A

Major Project

On

COMPUTER AUTOMATION USING GESTURE RECOGNITION AND MEDIAPIPE

(Submitted in partial fulfillment of the requirements for the award of degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

BY

Aditya Madhira (187R1A05F9)

Naresh Mote (187R1A05G6)

Under the Guidance of

V. Naresh Kumar

Assistant professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2018-22

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GROUP OF INSTITUTIONS

EXPLORE TO INVENT

CERTIFICATE

This is to certify that the project entitled "**COMPUTER AUTOMATION USING GESTURE RECOGNITION AND MEDIAPIPE**" being submitted by **ADITYA MADHIRA 187R1A05F9 & NARESH MOTE 187R1A05G6** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2021-22.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Mr. V. NARESH KUMAR
Assistant Professor
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on ____________________

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **Mr. V. Naresh Kumar,** Assistant Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) **Mr. J. Narasimha Rao, Dr. T. S. Mastan Rao, Mr. A. Uday Kiran, Mr. A. Kiran Kumar, Mrs. G. Latha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

ADITYA MADHIRA (187R1A05F9)
NARESH MOTE (187R1A05G6)

ABSTRACT

Human Computer Interaction has been a multidisciplinary field of study focusing on the design of computer technology and, in particular, the interaction between humans (the users) and computers. There are multiple ways to interact with a computer and are not limited to physical hardware devices. Gesture recognition is a computing process that attempts to recognize and interpret human gestures through the use of mathematical algorithms. Gesture recognition is not limited to just human hand gestures, but rather can be used to recognize everything from head nods to different walking gaits. Computer automation is another area where scientists are trying to automate mundane, time taking tasks. Basic tasks like "Shutting down", "Opening apps", "Visiting a particular URL" are tedious and can be automated. Utilizing the power of "Hand Tracking" and "Gesture Recognition", we can use our hands to control our system without ever touching mouse or keyboard.

LIST OF FIGURES

LIST OF SCREENSHOTS

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

This project is titled as “Computer automation using gesture recognition and mediapipe”. This software provides facility to use hand gestures to automate few computer tasks quickly. This project uses pre trained “convolutional neural network” to predict the gesture and mediapipe to perform hand tracking.

1.2 PROJECT PURPOSE

This has been developed to automate mundane computer tasks which require quickly and efficiently using gestures. The user can assign tasks to gestures which later can be used to perform few tasks. Mediapipe ensures constant tracking of both left and right hand for seamless performance.

1.3 PROJECT FEATURES

The main feature of this project is ‘Gesture Recognition’ and ‘Mediapipe’. MediaPipe offers cross-platform, customizable ML solutions for live and streaming media. End-to-End acceleration: Built-in fast ML inference and processing accelerated even on common hardware. Build once, deploy anywhere: Gesture recognition is doneusing a pre trained CNN, A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

Motivated by different real-world applications, researchers have considered a wide range of problems over a variety of different types of corpora. We now examine the key concepts involved in these problems. This discussion also serves as a loose grouping of the major problems, where each group consists of problems that are suitable for similar treatment as learning tasks. One set of problems share the following general character: a food image, where in it is assumed that as an unknown food image, classify the image into respective food recipe and also predict food ingredients present int the food recipe.

2.2 EXISTING SYSTEM

In the existing system methods and algorithms are not that much accurate and requires complex code or physical input which makes the idea of non-physical computer interaction redundant.

PYTHON MODULES:

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python has many modules for automation of tasks but it still requires physical contact with the computer. Modules alone cannot provide a fast, contactless automation tool and need to be utilized in another way.

SELENIUM:

Selenium is an open-source umbrella project for a range of tools and libraries aimed at supporting browser automation. It provides a single interface that lets you write test scripts in programming languages like Ruby, Java, NodeJS, PHP, Perl, Python, and C#, among others. Selenium can be useful for automation of browser related tasks but is limited to it. Like, python modules it needs to be combined for efficient utilization.

OpenCV:

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source Apache 2 License.

Gesture recognition can be performed with a ‘CNN’ model trained on ‘dataset’ and ‘OpenCV’. This method lacks ‘Hand Tracking’ capabilities. Hand tracking can also be performed using ‘Object Detection Algorithms’ like ‘Faster R-CNN’, ‘Single Shot Detector’. This approach requires huge data and GPU for training and deployment.

Figure 2.1: OpenCV

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- Use of python modules requires executing the code for automation. Hence, use of hand gestures might be redundant.
- Use of OpenCV requires complex code to perform hand tracking.
- Accuracy of hand tracking can be low.
- Manual automation using python can be time consuming.

.

2.3 PROPOSED SYSTEM

In the proposed system, we plan on using ‘Mediapipe’. Mediapipe is a cross-platform library developed by Google that provides amazing ready-to-use ML solutions for computer vision tasks. Mediapipe is currently the easiest way to achieve fast “Hand Tracking. With Mediapipe we can also detect and mark, ‘hand landmarks. Along with mediapipe, a trained neural net is used for “Gesture Recognition”. Combining both, we get a “hand tracking” and “Gesture Recognizing” tech. Each gesture can be assigned a task, i.e., opening app, shutting down the system. Using python modules, combined with the above tech, we can perform automation of tasks using gestures.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- In mediapipe Object localization is temporally consistent with the help of tracking, meaning less jitter is observable across frames.
- Mediapipe provides more accurate hand tracking when compared to OpenCV.
- MediaPipe offers cross-platform, customizable ML solutions for live and streaming media.
- Pre-defined and changeable gesture – task mapping.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis.

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

Development of this application is highly economically feasible. The organization needed not spend much money for the development of the system already available. The only thing is to be done is making an environment for the development with an effective supervision. If we are doing so, we can attain the maximum usability of the corresponding resources. Even after the development, the organization will not be in condition to invest more in the organization. Therefore, the system is economically feasible.

2.4.2 TECHNICAL FEASIBILITY

We can strongly say that it is technically feasible, since there will not be much difficulty in getting required resources for the development and maintaining the system as well. All the resources needed for the development of the software as well as the maintenance of the same is available in the organization here we are utilizing the resources which are available already.

2.4.3 BEHAVIORAL FEASIBILITY

Whatever we think need not be feasible. It is wise to think about the feasibility of any problem we undertake. Feasibility is the study of impact, which happens in the organization by the development of a system. The impact can be either positive or negative. When the positives nominate the negatives, then the system is considered feasible. Here the feasibility study can be performed in two ways such as technical feasibility and Economical Feasibility.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

• Processor	:	Minimum Intel i5 @CPU 2.9GHz
• Hard Disk	:	16GB and Above.
• RAM	:	8GB and Above
• Devices	:	HD webcam

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements.

• Operating system	:	Windows 10 and above
• Languages	:	Python
• Framework / Modules	:	CMake, dlib, Mediapipe
• IDE	:	PyCharm

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for automation using gestures with facial recognition as authentication system.

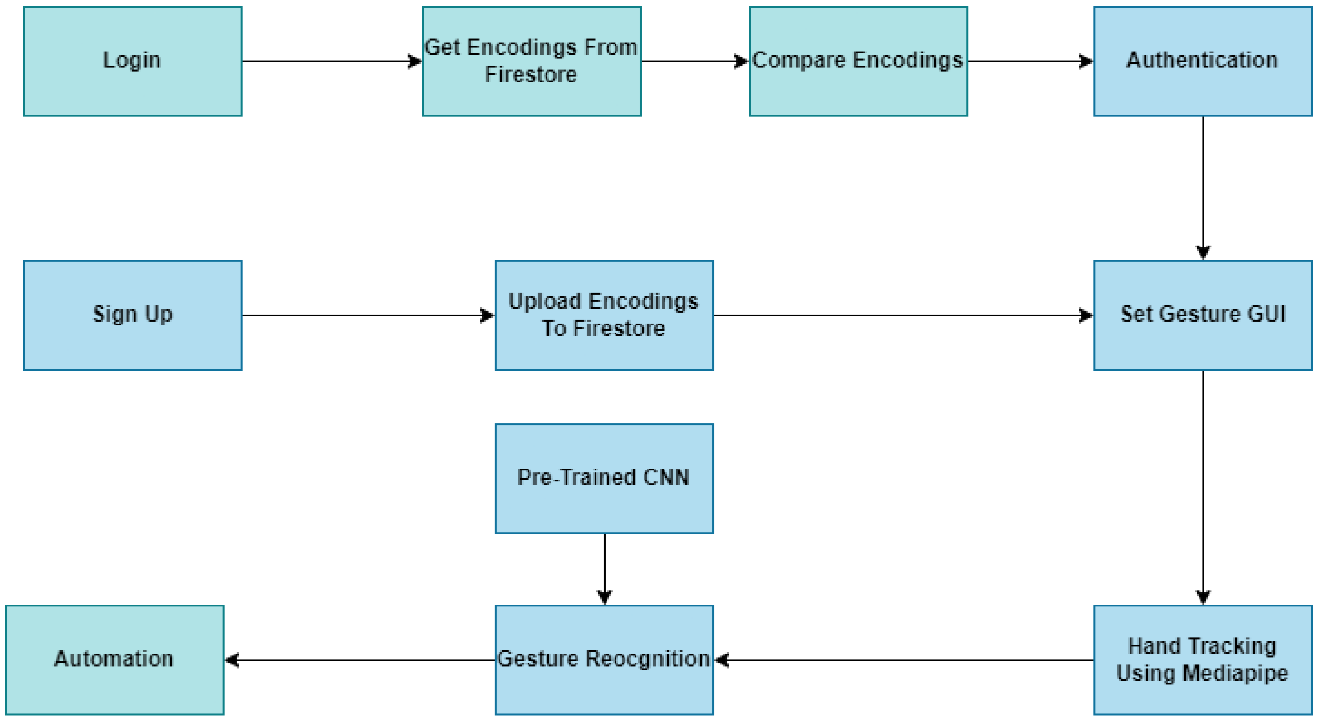


Figure 3.1: Project Architecture

3.2 DESCRIPTION

The project has been classified into four modules (or stages) in a sequential order This modular approach of the project is shown below sequentially.

Module 1: FACIAL AUTHENTICATION / FACIAL ENCODING

- A facial authentication system is a technology capable of matching a human face from a digital image or a video frame against a database of faces.
- A face encoding is basically a way to represent the face using a set of 128 computer-generated measurements. Two different pictures of the same person would have similar encoding and two different people would have totally different encoding.
- Face_recognition is a python module which can recognize and manipulate faces from Python or from the command line and is the world's simplest face recognition library.
- For users who are signing up, a new pic should be uploaded with a clear visibility of face. The module generates a 128 vector of facial key points. This vector or encoding is used to match and compare faces. The same comparison is applied during login.

Module 2: SETTING OF GESTURES:

- A gesture is a movement that you make with a part of your body, especially your hands, to express emotion or information. Gestures are the main concept of this project as they are used to perform automation of tasks.
- A ‘GUI’ window has been developed by using ‘’PyQt5’ python module. It is a Python interface for Qt, one of the most powerful, and popular cross-platform GUI library. PyQt5 is a blend of Python programming language and the Qt library. The user is given option to assign pre-defined tasks to pre-defined gestures. This can be done by both new and authenticated users.

Figure 3.2: PyQt

Module 3: HAND TRACKING USING MEDIAPIPE

- MediaPipe is a Framework for building machine learning pipelines for processing time-series data like video, audio, etc. This cross-platform Framework works in Desktop/Server, Android, iOS, and embedded devices like Raspberry Pi and Jetson Nano.
- MediaPipe Hands is a high-fidelity hand and finger tracking solution. It employs machine learning (ML) to infer 21 3D landmarks of a hand from just a single frame. Whereas current state-of-the-art approaches rely primarily on powerful desktop environments for inference, our method achieves real-time performance on a mobile phone, and even scales to multiple hands.

Palm Detection

- To detect initial hand locations, a single-shot detector model optimized for mobile real-time uses in a manner similar to the face detection model in MediaPipe Face Mesh. Detecting hands is a decidedly complex task: our lite model and full model have to work across a variety of hand sizes with a large scale span (~20x) relative to the image frame and be able to detect occluded and self-occluded hands.

Hand Landmark Detection

- After the palm detection over the whole image our subsequent hand landmark model performs precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.

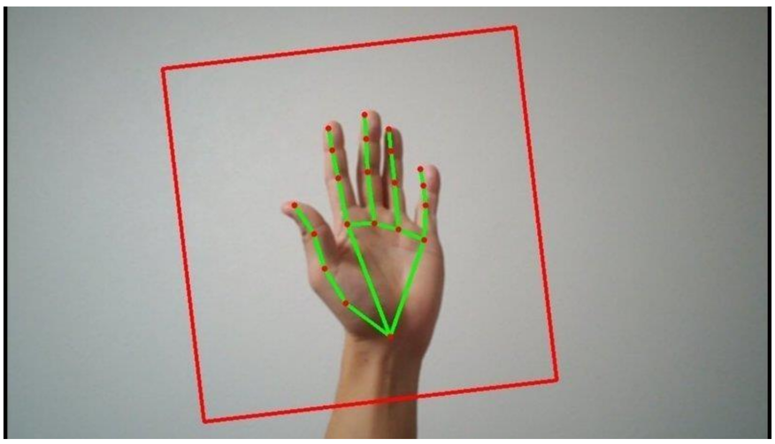
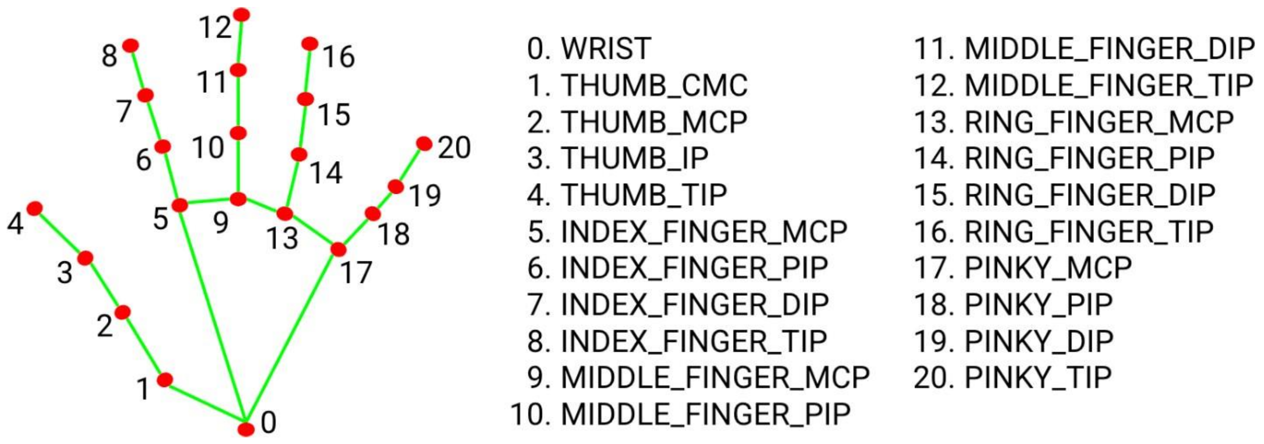


Figure 3.3 Hand tracking Using Mediapipe



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 3.4: Hand Landmarks

Module 4: RECOGNITION OF GESTURES

- Gesture recognition is the fast-growing field in image processing and artificial technology. The gesture recognition is a process in which the gestures or postures of human body parts are identified and are used to control computers and other electronic appliances.
- While user's hand is being tracked, user can perform predefined gestures to automate. A pre-defined 'Convolutional Neural Network' is used for recognition of the gesture.

Pre-Trained CNN

- pre-trained model is a model created by someone else to solve a similar problem. Instead of building a model from scratch to solve a similar problem, you use the model trained on other problem as a starting point.

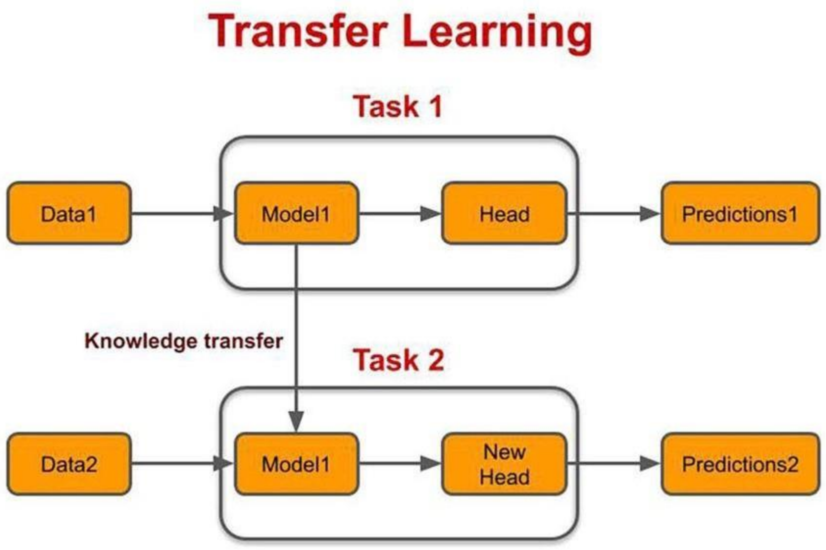


Figure 3.5: Transfer Learning

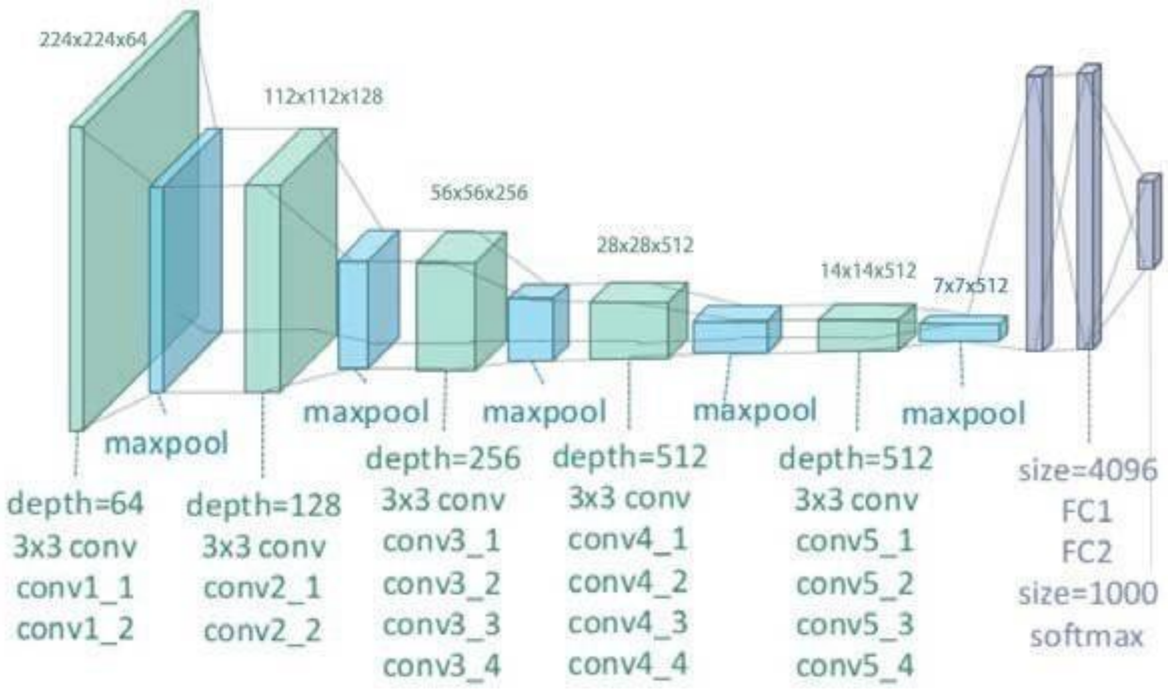


Figure 3.6: CNN Architecture

Module 5: AUTOMATION

- Automation is the use of technology to accomplish a task with as little human interaction as possible. In computing, automation is usually accomplished by a program, a script, or batch processing. For example, a website operator may write a script to parse the logs of the website traffic and generate a report.
- When the gesture is recognized by the model which was pre trained, the associated task will be performed by the computer with utilization of python modules, selenium driver etc.

Figure 3.7: Automation of Tasks

3.3 USE CASE DIAGRAM

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors. In this diagram we have multiple actors interacting with the system.

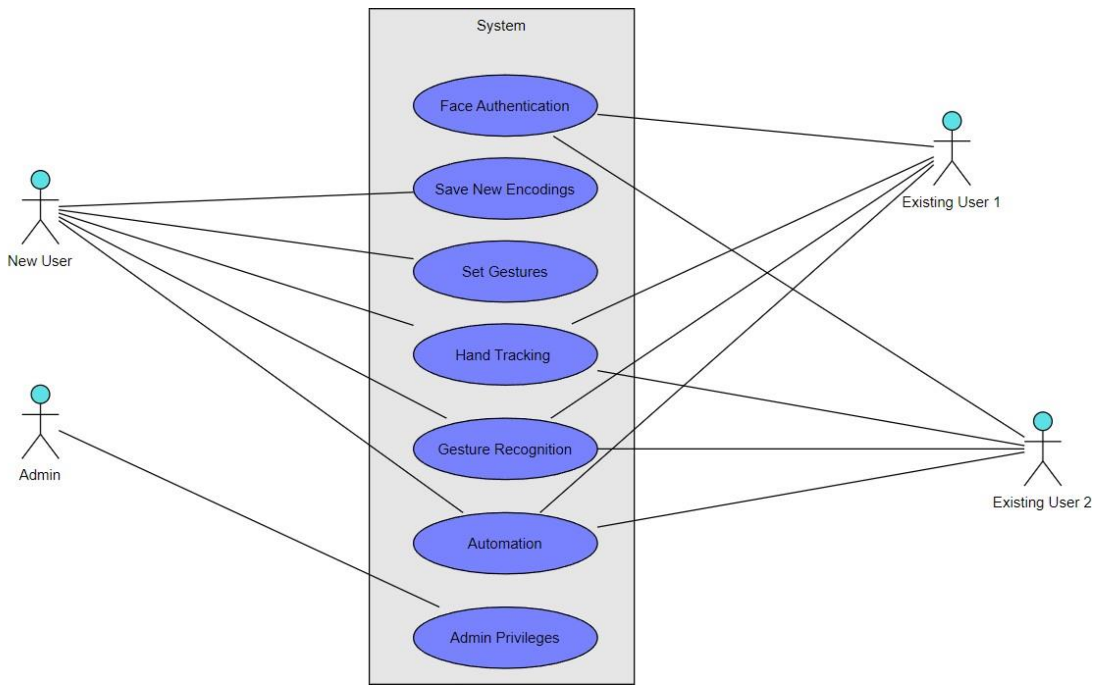


Figure 3.8 : Use Case Diagram

3.4 CLASS DIAGRAM

Class diagrams model social organization and its contents using design elements like classes, packages, and objects, thereby describing various objects used during a system and their relationships.” They define the cognitive, requirement, and functionality paradigms when developing a system by illustrating the classes in the program, attributes, functions of each class, and the relationship that exists between each class.

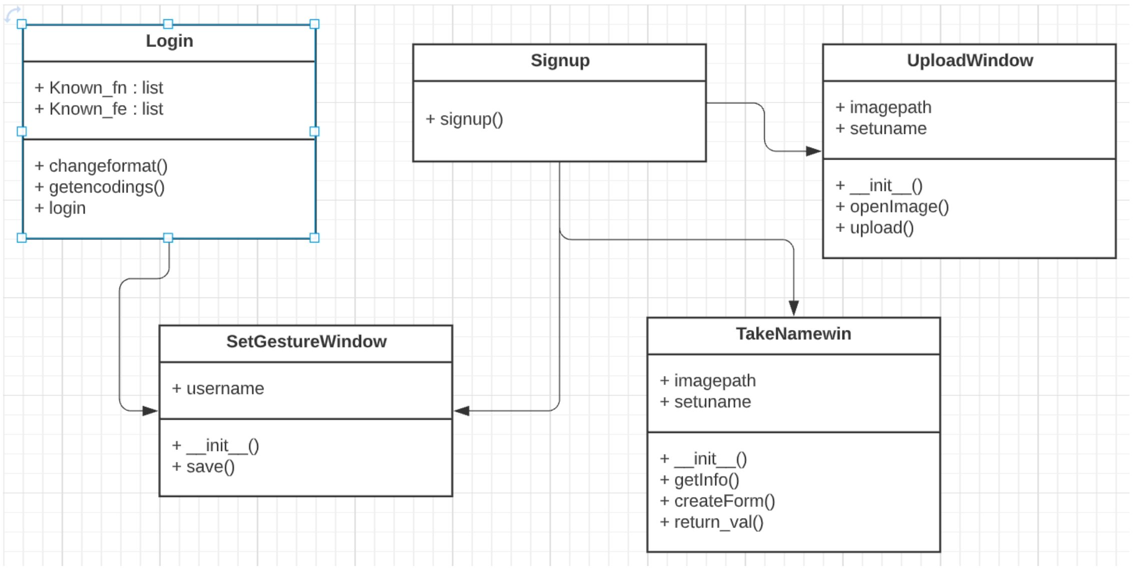


Figure 3.9: Class Diagram

3.5 SEQUENCE DIAGRAM

Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It's important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are show as arrows. This article explains the purpose and the basics of Sequence diagrams.

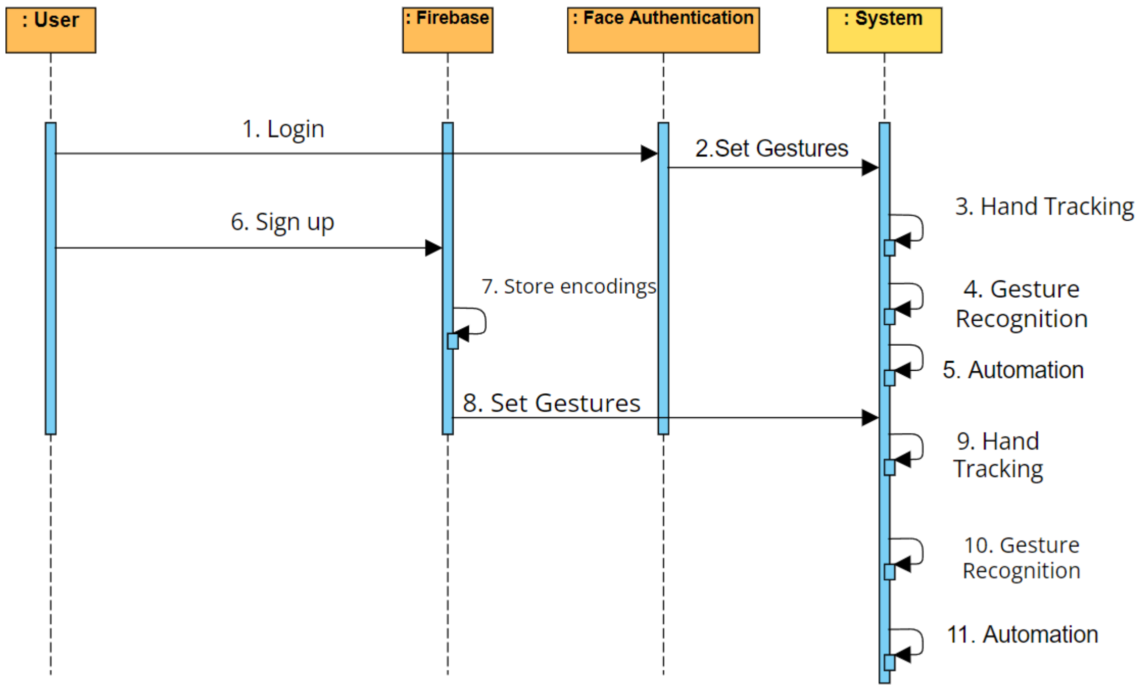


Figure 3.10: Sequence Diagram

3.6 ACTIVITY DIAGRAM

It describes about flow of activity states.

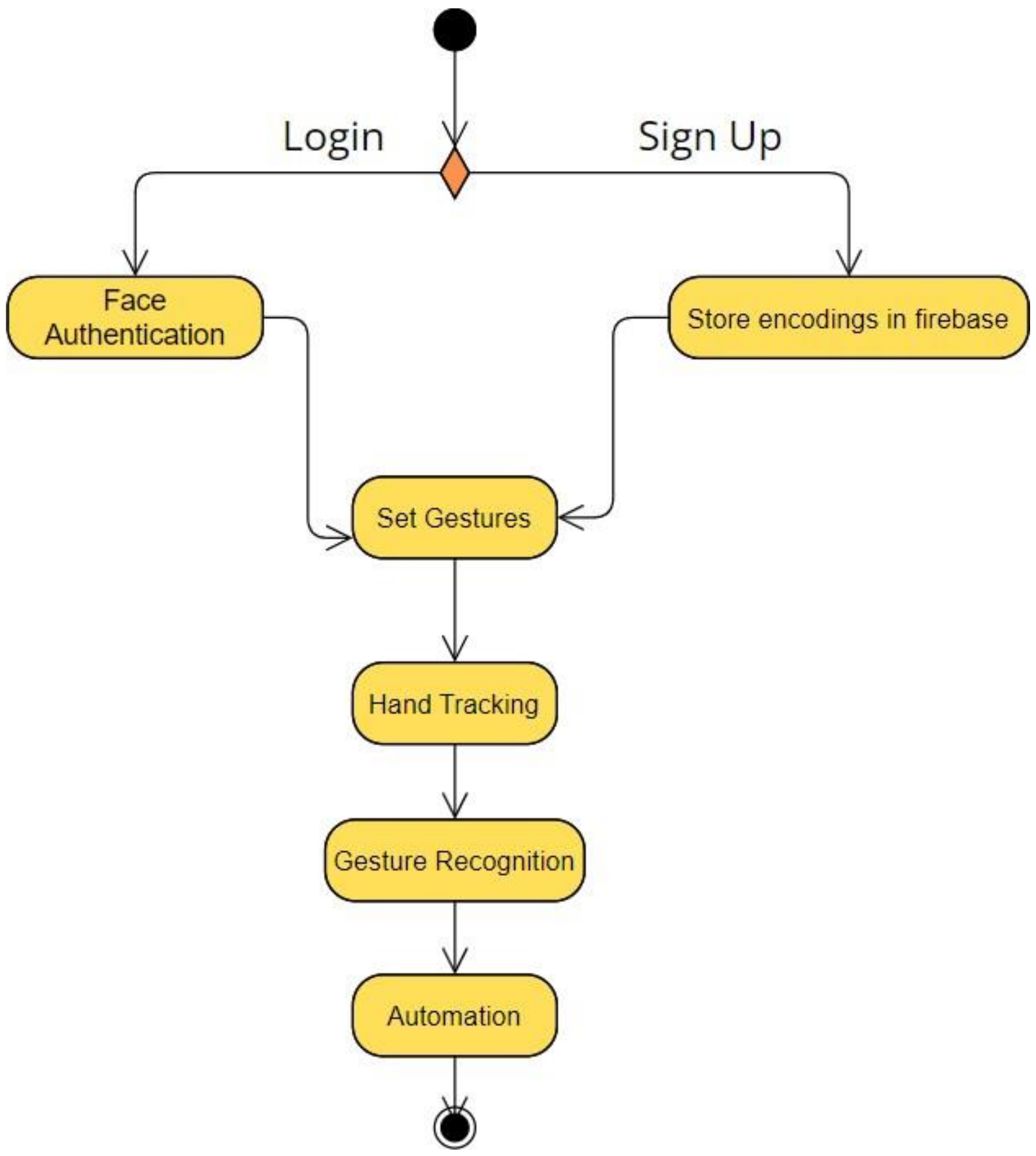


Figure 3.11: Activity Diagram

4. IMPLEMENTATION

4. IMPLEMENTATION

3.1 SAMPLE CODE

Main.py

```
import time
import sys
import face_recognition
import cv2
from firebase_admin import credentials,initialize_app,firestore
import firebase_admin
import numpy as np
from PyQt5.QtWidgets import  QApplication

import upimg
import gestset
import automate


#getting custom gestures stored in firebase
def  getgest(uname):
   doc_ref = db.collection(u'Gests').document(uname)
    doc = doc_ref.get()
    tempd=doc.to_dict()
    myl=[]
    myl.append(tempd['Thumbs up'])
    myl.append(tempd['Thumbs Down'])
    myl.append(tempd['Stop'])
    myl.append(tempd['Rock'])
return myl


class Signup():
   def signup(self):
       app = QApplication(sys.argv)
        window = upimg.TakeNamewin()
        window.show()
        app.exec()
        val = window.return_val()
```

```
 app.quit()
 newapp = QApplication(sys.argv)
 newwin = upimg.UploadWindow(val)
 newwin.show()
 newapp.exec_()
 newapp.quit()
 gestapp = QApplication(sys.argv)
 gestwin = gestset.SetGesturewindow(val)
 gestwin.show()
 gestapp.exec_()
  gestapp.quit()
 #####get custom gestures
 myl = getgest(val)
 automate.startnewcap(myl)



class Login():
    known_fe = []
    known_fn = []

    def changeformat(self,value):
       newl = []
        li = list(value.split(" "))
        li[0] = li[0].replace("[", "")
        li[-1] = li[-1].replace("]", "")
                for i in li:
            if i.strip():
                i = float(i)
                newl.append(i)
        final = np.array(newl)
        return final

     def getencodings(self):
        users_ref = db.collection(u'localdb')
        docs = users_ref.stream()
        try:
            for doc in docs:
                self.known_fn.append(doc.id)
                d = doc.to_dict()
                final = self.changeformat(d['value'])
                self. known_fe.append(final)
```

```
def login(self):
        print("here1")

        self.getencodings()
        print('here2')
        if len(self.known_fe) == 0:
            print("no users")
        else:

           video_capture = cv2.VideoCapture(0)

            patience = 0

            gotit = 0

            username = ""

            # Initialize some variables
            face_locations = []
            face_encodings = []
            face_names = []
            process_this_frame = True

            while True:
               # Grab a single frame of video
                ret, frame = video_capture.read()

                # Resize frame of video to 1/4 size for faster face recognition processing
                small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

        # Convert the image from BGR color (which OpenCV uses) to RGB color (which
       face_recognition uses)
                rgb_small_frame = small_frame[:, :, ::-1]

                if process_this_frame:
                  # Find all the faces and face encodings in the current frame of video
                  face_locations = face_recognition.face_locations(rgb_small_frame)

                    face_encodings = face_recognition.face_encodings(rgb_small_frame,
       face_locations)

                    face_names = []
                    for face_encoding in face_encodings:
                        # See if the face is a match for the known face(s)
# Or instead, use the known face with the smallest distance to the new face face_distances =

face_recognition.face_distance(self.known_fe, face_encoding)
best_match_index = np.argmin(face_distances)if
matches[best_match_index]:
name = self.known_fn[best_match_index]

face_names.append(name)if len(face_names) ==
0:
passelse:
if face_names[0] == "No Match":patience = patience + 1
if (patience == 15):
print("did not find a match")break

else:
username = face_names[0]gotit = 1
                cv2.putText(frame, "press 'r' to sign up", (10,
                20),cv2.FONT_HERSHEY_DUPLEX, 1.0, (255, 255,
                255), 1)

process_this_frame = not process_this_frame# Display the

results
for (top, right, bottom, left), name in zip(face_locations, face_names):
# Scale back up face locations since the frame we detected in was scaled to
1/4 size
                         top *= 4
                         right *= 4
                         bottom *= 4
                         left *= 4
                         # Draw a box around the face
                         cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

# Draw a label with a name below the face
cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255),cv2.FILLED)
cv2.putText(frame, name, (left + 6, bottom - 6),
# Display the resulting image
cv2.imshow('Video', frame)

if (gotit == 1): time.sleep(1)
video_capture.release()
cv2.destroyAllWindows() myl =
getgest(username)
automate.startnewcap(myl)

key = cv2.waitKey(1)if key ==
ord('q'):
        break
elif key == ord('r'):

video_capture.release()
cv2.destroyAllWindows()obj = Signup()
obj.signup()
break

video_capture.release()
cv2.destroyAllWindows()

loginobj=Login()
loginobj.login()
```

Upimg.py

```
import sys
from PyQt5.QtWidgets import  QApplication, QLabel, QFileDialog, QAction,QLineEdit
from PyQt5.QtWidgets import *
from PyQt5.QtGui import QPixmap
import face_recognition
import firebase_admin
from firebase_admin import credentials,initialize_app,firestore
import cv2
import numpy as np




username=""
```

```
class UploadWindow(QMainWindow):

  myimagepath=""

  def___init_(self, username,parent = None):
    super(UploadWindow, self)._init_(parent)
    self.setuname=username



    menubar = self.menuBar()
    fileMenu  =  menubar.addMenu('File')
    editMenu = menubar.addMenu('Edit')
    upload=menubar.addMenu('Upload ')
    self.resize(500, 500)

    openAction = QAction('Open Image', self)
    openAction.triggered.connect(self.openImage)
    fileMenu.addAction(openAction)

    uploadaction=QAction('upload to mongo',self)
    uploadaction.triggered.connect(self.upload)
    upload.addAction(uploadaction)

    closeAction = QAction('Exit', self)
    closeAction.triggered.connect(self.close)
    fileMenu.addAction(closeAction)
    self.label = QLabel()
    self.setCentralWidget(self.label


  def openImage(self):
    imagePath, _ = QFileDialog.getOpenFileName()
    pixmap = QPixmap(imagePath)
    self.label.setPixmap(pixmap)
    self.resize(pixmap.size())
    self.adjustSize()
    self.myimagepath=imagePath
    def upload(self):
    self.tempimg=face_recognition.load_image_file(self.myimagepath)
    self.enc=face_recognition.face_encodings(self.tempimg)[0]
    self.arrep=np.array_str(self.enc)
    self.db.collection(u'localdb').document(self.setuname).set({
            'value':self.arrep,
        })



class TakeNamewin(QDialog):


    # constructor
    def___init__(self):
        super(TakeNamewin, self)._init_()

        # setting window title
        self.setWindowTitle("Python")

        # setting geometry to the window
        self.setGeometry(100, 100, 300, 400)

        # creating a group box
        self.formGroupBox = QGroupBox("name Form")

        # creating spin box to select age

        # creating a line edit



        self.nameLineEdit = QLineEdit()

        # calling the method that create the form
        self.createForm()

        # creating a dialog button for ok and cancel
        self.buttonBox = QDialogButtonBox(QDialogButtonBox.Ok |
       QDialogButtonBox.Cancel)

        # adding action when form is accepted
        self.buttonBox.accepted.connect(self.getInfo)
        # adding action when form is rejected
        self.buttonBox.rejected.connect(self.reject)

        # creating a vertical layout
        mainLayout = QVBoxLayout()

        # adding form group box to the layout
        mainLayout.addWidget(self.formGroupBox)

        # adding button box to the layout
        mainLayout.addWidget(self.buttonBox)

        # setting lay out
        self.setLayout(mainLayout)

    # get info method called when form is accepted
    def getInfo(self):
        # closing the window
        self.close()

    # create form method
    def createForm(self):
        # creating a form layout
        layout = QFormLayout()

        # adding rows
        # for name and adding input text
        layout.addRow(QLabel("Name"), self.nameLineEdit)

        # setting layout
        self.formGroupBox.setLayout(layout)

    def return_val(self):
        return self.nameLineEdit.text()
```

Gestset.py

```
import firebase_admin.firestore
from PyQt5.QtWidgets import (
    QApplication,
    QPushButton,
    QVBoxLayout,
    QWidget,
    QLabel,
    QComboBox
)
```

```
from PyQt5.QtGui import QFont



class SetGesturewindow(QWidget):


   def___init_(self,username):
      self.db=firebase_admin.firestore.client()

      self.uname=username
      super()._init_()
      self.setWindowTitle("Gesture window")
      self.resize(850, 400)
      # Create a QVBoxLayout instance
      #welcome label
      infolab = QLabel(self)
      infolab.setFont(QFont('Ariel', 20))
      infolab.move(400, 2)
      infolab.setText("Hello"+" "+self.uname)
      infolab.show()

      infolab=QLabel(self)
      infolab.setFont(QFont('Ariel',10))
      infolab.move(20,40)
      infolab.setText("Available Gestures")
      infolab.show()




      #Thumbsuplabel
      tuplab=QLabel(self)
      tuplab.move(20,80)
      tuplab.setText("Thumbs Up")
      tuplab.show()


      # ThumbsdownLabel
      tdlab = QLabel(self)
      tdlab.move(20,140)
      tdlab.setText("Thumbs Down")
      tdlab.show()

      # FistLabel
      stlab = QLabel(self)
      stlab.move(20, 200)
      stlab.setText("Fist")
      stlab.show()
```

```
 # Rock OnLabel
 rlab = QLabel(self)
 rlab.move(20, 260)
 rlab.setText("Rock On ")
 rlab.show()

 #combox
 self.cb1=QComboBox(self)
 self.cb1.move(150,80)
 self.cb1.addItems(["Open Notepad","Open Chrome","Open cmd","Turn on sleep
mode"])
 self.cb1.show()
 self.cb2 = QComboBox(self)
 self.cb2.move(150, 140)
 self.cb2.addItems(["Open Notepad", "Open Chrome","Open cmd","Turn on sleep
mode"])
 self.cb2.show()
 self.cb3 = QComboBox(self)
 self.cb3.move(150, 200)
 self.cb3.addItems(["Open Notepad", "Open Chrome","Open cmd","Turn on sleep
mode"])
 self.cb3.show()
 self.cb4 = QComboBox(self)
 self.cb4.move(150, 260)
 self.cb4.addItems(["Open Notepad", "Open Chrome","Open cmd","Turn on sleep
mode"])
 self.cb4.show()




 self.bu=QPushButton(self)
 self.bu.show()
 self.bu.move(130,300)
 self.bu.setText("Save")
 self.bu.clicked.connect(self.save)
```

```
    def save(self):
        self.tupt=self.cb1.currentText()
        self.tdt=self.cb2.currentText()
        self.st=self.cb3.currentText()
        self.rt=self.cb4.currentText()
        self.db.collection(u'Gests').document(self.uname).set(

            {
                'Thumbs up':self.tupt,
                'Thumbs Down':self.tdt,
                'Stop':self.st,
                'Rock':self.rt
            }
        )




if__name__== "__main__":
    app = QApplication(sys.argv)
    window = SetGesturewindow()
    window.show()
    sys.exit(app.exec_())
```

Automate.py

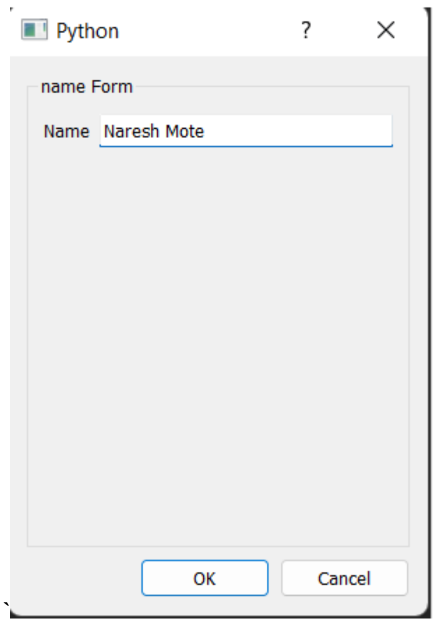
```
import cv2
import numpy as np
import mediapipe as mp
import tensorflow as tf
import time
import os
import webbrowser
import sel_code
```

```
# initialize mediapipe
mpHands = mp.solutions.hands
hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.7)
mpDraw = mp.solutions.drawing_utils

# Load the gesture recognizer model
model =
    tf.keras.models.load_model(r'C:\Users\Aditya\Desktop\Myproject\mets\mp_hand_ges
    ture')
# Load class names
f = open(r'C:\Users\Aditya\Desktop\Myproject\mets\gesture.names', 'r')
classNames = f.read().split('\n')
f.close()


def change_gest():
  pass

def startnewcap(myl):
  tupval=''
  tdownval=''
  fstval=''
  ronval=''
  for i in myl:
     if "Chrome" in i:
       indval = myl.index(i)
       if indval == 0:
          tupval="os.startfile(r'C:\Program
    Files/Google/Chrome/Application/chrome.exe')"
       if indval==1:
          tdownval="os.startfile(r'C:\Program
    Files/Google/Chrome/Application/chrome.exe')"
       if indval==2:
         fstval="os.startfile(r'C:\Program Files/Google/Chrome/Application/chrome.exe')"
       if indval==3:
          ronval="os.startfile(r'C:\Program
    Files/Google/Chrome/Application/chrome.exe')"
    elif "cmd" in i:
       indval = myl.index(i)
       if indval == 0:
```

```
                tupval = "os.system(“start cmd”) "
            if indval == 1:
                tdownval = "os.system(“start cmd”) "
            if indval == 2:
               fstval = "os.system(“start cmd”) "
            if indval == 3:
                ronval = "os.system(“start cmd”) "
        elif "Notepad" in i:
            indval = myl.index(i)
            if indval == 0:
                tupval = "os.system('notepad')"
            if indval == 1:
                tdownval = "os.system('notepad')"
            if indval == 2:
               fstval = "os.system('notepad')"
            if indval == 3:
                ronval = "os.system('notepad')"
        elif "sleep" in i:
            indval = myl.index(i)
            if indval == 0:
                tupval = "os.system('rundll32.exe powrprof.dll,SetSuspendState 0,1,0 ')"
            if indval == 1:
                tdownval = "os.system('rundll32.exe powrprof.dll,SetSuspendState 0,1,0 ')"
            if indval == 2:
               fstval = "os.system('rundll32.exe powrprof.dll,SetSuspendState 0,1,0 ')"
            if indval == 3:
                ronval = "os.system('rundll32.exe powrprof.dll,SetSuspendState 0,1,0 ')"


newcap = cv2.VideoCapture(0)
    while True:
        # Read each frame from the webcam
        _, frame = newcap.read()

        x, y, c = frame.shape

        # Flip the frame vertically
        frame = cv2.flip(frame, 1)
        framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Get hand landmark prediction
        result = hands.process(framergb)
        # print(result)
        className = ''

        # post process the result
        if result.multi_hand_landmarks:
            landmarks = []
            for handslms in result.multi_hand_landmarks:
                for lm in handslms.landmark:
                    # print(id, lm)
                    lmx = int(lm.x * x)
                    lmy = int(lm.y * y)

                    landmarks.append([lmx, lmy])

                # Drawing landmarks on frames
                mpDraw.draw_landmarks(frame, handslms,
       mpHands.HAND_CONNECTIONS)

                # Predict gesture
                prediction = model.predict([landmarks])

                classID = np.argmax(prediction)
                className = classNames[classID]

        # show the prediction on the frame
        cv2.putText(frame, className, (10, 50), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 0, 255), 2, cv2.LINE_AA)

        if className == "thumbs up":
            exec(tupval)
        if className == "stop":
            exec(stval)
        if className== "rock":
            exec(rval)
        if className=="thumbs down":
            exec(tdownval)

        # Show the final output
        cv2.imshow("Output", frame)

        if cv2.waitKey(1) == ord('q'):
            break
        if cv2.waitKey(1) == ord('c'):
            pass
    newcap.release()

    cv2.destroyAllWindows()
```

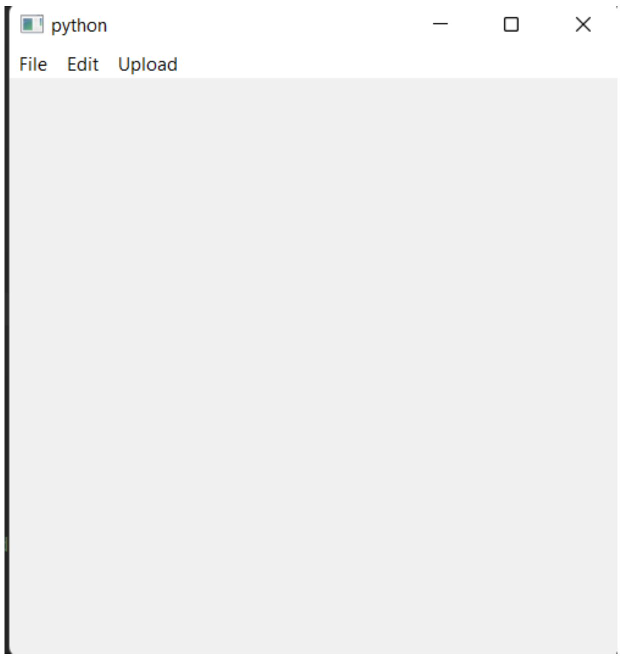
5. SCREENSHOTS

5. SCREENSHOTS

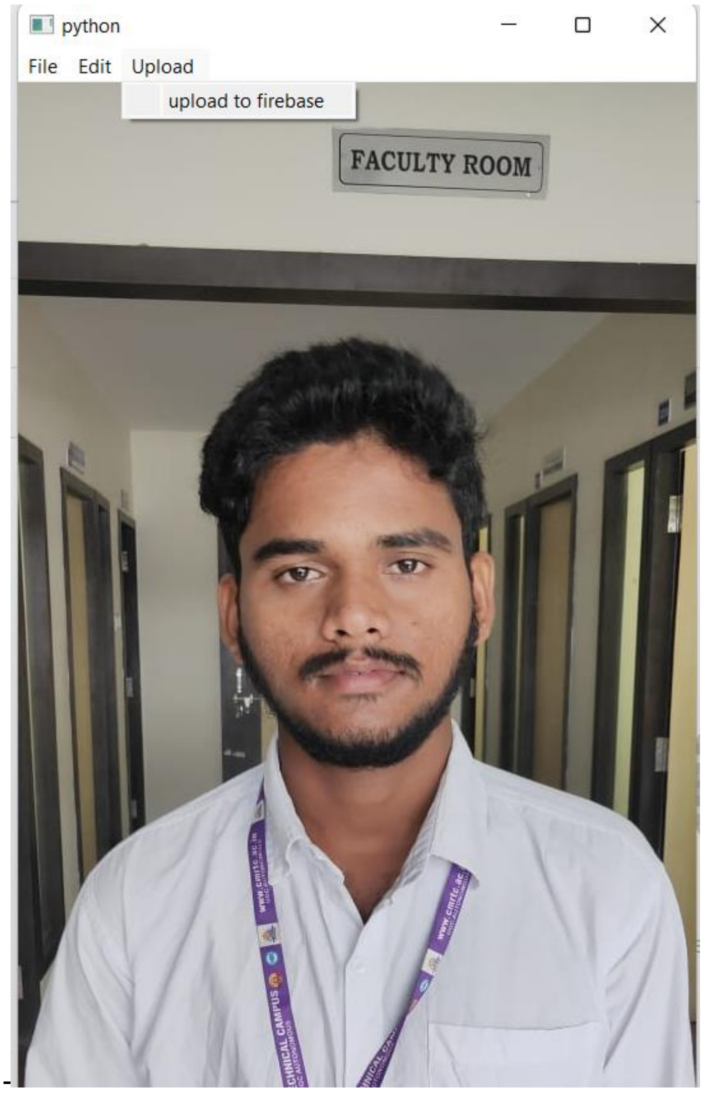


Name: Naresh Mote

Screenshot 5.1 Upload Name GUI

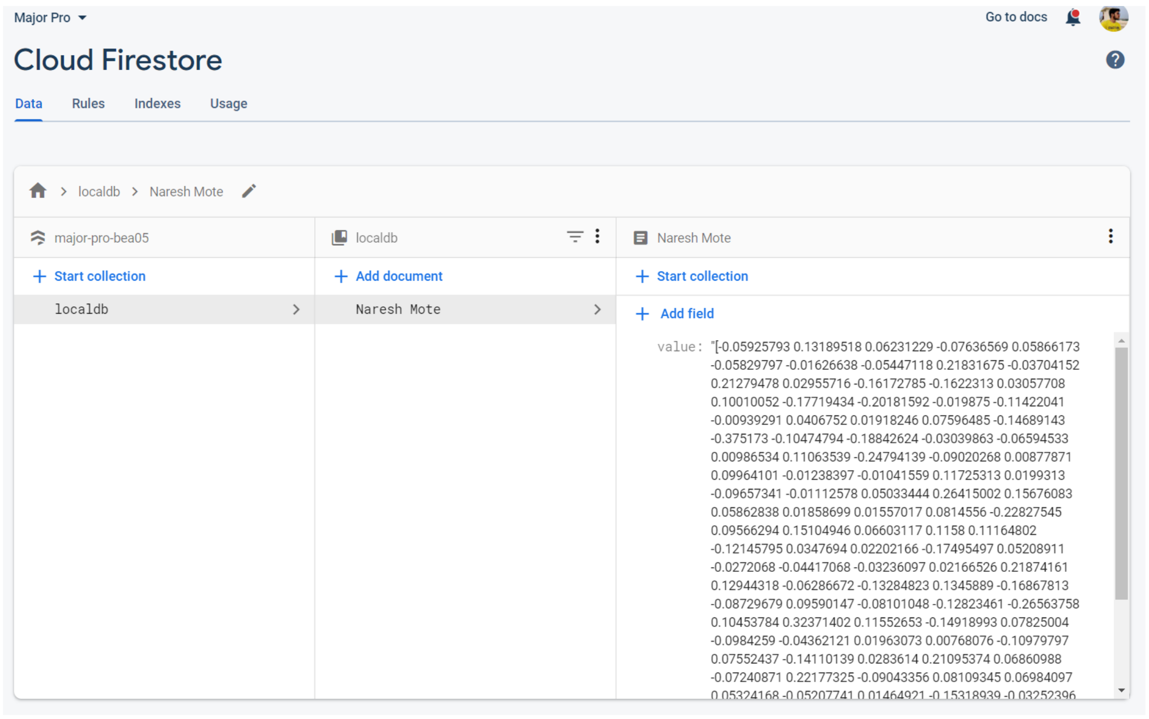


Screenshot 5.2 Upload Image GUI



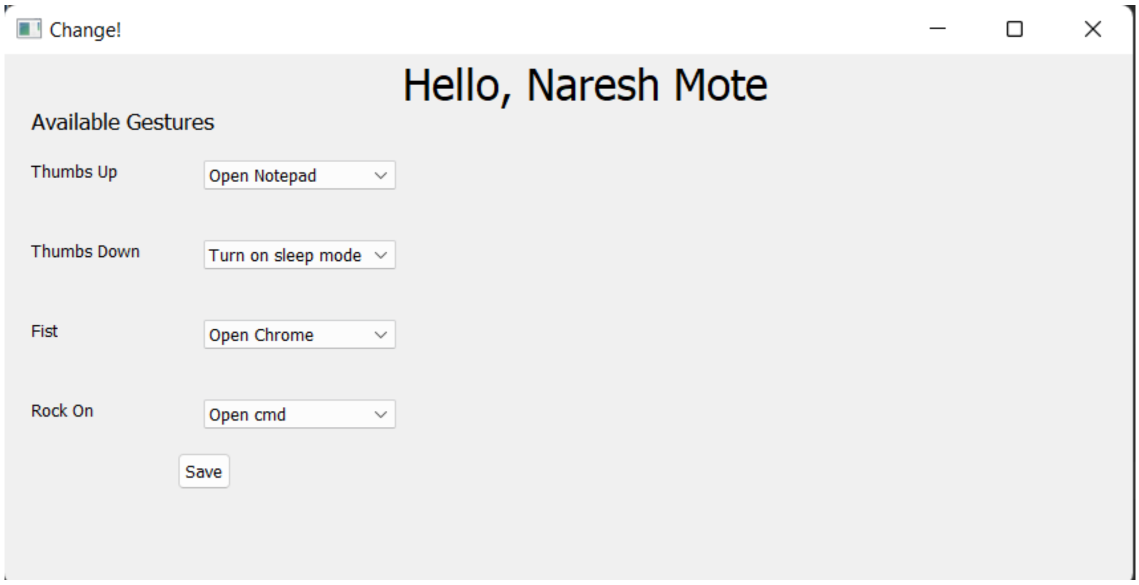
-

Screenshot 5.3 Upload Image to Firebase



```
value: "[-0.05925793 0.13189518 0.06231229 -0.07636569 0.05866173
-0.05829797 -0.01626638 -0.05447118 0.21831675 -0.03704152
0.21279478 0.02955716 -0.16172785 -0.1622313 0.03057708
0.10010052 -0.17719434 -0.20181592 -0.019875 -0.11422041
-0.00939291 0.0406752 0.01918246 0.07596485 -0.14689143
-0.375173 -0.10474794 -0.18842624 -0.03039863 -0.06594533
0.00986534 0.11063539 -0.24794139 -0.09020268 0.00877871
0.09964101 -0.01238397 -0.01041559 0.11725313 0.0199313
-0.09657341 -0.01112578 0.05033444 0.26415002 0.15676083
0.05862838 0.01858699 0.01557017 0.0814556 -0.22827545
0.09566294 0.15104946 0.06603117 0.1158 0.11164802
-0.12145795 0.0347694 0.02202166 -0.17495497 0.05208911
-0.0272068 -0.04417068 -0.03236097 0.02166526 0.21874161
0.12944318 -0.06286672 -0.13284823 0.1345889 -0.16867813
-0.08729679 0.09590147 -0.08101048 -0.12823461 -0.26563758
0.10453784 0.32371402 0.11552653 -0.14918993 0.07825004
-0.0984259 -0.04362121 0.01963073 0.00768076 -0.10979797
0.07552437 -0.14110139 0.0283614 0.21095374 0.06860988
-0.07240871 0.22177325 -0.09043356 0.08109345 0.06984097
0.05324168 -0.05207741 0.01464921 -0.15318939 -0.03252396
```

Screenshot 5.4 Facial Encodings stored

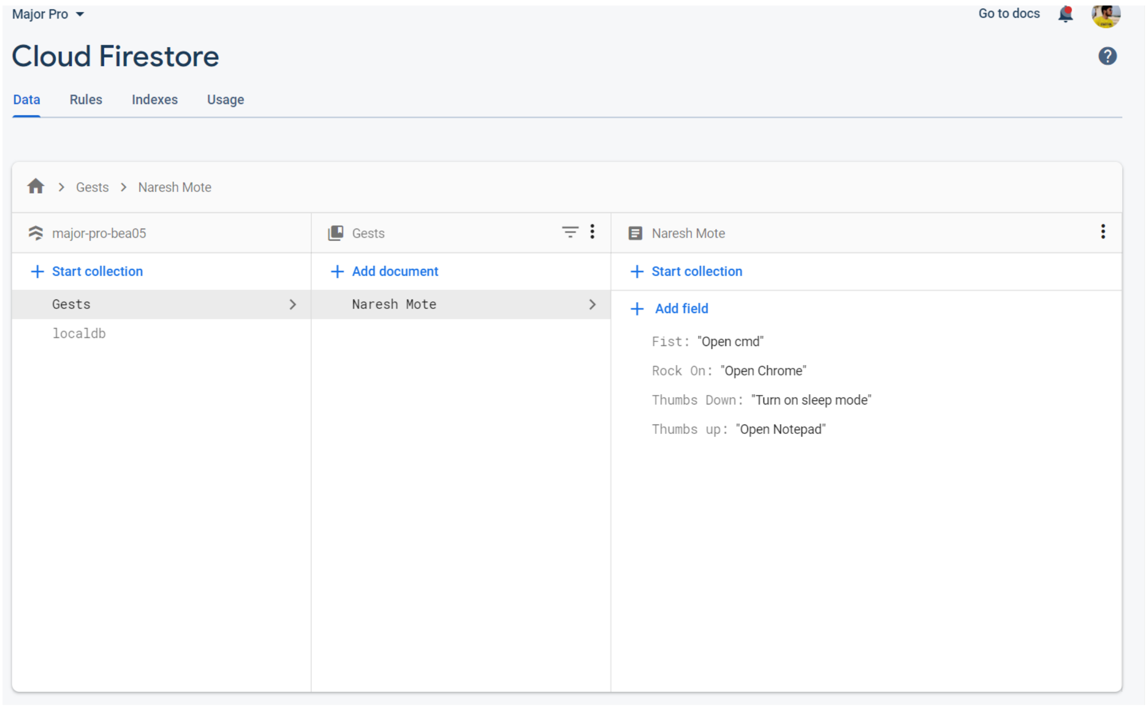


Hello, Naresh Mote

Available Gestures

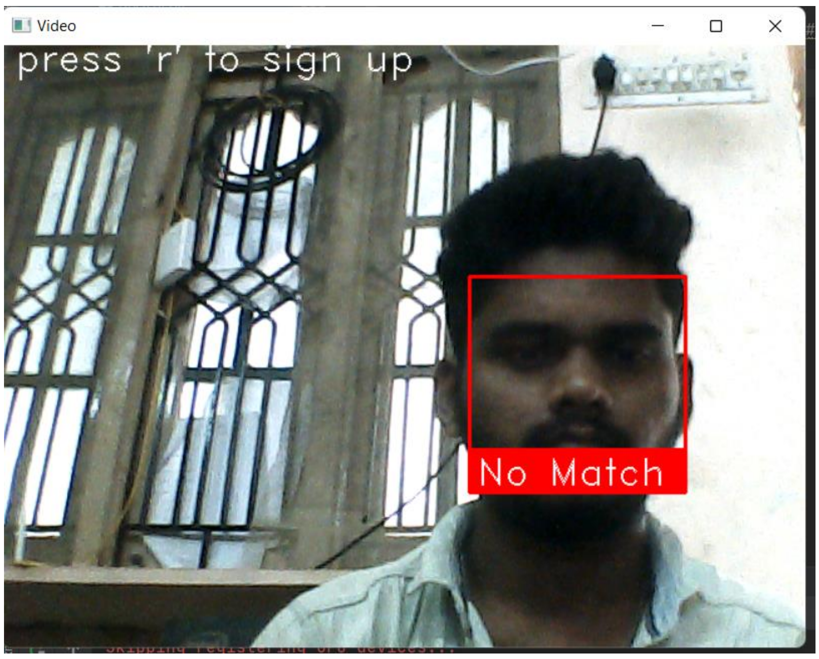
Thumbs Up	Open Notepad
Thumbs Down	Turn on sleep mode
Fist	Open Chrome
Rock On	Open cmd

Screenshot 5.5 Gesture GUI

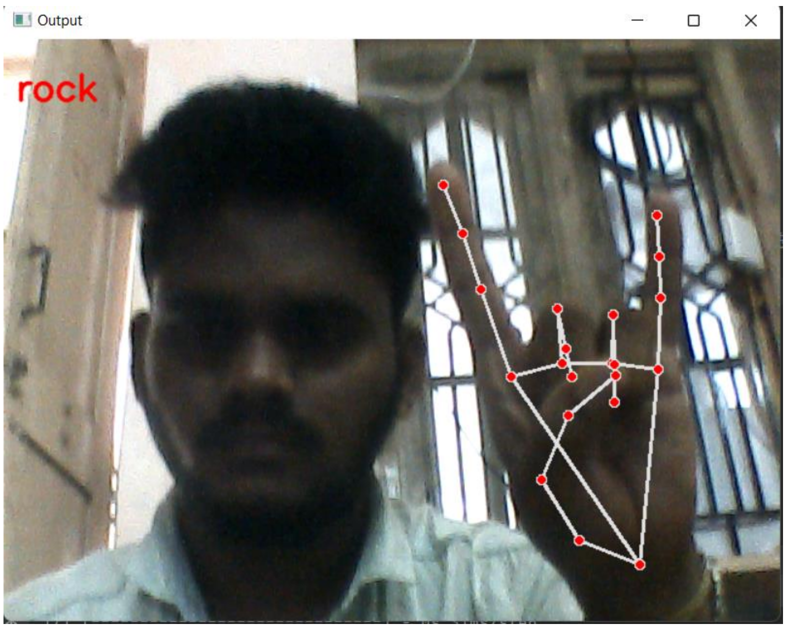


Fist: "Open cmd"
Rock On: "Open Chrome"
Thumbs Down: "Turn on sleep mode"
Thumbs up: "Open Notepad"

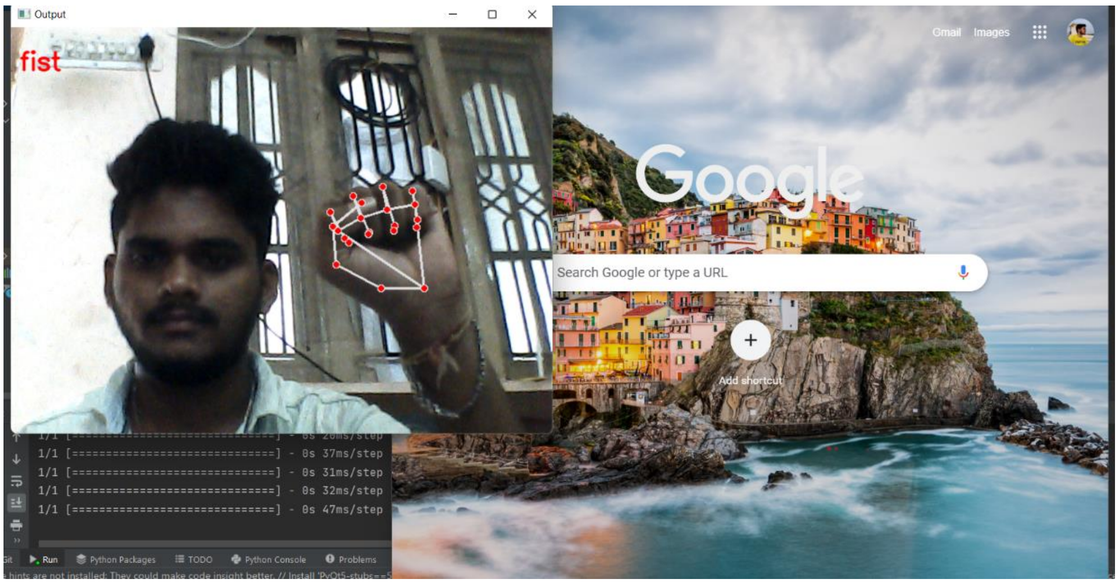
Screenshot 5.6 Gesture Data stored



Screenshot 5.7 Facial Match Not Found



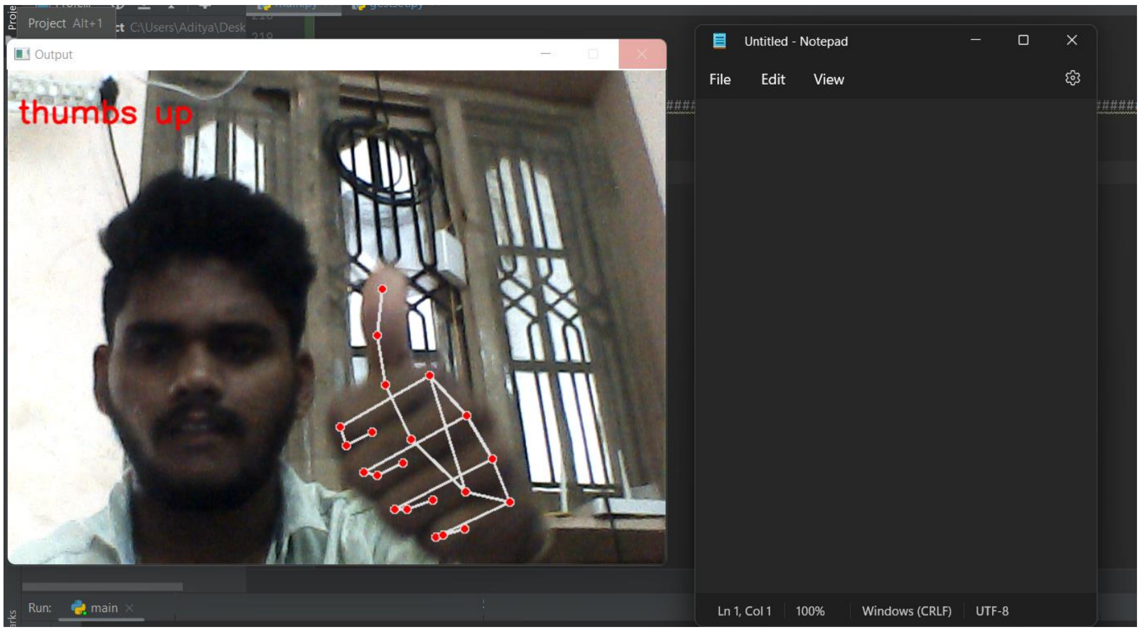
Screenshot 5.8: Rock On Gesture



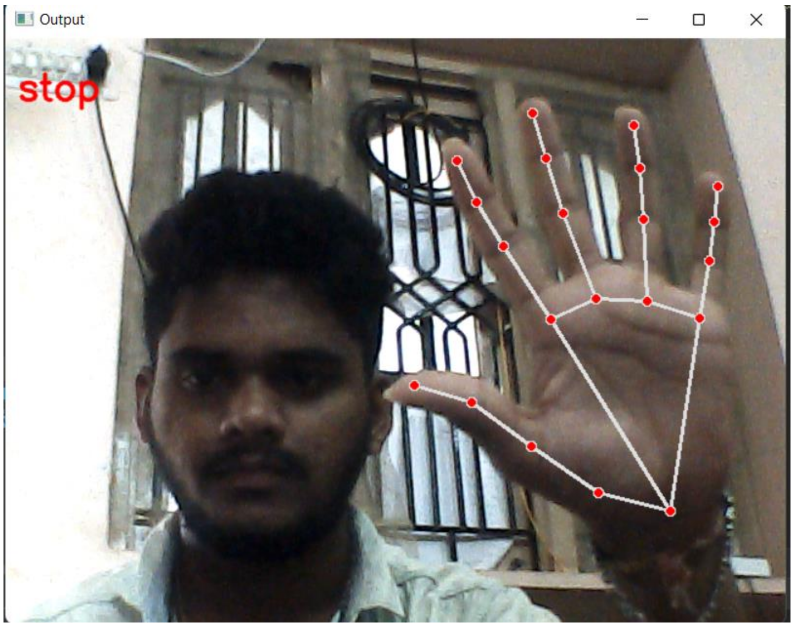
```
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 47ms/step
```

```
e hints are not installed: They could make code insight better. // Install 'PyQt5-stubs==5
```

Screenshot 5.9   Fist Gesture



Screenshot 5.10: Thumbs Up Gesture



Screenshot 5.11   Stop Gesture

6. TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specifictesting requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent.

Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONALTESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.3 TESTCASES

6.3.1 UPLOADING DATASET

Test case ID	Test case name	Purpose	Test Case	Output
1	User uploads image	Use it for authentic ation	The user uploads a clear image of his/her face	Uploaded successfully
2	User uploads gestures	Use it for recognitio n	The user tries to upload gestures	Uploaded successfully

6.3.2 TEST CASES

Test case ID	Test case name	Purpose	Expected Results	Actual Result	Result
1	Check correct facial Authentication	Verify wrong face authentication against stored encodings	Authentication Error	Authentication Denied	Positive
2	Hand Gesture Check	Check integrity of gesture recognition	Gesture is recognized	Gesture is recognized	Positive
3	Check wrong facial authentication	Check negative case of authentication	Access denied	Access granted	negative
4	Check wrong gesture	To check the case for wrong task	Correct task is performed	Wrong task is performed	negative

7. CONCLUSION

7 .CONCLUSION & FUTURESCOPE

7.1 PROJECT CONCLUSION

In the present thesis, we have presented a software capable of automating tasks using gesture recognition. This is a project that incorporates several technologies to create an efficient software. We identified that numerous technologies/techniques could be accompanied in this process namely:

- Mediapipe
- Convolutional Neural Network
- Facial Authentication
- Gesture Recognition

Mediapipe stands out in this project providing seamless hand tracking with accurate landmarks, the proposed project provides authentication and interfaces for the users which enhances the user experience. The data stored in firebase which is a cloud service provided by google utilizes NoSQL for storing the facial encodings.

We designed the project so that it can be easily used and deployed in multiple sectors. The existing system requires improvement and is not really efficient at performing hand tracking and automation. By harnessing the power of mentioned technologies, wecan perform automation of computer tasks.

The constraints are met and overcome successfully. The system is designed as like it was decided in the design phase. The project gives good idea on developing a full-fledged application satisfying the user requirements.

The system is very flexible and versatile. Validation checks induced have greatly reduced errors. Provisions have been made to upgrade the software. The application has been tested with live data and has provided a successful result.

7.2 PROJECT FUTURESCOPE

The future scope for this project would be boundless with changing times and has a much broader scope for enhancement and addition of auxiliary features. Some of which are listed below:

- Storage of data can be improved by migrating from online mode to offline mode. Although firebase is reliable, network issues might cause inaccessibility.

- The tasks which can be performed and the type of gestures can be increased. More tasks can completely change the way user interact with the computer.

- This project can be implemented or paired with other devices like Mobiles or IOT. This same idea can also be implemented for different operating systems.

- The method and accuracy of face authentication can be drastically improved, this provides better security and efficiency.

- User experience can be improved with better themed and solid looking GUI and with better approaches.

8. BIBILOGRAPHY

8. BIBILOGRAPHY

8.1 REFERENCES

[1] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, Matthias Grundmann:
MediaPipe: A Framework for Building Perception Pipelines

[2] Rafiqul Zaman Khan and Noor Adnan Ibraheem HAND GESTURE RECOGNITION: A LITERATURE REVIEW. International Journal of Artificial Intelligence & Applications (IJAIA), Vol.3, No.4, July 2012Jing-Jing Chen and Chong-Wah Ngo. Deep-based ingredient recognition for cooking recipe retrieval. In ACM Multimedia. ACM, 2016.

[3] Summerfield, Mark: Rapid GUI programming with Python and Qt: the definitive guide to PyQt programming

[4] Prabhu Ramachandran∗ automan: a simple, Python-based, automation framework for numerical computing

8.2 WEBSITES

1. https://google.github.io/mediapipe/
2. https://pypi.org/project/face-recognition/
3. https://www.riverbankcomputing.com/static/Docs/PyQt5/
4. https://firebase.google.com/docs/firestore
5. https://www.optisolbusiness.com/insight/alphabet-hand-gestures-recognition-using-media-pipe
6. https://docs.opencv.org/4.x/