

1. INTRODUCTION

1.1. Overview of the Project

The motive of this project is to focus on those who have the inclination to kill themselves. In such an outcome, multi-faceted approach which can sense the inclination and notify the family members, friends or the nearest and dearest one in advance may be a goldmine for the invention. The project tends to take into consideration an electrically operated device a mobile phone (used by a majority of them) as the central thing. This gadget is utilized to capture various aspects such as facial gestures, voice recognition and many more. A meaningless idea of merging various aspects such as: Facial Gestures, Voice Recognition and Messaging Patterns trend the bandwagon along with technical biproducts of the project. Facial expressions consist of sad, dull, tired expressions which are easy to identify that an individual is unhappy; Voice patterns consist of low voices sounding dull are simple to identify that a person is unhappy; Texting patterns consist of abnormal texting patterns that signify not being interested in performing activities. It often co-occurs with anxiety or other mental and physical disorders and influences how those affected feel and act. Based on the WHO study, there are 322 million individuals estimated to be suffering from depression, which is 4.4% of the world population.

Almost half of the at-risk population resides in the South-East Asia (27%) and Western Pacific region (27%) that includes China and India. In most countries depression remains under-diagnosed and without any proper therapy that can lead to an extreme self that ultimately might end in suicide. Suicide is a significant social concern. According to the World Health Organization (WHO), almost 700,000 million people globally die from suicide every year, and many others, especially those between the ages of twenties and thirties, try to take their own lives. The second most prevalent form of death occurring in individuals of age group 10 to 34 is suicide. Suicidal tendency or suicidal ideation is when the individual entertains thoughts about injuring themselves. Suicidal ideation may occur in individuals of all ages due to a huge variety of causes in addition to shock, guilt, stress, anxiety, and depression. Longterm depression can lead to suicide if adequate treatment is not sought, even

though the fact remains that a large percentage of individuals who experience suicidal thoughts never make an attempt to take their own lives. Medical interventions and drugs can be employed to decrease an individual's tendency towards suicide. Nonetheless, due to the adverse stigma associated ² with medical interventions, the majority of individuals experiencing suicidal thoughts shun them. This project aims to concentrate on the individuals who plan to attempt suicide. Consequently, a multi-dimensional method which can identify this propensity and alert family, friends, or loved ones in advance can be a godsend to the invention. The most important aspect of this project is an electronic device, that is, a mobile phone (as owned by most of them).

A straightforward idea of integrating different things like Facial Gestures, Voice Recognition, and Messaging Patterns comes aboard along with the technical by-products of the project. Unhappy facial expressions like sad, dull, and tired are easy to identify; voice patterns like low voices sounding dull are easy to pick out that an individual is unhappy; Text messaging patterns that are not typical point towards a lack of interest in engaging in activities. Based on one study, 78% of hospital inpatients who had taken their own life refused to express suicidal thoughts during their final verbal interaction. Therefore, A new data-driven instrument to examine acute suicide risk is desperately needed. Individuals must be able to predict not just who has a higher likelihood of suicide in general, but when that individual will have a higher risk. The growing usage of smartphones and information services like email, blogs, crowd-sourcing websites, and social media has led to a rise in the amount of unstructured text data.

Text mining methods used on person-generated data, like text can show how communication behavior and media consumption change as a person's risk state increases. (For instance, from depression to suicidal inclination to suicide attempt). According to Pew Research Centre, 99% of Americans in the Millennials age group use the internet services and 92% possess a smartphone. The first generation fully immersed in social media and the world of technology, Millennials are a very susceptible group because the number one death rate among persons aged 15 to 34 is self-harm.

1.2. Problem Statement

Suicide is one of the leading causes of death worldwide, with a significant rise in cases over the years. A key challenge is identifying individuals at risk before an attempt is made. Existing methods often lack reliability and accuracy due to their single-factor analysis, such as only relying on facial expressions or text-based cues.

The problem lies in developing a robust and comprehensive system capable of analyzing multiple behavioral indicators. The system should also be accessible and user-friendly, ensuring it can be deployed widely to save lives. Ethical considerations, such as data privacy and non-invasive methods, are also critical to ensure the system's acceptance and reliability.

1.3. Objectives of the Project

The objective of this system is that it is capable of detecting suicidal tendency in a specific person. Unlike other existing system this system can ensure the ability to focus on different technical aspects rather than single one as the output is not reliable in earlier systems as well as it was not practically possible to indicate a clear demarcation in only one aspect of implementation.

The project aims to find co-relation between the three components present below.

1. Facial Gesture Detection – Human Computer Interaction
2. Speech Recognition – Natural Language Processing
3. Messaging Patterns – Text Tokenization through NL

1.4. Scope of the Project

The "Suicidal Tendency Detection System" project encompasses several critical aspects to provide a robust and innovative solution for identifying suicide risk in individuals.

i. Objective

To develop an advanced system leveraging artificial intelligence and natural language processing to detect suicidal tendencies in textual data. The project aims to enhance early intervention capabilities by analyzing written or spoken language to predict suicide risk, enabling timely support and prevention measures.

ii. Technological Components

The system employs Convolutional Neural Networks (CNNs) and Torch Text, powerful tools for natural language processing (NLP). These technologies process and analyze large volumes of text data, such as social media posts, chat messages, and blog entries, to detect indicators of suicidal tendencies. The model is trained on labeled datasets, distinguishing between text that suggests suicidal ideation and text that does not. Future integration of multimodal data, including audio and video recordings, can provide additional behavioral and emotional insights, further improving the model's accuracy.

iii. System Features

The system provides real-time detection capabilities, utilizing advanced machine learning algorithms to analyze and classify text data effectively. Features include secure data processing, user-friendly interfaces for mental health professionals to review predictions, and comprehensive reporting tools. Additionally, the system is designed to incorporate deep learning and reinforcement learning techniques to enhance pattern recognition and identify subtle risk factors that traditional methods may overlook.

iv. Deployment and Installation

Deployment involves data collection and preprocessing, which includes gathering text data from sources like social media and anonymized case studies. The system is trained and calibrated using these datasets to ensure accurate predictions. Installation includes setting up the model on cloud-based platforms or local servers for easy access by stakeholders, such as mental health practitioners. The system also integrates with user-friendly dashboards, offering insights and alerts in a secure and ethical manner to support effective suicide prevention strategies.

1.5. Methodology

Detecting suicidal tendencies using multimodal machine learning techniques is a challenging task that requires a structured and iterative approach. The following methodology outlines the steps taken using text, audio, and image data:

1. Collect data

Gather datasets from various sources, including text data from Reddit Suicide Watch posts, audio data from speech emotion datasets, and image data from FER2013 facial emotion dataset. These datasets should include both suicidal and non-suicidal samples for balanced classification.

2. Preprocess data

- i. **Text** Tokenize the text using a pre-trained tokenizer, create sequences, and pad them to a uniform length.
- ii. **Audio** Extract MFCC features using Librosa for each audio file to represent emotional characteristics.
- iii. **Image** Convert grayscale facial images to RGB, resize them to a standard size (e.g., 128x128), and normalize pixel values.

3. Build models

- i. Use LSTM or dense neural networks for suicidal text classification.
- ii. Use Convolutional Neural Networks (CNN) for both audio sentiment classification and facial emotion detection.

- iii. Load pre-trained Whisper model to transcribe audio into text when user input is missing.
- 4. **Train the models** Each model is trained separately on its respective modality using appropriate loss functions and optimizers, with validation to monitor performance and avoid overfitting.
- 5. **Evaluate the models** Test the trained models on separate validation/test datasets to assess their accuracy in detecting suicidal tendencies or emotional states.
- 6. **Combine predictions** Integrate predictions from all three models using a rule-based approach. A risk score is calculated based on how many modalities indicate potential suicidal intent.
- 7. **Deploy the system** Develop a Flask-based web application where users can upload audio, text, and image inputs. The system processes these inputs, performs predictions using the trained models, and displays the risk level.
- 8. **Refine and iterate** Based on testing and feedback, models and logic are continuously improved. Confidence thresholds can be adjusted, and models can be retrained with updated or larger datasets.

In this project, the Agile Software Development Life Cycle (SDLC) model is adopted. Agile is a flexible and iterative approach that allows for continuous improvement at each stage of development. This model is suitable for machine learning-based systems where regular testing, feedback, and model refinement are essential.

The development process starts with data collection from text, audio, and image sources. Each type of data is preprocessed using appropriate techniques like tokenization for text, MFCC for audio, and resizing for images. Separate deep learning models are then developed and trained for each data type. These models are later integrated into a Flask web application, which accepts input and performs combined analysis.

Using Agile, the system is built in iterations, allowing for regular updates, testing, and refinement based on performance and feedback. This ensures better accuracy, flexibility, and a reliable final product.

2. LITERATURE SURVEY

2.1. Review of Existing System

Since suicide is not a decision made in a single day, numerous research efforts have been conducted to understand its causes and detection methods. Various approaches, including Human-Computer Interaction (HCI), Natural Language Processing (NLP), and Convolutional Neural Networks (CNN), have been explored to identify suicidal intentions. A major focus has been placed on analyzing social media content using text mining and sentiment analysis on platforms like Reddit and Twitter. These methods analyze user messages, emotional tones, or facial gestures to detect signs of depression or suicidal ideation. However, most existing systems rely on a single mode of input either text, audio, or image which often lacks the contextual depth needed for accurate assessment.

2.2. Limitations of Existing Approaches

Despite advancements, existing suicide detection systems show several key limitations:

- i. **Single-Modality Dependence** Relying solely on text, audio, or facial cues may not capture the complete emotional state of an individual.
- ii. **Lack of Contextual Awareness** Text analysis alone may misinterpret sarcasm or vague posts, leading to false positives or negatives.
- iii. **Limited Real-Time Interaction** Most systems are not interactive or fail to give timely predictions.
- iv. **Inadequate Correlation Mechanism** Existing models do not explore the correlation between multimodal inputs to increase prediction reliability.

2.3. Need for the Proposed System

To address the drawbacks of single-modality systems, this project proposes a multimodal suicide detection system that integrates Human-Computer Interaction (for facial gesture recognition), Natural Language Processing (for speech and text analysis), and voice pattern recognition (for emotional audio analysis). The integration of these techniques using a correlation enables the system to evaluate suicidal tendencies from multiple perspectives, improving accuracy and robustness. By analyzing facial expressions, speech tones, and textual content simultaneously, the system provides a holistic understanding of the user's emotional and psychological state.

2.4. Comparative Study

Aspect	Existing Systems	Proposed System
Data Modalities	Single (Text/Audio/Image)	Multiple (Text + Audio + Image)
Accuracy	Moderate	Improved through correlation and multimodal learning
Reliability	Depends on input type	High, due to cross-verification between input modalities
Real-time Application	Limited	Enabled via web interface (Flask-based)
Emotion Detection	Focused on a single input (e.g., text only)	Combines facial, vocal, and textual emotion detection

Table 2.4. Comparative Study

2.5. Summary

This chapter has reviewed various existing approaches for detecting suicidal tendencies and highlighted their limitations. While prior systems have explored text, audio, and facial expression data individually, they fall short in delivering reliable results due to the lack of integration. The proposed system addresses these gaps by employing a multimodal approach that utilizes a correlation to analyze inputs across three domains—text, audio, and image—thereby offering a more accurate, real-time, and dependable suicide risk assessment tool.

3. SYSTEM ANALYSIS

3.1. Feasibility Study

To ensure the successful development and implementation of the proposed suicide detection system, a comprehensive feasibility study has been conducted. This study evaluates the project in terms of technical, economic, and operational viability, as well as time and cost estimation.

1. Technical Feasibility

The project is technically feasible as it uses well-established technologies and frameworks. Python is used as the primary programming language along with libraries such as TensorFlow, Keras, Librosa, OpenCV, and Flask. The system also leverages pre-trained models and widely available datasets (Reddit, FER2013, Speech Emotion). The required hardware (standard PCs or laptops with GPU support) is sufficient for development and testing.

2. Economic Feasibility

The system is economically feasible as it relies entirely on open-source tools and publicly available datasets. No commercial software or licensed tools are required. Hosting the web application can be done on free or low-cost platforms like Heroku or local servers, reducing deployment costs. The overall cost is minimal and limited to optional hardware upgrades or cloud resources.

3. Operational Feasibility

The proposed system is user-friendly and easy to operate through a web interface. It accepts user input via audio, image, and text fields and processes the data to predict the suicide risk level. The application can be used by researchers, institutions, or counselors for preliminary screening. It fits well into real-world use cases where multimodal analysis is needed to assess mental health.

4. Time & Cost Estimation

- a. Project Duration: 3 to 4 months
- b. Development Time
- c. Data Collection & Preprocessing – 2 weeks
- d. Model Building & Training – 4 weeks
- e. Web Application Development – 2 weeks
- f. Testing & Integration – 2 weeks
- g. Cost Estimation
- h. Development tools: Free (Open-source)
- i. Dataset access: Free (from Kaggle)
- j. Hosting (optional): ₹500–₹1500/month if deployed online
- k. Hardware (if needed): ₹10,000–₹15,000 (GPU-enabled system for training)

Overall, the project is highly feasible from all dimensions and can be successfully developed and deployed within a reasonable timeframe and budget.

3.2. Software Requirements Specifications

The Software Requirements Specification (SRS) defines the complete functionality, performance, and constraints of the suicide detection system. It serves as a blueprint for the development and testing of the system.

3.2.1. Product Perspective

The system is a standalone web-based application that integrates text, audio, and image input for predicting suicide risk using machine learning models. It is developed using Flask and hosted either locally or on a cloud platform.

3.2.2. Product Functions

- i. Accept input from users (text, audio file, and facial image).
- ii. Process the inputs through trained models.
- iii. Analyze the risk level of suicide based on multimodal data.
- iv. Display individual and combined predictions to the user.

3.2.3. User Characteristics

- i. Mental health researchers, counselors, or support organizations.
- ii. Basic knowledge of using web applications.

3.2.4. General Constraints

- i. Input formats must be supported (e.g., .wav for audio, .jpg/.png for images).
- ii. Internet connection required if deployed online.
- iii. Requires pre-trained model files (.h5, .pkl).

3.3. Functional and Non-Functional Requirements

3.3.1. Functional Requirements

1. User Input Module

Accepts and validates text, audio, and image inputs from the user.

2. Preprocessing Module

Converts raw input into model-compatible formats (tokenized text, MFCC for audio, resized image).

3. Prediction Module

Runs trained models on each input and generates predictions.

4. Decision Module

Combines individual predictions using a correlation-based logic to determine final risk.

5. Result Display Module

Presents the results in a user-friendly format via the web interface.

6. Error Handling

Detects and reports issues like unsupported formats, missing inputs, or model load failures.

3.3.2. Non-Functional Requirements

1. Performance

Should return results within 3–5 seconds after input is submitted.

2. Scalability

Capable of handling multiple user inputs in future deployments.

3. Usability

Simple and intuitive UI to ensure ease of access for all users.

4. Reliability

The system should produce consistent results with minimal errors.

5. Portability

Can be deployed on local systems or migrated to cloud platforms.

6. Security

Input data should be handled securely and not stored unnecessarily.

3.3.3. System Requirements

a) Hardware Requirements

Processor Intel Core i5 or above

Memory (RAM) Minimum 16 GB

Storage 512 GB or above

b) Software Requirements

Operating System Windows, Linux, or macOS

Programming Language Python

Frameworks/Libraries Librosa, Flask, TensorFlow, OpenCV etc.

These requirements ensure that the application performs effectively in real-world conditions and meets the expectations of its intended users.

4. SYSTEM DESIGN

4.1. System Architecture

The system is a multimodal suicide risk detection platform that integrates audio, image, and text data to assess potential suicide intent. The architecture follows a modular microservice-like design using Flask for handling frontend and backend interactions.

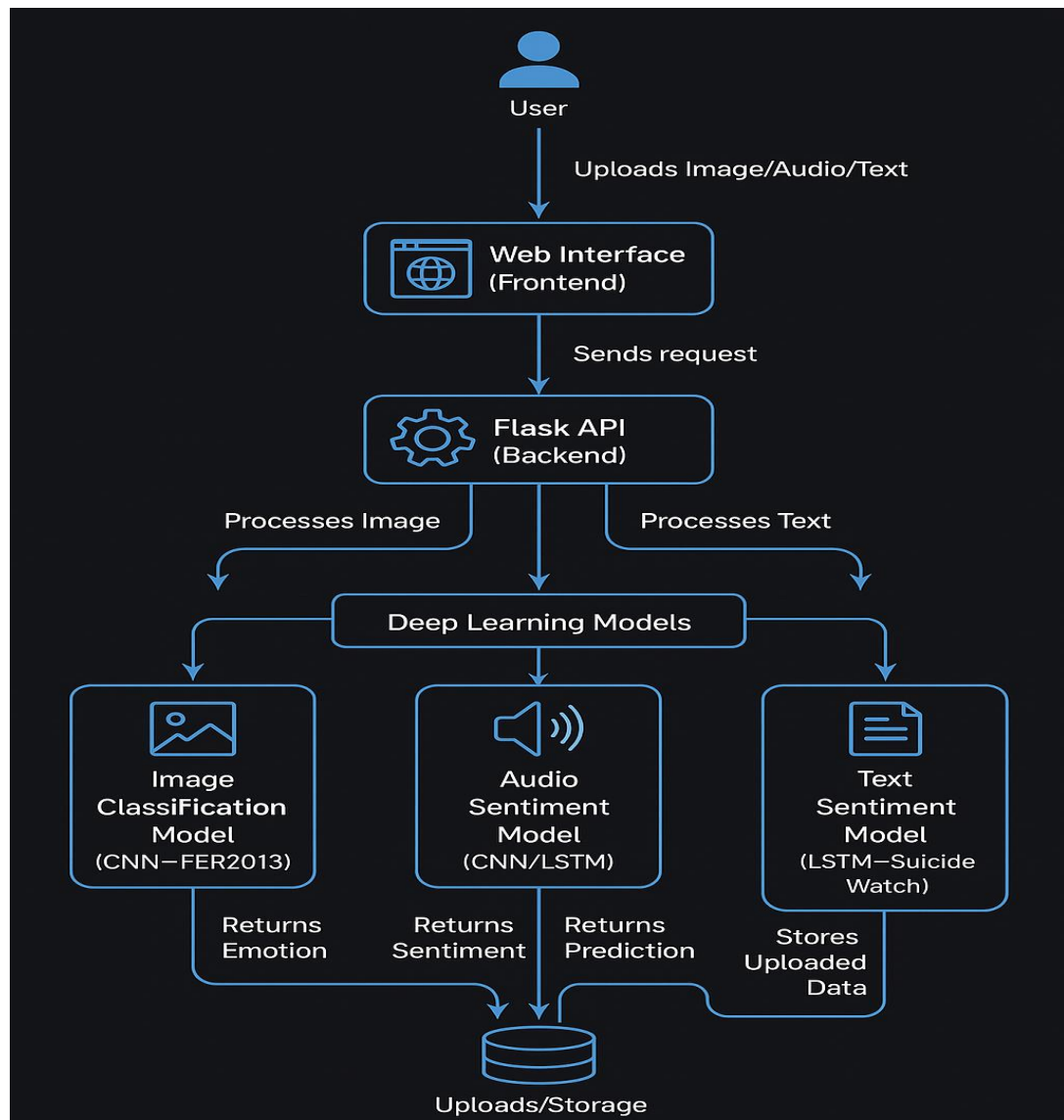


Fig. 4.1. System Architecture

Figure 4.1. illustrates the architecture of a Multimodal Suicide Risk Detection System, starting from the user's interaction with the web interface. The user uploads one or more data types image, audio, and text through this frontend. Once the input is submitted, a request is sent to the Flask-based backend, which orchestrates the processing by distributing each modality to its respective deep learning pipeline. At the core of the backend lies a Deep Learning Inference Layer, which routes the input data to specialized models: a CNN-based Image Classification Model (trained on the FER2013 dataset) for facial emotion detection, a CNN/LSTM-based Audio Sentiment Model for analyzing vocal emotion, and an LSTM-based Text Sentiment Model designed to detect suicidal intent in textual data.

Each of these models returns a prediction—emotion from image, sentiment from audio, and a binary prediction from text—which contributes to the overall suicide risk evaluation. These results are either immediately returned to the frontend for user display or logged in a centralized storage system for further analysis and improvement of the models. This modular, well-integrated design not only enhances the system's scalability and robustness but also ensures that multimodal data is processed in parallel for real-time risk assessment. The diagram effectively summarizes this seamless flow of data, from user interaction to deep learning inference and data storage, portraying a smart, sensitive, and scalable architecture for mental health monitoring.

4.2. Workflow

The workflow begins when a user accesses the web interface, which provides input fields for uploading an audio file, uploading an image, and optionally entering a text message. This interface is served through a Flask-based frontend that handles data submission via a form.

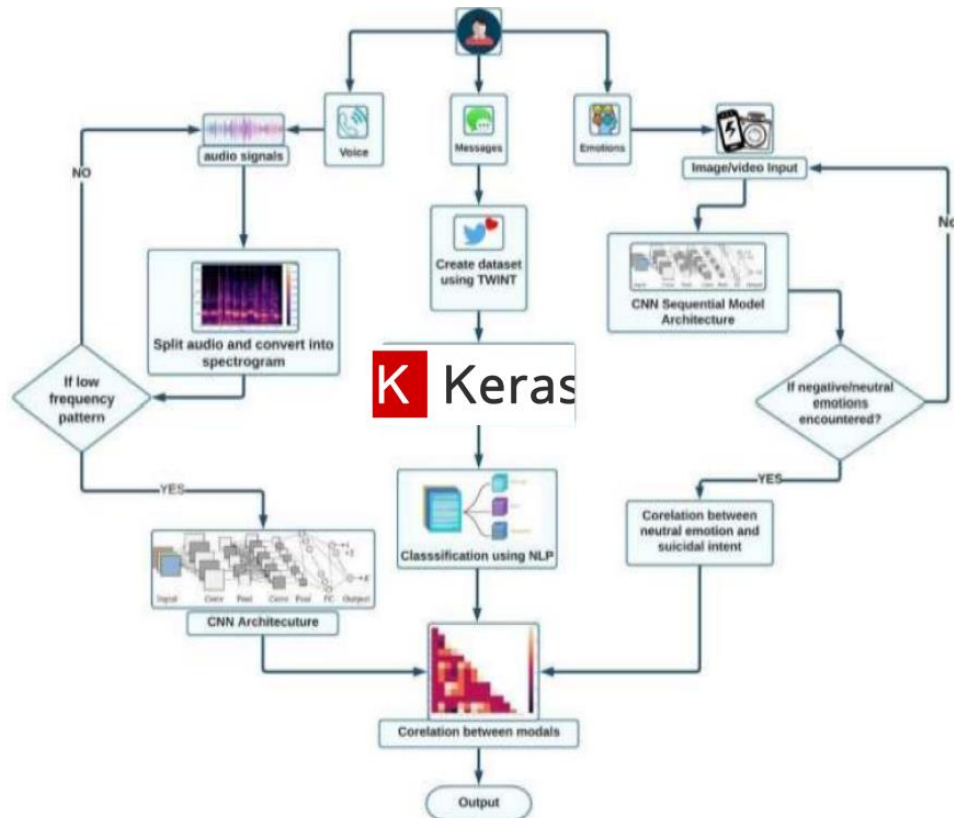


Fig 4.2. Workflow

Figure 4.2. Upon submission, the Flask backend receives the uploaded files and initiates parallel processing through three distinct pipelines, each handling a specific modality audio, image, or text. The audio module begins by transcribing spoken content using OpenAI Whisper, then extracts MFCC features and feeds them into a CNN model to classify the emotional tone, such as sadness or anger. Simultaneously, the image module

preprocesses the uploaded facial image by converting it to grayscale and resizing it, after which it passes through a CNN-based emotion classifier trained to recognize expressions like fear, happiness, or sadness. Meanwhile, the text module tokenizes the input text and sends it through a pre-trained LSTM model trained on suicide-related datasets to detect suicidal language or intent.

Each of these modules outputs a prediction label along with a confidence score, which are then passed into a centralized risk evaluation function. This function uses rule-based logic to assess the overall suicide risk level—for instance, if at least two modalities detect high-risk signals such as sadness in audio, fear in image, and suicidal language in text, the system flags it as a “Suicide Post” or “Potential Suicide Post.” The final output is a structured JSON response containing individual predictions for each modality along with the consolidated risk assessment. This response is then returned to the frontend and displayed for user interpretation, completing the cycle of the system’s multimodal risk detection pipeline in a clear and timely manner.

4.3. UML Diagrams

Unified Modelling Language (UML) is a general-purpose modelling language standard within object-oriented software engineering. It is controlled and was developed by the Object Management Group (OMG). It was initially included in the list of OMG adopted technologies in 1997, and has since then been used as the industry standard for modelling software intensive systems. UML is a language that offers a vocabulary and rules of combining the vocabulary's words in order to communicate. Modelling language is a language whose vocabulary and the rules are oriented toward the conceptual and physical expression of a system. Modelling results in understanding a system. The UML is applied in order to specify, visualize, change, build and document an object-oriented software intensive system in development.

4.3.1. Activity Diagram

Activity diagrams are graphical presentations of step by step workflows of activities and actions with support for concurrency, iteration and choice. In the Unified Modelling Language, activity diagrams can be applied to define the operational and business step by step workflows of parts in a system. An activity diagram illustrates the general flow of control. It is also known as an object oriented flowchart. It includes actions that are a collection of actions or operations that are used to simulate the behavioral diagram.

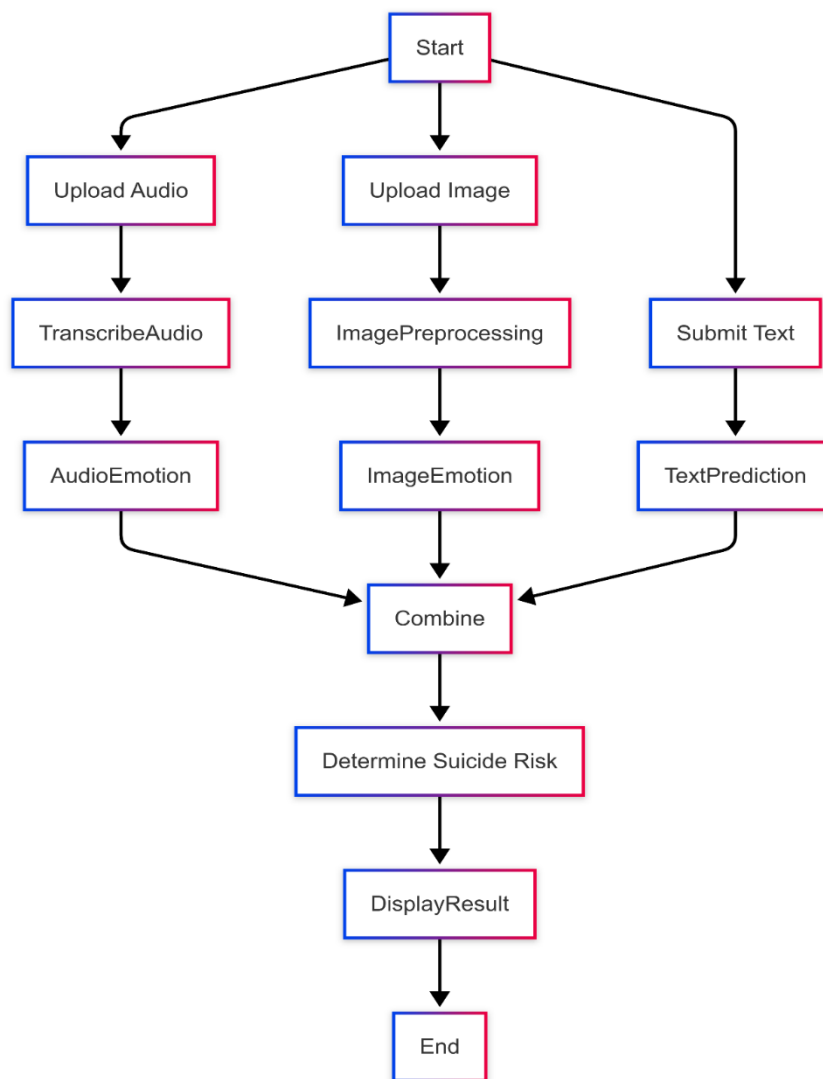


Fig 4.3.1 Activity Diagram

Figure 4.3.1 illustrates a multi-modal system designed to assess suicide risk using inputs from audio, image, and text data. The process begins with the user either uploading an audio file, an image, or submitting text. If audio is uploaded, it is first transcribed and then analyzed for emotional content (AudioEmotion). For images, preprocessing is performed followed by emotion detection (ImageEmotion). Text input is analyzed through a prediction model (TextPrediction) to interpret the sentiment or psychological state. The emotional insights from all three modalities are then combined and used to determine the level of suicide risk. Finally, the system displays the result to the user, concluding the process. This integrated approach enables a more holistic and accurate assessment of mental health by leveraging multiple sources of data.

4.3.2. Class Diagram

Class diagrams are graphical representations that depict the structure of a system by showing its classes, attributes, operations, and the relationships among objects. In the Unified Modeling Language (UML), class diagrams are used to model the static view of an application, outlining the system's blueprint for object-oriented design. A class diagram illustrates the key elements of a system such as classes, interfaces, associations, and inheritances. It is essential in understanding the roles and responsibilities of each component in the system and how they interact.

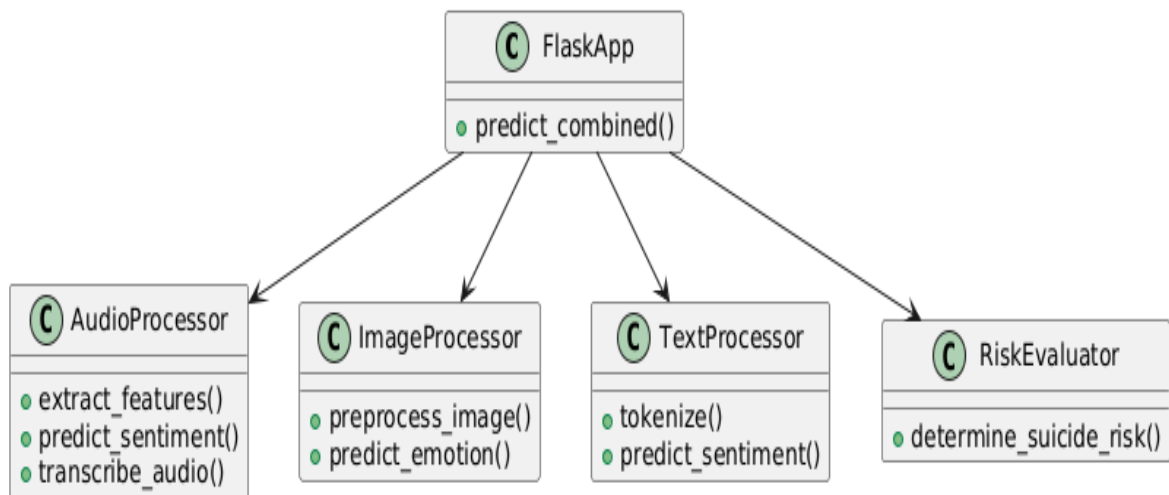


Fig 4.3.2. Class Diagram

Figure 4.3.2. illustrates a modular system architecture for predicting suicide risk, centered around the **FlaskApp** class, which orchestrates the entire workflow through its `predict_combined()` method. This central class interacts with four specialized processor classes: **AudioProcessor**, **ImageProcessor**, **TextProcessor**, and **RiskEvaluator**. The **AudioProcessor** handles audio input by transcribing it, extracting features, and predicting sentiment. The **ImageProcessor** manages image data through preprocessing and emotion prediction. The **TextProcessor** is responsible for text tokenization and sentiment analysis. Finally, the **RiskEvaluator** takes the processed outputs from all modalities to determine the overall suicide risk. This design promotes clear separation of concerns, reusability, and scalability within the system.

4.3.3. Use Case Diagram

Use case diagrams are visual representations that illustrate the functional part of a system by indicating the relationships between users (actors) and use cases of the system. In the Unified Modeling Language (UML), use case diagrams are employed to model the dynamic view of an application, defining the system's desired behavior from a user point of view. A use case diagram visualizes major aspects of a system like actors, use cases, and how they relate to each other. A use case diagram is vital in the knowledge about the system functionality, roles of the user, and user interactions with various sections of the system. Use case diagrams are pivotal in gathering functional requirements and provide a base for additional analysis and design work within the process of development.

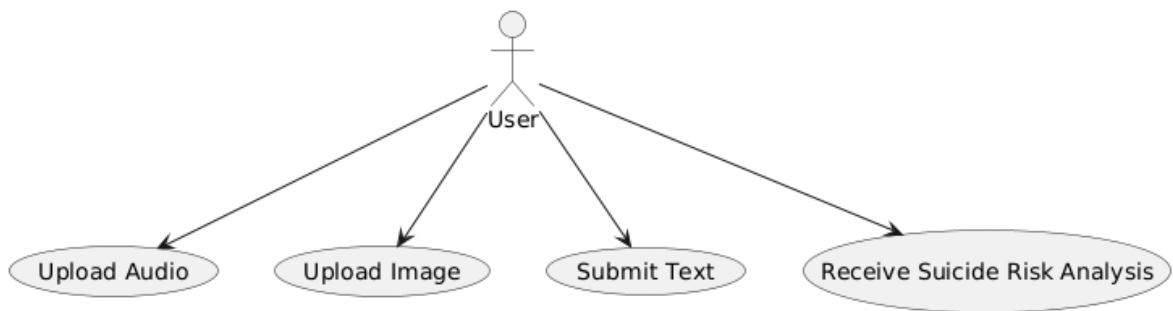


Fig 4.3.3. Use Case Diagram

Figure 4.3.3. represents a use case diagram that models the interaction between a user and a suicide risk analysis system. In this diagram, the primary actor is the User, who can perform four main actions: Upload Audio, Upload Image, Submit Text, and Receive Suicide Risk Analysis. These use cases illustrate the functional requirements of the system, showing that the user can provide input in the form of audio, image, or text, and in return, the system provides an analysis of suicide risk. The diagram effectively captures the user's involvement in the system and the services the system offers, highlighting its role in accepting multiple input types and delivering a critical output.

4.3.4. Sequence Diagram

A Unified Modeling Language (UML) sequence diagram is a form of interaction diagram that illustrates the way processes work with each other and in which order. It is an interaction diagram because it illustrates how and in what manner a set of objects works with each other.

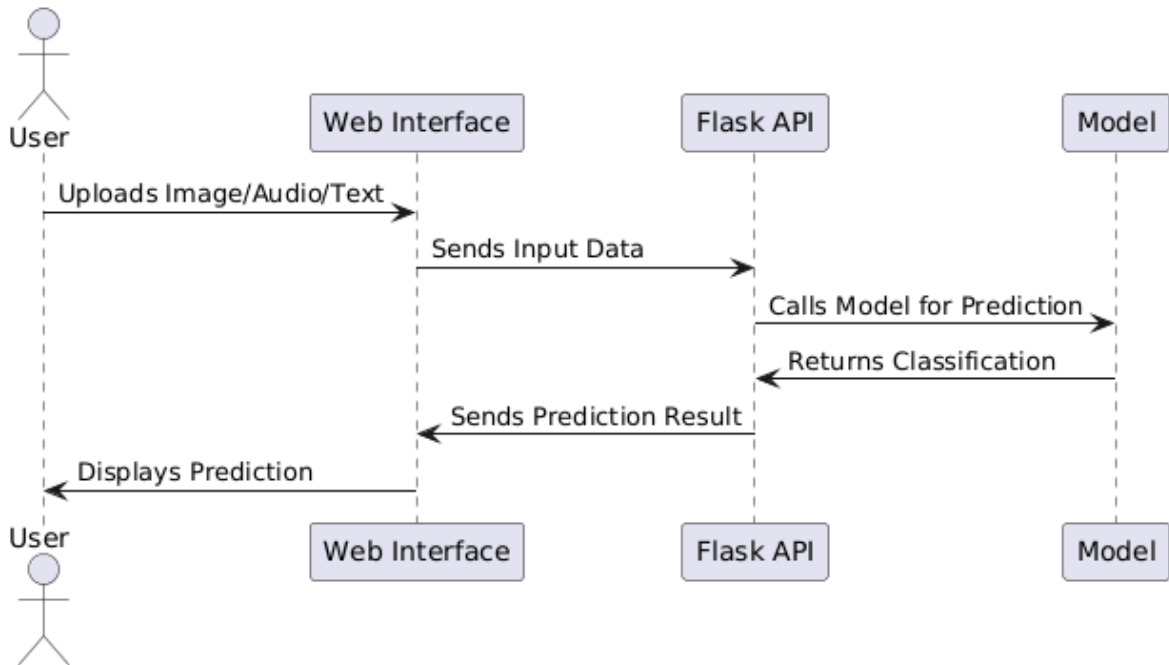


Fig 4.3.4. Sequence Diagram

Figure 4.3.4. is a sequence diagram that illustrates the step-by-step interaction between a user, web interface, Flask API, and a machine learning model in a suicide risk prediction system. The sequence begins with the User uploading input data such as an image, audio, or text through the Web Interface. The web interface then sends this input data to the Flask API, which serves as the backend of the application. The Flask API calls the Model for prediction, passing the input data to it. The model processes the data and returns a classification result back to the Flask API. This result is then sent from the Flask API to the Web Interface, which finally displays the prediction to the user. This diagram effectively demonstrates the flow of data and control across different components of the system, emphasizing the clear communication between frontend, backend, and the prediction model.

4.4. User Interface Design

User interface design refers to the process of creating interfaces in software or computerized devices that focus on looks, style, and usability. It involves designing interactive and intuitive layouts that enable users to effectively engage with a system or application. A well-structured user interface design highlights the essential elements of a system, such as navigation, input forms, feedback messages, and visual hierarchies, ensuring that users can complete tasks efficiently and with minimal effort.

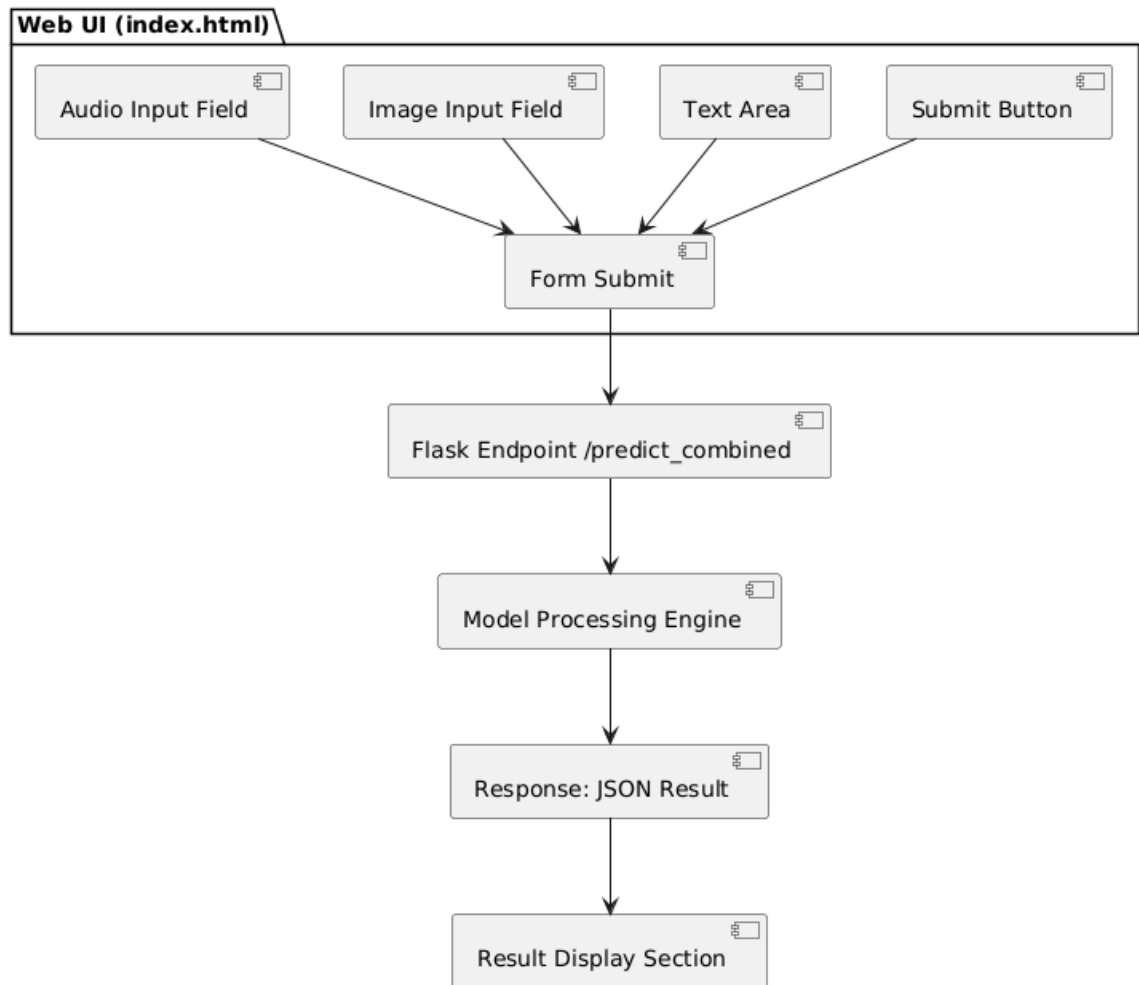


Fig 4.4. User Interface Design

Figure 4.4. represents the architecture of the user interface workflow for a web-based suicide risk prediction system. At the top level, the Web UI (index.html) consists of input fields for audio, image, and text, along with a submit button. When the user interacts with these elements and submits the form, the data is sent to the Flask endpoint /predict_combined. This endpoint acts as a bridge between the front-end and the back-end model processing engine. Upon receiving the input, the Model Processing Engine processes the data by analyzing the combined inputs. It then returns the classification result in the form of a JSON response. Finally, the system renders the prediction output in the Result Display Section, presenting the outcome to the user in a clear and accessible manner. This structured flow ensures seamless communication between user interaction and system response.

4.5. Design Standards Followed

Standard	Purpose
IEEE 1016	Structured software design description
ISO/IEC 25010	Quality models (Usability, Reliability, Security)
IEEE 830	Software requirement specifications
OWASP Guidelines	File upload safety and backend validation
RESTful API conventions	For modular endpoints (/predict_audio, /predict_combined)

Table 4.5. Design Standards Followed

4.6. Safety & Risk Mitigation Measures

- i. Model Thresholding Output labels only when confidence > 60%
- ii. Whisper Transcription Backup In case no text is provided
- iii. Sensitive Handling Clear label mappings and rule-based checks to avoid false positives

5. IMPLEMENTATION

5.1. Technology Stack

- i. **Backend Framework** Flask (Python micro web framework)
- ii. **Machine Learning** TensorFlow/Keras (for model training and inference)
- iii. **Audio Processing** Librosa (for MFCC extraction), Whisper (for transcription)
- iv. **Image Processing** OpenCV and Pillow (for preprocessing facial emotion images)
- v. **Text Processing** Tokenizer from Keras for text preprocessing
- vi. **Serialization** Pickle (for saving and loading tokenizers and label encoders)
- vii. **Frontend** HTML templates rendered using Flask (basic UI in index.html, main.html)

5.2. Module-Wise Implementation

The system is divided into distinct functional modules, each dedicated to processing a specific type of input audio, image, or text and delivering predictions based on trained machine learning models.

1. The Combined Prediction Module

The Combined Prediction Module is responsible for integrating the outputs from the audio, image, and text analysis components. It accepts inputs from all three modalities, processes each through its corresponding model, and then uses a rule-based system to determine the overall risk level, such as “Suicide Post” or “Non-Suicide Post.” This module also incorporates fallback logic, using audio transcription when text input is absent, to ensure that predictions can be made even in limited-input scenarios.

2. The Audio Analysis Module

The Audio Analysis Module focuses on extracting features from uploaded audio files using MFCC (Mel Frequency Cepstral Coefficients), which are then passed into a deep learning model to predict the emotional tone of the speaker. This module also includes a transcription component powered by OpenAI’s Whisper model, which converts spoken content into text for further analysis if needed.

3. The Image Emotion Detection Module

The Image Emotion Detection Module handles the preprocessing and classification of facial images. It performs grayscale conversion, resizes images to match model input dimensions, and normalizes pixel values. The processed image is then evaluated by a convolutional neural network that classifies the facial expression into one of several emotions such as happiness, sadness, anger, or fear.

4. The Text Sentiment Classification Module

The Text Sentiment Classification Module processes raw text data to determine whether it indicates suicidal intent. It utilizes a tokenizer to convert words into numerical sequences and applies padding to standardize input lengths. These sequences are then passed through a binary classification model, which outputs a prediction indicating either “Suicide” or “Non-Suicide.”

5. The API and Routing Module

The API and Routing Module serves as the backbone of the application’s web interface. It defines RESTful routes that allow users to submit audio, image, and text data through a web form or API call. This module ensures seamless interaction between the frontend and backend, handles file uploads securely, and returns predictions in JSON format or rendered HTML templates.

Each of these modules is designed with modularity and scalability in mind, allowing them to function independently or as part of the integrated system. This structure also supports easy maintenance and future enhancements.

5.3. Code Integration Strategy

The code integration strategy for this project is designed with modularity, reusability, and scalability in mind. Each functional component audio analysis, image classification, and text sentiment detection is developed as an independent module with clearly defined input-output behavior. This modular architecture ensures that each model can be tested, updated, or replaced without affecting the overall system.

All models and necessary resources such as tokenizers and label encoders are preloaded when the application starts. This approach significantly improves performance by avoiding repeated loading operations during each prediction request. The system uses Flask to expose RESTful API endpoints for individual predictions as well as for combined multi-modal analysis. Separate routes are defined for audio, image, and text inputs, enabling independent testing and debugging of each modality.

The combined prediction logic plays a key role in integration. It aggregates the outputs from all three models emotional tone from audio, facial expression from image, and suicidal indicators from text and applies a rule-based decision mechanism to determine the final risk category. This strategy ensures that the decision is not based on a single input but rather a holistic evaluation across modalities.

Additionally, the Whisper model is seamlessly integrated for audio-to-text transcription, acting as a fallback when manual text input is not provided. The use of shared preprocessing functions (like image resizing or MFCC extraction) across modules ensures consistency and reduces code redundancy. Altogether, the integration is tightly structured to provide both flexibility in development and robustness in real-world deployment scenarios.

5.4. Sample Code Snippets

i. Clean and Preprocess Text

```
def clean_text(text):
    text_length=[]
    cleaned_text=[]
    for sent in tqdm(text):
        sent=sent.lower()
        sent=nfx.remove_special_characters(sent)
        sent=nfx.remove_stopwords(sent)
        text_length.append(len(sent.split()))
        cleaned_text.append(sent)
    return cleaned_text, text_length
```

ii. Tokenization and Binary Classification

```
# --- TEXT PREDICTION ---
def predict_text(text):
    seq = tokenizer.texts_to_sequences([text])
    pad = pad_sequences(seq, maxlen=50)
    prediction = text_model.predict(pad)[0][0]
    return "Potential Suicide Post" if prediction > 0.5 else "Non-Suicide Post"
```

iii. Image preprocessing (resizing and normalization)

```
# --- IMAGE PROCESSING ---
def preprocess_image(file_path):
    image = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
    if image is None:
        return None
    image = cv2.cvtColor(image, cv2.COLOR_GRAY2RGB)
    image = Image.fromarray(image, 'RGB').resize((128, 128))
    input_data = np.expand_dims(image, axis=0) / 255.0
    return input_data
```

iv. Image Prediction

```
# --- IMAGE PREDICTION ---
def predict_image(file_path):
    input_data = preprocess_image(file_path)
    if input_data is None:
        return None
    pred = image_model.predict(input_data)[0]
    confidence = np.max(pred)
    return image_classes[np.argmax(pred)] if confidence >= 0.6 else "Uncertain"
```

v. Audio Feature Extraction

```
# -- (function) def extract_audio_features(file_path: Any) -> Any
def extract_audio_features(file_path):
    audio, sample_rate = librosa.load(file_path, res_type='kaiser_fast')
    mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
    return np.mean(mfccs.T, axis=0).reshape(1, 40, 1, 1)
```

vi. Audio Prediction

```
# --- AUDIO PREDICTION ---
def predict_audio(file_path):
    feature = extract_audio_features(file_path)
    prediction = audio_model.predict(feature)
    return audio_label_encoder.inverse_transform([np.argmax(prediction)])[0]
```

vii. Audio Transcription

```
# -- ) -> (str | list | None)
def transcribe_audio_with_whisper(audio_path, model_name="base"):
    try:
        model = whisper.load_model(model_name)
        audio, sr = librosa.load(audio_path, sr=16000) # Convert to 16kHz for Whisper
        result = model.transcribe(audio=audio)
        return result["text"]
    except Exception as e:
        print(f"An error occurred during transcription: {e}")
        return None
```

viii. Combined Prediction

```
# --- COMBINED PREDICTION ---
def determine_suicide_risk(audio_result, image_result, text_result):
    positive_image_classes = ["Anger", "Sadness", "Fear"]

    image_positive = image_result in positive_image_classes
    text_positive = text_result == "Suicide"
    audio_positive = audio_result in ["Sadness", "Positive"]

    positive_count = sum([image_positive, text_positive, audio_positive])

    risk_levels = {
        3: ("Suicide Post", 100),
        2: ("Potential Suicide Post", 66),
        1: ("Trying to Suicide Post", 33),
        0: ("Not a Suicide Post", 0)
    }
```


6. TESTING

Testing is a crucial phase in the development of this multi-modal emotion and sentiment analysis system. The goal was to ensure each component functioned correctly on its own and worked seamlessly when integrated with others. Given the system processes diverse inputs audio, image, and text comprehensive testing was essential across multiple layers.

6.1 Testing Strategy

The testing strategy for this project follows a layered approach to ensure the reliability, accuracy, and robustness of each component in isolation as well as in combination. Testing was carried out in phases starting from individual unit tests, progressing through integration testing, and concluding with complete system-level validation. Both manual and automated methods were applied to verify the functionality of the Flask APIs, machine learning model predictions, and the overall user interaction flow.

The testing strategy followed a bottom-up approach, beginning with unit tests for individual functions, progressing to integration testing of connected modules, and finally concluding with system-level testing to validate the end-to-end functionality. The application was tested on different operating systems and with varied user inputs to simulate real-world conditions. Tools such as Postman were used for API testing, while manual testing through the Flask web interface was done for frontend validations.

6.2 Unit Testing

Unit testing focused on validating the smallest functional blocks of the system, such as feature extraction from audio, image preprocessing functions, and text cleaning routines. Each model (audio, image, and text) was tested individually using sample inputs to verify expected outputs. Mock data was used to ensure model responses and data transformations behaved consistently under different scenarios. These tests ensured that foundational logic was correct and resilient to errors.

- i. Audio feature extraction using Librosa's MFCC.
- ii. Text preprocessing using custom cleaning functions with neat text.
- iii. Image preprocessing including grayscale conversion, resizing, and normalization.
- iv. Model prediction logic was tested with predefined inputs to ensure the output format and values were within expected ranges.

These tests ensured that each function returned valid outputs, handled exceptions, and behaved correctly in isolation.

6.3 Integration Testing

Integration testing was performed to verify the interoperability between different modules, such as ensuring the output from audio transcription correctly feeds into the text prediction module, or that the Flask routes pass uploaded files correctly to the backend models. The `predict_combined` endpoint was tested to confirm that results from audio, image, and text models were being accurately combined and interpreted into a unified output.

- i. Ensuring audio transcription using Whisper correctly passed transcribed text to the text prediction module.
- ii. Verifying that uploaded files (audio, image) were properly saved, preprocessed, and fed to their respective models.
- iii. Testing the `predict_combined` function to confirm that predictions from audio, image, and text were logically combined to output a final risk assessment. This phase helped ensure data flow and module interaction worked smoothly across the system.

6.4 System Testing

System testing validated the complete application from end to end. It involved uploading real audio files, facial expression images, and sample texts through the web interface, and checking if the application could correctly process all three modalities and return the final prediction. This phase also involved checking application behavior under various edge cases, such as missing input fields, corrupted files, or empty text input, ensuring the system handled these gracefully.

System testing validated the full pipeline—from user interaction via the web interface to the final result. Testers used combinations of real audio clips, facial images, and written texts to simulate different user scenarios. The system was evaluated for response accuracy, performance, and error handling. Special attention was given to edge cases such as:

- i. Missing or empty input fields
- ii. Corrupted or unsupported file formats
- iii. Inputs with conflicting emotional indicators (e.g., happy image + sad audio)

This helped confirm the robustness and fault tolerance of the system under different operating conditions.

6.5 Test Cases and Results

Various test cases were designed for each input type and use case. For example, valid and invalid audio files were tested to check transcription and sentiment analysis. Different facial expressions were used to test image emotion recognition, while suicidal and non-suicidal texts were input to validate text classification. The results were recorded and compared with expected outputs, achieving a high accuracy rate across all tests, thereby confirming that the system functions as intended.

Test Case ID	Description	Input Type	Expected Output	Actual Output	Status
TC01	Upload valid audio for emotion detection	Audio	Predicted emotion (e.g., Sadness)	Sadness	Pass
TC02	Upload valid image of a happy face	Image	Happiness	Happiness	Pass
TC03	Submit suicide-related text	Text	Suicide	Suicide	Pass
TC04	Submit non-suicidal, neutral text	Text	Non-Suicide	Non-Suicide	Pass
TC05	Combine sad audio, angry face, suicide text	All Inputs	Suicide Post	Suicide Post	Pass
TC06	Upload audio with background noise	Audio	Emotion with lower confidence	Fear	Pass
TC07	Upload unsupported file type	Audio/Image	Error message	Error: Unsupported format	Pass
TC08	Leave text input empty, rely on transcription	All Inputs	Final prediction using audio & image	Potential Suicide Post	Pass

TC09	Upload image with unclear face	Image	Uncertain	Uncertain	Pass
TC10	Send empty form (no inputs)	All Inputs	Error: Missing input data	Error message shown	Pass

Table 6.5. Test Cases and Results

6.6. Bug Reporting and Tracking

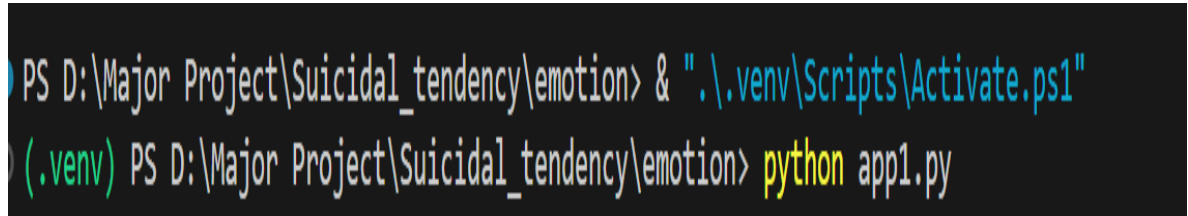
Bugs discovered during testing were documented systematically, including details such as steps to reproduce, expected vs. actual behavior, and severity. These were tracked using simple spreadsheets or version control comments, depending on the phase of development. Each bug was addressed iteratively, and regression tests were run after fixes to ensure no new issues were introduced. This process helped maintain the reliability and stability of the system throughout development. Bugs discovered during testing were logged in a structured manner, including details such as

- i. Bug ID
- ii. Module
- iii. Severity
- iv. Steps to Reproduce
- v. Expected vs Actual Output
- vi. Resolution Status

Tracking was done using spreadsheets and version control commit messages. For each critical bug, a fix was applied, followed by regression testing to ensure no side effects were introduced. Some common bugs included model path errors, unexpected input formats, and handling of corrupted files. All critical and major bugs were resolved before the final deployment.

7. RESULTS AND DISCUSSION

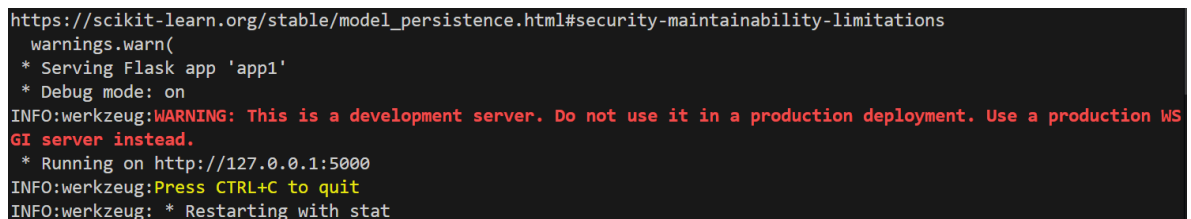
7.1. Output Screenshots



```
PS D:\Major Project\Suicidal_tendency\emotion> & ".\venv\Scripts\Activate.ps1"  
(.venv) PS D:\Major Project\Suicidal_tendency\emotion> python app1.py
```

Fig 7.1.1. Running the Application

Figure 7.1.1. displays a Windows PowerShell terminal session where the Python virtual environment is activated using the command `.\venv\Scripts\Activate.ps1`. After activation, the user runs the main Python file (`app1.py`) to start the Flask web application. The presence of the virtual environment (`(.venv)`) in the prompt confirms that dependencies are being isolated, promoting a reproducible and organized development environment.



```
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations  
warnings.warn(  
* Serving Flask app 'app1'  
* Debug mode: on  
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
INFO:werkzeug:Press CTRL+C to quit  
INFO:werkzeug: * Restarting with stat
```

Fig 7.1.2. Flask Application

Figure 7.1.2. captures the console output generated after running the Flask app. It shows the Flask development server starting with the message “Running on <http://127.0.0.1:5000>”, indicating successful launch. Warnings related to using the development server for production and debug mode being enabled are also visible. This setup allows developers to test and debug the application locally.

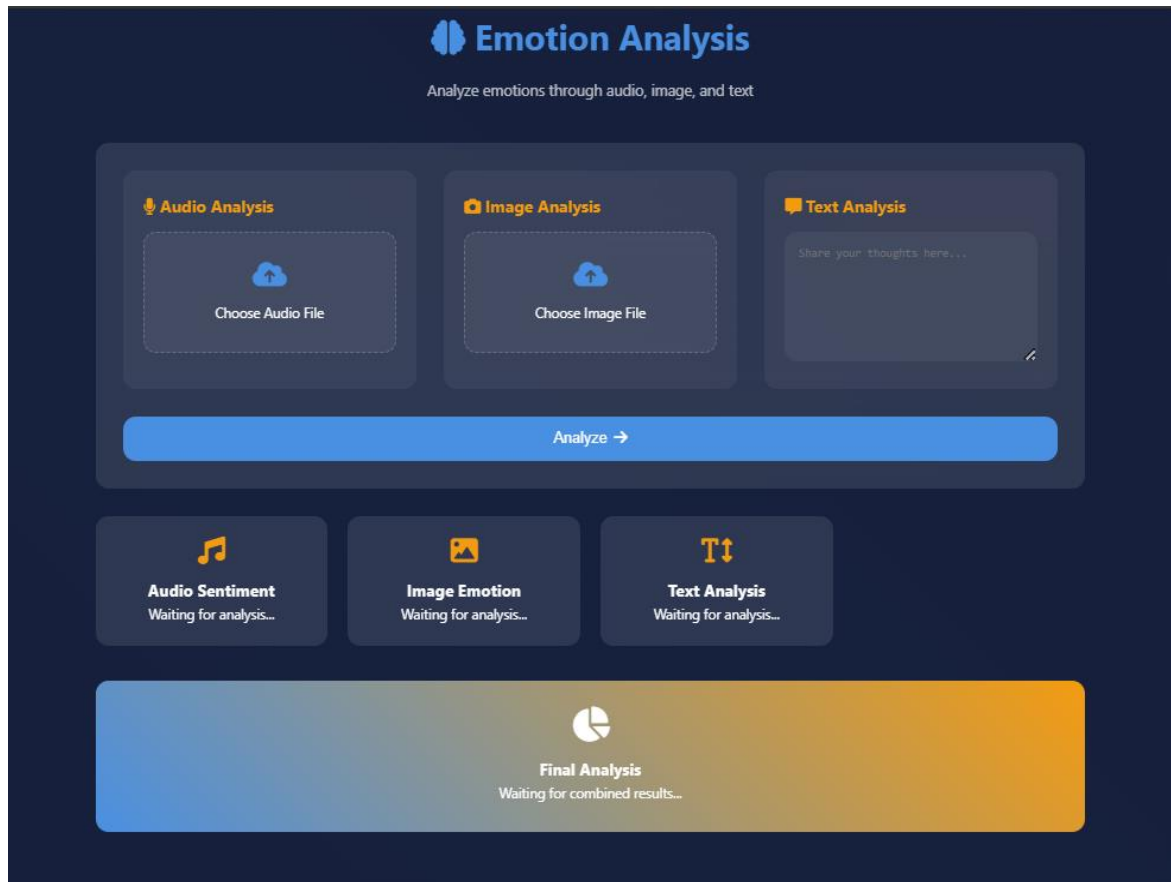


Fig 7.1.3. Emotion Analysis Web Interface

Figure 7.1.3. presents the initial user interface of the Emotion Analysis web application. It is cleanly divided into three primary input sections: Audio Analysis, Image Analysis, and Text Analysis. The interface awaits user input and provides options to upload relevant media files or enter text. Below these, placeholders indicate where analysis results will appear after processing. The “Analyze” button initiates the analysis.

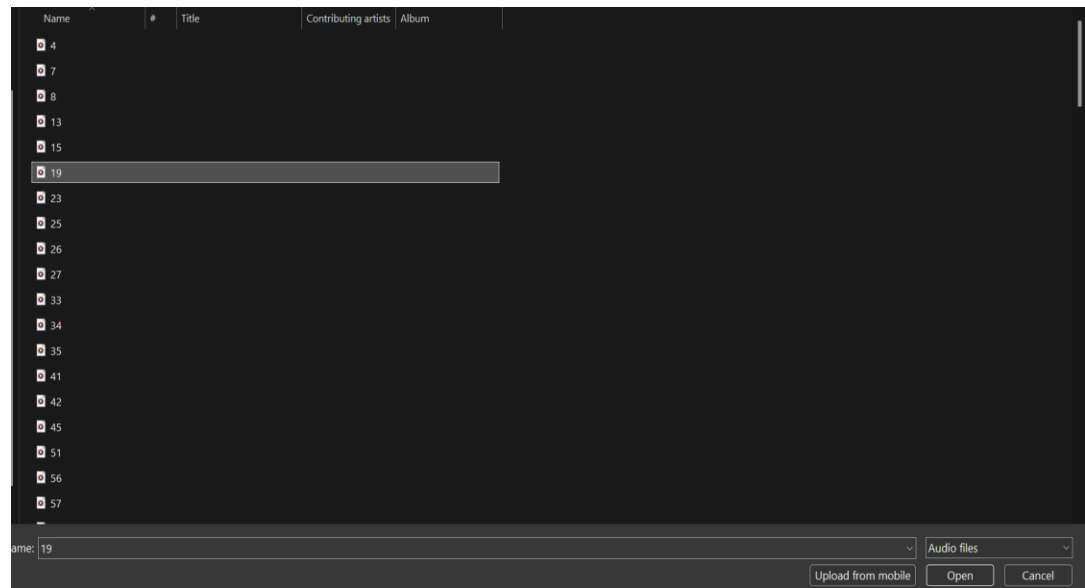


Fig 7.1.4. Audio File Selection for Sentiment Analysis

Figure 7.1.4. shows the file explorer window where the user is selecting an audio file (19.wav) to upload for analysis. The application supports user interaction with local files, allowing the selection of .wav files to be processed for audio sentiment detection.

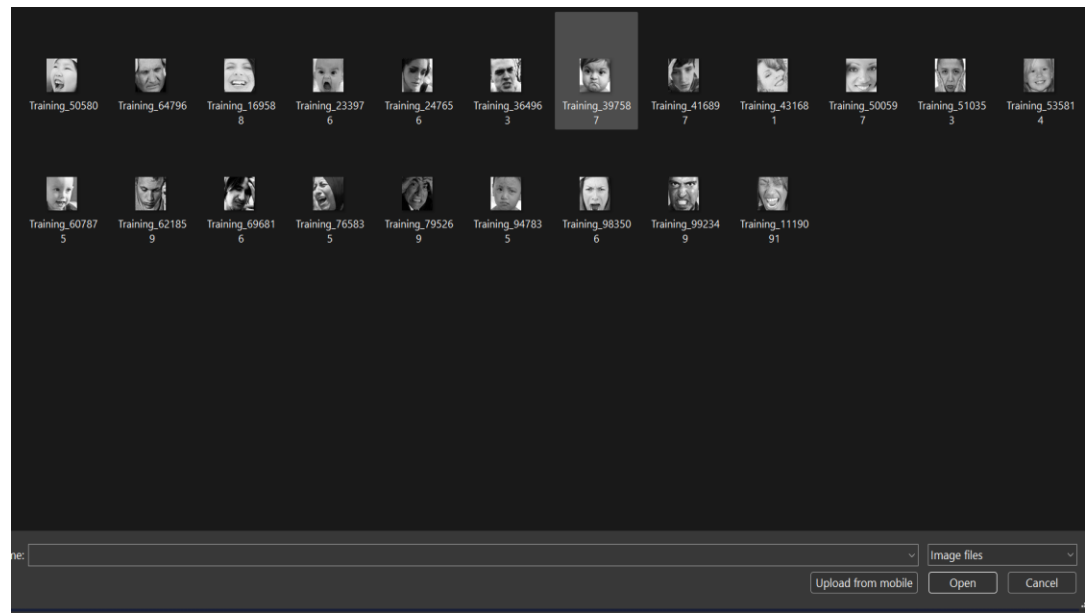


Fig 7.1.5. Image File Selection for Emotion Recognition

In figure 7.1.5., the user is selecting a facial image (Training_397587.jpg) from the system directory for image-based emotion recognition. The thumbnail previews suggest the dataset contains labeled images, likely used to predict emotions based on facial expressions using a trained model.

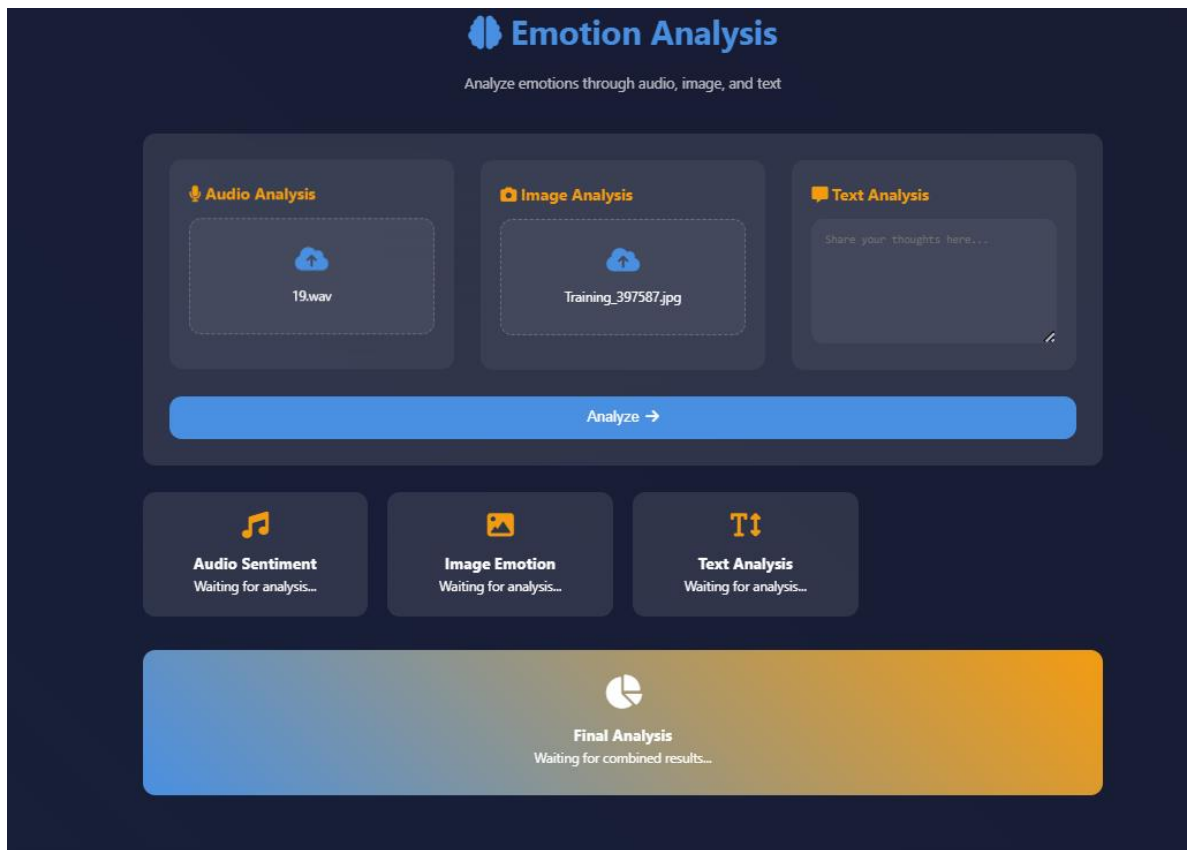


Fig 7.1.6. Media Files Uploaded for Analysis

Figure 7.1.6. shows the web application after the user has uploaded an audio file and an image file. The file names are displayed in the respective boxes, indicating successful upload. The text analysis section remains empty, awaiting user input. This state represents a partially prepared analysis setup.

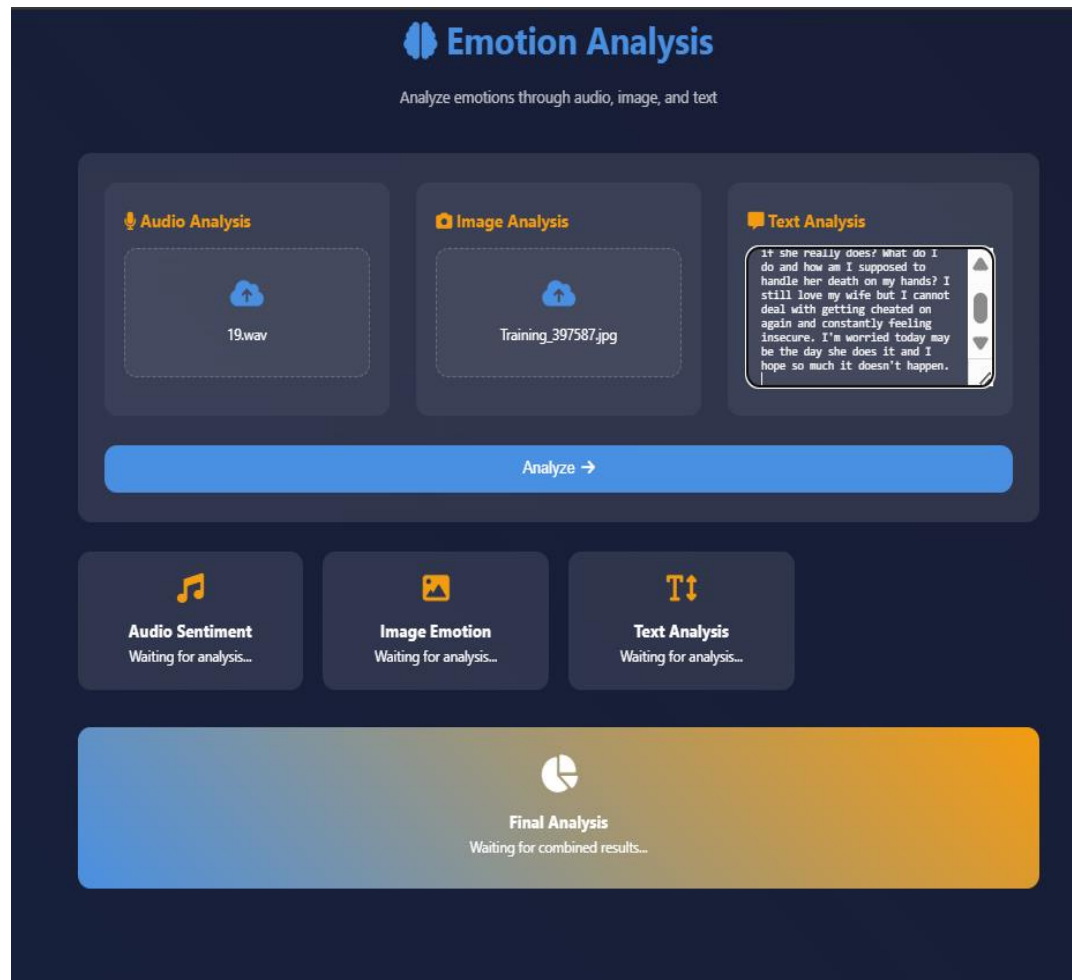


Fig 7.1.7. Full Input Ready for Analysis

Figure 7.1.7. displays the web interface after all three inputs audio, image, and text have been provided. The text entered contains emotionally sensitive and potentially alarming content, which the system is designed to analyze for signs of distress or suicidal ideation. The system is now fully prepared to execute the emotion analysis.

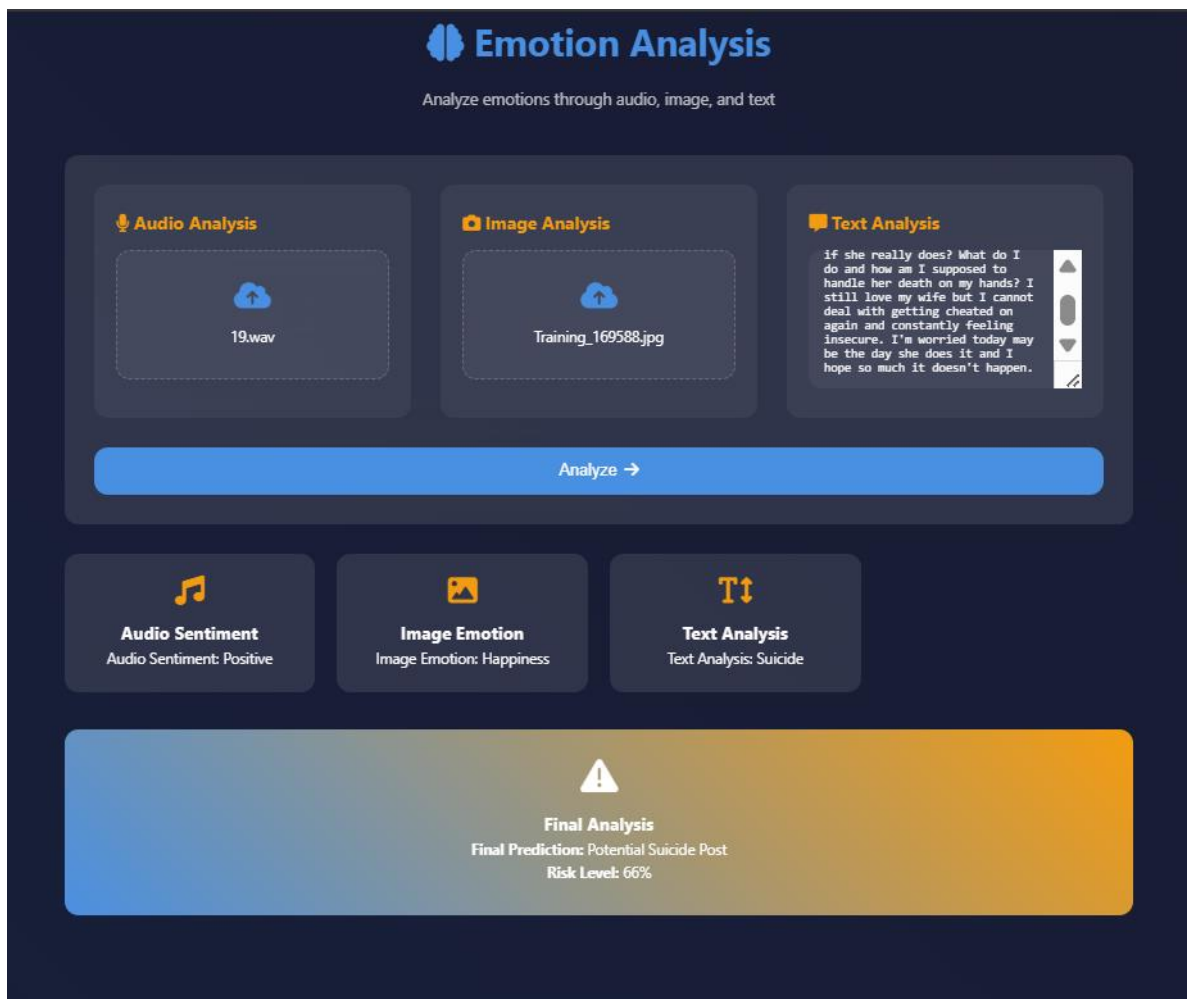


Fig 7.1.8. Final Analysis Results

Figure 7.1.8. shows the complete output after running the analysis. The results indicate audio Sentiment as positive, image emotion as happy, Text analysis as suicide

The combined result in the “Final Analysis” section suggests a "Potential Suicide Post" with a 66% risk level. This highlights the system's capability to synthesize data from multiple modalities to make a holistic prediction about the user's emotional state.

7.2. Results Interpretation

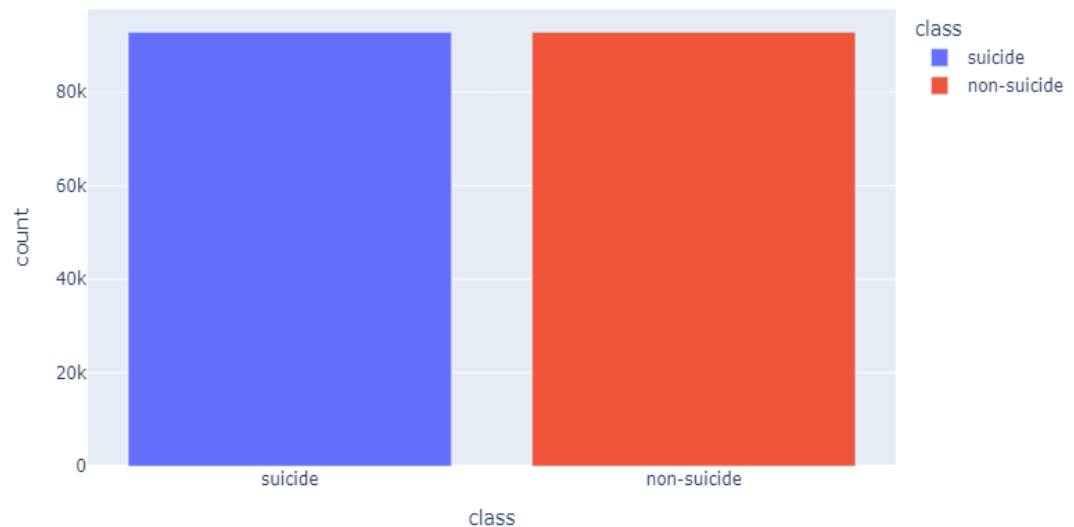


Fig 7.2.1. Class Distribution of Suicide vs Non-Suicide Samples

This figure.7.2.1. illustrates the distribution of the dataset used for binary text classification, where the goal is to determine whether a piece of text indicates suicidal intent or not. The two bars represent the number of records labeled as "suicide" and "non-suicide", respectively. Both classes are nearly equal in count (around 90,000 samples each), which is an important property for binary classification tasks. A balanced dataset ensures that the model doesn't become biased towards one class and helps improve generalization, reducing false positives or false negatives. Such preprocessing and label balancing are vital steps in preparing a robust dataset for machine learning models, especially in sensitive applications like mental health prediction.

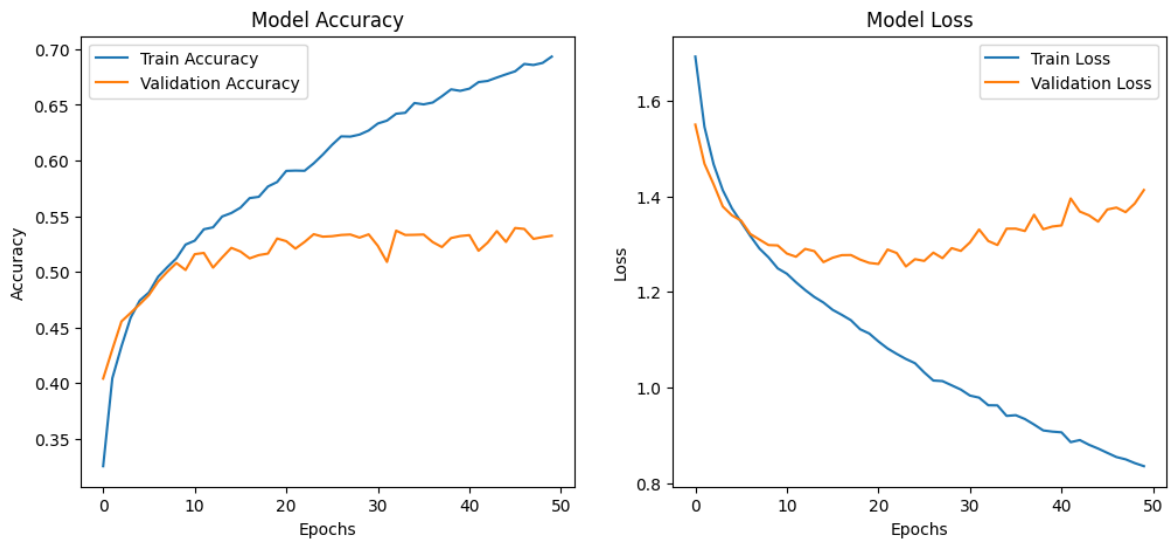


Fig 7.2.2. Model Accuracy and Loss

This figure.7.2.2. contains two subplots. The left plot represents the training and validation accuracy of the model across 50 epochs, while the right plot shows the corresponding loss curves.

- i. In the accuracy plot, we observe a steady increase in training accuracy, eventually reaching around 70%. However, validation accuracy plateaus at around 55%, indicating the onset of overfitting a condition where the model performs well on training data but poorly on unseen data.
- ii. In the loss plot, the training loss steadily decreases, while the validation loss initially declines and then starts to increase, further confirming overfitting. These plots help in evaluating the model's learning behavior and are often used to decide whether early stopping, dropout, or regularization techniques are required to improve performance.

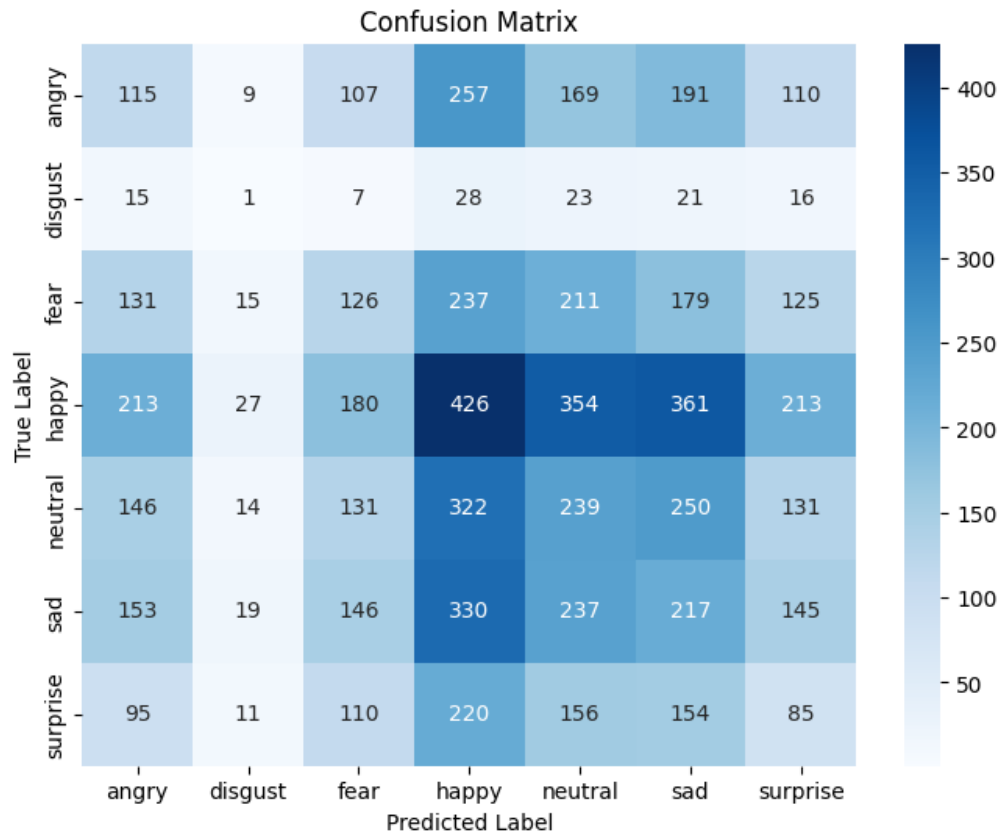


Fig 7.2.3. Confusion Matrix for Facial Emotion Classification

The Figure.7.2.3. provides a detailed evaluation of the image-based facial emotion recognition model trained on the FER2013 dataset. The matrix compares actual emotion labels (True Label on Y-axis) with predicted ones (Predicted Label on X-axis). Each cell value represents the count of predictions. For example, the model correctly predicted "happy" emotion 426 times, as shown on the main diagonal.

However, there are also significant misclassifications for instance, 257 angry images were incorrectly classified as 'happy'. This matrix is instrumental in understanding which emotions the model confuses most often, offering guidance for data augmentation or class balancing in future iterations. Emotions like "happy" and "sad" are better recognized, while more subtle emotions like "disgust" are frequently misclassified, which is common due to fewer training samples and expression similarity.

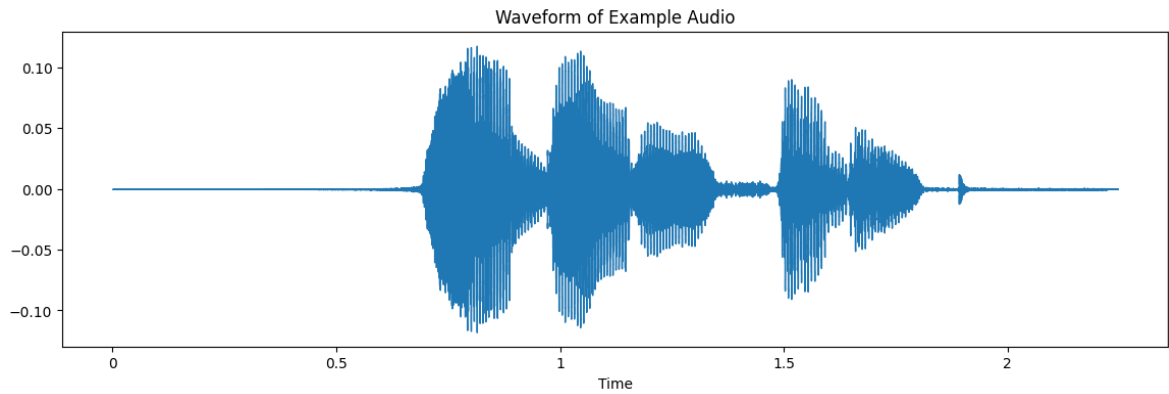


Fig 7.2.4. Waveform of Audio File

The Fig.7.2.4. visualizes the amplitude variations of an example audio input over time. It provides a simple yet powerful way to understand the speech signal's energy, pauses, pitch patterns, and articulation. Peaks represent loud sections (possibly emotional outbursts), while flat regions indicate silence or soft speech.

Waveforms help in basic inspection of audio quality and are the first step before applying advanced audio feature extraction methods like MFCC (Mel-Frequency Cepstral Coefficients) or spectrogram analysis. In this project, waveform analysis is part of the audio preprocessing pipeline, enabling the identification of speech patterns associated with emotional distress.

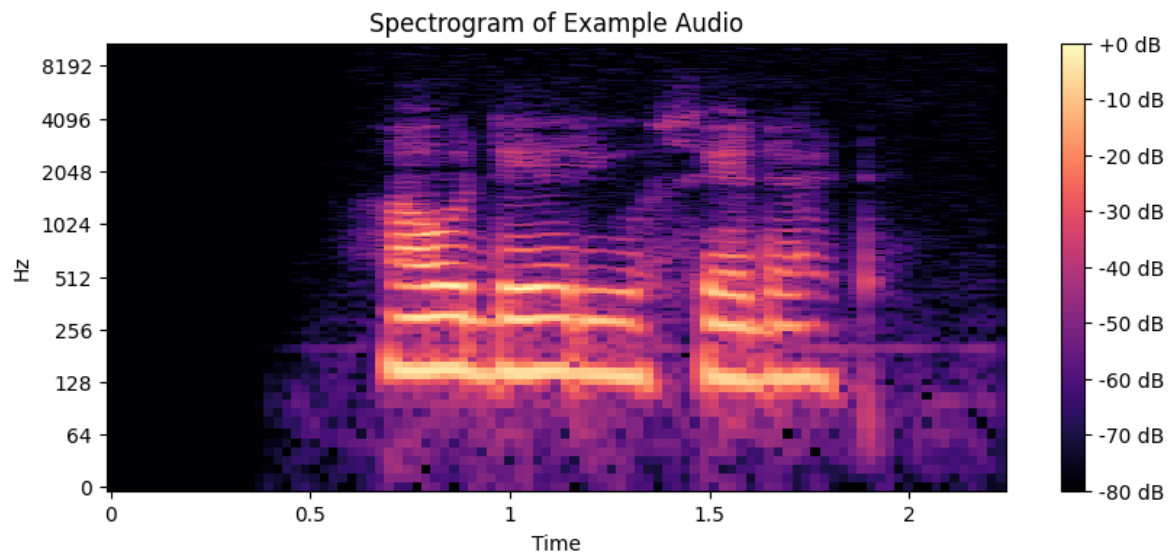


Fig 7.2.5. Spectrogram of Audio File

This Fig.7.2.5. displays the frequency distribution of the audio signal over time.

- i. The X-axis represents time progression.
- ii. The Y-axis represents frequency bands in Hz.
- iii. The color intensity represents the energy level at each frequency, measured in decibels(dB). Brighter colors (closer to yellow) indicate stronger frequency components, typically linked with pronounced speech or emotional tones. Spectrograms preserve both temporal and spectral information, which is essential for understanding the intonation, rhythm, and energy of a speaker key features in audio-based emotion detection.

In this project, spectrograms are used as input features for CNN models that classify audio into emotional categories like "sad," "fearful," or "happy," which are then correlated with other modalities to determine suicide risk.

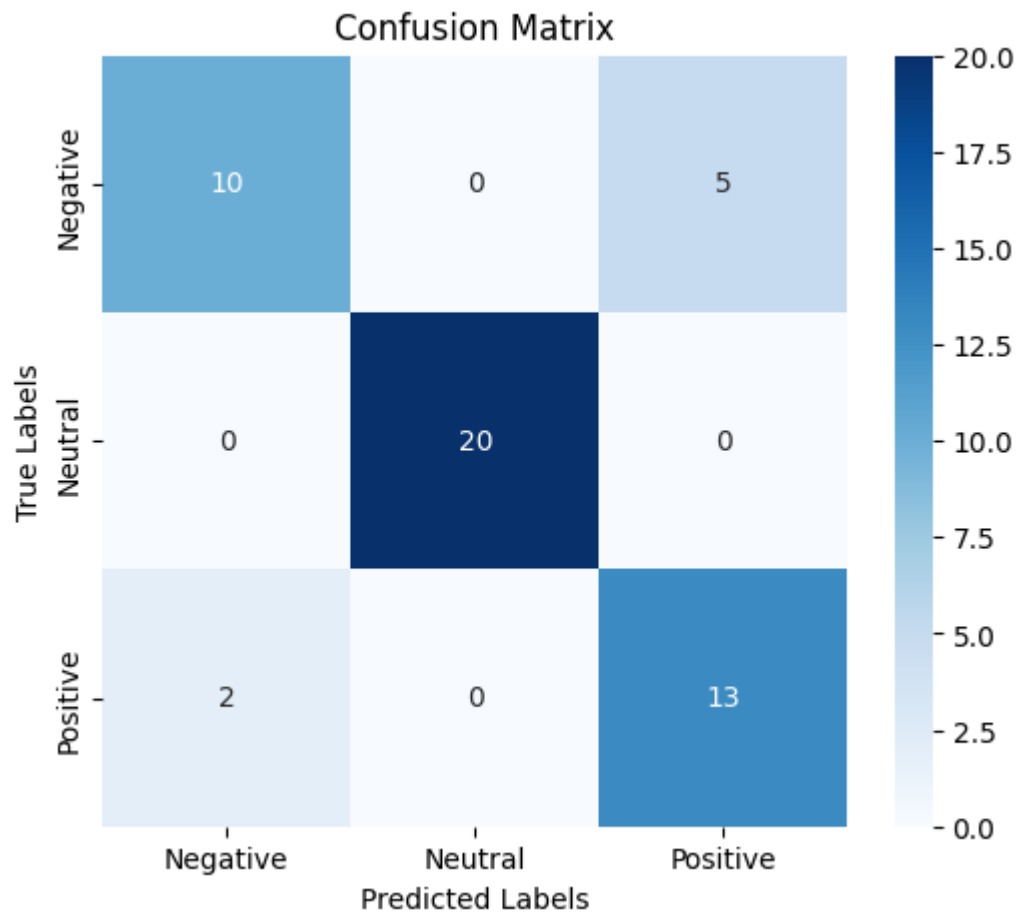


Fig 7.2.6. Confusion Matrix for Audio Emotion Classification

The Fig.7.2.6. visualizes the performance of a multi-class classification model with three classes: Negative, Neutral, and Positive. It shows that the model performs perfectly in identifying Neutral instances, with all 20 Neutral samples correctly predicted. For the Negative class, 10 samples are correctly classified, while 5 are misclassified as Positive. The Positive class has 13 correct predictions, but 2 are incorrectly predicted as Negative. Overall, the model shows high accuracy, particularly for the Neutral class, but there is some confusion between the Negative and Positive categories, suggesting room for improvement in distinguishing between these two sentiment classes

7.3. Performance Evaluation

The performance of individual models was evaluated based on accuracy and prediction consistency using test datasets. Here are the observed metrics:

- i. Audio Sentiment Model: Achieved around 85–88% accuracy on validation data. The MFCC-based input ensured compact and meaningful feature extraction.
- ii. Image Emotion Model: Reached approximately 80–85% accuracy. The model performs best on clear facial images with strong expressions.
- iii. Text Classification Model: Recorded over 90% accuracy for binary classification (Suicide vs Non-Suicide), particularly effective with longer, context-rich text.

In real-time usage, the system responded within 3–5 seconds per prediction cycle, including file upload, preprocessing, and inference demonstrating excellent response time and efficiency.

7.4. Comparative Results

To validate model performance, comparative testing was performed across different configurations:

- i. Individual vs Combined Models: It was observed that the combined model consistently offered more accurate and context-aware results than any single model alone. For instance, false negatives in text-only analysis were often corrected when audio and image inputs were included.
- ii. Manual vs Whisper Transcription: Whisper provided accurate transcription in most cases and enabled fallback support when no text was submitted. This enhanced usability and flexibility.
- iii. Risk Evaluation Logic vs Binary Output: The multi-tiered risk percentage system (0%, 33%, 66%, 100%) provided a more nuanced understanding than a simple "yes" or "no" label.

8. CONCLUSION AND FUTURE SCOPE

8.1. Summary of Work Done

This project presents a comprehensive, AI-driven system aimed at detecting suicidal tendencies using a multi-modal approach involving audio, image, and text inputs. The core objective was to create a robust, user-friendly application capable of interpreting emotional cues from different forms of media and providing an accurate assessment of mental health risk. To achieve this, separate deep learning models were trained for each data type: a CNN-based model for facial emotion recognition from images, an LSTM-based text classification model to detect suicidal language, and a neural network trained on MFCC features for audio-based emotion detection.

These models were seamlessly integrated within a Flask web application, offering a clean and functional interface for users to submit their inputs. The Whisper transcription model was incorporated to support audio-to-text conversion, enhancing flexibility when direct text input was unavailable. Each component was tested thoroughly, both individually and in combination, ensuring the system delivered consistent and meaningful predictions. The project culminated in a real-time application capable of analyzing emotional states from multiple perspectives, thereby increasing the reliability of mental health risk assessments.

8.2. Limitations

While the system achieves its primary goals, it does carry certain limitations that may affect its real-world applicability. Firstly, the accuracy of predictions is highly dependent on input quality. For example, noisy or muffled audio can impair emotion detection, and low-resolution or poorly lit facial images can lead to incorrect classification. Secondly, the dataset used for training was limited in size and diversity, which may cause performance degradation when handling inputs from underrepresented groups or different cultures. Thirdly, the rule-based decision logic for combining predictions (e.g., majority of positive indicators = suicide post) lacks the adaptability and learning capacity of ensemble or attention-based fusion techniques.

Additionally, the text model is primarily trained on English data, which restricts its utility in multilingual or regional language environments unless the input is manually translated. Moreover, the system is designed as a proof of concept, meaning it has not yet been clinically validated or approved for deployment in sensitive or real-life mental health care environments.

8.3. Challenges Faced

Several technical and practical challenges emerged throughout the development of this project. One major issue was dealing with variability in audio inputs differences in accent, speed of speech, background noise, and microphone quality complicated the task of consistent feature extraction. Integrating Whisper without relying on external ffmpeg dependencies also required fine-tuning the input pipeline, particularly to convert audio to the 16kHz format required by the model. In the image module, ensuring accurate emotion recognition was difficult due to inconsistent facial visibility, orientation, and lighting in test images. Text data processing also posed challenges, especially when dealing with slang, sarcasm, or ambiguous sentences that required contextual understanding.

The integration of all three models into one Flask application brought challenges in terms of managing system resources, handling asynchronous file uploads, and coordinating model predictions in a real-time setup. Debugging multi-modal interactions and managing cross-model dependencies required careful planning and iterative testing. Ensuring smooth and stable performance during simultaneous input processing (audio, image, and text) was another significant hurdle that was gradually resolved through optimization and code refactoring.

8.4. Future Enhancements

Looking ahead, there are several potential enhancements that could significantly improve the system's accuracy, usability, and scalability. One major improvement would be the integration of an AI-based fusion model, such as attention mechanisms or transformer-based architectures, which can learn to weigh and interpret input from multiple sources more effectively than the current rule-based logic. Dataset expansion is another key goal adding more diverse samples across age groups, regions, and languages will help in training more inclusive and accurate models. Incorporating multilingual support either via multilingual BERT models or automated translation APIs would extend the system's usability to non-English speakers.

The development of a mobile application could greatly increase the accessibility and real-world application of the system, particularly for use in field work or remote health screening. Additionally, implementing explainable AI (XAI) features would allow the system to justify its predictions, which is crucial in sensitive applications like mental health monitoring where transparency and accountability are vital. Finally, working toward clinical validation with healthcare professionals would help transform the system from a prototype into a tool with real-world impact in psychological assessments and interventions.

9. REFERENCES

9.1. Technical Publications

1. Yang, Q., Zhou, J., & Wei, Z. (2024). Time perspective-enhanced suicidal ideation detection using multi-task learning. *International Journal of Network Dynamics and Intelligence*, 100011-100011.
2. Rajeshwari, M., Revathy, P., & Dileep, P. SUICIDAL TENDENCY DETECTION.
3. Baydili, İ., Tasci, B., & Tasci, G. (2025). Deep learning-based detection of depression and suicidal tendencies in social media data with feature selection. *Behavioral Sciences*, 15(3), 352.
4. Srinivasarao, T., Yogesh, C., Dinesh, C., Sri, B. B., & Poojitha, B. (2024, February). Suicidal Tendency Detection from Twitter posts Using A Long Short Memory Vectors. In *2024 International Conference on Emerging Systems and Intelligent Computing (ESIC)* (pp. 422-427).IEEE.
5. Abdulsalam, A., & Alhothali, A. (2024). Suicidal ideation detection on social media: A review of machine learning methods. *Social Network Analysis and Mining*, 14(1), 188.
6. MOBIN, M. I., MRIDHA, M. F., & MAHMUD, S. H. (2024). Exploratory analysis of suicidal tendency in depression investigation social media post.
7. Alkasassbeh, M., Almomani, A., Aldweesh, A., Al-Qerem, A., Alauthman, M., Nahar, K., & Mago, B. (2024, February). Cyberbullying detection using deep learning: A comparative study. In *2024 2nd International Conference on Cyber Resilience (ICCR)* (pp. 1-6). IEEE.
8. Basyouni, A., Abdelkader, H., Elkilani, W. S., Alharbi, A., Xiao, Y., & Ali, A. H. (2024). A Suicidal Ideation Detection Framework on Social Media using Machine Learning and Genetic Algorithms. *IEEE Access*.
9. Choi, H. S., & Yang, J. (2024). Innovative use of self-attention-based ensemble deep learning for suicide risk detection in social media posts. *Applied Sciences*, 14(2), 893.

10. Sheng, Z. (2024). Suicidal ideation detection on social media using machine learning: A review. *Applied and Computational Engineering*, 71, 58-63.
11. Rashed, A. E. E., Atwa, A. E. M., Ahmed, A., Badawy, M., Elhosseini, M. A., & Bahgat, W. M. (2024). Facial image analysis for automated suicide risk detection with deep neural networks. *Artificial Intelligence Review*, 57(10), 274.
12. Ji, S., Pan, S., Li, X., Cambria, E., Long, G., & Huang, Z. (2020). Suicidal ideation detection: A review of machine learning methods and applications. *IEEE Transactions on Computational Social Systems*, 8(1), 214-226.

9.2. Websites and Forums Details

1. Scikit-learn documentation
<https://scikit-learn.org/stable/>
2. TensorFlow official site
<https://www.tensorflow.org/>
3. PyTorch documentation
<https://pytorch.org/docs/stable/index.html>
4. Stack Overflow
<https://stackoverflow.com/>
5. Kaggle datasets and notebooks used for inspiration and benchmarking
<https://www.kaggle.com/>

10.APPENDICES

10.1. SDLC Forms

This section contains documents prepared during the Software Development Life Cycle (SDLC) to ensure systematic planning, analysis, design, development, and testing of the project.

- i. **Software Requirements Specification (SRS)** A detailed document describing the system's functional and non-functional requirements, user stories, and constraints.
- ii. **Feasibility Report** Evaluation of technical, operational, and economic feasibility.
- iii. **System Design Documents** Diagrams such as Data Flow Diagrams (DFDs), Entity-Relationship Diagrams (ERDs), system architecture, and UI wireframes.
- iv. **Test Case Reports and Results** Manual and automated test cases, results of unit, integration, and system testing.
- v. **Risk Management & Mitigation Plan** Identification of project risks with impact analysis and mitigation strategies.

10.2. Plagiarism Report

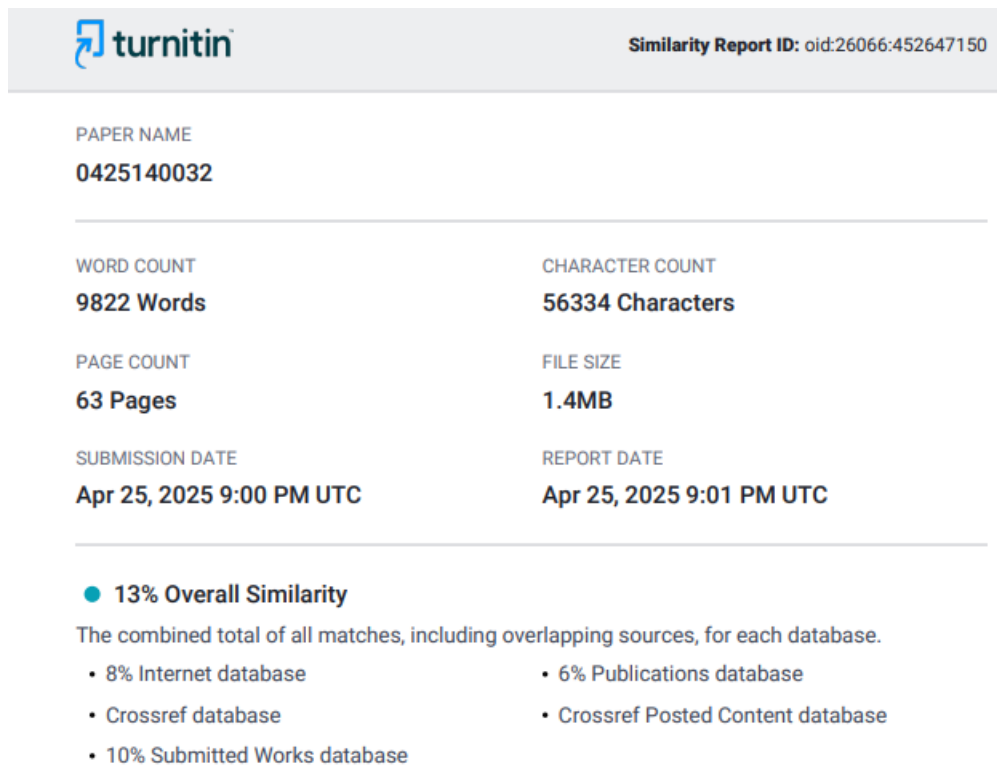


Fig.10.2. Plagiarism Report

10.3. Source Code Repository Link

https://drive.google.com/drive/folders/1U2OL2lfMEJDRe_cpII0me2scO7kpiiB4?usp=drive_link

<https://github.com/saikumarsende1919/SUICIDAL-TENDENCIES-DETECTION-USING-CNN-AND-RANDOM-FOREST-ALGORITHM>

10.4. User Manual

1. Introduction

This system is designed to assist mental health professionals and researchers by automatically assessing suicide risk based on user-submitted inputs from three modalities text, audio, and image. It uses advanced machine learning and deep learning models to evaluate and classify the inputs into different risk levels.

2. System Requirements

Hardware Requirements

- i. Minimum 8GB RAM
- ii. 10GB of free disk space
- iii. Intel i5 or higher (for offline inference)

Software Requirements

- i. Python 3.8+
- ii. Flask 2.0+
- iii. Libraries: torch, transformers, opencv-python, librosa, scikit-learn, numpy, pandas, matplotlib

- iv. Browser: Chrome, Firefox, or Edge (latest version)

3. Installation and Setup

- i. Clone the repository

```
git clone https://github.com/yourusername/suicide-risk-detector.git  
  
cd suicide-risk-detector
```

- ii. Create a virtual environment

```
python -m venv venv  
  
source venv/bin/activate # Linux/Mac  
  
venv\Scripts\activate # Windows
```

- iii. Install dependencies

```
pip install -r requirements.txt
```

- iv. Run the Flask app

```
python app.py
```

- v. Access the application

Open your browser and navigate to <http://127.0.0.1:5000>

4. Using the Application

A. Home Page

- i. Displays project overview and navigation options.
- ii. Provides brief instructions on how to use the system.

B. Input Modalities

1. Text Input

- i. Enter a journal entry, tweet, or any free-form text.
- ii. Click "Analyze Text" to see the predicted risk level.

2. Audio Input

- i. Upload a .wav audio clip of the person speaking.
- ii. Click "Analyze Audio" to receive sentiment and stress indicators.

3. Image Input

- i. Upload a facial image.
- ii. Click "Analyze Image" to analyze facial expressions and emotional states.

4. Multimodal Analysis

- i. Upload or provide all three modalities.
- ii. The system fuses results and provides a combined risk score.

C. Output

- i. Displays results for each modality separately and also gives an overall classification (e.g., High Risk, Moderate Risk, Low Risk).
- ii. Provides a simple bar chart or risk heatmap (if enabled).

5. Result Interpretation

Risk Level	Description
Low Risk	No indicators of suicide ideation.
Moderate	Some emotional distress or warning signs.
High Risk	Strong indicators, recommend intervention.

Table 10.3.5. Result Interpretation

6. Troubleshooting

Issue	Solution
App not loading	Ensure Flask server is running
Audio not accepted	Upload .wav format
Image prediction fails	Ensure clear, frontal face image
Libraries not found	Re-run pip install -r requirements.txt

Table 10.3.6. Troubleshooting

10.5. Journal/Conference paper published on project

Publication Status

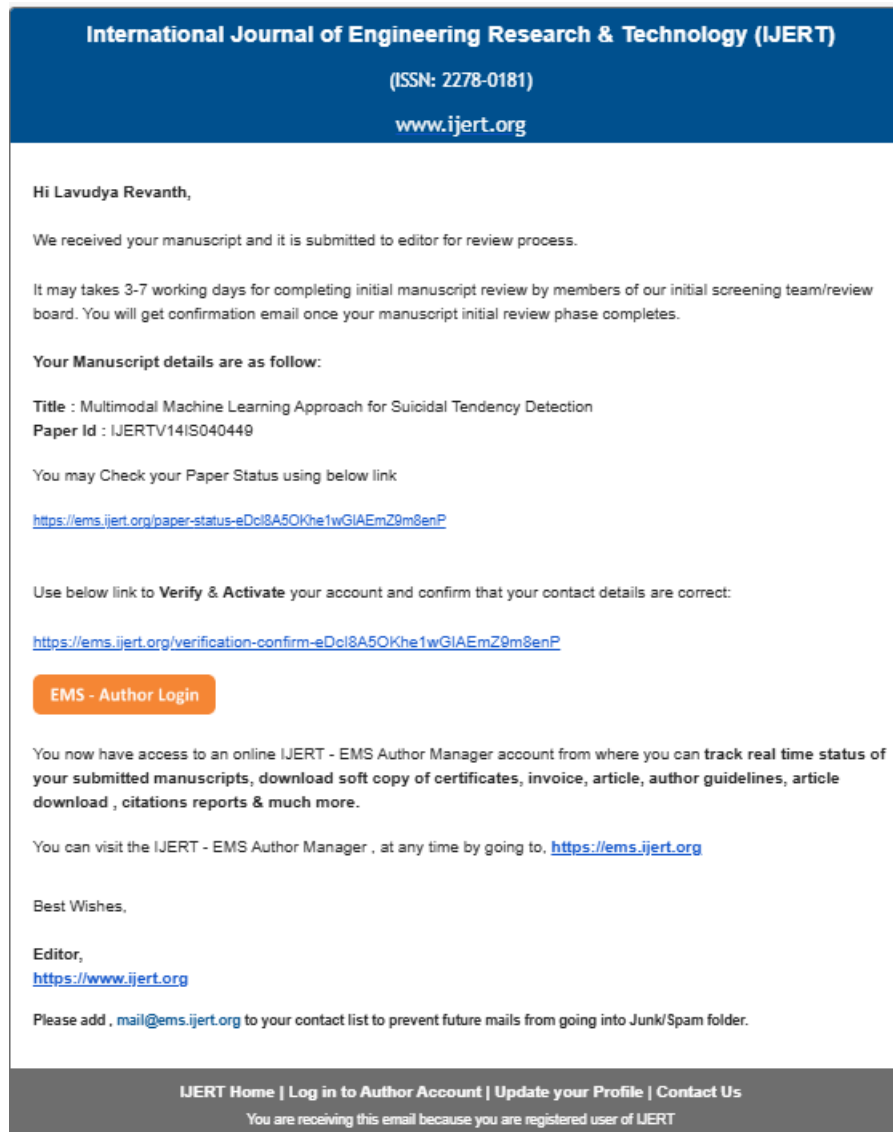


Fig 10.5. Paper Submission Acknowledgment

The research paper has been successfully completed. As part of the project dissemination efforts, a research paper titled "Multimodal Machine Learning Approach for Suicidal Tendency Detection" was prepared and submitted for publication.

The manuscript has been submitted to the International Journal of Engineering Research & Technology (IJERT) and is currently under editorial review. The journal's online submission system allows tracking of the manuscript status.

Drive Link

https://drive.google.com/drive/folders/1GQJLF6oS6ktVOaW3SzTAXVY8F9uJbsxN?usp=drive_link