Annexure -1

SRS for Minor Project-II

Title Of Project: Emotion Detector	
Name of the Supervisor: Ms Sandhya Rani Biswal	
Section: D	**Group No: 3**

Roll No	Regd No	Name of the Student
23CSE234	23UG010318	GOURAB KUMAR JENA
23CSE249	23UG010333	RUTURAJ PADHY
23CSE194	23UG010278	SIBAM DASH

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements for a Real-Time Emotion Detection System. While traditional computer vision tasks focus on simply locating a face, this project aims to interpret the underlying sentiment. The system will detect human faces in live video streams and classify their facial expressions into distinct emotional categories (e.g., Happy, Sad, Angry, Neutral) using the PyTorch deep learning framework.

1.2 Scope

The software will be a desktop application that processes live video feeds. It utilizes a two-step pipeline: first, detecting faces using robust algorithms (such as Haar Cascades or MTCNN), and second, analyzing the "Region of Interest" (the face) using Deep Convolutional Neural Networks (CNNs) to predict emotions. The project aims to bridge the gap between theoretical deep learning and practical deployment by comparing lightweight models against heavy-duty architectures in terms of accuracy and real-time performance (FPS).

1.3 Definitions, Acronyms, and Abbreviations

FER: Facial Expression Recognition.
ROI: Region of Interest (the specific area of the image containing the face)**.**
CNN: Convolutional Neural Network.
MTCNN: Multi-task Cascaded Convolutional Networks (used for face alignment/detection).
FPS: Frames Per Second (measure of system speed).
GUI: Graphical User Interface.
PyTorch: The primary deep learning library used for model training and inference.

1.4 Overview

The remainder of this document describes the overall factors that affect the product and its requirements. Section 2 provides a high-level overview of the system environment and user interaction. Section 3 details the specific functional requirements (logic and processing), and Section 4 defines the interface requirements (hardware, software, and user interaction).

2. Overall Description

2.1 Product Perspective

This system stands as a cornerstone application in the field of Affective Computing. It is designed as a standalone module that can be integrated into larger systems such as automated customer feedback terminals, mental health monitoring tools, or interactive AI systems. It replaces manual observation with automated, real-time sentiment analysis.

2.2 Product Functions

The major functions of the system include:

- Video Acquisition: Capturing live streams via webcam.
- Face Detection: Locating faces within the frame using MTCNN or Haar Cascades.
- Preprocessing: Grayscale conversion, normalization, and resizing of the face ROI.
- Emotion Classification: Passing the processed face data through a PyTorch-based CNN to determine the probability of specific emotions.
- Visualization: Drawing bounding boxes around faces and labeling them with the predicted emotion and confidence score.

2.3 User Classes and Characteristics

- **End Users:** General users who wish to test the system or use it for basic sentiment tracking. No technical expertise required.
- **Developers/Researchers:** Users capable of modifying the underlying PyTorch models, swapping datasets (like FER-2013 or CK+), or adjusting hyper-parameters for optimization.

2.4 Operating Environment

- OS: Windows 10/11 or Linux (Ubuntu).
- Language: Python 3.8+.
- Libraries: PyTorch, OpenCV, NumPy.
- Hardware: Standard PC with a webcam; an NVIDIA GPU is recommended for optimal deep learning inference speed but not strictly required.

2.5 Design and Implementation Constraints

- **Latency**: The system must process frames quickly enough to appear "real-time" to the user (aiming for >15 FPS).
- **Lighting**: Extreme low light or high glare may affect detection accuracy.
- **Pose**: Extreme head angles (profiles) may hinder the ability of the model to classify expressions accurately.

2.6 User Interface

The system will feature a user-friendly Graphical User Interface (GUI). It will provide a live video window and controls to toggle between different detection modes (e.g., "Fast Mode" using Haar vs. "Accurate Mode" using Deep Learning) and display real-time performance metrics.

2.7 Assumptions and Dependencies

- It is assumed the user has a functional webcam connected.
- It is assumed the user has the necessary Python drivers and libraries installed.
- The system relies on the availability of pre-trained weights for the detection and classification models.

3. Functional Requirements

3.1 Introduction

This section outlines the specific behaviors and functions the software must support to convert raw video input into labeled emotional data.

3.2 Functional Requirements

- FR-01 Stream Processing: The system shall accept video input from a default camera device using OpenCV.
- FR-02 Face Localization: The system shall identify the coordinates (x, y, width, height) of any faces present in the frame.
- FR-03 ROI Extraction: The system shall crop the detected face from the frame.
- FR-04 Image Preprocessing: The system shall convert the cropped face to grayscale and resize it to the input dimension required by the model (e.g., 48x48 pixels).
- FR-05 Emotion Inference: The system shall use the loaded PyTorch model to calculate probability scores for emotion classes (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral).
- FR-06 Output Rendering: The system shall overlay a bounding box and the text of the dominant emotion on the original video frame.

3.3 External Interfaces

The system interacts primarily with the Operating System's camera drivers and the file system (to load model weights like .pth or .h5 files). It utilizes the PyTorch API for tensor operations and the OpenCV API for image manipulation.

4. Interface Requirements

4.1 User Interfaces

- **Main Window**: A primary window displaying the video feed.
- **Control Panel**: A set of inputs allowing the user to:
  - Start/Stop the webcam.
  - Select the detection algorithm (Haar Cascade vs. MTCNN).
  - Toggle the display of confidence probabilities.
- **Status Bar**: A display area showing the current Frames Per Second (FPS) to evaluate computational speed.

4.2 Hardware Interfaces

- **Camera**: USB or integrated webcam supporting at least 640x480 resolution.
- **Display**: Monitor capable of displaying the GUI application.
- **Processor**: CPU for logic handling; optional GPU (CUDA cores) for accelerating tensor calculations.

4.3 Software Interfaces

- **Operating System:** Windows or Linux.
- **Frameworks:**
  - PyTorch: For loading the Neural Network architecture.
  - OpenCV (cv2): For video capture and image processing.
  - NumPy: For matrix and array operations.

4.4 Communication Interfaces

The current version operates as a local, standalone application and does not require active network communication (internet) during runtime, provided all model weights and dependencies are downloaded locally. Future versions may implement API endpoints to send emotion data to a cloud server.

Supervisor **Project Coordinator**