

SMART SENIOR CITIZEN FACIAL EMOTION ANALYSIS WITH ROBOTIC ASSISTANT

**Submitted in partial fulfilment of the requirements for the award of
Master of Computer Science**

By

RENUGADEVI.K(Reg. No. 42734017)



**DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF SCIENCE AND HUMANITIES
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A++” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

**DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF SCIENCE & HUMANITIES**

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bona fide work of RENUGA DEVI.K (42734017) who carried out the project entitled "SMART SENIOR CITIZEN FACIAL EMOTION ANALYSIS WITH ROBOTIC ASSISTANT" under my supervision from January 2024 to April 2024.

S. Rajeshwary
Internal Guide

Ms. RAJESHWARY S M.Sc., Assistant Professor

DEAN AND HEAD OF THE DEPARTMENT

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.,

Submitted for Viva voce Examination held on 20-04-2024

Internal Examiner

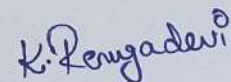
External Examiner

DECLARATION

I am **RENUGADEVI.K (42734018)** hereby declare that the Project Report entitled " **SMART SENIOR CITIZENFACIAL EMOTION ANALYSIS WITH ROBOTIC ASSISTANT**" done by me under the guidance of **Ms. RAJESHWARY.S M.Sc., Assistant Professor**, Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600119 is submitted in partial fulfilment of the requirements for the award of Master of Science degree in Computer Science.

DATE: 20-04-2024

PLACE: CHENNAI


RENUGADEVI K

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully.

I am grateful to them. I convey my thanks to **DR. REKHA CHAKRAVARTHI M.E., Ph.D., Dean, School of Arts and Science** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.Rajeshwary S M.Sc., Assistant Professor**, Department of computer science for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science who were helpful in many ways For the completion of the project.

RENUGADEVI K

ABSTRACT

The problems faced by elder people who are living independently, are considered to be one of the most important motivations of the activity recognition research. Mostly the people will face loneliness where no one will be with them to take care. The advances in sensing technologies allow collecting several types of data and communicate it wirelessly. However, most existing activity recognition systems require pre-calculated pattern recognition models. This paper lays the theoretical foundations of a real-time methodology for activity and emotional recognition using a camera based accessing pattern where the emotion of sad, cry, depressed, etc will be shown here. The monitoring person will be intimated with an emotion analysis using a mail based accessing platform system. Thus an efficient analysis serves as a foundation for a proposed conceptual framework toward an autonomous monitoring system for geriatric people who could minimize the need for explicit user input or interaction while still monitoring the geriatric person(s) well-being. The system will be a complete best framework where the virtual robotic chats will be applied.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	V
1.	INTRODUCTION	1
1.1	PROJECT OBJECTIVE	1
2.	LITERATURE SURVEY	3
3.	SCOPE OF THE PRESENT INVESTIGATION	4
3.1	SCOPE AND OBJECTIVE	4
3.2	SYSTEM REQUIREMENTS	4
3.2.1	HARDWARE REQUIREMENTS	4
3.2.2	SOFTWARE REQUIREMENTS	4
4.	SYSTEM DESIGN	5
4.1	UML DIAGRAMS	5
4.1.1	USE CASE DIAGRAM	5
4.1.2	SEQUENCE DIAGRAM	7
4.1.3	ACTIVITY DIAGRAM	8
4.1.4	CLASS DIAGRAM	9
5.	SYSTEM ARCHITECTURE	11
6.	METHODOLOGIES	12
6.1	EXISTING SYSTEM	12

6.2	PROPOSED SYSTEM	13
6.3	MODULES	13
6.3.1	MODULES DESCRIPTION	14
7.	CODING AND SCREENSHOT	16
7.1	SAMPLE CODING	16
7.2	SAMPLE SCREENSHOT	40
8.	CONCLUSION AND FUTURE WORK	47
8.1	CONCLUSION	47
8.2	FUTURE ENHANCEMENT	47
	REFERENCES	

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
4.1	USE CASE DIAGRAM	6
4.2	SEQUENCE DIAGRAM	7
4.3	ACTIVITY DIAGRAM	8
4.4	CLASS DIAGRAM	10
5.1	SYSTEM ARCHITECTURE	11

CHAPTER 1

INTRODUCTION

Mostly in the old age home or normal home old people called as senior citizen are kept lonely. They feel very bad on being lonely in the home alone and sometimes they feel very depressed due to this problem. Thus a solution needs to be generated on overcoming this software issue with elimination over this loneliness of the people who are suffering from a big problem.

The generated solution Need to be more effectual for the old people where they need to feel happy with the implementation created. With the development of technology, ML technology and people's life have been greatly improved. Many countries are faced with the problem of "population aging", in which more elderly means more medical, social and nursing services. If only the traditional way including build more nursing homes, equipment and employ more nurses were adopted, it would be hard to solve this problem

1.1 PROJECT OBJECTIVE

Many older adults will experience one or more chronic conditions, disabilities, or diseases that will need management to maintain independent living. The Journal of the American Medical Association discusses the preparation needed for better health and health care for an aging population. The implemented artificial intelligence (AI) system that integrates multiple intelligent components to actively assist an elderly person at home. Specifically, we describe how constraint-based scheduling technology is used to actively monitor a pattern of activities executed by the person and how detected temporal constraint violations are used to trigger meaningful and contextualized proactive interactions. This project also presents a psychological evaluation of the system focusing on elderly people's attitudes, in which system acceptability, perceived utility, interaction modality, and emotional response are considered.

To sum up, the proposed system can see that the elderly care technology has become a relatively mature field. It is true that today's elderly assistant equipment can meet the needs of the elderly through different types of auxiliary equipment, however, these are only assistant machines, not a robot. They need the elderly to use the remote control to manually manipulate or shout a word of speech

CHAPTER 2

LITERATURE SURVEY

Year and Publication	Name Of Authors	Title	Approach Used	Merits	Critic
IEEE and 2020	Niti Naik; Mayuri A. Mehta	An Improved Method to Recognize Hand-over-Face Gesture based Facial Emotion using Convolutional Neural Network	Hand-over-Face Gesture based Facial Emotion Recognition Method (HFG_FERM)	Face gesture emotion detection	Accuracy is just 82%
IEEE and 2021	Jing Yu; Jingjing Shi; <u>Yunwen</u> Chen	Leveraging Large Face Recognition Data for Emotion Classification	Spatial feature analysis	The feature based extraction is done	The identification on small face is not done
IEEE and 2020	Viha Upadhyay; Devangi Kotak	A Review on Different Facial Feature Extraction Methods for Face Emotions Recognition System	A research based four paper analysis is done	The best algorithm for emotion detection is done	Normal simulation works are only used
IEEE and 2019	Yoshiyuki Takahashi; Motoki Takagi; Kaoru Inoue	Development of a Communication Robot System for Monitoring Older People	Convolution Neural Network	The robot communication system is done	Efficient loneliness prediction is not done
IEEE and 2022	Kohichi Ogawa	A Smart LED Cane to Monitor Older People Living Alone for Considering Privacy	LED display analysis	The system helps to analyze the old people	Costly implementation

CHAPTER 3

SCOPE OF THE PRESENT INVESTIGATION

3.1 SCOPE AND OBJECTIVE

The main objective of the system is to generate a friendly bot to old people/senior citizen. The system analyzed the people emotion and classify the existence of the personal state. Then on a lower emotion the bot will make a audio response system.

3.2 SYSTEM REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS

- **Processor** : Dual core processor 2.6.0 GHz
- **RAM** : 4GB
- **Hard disk** : 160 GB
- **Compact Disk** : 650 MB
- **Keyboard** : Standard keyboard
- **Monitor** : 15 inch color monitor
- **External Hardware** : Camera

3.2.2 SOFTWARE REQUIREMENTS

- **Front End** : Python
- **Back End** : MySQL
- **IDE** : PyCharm Community Edition
- **Interface** : Windows Application (T Kinter)
- **Platform** : Windows 10 Pro

CHAPTER 4

SYSTEM DESIGN

4.1 UML DIAGRAMS

A UML (Unified Modeling Language) diagram is a graphical representation of a system or process using standardized symbols and notation. UML diagrams are widely used in software engineering and other fields to visually depict various aspects of a system, such as its structure, behavior, interactions, and architecture.

There are several types of UML diagrams, each serving a different purpose and focusing on different aspects of the system being modeled. Some common types of UML diagrams include:

Unified Modeling Language (UML) diagram for the proposed automated college timetable scheduling system. Here's an example of a high-level UML class diagram for the system:

4.1.1 Use Case Diagram

A Use Case Diagram is a type of UML (Unified Modeling Language) diagram that illustrates the interactions between actors (users or external systems) and a system under consideration. It provides a high-level view of the functionalities or features of a system from the perspective of its users. Use case diagrams are often used during the requirements analysis phase of software development to capture and communicate the system's behavior and requirements.

This diagram provides a high-level overview of the interactions between users, administrators, and the animal detection system, as well as the main functionalities supported by the system.

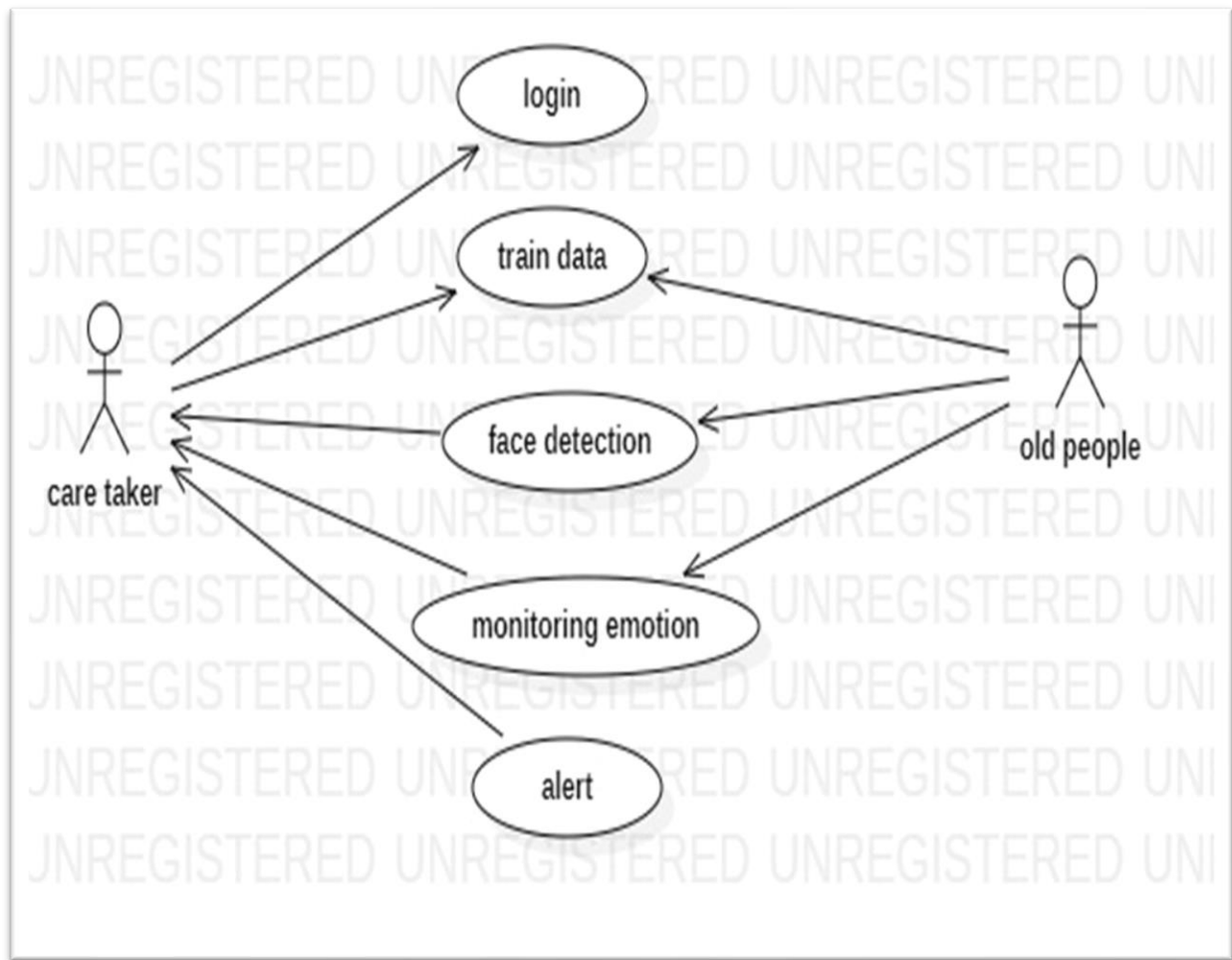


Fig 4.1 Use case diagram

4.1.2 Sequence Diagram

A Sequence Diagram is a type of UML (Unified Modeling Language) diagram that depicts the interactions between objects or components within a system over time. It illustrates the sequence of messages exchanged between these objects to accomplish a specific behavior or scenario. Sequence diagrams are particularly useful for visualizing the flow of control and collaboration among objects during the execution of a use case or scenario.

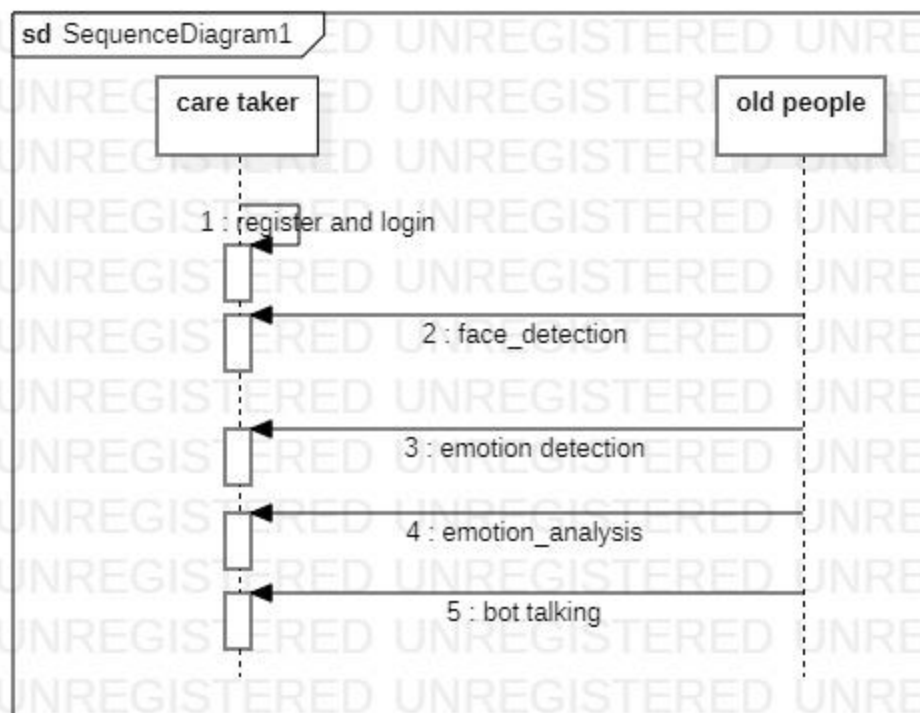


Fig 4.2 Sequence Diagram

4.2.3 Activity Diagram

An Activity Diagram is a type of UML (Unified Modeling Language) diagram that illustrates the flow of activities or actions within a system or a specific business process. It visually represents the sequence of actions, decisions, and transitions from one activity to another, allowing stakeholders to understand the workflow or behavior of a system or process.

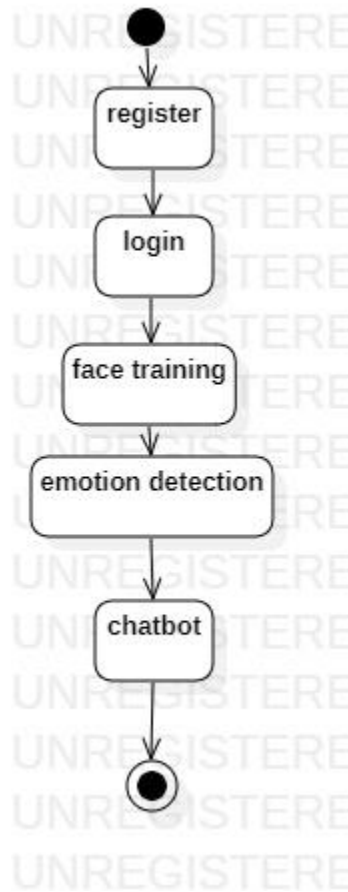


Fig 4.3 Activity Diagram

4.2.4 Class Diagram

A class diagram is a type of static structure diagram in the Unified Modeling Language (UML) that represents the structure and relationships of classes within a system. It provides a visual representation of the classes, their attributes, methods, and the associations between them.

Here's a breakdown of the components typically found in a class diagram:

Class: A class is represented as a rectangle with three compartments:

The top compartment contains the class name.

The middle compartment contains the attributes (data members) of the class.

The bottom compartment contains the methods (member functions) of the class.

Association: An association represents a relationship between two classes. It is depicted as a line connecting the classes, with optional multiplicity and role names indicating the nature of the association.

Generalization (Inheritance): Generalization represents an "is-a" relationship between classes, where one class (subclass or child) inherits attributes and behaviors from another class (superclass or parent). It is depicted as a solid line with a hollow arrowhead pointing from the subclass to the superclass.

Dependency: Dependency represents a relationship where one class depends on another class. It is typically shown as a dashed line with an arrowhead pointing from the dependent class to the independent class.

Aggregation and Composition: Aggregation and composition represent "whole-part" relationships between classes. Aggregation implies a weaker form of ownership, where the parts can exist independently of the whole, while composition implies a stronger form of ownership, where the parts are destroyed when the whole is destroyed. They are represented using diamond-shaped lines with filled or unfilled diamonds.

Multiplicity: Multiplicity indicates the number of instances of one class that are associated with a single instance of another class. It is denoted using numbers or symbols near the ends of association lines.

Class diagrams are commonly used during the design phase of software development to model the structure of object-oriented systems. They provide a high-level overview of the system's architecture, aiding in communication among stakeholders, software designers, and developers. Additionally, class diagrams serve as a blueprint for generating code and implementing the system.

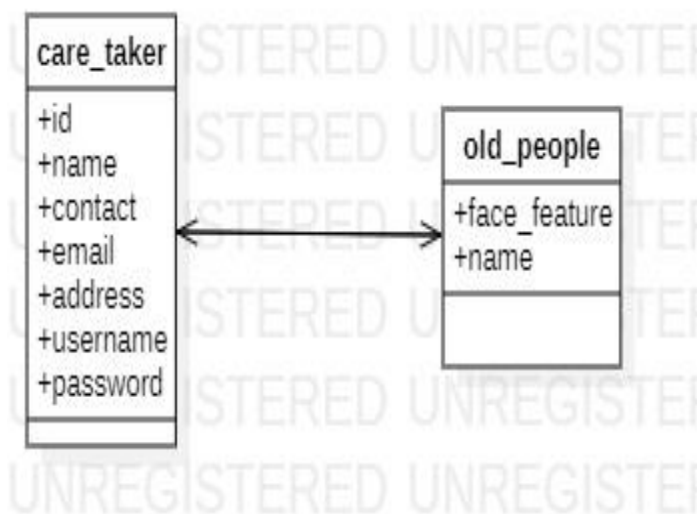


Fig 4.4 Class diagram

CHAPTER 5

SYSTEM ARCHITECTURE

The system architecture for animal detection typically involves a combination of hardware and software components designed to identify and classify recommendation based on input data, such as images or sensor readings. Below is a conceptual overview of the system architecture for virtual blind people detection:

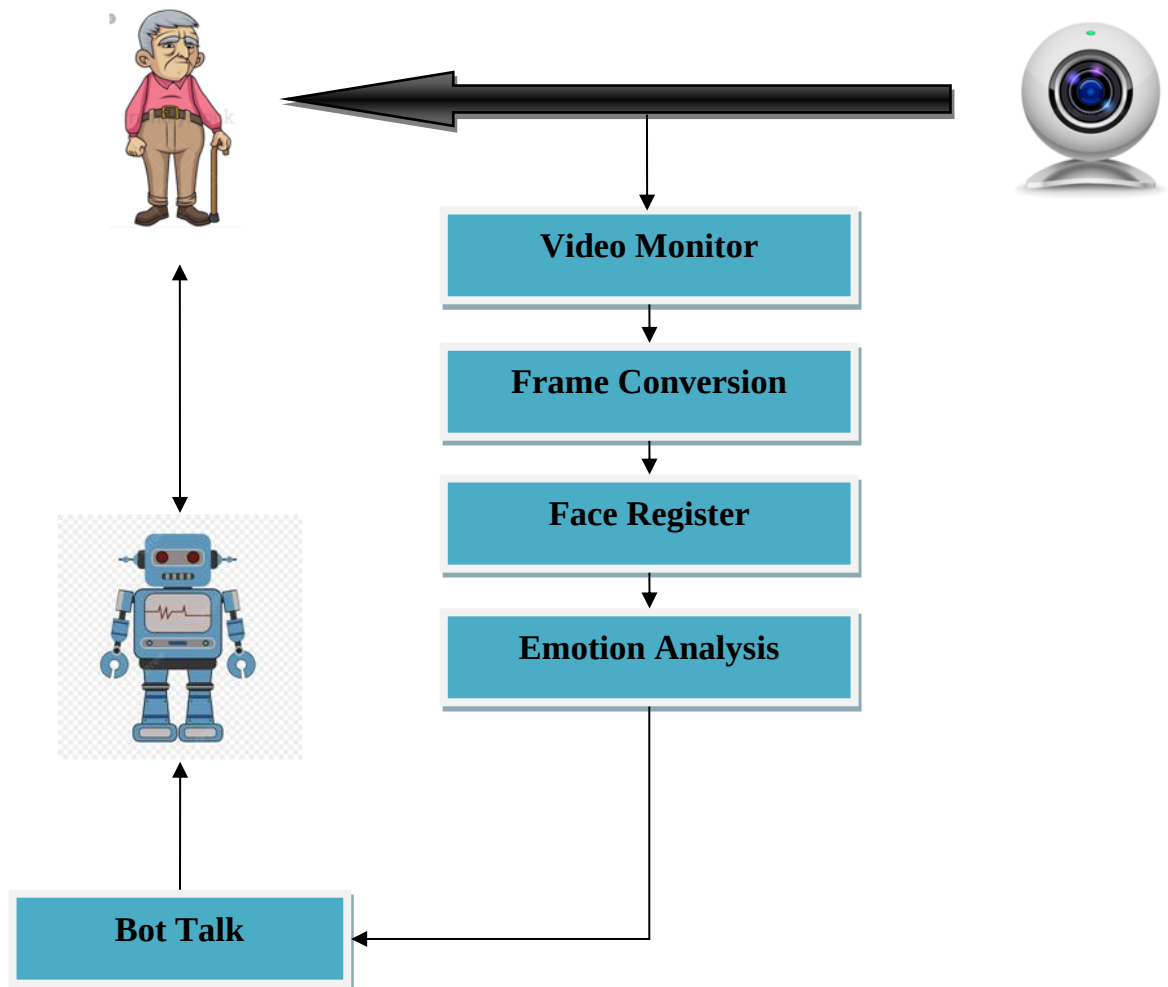


Fig 5.1 System Architecture

CHAPTER 6

METHODOLOGIES

6.1 EXISTING SYSTEM:

In the existing system, a normal emotion analysis can be made. The old people can live a lonely life without anyone to take care of them. Normal emotion maintenance can be done with the existing systems.

Disadvantages:

- **Privacy Concerns:** Analyzing facial expressions for emotional assessment raises privacy concerns, as it involves capturing and processing sensitive personal data.
- **Ethical Considerations:** There are ethical implications surrounding the use of facial emotion analysis, particularly in vulnerable populations like seniors. Issues such as consent, autonomy, and the potential for manipulation need to be carefully addressed.
- **Accuracy Limitations:** Facial emotion analysis algorithms may not always accurately interpret emotions, leading to misunderstandings or inappropriate responses from the robotic assistant.
- **Technological Limitations:** The effectiveness of the system heavily depends on the accuracy and reliability of the technology, which may still be in development and prone to errors or biases.
- **Dependency Risk:** Overreliance on robotic assistants for emotional support and companionship may reduce human interaction and lead to a decline in social skills and emotional resilience among seniors.
- **Cost:** Implementing and maintaining a system for smart senior citizen facial emotion analysis with robotic assistance can be costly, potentially limiting its accessibility to certain populations or healthcare providers.
- **Cultural Sensitivity:** Facial expressions and emotional cues can vary across cultures, which may pose challenges in developing universal algorithms that accurately interpret emotions for senior citizens from diverse backgrounds.

6.2 PROPOSED SYSTEM:

In the proposed system, HAAR cascade and BPNN based face and emotion detection can be done. The system can be implemented for the senior citizen where their face detection are done with multiple featured attribute. Matching with the trained feature attribute the facial emotion are detected.

Advantages:

- **Enhanced Care:** Facial emotion analysis can help robotic assistants better understand the emotional state of senior citizens, allowing them to provide more personalized and empathetic care.
- **Early Detection of Health Issues:** By analyzing facial expressions, the system can detect early signs of health issues such as depression or cognitive decline, enabling timely interventions.
- **Improved Communication:** Robotic assistants equipped with emotion analysis capabilities can communicate more effectively with seniors, leading to better engagement and understanding.
- **Assistance with Daily Tasks:** Understanding emotions can enable the robot to assist seniors with daily tasks in a more sensitive and responsive manner, enhancing their independence and quality of life.
- **Companionship:** Seniors may benefit from the companionship provided by robotic assistants that can understand and respond to their emotional cues, reducing feelings of loneliness and isolation.

6.3 MODULES

- Video acquisition
- Frame Conversion
- Haar cascade
- BPNN Classification
- Emotion Detection

- Chat Training
- HMM classification

6.4 MODULES DESCRIPTION

VIDEO ACQUISITION

The system's main objective is to improve the videos produced by thermal imagers and provide them the ability to acquire, stream, and analyze videos. In the FPGA's PL section, video capturing and processing units will be installed. These components will be in charge of PAL video data reception at 25 frames per second and some video processing, including 2D-FFT/IFFT and bilateral/Gaussian filtering. PAL video at 25 frames per second generates frames at a raw data rate of 20 Mbytes per second; Ethernet and USB are unable to handle this high pace. As a result, the video data must be compressed before being streamed over Ethernet. JPEG compression is what is utilized and the taken video from the surveillance camera is acquainted.

FRAME CONVERSION

Frame rate conversion is typically used when producing content for devices that use different standards (e.g. NTSC vs. PAL) or different content playback scenarios (e.g. film at 24 fps vs. television at 25 fps or 29.97 fps). Frame processing allows for the combination of multiple captured frames into one recorded frame. The combination occurs before the resulting frame is encoded. You can select the following frame processing settings: No Frame Processing, Frame Summing, Frame Averaging.

HAAR CASCADES

Haar cascade is an algorithm that can detect objects in images, irrespective of their scale in image and location. This algorithm is not so complex and can run in real-time. We can train a haar-cascade detector to detect various objects like cars, bikes, buildings, fruits, Haar Cascade Detection is one of the oldest yet powerful face detection algorithms invented. It has been there since long, long before Deep Learning became famous. Haar Features were not only used to detect faces, but also for eyes, lips, etc

BPNN CLASSIFICATION

Back propagation is the essence of neural network training. It is the method of fine-tuning the weights of a neural network based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and make the model reliable by increasing its generalization. Backpropagation in neural network is a short form for “backward propagation of errors.” It is a standard method of training artificial neural networks. This method helps calculate the gradient of a loss function with respect to all the weights in the network it is used to classify the different types of emotions.

EMOTION DETECTION

The emotion of the old people is detected using the BPNN classification algorithm. This emotion details are transferred to the chatting assistance.

CHAT TRAINING

The chat training module will define the process of steps to handle the old people to change their low level emotion to happy mood.

HMM CHAT

Hidden mor Kov model chat is a voice communication process shown out on an extreme level of data analysis where the given sign with the communication text will be converted as voice Based process. The maintenance over the application is done with Hidden Markov Model (HMM) technique where the text signals are added with noise and converted to speech.

CHAPTER 7

CODING AND SCREENSHOT

7.1 SAMPLE CODING

Home

```
import ar_masterTk
ar=ar_masterTk.tk_master()
root=ar.set_window_design()
```

Master flask code

```
import pymysql
class master_flask_code:
def __init__(self):
self.user = 'root'
self.password = ""
self.host = 'localhost'
self.database = 'python_elder_person_monitoring'
def find_max_id(self,table):
conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
database=self.database)
cursor = conn.cursor()
cursor.execute("SELECT id FROM "+table)
data = cursor.fetchall()
maxin = len(data)
if maxin == 0:
maxin = 1
else:
maxin += 1
return maxin
def insert_query(self,qry):
conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
database=self.database)
cursor = conn.cursor()
result=cursor.execute(qry)
conn.commit()
conn.close()
return result
def select_login(self,qry):
conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
database=self.database)
cursor = conn.cursor()
```

```

        cursor.execute(qry)
        data = cursor.fetchall()
        check = len(data)
        if check == 0:
            return 'no'
        else:
            return 'yes'
    def select_single_colum(self, table, column):
        conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
        database=self.database)
        qry1 = ("select " + column + " from " + table)
        cursor = conn.cursor()
        cursor.execute(qry1)
        data = cursor.fetchall()
        return data
    def select_entire_colum(self,table,column):
        conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
        database=self.database)
        qry1=("select * from "+table+" where symptoms LIKE '%" +column+"%'")
        cursor = conn.cursor()
        cursor.execute(qry1)
        data = cursor.fetchall()
        return data
    def select_direct_query(self,qry):
        conn = pymysql.connect(user=self.user, password=self.password, host=self.host,
        database=self.database)
        cursor = conn.cursor()
        cursor.execute(qry)
        data = cursor.fetchall()
        return data

```

VIDEO ANALYSIS

```

import numpy as np
import matplotlib.pyplot as plt
import cv2
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense
from keras.optimizers import Adam
from keras.models import Sequential
import os
from keras.preprocessing.image import ImageDataGenerator
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from tkinter import messagebox

from tkinter import *

```

```

def plot_model_history(model_history):
    """
    Plot Accuracy and Loss curves given the model_history
    """
    fig, axs = plt.subplots(1,2,figsize=(15,5))

    axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy'])

    axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_accuracy'])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')

    axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.history['accuracy']/10))
    axs[0].legend(['train', 'val'], loc='best')
    # summarize history for loss
    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])

    axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')

    axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['loss']/10))
    axs[1].legend(['train', 'val'], loc='best')
    fig.savefig('plot.png')
    plt.show()
    train_dir = 'data/train'
    val_dir = 'data/test'

    num_train = 28709
    num_val = 7178
    batch_size = 64
    num_epoch = 50

    train_datagen = ImageDataGenerator(rescale=1./255)
    val_datagen = ImageDataGenerator(rescale=1./255)

    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(48,48),
        batch_size=batch_size,
        color_mode="grayscale",

```

```

class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
mode="display"
from PIL import ImageTk,Image
import sample_data

def testing1():
    result=sample_data.student.result
    if result=='Failed':
        messagebox.showinfo(title="info", message=str(result))
    else:
        print(result)

def testing():
    if mode == "train":
        model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0001, decay=1e-6),
metrics=['accuracy'])
        model_info = model.fit_generator(
            train_generator,
            steps_per_epoch=num_train // batch_size,
            epochs=num_epoch,
            validation_data=validation_generator,
            validation_steps=num_val // batch_size)

```

```

plot_model_history(model_info)
model.save_weights('model.h5')
elif mode == "display":
    model.load_weights('model.h5')
    cv2.ocl.setUseOpenCL(False)
    emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5:
"Sad", 6: "Surprised"}
    rr=0
    result="Failed"
    if rr==0:
        dd= ('photo.jpg')
        frame = cv2.imread(dd)
        facecasc = cv2.CascadeClassifier('data/haarcascades/haarcascade_frontalface_default.xml')
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = facecasc.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y - 50), (x + w, y + h + 10), (255, 0, 0), 2)
            roi_gray = gray[y:y + h, x:x + w]
            cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
            prediction = model.predict(cropped_img)
            maxindex = int(np.argmax(prediction))
            print(emotion_dict[maxindex])
            result=emotion_dict[maxindex]
            sample_data.student.result=result
            messagebox.showinfo(title="info", message=str(result))
            root = Tk()
            w=750
            h=550
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample_data.student.title)
root.resizable(False, False)
root.configure(background=sample_data.student.background)
message = Label(root,
text=sample_data.student.titlec,fg=sample_data.student.text_color,bg=sample_data.student.back
ground, width=35,height=3, font=('times', 30, 'italic bold '))
message.place(x=00, y=20)
ri2 = Image.open('photo.jpg')
ri2 = ri2.resize((230, 200), Image.ANTIALIAS)
r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample_data.student.background)

```

```

lbl = Label(root, image=r2, background=sample_data.student.background,
fg=sample_data.student.text_color, font=('times', 15, ' bold '))
lbl.place(x=300, y=150)

button with command function
compare_dataset = Button(root, text="Emotion",width=15,command=testing
,height=1,fg="#FFF",bg="#004080", activebackground = "#ff8000",activeforeground="white"
,font=('times', 15, ' bold '))
compare_dataset.place(x=250, y=400)
resust_dataset = Button(root, text="Play Music",width=15
,height=1,command=testing1,fg="#FFF",bg="#004080", activebackground =
"#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))
resust_dataset.place(x=450, y=400)

root.mainloop()
import numpy as np
import matplotlib.pyplot as plt
import cv2
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense
from keras.optimizers import Adam
from keras.models import Sequential
import os
from keras.preprocessing.image import ImageDataGenerator
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from tkinter import messagebox

from tkinter import *
def plot_model_history(model_history):

Plot Accuracy and Loss curves given the model_history

fig, axs = plt.subplots(1,2,figsize=(15,5))
summarize history for accuracy

axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy'])

axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_acc
uracy'])
axs[0].set_title('Model Accuracy')
axs[0].set_ylabel('Accuracy')
axs[0].set_xlabel('Epoch')

axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.histor
y['accuracy']/10)
axs[0].legend(['train', 'val'], loc='best')
# summarize history for loss

```

```

    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])

axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')

axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['loss']/10))
    axs[1].legend(['train', 'val'], loc='best')
    fig.savefig('plot.png')
    plt.show()
train_dir = 'data/train'
val_dir = 'data/test'

num_train = 28709
num_val = 7178
batch_size = 64
num_epoch = 50

train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))

```

```

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
mode="display"
from PIL import ImageTk,Image
import sample_data

def testing1():
    result=sample_data.student.result
    if result=='Failed':
        messagebox.showinfo(title="info", message=str(result))
    else:
        print(result)

def testing():
    if mode == "train":
        model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0001, decay=1e-6),
metrics=['accuracy'])
        model_info = model.fit_generator(
            train_generator,
            steps_per_epoch=num_train // batch_size,
            epochs=num_epoch,
            validation_data=validation_generator,
            validation_steps=num_val // batch_size)
        plot_model_history(model_info)
        model.save_weights('model.h5')
    elif mode == "display":
        model.load_weights('model.h5')
        cv2.ocl.setUseOpenCL(False)
        emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5:
        "Sad", 6: "Surprised"}
        rr=0
        result="Failed"
        if rr==0:
            dd= ('photo.jpg')
            frame = cv2.imread(dd)
            facecasc =
cv2.CascadeClassifier('data/haarcascades/haarcascade_frontalface_default.xml')
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = facecasc.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
            for (x, y, w, h) in faces:
                cv2.rectangle(frame, (x, y - 50), (x + w, y + h + 10), (255, 0, 0), 2)

```

```

        roi_gray = gray[y:y + h, x:x + w]
        cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
        prediction = model.predict(cropped_img)
        maxindex = int(np.argmax(prediction))
        print(emotion_dict[maxindex])
        result=emotion_dict[maxindex]
        sample_data.student.result=result
        messagebox.showinfo(title="info", message=str(result))
root = Tk()
w=750
h=550
ws = root.winfo_screenwidth()
hs = root.winfo_screenheight()
x = (ws/2) - (w/2)
y = (hs/2) - (h/2)
root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample_data.student.title)
root.resizable(False, False)
root.configure(background=sample_data.student.background)
message = Label(root,
text=sample_data.student.titlec,fg=sample_data.student.text_color,bg=sample_data.student.back
ground, width=35,height=3, font=('times', 30, 'italic bold '))
message.place(x=00, y=20)
ri2 = Image.open('photo.jpg')
ri2 = ri2.resize((230, 200), Image.ANTIALIAS)
r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample_data.student.background)
lbl = Label(root, image=r2, background=sample_data.student.background,
fg=sample_data.student.text_color, font=('times', 15, ' bold '))
lbl.place(x=300, y=150)

##### button with command function
compare_dataset = Button(root, text="Emotion",width=15,command=testing
,height=1,fg="#FFF",bg="#004080", activebackground = "#ff8000",activeforeground="white"
,font=('times', 15, ' bold '))
compare_dataset.place(x=250, y=400)
resust_dataset = Button(root, text="Play Music",width=15
,height=1,command=testing1,fg="#FFF",bg="#004080", activebackground =
"#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))
resust_dataset.place(x=450, y=400)

root.mainloop()
import numpy as np
import matplotlib.pyplot as plt

```

```

import cv2
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense
from keras.optimizers import Adam
from keras.models import Sequential
import os
from keras.preprocessing.image import ImageDataGenerator
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from tkinter import messagebox

from tkinter import *
def plot_model_history(model_history):
    """
    Plot Accuracy and Loss curves given the model_history
    """
    fig, axs = plt.subplots(1,2,figsize=(15,5))
    # summarize history for accuracy

    axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy'])

    axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_acc
uracy'])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')

    axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.histor
y['accuracy']/10)
    axs[0].legend(['train', 'val'], loc='best')
    # summarize history for loss
    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])

    axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')

    axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['lo
ss']/10)
    axs[1].legend(['train', 'val'], loc='best')
    fig.savefig('plot.png')
    plt.show()
train_dir = 'data/train'
val_dir = 'data/test'

num_train = 28709
num_val = 7178

```

```

batch_size = 64
num_epoch = 50

train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
mode="display"
from PIL import ImageTk,Image
import sample_data

def testing1():
    result=sample_data.student.result
    if result=='Failed':
        messagebox.showinfo(title="info", message=str(result))
    else:
        print(result)

```

```

def testing():
    if mode == "train":
        model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0001, decay=1e-6),
metrics=['accuracy'])
        model_info = model.fit_generator(
            train_generator,
            steps_per_epoch=num_train // batch_size,
            epochs=num_epoch,
            validation_data=validation_generator,
            validation_steps=num_val // batch_size)
        plot_model_history(model_info)
        model.save_weights('model.h5')
    elif mode == "display":
        model.load_weights('model.h5')
        cv2.ocl.setUseOpenCL(False)
        emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5:
"Sad", 6: "Surprised"}
        rr=0
        result="Failed"
        if rr==0:
            dd= ('photo.jpg')
            frame = cv2.imread(dd)
            facecasc =
cv2.CascadeClassifier('data/haarcascades/haarcascade_frontalface_default.xml')
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = facecasc.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
            for (x, y, w, h) in faces:
                cv2.rectangle(frame, (x, y - 50), (x + w, y + h + 10), (255, 0, 0), 2)
                roi_gray = gray[y:y + h, x:x + w]
                cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
                prediction = model.predict(cropped_img)
                maxindex = int(np.argmax(prediction))
                print(emotion_dict[maxindex])
                result=emotion_dict[maxindex]
                sample_data.student.result=result
            messagebox.showinfo(title="info", message=str(result))
        root = Tk()
        w=750
        h=550
        ws = root.winfo_screenwidth()
        hs = root.winfo_screenheight()
        x = (ws/2) - (w/2)
        y = (hs/2) - (h/2)
        root.geometry('%dx%d+%d+%d' % (w, h, x, y))

```

```

root.title(sample_data.student.title)
root.resizable(False, False)
root.configure(background=sample_data.student.background)
design
message = Label(root,
text=sample_data.student.titlec,fg=sample_data.student.text_color,bg=sample_data.student.back
ground, width=35,height=3, font=('times', 30, 'italic bold '))
message.place(x=00, y=20)
#####
ri2 = Image.open('photo.jpg')
ri2 = ri2.resize((230, 200), Image.ANTIALIAS)
r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample_data.student.background)
lbl = Label(root, image=r2, background=sample_data.student.background,
fg=sample_data.student.text_color, font=('times', 15, ' bold '))
lbl.place(x=300, y=150)

##### button with command function
compare_dataset = Button(root, text="Emotion",width=15,command=testing
,height=1,fg="#FFF",bg="#004080", activebackground = "#ff8000",activeforeground="white"
,font=('times', 15, ' bold '))
compare_dataset.place(x=250, y=400)
resust_dataset = Button(root, text="Play Music",width=15
,height=1,command=testing1,fg="#FFF",bg="#004080", activebackground =
"#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))
resust_dataset.place(x=450, y=400)

root.mainloop()
import numpy as np
import matplotlib.pyplot as plt
import cv2
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense
from keras.optimizers import Adam
from keras.models import Sequential
import os
from keras.preprocessing.image import ImageDataGenerator
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from tkinter import messagebox

from tkinter import *
def plot_model_history(model_history):
    """
    Plot Accuracy and Loss curves given the model_history
    """

```

```

fig, axs = plt.subplots(1,2,figsize=(15,5))
# summarize history for accuracy

axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy'])

axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_acc
uracy'])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')

axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.histor
y['accuracy']/10)
    axs[0].legend(['train', 'val'], loc='best')
# summarize history for loss
axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])

axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')

axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['lo
ss']/10)
    axs[1].legend(['train', 'val'], loc='best')
    fig.savefig('plot.png')
    plt.show()
train_dir = 'data/train'
val_dir = 'data/test'

num_train = 28709
num_val = 7178
batch_size = 64
num_epoch = 50

train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(

```

```

    val_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
mode="display"
from PIL import ImageTk,Image
import sample_data

def testing1():
    result=sample_data.student.result
    if result=='Failed':
        messagebox.showinfo(title="info", message=str(result))
    else:
        print(result)

def testing():
    if mode == "train":
        model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0001, decay=1e-6),
metrics=['accuracy'])
        model_info = model.fit_generator(
            train_generator,
            steps_per_epoch=num_train // batch_size,
            epochs=num_epoch,
            validation_data=validation_generator,
            validation_steps=num_val // batch_size)
        plot_model_history(model_info)
        model.save_weights('model.h5')
    elif mode == "display":

```

```

model.load_weights('model.h5')
cv2ocl.setUseOpenCL(False)
emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5:
"Sad", 6: "Surprised"}
rr=0
result="Failed"
if rr==0:
    dd= ('photo.jpg')
    frame = cv2.imread(dd)
    facecasc =
cv2.CascadeClassifier('data/haarcascades/haarcascade_frontalface_default.xml')
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = facecasc.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y - 50), (x + w, y + h + 10), (255, 0, 0), 2)
        roi_gray = gray[y:y + h, x:x + w]
        cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
        prediction = model.predict(cropped_img)
        maxindex = int(np.argmax(prediction))
        print(emotion_dict[maxindex])
        result=emotion_dict[maxindex]
        sample_data.student.result=result
    messagebox.showinfo(title="info", message=str(result))
root = Tk()
w=750
h=550
ws = root.winfo_screenwidth()
hs = root.winfo_screenheight()
x = (ws/2) - (w/2)
y = (hs/2) - (h/2)
root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample_data.student.title)
root.resizable(False, False)
root.configure(background=sample_data.student.background)
design
message = Label(root,
text=sample_data.student.titlec,fg=sample_data.student.text_color,bg=sample_data.student.back
ground, width=35,height=3, font=('times', 30, 'italic bold '))
message.place(x=00, y=20)
#####
ri2 = Image.open('photo.jpg')
ri2 = ri2.resize((230, 200), Image.ANTIALIAS)
r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample_data.student.background)

```

```

lbl = Label(root, image=r2, background=sample_data.student.background,
fg=sample_data.student.text_color, font=('times', 15, ' bold '))
lbl.place(x=300, y=150)

compare_dataset = Button(root, text="Emotion",width=15,command=testing
,height=1,fg="#FFF",bg="#004080", activebackground = "#ff8000",activeforeground="white"
,font=('times', 15, ' bold '))
compare_dataset.place(x=250, y=400)
resust_dataset = Button(root, text="Play Music",width=15
,height=1,command=testing1,fg="#FFF",bg="#004080", activebackground =
"#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))
resust_dataset.place(x=450, y=400)

root.mainloop()
import numpy as np
import matplotlib.pyplot as plt
import cv2
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense
from keras.optimizers import Adam
from keras.models import Sequential
import os
from keras.preprocessing.image import ImageDataGenerator
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from tkinter import messagebox

from tkinter import *
def plot_model_history(model_history):
    """
    Plot Accuracy and Loss curves given the model_history
    """
    fig, axs = plt.subplots(1,2,figsize=(15,5))
    # summarize history for accuracy

    axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy'])

    axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_acc
uracy'])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')

    axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.histor
y['accuracy']/10)
    axs[0].legend(['train', 'val'], loc='best')
    # summarize history for loss
    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])

```

```

axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')

axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['loss'])/10)
    axs[1].legend(['train', 'val'], loc='best')
    fig.savefig('plot.png')
    plt.show()
train_dir = 'data/train'
val_dir = 'data/test'

num_train = 28709
num_val = 7178
batch_size = 64
num_epoch = 50

train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

```

```

model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
mode="display"
from PIL import ImageTk,Image
import sample_data

def testing1():
    result=sample_data.student.result
    if result=='Failed':
        messagebox.showinfo(title="info", message=str(result))
    else:
        print(result)

def testing():
    if mode == "train":
        model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0001, decay=1e-6),
metrics=['accuracy'])
        model_info = model.fit_generator(
            train_generator,
            steps_per_epoch=num_train // batch_size,
            epochs=num_epoch,
            validation_data=validation_generator,
            validation_steps=num_val // batch_size)
        plot_model_history(model_info)
        model.save_weights('model.h5')
    elif mode == "display":
        model.load_weights('model.h5')
        cv2.ocl.setUseOpenCL(False)
        emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5:
"Sad", 6: "Surprised"}
        rr=0
        result="Failed"
        if rr==0:
            dd= ('photo.jpg')
            frame = cv2.imread(dd)
            facecasc =
cv2.CascadeClassifier('data/haarcascades/haarcascade_frontalface_default.xml')
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = facecasc.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
            for (x, y, w, h) in faces:
                cv2.rectangle(frame, (x, y - 50), (x + w, y + h + 10), (255, 0, 0), 2)
                roi_gray = gray[y:y + h, x:x + w]

```

```

        cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
        prediction = model.predict(cropped_img)
        maxindex = int(np.argmax(prediction))
        print(emotion_dict[maxindex])
        result=emotion_dict[maxindex]
        sample_data.student.result=result
        messagebox.showinfo(title="info", message=str(result))
root = Tk()
w=750
h=550
ws = root.winfo_screenwidth()
hs = root.winfo_screenheight()
x = (ws/2) - (w/2)
y = (hs/2) - (h/2)
root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample_data.student.title)
root.resizable(False, False)
root.configure(background=sample_data.student.background)
message = Label(root,
text=sample_data.student.titlec,fg=sample_data.student.text_color,bg=sample_data.student.back
ground, width=35,height=3, font=('times', 30, 'italic bold '))
message.place(x=00, y=20)
#####
ri2 = Image.open('photo.jpg')
ri2 = ri2.resize((230, 200), Image.ANTIALIAS)
r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample_data.student.background)
lbl = Label(root, image=r2, background=sample_data.student.background,
fg=sample_data.student.text_color, font=('times', 15, ' bold '))
lbl.place(x=300, y=150)

##### button with command function
compare_dataset = Button(root, text="Emotion",width=15,command=testing
,height=1,fg="#FFF",bg="#004080", activebackground = "#ff8000",activeforeground="white"
,font=('times', 15, ' bold '))
compare_dataset.place(x=250, y=400)
resust_dataset = Button(root, text="Play Music",width=15
,height=1,command=testing1,fg="#FFF",bg="#004080", activebackground =
"#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))
resust_dataset.place(x=450, y=400)

root.mainloop()

master tk

```

```

import numpy as np
import matplotlib.pyplot as plt
import cv2
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense
from keras.optimizers import Adam
from keras.models import Sequential
import os
from keras.preprocessing.image import ImageDataGenerator
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
from tkinter import messagebox

from tkinter import *
def plot_model_history(model_history):
    """
    Plot Accuracy and Loss curves given the model_history
    """
    fig, axs = plt.subplots(1,2,figsize=(15,5))
    # summarize history for accuracy

    axs[0].plot(range(1,len(model_history.history['accuracy'])+1),model_history.history['accuracy'])

    axs[0].plot(range(1,len(model_history.history['val_accuracy'])+1),model_history.history['val_acc
uracy'])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')

    axs[0].set_xticks(np.arange(1,len(model_history.history['accuracy'])+1),len(model_history.histor
y['accuracy']/10)
    axs[0].legend(['train', 'val'], loc='best')
    # summarize history for loss
    axs[1].plot(range(1,len(model_history.history['loss'])+1),model_history.history['loss'])

    axs[1].plot(range(1,len(model_history.history['val_loss'])+1),model_history.history['val_loss'])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')

    axs[1].set_xticks(np.arange(1,len(model_history.history['loss'])+1),len(model_history.history['lo
ss']/10)
    axs[1].legend(['train', 'val'], loc='best')
    fig.savefig('plot.png')
    plt.show()
train_dir = 'data/train'
val_dir = 'data/test'

```

```

num_train = 28709
num_val = 7178
batch_size = 64
num_epoch = 50

train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(48,48),
    batch_size=batch_size,
    color_mode="grayscale",
    class_mode='categorical')
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
mode="display"
from PIL import ImageTk,Image
import sample_data

def testing1():
    result=sample_data.student.result
    if result=='Failed':

```

```

        messagebox.showinfo(title="info", message=str(result))
    else:
        print(result)

def testing():
    if mode == "train":
        model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0001, decay=1e-6),
metrics=['accuracy'])
        model_info = model.fit_generator(
            train_generator,
            steps_per_epoch=num_train // batch_size,
            epochs=num_epoch,
            validation_data=validation_generator,
            validation_steps=num_val // batch_size)
        plot_model_history(model_info)
        model.save_weights('model.h5')
    elif mode == "display":
        model.load_weights('model.h5')
        cv2.ocl.setUseOpenCL(False)
        emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5:
        "Sad", 6: "Surprised"}
        rr=0
        result="Failed"
        if rr==0:
            dd= ('photo.jpg')
            frame = cv2.imread(dd)
            facecasc =
cv2.CascadeClassifier('data/haarcascades/haarcascade_frontalface_default.xml')
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = facecasc.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
            for (x, y, w, h) in faces:
                cv2.rectangle(frame, (x, y - 50), (x + w, y + h + 10), (255, 0, 0), 2)
                roi_gray = gray[y:y + h, x:x + w]
                cropped_img = np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1), 0)
                prediction = model.predict(cropped_img)
                maxindex = int(np.argmax(prediction))
                print(emotion_dict[maxindex])
                result=emotion_dict[maxindex]
                sample_data.student.result=result
            messagebox.showinfo(title="info", message=str(result))
        root = Tk()
        w=750
        h=550
        ws = root.winfo_screenwidth()
        hs = root.winfo_screenheight()

```

```

x = (ws/2) - (w/2)
y = (hs/2) - (h/2)
root.geometry('%dx%d+%d+%d' % (w, h, x, y))

root.title(sample_data.student.title)
root.resizable(False, False)
root.configure(background=sample_data.student.background)
design
message = Label(root,
text=sample_data.student.titlec,fg=sample_data.student.text_color,bg=sample_data.student.back
ground, width=35,height=3, font=('times', 30, 'italic bold '))
message.place(x=00, y=20)
#####
ri2 = Image.open('photo.jpg')
ri2 = ri2.resize((230, 200), Image.ANTIALIAS)
r2 = ImageTk.PhotoImage(ri2)

label2 = Label(root, image=r2, background=sample_data.student.background)
lbl = Label(root, image=r2, background=sample_data.student.background,
fg=sample_data.student.text_color, font=('times', 15, ' bold '))
lbl.place(x=300, y=150)

##### button with command function
compare_dataset = Button(root, text="Emotion",width=15,command=testing
,height=1,fg="#FFF",bg="#004080", activebackground = "#ff8000",activeforeground="white"
,font=('times', 15, ' bold '))
compare_dataset.place(x=250, y=400)
resust_dataset = Button(root, text="Play Music",width=15
,height=1,command=testing1,fg="#FFF",bg="#004080", activebackground =
"#ff8000",activeforeground="white" ,font=('times', 15, ' bold '))
resust_dataset.place(x=450, y=400)

root.mainloop()

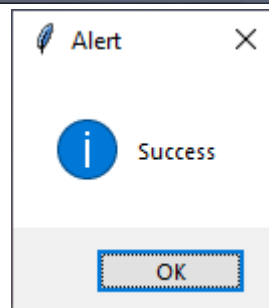
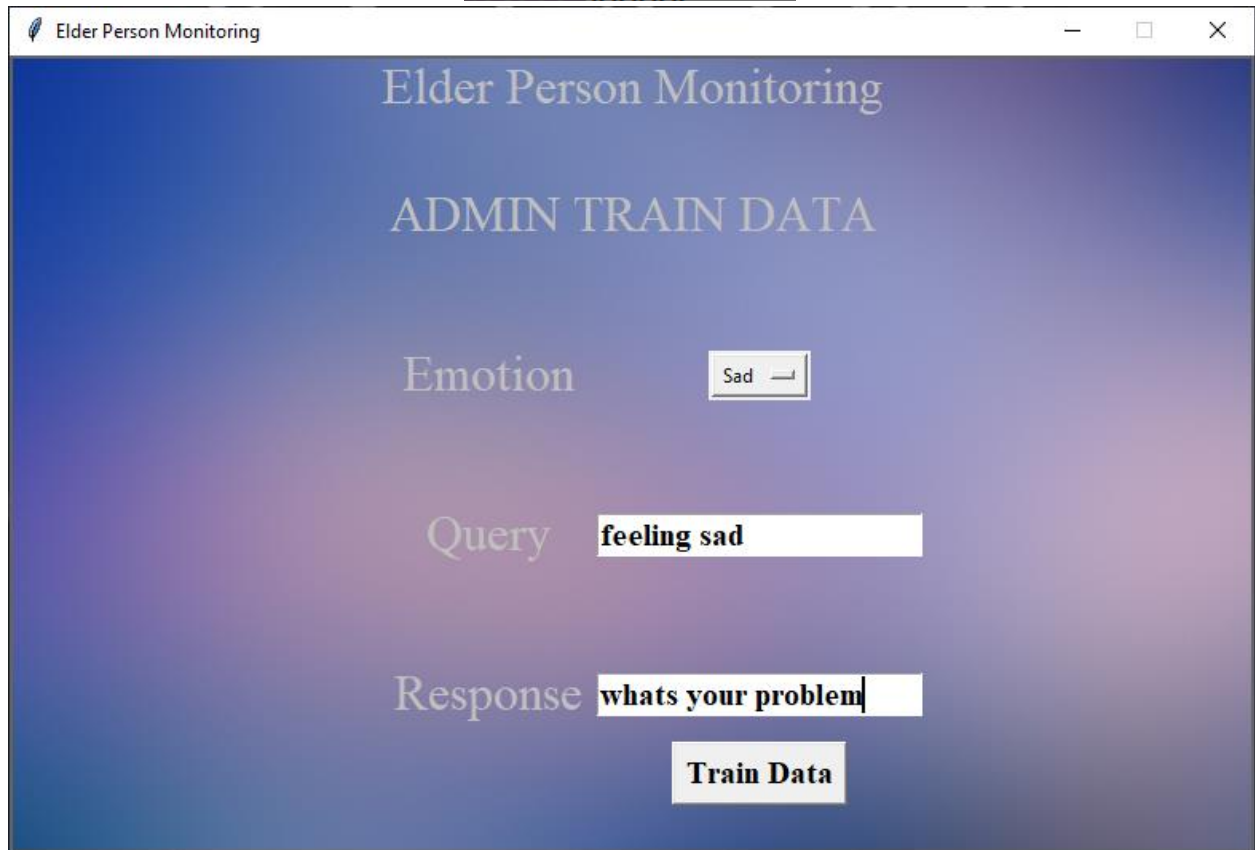
```

7.2 SAMPLE SCREENSHOT AND OUTPUT

The image displays two screenshots of a web application titled "Elder Person Monitoring".

The top screenshot shows the main menu with a blue background. It features the title "Elder Person Monitoring" at the top. Below the title, there are two options: "Admin" and "Care Taker". The "Admin" option is accompanied by a circular icon containing a gear and a person silhouette. The "Care Taker" option is accompanied by an icon showing two stylized figures, one purple and one blue, standing on a green mound.

The bottom screenshot shows the "ADMIN LOGIN" screen. It has a blue background and the title "Elder Person Monitoring" at the top. Below the title, the text "ADMIN LOGIN" is displayed. There are two input fields: "Username" with the value "admin" and "Password" with the value "*****". A "Login" button is located at the bottom center of the screen.



Elder Person Monitoring

CARE TAKER REGISTRATION

Name	helan
Contact	8273498237
Email	helan@gmail.com
Address	trichy
Username	helan
Password	123

Register



Elder Person Monitoring

CARE TAKER LOGIN

Username

Password

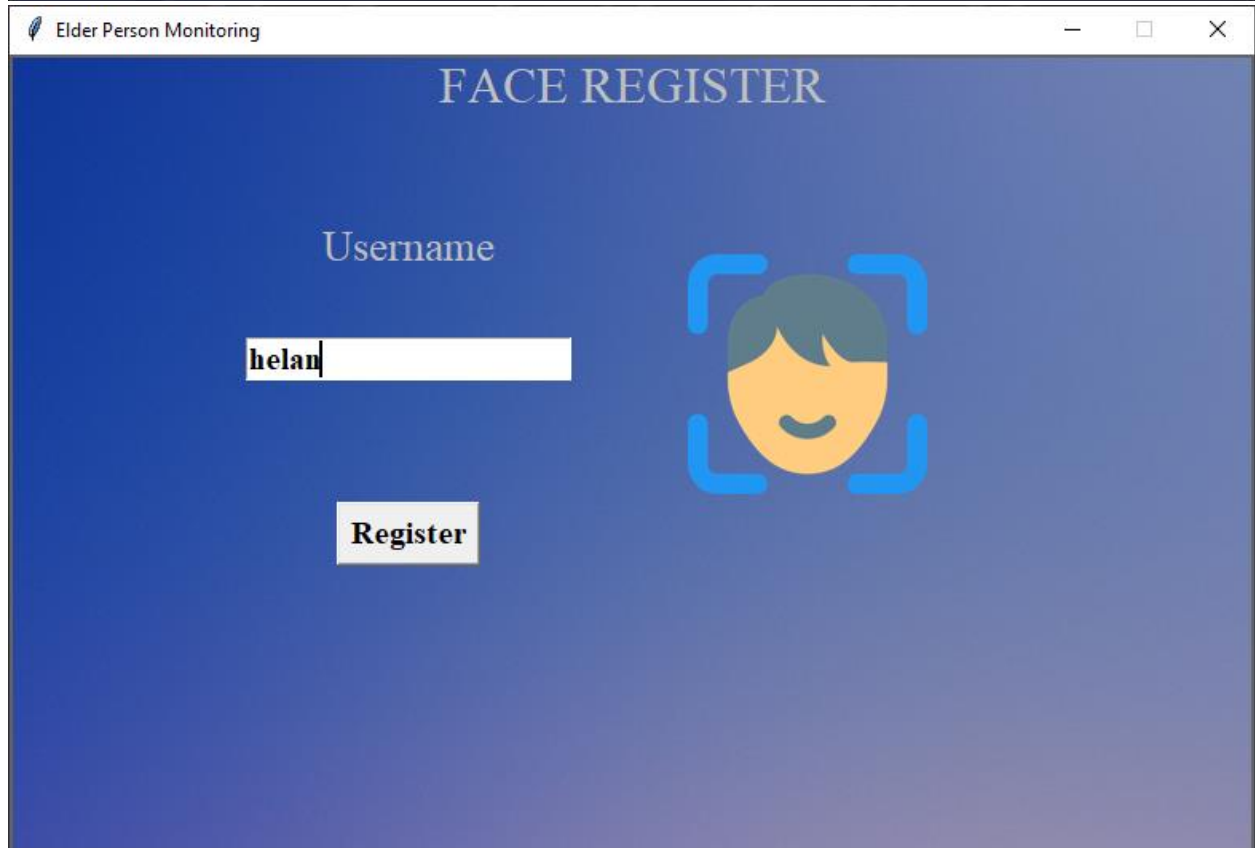
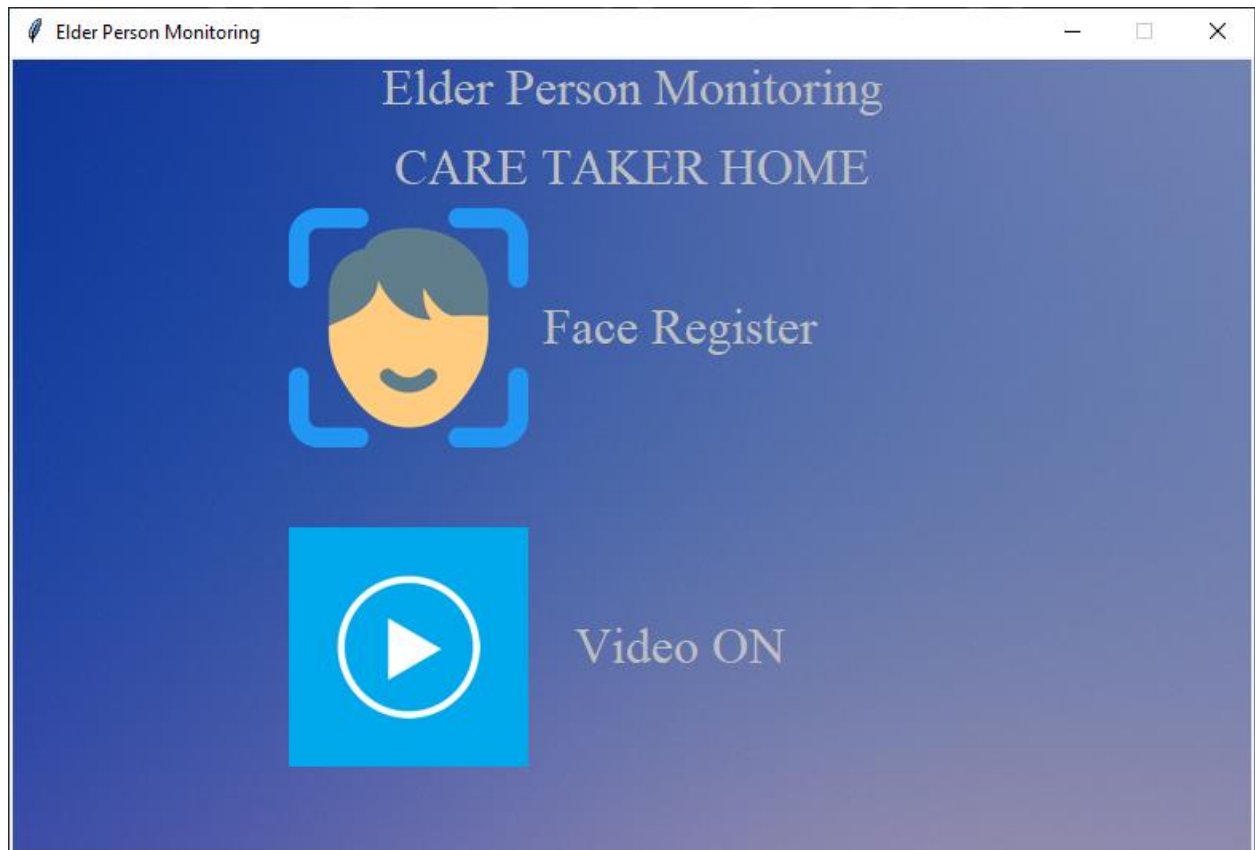
Login

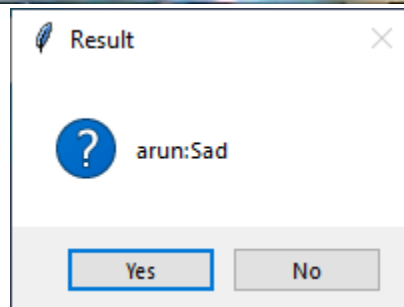
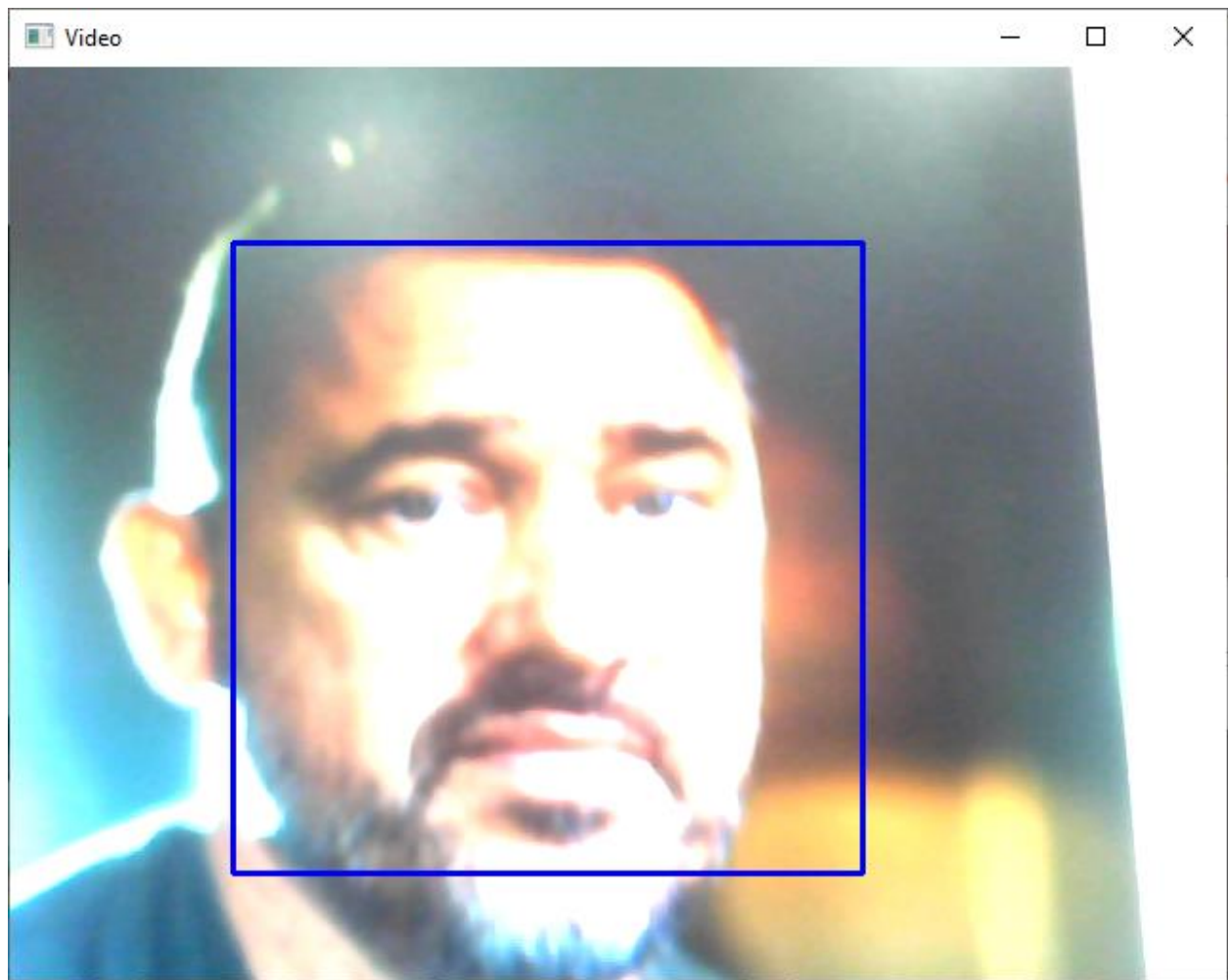
New Registration Here...

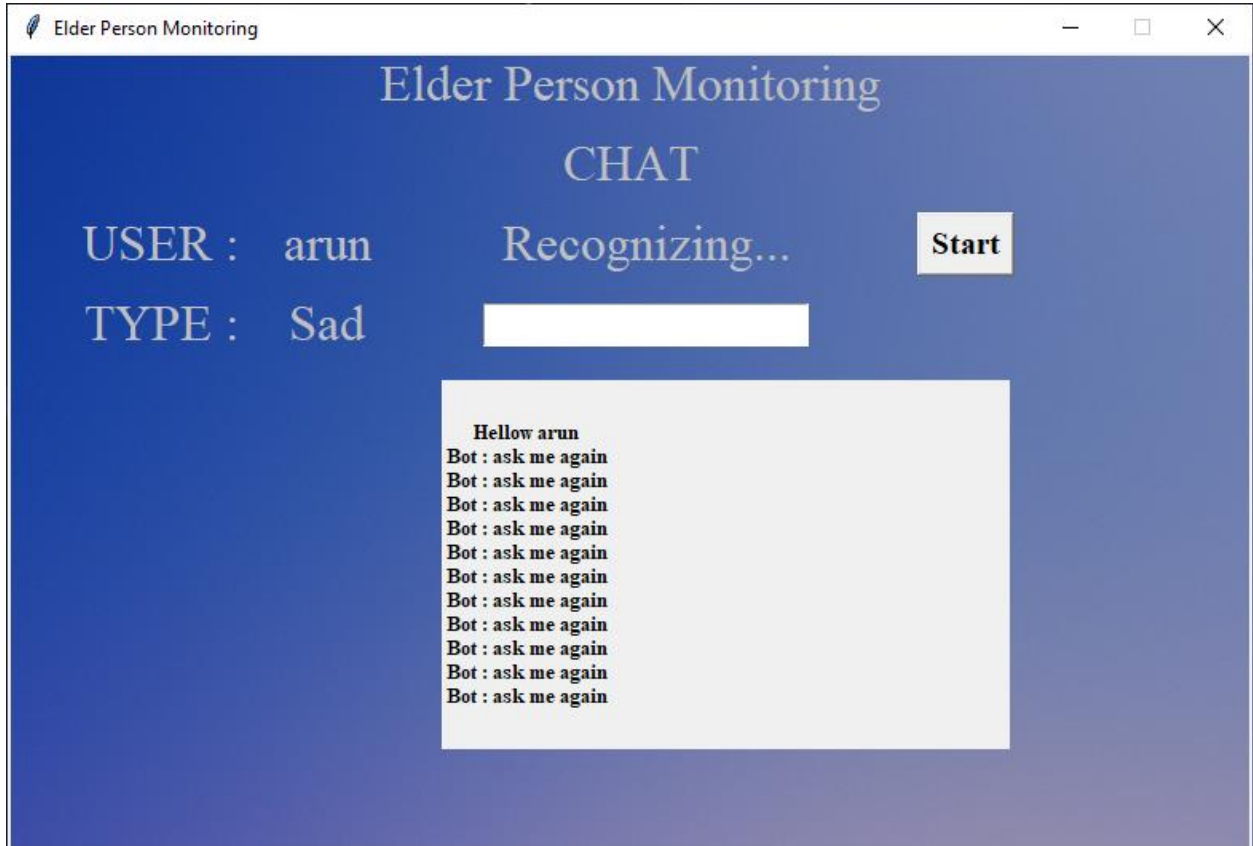
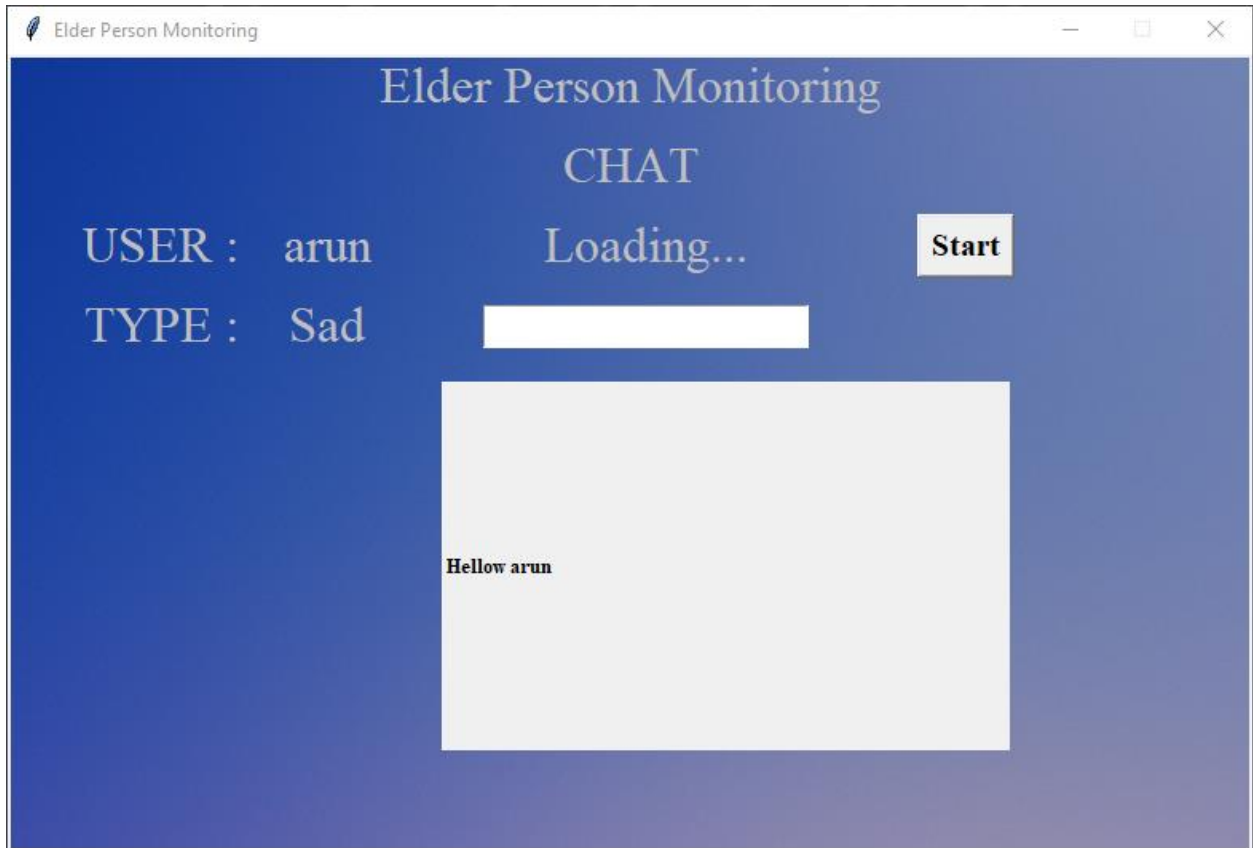
Result

 Login Success

OK







CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

This project contributes to the idea that the methodology used for examining emotion recognition in facial expressions may affect outcomes. This study explored differences between labeling and semi-open categorization paradigms. The attribution to found differences remains unclear and may urge researchers to take a different approach when studying emotion recognition, taking into account the effect of context on the interpretation of facial expressions and emotions, moving toward paradigms that come closer to representing real life emotion recognition. The project also helps elderly to change their emotion using chat training and HMM chat.

8.2 FUTURE ENHANCEMENT

In future development, the old people can be monitored with a real time surveillance with an IOT based robot creation. The emotion can be detected at a accurate rate of 100% generation system.

REFERENCES

- [1] Taneja Sheetal , “Python Programming |A modular approach | First Edition | By Pearson”, Pearson Education; First edition (26 September 2017)
- [2] Eric Matthes,” Python Crash Course”, No Starch Press; 2nd edition (3 May 2019)
- [3] Mark Lutz “Python Pocket Reference: Python in Your Pocket”, Shroff/O'Reilly; First edition (1 January 2014)
- [4] EUROSTAT. Mortality and Life Expectancy Statistics. 2021. Available online: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Mortality_and_life_expectancy_statistics (5 March 2021).
- [5] Freire Rodríguez, C.; Ferradás Canedo, M.d.M. Calidad de Vida y Bienestar en la Vejez; Ediciones Pirámide: Madrid, Spain, 2016.
- [6] INE. Encuesta Continua de Hogares (ECH) 2018. 2018. Available online: https://www.ine.es/prensa/ech_2018.pdf (10 March 2021).
- [7]] Unsar, S.; Dindar, I.; Kurt, S. Activities of daily living, quality of life, social support and depression levels of elderly individuals in Turkish society. J. Pak. Med. Assoc. 2015, 65, 14.
- [8] Holt-Lunstad, J. The Potential Public Health Relevance of Social Isolation and Loneliness: Prevalence, Epidemiology, and Risk Factors. Public Policy Aging Rep. 2017, 27, 127–130, doi:10.1093/ppar/prx030.
- [9] Holt-Lunstad, J.; Smith, T.B.; Layton, J.B. Social relationships and mortality risk: A meta-analytic review. PLoS Med. 2010, 7, e1000316, doi:10.1371/journal.pmed.1000316.
- [10] INE. Encuesta Continua de Hogares (ECH) 2020. 2021. Available online: https://www.ine.es/prensa/ech_2020.pdf (12 March 2021).
- [11] Pinazo-Hernandis, S. Impacto psicosocial de la COVID-19 en las personas mayores: Problemas y retos. Rev. Esp. Geriatr. Gerontol. 2020, 55, 249-252, doi:10.1016/j.regg.2020.05.006.

- [12] Turtlebot. TurtleBot2. Available online: <https://www.turtlebot.com/turtlebot2/> (14 March 2021).
- [13] Bautista-Salinas, D.; Gonzalez, J.R.; Mendez, I.; Mozos, O.M. Monitoring and Prediction of Mood in Elderly People during Daily Life Activities. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Berlin, Germany, 23–27 July 2019; doi:10.1109/EMBC.2019.8857847.
- [14] un, F.T.; Kuo, C.; Cheng, H.T.; Buthpitiya, S.; Collins, P.; Griss, M. Activity-aware mental stress detection using physiological sensors. In Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST; Springer: Berlin/Heidelberg, Germany, 2012; Volume 76.
- [15] Mozos, O.M.; Sandulescu, V.; Andrews, S.; Ellis, D.; Bellotto, N.; Dobrescu, R.; Ferrandez, J.M. Stress detection using wearable physiological and sociometric sensors. *Int. J. Neural Syst.* 2017, 27, 1650041, doi:10.1142/S0129065716500416.
- [16] Hovsepien, K.; Al’absi, M.; Ertin, E.; Kamarck, T.; Nakajima, M.; Kumar, S. CStress: Towards a gold standard for continuous stress assessment in the mobile environment. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Osaka, Japan, 7–11 September 2015, doi:10.1145/2750858.2807526.
- [17] Healey, J.A.; Picard, R.W. Detecting stress during real-world driving tasks using physiological sensors. *IEEE Trans. Intell. Transp. Syst.* 2005, 6, 156–166, doi:10.1109/TITS.2005.848368.
- [18] Magazine, D. El Robot Panasonic Hospi Rimo Ayuda en el Transporte de Fármacos a los Pacientes Hospitalizados. Available online: <https://www.digitalavmagazine.com/2013/10/29/el-robot-panasonic-hospi-rimo-ayuda-en-el-transporte-de-farmacos-a-los-pacientes-hospitalizados/> (16 March 2021).