# VIRTUALMOUSE

*A*

*Mini Project Report Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

## BACHELOR OF ENGINEERING

IN

## INFORMATION TECHNOLOGY

By

## <HARSHITH PAGINDLA ><1602-20-737-016>

**<MD ASADUDDIN><1602-20-737-024>**

## <SHREYA CHEPURI><1602-20-737-040>



**Department of Information Technology Vasavi College of Engineering (Autonomous)**

## VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)

### (AFFILIATED TO OSMANIA UNIVERSITY) HYDERABAD - 500 030

**Department of Information Technology**



### DECLARATION BY CANDIDATE

We, **<HARSHITH PAGINDLA>, <MD. ASADUDDIN>, <SHREYA CHEPURI>,** bearing hall ticket number, **<1602-20-737-016>,<1602-20-737-024>,<1602-20-737-040>**hereby declare that the project report entitled **<”VIRTUAL MOUSE”>** Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by me and the results embodied in this project report has not been submitted to any other university or institute for the award of any other degree or diploma.

### < HARSHITH PAGINDLA ><1602-20-737-016>

**<MD ASADUDDIN><1602-20-737-024>**

**<SHREYA CHEPURI><1602-20-737-040>**

## VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)

### (AFFILIATED TO OSMANIA UNIVERSITY) HYDERABAD - 500 030

**Department of Information Technology**



### BONAFIDE CERTIFICATE

This is to certify that the project entitled “**VIRTUAL MOUSE**” being submitted by

### HARSHITH, ASADUDDIN,SHREYA bearing 1602-20-737-016, 1602-20-737-024,

**1602-20-737-040**, in partial fulfillment of the requirements for the completion of MINI PROJECT of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by them under my guidance.

Internal Guide External Examiner Dr.K Ram Moahn Rao Mrs.Sireesha HOD, I

## ACKNOWLEDGEMENT

We thank the department of INFORMATION TECHNOLOGY, for introducing the subject “Mini Project-2” in BE fifth semester.

We would also like to show our appreciation to our Honorable principal, Dr S V Ramana sir, our HOD K. Ram Mohan Rao for supporting us and our mini project lecturer, Mrs sireesha mam, for letting us properly understand the process of doing a project and for providing valuable insight and expertise that has greatly assisted us in the making of the project.

### TABLE OF CONTENTS:

1. INTRODUCTION
   1. PURPOSE

1.2 PRODUCT SCOPE

* 1. PROBLEM DEFINITON

1. PROPOSED WORK
   1. ARCHITECTURE/DESIGN
   2. UI PROTOTYPES OR SCREENSHOTS
   3. TECHNOLOGIES USED
   4. IMPLEMENTATION
      * MODULES
      * CODE AND GITHUB LINKS
   5. TESTING
2. RESULTS
3. DISCUSSION AND FUTURE WORK
4. REFERENCES

**ABSTRACT**

Virtual mouse is a project that allow users to control their computers using a mouse-like interface, but without the need for a physical mouse using gesture recognition. It can be useful for people who have difficulty using a physical mouse, such as individuals with mobility impairments or other physical disabilities. Virtual mouse can also be useful in situations where a physical mouse is not practical, such as when using a laptop or other portable device. Additionally, some virtual mouse offer additional features or customization options that are not available with physical mouse, making them appealing to users who want more control over their computing experience.

# CHAPTER 1 INTRODUCTION

**What is a virtual mouse?**

The virtual mouse is a mouse which is not like our physical mouse. It allows a user to control the cursor on a computer screen using hand gestures or body movements. The virtual mouse uses software to track the user's movements and translate them into cursor movements on the screen, providing an alternative to the traditional physical mouse.

### PURPOSE

The virtual mouse system is useful for many applications. It can be used to reduce the space for using the physical mouse, and it can be used in situations where we cannot use the physical mouse. The system eliminates the usage of devices, and it improves the human-computer interaction.

* 1. **PRODUCT SCOPE**

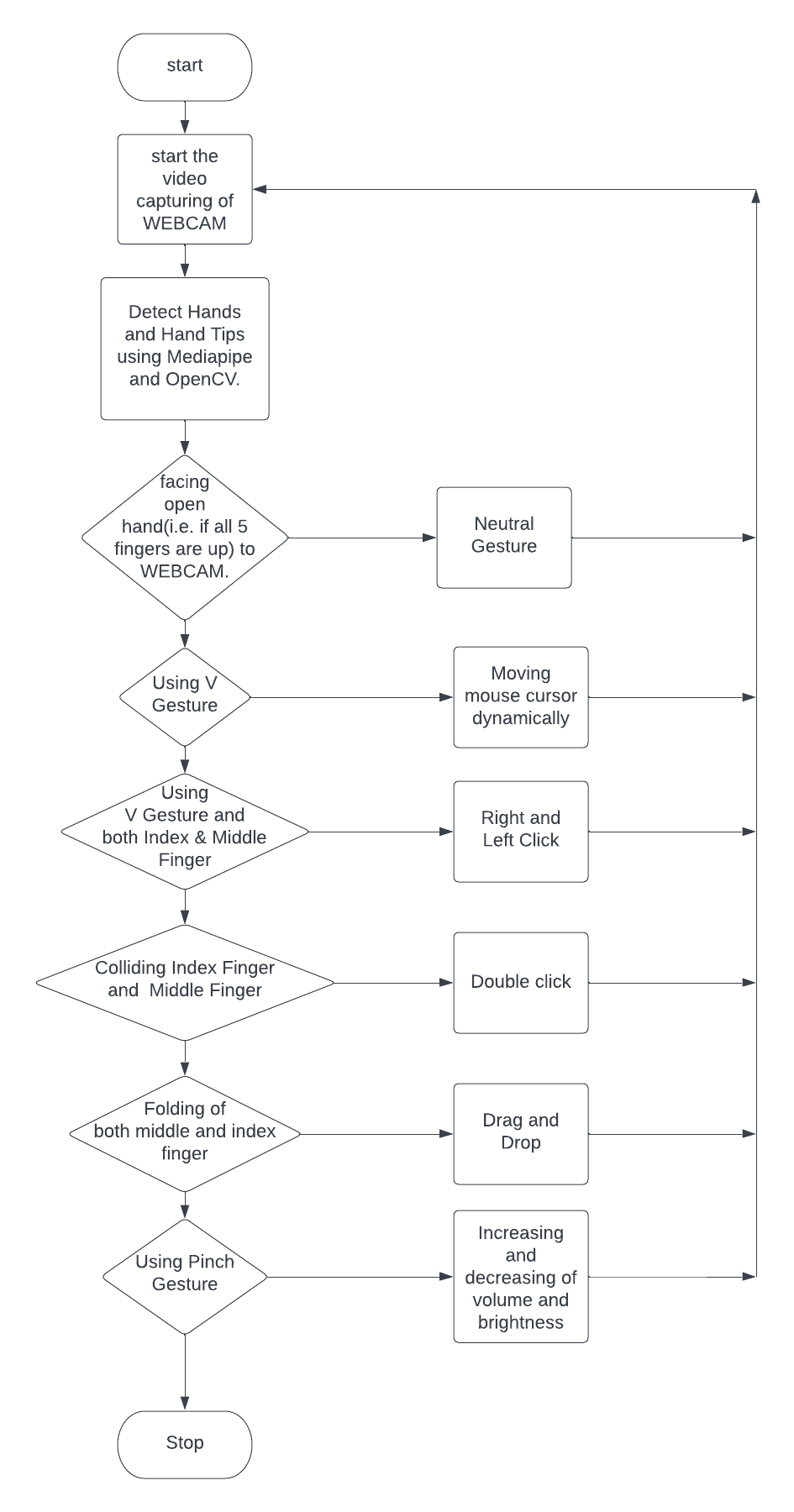
Hand gestures also play a vital role in our present world. Gesture Controlled Virtual Mouse makes human computer interaction simple by making use of Hand Gestures and Voice Commands. The computer requires almost no direct contact. It can be extended to medical operations, Gaming and to control home appliances (MP3 player, TV)

### PROBLEM DEFINATION

### Virtual mouse is a project that allow users to control their computers using a mouse-like interface, but without the need for a physical mouse using gesture recognition. It can be useful for people who have difficulty using a physical mouse, such as individuals with mobility impairments or other physical disabilities.

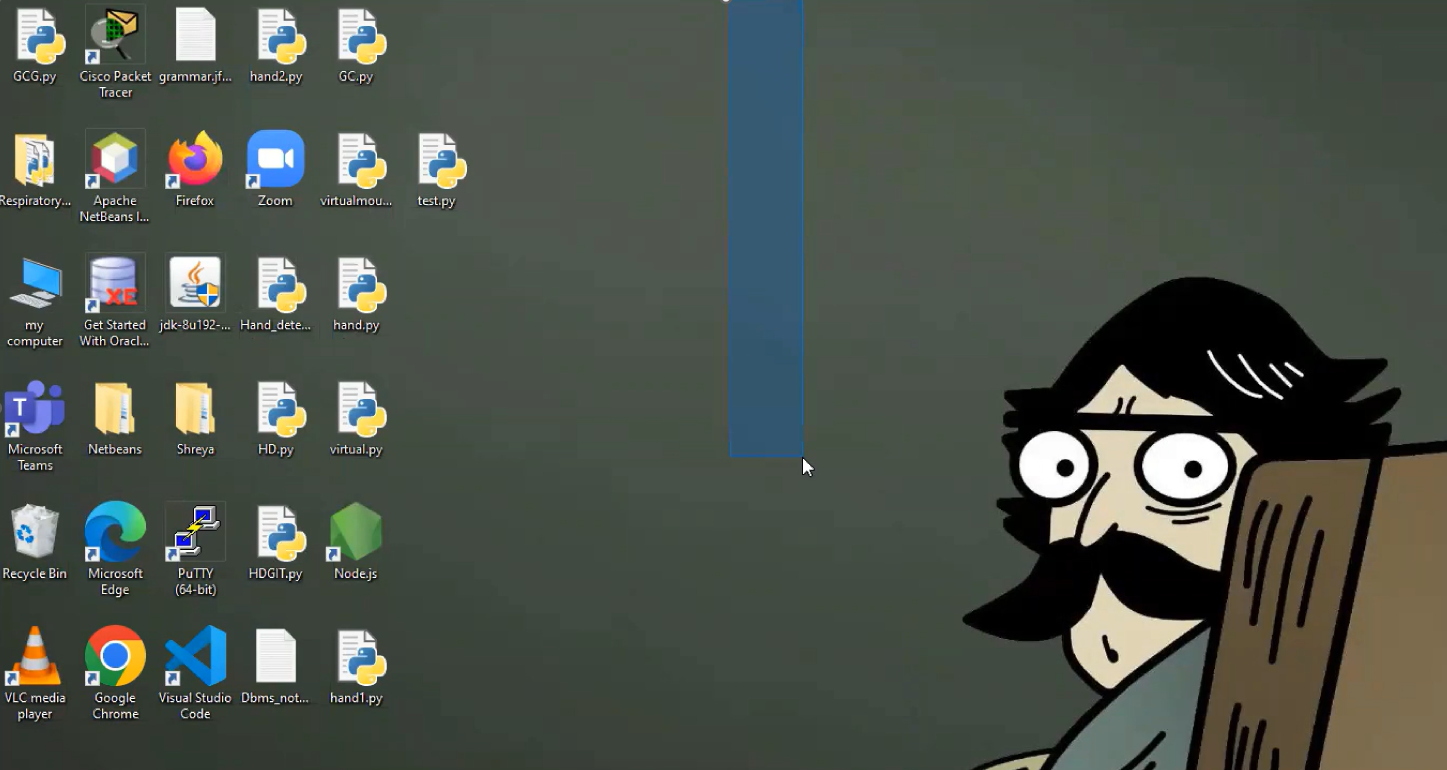
# CHAPTER 3 PROPOSED WORK –

* 1. **Architecture–**



# UI prototypes or screenshots





**2.3 TECHNOLOGIES USED –**

The tool using which the “Virtual mouse” was made is Python.

Python editor

Python modules

**2.4 MODULES –**

**cv2:** Open cv is an open-source library which is very useful for computer vision applications such as video analysis, CCTV footage analysis and image analysis.  We can access our webcam using pass 0 in the function parameter.

**● Media pipe:** Media pipe is a cross-platform library developed by Google that provides amazing ready-to-use ML solutions for computer vision tasks.(Detection of hand with lines and dots)

**● pyautogui:** PyAutoGUI is a Python module which can automate your GUI and programmatically control your keyboard and mouse.

**● math**: math module to perform various mathematical calculations.

● **Enum**: used for set of symbolic names (members) bound to unique values can be iterated over to return its members in definition order.

● **ctypes**: is a foreign function library for Python. It provides C compatible data types(for casting volume pointer i.e to increase or decrease volume using pointer).

● **screen\_brightness\_control:** for controlling screen brightness

**Code –**

import cv2

import mediapipe as mp

import pyautogui

import math

from enum import IntEnum

from ctypes import cast, POINTER

from comtypes import CLSCTX\_ALL

from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume

from google.protobuf.json\_format import MessageToDict

import screen\_brightness\_control as sbcontrol

pyautogui.FAILSAFE = False

mp\_drawing = mp.solutions.drawing\_utils

mp\_hands = mp.solutions.hands

class Gest(IntEnum):

FIST = 0

PINKY = 1

RING = 2

MID = 4

LAST3 = 7

INDEX = 8

FIRST2 = 12

LAST4 = 15

THUMB = 16

PALM = 31

V\_GEST = 33

TWO\_FINGER\_CLOSED = 34

PINCH\_MAJOR = 35

PINCH\_MINOR = 36

class HLabel(IntEnum):

MINOR = 0

MAJOR = 1

class HandRecog:

def \_init\_(self, hand\_label):

self.finger = 0

self.ori\_gesture = Gest.PALM

self.prev\_gesture = Gest.PALM

self.frame\_count = 0

self.hand\_result = None

self.hand\_label = hand\_label

def update\_hand\_result(self, hand\_result):

self.hand\_result = hand\_result

def get\_signed\_dist(self, point):

sign = -1

if self.hand\_result.landmark[point[0]].y < self.hand\_result.landmark[point[1]].y:

sign = 1

dist = (self.hand\_result.landmark[point[0]].x - self.hand\_result.landmark[point[1]].x)\*\*2

dist += (self.hand\_result.landmark[point[0]].y - self.hand\_result.landmark[point[1]].y)\*\*2

dist = math.sqrt(dist)

return dist\*sign

def get\_dist(self, point):

dist = (self.hand\_result.landmark[point[0]].x - self.hand\_result.landmark[point[1]].x)\*\*2

dist += (self.hand\_result.landmark[point[0]].y - self.hand\_result.landmark[point[1]].y)\*\*2

dist = math.sqrt(dist)

return dist

def get\_dz(self,point):

return abs(self.hand\_result.landmark[point[0]].z - self.hand\_result.landmark[point[1]].z)

def set\_finger\_state(self):

if self.hand\_result == None:

return

points = [[8,5,0],[12,9,0],[16,13,0],[20,17,0]]

self.finger = 0

self.finger = self.finger | 0 #thumb

for idx,point in enumerate(points):

dist = self.get\_signed\_dist(point[:2])

dist2 = self.get\_signed\_dist(point[1:])

try:

ratio = round(dist/dist2,1)

except:

ratio = round(dist1/0.01,1)

self.finger = self.finger << 1

if ratio > 0.5 :

self.finger = self.finger | 1

def get\_gesture(self):

if self.hand\_result == None:

return Gest.PALM

current\_gesture = Gest.PALM

if self.finger in [Gest.LAST3,Gest.LAST4] and self.get\_dist([8,4]) < 0.05:

if self.hand\_label == HLabel.MINOR :

current\_gesture = Gest.PINCH\_MINOR

else:

current\_gesture = Gest.PINCH\_MAJOR

elif Gest.FIRST2 == self.finger :

point = [[8,12],[5,9]]

dist1 = self.get\_dist(point[0])

dist2 = self.get\_dist(point[1])

ratio = dist1/dist2

if ratio > 1.7:

current\_gesture = Gest.V\_GEST

else:

if self.get\_dz([8,12]) < 0.1:

current\_gesture = Gest.TWO\_FINGER\_CLOSED

else:

current\_gesture = Gest.MID

else:

current\_gesture = self.finger

if current\_gesture == self.prev\_gesture:

self.frame\_count += 1

else:

self.frame\_count = 0

self.prev\_gesture = current\_gesture

if self.frame\_count > 4 :

self.ori\_gesture = current\_gesture

return self.ori\_gesture

class Controller:

tx\_old = 0

ty\_old = 0

trial = True

flag = False

grabflag = False

pinchmajorflag = False

pinchminorflag = False

pinchstartxcoord = None

pinchstartycoord = None

pinchdirectionflag = None

prevpinchlv = 0

pinchlv = 0

framecount = 0

prev\_hand = None

pinch\_threshold = 0.3

def getpinchylv(hand\_result):

"""returns distance beween starting pinch y coord and current hand position y coord."""

dist = round((Controller.pinchstartycoord - hand\_result.landmark[8].y)\*10,1)

return dist

def getpinchxlv(hand\_result):

"""returns distance beween starting pinch x coord and current hand position x coord."""

dist = round((hand\_result.landmark[8].x - Controller.pinchstartxcoord)\*10,1)

return dist

def changesystembrightness():

"""sets system brightness based on 'Controller.pinchlv'."""

currentBrightnessLv = sbcontrol.get\_brightness(display=0)/100.0

currentBrightnessLv += Controller.pinchlv/50.0

if currentBrightnessLv > 1.0:

currentBrightnessLv = 1.0

elif currentBrightnessLv < 0.0:

currentBrightnessLv = 0.0

sbcontrol.fade\_brightness(int(100\*currentBrightnessLv) , start = sbcontrol.get\_brightness(display=0))

def changesystemvolume():

"""sets system volume based on 'Controller.pinchlv'."""

devices = AudioUtilities.GetSpeakers()

interface = devices.Activate(IAudioEndpointVolume.iid, CLSCTX\_ALL, None)

volume = cast(interface, POINTER(IAudioEndpointVolume))

currentVolumeLv = volume.GetMasterVolumeLevelScalar()

currentVolumeLv += Controller.pinchlv/50.0

if currentVolumeLv > 1.0:

currentVolumeLv = 1.0

elif currentVolumeLv < 0.0:

currentVolumeLv = 0.0

volume.SetMasterVolumeLevelScalar(currentVolumeLv, None)

def scrollVertical():

"""scrolls on screen vertically."""

pyautogui.scroll(120 if Controller.pinchlv>0.0 else -120)

def scrollHorizontal():

"""scrolls on screen horizontally."""

pyautogui.keyDown('shift')

pyautogui.keyDown('ctrl')

pyautogui.scroll(-120 if Controller.pinchlv>0.0 else 120)

pyautogui.keyUp('ctrl')

pyautogui.keyUp('shift')

def get\_position(hand\_result):

point = 9

position = [hand\_result.landmark[point].x ,hand\_result.landmark[point].y]

sx,sy = pyautogui.size()

x\_old,y\_old = pyautogui.position()

x = int(position[0]\*sx)

y = int(position[1]\*sy)

if Controller.prev\_hand is None:

Controller.prev\_hand = x,y

delta\_x = x - Controller.prev\_hand[0]

delta\_y = y - Controller.prev\_hand[1]

distsq = delta\_x\*2 + delta\_y\*2

ratio = 1

Controller.prev\_hand = [x,y]

if distsq <= 25:

ratio = 0

elif distsq <= 900:

ratio = 0.07 \* (distsq \*\* (1/2))

else:

ratio = 2.1

x , y = x\_old + delta\_x\*ratio , y\_old + delta\_y\*ratio

return (x,y)

def pinch\_control\_init(hand\_result):

Controller.pinchstartxcoord = hand\_result.landmark[8].x

Controller.pinchstartycoord = hand\_result.landmark[8].y

Controller.pinchlv = 0

Controller.prevpinchlv = 0

Controller.framecount = 0

# Hold final position for 5 frames to change status

def pinch\_control(hand\_result, controlHorizontal, controlVertical):

if Controller.framecount == 5:

Controller.framecount = 0

Controller.pinchlv = Controller.prevpinchlv

if Controller.pinchdirectionflag == True:

controlHorizontal() #x

elif Controller.pinchdirectionflag == False:

controlVertical() #y

lvx = Controller.getpinchxlv(hand\_result)

lvy = Controller.getpinchylv(hand\_result)

if abs(lvy) > abs(lvx) and abs(lvy) > Controller.pinch\_threshold:

Controller.pinchdirectionflag = False

if abs(Controller.prevpinchlv - lvy) < Controller.pinch\_threshold:

Controller.framecount += 1

else:

Controller.prevpinchlv = lvy

Controller.framecount = 0

elif abs(lvx) > Controller.pinch\_threshold:

Controller.pinchdirectionflag = True

if abs(Controller.prevpinchlv - lvx) < Controller.pinch\_threshold:

Controller.framecount += 1

else:

Controller.prevpinchlv = lvx

Controller.framecount = 0

def handle\_controls(gesture, hand\_result):

"""Impliments all gesture functionality."""

x,y = None,None

if gesture != Gest.PALM :

x,y = Controller.get\_position(hand\_result)

# flag reset

if gesture != Gest.FIST and Controller.grabflag:

Controller.grabflag = False

pyautogui.mouseUp(button = "left")

if gesture != Gest.PINCH\_MAJOR and Controller.pinchmajorflag:

Controller.pinchmajorflag = False

if gesture != Gest.PINCH\_MINOR and Controller.pinchminorflag:

Controller.pinchminorflag = False

# implementation

if gesture == Gest.V\_GEST:

Controller.flag = True

pyautogui.moveTo(x, y, duration = 0.1)

elif gesture == Gest.FIST:

if not Controller.grabflag :

Controller.grabflag = True

pyautogui.mouseDown(button = "left")

pyautogui.moveTo(x, y, duration = 0.1)

elif gesture == Gest.MID and Controller.flag:

pyautogui.click()

Controller.flag = False

elif gesture == Gest.INDEX and Controller.flag:

pyautogui.click(button='right')

Controller.flag = False

elif gesture == Gest.TWO\_FINGER\_CLOSED and Controller.flag:

pyautogui.doubleClick()

Controller.flag = False

elif gesture == Gest.PINCH\_MINOR:

if Controller.pinchminorflag == False:

Controller.pinch\_control\_init(hand\_result)

Controller.pinchminorflag = True

Controller.pinch\_control(hand\_result,Controller.scrollHorizontal, Controller.scrollVertical)

elif gesture == Gest.PINCH\_MAJOR:

if Controller.pinchmajorflag == False:

Controller.pinch\_control\_init(hand\_result)

Controller.pinchmajorflag = True

class GestureController:

gc\_mode = 0

cap = None

CAM\_HEIGHT = None

CAM\_WIDTH = None

hr\_major = None # Right Hand by default

hr\_minor = None # Left hand by default

dom\_hand = True

def \_init\_(self):

GestureController.gc\_mode = 1

GestureController.cap = cv2.VideoCapture(0)

GestureController.CAM\_HEIGHT = GestureController.cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)

GestureController.CAM\_WIDTH = GestureController.cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH)

def classify\_hands(results):

left , right = None,None

try:

handedness\_dict = MessageToDict(results.multi\_handedness[0])

if handedness\_dict['classification'][0]['label'] == 'Right':

right = results.multi\_hand\_landmarks[0]

else :

left = results.multi\_hand\_landmarks[0]

except:

pass

try:

handedness\_dict = MessageToDict(results.multi\_handedness[1])

if handedness\_dict['classification'][0]['label'] == 'Right':

right = results.multi\_hand\_landmarks[1]

else :

left = results.multi\_hand\_landmarks[1]

except:

pass

if GestureController.dom\_hand == True:

GestureController.hr\_major = right

GestureController.hr\_minor = left

else :

GestureController.hr\_major = left

GestureController.hr\_minor = right

def start(self):

handmajor = HandRecog(HLabel.MAJOR)

handminor = HandRecog(HLabel.MINOR)

with mp\_hands.Hands(max\_num\_hands = 2,min\_detection\_confidence=0.5, min\_tracking\_confidence=0.5) as hands:

while GestureController.cap.isOpened() and GestureController.gc\_mode:

success, image = GestureController.cap.read()

if not success:

print("Ignoring empty camera frame.")

continue

image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR\_BGR2RGB)

image.flags.writeable = False

results = hands.process(image)

image.flags.writeable = True

image = cv2.cvtColor(image, cv2.COLOR\_RGB2BGR)

if results.multi\_hand\_landmarks:

GestureController.classify\_hands(results)

handmajor.update\_hand\_result(GestureController.hr\_major)

handminor.update\_hand\_result(GestureController.hr\_minor)

handmajor.set\_finger\_state()

handminor.set\_finger\_state()

gest\_name = handminor.get\_gesture()

if gest\_name == Gest.PINCH\_MINOR:

Controller.handle\_controls(gest\_name, handminor.hand\_result)

else:

gest\_name = handmajor.get\_gesture()

Controller.handle\_controls(gest\_name, handmajor.hand\_result)

for hand\_landmarks in results.multi\_hand\_landmarks:

mp\_drawing.draw\_landmarks(image, hand\_landmarks, mp\_hands.HAND\_CONNECTIONS)

else:

Controller.prev\_hand = None

cv2.imshow('Gesture Controller', image)

if cv2.waitKey(5) & 0xFF == 13:

break

GestureController.cap.release()

cv2.destroyAllWindows()

gc1 = GestureController()

gc1.start()

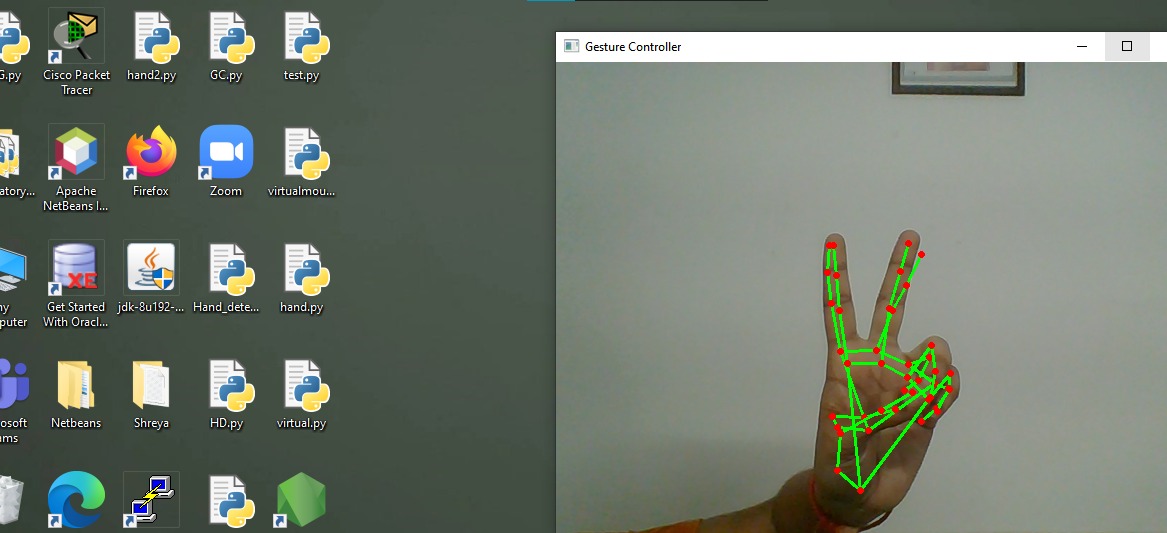
3.5.3 – GitHub Links –

<https://github.com/Shreyachepuri/MiniProject>

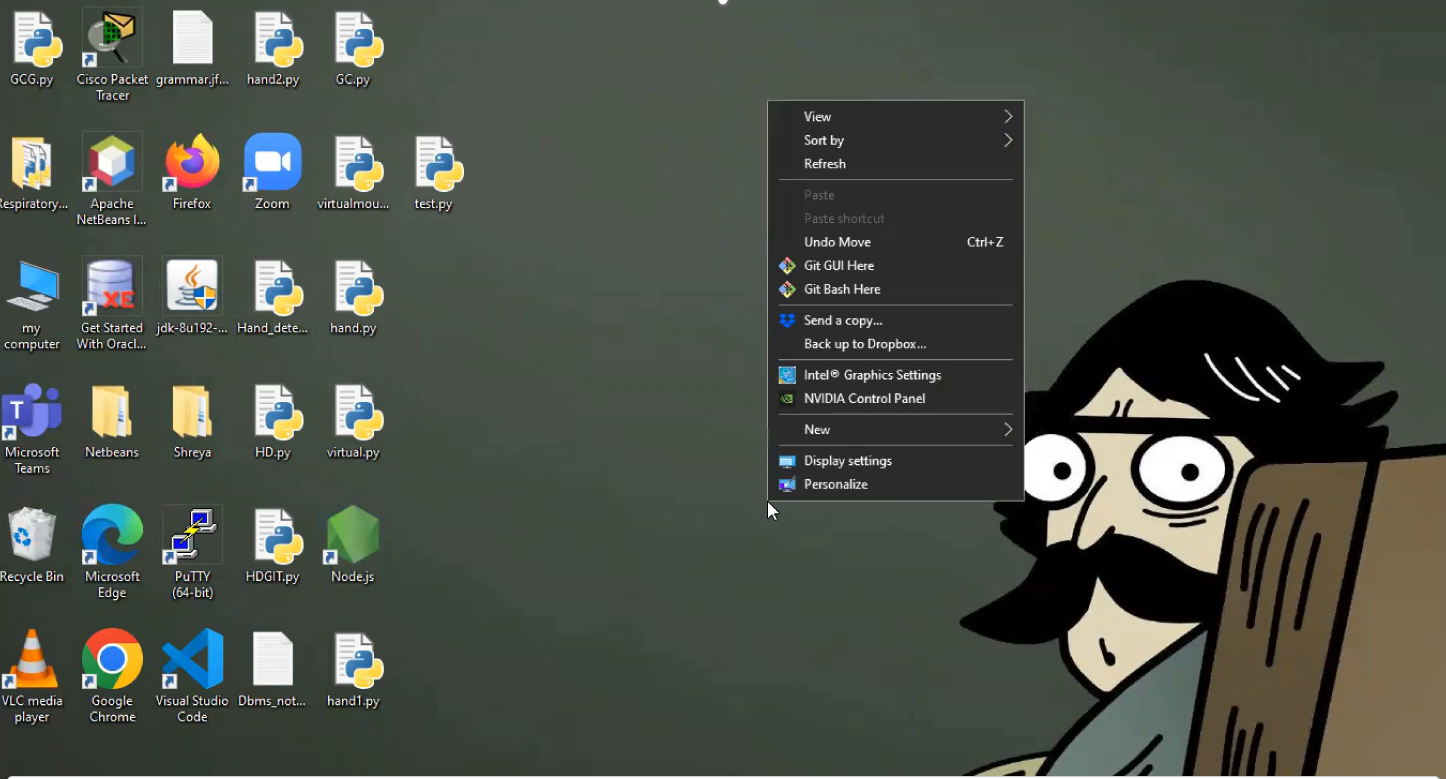
RESULTS:

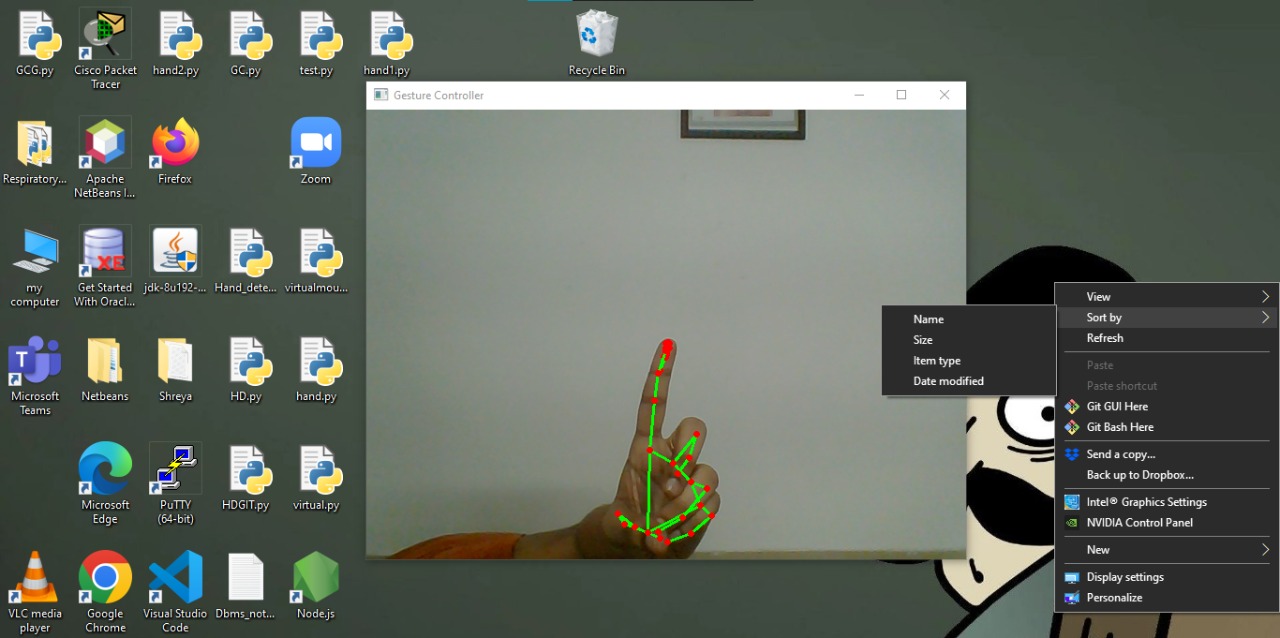
Detection of hand:

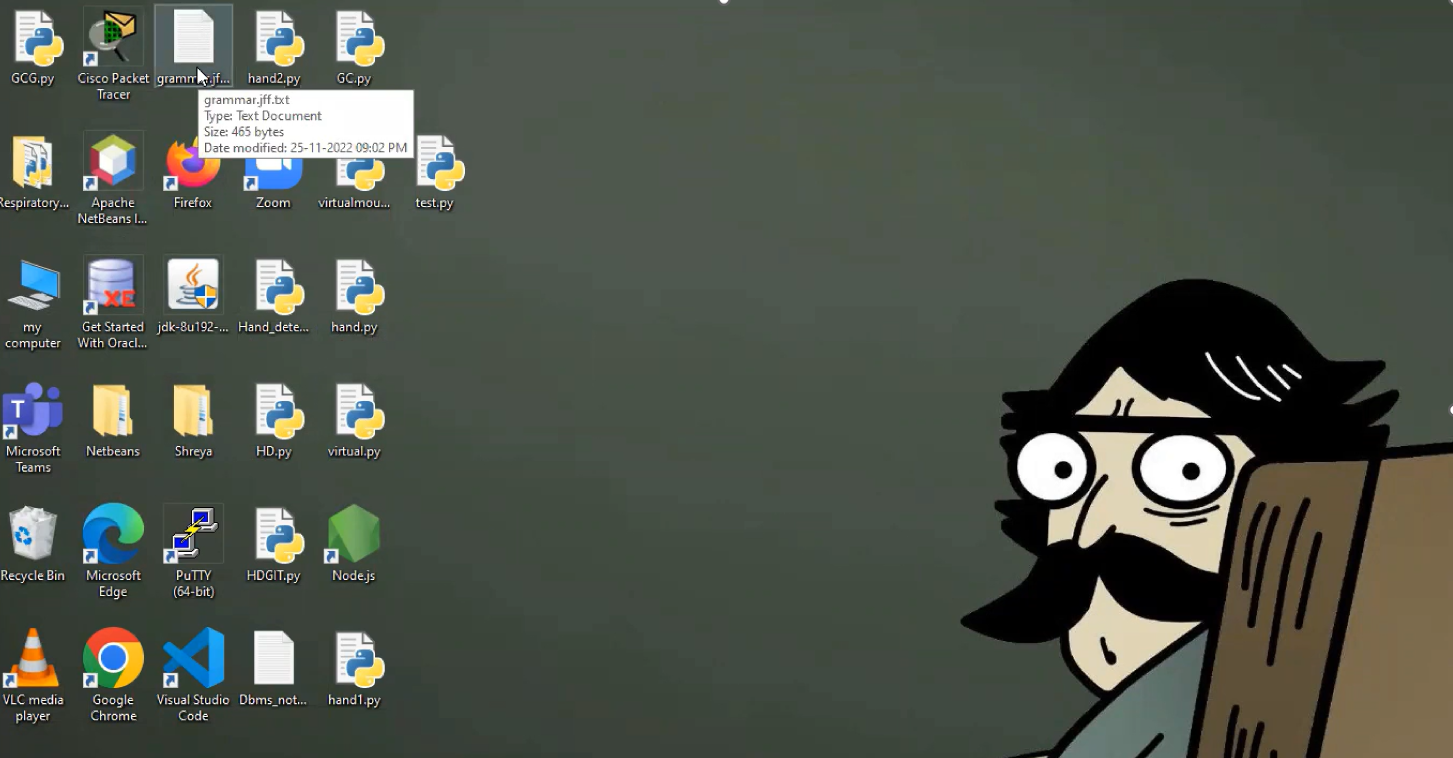
Dynamic movement of cursor using V gesture:

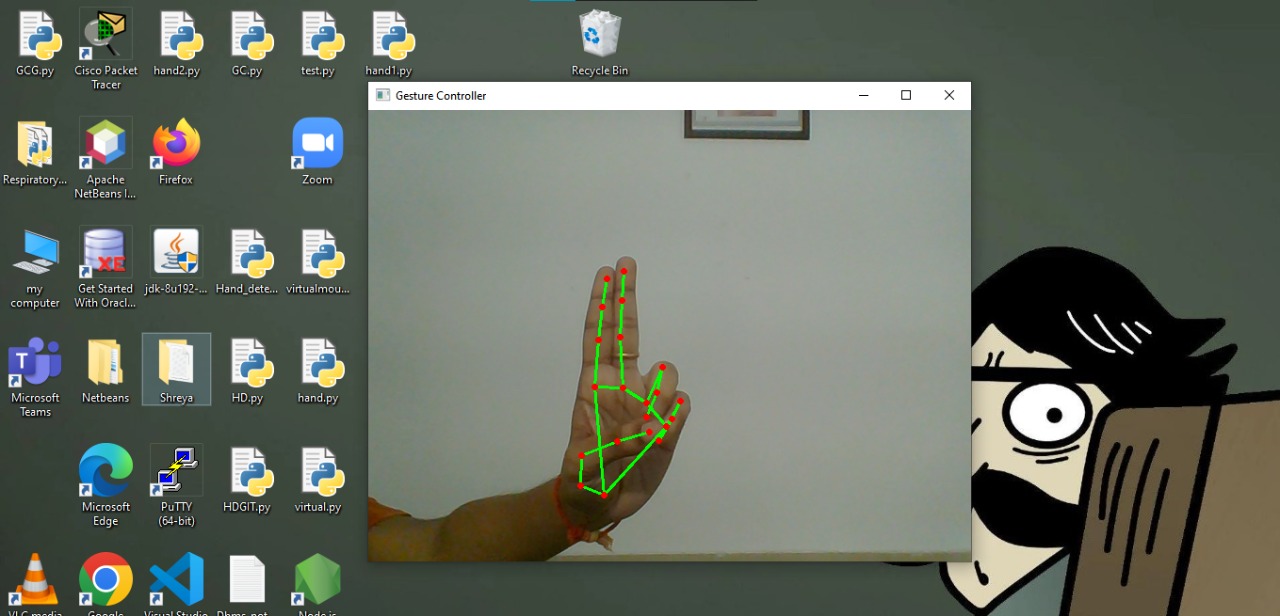


Right click using index finger:

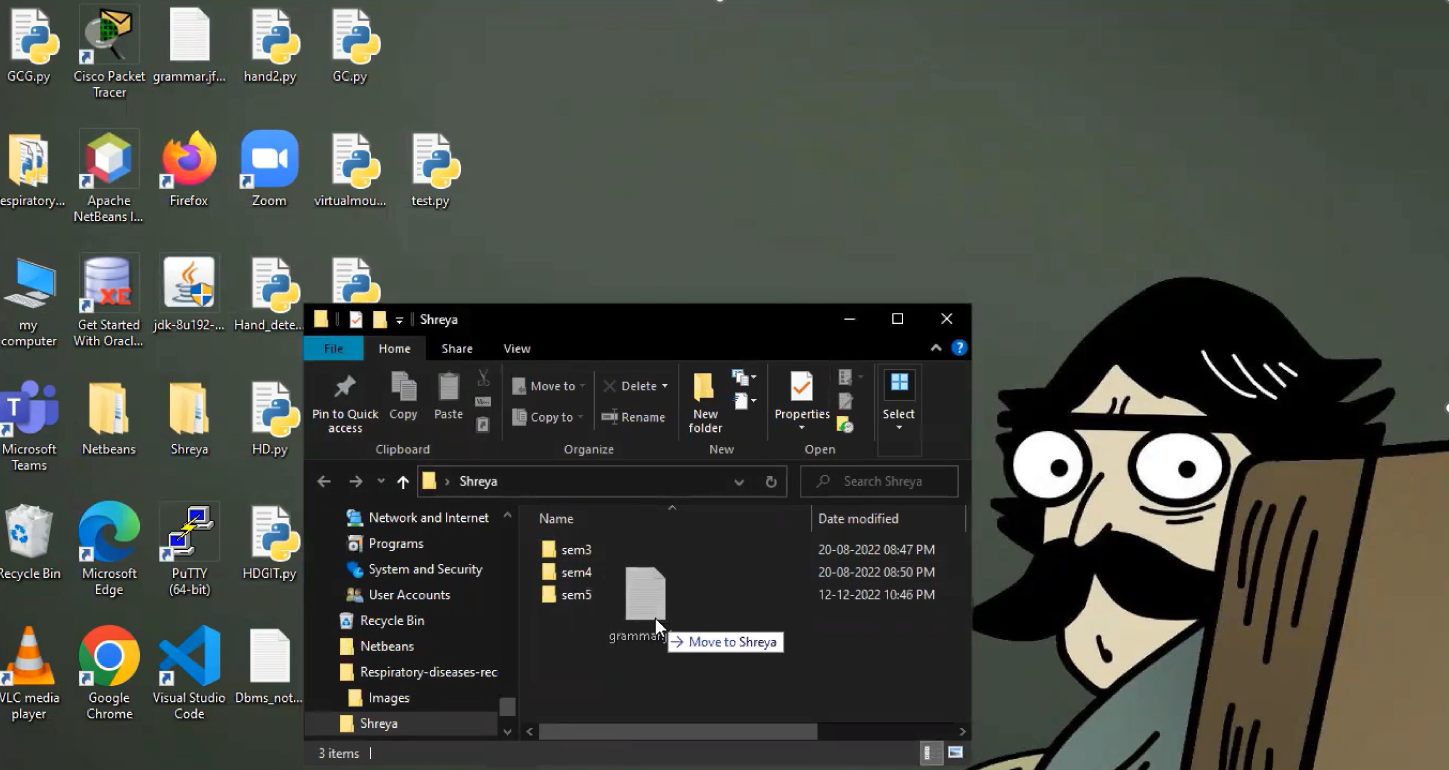




Double click using index and middle finger: 



Drag and drop using folding of index and middle finger



CHAPTER 5

DISCUSSION AND FUTURE WORK –

* As we know Gesture recognition is playing a major role in present world as AI is emerging.
* So it has a wide range of applications ranging from home appliances to medical field.
* This project can further extended to gaming or any application in medical field.
* In future, we are planning to add voice recognition to enable more features and flexibility.

# CHAPTER 6 REFERENCES

[**https://ieeexplore.ieee.org/document/8404314/**](https://ieeexplore.ieee.org/document/8404314/)

[**https://ieeexplore.ieee.org/document/8597147**](https://ieeexplore.ieee.org/document/8597147)