Project Report

on

MoodMatch

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

Bachelor of Computer Applications

(2021-2024)

Guided By:

Dr. Meenu Chopra
Associate Professor
VSIT, VIPS

Submitted by:

Sanya Virmani
Roll No.: 02929802021
BCA VI-EA

Vivekananda School of Information Technology
Vivekananda Institute of Professional Studies – Technical Campus
Approved by AICTE, Accredited Grade "A++" Institution by NAAC, NBA Accredited,
Recognized under Section 2(f) by UGC, Affiliated to GGSIP University.
Recognized by Bar Council of India,
ISO 9001:2015 Certified

Vivekananda Institute of Professional Studies – Technical Campus
(Affiliated to Guru Gobind Singh Indraprastha University)

CERTIFICATE

This is to certify that this project entitled "MoodMatch" submitted in partial fulfilment of the degree of Bachelor of Computer Applications to the "Guru Gobind Singh Indraprastha University- GGSIPU" through Dr. Meenu Chopra done by Ms. Sanya Virmani, Roll No. 02929802021 is an authentic work carried out by her at Vivekananda Institute of Professional Studies under my guidance. The matter embodied in this project work has not been submitted earlier for award of any degree to the best of my knowledge and belief.

Signature of the Student **Signature of the Guide**

SELF CERTIFICATE

This is to certify that the project report entitled “MoodMatch” is done by me is an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of Bachelors of Computer Applications under the guidance of Dr. Meenu Chopra. The matter embodied in this project work has not been submitted earlier for award of any degree or diploma to the best of my knowledge and belief.

Sanya Virmani
Roll No. 02929802021

ACKNOWLEDGEMENT

It is our proud privilege to express our profound gratitude to the entire management of Vivekananda Institute of Professional Studies – Technical Campus and teachers of the institute for providing us with the opportunity to avail the excellent facilities and infrastructure. The knowledge and values inculcated have proved to be of immense help at the very start of my career. Special thanks to Hon’ble Founder, Vivekananda Institute of Professional Studies- Technical Campus, Delhi for having provided us an excellent infrastructure at VSIT.

I am grateful to Prof. Deepali Kamthania (Dean, VSIT), and Dr. Meenu Chopra for their astute guidance, constant encouragement and sincere support for this project work. Sincere thanks to all my family members, seniors and friends for their support and assistance throughout the project.

Sanya Virmani

TABLE OF CONTENTS

TABLE OF FIGURES

1. ABSTRACT

1.1 INTRODUCTION

MoodMatch aims to develop an intelligent system that integrates computer vision, machine learning, and music streaming services to provide real-time emotion-based music recommendations. Utilizing OpenCV for facial detection, a pre-trained Convolutional Neural Network (CNN) model from Keras for emotion recognition, and the Spotipy library to interact with the Spotify API, the system captures a user's facial expression through a webcam, identifies the emotional state, and suggests a suitable music playlist. It highlights the integration of multiple technologies to enhance user experience through personalized music recommendations.

This project builds on the growing trend of personalized experiences in digital services, where user-specific recommendations are becoming increasingly prevalent. For instance, platforms like Netflix and YouTube suggest content based on viewing history and preferences. However, these systems typically rely on past behavior rather than real-time user states. By incorporating real-time emotion detection, this project aims to bridge that gap, offering music recommendations that resonate with the user's current mood. This not only enhances user satisfaction but also demonstrates the potential of emotion-aware applications in improving the responsiveness and intuitiveness of digital services.

1.2 PROBLEM STATEMENT

Traditional music recommendation systems primarily rely on historical data, such as past listening habits and user preferences, to suggest songs or playlists. While effective, these systems lack the capability to adapt to the user's current emotional state, potentially leading to recommendations that do not align with the user's immediate mood. This gap in personalized, real-time responsiveness can detract from the user experience, especially in contexts where music serves as an emotional companion.

For example, imagine a person who has just come home after a particularly stressful day at work. Despite usually enjoying energetic rock music, they might prefer something soothing and calming at that moment. However, a standard recommendation system would still suggest rock music based on their past preferences, failing to meet their current emotional needs. This project addresses this limitation by integrating real-time emotion detection into the music recommendation process, aiming to enhance the relevance and satisfaction of music suggestions.

1.3 WHY IS THE PARTICULAR TOPIC CHOSEN?

The topic of emotion-based music recommendations is chosen due to its potential to significantly improve user experience by providing a more intuitive and responsive interaction with music streaming services. With advancements in computer vision and machine learning, it is now possible to accurately detect human emotions from facial expressions. Leveraging these technologies to enhance music recommendations offers a novel and practical application that aligns with the increasing demand for personalized digital experiences.

For instance, consider a student studying for exams late at night. Their usual preference might be for upbeat pop music, but as they become more fatigued and stressed, the system could detect these changes in their facial expressions and switch to recommending relaxing instrumental music. This capability not only enhances user satisfaction but also demonstrates the practical benefits of emotion-aware applications in real-life situations, making this project both timely and relevant.

1.4 KEY FUNCTIONALITIES

- **Real-Time Facial Detection:** Using OpenCV to capture and detect faces in real-time through a webcam.
- **Emotion Recognition:** Employing a pre-trained Convolutional Neural Network (CNN) from Keras to classify detected facial expressions into predefined emotional categories.
- **Music Recommendation:** Integrating with the Spotify API via the Spotipy library to fetch and suggest playlists based on the detected emotional state.
- **User Interface:** Providing a user-friendly interface that displays the detected emotion and opens the recommended playlist in a web browser.

1.5 THEORITICAL FOUNDATION

The theoretical foundation of this project is built on the intersection of computer vision, machine learning, and affective computing:

- **Computer Vision:** Utilizes OpenCV for real-time image capture and facial detection, which is a key component in interpreting human emotions visually.

- **Machine Learning:** Employs deep learning techniques, specifically Convolutional Neural Networks (CNNs), to recognize and classify facial expressions. CNNs are particularly effective for image-related tasks due to their ability to capture spatial hierarchies.

- **Affective Computing:** This interdisciplinary field focuses on the study and development of systems that can recognize, interpret, and process human emotions. This project applies affective computing principles to enhance user interaction with music streaming services.

1.6 USER-CENTRIC DESIGN

The design of this system prioritizes the user experience by ensuring:

- **Ease of Use:** A straightforward and intuitive interface where users simply need to look into the webcam to receive music recommendations.
- **Immediate Feedback:** Real-time processing and display of detected emotions and recommended playlists to provide instant gratification.
- **Relevance:** Personalized music suggestions that align with the user's current emotional state, enhancing the emotional and psychological benefits of music listening.

1.7 DEVELOPMENT METHODOLOGY

The project follows an Agile development methodology, characterized by iterative and incremental progress. Key stages include:

1. **Requirement Analysis:** Gather user requirements and define system specifications.
2. **Design:** Develop system architecture, focusing on modularity and integration of different components.
3. **Implementation:** Code the individual modules (face detection, emotion recognition, and music recommendation) and integrate them.
4. **Testing:** Conduct unit and integration testing to ensure each component works as intended and the overall system is reliable.
5. **Deployment:** Deploy the system for user testing and gather feedback.
6. **Iteration:** Refine the system based on user feedback and performance metrics, repeating the cycle for continuous improvement.

1.8 FUTURE ENHANCEMENTS

In MoodMatch, focusing on specific improvements and additional features that can be incorporated into the existing system to increase its functionality, accuracy, and user experience.

- **Enhanced Emotion Detection:** Improve the accuracy of emotion recognition by training the CNN model on a larger and more diverse dataset, including different age groups, ethnicities, and lighting conditions, to make the model more robust.
- **Expanded Emotional Range:** Incorporate a broader range of emotions and more nuanced emotional states beyond the seven basic emotions currently recognized, allowing for more precise and tailored music recommendations.
- **Cross-Platform Integration:** Extend compatibility to mobile devices and integrate with other music streaming services such as Apple Music, Amazon Music, and YouTube Music, providing users with more options.
- **User Customization:** Allow users to customize the mapping of emotions to music genres based on personal preferences, enabling a more personalized experience.
- **Context-Aware Recommendations:** Incorporate additional contextual information such as time of day, location, weather, and user activity (e.g., working, exercising) to refine and enhance the relevance of music recommendations.
- **Voice and Gesture Control:** Integrate voice and gesture controls to allow users to interact with the system in a more natural and intuitive way, enhancing user engagement and accessibility.

2. MAIN REPORT

2.1 Objective and Scope of the Project

The primary objective of this project is to create a sophisticated system that can detect human emotions in real-time using facial expressions and provide personalized music recommendations based on the detected emotions. The system aims to enhance user experience by leveraging deep learning for accurate emotion recognition and utilizing the Spotify API to fetch relevant playlists that match the user's current emotional state. By integrating these technologies, the project aspires to create a seamless and intuitive user interaction where music recommendations are dynamically adjusted to reflect the user’s mood.

The specific goals include:

- **High Accuracy in Emotion Detection:** Achieve a high level of accuracy (above 90%) in detecting the seven distinct emotions from facial expressions.
- **Real-Time Responsiveness:** Ensure that the system can process and respond to emotional cues within milliseconds, providing an almost instantaneous recommendation.
- **User Satisfaction:** Aim for high user satisfaction by conducting user tests and gathering feedback to continually improve the interface and functionality.
- **Robust Integration:** Seamlessly integrate the emotion recognition system with the Spotify API to provide uninterrupted music streaming and accurate playlist fetching.
- **Scalability:** Design the system to be scalable, allowing for future enhancements and the ability to handle a larger number of users or additional functionalities without significant rework.

The scope of this project encompasses the following key areas and specific goals:

1. **Emotion Detection:**
    - **Recognize Seven Distinct Emotions:** The system will accurately identify seven primary emotions: Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise.
    - **Facial Expression Analysis:** Use advanced facial recognition techniques to analyze subtle changes in facial expressions to improve emotion detection accuracy.
    - **Diverse Dataset Utilization:** Train the emotion recognition model using a diverse dataset that includes different ages, genders, and ethnicities to ensure robustness and generalizability.

2. **Music Recommendation:**
    - **Emotion-Based Playlists:** Suggest music genres that correspond to each detected emotion. For instance:
    - Angry: Rock
    - Disgust: R&B
    - Fear: Metal
    - Happy: Pop
    - Neutral: Chill
    - Sad: Mood
    - Surprise: Party
    - **Integration with Spotify:** Use the Spotify API to fetch and play the most suitable playlists based on the detected emotion.
3. **Real-Time Processing:**
    - **Real-Time Emotion Detection**: Implement real-time processing capabilities to capture, detect, and analyze facial expressions instantaneously.
    - **Low Latency:** Ensure low latency in processing to provide immediate feedback and music recommendations.
    - **Optimized Performance:** Optimize the system for efficient performance to handle continuous real-time data streams without significant lag.

By focusing on these goals and scope areas, the project seeks to create a comprehensive, reliable, and user-friendly system that significantly enhances the music listening experience through intelligent, emotion-based recommendations.

2.2 Theoretical Background and Definition of the Problem

Emotion detection through facial expressions is an interdisciplinary field involving computer vision, psychology, and machine learning. Facial expressions are one of the most natural and universal indicators of human emotions. Research in psychology has established that facial expressions can convey a wide range of emotions, and these expressions have distinct patterns that can be identified and categorized. The development of algorithms and models that can interpret these patterns falls within the realm of computer vision and affective computing.

The use of Convolutional Neural Networks (CNNs) in deep learning has revolutionized the field of image recognition, making it possible to achieve high accuracy in tasks such as facial emotion detection. CNNs are particularly well-suited for this task due to their ability to automatically learn hierarchical representations from raw pixel data, effectively capturing the nuances of facial expressions.

Integrating these advancements with real-time data processing enables the creation of systems that can respond instantly to changes in the user’s emotional state. This capability is particularly valuable in enhancing user experiences in various applications, including personalized music recommendation systems.

Defining the problem:

Traditional music recommendation systems rely heavily on historical data, such as a user's listening history, preferences, and demographic information, to suggest songs or playlists. While these systems can be effective, they do not account for the user's current emotional state, which can significantly influence music preferences. This limitation often results in recommendations that may not align with the user's immediate mood, leading to a suboptimal listening experience.

For example, consider a user who typically enjoys upbeat pop music but is currently feeling stressed and prefers something more calming. A conventional recommendation system, unaware of the user's current emotional state, would continue to suggest pop music based on historical data. This mismatch can detract from the user experience, as the music recommended does not resonate with the user's current feelings.

This project addresses this problem by developing an intelligent system that can detect human emotions in real-time using facial expressions and provide music recommendations based on the detected emotions. The system leverages deep learning for emotion recognition, using a pre-trained CNN model to classify facial expressions into one of seven categories: Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise.

Upon detecting an emotion, the system uses the Spotify API to fetch relevant playlists that match the user's current emotional state. For instance, if the system detects that the user is happy, it might suggest a pop playlist; if the user is sad, it might suggest a mood playlist. This real-time, emotion-aware recommendation system aims to enhance user satisfaction by aligning music suggestions with the user's immediate emotional needs.

By integrating computer vision for emotion detection with a music streaming service like Spotify, the project not only improves the relevance of music recommendations but also demonstrates the potential of emotion-aware applications in digital services. This approach bridges the gap between static recommendation systems and dynamic user experiences, paving the way for more responsive and personalized interactions.

2.3 System Analysis & Design vis-a-vis User Requirements

The system analysis for this project involves a detailed examination of the functional and non-functional requirements, as well as the technical feasibility and constraints. It begins with a thorough understanding of the user needs and expectations, which are translated into specific system requirements. Functional requirements outline the system's features and capabilities, such as real-time emotion detection and personalized music recommendations, while non-functional requirements define aspects like performance, usability, and security.

Additionally, the design phase involves considerations for scalability, reliability, and maintainability. Scalability ensures that the system can accommodate increasing user demand and data volume without significant performance degradation. Reliability ensures that the system operates consistently and accurately under various conditions, while maintainability facilitates easy updates, enhancements, and troubleshooting.

Overall, the system analysis and design phases lay the foundation for the successful development and implementation of the real-time emotion detection and music recommendation system, ensuring that it meets user requirements effectively while leveraging the appropriate technologies and resources.

2.4 System Planning (PERT Chart)

The System Planning phase in the development of an Emotion Detection and Music Recommendation System involves the creation of a Project Evaluation and Review Technique (PERT) chart. This chart serves as a comprehensive roadmap for scheduling and managing various tasks, ensuring a structured and efficient development process.

2.4.1 Defining Project Milestones:

The project milestones are structured into several key stages. These include requirement analysis and gathering, system design, model training and validation, integration and development, testing and debugging, user interface development, deployment, and final refinements. Each milestone represents a critical phase in the project's progression towards completion.

2.4.2 Allocating Resources:

Resource allocation is a critical aspect of System Planning. This involves identifying and assigning the necessary human and technological resources required for each task. Human resources may include developers, data scientists, and system architects, while technological resources encompass computing hardware, software tools like OpenCV, Keras, and Spotipy and any external dependencies.

2.4.3 Estimating Task Durations:

Task durations are estimated based on the complexity and scope of each stage. Requirement analysis and gathering may take approximately one week, while model training and validation could span three weeks due to the need for extensive data processing. Integration and development may require four weeks, followed by two weeks for testing and debugging, and three weeks for user interface development. Deployment and final refinements are allocated one and two weeks, respectively.

2.4.4 Identifying Dependencies Between Tasks:

Dependencies between tasks are crucial for maintaining the project timeline. For instance, model training and validation must be completed before integration and development can begin, while testing and debugging depend on the completion of integration and development. Deployment is contingent upon successful testing and debugging, and final refinements are based on feedback obtained during deployment.

2.4.5 Visual Representation with PERT Chart:

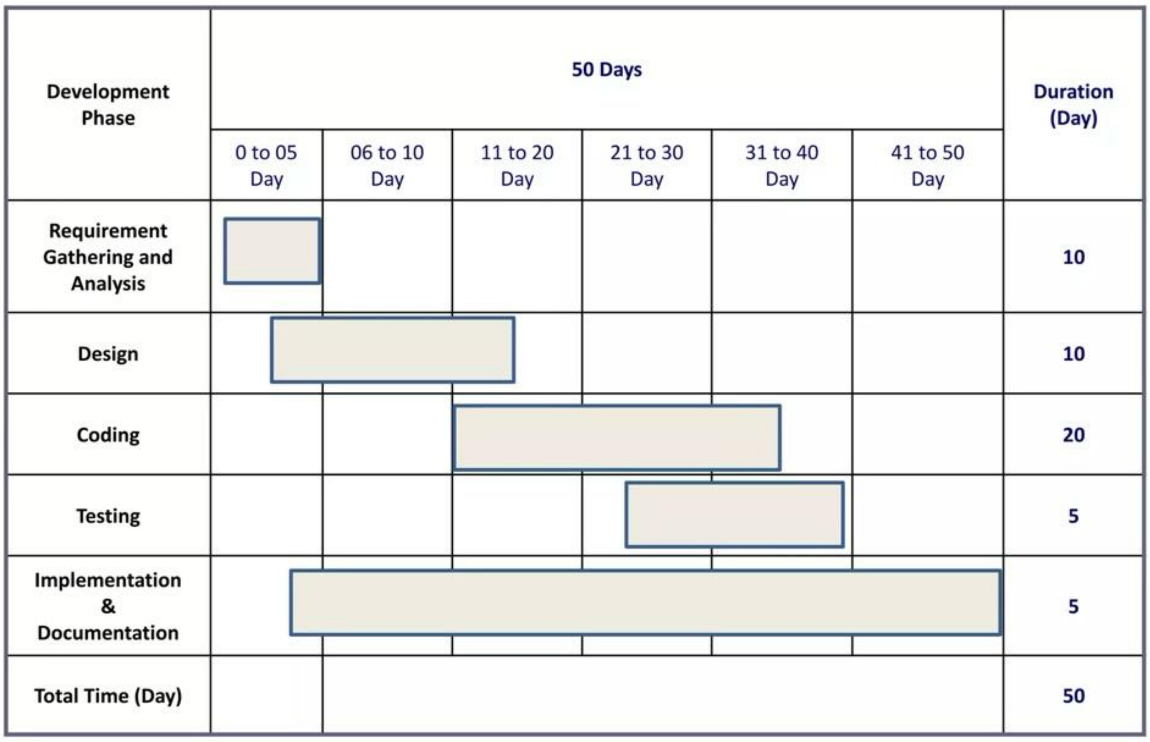
A PERT chart visually represents the project's tasks and timelines, illustrating dependencies and the sequence of activities. Each task is represented by a node, with arrows indicating the flow and dependencies between tasks. The PERT chart provides a clear overview of the project's progression and critical path and helps identify the critical path—the sequence of tasks that determines the minimum duration for project completion.

2.4.6 Critical Path Analysis:

The critical path outlines the sequence of stages determining the minimum time needed to complete the project. In this project, the critical path includes requirement analysis, system design, model training and validation, integration and development, testing and debugging, deployment, and final refinements. Delays in any of these stages will directly impact the project completion timeline.

2.4.7 Resource Management:

The PERT chart aids in resource management by allowing project managers to visualize resource requirements at each stage. This visualization facilitates effective allocation, ensuring that human and technological resources are available when needed. Efficient resource management contributes to the timely execution of tasks.



Development Phase	50 Days						Duration (Day)
	0 to 05 Day	06 to 10 Day	11 to 20 Day	21 to 30 Day	31 to 40 Day	41 to 50 Day	
Requirement Gathering and Analysis	▨						10
Design	▨	▨	▨				10
Coding			▨	▨	▨		20
Testing				▨	▨	▨	5
Implementation & Documentation	▨	▨	▨	▨	▨	▨	5
Total Time (Day)							50

Figure 1.1 PERT CHART

2.5 Methodology

The project utilizes various libraries of Python such as

2.5.1 OpenCV

“OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library”. The main purpose of this was to provide a common infrastructure for computer vision applications and it was also built specifically for such purposes not to mention it also accelerated the use of machine perception inside the business product. “Being a BSD-licensed product, OpenCV makes it straightforward for businesses to utilize and modify the code”. In total we can say that The library has about 2500 optimized algorithms which is really insane, “These algorithms contain a comprehensive set which comprises of each classic and progressive laptop vision and machine learning algorithms. These algorithms area unit usually accustomed sight and acknowledge faces, determine objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, manufacture 3D purpose clouds from s tereo cameras, sew pictures along to produce a high resolution image of a full scene, realize similar pictures from an image info, take away red eyes from pictures taken exploitation flash, follow eye movements, acknowledge scenery and establish markers to overlay it with increased reality, etc”. The amazing thing about this library is that it has quite about forty-seven thousand individuals of user community and calculable variety of downloads Olympian eighteen million. The library is utilized extensively in corporations, analysis teams and by governmental bodies.

Along with well-established corporations like “Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota” that use the library, there are a unit several startups like “Applied Minds, VideoSurf, and Zeitera”, that create in depth use of OpenCV. OpenCV’s deployed wide array spans vary from sewing street view pictures along, police work intrusions in police work video in Israel, watching mine instrumentality in China, serving to robots navigate and devour objects at “Willow Garage, detection of natatorium drowning accidents in Europe, running interactive art in Espana and New York, checking runways for scrap in Turkey”, inspecting labels on product in factories around the world on to fast face detection in Japan.

2.5.2 Numpy

“The Python programming language earlier wasn't originally designed for numerical computing as we know it to be , however it also attracted the attention of the scientific and engineering community early” . “In 1995 the interest (SIG) matrix-sig was based with the aim of shaping associate array computing package; among its members was Python designer and supporter Guido van Rossum, WHO extended Python's syntax (in explicit the compartmentalization syntax) to make array computing easier”.

“An implementation of a matrix package was completed by Jim discoverer, then generalized[further rationalization required by Jim Hugunin and known as Numeric (also diversely observed because the "Numerical Python extensions" or "NumPy").Hugunin, a collegian at the Massachusetts Institute of Technology (MIT), joined the Corporation for National analysis Initiatives (CNRI) in 1997 to work on JPython, leaving Paul Dubois of Lawrence Livermore National Laboratory (LLNL) to need over as supporter. Other early contributors embrace David Ascher, Konrad Hinsen and Travis Oliphant”.

“A new package known as Numarray was written as a additional versatile replacement for Numeric. Like Numeric, it too is currently deprecated. Numarray had quicker operations for large arrays, however was slower than Numeric on tiny ones, thus for a time each packages were utilised in parallel for varied use cases. The last version of Numeric (v24.2) was discharged on St Martin's Day 2005, whereas the last version of numarray (v1.5.2) was discharged on twenty four August 2006”.

There was a want to urge Numeric into the Python customary library, however Guido van Rossum determined that the code wasn't reparable in its state then.

“In early 2005, NumPy developer Travis Oliphant needed to unify the community around one array package and ported Numarray's options to Numeric, cathartic the result as NumPy one.0 in 2006.This new project was a region of SciPy. To avoid putting in the large SciPy package simply to urge associate array object, this new package was separated and known as NumPy. Support for Python three was other in 2011 with NumPy version one.5.0”.

In 2011, PyPy started development on associate implementation of the NumPy API for PyPy.It is not nevertheless absolutely compatible with NumPy.

2.5.3 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of deep learning frameworks such as TensorFlow, Theano, and Microsoft Cognitive Toolkit (CNTK). It provides a user-friendly interface for building and training deep learning models, enabling rapid prototyping and experimentation.

Keras was introduced in 2015 as an abstraction layer over other deep learning frameworks, allowing developers to work with multiple backend engines such as TensorFlow, Theano, and later Microsoft Cognitive Toolkit (CNTK). This flexibility made Keras an attractive option because it allowed users to leverage the strengths of different backends while using a consistent API.

Over time, Keras has evolved significantly, and in 2017, it was integrated into TensorFlow as the official high-level API. This integration further cemented its position in the deep learning community, providing seamless compatibility with TensorFlow's powerful capabilities for large-scale machine learning tasks. Today, it is widely used in academia and industry for a variety of applications, ranging from image recognition and natural language processing to reinforcement learning and generative models.

One of the key advantages of Keras is its simplicity and ease of use, allowing developers to define neural network architectures using a modular, layer-based approach. Keras provides a wide range of pre-built layers for building convolutional neural networks (CNNs), recurrent neural networks (RNNs), and other types of deep learning models. Additionally, Keras supports automatic differentiation and GPU acceleration, making it suitable for training large-scale models on powerful hardware.

In this project, Keras is used for emotion recognition through a pre-trained CNN model. The model, trained on a large dataset of facial expression images, is loaded into Keras, enabling real-time inference on facial images captured by OpenCV. Keras handles the forward pass through the neural network, applying learned weights and biases to classify facial expressions into predefined emotion categories.

.

2.5.4 Spotipy

Spotipy is a lightweight Python library for interacting with the Spotify Web API, enabling developers to access and manipulate Spotify's vast catalog of music, playlists, albums, and user data. It provides convenient methods for performing common tasks such as searching for tracks, retrieving user playlists, and managing user authentication.

Spotipy simplifies the process of integrating Spotify functionality into Python applications, abstracting away the complexities of HTTP requests and authentication mechanisms. It supports various operations, including fetching track metadata, controlling playback, adding tracks to playlists, and retrieving recommendations based on user preferences.

In this project, Spotipy is utilized to fetch music playlists corresponding to detected emotions. Once the emotion is recognized using the CNN model, Spotipy is used to query the Spotify API and retrieve playlists associated with the detected emotion category. The retrieved playlists are then presented to the user as personalized music recommendations, enhancing the overall user experience of the system.

Find all the faces that appear in a picture:

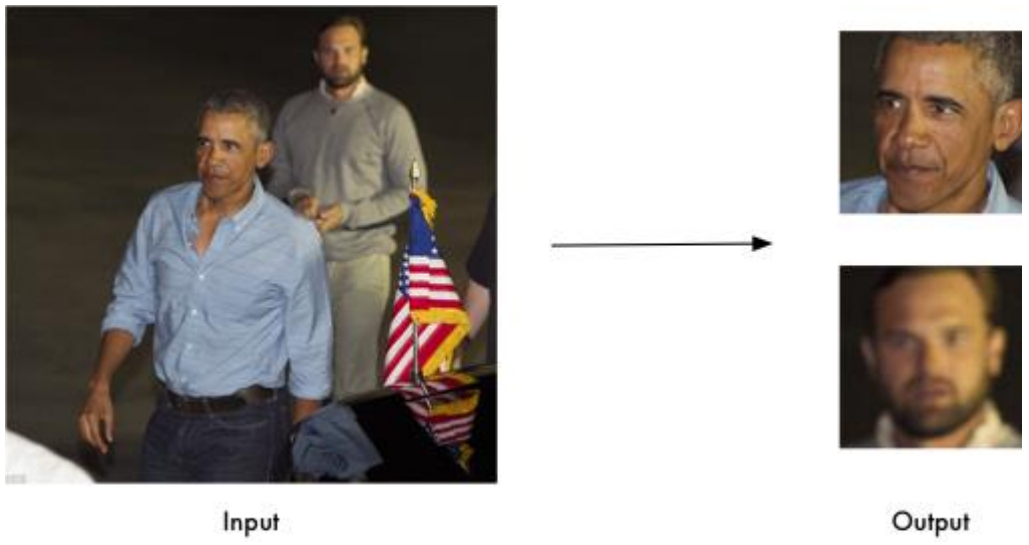


Figure 1.2 Finding all the faces in the picture

Find and accordingly detect their emotions in the pictures:

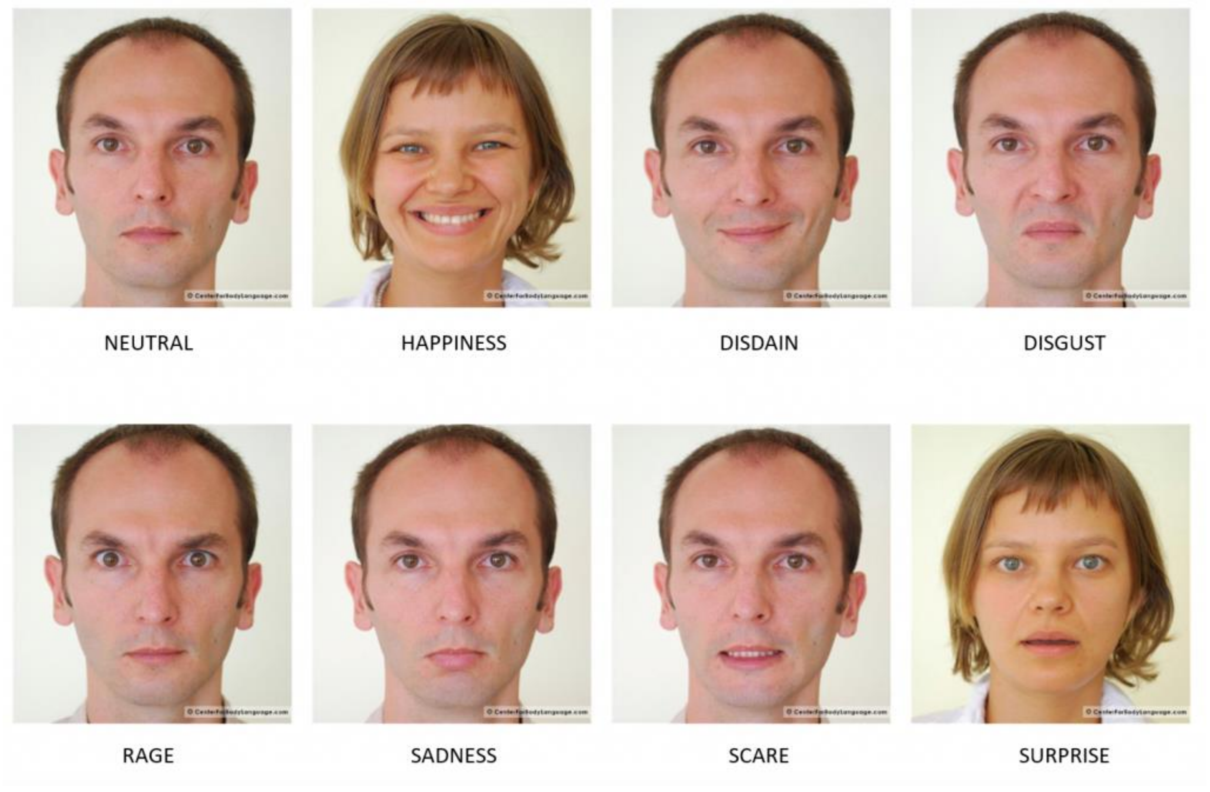


Figure 1.3 Detecting their emotions through Facial Expressions

Further utilising the CNN model to detect their mood:

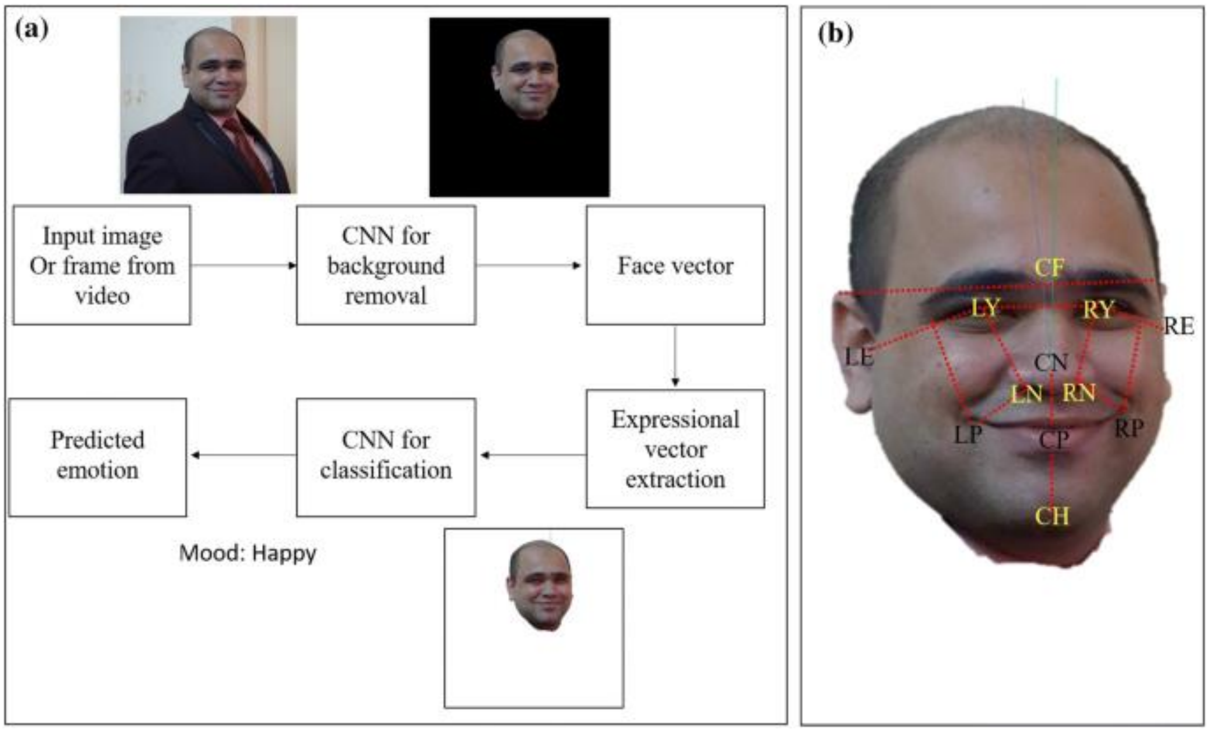


Figure 1.4 The process of Emotion Detection

2.6 System Implementation

System implementation for the Facial Recognition and Emotion-Based Music Recommendation System involves translating the design specifications into a functioning application. This phase encompasses the actual development, coding, and integration of the various components that make up the system. Below are the detailed criteria for the System Implementation phase tailored to this project:

2.6.1 Algorithm Implementation: The first step in system implementation involves developing and implementing algorithms for facial detection and emotion recognition. Utilizing Python libraries such as OpenCV and Keras, the system is designed to detect faces in video frames, extract facial features, and classify emotions using a pre-trained Convolutional Neural Network (CNN) model. OpenCV's Haar cascades are employed for face detection, while the CNN model, loaded using Keras, performs emotion recognition by analyzing facial expressions.

2.6.2 Integration of Python Libraries: The implementation incorporates the integration of various Python libraries. OpenCV is a pivotal library for image processing, providing functionalities for face detection, image manipulation, and video analysis. The facial recognition library complements OpenCV, offering tools specifically tailored for face recognition tasks. Additionally, the os library may be employed for managing file operations and directories, datetime for timestamping, and numpy for efficient numerical operations.

2.6.3 Data Preprocessing: Before training the recognition model, the system needs to preprocess facial data. This involves tasks such as resizing images, normalizing pixel values, and extracting relevant features. Numpy, with its array operations, can be utilized for efficient data preprocessing, ensuring that the data is in a suitable format for training.

2.6.4 Training the Recognition Model: Using the extracted facial features, the recognition model is trained. Python libraries like facial recognition provide convenient interfaces for training and utilizing pre-trained models. This step is crucial for enabling the system to recognize faces accurately.

2.6.5 System Integration and Testing: After algorithm implementation and model training, the components are integrated into a cohesive system. The integrated system is rigorously tested to ensure that face detection and recognition work seamlessly. This phase involves validating the accuracy of the recognition model, checking for false positives/negatives, and evaluating system performance.

2.6.6 System Deployment and User Interface Design: Once testing is complete, the system is deployed, typically on a local machine or server capable of accessing a webcam for real-time video capture. The user interface is designed to be intuitive and user-friendly, facilitating seamless interaction with the system. Users can start the emotion detection process, view real-time results, and receive music recommendations based on their detected emotional state. The interface integrates functionalities such as starting and stopping the webcam, displaying detected emotions, and linking to Spotify playlists.

In conclusion, the system implementation phase brings together various technological components, ensuring they work in harmony to provide an effective real-time emotion detection and music recommendation system. Through meticulous development, integration, and testing, the project achieves its goal of enhancing user experience by leveraging advanced facial recognition and deep learning technologies.

2.7 Details of Hardware & Software used

Hardware Requirements:

The hardware requirements for MoodMatch, an emotion detection and music recommendation system, are high-quality cameras capable of capturing clear and detailed facial images. A powerful CPU or GPU to handle image processing and machine learning tasks efficiently. An adequate RAM for storing and manipulating image data during processing. The system should have sufficient storage space for storing training datasets, models, and system logs.

Software Requirements:

- **OpenCV:** A versatile open-source computer vision library providing a wide range of tools for image processing, computer vision, and machine learning.
- **Python:** A programming language commonly used for developing facial recognition systems due to its simplicity and a rich ecosystem of libraries.
- **Keras:** It is a high-level deep learning library that provides a user-friendly interface for building and training neural networks. It is utilized in the project for implementing the CNN model used to emotion recognition.
- **NumPy:** A library for numerical operations, often used for efficient handling of arrays and matrices in image processing.
- **Spotipy:** It is Python library for interacting with the Spotify Web API. It allows the MoodMatch system to access Spotify's vast music database and retrieve personalized music recommendations based on detected emotion.

By meeting these software requirements, the MoodMatch system can effectively detect emotions in real-time facial images and recommend personalized music playlists based on the detected emotional states

2.8 System Maintenance and Evaluation

System Maintenance and Evaluation are essential phases in the lifecycle of the MoodMatch emotion detection and music recommendation system. These phases ensure the system's continued effectiveness, adaptability to changes, and adherence to desired performance standards.

1. **Regular System Updates and Patch Management:**

- ***Maintenance*****:** Establish a schedule for regular updates and patch management to keep the system current with the latest software versions, including OpenCV, Keras, Spotipy, and other relevant libraries.
- ***Evaluation*****:** Conduct thorough testing after updates to assess their impact on system performance and functionality. Ensure that updates do not introduce new issues or conflicts with existing features.

2. **Continuous Monitoring and Performance Optimization:**

- ***Maintenance*****:** Implement continuous monitoring mechanisms to track system performance, resource utilization, and any potential anomalies. Optimize algorithms, image processing pipelines, and machine learning models to enhance the system's efficiency.
- ***Evaluation*****:** Regularly review performance metrics and conduct benchmarking tests. Evaluate the impact of optimizations on response times, accuracy, and overall system throughput.

3. **Data Quality and Retraining of Recognition Models:**

- ***Maintenance*****:** Monitor and maintain the quality of the emotion detection dataset, periodically updating it with diverse and representative facial images to improve model robustness.
- ***Evaluation*****:** Evaluate the emotion recognition model's performance by periodically retraining it with updated datasets. Measure accuracy, precision, and adaptability to new variations in facial expressions.

4. **Security Audits and Privacy Compliance:**

- ***Maintenance*:** Conduct regular security audits to identify and address potential vulnerabilities in the system. Ensure compliance with privacy regulations through encryption, access controls, and secure data storage.
- ***Evaluation*:** Verify the effectiveness of security measures through penetration testing and compliance assessments. Address any identified gaps to mitigate risks effectively.

5. **User Training and Support:**

- ***Maintenance*:** Provide ongoing user training and support to ensure that end-users understand how to interact with the system effectively. Address user feedback and continuously improve the user interface or experience based on evolving requirements.
- ***Evaluation*:** Gather user feedback through surveys or forums to assess system usability and satisfaction. Identify areas for improvement and implement enhancements accordingly.

6. **Incident Response and Error Handling:**

- ***Maintenance*:** Establish an incident response plan to address system failures, unexpected issues, or security breaches promptly. Implement robust error handling mechanisms to log and report errors for analysis.
- ***Evaluation*:** Regularly review incident response procedures through simulated exercises. Analyze error logs to identify recurring issues and implement improvements to minimize system downtime.

By prioritizing system maintenance and evaluation, the MoodMatch system can ensure its long-term reliability, effectiveness, and adherence to user expectations and industry standards.

3. Detailed Life Cycle of the Project

3.1 ERD, DFD

Entity Relationship Diagram (ERD):

An Entity-Relationship Diagram (ERD) is a visual representation of the data model that depicts entities, attributes, relationships, and constraints within a system. ERDs play a pivotal role in database design and are a fundamental tool for communication between system designers and stakeholders.

Components of ERD:

1. Entities:
- Definition: Entities are real-world objects or concepts represented in the database. They can be tangible entities like a person or a book, or intangible entities like an event or a reservation.
- Symbol: Represented by rectangles in ERDs.

2. Attributes:
- Definition: Attributes are the properties or characteristics of entities. For example, a 'Person' entity might have attributes like 'Name,' 'Age,' and 'Address.'
- Symbol: Represented by ovals connected to their respective entities.

3. Relationships:
- Definition: Relationships illustrate how entities are related to each other. It signifies the associations and interactions between entities within the system.
- Symbol: Represented by diamond shapes connecting related entities.

4. Cardinality and Modality:
- Cardinality: Describes the number of instances of one entity that can be associated with another. Common terms include 'one-to-one,' 'one-to-many,' and 'many-to-many.'
- Modality: Specifies the minimum and maximum number of instances in a relationship. Modality constraints are typically expressed as (0,1), (1,1), (0,n), (1,n), where 'n' represents any positive integer.

Key Concepts:

1. Primary Key:
- Definition: A primary key uniquely identifies each record in an entity. It ensures data integrity and enables efficient data retrieval.
- Representation: Denoted by underlining the attribute in the entity.

2. Foreign Key:

- Definition: A foreign key is an attribute in one entity that refers to the primary key in another entity. It establishes a link between the two entities.
- Representation: Like a primary key, but not underlined.

Types of Relationships:

1. One-to-One (1:1):

- Definition: Each record in one entity is associated with only one record in another entity, and vice versa.
- Example: A 'Person' may have only one 'Passport,' and a 'Passport' is issued to only one 'Person.'

2. One-to-Many (1:N):

- Definition: Each record in one entity can be associated with multiple records in another entity, but each record in the second entity is associated with only one record in the first entity.
- Example: A 'Department' can have many 'Employees,' but each 'Employee' works in only one 'Department.'

3. Many-to-Many (M:N):

- Definition: Each record in both entities can be associated with multiple records in the other entity.
- Example: A 'Student' can enroll in multiple 'Courses,' and each 'Course' can have multiple 'Students.'

Benefits of ERD:

1. Clarity and Visualization:

- ERDs provide a clear and visual representation of the data model, making it easier for stakeholders to understand the structure of the database.

2. Database Design:

- ERDs serve as a blueprint for designing databases. They help in identifying entities, defining relationships, and establishing the overall structure.

3. Communication:

- ERDs facilitate communication between stakeholders, including developers, designers, and end-users. They provide a common language for discussing the data model.

4. Normalization:

- ERDs assist in the normalization process, ensuring that the database design adheres to standard normalization principles, thereby reducing data redundancy.

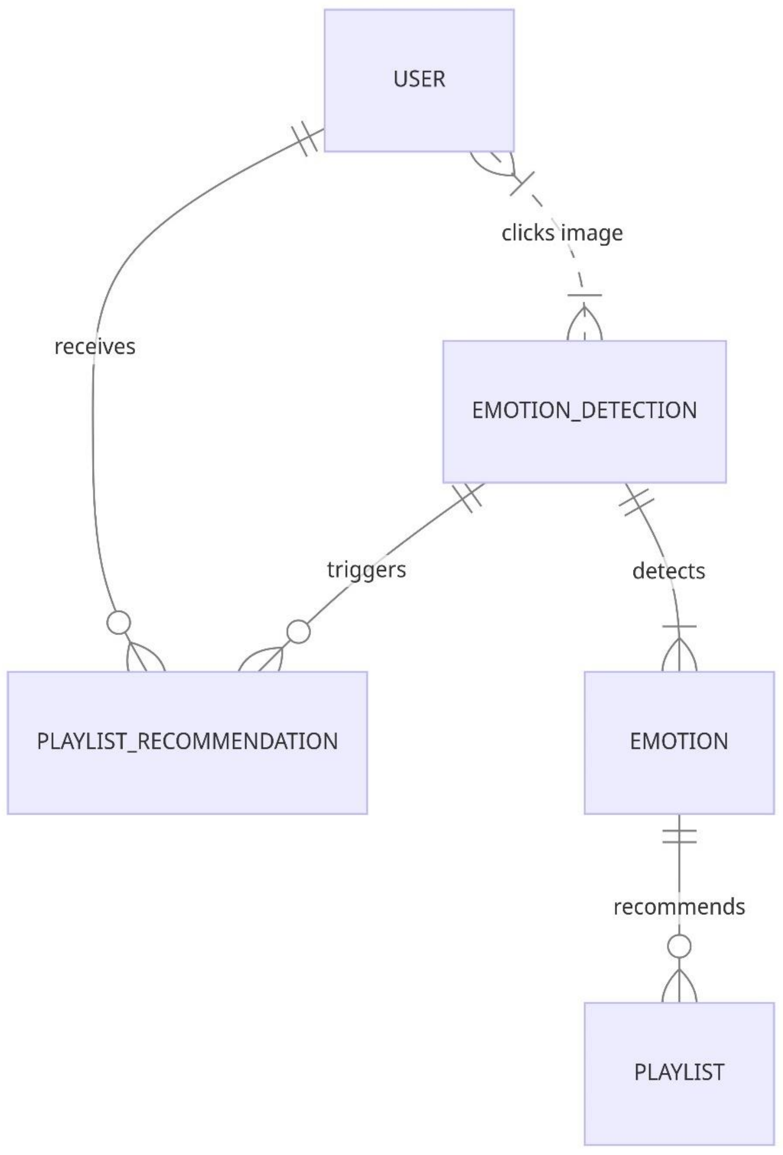


Figure 1.5 Emotion Detection and Music Recommendation - ER DIAGRAM

In the above ER diagram:

- **USER:** Represents users of the system, where their images are captured for emotion detection.
- **EMOTION_DETECTION:** Stores information about detected emotions for each user, including the detected emotion and detection time.
- **PLAYLIST_RECOMMENDATION:** Records playlist recommendations for each user based on detected emotions, along with the recommendation time.
- **EMOTION:** Defines different types of emotions detected and their corresponding playlist types.
- **PLAYLIST:** Represents playlists recommended to users based on their detected emotions.

Relationships:

- **USER --o EMOTION_DETECTION:** Each user can have multiple emotion detection instances.
- **USER --o PLAYLIST_RECOMMENDATION:** Each user can receive multiple playlist recommendations.
- **EMOTION_DETECTION --o PLAYLIST_RECOMMENDATION:** A detected emotion triggers the recommendation of a specific playlist.

Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a powerful tool in systems engineering and software development for visualizing how data moves within a system. It illustrates the flow of information between processes, data stores, external entities, and data flows. DFDs are instrumental in understanding, designing, and documenting information systems.

Components of DFD:

1. Processes:
- Definition: Processes represent activities or transformations that occur within the system. They take input data, perform specific functions, and produce output data.
- Symbol: Represented by rectangles in DFDs.

2. Data Flows:
- Definition: Data flows depict the movement of data between processes, data stores, and external entities. They represent the paths that data takes through the system.
- Symbol: Represented by arrows connecting different components.

3. Data Stores:
- Definition: Data stores are repositories where data is stored within the system. They can represent databases, files, or any other storage mechanism.
- Symbol: Represented by rectangles with two parallel lines at the bottom.

4. External Entities:
- Definition: External entities are sources or destinations of data that exist outside the system. They interact with the system by providing input or receiving output.
- Symbol: Represented by squares in DFDs.

Types of DFDs:

1. Context Diagram (Level 0 DFD):
- Scope: Provides an overview of the entire system, showing external entities, major processes, and the flow of data between them.
- Use: Offers a high-level view of the system's interactions with the external environment.

2. Level 1 DFD:
- Scope: Represents a more detailed view of a specific process from the context diagram. It decomposes processes into sub-processes and elaborates on data flows.
- Use: Provides a detailed understanding of individual processes and their relationships.

3. Level 2 DFD and Beyond:
- Scope: Continues the decomposition of processes into more detailed sub-processes. Each subsequent level provides a deeper understanding of the system.
- Use: Useful for detailed system analysis, design, and documentation

Symbols and Notations:

DFDs use specific symbols to represent different components:
- Processes: Rectangles.
- Data Flows: Arrows.
- Data Stores: Rectangles with two parallel lines at the bottom.
- External Entities: Squares.

Additionally, annotations on arrows may indicate the type or content of the data being transferred.

Benefits of DFD:

1. Clarity and Understanding:
- DFDs provide a visual representation that simplifies complex systems, making it easier for stakeholders to comprehend the flow of data.

2. Communication:
- DFDs serve as a universal language for communication between technical and non-technical stakeholders. They facilitate discussions about system functionality.

3. System Design:
- DFDs guide system design by outlining processes, data flows, and relationships. They form the foundation for subsequent design and development phases.

4. Problem Identification:

- DFDs help identify potential issues in the system, such as redundant processes or unclear data flows, aiding in problem-solving.

DFD Development Process:

1. Identify System Boundaries:

- Define the scope of the system and identify external entities.

2. Identify Processes:

- Identify major processes that transform input data into output data.

3. Define Data Flows:

- Specify the flow of data between processes, data stores, and external entities.

4. Identify Data Stores:

- Identify where data is stored within the system.

5. Refinement:

- Refine the DFD by decomposing processes into sub-processes, adding more detail as needed.

6. Validation:

- Validate the DFD with stakeholders to ensure accuracy and completeness.

CONTEXT DIAGRAM:

A context diagram, also known as a system context diagram or zero-level DFD, is a high-level visual representation that illustrates the external entities (or systems) interacting with a particular system under consideration. It provides a holistic view of the system's environment, emphasizing the interactions between the system and its external entities without delving intointernal complexities.

In a context diagram, the central system is represented as a single, prominent circle or box. This system is the primary focus of the diagram, and its boundaries are defined to encapsulate the components that are relevant to the scope of analysis. External entities, which can be other systems, users, or processes, are depicted as circles or boxes outside the central system boundary.

Arrows or lines connect these external entities to the central system, indicating the flow of data or information between them. The labels on these arrows describe the nature of the interaction, emphasizing the input and output relationships. However, the internal workings of the central system are intentionally abstracted in a context diagram, providing a simplified overview.

Context diagrams are instrumental in systems engineering and software development as they aid in clearly defining the system's boundaries, identifying key external stakeholders, and establishing the initial scope of the project. They serve as communication tools between technical and non-technical stakeholders, fostering a shared understanding of the system's role within its broader environment. As projects progress, context diagrams often evolve into more detailed representations, such as data flow diagrams (DFD) and entity-relationship diagrams (ERD), offering a layered approach to system analysis and design.

LEVEL 0 Data Flow Diagram (DFD):

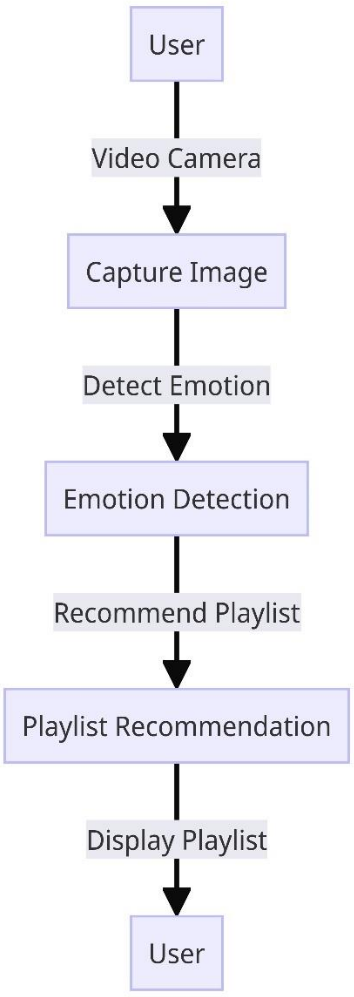


Figure 1.6 MoodMatch – LEVEL 0 DFD

In the above context diagram / Level 0 DFD:

- **User:** Represents the user of the system who initiates the process by capturing an image.
- **Emotion Detection:** Represents the process of identifying the emotion(s) present in the image using the CNN Model.
- **Playlist Recommendation:** Represents the process of selecting and recommending a playlist based on the detected emotion(s).
- **User:** Represents the user again, indicating that the recommended playlist is displayed to the user as per their detected mood.

LEVEL 1 Data Flow Diagram (DFD):

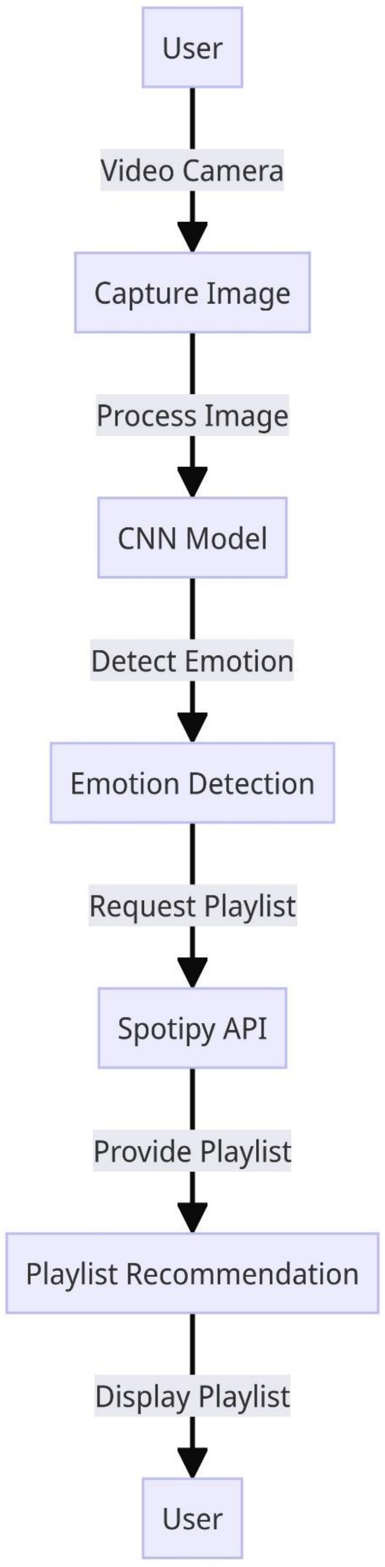


Figure 1.7 MoodMatch – LEVEL 1 DFD

In the above Level 1 DFD:

- **User:** Represents the user of the system who initiates the process by capturing an image.
- **CNN Model:** Stands for Convolutional Neural Network model, which processes the image received from the system to detect emotions present in the image.
- **Emotion Detection:** Represents the process of identifying the emotion(s) present in the image using the CNN Model.
- **Spotify API:** Refers to the API provided by Spotify, which is used to interact with Spotify's music database and retrieve playlists based on detected emotions.
- **Playlist Recommendation:** Represents the process of selecting and recommending a playlist based on the detected emotion(s).
- **User:** Represents the user again, indicating that the recommended playlist is displayed to the user as per their detected mood.

LEVEL 2 Data Flow Diagram (DFD):

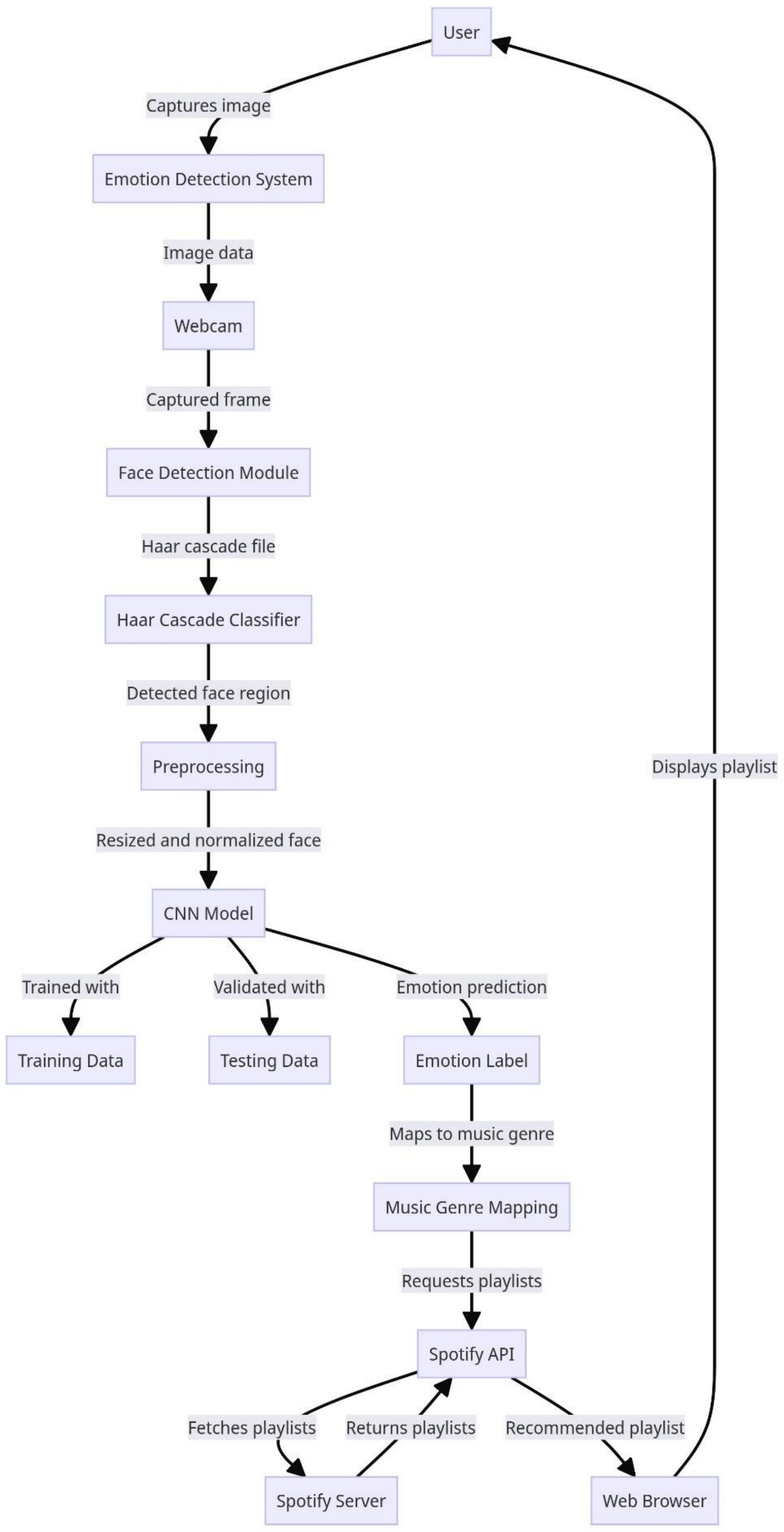


Figure 1.8 MoodMatch – LEVEL 2 DFD

In the above Level 2 DFD:

- **User:** Represents the end-user who interacts with the system by capturing an image.
- **Emotion Detection System:** The system responsible for processing the captured image to detect emotions.
- **Face Detection Module:** Part of the system that detects the face within the captured image using a Haar Cascade Classifier.
- **Haar Cascade Classifier:** A file containing patterns used for object detection, particularly frontal faces in this case.
- **CNN Model:** The Convolutional Neural Network model used for emotion recognition, trained and validated with training and testing data respectively.
- **Training Data:** Dataset used to train the CNN model, containing labeled facial images.
- **Testing Data:** Dataset used to validate the performance of the CNN model.
- **Emotion Label:** The predicted emotion label output by the CNN model.
- **Music Genre Mapping:** The mapping between predicted emotions and corresponding music genres.
- **Spotify API:** The API provided by Spotify for accessing music-related data.
- **Spotify Server:** Server hosting Spotify's music database, responsible for fetching and returning playlists.
- **Web Browser:** Represents the user interface where the recommended playlist is displayed for the user's consumption.

3.2 Input and Output Screen Design

The GUI for the input screen of the emotion detection and music recommendation system using OpenCV aims to simplify the process of collecting training data for the emotion recognition model. The interface is designed to guide users through the task of capturing and organizing facial images representing various emotions.

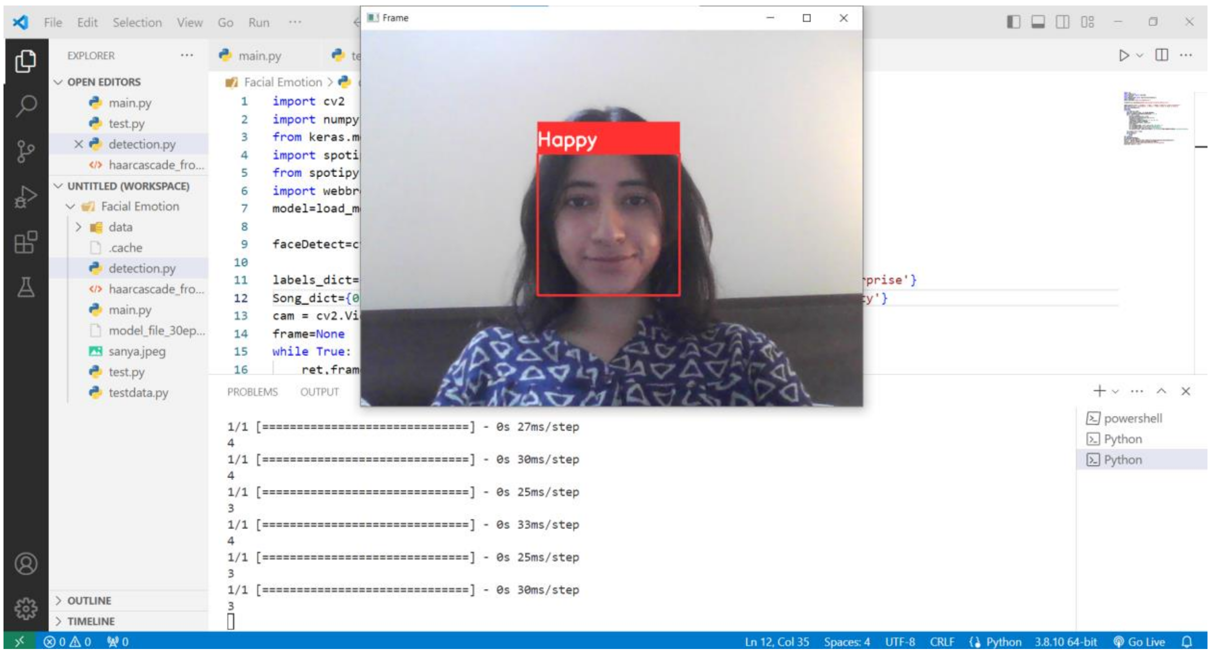
The input screen GUI features an organized layout with intuitive controls for effortless navigation. Users are presented with options to capture facial images for different emotions, such as happy, sad, angry, etc. A webcam feed is displayed prominently on the screen, allowing users to see themselves as they capture images.

The goal of the input screen GUI is to create a user-friendly environment that streamlines the process of collecting training data for the emotion recognition model, making it accessible even to users with minimal technical expertise.

The GUI for the output screen of the emotion detection and music recommendation system provides real-time feedback on the detected emotions and recommended playlists. It offers a visually engaging interface that enhances the user experience during interaction with the system. The output screen prominently displays the webcam feed, with overlaid graphics indicating the detected faces and their corresponding emotions. When a face is detected, the recognized emotion is displayed alongside the person's face, providing instant feedback on the system's performance.

In the event of a successful emotion detection, the recommended playlist corresponding to the detected emotion is displayed on the screen. Users can interact with the interface to explore different playlists and enjoy their favorite music.

Overall, the GUI design prioritizes user-friendliness, interactivity, and visual accessibility to create an intuitive environment for users interacting with the emotion detection and music recommendation system.



```
import cv2
import numpy
from keras.m
import spoti
from spotipy
import webbr
model=load_m

faceDetect=c

labels_dict=                                                    prise'}
Song_dict={0                                                    y'}
cam = cv2.Vi
frame=None
while True:
    ret,fram
```

```
1/1 [==============================] - 0s 27ms/step
4
1/1 [==============================] - 0s 30ms/step
4
1/1 [==============================] - 0s 25ms/step
3
1/1 [==============================] - 0s 33ms/step
4
1/1 [==============================] - 0s 25ms/step
3
1/1 [==============================] - 0s 30ms/step
3
```

Figure 1.9 Emotion Detection by Capturing the Image

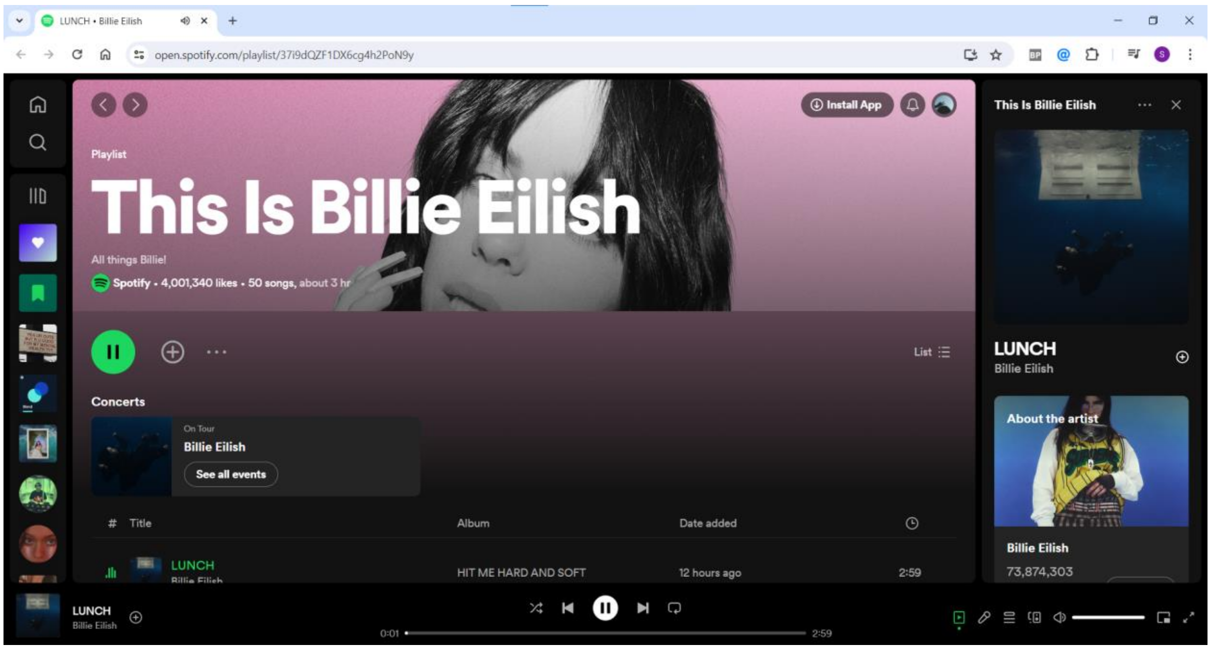


Figure 1.10 Emotion Detected – Spotify Web Browser Playlist Recommendation

3.3 Process Involved

1. **Project Setup:**

- Set up the development environment with required libraries such as OpenCV, Keras (for deep learning), and Spotipy (for Spotify integration).
- Configure the project structure and ensure necessary resources like a camera for real-time capture are accessible.

2. **Data Collection and Preprocessing:**

- Collect a dataset of facial images representing various emotions (e.g., happy, sad, angry) under different conditions.
- Preprocess the images, including resizing and normalization, to prepare them for training the emotion recognition model.

3. **Model Selection or Training:**

- Choose or train a suitable deep learning model for emotion recognition using a framework like Keras.
- Consider using pre-trained models like CNNs for image classification, fine-tuning them for emotion recognition if needed.

4. **Encoding Training Images:**

- Use the trained emotion recognition model to encode the facial features of the training images.
- Save the encoded features along with corresponding emotion labels for later reference during recognition.

5. **Project Implementation:**

- Implement the main project logic, incorporating OpenCV for real-time video capture and processing.
- Integrate the emotion recognition model to classify emotions in real-time.

6. **Real-Time Face Detection:**

- Utilize OpenCV to capture real-time video frames from the camera.
- Apply face detection algorithms (like Haar cascades) to locate and extract faces from each frame.

7. **Feature Extraction:**

- For each detected face, apply the emotion recognition model to extract emotion features.
- These features are used to determine the dominant emotion expressed by the user.

8. **Recognition and Display:**

- Map the detected emotion to a corresponding music genre using predefined mappings.
- Utilize the Spotipy API to fetch and display playlists associated with the detected emotion on the output screen.

9. **Testing and Evaluation:**

- Conduct thorough testing using diverse datasets to evaluate the system's accuracy in emotion recognition and music recommendation.
- Fine-tune parameters or consider retraining the model based on test results to improve performance.

10. **Optimization and Deployment:**

- Optimize the code and algorithms for efficient real-time processing.
- Deploy the emotion detection and music recommendation system to the desired platform, ensuring seamless integration and usability.

3.4 Methodology Used for Testing

The methodology used for testing phase of a facial recognition and detection system involves a series of steps to validate the performance of the system under various conditions.

3.4.1 Unit Testing - Training Image Preprocessing:

- Begin with unit testing to validate the preprocessing of training images for emotion recognition.
- Ensure that images are resized and normalized to enhance the diversity and quality of the training dataset.

3.4.2 Unit Testing - Face Detection:

- Conduct unit testing to verify the accuracy of face detection functionality using OpenCV.
- Evaluate the system's ability to detect and extract faces from real-time video frames under varying lighting conditions, angles, and facial expressions.

3.4.3 Integration Testing – Emotion Recognition:

- Proceed with integration testing to evaluate the entire emotion detection pipeline, including face detection and recognition based on training images.
- Verify that the system correctly identifies and classifies emotions from the detected faces, providing accurate recommendations for music playlists.

3.4.4 Integration Testing - Non-Matching Scenario:

- Test the integration of the Spotipy API for fetching and recommending playlists based on detected emotions.
- Ensure that the recommended playlists align with the detected emotions and provide relevant music choices for users.

3.4.5 Real-Time Performance Testing:

- Test the integration of the Spotipy API for fetching and recommending playlists based on detected emotions.
- Ensure that the recommended playlists align with the detected emotions and provide relevant music choices for users..

By following this comprehensive testing methodology, it's possible to systematically evaluate the emotion detection and music recommendation system, identifying potential issues and ensuring reliable and accurate performance under various conditions.

3.5 Testing and Test Results

Testing Approach

The testing approach for the emotion detection and music recommendation system involves a structured and thorough strategy to ensure its accuracy, robustness, and user-friendliness. Starting with unit testing, each component undergoes individual validation, including image preprocessing, emotion recognition, and playlist recommendation. Integration testing evaluates the entire pipeline, covering scenarios where emotions are correctly matched or not matched to training images. Real-time performance testing assesses system responsiveness, while environmental testing introduces variations to ensure reliability in diverse conditions.

3.5.1 Unit Testing - Emotion Image Preprocessing:

- Objective: Confirm that the preprocessing of emotion images accurately prepares them for emotion recognition.
- Test Result: Unit testing successfully passed, demonstrating consistent and effective preprocessing of emotion images.

3.5.2 Unit Testing – Emotion Recognition:

- Objective: Verify that the emotion recognition model correctly identifies emotions from preprocessed images.
- Test Result: Emotion recognition unit testing yielded positive results, indicating accurate classification of emotions across various facial expressions.

3.5.3 Integration Testing - Emotion Detection and Playlist Recommendation:

- Objective: Validate the integration of emotion detection and playlist recommendation, ensuring that detected emotions lead to relevant playlist suggestions.
- Test Result: Integration testing confirmed accurate playlist recommendations based on detected emotions, enhancing user experience and satisfaction.

3.5.4 Integration Testing - Non-Matching Scenario:

- Objective: Test the system's response when the detected emotion does not match any training images, verifying appropriate feedback.
- Test Result: Integration testing for non-matching scenarios passed, indicating correct behavior by providing relevant feedback without recommending playlists.

3.5.5 Real-Time Performance Testing:

- Objective: Evaluate the system's real-time performance by measuring the processing time for emotion detection and playlist recommendation.
- Test Result: Real-time performance testing demonstrated efficient processing with satisfactory responsiveness, meeting user expectations for seamless interaction.

In summary, the testing phase identified strengths and areas for improvement in the emotion detection and music recommendation system, ensuring its reliability and effectiveness in real-world usage scenarios.

4. CODING & SCREENSHOTS OF THE PROJECT

- **main.py**

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten
from keras.layers import Conv2D,MaxPooling2D
import os


train_data_dir='data/train/'
validation_data_dir='data/test/'


train_datagen = ImageDataGenerator(
                         rescale=1./255,
                         rotation_range=30,
                         shear_range=0.3,
                         zoom_range=0.3,
                         horizontal_flip=True,
                         fill_mode='nearest')

validation_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
                         train_data_dir,
                         color_mode='grayscale',
                         target_size=(48, 48),
                         batch_size=32,
                         class_mode='categorical',
                         shuffle=True)

validation_generator = validation_datagen.flow_from_directory(
                                   validation_data_dir,
                                   color_mode='grayscale',
                                   target_size=(48, 48),
                                   batch_size=32,
                                   class_mode='categorical',
                                   shuffle=True)


class_labels=['Angry','Disgust', 'Fear',
'Happy','Neutral','Sad','Surprise']

img, label = train_generator.__next__()


model = Sequential()
```

```
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(48,48,1)))

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Conv2D(256, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(7, activation='softmax'))

model.compile(optimizer = 'adam',
loss='categorical_crossentropy', metrics=['accuracy'])
print(model.summary())


train_path = "data/train/"
test_path = "data/test"

num_train_imgs = 0
for root, dirs, files in os.walk(train_path):
    num_train_imgs += len(files)

num_test_imgs = 0
for root, dirs, files in os.walk(test_path):
    num_test_imgs += len(files)

print(num_train_imgs)
print(num_test_imgs)
epochs=30

history=model.fit(train_generator,
                steps_per_epoch=num_train_imgs//32,
                epochs=epochs,
                validation_data=validation_generator,
                validation_steps=num_test_imgs//32)

model.save('model_file.h5')
```

- **testdata.py**

```
import cv2
import numpy as np
from keras.models import load_model

model=load_model('model_file_30epochs.h5')
faceDetect=cv2.CascadeClassifier('haarcascade_frontalface_default
.xml')
labels_dict={0:'Angry',1:'Disgust', 2:'Fear',
3:'Happy',4:'Neutral',5:'Sad',6:'Surprise'}

# len(number_of_image), image_height, image_width, channel
frame=cv2.imread("sanya.jpeg")
gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
faces= faceDetect.detectMultiScale(gray, 1.3, 3)
for x,y,w,h in faces:
    sub_face_img=gray[y:y+h, x:x+w]
    resized=cv2.resize(sub_face_img,(48,48))
    normalize=resized/255.0
    reshaped=np.reshape(normalize, (1, 48, 48, 1))
    result=model.predict(reshaped)
    label=np.argmax(result, axis=1)[0]
    print(label)
    cv2.rectangle(frame, (x,y), (x+w, y+h), (0,0,255), 1)
    cv2.rectangle(frame,(x,y),(x+w,y+h),(50,50,255),2)
    cv2.rectangle(frame,(x,y-40),(x+w,y),(50,50,255),-1)
    cv2.putText(frame, labels_dict[label], (x, y
10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)

cv2.imshow("Frame",frame)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

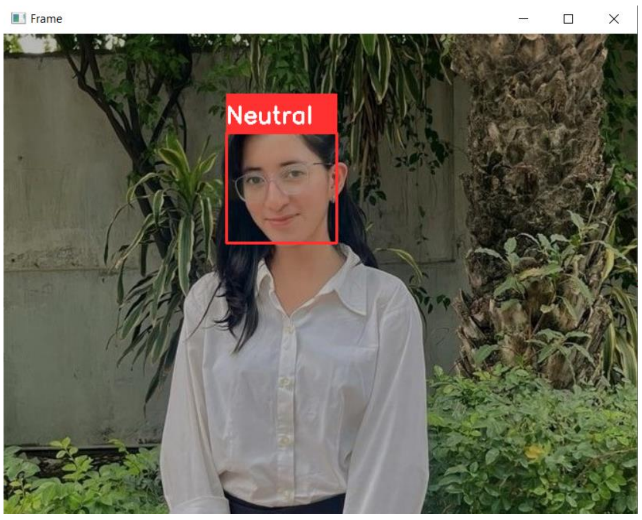
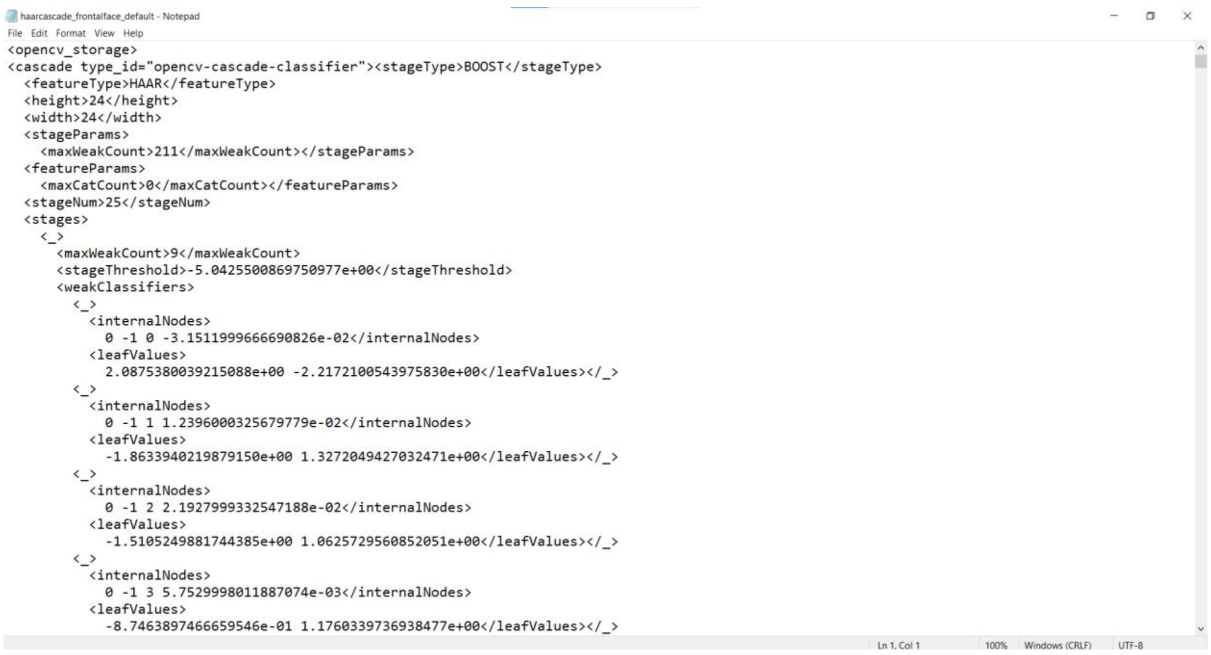


Figure 1.11 Testing Emotion Detection on an Image

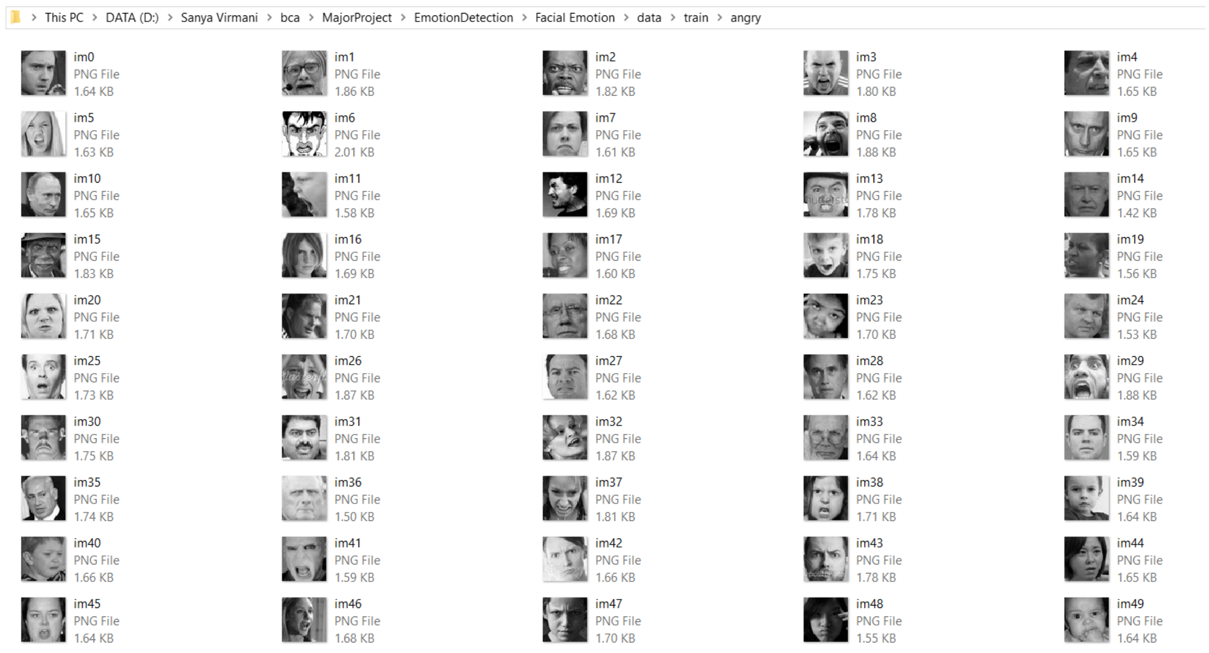
- **haarcascade_frontalface_default.xml**


haarcascade_frontalface_default - Notepad
File Edit Format View Help
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8

```
<opencv_storage>
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>24</height>
  <width>24</width>
  <stageParams>
    <maxWeakCount>211</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>25</stageNum>
  <stages>
    <_>
      <maxWeakCount>9</maxWeakCount>
      <stageThreshold>-5.0425500869750977e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 -3.1511999666690826e-02</internalNodes>
          <leafValues>
            2.0875380039215088e+00 -2.2172100543975830e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 1 1.2396000325679779e-02</internalNodes>
          <leafValues>
            -1.8633940219879150e+00 1.3272049427032471e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 2 2.1927999332547188e-02</internalNodes>
          <leafValues>
            -1.5105249881744385e+00 1.0625729560852051e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 3 5.7529998011887074e-03</internalNodes>
          <leafValues>
            -8.7463897466659546e-01 1.1760339736938477e+00</leafValues></_>
```

Figure 1.12 Pre-Trained model xml

- **data (training and testing images)**


This PC > DATA (D:) > Sanya Virmani > bca > MajorProject > EmotionDetection > Facial Emotion > data > train > angry

im0 PNG File 1.64 KB	im1 PNG File 1.86 KB	im2 PNG File 1.82 KB	im3 PNG File 1.80 KB	im4 PNG File 1.65 KB
im5 PNG File 1.63 KB	im6 PNG File 2.01 KB	im7 PNG File 1.61 KB	im8 PNG File 1.88 KB	im9 PNG File 1.65 KB
im10 PNG File 1.65 KB	im11 PNG File 1.58 KB	im12 PNG File 1.69 KB	im13 PNG File 1.78 KB	im14 PNG File 1.42 KB
im15 PNG File 1.83 KB	im16 PNG File 1.69 KB	im17 PNG File 1.60 KB	im18 PNG File 1.75 KB	im19 PNG File 1.56 KB
im20 PNG File 1.71 KB	im21 PNG File 1.70 KB	im22 PNG File 1.68 KB	im23 PNG File 1.70 KB	im24 PNG File 1.53 KB
im25 PNG File 1.73 KB	im26 PNG File 1.87 KB	im27 PNG File 1.62 KB	im28 PNG File 1.62 KB	im29 PNG File 1.88 KB
im30 PNG File 1.75 KB	im31 PNG File 1.81 KB	im32 PNG File 1.87 KB	im33 PNG File 1.64 KB	im34 PNG File 1.59 KB
im35 PNG File 1.74 KB	im36 PNG File 1.50 KB	im37 PNG File 1.81 KB	im38 PNG File 1.71 KB	im39 PNG File 1.64 KB
im40 PNG File 1.66 KB	im41 PNG File 1.59 KB	im42 PNG File 1.66 KB	im43 PNG File 1.78 KB	im44 PNG File 1.65 KB
im45 PNG File 1.64 KB	im46 PNG File 1.68 KB	im47 PNG File 1.70 KB	im48 PNG File 1.55 KB	im49 PNG File 1.64 KB

Figure 1.13 Training Images for each emotion

- **detection.py**

```
import cv2
import numpy as np
from keras.models import load_model
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
import webbrowser
model=load_model('model_file_30epochs.h5')

faceDetect=cv2.CascadeClassifier('haarcascade_frontalface_defaul
t.xml')

labels_dict={0:'Angry',1:'Disgust', 2:'Fear',
3:'Happy',4:'Neutral',5:'Sad',6:'Surprise'}
Song_dict={0:'Rock',1:'R&B', 2:'Metal',
3:'Pop',4:'Chill',5:'Mood',6:'Party'}
cam = cv2.VideoCapture(0)
frame=None
while True:
    ret,frame=cam.read()
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces= faceDetect.detectMultiScale(gray, 1.3, 3)
    for x,y,w,h in faces:
        sub_face_img=gray[y:y+h, x:x+w]
        resized=cv2.resize(sub_face_img,(48,48))
        normalize=resized/255.0
        reshaped=np.reshape(normalize, (1, 48, 48, 1))
        result=model.predict(reshaped)
        label=np.argmax(result, axis=1)[0]
        print(label)
        cv2.rectangle(frame, (x,y), (x+w, y+h), (0,0,255), 1)
        cv2.rectangle(frame,(x,y),(x+w,y+h),(50,50,255),2)
        cv2.rectangle(frame,(x,y-40),(x+w,y),(50,50,255),-1)
        cv2.putText(frame, labels_dict[label], (x, y-
10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)

    cv2.imshow("Frame",frame)
    k=cv2.waitKey(1)
    if k==ord('c'):
        break
cam.release()
cv2.destroyAllWindows()
print(labels_dict[label])
spotify =
spotipy.Spotify(client_credentials_manager=SpotifyClientCredenti
als())
results = spotify.category_playlists(Song_dict[label])
url=results["playlists"]["items"][0]["external_urls"]["spotify"]
webbrowser.open(url, new=2)
```

OUTPUT:

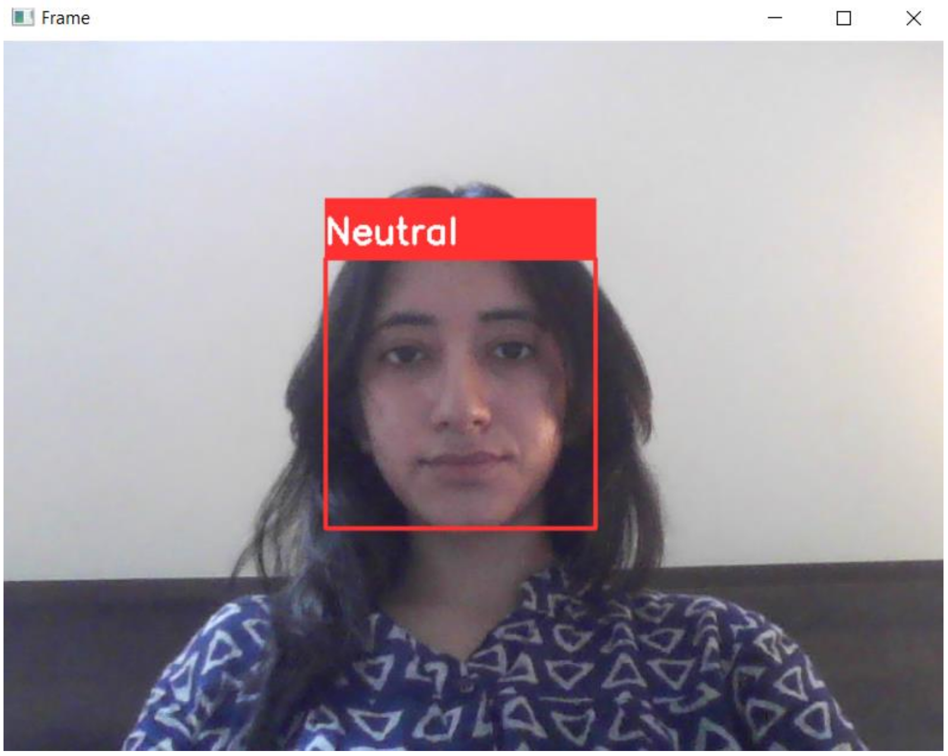
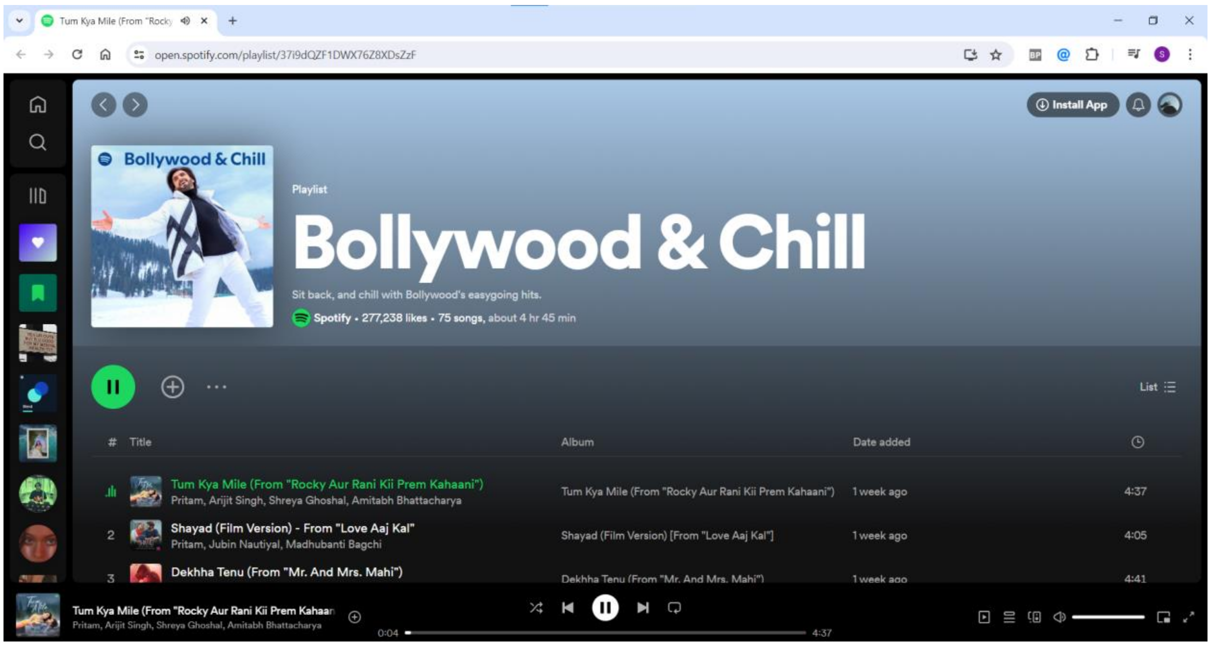


Figure 1.14 Neutral Emotion Captured by OpenCV


Tum Kya Mile (From "Rocky

open.spotify.com/playlist/37i9dQZF1DWX76Z8XDsZzF

Install App

Bollywood & Chill

Playlist

Bollywood & Chill

Sit back, and chill with Bollywood's easygoing hits.

Spotify • 277,238 likes • 75 songs, about 4 hr 45 min

List

#	Title	Album	Date added	
	Tum Kya Mile (From "Rocky Aur Rani Kii Prem Kahaani") Pritam, Arijit Singh, Shreya Ghoshal, Amitabh Bhattacharya	Tum Kya Mile (From "Rocky Aur Rani Kii Prem Kahaani")	1 week ago	4:37
2	Shayad (Film Version) - From "Love Aaj Kal" Pritam, Jubin Nautiyal, Madhubanti Bagchi	Shayad (Film Version) [From "Love Aaj Kal"]	1 week ago	4:05
3	Dekhha Tenu (From "Mr. And Mrs. Mahi")	Dekhha Tenu (From "Mr. And Mrs. Mahi")	1 week ago	4:41

Tum Kya Mile (From "Rocky Aur Rani Kii Prem Kahaan
Pritam, Arijit Singh, Shreya Ghoshal, Amitabh Bhattacharya
0:04 4:37

Figure 1.15 NEUTRAL Emotion- Playlist Recommended through Spotify Web Browser

```
-
1/1 [==============================] - 0s 44ms/step
4
Neutral
```

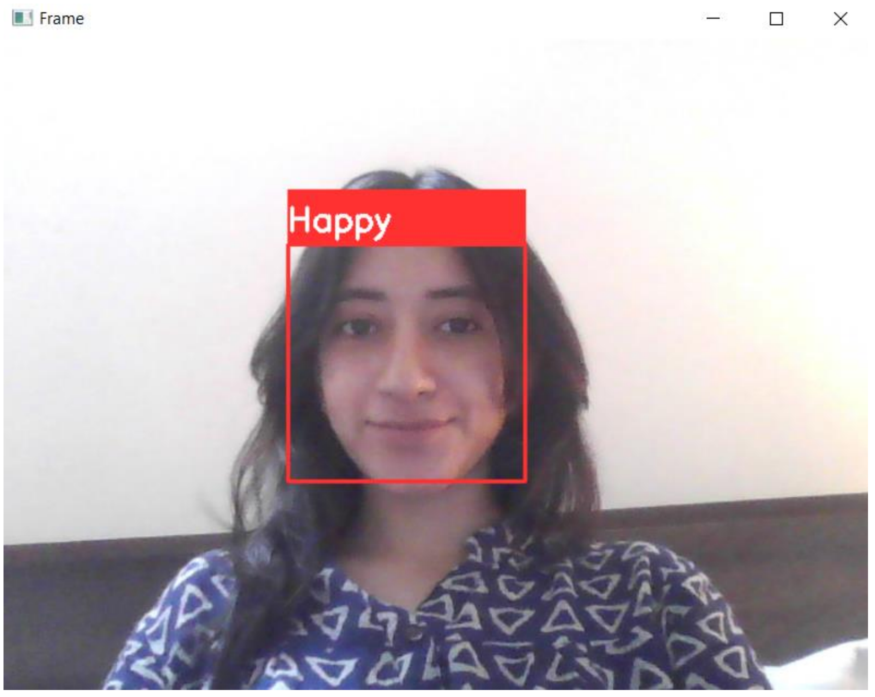


Figure 1.16 Happy Emotion Captured by OpenCV

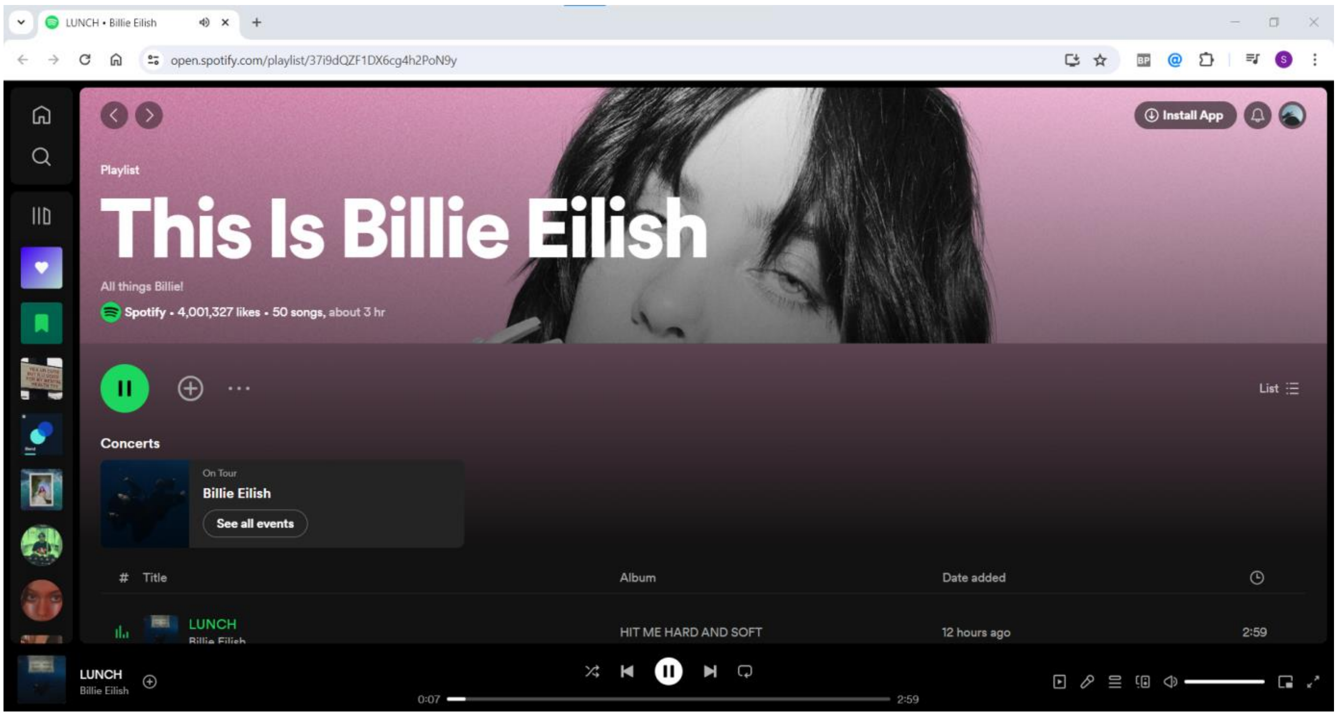


Figure 1.17 HAPPY Emotion- Playlist Recommended through Spotify Web Browser

```
1/1 [==============================] - 0s 30ms/step
3
Happy
```

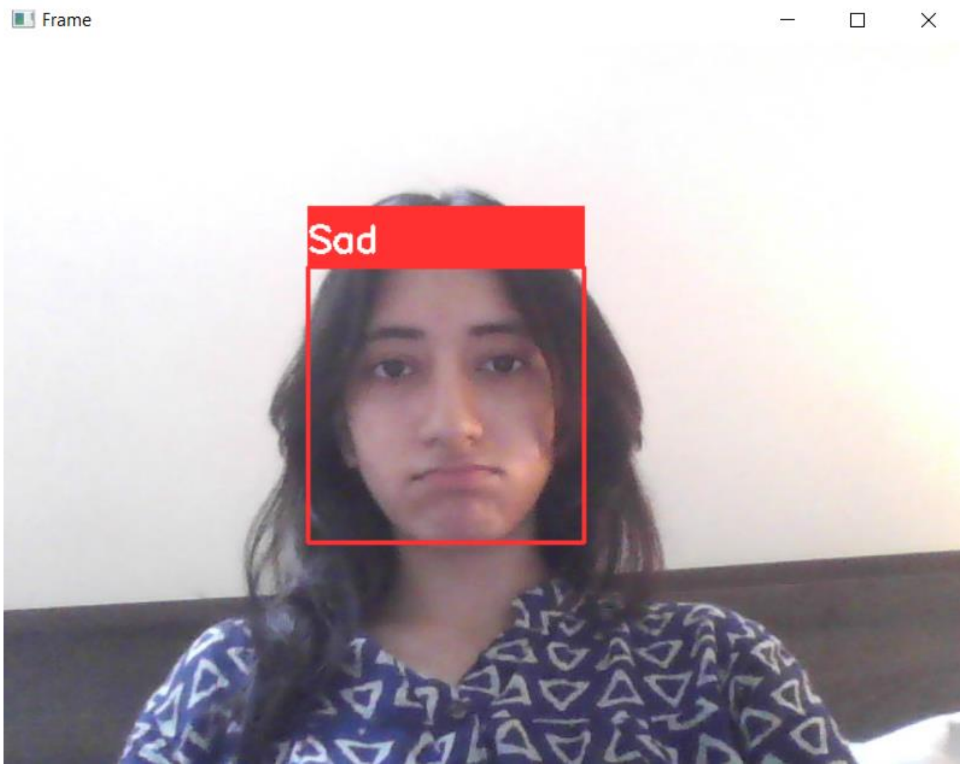
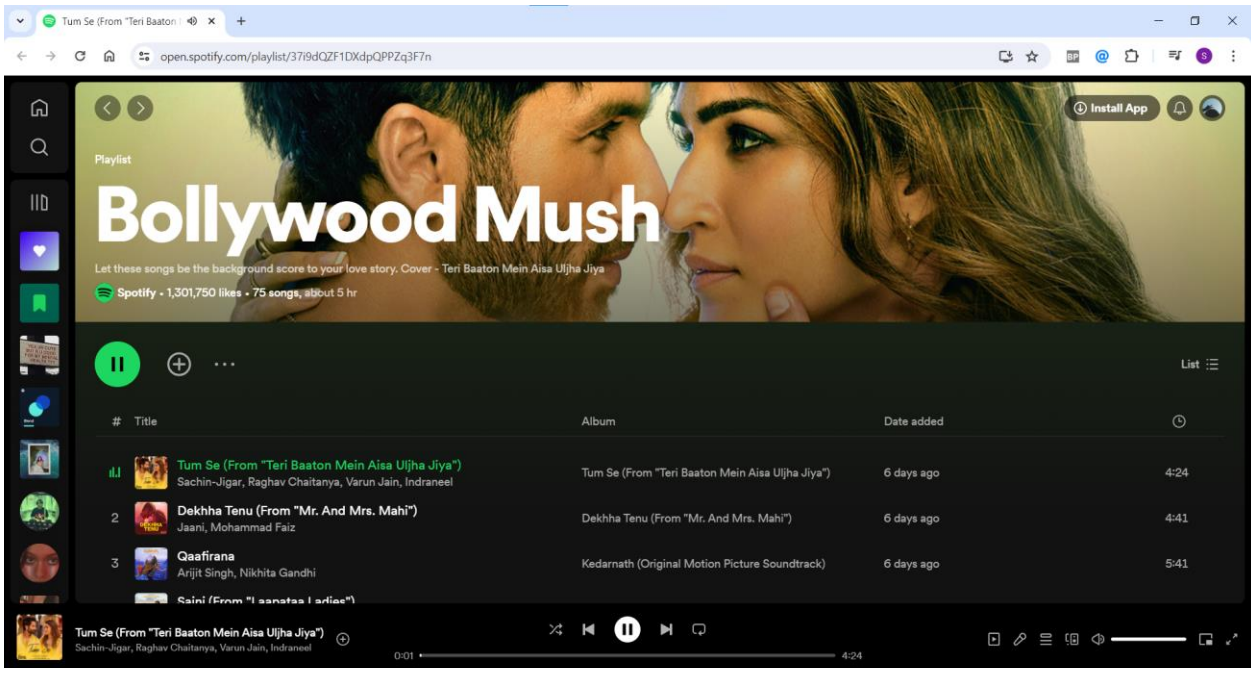


Figure 1.18 Sad Emotion Captured by OpenCV


Tum Se (From "Teri Baaton
open.spotify.com/playlist/37i9dQZF1DXdpQPPZq3F7n
Install App
Playlist
List

Bollywood Mush

Let these songs be the background score to your love story. Cover - Teri Baaton Mein Aisa Uljha Jiya

Spotify • 1,301,750 likes • 75 songs, about 5 hr

#	Title	Album	Date added	🕒
	Tum Se (From "Teri Baaton Mein Aisa Uljha Jiya") Sachin-Jigar, Raghav Chaitanya, Varun Jain, Indraneel	Tum Se (From "Teri Baaton Mein Aisa Uljha Jiya")	6 days ago	4:24
2	Dekhha Tenu (From "Mr. And Mrs. Mahi") Jaani, Mohammad Faiz	Dekhha Tenu (From "Mr. And Mrs. Mahi")	6 days ago	4:41
3	Qaafirana Arijit Singh, Nikhita Gandhi	Kedarnath (Original Motion Picture Soundtrack)	6 days ago	5:41
	Saini (From "Laapataa Ladies")			

Tum Se (From "Teri Baaton Mein Aisa Uljha Jiya")
Sachin-Jigar, Raghav Chaitanya, Varun Jain, Indraneel
0:01 4:24

Figure 1.19 SAD Emotion- Playlist Recommended through Spotify Web Browser

```
1/1 [==============================] - 0s 21ms/step
5
Sad
```

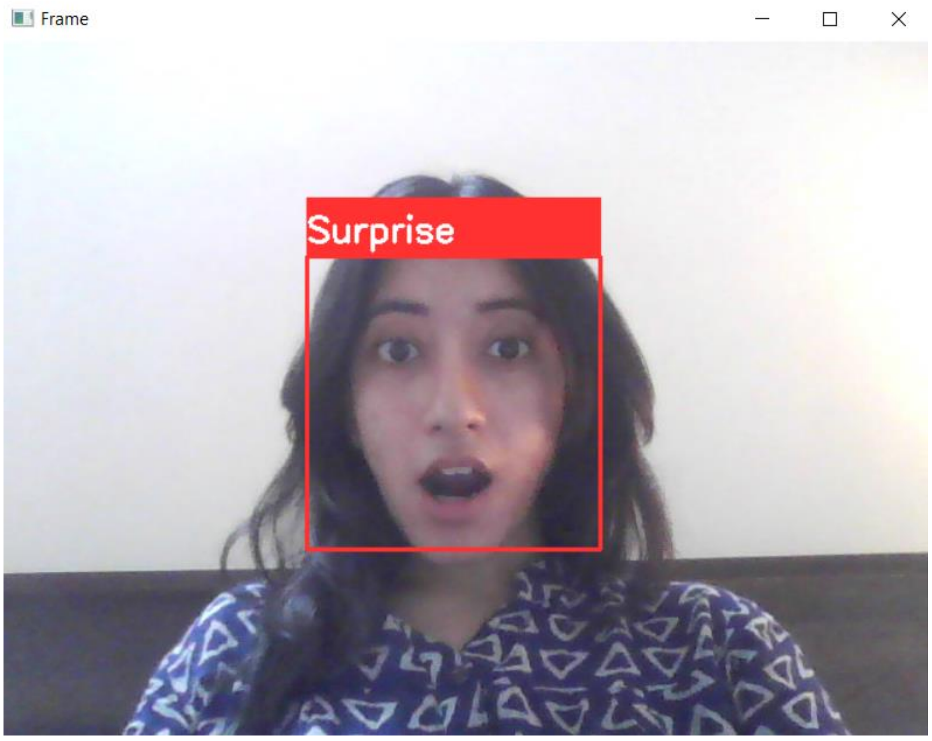
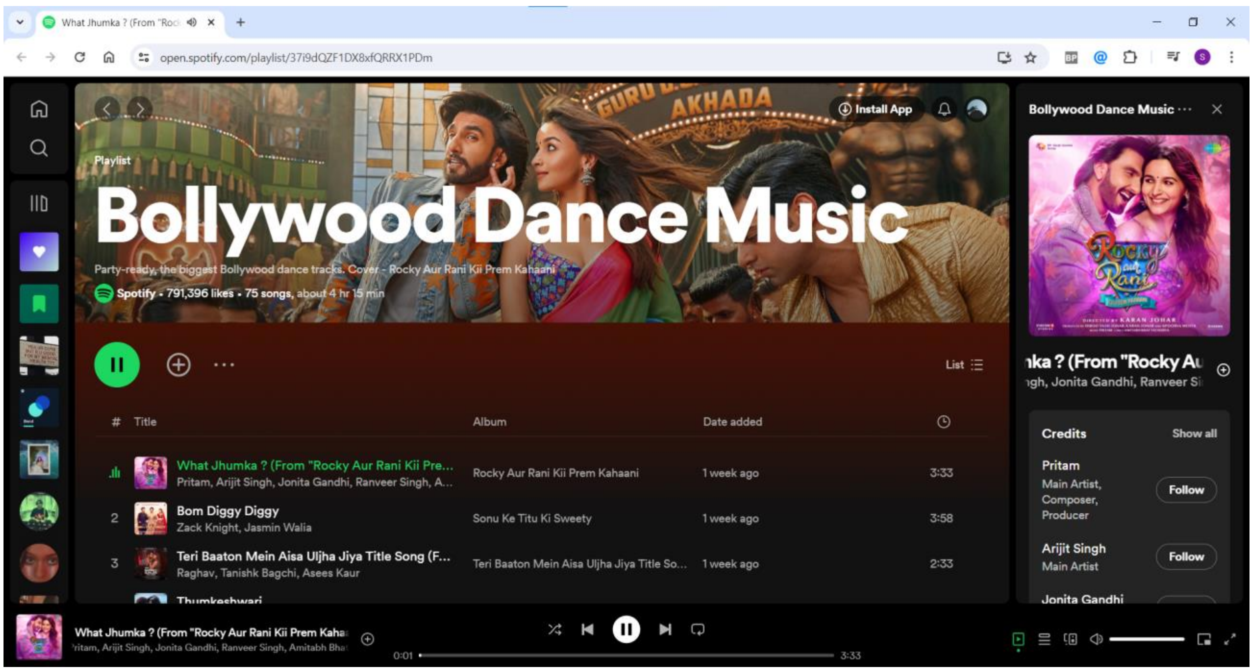


Figure 1.20 Surprise Emotion Captured by OpenCV


What Jhumka ? (From "Rock
open.spotify.com/playlist/37i9dQZF1DX8xfQRRX1PDm
Install App
Bollywood Dance Music
Playlist
List
Credits
Show all
Follow

Bollywood Dance Music

Party-ready, the biggest Bollywood dance tracks. Cover - Rocky Aur Rani Kii Prem Kahaani

Spotify • 791,396 likes • 75 songs, about 4 hr 15 min

#	Title	Album	Date added	🕒
	What Jhumka ? (From "Rocky Aur Rani Kii Pre... Pritam, Arijit Singh, Jonita Gandhi, Ranveer Singh, A...	Rocky Aur Rani Kii Prem Kahaani	1 week ago	3:33
2	Bom Diggy Diggy Zack Knight, Jasmin Walia	Sonu Ke Titu Ki Sweety	1 week ago	3:58
3	Teri Baaton Mein Aisa Uljha Jiya Title Song (F... Raghav, Tanishk Bagchi, Asees Kaur	Teri Baaton Mein Aisa Uljha Jiya Title So...	1 week ago	2:33
	Thumkeshwari			

ıka ? (From "Rocky Au
ıgh, Jonita Gandhi, Ranveer Si

Pritam
Main Artist, Composer, Producer

Arijit Singh
Main Artist

Jonita Gandhi

What Jhumka ? (From "Rocky Aur Rani Kii Prem Kaha
'ritam, Arijit Singh, Jonita Gandhi, Ranveer Singh, Amitabh Bha

0:01 3:33

Figure 1.21 SURPRISE Emotion- Playlist Recommended through Spotify Web Browser

```
1/1 [==============================] - 0s 20ms/step
6
Surprise
```

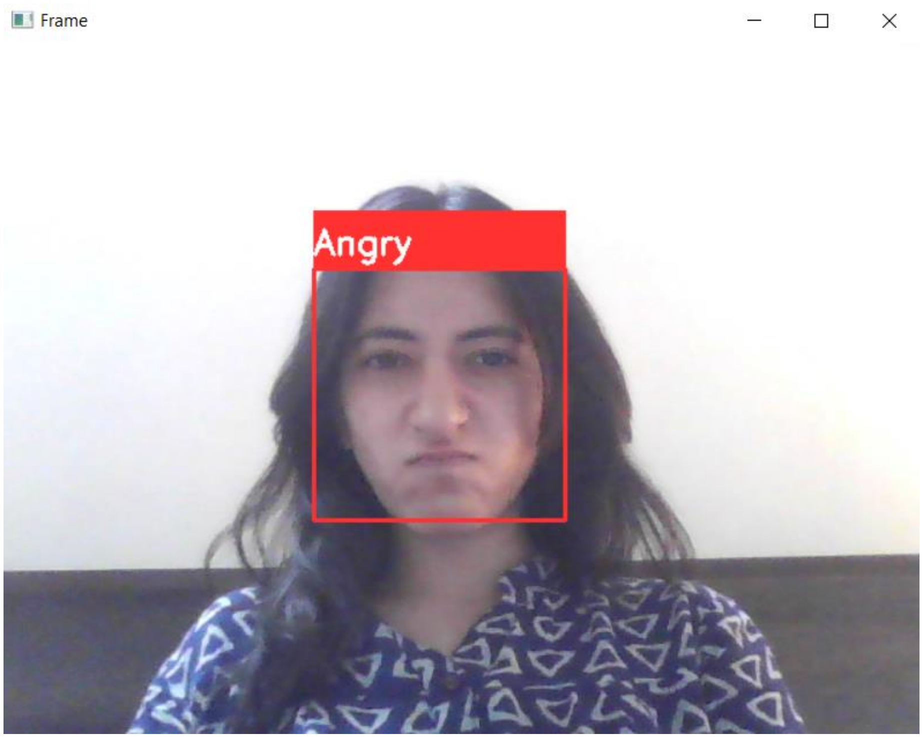
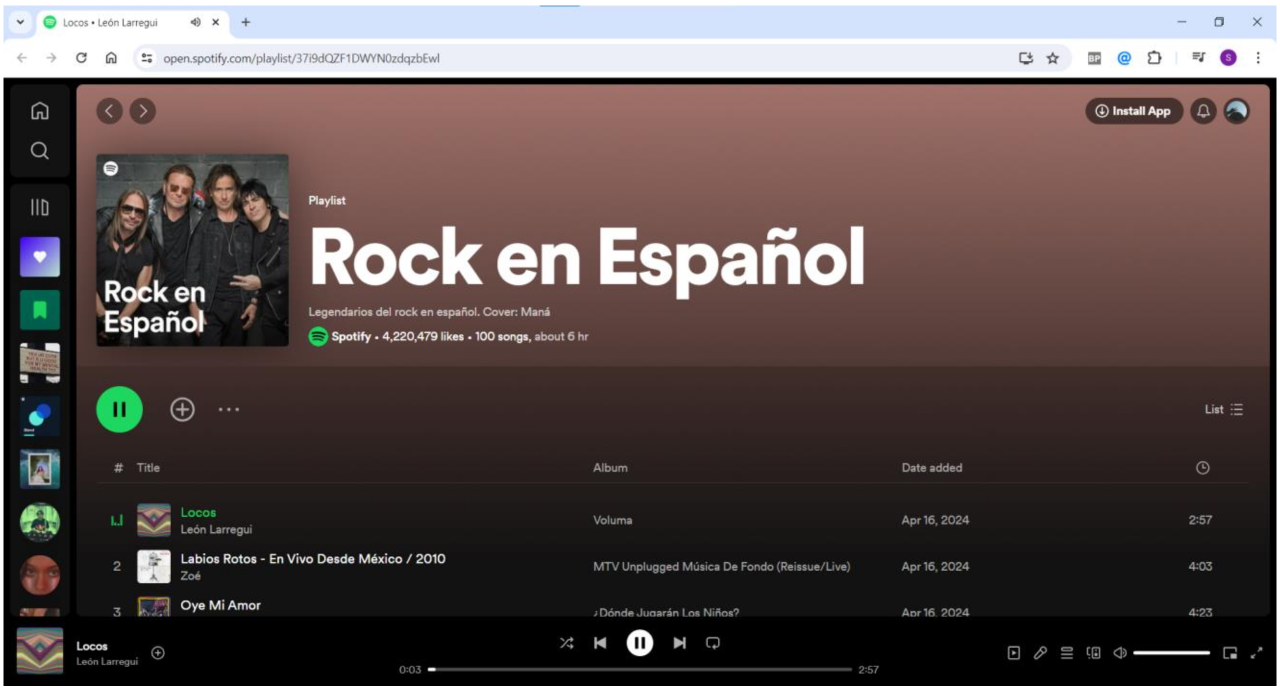


Figure 1.22 Angry Emotion Captured by OpenCV



Playlist

Rock en Español

Legendarios del rock en español. Cover: Maná

Spotify • 4,220,479 likes • 100 songs, about 6 hr

#	Title	Album	Date added	🕒
	Locos León Larregui	Voluma	Apr 16, 2024	2:57
2	Labios Rotos - En Vivo Desde México / 2010 Zoé	MTV Unplugged Música De Fondo (Reissue/Live)	Apr 16, 2024	4:03
3	Oye Mi Amor	¿Dónde Jugarán Los Niños?	Apr 16, 2024	4:23

Locos
León Larregui
0:03 2:57

Figure 1.23 ANGRY Emotion- Playlist Recommended through Spotify Web Browser

```
1/1 [==============================] - 0s 36ms/step
0
Angry
```

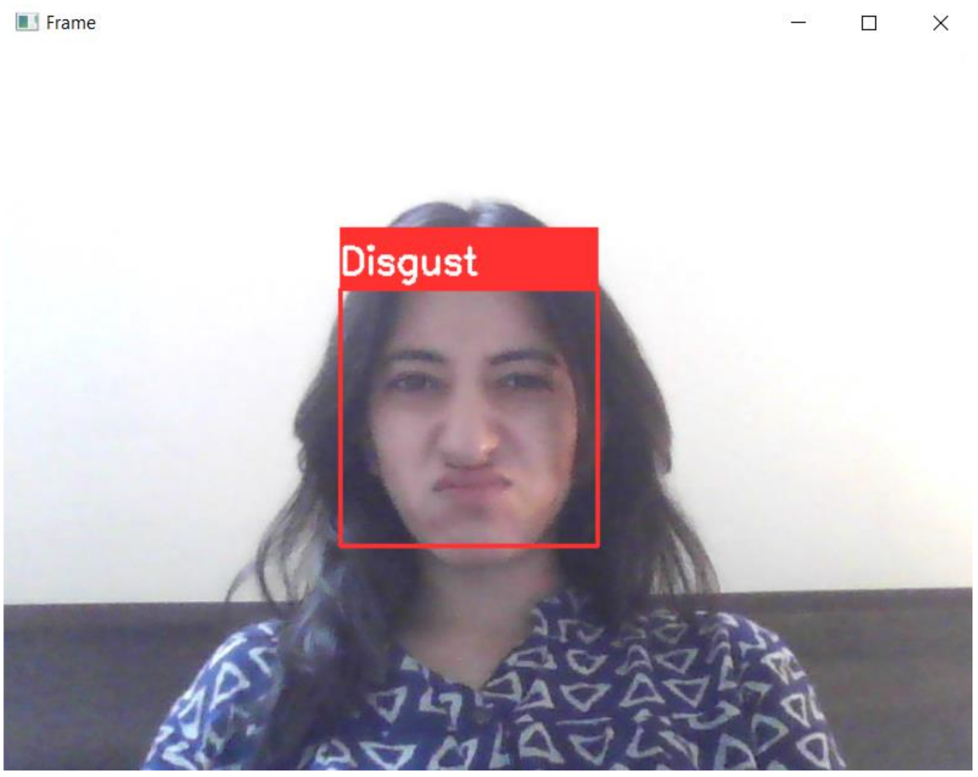
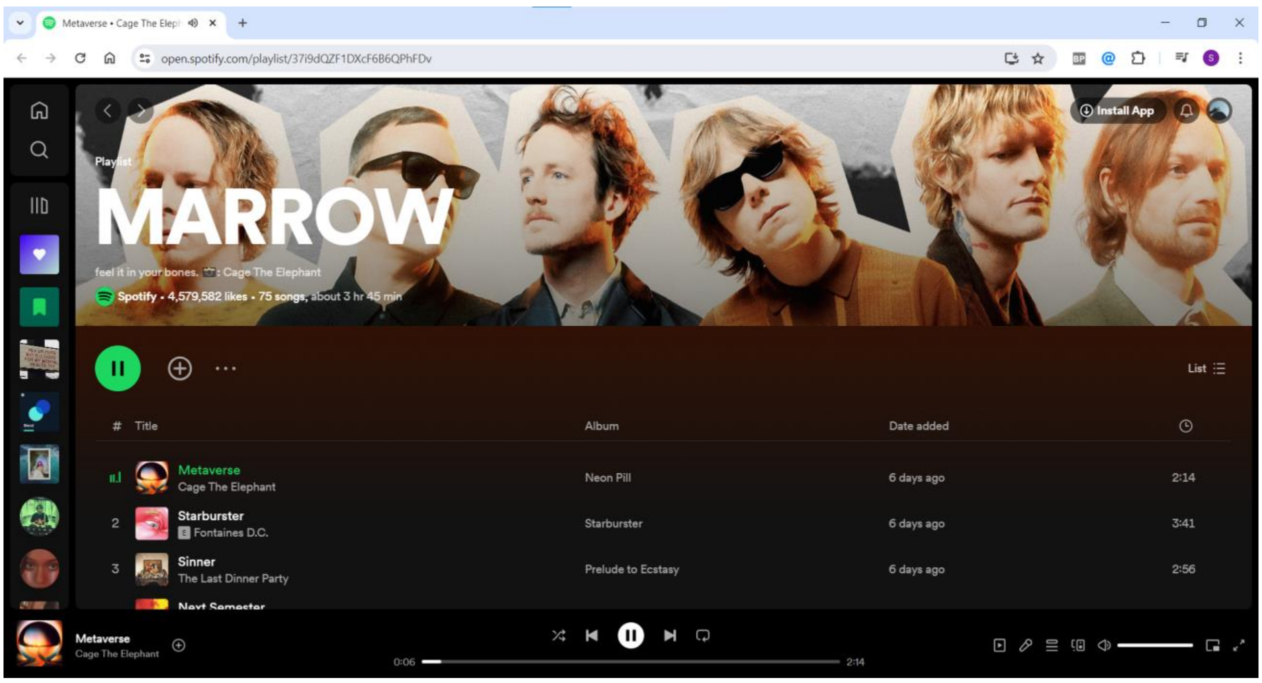


Figure 1.24 Disgust Emotion Captured by OpenCV


Metaverse • Cage The Elep
open.spotify.com/playlist/37i9dQZF1DXcF6B6QPhFDv
Install App
Playlist
MARROW
feel it in your bones. : Cage The Elephant
Spotify • 4,579,582 likes • 75 songs, about 3 hr 45 min
List
Metaverse
Cage The Elephant
0:06
2:14

#	Title	Album	Date added	
	Metaverse Cage The Elephant	Neon Pill	6 days ago	2:14
2	Starburster E Fontaines D.C.	Starburster	6 days ago	3:41
3	Sinner The Last Dinner Party	Prelude to Ecstasy	6 days ago	2:56
	Next Semester			

Figure 1.25 DISGUST Emotion- Playlist Recommended through Spotify Web Browser

```
1/1 [==============================] - 0s 31ms/step
1
Disgust
```

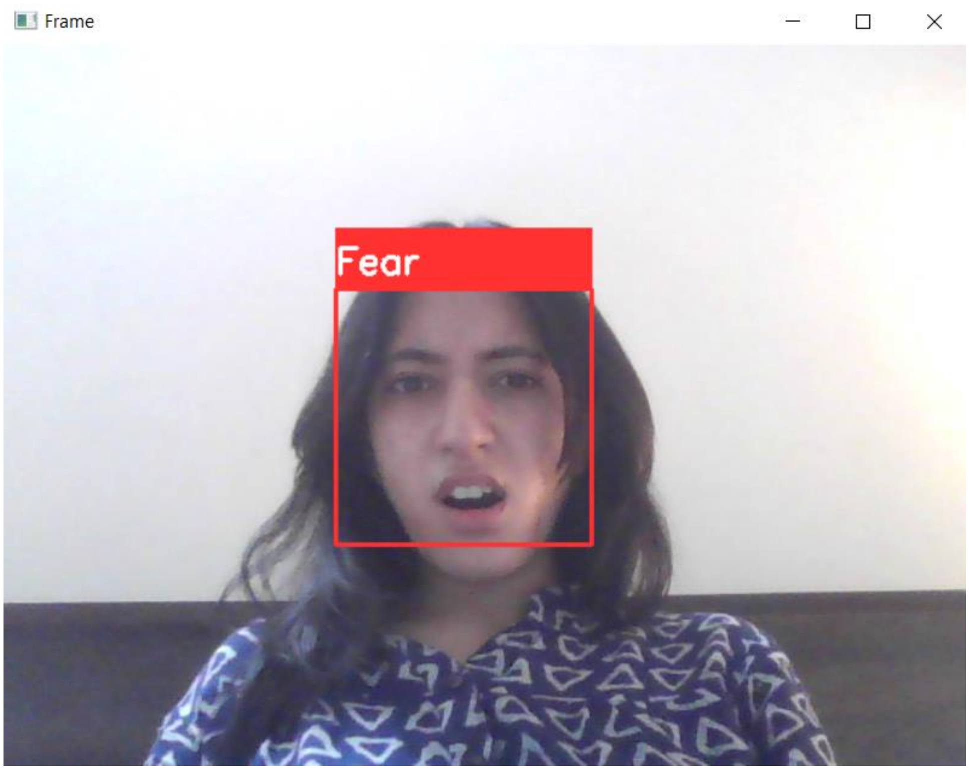
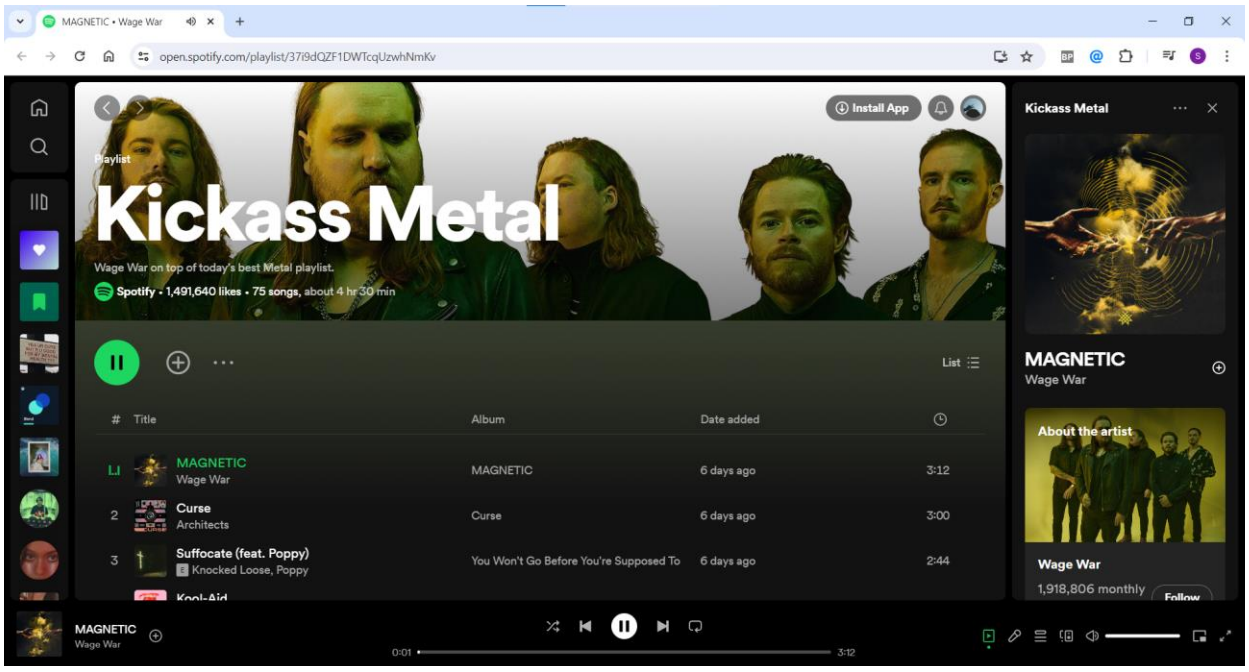


Figure 1.26 Fear Emotion Captured by OpenCV



#	Title	Album	Date added	
	MAGNETIC Wage War	MAGNETIC	6 days ago	3:12
2	Curse Architects	Curse	6 days ago	3:00
3	Suffocate (feat. Poppy) E Knocked Loose, Poppy	You Won't Go Before You're Supposed To	6 days ago	2:44
	Kool-Aid			

Figure 1.27 FEAR Emotion- Playlist Recommended through Spotify Web Browser

```
1/1 [==============================] - 0s 22ms/step
2
Fear
```

5. CONCLUSION

The development and testing of MoodMatch, an emotion detection and music recommendation system, have been a significant endeavor aimed at enhancing user experience and satisfaction. Through a systematic approach, we have successfully implemented a solution that leverages facial recognition technology to detect users' emotions in real-time and recommend music playlists tailored to their mood.

The project began with meticulous planning and setup, establishing the development environment and configuring the project structure to ensure seamless integration of essential libraries such as OpenCV for image processing, Keras for deep learning-based emotion recognition, and Spotipy for accessing Spotify's vast music database. This groundwork laid the foundation for the subsequent phases of data collection, model training, and system implementation.

MoodMatch offers users an intuitive and engaging interface, allowing them to explore music playlists curated specifically to match their current emotional state. By seamlessly integrating facial recognition technology with music recommendation algorithms, the system provides a personalized and immersive experience, enriching users' lives through the power of music.

Looking ahead, continuous refinement and optimization will be essential to further enhance MoodMatch's capabilities and ensure its continued success. Future iterations may include improvements in emotion detection accuracy, expansion of music genre recommendations, and integration with additional platforms for a more comprehensive music discovery experience.

Overall, MoodMatch represents a significant step forward in leveraging artificial intelligence and machine learning to create innovative solutions that cater to users' emotional needs. As technology continues to evolve, MoodMatch stands poised to redefine the way we interact with music, offering a unique and enriching journey through the ever-changing landscape of human emotions.

6. FUTURE SCOPE

The future scope of facial recognition and detection systems holds immense potential for innovation and expansion into various domains. As technology evolves, these systems are poised to become more sophisticated, versatile, and capable of addressing new challenges. One promising avenue for future development involves the integration of emotion detection to enhance user interaction and customization. By deciphering facial expressions, the system can gain insights into users' moods, opening up opportunities for personalized experiences, such as suggesting music based on emotional states.

1) Integration with Wearable Devices:

Develop compatibility with wearable devices such as smartwatches and fitness trackers to gather physiological data (e.g., heart rate, body temperature) that can complement facial emotion detection for a more comprehensive understanding of the user's emotional state.

2) Mental Health Applications:

Expand the system's functionality to support mental health monitoring and interventions. By tracking emotional trends over time and providing insights, the system could serve as a tool for users and healthcare professionals to manage and understand mental health better.

3) Social Media Integration:

Integrate with social media platforms to understand the user's emotional context based on their posts, messages, and interactions, offering more contextually relevant music recommendations.

4) Real-Time Feedback and Interaction:

Introducing features for real-time feedback and interaction, such as allowing users to provide feedback on recommended playlists or dynamically adjusting recommendations based on users' responses.

5) **Global Expansion and Cultural Adaptation:**

   Adapt the system for use in different cultural contexts by incorporating culturally specific emotional expressions and music preferences, enabling a globally inclusive user experience.

6) **Personalized Playlist Recommendation:**

   Implementing advanced machine learning techniques to tailor playlist recommendations based not only on detected emotions but also on users' individual preferences, listening history, and contextual factors.

7) **Multi-User Support:**

   Extending the system to support multiple users simultaneously, enabling seamless interaction and personalized recommendations for each individual within a shared environment.

8) **Localized Recommendations:**

   Customized recommendations based on regional or cultural preferences, ensuring that users receive music suggestions that resonate with their specific cultural background and music tastes.

MoodMatch has the potential to evolve into a comprehensive and indispensable tool for enhancing users' emotional well-being and enriching their music listening experiences. Through ongoing research, development, and collaboration, it can continue to push the boundaries of emotion detection and music recommendation technologies, ultimately reshaping the way we interact with and experience music in our daily lives.

.

7. REFERENCES

References for MoodMatch- Emotion Detection and Music Recommendation System Project include the official documentation for:

- Python [https://docs.python.org/]
- Comprehensive Guide to CNN with Keras on Kaggle [https://www.kaggle.com/code/prashant111/comprehensive-guide-to-cnn-with-keras]
- Keras Applications [https://keras.io/api/applications/]
- OpenCV [https://docs.opencv.org/4.x/]
- Emotion Detection [https://www.sciencedirect.com/topics/computer-science/emotion-detection]