Department of Computer Science

Spring 2024

Project Documentation on

MOOD-BASED MUSIC RECOMMENDATION SYSTEM USING CONVOLUTIONAL NEURAL NETWORKS

by

MET-A-FOUR

PREPARED FOR

Dr. Hubert A. Johnson
CSIT 515-02 Software Engineering & Reliability, Spring 2024

TEAM MEMBERS

Chiru Anand Vaka
Venkata Roshan Reddy Rajala
Saikumar Atchi
Heet Patel

Table of Contents

Documentation Revision History 3
Contact Information 3
Section A: Specification 4
Section B: Design 12
Section C: Test Plan 48
Maintenance and Support 65
Documentation and Training 65
References 66

Documentation Revision History

Date	Notes	Version
February 29, 2024	Initial Release 1.0	1.0
March 15, 2024	Modifications to SRS and Design	1.1
March 16, 2024	Addition of Test Plan	1.2
March 28, 2024	The final revision of all sections	1.3
April 7, 2024	Addition of new functionality – Track Attendance	1.4
April 20, 2024	The final revision of all sections	1.5

Contact Information

Name	Email	Mobile Number
Chiru Anand Vaka	vakac1@montclair.edu	+1 862 381 2748
Venkata Roshan Reddy Rajala	rajalav1@montclair.edu	+1 838 207 0559
Saikumar Atchi	atchis1@montclair.edu	+1 862 297 1012
Heet Patel	patelh57@montclair.edu	+1 919 548 7135

Professor Hubert Johnson Signature Date

Section A: Specification

Contents

Documentation Revision History 3
Contact Information 3
Section A : Specification 4
1. Introduction 6
1.2 Product Scope 6
1.3 Product Features 7
1.4 Intended Use 7
1.5 Intended Audience 7
2. Overall Description 7
2.1 Terminology 8
2.2 User Needs 8
2.3 User Characteristics 9
2.4 Design and Implementation Constraints 9
2.5 Assumptions and Dependencies 9
2.6 Feasibility 10
3.System Features and Requirements 11
3.1 Functional Requirements 11
3.1.1 Facial Expression Recognition 11
3.1.2 Music Recommendation 11
3.1.3 Real-time Processing. 11
3.1.4 User Interaction 11
3.1.5 Data Handling 11
3.2 Non-Functional Requirements 11
4. System Requirements 12
4.1 Hardware Requirements 12
4.2 Software Requirements 12

1. Introduction

"In the rapidly evolving world of music consumption, personalization will become paramount in enhancing the listener's experience. Recognizing the deep connection between music and emotions, our project will introduce an innovative Mood-Based Music Recommendation System. This system will leverage advanced technology to analyse users' facial expressions in real-time. By accurately detecting the user's current emotional state from their facial cues, our system will dynamically recommend songs from their existing playlist that align with their emotions. This approach will not only enhance the user's engagement by providing a personalized listening experience but also will pave the way for applications extending beyond entertainment, including therapeutic and mood-enhancement scenarios. The system promises a unique blend of emotional intelligence and technological sophistication, aiming to revolutionize the way users interact with music."

1.1 Purpose

The purpose of the Mood-Based Music Recommendation System will be to offer a personalized and dynamic listening experience by utilizing advanced facial recognition and deep learning technologies. The system will aim to analyze a user's current emotional state through their facial expressions and recommend music that aligns with their mood. This innovative approach will enhance user engagement by adapting music selections in real time to fit the emotional needs of the listener, ensuring a highly personalized and satisfying musical experience. This technology will have the potential to transform music listening by making it more interactive and responsive, thereby catering to the emotional well-being of users.

1.2 Product Scope

The product scope of the "Mood Based Music Recommendation System Using Convolutional Neural Networks" will involve creating a sophisticated system that will use advanced facial recognition and deep learning technologies to analyze users' emotional states and recommend music accordingly. This system will be capable of real-time emotion detection and music recommendation, providing users with playlists that reflect their current moods to enhance their listening experience.

1) **Real-Time Emotion Detection:** Cutting-edge facial recognition technology will be utilized to assess users' moods quickly and accurately.

2) **Deep Learning Integration:** The Inception model will be employed to process and understand the nuances of facial expressions associated with different emotional states.

3) **Personalized Music Recommendations:** The platform will enable precise identification and authentication of detected faces using a stored face dataset of registered individuals. The system shall identify faces as "unrecognized" if their facial features are not present or registered in the dataset.

4) **User-Friendly Interface:** The system will be accessible through a web application that will be easy to navigate, enhancing user interaction and satisfaction.

5) **Scalability and Extensibility:** The system will empower organizers with seamless attendance tracking, automating the process efficiently.

6) **Research and Development:** The system will provide administrators with the capability to access and manage system logs, allowing for effective monitoring and oversight of system activities. Admins will be able to review logs for security and system performance analysis, ensuring a comprehensive administrative control layer.

1.3 Product Features

The Face Recognition System will encompass essential functionalities, including:

- Facial Detection for users
- Feature Extraction for detected faces
- Mood Detection
- Suggests music from existing playlists.
- User Interface

1.4 Intended Use

The intended use of the "Mood Based Music Recommendation System Using Convolutional Neural Networks" project, as described in the uploaded document, is to create a system that analyses the user's facial expressions to determine their mood and then recommends music based on the detected mood. The system aims to provide a personalized music listening experience that enhances the user's mood by suggesting songs that match their current emotional state. This is achieved through the application of deep learning techniques, specifically convolutional neural networks (CNNs) and facial recognition technologies, to efficiently recognize emotions and suggest suitable music. The project is intended to improve user interaction with music platforms by offering more relevant music choices that could improve the listener's overall experience and emotional well-being.

1.5 Intended Audience

The target audience for the "Mood-Based Music Recommendation System will include music listeners seeking personalized playlists, mental health advocates, tech enthusiasts, and early adopters, computer science students and developers, and music industry stakeholders interested in enhancing user engagement.

2. Overall Description

The "Mood Based Music Recommendation System Using Convolutional Neural Networks" will significantly enhance user experiences by meeting specific preferences and technical requirements, carefully considering essential assumptions and feasibility to align with music listeners' needs for personalized playlists based on their emotional state.

2.1 Terminology

The provided terms aim to enhance clarity and maintain consistency when denoting concepts and entities in the context of developing the Mood-Based Music Recommendation System.

- **RAM:** Random Access Memory
- **GB:** Gigabyte
- **CNN:** Convolutional Neural Network
- **GUI:** Graphical User Interface
- **P:** Pixel
- **D:** Dimensional
- **Developers / The Team:** Met-a-Four.
- **UCD:** Use Case Diagram
- **Admin:** Administrator
- **Users:** Music Listeners

2.2 User Needs

Mood based Music Recommendation system will be designed to meet the following user needs:

1) **Personalized Music Recommendations:** Users will need the system to analyse their current emotional state through facial expressions and recommend music playlists that match their mood, enhancing their listening experience.

2) **Secure Authentication:** The system will need to recognize the user's emotions efficiently and accurately in real-time to provide instant music suggestions that are appropriate for their current feelings.

3) **Efficient Attendance Tracking:** The interface will need to be user-friendly, allowing users to easily interact with the system without requiring technical expertise, ensuring they can enjoy music seamlessly according to their emotional state.
4) **Real-time Emotion Recognition:** Users will need the system to be highly accurate in detecting emotions to avoid inappropriate music recommendations, enhancing user satisfaction and trust in the system.
5) **Personalized Music Recommendations:** Privacy and Security: As the system involves processing sensitive personal data (facial expressions), users will need assurance that their information is handled securely and privately, with clear policies on data usage and protection.
6) **Speed and Efficiency:** The system will need to be fast enough to analyze facial expressions and suggest music without noticeable delays, providing a smooth and uninterrupted music listening experience.

2.3 User Characteristics

The system will cater to the following user classes in the future:

1) **Administrator:** The Administrator will be responsible for configuring and managing the system, overseeing user data security, maintaining system functionality, providing technical support, and implementing updates based on user feedback and compliance requirements.
2) **End Users:** Individuals will seek a personalized music experience that adjusts dynamically to their mood, utilizing the system to enhance their emotional well-being or match their current feelings with suitable music.

2.4 Design and Implementation Constraints

Environmental Conditions: The system's accuracy in emotion recognition will be contingent on environmental factors, such as lighting and background noise, which could potentially interfere with facial expression analysis.

System Compatibility: The system will need to be compatible with a wide range of devices and operating systems to ensure broad accessibility and usability, requiring comprehensive testing and optimization across different hardware configurations.

Real-Time Processing: The system will require robust computational power to process facial recognition data in real time, posing a challenge in terms of maintaining system responsiveness and speed for all users, particularly those with less capable hardware.

Integration with Music Streaming APIs: Effective integration with multiple music streaming services will be crucial. This involves managing API limitations and ensuring consistent functionality across various platforms to provide seamless music recommendations.

Scalability and Maintenance: The system will need to be scalable to handle growing numbers of users and data without degradation in performance. Regular maintenance and updates will be required to address new challenges and incorporate feedback.
Data Privacy and Security: Ensuring the privacy and security of sensitive user data, particularly facial recognition data, will be critical. The system must comply with relevant data protection regulations and ensure robust security measures to protect user information.

2.5 Assumptions and Dependencies

Certain assumptions and dependencies are considered for the successful implementation of Mood-Based Music Recommendation System

1) **Reliable Internet Connection:** The system will assume that users have access to a reliable Internet connection, which is essential for participating in video conferences.

2) **Functional Cameras**: Successful facial recognition will rely on users having functional cameras on their devices, ensuring the system can capture and analyse facial features effectively.

3) **User Interaction:** In certain scenarios, the system will rely on user collaboration, necessitating actions such as actively engaging with the camera, addressing potential obstructions like challenging lighting conditions, and ensuring an unobstructed camera lens for precise recognition. Additionally, users will be expected to remain stationary in front of the camera during the recognition process.

4) **Hardware and System Maintenance:** The reliable performance of hardware components, including adequate power backup, shall play a critical role in ensuring the system functions optimally.

2.6 Feasibility

The feasibility of Mood Based Music Recommendation System will involve assessing various aspects:

1) **Operational Feasibility:** Operationally, the system will integrate seamlessly into users' music listening routines, providing enhanced personalization without disrupting user experience.

2) **Technical Feasibility:** The system's ability to accurately detect emotions using facial recognition technology will be technically feasible with current AI advancements.

3) **Economic Feasibility:** Economically, the development and maintenance costs will be evaluated against the potential market size, ensuring justifiable investments aligned with anticipated user adoption and revenue streams.

4) **Legal and Ethical Feasibility:** Legally and ethically, the system will comply with data protection laws and regulations, ensuring user consent and data handling conformity to international standards.

5) **Sustainability Feasibility:** Sustainability involves adapting to evolving trends and technological advances, ensuring flexibility and scalability for updates and improvements to meet changing user needs and technology standards. 3.System Features and Requirements

3.System Features and Requirements

3.1 Functional Requirements

3.1.1 Facial Expression Recognition: The system will need to accurately recognize facial expressions using a webcam. It should detect emotions like fear, surprise, anger, happiness, sadness, disgust, and neutrality.

3.1.2 Music Recommendation: Based on the detected emotion, the system will recommend a YouTube playlist that corresponds to the user's current emotional state to help modify or enhance the user's mood.

3.1.3 Real-time Processing: The system must process data in real-time, capturing facial expressions via webcam and immediately providing music recommendations**.**

3.1.4 User Interaction: The system should have a user-friendly interface that will allow users to interact with the system seamlessly, including starting the webcam, viewing detected emotions, and accessing recommended music playlists.

3.1.5 Data Handling: The system should manage and store facial expression data efficiently, allowing for quick retrieval and analysis.

3.2 Non-Functional Requirements

3.2.1 Performance:

The system should recognize facial expressions and provide music recommendations quickly and accurately, with a high success rate in real-time.

3.2.2 Usability:

The interface will need to be intuitive and easy to use for a wide range of users, requiring minimal technical knowledge.

3.2.3 Scalability:

The system should be scalable and capable of handling an increased load, such as multiple users simultaneously without degradation in performance.

3.2.4 Reliability:

The system should operate reliably under various conditions and should correctly identify facial expressions with a high degree of accuracy.

3.2.5 Security:

User data, including images and interaction logs, will need to be handled securely to protect privacy.

3.2.6 Maintainability:

The system should be easy to maintain and update, with clear documentation and modular design to facilitate enhancements and bug fixes.

4. System Requirements

4.1 Hardware Requirements

- Processor: Intel I3 processor
- Storage Space: 500 GB.
- Screen size: 15” LED Devices Required: Web camera, Mouse, and Keyboard
- Minimum RAM: 4GB and a good Internet connection.

4.2 Software Requirements

OS: Windows 7 and above /Ubuntu Programming
Language: Python
Software: JetBrains PyCharm Community Edition 2017.1.4 x64 Backend: Keras
Additional requirements: TensorFlow

Table of Contents – Section B

Section B: Design 12
1. System Design 14
1.1 Purpose 14
1.2 Description of Program 14
1.2.1 Workflow Diagram 14
1.2.2 Use Case Diagram 16
1.2.3 Class Diagram 18
1.2.4 Activity Diagram 20
1.2.5 Sequence Diagram 22
1.3 High-Level Design 23
1.3.1 Components 23
1.3.2 Architecture 24
1.3.3 Technology 37
1.4. Detailed Design 39
1.4.1 Purpose 39
1.4.2 Interface Design 39
1.5 System Code 42
1.6 Output 43

1. System Design

The system design will focus on developing an application that recommends music based on the user's emotional state. This will be achieved by analyzing the user's facial expressions through a camera input. The system will utilize a combination of facial recognition algorithms and emotion recognition models to determine the user's current mood and then suggest music that matches this mood.

1.1 Purpose

The primary purpose of this system will be to enhance the listening experience by dynamically selecting music that corresponds to the user's emotional state. It will aim to provide a personalized music experience that can help in mood enhancement, relaxation, or motivation depending on the detected emotion.

1.2 Description of Program

1.2.1 Workflow Diagram

The workflow will begin with the user interacting with the application through a graphical user interface. The user's face will be captured via a camera, and the image data will be processed using facial recognition technology to identify emotional expressions. The identified emotion will then be used to query a music database that will fetch a list of songs corresponding to the emotion detected. The system will play back the selected music, and the user will have the option to provide feedback which will be used to refine future music recommendations.

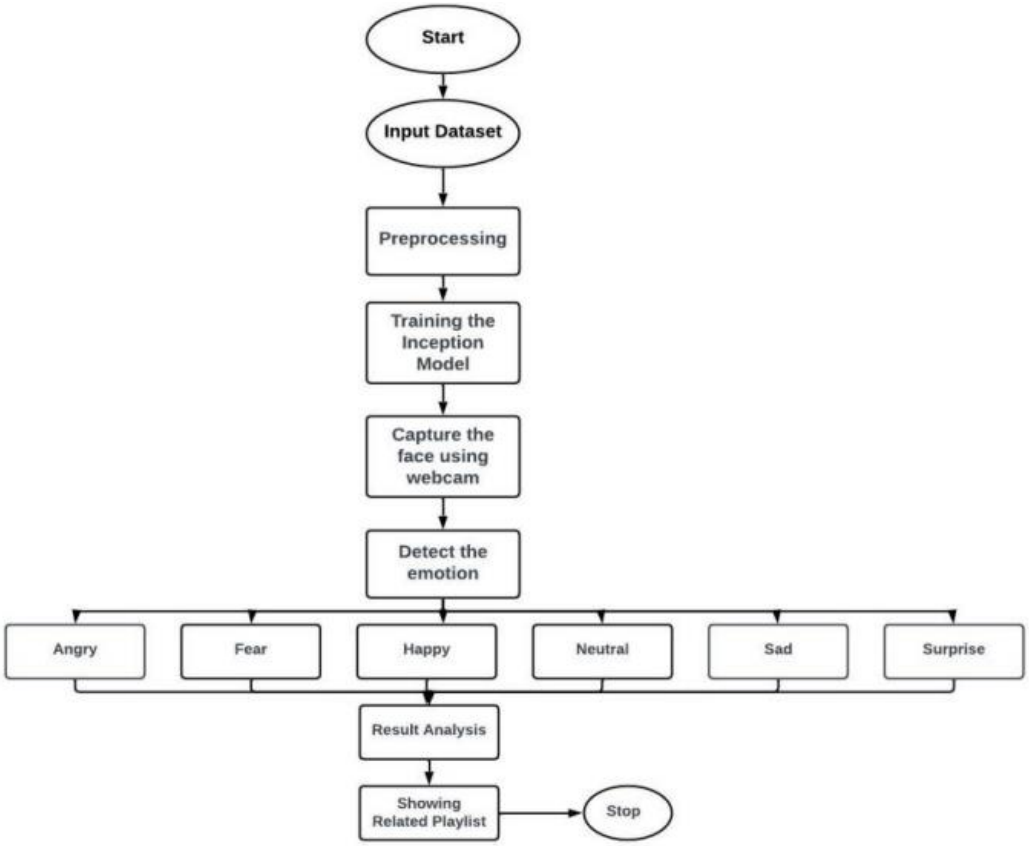


Figure 1.Workflow Diagram

Figure 2 shows the flowchart for a facial expression detection system. The process starts with loading or capturing the image and then detecting the eyes and lips of the image. Next, the system compares the image with a database to identify the emotion. If no emotion is detected, the process loops back to the detection of eyes and lips. If an emotion is detected, it sends this information to the music selection system to trigger the music player, resulting in songs being played. The flowchart is clearly outlined and organized in a logical sequence.

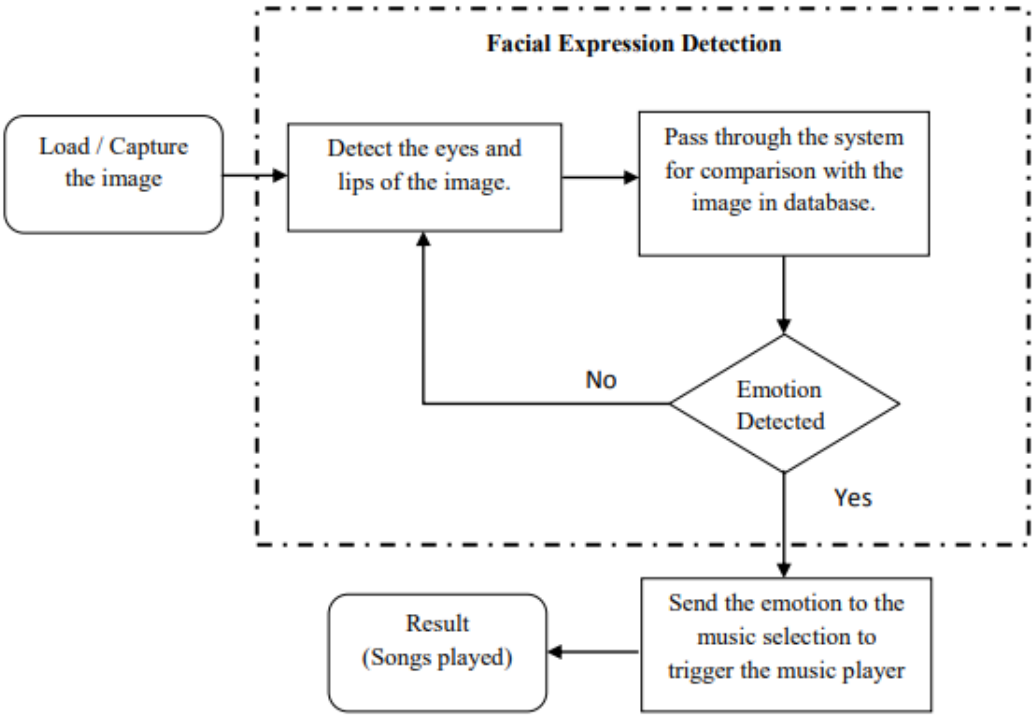


Figure 2.Workflow Diagram for Emotion Detection

1.2.2 Use Case Diagram

- **User:** This will be the primary actor who will interact directly with the system. The User's actions will initiate the use cases and will benefit from the outcomes. For instance, the User will be able to Capture Image, Approve Mood, and interact with the Playlist through actions like Play/Pause, Stop, and selecting the Previous Song. The User will also be able to manage the playlist by Adding/Removing Songs, Editing Song Details, and Creating Playlists.
- **The System:** Often represented in use case diagrams as a collection of processes rather than a human actor, the System will be an actor that will perform background actions or provide services that support the user's goals. In this diagram, the System will include automated processes like Accessing the Camera, Face Detection, Pre-processing Images, and Retrieving Mood based on the captured image. It will also include Database Access for managing user data and music-related information.

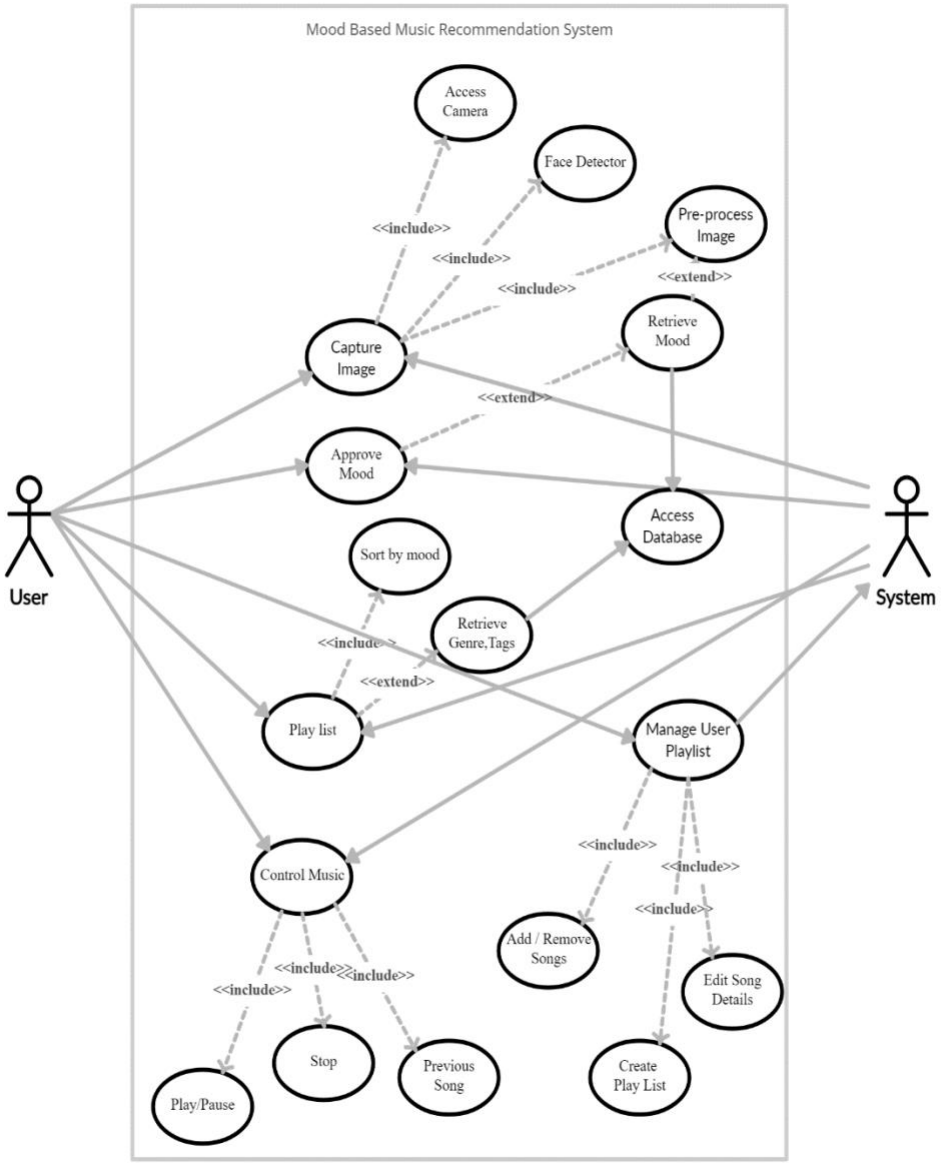


Figure 3.Use Case Diagram for Mood-Based music Recommendation System

1.2.3 Class Diagram

The presented class diagram, depicted in Figure 6, outlines the framework of a Mood-Based Music Recommendation system.

- **UserInterface:** This class represents the user interaction interface of the application. It includes a method interact() that facilitates user interaction with the system.
- **Camera:** This class is responsible for capturing the user's image. It has a method captureImage() which captures and sends the image data to the Facial Recognition system.
- **FacialRecognition:** This class processes the captured image data to identify the user's emotional expression. It has a method identifyEmotion(ImageData) : String that analyzes the image and returns the detected emotion as a string.
- **MusicDatabase:** This class manages the music content. It has a method fetchSongs(String emotion) : List that queries the database for songs matching the detected emotion and returns a list of songs.
- **MusicPlayer:** Responsible for playing music, this class has methods playMusic(List songList) for playing the fetched songs and

recommendSongs(String emotion) for recommending songs based on the detected emotion.

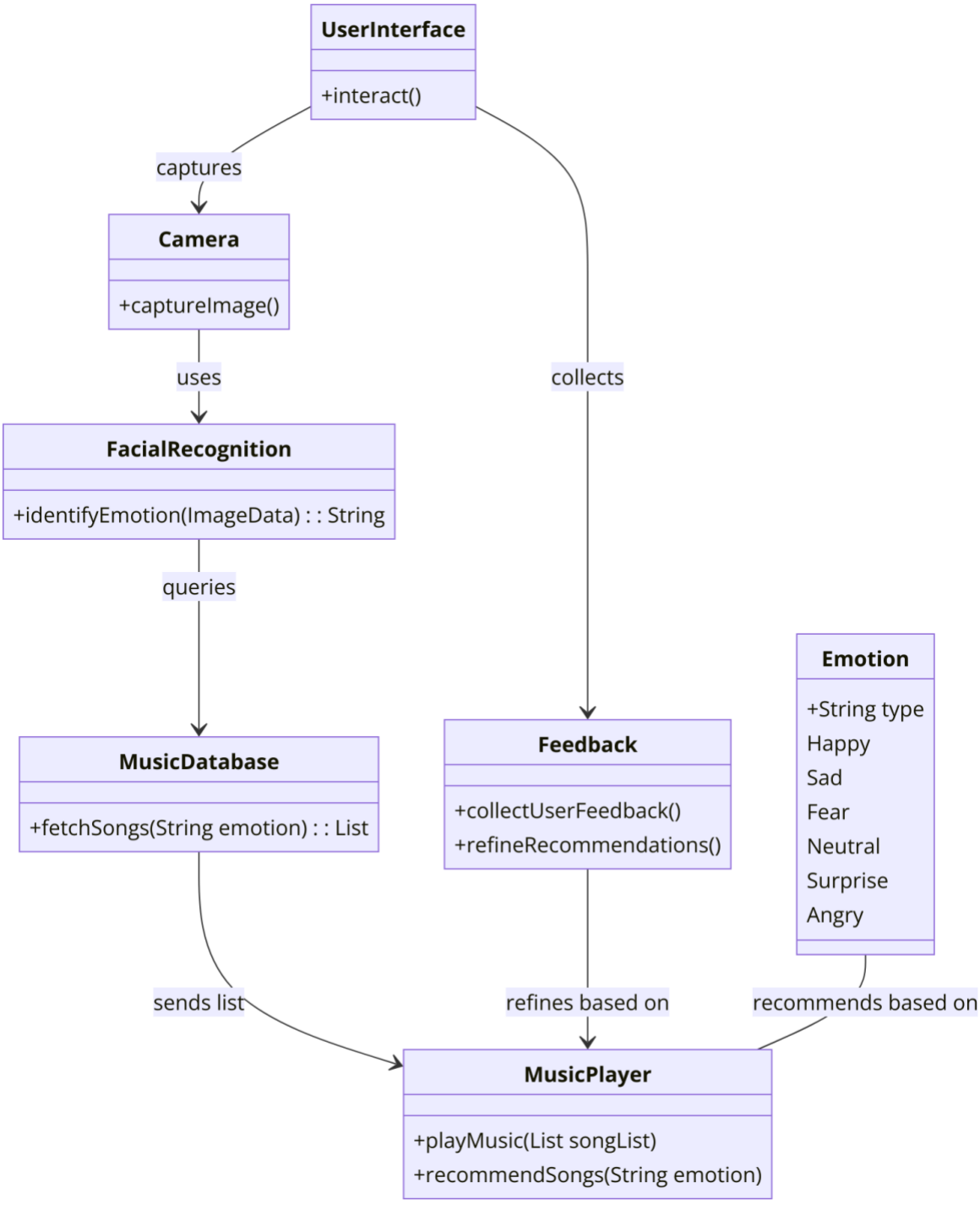


Figure 4.Class Diagram for SecureFace Connect

1.2.4 Activity Diagram

Figure 3 depicts a flowchart describing the process of song selection on a computer. It begins with the user turning on the computer, followed by opening the song folders. The user must decide whether to choose specific songs or not. If they do not choose songs, they select all songs in the folder(s). If they choose to select songs, they browse through the songs one by one and make their selection(s). The process loops until the user is done with song selection. Once the selection is complete, the user opens the selected song(s) with a music player.

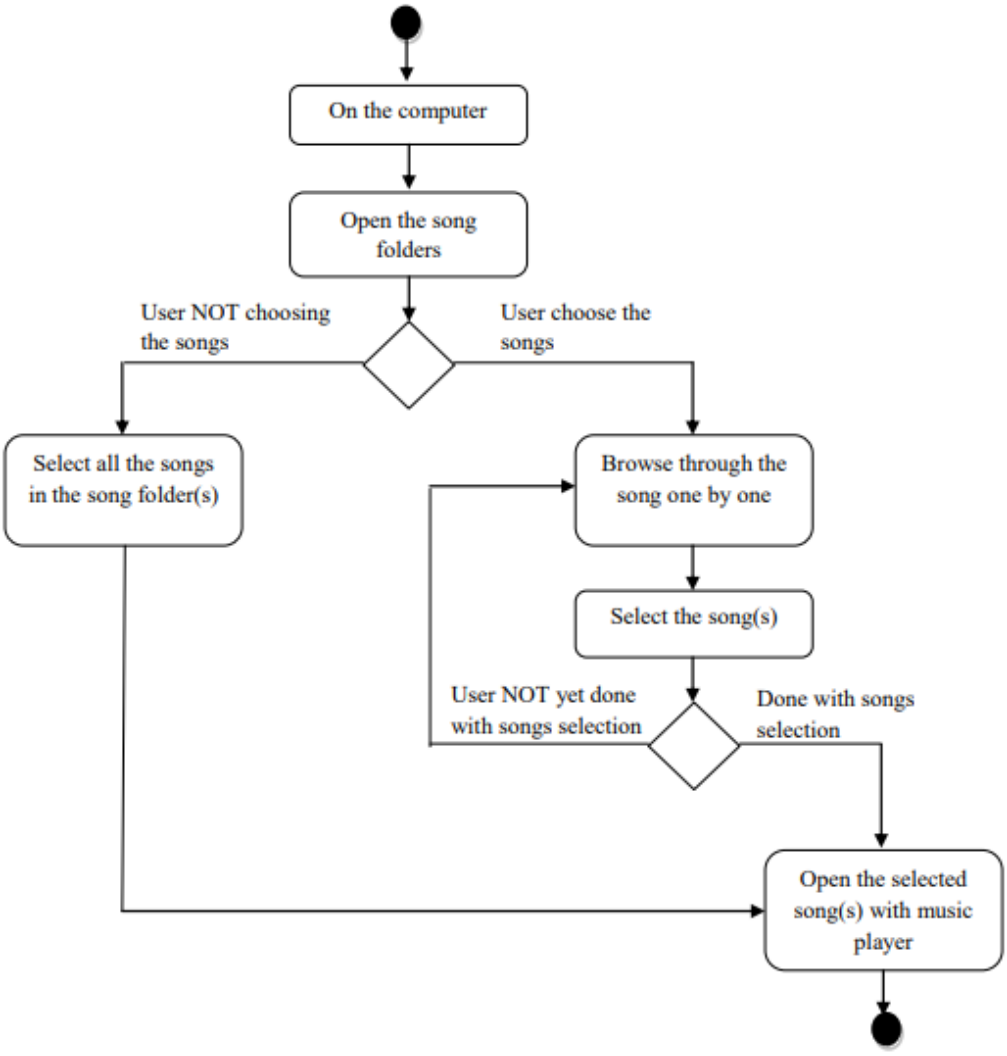


Figure 5. Activity Diagram for Mood-Based Music PlayerSystem

Figure 4 outlines the operation of an emotion-based music player, divided into two main sequences. On the left side, it begins with launching the player and prompts the user to open music. It then allows the user to customize songs into different emotion-based categories if desired, concluding with saving changes. On the right side, it starts with loading or capturing an image, waits for facial feature detection, and allows the user to listen to music played by the player. The system checks if the user wants to change emotions; if not, it continues playing music, otherwise, the user can close the system. The flowchart incorporates decision points where users can customize music or change emotions, influencing the flow of operations.

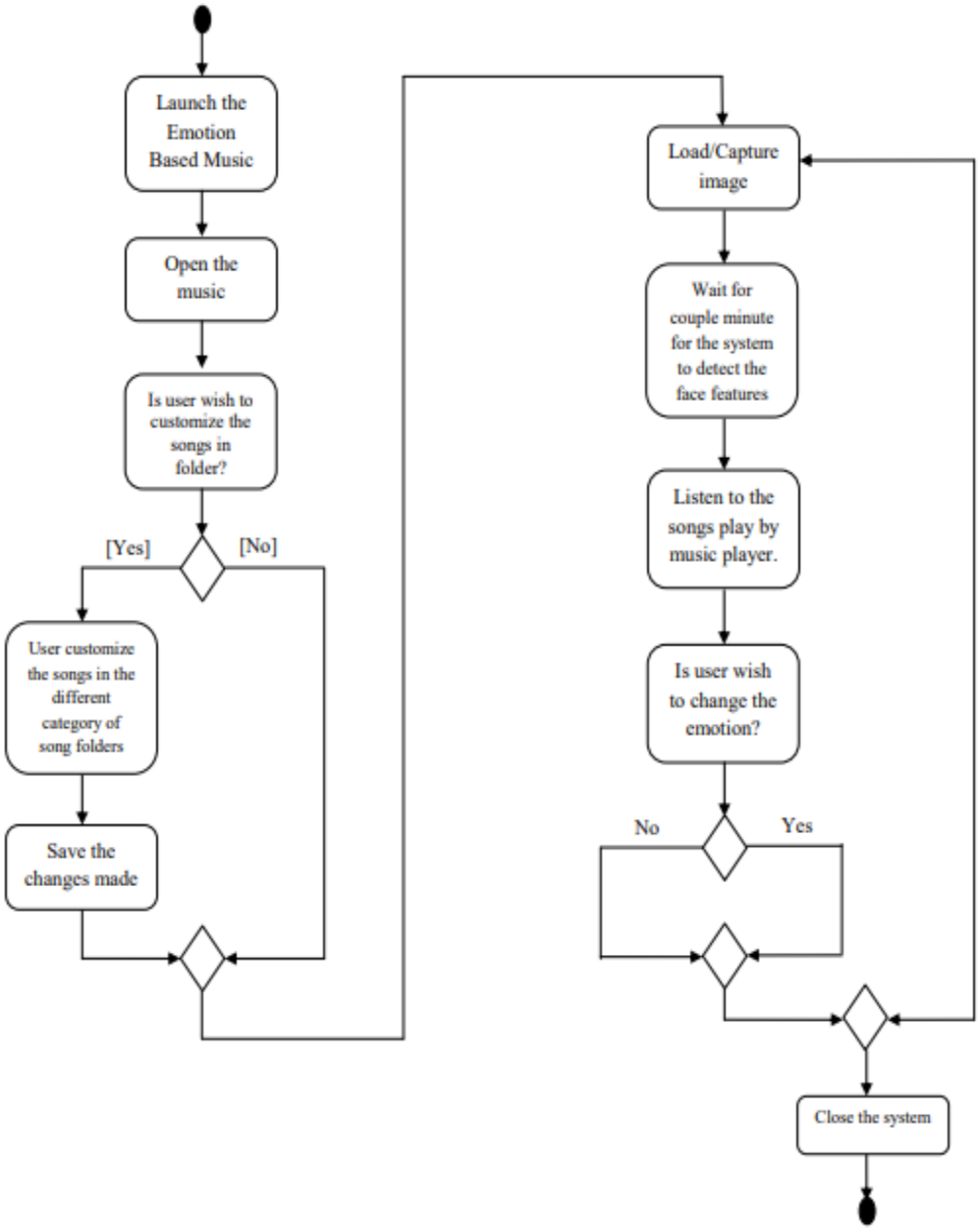


Figure 6. Activity Diagram for ***Emotion-Based Music Player Workflow.***

1.2.5 Sequence Diagram

The sequence diagram in Figure 6 will outline the workflow of a mood-based music recommendation application, starting with the user opening the application, which will prompt the device to capture a photo. Subsequently, facial detection will be conducted to recognize the user's mood, and this information will be transmitted to the device from the database. Upon receiving the mood assessment, the user will confirm its accuracy, prompting the database to retrieve music suited to the identified mood. Once the playlist is generated, it will be sent back to the device for playback. Throughout this process, the user will retain control over the music playback, with options to play, pause, or skip tracks as

desired. Finally, when finished, the user will close the application, concluding the interaction. This sequential and user-centric approach will ensure a smooth and personalized experience for the user, from initiating the app to enjoying tailored music recommendations and managing playback, ultimately leading to the app's closure.

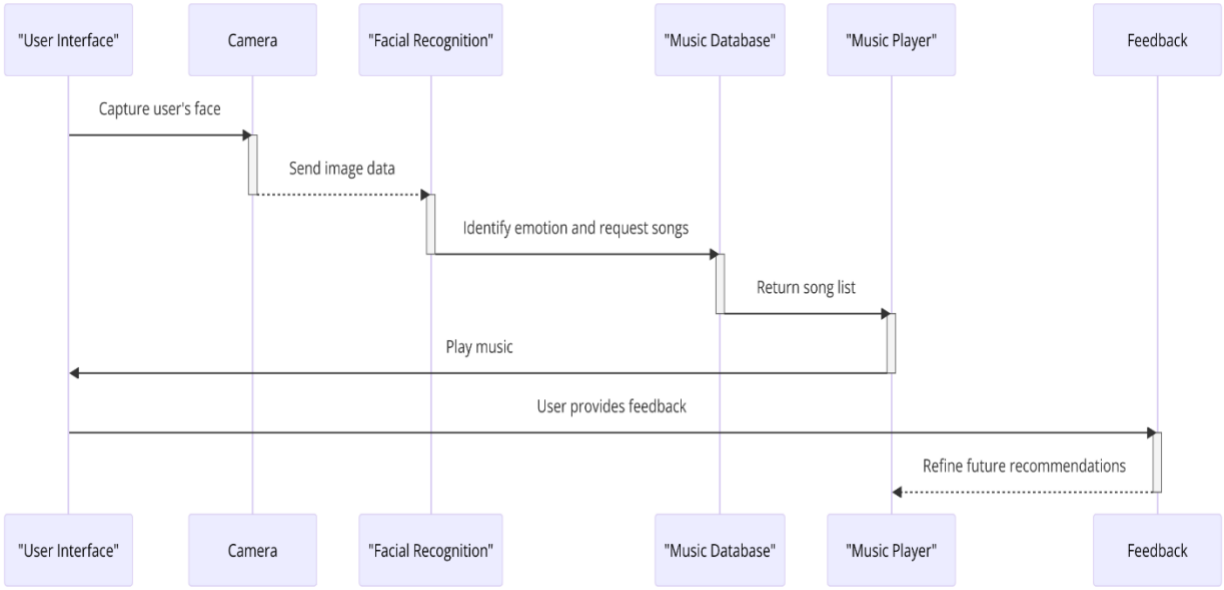


Figure 7 Sequence Diagram

1.3 High-Level Design

1.3.1 Components

The System will include the following components:

Modules which need to be installed are:

- flask
- json
- base64
- OpenCV
- keras.models

- keras.utils.np_utils
- pandas
- numpy
- random
- keras.models
- keras.preprocessing.image
- cv2 (OpenCV library)
- flask

1.3.2 Architecture

There are three main blocks of implementation:

1) **Dataset**

   The dataset used in developing this system is `fer2013.csv`, commonly referred to as the Facial Emotion Recognition 2013 (FER-2013) dataset. This dataset is widely utilized for training machine learning models to recognize human emotions from facial expressions. It is structured in a CSV format, containing columns for emotion labels (represented as integers for emotions like Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral), pixel data (a space-separated string of grayscale pixel values), and usage type (indicating subsets like Training, Public Test, or PrivateTest). The code processes this data by filtering specific usage types, converting pixel strings into 48x48 numpy arrays, and normalizing them for neural network input. Additionally, the 'Disgust' emotion is reclassified as 'Angry', possibly to address class imbalance or focus the model training on a subset of emotions. This dataset is essential for projects in emotion classification using computer vision techniques, providing a robust foundation for developing and testing emotion recognition models.

2) **User Interface**

   Index page:

```
<!-- <!DOCTYPE html>
<html>

<head>
    <title>Live Video Capture</title>
</head>

<body>
    <h1>Live Video Capture</h1>
    <div>
```

```
        <video id="video" width="640" height="480" autoplay></video>
    </div>
    <div>
        <button id="capture-btn">Capture</button>
    </div>
    <script>
        // get video element
        const video = document.getElementById('video');

        // request camera access and set video source
        navigator.mediaDevices.getUserMedia({ video: true })
            .then(stream => {
                video.srcObject = stream;
            })
            .catch(err => {
                console.error('Error accessing camera:', err);
            });

        // add event listener to capture button
        const captureBtn = document.getElementById('capture-btn');
        captureBtn.addEventListener('click', () => {
            // get canvas element and draw video frame on it
            const canvas = document.createElement('canvas');
            canvas.width = video.videoWidth;
            canvas.height = video.videoHeight;
            const context = canvas.getContext('2d');
            context.drawImage(video, 0, 0, canvas.width, canvas.height);

            // convert canvas to data URL
            const dataURL = canvas.toDataURL('image/png');

            // send data URL to Flask route
            fetch('/process-image', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ image_data: dataURL }),
                redirect:"follow"
            })
                .then(response => {
                    console.log('Image processed successfully:', response);
                    window.location.href=response.url;
                })
                .catch(err => {
```

```
                console.error('Error processing image:', err);
            });
    });
        </script>
 </body>

</html> -->



<!DOCTYPE html>
<html>

<head>
    <title>Live Video Capture</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f7f7f7;
            margin: 0;
            padding: 0;
        }

        h1 {
            text-align: center;
            margin-top: 50px;
            margin-bottom: 30px;
        }

        #video {
            display: block;
            margin: 0 auto;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
        }

        #capture-btn {
            display: block;
            margin: 30px auto;
            padding: 10px 20px;
            font-size: 16px;
            font-weight: bold;
            color: #fff;
            background-color: #4caf50;
            border: none;
```

```
            border-radius: 4px;
            box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);
            cursor: pointer;
            transition: background-color 0.3s;
        }

        #capture-btn:hover {
            background-color: #388e3c;
        }
    </style>
</head>

<body>
    <h1>Live Video Capture</h1>
    <div>
        <video id="video" width="640" height="480" autoplay></video>
    </div>
    <div>
        <button id="capture-btn">Capture</button>
    </div>
    <script>
        // get video element
        const video = document.getElementById("video");

        // request camera access and set video source
        navigator.mediaDevices
            .getUserMedia({ video: true })
            .then((stream) => {
                video.srcObject = stream;
            })
            .catch((err) => {
                console.error("Error accessing camera:", err);
            });

        // add event listener to capture button
        const captureBtn = document.getElementById("capture-btn");
        captureBtn.addEventListener("click", () => {
            // get canvas element and draw video frame on it
            const canvas = document.createElement("canvas");
            canvas.width = video.videoWidth;
            canvas.height = video.videoHeight;
            const context = canvas.getContext("2d");
            context.drawImage(video, 0, 0, canvas.width, canvas.height);

            // convert canvas to data URL
```

```
                const dataURL = canvas.toDataURL("image/png");

                // send data URL to Flask route
                fetch("/process-image", {
                    method: "POST",
                    headers: {
                        "Content-Type": "application/json",
                    },
                    body: JSON.stringify({ image_data: dataURL }),
                    redirect: "follow",
                })
                    .then((response) => {
                        console.log("Image processed successfully:", response);
                        window.location.href = response.url;
                    })
                    .catch((err) => {
                        console.error("Error processing image:", err);
                    });
            });
        </script>
    </body>

    </html>
```

Figure 3.Indexpage

Overall, this code creates a simple web interface for capturing live video and processing the captured frames using Flask on the server side. It can be extended to include additional features such as image filtering, real-time analysis, or saving captured images to a database.

Music player Interface:

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>Music Player</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
        integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
```

```
        <style>
            body {
                background-size: cover;
                background-image: url('../static/bg.jpg');
    background-color: #f5f5f5;
    font-family: Arial, sans-serif;
}

.container {
    background-color:transparent;
    border-radius: 5px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.3);
    padding: 20px;
}

h1 {
    font-size: 2.5rem;
    color: #f5f5f5;
    font-weight: bold;
    font-family: 'Courier New', Courier, monospace;
}

audio {
    margin-bottom: 2rem;
}

.btn {
    font-size: 1.2rem;
    padding: 0.6rem 1.5rem;
}

#play-btn {
    background-color: #007bff;
    border-color: #007bff;
}

#pause-btn {
    background-color: #ffc107;
    border-color: #ffc107;
}

#stop-btn {
    background-color: #dc3545;
    border-color: #dc3545;
}
```

```
#play-btn:hover,
#pause-btn:hover,
#stop-btn:hover {
    opacity: 0.8;
    cursor: pointer;
}

        </style>
</head>

<body>
    <div class="container mt-5">
        <div class="row justify-content-center">
            <div class="col-md-6">
                <h1 class="text-center mb-4">Music Player</h1>
                <audio id="audio-player" controls class="w-100 mb-4">
                    <source src="{{ url_for('static',
filename=prediction+'/'+song) }}" type="audio/mpeg">
                </audio>
                <div class="text-center">
                    <button id="play-btn" class="btn btn-primary mr-
3">Play</button>
                    <button id="pause-btn" class="btn btn-primary mr-
3">Pause</button>
                    <button id="stop-btn" class="btn btn-primary">Stop</button>
                </div>
                </div>
                </div>
                </div>
                <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
                    integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
                    crossorigin="anonymous"></script>
                <script>
                    // get audio player and control buttons
                    const audioPlayer = document.getElementById('audio-player');
                    const playBtn = document.getElementById('play-btn');
                    const pauseBtn = document.getElementById('pause-btn');
                    const stopBtn = document.getElementById('stop-btn');

                    // add event listeners to control buttons
                    playBtn.addEventListener('click', () => {
                        audioPlayer.play();
                    });
```

```
        pauseBtn.addEventListener('click', () => {
            audioPlayer.pause();
        });

        stopBtn.addEventListener('click', () => {
            audioPlayer.pause();
            audioPlayer.currentTime = 0;
        });
    </script>
     </body>

    </html>
```

Figure 4.Music Player Code

This code provides a basic interface for playing audio files with controls for play, pause, and stop. The audio file source is expected to be provided dynamically using Flask's url_for() function. You may need to adjust the Flask part of the code to ensure the correct URL for the audio file is generated.

3) Face Detection and Recognition

Emotion Detection:

Overall, the below code demonstrates the process of loading facial expression data, preprocessing it, defining and training a CNN model for emotion recognition, and saving the trained model for future use. It's a comprehensive approach to building an emotion recognition system based on facial expressions.

```
from keras.utils.np_utils import to_categorical
import pandas as pd
import numpy as np
import random

def emotion_count(y_train, classes):
    """
    The function re-classify picture with disgust label into angry label
    """
```

```
    emo_classcount = {}
    print('Disgust classified as Angry')
    y_train.loc[y_train == 1] = 0
    classes.remove('Disgust')
    for new_num, _class in enumerate(classes):
        y_train.loc[(y_train == emotion[_class])] = new_num
        class_count = sum(y_train == (new_num))
        emo_classcount[_class] = (new_num, class_count)
    return y_train.values, emo_classcount


def load_data(sample_split=0.3, usage='Training', classes=['Angry', 'Happy'],
filepath='./data/fer2013.csv'):
    """
    The function load provided CSV dataset and further reshape, rescale the data
for feeding
    """
    df = pd.read_csv(filepath)
    df = df[df.Usage == usage]
    frames = []
    classes.append('Disgust')
    for _class in classes:
        class_df = df[df['emotion'] == emotion[_class]]
        frames.append(class_df)
    data = pd.concat(frames, axis=0)
    rows = random.sample(list(data.index), int(len(data) * sample_split))
    data = data.loc[rows]
    x = list(data["pixels"])
    X = []
    for i in range(len(x)):
        each_pixel = [int(num) for num in x[i].split()]
        X.append(each_pixel)
    ## reshape into 48*48*1 and rescale
    X = np.array(X)
    X = X.reshape(X.shape[0], 48, 48, 1)
    X = X.astype("float32")
    X /= 255

    y_train, new_dict = emotion_count(data.emotion, classes)
    y_train = to_categorical(y_train)
    return X, y_train

## All three datasets are well loaded accordingly
emotion = {'Angry': 0, 'Disgust': 1, 'Fear': 2, 'Happy': 3,
           'Sad': 4, 'Surprise': 5, 'Neutral': 6}
```

```
emo     = ['Angry', 'Fear', 'Happy',
           'Sad', 'Surprise', 'Neutral']

X_test, y_test = load_data(sample_split=1.0,classes=emo,
usage='PrivateTest')

X_train, y_train = load_data(sample_split=1.0,classes=emo,
usage= 'Training')

X_val,y_val = load_data(sample_split=1.0,classes=emo,
usage= 'PublicTest')

def save_data(X_test, y_test, fname=''):
    """
    The function stores loaded data into numpy form for further processing
    """
    np.save( 'X_test' + fname, X_test)
    np.save( 'y_test' + fname, y_test)
save_data(X_test, y_test,"_privatetest6_100pct")
X_fname = 'X_test_privatetest6_100pct.npy'
y_fname = 'y_test_privatetest6_100pct.npy'
X = np.load(X_fname)
y = np.load(y_fname)
print ('Private test set')
y_labels = [np.argmax(lst) for lst in y]
counts = np.bincount(y_labels)
labels = ['angry', 'fear', 'happy', 'sad', 'surprise', 'neutral']
print (zip(labels, counts))



# Final Model Architecture:
from keras import layers
from keras import models

modelN = models.Sequential()
modelN.add(layers.Conv2D(32, (3, 3), padding='same', activation='relu',
                         input_shape=(48, 48, 1)))
modelN.add(layers.Conv2D(32, (3, 3), padding='same', activation='relu'))
modelN.add(layers.Conv2D(32, (3, 3), padding='same', activation='relu'))
modelN.add(layers.MaxPooling2D(pool_size=(2, 2)))

modelN.add(layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
modelN.add(layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
modelN.add(layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
```

```
modelN.add(layers.MaxPooling2D(pool_size=(2, 2)))

modelN.add(layers.Conv2D(128, (3, 3), padding='same', activation='relu'))
modelN.add(layers.Conv2D(128, (3, 3), padding='same', activation='relu'))
modelN.add(layers.Conv2D(128, (3, 3), padding='same', activation='relu'))
modelN.add(layers.MaxPooling2D(pool_size=(2, 2)))

modelN.add(layers.Flatten())  # this converts our 3D feature maps to 1D feature
vectors
modelN.add(layers.Dense(64, activation='relu'))
modelN.add(layers.Dense(64, activation='relu'))
modelN.add(layers.Dense(6, activation='softmax'))

# optimizer:
modelN.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
print ('Training....')


#fit
nb_epoch = 32
batch_size = 128

model = modelN.fit(X_train, y_train, epochs=nb_epoch, batch_size=batch_size,
          validation_data=(X_val, y_val), shuffle=True, verbose=1)


model.save("model.h5")#saves model configuration as h5 file
```

Figure 5.Model Training

Below Figure 20, evaluates the performance of the trained CNN model on the test set by calculating its accuracy. It provides insights into how well the model generalizes to unseen data.

Testing :

```
from keras.models import load_model
from sklearn.metrics import accuracy_score
import numpy as np
```

```
model = load_model('model.h5')
X_fname = 'X_test_privatetest6_100pct.npy'
y_fname = 'y_test_privatetest6_100pct.npy'
X = np.load(X_fname)
y = np.load(y_fname)
print ('Private test set')
y_labels = [np.argmax(lst) for lst in y]
counts = np.bincount(y_labels)
labels = ['angry', 'fear', 'happy', 'sad', 'surprise', 'neutral']
print (zip(labels, counts))

test_pred = np.argmax(model.predict(X), axis=1)
print("CNN Model Accuracy on test set: {:.4f}".format(accuracy_score(y_labels,
test_pred)))
# print(test_pred[:5])
```

Figure 6.Testing

Model Loading:

This Flask application integrates image processing, facial emotion recognition, and music playback functionalities. Upon loading, the main page (`index.html`) is displayed. When an image is uploaded through the web interface, the application decodes it, detects faces using the Haar cascade classifier, and predicts the emotion expressed in each face using a pre-trained convolutional neural network (CNN) model. The predicted emotion is then used to redirect to a page (`songs_list.html`) displaying a list of songs associated with that emotion. Users can select a song, which triggers the `music player` route, rendering a music player interface (`music_player.html`) to play the selected song. If any error occurs during this process, users are directed to an error page (`error_page.html`). This application offers an interactive and engaging experience, combining image processing, machine learning, and multimedia functionalities to provide personalized music recommendations based on detected emotions.

```
from keras.models import load_model
import base64
import io
import os
import cv2
import json
import numpy as np
from flask import Flask, render_template, request, jsonify, url_for,
redirect,session

app = Flask(__name__)
```

```
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 1
haarcascade = "haarcascade_frontalface_default.xml"
label_map = ['Anger', 'Neutral', 'Fear', 'Happy', 'Sad', 'Surprise']
print("+"*50, "loading model")
model = load_model('model.h5')
cascade = cv2.CascadeClassifier(haarcascade)


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/process-image', methods=['POST'])
def process_image():
    # get data URL from request body
    data = json.loads(request.data)
    dataURL = data.get('image_data')
    image_data = dataURL.split(',')[1]
    decoded_data = base64.b64decode(image_data)
    np_data = np.frombuffer(decoded_data, np.uint8)
    img = cv2.imdecode(np_data, cv2.IMREAD_COLOR)

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = cascade.detectMultiScale(gray, 1.4, 1)
    for x, y, w, h in faces:
        roi = gray[y:y+h, x:x+w]
    try:
        roi = cv2.resize(roi, (48, 48))
        roi = roi/255.0
        roi = np.reshape(roi, (1, 48, 48, 1))
        prediction = model.predict(roi)
        prediction = np.argmax(prediction)
        prediction = label_map[prediction]
        print("prediction is",prediction)
        print("process-image")
        return redirect(url_for('songs_list', prediction=prediction))
    except:
        return redirect(url_for('error'))

@app.route('/songs_list/<prediction>')
def songs_list(prediction):
    f_path='static/'+prediction
    folder_contents = os.listdir(f_path)
```

```
    return
render_template('songs_list.html',folder_contents=folder_contents,prediction=pred
iction)

@app.route('/music-player')
def music_player():
    prediction=request.args.get('prediction')
    song=request.args.get('song')
    return render_template('music_player.html', prediction=prediction,song=song)


@app.route('/error-page')
def error():
    return render_template('error_page.html')
```

Figure 7.Loading the model

1.3.3 Technology

Several modules are being used in developing this system. Here is the list of modules and their usage information:

1. OpenCV

OpenCV, or Open-Source Computer Vision Library, is a powerful open-source software library extensively used for various computer vision and machine learning tasks. With its vast array of functions and algorithms, OpenCV enables developers to perform real-time image and video processing tasks, including object detection, facial recognition, and gesture recognition. Its features encompass image processing capabilities such as resizing, cropping, and filtering, alongside sophisticated methods for object detection, like the Haar cascade classifier and deep learning-based models such as SSD and YOLO. OpenCV also offers tools for feature detection and description, supporting algorithms like SIFT and SURF. Additionally, it provides functionalities for machine learning tasks such as classification, clustering, and regression, complemented by support for video processing, camera calibration, and graphical user interface tools. With its extensive documentation and active community, OpenCV serves as an essential resource for developers seeking to implement advanced computer vision applications across diverse domains.

2. Keras:

Keras is a Python-based high-level neural networks API that simplifies deep learning model development and deployment. It integrates seamlessly with popular backend frameworks like TensorFlow and Theano, offering an intuitive interface for building, training, and deploying neural networks. With modular design, support for sequential and functional models, and extensibility for custom layers and functions, Keras enables rapid

prototyping and experimentation. It is widely used across platforms, supported by a vibrant community and extensive documentation, making it a go-to choice for researchers and developers in various deep learning applications.

3. Pandas:
Pandas is a Python library for data manipulation and analysis, offering powerful data structures like DataFrame and Series. It simplifies tasks such as reading/writing data from various sources, cleaning, transforming, and reshaping data, and performing descriptive statistics and visualization. With an intuitive syntax and extensive functionality, Pandas is widely used in data science for efficient tabular data handling.

4. os
The `os` module in Python facilitates interaction with the operating system, offering functions for file and directory operations, environment variables, and process management. It simplifies tasks like manipulating file paths, handling files and directories, accessing environment variables, and executing shell commands. Overall, `os` serves as a bridge between Python and the operating system, enabling seamless system-related tasks.

5. numpy:
NumPy is a fundamental Python library for numerical computing, offering support for multi-dimensional arrays and matrices along with a collection of mathematical functions to operate on these arrays efficiently. With NumPy, users can perform various operations such as mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation, and much more. It serves as a foundation for numerical and scientific computing in Python, providing the essential building blocks for tasks such as data analysis, machine learning, image processing, and computational science. NumPy's array operations are significantly faster than Python's built-in list operations, making it a popular choice for numerical computing tasks requiring high performance and efficiency.

6. random
The `random` module in Python is used for generating random numbers and performing random selections. It provides various functions for generating random numbers from different distributions, such as uniform, Gaussian, and exponential distributions. Additionally, the `random` module offers functions for shuffling sequences, selecting random elements, generating random samples with or without replacement, and seeding the random number generator for reproducibility. This module is widely used in tasks such as simulation, random sampling, cryptography, and generating test data. Overall, the `random` module is essential for introducing randomness into Python programs and applications.

7. flask
Flask is a lightweight and versatile web framework for Python, designed to make web development simple and scalable. It provides essential tools and libraries for building web applications, APIs, and microservices with minimal boilerplate code. Flask follows the WSGI (Web Server Gateway Interface) standard, making it compatible with various web

servers and deployment environments. Key features of Flask include routing, request handling, template rendering, and session management. It also offers support for extensions, allowing developers to add additional functionality, such as authentication, database integration, and RESTful API support. Flask's simplicity, flexibility, and extensive documentation make it a popular choice for both beginners and experienced developers in the Python web development community.

1.4. Detailed Design

1.4.1 Purpose

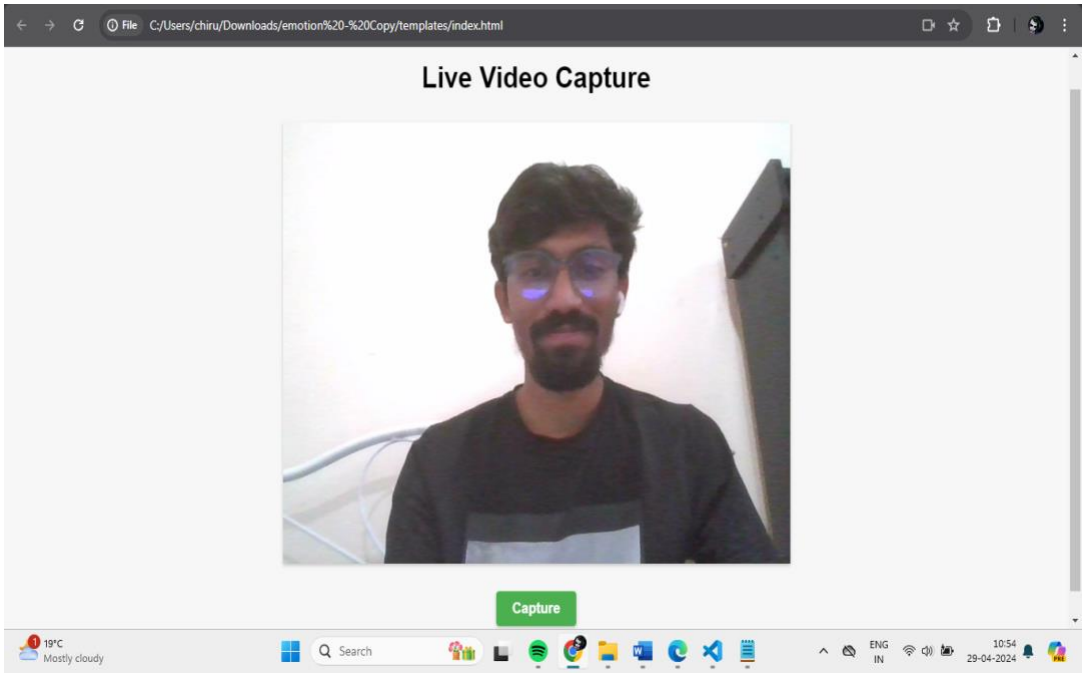
In detailed design the internal logic of each module is specified. Detailed design identifies the major modules and the major data flow among them. The method used to specify the system design typically focuses on the external interfaces of the modules and cannot be extended to specify the internal interfaces of the modules. The focus of verification in the detailed design phase is on showing that the detailed design meets the specification laid down in the system design. Below are the implementation details for each functionality of the SecureFace Connect system, used for conference registration and authentication.

1.4.2 Interface Design

There are four main interfaces in this project:

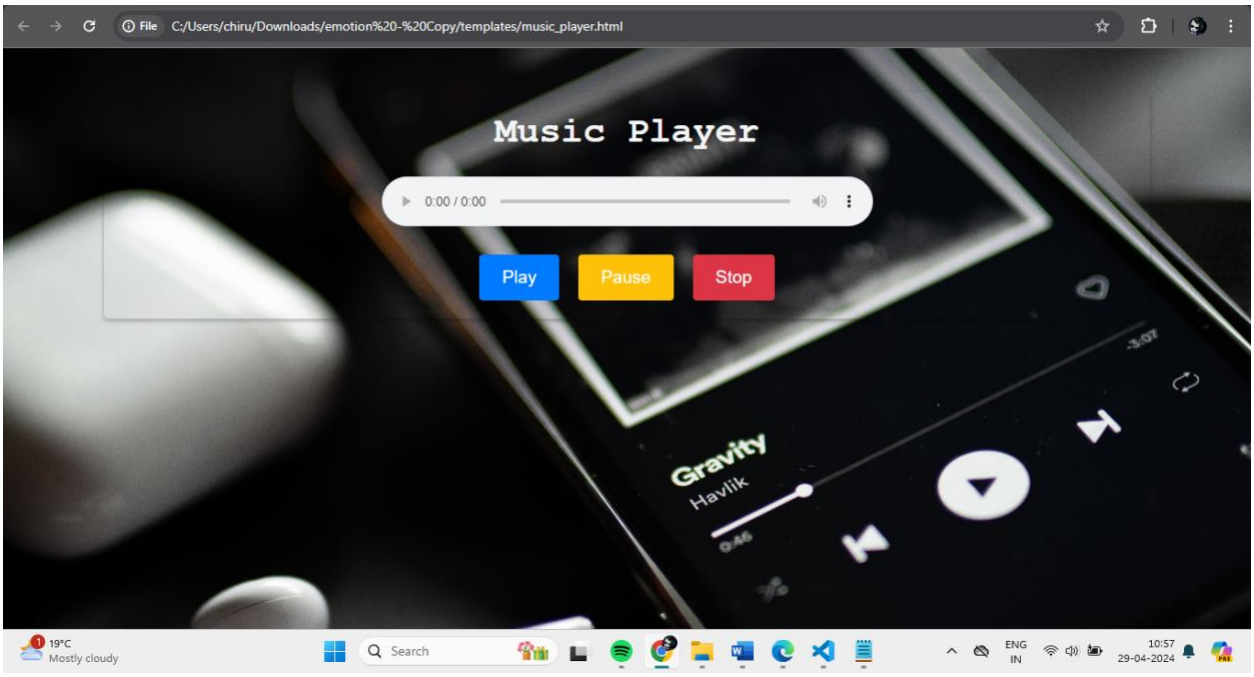
1) Home Page

This web page will serve as a platform for users to access and interact with their device's camera through a web browser. Its primary purpose will be to facilitate live video capture and snapshot functionality, enabling users to capture moments directly from their camera feed without the need for additional software. By embedding a video element and a capture button, the page will allow users to view the live video feed and trigger the capture of snapshots with a simple click. Behind the scenes, JavaScript will handle the camera access and snapshot capture functionality, while CSS styles will ensure a visually appealing and user-friendly interface. Overall, this page will provide a convenient and accessible solution for users to leverage their device's camera capabilities within a web environment, catering to a variety of potential applications such as video conferencing, live streaming, or multimedia content creation.



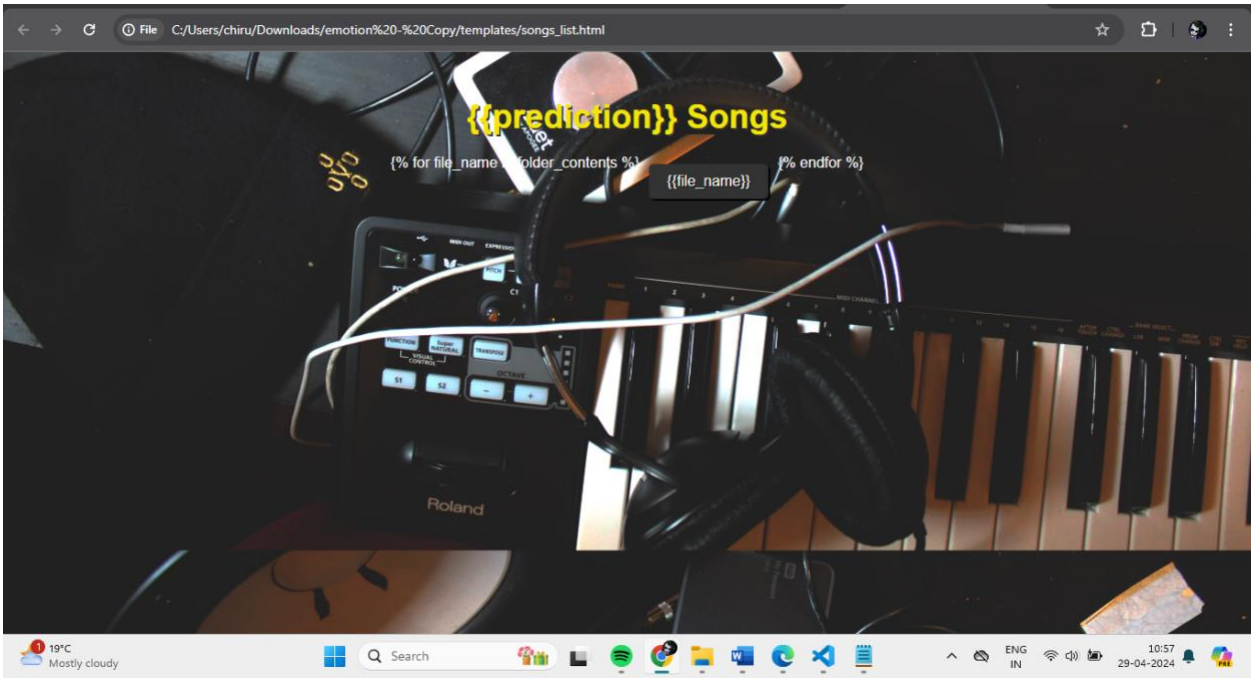
2) Music Player:

This webpage serves as a user interface for playing audio files. It presents a music player with basic playback controls, allowing users to play, pause, and stop audio playback. Users can interact with the player by clicking the corresponding control buttons displayed on the page. The audio player supports various audio formats and provides a seamless experience for listening to music or other audio content. Additionally, the page is visually appealing, with a clean layout and styling provided by the Bootstrap CSS framework, ensuring a consistent and attractive presentation across different devices. Overall, the webpage provides a simple yet effective solution for playing audio files directly within a web browser.



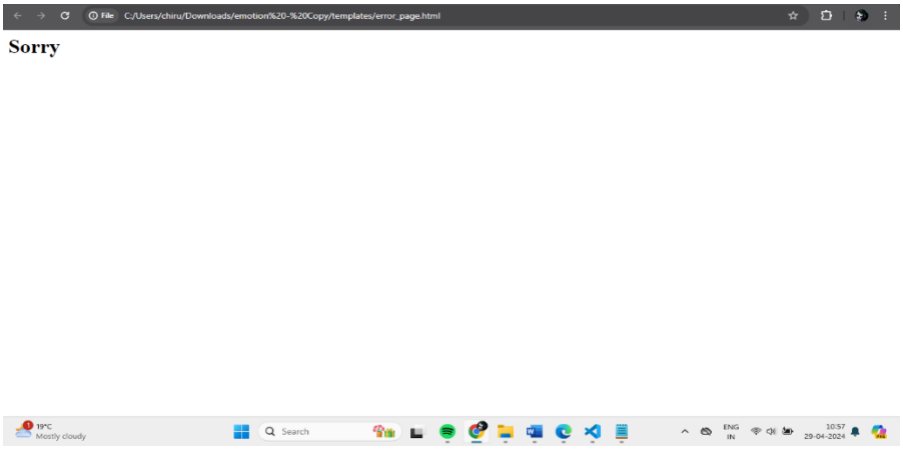
3) Playlist:

This HTML document creates a webpage titled "Song List" displaying songs based on a prediction. It features a background image, white text, and a list of songs presented as buttons. Each button triggers a JavaScript function to redirect users to a music player page with the selected song. The styling emphasizes dark backgrounds and hover effects for buttons, enhancing user experience. Overall, it offers a streamlined interface for users to browse and select songs based on predictions.



4) Error page

The error page will function as a user interface component to inform users about problems within the system. If the mood-based music recommendation system fails to process a user's emotion or encounters a technical issue, the error page will display messages explaining the error, suggest corrective actions, and provide links to other helpful pages. This helps maintain a positive user experience by managing system failures gracefully and keeping users engaged despite potential issues.



1.5 System Code

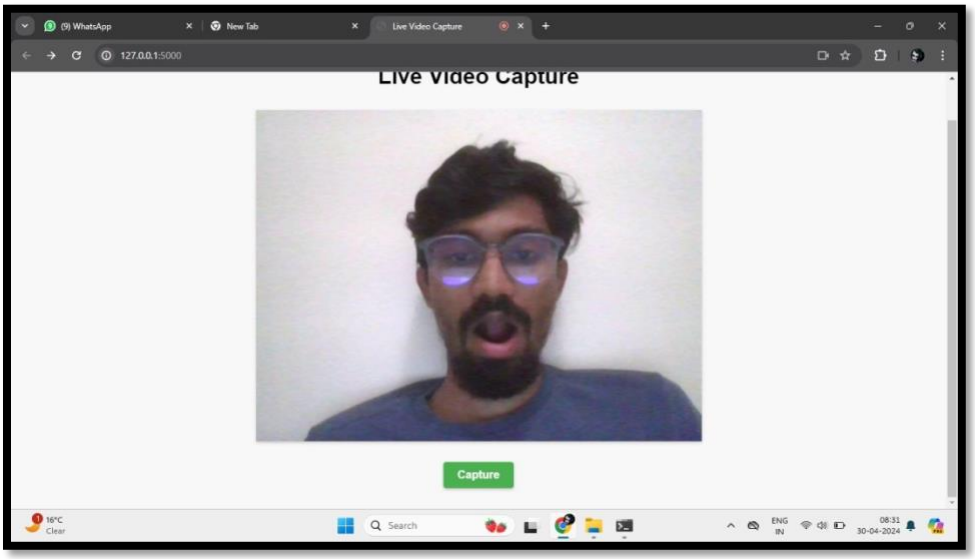
```
from keras.models import load_model
```

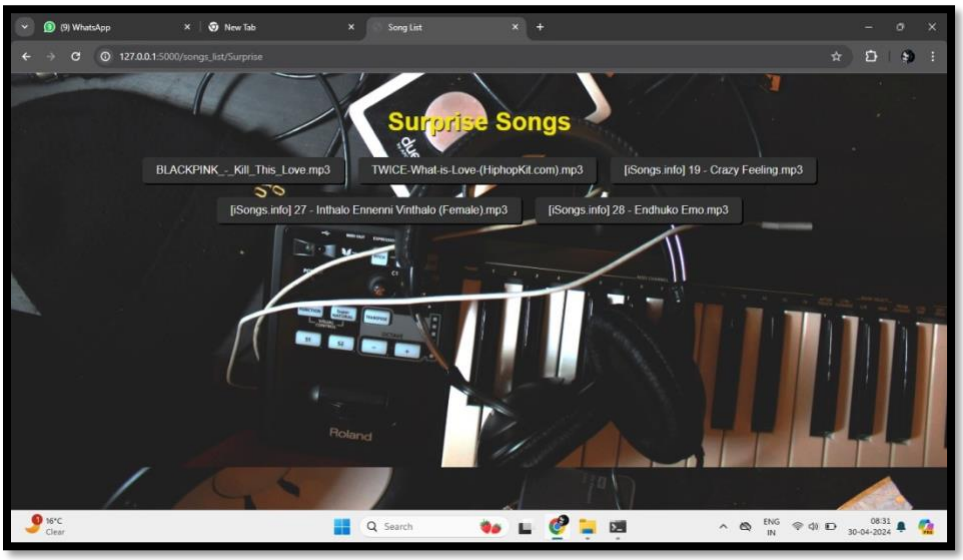
```
import base64
import io
import os
import cv2
import json
import numpy as np
from flask import Flask, render_template, request, jsonify, url_for,
redirect,session

app = Flask(__name__)
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 1
haarcascade = "haarcascade_frontalface_default.xml"
label_map = ['Anger', 'Neutral', 'Fear', 'Happy', 'Sad', 'Surprise']
print("+"*50, "loading model")
model = load_model('model.h5')
cascade = cv2.CascadeClassifier(haarcascade)


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/process-image', methods=['POST'])
def process_image():
    # get data URL from request body
    data = json.loads(request.data)
    dataURL = data.get('image_data')
    image_data = dataURL.split(',')[1]
    decoded_data = base64.b64decode(image_data)
    np_data = np.frombuffer(decoded_data, np.uint8)
    img = cv2.imdecode(np_data, cv2.IMREAD_COLOR)

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = cascade.detectMultiScale(gray, 1.4, 1)
    for x, y, w, h in faces:
        roi = gray[y:y+h, x:x+w]
    try:
        roi = cv2.resize(roi, (48, 48))
        roi = roi/255.0
        roi = np.reshape(roi, (1, 48, 48, 1))
        prediction = model.predict(roi)
        prediction = np.argmax(prediction)
        prediction = label_map[prediction]
        print("prediction is",prediction)
```

```
        print("process-image")
        return redirect(url_for('songs_list', prediction=prediction))
    except:
        return redirect(url_for('error'))

@app.route('/songs_list/<prediction>')
def songs_list(prediction):
    f_path='static/'+prediction
    folder_contents = os.listdir(f_path)
    return
render_template('songs_list.html',folder_contents=folder_contents,prediction=pred
iction)

@app.route('/music-player')
def music_player():
    prediction=request.args.get('prediction')
    song=request.args.get('song')
    return render_template('music_player.html', prediction=prediction,song=song)


@app.route('/error-page')
def error():
    return render_template('error_page.html')
```

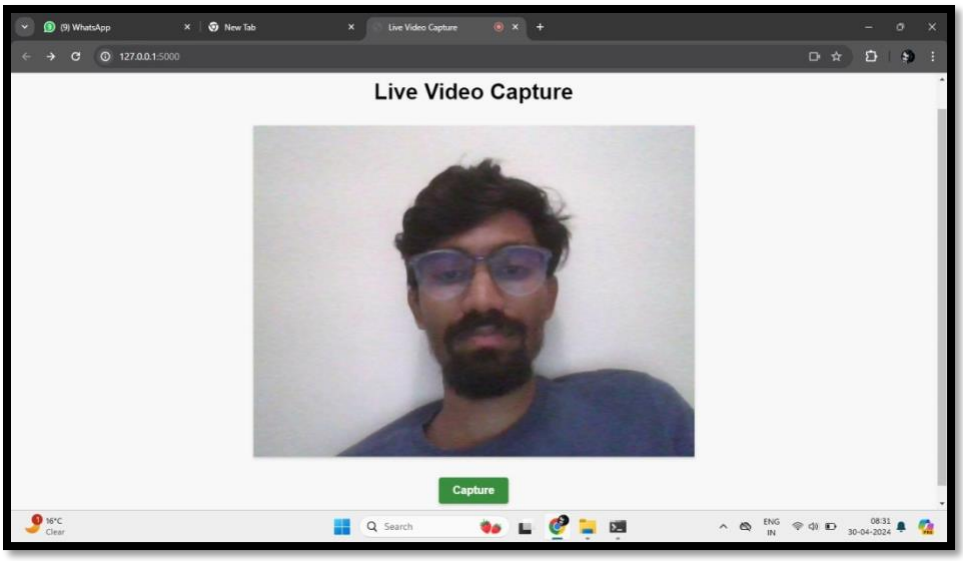
1.6 OUTPUT:

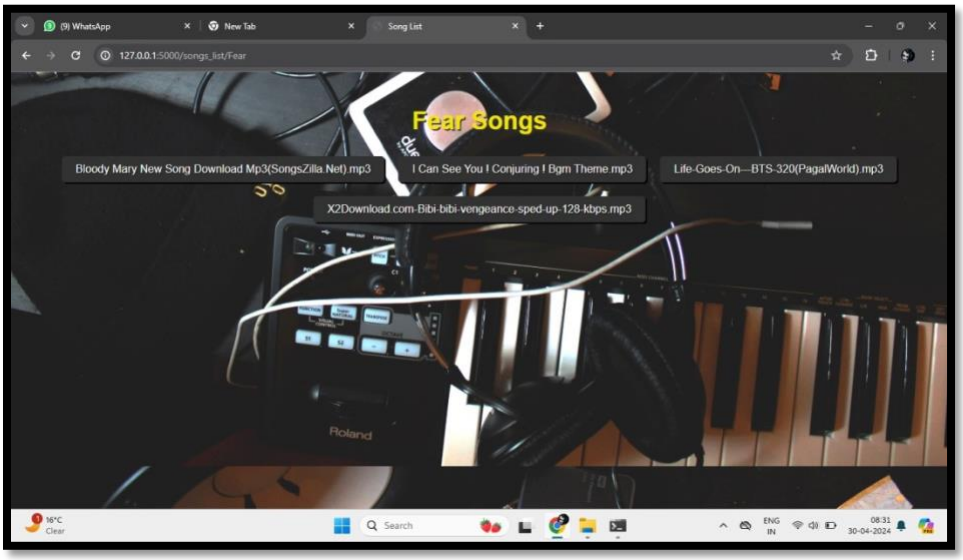
Surprise:



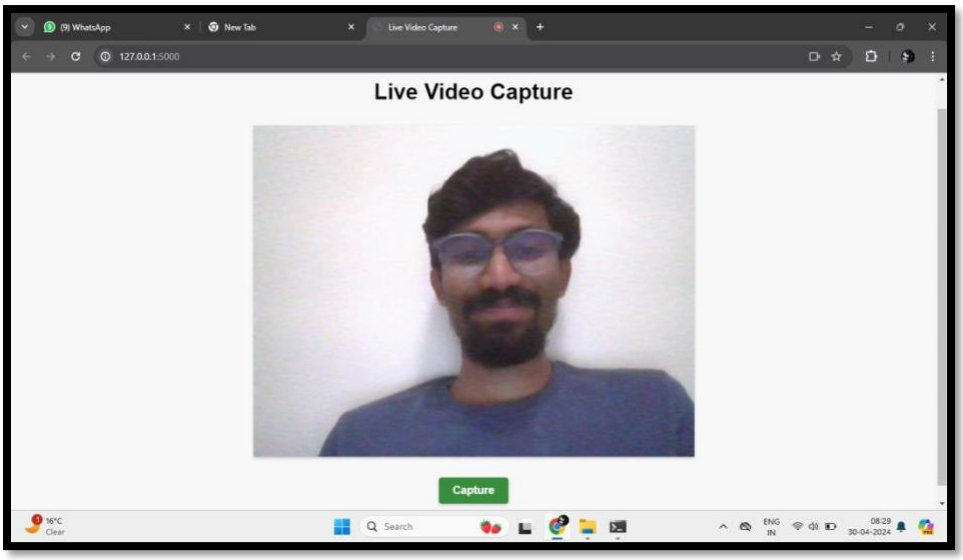


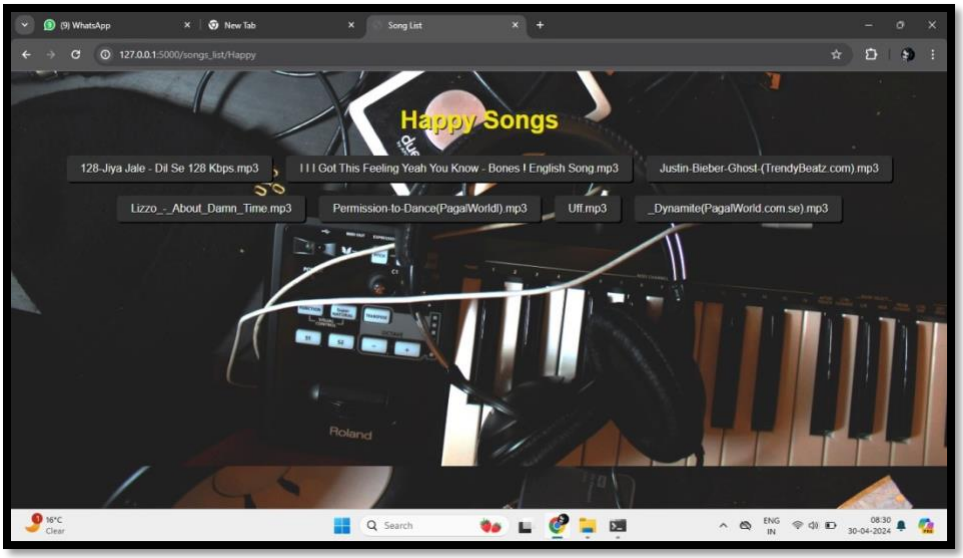
Fear:


Live Video Capture
Capture

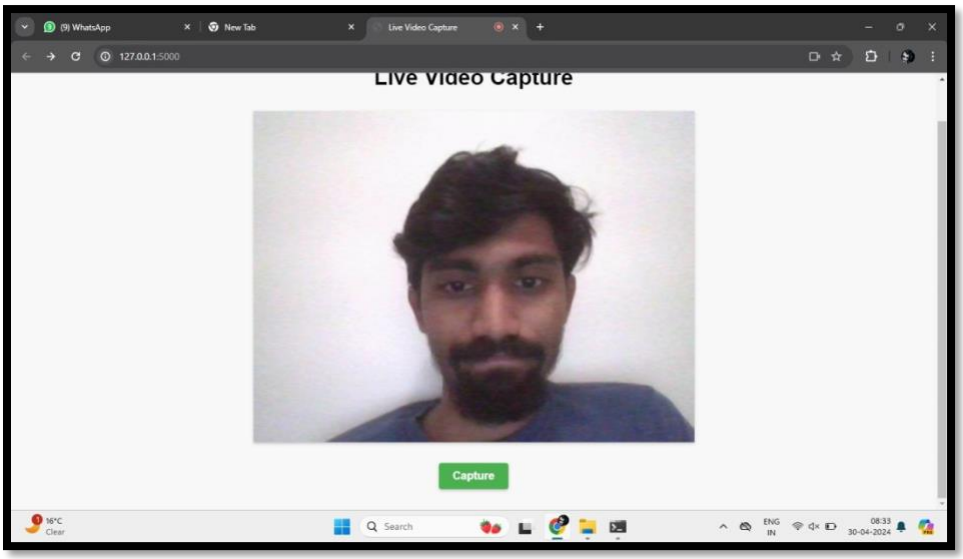

Fear Songs
Bloody Mary New Song Download Mp3(SongsZilla.Net).mp3
I Can See You ǃ Conjuring ǃ Bgm Theme.mp3
Life-Goes-On---BTS-320(PagalWorld).mp3
X2Download.com-Bibi-bibi-vengeance-sped-up-128-kbps.mp3

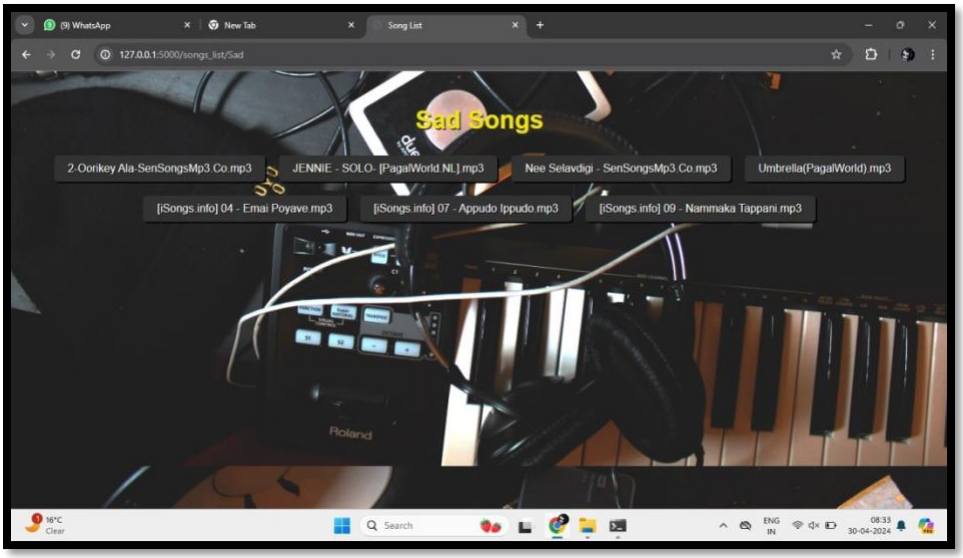
Happy:


Live Video Capture
Capture

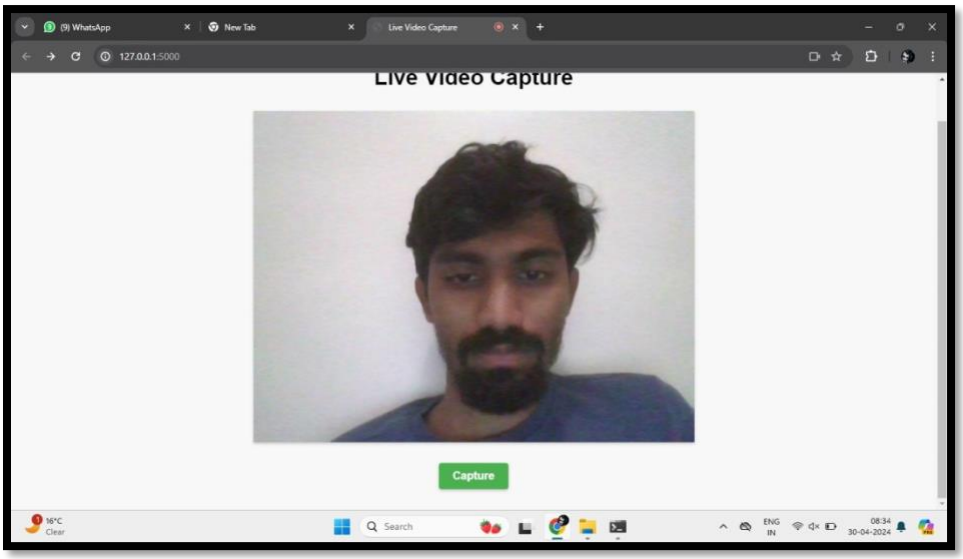

Happy Songs
128-Jiya Jale - Dil Se 128 Kbps.mp3
I I I Got This Feeling Yeah You Know - Bones I English Song.mp3
Justin-Bieber-Ghost-(TrendyBeatz.com).mp3
Lizzo_-_About_Damn_Time.mp3
Permission-to-Dance(PagalWorld).mp3
Uff.mp3
_Dynamite(PagalWorld.com.se).mp3

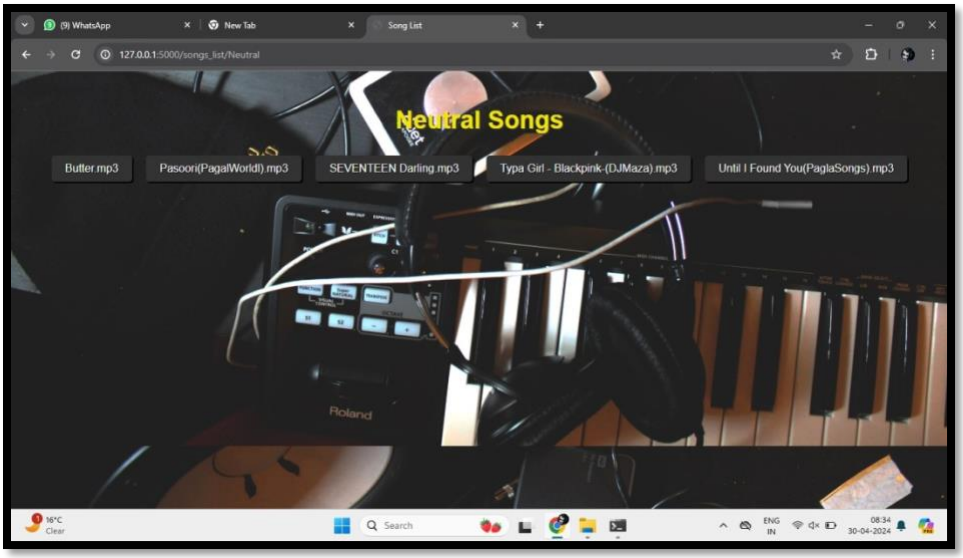
Sad:


Live Video Capture
Capture


Sad Songs
2-Oorikey Ala-SenSongsMp3.Co.mp3
JENNIE - SOLO- [PagalWorld.NL].mp3
Nee Selavdigi - SenSongsMp3.Co.mp3
Umbrella(PagalWorld).mp3
[iSongs.info] 04 - Emai Poyave.mp3
[iSongs.info] 07 - Appudo Ippudo.mp3
[iSongs.info] 09 - Nammaka Tappani.mp3

Neutral:


Live Video Capture
Capture


Neutral Songs
Butter.mp3
Pasoori(PagalWorld).mp3
SEVENTEEN Darling.mp3
Typa Girl - Blackpink-(DJMaza).mp3
Until I Found You(PaglaSongs).mp3

Section C: Test Plan

Table of Contents

1. Test Plan: 52
1.1 Introduction 52
1.2 Purpose 52
1.3 Scope 52
2. Objective 54
2.1 Objective 54
2.2 Success Criteria 54
3. Testing Scope 55
3.1 Inclusions - Features, Modules, and Components to be Tested. 55
1. Emotion Detection Module 55
2. Music Recommendation Engine 55
3. Data Interaction Module 55
4. User Interface Module 55
3.2 Exclusions - Features, Modules, Components not to be Tested. 55
4. Testing Methodologies 55
4.1 Summary of the Integration Plan 55
4.2 Module-to-Test-Technique Mapping 56
5. Monitoring, Reporting, and Correcting Procedures 57
6. Proposed Dates for Submission of Individual Test Reports 57
7. Test Schedule 57
7.1 Milestones and Deadlines 57
7.2 Testing Timeline Overview 58
8. Integration Plan: 59
8.1 Integration testing strategy: 59
8.2 Sequence of integration 59
9. Test Cases and Result: 59
9.1 Modules with sample inputs and outputs 59
9.3 Module Test Cases 62

9.3.1 Emotion Detection Module 62
9.3.2 Music Recommendation Module 63
9.3.3 Data Interaction Module 64
9.3.4 User Interface Module 64
10. Distribution Of workload: 65
10.1 Scheduled Working Activities: 65
10.2 Member Activities or Tasks: 65

1. Test Plan:

1.1 Introduction

The test plan for the mood-based music recommendation system outlines the systematic approach to validate its functionality, accuracy, and user experience. It details objectives, scope, and strategies to ensure robust performance in suggesting music based on users' moods. The plan emphasizes enhancing the system's ability to accurately interpret and respond to user emotions while providing an enjoyable and intuitive interface. By focusing on these aspects, the plan aims to guarantee a seamless user experience and successful deployment of the mood-based music recommendation system.

1.2 Purpose

The primary goal of this test plan is to establish a structured approach for verifying and validating an emotion-based music recommendation system. It emphasizes accurately interpreting user emotions to ensure system functionality, effectiveness, and user satisfaction. Additionally, it aims to provide a seamless and personalized music listening experience based on emotional states. This document guides the testing team with clear objectives for an efficient and effective testing process tailored to the system's unique requirements.

1.3 Scope

The testing scope for the "Mood Based Music Recommendation System Using Convolutional Neural Networks" focuses on evaluating its critical components to ensure efficiency and accuracy. Tests include verifying data acquisition and preprocessing for music and facial features, assessing the performance of the CNN architecture and the Inception model, and ensuring mood detection algorithms are accurate. The music recommendation logic is tested for relevance and responsiveness, while system integration and user interaction are checked for seamless user experience. Security tests ensure compliance with data protection standards, and robustness tests evaluate the system's performance under varying conditions.

1. **Module: Emotion Detection**

   Emotion Detection Accuracy:

   i. Verify the system’s accuracy in identifying users’ emotions through facial analysis or voice tone analysis.

   ii. Test the system's robustness under varying lighting conditions, facial expressions, or voice modulations to simulate real-world scenarios.

2. **Module: Music Recommendation Engine**

   Recommendation System Testing:

i. Confirm that the music recommendations are appropriate for the detected emotions, ensuring relevance and personalization.

ii. Test the recommendation logic under edge cases and less common emotional states to ensure the system's adaptability.

3. **Module: Integration and User Interaction**

System Integration Testing:

i. Validate the seamless integration between the emotion detection module and the music recommendation engine.

ii. Assess error handling and system behaviour when integration points fail, including fallback and recovery strategies.

User Interface Usability:

i. Evaluate the user interface for accessibility, ease of use, and responsiveness, ensuring a positive user experience.

ii. Verify the effective display of error messages, warnings, and user guidance through the interface.

4. **Module: Data Management and Security**

Data Privacy and Security:

i. Confirm that user data, especially sensitive emotional data, is handled securely, adhering to compliance and privacy standards.

ii. Test data integrity and security measures against potential breaches or data leaks.

Session Management:

i. Ensure robust session management with secure authentication and timely session expiration.

ii. Simulate scenarios to test the system's response to unauthorized access attempts and session hijacking.

5. **Module: Performance and Load Testing**

Load Testing:

i. Evaluate the system's performance under peak and normal load conditions to gauge scalability and resource management.

ii. Conduct stress tests to determine the system's limits and identify potential performance bottlenecks and system anomalies in real-time.

6. **Module: Compliance and Regulatory Testing**

Regulatory Compliance:

i. Ensure that the system meets all relevant regulatory requirements related to music licensing and digital content distribution.

ii. Test the system's compliance with international data protection regulations, especially those governing biometric data.

2. Objective

2.1 Objective

The primary goal of the testing phase for the "Emotion Based Music Recommendation System" is to ensure the system effectively and securely recommends music based on users' emotions. This involves confirming accurate emotion detection, delivering suitable music recommendations, and handling user data with high security and privacy standards. The testing aims to guarantee a seamless and responsive user experience while adhering to data protection regulations.

2.2 Success Criteria

Below are the success criteria:

- The system will achieve an emotion detection accuracy rate of at least 95%, ensuring it can accurately identify the user’s emotional state based on facial analysis or voice tone.
- The system will respond within 2 seconds when recognizing emotions and generating music recommendations, providing a smooth and responsive user experience.
- Security measures will be confirmed to prevent unauthorized access to users’ emotional data and music preferences, ensuring high confidentiality and data protection.
- The system will accurately adapt music recommendations to the detected emotional state, ensuring that the music selections enhance user satisfaction and engagement.

3. Testing Scope

3.1 Inclusions - Features, Modules, and Components to be Tested.

1. Emotion Detection Module
2. Music Recommendation Engine
3. Data Interaction Module
4. User Interface Module

3.2 Exclusions - Features, Modules, Components not to be Tested.

1. **External Integrations:** Testing of integrations with third-party music streaming services or external APIs will not be included. These external systems are assumed to function as specified by their providers and are outside the scope of this system's internal testing.

2. **Network Stability:** Exhaustive testing related to varying network conditions and bandwidth constraints will be excluded, focusing primarily on the application's functionality under standard operational conditions.

4. Testing Methodologies

The testing methodology for the Mood-Based Music Recommendation System advances in phases, with progressive testing performed alongside system development. This strategy guarantees the methodical validation of the system's efficiency and effectiveness in accurately identifying emotions and recommending music accordingly.

4.1 Summary of the Integration Plan

We have structured a comprehensive integration plan that encompasses various tests to evaluate the functionality and security of the System. The integration plan includes the following tests:

1. **Unit Testing:**

   Testing individual components or units of the music recommendation system to ensure that each part functions correctly in isolation. This involves testing each module, such as emotion detection, music recommendation logic, and data interaction separately.
2. **Integration Testing:**

   After unit testing, this methodology tests the interaction between integrated modules to ensure they operate together seamlessly. For example, ensuring that the emotion detection module properly interfaces with the music recommendation engine.
3. **System Testing:**

   This involves testing the complete system to verify that it meets all specified requirements. System testing checks the end-to-end functionality of the system and its compliance with the requirements.
4. **Acceptance Testing:**

   Conducted to determine if the system is ready for release. This testing is usually done with input from end-users to ensure the system meets their needs and expectations.
5. **Performance Testing:**

   Tests the performance and responsiveness of the system under various conditions. It includes load testing to determine how the system behaves under normal and peak loads, and stress testing to find the limits at which the system fails.

4.2 Module-to-Test-Technique Mapping

The following table summarizes the mapping of modules to testing techniques for the required testing phases:

Sr. No.	Module	Unit Testing	Integration Testing	System Testing
1	Emotion Detection	White Box Testing	Integration Testing	End-to-End Testing
2	Music Recommendation	Unit Testing	Integration Testing	System Testing
3	User Interface	Usability Testing	Integration Testing	System Testing
4	User Data Interaction	Data Integrity Testing	Integration Testing	System Testing
5	Playlist Management	Unit Testing	Integration Testing	System Testing
6	Audio Data Processing	Unit Testing	Integration Testing	Performance Testing

5. Monitoring, Reporting, and Correcting Procedures

To ensure an effective testing process, we have established the following procedures:

- **Monitoring:** Regular progress meetings will be held to monitor the status of each testing phase. Continuous communication channels will be maintained to address any issues promptly.
- **Reporting:** A detailed test report will be generated after each testing phase, outlining the findings, test results, and any identified issues or areas for improvement.
- **Correcting Procedures:** In the event of identified issues, a dedicated team will be responsible for implementing corrections. The corrections will undergo retesting to ensure successful resolution.

6. Proposed Dates for Submission of Individual Test Reports

Sr. No.	Testing Type	Timeline
1)	Unit Testing Report	February 27
2)	Integration Testing Report	March 15
3)	System Testing Report	April 2

7. Test Schedule

7.1 Milestones and Deadlines

The integration plan for the Mood-Based Music Recommendation System is structured to progressively combine individual modules, beginning with unit tests and advancing towards comprehensive end-to-end system testing. Regular communication channels will be maintained among the development and testing teams to promptly address any issues arising during integration. This phased approach ensures a meticulous and systematic evaluation, leading to a robust and reliable system ready for deployment. The integration process encompasses the following key components, each accompanied by specific target dates.

Milestone 1: Setup and Configuration (Week 1-2)
Tasks:

1. Infrastructure Setup (Week 1):
   - Identify and allocate the necessary hardware resources for testing.
   - Set up the hardware components, including cameras and computing devices.

2. Software Configuration (Week 1-2):
   - Install and configure the operating systems on designated machines.
   - Set up the required software dependencies, including the Face Recognition software.

- Validate that the software versions align with the system specifications.

Milestone 2: Unit Testing (Week 3-4)
Tasks:

1. Unit Testing (Week 3):
   - Objective: Verify the correctness of individual modules and components before their integration into the broader system.
   - Approach:
     - Conduct unit tests for the Emotion Detection module, User Interface, Face Dataset Interaction.
     - Evaluate the functionality of each module in isolation to ensure it meets specified requirements.

Milestone 3: Integration Testing (Week 5-6)
Tasks:

1. Integration Testing (Week 5):
   - Objective: Ensure seamless interaction and functionality of integrated modules.
   - Approach:
     - Integrate Face Detection and Mood detection to verify accurate detection and identification.
     - Combine User Interface with Face Dataset Interaction to assess the flow of data and user interaction.

Milestone 4: System Testing (Week 7-9)
Tasks:

1. Functional Testing (Week 7):
   - Objective: Validate the overall performance and functionality of the integrated System.
   - Approach:
     - Conduct end-to-end tests to simulate real-world scenarios, assessing the entire user authentication process.
     - Evaluate the system's response time and performance under normal and peak loads.
     - Verify the proper functioning of the integrated modules collectively.

2. Performance Testing (Week 9):
   - Assess the system's performance on different hardware setups.

Deadline: End of Week 10

7.2 Testing Timeline Overview

- Weeks 1-2: Setup and Configuration
- Weeks 3-4: Unit Testing
- Weeks 5-6: Integration Testing

- Weeks 7-9: System Testing
- Week 10: Deadline

8. Integration Plan:

8.1 Integration testing strategy:

1. **Sequential Integration:** Modules will be integrated in a sequence that follows the logical flow of the system, starting from emotion detection to music recommendation delivery. Each integration point will be tested individually to isolate and resolve issues without the interference of other components.

2. **Incremental Integration:** The system will be built and tested incrementally, adding one module at a time to the already tested units. This approach helps in identifying defects at early stages.

8.2 Sequence of Integration

The chosen sequence of integration follows a logical progression based on the dependencies and interactions between modules:

1. Emotion Detection Module
2. Music Recommendation System
3. User Interface
4. Data Management System

9. Test Cases and Result:

The following test case documentation outlines the planned scenarios, conditions, and expected outcomes for validating the Face Recognition and Authentication System.

9.1 Modules with sample inputs and outputs

Test Case ID	Module	Test Case Description	Steps to Execute	Expected Result
TC-FD01	Face Detection	Face Detection in Ideal Conditions	1. Initialize the camera in a well-lit environment. 2. Capture an image and detect the face in the frame.	Face is accurately detected and localized in the frame.

TC-FD02	Face Detection	Face Detection under Poor Lighting	1. Initialize the camera in low-light conditions. 2. Capture an image and detect the face in the frame.	System handles poor lighting and still detects the face accurately.
TC-FR01	Face Registration	Successful Registration	1. User enters valid credentials. 2. Capture user's facial image.	User registration is successful; details and image are stored in the dataset.
TC-FE01	Feature Extraction	Successful Feature Extraction	1. Process the detected face to extract features. 2. Send features to the Emotion Detection module.	Features are successfully extracted and sent for emotion analysis.
TC-ER01	Emotion Recognition	Emotion Recognition for Registered User	1. Compare extracted features with the emotion model. 2. Identify the detected emotion.	Correct emotion is identified based on the model analysis.
TC-ER02	Emotion Recognition	Emotion Recognition Failure (Unrecognized Face)	1. Use features that do not match typical patterns. 2. Attempt to identify emotion.	System recognizes the anomaly and triggers a recognition failure scenario.
TC-MR01	Music Recommendation	Successful Music Recommendation	1. Based on identified emotion, fetch appropriate music playlist. 2. Display	Playlist matching the detected emotion is successfully recommended.

			playlist to the user.	
TC-MR02	Music Recommendation	Music Recommendation Failure	1. Provide an unrecognized emotion. 2. Attempt to fetch music.	System fails gracefully, providing an error message or fallback option.
TC-AT01	Attendance Tracking	Successful Attendance	1. Log user's attendance in a virtual session using facial recognition. 2. Record the time and session details.	Attendance is successfully logged with accurate time and session data.
TC-LG01	Logs	Proper Logging of System Activities	1. Perform various system activities. 2. Check logs for entries of these activities.	All activities are correctly logged with detailed information.
TC-LG02	Logs	Retrieval of Logs for Analysis	1. Attempt to access and retrieve logs for a specific period. 2. Analyze the log entries.	Logs are accessible and contain all required data for analysis without errors.

9.3 Module Test Cases

9.3.1 Emotion Detection Module

Serial No.	Condition to be Tested	Test Data	Expected Output	Remarks
1	Accurate Emotion Detection for Basic Emotions	User images showing basic emotions (e.g., Happy, Sad)	Correct emotion detected for each image.	PASS
2	Emotion Detection with Accessories	Images with faces wearing glasses or hats.	Emotions correctly identified despite obstructions.	PASS
3	Emotion Detection in Various Lighting Conditions	Images taken in low light, backlight, and bright conditions.	Correct emotions detected across all lighting conditions.	PASS
4	Emotion Detection across Different Ages and Ethnicities	Diverse demographic images displaying various emotions.	Emotions accurately detected regardless of age or ethnicity.	PASS
5	Detection of Subtle Emotions	Images showing subtle emotions (e.g., confusion, disappointment).	Subtle emotions correctly identified.	PASS
6	Rapid Emotion Changes	Video clips where the subject changes facial expressions quickly.	Each emotion transition detected accurately and timely.	PASS
7	Stress Test for Emotion Detection	High volume of images processed simultaneously.	System accurately detects emotions without delays.	PASS
8	Integration with Music Recommendation Module	Images with clear emotional expressions.	Detected emotions trigger appropriate music recommendations.	PASS
9	Emotion Misidentification Scenario	Images with ambiguous or mixed emotions.	System either identifies dominant emotion	PASS

			or flags ambiguity.	
10	Emotion Detection Failure Due to Image Quality	Blurry or low-resolution images.	System flags images as inadequate for processing.	PASS

9.3.2 Music Recommendation Module

Test Case ID	Emotion	Expected Music Genre	Input Data	Expected Result	Status	
TC001	Happy	Upbeat, Pop	Smile image	Playlist from Happy folder	PASS	
TC002	Sad	Classical, Soft Rock	Frowning image	Playlist from Sad folder	PASS	
TC003	Anger	Heavy Metal, Rock	Scowling image	Playlist from Anger folder	PASS	
TC004	Neutral	Jazz, Blues	Neutral image	Playlist from Neutral folder	PASS	
TC005	Fear	Ambient, Soundtrack	Surprised image	Playlist from Fear folder	PASS	
TC006	Surprise	Electronic, Dance	Shocked image	Playlist from Surprise folder	PASS	
TC007	None	None	Unclear image	Error message: 'No emotion detected'	PASS	

9.3.3 Data Interaction Module

Test Case ID	Functionality	Input Data	Expected Result	Remarks	
DI001	Data Retrieval	Query for 'Happy' playlist	Successful retrieval of Happy playlist tracks	PASS	
DI002	Data Storage	Store new user preferences	Data stored without errors	PASS	
DI003	Data Integrity	Check consistency of playlist data	No discrepancies in data	PASS	
DI004	Performance	Simulate high load data retrieval	Timely data retrieval under high load	PASS	
DI005	Error Handling	Query with invalid data	Error message returned	PASS	

9.3.4 User Interface Module

Test Case ID	Aspect Tested	Action Performed	Expected Outcome	Remarks
MP001	Functionality	Click 'Play' button	Music starts playing	PASS
MP002	Functionality	Click 'Pause' button	Music pauses	PASS
MP003	Functionality	Click 'Stop' button	Music stops and resets to start	PASS
MP004	Usability	Load page	All controls are visible and operational	PASS
MP005	Usability	Resize browser window	Player and buttons adjust to screen size	PASS

10. Distribution Of workload:

10.1 Scheduled Working Activities:

Activity	Period	Comment
Setup and Configuration	Week 1-2	Establish test environment, configure hardware/software components
Unit Testing	Week 3-4	Conduct unit testing for individual modules, document test cases and outcomes
Assist in Unit Testing	Week 3-4	Collaborate in unit testing, begin drafting individual test reports
Integration Testing	Week 5-6	Lead integration testing efforts, verify module interactions
Collaborate in Integration	Week 5-6	Collaborate in integration testing, document integrated module functionalities
System Testing	Week 7-8	Oversee system testing, evaluate overall system behaviour and performance
Engage in System Testing	Week 7-8	Participate in system testing, begin drafting individual test reports
Finalization and Reporting	Week 11-12	Ensure all testing phases are completed, compile a comprehensive test summary report
Review and Contribute	Week 11-12	Review and contribute to the test summary report, finalize individual test reports

10.2 Member Activities or Tasks:

The designated initials for each team member as specified-

- M1= Chiru Anand
- M2= Roshan Rajala
- M3= Sai Kumar
- M4= Heeth

Member	Activity	Period	Start Date	End Date	Comment
M1	Setup and Configuration	Week 1-2	Jan 19, 2024	Jan 30, 2024	Establish test environment, configure hardware/software components
M2, M3, M4	Collaborate in Setup	Week 1-2	Jan 18, 2024	Jan 28, 2024	Assist M1 in ensuring consistent setup
M1	Unit Testing	Week 3-4	Feb 2, 2024	Feb 13, 2024	Conduct unit testing for

					individual modules, document test cases and outcomes
M2, M3, M4	Assist in Unit Testing	Week 3-4	Feb 2, 2024	Feb 12, 2024	Collaborate in unit testing, begin drafting individual test reports
M1,M2	Integration Testing	Week 5-6	Feb 14, 2024	Feb 25, 2024	Lead integration testing efforts, verify module interactions
M3, M4	Collaborate in Integration	Week 5-6	Feb 15, 2024	Feb 24, 2024	Collaborate in integration testing, document integrated module functionalities
M2	System Testing	Week 7-8	Feb 28, 2024	Mar 7, 2024	Oversee system testing, evaluate overall system behavior and performance
M3, M4	Engage in System Testing	Week 7-8	Feb 28, 2024	Mar 10, 2024	Participate in system testing, begin drafting individual test reports
M2	Finalization and Reporting	Week 11-12	Mar 27, 2024	Apr 6, 2024	Ensure all testing phases are completed, compile a comprehensive test summary report
M2, M3, M4	Review and contribute	Week 11-12	Mar 27 , 2024	Apr 10, 2024	Review and contribute to the test summary report, finalize individual test reports
M1	Unit Testing	Week 13	Apr 10, 2024	Apr 11, 2024	Conduct unit testing for attendance module,

					document test cases and outcomes
M1,M2	Integration Testing	Week 13	Apr 10, 2024	Apr11 , 2024	Conduct integration testing for attendance module, verify module interactions
M2, M3,M4	System Testing	Week 13	Apr 10, 2024	Apr 12, 2024	Oversee system testing, evaluate overall system behavior and performance
M2,M3,M4	Finalization and Reporting	Week 13	Apr 10, 2024	Apr 14, 2024	Ensure all testing phases are completed and document all results.

Maintenance and Support

Following the system's launch, complimentary maintenance will be provided for the initial year. Subsequently, a maintenance fee of $22 per hour will apply.

Documentation and Training

A user manual is included with the website, and the client will receive the system source code. Additionally, there is an option to deploy a training team at a rate of $25 per member per hour.

Future enhancements

- The mood-based music recommendation system powered by an Inception model, designed to analyze users' emotional states and tailor music recommendations accordingly. Leveraging data collection, feature extraction, and machine learning, the system will be assessed using metrics such as accuracy, precision, recall, F1 score, and ROC curve analysis.

- The proposed system is anticipated to significantly enhance personalized music experiences, thereby positively affecting users' emotional well-being. It will be suitable for a variety of applications, including entertainment, healthcare, and therapy. Future enhancements will likely include the integration of lyrics and voice recognition for more detailed emotional analysis. Additionally, the implementation of real-time emotion recognition will enable the system to dynamically adjust music recommendations based on users' changing emotional states. Exploring applications in mental health and wellness,

where music has demonstrated a positive impact, will also be a priority. These advancements are expected to improve the precision of the emotion identification model and broaden the system's applicability, enhancing user engagement and satisfaction.

References

a. Li, X., Li, C., Li, T., & Liu, Q. (2019). An emotion-based music recommendation system using deep learning. IEEE Access, 7, 13888–13899.

b. Metilda Florence S and Uma M, 2020, “Emotional Detection and Music Recommendation System based on User Facial Expression”, IOP Conf. Ser.:Mater. Sci. Eng. 912,06/2007.

c. Sarkar, R.; Choudhury, S.; Dutta, S.; Roy, A.; Saha, S.K. Recognition of emotion in music based on deep convolutional neural network. Multimed. Tools Appl. 2020, 79, 765–783

d. Lee, S. Y., & Kim, H. J. (2019). An emotion-based music recommendationsystem using deep learning and physiological signals. Applied Sciences, 9(22), 4716.

e. Chaudhary, D.; Singh, N.P.; Singh, S. Development of music emotion classification system using convolution neural network. Int. J. Speech Technol. 2021, 24, 571–580.

f. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings ofthe IEEE conference on computer vision and pattern recognition (pp. i. 2818-2826).

g. Ren, Jianfeng, Nasser Kehtarnavaz, and Leonardo Estevez. "Real-time optimization of Viola- Jones face detection for mobile platforms." 2008 IEEEDallas Circuits and Systems Work- shop: System-on-ChipDesign, Applications,

h. Castrilln, M., et al. "ENCARA2: Real-time detection of multiple faces at different resolutions in video streams." Journal of visual communication andimage representation 18.2 (2007): 130-140.

i. Renuka R. Londhe, Dr.Vrushshen P. Pawar, “Analysis of Facial Expression and Recognition Based On Statistical Approach”, International Journal of Soft Computing and Engineering (IJSCE)Volume-2, May 2012.

j. Integration, and Software. IEEE, 2008.

k. Shakhnarovich, Gregory, Paul A. Viola, and Baback Moghaddam. "A unified learning frame- work for real time face detection and classification." Proceedings of Fifth IEEE international conference on automatic face gesturerecognition. IEEE, 2002.

l. Dr. Shaik Asif Hussain and Ahlam Salim Abdallah Al Balushi, “A realtime face emotion classification and recognition using deep learningmodel”, 2020 Journal. of Phys.: Conf. Ser. 1432 012087.

m. Luh, Guan-Chun. "Face detection using combination of skin color pixeldetection and Viola- Jones face detector." 2014 International Conference on Machine Learning and ybernetics. Vol. 1. IEEE, 2014