

Project Report
on

MoodMusic

to be developed to fulfil the requirements for

Integrated Project (CA329)

Submitted to

Department of Computer Applications Chitkara University, Punjab



under the supervision of

Dr. Amanpreet Kaur
Assistant Professor
(CURIN)

Dr. Bhanu Sharma
Assistant Professor
(CURIN)

Dr. Anchal Thakur
Assistant Professor
(CSE)

Submitted by

Shyam
2410987129

Muktali
2410987084

Master of Computer Applications
(Batch 2024-26)

CERTIFICATE

This is to certify that the report on software project titled “MoodMusic” submitted by the student team comprising Shyam Dua (2410987129) to the Department of Computer Applications, Chitkara University, Punjab in partial fulfilment for the completion of the course Integrated Project (CA329) in the second semester of Masters of Computer Applications is a Bona fide record of work carried out by the team under my supervision.

Dr. Amanpreet Kaur
Assistant Professor
(CURIN)

Dr. Bhanu Sharma
Assistant Professor
(CURIN)

Dr. Anchal Thakur
Assistant Professor
(CSE)

DATE:

TABLE OF CONTENTS

S.No.	Topic	Page No.
1.	Abstract	1
2.	Profile of Problems Assigned	1-2
3.	Study of Existing Systems	2-3
4.	System Requirements 4.1 Product Definition a) Problem Statement b) Function to be provided c) Processing Environment: H/W, S/W. d) Solution Strategy e) Acceptance Criteria 4.2. Feasibility Analysis 4.3 . Project Plan a) Team Structure b) Development Schedule c) Programming Languages and Development Tools	3-6
5.	System Requirement Specifications 5.1. Developing / Operating / Maintenance Environments 5.2. External Interface and Data Flows a) User display and report format, user command summary. b) High level DFD and data dictionary. 5.3. Functional and performance specifications.	6-8
6.	Design 6.1. Detailed DFD's and structure diagrams. 6.2. Data structures, database and file specifications. 6.3. Pseudo code	8-13
7.	Test Plan 7.1. Functional, Performance, Stress tests etc.	13-16
8.	Implementation / Conversion Plan	16-17
9.	Project Legacy 9.1. Current Status of project. 9.2. Remaining areas of concern. 9.3. Technical and Managerial lessons learnt. 9.4. Future Recommendations.	17-18
10.	Bibliography	18-19
11.	Source Code	19-21

1. Abstract:

MoodMusic is an innovative AI-based system designed to enhance the music listening experience by offering personalized music recommendations based on the user's emotional state. The system utilizes facial expression recognition to detect emotions such as happiness, sadness, anger, and neutrality [12]. This is achieved through the FER (Facial Expression Recognition) library, which analyzes the user's facial expressions captured via a webcam or uploaded images[1][12]. Upon detecting an emotion, the system uses the YouTube Data API to fetch and recommend music videos that match the user's mood.

The backend of the system is built using Flask, a lightweight Python web framework, which handles API requests for emotion detection and music recommendations[4]. MongoDB is used as the database to store user information, including their emotion history and previous music recommendations, allowing the system to learn and adapt over time[5]. On the frontend, the system is developed using React.js, providing an interactive user interface where users can upload images, view real-time emotion detection, and receive their personalized music suggestions[6][7].

This project bridges the gap between emotion recognition and music recommendation, creating a dynamic and personalized user experience. The MoodMusic system not only recommends music based on user emotions but also adapts to individual preferences over time. Future enhancements could include additional music platforms, more advanced emotion detection models, and deeper integration with personal user preferences [10].

2. Profile of Problems Assigned:

The MoodMusic project aimed to create an emotion-based music recommendation system. The key problems assigned and the solutions implemented are as follows:

2.1. Emotion Detection from Facial Expressions:

2.1.1. Problem: The main challenge was accurately detecting emotions from user facial expressions, considering variables such as lighting, facial variations, and camera quality.

2.1.2. Solution: The project used the FER (Facial Expression Recognition) library, which utilizes trained models for emotion recognition[9][12], allowing the system to detect emotions such as happiness, sadness, and anger with high accuracy under varied conditions.

2.2. Real-Time Emotion Detection and Music Recommendation:

2.2.1. Problem: The system required real-time detection of emotions and immediate music recommendations to ensure a smooth user experience.

2.2.2. Solution: Real-time processing was achieved by optimizing emotion detection algorithms and using efficient image processing techniques. Music recommendations

were fetched quickly using the YouTube API, with dynamic queries based on the detected emotion.[1]

2.3. Personalized Music Recommendations Based on Emotions:

2.3.1. Problem: The system needed to offer music recommendations not based on user preferences but according to their current emotional state.

2.3.2. Solution: Emotions were mapped to specific music genres (e.g., upbeat for happiness, calm for sadness)[10], and the system used the YouTube API to fetch relevant videos matching these emotions, offering a personalized recommendation experience.

2.4. Frontend and User Interaction:

2.4.1. Problem: The system needed to provide an interactive and user-friendly interface for non-technical users to upload images, view detected emotions, and receive recommendations.

2.4.2. Solution: The frontend was developed using React.js, which provided a dynamic interface where users could easily interact with the system, upload images, track emotional history, and receive real-time music recommendations[6].

3. Study of Existing System:

3.1. Music Recommendation Systems:

Music recommendation systems have existed for years, primarily focusing on user preferences, genres, or listening history. Some of the most popular systems include Spotify, Apple Music, and YouTube. These systems use algorithms such as collaborative filtering, content-based filtering, and hybrid approaches [8][10]

3.1.1. Collaborative Filtering:

This technique recommends music based on the preferences of users with similar tastes. For example, if two users listen to similar genres, the system will recommend the same songs to both [8]

3.1.2. Content-Based Filtering:

This method recommends songs based on attributes such as genre, artist, tempo, or mood (which can be manually tagged). However, it is limited to the attributes tagged in the song and does not consider the emotional state of the user[10].

3.1.3. Hybrid Systems:

Hybrid systems combine collaborative and content-based filtering for better accuracy. Pandora and Spotify utilize this approach to recommend music based on user history and song features (e.g., tempo, energy).[10]

3.2. Limitations of Existing Music Recommendation Systems:

3.2.1. Lack of Emotion-Based Recommendations:

While most music platforms offer personalized recommendations based on listening

habits, they do not take the user's emotional state into account [12]. Users may often find that suggested songs do not align with how they are feeling at a given moment.

4. System Requirements:

4.1. Product Definition:

4.1.1. Problem Statement:

The MoodMusic project aims to provide a personalized music recommendation system based on the user's emotional state. Unlike traditional systems that rely on user preferences or listening history, MoodMusic integrates real-time emotion detection via facial recognition and suggests music from YouTube tailored to the user's mood[12].

4.1.2. Functions to be Provided:

- **Emotion Detection:**
Real-time detection of the user's emotions (happiness, sadness, anger, neutrality) using facial expression recognition from live video or uploaded images [1][12]
- **Music Recommendation:**
Based on the detected emotion, the system will fetch relevant music recommendations from the YouTube API, mapping emotions to specific music genres or moods [10]
- **User Interaction:**
Allow users to interact with the system by uploading images or using their webcam to detect emotions. Display music recommendations based on the detected emotion in an interactive user interface [6].
- **Emotion History Tracking:**
Keep a record of the user's emotional states and associated music recommendations for later viewing[5].
- **Social Media Sharing:**
Enable users to share their current emotional state and music recommendation via social media platforms (e.g., Twitter, Facebook).
- **User Account Management:**
Provide functionality for user registration, login, and authentication using JWT (JSON Web Tokens) for secure access to personalized features [4].

4.2. Processing Environment: H/W, S/W:

4.2.1. Hardware Requirements:

- **Camera:** A webcam or camera capable of capturing facial expressions for real-time emotion detection.
- **Computer/Server:** A personal computer or server with adequate processing power to handle real-time emotion detection and API requests [2][3].
- **Internet Access:** Required for fetching music recommendations from the

YouTube API and cloud-based deployments.

4.2.2. Software Requirements:

- **Backend Development:** Python 3.x with Flask framework for developing RESTful APIs [4].
- **Emotion Detection:** FER (Facial Expression Recognition) library for analyzing facial expressions and detecting emotions [12].
- **Database:** MongoDB for storing user data, emotion history, and music recommendations [5].
- **Frontend Development:** React.js for building a responsive and interactive user interface [6].
- **API Integration:** YouTube Data API for fetching music recommendations
- **Authentication:** Flask-JWT-Extended for handling user authentication using JSON Web Tokens [4].

4.3. Solution Strategy:

4.3.1. Frontend:

The user interacts with the system via a web-based interface developed using React.js. [6] The frontend is responsible for capturing webcam feeds or allowing users to upload images for emotion detection. It also displays the recommended music and tracks user history.

4.3.2. Backend:

The Flask backend processes requests, including user registration/login, emotion detection (using the FER library), and music recommendations (using the YouTube API). The backend also stores user data and emotion history in MongoDB for personalization. [5]

4.3.3. Emotion Detection:

The FER library is used to analyze images or video feeds and detect emotions from the user's facial expressions [12]. The detected emotion is then mapped to a corresponding music genre or mood, and relevant music is fetched using the YouTube Data API [10].

4.3.4. User Interaction:

Users interact with the system by uploading images or using the webcam to detect emotions. Based on the detected emotion, the system fetches relevant music from YouTube and displays the results on the frontend.

4.4. Acceptance Criteria:

4.4.1. Accuracy of Emotion Detection:

The system must accurately detect emotions like happiness, sadness, anger, and neutrality from webcam or uploaded images [12].

4.4.2. Real-Time Processing:

The system must process emotions in real-time, with minimal delay between the user's facial expression and the music recommendation [2] [9].

4.4.3. Personalization of Music Recommendations:

The system must offer relevant and personalized music recommendations based on the detected emotion [10] .

4.4.4. User Engagement:

The user interface must be interactive, responsive, and easy to use, ensuring that users can easily upload images or access webcam feeds and view recommendations [6][7].

4.4.5. Scalability:

The system should be able to handle a reasonable number of users concurrently, without significant degradation in performance [5].

4.4.6. Security:

The user registration, login, and history must be securely managed using JWT for authentication [4].

4.5. Feasibility Analysis:

4.5.1. Technical Feasibility:

- **Emotion Detection:** The FER library, built on deep learning models, is well-suited for real-time emotion detection [12], providing accurate and robust results across various lighting conditions and face orientations.
- **Backend Framework:** **Flask** is lightweight and efficient for building RESTful APIs [4], making it a suitable choice for the project. MongoDB complements this by offering a flexible schema for storing user data and emotion history.
- **Frontend Development:** **React.js** is a powerful framework for building responsive user interfaces, ensuring a smooth and interactive user experience.

4.5.2. Operational Feasibility:

The system can be deployed on cloud platforms such as AWS or Heroku, allowing for scalable and reliable operation. The project does not require extensive hardware beyond a basic webcam and computer/server infrastructure.

4.5.3. Economic Feasibility:

- **YouTube API** usage is free for basic usage (with rate limits), making it a cost-effective solution for fetching music recommendations.
- **MongoDB** provides free-tier access with sufficient storage for the project's initial scope, minimizing costs [5].
- **Flask** and **React.js** are both open-source frameworks, ensuring minimal costs for the development process [4][6].

4.6. Project Plan:

4.6.1. Team Structure:

- **Backend Developer:** [Shyam Dua]
Responsible for developing the backend using Flask, integrating FER for emotion detection, and connecting to MongoDB and YouTube API [4] [12].

- **Frontend Developer:** [Shyam Dua]
Responsible for developing the frontend interface using React ensuring a smooth and interactive user experience [6].

4.6.2. Development Schedule:

- **Week 1-2:**
Requirement gathering and system design.
Set up project repository, tools, and frameworks.
- **Week 3-4:**
Develop backend logic for emotion detection and API integration (Flask, FER, YouTube API) [2][3][12]
Set up MongoDB for user data storage[5].
- **Week 5-6:**
Develop frontend using React.js, integrating webcam functionality and displaying recommendations[6][7].
- **Week 7:**
Integrate frontend with backend.
Implement user authentication and data handling.

4.6.3. Programming Languages and Development Tools

- **Programming Languages:**
Python (for backend development with Flask) [4]
Javascript (for frontend development with React)[6]
- **Frameworks & Libraries:**
Flask (for building the backend and APIs)
React.js (for building the frontend)
FER (for facial expression recognition)
MongoDB (for storing user data and emotion history) [5]
YouTube Data API (for fetching music recommendations)
- **Development Tools:**
Visual Studio Code (IDE)
Git (for version control)

5. System Requirement Specifications:

5.1. Developing / Operating / Maintenance Environments:

5.1.1. Developing Environment:

- **Backend Development:**
Language: Python 3.x
Framework: Flask [4] (for RESTful API development)
Libraries: FER (Facial Expression Recognition), OpenCV (for video feed processing), Flask-JWT-Extended (for authentication)

Database: MongoDB (for user data storage, emotion history, and music preferences)

IDE: Visual Studio Code or PyCharm

- **Frontend Development:**

Language: JavaScript

Framework: React.js [6] (for building the user interface)

CSS Framework: Tailwind CSS [7] (for responsive UI design)

Libraries: Axios (for API calls), React Router (for navigation), React-Toastify (for notifications)

5.1.2. Operating Environment:

- **Servers:** Can be deployed on cloud platforms like AWS or Heroku for scalability and performance. A local server setup using Flask is also possible for development and testing.
- **Operating System:** Linux/Ubuntu or Windows/Mac OS for development; the application will run on any OS supporting the above software.

5.1.3. Maintenance Environment:

- **Cloud Deployment:** AWS EC2 or Heroku for easy deployment, automatic scaling, and continuous integration.
- **Database Management:** MongoDB Atlas for cloud-based database management and automatic backups [5]
- **Security:** Regular updates and security patches for Flask, React, MongoDB, and external APIs.

5.2. External Interface and Data Flows:

5.2.1. User Display and Report Format, User Command Summary

- **User Display:**

The frontend displays a live video feed (via webcam) where the user can upload an image for emotion detection. Detected emotions are shown on the screen along with corresponding music recommendations (thumbnails and video titles) [6]. A history panel allows users to see previously detected emotions and the music that was recommended. Login and registration screens are available for users to access personalized data.

- **User Command Summary:**

Emotion Detection: Users can click a “Start Analysis” button to begin detecting emotions via the webcam or upload an image for offline detection.

Music Recommendations: Music videos are shown in a list with thumbnails, titles, and play buttons. Users can click on video titles to open them on YouTube.

Account Management: Login/Logout buttons for user authentication. History View for tracking past emotions and recommended music.

5.2.2. Data Dictionary

Table 1 defines the data dictionary of MoodMagic.

Table 1: Data Dictionary

Entity	Description
User	Contains information about the user including username, email, password.
Emotion	Represents a detected emotion with attributes like emotionType, confidence.
Music	Represents a music video with attributes like videoId, title, url.

5.2.3. Functional and Performance Specifications

- **Functional Specifications:**

Emotion Detection: The system should analyze images or webcam feeds to detect emotions such as happy, sad, angry, or neutral with at least 85% accuracy [12] .

Music Recommendation: Based on the detected emotion, the system should query the YouTube Data API to recommend music videos. The system should fetch the top 25 relevant videos for the detected emotion [10] .

User Authentication: The system must support user registration, login, and JWT-based authentication [4] .

Emotion History: The system must store the user's detected emotions along with associated music recommendations in the database.

- **Performance Specifications:**

Response Time: The system should provide emotion detection and music recommendations in under 5 seconds from the moment the user uploads an image or activates the webcam.

Scalability: The system must support at least 100 simultaneous users without performance degradation.

Reliability: The system should have 99.9% uptime in the production environment, with automatic failover for critical services.

Security: User data must be securely stored, and passwords must be hashed. JWT authentication should be implemented for user sessions.

6. Design:

6.1. Detailed DFD's and Structure Diagrams:

6.1.1. High-Level DFD:

Figure 1 represents the high level DFD for MoodMusic.

- **User Login and Registration:** Manages user authentication and sign-up.

- **Stored Credentials:** Stores user credentials securely.
- **Great:** Confirms successful login or registration.
- **Database:** Stores user data and emotion history.
- **Backend Processing:** Handles logic for emotion detection and music recommendations.
- **Frontend Display:** Displays the user interface and recommendations.
- **Playlist Mapping:** Matches detected emotions to music playlists.
- **Emotion Detection:** Detects emotions from images or video.
- **Webcam:** Captures video feed for emotion detection.

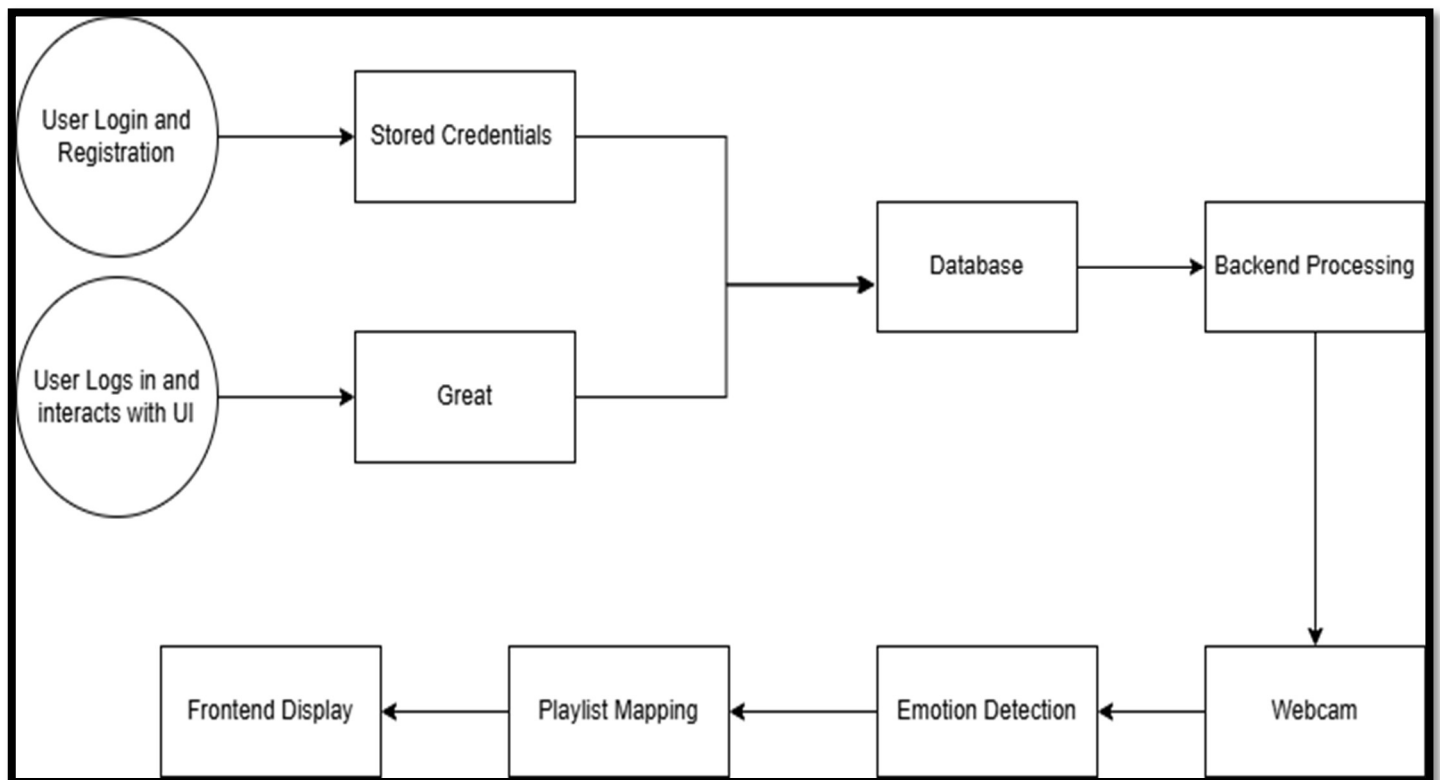


Figure `1 DFD of MoodMusic

6.1.2. Structure Diagram:

Figure 2 represents the structure diagram for MoodMagic.

- **User Interacts with UI (Frontend):** Sends image/webcam data to the Emotion Detection Process.
- **Emotion Detection:** Detects emotion, sends it to the Music Recommendation Process.
- **Music Recommendation:** Queries the YouTube API for music based on the detected emotion and sends results to the Frontend.
- **Frontend Display:** Shows the recommended music to the user

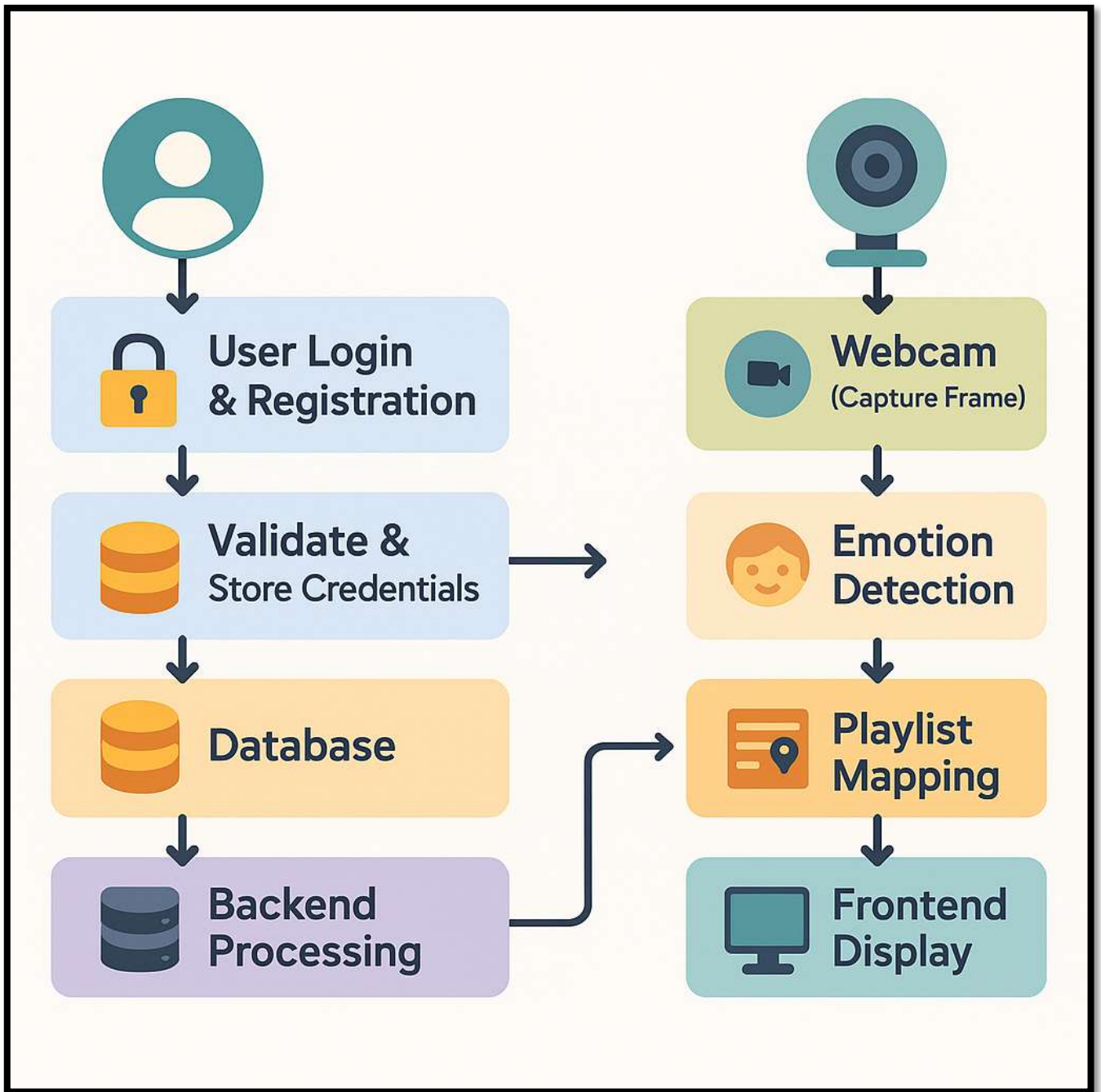


Figure 2 Structure Diagram

6.2. Data Structures, Database and File Specifications:

6.2.1. Data Structures:

- **User Object (for MongoDB)**
 - **Fields:**
 - username: String
 - password: String (hashed)
 - email: String
 - history: Array of objects (each object contains emotion, timestamp, and

musicRecommendations) [5]

- **Emotion Object (for MongoDB)**

- **Fields:**

- emotionType: String (e.g., Happy, Sad, Angry)

- confidenceLevel: Float (measuring how confident the system is in its emotion detection)

- **MusicRecommendation Object**

- **Fields:**

- musicID: String (unique identifier from YouTube)

- title: String (song title)

- url: String (YouTube video URL)

- genre: String (music genre associated with the emotion)

6.2.2. Database (MongoDB):

- **Collections:**

- Users:** Stores user-related data (username, email, password).

- EmotionHistory:** Stores the user's emotional states over time,

- MusicRecommendations:** Stores the recommended music and the emotion it was mapped to.

6.3. Pseudo Code:

6.3.1. Emotion Detection Pseudocode:

Function DetectEmotion(image):

Load pre-trained FER model [2] [3] [12]

Process image (resize, normalize) [1]

Detect faces in image [11]

For each face in the image:

Predict emotion (happy, sad, angry, neutral) [12]

Return the most confident emotion detected

6.3.2. Music Recommendation Pseudocode:

Function GetMusicRecommendation(emotion):

Map emotion to genre:

- Happy -> Upbeat music

- Sad -> Calm music

- Angry -> High-energy music

Query YouTube API with mapped genre

Fetch top 25 music results based on emotion

Return list of music videos (title, URL)

6.3.3. User Login Pseudocode:

Function UserLogin(username, password):

Retrieve stored credentials from Database [5]

If credentials match:

Generate JWT token for session [4]

Return success message with token

Else:

Return error message "Invalid credentials"

6.3.4. Frontend Pseudocode (for Displaying Music Recommendations):

Function DisplayRecommendations(emotion, recommendations: Display detected emotion on the screen

For each recommendation in recommendations:

Display music title and thumbnail

Provide link to open music on YouTube

6.4. Screenshots:

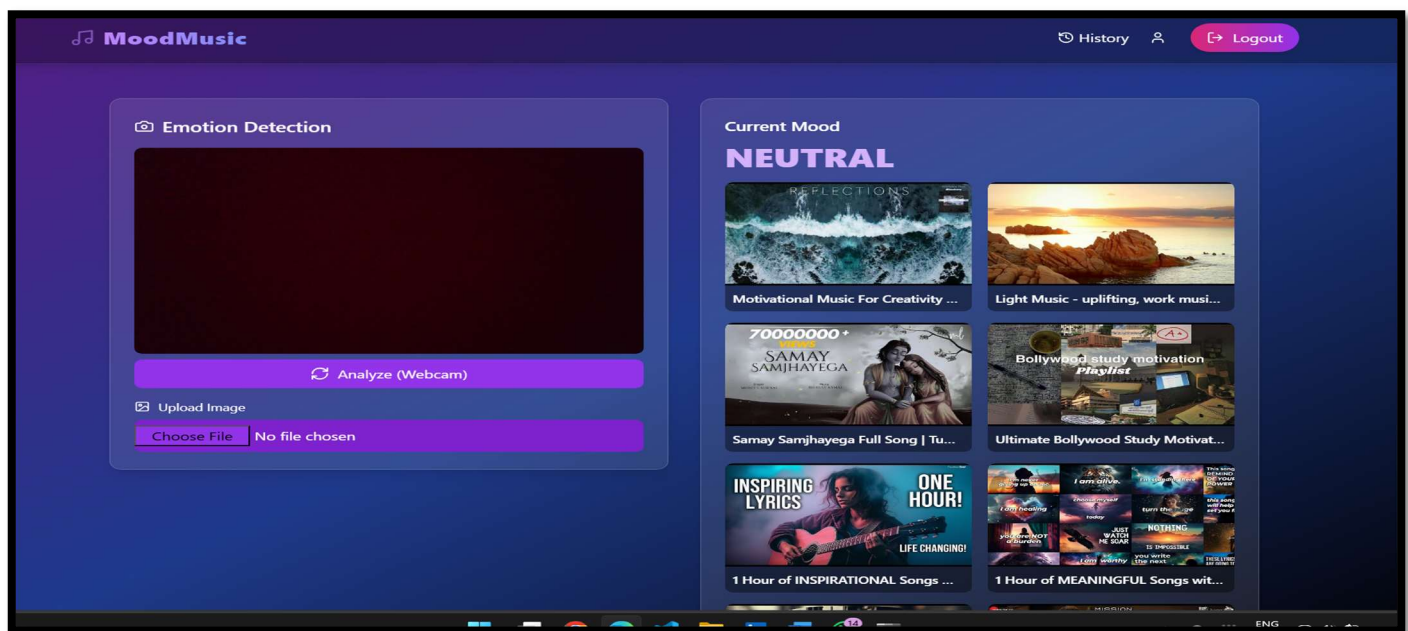




Figure 3 Emotion Detection

 MoodMusic


[→\] Login](#) [Sign Up](#)

Create Account


Username


 Username

Password




Confirm Password



 Create Account

Already have an account? [Sign in](#)


Figure 4 Sign up

 MoodMusic


[→\] Login](#) [Sign Up](#)


Welcome Back

Username

 Username

Password



 Sign In

Don't have an account? [Sign up](#)

Figure 5 Login

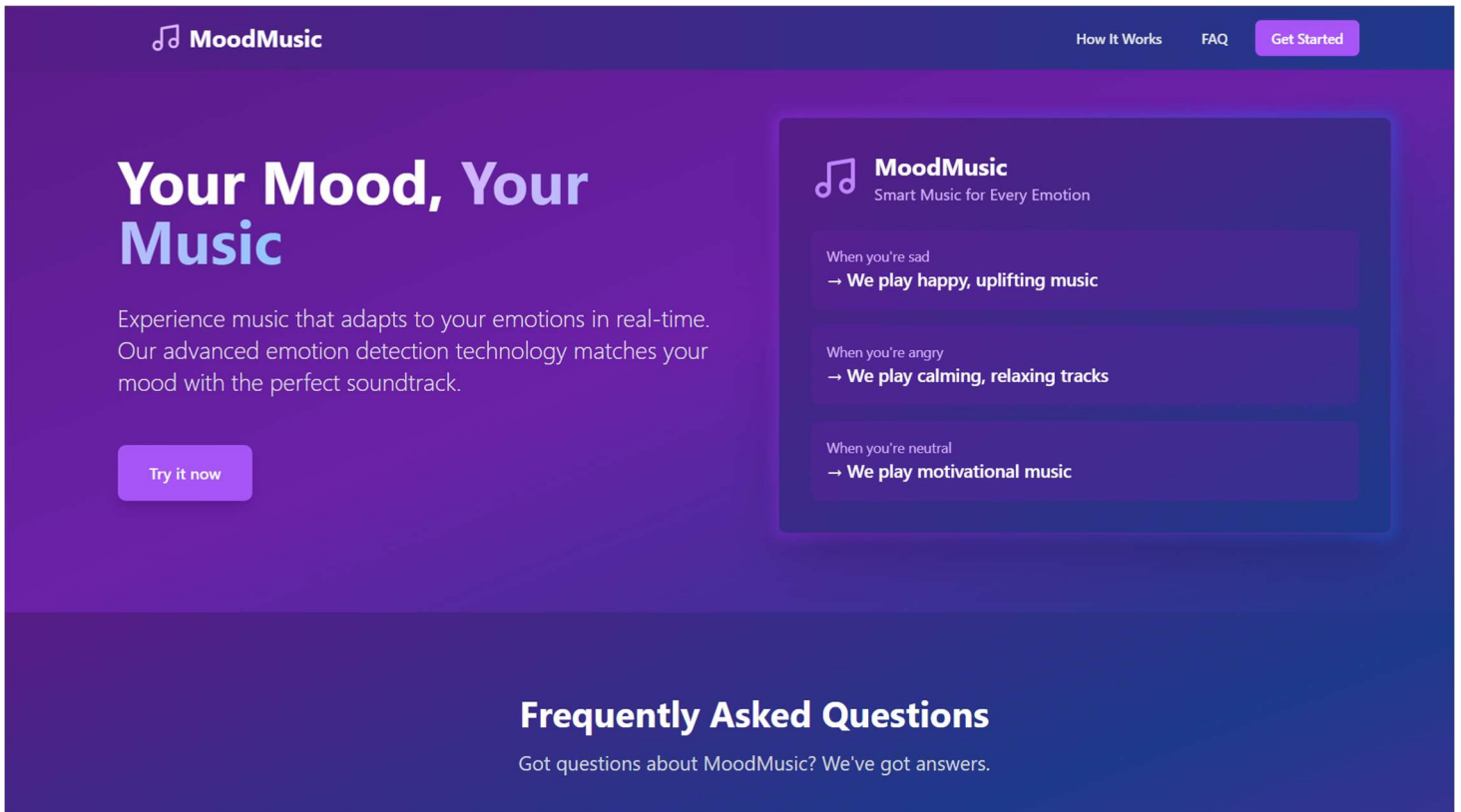


Figure 6 Landing Page

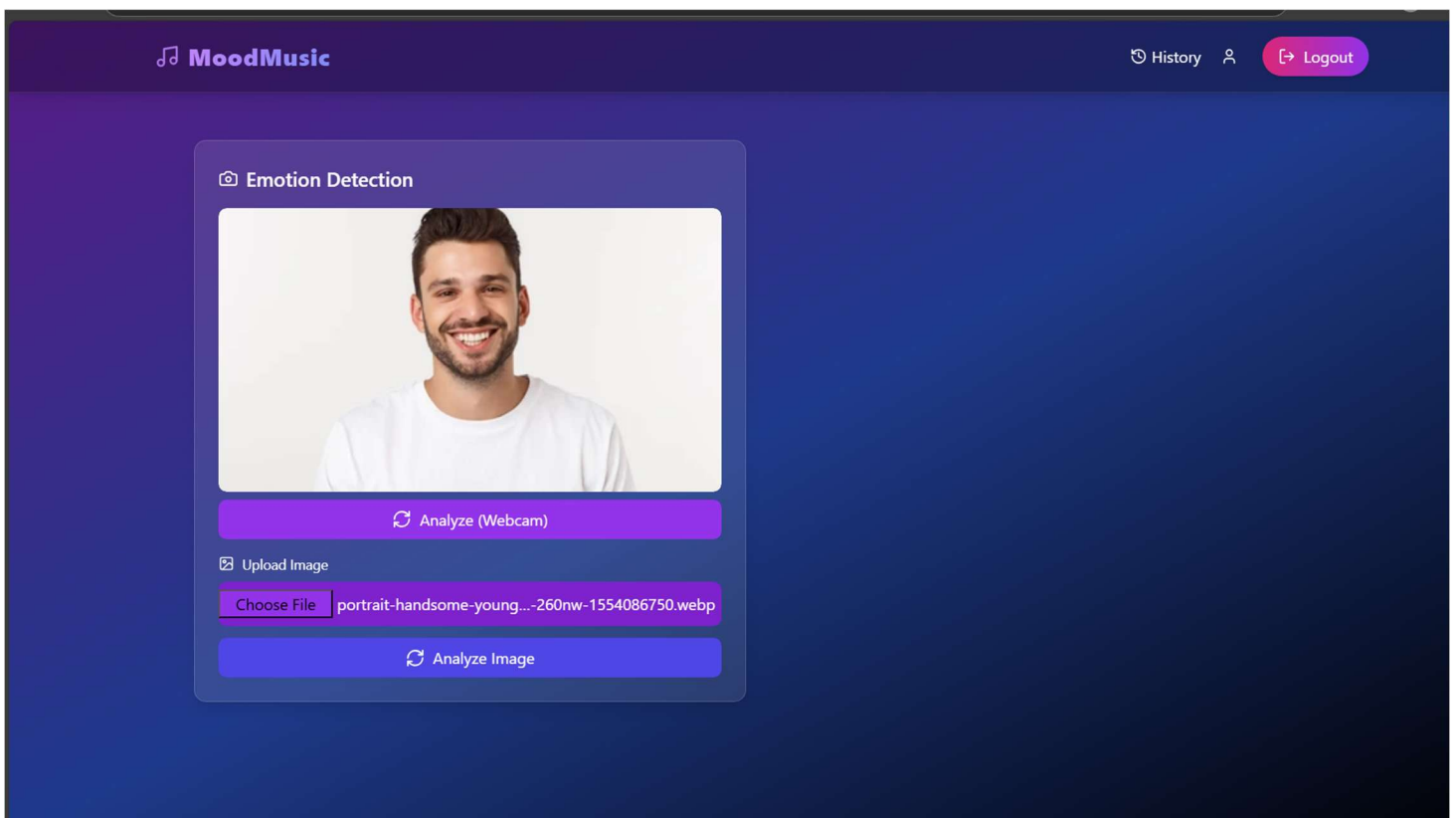


Figure 7 Upload Image

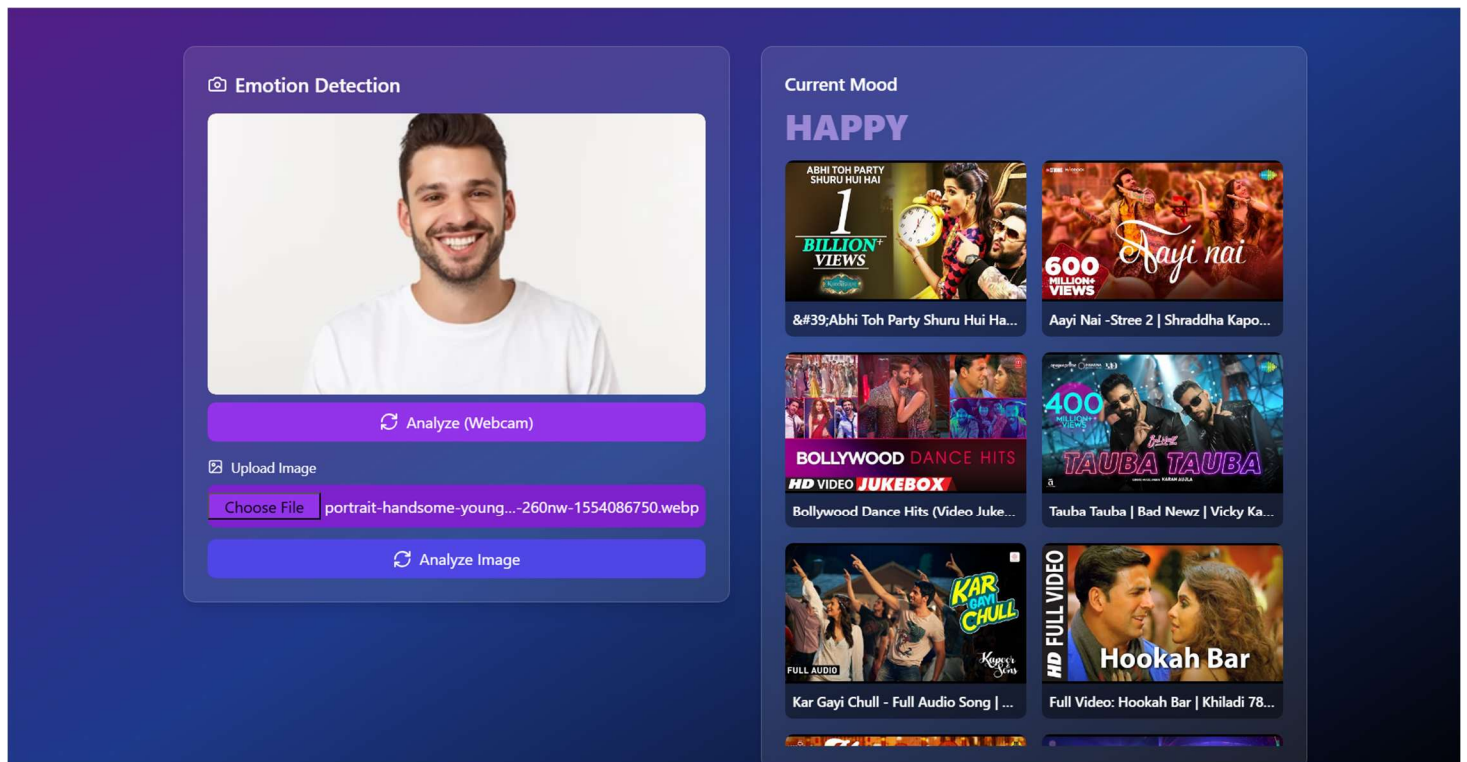


Figure 8 Happy Emotion Detected

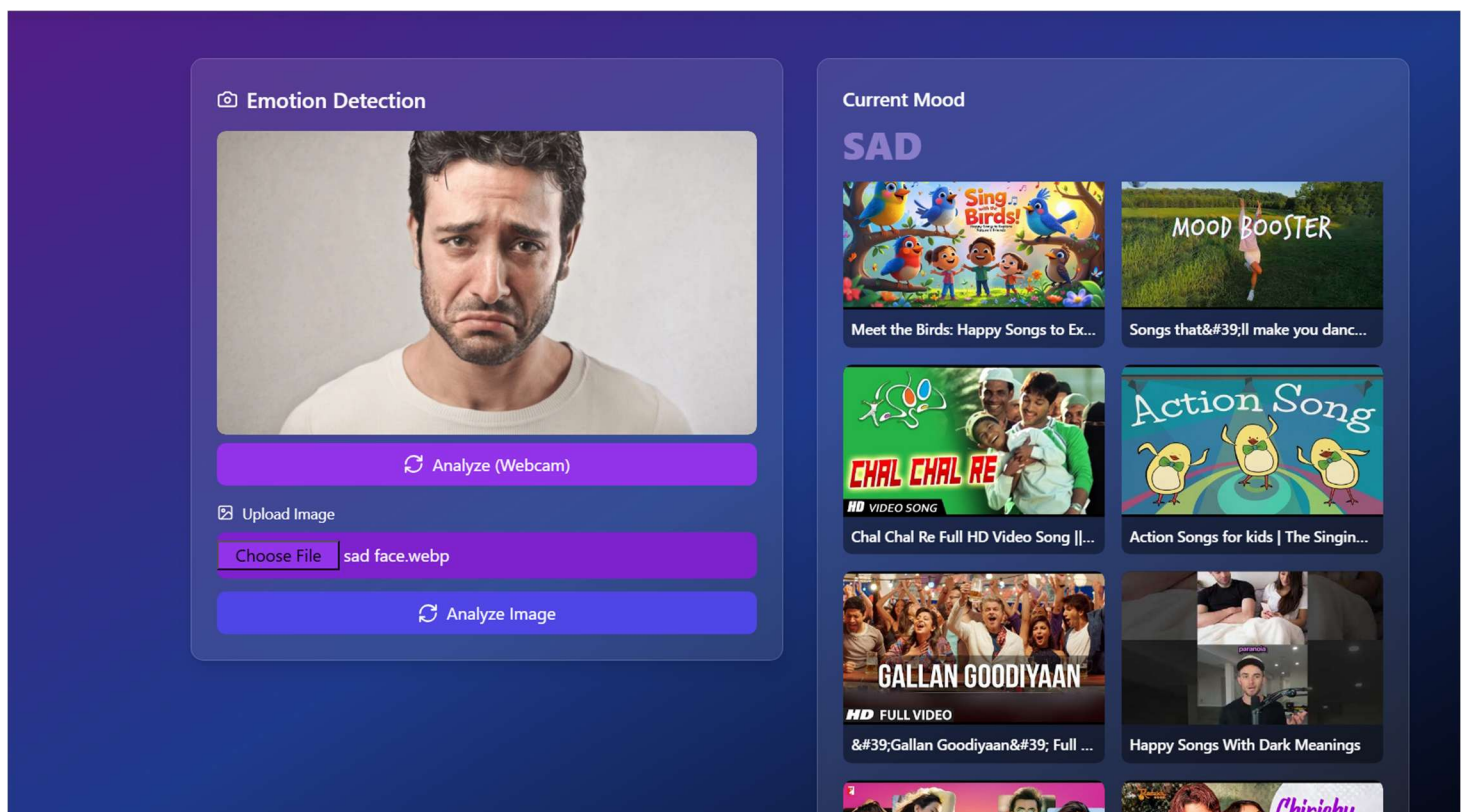


Figure 9 Sad Emotion Detected

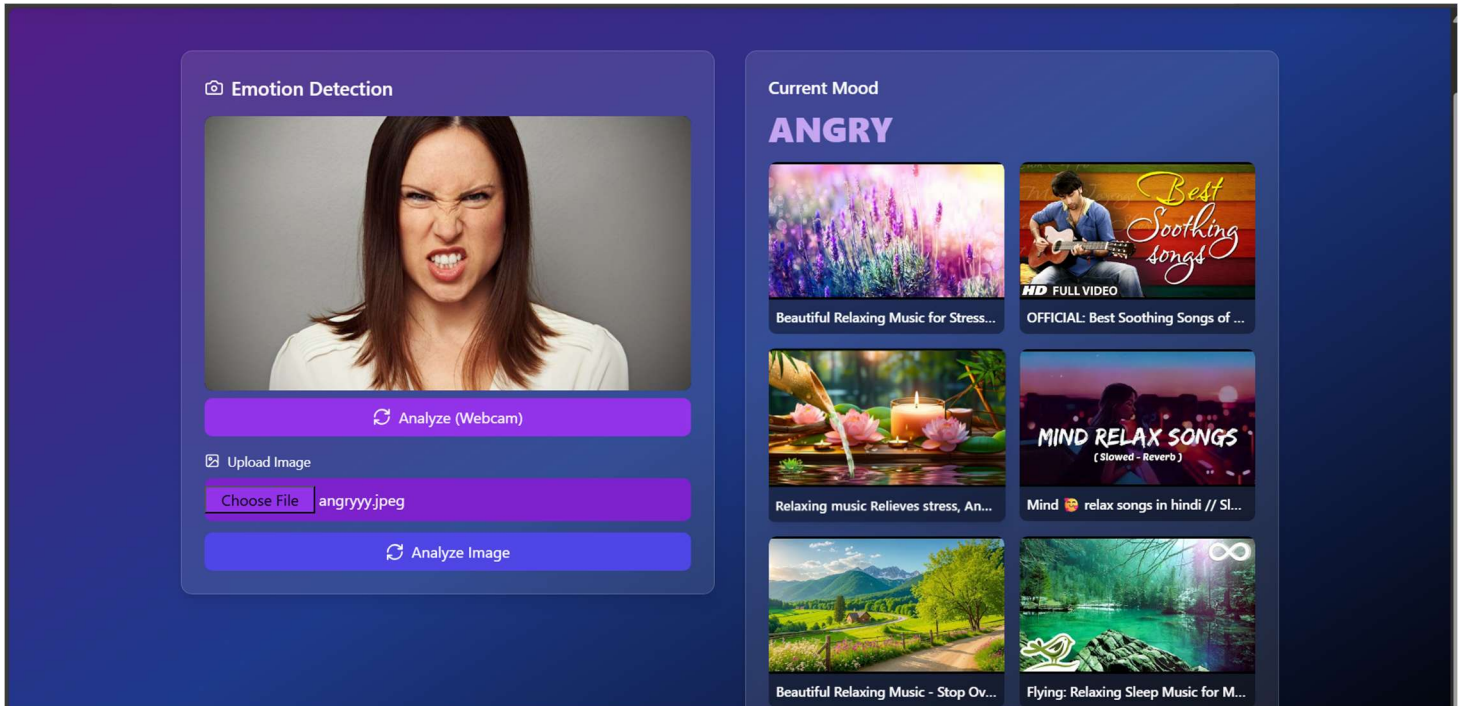


Figure 10 Angry Emotion Detected

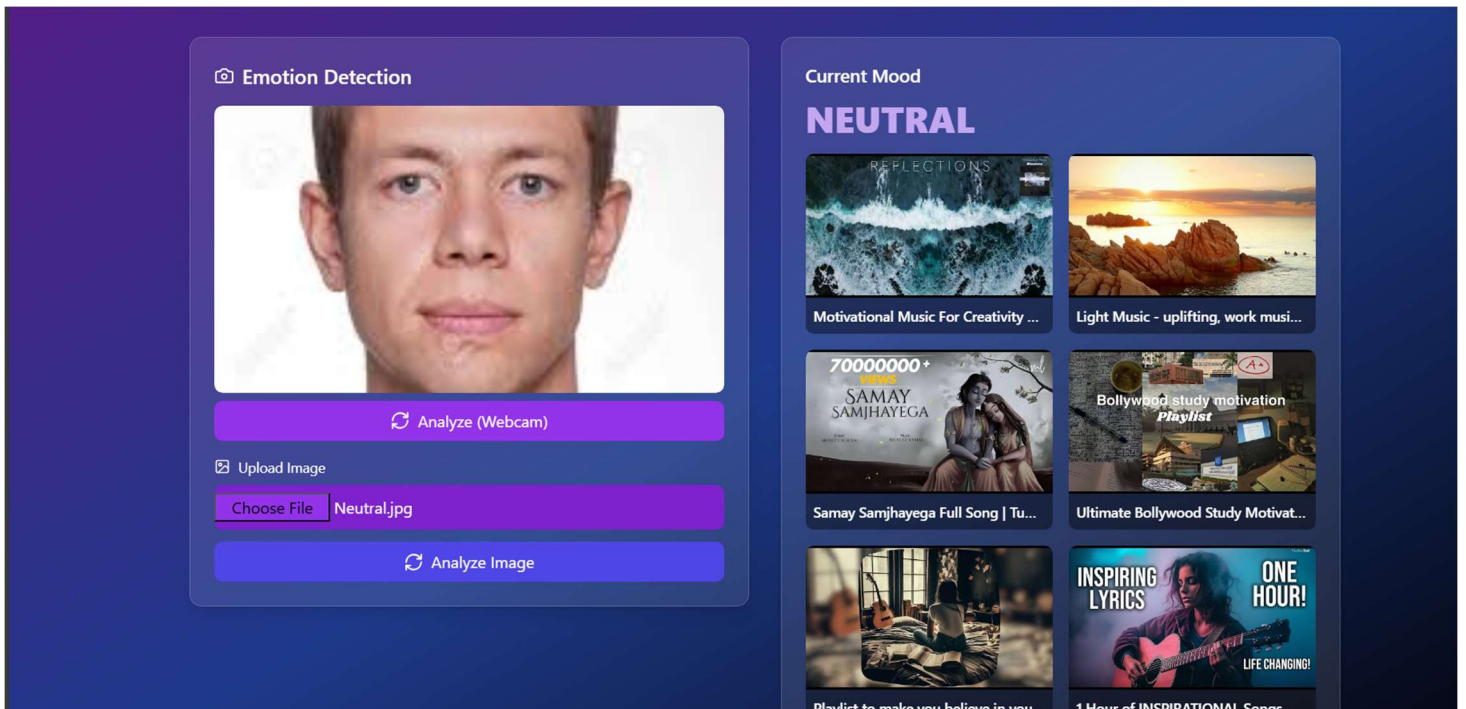


Figure 11 Neutral Emotion Detected

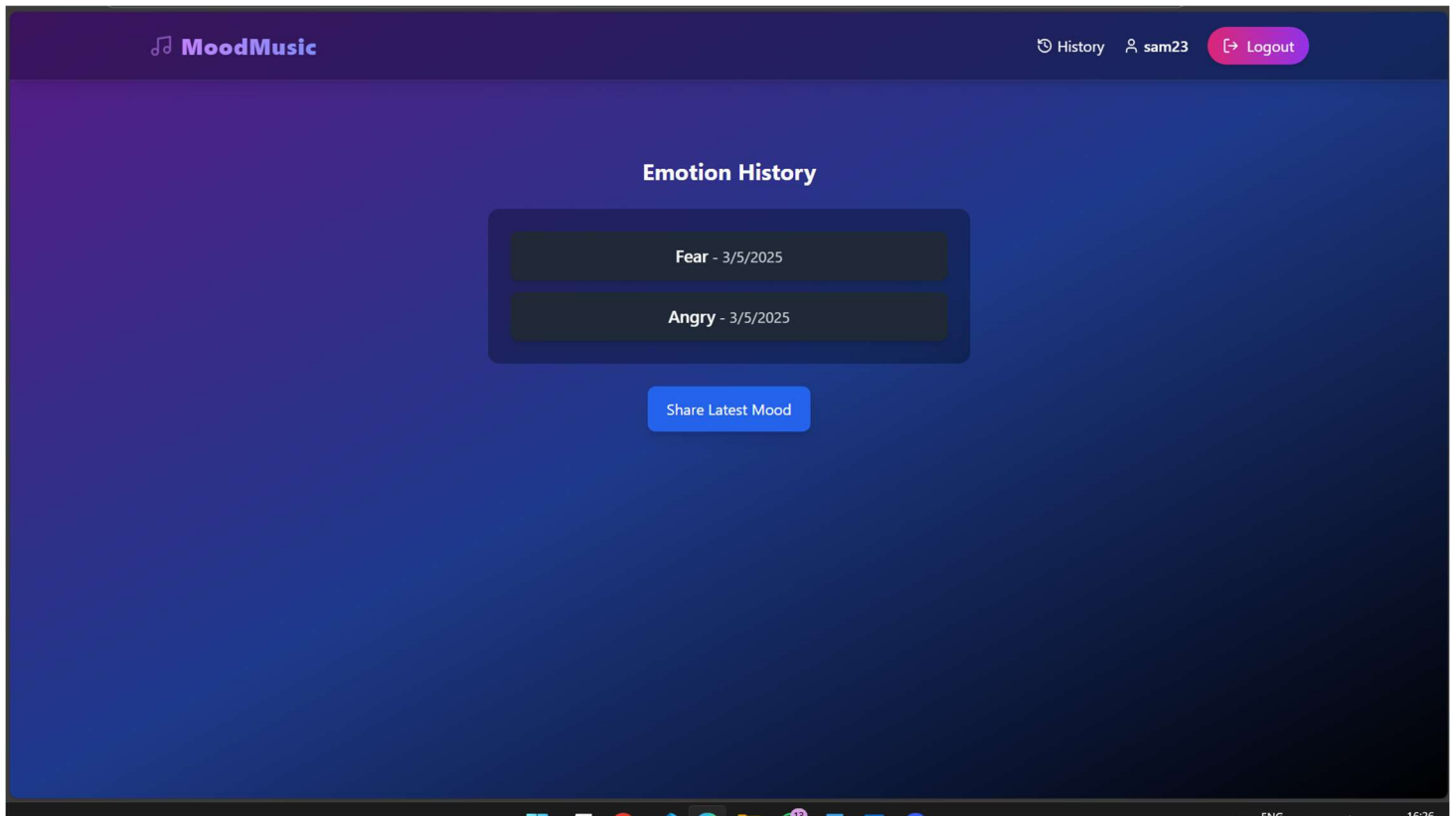


Figure 12 Emotion History Page

7. Test Plan

7.1. Functional Tests

Functional tests focus on verifying that each feature of the system functions as intended.

7.1.1. User Registration and Login Test

- **Objective:**
Verify that users can successfully register and log in.
- **Test Steps:**
Go to the registration page and enter valid credentials (username, email, password).
Submit the registration form.
Check if the user is redirected to the login page after registration.
Log in using the registered credentials.
Check if the user is successfully logged in and redirected to the home page or user dashboard.
- **Expected Result:**
User is able to register and log in successfully, and is redirected to the appropriate page after login [4].

7.1.2. Emotion Detection Test

- **Objective:**
Verify that the system can accurately detect emotions from the webcam feed or uploaded images.
- **Test Steps:**
Upload an image or enable the webcam.

The system should detect the user's facial expression (e.g., happiness, sadness, anger).

Check if the detected emotion matches the user's facial expression

- **Expected Result:**

The system detects the correct emotion with high accuracy [12]

7.1.3. Music Recommendation Test

- **Objective:**

Ensure that the system recommends music based on the detected emotion.

- **Test Steps:**

After the emotion is detected (e.g., happy), check if the system queries the YouTube API for appropriate music.

Ensure the correct genre of music is recommended.

Check the frontend display for correct video thumbnails, titles, and links to YouTube.

- **Expected Result:**

Music recommendations are relevant to the detected emotion [10] , and users can view and interact with the music recommendations on the UI.

7.1.4. User History Tracking Test

- **Objective:**

Verify that the system correctly stores and displays user emotion history.

- **Test Steps:**

Detect multiple emotions (e.g., happy, sad).

Check if the system stores emotion history in the database[5] .

Check if the history is displayed correctly in the user interface.

- **Expected Result:**

The user's emotion history is correctly stored and displayed, showing the corresponding music recommendations for each emotion

7.1.5. Social Media Sharing Test

- **Objective:**

Ensure users can share their emotions and music recommendations on social media platforms (e.g., Facebook, Twitter).

- **Test Steps:**

After emotion detection and music recommendation, click the Share button.

Verify that the correct music and emotion details are shared to the selected platform (e.g., Twitter, Facebook).

- **Expected Result:**

The correct data is shared on the selected social media platform, and the link to the music video is included.

7.2. Performance Tests

Performance tests focus on verifying that the system performs well under normal and peak

conditions.

7.2.1. Response Time Test

- **Objective:**
Measure how long it takes for the system to detect emotions and provide music recommendations.
- **Test Steps:**
Upload an image or use the webcam for emotion detection.
Measure the time taken from the moment the image is uploaded to the time the music recommendations are displayed
- **Expected Result:**
The system should detect emotions and provide music recommendations within 5 seconds or less. [5]

7.2.2. Load Test (Normal Load)

- **Objective:**
Ensure the system can handle the expected number of users
- **Test Steps:**
Simulate normal usage (e.g., 20-50 concurrent users) accessing the system.
Check if the system performs well without crashes or significant delays.
- **Expected Result:**
The system should perform with no major slowdowns or errors under normal load conditions.

7.2.3. Scalability Test

- **Objective:**
Ensure the system can scale to support an increasing number of users
- **Test Steps:**
Gradually increase the number of users accessing the system concurrently (e.g., from 50 to 500).
Monitor system performance and resource usage.
- **Expected Result:**
The system should remain stable and responsive as the number of concurrent users increases. The response time should not degrade significantly under load.

8. Implementation/Conversion Plan

8.1. Requirement Gathering and System Design (Week 1-2)

Gather functional and non-functional requirements.

Finalize technologies: Flask for backend, React.js for frontend, MongoDB for database, YouTube API for music recommendations.

Design the system architecture (client-server, database design).

Set up the GitHub repository for version control.

8.2. Backend Development (Week 3-4)

Initialize Flask backend and set up RESTful APIs for user registration, emotion detection, and music recommendation. [2]

Integrate FER for emotion detection and connect to YouTube Data API for fetching music recommendations. [3] [12]

Set up MongoDB for storing user data and emotion history.[5]

Test the backend services locally.

8.3. Frontend Development (Week 5-6)

Set up React.js for the frontend interface, including registration, login, and emotion detection features. [6]

Integrate frontend with backend services using Axios for API calls.

Implement responsive design with Tailwind CSS.[7]

Test frontend locally and ensure smooth interaction between frontend and backend.

8.4. Testing and Debugging (Week 7)

8.4.1. Functional Testing:

Test user registration, emotion detection, music recommendation, and history tracking.

Verify the accuracy of emotion detection and the relevance of music recommendations.

8.4.2. Performance Testing:

Measure the response time for emotion detection and music recommendations.

Ensure the system handles a reasonable number of concurrent users without significant performance issues.

8.4.3. Bug Fixing:

Identify and fix bugs or errors in both the frontend and backend.

Test edge cases such as invalid user data, non-detectable facial expressions, and API failures.

8.4.4. Debugging:

Check for memory leaks, incorrect API responses, and unexpected crashes.

Expected Outcome: The system should pass all functional tests, meet performance expectations, and be free of major bugs or issues.

9. Project Legacy

9.1. Current Status of Project

9.1.1. User Authentication:

Registration and login processes are working seamlessly with JWT-based authentication. [4]

9.1.2. Emotion Detection:

The system accurately detects emotions (happiness, sadness, anger, etc.) from both

uploaded images and live webcam feeds using the FER library.[12]

9.1.3. Music Recommendation:

The system queries the YouTube Data API to fetch and display music recommendations based on the detected emotion.

9.1.4. Frontend and Backend Integration:

The React [6] frontend is fully integrated with the Flask [4] backend, ensuring smooth communication for emotion detection and music recommendations.

9.2. Remaining Areas of Concern

9.2.1. Frontend User Interface:

While the user interface is functional, it could benefit from a more refined design. Visual feedback (e.g., progress indicators while the emotion detection model is processing) can improve the user experience.

There's potential to add more engaging animations and transitions to the UI to make it more visually appealing and interactive.

9.2.2. Mobile Compatibility:

The current web application is designed for desktop browsers, but extending it to mobile platforms is important. Mobile users will need a responsive or native app to interact with the system on the go, especially for real-time emotion detection using mobile cameras

9.3. Technical and Managerial Lessons Learned

9.3.1. Technical Lessons:

- **Frontend-Backend Communication:**

Integrating React.js with a Flask backend was an interesting challenge, particularly around managing state and data flow between the two components. Using Axios for API calls helped, but ensuring proper error handling and data validation on both ends was critical for system stability.

- **Database Schema Design:**

MongoDB is flexible and works well with the document-based structure for storing user and emotion data. However, data consistency and performance are areas to consider as the user base expands. Indexing important fields like emotion history and user preferences can help with faster queries and scalability.

9.3.2. Managerial Lessons:

- **Iterative Development:**

The project followed an iterative development model, which focused on building and testing small functional components one at a time. This allowed the team to focus on core features like emotion detection and music recommendation early on and debug them before adding more complex elements. Each iteration brought valuable user feedback and real-world testing

data, preventing accumulation of unresolved issues.

- **Time Management:**

Given the project's timeline, effective time management was a priority. Tasks were broken down into smaller milestones, such as backend setup, emotion detection integration, and UI design, ensuring the team remained on track. Regular check-ins helped identify bottlenecks early, allowing the team to adapt and prioritize tasks as necessary.

9.4. Future Recommendations:

9.4.1. Enhanced Music Recommendation System:

Currently, the system relies on the YouTube Data API for music recommendations, but adding more sources such as Spotify or Apple Music could diversify the music library and provide more personalized suggestions. Additionally, leveraging AI-driven music personalization, where the system learns from the user's listening behavior over time, can help in making smarter recommendations.

9.4.2. Integration of Voice-Based Emotion Detection:

Currently, emotion detection is based on facial expressions. Expanding this to include voice-based emotion detection could further enhance the system, especially for users who prefer using voice commands or audio instead of images.

10. Bibliography:

- [1] OpenCV, "OpenCV Documentation," 2021. [Online]. Available: <https://docs.opencv.org/4.x/d1/dfb/intro.html>. [Accessed: Feb. 10, 2025].
- [2] TensorFlow, "TensorFlow Documentation," 2021. [Online]. Available: <https://www.tensorflow.org/>. [Accessed: Feb. 10, 2025].
- [3] PyTorch, "PyTorch Documentation," 2021. [Online]. Available: <https://pytorch.org/>. [Accessed: Feb. 10, 2025]
- [4] Flask, "Flask Documentation," 2021. [Online]. Available: <https://flask.palletsprojects.com/>. [Accessed: Feb. 9, 2025].
- [5] MongoDB, "MongoDB Documentation," 2021. [Online]. Available: <https://docs.mongodb.com/>. [Accessed: Feb. 10, 2025].
- [6] React, "React Documentation," 2021. [Online]. Available: <https://react.dev/>. [Accessed: Feb. 17, 2025].
- [7] Tailwind CSS, "Tailwind CSS Documentation," 2021. [Online]. Available: <https://tailwindcss.com/>. [Accessed: Feb. 10, 2025].
- [8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: [10.1038/nature14539](https://doi.org/10.1038/nature14539). [Accessed: Feb. 12, 2025]
- [9] Z. Zhang, W. Chen, and Y. Chen, "Facial expression recognition based on convolutional neural networks," *IEEE Trans. Affective Comput.*, vol. 8, no. 4, pp. 379–391, Oct.-Dec. 2017, doi: [10.1109/TAFFC.2016.2621400](https://doi.org/10.1109/TAFFC.2016.2621400). [Accessed: Feb. 12, 2025].

- [10] L. Yao and J. Huang, "Content-based recommendation system using deep learning," in *Proc. IEEE 13th Int. Conf. Semantic Comput. (ICSC)*, 2017, pp. 329–332, doi: [10.1109/ICSC.2017.61](https://doi.org/10.1109/ICSC.2017.61). [Accessed: Feb. 12, 2025].
- [11] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. I–511–I–518, Dec. 2001, doi: [10.1109/CVPR.2001.990517](https://doi.org/10.1109/CVPR.2001.990517). [Accessed: Feb 12, 2025]
- [12] S. Li and W. Deng, "Deep facial expression recognition: A survey," *IEEE Trans. Affective Comput.*, vol. 10, no. 4, pp. 471–493, Oct.–Dec. 2019, doi: [10.1109/TAFFC.2018.2858791](https://doi.org/10.1109/TAFFC.2018.2858791). [Accessed: Feb 14, 2025]
- [13] B. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, Jun. 2015, doi: [10.1109/CVPR.2015.7298682](https://doi.org/10.1109/CVPR.2015.7298682). [Accessed: Feb 14, 2025]

11. Source Code:

11.1. Backend:

11.1.1. Code:

```
from flask import Flask, request, jsonify
from flask_jwt_extended import JWTManager, create_access_token, jwt_required
from pymongo import MongoClient
import cv2

app = Flask(__name__)
jwt = JWTManager(app)

@app.route('/api/register', methods=['POST'])
def register():
    Create a new user with hashed password
    data = request.get_json()
    Validate input
    Hash password and insert user into MongoDB
    return jsonify(message='Registered'), 201

@app.route('/api/login', methods=['POST'])
def login():
    Authenticate user and return a JWT.
```

```

data = request.get_json()
token = create_access_token(identity=data.get('username'))
return jsonify(token=token), 200

@app.route('/api/detect_emotion', methods=['POST'])
@jwt_required(optional=True)
def detect_emotion():
    Capture frame, predict emotion, and fetch matching videos.
    _, frame = camera.read()
    emotion = emotion_recognizer.predict_emotion(frame)
    videos = fetch_youtube_videos(emotion)
    return jsonify(emotion=emotion, videos=videos)

```

11.1.2. Explanation:

- Initializes Flask with JWT for secure routes.
- register: handles new user creation.
- login: checks credentials and issues a JWT.
- detect_emotion: reads a camera frame, uses FER to detect emotion, and returns video recommendations.

11.2. Frontend:

11.2.1. Code:

```

import React from 'react';
import { Routes, Route } from 'react-router-dom';

function App() {
    State hooks manage mood, videoList, loading, error, and file uploads
    fetchSongs(), startAnalysis(), and analyzeImage() handle API interactions

    return (
        <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Register />} />
        <Route path="/history" element={<History />} />
        </Routes>
    );
}

export default App;

```

11.2.2. Explanation:

- Defines main routes for the app.
- Manages global state and delegates emotion analysis to helper functions.
- Renders different pages (Home, Login, Register, History).

11.3. EmotionDetector:

11.3.1. Code:

```
import React, { useState } from 'react';

const EmotionDetector = () => {
  const [emotion, setEmotion] = useState<string | null>(null);
  const [loading, setLoading] = useState(false);

  const analyze = async () => {
    setLoading(true);
    Simulate or call backend API
    setEmotion('happy');
    setLoading(false);
  };

  return (
    <div>
      <button onClick={analyze} disabled={loading}>
        {loading ? 'Analyzing...' : 'Analyze Emotion'}
      </button>
      {emotion && <p>Detected: {emotion}</p>}
    </div>
  );
};
```

11.3.2. Explanation:

- Uses useState to track detected emotion and loading status.
- analyze: triggers emotion detection (simulated or via API).
- Renders a button and displays the resulting emotion.