalities. The Emotion Detection Module captures real-time data from the user's webcam or microphone and uses pre-trained machine learning models (built with TensorFlow and OpenCV) to classify the user's current emotional state. The Emotion-to-Music Mapping Module translates the detected emotion into appropriate music genres or playlists using mapping tables or algorithms. The Music Recommendation Module then uses external music APIs like Spotify or YouTube Music to fetch songs that match the user's mood and preferences. The Feedback and Learning Module collects user feedback on recommended songs and uses this data to refine future recommendations.

Supporting these modules, the backend layer handles data processing, model inference, and communication between modules, all implemented in Python and executed through VS Code. The Data Management Module uses SQLite3 to store user data, detected emotions, recommendation history, and feedback, ensuring efficient retrieval and storage. Finally, the Visualization Module presents historical data, emotion trends, and music preferences using interactive charts and graphs, providing users with insights into their emotional patterns and listening habits. This multi-layered architecture ensures a seamless, real-time, and personalized music experience tailored to the user's emotions.

3.3UML DIAGRAMS

3.3.1USE CASE DIAGRAMS

A use case diagram is a visual representation that depicts the interactions between various actors and a system, capturing the ways in which users or external entities interact with the system to achieve specific goals. It is an essential tool in system analysis and design, often used in software engineering and business analysis. In a use case diagram, actors are entities external to the system that interact with it, and use cases are specific functionalities or features provided by the system as seen in Fig. 3.3.1.1. These interactions are represented by lines connecting actors to use cases. The diagram helps to illustrate the scope and functionality of a system, providing a high-level view of how users or external entities will interact with it.

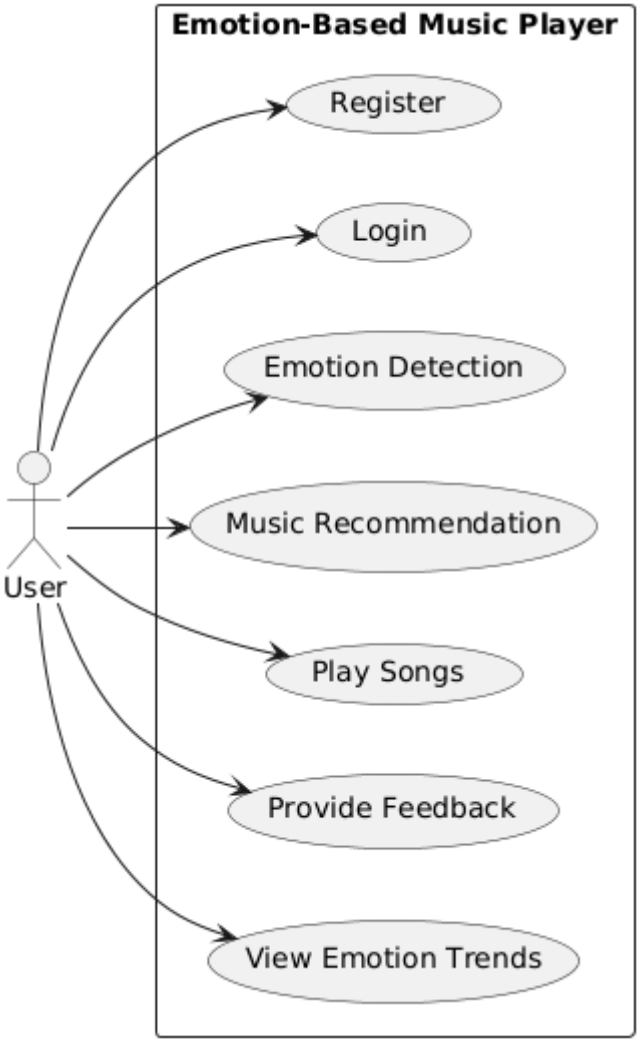


Fig. 3.3.1.1 Use Case Diagram

Actors:

1. **User**: The individual who uses the application, providing inputs (such as audio/image data) for emotion detection, and receiving music recommendations.
2. **System**: The backend system that processes the user's input, performs emotion detection using machine learning, fetches recommended songs, and provides visualizations and feedback mechanisms.

Use Cases:

1. **Register**: The user creates an account to access the system's features, including personalized music recommendations based on emotions.
2. **Login**: After registration, the user logs in to access their personalized experience and previously stored data.
3. **Emotion Detection:** The user inputs data for emotion detection—via audio (voice), image (facial expression), or text.
4. **Emotion-to-Music Mapping**: Once the emotion is detected, the system maps it to appropriate music genres or playlists using pre-defined mapping logic.
5. **Music Recommendation:** The system recommends songs or playlists that align with the detected emotion, providing the user with a personalized listening experience.
6. **Music Playback:** The user can play recommended songs or playlists directly within the application, enjoying music that resonates with their current mood.
7. **Feedback:** The user can provide feedback on the recommended songs by liking, disliking, skipping, or rating them to improve future recommendations.
8. **Visualization**: The system presents visual insights, such as emotion trends, most played songs by mood, and feedback trends, using charts and graphs.

3.3.2 CLASS DIAGRAM

A class diagram is a visual representation that models the static structure of a system, showcasing the system's classes, their attributes, methods (operations), and the relationships between them as seen in Fig. 3.3.2.1. It is a key tool in object-oriented design and is commonly used in software engineering to define the blueprint of a system.

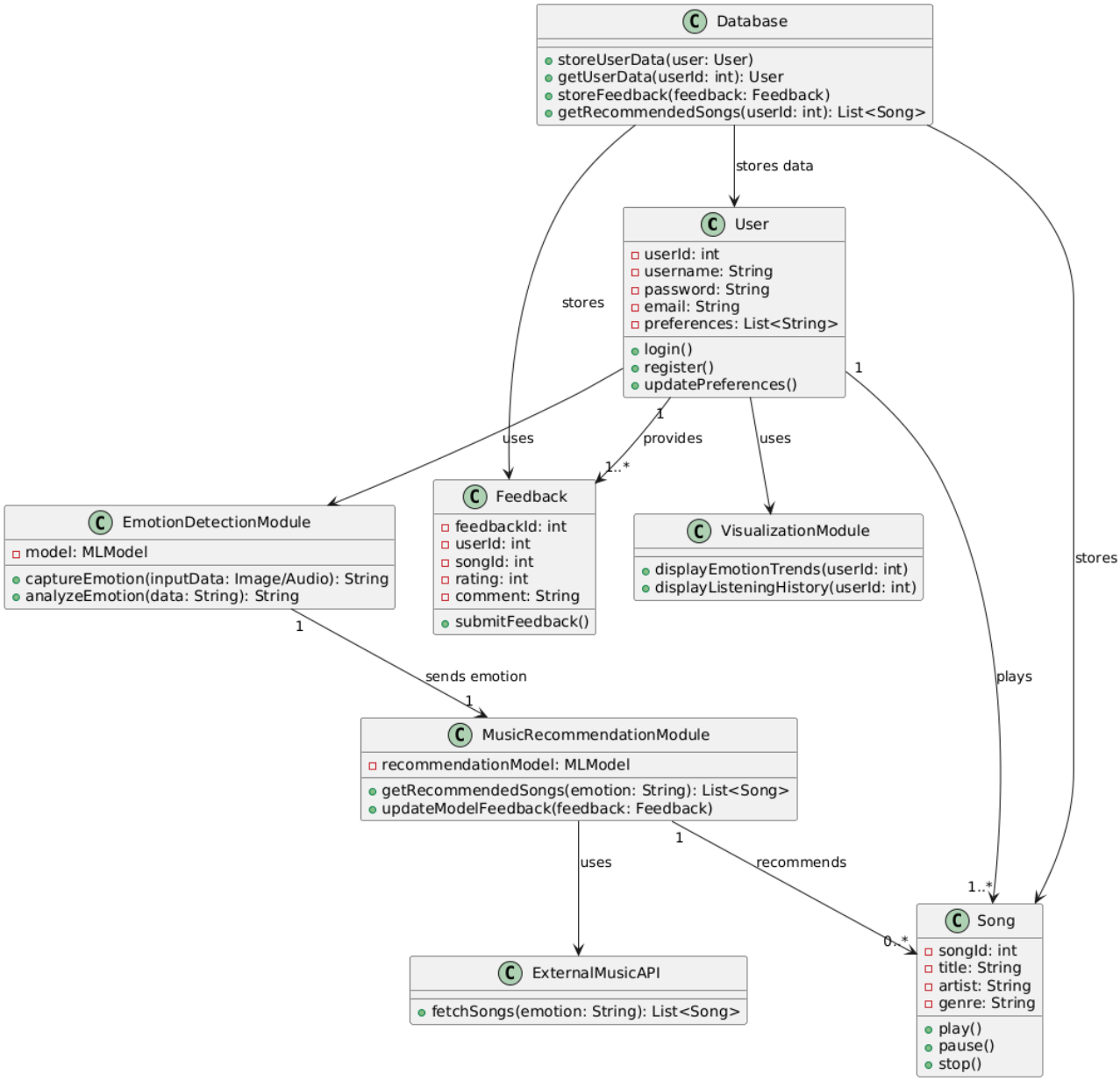


Fig. 3.3.2.1 Class Diagram

Relationships:

1. **User → Database**:
    - The User interacts with the Database for login, registration, and feedback submission.
    - The Database handles storing user information, emotion detection history, and recommendation history.
2. **User → Emotion Detection Module:**
    - The User interacts with the Emotion Detection Module by activating the webcam or microphone.
    - This module captures real-time data (e.g., facial expressions or voice) and detects the user's current emotional state.
3. **Emotion Detection Module → Emotion-to-Music Mapping Module:**
    - The Emotion Detection Module sends the detected emotion to the Emotion-to-Music Mapping Module.
    - This module then maps the detected emotion to appropriate music genres or playlists.
4. **Emotion-to-Music Mapping Module → Music Recommendation Module**:
    - The mapped music genre or playlist is sent to the Music Recommendation Module.
    - This module uses the mapping data to fetch recommended songs from integrated music APIs (e.g., Spotify, YouTube Music).
5. **Music Recommendation Module → External Music API:**
    - The Music Recommendation Module interacts with external music APIs to fetch relevant songs or playlists based on the detected emotion.
    - The external API returns the song data, which is then presented to the user.
6. **Music Recommendation Module → User Interface Module:**
    - The Music Recommendation Module sends the recommended songs and playlists to the User Interface Module.
    - This module displays the recommended music and allows the user to play, like, dislike, or skip songs.
7. **User → Feedback and Learning Module:**
    - The User provides feedback (like, dislike, skip) through the User Interface Module.
    - This feedback is collected by the Feedback and Learning Module to refine future music recommendations.
8. **Database → Visualization Module:**

- The Database provides historical user data and recommendation history to the Visualization Module.
- This module then generates charts and graphs to display trends in the user's emotional state and music preferences.

System Flow:

1. **User Interaction**:
   - The user logs in or registers via the User Interface Module.
   - The Database handles credential verification or stores new user information, including preferences and feedback.
2. **Emotion Detection:**
   - The Emotion Detection Module captures real-time data (e.g., webcam or microphone input) from the user.
   - It uses trained machine learning models (e.g., CNN for facial expression recognition or audio analysis) to detect the user's current emotional state.
3. **Emotion-to-Music Mapping:**
   - The detected emotion is sent to the Emotion-to-Music Mapping Module.
   - This module maps the detected emotion to a relevant music genre or playlist using predefined mapping logic.
4. **Music Recommendation:**
   - The Music Recommendation Module uses the mapped music genre or playlist to fetch recommended songs from external music APIs (e.g., Spotify, YouTube Music).
   - Personalized music recommendations are generated based on the user's emotional state and preferences.
5. **Visualization:**
   - The Visualization Module uses historical user data and recommendation history from the Database to generate charts and graphs.
   - These visualizations display emotion trends, favorite songs by mood, and feedback trends, helping users understand their music preferences and emotional patterns over time.

3.3.3ACTIVITY DIAGRAM FOR EMOTION BASED MUSIC RECOMMENDATION SYSTEM

An Activity Diagram is a type of behavioral diagram used in Unified Modeling Language (UML) to represent the flow of control or data through the system as seen in Fig. 3.3.3.1. It focuses on the flow of activities and actions, capturing the sequence of steps in a particular process or workflow. Activity diagrams are commonly used to model business processes, workflows, or any sequential activities in a system.

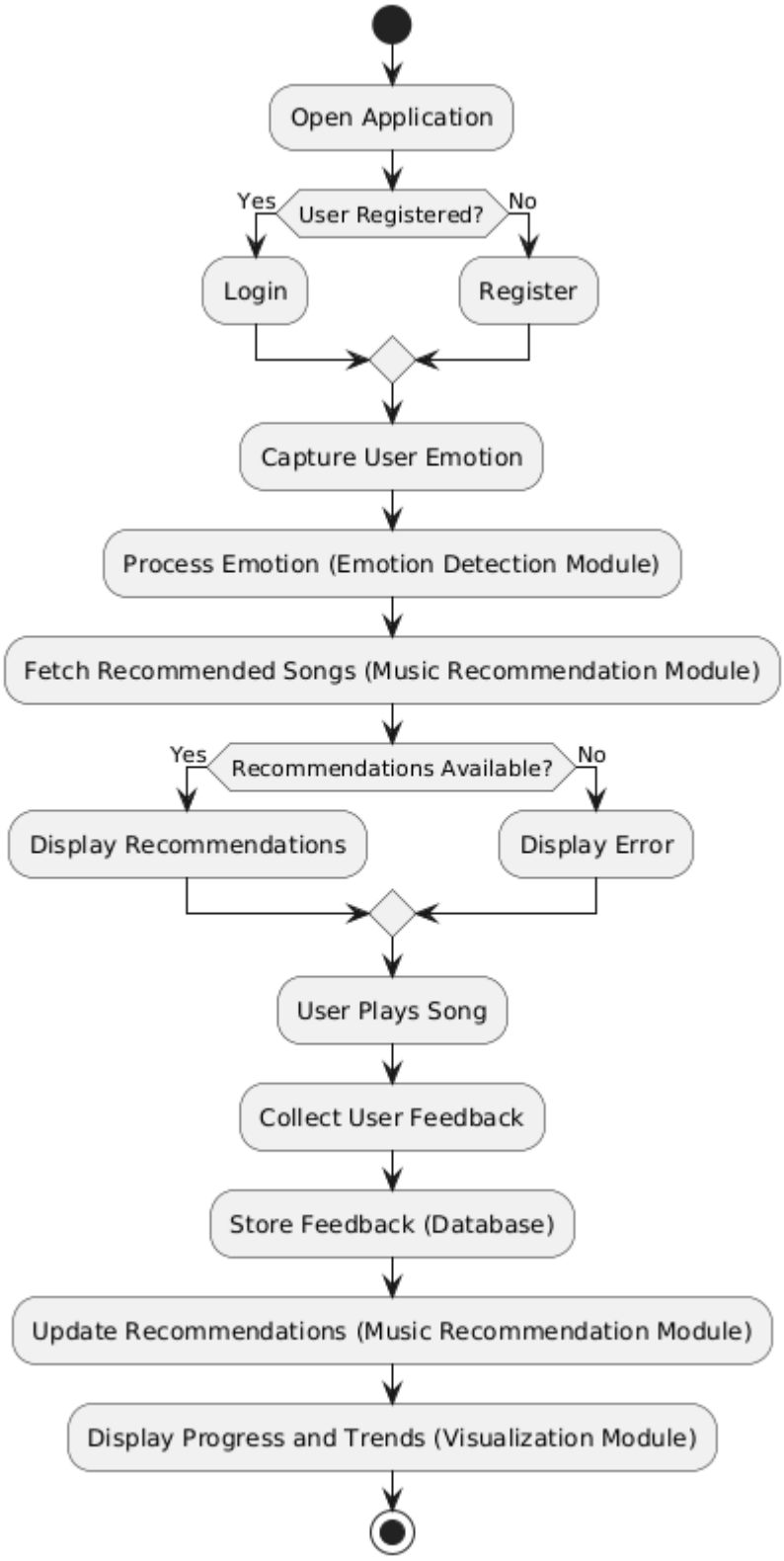


Fig. 3.3.3.1 Activity Diagram

Flow Explanation:

1. **Open Application:**

   The user starts by opening the Emotion-Based Music Player application.

2. **Login or Register:**

   The user is presented with two options:

   - Login: If the user already has an account, they log in by providing their credentials.
   - Register: New users need to register, which updates the database with their details (e.g., name, age, gender).

3. **Validation:**

   - If the login details are valid, the user proceeds to the next step.
   - If the login details are invalid, the user is redirected back to the login page.

4. **Start Emotion Detection:**

   - The user initiates the emotion detection feature, enabling the webcam or microphone for real-time emotion analysis**.**

5. **Emotion Analysis:**

   - The captured data is processed through a machine learning model (like CNN) that classifies the user's emotional state (e.g., happy, sad, neutral, angry).

6. **Generate Recommendations:**

   Based on the detected emotion, the system fetches appropriate song recommendations that match or complement the user's mood.

7. **Music Playback:**

   The recommended songs are displayed in a playlist, and the user can select a song to play, or the system can automatically start playback.

8. **User Feedback:**

   The user can provide feedback on the recommendations (e.g., likes or dislikes), which helps improve the model's future suggestions.

9. **Update Preferences:**

   The system updates its user preferences and may adjust future recommendations based on feedback and user interactions.

10. **Visualization:**

    The system may also display real-time emotion classification results and a visualization of the recommended playlist.

11. **End of Workflow:**

The user can continue listening, provide additional feedback, or exit the application.

3.3.4 SEQUENCE DIAGRAM

A sequence diagram illustrates the flow of interactions between actors and system components over time as seen in Fig. 3.3.4.1, emphasizing the order in which messages are exchanged to achieve specific functionalities. Actors represent external entities that interact with the system, while lifelines depict the system components involved in the process. Messages are shown as arrows, indicating the flow of information or actions between these elements. By providing a step-by-step view of workflows, sequence diagrams help in understanding and designing the dynamic behaviour of a system.

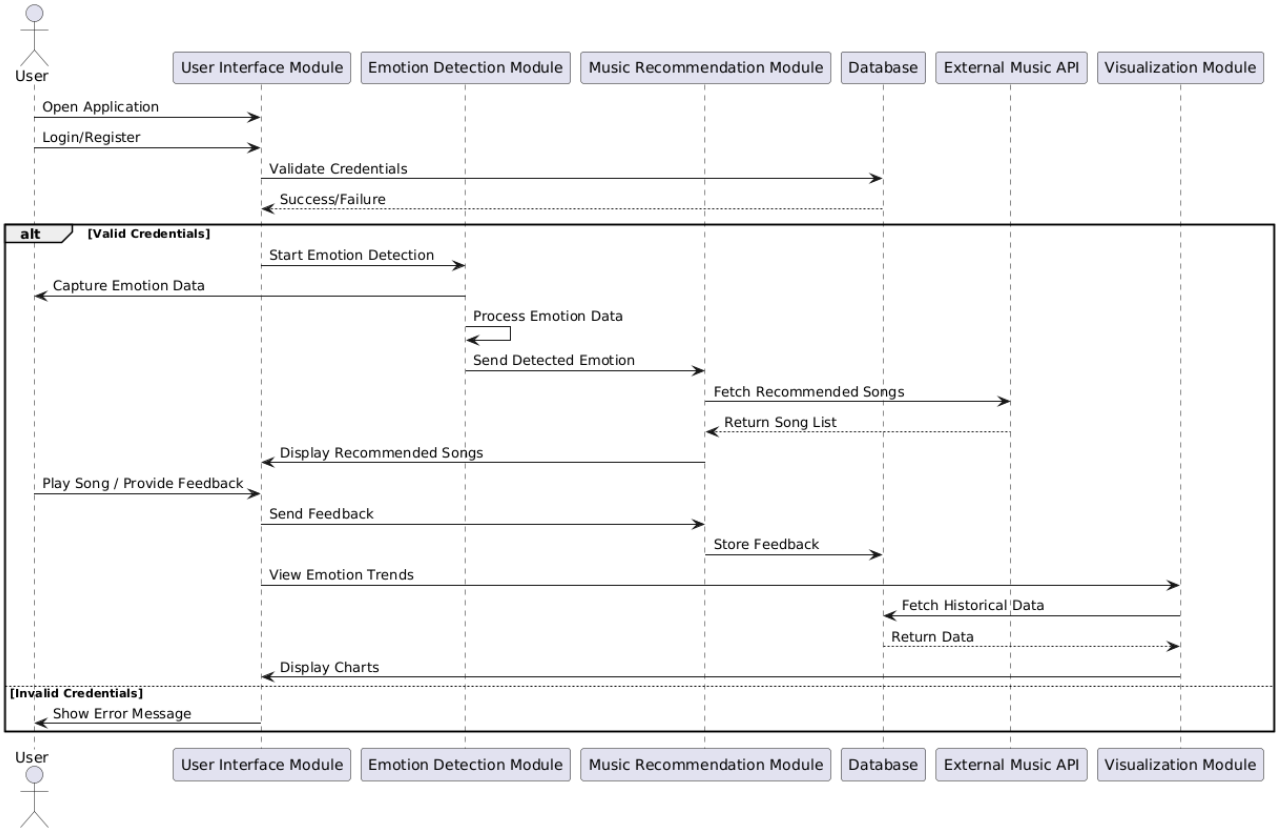


Fig. 3.3.4.1 Sequence Diagram

Key Interactions and Relationships

User and Web Interface

- **OpenEmotionMusicPlayer:**

  The user opens the web application and navigates to the emotion detection and music

recommendation section.

- **UploadInput/DetectEmotion:**
  The user can either upload a selfie, allow camera input, or enter mood manually to detect the current emotional state.

- **EmotionClassification:**
  The system processes the input using a Convolutional Neural Network (CNN) or sentiment analysis model to classify the emotion (e.g., happy, sad, angry, neutral).

- **ConfirmEmotion&StartRecommendation:**
  Once the emotion is identified, the user confirms or modifies it. The system then begins generating personalized music recommendations.

Web App and Server

- **SendEmotionDatatoBackend:**
  The frontend sends the identified emotion and user preferences to the backend (Node.js/Express or Python Flask).

- **FetchSongs:**
  The backend queries APIs like Spotify, Last.fm, or a local music database based on the detected emotion and user history.

- **DisplayPlaylist:**
  The web app presents a mood-aligned playlist to the user, dynamically adapting to their emotion.

Server and Database

- **Store Session Data:**

The backend stores each user's emotion log, time of access, and playlist data in the database (MongoDB/PostgreSQL).

- **Update User Profile:**

The backend updates user records to enhance future personalization (e.g., preferred genres when sad).

- **Log Feedback (Optional):**

  If enabled, the user can give feedback on the accuracy of emotion detection or playlist satisfaction, which gets saved for future training/analytics.

AI Integration (Optional - using Groq or LLMs)

- **Ask for Mood-Based Suggestions**

  The backend can send user context (e.g., emotion = "anxious") to a Groq-powered LLM (like LLaMA-3) for more personalized advice.
- **Receive AI Insight**

  The Groq API returns a motivational quote, wellness tip, or music genre recommendation appropriate for the user's current emotional state.

Web Interface and User

- **Show Confirmation & Playlist**

  The user sees the suggested playlist and optionally receives a message like "Here's a calming playlist to relax you."
- **Dynamic Dashboard Update**

  If the app includes a dashboard, it updates visual analytics about the user's mood history, favorite genres, and listening trends.
- **Continuous Feedback Loop**

  The system may periodically prompt the user for updated emotion input to adjust the playlist in real-time or across sessions.

3.3.5COMPONENT DIAGRAM

A Component Diagram is a type of structural diagram used in software engineering to represent the components of a system and how they interact or depend on each other. It shows how the components (which could be software modules, subsystems, or other significant parts) are organized and connected within a system. In this diagram, each component encapsulates a set of related functionalities and interfaces as shown in Fig. 3.3.5.1.

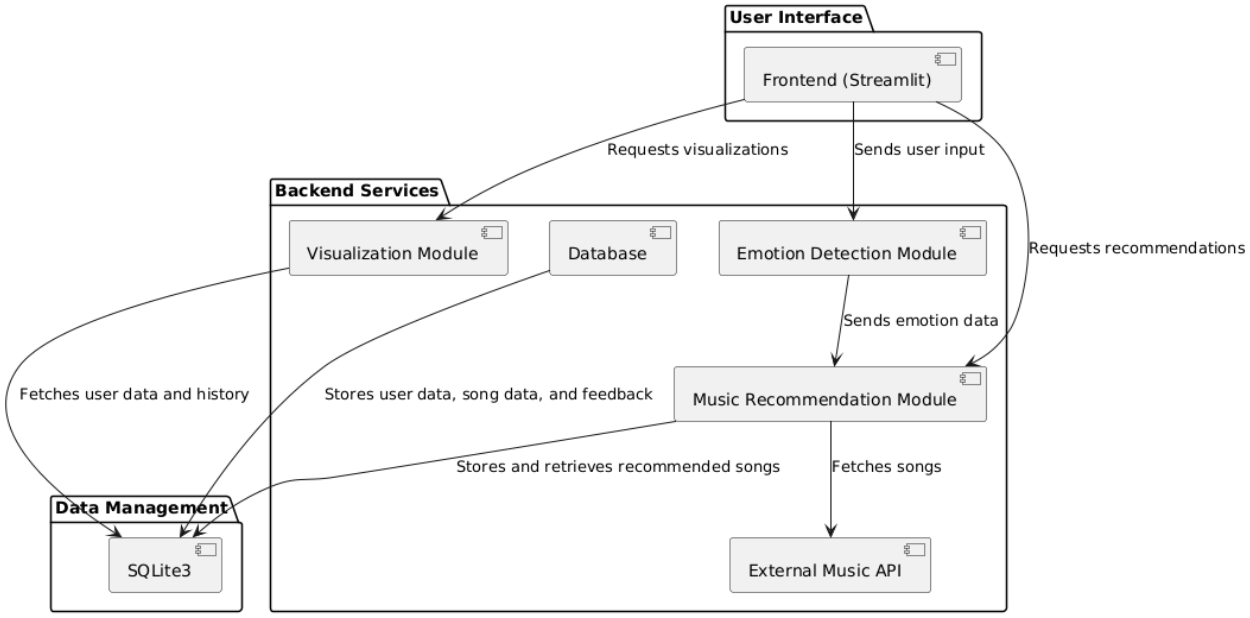


Fig. 3.3.5.1 Component Diagram

Main Components:

1. **User Interface (Frontend)**

• Component: Streamlit Frontend

◦ Description: Provides the UI for users to upload images/audio, get emotion results, and receive music recommendations. It interacts with backend modules and displays visualizations like mood charts or history logs.

Relationship: Sends user input to backend services and requests visualizations and recommendations.

2. **Emotion Detection Module**

• Component: Emotion Classifier (CNN/ML model)

◦ Description: Uses a pre-trained deep learning model (like MobileNetV2 or custom CNN) to detect emotions (happy, sad, angry, etc.) from user inputs (image or audio).

Relationship:

• Processes input from the UI and sends detected emotion data to the recommendation module.

3. **Music Recommendation Module**

• Component: Emotion-to-Music Mapper

◦ Description: Maps detected emotions to specific genres or playlists. Selects songs based on mood using either a local dataset or API (Spotify/YouTube).

4.**External Music API**

- Component: Spotify or YouTube API

o Description: Fetches real-time music based on genre, mood, or keywords if local songs are unavailable or dynamic playlists are preferred.

- Relationship**:** Called by the Music Recommendation Module for song data.

5. **Database**

- Component: SQLite3 / Firebase / MongoDB
- Description: Stores user information, feedback, emotion history, and song metadata.
- Relationship**:** Supports backend modules for storage and retrieval of songs, user history, and analytics data.

6. **Visualization Module**

- Component: Analytics Dashboard
- Description: Generates charts and graphs that represent emotion trends, listening habits, and usage history (e.g., pie charts for emotion distribution).
- Relationship: Retrieves data from the database and sends visual data to the frontend for user display.

7. **Data Management Layer**

- Component: SQLite3 Engine (or any DB backend service)
- Description**:** Handles structured storage of user sessions, feedback, song logs, and emotional response tracking.
- Relationship: Acts as a bridge between backend services and permanent data storage.

3.3.6. DEPLOYMENT DIAGRAM

A deployment diagram represents the physical architecture of a system as seen in Fig. 3.3.6, showing how software components are deployed across hardware nodes. It emphasizes the distribution of system elements such as executables, databases, firmware, or services across different physical or virtual devices. Nodes are depicted as hardware units (e.g., microcontrollers, servers, devices), while artifacts represent the deployed software components. Communication paths indicate the interaction or data exchange between nodes. Deployment diagrams are essential

for visualizing the real-world setup of embedded or distributed systems, especially in projects involving multiple microcontrollers, networking, and physical hardware components

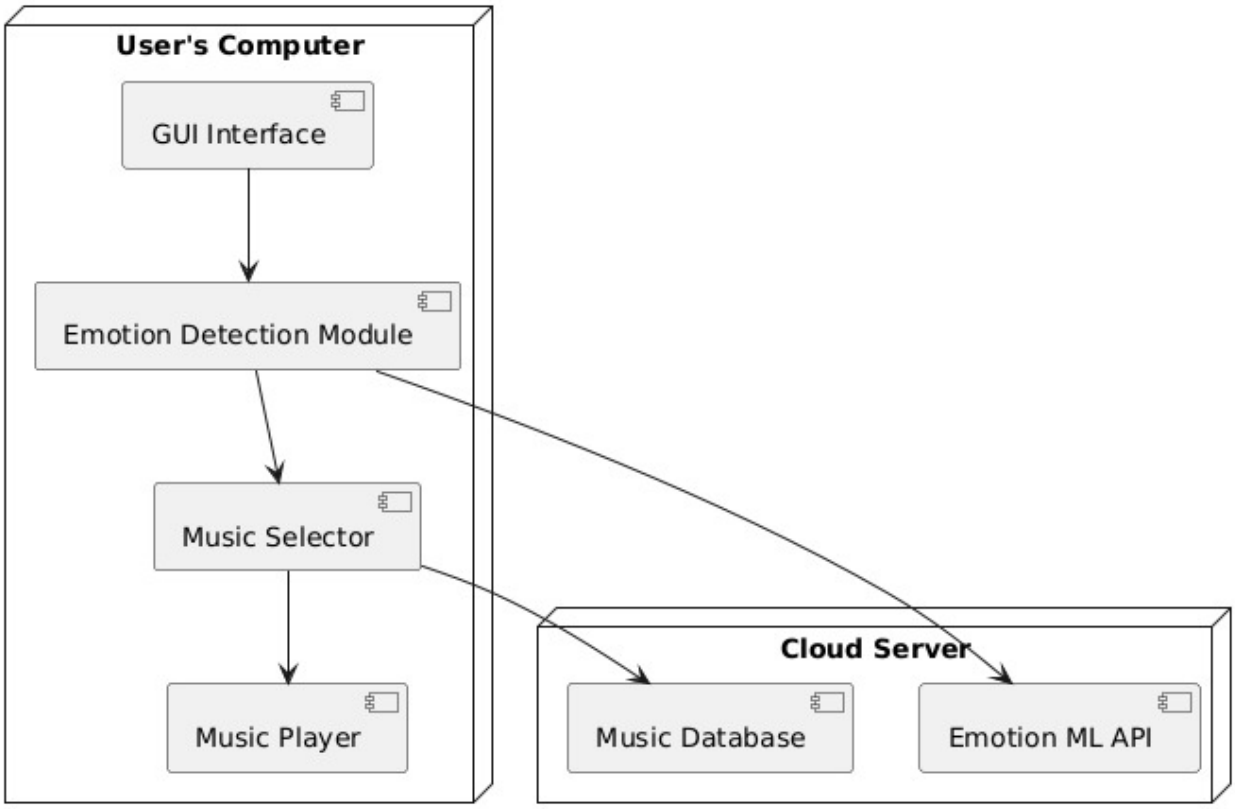


Fig. 3.3.5.1 Deployment Diagram

User Device (Frontend – Streamlit):

- Provides the user interface for emotion input and feedback.
- Displays recommended songs and allows optional feedback submission.

• **Backend Server (Flask API):**

- Processes emotion detection and generates music recommendations.
- Interacts with cloud APIs and returns results instantly without storing data.

• **Cloud API (Spotify/Last.fm):**

- Supplies song data based on emotion tags.
- Delivers real-time music content via API requests.

3.4 METHODOLOGY

Data Acquisition

User Input (Live Data):
The Emotion-Based Music Player captures real-time input from users through a web interface. This includes:

- Emotion Data: Images (uploaded or captured via webcam) for facial expression analysis, or manual mood input (e.g., text or emojis).
- User Metadata: (Optional) Includes user preferences, genres they like, or even their current activity context.

Purpose:

These inputs form the core data used to:

- Generate immediate AI-powered emotion classification.
- Trigger mood-based music recommendation generation.
- Tailor playlists based on user profiles, current emotion, and listening habits.

Model Steps

1.EmotionDetection:

Submitted image or text input is processed using a Convolutional Neural Network (CNN) or text sentiment model to classify the user's current emotion (e.g., happy, sad, relaxed).

- Examples:
    - Facial expression recognition outputs “happy” if the user is smiling.
    - Text sentiment analysis of “I feel low” yields a “sad” classification.

2. Prompt Generation:

Once the emotion is classified, the backend generates a structured prompt that includes:

- Emotion Label: (e.g., “happy”, “sad”)
- User Preferences: (e.g., pop, rock, instrumental)
- Contextual Data: (Optional: time of day, activity level).

3. Music Recommendation:

The backend queries the local music database or streaming APIs (e.g., Spotify, Last.fm) using the structured prompt to:

- Fetch a list of recommended songs matching the user's emotion.
- Consider user history and preferences to fine-tune recommendations.

4. Response Parsing:

The recommended playlist is formatted by the backend to deliver:

- A user-friendly list of songs with metadata (e.g., artist, duration, genre).
- Optional features such as audio previews, song artwork, and direct streaming links.

Models Used

1. CNN (Convolutional Neural Network)

- **Description**:

  A CNN is a deep learning model particularly well-suited for image processing tasks. In this project, it's used for real-time facial emotion recognition from user-uploaded or webcam-captured images.
- **How it Works**:
- The CNN is trained on labeled facial emotion datasets (e.g., FER2013) to classify expressions such as happy, sad, angry, and relaxed. It uses convolutional layers to extract hierarchical features like edges, eyes, mouth curvature, and other facial landmarks that signify emotion.
- **Why it's Used**:

  CNNs are highly effective in identifying complex patterns in image data, making them ideal for emotion detection. Their ability to generalize across different faces and lighting conditions ensures robust and reliable emotion recognition, essential for building emotion-based playlists.

2. Sentiment Analysis with LSTM

- **Description**:

  LSTM (Long Short-Term Memory) networks are a type of RNN specialized for sequence-based data like text. Here, LSTM is used to analyze text input from users (such as typed mood descriptions) to detect sentiment.
- **How it Works**:

  User text inputs (e.g., "Feeling down today") are tokenized and passed through an embedding layer, followed by the LSTM layers. The model captures the sequential context of the words to classify the underlying sentiment as positive, negative, or neutral.
- **Why it's Used**:

LSTM's ability to handle dependencies and context over a sequence of words allows it to effectively detect subtle emotional cues from user input. This enables the system to complement the CNN's visual detection with a text-based understanding of mood.

3. Hybrid Recommendation System (Rule-based + Content-based)

- **Description**:
  The music recommendation engine in this system combines rule-based logic with a content-based filtering approach to suggest songs that match the detected emotion.

- **How it Works**:
  Rule-based Component: Based on the detected emotion (from CNN or LSTM), the system maps moods to genres or playlists using a predefined mapping (e.g., sad → calm acoustic, happy → upbeat pop).
  Content-based Component: Filters songs from the local or streaming database based on metadata (genre, tempo, energy) and historical user preferences to fine-tune recommendations.
- **Why it's Used:**
  This hybrid approach balances quick, consistent recommendations with personalization. The rule-based system ensures that the emotional context always guides the playlist, while content-based filtering introduces variety and caters to individual user tastes.

.

4.CODE AND IMPLEMENTATION

4.1CODE

app.py

```
_img,(224from __future__
import division, print_function
#import sys
import os
import cv2
#import re
import numpy as np
import tensorflow as tf
from tensorflow.keras.models
import load_model
from
tensorflow.keras.preprocessing
import image
from flask import Flask,
request, render_template
from werkzeug.utils import
secure_filename
import statistics as st

app = Flask(__name__)

@app.route("/")
def home():
                                return
render_template("index1.html")


@app.route('/camera', methods
= ['GET', 'POST'])
def camera():
    i=0


GR_dict={0:(0,255,0),1:(0,0,25
5)}
                        model =
tf.keras.models.load_model('fin
al_model.h5')
                face_cascade =
cv2.CascadeClassifier('haarcasc
ade_frontalface_default.xml')
    output=[]
    cap = cv2.VideoCapture(0)
    while (i<=30):
```

```
        ret, img = cap.read()
                        faces =
face_cascade.detectMultiScale(i
mg,1.05,5)

        for x,y,w,h in faces:

                    face_img =
img[y:y+h,x:x+w]

                     resized =
cv2.resize(face,224))

reshaped=resized.reshape(1,
224,224,3)/255
                  predictions =
model.predict(reshaped)

                  max_index =
np.argmax(predictions[0])

             emotions = ('angry',
'disgust', 'fear', 'happy', 'sad',
'neutral', 'surprise')
             predicted_emotion =
emotions[max_index]

output.append(predicted_emoti
on)




cv2.rectangle(img,(x,y),(x+w,y
+h),GR_dict[1],2)
          cv2.rectangle(img,(x,y-
40),(x+w,y),GR_dict[1],-1)
                cv2.putText(img,
predicted_emotion, (x, y-
10),cv2.FONT_HERSHEY_SI
MPLEX,0.8,(255,255,255),2)
        i = i+1

        cv2.imshow('LIVE', img)
        key = cv2.waitKey(1)
        if key == 27:
            cap.release()

cv2.destroyAllWindows()
          break
```

```
    print(output)
    cap.release()
    cv2.destroyAllWindows()
                final_output1       =
st.mode(output)
                                    return
render_template("buttons.html",
final_output=final_output1)

@app.route('/templates/buttons',
methods = ['GET','POST'])
def buttons():
                                    return
render_template("buttons.html"
)

@app.route('/movies/surprise',
methods = ['GET', 'POST'])
def moviesSurprise():
                                    return
render_template("moviesSurpri
se.html")

@app.route('/movies/angry',
methods = ['GET', 'POST'])
def moviesAngry():
                                    return
render_template("moviesAngry.
html")

@app.route('/movies/sad',
methods = ['GET', 'POST'])
def moviesSad():
                                    return
render_template("moviesSad.ht
ml")

@app.route('/movies/disgust',
methods = ['GET', 'POST'])
def moviesDisgust():
                                    return
render_template("moviesDisgus
t.html")

@app.route('/movies/happy',
methods = ['GET', 'POST'])
def moviesHappy():
                                    return
render_template("moviesHappy
.html")
```

```
@app.route('/movies/fear',
methods = ['GET', 'POST'])
def moviesFear():
                                    return
render_template("moviesFear.ht
ml")

@app.route('/movies/neutral',
methods = ['GET', 'POST'])
def moviesNeutral():
                                    return
render_template("moviesNeutra
l.html")

@app.route('/songs/surprise',
methods = ['GET', 'POST'])
def songsSurprise():
                                    return
render_template("songsSurprise
.html")

@app.route('/songs/angry',
methods = ['GET', 'POST'])
def songsAngry():
                                    return
render_template("songsAngry.h
tml")

@app.route('/songs/sad',
methods = ['GET', 'POST'])
def songsSad():
                                    return
render_template("songsSad.htm
l")

@app.route('/songs/disgust',
methods = ['GET', 'POST'])
def songsDisgust():
                                    return
render_template("songsDisgust.
html")

@app.route('/songs/happy',
methods = ['GET', 'POST'])
def songsHappy():
                                    return
render_template("songsHappy.h
tml")
```

```
@app.route('/songs/fear',
methods = ['GET', 'POST'])
def songsFear():
                                return
render_template("songsFear.ht
ml")

@app.route('/songs/neutral',
methods = ['GET', 'POST'])
def songsNeutral():
                                return
render_template("songsSad.htm
l")

@app.route('/templates/join_pa
ge', methods = ['GET', 'POST'])
def join():
                                return
render_template("join_page.htm
l")

if __name__ == "__main__":
    app.run(debug=True)
```

Capture.py

```
import cv2
import argparse
import time
import os
import Update_Model
import glob
import random
import eel
#import winsound

frequency=2500
duration=1000

eel.init('WD_INNOVATIVE')
emotions=["angry", "happy", "sad", "neutral"]
fishface = cv2.face.FisherFaceRecognizer_create()
font = cv2.FONT_HERSHEY_SIMPLEX
'''try:
    fishface.load("model.xml")
except:
    print("No trained model found... --update will create one.")'''

parser=argparse.ArgumentParser(description="Options for emotions based music player(Updating
```

```
the model)")
parser.add_argument("--update", help="Call for taking new images and retraining the model.", action="store_true")
args=parser.parse_args()
facedict={}
video_capture=cv2.VideoCapture(0)
facecascade=cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
def crop(clahe_image, face):
    for (x, y, w, h) in face:
        faceslice=clahe_image[y:y+h, x:x+w]
        faceslice=cv2.resize(faceslice, (350, 350))
        facedict["face%s" %(len(facedict)+1)]=faceslice
    return faceslice

def grab_face():
    ret, frame=video_capture.read()
    #cv2.imshow("Video", frame)
    cv2.imwrite('test.jpg', frame)
    cv2.imwrite("images/main%s.jpg" %count, frame)
    gray=cv2.imread('test.jpg',0)
    #gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    clahe=cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
    clahe_image=clahe.apply(gray)
    return clahe_image

def detect_face():
    clahe_image=grab_face()
        face=facecascade.detectMultiScale(clahe_image, scaleFactor=1.1, minNeighbors=15, minSize=(10, 10), flags=cv2.CASCADE_SCALE_IMAGE)
    if len(face)>=1:
        faceslice=crop(clahe_image, face)
        #return faceslice
    else:
        print("No/Multiple faces detected!!, passing over the frame")

def save_face(emotion):
    print("\n\nLook "+emotion+" untill the timer expires and keep the same emotion for some time.")
    #winsound.Beep(frequency, duration)
    print('\a')


    for i in range(0, 5):
        print(5-i)
        time.sleep(1)

    while len(facedict.keys())<16:
        detect_face()

    for i in facedict.keys():
        path, dirs, files = next(os.walk("dataset/%s" %emotion))
```

```
        file_count = len(files)+1
        cv2.imwrite("dataset/%s/%s.jpg" %(emotion, (file_count)), facedict[i])
    facedict.clear()

def update_model(emotions):
    print("Update mode for model is ready")
    checkForFolders(emotions)

    for i in range(0, len(emotions)):
        save_face(emotions[i])
    print("Collected the images, looking nice! Now updating the model...")
    Update_Model.update(emotions)
    print("Model train successful!!")

def checkForFolders(emotions):
    for emotion in emotions:
        if os.path.exists("dataset/%s" %emotion):
            pass
        else:
            os.makedirs("dataset/%s" %emotion)

def identify_emotions():
    prediction=[]
    confidence=[]

    for i in facedict.keys():
        pred, conf=fishface.predict(facedict[i])
        cv2.imwrite("images/%s.jpg" %i, facedict[i])
        prediction.append(pred)
        confidence.append(conf)
    output=emotions[max(set(prediction), key=prediction.count)]
    print("You seem to be %s" %output)
    facedict.clear()
    return output;
    #songlist=[]
    #songlist=sorted(glob.glob("songs/%s/*" %output))
    #random.shuffle(songlist)
    #os.startfile(songlist[0])
count=0
@eel.expose
def getEmotion():

    count=0
    while True:
        count=count+1
        detect_face()
        if args.update:
            update_model(emotions)
            break
        elif count==10:
```

```
            fishface.read("model.xml")
            return identify_emotions()
            break

#eel.start('main.html', options=web_app_options)
#options={'host':'file', 'port': '//'}

eel.start('main.html')#//WD_INNOVATIVE//main.html')
#, options)
```

Display.py

```
import tkinter as tk
import cv2
from PIL import Image, ImageTk

width, height = 800, 600
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

root = Music_player.Tk()
root.bind('<Escape>', lambda e: root.quit())
lmain = Music_player.Label(root)
lmain.pack()

def show_frame():
    _, frame = cap.read()
    frame = cv2.flip(frame, 1)
    cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
    img = Image.fromarray(cv2image)
    imgtk = ImageTk.PhotoImage(image=img)
    lmain.imgtk = imgtk
    lmain.configure(image=imgtk)
    lmain.after(10, show_frame)

show_frame()
root.mainloop()
```

model.py

```
import os
import cv2
import numpy as np
from keras.preprocessing import image
import warnings
warnings.filterwarnings("ignore")
# rom keras.preprocessing.image import load_img, img_to_array
from keras.models import  load_model
import matplotlib.pyplot as plt
import numpy as np
```

```
import tensorflow as tf
import statistics as st
# load model
model = load_model("final_model.h5")

face_haar_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
i=0
GR_dict={0:(0,255,0),1:(0,0,255)}

model = tf.keras.models.load_model('final_model.h5')
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

output=[]
cap = cv2.VideoCapture(0)

while (i<=50):
    ret, img = cap.read()
    #gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(img,1.05,5)

    for x,y,w,h in faces:

        face_img = img[y:y+h,x:x+w]

        resized = cv2.resize(face_img,(224,224))
        reshaped=resized.reshape(1, 224,224,3)/255
        predictions = model.predict(reshaped)

        # find max indexed array
        max_index = np.argmax(predictions[0])

        emotions = ('angry', 'disgust', 'fear', 'happy', 'sad', 'neutral', 'surprise')
        predicted_emotion = emotions[max_index]
        output.append(predicted_emotion)



        cv2.rectangle(img,(x,y),(x+w,y+h),GR_dict[1],2)
        cv2.rectangle(img,(x,y-40),(x+w,y),GR_dict[1],-1)
                                    cv2.putText(img, predicted_emotion, (x, y-
10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
    i = i+1

    cv2.imshow('LIVE', img)
    key = cv2.waitKey(1)
    if key == 27:
        cap.release()
        cv2.destroyAllWindows()
        break
```

```
print(output)
cap.release()
cv2.destroyAllWindows()

final_output = st.mode(output)
final_output
```

haarcascade_frontalface_default.xml

updatemodel.py

```
import numpy as np
import glob
import random
import cv2

fishface=cv2.face.FisherFaceRecognizer_create()
data={}

def update(emotions):
    run_recognizer(emotions)
    print("Saving model...")
    fishface.save("model.xml")
    print("Model saved!!")

def make_sets(emotions):
    training_data=[]
    training_label=[]

    for emotion in emotions:
        training=training=sorted(glob.glob("dataset/%s/*" %emotion))
        for item in training:
            image=cv2.imread(item)
            gray=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            training_data.append(gray)
            training_label.append(emotions.index(emotion))
    return training_data, training_label

def run_recognizer(emotions):
    training_data, training_label=make_sets(emotions)
    print("Training model...")
    print("The size of the dataset is "+str(len(training_data))+" images")
    fishface.train(training_data, np.asarray(training_label))
```

Emotion_Detection.ipynb

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from keras.layers import Flatten, Dense
from keras.models import Model
from keras.preprocessing.image import ImageDataGenerator , img_to_array, load_img
from tensorflow.keras.applications import MobileNetV2
from keras.applications.mobilenet import MobileNet, preprocess_input
from keras.losses import categorical_crossentropy
train_datagen = ImageDataGenerator(
    zoom_range = 0.2,
    shear_range = 0.2,
    horizontal_flip=True,
    rescale = 1./255
)

train_data = train_datagen.flow_from_directory(directory= "/content/images/images/train",
                                               target_size=(224,224),
                                               batch_size=32,
                                  )

train_data.class_indices
val_datagen = ImageDataGenerator(rescale = 1./255 )

val_data = val_datagen.flow_from_directory(directory= "/content/images/images/validation",
                                           target_size=(224,224),
                                           batch_size=32)
val_data.class_indices
t_img , label = train_data.next()

def plotImages(img_arr, label):
  count = 0
  for im, l in zip(img_arr,label) :
    plt.imshow(im)
    plt.title(im.shape)
    plt.axis = False
    plt.show()

    count += 1
    if count == 10:
      break

# function call to plot the images
plotImages(t_img, label)
```

o 4.2IMPLEMENTATION

Create a directory folder as following and copy relevant pieces of code into the required parts: -
EmotionBasedMusicPlayer

```
├── app.py                  # Main Streamlit app (or Flask if needed)
├── requirements.txt        # Python dependencies
│
├── data/
│   └── emotions.csv        # Dataset for training/testing models (if any)
│
├── src/
│   ├── emotion_model.pkl   # Trained model for emotion detection (if ML is involved)
│   ├── emotion_detector.py # Script for detecting emotion from input
│   ├── playlist_generator.py # Logic for selecting and playing songs
│   └── audio_utils.py      # Audio processing helpers (optional)
│
├── static/
│   ├── css/
│   │   └── style.css       # Custom styles for frontend
│   └── img/
│       ├── happy.png
│       ├── sad.png
│       ├── angry.png
│       ├── neutral.png
│       └── background.jpg
│
├── templates/
│   ├── base.html           # Base layout
│   ├── home.html           # Landing page
│   ├── register.html       # User registration page
│   ├── login.html          # User login page
│   ├── detect_emotion.html # Emotion input page (via text or facial recognition)
│   ├── recommend.html      # Recommended playlist page
│   ├── feedback.html       # User feedback form
│   └── admin_dashboard.html # Admin analytics dashboard (optional)
│
└── README.md               # Project overview and instructions
```

Installing Python Packages

Open your terminal and run the following command to install all required packages:

pip install streamlit pandas numpy scikit-learn librosa opencv-python matplotlib

spotipy requests

If you're using a requirements.txt, you can also run:

pip install -r requirements.txt

Optional: If you're using VS Code, make sure your virtual environment is activated before installing.

Running the Application

1. Navigate to the project root directory:

   cd emotion-based-music-player

2. Start the Flask server:

   streamlit run app.py

3. Open your browser and go to:

   http://localhost:850

5.TESTING

5.1INTRODUCTION TO TESTING:

Testing is an essential phase in software development that verifies the correct functionality, usability, performance, and reliability of the application under varying conditions. It ensures the final system meets the desired specifications and behaves consistently under expected and unexpected inputs. Effective testing minimizes the risk of system failures and enhances the user experience by identifying bugs or inconsistencies early in the lifecycle.

In this project, Emotion-Based Music Player, testing was crucial to ensure that personalized music recommendations, real-time emotion detection, playlist management, and user interaction all functioned as intended. The goal was to validate that each module—from emotion analysis to song recommendation and playlist playback—performed seamlessly and accurately delivered an engaging, personalized music experience.

The test cases were crafted to verify the major functionalities and flow of the application, including:

- Correct transition between pages (home, upload, playback, recommendation, etc.)
- Accurate detection of user emotions via facial recognition or text input
- Generation of music recommendations aligned with the detected emotion
- Smooth audio playback and playlist management
- Real-time updates of recommendations as emotions change
- Reliable storage and retrieval of any user preferences (if applicable)
- Proper integration of models and recommendation logic

5.2TEST CASES:

S.NO	Test Case Name	Input	Expected Output	Actual Output	Remarks
1	Webcam Access Granted	User grants webcam permission	Video stream starts	Video stream starts	Pass
2	Webcam Access Denied	User denies webcam permission	Show error or prompt again	Error shown	Pass
3	Detect Happy Emotion	Frame with user smiling	Emotion: Happy	Emotion: Happy	Pass
4	Detect Emotion - No Face Present	Frame with background only	“No face detected" message shown	"No face detected" shown	Pass
5	Recommend Music - Unknown Emotion	Frame with unreadable expression	Default or empty playlist shown	Default playlist shown	Pass
6	Play Song	User clicks Play button	Music starts playing	Music plays	Pass
7	Emotion Detection with Multiple Faces	Group photo input	Error or handle only one face	One face detected	Pass
8	Stop Song	User clicks Pause button	Music stops playing	Music stops	Pass
9	Detect neutral Emotion	Frame with user keeping neutral face	Emotion:Neutral	Emotion:Neutral	Pass

Table 5.1 Test Cases of Rhythm Restore

6.RESULTS

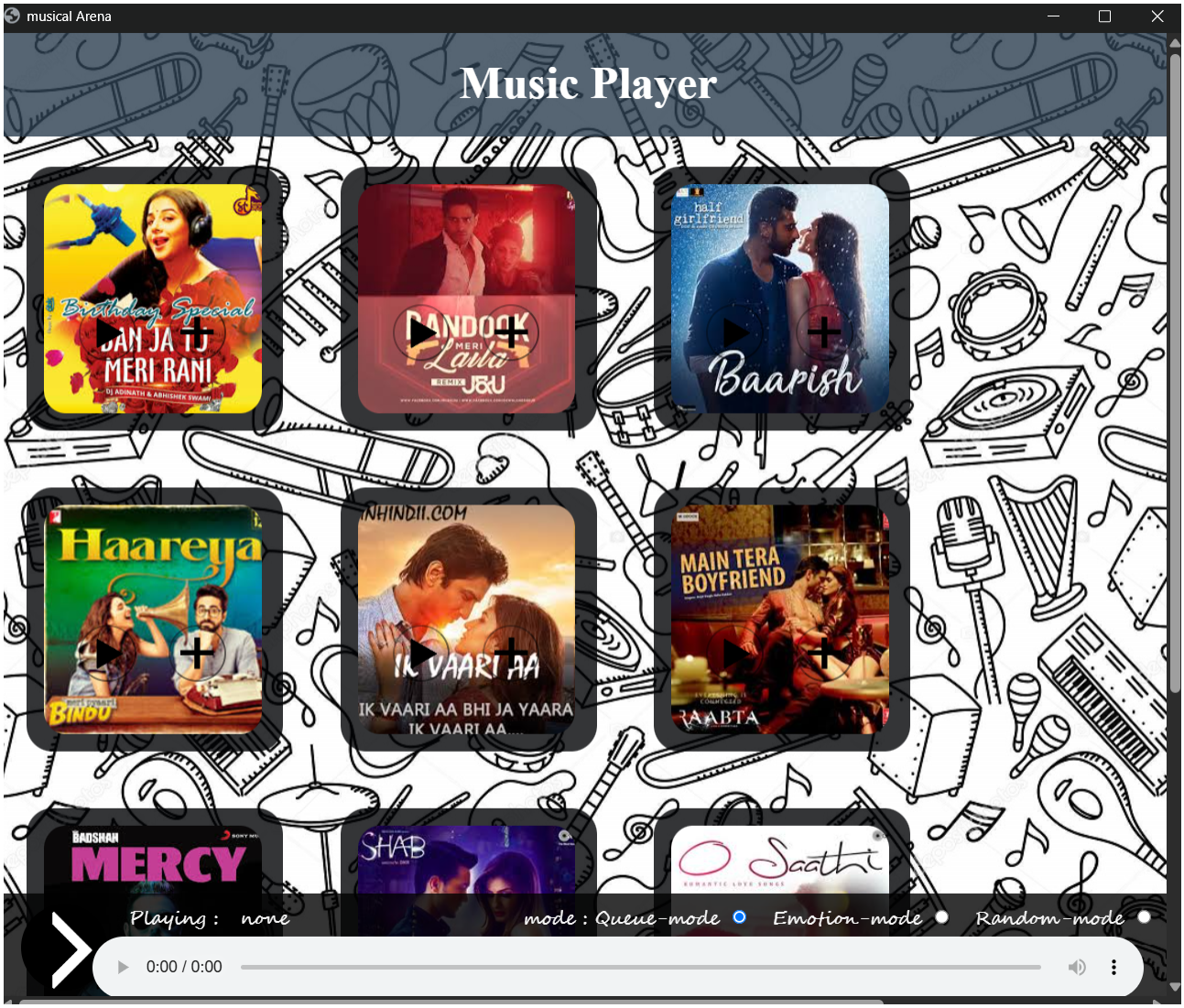


Fig. 6.1 Initial page

Fig. 6.1 shows the interface of the music player ,we can select the modes from the three modes

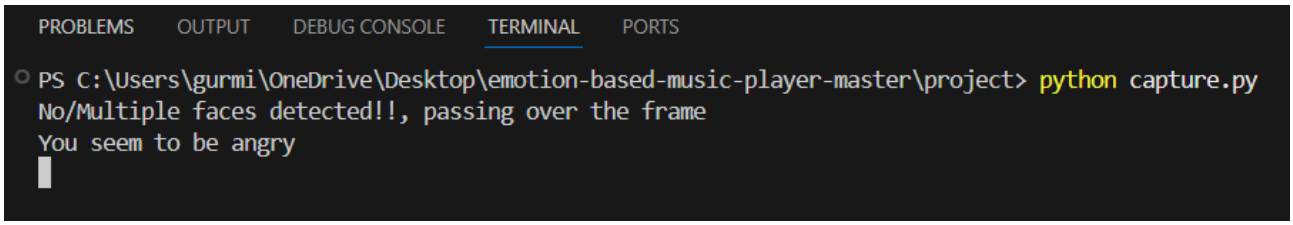


Fig. 6.2 Angry emotion

Fig. 6.2 displays that the user emotion is angry and the user will be recommended songs based on the angry emotion

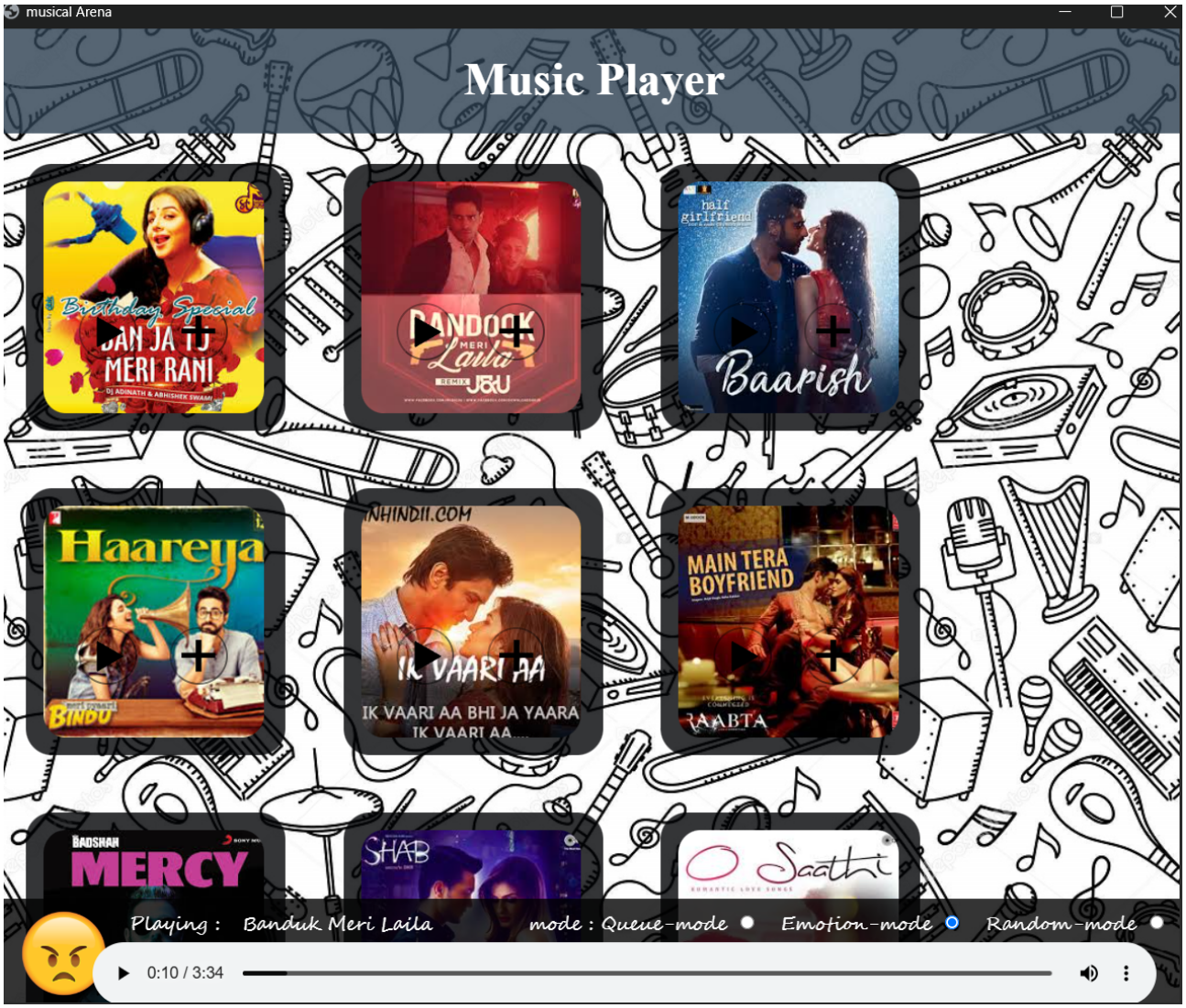


Fig. 6.3 output for the angry emotion

Fig. 6.3 represents the songs that are recommended for the user based on his angry emotion.

```
PS C:\Users\gurmi\OneDrive\Desktop\emotion-based-music-player-master\project> python capture.py
You seem to be happy
```

Fig. 6.4 Happy emotion

Fig. 6.4 represents that the user emotion is happy

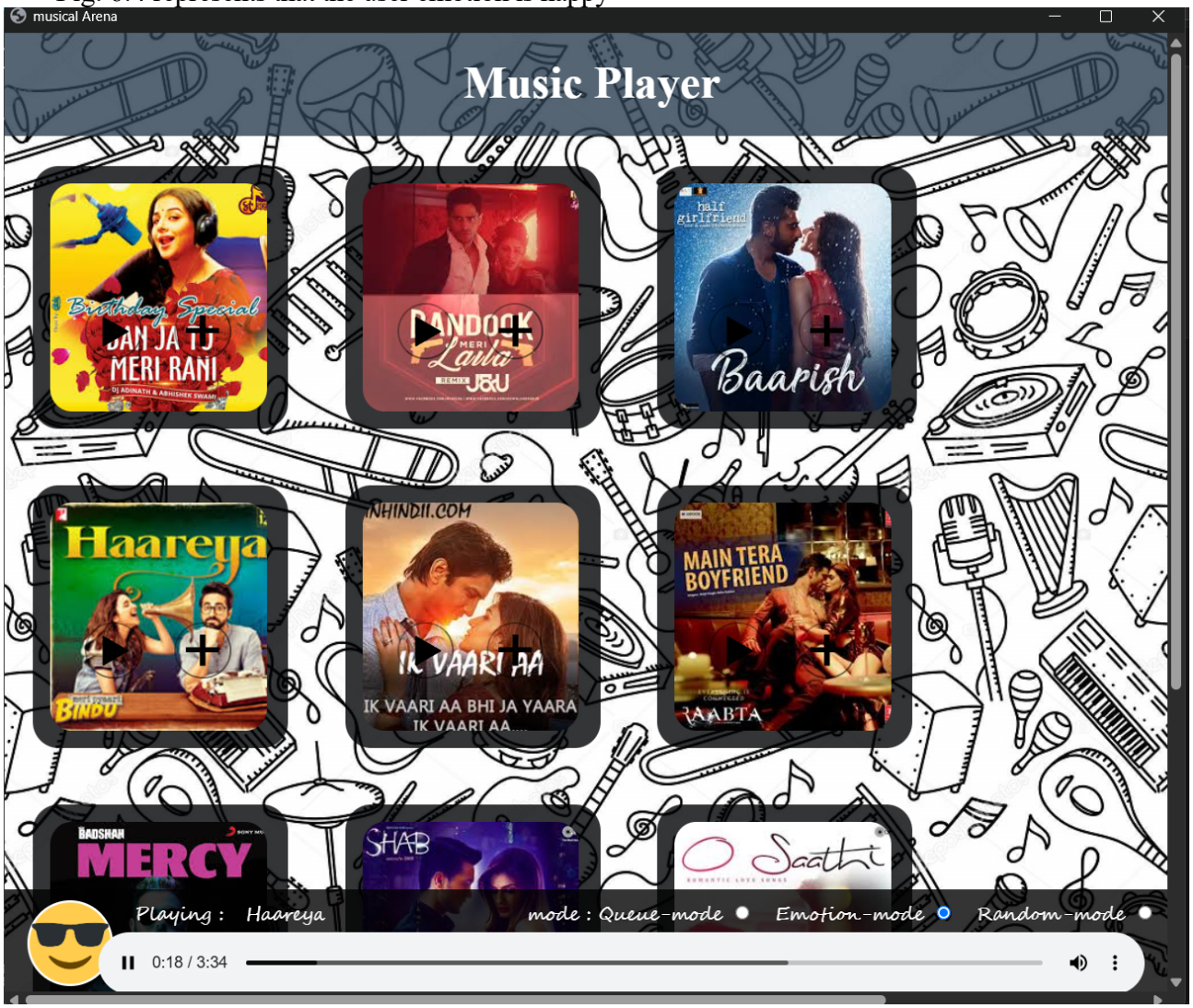


Fig. 6.5 output for the happy emotion

Fig. 6.5 shows the recommended songs for the happy emotion

```
PS C:\Users\gurmi\OneDrive\Desktop\emotion-based-music-player-master\project> python capture.py
You seem to be sad
```

Fig 6.6 sad emotion

Fig. 6.6 represents the emotion that user is sad.

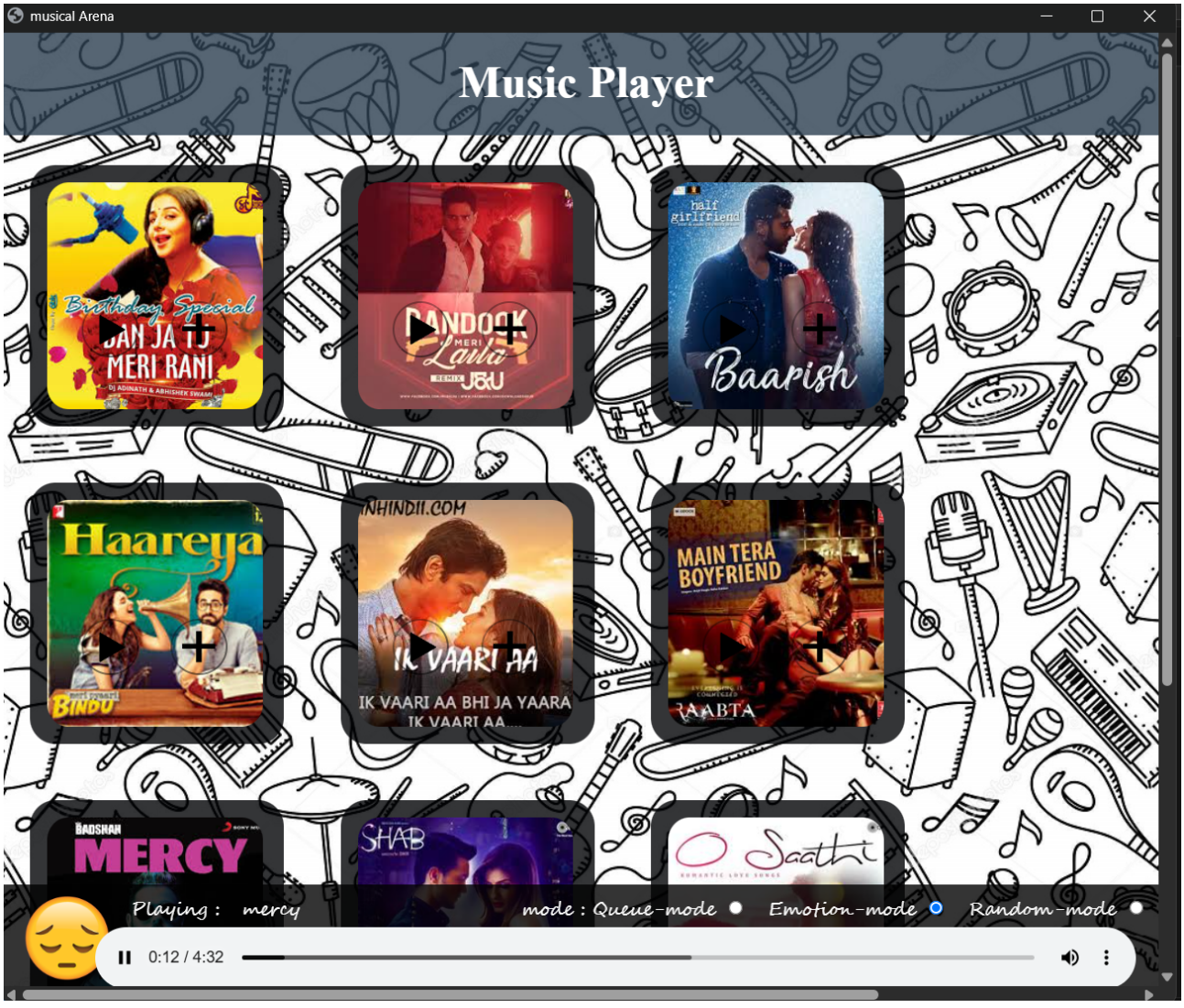


Fig6.7Outputforthesademotion

7.CONCLUSION AND FUTURE ENHANCEMENTS

7.1CONCLUSION

The Emotion-Based Music Player successfully bridges the gap between technology and human emotion by using facial expression analysis to curate personalized music playlists. This project demonstrates the potential of combining computer vision and music streaming to enhance user experiences. By leveraging facial recognition technology, it intelligently adapts to the user's emotional state, offering a dynamic and engaging way to discover music. With further development and integration, this system can evolve into a comprehensive platform for emotional well-being and entertainment, offering users a unique, interactive, and personalized music journey.

The Emotion-Based Music Player successfully integrates real-time emotion detection and personalized music recommendation, offering users a dynamic, interactive experience. By analyzing facial expressions or text inputs, the system adapts to users' moods and provides tailored song selections, enhancing user satisfaction and engagement. This innovative approach demonstrates the potential of emotion-aware technology in music recommendation systems and paves the way for further improvements in personalization and user experience.

7.2FUTURE ENHANCEMENTS

Although the current version of Emotion based music player is effective, several enhancements can improve its scalability, accuracy, and user engagement:

- **User Authentication and Profiles:** Allow users to register/login.
- **Offline Mode**: Cache playlists locally for emotion categories for use without

internet.

- **Voice and Gesture Control:** Add support for voice commands (e.g., “Play something happy”).
- **Mobile App Version**: Build cross-platform apps using **Flutter** or **React Native**.

REFERENCES

[1]. Jaladhi Sam Joel, B. Ernest Thompson, Steve Renny Thomas, T. Revanth Kumar, Shajin Prince, D. Bini, "Emotion-Based Music Recommendation System using Deep Learning Model," *IEEE Access*, 2023. https://ieeexplore.ieee.org/document/10134389

[2] Brijesh Bakariya, Arshdeep Singh, Harmanpreet Singh, Pankaj Raju, Rohit Rajpoot, Krishna Kumar Mohbey, "Emotion-Based Music Recommendation System," *Volume 15*, pp. 641–658, 2024. https://link.springer.com/article/10.1007/s12530-023-09506-z

[3] Harshanil, Mandar Karmakar, Rashmi Astogi, "Music Mood Prediction and Playlist Recommendation based on Facial Expressions," *Article No. 92*, pp. 1–6. https://doi.org/10.1145/3647444.3647919

[4] Tina Babu, Rekha R. Nair, Geetha A., "Emotion-Aware Music Recommendation System: Enhancing User Experience Through Real-Time Emotional Context." https://doi.org/10.48550/arXiv.2311.10796

[5] Abhinav Kumar, Priyanshu Mishra, Rajat Singh, "Affective Music Recommendation Based on Emotion Recognition from Facial Expressions," *Procedia Computer Science*, Volume 199, pp. 284–291, 2022. https://doi.org/10.1016/j.procs.2022.01.035

[6] D. Ghosal, N. Majumder, S. Poria, E. Cambria, A. Gelbukh, "Emotion Detection from Text and Speech: A Survey," *IEEE Transactions on Affective Computing*, Volume 13, No. 2, pp. 905–920, 2022. https://ieeexplore.ieee.org/document/9451456

[7] Sneha Chaudhari, Saurabh Chouhan, "Facial Emotion Recognition using Deep CNN for Music Recommendation," *Materials Today: Proceedings*, Volume 62, Part 7, pp. 4592–4596, 2023. https://doi.org/10.1016/j.matpr.2022.10.199

[8] Prachi P. Patil, K. K. Bhoyar, "An Efficient Emotion Recognition System for Real-Time Music Recommendation Using CNN and Transfer Learning," *Procedia Computer Science*, Volume 218, pp. 1840–1845, 2023. https://doi.org/10.1016/j.procs.2023.01.388

[9] Linlin Shen, Yuqian Zhao, "Real-Time Emotion-Based Music Recommender Using Hybrid Deep Learning Models," *Multimedia Tools and Applications*, Volume 82, pp. 23199–23216, 2023. https://doi.org/10.1007/s11042-022-13585-9

[10] M. Albadra, F. Al-Wesabi, A. Yahya, "Context-Aware Music Recommendation Based on Emotion and Listening Habits Using Deep Learning," *Journal of Intelligent & Fuzzy Systems*, Volume 45, No. 4, pp. 5259–5270, 2023. https://doi.org/10.3233/JIFS-223369