NURSING BOT

A PROJECT REPORT
191ROC513L – INNOVATION LABORATORY

Submitted by

AATHISHKUMAR S (310622125002)

ANANYA KANNAN (310622125005)

NAVEEN K.M (310622125032)

BACHELOR OF ENGINEERING
in
ROBOTICS AND AUTOMATION

EASWARI ENGINEERING COLLEGE, CHENNAI
(AutonomousInstitution)

Affiliated *to*

ANNA UNIVERSITY: CHENNAI - 600025

NOV 2024

EASWARI ENGINEERING COLLEGE, CHENNAI

(Autonomous Institution)

AFFILIATED TO ANNA UNIVERSITY, CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"NURSING BOT"** is the bonafide work of

"AATHISH KUMAR (310622125002), ANANYA KANNAN (310622125005), and NAVEEN K.M (310622125032)" carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. V. ELANGO, M.E, Ph.D	**Dr. V. ELANGO, M.E, Ph.D**
HEAD OF DEPARTMENT	**SUPERVISOR**
Professor and Head	Professor and Head
Department of Robotics and Automation	Department of Robotics and Automation
Easwari Engineering College	Easwari Engineering College
Ramapuram, Chennai 600089	Ramapuram, Chennai 600089

Submitted for the Semester Examination held on ………………..

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We sincerely thank Dr R. Sivakumar, our Beloved Chairman, and **Mr S. Niranjan**, our Beloved **Co-Chairman** for their Blessings.

We are grateful to **Dr. P. Deiva Sundari, Principal**, for providing an excellent environment in which to carry out our project work successfully. We are deeply indebted to **Dr. V. Elango**, **Professor and Head** of Robotics and Automation, for his continuous support at every stage of our project work.

We convey our sincere thanks to **Dr. V. Elango**, **Professor and Head and our supervisor, for his technical support and guidance in completing our project. We would like to thank Dr. S. Santhosh, Class Coordinator and Assistant Professor, and Dr. S. Rajkamal**, **Assistant Professor**, for project coordination and support.

Our sincere thanks to all the faculty and staff members of our department for their valuable guidance at various stages of our project development.

UNITED NATIONS 17 SUSTAINABLE DEVELOPMENT GOALS

1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL-BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE AND JUSTICE STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
THE GLOBAL GOALS
For Sustainable Development

Our Project Work towards 17 SDGs is mapped as follows:

Target 9.4: Upgrade all industries and infrastructures for sustainability
"By 2030, upgrade infrastructure and retrofit industries to make them sustainable, with increased resource-use efficiency and greater adoption of clean and environmentally sound technologies and industrial processes, with all countries taking action in accordance with their respective capabilities."

Target 12.2: Sustainable management and use of natural resources
"By 2030, achieve the sustainable management and efficient use of natural resources."

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

This project aims to enhance the care of paralyzed patients by using AI-powered facial recognition and emotion detection by monitoring their emotional state in real time. Equipped with a camera system, the device continuously analyses facial expressions to identify emotions such as distress, sadness, or anger. When significant emotional changes are detected, caregivers are instantly alerted through real-time notifications and alerts on the platform Telegram. This system ensures timely intervention, improving response times and overall patient well-being by providing crucial emotional insights, particularly for non-verbal patients.

CHAPTER 1:

INTRODUCTION

The paralysed patients' emotion and facial recognition system is designed to monitor and communicate their emotional states using facial recognition and emotion detection technologies. A camera continuously tracks the patient's facial expressions to identify emotions like happiness, sadness, or distress. When emotional changes are detected, real-time alerts are sent to caregivers via message notifications on platforms like Telegram, ensuring timely intervention.

The system analyses facial data and transmits emotional insights wirelessly, allowing caregivers to monitor the patient's emotional well-being. This technology improves patient care by providing crucial emotional feedback and ensuring that even non-verbal patients receive the necessary attention promptly.

The device employs advanced machine learning algorithms to interpret subtle facial expressions accurately, ensuring precise emotion detection. By delivering real-time updates on the patient's emotional state, it enhances the ability of caregivers and healthcare professionals to respond quickly and effectively. This system offers a reliable way to monitor the emotional well-being of paralyzed patients, ultimately improving their quality of life and care.

CHAPTER 2

LITERATURE REVIEW

Dr. Aditi Sharma (2020):

Dr Aditi Sharma's study developed a deep learning framework for real-time emotional assessment in patients. Our project enhances this with the Human Library, enabling efficient emotion detection through a Node.js server, React-based client, and Telegram bot integration.

Prof. Ravi Kumar (2021):

Prof. Ravi Kumar created an IoT-based emotion recognition system with caregiver alerts via a mobile app. We build on this by using the Human Library and Node.js to offer a browser-based solution with Telegram bot notifications, removing the need for dedicated apps.

Dr. Priya Verma (2022):

Dr. Priya Verma proposed a device for real-time emotion monitoring in paralyzed patients. Our system enhances accessibility by processing facial data through Node.js servers and React clients, delivering alerts via Telegram bots for broader caregiver support.

CHAPTER 3

OVERVIEW OF THE PROPOSED SYSTEM

This system uses facial recognition, AI algorithms, Node.js, React, cloud integration, wireless communication, and Telegram bots to detect emotions, upload data, and alert caregivers in real time for improved patient care.

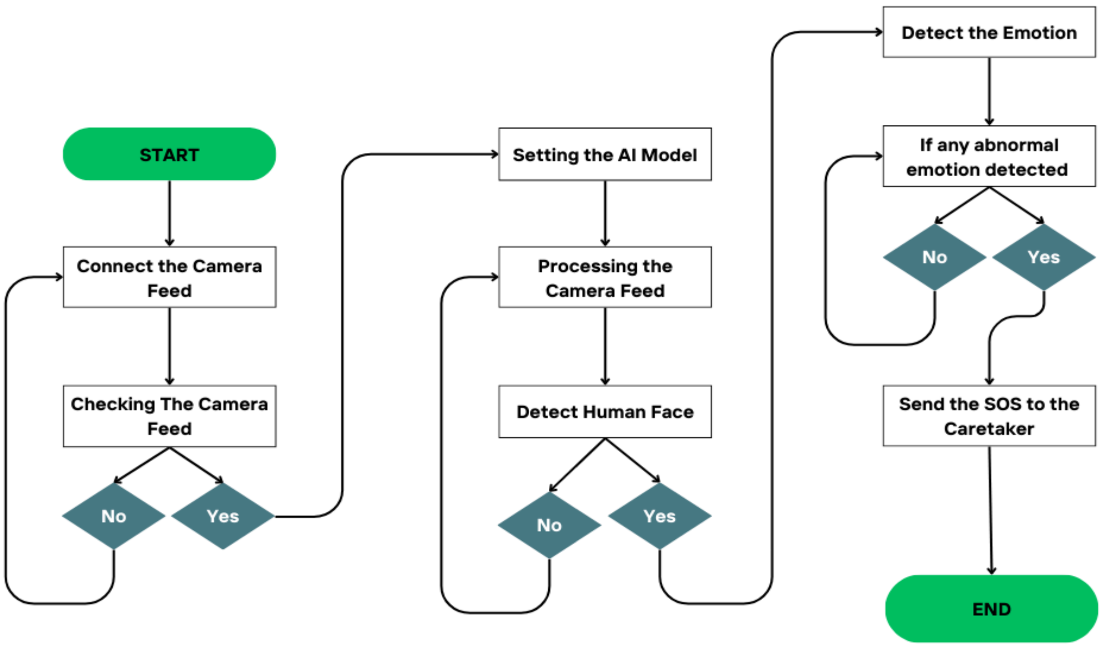


Fig 3.1 Process Flow of the Proposed Emotion Detection System

The proposed system collects real-time emotional data from a facial recognition camera and processes it using emotion detection algorithms. This information is uploaded to a cloud-based platform via a wireless communication module, allowing caregivers to monitor the emotional states of paralyzed patients remotely. Timely alerts are generated for significant emotional changes, facilitating immediate intervention and improving patient care.

CHAPTER 4

PARTS DESCRIPTION

4.1 RASPBERRY PI ZERO 2 W

The Raspberry Pi Zero 2 W, as shown in Figure 4.1, is a compact, low-cost single-board computer designed for many IoT and embedded applications.

Its key specifications are:

- Processor: Quad-core ARM Cortex-A53 running at 1 GHz
- RAM: 512 MB LPDDR2 SDRAM
- Connectivity: Integrated Wi-Fi (802.11 b/g/n) and Bluetooth 5.0
- Size: 65 mm x 30 mm x 5 mm
- Power: Operates at 5V via a micro-USB power supply
- Ports: One mini-HDMI port, one USB On-The-Go port, and a 40-pin GPIO header

The Raspberry Pi Zero 2 W is perfect for your project due to its compact size and efficient processing power, making it ideal for real-time data processing and communication in your mobile robot.

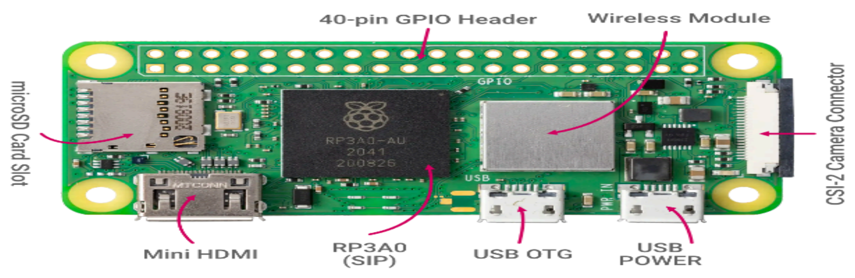


Fig 4.1: Image of Raspberry Pi Zero 2W

4.2 PI CAMERA

The Pi Camera V2, as shown in Figure 4.2, is an official camera module for the Raspberry Pi, designed to capture high-quality images and videos. Its compact design and powerful features make it ideal for various applications, including real-time facial data capture and emotion detection in healthcare projects.

Its specifications include:

- Resolution: 8 MP (3280 x 2464 pixels) for still images
- Video Recording: Supports 1080p30, 720p60, and VGA90 video recording
- Interface: Connects to the Raspberry Pi via a dedicated camera interface (CSI)
- Size: Compact design, ideal for integration into projects

The Pi Camera V2 plays a crucial role in your project by enabling the real-time capturing of facial expressions, which is essential for emotion detection in paralysed patients.

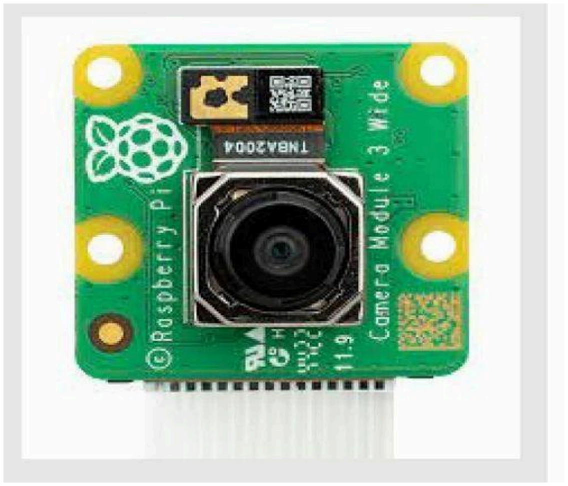


Fig 4.2 Image of Pi Camera

CHAPTER 5

SOFTWARE DESIGN

This chapter describes a React application that uses webcam streaming and emotion detection through the @vladmandic/human library, sending Telegram alerts for persistent emotional states, and managing user interactions with state and logs.

5.1 ALGORITHM

Initialize State and References:

- Define states for streaming, logs, dominant emotion, webcam stream, and human emotion detection model.
- Create references for the video element and manage stream data.

Load Human Emotion Detection Model:

- In the useEffect hook, load the human library (which handles emotion detection) and set it up with the required settings for face emotion detection.
- Set the model to load using the webGL backend.

Start Webcam Streaming:

- On clicking the "Start Stream" button:
  - Use navigator.mediaDevices.getUserMedia() to access the user's webcam and start the video stream.
  - Set the video element source to the webcam stream.
  - Log that the webcam stream has started.

Stop Webcam Streaming:

- On clicking the "Stop Stream" button:
    - Stop all active webcam tracks to end the stream.
    - Log that the webcam stream has been stopped.

Emotion Detection:

- Set Interval: Run emotion detection every 200ms.
- Detect Emotion: Use human. detect() to analyze the webcam feed.
- Process Emotion: If the confidence score > 0.5, track the dominant emotion.
- Track Duration: Keep track of how long the emotion persists.
- Trigger Notification: If the emotion lasts for 2+ seconds, send a Telegram message about the emotion.
- Reset: After sending the message, reset emotion tracking.

Send Telegram Message:

- If a persistent emotion (e.g., "sad", or "angry") is detected, send a message to the predefined Telegram bot using the Telegram API.
- Include details about the emotion (e.g., type and confidence score).

Display Detected Emotion:

- If an emotion is detected, display the emotion and its confidence score on the dashboard.
- If no emotion is detected, show a message indicating no strong emotion detected.

5.2 DEPENDENCIES:

@testing-library/jest-dom (^5.17.0): For DOM testing with Jest; optimize by keeping in dev mode.

@testing-library/react (^13.4.0): Simulates user interactions in React testing; optimize for dev mode.

@testing-library/user-event (^13.5.0): Simulates user events for UI testing; essential for behavior-driven tests.

@vladmandic/human (^3.3.4): AI for emotion and face detection; optimize model loading.

Axios (^1.7.7): HTTP client for API requests; handles network errors and retries.

CORS (^2.8.5): Enables cross-origin requests; configure for trusted origins.

Express (^4.21.1): Backend framework for APIs; optimise for production (secure headers, compression).

face-api.js (^0.22.2): Alternative face detection library; use if needed alongside @vladmandic/human.

fluent-FFmpeg (^2.1.3): Handles multimedia streams; include only if processing video on the backend.

React (^18.3.1) & React-DOM (^18.3.1): Core for building React apps; ensure consistent versioning.

react-scripts (5.0.1): Simplifies React app setup; consider ejecting for custom configs.

Socket.IO (^4.8.1): Real-time communication; include only if needed for live updates.

web-vitals (^2.1.4): Tracks app performance metrics; use in production/dev for monitoring.

5.3 PROGRAM CODE:

```
import React, { useState, useEffect, useRef } from "react";
import Human from "@vladmandic/human";
import "./App.css";

const App = () => {
  const [isStreaming, setIsStreaming] = useState(false);
  const [logs, setLogs] = useState([]);
  const [dominantEmotion, setDominantEmotion] = useState(null);
  const videoRef = useRef(null);
  const [stream, setStream] = useState(null);
  const [human, setHuman] = useState(null);
  const [isModelLoaded, setIsModelLoaded] = useState(false);

  const botToken = "7700637561:AAFLI5APdlQsK5wXFmWq3gEotnVeG-H06jI";
  const userId = "7082124011";

  const sendMessage = async (message) => {
    try {
      const response = await fetch(
        `https://api.telegram.org/bot${botToken}/sendMessage`,
        {
          method: "POST",
          headers: { "Content-Type": "application/json" },
          body: JSON.stringify({
            chat_id: userId,
 text: message,
          }),
        }
      );

      if (!response.ok) {
        throw new Error(`Telegram API error: ${response.statusText}`);
      }
      console.log("Message sent successfully!");
    } catch (error) {
      console.error("Error sending Telegram message:", error);
    }
  };

  useEffect(() => {
    const loadHuman = async () => {
      const humanInstance = new Human({
        backend: "webgl",
        modelBasePath: "https://vladmandic.github.io/human/models/",
        face: {
          emotion: { enabled: true, minConfidence: 0.5 },
        },
        body: { enabled: false },
        hand: { enabled: false },
      });
      await humanInstance.load();
      setHuman(humanInstance);
```

```
setIsModelLoaded(true);
  };

  loadHuman();
 }, []);

 useEffect(() => {
  if (videoRef.current && isModelLoaded) {
   videoRef.current.srcObject = stream;
   videoRef.current.addEventListener("loadeddata", () => {
    console.log("Video stream loaded and ready for emotion detection");
   });
  }
 }, [isModelLoaded, stream]);

 useEffect(() => {
  let emotionState = { emotion: null, duration: 0 };

  const detectEmotions = async () => {
   try {
    if (isModelLoaded && videoRef.current && videoRef.current.readyState ===
{
     const result = await human.detect(videoRef.current);

     if (result.face && result.face.length > 0 && result.face[0].emotion) {
      const emotions = result.face[0].emotion;
```

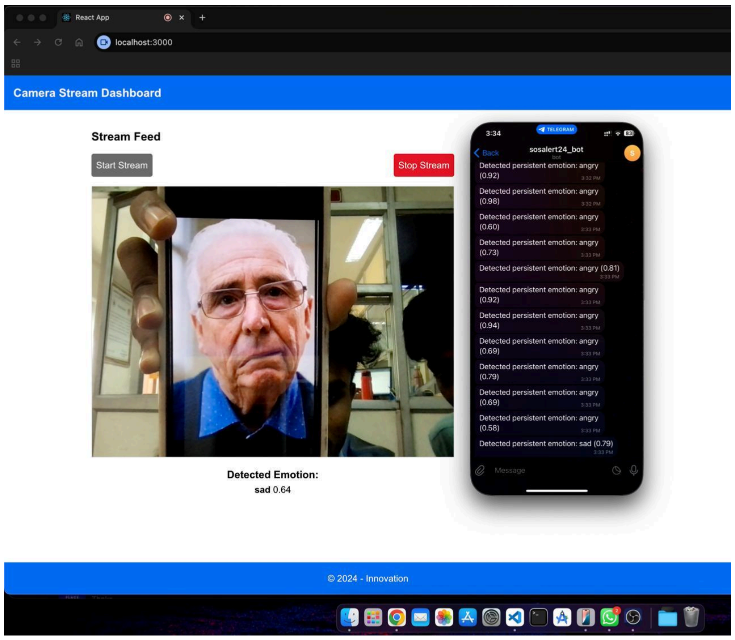
```
if (emotions.length > 0) {
          const highestEmotion = emotions.reduce((prev, current) =>
            prev.score > current.score ? prev : current
          );

          if (highestEmotion.score > 0.5) {
            if (emotionState.emotion === highestEmotion.emotion) {
              emotionState.duration += 200;
            } else {
              emotionState = { emotion: highestEmotion.emotion, duration: 200 };
            }

            if (
              emotionState.duration >= 2000 &&
              ["sad", "angry", "fear", "surprise", "abnormal"].includes(
                emotionState.emotion.toLowerCase()
              )
            ) {
              const message = `Detected persistent emotion: ${emotionState.emotion}
(${highestEmotion.score.toFixed(
                2
              )})`;
              sendMessage(message);

              emotionState = { emotion: null, duration: 0 };
            }
setDominantEmotion(highestEmotion);
              } else {
                emotionState = { emotion: null, duration: 0 };
                setDominantEmotion(null);
              }
            } else {
              setDominantEmotion(null);
            }
          }
        }
      } catch (error) {
        console.error("Emotion detection error:", error);
      }
    };
    const interval = setInterval(detectEmotions, 200);
    return () => clearInterval(interval);
  }, [isModelLoaded, human]);

  const handleStartStreaming = async () => {
    try {
      const mediaStream = await navigator.mediaDevices.getUserMedia({ video: true
});
      setStream(mediaStream);
      setIsStreaming(true);
      setLogs((prevLogs) => [...prevLogs, "Started webcam stream"]);
    } catch (error) {
      setLogs((prevLogs) => [
 ...prevLogs,
   `Error starting webcam stream: ${error.message}`,
  ]);
 }
};
const handleStopStreaming = () => {
 if (stream) {
  stream.getTracks().forEach((track) => track.stop());
  setStream(null);
 }
 setIsStreaming(false);
 setLogs((prevLogs) => [...prevLogs, "Stopped webcam stream"]);
};
return (
 <div className="App">
  <header className="bg-blue-500 p-4 text-white">
   <h1 className="text-xl font-bold">Camera Stream Dashboard</h1>
  </header>

  <div className="container mx-auto p-4 flex">
   <div className="w-2/3 p-4">
    <h2 className="text-xl font-semibold mb-4">Stream Feed</h2>
    <div className="flex justify-between mb-4">
     <button
      onClick={handleStartStreaming}
      className="bg-green-500 text-white p-2 rounded"
```

```
disabled={isStreaming}
  >
            Start Stream
          </button>
          <button
            onClick={handleStopStreaming}
            className="bg-red-500 text-white p-2 rounded"
            disabled={!isStreaming}
          >
            Stop Stream
          </button>
        </div>
        <div>
          <video
            ref={videoRef}
            autoPlay
            playsInline
            muted
            className="border w-full"
            style={{ display: isStreaming ? "block" : "none" }}
          />
        </div>
        {dominantEmotion ? (
          <div className="mt-4 text-center">
            <h2 className="text-lg font-semibold">Detected Emotion:</h2>
            <p>
              <strong>{dominantEmotion.emotion}</strong>{" "}
{dominantEmotion.score.toFixed(2)}
  </p>
          </div>
        ) : (
          <div className="mt-4 text-center">
            <h2 className="text-lg font-semibold">No strong emotion detected
yet.</h2>
          </div>
        )}
      </div>


      <div className="w-1/3 p-4">
        <h2 className="text-xl font-semibold mb-4">Logs</h2>
        <div
          className="h-64 overflow-y-auto bg-gray-200 p-4 rounded border
border-gray-300"
          style={{ backgroundColor: "#f0f0f0" }}
        >
          {logs.map((log, index) => (
            <p key={index}>{log}</p>
          ))}
        </div>
      </div>
    </div>
```

```
<footer className="bg-blue-500 p-4 text-white text-center">
        <p>&copy; 2024 - Innovation</p>
      </footer>
    </div>
  );
};

export default App;
```

CHAPTER 6

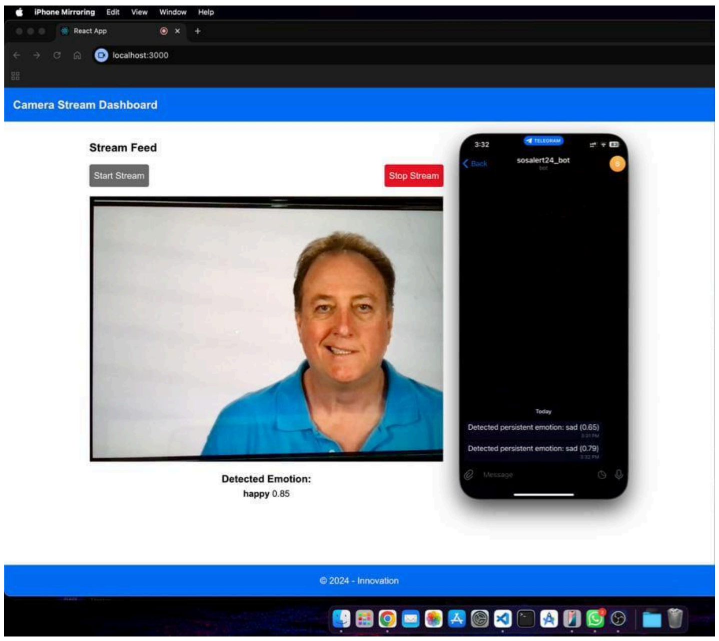
OUTCOME



Detected Emotion:
sad 0.64

Detected persistent emotion: angry (0.92) 3:32 PM
Detected persistent emotion: angry (0.98) 3:32 PM
Detected persistent emotion: angry (0.60) 3:33 PM
Detected persistent emotion: angry (0.73) 3:33 PM
Detected persistent emotion: angry (0.81) 3:33 PM
Detected persistent emotion: angry (0.92) 3:33 PM
Detected persistent emotion: angry (0.94) 3:33 PM
Detected persistent emotion: angry (0.69) 3:33 PM
Detected persistent emotion: angry (0.79) 3:33 PM
Detected persistent emotion: angry (0.69) 3:33 PM
Detected persistent emotion: angry (0.58) 3:33 PM
Detected persistent emotion: sad (0.79) 3:33 PM

Fig 6.1 Image of The Bot Detecting Sad emotion



Detected Emotion:
happy 0.85

Detected persistent emotion: sad (0.65) 3:31 PM
Detected persistent emotion: sad (0.79) 3:32 PM

Fig 6.2 Image of The Bot Detecting Happy emotion

CHAPTER 7

COST ESTIMATION

S.NO	PRODUCT	QUANTITY	AMOUNT
1	RASPBERRY PI ZERO 2W	1	1648/-
2	RASPBERRY PI ZERO CAMERA CABLE	1	79/-
3	RASPBERRY PI ZERO CAMERA MODULE	1	1770/-

Total: 4298.00

CHAPTER 8

CONCLUSION

In conclusion, this project effectively enhances the care of paralysed patients by leveraging AI-powered facial recognition and emotion detection to monitor their emotional states in real time. The system continuously analyses facial expressions to detect emotions such as distress, sadness, or anger, providing valuable insights into the patient's well-being. Upon detecting significant emotional changes, caregivers are notified via real-time alerts, including message notifications on the platform Telegram. This timely intervention ensures a prompt response, improving overall care and safety of non-verbal patients and significantly contributing to their emotional and physical well-being.

REFERENCE

[1] Liu, Y. J., & Chang, L. S. (2020). "An emotion detection framework for improving the well-being of patients using IoT and machine learning." IEEE Transactions on Biomedical Engineering, 67(11), 3127-3134.

[2] Mandic, G. Vlad. (2020). "Human: AI-powered facial recognition and emotion detection library." Journal of AI and Computer Vision Technologies.

[3] Reddy, C. R. K. M., & Gupta, N. K. (2019). "Integrating facial emotion recognition for healthcare with IoT: Real-time applications and challenges." International Journal of Computer Science and Healthcare, 8(3), 142-153.

[4] Romero, M. J., Smith, L. M., Williams, A. T., & Jones, S. R. (2021). "Emotion detection systems for healthcare applications: A comprehensive survey." IEEE Access, 9, 12490-12503.

[5] Shah, P. L., Sharma, R. R., & Gupta, S. K. (2020). "Real-time emotion detection using facial expression recognition: A study on mobile devices." Proceedings of the International Conference on Artificial Intelligence and Machine Learning.

[6] Thomas, J. D., Cruz, R. A., & Patel, S. K. (2021). "Real-time emotion detection in healthcare applications: A review of methods and applications." Healthcare Technologies Letters, 8(2), 41-48.