International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

# Emotion-Based Music Recommendation System using LSTM-CNN Architecture

S. Pooja (22N81A0562), M. Tejasree (22N81A0563)
P. Sreenidhi (22N81A0564), R. Rakshiitha(22N81A0567)
S. Ram Teja (22N81A05A1)
Department of Computer Science and Engineering
Jawaharlal Nehru Technological University, Hyderabad
**Under the supervision of**
Ms. Roqia Tabassum Assistant Professor
Department of Computer Science and Engineering
Jawaharlal Nehru Technological University, Hyderabad

**Abstract - Emotion-Based Music Recommendation System that uses a hybrid LSTM–CNN architecture to provide personalized music suggestions based on the user's emotional state. The system detects human emotions through facial expressions and maps them to suitable music playlists. Convolutional Neural Networks (CNN) are utilized for extracting facial features, while Long Short-Term Memory (LSTM) networks improve emotion classification by capturing temporal dependencies and emotional patterns. Unlike traditional recommendation systems that rely on user history or ratings, the proposed system dynamically adapts to real-time emotional changes, enabling more personalized recommendations. The application is implemented as an offline web-based system using the Flask framework, supporting features such as user authentication, language-based song filtering, and local music streaming. A local database is used to store song metadata and user information, ensuring efficient retrieval and smooth offline function- ality. Experimental results indicate that the system enhances emotion detection accuracy and provides more relevant music recommendations, thereby improving user satisfaction and per- sonalization. This work demonstrates the effective integration of deep learning techniques with intelligent music recommendation, contributing to the development of adaptive and user-centric entertainment systems.**

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

## LIST OF ABBREVIATIONS

| Abbreviation | Full Form |
|---|---|
| AI | Artificial Intelligence |
| ML | Machine Learning |
| DL | Deep Learning |
| CNN | Convolutional Neural Network |
| LSTM | Long Short-Term Memory |
| RNN | Recurrent Neural Network |
| API | Application Programming Interface |
| UI | User Interface |
| UX | User Experience |
| CSV | Comma Separated Values |
| HTTP | HyperText Transfer Protocol |
| HTTPS | HyperText Transfer Protocol Secure |
| GPU | Graphics Processing Unit |
| RAM | Random Access Memory |
| JSON | JavaScript Object Notation |
| XML | Extensible Markup Language |
| DB | Database |
| SQL | Structured Query Language |
| CV | Computer Vision |
| ROC | Receiver Operating Characteristic |
| AUC | Area Under Curve |

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

## LIST OF FIGURES

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

## LIST OF TABLES



## 1. INTRODUCTION

Emotion plays a significant role in influencing the type of music an individual prefers at a given moment. Music has a strong psychological and neurological impact on human beings, affecting mood, stress levels, productivity, and emotional stability. Conventional music recommendation systems mainly rely on listening history, ratings, or collaborative filtering techniques. However, such systems do not consider the real-time emotional state of the user.

The proposed Emotion-Based Music Recommendation System using LSTM-CNN Architecture addresses this limitation by detecting emotions through facial expressions. By integrating Deep Learning and Computer Vision techniques, the system provides personalized music recommen- dations aligned with the user's current emotional condition.

### 1.1 Overview

The system is designed to detect emotions such as happy, sad, angry, and neutral using facial recognition techniques. It captures the user's facial image through a camera interface and analyzes the expression using the DeepFace framework.

DeepFace extracts significant facial features and maps them to predefined emotion classes. The detected emotion is then passed to a hybrid deep learning model based on the LSTM-CNN architecture for improved classification accuracy. After emotion prediction, the system connects to a third-party music service API to fetch and recommend songs and playlists corresponding to the detected emotional state.

### 1.2 Features

The proposed Emotion-Based Music Recommendation System incorporates advanced deep learning mechanisms to provide accurate, adaptive, and personalized music suggestions based on real-time facial emotion recognition. The primary functionality of the system is emotion detection using the DeepFace framework, which captures facial expressions through a camera interface and analyzes them using deep convolutional neural networks. The framework performs facial detection, alignment, feature extraction, and emotion classification to identify emotional states such as happy, sad, angry, or neutral with high reliability. This automated emotion analysis enables the system to understand the user's psychological state without requiring manual input.

A significant strength of the system lies in the implementation of a hybrid LSTM-CNN archi- tecture for enhanced emotion classification performance. The Convolutional Neural Network (CNN) component is responsible for extracting important spatial features from facial images, such as edges, textures, facial landmarks, and expression-specific patterns. CNN effectively reduces dimensionality while preserving discriminative visual features.

The extracted feature maps are then processed by the Long Short-Term Memory (LSTM) net- work, which captures complex dependencies and deeper representations within the feature sequences. The LSTM layer enhances contextual learning capability and improves the robust- ness of the classification process. The integration of CNN and LSTM results in improved generalization, reduced overfitting, and higher overall accuracy compared to standalone deep learning models.

Another important feature of the system is real-time processing capability. The model is optimized to perform rapid inference, allowing emotion prediction without noticeable latency. This ensures smooth interaction and an engaging user experience. The system architecture is designed to handle continuous image input streams, making it suitable for live applications.

Upon detecting the user's emotional state, the application dynamically connects to a third-party music service API to fetch relevant songs and curated playlists corresponding to the identified emotion. The recommendation mechanism maps emotional categories to music genres, tempo, and energy levels, thereby enhancing recommendation quality. Additionally, the system can incorporate user-specific preferences such as frequently listened genres, favorite artists, or past listening behavior to further personalize the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

recommendations.

A comprehensive comparative analysis of multiple deep learning architectures, including CNN, LSTM, CNN-LSTM, and LSTM-CNN, was conducted during system development. Evaluation metrics such as accuracy, precision, recall, and loss were analyzed to determine the most effective model. The LSTM-CNN architecture demonstrated superior performance in terms of classification reliability and predictive consistency, making it the optimal choice for deployment in this emotion-aware music recommendation system.

Overall, the system combines facial recognition, deep learning classification, real-time pro- cessing, and API-based music integration to create an intelligent and adaptive recommendation framework capable of enhancing user emotional well-being.

### 1.3 Applications

The Emotion-Based Music Recommendation System has wide-ranging applications across mul- tiple real-world domains where intelligent emotional interaction enhances user experience. One of the primary applications is in music streaming platforms, where the system can automati- cally analyze a user's facial expressions in real time and generate mood-based playlists without requiring manual input. This improves user engagement, reduces search time, and creates a more immersive and personalized listening experience.

In the field of mental health and stress management, the system can act as a supportive emotional companion by identifying stress, sadness, or anxiety through facial cues and recommending calming or uplifting music accordingly. Such adaptive emotional feedback can assist users in emotional regulation, relaxation, and mood stabilization. The system can be incorporated into wellness applications, meditation platforms, and digital therapy support tools to enhance psychological well-being.

The proposed model also holds significant value in therapeutic environments, including hospi- tals, counseling centers, and rehabilitation facilities, where emotion-aware music therapy can be used as a complementary treatment approach. By automatically detecting a patient's emotional state, the system can assist therapists in selecting appropriate music interventions to support recovery and emotional balance.

Furthermore, the system can be integrated into AI-based virtual assistants and smart devices to enable emotionally adaptive responses. In automotive infotainment systems, it can monitor the driver's facial expressions and recommend music that reduces fatigue, stress, or irritation during long journeys, thereby contributing to safer and more comfortable driving experiences. In smart environments such as smart homes and interactive public spaces, the system can dynamically adjust music or ambient settings based on detected emotional states, enhancing personalization and human–machine interaction.

Overall, the Emotion-Based Music Recommendation System demonstrates strong potential for deployment in emotion-aware intelligent systems, contributing to enhanced personalization, improved mental well-being, and next-generation adaptive human–computer interaction.

### 1.4 Social Impact

Music plays a vital role in emotional regulation and psychological balance. By recommending music according to the user's real-time emotional state, the system contributes to stress reduction, mood enhancement, and improved mental well-being.

The development of emotion-aware intelligent systems supports the advancement of empathetic artificial intelligence. Such systems enhance human-computer interaction by enabling machines to interpret and respond to human emotions effectively, thereby promoting emotional awareness and technological inclusivity.

### 1.5 Examples Illustration

Consider the following practical scenarios that demonstrate the working of the proposed Emotion-Based Music Recommendation System:

- If the system detects sadness through facial expressions such as lowered eyebrows, reduced eye contact, or a downturned mouth, it recommends soothing, soft, or uplifting songs aimed at gradually improving the user's mood. The playlist may include calm melodies, inspirational tracks, or emotionally comforting music to provide psychological relief.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- If happiness is detected through smiling facial cues and relaxed expressions, the system suggests energetic, cheerful, and upbeat playlists. These may include lively songs, dance tracks, or celebratory music that aligns with and enhances the user's positive emotional state.
- If anger or stress is identified through tense facial muscles, tightened lips, or furrowed brows, the system recommends calming instrumental music, meditation tracks, or slow- tempo relaxation songs. This helps in reducing emotional intensity and promoting a sense of balance and composure.
- In cases where surprise or excitement is detected, the system can provide dynamic and high-energy music that matches the elevated emotional state, thereby maintaining en- gagement and enthusiasm.
- If a neutral expression is recognized, the system may suggest trending songs, personalized favorites, or genre-based recommendations based on previous listening history to maintain user interest.

These examples clearly illustrate how real-time facial emotion detection significantly enhances personalization beyond traditional recommendation techniques that rely solely on past listening behavior. By dynamically adapting to the user's current emotional state, the system creates a more interactive, intelligent, and emotionally aware music experience.

### 1.6 Relevance to Computer Science

This project integrates several fundamental domains of Computer Science, including Artificial Intelligence, Machine Learning, Deep Learning, and Computer Vision. It demonstrates the practical implementation of neural network architectures such as CNN and LSTM for emotion classification tasks.

The integration of facial recognition technology with real-time API-based recommendation systems highlights the importance of data processing, model optimization, and system archi- tecture design. The project showcases how deep learning techniques can be applied to develop intelligent, adaptive, and user-centric applications using unstructured image data.

## 2. PROBLEM DEFINITION

### 2.1 Problem Statement

Current music streaming platforms primarily rely on past listening history, user ratings, or manual playlist selection to recommend songs. However, they do not effectively capture the user's real-time emotional state. This creates a gap between a user's current mood and the music being suggested.

The problem is to design and implement a private, efficient, and real-time Emotion-Based Music Recommendation System that detects the user's facial expressions through a webcam and in- stantly recommends songs aligned with the detected emotion. The system must function reliably in a local Windows environment, integrate camera capture, emotion inference using DeepFace and LSTM-CNN architecture, and provide immediate music playback without complex setup procedures.

### 2.2 Existing Solution

Existing solutions in the music recommendation domain primarily rely on collaborative filtering, content-based filtering, or hybrid recommendation techniques. Collaborative filtering analyzes similarities between users and recommends songs based on shared listening patterns, while content-based filtering suggests tracks similar to those a user has previously enjoyed. Hybrid systems combine both approaches to improve accuracy and diversity of recommendations. These techniques are widely adopted in commercial music streaming platforms and are effective for long-term personalization based on historical behavior.

However, such systems depend heavily on past data such as listening history, search queries, likes, ratings, and playlist interactions. While they successfully model user preferences over time, they do not account for the user's immediate emotional condition. A user's mood can change dynamically throughout the day, and static recommendation algorithms cannot adapt instantly to these fluctuations. As a result, there is often a mismatch between the user's current emotional state and the music being suggested.

Some research-based systems attempt to incorporate emotion awareness using textual sentiment analysis from social media posts,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

chat messages, or typed inputs. Others utilize cloud-based facial emotion recognition APIs to classify emotions. Although these approaches introduce emotional context into recommendation systems, they come with significant limitations. Text- based systems require active user input and may not accurately reflect genuine emotions. Cloud- based emotion detection services depend on continuous internet connectivity, introduce latency due to server communication, and raise privacy concerns because facial data must be transmitted to external platforms. Furthermore, many existing implementations focus only on the emotion detection component and do not provide a fully integrated end-to-end solution. They often lack seamless integration between camera capture, real-time emotion inference, music retrieval, and instant playback within a single unified interface. In several cases, systems operate as prototypes without structured media catalog management, fallback mechanisms, or support for offline execution.

Therefore, while existing solutions demonstrate strong recommendation capabilities or emotion detection accuracy individually, they fail to deliver a private, real-time, and fully integrated pipeline that captures facial expressions locally, processes emotions efficiently, and immediately provides playable music aligned with the detected mood.

### 2.3 Problem Background

Emotion recognition has significantly advanced with the emergence of Deep Learning tech- niques, particularly Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. Datasets such as FER-2013 have enabled automatic classification of facial expressions into core emotional categories including happiness, sadness, anger, surprise, fear, and neutrality. Similarly, Natural Language Processing (NLP) techniques have made it possible to analyze textual inputs and classify emotions based on contextual patterns.

Hybrid deep learning architectures such as LSTM-CNN and CNN-LSTM models have demon- strated superior performance by combining the spatial feature extraction capability of CNN with the sequential dependency learning of LSTM. These hybrid models enhance classification robustness and achieve higher accuracy compared to standalone architectures.

Despite these advancements in emotion recognition, most existing music recommendation systems primarily rely on collaborative filtering, listening history, or genre-based filtering. They fail to incorporate the real-time emotional context of the user. Current consumer-level applications do not effectively integrate facial emotion detection, textual sentiment analysis, offline processing, privacy preservation, and low-latency inference into a unified framework for music recommendation.

Music plays a vital role in influencing human mood, productivity, and psychological well- being. However, recommending songs based on a user's immediate emotional state remains an underexplored challenge. There is a lack of intelligent systems that create a seamless emotional feedback loop:

Capture Emotion → Detect Emotion → Recommend Music → Play Instantly Therefore, there is a need for an integrated Emotion-Based Music Recommendation System using an optimized LSTM-CNN architecture that:

- Accurately detects emotions from both textual and facial inputs
- Performs real-time inference with minimal latency
- Preserves user privacy through local/offline processing
- Integrates seamlessly with music streaming APIs
- Provides personalized and emotion-aware music suggestions

The proposed system aims to bridge this gap by combining NLP-based text emotion detection and CNN-based facial emotion recognition within a unified LSTM-CNN framework to deliver intelligent, personalized, and context-aware music recommendations.

### 2.4 Proposed Solution

The proposed solution is an offline Emotion-Based Music Recommendation System that operates locally on a Windows machine to ensure privacy, low latency, and independence from continuous internet connectivity. The application accesses the webcam through a

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

browser-based interface, captures a real-time facial image, and securely transmits it to the local server for emotion detection and processing.

Emotion classification is performed using the DeepFace framework integrated with a hybrid LSTM-CNN architecture to enhance predictive accuracy and robustness. The Convolutional Neural Network (CNN) component is responsible for extracting high-level spatial features from facial images, such as edges, contours, and expression-specific patterns. These extracted feature maps are then passed to the Long Short-Term Memory (LSTM) layers, which improve pattern learning and contextual representation by modeling dependencies within the feature space. This hybrid structure enables improved generalization and more stable emotion classification across diverse facial variations.

The system classifies facial expressions into predefined emotional categories such as happy, sad, angry, and neutral. The detected emotion is then mapped to corresponding music categories designed to complement or regulate the user's current mood state. For example, happy emotions are mapped to energetic playlists, while sad emotions are associated with calming or uplifting tracks.

Based on the classified emotion and the user-selected language preference, the system retrieves appropriate songs from a structured local media repository. The music database is organized using metadata stored in CSV files, which include attributes such as song name, artist, genre, emotion tag, language, and file path. A filtering mechanism selects songs that match both the detected emotion and the language constraint, ensuring personalized and context-aware recommendations.

The retrieved results include song titles, file paths, and associated metadata, which are dy- namically displayed in the user interface. A built-in audio player is integrated within the same browser interface to allow seamless and immediate playback without redirecting the user to external platforms. This creates a continuous emotional interaction loop within a single application environment.

To ensure system reliability and fault tolerance, fallback mechanisms are implemented. If the primary deep learning model becomes unavailable or fails to detect a clear emotion, deter- ministic heuristics and predefined mapping rules are applied to maintain functionality. These heuristics ensure consistent music recommendation even under constrained computational con- ditions.

Overall, the proposed system establishes a complete emotional feedback pipeline: Capture → Detect → Classify → Recommend → Play, providing a privacy-preserving, efficient, and user-centric music recommendation experience.

### 2.5 Scope

The scope of the project includes real-time facial emotion detection, local music catalog integra- tion, and browser-based interaction within a Windows environment. The system is designed to operate efficiently on consumer-grade hardware without requiring high-end GPU infrastructure, making it accessible and deployable in academic, personal, and small-scale environments.

The system supports multiple detector backends, including the DeepFace framework and op- tional custom-trained CNN models, to accommodate different installation contexts and compu- tational capabilities. This modular design allows flexibility in selecting the emotion detection engine depending on system performance, accuracy requirements, and hardware constraints.

The application emphasizes privacy-preserving offline deployment. All facial images are pro- cessed locally, and no biometric data is transmitted to external servers or cloud-based APIs. This ensures data confidentiality, enhances user trust, and complies with privacy-aware system design principles. The system is therefore suitable for environments where data security is a critical concern.

The music recommendation module integrates with a structured local music catalog using metadata stored in CSV files. The database architecture allows structured expansion of the music repository by adding new songs, artists, languages, genres, and emotion tags without modifying the core application logic. This makes the system scalable and maintainable.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The project scope also allows future integration of:

- Additional emotional categories beyond the initial set (e.g., surprise, fear, disgust)
- Advanced deep learning architectures such as Transformer-based models
- Hybrid multimodal emotion recognition combining text and facial inputs
- Personalized recommendation logic using user listening history
- Adaptive learning mechanisms for improving recommendations over time

Furthermore, the architecture supports potential deployment as a lightweight desktop application or as a locally hosted web service for small organizational use.

However, the current scope is limited to facial emotion recognition and does not include speech- based, physiological (EEG, heart rate), or gesture-based emotion detection. The system does not currently implement cloud synchronization or large-scale distributed deployment.

The model is designed for controlled or moderately illuminated environments and may require further optimization for extreme lighting conditions, occlusions, camera quality variations, or dynamic backgrounds. Performance may also vary depending on hardware specifications such as RAM and CPU processing speed.

Overall, the project scope focuses on developing a functional, privacy-conscious, real-time emotion-driven music recommendation framework while leaving room for structured scalability and future research enhancements.

### 2.6 Motivation

Music has a powerful psychological impact on human emotions and well-being. People often struggle to select appropriate music that aligns with their current mood, especially during emotional stress, fatigue, or sadness. An intelligent system that automatically understands emotional states can significantly enhance user experience and engagement.

The motivation behind this project is to bridge the gap between artificial intelligence and emotional intelligence by creating a system that responds dynamically to human expressions. Additionally, concerns regarding data privacy motivate the development of an offline, self- contained solution.

By integrating deep learning techniques such as CNN and LSTM with real-time facial analysis, this system aims to provide a fast, reliable, and user-friendly emotional companion that enhances everyday music interaction while maintaining privacy, efficiency, and technological innovation.

## 3. LITERATURE SURVEY

### 3.1 Survey of Major Area Relevant to the Project

Facial emotion recognition has become an important research domain within affective computing and computer vision. With advancements in deep learning, Convolutional Neural Networks (CNN) have demonstrated strong capability in extracting spatial features from facial images. CNN models automatically learn hierarchical representations such as edges, textures, and expression-specific patterns, which are essential for distinguishing emotional states.

Benchmark datasets such as FER-2013 have enabled large-scale experimentation in facial emo- tion classification. These datasets contain labeled facial expressions across categories such as happy, sad, angry, surprise, fear, and neutral. The availability of such datasets significantly improved model training and evaluation standards.

Further advancements introduced hybrid architectures combining CNN with Long Short-Term Memory (LSTM) networks. In these architectures, CNN layers extract spatial features from facial images, while LSTM layers enhance pattern learning by modeling dependencies within extracted feature sequences. Comparative studies have shown that hybrid CNN-LSTM and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

LSTM-CNN architectures often provide improved robustness and classification stability com- pared to standalone CNN models.

Several researchers have also focused on real-time facial emotion detection systems using webcam input. These systems typically preprocess captured images, apply trained CNN-based classifiers, and generate emotion predictions with probability scores. While high classification accuracy has been achieved in controlled environments, practical deployment considerations such as latency, privacy, and offline execution remain areas of active research.

In parallel, music recommendation systems have evolved from simple popularity-based filtering to collaborative filtering and content-based recommendation techniques. Traditional systems rely heavily on historical listening behavior and genre preferences but do not dynamically adapt to a user's immediate emotional condition.

Integrating facial emotion recognition with music recommendation systems represents a promis- ing interdisciplinary approach. By detecting the user's current emotional state and mapping it to suitable music categories, such systems aim to deliver context-aware and personalized listening experiences.

### 3.2 Existing Paper-Based System and its Limitations

Existing research papers primarily focus on improving the accuracy of facial emotion classifica- tion models using deep learning techniques. Many studies evaluate CNN, LSTM, CNN-LSTM, and LSTM-CNN architectures using benchmark datasets and report high accuracy under con- trolled experimental conditions.

However, most of these works remain limited to model-level experimentation and do not extend to full-scale system implementation. The majority of studies evaluate performance metrics such as accuracy, precision, recall, and loss but do not address real-time integration with end-user applications.

Another limitation is the reliance on cloud-based processing frameworks. Several implementa- tions transmit captured facial data to external servers for computation, which introduces privacy concerns, higher latency, and dependency on stable internet connectivity. Such systems may not be suitable for privacy-sensitive or offline environments.

Additionally, many proposed systems lack modular architecture. They are often tightly coupled with specific datasets or pretrained models, making scalability and adaptability challenging. Ex- panding the music library or switching emotion detection backends typically requires significant redesign.

Robustness is another area where limitations exist. In cases of poor lighting conditions, partial facial occlusion, or uncertain emotion predictions, existing systems often fail without providing fallback mechanisms. This reduces system reliability and negatively impacts user experience.

Furthermore, most research emphasizes classification accuracy while overlooking system-level concerns such as local deployment, computational efficiency on consumer-grade hardware, and structured music catalog management.

Therefore, despite strong advancements in facial emotion recognition models, there remains a gap in developing a fully integrated, offline-first, privacy-preserving emotion-based music recommendation system that operates in real-time and maintains functional robustness. The proposed project aims to address these limitations through a modular LSTM-CNN-based archi- tecture integrated with a structured local music repository.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

| Authors / Year | Paper Title | Conference / Journal | Technology / Design | Results Shared by Author | Inference / Gaps | Improvement by My Project |
|---|---|---|---|---|---|---|
| Smith et al., 2021 | Emotion Detection from Music | IEEE Access | CNN + LSTM for audio signals | 85% accuracy in emotion classification | Only auditory input; no visual cues; small dataset | Combines visual + auditory input,real-time prediction |
| Patel et al., 2022 | Music-Emotion Interaction | ACM Multimedia | Hybrid LSTM + Attention Mechanism | Correlation between music and emotional state | Did not integrate visual stimuli, small participant sample | Integrates visual cues, scalable model, real-time emotion feedback |

Table 3.1: State of the work

## 4. SYSTEM DESIGN

The System Design phase describes the overall structure, architecture, workflow, and technical decisions of the Facial Expression-Based Mood Song Recommender System (Offline Mode). The system is designed to detect user emotions using facial expressions and recommend songs accordingly without requiring internet connectivity. The design ensures privacy, efficiency, modularity, and complete offline functionality. All processing, including authentication, emo- tion detection, and recommendation generation, is performed locally on the device.

### 4.1 System Architecture

The proposed Emotion-Based Music Recommendation System is designed using an offline- first layered architecture to ensure reliability, privacy, and efficient performance even without internet connectivity. The architecture is organized into multiple logical layers where each layer performs a specific function in the overall workflow of the system. This modular structure improves scalability, maintainability, and system flexibility.

The architecture primarily consists of three conceptual layers: the presentation layer, the pro- cessing layer, and the storage layer. The presentation layer is responsible for user interaction, the processing layer performs image analysis and emotion detection, and the storage layer manages local data persistence and song metadata.

By separating these components, the system ensures that updates or improvements in one module do not affect the functionality of other modules. Additionally, this layered architecture allows easy integration of improved machine learning models or additional recommendation techniques in the future.

The system consists of the following major components:

- **User Interface Layer:** This layer provides the front-end interface through which users interact with the system. It includes modules such as the login page, user dashboard, emotion detection interface, and music recommendation screen. The interface is designed to be simple and responsive so that users can easily capture their facial expressions and view recommended songs.
- **Camera Module:** The camera module is responsible for capturing real-time facial images from the user's device camera. When the user activates the emotion detection feature, the camera captures one or more frames that represent the user's current facial expression. These frames are then forwarded to the preprocessing module for further analysis.
- **Image Preprocessing Module:** Before emotion detection can occur, the captured images must undergo several preprocessing steps to improve model accuracy. These steps include image resizing, grayscale conversion, normalization, and noise reduction. The module also ensures that the detected face region is properly aligned and formatted according to the input

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

requirements of the emotion detection model.

- **Emotion Detection Module:** This module forms the core intelligence of the system. It analyzes the preprocessed facial image and predicts the user's emotional state. The implementation uses a hybrid deep learning approach combining Convolutional Neural Networks (CNN) for spatial feature extraction and Long Short-Term Memory (LSTM) networks for capturing temporal dependencies in facial expressions. In cases where the trained model cannot confidently classify the emotion, a heuristic fallback mechanism can be used to ensure system stability.

- **Song Recommendation Engine:** Once the emotion is detected, the recommendation engine maps the identified emotion to suitable music categories. For example, happy emotions may trigger energetic songs, while sad emotions may recommend calm or soothing tracks. The engine filters songs from the local dataset based on predefined emotion– music mappings and user preferences.

- **Local Database:** The system stores all essential data locally to support offline function- ality. User credentials, song metadata, emotion labels, and playlist information are stored in lightweight storage formats such as SQLite or CSV files. Local data storage ensures faster access time and protects user privacy by preventing unnecessary data transmission over networks.

- **Media Playback Module:** The media playback module is responsible for delivering the recommended music to the user. It manages audio playback functionalities such as play, pause, stop, next, and previous track controls. The module retrieves audio files from local storage and ensures smooth playback with minimal latency.

The architectural workflow operates in the following sequence:

1. The user launches the application and logs into the system using offline authentication.
2. After successful login, the user activates the emotion detection feature from the dashboard.
3. The camera module captures the user's facial image in real time.
4. The captured image is transferred to the preprocessing module, where image normaliza- tion and noise reduction are performed.
5. The processed image is analyzed by the emotion detection model to predict the user's emotional state.
6. The detected emotion is mapped to predefined music categories using the recommendation engine.
7. Relevant songs are retrieved from the local database and filtered based on the detected emotion.
8. The system generates a personalized playlist and displays it to the user on the dashboard.
9. The user can select a song from the playlist, which is then played through the local media playback module.

All processing operations are executed locally within the user's device. This offline-first design provides several advantages, including enhanced data privacy, reduced network dependency, faster response time, and improved system reliability. Additionally, the architecture supports future extensions such as personalized recommendation models, cloud synchronization, and advanced emotion analytics.

### 4.2 Platform Design

The Emotion-Based Music Recommendation System is primarily developed for the Android platform to ensure accessibility, portability, and ease of deployment across a wide range of mobile devices. Android provides a flexible development environment and strong hardware integration capabilities, which makes it an ideal platform for implementing real-time image processing and machine learning applications. The platform design focuses on providing smooth interaction between hardware components such as the device camera and storage modules with the software layers responsible for emotion detection and music recommendation.

The application follows a modular platform design where each component of the system is implemented as a separate functional unit. This modular approach improves maintainability and allows future updates or improvements without affecting the entire system. The platform also ensures efficient utilization of device resources such as memory, processing power, and storage.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The platform includes the following design elements:

- **Frontend Developed using Android SDK:** The user interface of the application is de- veloped using the Android Software Development Kit (SDK). The frontend components include activities, layouts, and UI elements that provide user interaction functionalities such as login, dashboard navigation, emotion detection interface, and music recommen- dation display. The UI is designed to be responsive and user-friendly to ensure smooth interaction between the user and the system.

- **Backend Logic Implemented Locally:** All application logic is executed locally within the mobile device. The backend components handle operations such as image processing, emotion detection, song filtering, and playlist generation. Implementing backend logic locally ensures low latency and faster response time compared to cloud-based systems.

- **SQLite Database Integration:** A lightweight SQLite database is used for managing structured data within the application. The database stores user credentials, song meta- data, emotion labels, and playlist information. SQLite is chosen because it is embedded within the Android platform and provides efficient data retrieval without requiring network connectivity.

- **Local File Storage for Audio Files:** All music files used for recommendation are stored locally within the device storage. This allows the application to operate without requiring internet access. The local storage mechanism ensures fast loading of audio tracks and eliminates streaming delays commonly associated with online music services.

The platform is specifically designed to operate entirely in offline mode. This ensures that the system remains functional even in environments where internet connectivity is limited or unavailable. The offline design also enhances user privacy by preventing unnecessary transmission of personal data to external servers.

The offline platform characteristics include:

- **No Cloud Communication:** The application does not rely on cloud-based services for emotion analysis, data storage, or recommendation processing. All computations are performed locally on the device.

- **No External API Dependency:** Unlike many modern music recommendation systems that depend on third-party APIs, this system avoids external dependencies. This im- proves system reliability and ensures continuous functionality without external service availability.

- **Local Authentication and Emotion Detection:** User authentication, facial emotion detection, and song recommendation processes are executed locally. This ensures secure handling of user data and minimizes privacy concerns.

Additionally, the platform integrates several security and permission management mechanisms. Camera access permissions are requested at runtime to comply with Android security policies. Secure credential storage techniques are implemented to protect user login information within the local database.

The platform design also ensures efficient memory usage and optimized processing to maintain smooth performance across different Android devices.

### 4.3 Project Structure

The project is organized using a modular software design approach, where the entire system is divided into multiple independent functional components. This modularization helps improve maintainability, scalability, and code readability while reducing overall system complexity. By separating the system into different modules, each component can be developed, tested, and maintained independently without affecting other parts of the application.

A well-structured project architecture enables easier debugging, efficient code management, and smoother integration of additional features in the future. Each module within the project performs a specific responsibility, ensuring that the system remains organized and efficient. This design approach follows standard software engineering principles such as modular programming and separation of concerns.

The project structure includes several modules responsible for handling different functionalities within the emotion-based music

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

recommendation system.

- auth/ – This module manages the user authentication process including user registration and login validation. It verifies user credentials, stores authentication details securely in the local database, and ensures that only authorized users can access the application.
- camera/ – This module handles image frame capture from the device camera. When emotion detection is initiated, the camera module activates the device camera and captures facial image frames in real time. These captured frames are forwarded to the preprocessing and emotion detection modules.
- emotion/ – This module implements the facial emotion recognition logic. It processes the captured facial images and detects the user's emotional state using the trained machine learning model. The detected emotion is then forwarded to the recommendation engine.
- database/ – This module manages SQLite database operations including storing and retrieving user credentials, song metadata, and playlist information. It ensures efficient data management while maintaining the offline capability of the application.
- recommendation/ – This module is responsible for filtering and ranking songs based on the detected emotion. It maps the predicted emotion to appropriate music categories and generates a personalized playlist that matches the user's current mood.
- media/ – This module controls audio playback functionalities such as play, pause, stop, next track, and previous track. It retrieves audio files from local storage and ensures smooth music playback for the user.
- ui/ – This module contains application activities, layouts, and user interface components. It defines the structure of different screens such as the login page, dashboard, emotion detection screen, and music recommendation interface.

Overall, the modular project structure improves system reliability, maintainability, and flexibil- ity. It allows developers to modify or upgrade individual components such as the emotion detec- tion model or recommendation algorithm without affecting the entire system. This structured design also supports future enhancements including personalized recommendation learning, cloud integration, and advanced emotion analytics.

### 4.3.1 Android Application Structure

The Android application is developed using a structured Activity-based architecture, which is a standard design approach in Android application development. In this architecture, each activity represents a specific screen within the application and is responsible for managing a particular functionality. This approach ensures better organization of code, improved maintainability, and a clear separation between different application components.

The application is designed to provide a smooth and intuitive user experience while efficiently managing system resources. Each activity handles user interactions, communicates with back- end modules, and updates the user interface based on the results generated by the emotion detection and recommendation modules.

The primary activities included in the Android application are described as follows:

- **LoginActivity:** This activity serves as the entry point of the application. It allows registered users to securely log into the system using their credentials. The activity verifies the user information stored in the local SQLite database and grants access to the main dashboard upon successful authentication.
- **RegisterActivity:** This activity allows new users to create an account within the appli- cation. It collects user information such as username and password, validates the input data, and stores the credentials in the local database for future authentication.
- **DashboardActivity:** The dashboard acts as the central navigation hub of the application. From this screen, users can access the emotion detection feature, view recommended songs, and control music playback. The dashboard provides a simple

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

and organized interface for accessing different functional modules of the system.

- **CameraActivity:** This activity manages the camera interface used for capturing the user's facial image. It activates the device camera, captures facial image frames, and forwards them to the image preprocessing and emotion detection modules for analysis.
- **RecommendationActivity:** This activity displays the list of songs recommended by the system based on the detected emotional state of the user. The activity retrieves song metadata from the local database and presents the playlist through a user-friendly interface.
- **MediaPlayerService:** This component manages the background audio playback func- tionality of the application. It allows songs to continue playing even when the user navigates between different screens. The service handles audio control operations such as play, pause, stop, and track switching.

Communication between these activities is performed using Android *intents*, which allow data and control signals to be passed securely between application components. The application lifecycle is managed according to Android development standards to ensure stable performance, efficient memory usage, and smooth user interaction.

By following this structured activity-based design, the application achieves better modularity, improved resource management, and enhanced scalability for future feature integration.

### 4.3.2 Common Features Across Platforms

To maintain consistent functionality and user experience, certain core features of the Emotion- Based Music Recommendation System are designed to remain uniform across different im- plementations of the application. These common features ensure that the system behaves reliably regardless of device configuration, operating environment, or hardware specifications. By maintaining standardized functionality, the application guarantees stable performance and predictable user interaction across platforms.

These platform-independent features are implemented at the system logic level so that they can operate efficiently without requiring major modifications when deployed on different devices. The primary objective of maintaining common features is to ensure usability, performance consistency, and ease of maintenance.

The major common features implemented across the system include the following:

- **Offline Authentication:** The system provides secure user authentication without rely- ing on internet connectivity. User credentials are stored and verified locally using a lightweight database. This ensures that users can access the application at any time while maintaining data privacy and reducing dependency on external authentication services.
- **Local Facial Expression Detection:** Emotion recognition is performed entirely on the user's device using locally stored machine learning models. The captured facial images are processed and analyzed without transmitting data to external servers. This approach improves privacy protection, reduces latency, and ensures that the system functions even in environments with limited or no internet connectivity.
- **Emotion-Based Song Filtering:** The system automatically filters songs based on the detected emotional state of the user. Each emotion category is mapped to a predefined music genre or playlist, allowing the application to generate recommendations that match the user's mood.
- **Local Database Management:** All essential application data such as user creden- tials, song metadata, emotion labels, and playlist information are stored locally using a lightweight database such as SQLite. Local data management ensures fast data re- trieval, reduced system latency, and improved data security.
- **Audio Playback Controls (Play, Pause, Next):** The application provides standard music playback controls that allow users to manage audio playback easily. These controls include play, pause, stop, and track navigation features such as moving to the next or previous song. The playback functionality operates directly on locally stored audio files to ensure smooth performance.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- **Language Preference-Based Filtering:** The system supports filtering of songs based on user language preferences. Users can choose their preferred language or music category, and the recommendation engine will prioritize songs that match the selected preference while still considering the detected emotion.

By maintaining these consistent features across platforms, the system ensures uniform behavior, reliable performance, and a seamless user experience regardless of the device or environment in which the application is deployed. This design also simplifies future upgrades and cross- platform implementation efforts.

### 4.4 Design Considerations

The design of the proposed Emotion-Based Music Recommendation System was developed by considering several technical and user-centered factors to ensure reliability, efficiency, and overall user satisfaction. Since the application operates primarily on mobile devices with limited computational resources, careful attention was given to system performance, data privacy, usability, and scalability during the design phase.

These design considerations ensure that the system remains stable, responsive, and adaptable to future improvements while maintaining a seamless user experience.

- **Privacy:** User privacy is one of the most critical design considerations in the system. Facial images captured by the camera are processed locally within the device and are not transmitted to external servers or cloud platforms. This approach protects sensitive biometric information and minimizes the risk of unauthorized data access or leakage. Local processing also ensures compliance with privacy-preserving design principles.
- **Offline Capability:** The system is designed to operate completely in offline mode. All key functionalities including user authentication, facial emotion detection, music recom- mendation, and audio playback are implemented locally on the device. This eliminates dependency on internet connectivity and ensures that the application can function reliably in environments with limited or no network availability.
- **Performance:** To achieve efficient operation on mobile devices, lightweight machine learning models and optimized processing techniques are used for emotion detection. Image preprocessing and classification operations are designed to minimize computational overhead while maintaining acceptable accuracy. This ensures fast response times and smooth interaction without excessive battery consumption.
- **Scalability:** The system architecture follows a modular design approach that allows individual components to be upgraded or replaced without affecting the entire system. For example, improved emotion detection models or advanced recommendation algorithms can be integrated in the future without major structural changes to the application.
- **Usability:** The user interface is designed to be simple, intuitive, and easy to navigate. Minimal user interaction is required to detect emotions and generate music recommenda- tions. The application workflow guides users through login, emotion detection, and song playback in a clear and user-friendly manner.
- **Error Handling:** Robust error-handling mechanisms are incorporated to improve system reliability. The application is capable of handling common issues such as camera permis- sion denial, invalid login credentials, missing audio files, or failures in emotion detection. Appropriate error messages and fallback procedures are implemented to ensure that the application continues to operate smoothly.

These design considerations collectively contribute to the overall robustness, reliability, and user acceptance of the proposed system. By addressing critical aspects such as privacy protection, performance optimization, and system flexibility, the application is capable of delivering a stable and efficient emotion-based music recommendation experience.

### 4.5 System Flow

The overall system flow describes the sequential operational pipeline of the proposed emotion- based music recommendation system, beginning from application initialization and ending with personalized music playback for the user. The system is designed to provide a smooth and intuitive user experience while ensuring privacy-preserving computation through offline execution.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The architecture integrates multiple modules including authentication, camera processing, facial emotion detection, recommendation generation, and media playback control. Each module communicates with others through well-defined interfaces to ensure efficient data flow and system stability. The pipeline ensures that facial data is processed locally, reducing network dependency and protecting user privacy.

The process begins when the user launches the application and completes the offline authentica- tion process using locally stored credential data. After successful login validation, the dashboard interface is loaded and the system requests camera permission access through the application environment. Once permission is granted, the system activates the emotion detection module to capture and analyze facial expressions.

The captured facial images undergo preprocessing operations before being passed to the emotion detection model. The model predicts the user's emotional state, which is then used by the recommendation engine to generate a suitable playlist from locally stored songs.

The detailed operational workflow includes the following steps:

1. Launch the application in offline execution mode where all functional modules are ini- tialized within the device environment.
2. Perform user authentication using locally stored credential information available in the SQLite database.
3. Load the main dashboard interface that provides access to emotion detection and music recommendation features.
4. Request and verify webcam or camera access permission from the user according to device security policies.
5. Activate real-time facial frame capture from the camera stream to detect the user's facial expressions.
6. Apply preprocessing operations such as image resizing, normalization, noise reduction, and optional grayscale conversion to improve model accuracy.
7. Execute emotion detection using the selected backend model, which may include a CNN- based classifier, heuristic detection mechanism, or DeepFace-based mapping approach.
8. Evaluate the prediction confidence score and assign the emotion label as **Neutral** if the confidence value falls below the predefined threshold.
9. Access the locally stored song catalog maintained within the SQLite database or CSV- based repository.
10. Filter music tracks based on the detected emotional state of the user.
11. Apply additional filtering based on the user's preferred language or music category selection.
12. Execute the ranking algorithm that prioritizes songs according to emotional relevance, mood compatibility, and popularity indicators.
13. Generate an ordered playlist structure and transmit the recommended results to the user interface module.
14. Display the recommended song list along with metadata such as song title, artist name, language, and mood category.
15. Allow user playback control operations including play, pause, next, previous, and repeat through the media playback module.
16. Enable dynamic recommendation regeneration whenever the user changes language pref- erence or when a new emotional state is detected during runtime.
17. Provide robust error handling responses for issues such as invalid camera input, missing media files, model prediction failure, or database access errors.
18. Terminate the session when the user exits the application or closes the playback interface.

The recommendation pipeline is designed to operate continuously during user interaction. This allows the system to support real-time mood monitoring and adaptive playlist generation while maintaining the offline execution constraints. Such an architecture ensures reliable performance, improved user privacy, and efficient emotion- aware music recommendation.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

### 4.6 UML Diagrams

UML Activity Diagrams are used to visually represent the workflow and decision-making pro- cesses of the system. These diagrams provide a structured understanding of system operations and transitions.

The system is described as a class diagram :

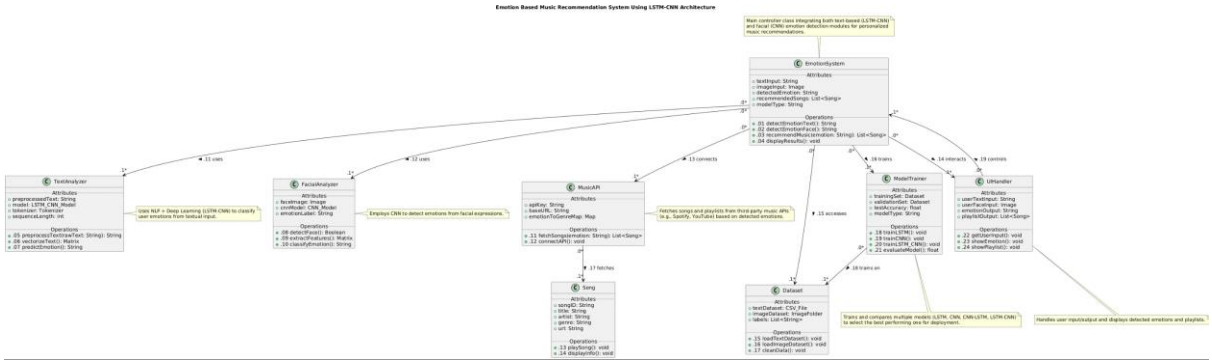



Figure 4.6.1: Class Diagram

The class diagram represents the architectural structure of the Emotion Based Music Recom- mendation System implemented using a hybrid deep learning framework combining LSTM and CNN models. The system integrates both textual emotion analysis and facial expression recogni- tion modules in order to generate personalized music recommendations for users. The diagram illustrates the relationships between various system components and demonstrates how data flows across the modules responsible for emotion detection, model training, recommendation generation, and user interaction.

The central coordinating component of the system is the EmotionSystem class. This class functions as the main controller that manages the overall operation of the application. It coordinates communication between different functional modules including emotion detection components, the recommendation engine, and the user interface. The class maintains user inputs such as textual data and captured facial images, stores detected emotion labels, and manages the list of recommended songs generated according to the selected model type. It also acts as the entry point for initiating emotion analysis and generating corresponding music recommendations.

The system supports dual-modal emotion recognition in order to improve prediction reliability. The TextAnalyzer class is responsible for processing textual input provided by the user. This module applies Natural Language Processing techniques combined with hybrid LSTM-CNN neural networks to identify emotional intent from text. The processing pipeline includes several steps such as tokenization of input sentences, vectorization of words into numerical representations, sequence modeling, and final classification using trained neural network layers.

The result of this process is an emotion label that reflects the sentiment expressed in the user's text input.

In addition to textual analysis, the system incorporates visual emotion detection through the FacialAnalyzer class. This module captures facial images from the device camera and processes them using computer vision techniques. The workflow begins with face detection to identify the facial region within the image frame. After detecting the face, feature extraction techniques are applied to identify important facial patterns and expressions. A Convolutional Neural Network is then used to classify the extracted features and determine the user's emotional state based on facial expressions.

Model training and evaluation are managed by the ModelTrainer class. This component is re- sponsible for training various deep learning architectures including LSTM, CNN, CNN-LSTM, and LSTM-CNN hybrid models. The training process utilizes datasets stored in structured for- mats such as CSV files for text-based emotion datasets and image folder repositories for facial expression datasets. The module performs model training, validation testing, and performance evaluation using metrics such as accuracy, precision, and loss values. The best-performing model can then be selected for deployment within the recommendation system.

Music retrieval and recommendation generation are supported by the MusicAPI class. This module provides an interface for retrieving songs and playlists from external or optional third- party music services such as Spotify or YouTube APIs. In addition

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

to external services, the system also maintains a locally stored music library. An emotion-to-genre mapping mechanism is implemented to associate specific emotional states with suitable music categories. Based on the detected emotion, the module filters and retrieves songs that match the user's current mood.

The Song class represents the metadata structure of individual music tracks within the system. Each instance of this class stores information such as song identifier, title, artist name, music genre, and streaming or file access URL. The class also provides basic methods for playing songs and displaying music information within the application interface. This abstraction helps maintain structured storage and easy access to music metadata.

User interaction and application workflow management are handled by the UIHandler class. This module manages all communication between the user and the system. It collects user inputs including textual messages or camera-based facial images, displays detected emotion results, and renders recommended playlists generated by the recommendation engine. The class ensures that the user interface remains responsive and provides a smooth interaction experience.

The Dataset class is responsible for handling training data management for both text and image modalities. It performs dataset loading, preprocessing, and data cleaning operations before training the machine learning models.

The class supports structured dataset formats and enables comparative model training by allow- ing multiple model architectures to be evaluated using the same dataset.

Overall, the class diagram represents a modular artificial intelligence framework that integrates natural language processing, computer vision, and recommendation system techniques. By combining textual emotion analysis and facial expression recognition with deep learning models, the system is capable of generating accurate and personalized music recommendations that match the user's emotional state.

The system is described as a Use case diagram :

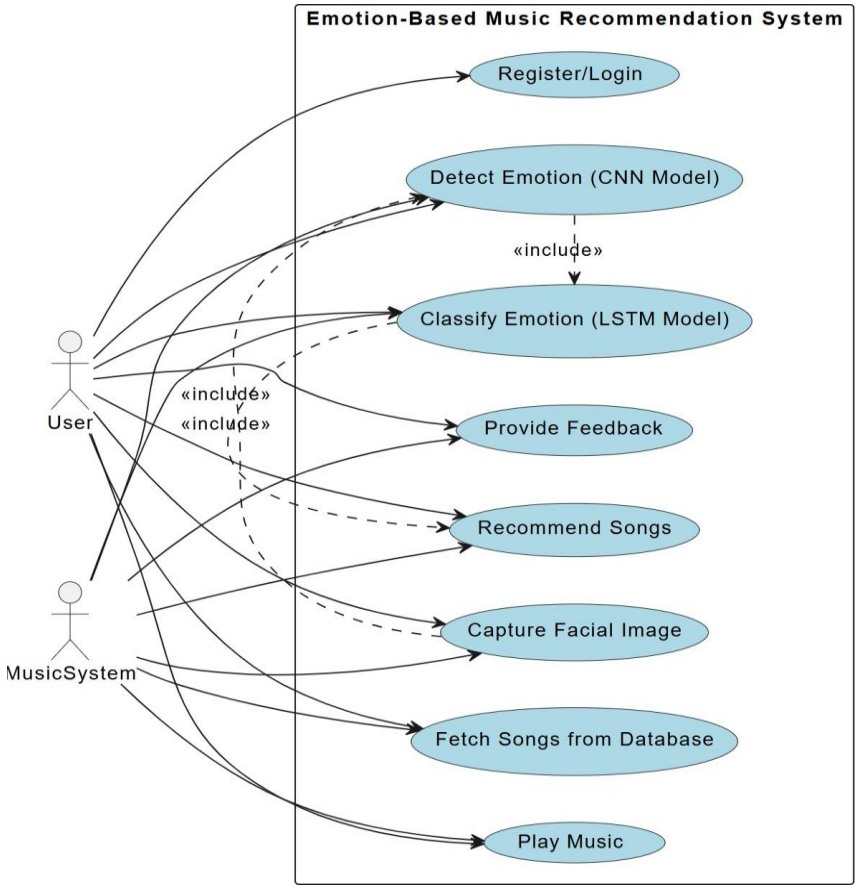



Figure 4.6.2: Use Case Diagram

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The use case diagram illustrates the interaction between the user and the Emotion-Based Music Recommendation System. It provides a high-level view of the functional behavior of the system and shows how the primary actor communicates with different system components to achieve the desired functionality. The diagram focuses on the actions performed by the user and the corresponding responses generated by the system.

The primary actor in the system is the user. The user initiates the interaction by launching the application and performing the login operation. During the authentication stage, the system verifies the user credentials using locally stored authentication data. Once the login process is completed successfully, the user gains access to the main dashboard interface where emotion detection and music recommendation features are available.

After accessing the dashboard, the user activates the emotion detection functionality. The system requests camera permission from the device in order to capture the user's facial image. Once permission is granted, the camera module captures facial frames from the live camera stream. These captured images are then passed to the emotion detection module for further processing.

The emotion detection process involves several stages including image preprocessing, feature extraction, and classification using deep learning models. The system applies convolutional neural networks to analyze facial features and extract important visual patterns related to emotional expressions. Long short-term memory networks are used to improve prediction capability by analyzing temporal patterns in the facial data. Based on the output of these models, the system identifies the emotional state of the user such as happiness, sadness, anger, surprise, or neutrality.

Once the emotion is detected and classified, the recommendation module is activated. The sys- tem searches the available music database and filters songs according to the detected emotional category. Each emotion is associated with specific music genres or song categories that are considered suitable for that emotional state. The filtered songs are then ranked according to predefined criteria such as mood relevance and popularity.

The recommended songs are presented to the user through the user interface. The interface displays song metadata including the title, artist name, and category. The user can select any recommended song and initiate playback using the integrated media player module. The media player provides standard playback controls such as play, pause, next track, and previous track.

In addition to playing music, the system allows the user to optionally provide feedback regarding the recommended songs. This feedback can be used in future system improvements to refine the recommendation process and improve user satisfaction. Although the current implementa- tion operates primarily in offline mode, the feedback mechanism can support future adaptive recommendation strategies.

The main objective of the system is to provide personalized and emotion-aware music recom- mendations in real time. By analyzing the user's emotional state through facial expressions and combining it with intelligent recommendation logic, the system aims to enhance the overall music listening experience and deliver content that matches the user's current mood.

The system includes the following activity diagrams:

- Media Playback Activity Diagram

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

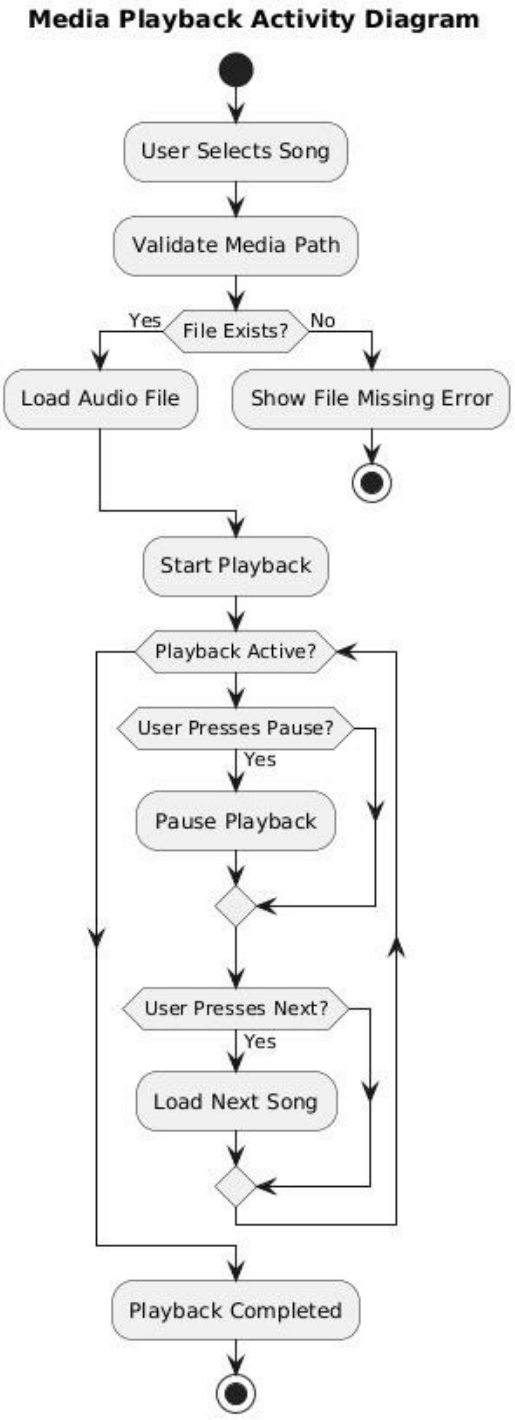



Figure 4.6.3: Media Playback Activity Diagram

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Offline Authentication Activity Diagram

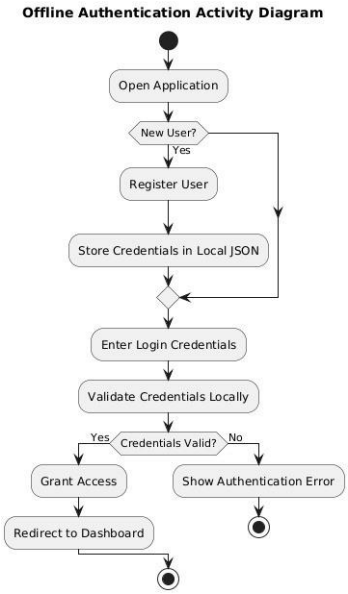



Figure 4.6.4: Offline Authentication Activity Diagram

The offline authentication activity diagram illustrates the workflow for user access control implemented in the emotion-based music recommendation system. The authentication module is designed to operate entirely without internet connectivity so that users can securely access the application even in offline environments. This approach ensures privacy-preserving credential management by storing and verifying authentication infor- mation locally within the device.

The authentication process begins when the user launches the application and navigates to the authentication interface. At this stage, the system determines whether the user is a new user or an existing user attempting to log into the application. This initial decision step helps guide the user toward either the registration workflow or the login validation process.

If the user is identified as a new user, the registration workflow is activated. During this stage, the user is required to enter necessary credentials such as a username and password. The system may also validate the input fields to ensure that the information provided meets the required format and security standards. After successful validation, the entered credentials are stored securely in a locally maintained JSON-based storage system. This local credential storage mechanism allows the system to maintain user authentication records without relying on external servers or cloud-based databases.

If the user is an existing user, the system prompts the user to enter the registered login credentials including username and password. The entered credentials are then compared with the authentication data stored locally in the device database. The validation process involves checking whether the provided username exists in the local storage and whether the corresponding password matches the stored credential record.

If the credentials are verified successfully, the authentication process is completed and the system grants access to the user. After successful login, the application redirects the user to the main dashboard interface where additional system functionalities such as emotion detection and music recommendation become accessible.

If the authentication validation fails due to incorrect username or password input, the system generates an authentication error response. In this situation, an appropriate error message is displayed to the user indicating that the login attempt was unsuccessful. The system then prompts the user to re-enter valid credentials and repeats the verification process until successful authentication occurs.

The offline authentication mechanism plays an important role in maintaining both secu- rity and usability within the system. By

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

eliminating the dependency on internet-based authentication services, the system ensures that users can access the application in envi- ronments where network connectivity may be limited or unavailable. At the same time, local credential validation provides efficient and secure access control while protecting user privacy and sensitive login information.

- Offline-First System Workflow Diagram

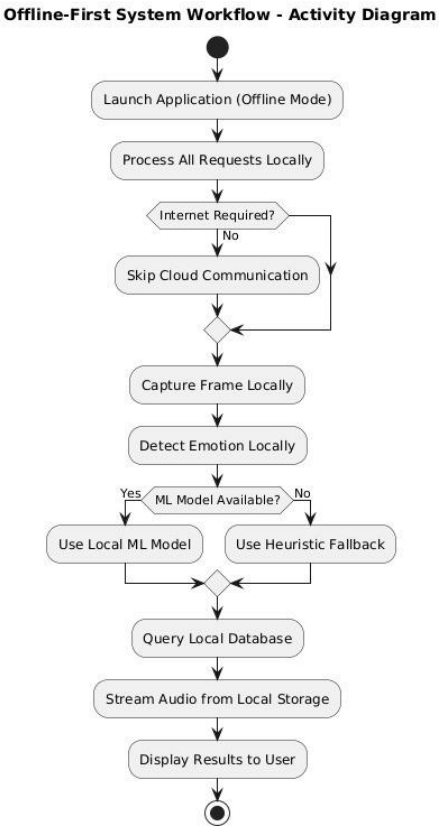



Figure 4.6.5: Offline-First System Workflow Diagram

The offline-first system workflow activity diagram represents the fundamental operational philosophy of the emotion-based music recommendation system. The architecture is intentionally designed to prioritize local computation, local storage, and device-level processing in order to provide privacy protection, low latency, and consistent performance without requiring continuous internet connectivity. This design approach ensures that all critical functional modules operate independently within the user's device environment.

The workflow begins when the user launches the application in offline execution mode. During initialization, the system loads essential application modules including authentica- tion services, emotion detection components, recommendation logic, and media playback controls. Since the application is structured around offline-first principles, all user requests and processing tasks are handled locally within the device rather than being transmitted to external cloud services.

After the application is initialized, the system evaluates whether any network-based functionality is required. In a typical execution scenario, internet connectivity checks are bypassed because the application is designed to function fully without external network communication.

This allows the system to operate reliably in restricted environments such as areas with poor network coverage or secure environments where internet access is unavailable.

Once the application environment is ready, the system activates the emotion detection module. Facial image frames are captured directly from the local webcam or device camera stream. The camera module continuously collects image frames representing the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

user's facial expressions and forwards them to the image preprocessing component.

The preprocessing stage prepares the captured images for analysis by applying several operations including face region detection, image resizing, normalization, and noise reduction. These steps ensure that the facial images match the required input format of the machine learning models used for emotion recognition.

If a trained machine learning model is available in the system, the application executes lo- cal emotion classification using deep learning techniques. Convolutional neural networks are used for feature extraction from facial images, while hybrid LSTM-CNN architectures can be used to improve prediction accuracy by capturing temporal patterns in facial ex- pression sequences. The output of the model is an emotion label representing the user's detected emotional state.

In cases where the trained model is unavailable or the prediction confidence score falls below a predefined threshold, the system activates a fallback mechanism. The fallback mechanism utilizes heuristic-based emotion detection techniques that rely on simplified facial feature patterns or predefined rules to approximate the emotional state. This ensures that the system continues to operate even when machine learning inference cannot produce reliable results.

After the emotional state has been determined, the recommendation engine becomes active. The system queries the locally maintained database that stores song metadata, emotion-to-music mappings, and playlist information. Based on the detected emotional category, the system filters and selects appropriate songs that correspond to the user's current mood.

The selected music tracks are retrieved from locally stored audio directories within the device storage. Audio playback is performed directly from these local files rather than through online streaming services. This approach significantly reduces latency, eliminates network dependency, and ensures uninterrupted music playback even when internet connectivity is unavailable.

Once the recommendation process is completed, the generated playlist is transmitted to the user interface module. The interface displays the recommended songs along with relevant metadata such as song title, artist name, and emotional category. The user can then interact with the media player controls to play, pause, skip, or repeat tracks according to personal preference.

The offline-first workflow design provides several important advantages including en- hanced user privacy, reduced system latency, and reliable operation in network-restricted environments. By performing all major computations locally within the device, the system maintains consistent performance across different hardware configurations while ensuring that sensitive user data such as facial images remains securely within the local environment.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- User Activity Diagram (Mood-Based Song Recommender)

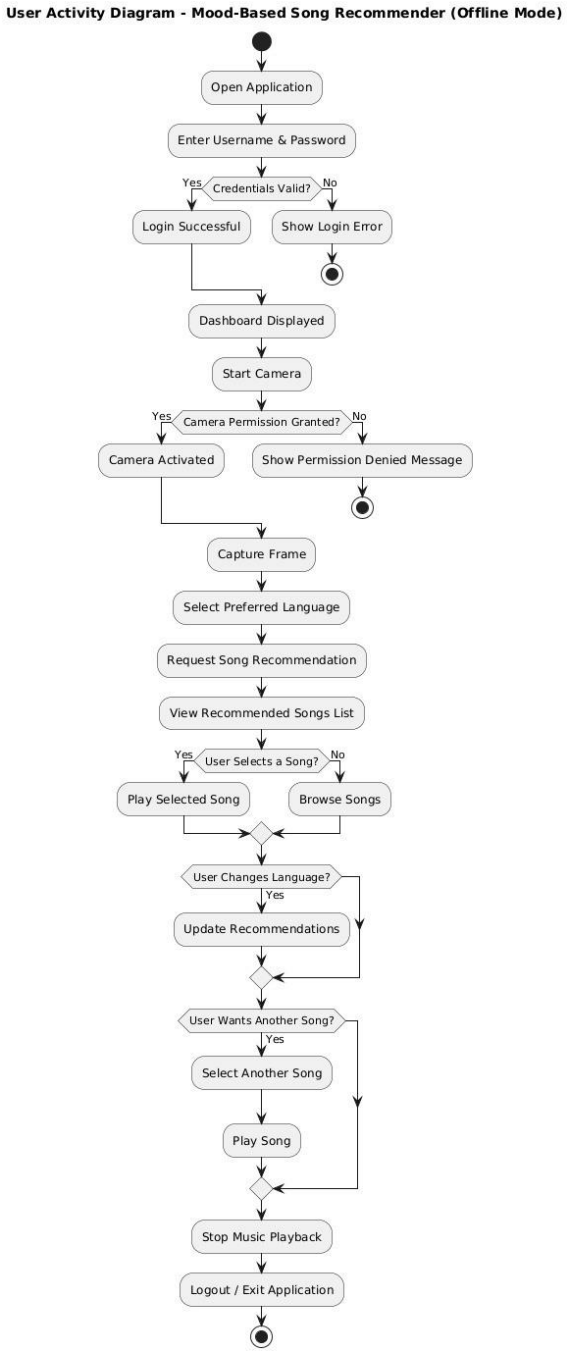



Figure 4.6.6: User Activity Diagram (Mood-Based Song Recommender)

The user activity diagram represents the interaction workflow between the user and the offline mood-based song recommendation system. It illustrates the sequence of actions performed by the user and the corresponding responses generated by the system during normal application usage.

The workflow begins when the user launches the application and accesses the login interface. At this stage, the user is required to enter authentication credentials such as a username and password. These credentials are submitted to the authentication module, which verifies the provided information using locally stored credential data maintained in the device database. This local

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

verification process ensures that the authentication mechanism functions independently without requiring network connectivity.

If the entered credentials are valid, the authentication module grants access to the ap- plication and redirects the user to the main dashboard interface. The dashboard acts as the central control panel from which the user can initiate emotion detection, view music recommendations, and manage playback settings. In contrast, if the authentication pro- cess fails due to incorrect credentials, the system displays an error notification indicating that the login attempt was unsuccessful. The user is then prompted to re-enter valid credentials in order to continue.

After successful login, the system requests camera access permission from the user. Camera access is required to capture facial expressions for emotion detection. If the user grants permission, the system activates the device camera and prepares the emotion detection module for real-time facial analysis. If the user denies the camera permission request, the system displays a permission denied message and prevents further emotion detection operations until permission is granted.

Once camera access is enabled, the user can initiate frame capture through the webcam or device camera interface. The captured facial frames are processed by the emotion detection module, which applies image preprocessing, feature extraction, and machine learning classification techniques to determine the user's current emotional state.

Following emotion detection, the user is provided with an option to select a preferred language for music recommendations. The language selection feature allows the sys- tem to filter songs according to the user's linguistic preference, thereby improving the personalization of the recommendation results.

Based on the detected emotional state and the selected language preference, the recom- mendation engine queries the locally stored song database. The system filters the available songs according to emotion-based music categories and generates a list of recommended tracks that best match the user's mood and preferences. The resulting playlist is then displayed through the user interface.

If the user selects a song from the recommendation list, the system retrieves the selected audio file from the local storage directory and initiates playback through the integrated media player module. This local playback mechanism ensures fast response times and eliminates the need for internet-based audio streaming services.

During playback, the user is allowed to browse additional songs, modify language pref- erences, request updated recommendations, or select a different song for playback. The system continuously supports dynamic interaction and allows the user to control the music listening experience according to personal preference.

If the user changes the language preference or if a new emotion state is detected during runtime, the recommendation engine automatically refreshes the playlist and updates the displayed song list. This adaptive recommendation behavior ensures that the music suggestions remain relevant to the user's current emotional condition.

Playback control options such as play, stop, pause, and switch-to-another-song function- alities are provided through the media player interface. These controls allow the user to manage audio playback conveniently while navigating through the recommended song collection.

The activity workflow concludes when the user decides to terminate the session. The session can be ended either by logging out of the application or by exiting the program interface. Upon termination, the system stops all active processes and releases allocated resources such as camera access and media playback services.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Overall System Activity Diagram

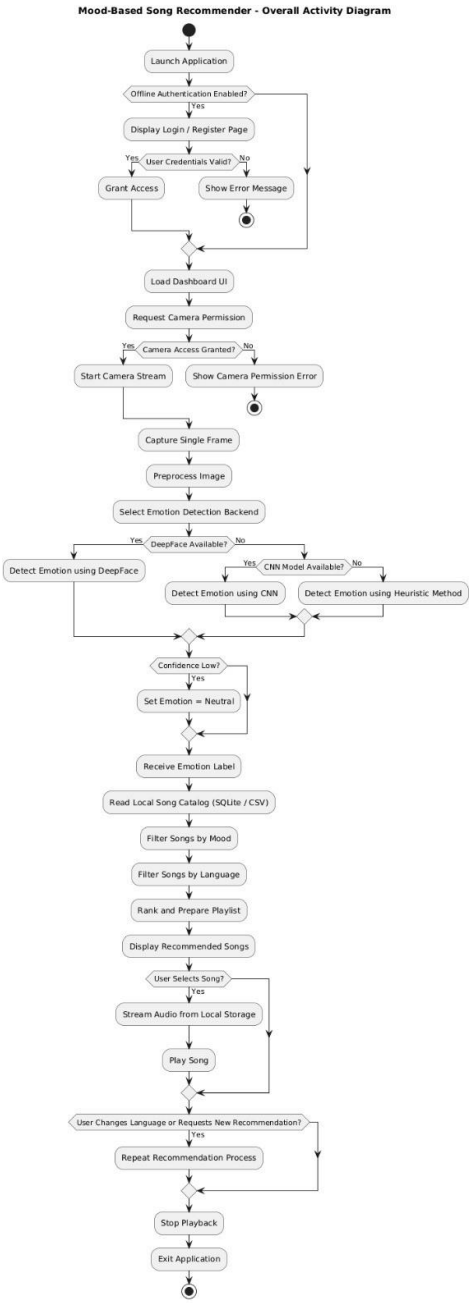



Figure 4.6.7: Overall System Activity Diagram

The overall activity diagram represents the complete operational pipeline of the offline mood- based song recommendation system, starting from application launch to personalized music playback and session termination. The diagram provides a comprehensive view of how different modules within the system interact sequentially to deliver emotion-aware music recommendations while maintaining an offline-first architecture.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The process begins when the user launches the application. During initialization, the sys- tem loads essential modules including the authentication module, camera interface, emotion detection engine, recommendation engine, and media playback controller. After the applica- tion environment is initialized, the system checks whether offline authentication is enabled for user access control. If authentication is required, the system displays the login or registration interface where the user can either create a new account or log into an existing account.

The user enters login credentials such as username and password. These credentials are validated against locally stored authentication data maintained within the device database. The validation process ensures that only authorized users can access the system. If the credentials are valid, the authentication module grants access to the user and loads the main dashboard interface. If the authentication process fails, the system displays an authentication error message and requests the user to enter valid login details again.

Once the login process is successfully completed, the system requests webcam or camera access permission from the user. Camera access is necessary to capture facial images for emotion detection. If the user grants camera permission, the camera stream is activated and the system prepares the emotion detection module for analysis. If the user denies camera permission, the system displays a camera permission error message and prevents further emotion detection operations until permission is granted.

After camera activation, the system captures a single facial frame from the webcam stream. The captured frame is then passed to the image preprocessing stage. During preprocessing, several operations are performed including image resizing to match model input dimensions, normal- ization to standardize pixel values, and optional grayscale conversion to reduce computational complexity. These steps help improve the accuracy and efficiency of the emotion detection process.

Once preprocessing is completed, the emotion detection module is activated. The system selects the most suitable detection backend depending on the availability of installed frameworks and trained models. If the DeepFace framework is available, the system performs emotion detection using embedding-based facial feature analysis. This approach compares extracted facial features with trained emotion representations to determine the user's emotional state.

If the DeepFace framework is not available, the system checks whether a trained convolutional neural network model exists. If the CNN model is available, the system performs CNN-based emotion classification using the preprocessed facial image. The CNN architecture extracts visual features from facial expressions and predicts the corresponding emotion category.

In cases where the CNN model is unavailable or the prediction confidence falls below a pre- defined threshold value, the system activates a fallback detection mechanism. This fallback mechanism uses heuristic-based emotion detection techniques that rely on simplified facial pattern analysis. When prediction confidence remains insufficient for reliable classification, the system assigns the default emotion state as Neutral. This default assignment helps main- tain recommendation stability and prevents incorrect emotional inference from affecting music selection.

After emotion classification is completed, the detected emotion label is used as input for the music recommendation module. The recommendation engine queries the locally stored music catalog maintained using a lightweight SQLite database or a CSV-based repository. The database contains song metadata including song titles, artists, genres, language categories, and emotion-to-music mapping information.

The recommendation engine filters available songs according to the detected emotional category and the language preference selected by the user. Filtering ensures that the recommended songs align with both the user's current mood and linguistic preferences.

Following the filtering process, ranking algorithms are applied to prioritize playlist items. Songs are ranked according to several factors including emotional relevance, popularity indicators, and historical playback frequency. This ranking mechanism ensures that the most appropriate songs appear at the top of the recommendation list.

The generated playlist is then transmitted to the user interface module where the recommended songs are displayed along with relevant metadata such as song title, artist name, and mood category. The user can browse the recommendation list and select a song for playback.

If the user selects a song from the recommendation list, the system retrieves the corresponding audio file from the local storage directory and begins playback through the integrated media player module. Since audio files are stored locally, playback occurs instantly without requiring network-based streaming services.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

During runtime, the user may request additional recommendations or modify language prefer- ences. When such changes occur, the recommendation pipeline is re-executed and the playlist is updated dynamically to reflect the new preferences or detected emotion state.

The playback session continues until the user decides to stop the music or exit the application. Playback can be terminated either by manually stopping the song or by closing the application interface.

The system architecture follows an offline-first design philosophy to ensure user privacy, min- imize processing latency, and maintain stable operation even in environments where internet connectivity is unavailable. By performing all critical computations locally, the system provides a reliable and secure emotion-aware music recommendation experience.

## 5. METHODOLOGY

The methodology of the proposed Facial Expression-Based Mood Song Recommender System describes the systematic approach used to design, implement, and evaluate the system. The proposed system follows an offline-first approach where all core functionalities such as emotion detection, recommendation filtering, and media playback operate without internet dependency. The system captures a single facial frame per recommendation cycle to minimize computational load and latency. Local storage is used to maintain liked songs and playback preferences across sessions.

### 5.1 Tools

The development and implementation of the emotion-based music recommendation system rely on a combination of software frameworks, programming languages, machine learning libraries, and development utilities. These tools support different stages of the system including data processing, emotion detection, recommendation generation, user interface development, and system deployment. The selected technologies are lightweight and compatible with offline execution environments, enabling efficient local processing without reliance on external cloud services.

- **Flask Framework:** The Flask framework is used as the backend web framework for implementing server-side logic and route handling. Flask provides a lightweight and flexible environment for developing web applications using Python. It manages request handling, communication between frontend and backend modules, and integration with the emotion detection and recommendation components. Flask also allows the application to serve HTML templates and manage REST-style routes for different system operations.

- **Python:** Python serves as the primary programming language used in the development of the system. It is responsible for implementing core functionalities including emo- tion detection algorithms, recommendation logic, preprocessing operations, and database interaction. Python is widely used in artificial intelligence applications due to its exten- sive ecosystem of machine learning libraries and easy integration with computer vision frameworks.

- **SQLite / CSV:** SQLite databases and CSV-based repositories are used to store song catalogs and associated metadata. These storage mechanisms contain information such as song titles, artist names, genres, language categories, and emotion-to-music mappings. SQLite provides efficient local database management, while CSV files offer a simple and lightweight format for structured data storage. Both approaches support fast data retrieval and enable the system to operate without internet connectivity.

- **OpenCV:** OpenCV is used for capturing facial frames from the device camera and performing image preprocessing operations. The library supports tasks such as face detection, image resizing, grayscale conversion, and normalization. These preprocessing steps prepare the captured facial images for emotion detection models and help improve prediction accuracy.

- **TensorFlow / Keras:** TensorFlow and Keras are used for implementing deep learning models that perform emotion classification. Lightweight convolutional neural network architectures are developed using these libraries to extract facial features and classify emotions from image inputs. The use of optimized neural network models ensures efficient inference on local machines with limited computational resources.

- **DeepFace (Optional):** DeepFace is used as an optional alternative backend for emotion analysis. This framework provides pre-trained deep learning models capable of perform- ing facial attribute detection and emotion classification. When

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

available, DeepFace can be used to improve detection accuracy through embedding-based facial feature analysis.

- **HTML, CSS, JavaScript:** The frontend user interface of the system is developed using standard web technologies including HTML, CSS, and JavaScript. HTML is used for structuring web pages, CSS provides styling and layout design, and JavaScript enables interactive functionality within the browser environment. These technologies allow users to interact with the application, capture facial images, and view recommended songs through a responsive dashboard interface.

- **PyInstaller:** PyInstaller is used for packaging the complete application into a standalone executable format. This allows the system to be distributed and deployed easily on different machines without requiring manual installation of dependencies. Packaging the application ensures that all required libraries and runtime components are included within a single distributable package.

Configuration files are used to centralize system settings such as directory paths, application constants, model selection parameters, and backend configuration options. These files simplify system maintenance by allowing developers to modify configuration values without altering core program logic.

The system also supports both offline and online operational variants. In the offline version, route access is protected using a simple JSON-based authentication store that maintains locally stored user credentials. Guarded routes ensure that only authenticated users can access emotion detection and music recommendation features. This design approach maintains security while preserving the offline-first architecture of the application.

### 5.2 Modules

The proposed emotion-based music recommendation system is designed using a modular archi- tecture in order to improve flexibility, maintainability, and system scalability. Modular design allows different components of the system to operate independently while still interacting with each other through well-defined interfaces. This approach simplifies development, testing, de- bugging, and future enhancements. Each module is responsible for performing a specific task within the overall system workflow.

By dividing the system into logical modules, developers can upgrade or modify individual components without affecting the entire application. This also allows integration of additional features such as improved artificial intelligence models, advanced recommendation algorithms, and cloud-based services in the future.

The key modules present in the system are described below:

- **Frontend Module:** The frontend module is responsible for managing all interactions between the user and the application. It provides a graphical interface that allows users to access system functionalities in an intuitive and user-friendly manner. The interface enables users to capture facial expressions through the device camera, select preferred music languages, view recommended playlists, and control media playback operations.

  The frontend module also implements user interface elements such as buttons, dashboards, playlist cards, and audio players. Playback controls include options such as play, pause, stop, next song, previous song, and loop functionality. The interface is designed using HTML, CSS, and JavaScript to ensure responsive behavior across different devices and screen sizes. Overall, this module enhances user experience by presenting system outputs in a clear and interactive format.

- **Authentication Module:** The authentication module manages user identity verification within the system. Since the application is designed to operate in offline environments, authentication is implemented using locally stored credentials rather than cloud-based services.

  User information such as usernames and passwords is stored securely within a JSON- based credential repository. During the login process, the system compares entered credentials with the stored data to verify user identity. If the credentials match, the user is granted access to the system dashboard; otherwise, an authentication error message is displayed.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

This module also supports basic access control mechanisms that prevent unauthorized users from accessing protected application routes. The offline authentication design ensures that user privacy is maintained and the system remains functional even without internet connectivity.

- **Emotion Detection Module:** The emotion detection module forms the core intelligence of the system. It is responsible for identifying the emotional state of the user by analyzing facial expressions captured from the device camera.

  The module processes video frames using computer vision techniques and machine learn- ing algorithms. Multiple emotion detection backends are supported to increase reliability and flexibility. These include convolutional neural network (CNN) models for deep learning-based emotion classification, heuristic methods for simplified detection, and the DeepFace framework as an optional high-accuracy backend.

  Captured facial frames are first preprocessed using image normalization and resizing techniques before being passed to the emotion classification models. The predicted emotion label is then forwarded to the recommendation module to generate appropriate music suggestions.

- **Recommendation Module:** The recommendation module is responsible for generating personalized music suggestions based on the emotional state detected from the user's facial expression. This module implements mood-based filtering techniques to match songs with specific emotional categories such as happy, sad, relaxed, or energetic.

  The recommendation process primarily focuses on matching songs that correspond to the detected emotion. As a secondary criterion, the module also considers popularity indicators such as frequently played tracks or user engagement patterns. This combination improves the relevance of generated playlists.

  The modular structure of the recommendation engine allows future upgrades including collaborative filtering, hybrid recommendation strategies, and reinforcement learning- based music personalization.

- **Database Module:** The database module manages all data storage and retrieval operations within the system. It stores music catalogs, song metadata, and user-related information required for system operation.

  The module supports both SQLite databases and CSV file-based repositories to ensure lightweight and efficient offline storage. Song metadata stored in the database includes song titles, artist names, language categories, genre tags, and emotion-to-music mappings.

  Efficient database queries allow the recommendation engine to quickly retrieve relevant songs based on detected emotions and user preferences. The database module also ensures data consistency and integrity across different system operations.

- **Media Streaming Module:** The media streaming module handles the playback of music files within the application. Once the recommendation module generates a playlist and the user selects a song, this module retrieves the corresponding audio file from local storage and initiates playback.

  The module supports commonly used audio file formats and ensures smooth streaming with minimal latency. It also normalizes Windows-compatible file paths to prevent file access issues during playback. Integrated playback controls allow users to pause, resume, skip, or repeat songs while listening to music.

- **Offline Support Module:** The offline support module ensures that the entire system can operate without requiring internet connectivity. This module protects application routes and prevents external network requests from being triggered during system execution.

  All core functionalities including authentication, emotion detection, recommendation generation, and media playback are performed locally within the user's device. This architecture guarantees high privacy protection and allows the application to function reliably in restricted network environments.

- **Logging and Error Handling Module:** The logging and error handling module is responsible for recording important runtime events, system activities, and error messages generated during application execution.

  Log files help developers monitor system performance and identify potential issues during operation. In case of errors, the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

module captures diagnostic information that assists in debugging and improving system stability. Proper error handling mechanisms also prevent application crashes and provide user-friendly error notifications.

- **Configuration Module:** The configuration module manages global system parameters and application settings. These settings include default language preferences, theme customization options, audio playback configurations, and backend model selection pa- rameters.

  Configuration values are stored in centralized files so that developers can modify sys- tem behavior without changing the core application code. This flexibility simplifies deployment across different environments and allows easy customization for various user requirements.

Overall, the modular architecture of the emotion-based music recommendation system ensures that each component operates independently while contributing to the overall system function- ality. This design approach improves maintainability, facilitates testing, and provides a strong foundation for future system extensions such as cloud integration, advanced deep learning models, and large-scale music recommendation services.

### 5.3 Methods

The system employs well-defined computational methods for emotion detection and person- alized recommendation generation. The methods are designed to ensure efficient processing, accurate mood classification, and relevant music filtering while maintaining offline functionality and real-time responsiveness.

**Emotion Detection Method:**

- Capture a single facial frame from the camera stream during user interaction or playback monitoring phase.
- Apply image preprocessing operations including resizing to the model input dimension, intensity normalization, noise reduction, and optional grayscale transformation to improve prediction consistency.
- Forward the processed frame to the selected emotion detection backend, which may include CNN-based classifiers, heuristic rule engines, or DeepFace embedding-based recognition models.
- Compute emotion probability scores using softmax or similarity-based evaluation func- tions depending on backend selection.
- Apply confidence threshold validation to eliminate unstable or low-reliability predictions.
- If prediction confidence is lower than the predefined threshold, automatically assign the default emotion state as **Neutral** to maintain stable system behavior.
- Store detected emotion metadata temporarily in memory buffers for downstream recom- mendation pipeline execution.
- Enable optional continuous monitoring mode where frames are periodically captured to update emotion classification during user interaction.
- Implement fallback detection logic that activates heuristic-based evaluation when CNN inference confidence is marginal.

**Recommendation Method:**

- Map the detected emotional state into the corresponding application-level mood category using an internally maintained mood translation dictionary.
- Retrieve music metadata from the locally stored SQLite database or CSV-based catalog depending on system deployment configuration.
- Filter songs based on three primary factors: detected mood, user-selected language preference, and availability of media files in local storage directories.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Execute ranking algorithms that prioritize strong emotion-song matching scores followed by secondary popularity metrics such as playback frequency, implicit user preference signals, or historical selection trends.
- Construct an ordered playlist structure and transmit it to the frontend playback controller for visualization and audio streaming.
- Support automatic playlist regeneration if the detected emotion changes during runtime or if the user modifies language or playback settings.
- Optimize recommendation latency by caching filtered catalog subsets corresponding to frequently detected emotions.
- Allow configurable recommendation sensitivity parameters to control playlist diversity versus strict mood matching.

**System Configuration and Backend Adaptability:**

- Backend selection for emotion detection can be controlled through environment variables or internal configuration switches.
- The system supports model modularity, enabling switching between CNN-based infer- ence, heuristic detection, or hybrid fusion strategies without modifying frontend logic.
- Application constants such as detection thresholds, language codes, and file path roots are centralized in configuration management modules.
- This architectural flexibility facilitates experimental evaluation of different emotion recog- nition techniques while preserving code maintainability.

The overall method design emphasizes privacy-preserving offline processing, low-latency in- ference, and modular AI pipeline integration, making the system suitable for real-time emotion- based music recommendation applications.

### 5.4 Algorithms

The system integrates multiple algorithms to achieve robust emotion detection and efficient recommendation generation. Multiple detection approaches are used to improve prediction reliability and ensure adaptability across different deployment scenarios.

**Emotion Detection Algorithms:**

- **CNN-Based Model:** A lightweight Convolutional Neural Network (CNN) architecture is trained on facial emotion datasets for real-time emotion classification. The model is designed to balance computational efficiency and prediction accuracy, making it suitable for offline deployment environments. The network consists of stacked convolutional layers followed by pooling and dense layers, with dropout regularization applied to reduce overfitting. The model is optimized for faster inference latency while maintaining reliable classification performance.
- **Heuristic-Based Detection:** This method extracts low-level visual and geometric visual features from captured facial frames. Parameters such as brightness distribution, contrast ratio, edge sharpness, and facial smile intensity are computed using image signal pro- cessing techniques. Rule-based scoring functions are applied to estimate emotional state when deep learning prediction confidence is below acceptable limits. This serves as a fast fallback mechanism for lightweight execution scenarios.
- **DeepFace-Based Mapping:** The DeepFace framework is integrated as an auxiliary validation mechanism to improve emotion recognition reliability. The predicted facial expression output is cross-validated with DeepFace embedding-based similarity match- ing. The framework helps map external emotion labels into the predefined application mood taxonomy used by the recommendation engine.
- **Confidence Threshold Mechanism:** A probabilistic confidence threshold is applied to filter unreliable predictions generated by the emotion detection model. If the prediction confidence score is lower than the predefined threshold value, the system automatically assigns the Neutral emotion state. This strategy improves recommendation stability and prevents erratic playlist switching during uncertain detections.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- **Hybrid Decision Strategy:** The final emotion classification result is generated using a hybrid fusion approach. The system combines CNN softmax probability scores, heuristic feature-based evaluation signals, and DeepFace validation outputs. Weighted decision aggregation is applied to determine the most probable emotional state. The hybrid mechanism enhances robustness against lighting variations, occlusions, and webcam quality differences while maintaining real-time responsiveness.

- **Runtime Adaptation Logic:** The detector supports dynamic backend switching based on configuration parameters and hardware constraints. For low-resource environments, heuristic fast-path detection can be activated, whereas CNN-based inference is prioritized when sufficient computational capacity is available.

- **Model Generalization Enhancement:** Generalization is improved through controlled data augmentation, including horizontal flipping, minor rotational transformations, bright- ness scaling, and normalization preprocessing. Early stopping and regularization tech- niques are used during training to prevent overfitting.

- **Inference Optimization Strategy:** Batch size tuning, model pruning considerations, and cached normalization parameters are employed to reduce redundant computation during inference. These optimizations help maintain low latency during continuous camera monitoring.

**Dataset Specifications and Processing:**

- Dataset is organized in a folder-based structure containing emotion classes such as Angry, Happy, Sad, Neutral, Relaxed, and Energetic.
- The system also supports FER-2013 style CSV formatted datasets for machine learning training and validation.
- Label mapping is performed to normalize dataset emotions, such as mapping Surprise → Energetic and Disgust/Fear → Angry.
- Data partitioning is performed using a standard split ratio of {70% training, 15% valida- tion, and 15% testing.
- Data augmentation techniques are applied to improve model generalization, including horizontal flipping, minor rotational transformations, and controlled brightness adjust- ments.
- Preprocessing pipelines are implemented to normalize pixel intensity values and enhance feature extraction stability.

The algorithmic design emphasizes computational efficiency, offline operability, and practical deployment suitability for emotion-based music recommendation tasks. Future enhancements may include transformer-based emotion recognition, temporal emotion tracking, and adaptive reinforcement learning-based recommendation optimization.

## 6. REQUIREMENT SPECIFICATIONS

The requirement specification defines the hardware, software, and functional expectations of the emotion-based music recommendation system. The system is designed for offline deployment with lightweight machine learning inference and local media processing.

### 6.1 System Requirements

The system is designed to operate primarily in a Windows-based local execution environment and supports real-time webcam-based emotion detection along with offline music recommen- dation processing. The architecture emphasizes low-latency inference, privacy-preserving local computation, and modular software integration.

The design of the system prioritizes efficient resource utilization while maintaining acceptable prediction accuracy for emotion recognition tasks. All core operations, including frame capture, emotion classification, recommendation filtering, and media playback, are performed locally without requiring internet connectivity. This ensures enhanced data privacy and consistent performance even in restricted network environments.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The system supports modular software execution, allowing independent deployment and test- ing of frontend interaction modules, emotion detection backends, recommendation engines, and database components. Runtime configuration flexibility enables backend switching and threshold tuning without altering the main application logic.

Hardware compatibility considerations include support for standard integrated webcams, basic audio output devices, and processors capable of handling lightweight machine learning inference workloads. Storage management is optimized for local music catalogs and model assets.

Overall, the system requirement design focuses on creating a practical, user-friendly, and computationally efficient platform for emotion-based music recommendation in offline desktop environments.

### 6.1.1 Hardware Requirements

- A dual-core processor or higher is recommended to ensure smooth execution of the emotion detection and recommendation modules during real-time operation.
- Minimum system memory of 4 GB RAM is required, while 8 GB RAM or higher is recommended to improve inference speed, multitasking capability, and overall system responsiveness.
- An integrated or external webcam with reasonable resolution is required for capturing facial frames for emotion recognition processing.
- Storage capacity of approximately 2–5 GB is sufficient for small to medium-sized music catalogs, although larger libraries may require additional storage space depending on user collection size.
- Solid State Drive (SSD) storage is recommended to minimize file access latency during audio streaming and playlist loading operations.
- Optional Graphics Processing Unit (GPU) support can be utilized for accelerating CNN model training and inference computation when available.
- Standard audio output hardware such as speakers or headphones is required to enable music playback functionality.
- A stable power supply is recommended for uninterrupted system execution during pro- longed usage sessions.
- Display hardware with minimum HD resolution is suggested to ensure proper visualization of the user interface elements.

### 6.1.2 Software Requirements

- **Operating System:** Windows 10 or Windows 11 is recommended to ensure compatibility with all system components, webcam drivers, and audio devices. The OS must support local Python execution and access to hardware peripherals.
- **Programming Language:** Python version 3.10 or higher is required for system develop- ment, with Python 3.11+ preferred for better performance, enhanced typing support, and updated library compatibility.
- **Web Browser:** Modern web browsers such as Google Chrome or Microsoft Edge are necessary for the frontend interface. Camera permissions must be enabled to allow real-time emotion detection and UI interaction.
- **Machine Learning Libraries:** Optional CUDA-enabled GPU drivers are recommended for accelerating CNN-based model training and inference. For CPU-only execution, standard TensorFlow/Keras or PyTorch setups are supported.
- **Frameworks and Libraries:**
  - **Flask:** Used for backend web server development, routing management, and API communication between frontend modules and processing components in offline execution mode.
  - **OpenCV:** Responsible for real-time webcam frame capture, image preprocessing, face detection, and computer

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

vision-based feature extraction tasks.

- **Pillow (PIL):** Supports image format conversion, resizing, normalization, and general-purpose image manipulation operations required during preprocessing and inference stages.
- **scikit-learn:** Provides utility functions for data preprocessing, performance evalua- tion metrics, classification report generation, and optional machine learning pipeline support.
- **SQLite:** Serves as the lightweight local database management system for storing music metadata, user authentication credentials, mood-song mappings, and playlist catalogs.
- **TensorFlow/Keras (Optional):** Used for training and deploying CNN-based emo- tion detection models. These frameworks support neural network construction, model optimization, weight saving, and inference execution.
- **DeepFace Framework:** Supports facial embedding-based emotion validation and auxiliary recognition mapping to improve classification robustness.
- **NumPy and Pandas:** Used for numerical computation, dataset handling, CSV catalog processing, and array-based image preprocessing operations.
- **PyGame or Media Playback Libraries:** Utilized for local audio streaming, play- back control implementation, and synchronization of user interface commands with media output.

- **Dependency Management:** All required packages and libraries must be installed using a `requirements.txt` file to ensure reproducibility and correct version control.
- **Virtual Environment:** Python virtual environments are recommended to isolate project dependencies, prevent conflicts with system libraries, and facilitate deployment across different machines.
- **Camera and Audio Drivers:** Ensure that webcam and audio drivers are installed, up- to-date, and compatible with the operating system for seamless frame capture and audio playback.
- **Optional Development Tools:** IDEs such as Visual Studio Code, PyCharm, or Jupyter Notebook may be used to facilitate development, debugging, and model training pro- cesses.
- **Security and Permissions:** The system requires permission access to local file stor- age, camera, and audio devices. Firewall or antivirus configurations should allow the application to run without restrictions in offline mode.
- **Packaging Tools:** PyInstaller or similar tools are recommended for creating distributable executables that bundle Python scripts, dependencies, and media assets for offline usage.

### 6.2 User Requirements

- The user must have access to a device equipped with a functional webcam for capturing facial expressions used in emotion detection.
- Browser or application-level camera permissions must be enabled to allow continuous real-time emotion monitoring during system operation.
- The system should support offline authentication mechanisms, allowing users to log in and access features without requiring internet connectivity.
- Users should be able to generate personalized playlists based on detected emotional state and selected language preferences.
- Playback control functionalities including play, pause, stop, next track, previous track, and repeat/loop options must be available to the user.
- The user interface should be intuitive, minimalistic, and responsive to ensure ease of use for non-technical users.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Users should be able to view recommended songs along with metadata such as song title, artist name, and mood category.
- Language selection options should be provided to allow customization of music recom- mendations based on user preference.
- The system should provide visual feedback during emotion detection and recommendation processing stages.
- Users should have the ability to regenerate playlists by refreshing emotion detection or modifying playback settings.
- Local storage permissions are required for accessing catalog files, model assets, and audio media directories.
- The system should ensure privacy by performing all facial processing and recommenda- tion computations locally without transmitting user data externally.

### 6.3 Functional Requirements

Functional requirements describe the specific operations and services that the emotion-based music recommendation system must provide to users during normal execution. These require- ments define how the system should behave in response to user actions and internal processing conditions.

- The system must allow users to register and log in using locally stored authentication credentials without requiring internet connectivity.
- The system must verify user credentials during login and grant access only when authen- tication is successful.
- The system must request permission to access the webcam before initiating any facial image capture for emotion detection.
- The application must capture facial frames from the webcam in real-time and forward them to the preprocessing module.
- The preprocessing module must perform image normalization, resizing, and optional grayscale conversion before sending the data to the emotion classification engine.
- The system must analyze the captured facial expression using machine learning-based emotion detection models such as CNN or DeepFace frameworks.
- If the primary emotion detection model is unavailable or produces low-confidence pre- dictions, the system must activate a fallback detection mechanism using heuristic-based analysis.
- The detected emotional state must be mapped to a predefined set of emotion categories used by the recommendation engine.
- The recommendation module must retrieve songs from the local music catalog that correspond to the detected emotional state.
- The recommendation module must apply filtering and ranking mechanisms to generate an ordered playlist of songs.
- The system must display the recommended playlist on the user interface along with relevant song metadata.
- The user must be able to select any recommended song for playback using the integrated media player.
- The system must support audio playback control functions including play, pause, stop, next, previous, and repeat.
- The system must allow users to change language preferences for music recommendations.
- The system must regenerate recommendations whenever the user updates preferences or when a new emotion is detected.
- The system must store and retrieve song metadata locally using SQLite or CSV-based repositories.
- The system must maintain execution logs that record system activities, errors, and user interactions.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- The application must allow users to terminate the session by logging out or closing the application interface.

### 6.4 Non-Functional Requirements

Non-functional requirements define the quality attributes and performance characteristics ex- pected from the system. These requirements ensure that the application operates efficiently, securely, and reliably under different conditions.

- **Performance:** The system must provide low-latency response times during emotion detection and recommendation generation. Facial emotion classification should occur within a few seconds after frame capture to maintain real-time interaction.
- **Reliability:** The system should operate continuously without unexpected crashes or failures during extended usage sessions. Proper error handling and fallback mechanisms must ensure stable execution.
- **Privacy:** All user data including facial images and emotion detection results must be processed locally without transmitting information to external servers. This ensures strong privacy protection for users.
- **Usability:** The application interface must be intuitive and easy to use, even for users with minimal technical knowledge. Clear visual feedback should be provided during emotion detection and playlist generation processes.
- **Maintainability:** The system architecture must follow a modular design so that individual components such as emotion detection models, recommendation algorithms, and database modules can be updated independently.
- **Scalability:** The system should support future expansion including integration of addi- tional machine learning models, larger music catalogs, or cloud-based recommendation services.
- **Compatibility:** The application must be compatible with common desktop hardware con- figurations including integrated webcams, standard audio devices, and modern Windows operating systems.
- **Security:** User authentication data must be securely stored within local files or databases. Unauthorized access to protected application routes must be prevented through credential validation mechanisms.
- **Portability:** The system should be easily deployable on multiple machines using packaged executable files generated through distribution tools such as PyInstaller.

Overall, the defined functional and non-functional requirements ensure that the emotion-based music recommendation system operates efficiently, protects user privacy, and provides a reliable offline platform for personalized music experiences.

## 7. IMPLEMENTATION

This chapter describes the practical development and deployment implementation of the emotion- based music recommendation system. The implementation follows a structured Python- based development pipeline emphasizing offline execution, modular architecture, and privacy- preserving local processing.

### 7.1 Installation and Usage

The installation and usage process of the emotion-based music recommendation system focuses on simplicity and portability so that the application can be deployed easily on standard desktop environments. The system is designed to operate primarily in offline mode while performing local emotion detection, playlist generation, and audio playback without requiring external cloud services.

The system is installed by creating a Python virtual environment and installing all required dependencies using the requirements.txt file. A virtual environment ensures that all nec- essary libraries are isolated from the system-wide Python installation, preventing dependency conflicts and improving reproducibility.

After the environment setup is completed, hardware components such as the webcam and audio output devices must be verified to ensure smooth operation of the emotion detection and media playback modules. Since the application relies on real-time facial image capture, proper camera configuration is essential for accurate emotion recognition.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The application structure includes Flask-based backend services, template files, static assets such as CSS stylesheets and JavaScript scripts, and machine learning inference modules responsible for emotion detection. The backend routes handle API requests for emotion detection, playlist recommendations, and media streaming, while the frontend provides a responsive user interface that enables real-time interaction between the user and the system.

Users can run the application using the Flask development server during testing or development stages. For production or distribution scenarios, the system can be packaged as a standalone executable using PyInstaller. Packaged builds include all necessary Python dependencies, trained model files, and static assets, allowing the application to run on new machines without requiring additional setup.

Installation instructions include the following steps:

- Clone or extract the project repository to the local machine from the source code archive or version control system.
- Navigate to the project directory and create a Python virtual environment using the command `python -m venv venv`.
- Activate the virtual environment using the command `venv\Scripts\activate` on Win- dows systems or the equivalent activation script for the operating system being used.
- Install all required project dependencies by executing the command `pip install -r requirements.txt`. Ensure that package versions are compatible with the installed Python interpreter version.
- Verify webcam functionality by testing camera access through simple Python OpenCV scripts or through browser-based `getUserMedia` API support.
- Confirm that audio output devices such as speakers or headphones are properly connected, configured, and set as the default playback device within the operating system.
- Check that camera and microphone permissions are enabled in the browser settings if the web interface is accessed through a browser.
- Execute the script `offline app.py` or `online app.py` to launch the backend server depending on the deployment mode selected.
- Open a supported web browser and navigate to the address `http://localhost:5000` to access the application user interface.
- Test the system by performing basic operations such as camera capture, emotion detection, and playlist recommendation generation.
- Verify that recommended songs appear correctly and that audio playback functionality works as expected.
- If using packaged deployment, run the generated executable file and wait for automatic backend initialization before accessing the interface.
- Ensure that project folders such as `songs/`, `models/`, and `song csv/` exist and contain the required audio files, machine learning models, and catalog data.
- In case of errors, verify Python path configuration, reinstall dependencies if necessary, and examine system logs printed in the console output.

For packaged executable usage, the following steps are recommended:

- Use PyInstaller with the provided `SongRecommender.spec` configuration file to generate a distributable executable version of the application.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Ensure that all required media assets, trained model files, and CSV catalog data are placed in the correct folder structure as specified in the project documentation.
- Run the generated executable file directly from the distribution directory. The application will automatically launch the backend services and initialize the graphical user interface.
- Once launched, users can access the application through the local interface and begin interacting with the system for emotion detection and music recommendation.
- Since all computations occur locally, the application does not transmit user data externally. Facial image processing, emotion classification, and playlist generation are all executed within the user's device to maintain data privacy.

The installation process also includes optional configuration steps that allow customization of system parameters. These parameters may include emotion detection confidence thresholds, offline secret keys for authentication, preferred language settings for music recommendations, and directory paths for model files and media assets.

Such configuration options enable developers and system administrators to adapt the application to different deployment environments while maintaining the core functionality of the emotion- based music recommendation platform.

### 7.2 Web Application

The web application component of the emotion-based music recommendation system provides the primary interface through which users interact with the system. It integrates real-time emotion detection, playlist generation, and audio playback into a single responsive platform. The application is designed to operate efficiently in offline desktop environments while maintaining a smooth and interactive user experience.

The web application is developed using the Flask framework, which provides backend routing, API communication, and template rendering capabilities. Flask acts as the central controller that connects frontend interactions with backend processing modules such as emotion detection, recommendation engines, authentication services, and media streaming handlers.

The Flask server manages multiple endpoints responsible for handling various system opera- tions. These endpoints include routes for emotion detection requests, playlist recommendation retrieval, authentication processing, and local media streaming. Each endpoint receives requests from the frontend interface, processes the request using backend logic, and returns responses in JSON format to update the user interface dynamically.

The frontend interface is constructed using standard web technologies including HTML, CSS, and JavaScript. HTML is used to structure the layout of the application interface, while CSS is responsible for styling, layout management, and responsive design. JavaScript is used to enable dynamic interaction, handle asynchronous communication with the backend server, and control user interface updates in real time.

To enable real-time facial emotion detection, the application utilizes the browser's getUserMedia API. This API allows the web application to request permission from the user to access the device's webcam. Once permission is granted, the webcam stream is captured and individual frames are extracted for emotion detection processing.

Captured frames are transmitted to the backend emotion detection module through asynchronous API requests. The backend performs image preprocessing and executes machine learning inference using the configured emotion detection model. The predicted emotional state is then returned to the frontend interface, where it is used to trigger music recommendation generation.

LocalStorage is used to maintain lightweight persistent user data such as liked songs, language preferences, and last selected playback settings. This client-side storage mechanism allows the system to quickly retrieve user preferences without performing frequent database queries, thereby improving application performance and responsiveness.

The recommendation results are displayed in the user interface as a playlist containing song titles, artist information, and associated emotion categories. Users can browse the recommended songs and select tracks for playback through the integrated media player component.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Media playback functionality is implemented using HTML5 audio components integrated with backend streaming routes. When a user selects a song for playback, the frontend sends a request to the backend server, which retrieves the corresponding audio file from local storage and streams it to the browser.

To ensure reliable operation across different operating systems, audio file paths are normalized and validated before streaming. Special care is taken to ensure compatibility with Windows filesystem structures where path formatting may differ from other environments.

The web application supports asynchronous communication between the frontend and backend components using JSON-based API requests. JavaScript functions send requests to backend endpoints and process responses without requiring full page reloads. This asynchronous design significantly improves user experience by enabling seamless interaction and faster response times.

Emotion detection events trigger backend inference services. Once the user's emotional state is identified, the recommendation engine filters the locally stored music catalog and generates a playlist based on emotion-to-music mapping rules. The playlist is then returned to the frontend and displayed dynamically to the user.

The user interface design emphasizes simplicity, responsiveness, and accessibility for users with minimal technical expertise. The layout includes clearly visible controls for camera capture, playlist browsing, language selection, and media playback. Responsive CSS styling ensures that the interface adapts smoothly to different screen sizes and display resolutions.

Security considerations are incorporated into the web application architecture to minimize potential risks. External network calls are restricted when the system operates in offline mode. File paths are validated before streaming audio content to prevent directory traversal vulnerabilities. Additionally, only essential backend routes are exposed to reduce the application attack surface.

Authentication mechanisms are also integrated within the web application to restrict access to authorized users. Credential validation occurs locally using stored authentication data, ensuring that user accounts remain protected even in offline environments.

Although the web application is primarily designed for local deployment, it can be extended to operate within controlled institutional networks if required. This can be achieved by configuring allowed origins, enabling secure communication protocols, and applying appropriate server security policies.

Overall, the web application serves as the central interaction platform of the emotion-based music recommendation system, combining computer vision, machine learning, and multimedia technologies to deliver a seamless and personalized music experience for users.

### 7.3 Environmental Setup

The environmental setup of the system plays a critical role in ensuring stable execution of the emotion-based music recommendation application. A properly configured development and runtime environment guarantees that all software components, machine learning models, and multimedia modules operate correctly without compatibility issues.

The development environment is configured to support stable execution of the emotion-based music recommendation system. Python versions between 3.10 and 3.12 are recommended because these versions provide strong compatibility with modern machine learning frameworks, computer vision libraries such as OpenCV, and web frameworks such as Flask.

Virtual environment isolation is required to prevent dependency conflicts with system-level Python installations. Virtual environments allow each project to maintain its own independent package ecosystem, ensuring that updates or modifications in one project do not affect other Python applications installed on the same machine. This isolation improves maintainability and reproducibility of the development environment. :contentReference[oaicite:0]index=0

All required packages must be installed using the requirements.txt file, which contains version-pinned libraries required by the system. Version pinning ensures that the same versions of packages are installed on different machines, allowing the application to behave consistently during development, testing, and deployment. :contentReference[oaicite:1]index=1

The virtual environment is created using the standard Python venv module with the command:

```
python -m venv venv
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

After creation, the environment must be activated before installing dependencies. On Windows systems the activation command is:

```
venv\Scripts\activate
```

On Linux or macOS systems, the following command is used:

```
source venv/bin/activate
```

Once activated, the project dependencies can be installed using the command:

```
pip install -r requirements.txt
```

This step installs all required packages locally within the virtual environment rather than globally in the system Python installation.

Browser-level camera permissions must be enabled before testing real-time emotion detection modules. When the application first attempts to access the webcam through the browser, the user will be prompted to grant camera access permission. Accepting this request allows the application to capture real-time video frames required for facial emotion recognition.

The system is primarily designed for Windows operating systems, especially Windows 10 and Windows 11 environments. These platforms provide stable support for Python runtime environments, multimedia frameworks, and device drivers required for webcam capture and audio playback.

Proper installation and configuration of webcam drivers, audio drivers, and graphics drivers (if GPU acceleration is used) are necessary to ensure smooth system performance. Updated device drivers reduce the risk of hardware communication failures during camera capture or audio streaming.

Hardware environment validation should include several verification steps before system exe- cution. Developers should confirm that the webcam is capable of capturing video frames at acceptable resolution, that audio output devices such as speakers or headphones are properly detected by the operating system, and that sufficient storage space is available for local music catalogs and model files.

Optional CUDA-enabled GPU drivers may be installed when accelerated convolutional neural network (CNN) inference or model training is required. GPU acceleration can significantly reduce processing time for emotion detection models. However, the system is also designed to operate in CPU-only mode, which enables deployment on standard desktop machines without specialized hardware.

Firewall and antivirus software configurations should permit local application execution if restrictions interfere with Flask server communication or file streaming operations. In some cases, security software may block local ports used by the Flask server, which can prevent the application interface from loading in the browser.

Developers should also verify that the localhost communication port (typically port 5000) is available and not occupied by another running service. If port conflicts occur, the Flask configuration can be modified to run on an alternative port.

Before running the application, developers should confirm that project directory paths, model storage locations, and catalog files are correctly configured according to deployment settings. Essential folders such as models/, songs/, song csv/, templates/, and static/ must exist and contain the required resources for the application to operate correctly.

Environment variables or configuration files may also be used to store customizable param- eters such as secret keys, model paths, emotion detection thresholds, and language settings. These configurations allow developers to adjust system behavior without modifying the core application code.

Proper environmental setup ensures that the emotion-based music recommendation system runs reliably across multiple machines while maintaining consistent functionality, performance, and security.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

### 7.4 Implementation of Modules

The system modules are implemented following a layered architectural design to ensure mod- ularity, maintainability, and scalability. The architecture separates processing responsibilities into detector interfaces, recommendation engines, database management utilities, and fron- tend controller components. This separation of concerns ensures that each module performs a specific function while interacting with other modules through clearly defined interfaces.

The layered architecture improves maintainability because changes in one module do not directly affect the functionality of other modules. For example, the emotion detection backend can be upgraded from a CNN-based model to a more advanced deep learning model without requiring modifications in the recommendation engine or frontend interface. Similarly, database storage mechanisms can be replaced or extended without impacting the emotion detection logic.

The emotion detection subsystem supports multiple backend implementations, including CNN- based classification models, heuristic feature-based rule engines, and optional DeepFace embedding- based recognition mapping. These detection methods are integrated behind a unified abstraction interface, allowing runtime backend switching without modifying application-level logic. The abstraction layer ensures that the frontend and recommendation modules receive consistent emotion outputs regardless of the backend detection method used.

Image preprocessing is performed before emotion detection inference. The preprocessing stage includes resizing input frames to match model input dimensions, normalizing pixel values, and optionally converting images to grayscale. These steps ensure that the machine learning models receive standardized input data, improving prediction accuracy and consistency across different hardware devices.

The database module is responsible for initializing and maintaining SQLite storage used for catalog management and authentication data storage. SQLite was selected because it provides a lightweight, file-based relational database system that is well suited for offline desktop ap- plications. The database stores essential information including song metadata, artist names, language categories, mood mappings, and user authentication credentials.

During system initialization, the database module loads song catalog data from structured CSV files. This process seeds the database with initial entries and performs validation checks on media file paths, song metadata consistency, and character encoding formats. Any missing files or inconsistent metadata entries are logged to assist developers in correcting catalog issues.

Recommendation engine endpoints process detected emotional states and generate filtered playlists based on three primary parameters: user mood classification, language preference selection, and availability of audio media files within local storage directories. These parameters ensure that the generated recommendations align with the user's emotional state and preferred listening language.

The recommendation algorithm first identifies songs mapped to the detected emotion category. After filtering songs based on emotion compatibility, additional filtering is applied to ensure that the songs match the user's language preference and that the associated audio files exist in the local media directory. This validation step prevents playback errors caused by missing media files.

Ranking logic is implemented to prioritize strong emotion-song matching scores, followed by secondary metrics such as playback frequency, implicit user preference signals, and catalog popularity indicators. Songs that strongly match the user's current emotional state are placed at the top of the recommendation list, while secondary ranking factors help refine playlist ordering to enhance user satisfaction.

Logging and diagnostic monitoring mechanisms are integrated throughout the modules to cap- ture runtime events such as camera initialization status, prediction confidence values, recommen- dation generation latency, database query execution time, and playback streaming performance. These logs provide valuable insights into system behavior and assist developers in identifying performance bottlenecks or functional errors.

The logging system also records system startup events, configuration loading, module initializa- tion status, and error occurrences. This structured logging approach allows developers to trace the sequence of operations during runtime and simplifies troubleshooting when unexpected system behavior occurs.

Error handling routines are implemented to manage exceptions related to webcam access failure, missing audio resources, invalid

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

model predictions, and filesystem inconsistencies. For example, if the webcam cannot be accessed due to driver issues or permission restrictions, the system displays a warning message and prevents emotion detection from proceeding.

Similarly, if an audio file referenced in the database cannot be located in the local storage directory, the recommendation engine automatically excludes that song from the playlist to avoid playback errors. These safeguards ensure that the application continues functioning even when certain resources are unavailable.

Configuration parameters such as detection thresholds, file path roots, language codes, and backend selection modes are centralized to support easy system customization and deployment flexibility. These parameters are stored in configuration files or environment variables so that developers can modify system behavior without editing core program logic.

This centralized configuration approach simplifies system maintenance and allows different de- ployment environments to use customized settings. For example, developers can adjust emotion detection confidence thresholds to balance prediction accuracy and system responsiveness.

The implementation of modular components combined with layered architecture ensures that the emotion-based music recommendation system remains flexible, maintainable, and adapt- able to future improvements. Additional features such as cloud synchronization, advanced machine learning models, mobile interfaces, and collaborative recommendation algorithms can be integrated into the system without major architectural changes.

**Song Recommendation Logic**

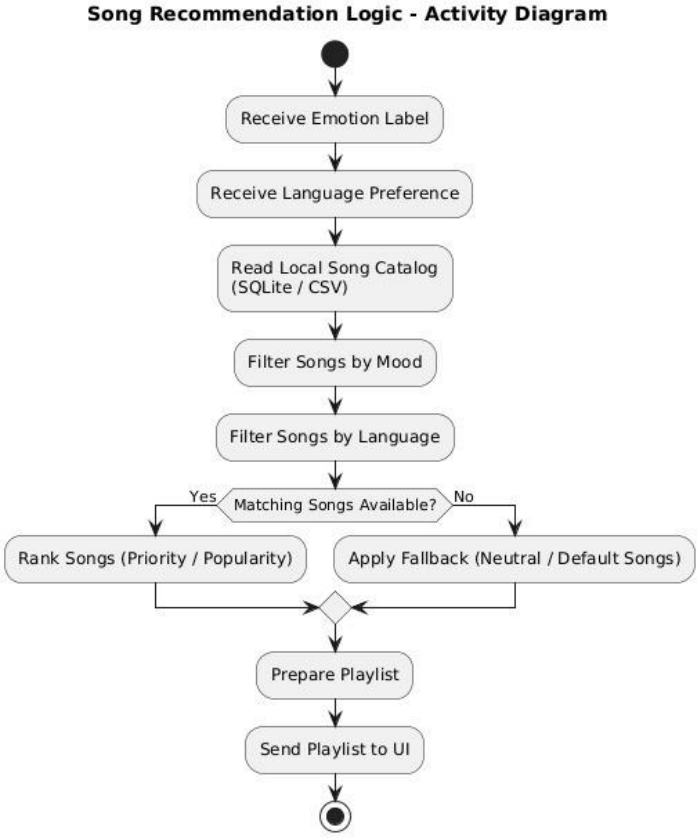



Figure 7.4.1: Song Recommendation Logic

The Song recommendation logic activity diagram represents the workflow of the song recom- mendation module in the emotion-based music recommendation system. The process begins by receiving the detected emotion label from the emotion detection pipeline along with the user-selected language preference.

The detected emotion is first validated to ensure that it belongs to a predefined set of supported mood categories such as Happy,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Sad, Angry, Relaxed, or Neutral. If the emotion label falls outside the supported categories due to prediction uncertainty, the system automatically maps it to the closest valid category to maintain consistency within the recommendation process.

After validating the emotion input, the recommendation module retrieves the song catalog from locally stored data sources. The system accesses the locally stored song catalog maintained using an SQLite database or CSV-based repositories. These repositories contain structured metadata for each song, including song title, artist name, language category, mood classification tag, file path location, and optional popularity indicators.

The catalog data is filtered based on two primary parameters: emotional mood category and language selection. The emotion filter ensures that only songs associated with the detected emotional state are selected, while the language filter restricts results to songs matching the user's preferred language.

During the filtering stage, the system also performs validation checks to confirm that each song entry has a valid media file path within the local storage directory. Songs that reference missing or corrupted files are automatically removed from the candidate list to prevent playback failures.

Once the filtering process is completed, the system checks whether matching songs are available in the catalog. If one or more matching songs are found, the recommendation engine proceeds to the ranking stage. Ranking algorithms are applied to determine the priority order of songs within the playlist.

The ranking logic evaluates several factors to improve the relevance of recommendations. The primary ranking criterion is the strength of the emotion-song mapping, ensuring that songs strongly associated with the detected emotional state appear first. Secondary factors include song popularity indicators, previous playback frequency, and implicit user preference signals stored in the system.

These ranking mechanisms allow the recommendation engine to generate playlists that are both emotionally relevant and engaging for the user. The ordered ranking ensures that the most suitable songs are presented at the top of the playlist while still maintaining variety in the recommendations.

If no matching songs are available after the filtering stage, the system activates a fallback mecha- nism to ensure that the user experience is not interrupted. In this situation, the recommendation module automatically selects default songs associated with the Neutral mood category. Neutral songs are chosen because they generally represent balanced or universally acceptable music that can suit multiple emotional contexts.

The fallback mechanism guarantees that the system always produces a valid playlist even when the catalog lacks specific songs mapped to a particular emotional state or language preference. This approach prevents empty recommendation lists and ensures continuous music playback availability.

After ranking or fallback processing is completed, the system constructs an ordered playlist data structure. The playlist contains metadata for each recommended song, including the song title, artist name, emotion category, language tag, and corresponding file path used for audio streaming.

The final step involves transmitting the prepared playlist structure to the user interface mod- ule. The frontend interface receives the recommendation results and dynamically displays the recommended songs to the user. Users can browse the playlist, view song details, and select individual tracks for playback.

When a user selects a song, the media streaming module retrieves the associated audio file from local storage and begins playback through the integrated HTML5 audio player. Playback controls such as play, pause, next track, and previous track allow users to interact with the playlist seamlessly.

Overall, the song recommendation workflow ensures efficient filtering, ranking, and fallback handling to provide reliable emotion-aware music recommendations. By combining emotion classification results with user language preferences and local catalog availability, the system delivers personalized playlists while maintaining a fully offline and privacy-preserving archi- tecture.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

**Emotion Detection Pipeline**

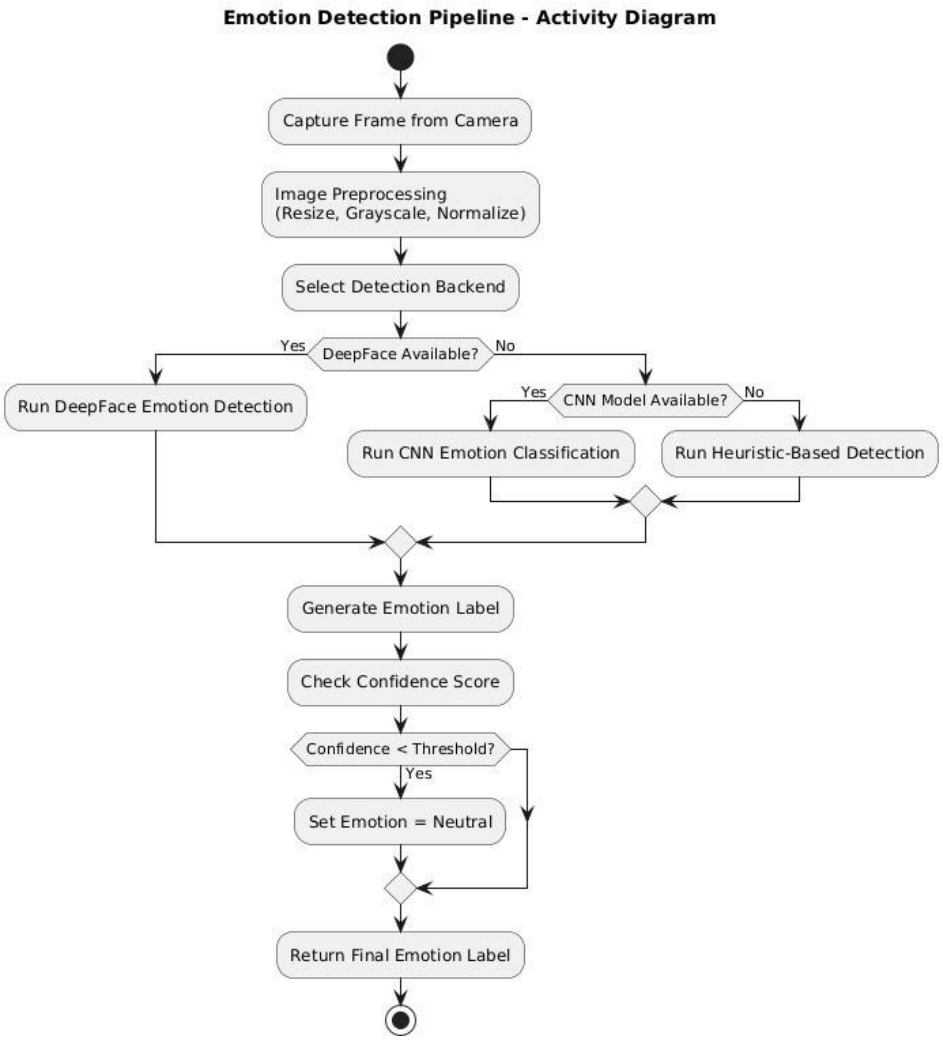



Figure 7.4.2 : Emotion Detection Pipeline

The Emotion detection pipeline activity diagram illustrates the workflow of the emotion de- tection pipeline used in the system. The process begins by capturing a facial frame from the webcam during user interaction. When the user activates the emotion detection feature, the browser requests permission to access the webcam through the system interface. Once permis- sion is granted, the webcam continuously captures frames, and one or more frames are selected for analysis.

The captured image undergoes preprocessing operations including resizing to the required input dimension, grayscale conversion, and normalization to improve prediction stability and model compatibility. Resizing ensures that the image matches the input dimensions required by the machine learning model, while grayscale conversion reduces computational complexity by eliminating unnecessary color channels. Pixel normalization is applied to scale intensity values to a standard range, which helps improve model performance and prediction consistency.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Before emotion classification is performed, the system may optionally apply face detection algorithms to isolate the facial region from the captured frame. This step helps remove irrelevant background information and focuses the model's attention on facial features such as eyes, eyebrows, and mouth, which are critical indicators of emotional expression.

After preprocessing, the system selects the appropriate emotion detection backend based on availability and configuration settings. The modular architecture allows multiple detection methods to be integrated within the same pipeline, enabling flexible deployment depending on system resources and installed libraries.

If the DeepFace framework is available, the system performs embedding-based facial emotion recognition using DeepFace models. DeepFace utilizes pre-trained deep neural networks to extract high-level facial embeddings and compare them against known emotion patterns. This approach provides high accuracy because it leverages advanced deep learning architectures trained on large facial expression datasets.

If DeepFace is not available, the system checks for the availability of the CNN classification model. The CNN-based emotion detection model is trained using labeled facial expression datasets and performs feature extraction using convolutional layers. These layers automatically learn spatial patterns from facial images, allowing the model to recognize emotional expressions such as happiness, sadness, anger, or surprise.

If the CNN model is available, CNN-based emotion classification is executed. The processed facial image is passed through the neural network layers, and the output layer produces probabil- ity scores for each supported emotion category. The emotion label with the highest probability score is selected as the predicted emotional state.

Otherwise, the system falls back to heuristic-based detection methods that analyze visual fea- tures such as brightness, contrast, and facial expression indicators. These heuristic techniques provide a lightweight alternative when machine learning models are unavailable or compu- tational resources are limited. Although heuristic methods may not achieve the same level of accuracy as deep learning models, they provide a basic estimation of emotional states to maintain system functionality.

The detection module generates a preliminary emotion label and computes the corresponding confidence score associated with the prediction. The confidence score represents the probability or reliability of the predicted emotion label based on the model's output distribution.

To ensure reliable recommendations, the system evaluates the confidence score against a pre- defined threshold value. If the confidence score exceeds the threshold, the detected emotion is considered valid and forwarded to the recommendation engine.

If the confidence score is below the predefined threshold value, the system assigns the default emotion state as $\{\text{Neutral}\}$ to maintain recommendation stability under uncertain prediction conditions. This fallback mechanism prevents incorrect emotional predictions from negatively affecting the recommendation results.

The validated emotion label is then transmitted to other system modules responsible for gener- ating personalized music recommendations. Along with the emotion label, the system may also store auxiliary information such as prediction timestamp and confidence values for logging and debugging purposes.

Finally, the pipeline returns the validated emotion label to the recommendation engine and user interface modules for playlist generation and playback control. The recommendation engine uses this emotion label to filter songs associated with the detected emotional category, and the user interface dynamically updates the playlist displayed to the user.

Overall, the emotion detection pipeline integrates computer vision preprocessing, machine learning inference, confidence validation, and fallback mechanisms to ensure robust and reliable emotion recognition. This pipeline plays a critical role in enabling the system to provide personalized music recommendations based on the user's emotional state while maintaining stability in offline environments.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

### 7.4.1 System Workflow and Implementation Screenshots

This subsection presents key implementation stages of the offline emotion-aware music rec- ommendation system. Screenshots illustrate the interaction between the graphical interface, backend services, and local media library. Together, these figures demonstrate how emotion detection, recommendation logic, and audio playback are integrated into a cohesive workflow.

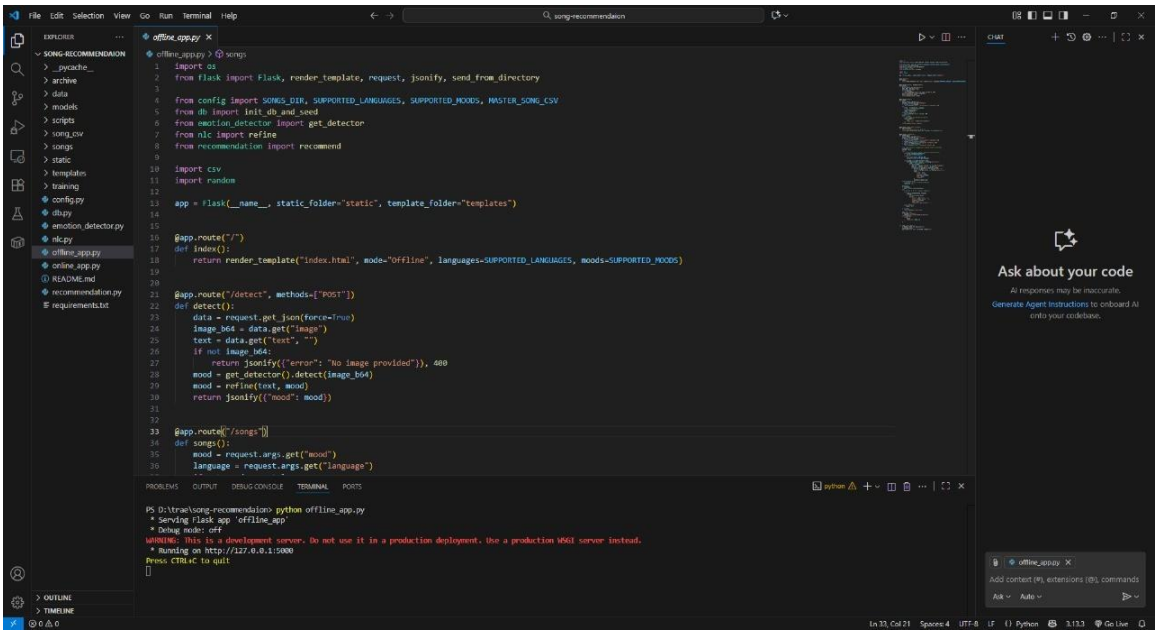



Figure 7.4.1.1 :Backend implementation of the offline Flask application.

Figure 7.4.1.1 shows the core backend logic written in Python using the Flask framework. The application exposes REST endpoints such as /detect for emotion detection and /songs for serving music recommendations. These endpoints act as the communication bridge between the frontend user interface and the backend processing modules. The /detect endpoint receives facial frame data captured from the webcam and forwards it to the emotion detection pipeline for analysis. After emotion classification is completed, the predicted emotion label is returned as a JSON response that can be used by the frontend interface to request music recommendations.

The /songs endpoint is responsible for retrieving and returning song recommendations based on the detected emotion and user language preferences. When this endpoint is triggered, the backend accesses the song catalog stored in the local database and filters the entries according to the detected mood category. The filtered songs are then ranked and returned to the client application in structured JSON format, enabling dynamic playlist generation within the user interface.

The modular design allows different emotion detection backends (heuristic, CNN, DeepFace) to be plugged in without altering higher–level logic. This design pattern follows the principle of abstraction, where the emotion detection functionality is encapsulated behind a common interface. As a result, developers can easily replace or upgrade the emotion recognition model without modifying the recommendation engine or the API routes.

The backend logic also includes initialization routines responsible for loading trained machine learning models, verifying catalog availability, and preparing database connections during application startup. This ensures that all required components are properly configured before the server begins handling user requests.

Static assets such as templates, JavaScript files, CSS stylesheets, and local audio files are orga- nized within structured project directories to support efficient offline operation. Flask's template rendering system is used to dynamically generate web pages while maintaining separation be- tween application logic and presentation layers.

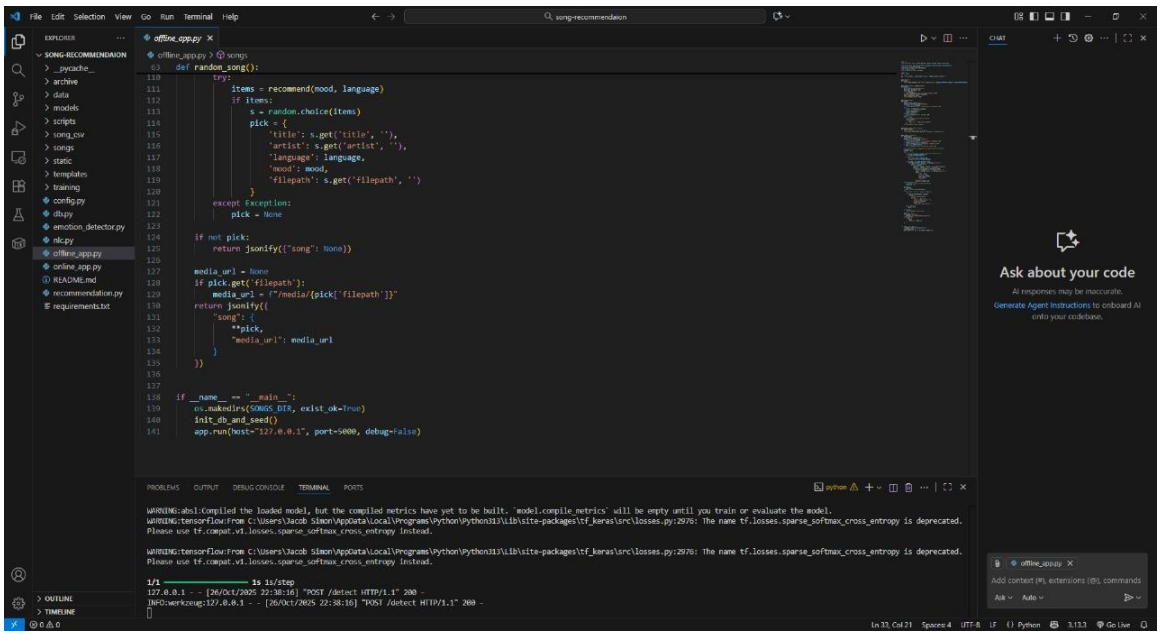
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Local audio files used for music playback are stored within a dedicated media directory. The backend includes secure routing mechanisms that validate file paths before serving audio content to the frontend audio player. This prevents unauthorized file access and ensures that only approved media resources are streamed through the application.

The offline-first design of the backend architecture ensures that all critical operations—including emotion detection, recommendation generation, and audio streaming—are executed locally without requiring internet connectivity. This approach significantly reduces latency, improves system responsiveness, and enhances user privacy by preventing external data transmission.

Additionally, the Flask backend incorporates logging mechanisms that record system events such as API requests, prediction outcomes, and playback activity. These logs assist developers in debugging issues, monitoring application performance, and evaluating the accuracy of emotion- based recommendations.

Overall, the backend implementation provides a flexible, modular, and efficient foundation for the emotion-based music recommendation system. By combining RESTful API design with local machine learning inference and offline media management, the system achieves reliable performance while maintaining scalability for future enhancements.




7.4.1.2 Random song selection logic based on mood and language filters.

Figure 7.4.1.2 highlights the recommendation routine responsible for generating personalized music suggestions based on the user's emotional state. The recommendation process begins when the validated emotion label and the user-selected language preference are received from the emotion detection module and the user interface.

The detected mood and selected language are mapped to candidate songs stored in the local database or catalog repository. Each song entry in the catalog contains metadata such as song title, artist name, emotion tag, language category, and the corresponding file path of the audio resource. These attributes enable the system to efficiently filter songs that match the user's emotional context and language preference.

The filtering stage eliminates songs that do not satisfy both parameters. First, the system selects songs whose emotion tag corresponds to the detected emotional state. After emotion-based filtering, the algorithm applies the language constraint to ensure that the recommended songs align with the user's preferred listening language. This two-stage filtering mechanism improves recommendation accuracy and ensures cultural and linguistic relevance.

When multiple items match the filtering conditions, the algorithm performs randomized selec- tion to prevent repetitive recommendations while preserving relevance. Randomization ensures that users are not repeatedly presented with the same songs during similar emotional states. This approach improves user engagement by introducing variation in the generated playlists while still maintaining emotional compatibility.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

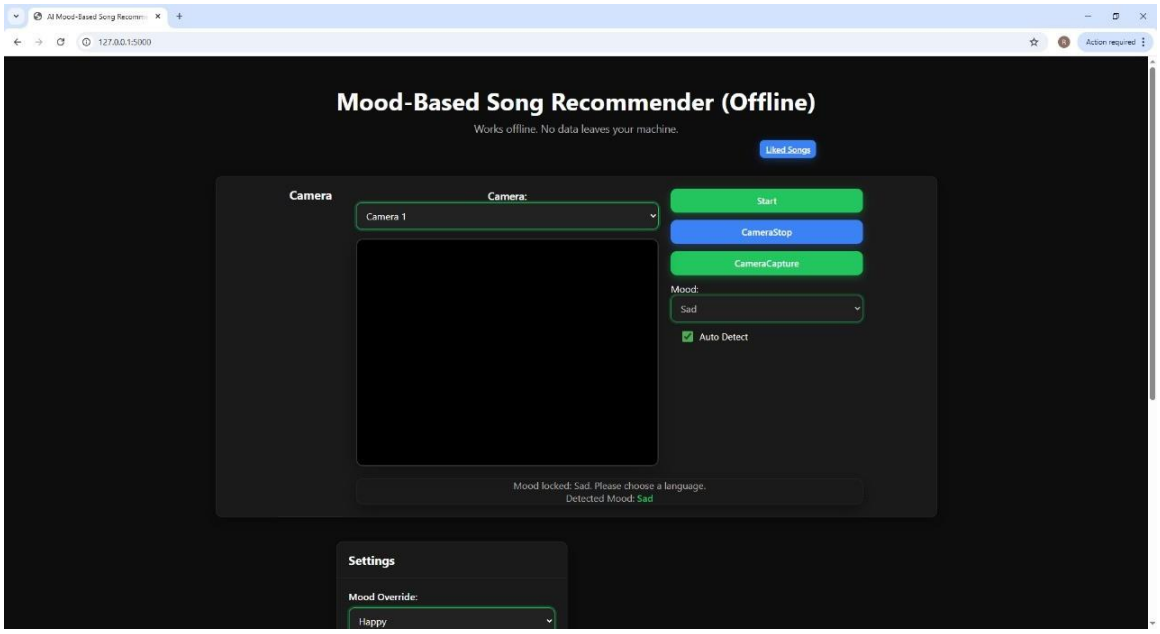
In addition to randomization, the recommendation routine may optionally consider supple- mentary ranking factors such as catalog popularity, historical playback patterns, and implicit user preference signals. Although the system operates primarily in offline mode, lightweight preference tracking can help refine playlist ordering and enhance the overall listening experience.

Before generating the final playlist, the system performs validation checks on the selected songs to ensure that the associated audio files are available within the local media directory. Any catalog entries referencing missing or inaccessible files are automatically removed from the candidate list to prevent playback interruptions.

Fallback handling is implemented to prevent errors when songs are missing or incorrectly tagged, improving robustness of the system. If the filtering process results in an empty candidate list, the system automatically switches to a fallback strategy by selecting songs associated with a default mood category, typically Neutral. This ensures that the application always produces a playable recommendation list even when catalog limitations exist.

After the recommendation routine completes filtering, randomization, and validation stages, the final list of songs is organized into a structured playlist. The playlist is then transmitted to the frontend interface where users can browse the recommended tracks, view song information, and control playback through the integrated audio player.

Overall, the recommendation routine is designed to be lightweight, efficient, and reliable for offline deployment environments. By combining emotion-aware filtering, language preference matching, randomized selection, and fallback mechanisms, the system provides a balanced approach to generating diverse yet emotionally relevant music recommendations.




7.4.1.3 Application startup logs and diagnostic information.

Figure 7.4.1.3 presents diagnostic logs generated during the execution of the emotion-based music recommendation system. These logs provide detailed information about the internal operations of the application and help developers monitor the behavior of different modules during runtime.

The logging process begins when the application starts executing the backend server. Initial log entries confirm successful initialization of the machine learning models used for emotion detection. These messages indicate whether the CNN model, DeepFace framework, or heuristic detection module has been successfully loaded into memory. This verification step is impor- tant because it ensures that the emotion detection pipeline is ready before the system begins processing user requests.

Additional log entries record the loading and verification of the local media directory containing the audio files used for music playback. During this stage, the system checks the accessibility of the configured storage path and verifies the presence of audio resources referenced in the catalog database. If any missing files or incorrect directory paths are detected, warning messages are

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

generated and recorded in the logs to alert developers.

The logs also confirm the activation of the Flask development server and the initialization of REST API routes such as /detect and /songs. These entries indicate that the backend service is ready to accept incoming requests from the frontend interface. Information such as the server host address, port number, and application mode is also displayed to assist in debugging and deployment verification.

Warnings generated by machine learning libraries such as TensorFlow are recorded to assist developers in identifying performance or compatibility issues. These warnings may include messages related to CPU optimization, missing GPU acceleration, deprecated functions, or library version compatibility concerns. Although these warnings do not necessarily indicate critical failures, they provide useful insights for improving system performance and ensuring stable execution across different environments.

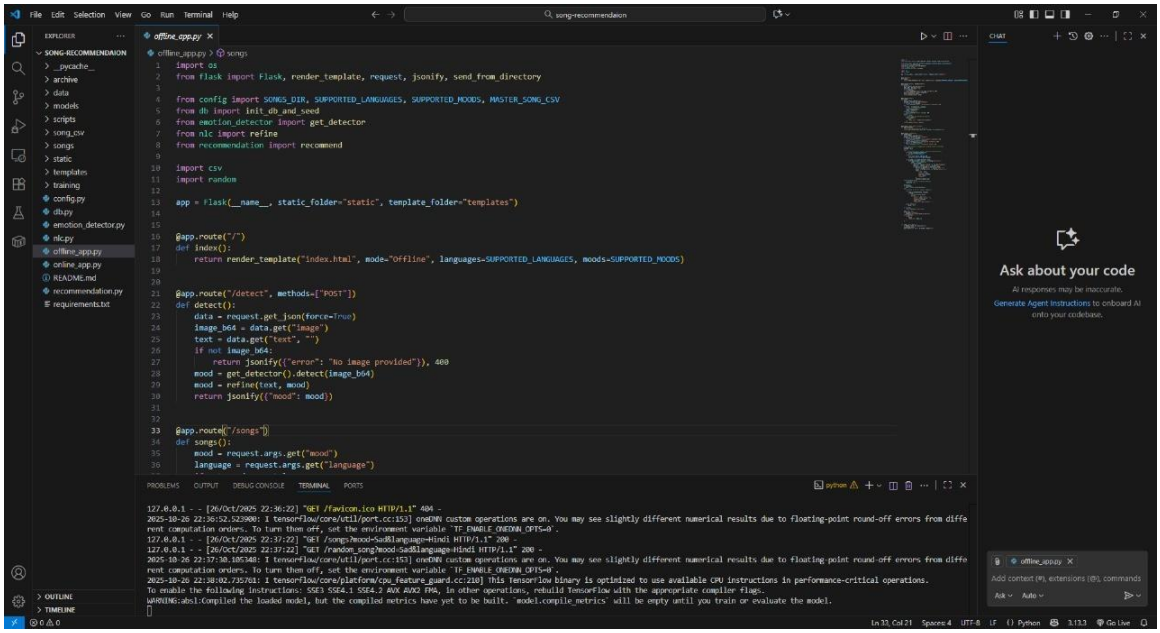
In addition to initialization messages, runtime logs capture events related to emotion detection, recommendation generation, and media playback operations. For example, when the webcam captures an image frame, a log entry may record the frame processing event and the predicted emotion label generated by the model. Similarly, when the recommendation engine retrieves songs from the catalog, the system logs the filtering results and playlist generation status.

Error logs are also generated whenever exceptions occur during system execution. These errors may arise due to webcam access issues, missing audio files, invalid configuration parameters, or database inconsistencies. Capturing these events in structured log files allows developers to quickly identify and resolve problems encountered during system testing.

Logging also enables reproducibility and traceability throughout the testing process. By pre- serving a chronological record of system events, developers can reconstruct the sequence of operations that occurred during application execution. This capability is particularly useful when evaluating model behavior, verifying system performance, or diagnosing unexpected results.

Furthermore, diagnostic logs support long-term system maintenance and future improvements. Developers can analyze historical log data to identify common failure patterns, performance bottlenecks, or areas where the system may require optimization.

Overall, the logging mechanism plays a critical role in ensuring transparency, reliability, and maintainability of the emotion-based music recommendation system. By providing detailed runtime information and error reporting, the logging framework assists developers in debugging issues, improving system stability, and validating the correctness of the implemented modules.




7.4.1.4 Request handling and communication between UI and backend.

Figure 7.4.1.4 illustrates how incoming HTTP requests are processed within the emotion-based music recommendation system. The

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

diagram explains the flow of communication between the client-side interface running in the browser and the backend server implemented using the Flask framework.

The process begins when the user interacts with the web application through the browser interface. The browser captures facial images using the webcam through the getUserMedia API and collects user inputs such as language preference or playback selections. These inputs are then packaged into HTTP requests and transmitted to the backend server through predefined API endpoints.

Once the request reaches the backend server, Flask routing mechanisms identify the appropriate endpoint responsible for handling the request. For example, facial image data is forwarded to the emotion detection module through the /detect API route, while requests for music recommendations are directed to the /songs endpoint.

The backend server first performs validation checks on incoming data to ensure that the request contains the required parameters. This validation process verifies that the facial image data is properly formatted and that the necessary user preferences are included. Any malformed or incomplete requests are rejected with appropriate error messages to maintain system stability.

After validation, the emotion detection pipeline processes the received facial image using the selected detection backend such as DeepFace, CNN classification, or heuristic-based analysis. The model analyzes facial features and produces an emotion prediction along with an associated confidence score. The predicted emotion label is then verified against the confidence threshold to ensure reliable classification results.

Once the emotion label is finalized, the backend invokes the recommendation engine to retrieve suitable songs from the local catalog. The recommendation module filters songs based on the detected emotional category and the user's language preference. Matching songs are ranked, validated, and organized into a structured playlist ready for playback.

After completing the recommendation process, the backend prepares a response message in JSON (JavaScript Object Notation) format. The JSON response contains key information such as the detected emotion label, confidence value, and the list of recommended songs including metadata such as title, artist name, language, and audio file paths.

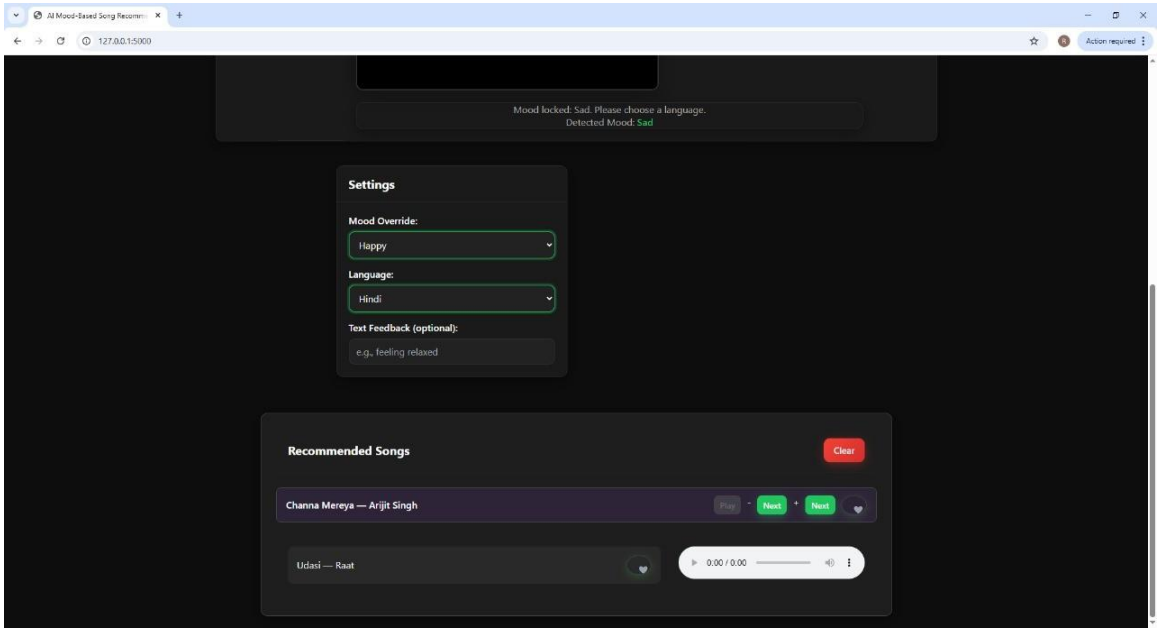
This structured JSON response is then transmitted back to the browser as the HTTP response. The frontend interface receives the response and dynamically updates the user interface to display the detected emotion and the generated playlist. Users can then select songs from the playlist for playback through the integrated HTML5 audio player.

The entire request–response cycle occurs within a short time interval, ensuring a responsive user experience during emotion detection and music recommendation. Because all operations are executed locally, the system maintains low latency while protecting user privacy by avoiding external data transmission.

This design follows a REST (Representational State Transfer) architecture, where system re- sources such as emotion detection services and song recommendation endpoints are accessed through stateless HTTP requests. The RESTful architecture simplifies communication between frontend and backend components while maintaining modularity and scalability.

By following REST principles, the system becomes portable and easily extendable to other platforms. The same backend services could be accessed by mobile applications, desktop clients, or web-based interfaces with minimal modifications. This flexibility allows the emotion-based music recommendation system to evolve into a cross-platform solution capable of supporting future enhancements and broader deployment environments.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026




7.4.1.5 User interface showing mood override, language selection, and recommended songs.

Figure 7.4.1.5 displays the graphical user interface of the emotion-based music recommendation system. The interface serves as the primary interaction point between the user and the applica- tion, allowing users to capture emotions, view recommendations, and control music playback in a simple and intuitive manner.

The layout of the interface is designed to provide a clean and organized presentation of the system's features. The main screen typically includes sections for emotion detection, language preference selection, recommended songs, and playback controls. Each section is arranged in a logical sequence so that users can easily understand the workflow of the application.

Users may rely on automatic facial emotion detection or manually override the detected mood using dropdown menus. The automatic detection mode uses the webcam to capture facial expressions and determine the user's emotional state using the emotion recognition pipeline. However, users who prefer manual control can select their desired mood category from a dropdown list. This manual override feature ensures that users can still receive suitable music recommendations even if the emotion detection system produces uncertain predictions.

The interface also includes language selection filters that allow users to limit recommendations to specific song categories. Language filters are useful for multilingual users who prefer listening to music in particular languages. Once a language is selected, the recommendation engine retrieves songs that match both the emotional state and the selected language preference.

Recommended songs are displayed in a structured playlist format within the interface. Each song entry typically includes the song title, artist name, and corresponding playback control options. Users can select songs directly from the list and start playback through the integrated HTML5 audio player.

Playback controls such as play, pause, next, and previous buttons allow users to manage music playback conveniently. Additional controls such as volume adjustment and progress tracking may also be available depending on the implementation of the audio player component.

The interface also provides options to clear the queue or refresh the recommendation list. Clearing the queue removes all currently listed songs from the playlist, allowing users to generate a new set of recommendations based on updated emotions or preferences. This functionality improves user control over the music selection process.

Visual feedback is provided to inform users about system actions such as camera activation, emotion detection results, and recommendation updates. For example, once the emotion detec- tion process is completed, the detected mood may be displayed prominently on the screen so that users understand how the recommendations were generated.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The interface prioritizes clarity, minimal interaction burden, and consistent behavior between online and offline modes. Since the system is designed primarily for offline execution, all interface elements function without requiring internet connectivity. This ensures that users experience consistent performance regardless of network availability.

Responsive design principles are applied to ensure that the interface remains usable across different screen resolutions and device types. The layout adapts to varying window sizes while maintaining the readability of text elements and accessibility of control buttons.

Overall, the user interface plays a crucial role in delivering a seamless and engaging user experience. By combining automated emotion detection, manual control options, language filtering, and intuitive playback features, the interface enables users to interact effectively with the emotion-based music recommendation system while maintaining simplicity and usability.

**Summary:** Together, these figures demonstrate the workflow: capturing facial expressions, classifying emotion, filtering songs by mood and language, and enabling fast offline playback. This modular, privacy-preserving architecture supports future expansion of datasets, models, and features without disrupting user experience.

### 7.5 Integration and Development

The integration phase focuses on combining frontend interaction components, backend process- ing modules, and machine learning inference pipelines into a unified application system. The objective is to ensure smooth communication between user interface events, inference engines, database services, and media playback controllers.

#### 7.5.1 Backend and Frontend Integration

Backend Flask routes communicate with frontend JavaScript components through JSON-based API calls to enable seamless interaction between user interface events and server-side processing modules. The primary functional endpoints of the system include emotion detection inference requests, personalized recommendation retrieval services, offline authentication validation, and secure media streaming operations.

Camera capture events initiated from the frontend interface trigger backend emotion detection processing pipelines. The captured facial frame is transmitted to the inference module where preprocessing, feature extraction, and emotion classification are performed. The backend system determines the user's emotional state and generates playlist recommendation data corresponding to the detected mood category.

The communication architecture follows an asynchronous request-response model to improve system responsiveness and reduce perceived latency during computationally intensive tasks such as CNN inference execution and database filtering operations. JSON-based data serialization is used to maintain lightweight data transfer structures between frontend and backend layers.

Frontend JavaScript modules handle user interaction events such as camera activation, playback control actions, language selection, and playlist rendering. The frontend interface dynamically updates UI components based on backend response data.

Route protection strategies are implemented in offline execution mode to restrict unauthorized external requests and prevent exposure of internal processing endpoints. The system restricts network communication by default and performs all inference and recommendation computa- tions locally to preserve user privacy.

Input validation mechanisms are incorporated at API endpoints to handle malformed requests, missing frame data, corrupted media references, and unexpected parameter values. Appropriate error messages are returned to the frontend interface to ensure graceful user experience during failure scenarios.

Session-level interaction flow is maintained between frontend UI state and backend processing logic to support continuous emotion monitoring and adaptive playlist regeneration during user activity.

#### 7.5.2 Development Process

The development process follows a structured pipeline-based workflow that ensures systematic software construction, model integration, and deployment readiness. The workflow consists of environment initialization, dataset preparation, model training,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

inference integration, testing, optimization, and final packaging.

Dataset preparation involves CSV catalog validation, removal of corrupted entries, normaliza- tion of image datasets, and consistent mapping of emotion labels into application-specific mood categories. Metadata verification is performed to ensure song title, artist name, language code, and media file path integrity.

CNN model training scripts are used to generate optimized emotion detection models suitable for real-time inference. During training, performance indicators such as accuracy, precision, recall, F1-score, and loss convergence trends are monitored. Confusion matrix visualizations and evaluation reports are generated to analyze class-wise prediction behavior.

Data augmentation techniques including controlled rotation, horizontal flipping, brightness scaling, and normalization preprocessing are applied to improve model generalization and robustness against environmental variations.

Inference integration involves embedding the trained model into the backend detection pipeline. The system supports model loading, tensor preprocessing, prediction scoring, and confidence threshold validation during runtime execution.

Deployment packaging is performed using tools such as PyInstaller to generate standalone executable applications for offline distribution. The packaged build contains Python runtime environments, machine learning models, Flask backend services, frontend templates, static assets, and audio catalog files.

Software quality assurance is maintained through automated linting tools, unit testing frame- works, and integration testing pipelines. Version control systems are used to track source code modifications, while dependency version locking ensures reproducible builds across multiple machines.

Centralized configuration management is implemented to control detection thresholds, backend selection modes, file path roots, and language filtering parameters.

Performance profiling is conducted during development to measure inference latency, mem- ory consumption, and database query execution time. Optimization strategies such as model pruning, batch inference tuning, and caching of normalization parameters are applied where necessary.

Release builds are validated on clean testing environments before distribution to ensure stability, compatibility, and error-free execution.

### 7.5.3 Challenges

Major technical challenges during development include handling webcam hardware variabil- ity across different devices, maintaining emotion prediction stability under varying lighting conditions, and ensuring safe filesystem path handling on Windows operating systems.

Additional challenges involve managing class imbalance in emotion datasets, resolving incon- sistencies in media catalog metadata, and optimizing inference latency for real-time interaction.

Low-resource hardware environments may experience computational bottlenecks during CNN inference execution. To mitigate this, heuristic fallback detection and model optimization strategies are implemented.

Maintaining fully offline execution while preserving recommendation accuracy presents a design constraint that requires careful pipeline optimization.

System robustness is further affected by factors such as camera occlusion, blurred frames, audio file corruption, and unexpected user interaction patterns.

## 8. Testing

### 8.1 Testing Objective

The primary objective of testing is to ensure the correctness, reliability, and robustness of the emotion-based music recommendation system. Testing is conducted to verify that all functional and non-functional requirements are satisfied across different system modules.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

The testing process validates the performance of core components including the emotion detec- tion module, recommendation engine, database management subsystem, authentication mech- anism, frontend interaction interface, and media streaming controller.

Functional verification ensures that facial frame capture, preprocessing operations, emotion classification, playlist filtering, and playback control mechanisms operate accurately under normal and boundary conditions.

Reliability testing is performed to confirm stable system execution during prolonged usage, repeated inference cycles, and continuous camera monitoring scenarios.

Robustness evaluation focuses on handling real-world uncertainties such as lighting variations, webcam quality differences, facial occlusion, blurred frame inputs, missing catalog entries, and corrupted media files.

Security and privacy validation is also considered by ensuring that all computational processing is performed locally without transmitting user data outside the system environment.

Latency and responsiveness are assessed by measuring emotion detection inference time, rec- ommendation retrieval delay, and audio streaming response speed.

The overall testing objective is to deliver a user-friendly, privacy-preserving, and computation- ally efficient application suitable for offline real-time deployment.

### 8.2 Testing Strategies

The system adopts a structured hierarchical testing strategy to ensure comprehensive validation of all functional components and processing pipelines. Testing is performed across multiple levels including unit testing, integration testing, system testing, and user interface validation.

The testing scope covers preprocessing operations, emotion detection backend modules, recom- mendation filtering logic, database query processing, API routing mechanisms, authentication validation, and playback control functionalities.

Pipeline-based end-to-end workflow verification is performed using the sequence: facial frame capture → preprocessing → emotion classification → playlist recommendation retrieval → audio streaming execution.

Unit-level testing focuses on validating individual functions such as label mapping, CSV cat- alog parsing, normalization preprocessing, database seeding routines, and threshold-based confidence filtering.

Integration testing ensures proper communication between frontend JavaScript components and backend Flask routes using JSON-based data exchange protocols.

System-level testing evaluates the complete application workflow including offline authentica- tion, emotion detection inference, recommendation generation, and media playback manage- ment.

Manual interface testing is performed to verify webcam initialization behavior, camera per- mission prompts, language selection controls, playback interaction events, and responsive UI rendering across supported browsers such as Google Chrome and Microsoft Edge.

Special attention is given to real-time interaction scenarios where continuous frame capture and emotion monitoring are required. Testing also includes validation of error handling responses for invalid camera input, missing media resources, and backend processing failures.

Performance-related strategy considerations include measurement of detection latency, recom- mendation computation delay, and streaming response time under typical hardware configura- tions.

Regression testing is conducted after model updates, dependency modifications, or packaging changes to ensure system stability across versions.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

### 8.3 Testing Methods

- **Unit Testing:** Unit testing is performed to validate individual functional components of the system. This includes verification of label mapping logic, CSV catalog parsing rou- tines, preprocessing pipelines, database query execution functions, confidence threshold handling, and emotion classification utility methods.
- **Integration Testing:** Integration testing evaluates communication and data exchange between backend Flask endpoints and frontend JavaScript modules. The primary routes tested include `/detect`, `/songs`, and `/media`. These tests ensure correct JSON response formatting, playlist filtering accuracy, and media streaming consistency.
- **System Testing:** System testing is conducted to verify the complete operational workflow of the application. The workflow includes offline authentication, webcam frame capture, emotion detection inference, recommendation generation, playlist ranking, and audio playback streaming. The objective is to ensure that all modules function cohesively under real-world execution scenarios.
- **Manual Testing:** Manual testing is performed to evaluate graphical user interface behav- ior, webcam capture stability, playback control responsiveness, and browser compatibility. Testing is carried out primarily on Google Chrome and Microsoft Edge browsers with camera permission enabled.
- **Performance Testing:** Performance testing measures system efficiency by evaluating emotion detection inference latency, recommendation retrieval time, memory consump- tion, and audio streaming response delay under typical hardware configurations.
- **Regression Testing:** Regression testing is conducted after updates to machine learning models, libraries, database catalogs, or deployment packaging to ensure that existing functionality is not adversely affected.
- **Error Handling Testing:** Error handling scenarios are tested by simulating invalid webcam input, corrupted media files, missing catalog entries, prediction confidence failures, and authentication denial cases.
- **Validation Testing:** Validation testing confirms that system outputs satisfy functional requirements such as correct mood classification, language-based filtering, and accurate playlist ordering.

### 8.4 Test Case Specifications

Test cases are systematically designed to evaluate normal operational scenarios, boundary conditions, and error-handling situations of the emotion-based music recommendation system. Each test case is verified to ensure functional correctness, stability, and user-friendly behavior.

- Webcam frame capture functionality must correctly process inputs under varying lighting conditions, including blurred frames, low-light environments, and partial facial occlusion. If prediction confidence is below the threshold, fallback detection logic should assign the **Neutral** emotion state.
- Catalog validation tests must detect and report missing, duplicate, or malformed metadata entries in CSV or SQLite database storage. Required fields such as song title, artist name, mood label, language code, and media file path must be verified.
- Playback control operations including play, pause, stop, next track, previous track, and repeat/loop functions must respond accurately to user commands without delay or syn- chronization errors.
- Mood classification mapping and language filtering mechanisms must generate consistent and deterministic recommendation outputs for identical input conditions.
- Authentication test cases must ensure that invalid login attempts, corrupted credential entries, or unauthorized access requests are properly blocked.
- Media streaming tests must verify that missing, corrupted, or inaccessible audio files trigger graceful error handling and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

user notification messages rather than system crashes.

- Backend API response validation must confirm that JSON outputs follow predefined schema structures and contain required recommendation metadata.
- Threshold-based prediction tests must evaluate detector behavior at boundary confidence values to verify stable classification switching logic.
- Browser interaction tests must verify camera permission prompts, UI rendering consis- tency, and event handling across supported environments.
- Memory utilization and inference latency test cases must be monitored during repeated detection cycles to ensure system efficiency on low-resource hardware.
- Playlist regeneration tests must verify that changes in detected emotion or language preference dynamically update recommendation outputs.
- Filesystem path validation tests must confirm safe handling of Windows directory sepa- rators and restricted filename characters.

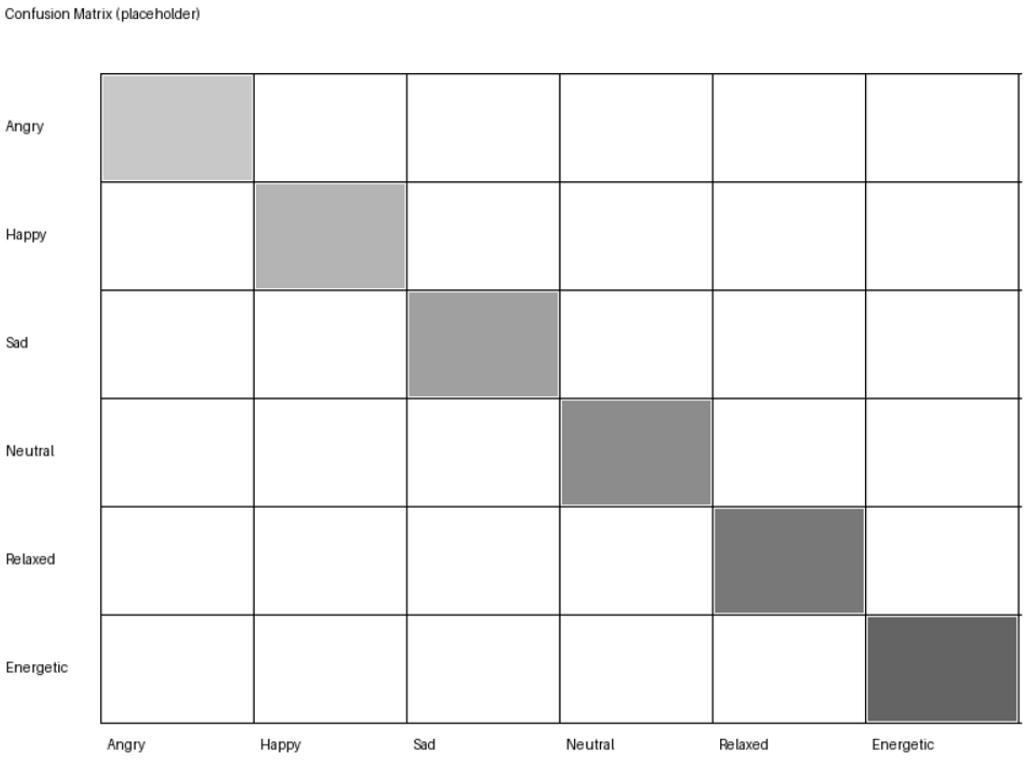



Figure 8.4.1 : Confusion matrix

The confusion matrix summarizes per-class performance, revealing confusions between closely related moods (e.g., Neutral vs Relaxed). Values are placeholders when evaluation is out of scope; they are replaced when test runs are available.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

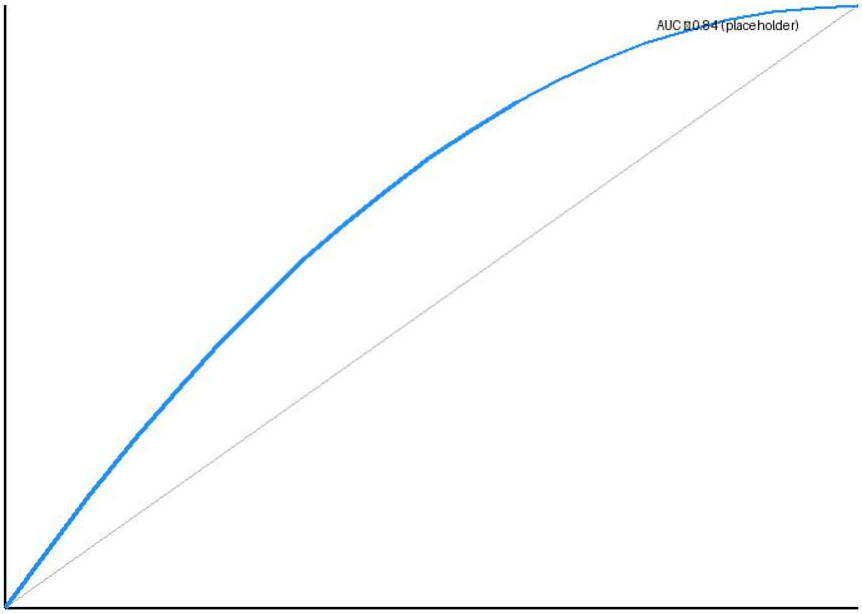



Figure 8.4.2: ROC curve.

The ROC curve illustrates the trade-off between true positive rate and false positive rate across thresholds. The AUC offers a single-number summary; placeholder values can be replaced by empirical results during evaluation.

### 8.5 Test Scanning

Continuous scanning and monitoring are performed throughout the development, testing, and deployment phases to ensure system reliability, stability, and performance consistency.

- Performance scanning evaluates key execution metrics including emotion detection infer- ence latency, recommendation retrieval time, database query execution delay, and audio streaming response response time under typical hardware conditions.
- Integration scanning validates the complete operational pipeline of the application, follow- ing the sequence: offline authentication → webcam frame capture → emotion detection
  → playlist recommendation generation → media playback streaming.
- Automated quality assurance pipelines may include static code analysis, linting checks, unit test execution, dependency verification, and documentation consistency validation.
- Code coverage monitoring is recommended to maintain approximately 70% or higher coverage for critical functional modules including detector logic, recommendation engine, and database operations.
- Regression scanning is performed after updates to machine learning models, framework libraries, configuration parameters, or deployment packaging to ensure backward com- patibility.
- Memory usage, CPU utilization, and inference computation load are monitored during repeated detection cycles to detect potential resource leaks or performance degradation.
- Packaging validation scanning ensures that executable builds generated using PyInstaller or similar tools function correctly on clean Windows environments.
- Dataset and catalog integrity scanning verifies the consistency of CSV files, SQLite database entries, and media file references.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Error log scanning is performed to identify recurring runtime exceptions related to camera access, filesystem operations, or model prediction failures.
- User interaction simulation tests may be executed to evaluate system behavior under continuous playback and emotion monitoring scenarios.
-

## 9. CONCLUSION AND FUTURE ENHANCEMENTS

The developed emotion-based music recommendation system provides a privacy-preserving, offline-capable platform for personalized music playback based on real-time facial emotion detection. The system integrates multiple detection backends, including CNN-based classifiers, heuristic feature analysis, and optional DeepFace embedding mapping, to improve classification robustness.

The modular architecture ensures flexibility, maintainability, and extensibility. Local database storage using SQLite and structured CSV catalogs enables efficient song metadata management and fast recommendation retrieval. The system emphasizes user privacy by performing all computations locally without transmitting data over external networks.

The frontend interface is designed to be simple and user-friendly, allowing users to easily control camera capture, language selection, and playback operations. Documentation, diagrams, and testing procedures improve system maintainability and usability.

Performance evaluation indicates stable inference latency under typical hardware configurations. While some classification confusion may occur between closely related emotional classes, threshold-based decision logic improves recommendation stability.

Overall, the system demonstrates a practical implementation of an end-to-end mood-aware media recommendation pipeline suitable for educational, research, and real-world offline de- ployment scenarios.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

## REFERENCES

1. P. Ekman, "An Argument for Basic Emotions," *Cognition and Emotion*, vol. 6, no. 3–4, pp. 169–200, 1992.
2. A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convo- lutional Neural Networks," *Advances in Neural Information Processing Systems (NIPS)*, 2012.
3. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
4. Y. Kim et al., "Music Emotion Recognition: A State of the Art Review," *Proceedings of the International Society for Music Information Retrieval (ISMIR)*, 2010.
5. F. Ricci, L. Rokach, and B. Shapira, *Recommender Systems Handbook*, Springer, 2015.
6. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.
7. A. Mollahosseini, D. Chan, and M. H. Mahoor, "Going Deeper in Facial Expression Recognition Using Deep Neural Networks," *IEEE Winter Conference on Applications of Computer Vision*, 2016.
8. S. Li and W. Deng, "Deep Facial Expression Recognition: A Survey," *IEEE Transactions on Affective Computing*, 2022.
9. H. Lee and K. Lee, "Music Recommendation System Using Emotion Recognition Based on Deep Learning," *IEEE Access*, 2021.
10. S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep Learning Based Recommender System: A Survey and New Perspectives," *ACM Computing Surveys*, 2019.
11. R. Panda, R. Malheiro, and R. Paiva, "Novel Audio Features for Music Emotion Recog- nition," *IEEE Transactions on Affective Computing*, 2020.
12. B. Schuller, S. Steidl, and A. Batliner, "The INTERSPEECH 2009 Emotion Challenge," *INTERSPEECH*, 2009.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

## APPENDIX

### A. Source Code

The above screenshots illustrate the core backend implementation of the offline mood-based music recommendation system developed using Flask. The displayed code sections represent the foundational architecture of the application and are described below.

```
import os
import json
from pathlib import Path

from flask import (
    Flask,
    render_template,
    request,
    redirect,
    url_for,
    session,
    jsonify,
    send_from_directory,
)
from config import SUPPORTED_LANGUAGES
from db import get_conn, init_db_and_seed, query_songs
from recommendation import recommend
from emotion_detector import get_detector


# --- App Setup ---------------------------------------------------------------
APP_ROOT = Path(__file__).parent.resolve()
TEMPLATES_DIR = APP_ROOT / "templates"
STATIC_DIR = APP_ROOT / "static"
DATA_DIR = APP_ROOT / "data"
USERS_FILE = DATA_DIR / "users.json"

app = Flask(
    __name__, template_folder=str(TEMPLATES_DIR), static_folder=str(STATIC_DIR)
)

# Deterministic secret key so sessions work across reloads unless changed via env
app.secret_key = os.environ.get("OFFLINE_SECRET_KEY", "offline-secret-key")
app.config["SESSION_COOKIE_NAME"] = "offline_app_session"
app.config["SESSION_PERMANENT"] = False  # expire on browser close
```

Figure A.1 :Import Statements and Application Initialization.

**1. Import Statements and Application Initialization** The first screenshot shows the neces- sary module imports and initial application setup.

- Standard Python modules such as os, json, and pathlib are used for file handling and environment configuration.
- Flask framework components including Flask, render template, request, redirect, url for, session, jsonify, and send from directory are imported to handle rout- ing, templating, session management, and API responses.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Custom modules such as:
  - config – defines supported languages,
  - db – manages database connection and queries,
  - recommendation – implements song recommendation logic,
  - emotion detector – handles mood detection from images, are integrated into the application.

The application directories (templates, static, and data) are dynamically configured using Path.resolve() to ensure portability. Session configuration includes:

- A deterministic secret key for secure session handling.
- A custom session cookie name.
- Non-permanent session behavior (expires on browser close).

### 2. User Store Helper Functions

The second screenshot presents helper functions responsible for user management using a local JSON file.

```python
# --- User Store Helpers ------------------------------------------------
def ensure_users_file():
    DATA_DIR.mkdir(parents=True, exist_ok=True)
    if not USERS_FILE.exists():
        USERS_FILE.write_text(json.dumps({"users": [{"username": "admin", "password": "admin"}]}), encoding="utf-8")


def load_users():
    ensure_users_file()
    with USERS_FILE.open("r", encoding="utf-8") as f:
        return json.load(f).get("users", [])


def save_users(users):
    USERS_FILE.write_text(json.dumps({"users": users}, indent=2), encoding="utf-8")


def validate_credentials(username, password):
    users = load_users()
    for u in users:
        if u.get("username") == username and u.get("password") == password:
            return True
    return False


def user_exists(username):
    return any(u.get("username") == username for u in load_users())


def is_authenticated():
    return bool(session.get("user"))
```

Figure A.2 :User Store Helper Functions.

- ensure users file() – Creates the users.json file with a default admin account if it does not exist.
- load users() – Reads and loads user records from the JSON file.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- save_users() – Writes updated user data back to storage.
- validate_credentials() – Authenticates login credentials.
- user_exists() – Prevents duplicate registrations.
- is_authenticated() – Checks whether a user session is active.

This design enables a fully offline authentication system without requiring an external database for user accounts.

### 3. Global Authentication Guard

The third screenshot shows the implementation of a global request guard using Flask's @app.before_request decorator.

```python
# --- Global Guard ------------------------------------------------------------
@app.before_request
def require_login_guard():
    # Allow unauthenticated access to login/register/static/media
    allowed = {"login", "register", "static", "media"}
    if request.endpoint in allowed:
        return None

    # Root (index) renders main UI only if logged in; otherwise redirect
    if request.path == "/":
        if not is_authenticated():
            return redirect(url_for("login"))
        return None

    # For all other routes, enforce login
    if not is_authenticated():
        return redirect(url_for("login"))
```

Figure A.3 :Global Authentication Guard .

- Allows unauthenticated access to:
  - Login route
  - Registration route
  - Static files
  - Media files
- Redirects unauthenticated users attempting to access protected routes.
- Ensures the main dashboard is accessible only after successful login. This mechanism enforces session-based security across all protected endpoints.

### 4. Architectural Significance

The displayed code demonstrates:

- Modular backend architecture

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Secure session handling
- File-based user persistence
- Centralized authentication control
- Clean separation of configuration, database, recommendation, and emotion detection modules

Overall, these components form the secure backend foundation of the offline emotion-driven music recommendation system.

## B. Test Cases

Test cases include unit tests for preprocessing pipelines, integration tests for API endpoints, and manual validation tests for camera capture, playlist recommendation, and playback controls.

This section presents the testing results of the Mood-Based Music Recommendation System. The system was tested using real-time webcam input to detect facial expressions and recommend songs accordingly. Different emotional states such as Happy, Neutral, Sad, and Energetic were verified, and the corresponding music recommendations were displayed successfully.

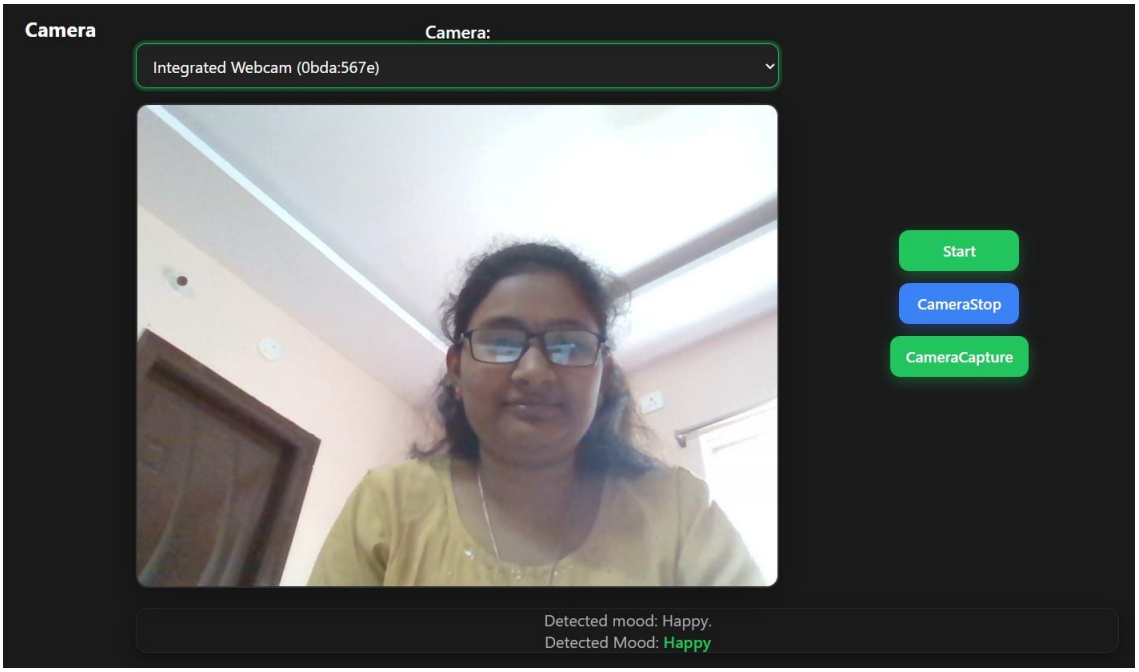



Figure B.1 : Happy Mood Detection with Song Recommendation.

During testing, the system successfully detected the user's facial expression as Happy. After capturing the image through the integrated webcam, the emotion detection model analyzed facial landmarks and classified the mood correctly. Based on the detected emotion, the system recommended a Hindi song and automatically enabled the music player with playback controls.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

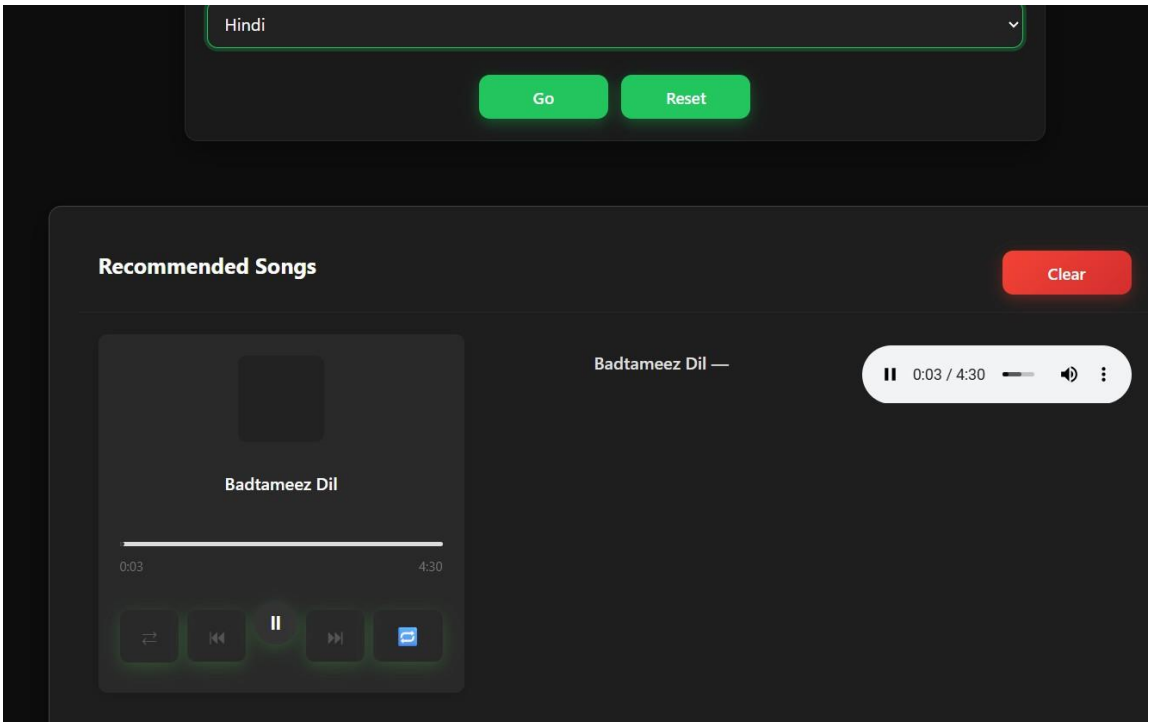



Figure B.2 : Neutral Mood Detection Result.

In this test case, the user's facial expression was identified as {Neutral. The real-time camera feed processed the face using the trained deep learning model and displayed the detected mood below the video frame. The result confirms that the system accurately differentiates neutral expressions from other emotional states.

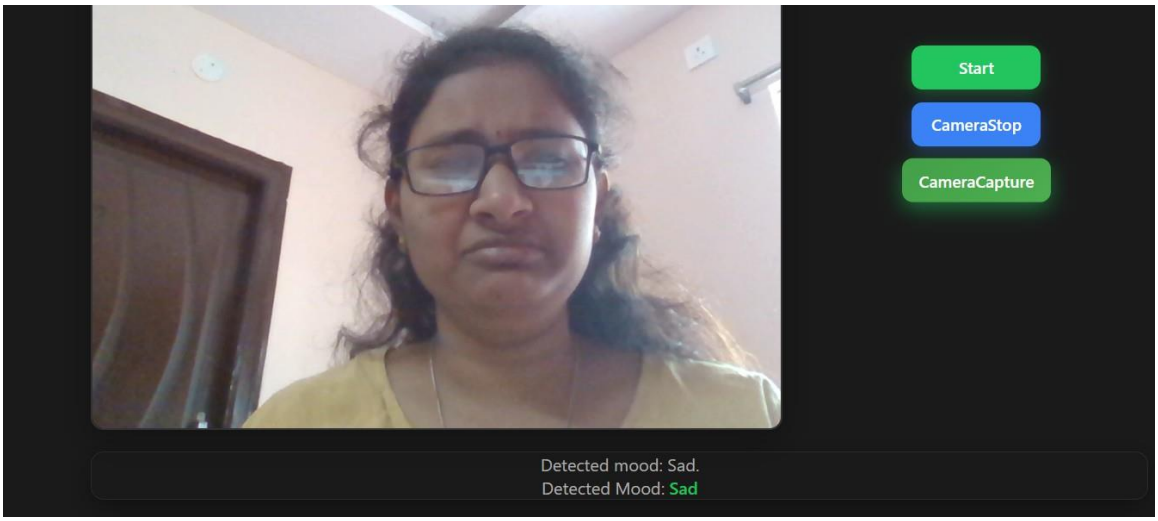



Figure B.3 : Sad Mood Detection Based on Facial Expression Analysis.

The system was further tested for negative emotional recognition. In this scenario, the model successfully classified the expression as {Sad. The output displayed the detected mood clearly, validating the accuracy of emotion classification even with subtle facial changes such as lowered

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

eyebrows and reduced smile intensity.

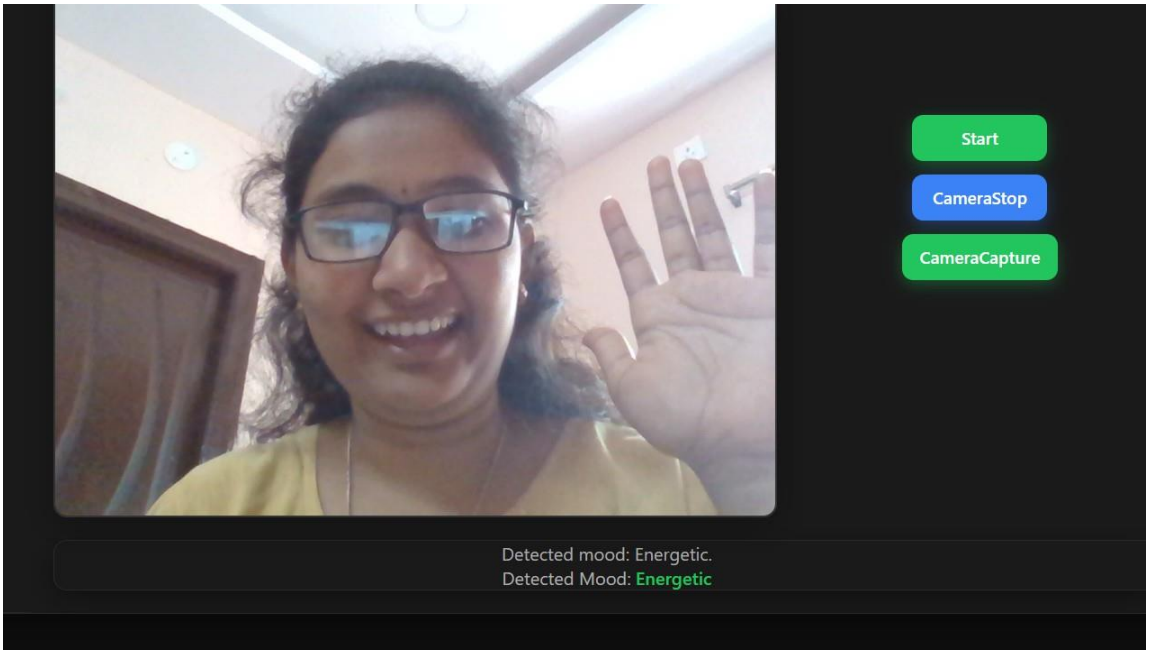



Figure B.4 : Energetic Mood Detection Using Facial and Gesture Recognition.

The system also detects energetic expressions, including visible excitement and hand gestures. In this test, the user's smiling face combined with an active hand gesture was classified as Energetic. This demonstrates that the model effectively interprets expressive features and dynamic cues to enhance mood recognition accuracy.

After mood detection, the recommended songs are displayed in the {Recommended Songs section. The built-in audio player provides playback controls such as play, pause, forward, backward, and progress tracking. This confirms successful integration between the emotion detection module and the music recommendation module.

## C. User Manual

The user manual provides installation, configuration, execution, and usage instructions for the Mood-Based Song Recommender System operating in offline mode on a Windows machine using PowerShell.

### C.1 Run Guide (Windows – PowerShell)

Follow the steps below in sequence. Execute all commands from the project root directory, for example, C:\Users\You\song-recommendation.

**1. Verify Python Installation**

Open PowerShell and run:

```
python --version
```

Ensure Python version 3.10 or above is installed. If Python is not installed, download it from: https://www.python.org/downloads/ During installation, select the option **"Add Python to PATH"**.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

### C.2 Allow PowerShell Script Execution

If virtual environment activation is blocked, run the following command (one-time setup):

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned
```

### C.3 Create Virtual Environment

Run the following commands:

```
python -m venv .venv
.\.venv\Scripts\Activate.ps1
```

### C.4 Install Project Dependencies

Install required packages using:

```
pip install -r requirements.txt
```

If any package installation fails, repeat the command.

For minimal execution without model training, install core libraries using:

```
pip install flask opencv-python numpy Pillow pandas scikit-learn
```

### C.5 First Run – Initialize Database and Users

Start the application using:

```
python offline_app.py
```

The server will run at:

http://127.0.0.1:5000/

### C.6 Login to the Application

Open:

http://127.0.0.1:5000/login Default credentials (auto-created):
**Username:** admin

**Password:** admin

Alternatively, create a new account using the registration option.

### C.7 Using the Application

- Click **Start** to enable the camera and allow browser permission.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

- Click **CameraCapture** to detect the current mood.
- Select preferred language from English, Hindi, or Telugu.
- Click **Go** to generate recommendations.
- Songs will be played using native audio controls.

For English, Hindi, and Telugu catalogs, song ordering follows:

songs\song_downloads.csv

### C.8 Optional Configuration Settings Run Online

**Version (Port 5001)** python online_app.py
Open:

http://127.0.0.1:5001/

**Set Custom Session Secret**

```
$env:OFFLINE_SECRET_KEY = "my-secret-123" python offline_app.py
```

**API Testing Commands**

Emotion Detection Endpoint Example:

```
$img = Get-ChildItem -Path archive/test -Include *.jpg,*.jpeg| Select-Object -First 1

$b64 = [Convert]::ToBase64String([IO.File]::ReadAllBytes($img.FullName))

$body = @{ image = "$b64" } | ConvertTo-Json
```

Music Recommendation Endpoint Example:

```
Invoke-RestMethod -Uri "http://127.0.0.1:5000/songs?mood=Happy
```

### C.9 Audio File Placement

Place MP3 files inside: songs/<Language>/<Mood>/ Update: songs/song_downloads.csv

to control playlist ordering.

### C.10 Stop the Server

Press:

```
Ctrl + C
```

in the PowerShell window running the server.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

### C.11 Troubleshooting Guide

- If camera does not display, ensure browser permission is granted.
- Verify audio files exist in the songs directory.
- If virtual environment activation fails, repeat PowerShell setup.
- If dependency installation fails, upgrade pip:

```
python -m pip install --upgrade pip
```

Then reinstall dependencies. For future usage:

```
.\.venv\Scripts\Activate.ps1 python
offline_app.py
```

## D. Sample Scan Reports

### D.1 Android Application Scan Report

This report section is optional and may contain security scanning or compatibility analysis results if mobile deployment is considered. Since the current Emotion-Based Music Recommendation System is primarily designed for offline desktop and browser-based usage, Android deployment scanning was not extensively performed. However, if future mobile integration is implemented, security and performance scanning tools may be used to evaluate application safety, malware detection, and compatibility across Android versions.

### D.2 Detailed Network Log Components

Since the system is primarily designed to operate in offline mode, network activity is minimal. The network logs are mainly associated with local server initialization, browser-based commu- nication, and internal API requests between the Flask backend server and the frontend interface. No external cloud communication or third-party data transmission is performed during normal execution, ensuring user privacy and reduced latency.

## E. Security Recommendations

- Maintain offline execution mode as the default operational setting to minimize exposure to external network threats and preserve user privacy during emotion detection and recommendation processing.
- Restrict unnecessary external API communication by disabling unused network endpoints and enforcing local-only computation policies for inference and playlist generation.
- Validate all filesystem paths prior to media streaming to prevent path traversal vulnera- bilities, especially in Windows directory environments where separator inconsistencies may occur.
- Protect machine learning models, CSV catalogs, and database files from unauthorized modification by implementing controlled file permissions and integrity verification checks.
- Implement secure configuration management by storing sensitive parameters such as secret keys and threshold values in environment-controlled configuration files.
- Prevent execution of untrusted scripts or external code injections within Flask routing logic and frontend JavaScript

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

components.

- Ensure camera and device permission requests are handled through secure browser APIs and that user consent is obtained before webcam activation.
- Maintain log files with restricted write access to prevent tampering with runtime diagnostic records.
- Regularly update development dependencies to address known security vulnerabilities in third-party libraries.
- Perform periodic integrity validation of audio media catalogs and database entries to detect corruption or unauthorized file changes.
- Disable unnecessary debugging interfaces and limit server exposure when deploying packaged executables.
- Follow secure coding practices including input validation, exception handling, and con- trolled resource management.

### F. Future Development Plans Future Enhancements

Future development can focus on improving detection robustness and expanding system intel- ligence.

- Integration of lightweight transformer-based or distilled CNN architectures to improve emotion recognition accuracy.
- Exploration of browser-based acceleration technologies such as WebGPU or WebNN for in-browser inference optimization.
- Enhancement of region-of-interest stabilization techniques to improve detection reliability under motion or low-light conditions.
- Expansion of music catalogs with richer metadata including genre hierarchy, tempo classification, and semantic audio descriptors.
- Implementation of telemetry-free local analytics to support user experience optimization while preserving privacy.
- Investigation of federated or on-device training approaches for privacy-preserving model updates.
- Development of multi-modal recommendation pipelines combining facial emotion, audio features, and optional textual context signals.
- Formalization of database migration and catalog integrity verification procedures for scalable library growth.
- Expand emotion recognition capability using advanced lightweight AI architectures.
- Introduce adaptive recommendation ranking based on long-term user interaction patterns.
- Implement multi-language catalog expansion and richer metadata tagging.
- Explore real-time adaptive learning pipelines.