Emotion Detection System Using Python

1. Introduction

The Emotion Detection System is a real-time application developed in Python to classify human emotions based on facial expressions captured through a webcam. Utilizing advanced computer vision and machine learning techniques, the system identifies **seven emotions**: Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprised.

This project integrates libraries such as **MediaPipe** for facial landmark detection, **OpenCV** for video processing, **Scikit-learn** for machine learning, and **NumPy** for numerical computations.

The primary objectives of the system include:

- Real-time emotion recognition.
- Applications in human-computer interaction, mental health monitoring, and customer feedback analysis.

The system processes video frames to extract 468 3D facial landmarks, trains a **Random Forest classifier** on labeled data, and predicts emotions in real time while providing visual feedback on the video feed.

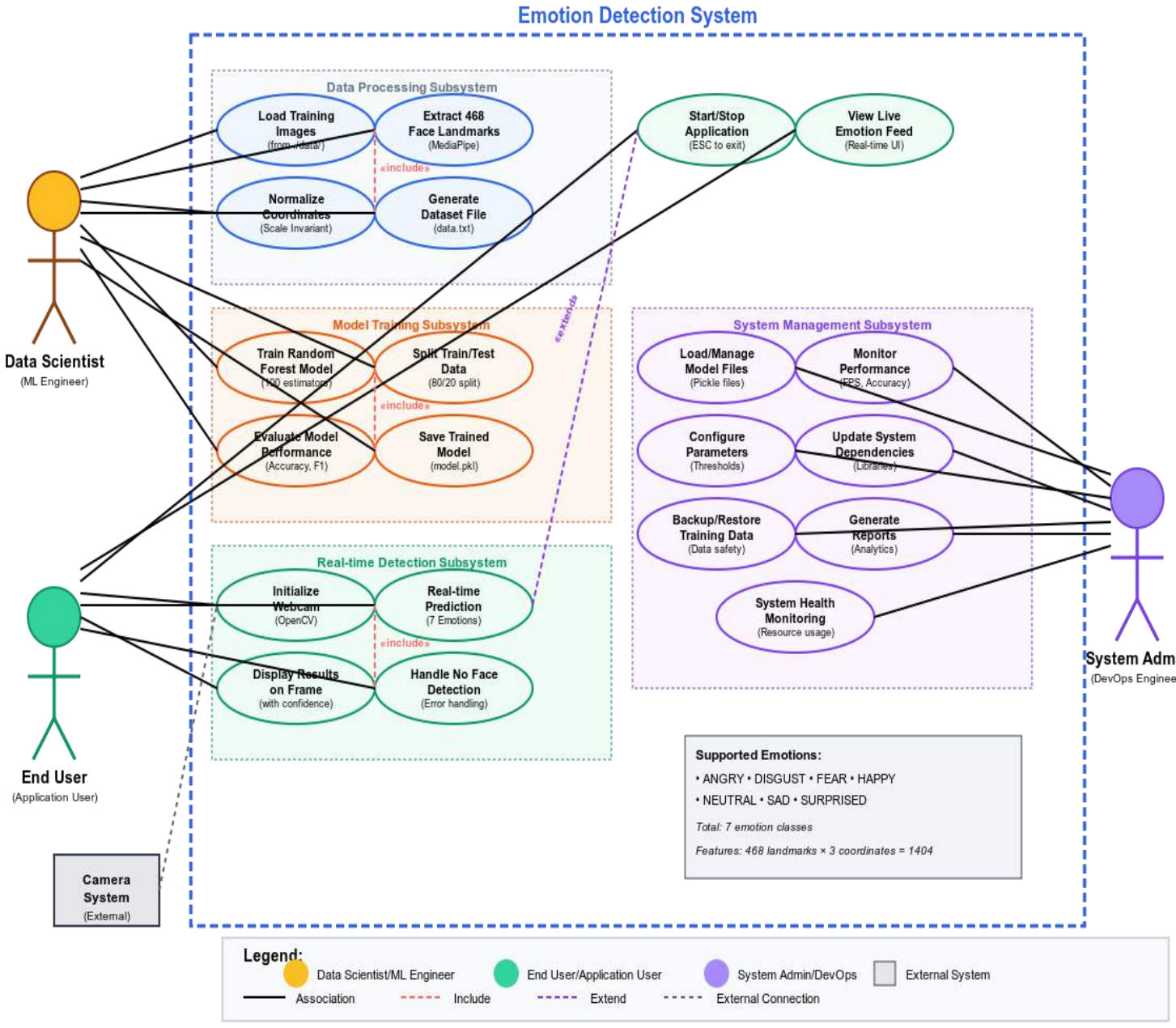
2. Features of the System

The Emotion Detection System offers the following key features:

- **Real-Time Emotion Detection**: Processes live webcam video to detect and classify emotions instantly, displaying the predicted emotion (e.g., "Happy") on the screen.
- **Facial Landmark Extraction**: Uses MediaPipe to extract 468 3D facial landmarks (x, y, z coordinates) for emotion classification.
- **Robust Machine Learning Model**: Employs a Random Forest classifier trained on normalized landmark data, with balanced class weights to handle imbalanced datasets.
- **Modular Design**: Separates functionality into distinct modules for data preparation, model training, real-time testing, and utility functions.
- **Visual Feedback**: Draws facial landmarks on the video feed (optional) and displays emotion labels or "No face detected" if no face is found.
- **Efficient Preprocessing**: Normalizes landmarks to ensure scale invariance, improving model robustness to variations in face size or position.

3. Use Case Diagram

The use case diagram illustrates the interactions between the Data Scientist, End User and the System Admin. Below is the Use Case Diagram for this scanerio:



4. System Architecture

The system architecture consists of **four main components**, which work together to achieve emotion detection:

4.1 Data Preparation Module (prepare_data.py)

- Reads images from a dataset directory organized by emotion.
- Extracts 468 normalized 3D landmarks per image.
- Saves valid data (1404 features + emotion label) to data.txt.

4.2 Model Training Module (train_model.py)

- Loads data from data.txt and splits it into training (80%) and testing (20%) sets.
- Trains a Random Forest classifier with tuned hyperparameters (100 trees, max depth 20).
- Evaluates performance using accuracy, confusion matrix, and classification report.
- Saves the trained model to model.pkl.

4.3 Real-Time Testing Module (test_model.py)

- Captures video frames using OpenCV.
- Extracts landmarks using utils.py.
- Loads the trained model and predicts emotions for each frame.
- Displays the video with the predicted emotion label or "No face detected."

4.4 Utility Module (utils.py)

- Provides functions for landmark extraction using MediaPipe and OpenCV.
- Normalizes landmarks for scale invariance.

The libraries collaborate as follows: MediaPipe detects landmarks, OpenCV captures frames, NumPy formats the data, and Scikit-learn performs classification.

5. Installation and User's Guide

5.1 Installation

To set up the Emotion Detection System:

1. **Install Python 3.8 or higher**: Download from Python.org.
2. **Install required libraries**: Run the following command:

   ```
   pip install opencv-python mediapipe numpy scikit-learn
   ```

3. **Prepare the dataset**: Create a directory named `data` with subfolders for each emotion (e.g., `data/Happy`, `data/Sad`) and place labeled images in the respective folders.
4. **Download project files**: Ensure all necessary files (`utils.py`, `prepare_data.py`, `train_model.py`, `test_model.py`) are in the project directory.

5.2 User's Guide

Step 1: Prepare the Dataset

- Run: **`python prepare_data.py`**
- Output: Generates `data.txt` with landmark data and labels.

Step 2: Train the Model

- Run: **`python train_model.py`**
- Output: Trains the model, displays performance metrics, and saves it to `model.pkl`.

Step 3: Detect Emotions in Real-Time

- Ensure a webcam is connected.
- Run: **`python test_model.py`**
- Output: Displays a window with the video feed, showing the predicted emotion or "No face detected."

Troubleshooting

- Ensure the webcam is properly connected and not in use by another application.
- If `model.pkl` is missing, rerun `train_model.py`.
- Verify the dataset contains valid images with detectable faces.