Emotional Assessment using Biosignals and Voice Patterns

by

Pallavi H H (2240135)
Velan E (2240153)
Arpit Tiwari (2240156)

Under the guidance of
Dr. Sreeja C. S

A Project report submitted in partial fulfillment of the requirements
for the award of degree of Bachelor of Science of
CHRIST (Deemed to be University)

March - 2025

CERTIFICATE

This is to certify that the report titled ***Emotional Assessment using Biosignals and Voice Patterns*** *is a bona fide record of work done by* ***Pallavi H H (2240135), Velan E (2240153), and Arpit Tiwari (2240156)*** *of CHRIST(Deemed to be University), Bengaluru, in partial fulfillment of the requirements of VI Semester BSc CME during the academic year 2024-25.*

Head of the Department **Project Guide**

Valued-by:

	Name	: Pallavi H H, Velan E, Arpit Tiwari
1.	Register Number	: 2240135, 2240153, 2240156
	Examination Centre	: CHRIST(Deemed to be University)
2.	Date of Exam	: 15/03/2025

Acknowledgements

We would like to express our heartfelt thanks to all those who supported and guided us throughout the course of this project.

First and foremost, we are grateful to God Almighty for His blessings, which enabled us to successfully complete this project. Our sincere appreciation goes to Vice Chancellor Dr. Fr. Joseph C. C. and Pro-Vice Chancellor Dr. Fr. Viju P. D. of CHRIST (Deemed to be University) for providing us with the opportunity to apply and enhance our knowledge.

We are deeply thankful to Dr. Ashok Immanuel V., Head of the Department of Computer Science, and Dr. Manjunatha Hiremath, Coordinator of BSc CME, for their constant support and guidance throughout this process. A special thanks to our mentor, Dr. Sreeja C. S., for her invaluable insights, expertise, and continuous encouragement, which played a key role in the successful completion of this project.

We also appreciate the contributions of our Alumni Evaluator, whose industry expertise and advice helped us align our work with professional standards. Lastly, we extend our gratitude to our peers, family, and everyone who contributed, either directly or indirectly, to the successful realization of this project.

Abstract

Mental health disorders, including depression, have become a growing concern, requiring objective and reliable diagnostic tools. This project aims to assist mental health professionals by developing a system that utilizes biosignals and voice patterns to assess emotional states and detect signs of depression.

The system integrates non-invasive physiological signals, such as Photoplethysmography (PPG) and Galvanic Skin Response (GSR), with speech analysis techniques to provide a multimodal assessment approach. A wearable device captures biosignals, while voice data is processed using Mel-Frequency Cepstral Coefficients (MFCC) and classified using a machine learning model.

A web-based interface is developed to display real-time results using Streamlit, providing professionals with actionable data for better diagnosis and treatment planning. By leveraging Web infrastructure and machine learning for emotion classification and detection of depression state, this system enhances the accuracy and reliability of mental health assessments.

Contents

List of Figures

List of Tables

Abbreviations

PPG	Photoplethysmography
GSR	Galvanic Skin Response
BVP	Blood Volume Pulse
MFCC	Mel Frequency Cepstrum Coefficient
CNN	Convolutional Neural Network
LDA	Linear Discriminant Analysis

Chapter 1

Introduction

1.1 Overview of the System

This system is designed to classify emotional states and detect depression using a combination of biosignals and voice pattern analysis. The system integrates non-invasive physiological signals like PPG and GSR with speech processing techniques to provide a reliable tool for mental health assessment. The core objective is to assist mental health professionals by providing objective insights into a person's emotional well-being.

A machine learning model trained on institutional data is used to classify emotional states based on multimodal inputs. The system includes:

- A wearable device that collects biosignals (heart rate and skin conductivity).
- Voice pattern analysis using speech features like Mel-Frequency Cepstral Coefficients (MFCC).
- Machine learning models for emotion classification and depression detection.
- A web interface developed using Streamlit for real-time data visualization.

By leveraging these technologies, the system provides an efficient and objective method for diagnosing emotional states and mental health conditions.

1.2 Problem Statement

Mental health disorders, including depression, are a significant global concern. Traditional methods of diagnosis rely on self-reported data and clinician assessments, which can be subjective and prone to bias. There is a growing need for objective, technology-driven solutions that can enhance the accuracy of emotional assessments and early detection of depression. The proposed system aims to fill this gap by integrating biosignal monitoring with voice pattern analysis to provide a multimodal, AI-powered assessment tool.

Chapter 2

System Analysis

2.1 Literature Review

Emotion recognition using biosignals and voice analysis has gained significant attention in recent years due to its potential applications in mental health assessment. By leveraging machine learning and deep learning models, researchers have successfully extracted and classified emotional states from physiological signals such as Photoplethysmography (PPG), Galvanic Skin Response (GSR), and speech features like Mel-Frequency Cepstral Coefficients (MFCCs).

Biosignal-based emotion classification has been extensively studied. Several studies have used PPG and GSR signals to detect emotional states based on heart rate variability and skin conductivity. Feature extraction methods such as mean, standard deviation, and differential features have been employed to enhance classification accuracy.

Machine Learning Approaches, Various models have been explored for biosignal-based emotion classification. A study trained a KNN on features extracted from PPG and GSR signals, achieving an accuracy of 40%. The classification report showed that emotions like anger, reverence, and romantic love had higher detection rates compared to more subtle emotions.

Feature Extraction in Voice-Based Analysis, Speech emotion recognition (SER) utilizes acoustic features to classify emotions. The MFCC feature extraction method is widely used for capturing frequency-based speech characteristics. The extracted MFCC features are fed into a fully connected neural network classifier for emotion classification.

These models process spectrogram-based input representations of speech signals to extract temporal and frequency-based emotional characteristics.

Multimodal Emotion Detection: Recent advancements have focused on combining biosignals and speech data for more robust emotion classification. Research has shown that integrating physiological and voice-based features improves overall accuracy in detecting emotional states.

Despite significant progress, challenges such as dataset variability, real-time processing, and generalization across different populations remain. Future work aims to improve classification accuracy through deep learning, transfer learning, and real-time multimodal fusion techniques.

Emotion recognition using biosignals and voice analysis has demonstrated promising results in mental health monitoring. While machine learning and deep learning techniques have significantly improved classification performance, further optimization and dataset expansion are necessary to enhance model accuracy.

2.2 Existing System

Current multimodal models for emotional assessment rely on invasive techniques to classify emotional states and detect depression. These systems often depend on clinical-grade devices and controlled environments, limiting accessibility and real-world applicability. Traditional methods focus on self-reported data, which can be subjective and prone to bias. Additionally, many existing systems either analyse voice or physiological data in isolation, reducing overall accuracy.

2.3 Proposed System

The proposed system enhances depression and emotional detection by integrating biosignals (PPG and GSR) with voice pattern analysis, providing a more comprehensive and objective evaluation. The system's key innovations include:

Biosignal Acquisition

- A wearable device continuously monitors blood volume pulse using PPG sensors and skin conductivity via GSR sensors.
- These signals are processed in real-time using a microcontroller to track physiological responses related to emotions and mental health.

Emotion and Depression Classification

- Machine learning models classify emotional states and detect depressive symptoms based on PPG and GSR data and voice, referencing Plutchik’s Wheel of Emotions for classification.
- The system leverages the DAIC-WOZ dataset, which includes structured interviews, to train models for detecting depression.

Audio Preprocessing & Feature Extraction

- Voice recordings undergo noise suppression, gain adjustment, and filtering to refine data quality.
- Mel-Frequency Cepstral Coefficients (MFCC) are extracted to analyse speech patterns for emotion recognition and depression detection.

Mental Health Assessment and Web Interface for Professionals

- A real-time dashboard, developed using Streamlit, displays key parameters such as heart rate, skin conductivity, detected emotions, and depression indicators.
- The system integrates streamlit for web interface.

Web-Based Infrastructure

- The trained machine learning model runs on servers, ensuring high-speed processing, scalability, and seamless access for mental health professionals.

This approach enables a data-driven, objective assessment of emotional states and depression, assisting healthcare professionals in early detection and intervention.

Chapter 3

System Requirements

3.1 System Model

A structured pipeline processes biosignals and voice data, feeding them into a web-based machine learning model.

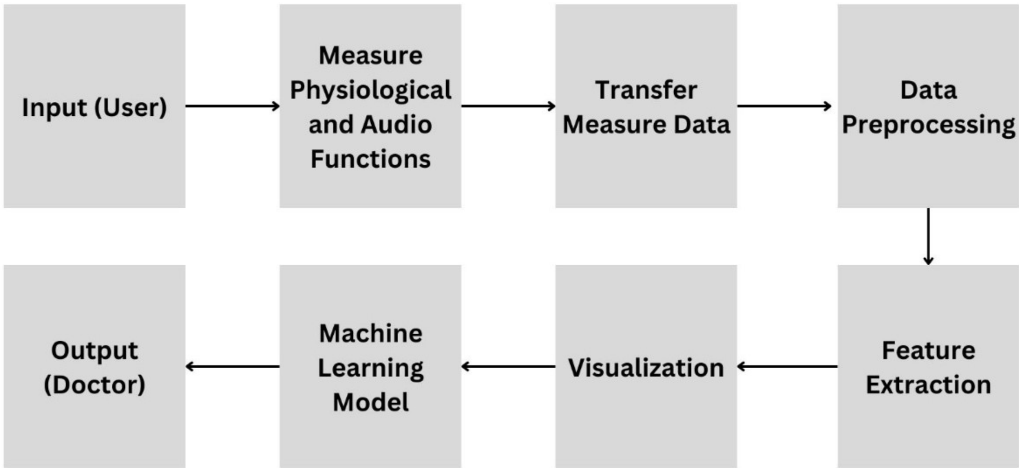

Figure 1: Block diagram

3.2 Functional Requirements

The system is designed to collect and preprocess biosignals and voice data to classify emotional states and detect depression. By analysing physiological signals and voice patterns, it provides real-time assessments of mental health conditions. The system functions by gathering biosignals from PPG and GSR sensors while simultaneously capturing and analysing voice data. These inputs are processed using machine learning models to classify emotional and depressive states. The results are then displayed through a web-based interface, allowing users to monitor emotional and mental health conditions in real time.

3.3 Hardware Requirements

To effectively gather and process biosignals and voice data, the system utilizes specific hardware components. PPG and GSR sensors are employed to measure blood volume pulse and skin conductivity, providing crucial physiological data linked to emotional states. An ESP32 microcontroller is responsible for processing these signals and transmitting them for further analysis. A smartphone serves as a mobile interface, facilitating real-time data visualization and remote monitoring, while a laptop is used for executing data processing tasks, running machine learning models, and displaying results on a dashboard.

3.4 Software Requirements

The software stack for this system consists of various tools for data collection, processing, and visualization. OBS Studio is utilized for screen recording and real-time streaming of voice data, while Virtual Audio Cable facilitates seamless audio routing for voice data collection and processing. VDO.ninja enables remote data collection from a mobile phone and live streaming of voice input. Python serves as the primary programming language for data handling and machine learning implementation, with TensorFlow and Scikit-learn employed to develop models for emotion and depression classification. Streamlit is used to build an interactive web-based dashboard that presents real-time results, ensuring a user-friendly experience. Additionally, Firebase acts as a cloud-based database to securely store and retrieve biosignal and voice data, ensuring scalability and remote access.

Chapter 4

Design Specification

4.1 System Architecture

A modular system connects data acquisition, machine learning processing, and web-based visualization.

Emotion Detection using Biosignals

The Random Forest classifier is used for emotion classification based on biosignals (PPG and GSR). The model is trained on extracted physiological features, including heart rate variability and skin conductivity levels.

Emotion Detection using Voice Model

The deep learning model follows a sequential architecture. The below table summarizes the model structure.

Table 1: Model Architecture of Emotion detection using voice model

Layer (Type)	Output Shape	Parameters
Conv2D	(508, 508, 32)	2,432
MaxPooling2D	(127, 127, 32)	0
Conv2D	(125, 125, 32)	9,248
MaxPooling2D	(62, 62, 32)	0
Flatten	(164000)	0
Dense	(128)	20,992,128
Dropout	(128)	0
Dense	(256)	33,024
Dropout	(256)	0
Dense	(1)	257

Depression Detection using Voice Model

The deep learning model for depression detection consists of fully connected layers, as detailed in Table.

Table 2: Model Architecture of Depression detection using voice model

Layer (Type)	**Input Features**	**Output Features**
Fully Connected (fc1)	27	200
Fully Connected (fc2)	200	200
Fully Connected (fc3)	200	200
Fully Connected (fc4)	200	7

4.2 Data Flow Diagrams

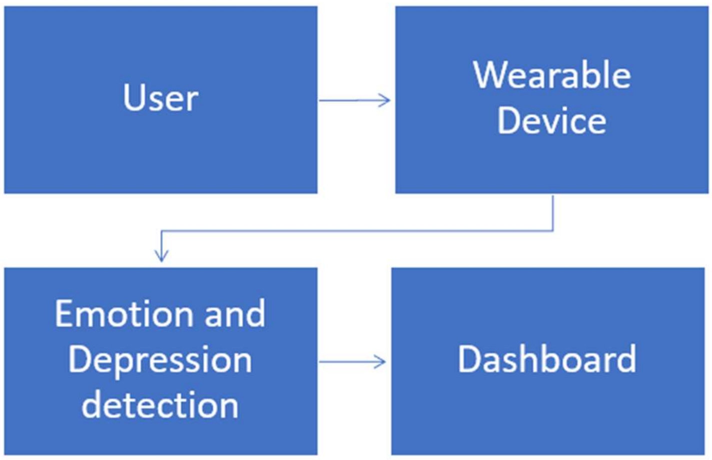

Figure 2.1: Data Flow Diagram (Level 0: Context Diagram)

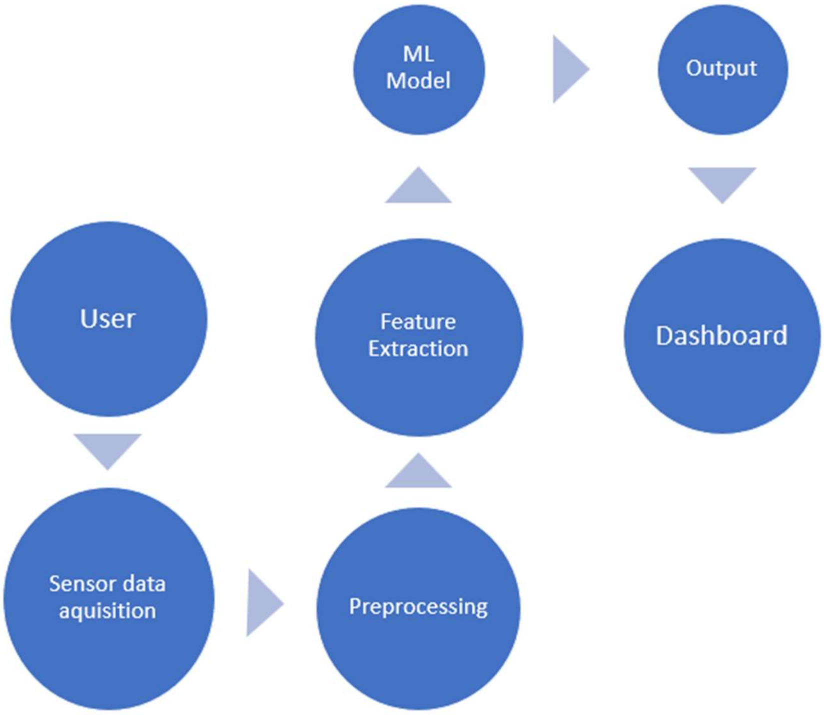

Figure 2.2: Data Flow Diagram (Level 1: Detailed Diagram)

4.3 User Interface Design

Emotion detection using biosignals

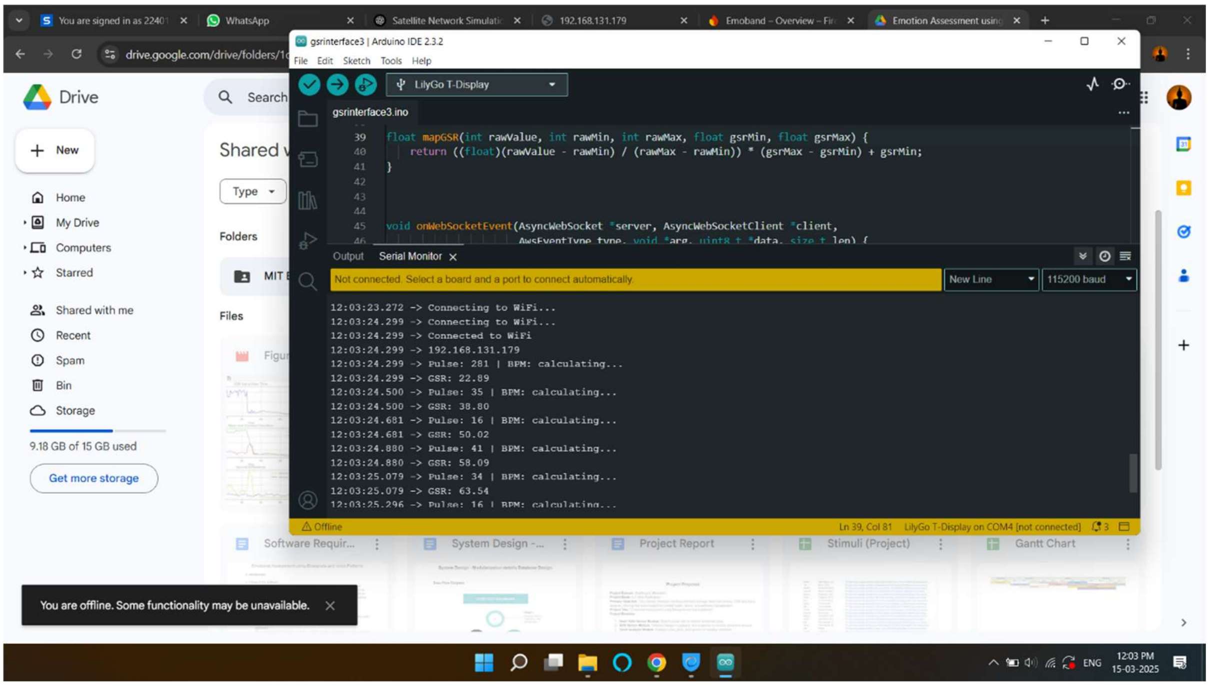

Figure 3.1: ESP IDE

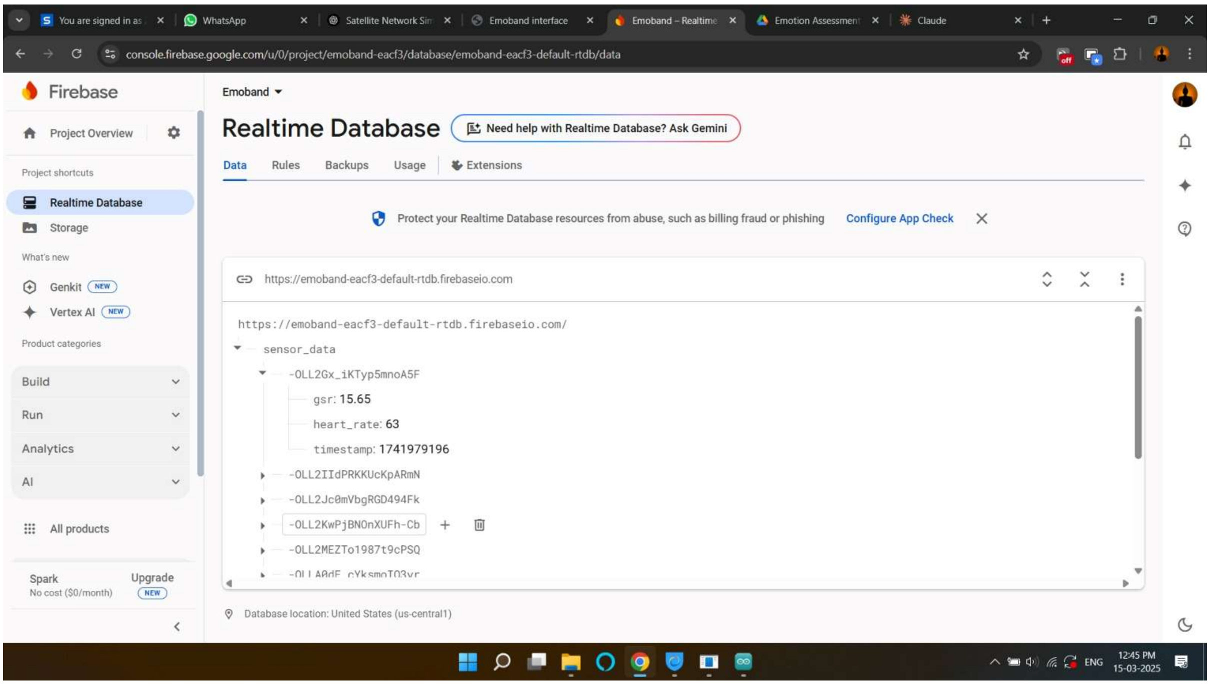

Figure 3.2 Google Firebase

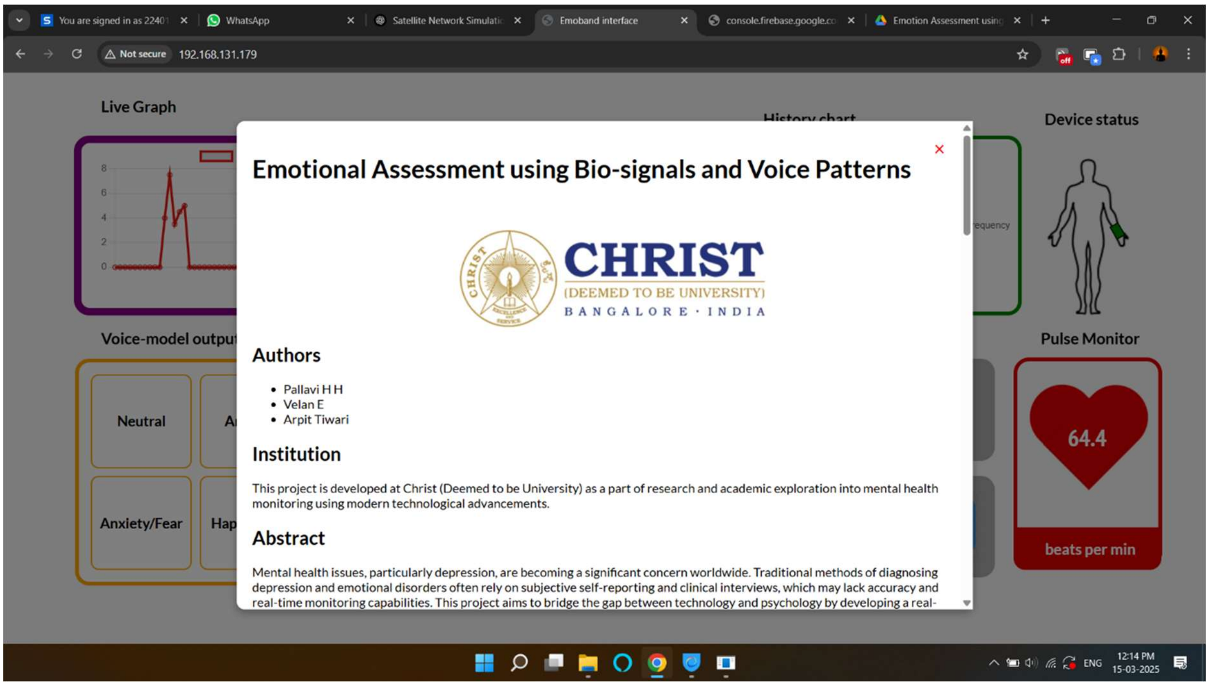

Figure 3.3: Web Interface

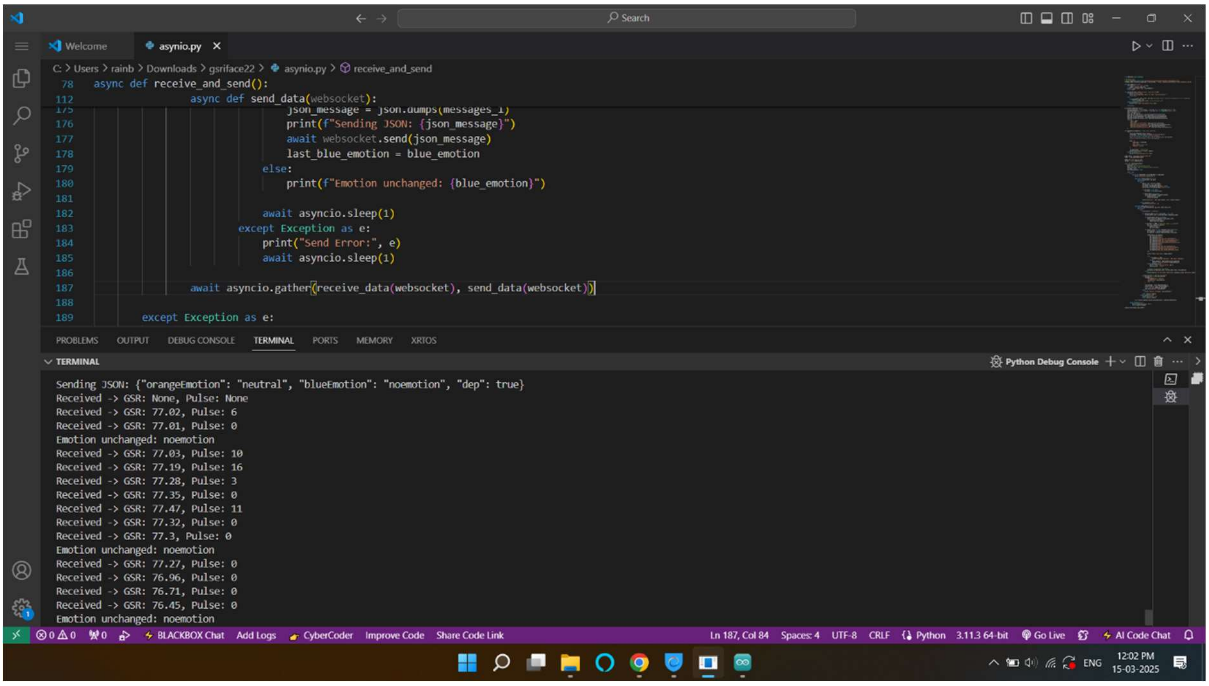

Figure 3.4: Server

Emotion detection using voice

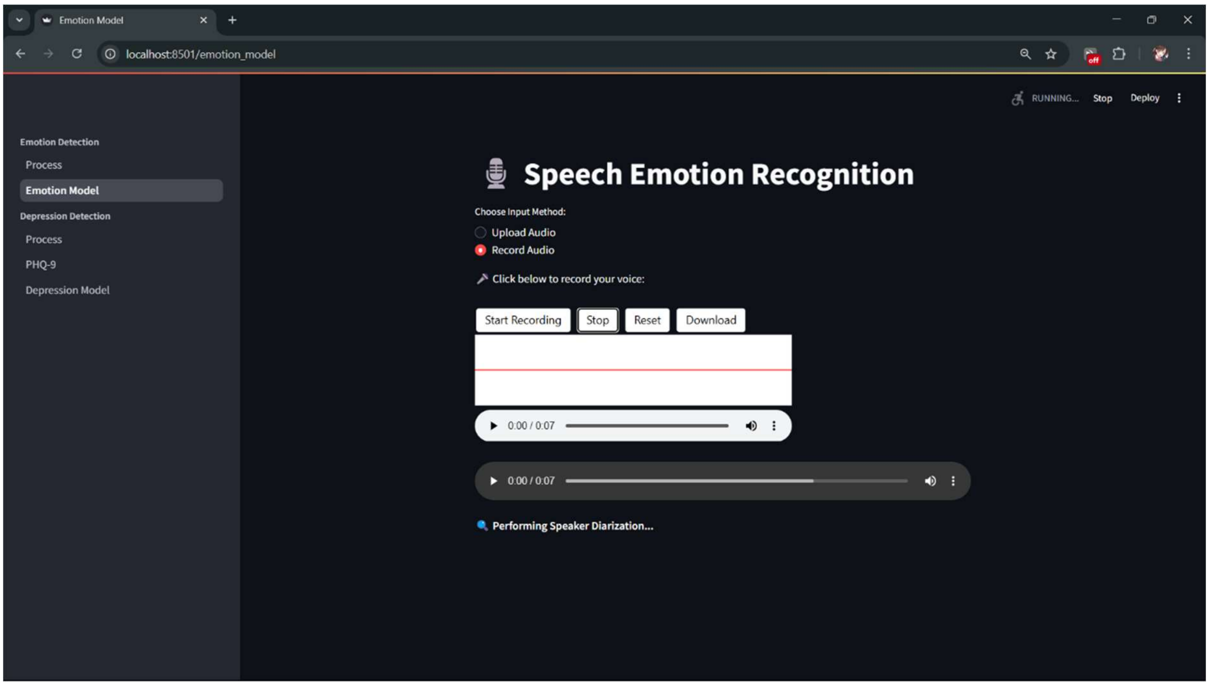

Figure 4.1: Speaker Diarization

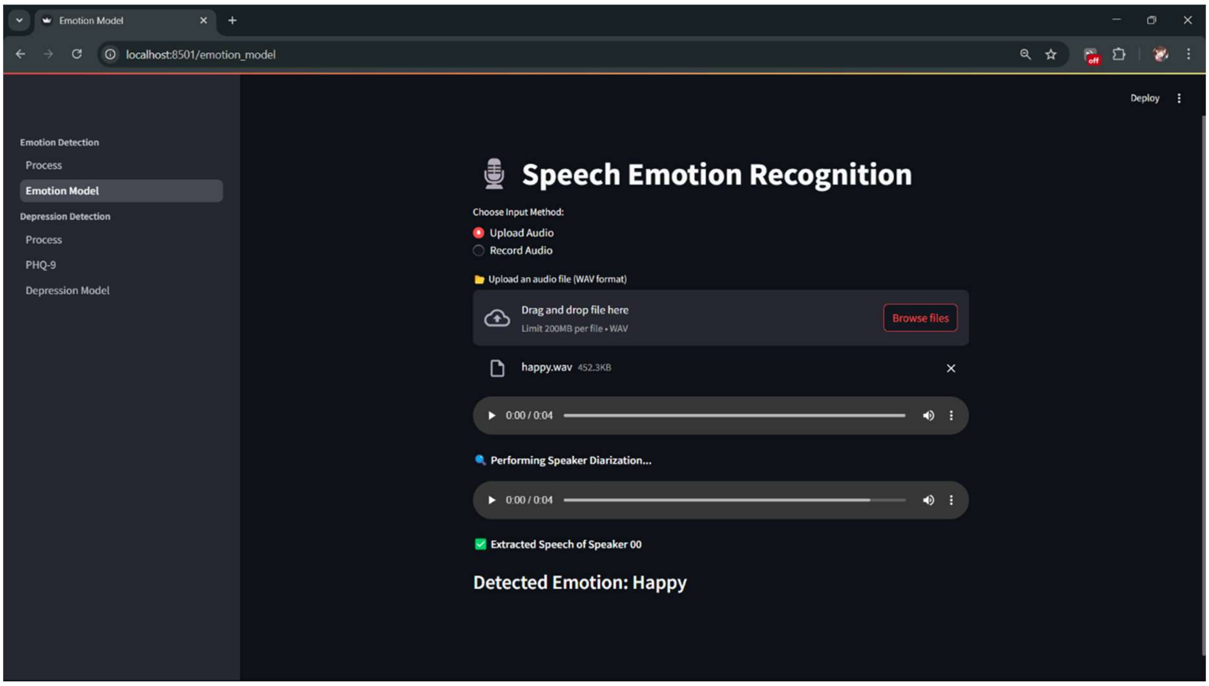

Figure 4.2: Model Prediction for Emotion detection using voice

Depression detection using voice

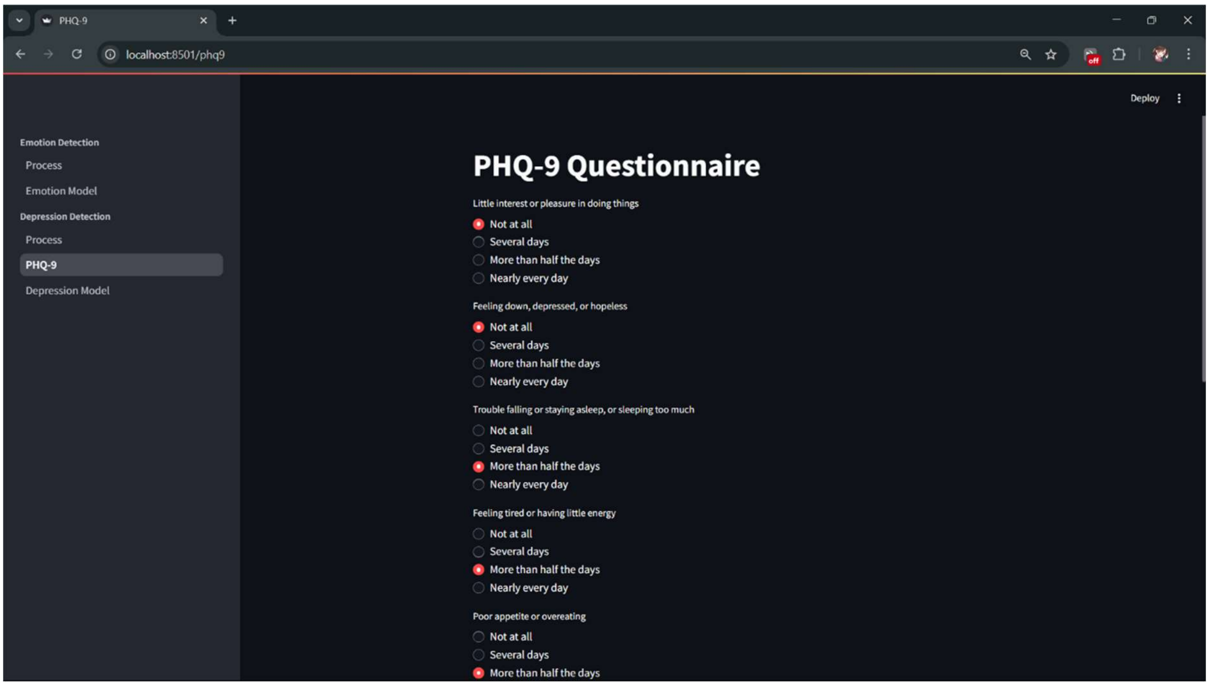

Figure 5.1: PHQ9 Form

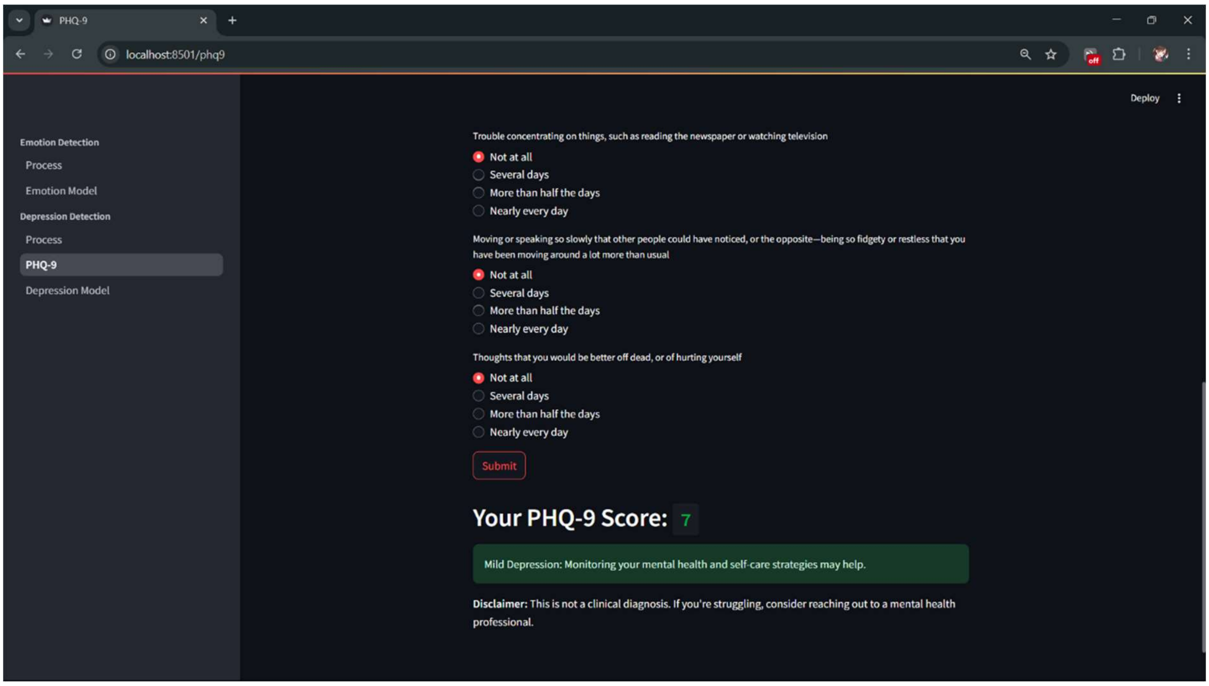

Figure 5.2: PHQ9 Score

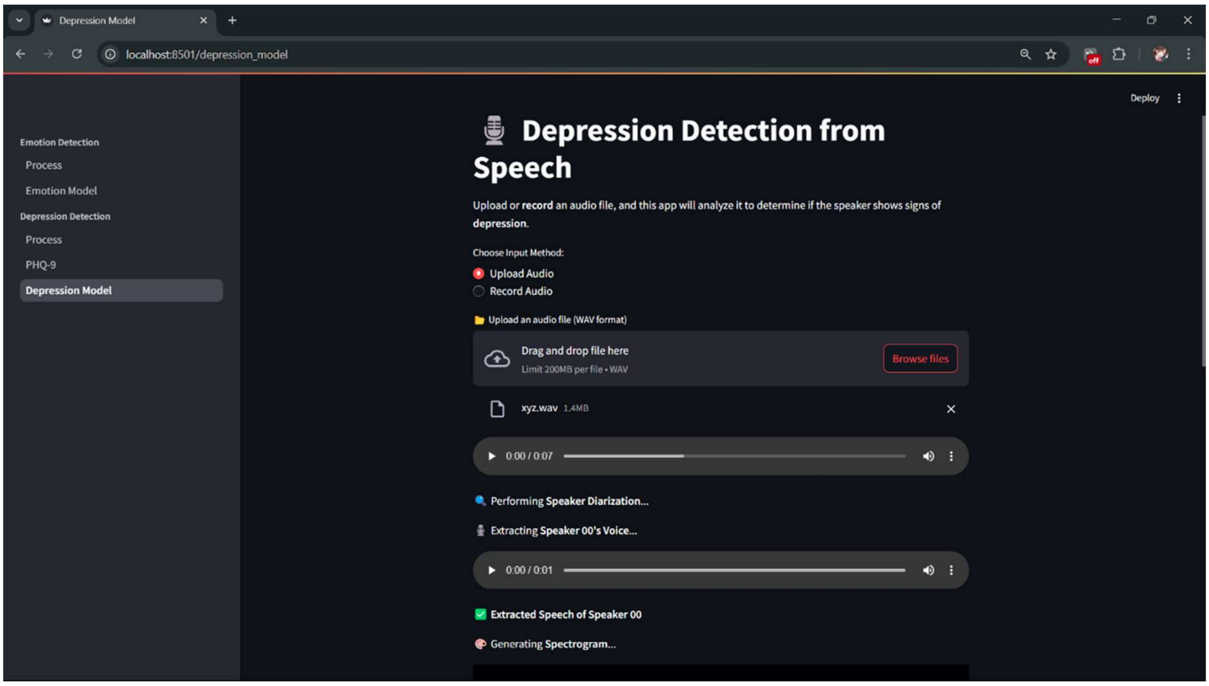

Figure 5.3: Spectrogram Generation

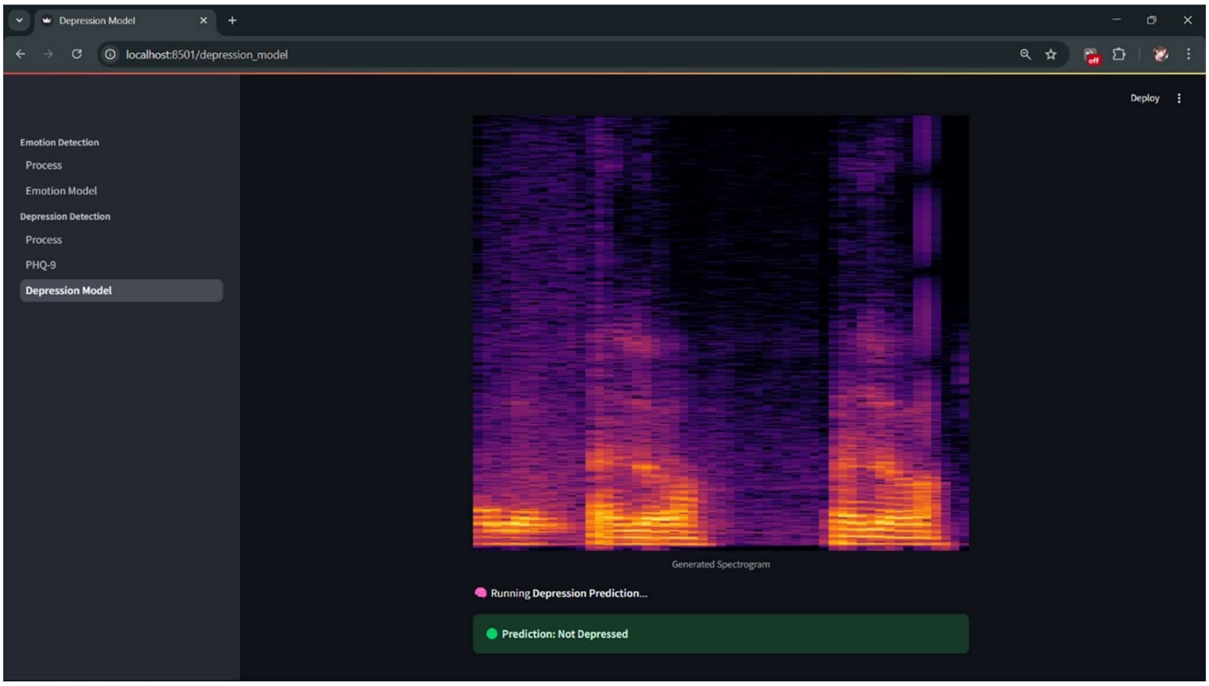

Figure 5.4: Model Prediction for Depression detection using voice

Chapter 5

Implementation

5.1 Emotion detection using biosignals

asynio.py

```
import asyncio
import websockets
import json
import joblib
import numpy as np
import biosppy.signals.bvp as bvp
import firebase_admin
from firebase_admin import credentials, db
from datetime import datetime
from sklearn.preprocessing import StandardScaler
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA

# Firebase setup
cred = credentials.Certificate("emoband-eacf3-firebase-adminsdk-fbsvc-b58ca68a64.json")
firebase_admin.initialize_app(cred, {"databaseURL": "https://emoband-eacf3-default-
rtdb.firebaseio.com/"})


def load_model(pkl_file):
    with open(pkl_file, 'rb') as file:
        model = joblib.load(file)
    print(f"Loaded model type: {type(model)}")
    return model


def calculate_heart_rate(bvp_signal, sampling_rate=20):
    if len(bvp_signal) < 27:
        print("Not enough data points for heart rate calculation.")
        return None
```

```
    try:
        _, _, _, ts_hr, heart_rate = bvp.bvp(signal=bvp_signal, sampling_rate=sampling_rate, 
show=False)
        if len(heart_rate) > 0:
            return np.mean(heart_rate)  # Use mean heart rate
    except Exception as e:
        print(f"Heart Rate Calculation Error: {e}")

    return None
def upload_to_firebase(gsr, heart_rate, emotion):
    try:
        ref = db.reference("sensor_data")
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        gsr = gsr if gsr is not None else "not entered"
        heart_rate = heart_rate if heart_rate is not None else "not entered"
        emotion = emotion if emotion else "not entered"

        data = {
            "timestamp": timestamp,
            "gsr": gsr,
            "heart_rate": heart_rate,
            "emotion": emotion
        }

        ref.push(data)  # Upload Data
        print(f"Uploaded to Firebase: {data}")
    except Exception as e:
        print(f"Firebase Upload Error: {e}")
def extract_features(signal):
    first_difference = [signal[i+1] - signal[i] for i in range(len(signal) - 1)]
    second_difference = [signal[i+2] - signal[i] for i in range(len(signal) - 2)]
    mean = np.mean(signal)
```

```
    std = np.std(signal)
    mean_abs_first_difference = np.mean(np.abs(first_difference))
    mean_abs_first_difference_norm = mean_abs_first_difference/std
    mean_abs_second_difference = np.mean(np.abs(second_difference))
    mean_abs_second_difference_norm = mean_abs_second_difference/std
    return {
        "mean": mean,
        "std": std,
        "mean_abs_first_difference": mean_abs_first_difference,
        "mean_abs_first_difference_norm": mean_abs_first_difference_norm,
        "mean_abs_second_difference": mean_abs_second_difference,
        "mean_abs_second_difference_norm": mean_abs_second_difference_norm
    }

model_path = "Biosignal_Emotion_model.pkl"
model = load_model(model_path)

ESP32_IP = "192.168.131.179"
WS_URL = f"ws://{ESP32_IP}/ws"

async def receive_and_send():
    pulse_data = []
    gsr_data = []

    while True:
        try:
            async with websockets.connect(WS_URL) as websocket:
                print("Connected to WebSocket!")

                async def receive_data(websocket):
                    nonlocal pulse_data, gsr_data
                    while True:
```

```
        try:
            data = await websocket.recv()
            parsed_data = json.loads(data)
            gsr_value = parsed_data.get("gsr", None)
            pulse_value = parsed_data.get("pulse", None)

            if gsr_value is not None:
                gsr_data.append(float(gsr_value))
                if len(gsr_data) > 30000:
                    gsr_data.pop(0)

            if pulse_value is not None:
                pulse_data.append(float(pulse_value))
                if len(pulse_data) > 30000:
                    pulse_data.pop(0)

            print(f"Received -> GSR: {gsr_value}, Pulse: {pulse_value}")

        except Exception as e:
            print("Receive Error:", e)

async def send_data(websocket):
    while True:
        try:
            blue_emotion = "noemotion"

            if len(pulse_data) >= 27 and gsr_data is not None:
                if heart_rate is not None:
                    input_data = []
                    for signal, name in zip([pulse_data, gsr_data], ["BVP", "GSR"]):
                        extracted = extract_features(signal)
                        for _, value in extracted.items():
```

```
                input_data.append(value)
            if name == "BVP":
                heart_rate = calculate_heart_rate(pulse_data)
                extracted_hr = extract_features(heart_rate)
                for _, value in extracted_hr.items():
                    input_data.append(value)
        X = np.array(input_data)
        scaler = StandardScaler()
        X_scaled = scaler.fit_transform(X)
        lda = LDA(n_components=min(7, X.shape[1]))
        input_data = lda.fit_transform(X_scaled)

        try:
            if gsr_data == -999:
                print("Skipping prediction - GSR sensor removed!")
            else:
                blue_emotion = model.predict(input_data)[0]  # Model output
        except Exception as e:
            print(f"Model Prediction Error: {e}")
            blue_emotion = "error"

        upload_to_firebase(gsr_data, heart_rate, blue_emotion)

    messages_1 = {
        "orangeEmotion": "neutral",  # Static value
        "blueEmotion": blue_emotion,  # Model output
        "dep": True
    }

    json_message = json.dumps(messages_1)
    print(f"Sending JSON: {json_message}")
    await websocket.send(json_message)
```

```
                await asyncio.sleep(1)
            except Exception as e:
                print("Send Error:", e)

        await asyncio.gather(receive_data(websocket), send_data(websocket))

    except Exception as e:
        print(f"WebSocket Error: {e}")
        await asyncio.sleep(2)
asyncio.run(receive_and_send())
```

5.2 Emotion detection using voice

main.py

```
import streamlit as st

emotion_process = st.Page("./emotion_detection/emotion_process.py", title="Process",
default=True)
emotion_model = st.Page("./emotion_detection/emotion_model.py", title="Emotion Model")

depression_process = st.Page("./depression_detection/depression_process.py",
title="Process")
phq9 = st.Page("./depression_detection/phq9.py", title="PHQ-9")
depression_model = st.Page("./depression_detection/depression_model.py",
title="Depression Model")

pg = st.navigation(
    {
        "Emotion Detection": [emotion_process, emotion_model],
        "Depression Detection": [depression_process, phq9, depression_model]
    }
  )
pg.run()
```

emotion_model.py

```
import os
import math
import librosa
import numpy as np
import python_speech_features
import streamlit as st
import torch
from torch import nn
from torch.nn import functional as F
from pydub import AudioSegment
from st_audiorec import st_audiorec

# Ensure FFmpeg is installed
AudioSegment.converter = "ffmpeg"
# Define feature list
features = [
    'length', 'mfcc_mean', 'mfcc_sd', 'mfcc_median', 'mfcc_max', 'mfcc_min',
    'spectral_centroid_mean', 'spectral_centroid_sd', 'spectral_centroid_median',
'spectral_centroid_max',
    'spectral_centroid_min', 'spectral_rolloff_mean', 'spectral_rolloff_sd',
'spectral_rolloff_median', 'spectral_rolloff_max',
    'spectral_rolloff_min', 'logfbank_mean', 'logfbank_sd', 'logfbank_median', 'logfbank_max',
'logfbank_min',
    'spectral_subband_centroid_mean', 'spectral_subband_centroid_sd',
'spectral_subband_centroid_median',
    'spectral_subband_centroid_max', 'spectral_subband_centroid_min', 'ratio'
]

# Define Neural Network Class
class Net(nn.Module):
    def _init_(self, nb_features=27):  # Ensure 27 input features
        super(Net, self)._init_()
```

```
        self.fc1 = nn.Linear(nb_features, 200)
        self.fc2 = nn.Linear(200, 200)
        self.fc3 = nn.Linear(200, 200)
        self.fc4 = nn.Linear(200, 7)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        return self.fc4(x)

# Function to split audio
def audio_split(audio, from_sec, to_sec, folder_name, split_filename):
    t1, t2 = from_sec * 1000, to_sec * 1000
    split_audio = audio[t1:t2]
    os.makedirs(f"./audio_files/{folder_name}", exist_ok=True)
    split_audio.export(f'./audio_files/{folder_name}/{split_filename}', format="wav")

# Streamlit UI
st.title("Speech Emotion Detection")
wav_audio_data = st_audiorec()
os.makedirs("./audio_files", exist_ok=True)

if wav_audio_data is not None:
    AUDIO_PATH = "./audio_files/sample.wav"
    with open(AUDIO_PATH, "wb") as f:
        f.write(wav_audio_data)

    audio_file = AudioSegment.from_wav(AUDIO_PATH)
    # total_seconds = math.ceil(audio_file.duration_seconds)

    # for i in range(0, total_seconds, 5):
```

```
#     split_fn = f"{i}_xyz.wav"
#     audio_split(audio_file, i, i+5, 'xyz', split_fn)
# Load and preprocess audio
y, sr = librosa.load(AUDIO_PATH, sr=16000)
length = len(y)

# Feature Extraction Function
def extract_feature(f, y):
    feat = f(y=y, sr=16000)
    return np.mean(feat), np.std(feat), np.median(feat), np.max(feat), np.min(feat)

# Extract Features
mean_mfcc, sd_mfcc, median_mfcc, maxi_mfcc, mini_mfcc =
extract_feature(librosa.feature.mfcc, y)
mean_spectral_centroid, sd_spectral_centroid, median_spectral_centroid,
maxi_spectral_centroid, mini_spectral_centroid =
extract_feature(librosa.feature.spectral_centroid, y)
mean_spectral_rolloff, sd_spectral_rolloff, median_spectral_rolloff, maxi_spectral_rolloff,
mini_spectral_rolloff = extract_feature(librosa.feature.spectral_rolloff, y)

# Fix for logfbank
feature_logfbank = python_speech_features.logfbank(y, 16000)
mean_logfbank, sd_logfbank, median_logfbank, maxi_logfbank, mini_logfbank =
np.mean(feature_logfbank), np.std(feature_logfbank), np.median(feature_logfbank),
np.max(feature_logfbank), np.min(feature_logfbank)

# Spectral subband centroid features
feature_spectral_subband_centroid = python_speech_features.ssc(y, 16000)
mean_spectral_subband_centroid, sd_spectral_subband_centroid,
median_spectral_subband_centroid, maxi_spectral_subband_centroid,
mini_spectral_subband_centroid = \
    np.mean(feature_spectral_subband_centroid),
np.std(feature_spectral_subband_centroid), np.median(feature_spectral_subband_centroid),
np.max(feature_spectral_subband_centroid), np.min(feature_spectral_subband_centroid)

# Silence ratio
```

```
    threshold = 0.01
    size = 5
    coord = [i for i in range(size, len(y) - size) if np.max(abs(y[i - size:i + size])) < threshold]
    ratio = len(coord) / (len(y) - len(coord))

    # Ensure 27 features
    feature_vector = torch.tensor([
        length, mean_mfcc, sd_mfcc, median_mfcc, maxi_mfcc, mini_mfcc,
        mean_spectral_centroid, sd_spectral_centroid, median_spectral_centroid, 
maxi_spectral_centroid, mini_spectral_centroid,
        mean_spectral_rolloff, sd_spectral_rolloff, median_spectral_rolloff, 
maxi_spectral_rolloff, mini_spectral_rolloff,
        mean_logfbank, sd_logfbank, median_logfbank, maxi_logfbank, mini_logfbank,
        mean_spectral_subband_centroid, sd_spectral_subband_centroid, 
median_spectral_subband_centroid, maxi_spectral_subband_centroid, 
mini_spectral_subband_centroid,
        ratio
    ]).float().unsqueeze(0)  # Ensure shape is (1, 27)

    # Load Model and Predict
    model = Net()
    try:

model.load_state_dict(torch.load("./models/fully_connected_nn_emotion_model/cross_val.pt
", map_location=torch.device('cpu'), weights_only=True), strict=False)
    except RuntimeError as e:
        st.error(f"Error loading model: {e}")
        st.stop()

    model.eval()

    with torch.no_grad():
        output = model(feature_vector)
        predicted_emotion = torch.argmax(output, dim=1).item()
```

```
    # Emotion mapping
    emotions_english = ["anxiety", "disgust", "happy", "boredom", "anger", "sadness",
"neutral"]
    st.write("Predicted Emotion:", emotions_english[predicted_emotion])
```

5.3 Depression detection using voice

depression_model.py

```
import streamlit as st
import numpy as np
import librosa
import librosa.display
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from pydub import AudioSegment
from pyannote.audio.pipelines import SpeakerDiarization
from huggingface_hub import login
import tempfile
from st_audiorec import st_audiorec

# Set Hugging Face API Token
HUGGING_FACE_TOKEN = "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"

# Authenticate with Hugging Face
login(HUGGING_FACE_TOKEN)

# Load Pyannote Speaker Diarization pipeline
pipeline = SpeakerDiarization.from_pretrained("pyannote/speaker-diarization-3.1",
use_auth_token=HUGGING_FACE_TOKEN)

# Load trained CNN model
model = load_model("./models/cnn_depressed_model")
```

```
# Streamlit UI
st.title("Depression Detection from Speech")
st.markdown("Upload or **record** an audio file, and this app will analyze it to determine if the speaker shows signs of **depression**.")

# Option to upload or record audio
option = st.radio("Choose Input Method:", ["Upload Audio", "Record Audio"])

uploaded_file = None
temp_audio_path = None

# Handling Audio Input
if option == "Upload Audio":
    uploaded_file = st.file_uploader("Upload an audio file (WAV format)", type=["wav"])
    if uploaded_file is not None:
        with tempfile.NamedTemporaryFile(delete=False, suffix=".wav") as temp_audio:
            temp_audio.write(uploaded_file.read())  # Read file bytes
            temp_audio_path = temp_audio.name  # Store temp file path

elif option == "Record Audio":
    st.write("Click below to record your voice:")
    recorded_audio = st_audiorec()
    if recorded_audio is not None:
        with tempfile.NamedTemporaryFile(delete=False, suffix=".wav") as temp_audio:
            temp_audio.write(recorded_audio)  # Save recorded bytes
            temp_audio_path = temp_audio.name

# Ensure we have a valid audio file before proceeding
if temp_audio_path:
    st.audio(temp_audio_path, format="audio/wav", start_time=0)
```

```
# Step 1: Speaker Diarization
st.write("Performing **Speaker Diarization**...")

def diarize_audio(audio_file):
    """Performs speaker diarization and returns speaker segments"""
    diarization_result = pipeline({"uri": "audio", "audio": audio_file})

    speaker_segments = {}
    for turn, _, speaker in diarization_result.itertracks(yield_label=True):
        start, end = turn.start, turn.end
        if speaker not in speaker_segments:
            speaker_segments[speaker] = []
        speaker_segments[speaker].append((start, end))

    return speaker_segments

speaker_timestamps = diarize_audio(temp_audio_path)

# Step 2: Extract Speaker 00
st.write("Extracting **Speaker 00's Voice**...")
with tempfile.NamedTemporaryFile(delete=False, suffix=".wav") as extracted_audio_file:
    output_audio_path = extracted_audio_file.name  # Store temp file path

def segment_audio(input_audio, speaker_segments, output_file):
    """Extracts the first speaker's segments and saves them as an audio file"""
    audio = AudioSegment.from_wav(input_audio)

    if "SPEAKER_00" in speaker_segments.keys():
        speaker_audio = AudioSegment.silent(duration=0)
        for start, end in speaker_segments["SPEAKER_00"]:
            segment = audio[int(start * 1000):int(end * 1000)]  # Convert sec → ms
            speaker_audio += segment
```

```
            speaker_audio.export(output_file, format="wav")
            return output_file
        else:
            return None

    extracted_audio = segment_audio(temp_audio_path, speaker_timestamps,
output_audio_path)

    if extracted_audio:
        st.audio(output_audio_path, format="audio/wav")
        st.write("**Extracted Speech of Speaker 00**")

        # Step 3: Generate Spectrogram
        st.write("Generating **Spectrogram**...")

        with tempfile.NamedTemporaryFile(delete=False, suffix=".png") as spectrogram_file:
            spectrogram_path = spectrogram_file.name  # Store temp file path

        def create_spectrogram(audio_path, output_image):
            """Creates and saves a spectrogram from an audio file"""
            y, sr = librosa.load(audio_path, sr=16000)
            D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)

            fig = plt.figure(figsize=(5.12, 5.12), dpi=100)
            ax = fig.add_axes([0, 0, 1, 1])
            librosa.display.specshow(D, sr=sr, cmap='inferno', ax=ax)
            ax.set_xticks([])
            ax.set_yticks([])
            ax.set_frame_on(False)

            plt.savefig(output_image, dpi=100, pad_inches=0)
            plt.close(fig)
```

```
        create_spectrogram(output_audio_path, spectrogram_path)

        st.image(spectrogram_path, caption="Generated Spectrogram",
use_container_width=True)

        # Step 4: Make Prediction
        st.write("Running **Depression Prediction**...")

        def preprocess_image(image_path, target_size=(512, 512)):
            """Prepares spectrogram image for CNN model"""
            img = image.load_img(image_path, target_size=target_size)
            img_array = image.img_to_array(img)
            img_array = img_array / 255.0  # Normalize
            img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
            return img_array

        processed_img = preprocess_image(spectrogram_path)
        prediction = model.predict(processed_img)[0][0]

        # Step 5: Display Prediction Result
        if prediction > 0.5:
            st.error("**Prediction: Depressed**")
        else:
            st.success("**Prediction: Not Depressed**")

    else:
        st.warning("No valid speech segments found for Speaker 00!")

else:
    st.warning("Please upload or record an audio file!")
```

Chapter 6

Testing

6.1 Test Plan

The system was tested to evaluate its efficiency in biosignal acquisition, speech emotion detection, feature extraction, and classification accuracy. The testing methodology includes:

- Unit Testing: Individual components, including biosignal processing and voice feature extraction, were tested.
- Integration Testing: The system's ability to process both biosignals and speech data simultaneously was evaluated.
- Performance Testing: Model accuracy was measured based on real-world

6.2 Test Cases

Table 3: Test Case for Emotion Classification using Biosignals

Test ID	Test Case Description	Expected Output	Result
T1	Biosignal data acquisition PPG and GSR	Signals are recorded and displayed	Pass
T2	Feature extraction from biosignals	Extracted values for classification	Pass
T3	Random Forest classification using biosignals	Predicted emotion is displayed	Pass

Table 4: Test Case for Emotion Classification using Voice

Test ID	Test Case Description	Expected Output	Result
T4	Handling noise in speech input	Background noise does not interfere with classification	Pass

T5	Audio feature extraction (MFCC)	Extracted MFCC features displayed	Pass
T6	Neural-Network-based voice emotion classification	Correct emotion label assigned	Pass

6.3 Test Reports

Model Accuracy Evaluation

- **Biosignal-Based Emotion Detection:**

Accuracy: 43.75%

- **Voice-Based Emotion Detection:**

Accuracy: 66.17%

- **Depression Detection Model Performance:**

Accuracy: 83.33%

6.4 Overall Test Summary

Types of Tests Conducted:

Unit testing, integration testing, and model accuracy evaluation.

Findings:

- Emotion Detection using Biosignal, Emotion Detecton using Voice and Depression Detection using Voice achieved an overall accuracy of 43.75%, 66.17%, and 83.33% respectively.
- The system detected strong emotions more accurately than subtle ones.

Improvements:

- Improve dataset diversity to handle more emotional variations.
- Enhance noise reduction techniques in voice-based models.

Chapter 7

Conclusion

The project presents a novel approach to mental health assessment by integrating biosignal data and voice analysis with machine learning. It successfully provides an objective, real-time system to detect emotional states and signs of depression. By combining wearable technology, AI-based classification models, and a web-based interface, the system enhances the accuracy and reliability of mental health monitoring. This solution bridges the gap between technology and psychology, offering a promising tool for both professionals and individuals seeking better emotional well-being tracking.

7.1 Advantages

- Objective & Reliable: Eliminates subjectivity by using physiological and voice-based data.
- Non-Invasive: Utilizes wearable sensors without requiring invasive procedures.
- Real-Time Monitoring: Provides instant feedback and continuous tracking of emotional states.
- AI-Powered Accuracy: Machine learning models improve classification and detection accuracy.
- User-Friendly Interface: A web-based dashboard allows easy access and visualization of mental health trends.
- Early Depression Detection: Helps in identifying potential signs of mental health issues before they worsen.

7.2 Limitations

- Data Variability: The accuracy may vary based on individual differences in physiological responses.
- Environmental Noise: External factors can affect voice recordings and biosignal measurements.
- Hardware Dependency: Requires specific wearable sensors and devices for data collection.
- Limited Dataset: Model performance depends on the quality and diversity of training data.

- Real-Time Constraints: Processing large amounts of data in real-time may lead to latency issues.

7.3 Future Scope

- Enhanced AI Models: Use deep learning and transfer learning to improve accuracy.
- Expanded Datasets: Incorporate larger and more diverse datasets for better generalization.
- Integration with Smart Wearables: Connect with commercially available smartwatches and fitness bands.
- Mobile App Development: Develop a smartphone application for easier access and user engagement.
- Personalized Recommendations: Implement AI-driven mental health insights and coping strategies.
- Multi-Language Support: Expand voice analysis to support multiple languages and dialects.

References

1. Gratch J, Artstein R, Lucas GM, Stratou G, Scherer S, Nazarian A, Wood R, Boberg J, DeVault D, Marsella S, Traum DR. The Distress Analysis Interview Corpus of human and computer interviews. InLREC 2014 May (pp. 3123-3128)
2. DeVault, D., Artstein, R., Benn, G., Dey, T., Fast, E., Gainer, A., Georgila, K., Gratch, J., Hartholt, A., Lhommet, M., Lucas, G., Marsella, S., Morbini, F., Nazarian, A., Scherer, S., Stratou, G., Suri, A., Traum, D., Wood, R., Xu, Y., Rizzo, A., and Morency, L.-P. (2014). "SimSensei kiosk: A virtual human interviewer for healthcare decision support". In Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'14), Paris
3. Research on IoT-Based Emotion Recognition Devices, ScienceDirect. Available: https://www.sciencedirect.com/science/article/pii/S1566253523003354.
4. Task Force of the European Society of Cardiology and the North American Society of Pacing and Electrophysiology, "Heart Rate Variability: Standards of Measurement, Physiological Interpretation, and Clinical Use," European Heart Journal, vol. 17, no. 3, pp. 354–381, 1996. Available: https://www.escardio.org/static-file/Escardio/Guidelines/Scientific-Statements/guidelines-Heart-Rate-Variability-FT-1996.pdf.
5. G. Sannino and I. De Falco, "A Comprehensive Review on Wearable Health Monitoring Systems," Open Biomedical Engineering Journal, vol. 15, pp. 213–230, 2021. Available: https://openbiomedicalengineeringjournal.com/VOLUME/15/PAGE/213/FULLTEXT/.
6. Emotion Recognition Using Multimodal Data, ScienceDirect. Available: https://www.sciencedirect.com/science/article/pii/S0167639324000785#b39.
7. J. Gratch, R. Artstein, G. M. Lucas, G. Stratou, S. Scherer, A. Nazarian, R. Wood, J. Boberg, D. DeVault, S. Marsella, and D. R. Traum, "The Distress Analysis Interview Corpus of Human and Computer Interviews," in Proc. Ninth Int. Conf. Language Resources and Evaluation (LREC), Reykjavik, Iceland, 2014, pp. 3123–3128.
8. MIT Media Lab, "Affective Computing Group Technical Report," Available: https://vismod.media.mit.edu/pub/tech-reports/TR-510.pdf.