

# Y.M.I.R: YIELDING MELODIES FOR INTERNAL RESTORATION

A PROJECT REPORT  
BY

TEAM NO. 04

TEAM MEMBER1 (E23CSEU0022)  
TEAM MEMBER2 (E23CSEU0014)



SUBMITTED TO

SCHOOL OF COMPUTER SCIENCE ENGINEERING AND  
TECHNOLOGY, BENNETT UNIVERSITY

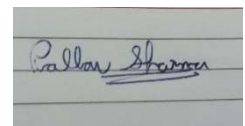
GREATER NOIDA, 201310, UTTAR PRADESH, INDIA

April 2025

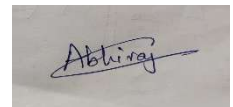
## DECLARATION

We hereby declare that the work which is being presented in the report entitled “Y.M.I.R.”, is an authentic record of our own work carried out during the period from January 2025 to April 2025 at School of Computer Science and Engineering and Technology, Bennett University Greater Noida.

The matters and the results presented in this report has not been submitted by me/us for the award of any other degree elsewhere.

A photograph of a handwritten signature in blue ink on lined paper. The signature appears to be 'Pallav Sharma'.

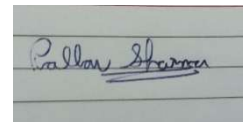
Pallav Sharma  
(Enroll. No. E23CSEU0022)

A photograph of a handwritten signature in blue ink on lined paper. The signature appears to be 'Abhiraj'.

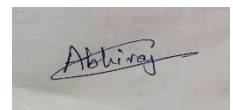
Abhiraj Ghose  
(Enroll. No. E23CSEU0014)

## ACKNOWLEDGEMENT

I/We would like to take this opportunity to express my/our deepest gratitude to my/our mentor, **Dr. Rishi Datt** for guiding, supporting, and helping me/us in every possible way. I/we was/were extremely fortunate to have him as my/our mentor as he provided insightful solutions to problems faced by me/us thus contributing immensely towards the completion of this capstone project. I/We would also like to express my/our deepest gratitude to VC, DEAN, HOD, faculty members and friends who helped me/us in successful completion of this capstone project.



Pallav Sharma  
(Enroll. No. E23CSEU0022)



Abhiraj Ghose  
(Enroll. No. E23CSEU0014)

## TABLE OF CONTENTS

LIST OF TABLES .....	vi
LIST OF FIGURES .....	vii
LIST OF ABBREVIATIONS .....	viii
ABSTRACT .....	ix
1. INTRODUCTION .....	1
1.1. Problem Statement .....	1
2. Background Research .....	2
2.1. Proposed System .....	4
2.2. Goals and Objectives .....	5
3. Project Planning .....	6
3.1. Project Lifecycle .....	6
3.2. Project Setup .....	6
3.3. Stakeholders .....	7
3.4. Project Resources .....	7
3.5. Assumptions .....	8
4. Project Tracking .....	9
4.1. Tracking .....	9
4.2. Communication Plan .....	9
4.3. Deliverables .....	10
5. SYSTEM ANALYSIS AND DESIGN .....	12
5.1. Overall Description .....	12
5.2. Users and Roles .....	12
5.3. Design diagrams/ UML diagrams/ Flow Charts/ E-R diagrams .....	13

5.3.1. Use Case Diagrams.....	14
5.3.2. Class Diagram .....	15
5.3.3. Activity Diagrams .....	16
5.3.4. Sequence Diagram.....	17
5.3.5. Data Architecture.....	18
6. User Interface.....	19
6.1. UI Description .....	19
6.2. UI Mockup .....	19
7. Algorithms/Pseudo Code .....	21
8. Project Closure.....	24
8.1. Goals / Vision.....	24
8.2. Delivered Solution.....	24
8.3. Remaining Work .....	25
REFERENCES .....	26

## LIST OF TABLES

<u>Table</u>	<u>Page</u>
Table 1: Goal and Objectives.....	5
Table 2: Sample 2 .....	6
Table 3: Sample 3 .....	7
Table 4: Sample 4 .....	7
Table 5: Sample 4 .....	8
Table 6: Sample 6 .....	9
Table 7: Regularly Scheduled Meetings .....	9
Table 8: Information To Be Shared Within Our Group.....	10
Table 9: Information To Be Provided To Other Groups.....	10
Table 10: Information Needed From Other Groups .....	10
Table 11: Deliverables .....	10
Table 12: Sample 12 .....	12

## LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
Figure 1: Sample use-case diagram .....	15
Figure 2: sample 2.....	15
Figure 3: sample 3.....	16
Figure 4: Sample 4.....	17
Figure 5: Sample 5.....	18
Figure 6: Sample 6.....	20

## **LIST OF ABBREVIATIONS**

AI	Artificial Intelligence
YMIR	Yielding Melodies for Internal Restoration
NLP	Natural Language Processing
FER	Facial Emotion Recognition
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network



## ABSTRACT

The modern digital era has exposed the current generation to growing challenges regarding their emotional well-being through stressful environments, increasing social anxiety due to excessive technology usage. The field of AI has achieved numerous accomplishments yet there aren't any complete developed systems capable of emotional user interaction with emotion detection abilities. Using both emotional cues present in facial expressions and text chat, **Y.M.I.R** creates an advanced **AI** system which then suggests suitable music across various contexts for mood improvement.

The system combines **FER** using DeepFace (Python's language inbuilt library) with **NLP** implementing **TextBlob** and **BERT** models for effective text-based emotion detection. Real-time emotion detection takes place through the webcam and chatbot interface of a Flask-based web application which operates two parallel tracking techniques. The system computes final dominant emotions using a combination of data from facial expressions and text analysis in real-time by either selecting results from confident sources or computing weighted averages.

Based on the determined final emotional state, a **content-based music recommendation engine** is used which is trained on a dataset scraped directly from **Spotify** with over 1000 records, that includes mood tags, mental health benefits, and audio features and incorporated ensemble learning to make predictions for recommending music. This system presents users with songs either matching their current mood selection or it provides optimistic audio content to positively influence their emotional states. The music application displays a mini-player that resembles the format used by **Spotify** along with other commercial platforms for smooth listening control.

This project proves how **Artificial Intelligence** can connect emotional intelligence with tailored personal well-being solutions between human beings and machines. The system employs real-time intelligence together with offline functionality to deliver music therapy as a mental health promotion through machine learning and user-centred design.

# 1. INTRODUCTION

Music has become recognized across various fields as an effective tool to modulate feelings while therapists incorporate it into their work for enhancing mood patterns. The primary method used by effective computing systems is facial recognition or written text analysis which results in suboptimal emotion recognition accuracy. The implementation of cloud-based models presents three main drawbacks because it affects privacy levels and generates performance slowness while minimizing system accessibility. An immediate offline system needs development which takes several data sources to measure emotions and automatically create appropriate responses including mood-based song recommendations. The **multi-modal AI** system of **Y.M.I.R** utilizes facial and text-based emotion detection to obtain comprehensive emotional states of users. Users can access mood-based music recommendations through Flask which operates through their detected emotion and provides platform support offline.

## **Key Motivations:**

- Growing demand for **AI** systems that go beyond utility to address emotional well-being.
- The gap in current technologies in detecting and responding to complex human emotions.
- An interest in the field of mental health and music.

## **1.1 Problem Statement**

Most **AI** systems detect emotions through single detection methods such as facial expressions and text while being unable to deliver emotionally sensitive responses. Popular music recommendation platforms such as Spotify, Amazon Music and many more suggest songs based on user history or genre preferences but fail to account for real-time emotional states. These limitations result in emotionally disconnected user experiences. System developers should create a single unified platform which detects emotions within real-time from face and writing data to generate music fitting the user's emotional state. This system accomplishes mental wellness goals.

## 2. BACKGROUND RESEARCH

The integration of **AI** into music recommendation systems has opened new avenues for enhancing user experience by aligning musical suggestions with users' emotional states. Traditional recommendation systems primarily rely on user preferences and listening history, often neglecting the real-time emotional context of the user. Recent research endeavors have focused on developing systems that can detect user emotions and recommend music accordingly, aiming to provide a more personalized and emotionally resonant listening experience.

The study by Mahadik et al. (2021) presents a mood-based music player that performs real-time mood detection using facial expressions and suggests songs that correspond to the detected mood. The system employs image analysis to predict user expressions and recommends music that aligns with the user's current emotional state, thereby enhancing user satisfaction and engagement.

Source: [https://www.researchgate.net/publication/352780489\\_Mood\\_based\\_music\\_recommendation\\_system](https://www.researchgate.net/publication/352780489_Mood_based_music_recommendation_system)

Similarly, the research published in the International Journal of Engineering Research & Technology (IJERT) discusses a system that analyzes the user's image to predict expressions and suggest suitable songs. This approach emphasizes the significance of incorporating mood detection into music players to improve user experience.

Source: <https://www.ijert.org/mood-based-music-recommendation-system>

Another study, available through IRJET, introduces a mood-sensitive music recommendation system that utilizes real-time facial emotion analysis to generate personalized music playlists. The system uses pre-trained convolutional neural networks (CNNs) for feature extraction and employs categorical cross-entropy loss functions to address dataset imbalances. By analyzing facial emotions, the system aims to enhance user experience by automating music recommendations based on emotional states.

Source: <https://www.irjet.net/archives/V10/i8/IRJET-V10I871.pdf>

These systems showcase a move towards more emotionally intelligent computing environments, emphasizing that emotional state detection can significantly enhance personalized recommendations. Each of these studies recognizes the importance of integrating emotion analysis into user interaction, particularly for entertainment and mental health applications.

While facial expression analysis is a powerful tool for detecting mood, it may not always accurately reflect the user's internal state. Contextual emotion analysis through chat or typed inputs can capture more nuanced emotional expressions. Some recent systems attempt to incorporate text analysis, but they often lack the depth or real-time capability needed for dynamic applications.

Text-based emotion detection involves natural language processing techniques that assess sentiment and emotional tone from written language. When combined with visual cues, it provides a more robust and accurate emotional profile. This multi-modal approach is a key innovation direction for emotion-aware systems and lays the foundation for projects like **Y.M.I.R.**

**Limitations of Existing Systems:** While these systems represent significant advancements in mood-based music recommendation, they exhibit certain limitations:

- Most systems rely solely on facial expression analysis, potentially overlooking the nuanced emotional cues present in textual communication.
- Many existing models are cloud-based, raising concerns about data privacy, latency, and accessibility in offline scenarios.
- The emotional categories recognized by these systems are often restricted to basic emotions, which may not capture the complexity of human affective states.
- There is a lack of personalization in the recommendation logic, as most systems match songs based on generic mood labels instead of personalized emotional patterns.

### **Motivation for Y.M.I.R.**

The identified limitations in current systems underscore the need for a more comprehensive approach to emotion-based music recommendation. Y.M.I.R. aims to address these gaps by:

- Combining facial expression analysis with text-based sentiment analysis to capture a broader spectrum of emotional cues.

- Developing an offline-capable system to ensure user privacy and accessibility without reliance on internet connectivity.

- Incorporating advanced AI models to recognize a wider range of emotions, facilitating more accurate and personalized music recommendations.

- Using music not only to reflect current moods but to actively uplift users' emotional states, contributing to emotional resilience and mental wellness.

By integrating these features, Y.M.I.R. aspires to create an empathetic AI companion that not only understands the user's emotional state but also actively contributes to emotional well-being through tailored music suggestions.

## 2.1 Proposed System

This project aims to enhance emotional well-being by developing an intelligent, multi-modal system that can detect a user's emotional state in real time through facial expressions and textual interaction, and respond by recommending emotionally appropriate music. By leveraging artificial intelligence, machine learning, and affective computing, the system intends to simulate human-like emotional understanding and deliver meaningful responses that go beyond typical rule-based interactions.

Using advanced facial emotion recognition (FER) algorithms, natural language processing (NLP), and a locally hosted AI chatbot, the system collects emotional cues from both webcam input and user conversations. These emotion readings are then aggregated, analysed, and used to compute a final dominant emotional state. Based on this emotion, the system triggers a content-based filtering model to recommend songs that either reflect the user's mood or uplift them if the emotion is negative.

The vision behind this system is to create a responsive AI companion that not only interacts with users but understands and supports their mental state, particularly during emotionally challenging moments. In a world where mental health tools are either inaccessible or impersonal, Y.M.I.R. provides a digital alternative that is quick to respond and help, empathetic, real-time, and privacy-conscious by design.

This system will improve users' daily lives by offering a calming or energizing musical intervention tailored to their emotional needs. As part of ongoing efforts to blend AI with emotional intelligence, Y.M.I.R. contributes a highly personalized and offline-capable solution for

emotional restoration through music—an approach that is currently rare in AI-driven recommendation engines.

## 2.2 Goals and Objectives

The primary goal of the Y.M.I.R. project is to develop a real-time, offline-capable, multi-modal AI system that accurately detects human emotions through both facial and textual inputs and responds with personalized music recommendations aimed at emotional restoration.

**Table 1: Goal and Objectives**

#	Goals or Objectives
1	Integrate facial emotion recognition using DeepFace into a real-time web interface.
2	Develop a text-based emotion detection module using NLP models such as BERT, TextBlob, and GPT4All.
3	Implement a hybrid emotion aggregation algorithm to compute a final dominant emotion from both inputs.
4	Build a content-based music recommendation engine using features such as mood labels and mental health benefits.
5	Design and deploy a responsive, user-friendly web UI with video feed, chatbot interface, and embedded music player.
6	Ensure complete offline functionality, using open-source models for privacy and accessibility.
7	Log emotion data and provide periodic summaries for insights and mood tracking.
8	Conduct internal user testing to evaluate emotional accuracy and musical relevance with at least 80% satisfaction rate.
9	Provide meaningful documentation for system architecture, user guidance, and technical configuration.
10	Maintain teamwork through agile development (SCRUM) with weekly sprint reviews and version-controlled development (Git).
11	Have fun working on the project

### 3. PROJECT PLANNING

This section outlines the structured approach taken to plan and implement the Y.M.I.R system. The planning involved selecting a suitable development lifecycle, identifying key stakeholders, determining necessary resources, and documenting any assumptions that guided the project's progress.

#### 3.1 Project Lifecycle

Given the iterative nature of the system's design — including continuous refinement of the emotion detection models and integration of new recommendation logic — the **Agile Software Development Lifecycle (SDLC)** was chosen. Agile allows for flexibility, adaptability to feedback, and incremental delivery of key components. Development was split into the following major sprints:

- **Sprint 1:** Research, dataset preparation, and emotion detection setup using DeepFace.
- **Sprint 2:** Integration of text-based emotion classification through chatbot conversation.
- **Sprint 3:** Development of music recommendation engine using content-based filtering.
- **Sprint 4:** Flask-based full stack integration with a modern frontend UI.
- **Sprint 5:** Emotion averaging logic, background processing, and UI enhancements (e.g., animated elements, floating player).
- **Sprint 6:** Final testing, optimization, and documentation.

Each sprint involved planning, development, testing, and review, which allowed continuous user-focused improvement of the system.

#### 3.1. Project Setup

The following decisions were made during the initial setup of the project. These decisions guided the project's direction, tooling, standards, and deployment strategy.

**Table 2: Sample 2**

#	Decision Description
1	<b>Operating System:</b> Windows 11 Home with Python 3.10; <b>Development Stack:</b> Flask, HTML/CSS, JavaScript; <b>AI Frameworks:</b> DeepFace for FER, scikit-learn for text emotion classification; <b>Version Control:</b> Git and GitHub.
2	Followed <b>PEP8 Python coding standards</b> , organized modular architecture, and adhered to best practices in HTML/CSS design. Emotion detection modules and chatbot logic were separated for maintainability.
3	No external cloud APIs were used to preserve user privacy; project runs entirely offline. No special access or licenses required. The final project can be released as open source.
4	The system was developed and tested on a local machine simulating real-world usage conditions (webcam input, offline processing, browser interaction). Deployment on Flask ensured environment portability.

### 3.2. Stakeholders

The stakeholders involved in this project include:

**Table 3: Sample 3**

Stakeholder	Role
Person A	Indirect User [Family Members]
Person B	Dr. Rishi Datt [ Mentor]
Person C	Dr. Rishi Datt [ Instructor]
Person D	Pallav Sharma [Project developer]
Person E	Abhiraj Ghose [Project developer]

### 3.3. Project Resources

To ensure successful development and working, the following resources were utilized:

#### Example:

**Table 4: Sample 4**

Resource	Resource Description	Quantity
Hardware:	Laptop with Intel i7 CPU, 16GB RAM, and NVIDIA RTX 4060 GPU for local development and model inference.	2
Capstone Team	Our team of students who will be the primary developers of the project.	2



Dr Rishi Datt [Mentor]	The mentor who will be able to provide us with technical assistance.	1
Python Libraries	OpenCV, DeepFace, scikit-learn, Flask, transformers, etc., used for development.	-
External Monitor	Used for extended workspace during UI/UX and video-based model testing.	1
Dataset	Pre-cleaned CSV dataset with Mood Labels, Mental Health Benefits, and Musical Features for training the recommendation engine.	1
Browser	Google Chrome for testing and rendering the real-time web-based system UI.	1

### 3.4. Assumptions

The following assumptions have been made during the course of the project to plan development, resourcing, and delivery timelines effectively:

**Table 5: Sample 4**

#	Assumption
A1	The project team and mentor will be able to meet regularly (virtually or physically) once a week.
A2	All required Python libraries and frameworks (e.g., DeepFace, Flask, Transformers) will remain open-source and stable during the development period.
A3	The team will have uninterrupted access to the development laptop with GPU support throughout the project.
A4	Team will have sufficient time to complete a working model to present by mid-semester
A5	Internet access will be available during testing phases that require fetching online content (e.g., YouTube/Spotify embeds).
A6	The development test data provided will be sufficient to create an accurate prediction of user actions
A7	Users interacting with the system will do so under reasonably controlled lighting and positioning for FER accuracy.
A8	All planned features (FER, chatbot, music recommendation, UI components) can be integrated into a single Flask web application.
A9	The models developed will be easily extended to other forms within the time frame

## 4. PROJECT TRACKING

### 4.1. Tracking

The progress of the Y.M.I.R. project was systematically tracked using collaborative tools for version control, documentation, testing, and communication. Git and GitHub were used for storing source code and managing version history, while manual regression testing and real-time code reviews were used to ensure the stability of each module. Regular updates were maintained to reflect progress, issues, and implementation status.

**Table 6: Sample 6**

Information	Description	Link
Code Storage	Project code was stored and managed using Git, with remote access via GitHub.	<a href="#">Link</a>
Bug Tracking	Issues and bugs were tracked using Notepad.	<a href="#">N/A</a>
Project Documents and Assignments	Weekly updates, diagrams, and progress reports were shared on WhatsApp.	<a href="#">N/A</a>
Continuous Integration	Code was manually tested on every feature merge; automated CI was planned but not implemented.	<a href="#">N/A</a>
Regression Testing	Manual regression testing was performed at the end of each sprint using test case checklists.	<a href="#">Local testing environment</a>

### 4.2. Communication Plan

Effective and regular communication was maintained between team members and the mentor. Weekly meetings, sprint discussions, and on-demand check-ins ensured the team was aligned and issues were promptly addressed.

**Table 7: Regularly Scheduled Meetings**

Meeting Type	Frequency/Schedule	Who Attends
Conference Call	Weekly	Project team and mentor

Team Meeting	Weekly	Project team
Short Meeting	Weekly in class	Project team
Sprint Planning Meeting	Start of each sprint	Project team and mentor
Sprint Review Meeting	End of each sprint	Project team, <i>mentor, and sponsor</i>

**Table 8: Information to Be Shared Within Our Group**

Who?	What Information?	When?	How?
Project team	Task assignments & General scrum information	Weekly	Team meetings, Google Docs, WhatsApp

**Table 9: Information To Be Provided To Other Groups**

Who?	What Information?	When?	How?
Sponsor and mentor	Final deliverables	At completion of project	Project specification doc., code, Power Point presentation
Sponsor and mentor	Weekly report	Weekly	Email and Trac site access
Mentor	Sprint summaries	At the end of each sprint	Google Docs and verbal update during review.

**Table 10: Information Needed From Other Groups**

Who?	What Information?	When?	How?
Mentor	Feedback and requirement changes	Start of each sprint	Conference call or meeting with sponsor and mentor.
Supervisor	Presentation guidelines	Before final submission	Email

### 4.3. Deliverables

Below is the list of deliverables planned for successful project execution and evaluation:

**Table 11: Deliverables**

#	Deliverable
1	Study results (emotion model performance, music impact findings)

2	Complete codebase (FER, chatbot, music recommender, Flask integration)
3	Test and test results
4	Build instructions and configuration files (environment setup)
5	Installation guide for running the Flask web app offline
6	User manual for interaction and testing
7	Postmortem document detailing challenges, resolutions, and team reflections
8	Final report (final PowerPoint presentation, 3 minute video, and final sprint)

## 5. SYSTEM ANALYSIS AND DESIGN

This section describes in detail about the design part of the system.

### 5.1. Overall Description

The Y.M.I.R. project aims to enhance human-computer interaction by building a system capable of understanding human emotions through both facial expressions and textual input, and then recommending music that suits or improves the user's emotional state. The system was developed using Python, with key libraries including DeepFace for facial emotion recognition, various NLP models for text-based sentiment analysis, and a content-based music recommender. These components are integrated into a Flask-based web application with an intuitive frontend.

The user interacts with the system in real-time via a camera feed and chatbot interface. The facial recognition module processes visual data frame by frame to identify the current emotion, while the chatbot analyses user messages to extract textual sentiment and the user's feelings. Both sources contribute to a dominant emotion score that dynamically evolves throughout the session. Once finalized, the emotion is used to query a custom dataset of songs using cosine similarity and mood-based metadata to return top musical recommendations aimed at emotional upliftment or reinforcement.

The entire process runs locally, ensuring full offline capability and data privacy. The UI also includes modern visual elements such as animated transitions, a floating mini music player, and interactive feedback. The technical design ensures modularity for future extensions—such as voice-based input, user account personalization, or mood journaling.

### 5.2. Users and Roles

**Table 12: Sample 12**

User	Description
Developer	Responsible for building the facial recognition, chatbot, recommender engine, and UI components.
Mentor	Oversees technical guidance, validates implementation, and provides

	feedback during reviews.
System (AI)	Autonomous agent that receives emotion input, processes it, and returns music suggestions
General User	Interacts with the camera and chatbot interface; receives music recommendations.

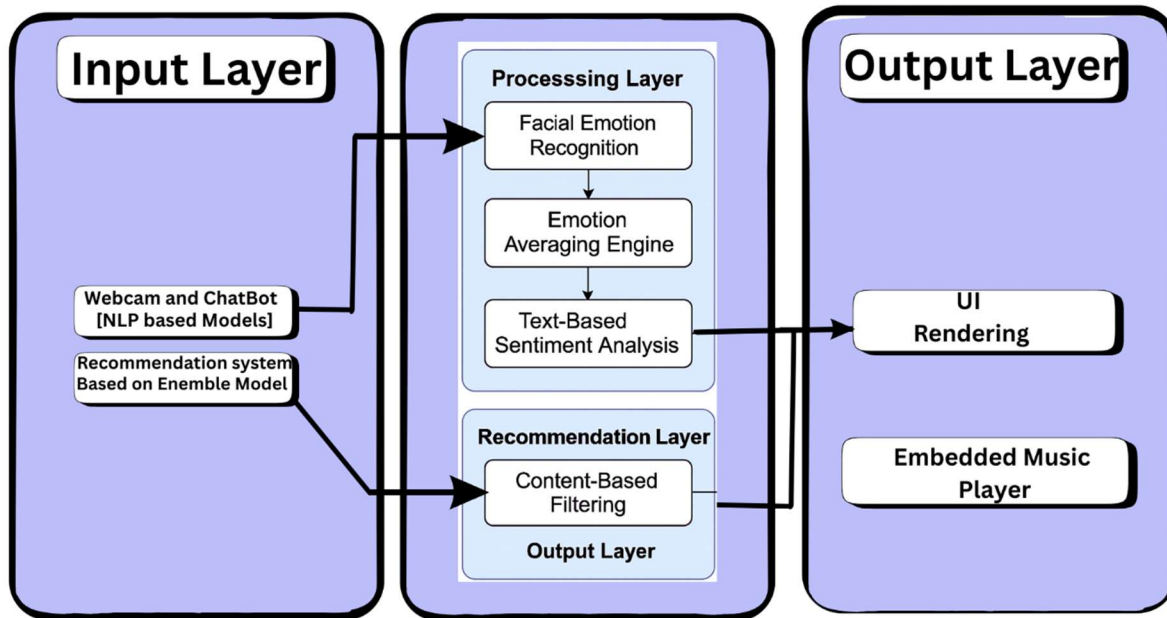
## 5.3 Design diagrams/Architecture/ UML diagrams/ Flow Charts/ E-R

### diagrams

#### 5.3.1 Product Backlog Items

- "As Max, I want to invite my friends, so we can enjoy this service together."
- "As Sasha, I want to organize my work, so I can feel more in control."
- "As a manager, I want to be able to understand my colleagues' progress, so I can better report our success and failures."
- "As a fitness enthusiast, I want to track my workouts and progress in an app so that I can monitor my improvements."
- "As a frequent shopper, I want to save my payment information so that I can check out quickly on future purchases."
- "As an online student, I want to receive notifications for upcoming classes so that I can easily manage my schedule."

### 5.3.2 Architecture Diagram



### 5.2.1. Use Case Diagram

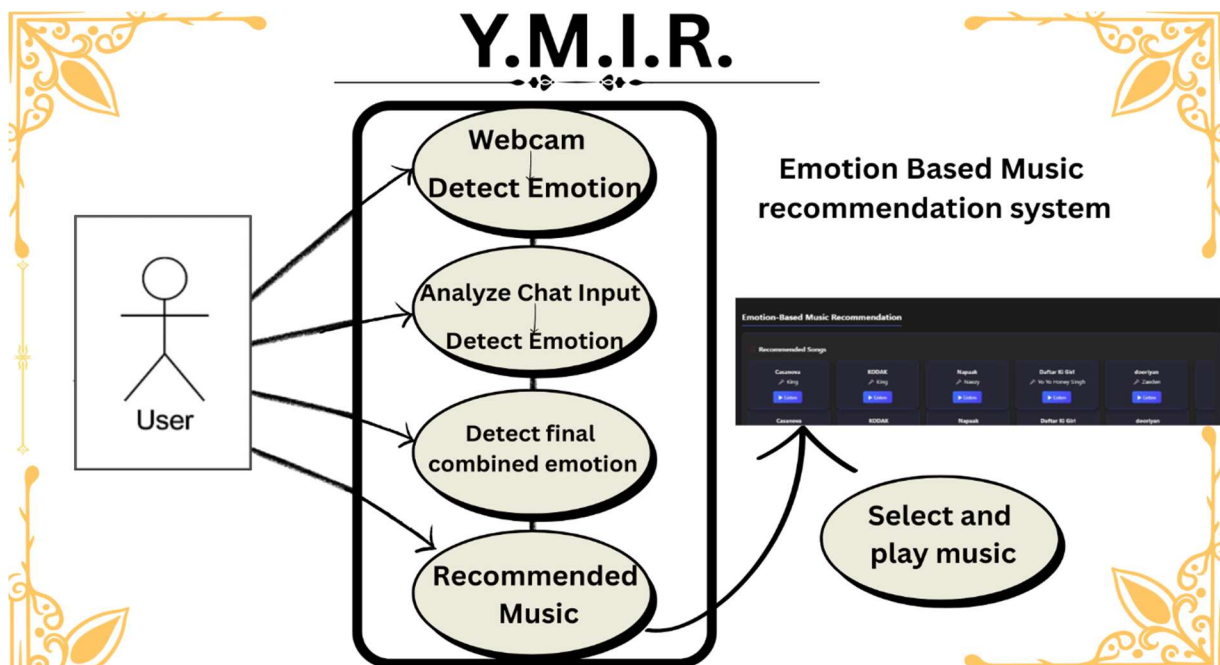


Figure 1: Sample use-case diagram

### 5.2.2. Class Diagram

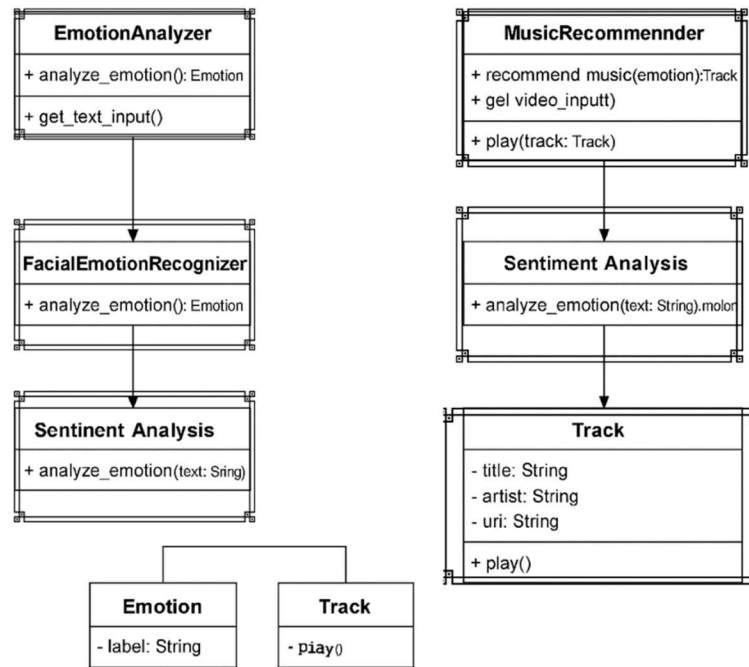


Figure 2: sample 2



### 5.2.3. Activity Diagrams

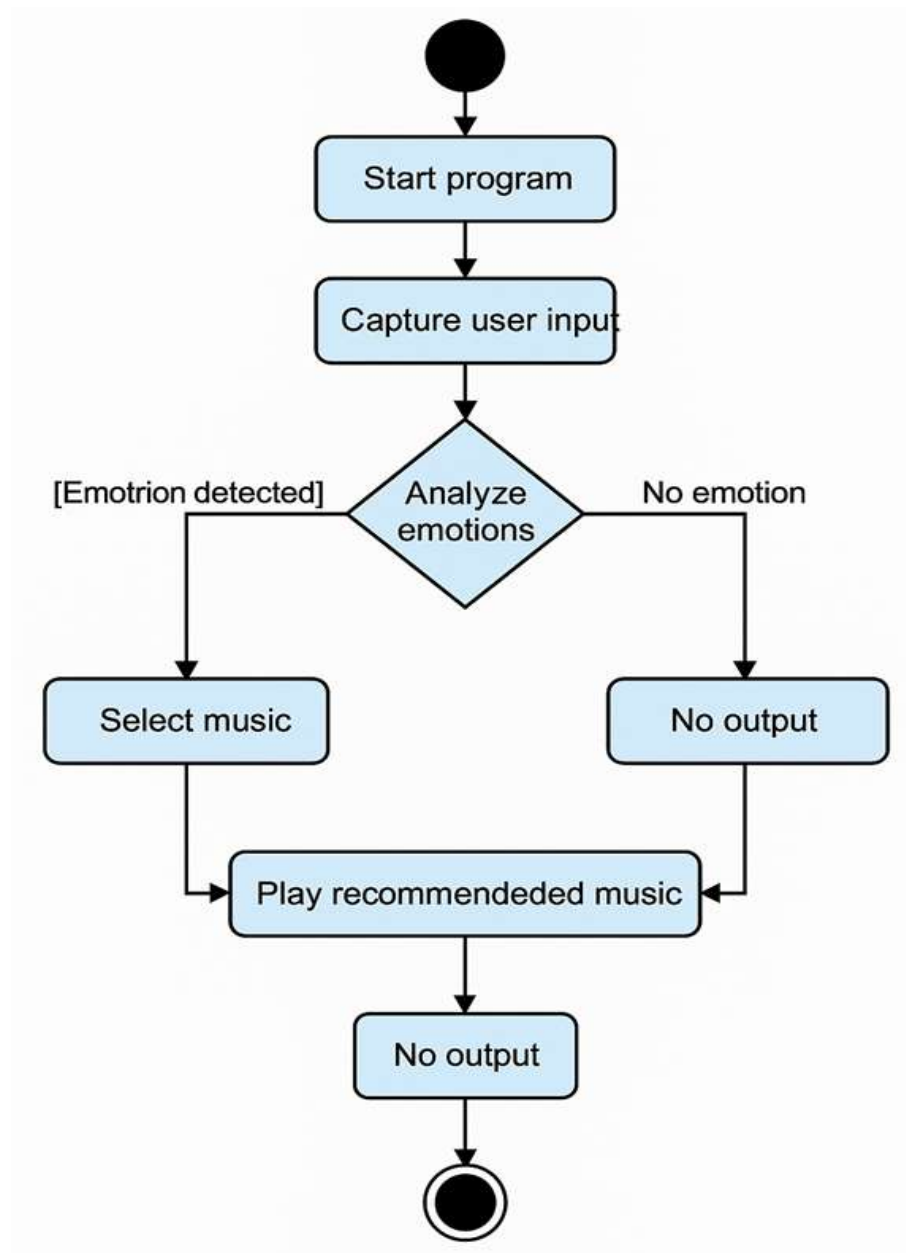


Figure 3: sample 3

#### 5.2.4. Sequence Diagram

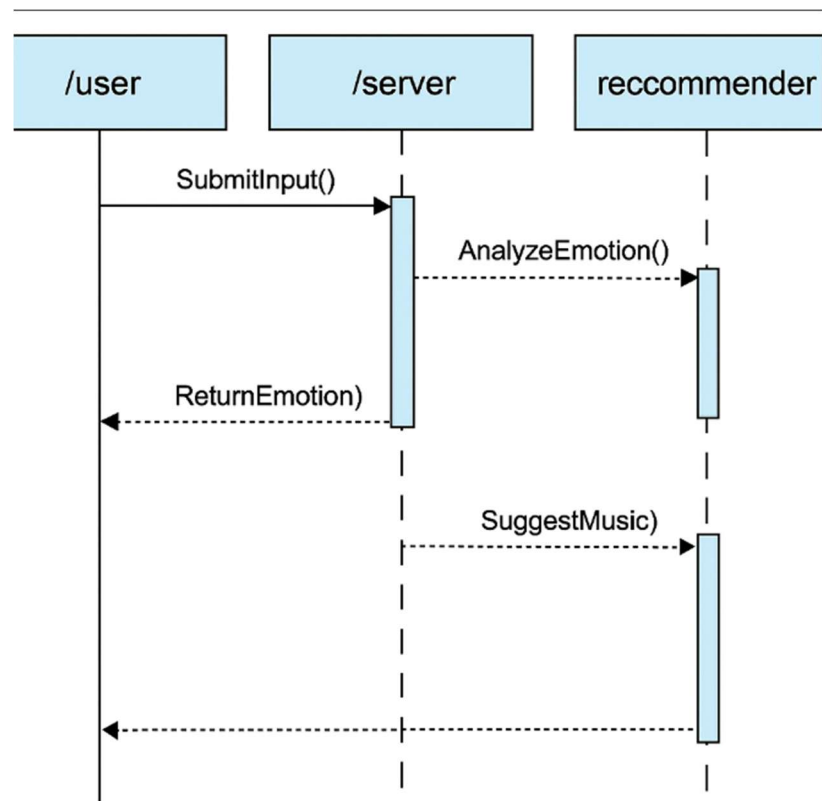


Figure 4: Sample 4

### 5.2.5. Data Architecture

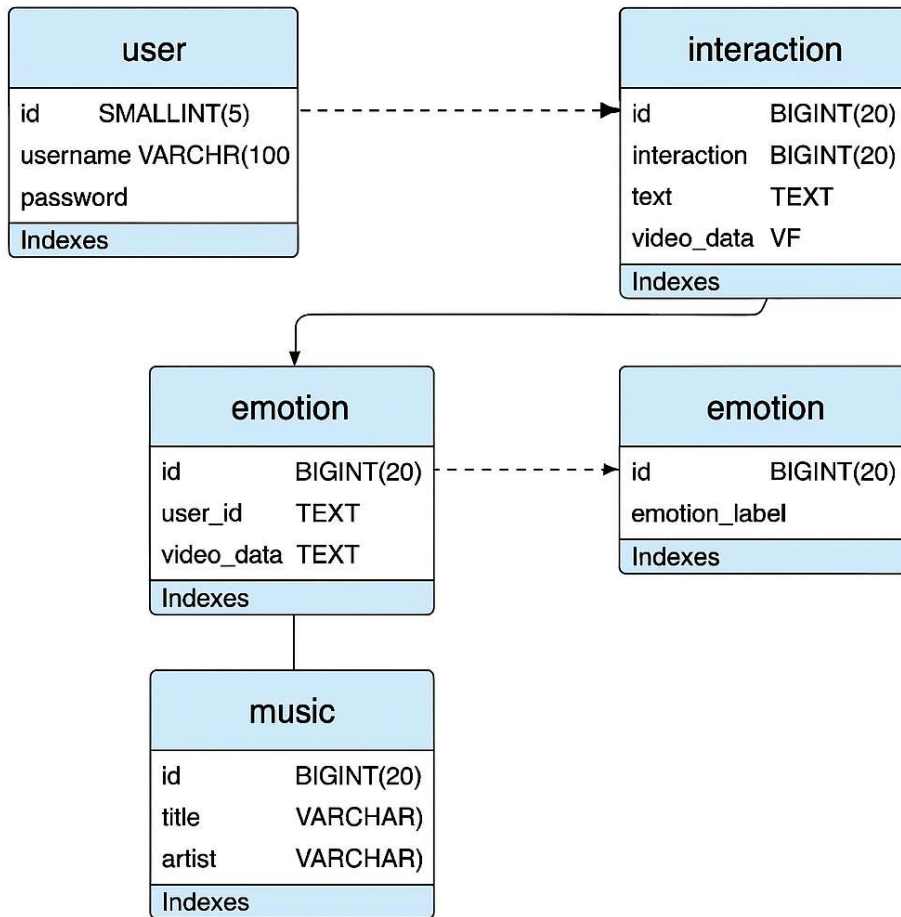


Figure 5: Sample 5

## 6. USER INTERFACE

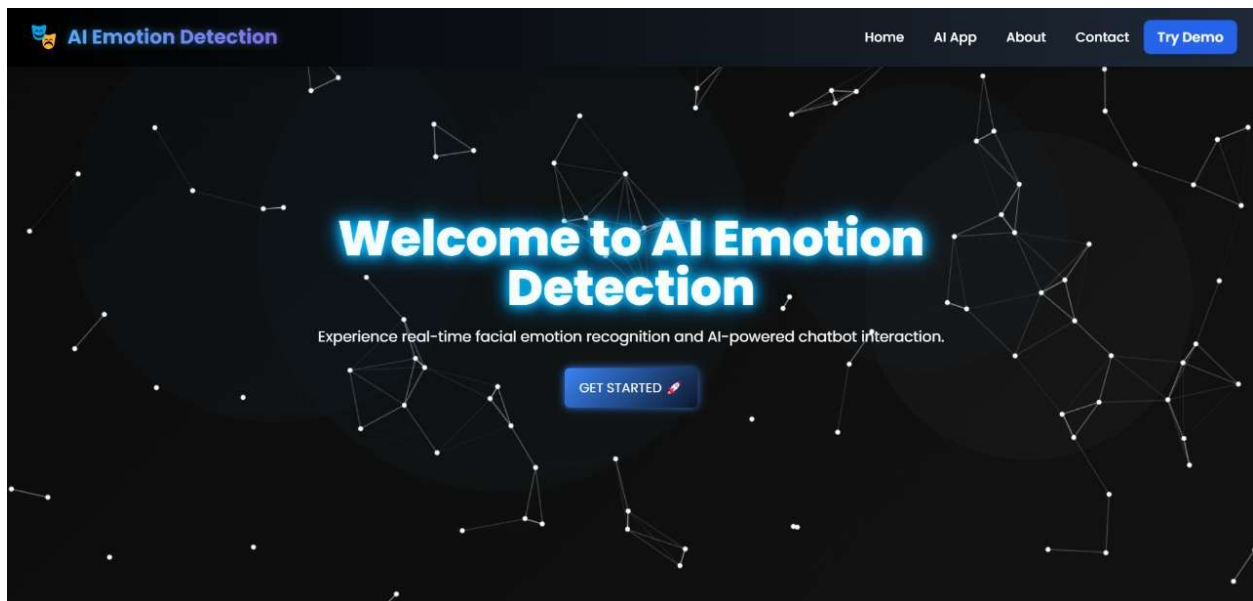
### 6.1. UI Description

The user interface (UI) for this project is a modern, web-based frontend designed to provide an intuitive and interactive experience. The homepage features a welcoming hero section with a glowing animated title that introduces the project — "AI Emotion Detection" — along with a call-to-action button to get started. The design incorporates a dark theme with a space-tech aesthetic, including animated node connections in the background for a futuristic feel.

Upon entering the application, users interact with two main modules: a real-time facial emotion recognition system on the left and a chatbot interface on the right. Below these, the UI displays an emotion-based music recommendation section that dynamically updates song suggestions based on the user's detected emotions. Each song card includes a title, artist name, and a "Listen" button, providing a seamless and user-friendly music discovery experience.

The UI ensures that users can engage with both visual and text-based emotion analysis, while the backend processes the data and updates the recommendations in real time. The interface is fully browser-based, eliminating the need for additional software installation and making it accessible and easy to use.

### 6.2. UI Mockup



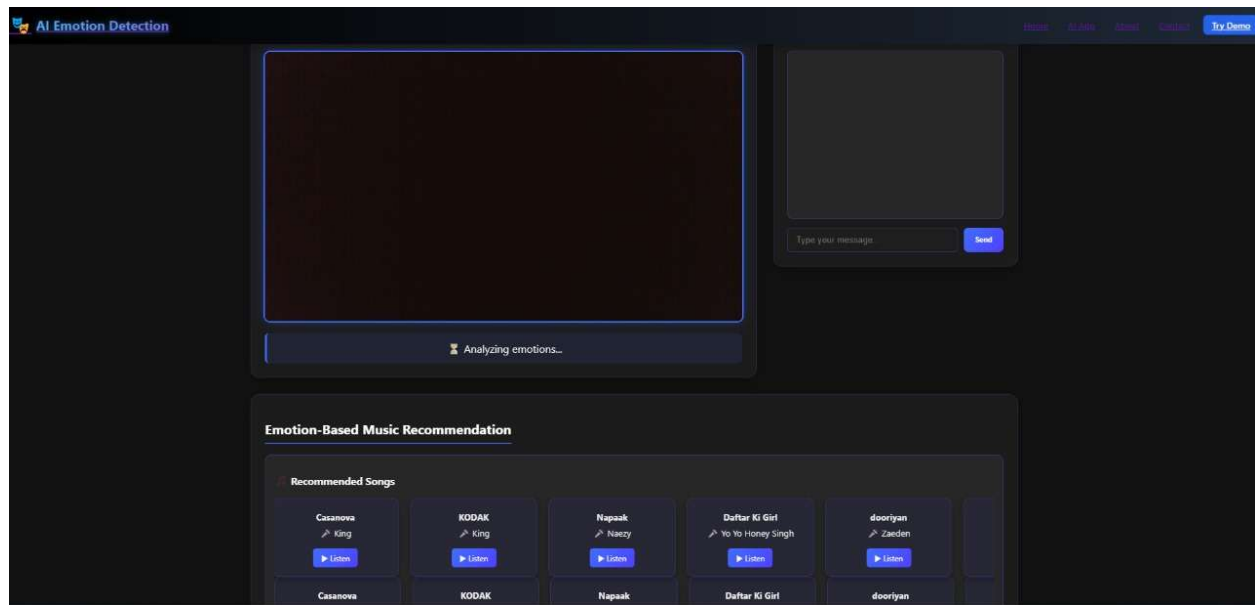


Figure 6: Sample 6

## 7. ALGORITHMS/PSEUDO CODE OF CORE FUNCTIONALITY

ALGORITHM: MultimodalEmotionRecognition

INPUT: Video feed from camera

OUTPUT: Real-time emotion analysis, facial landmarks, body posture data

FUNCTION InitializeComponents():

- Initialize camera and video capture
- Load face detection model (MediaPipe)
- Load facial landmark model (MediaPipe Face Mesh)
- Load pose detection model (MediaPipe Pose)
- Load hand tracking model (MediaPipe Hands)
- Load emotion recognition model (DeepFace)
- Initialize threading for parallel processing
- Set up display window and parameters

FUNCTION AnalyzeEmotion(face\_region):

- // Runs in separate thread for performance
- Resize face image to standard dimensions
- emotion\_result = DeepFace.analyze(face\_region, actions=['emotion'])
- Extract emotion confidence scores
- Log emotions with timestamp
- Update shared emotion data structure
- Write emotion data to JSON log

FUNCTION EyeAspectRatio(eye\_points):

- // Calculate ratio for gaze detection
- Calculate vertical distances between eye points
- Calculate horizontal distance
- RETURN ratio of distances

FUNCTION DrawFacialLandmarks(frame, landmarks):

- FOR EACH landmark IN landmarks:
- Convert relative coordinates to pixel positions
- Draw landmark point on frame

FUNCTION ProcessFrame(frame):

- Convert frame to RGB and grayscale
- Detect faces using cascade and MediaPipe
- IF faces detected:
- Update last\_known\_faces
- Reset no\_face\_counter
- ELSE:
- Increment no\_face\_counter
- IF no\_face\_counter < threshold:
- Use last\_known\_faces

- // Process multiple modalities
- Extract face mesh landmarks
- Extract body pose landmarks
- Extract hand landmarks

- // Process primary face for emotion analysis
- IF processing\_interval\_reached:
- Submit face\_region to emotion analysis thread

- // Draw visualizations
- Draw face bounding boxes
- Draw facial landmarks
- Display emotion labels and scores
- Draw gaze direction indicators
- Draw body pose overlay
- Draw hand tracking points

RETURN annotated\_frame

FUNCTION MainLoop():

- WHILE True:
- Capture frame from camera
- IF frame capture failed:
- Continue

- processed\_frame = ProcessFrame(frame)
- Calculate and display FPS
- Display processed frame

- IF exit\_condition:
- BREAK

FUNCTION ComputeAverageEmotions():

- Load emotion log data
- Initialize emotion counters
- FOR EACH entry IN log\_data:
- Accumulate emotion confidence scores

- Calculate average emotion scores
- RETURN formatted\_emotion\_summary

# Pseudo-code for Facial Emotion Recognition

ALGORITHM: EmotionalChatbotSystem  
INPUT: User text input  
OUTPUT: Chatbot response with emotion analysis

```
FUNCTION InitializeComponents():
    Initialize API connections and credentials
    Load emotion detection models (3 distinct models)
    Load sentiment analysis model
    Initialize data structures for chat history

FUNCTION DetectEmotions(text):
    Initialize emotion_scores as empty dictionary

    FOR EACH model IN emotion_detection_models:
        results = model.predict(text)
        FOR top_predictions IN results:
            Map raw emotion label to standardized category
            Add weighted score to emotion_scores

    IF negation_patterns_detected(text):
        Apply emotion inversion rules

    sentiment = SentimentAnalysisModel.predict(text)
    IF sentiment == "negative" AND "sad" IN emotion_scores:
        Increase sad score

    RETURN dominant_emotion, emotion_scores

FUNCTION GenerateChatbotResponse(user_input):
    Format API request with user input
    Send request to LLM API (Groq)
    Parse and validate response
    RETURN formatted_response

FUNCTION MainLoop():
    WHILE True:
        user_input = GetUserInput()
        IF user_input == "exit":
            SaveChatResults()
            BREAK

        chatbot_response = GenerateChatbotResponse(user_input)
        UpdateChatHistory(user_input, chatbot_response)

        dominant_emotion, scores = DetectEmotions(GetRecentMessages())
        LogEmotionData(dominant_emotion, scores)
        DisplayResponse(chatbot_response)

FUNCTION SaveChatResults():
    Format chat history and emotion analysis
    Write to JSON file for future analysis
```

Pseudo-Code

For

ChatBot

ALGORITHM: EmotionBasedMusicRecommendation  
 INPUT: Emotion data files from chat and facial analysis  
 OUTPUT: Music recommendations based on emotional state

```

FUNCTION LoadModels():
  Load ensemble classification model
  Load label encoder for mood categories
  Load feature scaler
  Load music dataset with mood labels

FUNCTION ProcessEmotions(emotion_file):
  // Convert detected emotions to audio feature space
  Load emotion data from JSON
  Extract numerical emotion scores

  // Emotional state to audio feature mapping
  Initialize weighted_audio_features as zero vector
  FOR EACH emotion, weight IN emotion_scores:
    IF emotion IN emotion_to_audio_mapping:
      feature_contribution = emotion_to_audio_mapping[emotion] * weight
      Add feature_contribution to weighted_audio_features

  Normalize audio features using scaler
  RETURN normalized_features, emotion_scores

FUNCTION RecommendSongs(emotion_file):
  emotion_vector, emotion_scores = ProcessEmotions(emotion_file)

  // Dimensionality reduction for better prediction
  Apply standardization
  Apply PCA transformation

  // Predict mood from emotion vector
  mood_probabilities = ensemble_model.predict_proba(emotion_vector)
  predicted_mood = ensemble_model.predict(emotion_vector)

  // Apply emotional context rules
  dominant_emotions = GetTopEmotions(emotion_scores, 2)
  mapped_moods = MapEmotionsToMoods(dominant_emotions)

  IF predicted_mood NOT IN mapped_moods:
    Adjust predicted_mood to first mapped_mood

  // Filter dataset by predicted mood
  filtered_songs = dataset[dataset.Mood_Label == predicted_mood]
  IF filtered_songs.empty:
    filtered_songs = dataset[dataset.Mood_Label IN mapped_moods]
  IF filtered_songs.empty: // Final fallback
    filtered_songs = dataset[dataset.Mood_Label == "Neutral"]

  recommended_songs = SampleDistinctSongs(filtered_songs, 10)
  RETURN FormatRecommendations(recommended_songs)

FUNCTION CalculateFinalEmotions():
  // Combine chat and facial emotion data
  Load chat emotion data
  Load facial emotion data

  Extract dominant emotions from chat analysis
  Extract average emotions from facial analysis

  // Normalize and combine data sources
  Convert percentages to decimal values
  Create dataframes from both emotion sources
  Calculate weighted average of both sources

  Save combined emotion profile to JSON
  RETURN final_emotion_profile

```

**Pseudo-code**

**For**

**Recommending**



## 8. PROJECT CLOSURE

This section elucidates the overall lookup at the project and some of the future works that may enhance the solution.

### 8.1. Goals / Vision

Our original goal for this project was to build a real-time emotion-aware system that could detect human emotions through facial expressions and textual conversations, and then provide music recommendations accordingly to improve the user's mood and mental well-being. Initially, we envisioned integrating emotion detection with a simple music suggestion engine. However, as the project evolved, we expanded our vision to include a dual-mode emotion analysis system using both facial recognition and chatbot-based interaction. The final solution not only detects emotions visually and textually but also provides a continuous stream of personalized music recommendations within a modern, interactive web interface. This aligns with our broader vision of using AI to enhance emotional wellness through technology.

### 8.2. Delivered Solution

The delivered solution is a comprehensive web-based application. It consists of three core modules:

1. **Facial Emotion Recognition** using computer vision and deep learning techniques to capture real-time expressions via webcam.
2. **AI-Powered Chatbot** that interacts with the user in natural language, refining emotion classification through textual analysis.
3. **Emotion-Based Music Recommendation System**, which continuously analyzes detected emotions and dynamically recommends mood-enhancing songs.

The system is deployed in a responsive and user-friendly UI. It features a homepage with an animated entry point, a dual-pane interface for FER and chatbot interaction, and a dynamically updating music section where users can click "Listen" to engage with suggested tracks. All background processes — emotion logging, averaging, and music selection — are seamlessly integrated into a single app.py file, ensuring a smooth and real-time user experience.

### 8.3. Remaining Work

While the project meets its intended functionality, several enhancements could further improve its effectiveness and scalability:

- **Multilingual Chatbot Support** to interact with users in different languages and broaden accessibility.
- **Integration with Spotify/Youtube APIs** to allow live streaming and real-time audio previews instead of placeholder buttons.
- **Improved Emotion Aggregation Models** to combine facial and textual emotion more accurately using weighted or ensemble models.
- **Mobile-Friendly Version** for increased accessibility across devices.
- **Emotion History Dashboard** for users to view their emotional trends over time and receive tailored wellness tips.
- **User Personalization Features** such as song feedback options or playlist creation based on mood profiles.

## REFERENCES

1. Mahadik, A., Milgir, S., Patel, J., Jagan, V. B., & Kavathekar, V. (2021). Mood based music recommendation system. International Journal of Engineering Research & Technology (IJERT), 10(6). Available at:  
[https://www.researchgate.net/publication/352780489\\_Mood\\_based\\_music\\_recommendation\\_system](https://www.researchgate.net/publication/352780489_Mood_based_music_recommendation_system)
2. Mood based Music Recommendation System. IJERT. Available at: <https://www.ijert.org/mood-based-music-recommendation-system>
3. Mood Sensitive Music Recommendation System. IRJET. Available at: <https://www.irjet.net/archives/V10/i8/IRJET-V10I871.pdf>
4. A. Mutz, R. Wolski, Efficient auction-based grid reservation using dynamic programming, in: IEEE/ACM Int'l Conf. on Super Computing, SC 2007.
5. M. Mezmaiz, N. Melab, Y. Kessaci, Y. C. Lee, E. G. Talbi, A. Y. Zomaya and D. Tuytens, "Parallel Bi-Objective Hybrid Metaheuristic for Energy-Aware Scheduling for Cloud Computing Systems", Journal of Parallel Distributed Computing, Elsevier, Vol. 71, pp. 1497-1508, 2011.
6. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype and Reality for Delivering Computing as the 5th Utility", Future Generation Computer Systems, Elsevier, Vol. 25, pp. 599-616, 2009.
7. Basements and crawl spaces. Retrieved from <http://www.hud.gov/offices/hsg/sfh/ref/sfhp1-25.cfm> (Access in June 2020).