

# **CURSOR MOVEMENT ON OBJECT MOTION**

**An Application Development – II (Project) Report Submitted  
In partial fulfilment of the requirement for the award of the degree of**

**Bachelor of Technology  
In  
Computer Science and Engineering  
(Internet of Things)**

**By**

<b>ALLTHOLA HARIKA</b>	<b>20N31A6905</b>
<b>KARNATI SRIKANTH</b>	<b>20N31A6925</b>
<b>P AKHIL VIGNAN</b>	<b>20N31A6942</b>

**Under the guidance of**

**Mrs. V. Divya**

**Assistant Professor**

**Department of Emerging Technologies  
MRCET (Autonomous Institution, UGC Govt. of India)**



**MRCET CAMPUS**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
(EMERGING TECHNOLOGIES)**

**MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(Autonomous Institution - UGC, Govt. of India)**

**(Affiliated to JNTU, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC – 'A' Grade, ISO 9001:2015 Certified)  
Maisammaguda (v), Near Dullapally, Via: Kompally, Hyderabad – 500 100, Telangana State, India 2022-**

**2023**



Estd : 2004

# MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

## (Autonomous Institution – UGC, Govt. of India)

(Sponsored by CMR Educational Society)

Recognized under 2(f) and 12 (B) of UGC ACT 1956

( Affiliated to JNTUH, Hyderabad, Approved by AICTE- Accredited by NBA & NAAC– 'A' Grade - ISO 9001:2015 Certified )



## CERTIFICATE

This is to certify that this is the bonafide record of the project titled “**CURSOR MOVEMENT ON OBJECT MOTION**”, submitted by A **HARIKA** (20N31A6905), **KARNATI SRIKANTH** (20N31A6925) and **P AKHIL VIGNAN** (20N31A6942) of **B. Tech III YEAR – II Semester** in the partial fulfilment of the requirements for the degree of **Bachelor of Technology in Computer Science and Engineering (Internet of Things)**, Dept. of CSE (Emerging Technologies) during the year 2022-2023. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Mrs. V. Divya  
**Project Guide**  
Department of CSE (ET)

Mrs. Lakshmi Prasudha  
**Project Coordinator**  
Department of CSE (ET)

Dr. P. Dileep  
**OVERALL COORDINATOR**

**EXTERNAL  
EXAMINER**

Dr. M. V. Kamal  
**HEAD OF THE DEPARTMENT**

Date of Viva-Voce Examination held on: \_\_\_\_\_

## **DECLARATION**

We hereby declare that the project entitled “**CURSOR MOVEMENT ON OBJECT MOTION**” submitted to **Malla Reddy College of Engineering and Technology**, affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) as part of III Year B. Tech – II Semester and for the partial fulfilment of the requirement for the award of **Bachelor of Technology in Computer Science and Engineering (Internet of Things)** is a result of original research work done by us.

It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

<b>ALLTHOLA HARIKA</b>	-	<b>20N31A6905</b>
<b>KARNATI SRIKANTH</b>	-	<b>20N31A6925</b>
<b>P AKHIL VIGNAN</b>	-	<b>20N31A6942</b>

## **ACKNOWLEDGEMENTS**

We feel ourself honoured and privileged to place our warm salutation to our college “Malla Reddy College of Engineering and Technology (Autonomous Institution – UGC Govt. of India) and our Principal **Dr. S Srinivasa Rao**, Professor who gave us the opportunity to do the Application Development -1 (Project) during our III Year B. Tech and profound the technical skills.

We express our heartiest thanks to our Director **Dr. V S K Reddy**, Professor for encouraging us in every aspect of our project and helping us realize our full potential.

We also thankful to our Head of the Department **Dr. M V Kamal**, Professor for providing training and guidance, excellent infrastructure, and a nice atmosphere for completing this project successfully.

We would like to express our sincere gratitude and indebtedness to our project supervisor **Mrs. V. Divya**, Assistant Professor for her valuable suggestions and interest throughout the course of this project.

We convey our heartfelt thanks to our Project Coordinator **Dr. P Dileep**, Professor for allowing for their regular guidance and constant encouragement during our dissertation work.

We would like to thank all our supporting **staff** of the Department of CSE (Emerging Technologies) and even all other department who have been helpful directly and in-directly in making our project a success.

Finally, we would like to take this opportunity to thank our **families** for their support and blessings for completion of our project that gave us the strength to do our project.

<b>A. Harika</b>	-	<b>20N31A6905</b>
<b>K. Srikanth</b>	-	<b>20N31A6925</b>
<b>P. Akhil Vignan</b>	-	<b>20N31A6942</b>

## **ABSTRACT**

This project is a mouse simulation system which performs all the functions performed by your mouse corresponding to your hand movements and gestures. System will detect hand which will act as cursor. In this the camera captures your video and depending on your hand gestures, you can move the cursor and perform left click, right click, drag, select and scroll up and down. The predefined gestures make use of only two fingers marked numerically. The project is essentially a program which applies image processing, retrieves necessary data and implements it to the mouse interface of the computer according to predefined notions. The code is written on Python. Its uses of the cross-platform image processing module OpenCV and implements the mouse actions using Python specific library PyAutoGUI. The system can be broken down in three main components, which are - Hand Tracking, Gesture Recognition, Cursor Control. It aims to provide the user a better understanding of the system and to let them use alternate ways of interacting with the computer for a task. The project can be useful for various professional and non-professional presentations. It can also be used at home by users for recreational purposes like while watching movies or playing games.

## **TABLE OF CONTENTS**

<b>S.No.</b>	<b>Topic</b>	<b>Page No.</b>
	CHAPTER 1: INTRODUCTION -----	1-3
	1.1: Problem Definition	
	1.2:Existing System	
	1.3:Proposed System	
	CHAPTER 2:SYSTEM REQUIREMENTS -----	4
	2.1:S/W Requirements	
	2.2:H/W Requirements	
	CHAPTER 3: SYSTEM DESIGN -----	5-11
	3.1:Software	
	3.2:UML Diagrams	
	3.3:System Architecture/ER Diagram/Sequential Diagram	
	CHAPTER 4:SOFTWARE DEVELOPMENT LIFECYCLE -----	12-16
	4.1: SDLC	
	4.2: Phase of SDLC	
	CHAPTER 5: IMPLEMENTATION -----	17-22
	CHAPTER 6: TESTING -----	23-27
	CHAPTER 7: CONCLUSION AND FUTURE SCOPE -----	28
	7.1: Conclusion	
	7.2: Future Scope	
	CHAPTER 8: REFERENCES -----	29

## **LIST OF FIGURES**

Figure No.	Figure Title	Page No.
1	Dataflow Diagram	7
2	Use Case Diagram	8
3	System Architecture	9
4	ER Diagram	10
5	Sequential Diagram	11
6	Phases of SDLC	12
7	Test Case for Right Click	23
8	Test Case for Selection	24
9	Test Case for Detection	25
10	Test Case for Opening Dialogue Box	26
11	Overall Test Cases	27

# **CHAPTER 1**

## **INTRODUCTION**

The importance of computers is increasing constantly. Computer can be used for many purposes. We often use hardware devices such as mouse and keyboard to interact with the computers. In today's world, technologies are evolving day by day. One of the examples is that we use Bluetooth installed in the system without having any wired connections. Computer Vision-Based Mouse is a system to control the cursor of our computer without using any physical device even a mouse. Our system basically uses image processing, object detection and motion tracking to control the mouse activities such as its movement, left-click, right-click and double click. A machine learning project based on python libraries like mediapipe and opencv. A virtual hand gesture mouse, that allows users to control a computer mouse using hand gestures instead of a physical mouse.

To develop this project, we will be using two functionalities like:

- Movement of cursor triggered by the hand movement.
- Click will be triggered based on the action of Index finger and Middle finger.

A hand landmark model is a machine learning model that is trained to identify and locate specific points or landmarks on a human hand in an image or video. These landmarks correspond to anatomical features of the hand such as the fingertips, knuckles, and wrist, and their precise locations can be used to track the movements and gestures of the hand.

➤ It performs features like:

- Click or select files.
- Drag and drop operation.
- Increase or decrease volume.
- Scroll pages, file or folders vertically
- Zoom in and out webpages.



## **1.1 PROBLEM DEFINITION**

The system uses computer vision and motion tracking technology to interpret the movements of the user's hand and translate them into mouse commands. It can also be used in situations where a physical mouse is not available or practical, such as in virtual reality or augmented reality environments. A hand landmark model is a machine learning model that is trained to identify and locate specific points or landmarks on a human hand in an image or video. These landmarks correspond to anatomical features of the hand such as the fingertips, knuckles, and wrist, and their precise locations can be used to track the movements and gestures of the hand.

## **1.2 EXISTING SYSTEM**

In the current scenario, the existing system has many flaws which make it inefficient to carry on with it. Here, the existing system uses an approach for controlling the mouse movement using colored objects like any RGB colors in front of the web cam for the movement of cursor. It consists of the simple mouse operation using the colored tips for detection which are captured by web-cam, hence colored fingers act as an object which the web-cam sense color like red, green, blue color to monitor the system. It performs the tasks like movement of the mouse and click events.

### **1.3 PROPOSED SYSTEM**

In the proposed system, we have generated the code individually for different actions like: Click or select file, drag and drop operation, Increase or decrease volume, scroll pages, file or folders vertically, zoom in and out webpages, there is no need of any RGB objects in the movement of cursor, it detects the hand as an object for the movement.

## **CHAPTER2**

### **SYSTEM REQUIREMENTS**

#### **2.1 Software Requirements:**

- Operating system: Windows
- Coding Language: Python with modules like-
  - Cv2 is used for video capturing.
  - Mediapipe is used for giving land mark for palm from 0-20.
  - PyAutoGUI provides the ability to simulate mouse cursor moves and clicks.

#### **2.2 Hardware Requirements:**

- Monitor
- Web Cam

#### **Programming Language:**

- Python Programming

## **CHAPTER 3**

### **SYSTEM DESIGN**

#### **3.1 SOFTWARE**

- Operating system: Windows
- Coding Language: Python with modules like-
  - Cv2 is used for video capturing.
  - Mediapipe is used for giving land mark for palm from 0-20.
  - Pyautogui provides the ability to simulate mouse cursor moves and clicks.

#### **PYTHON PROGRAMMING LANGUAGE:**

Python is a high-level, general-purpose programming language that is designed to be easy to read and write. It was first released in 1991 by Guido van Rossum and has since become one of the most popular programming languages in the world. Python's syntax emphasizes code readability, and it has a vast standard library and a large community of developers who have created many useful third-party libraries. Python can be used for a wide range of applications, including web development, data analysis, machine learning, scientific computing, and more. Additionally, it supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

#### **FEATURES OF PYTHON PROGRAMMING LANGUAGE:**

- Dynamic typing: variable types are determined at runtime, which can make code easier to write and read.

- Automatic memory management: memory is managed by the interpreter, which frees developers from having to manage memory explicitly.
- Strong support for data structures: Python provides built-in support for lists, dictionaries, and sets, which makes it easy to work with complex data.
- Easy to learn and use: Python's simple syntax and straightforward design make it an accessible language for beginners and experienced programmers alike.

## 3.2 Data Flow Diagrams / UML Diagrams

### 3.2.1 Data Flow Diagram

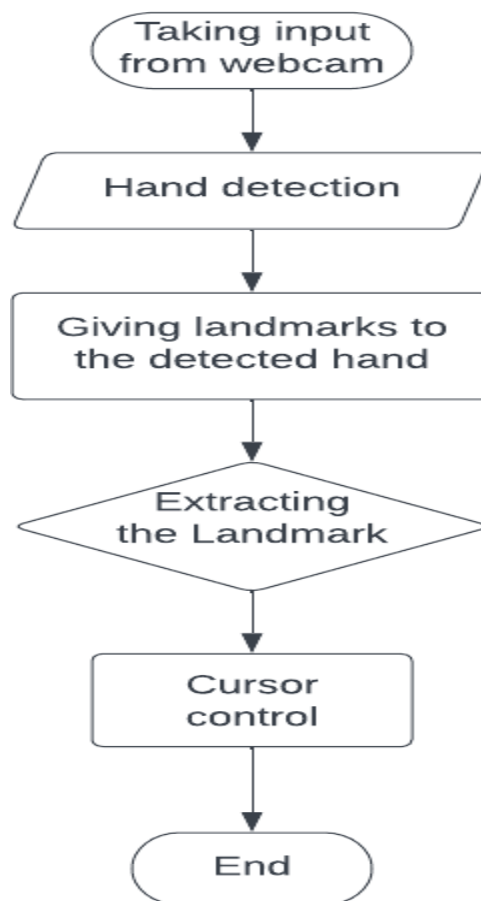


Fig 1: Data Flow Diagram

### 3.2.2 Use Case Diagram:



Fig 2: Use Case Diagram

### 3.3 System Architecture/ ER Diagram/ Sequential Diagram:

#### 3.3.1 System Architecture

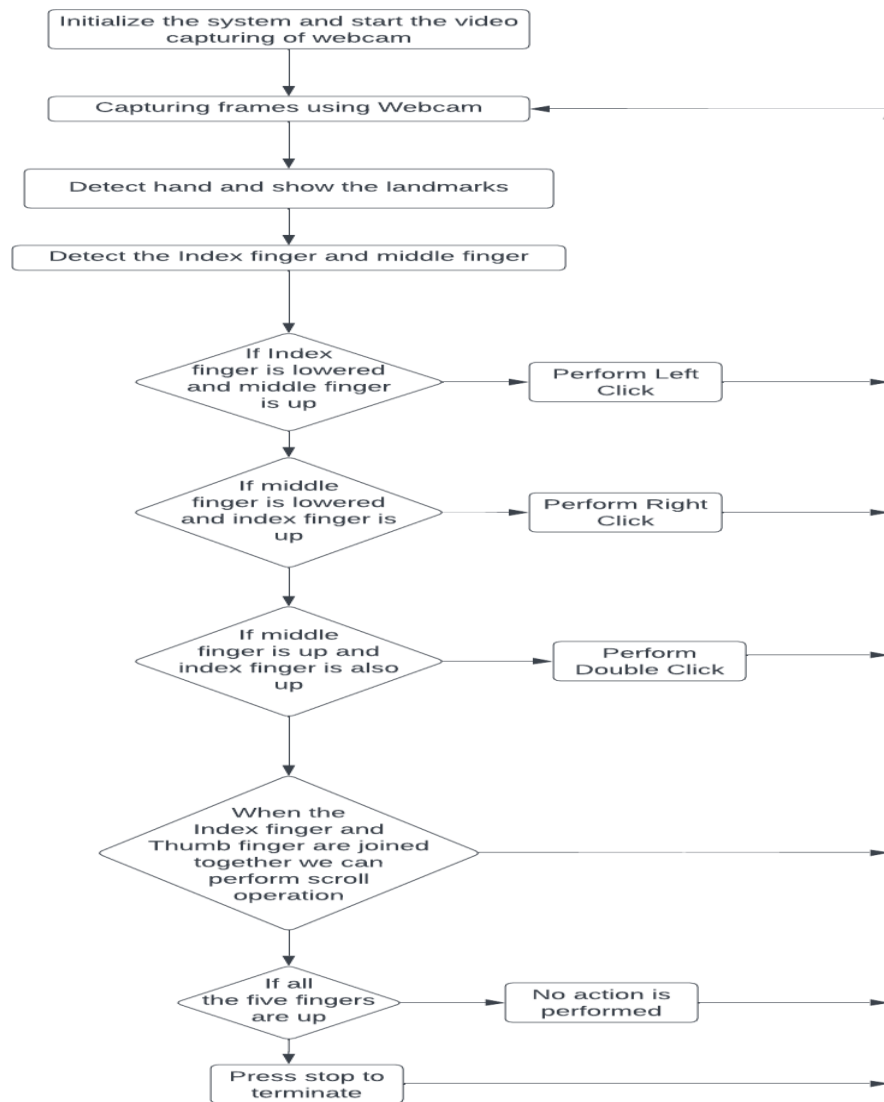


Fig 3: System Architecture



### 3.3.2 ER Diagram:

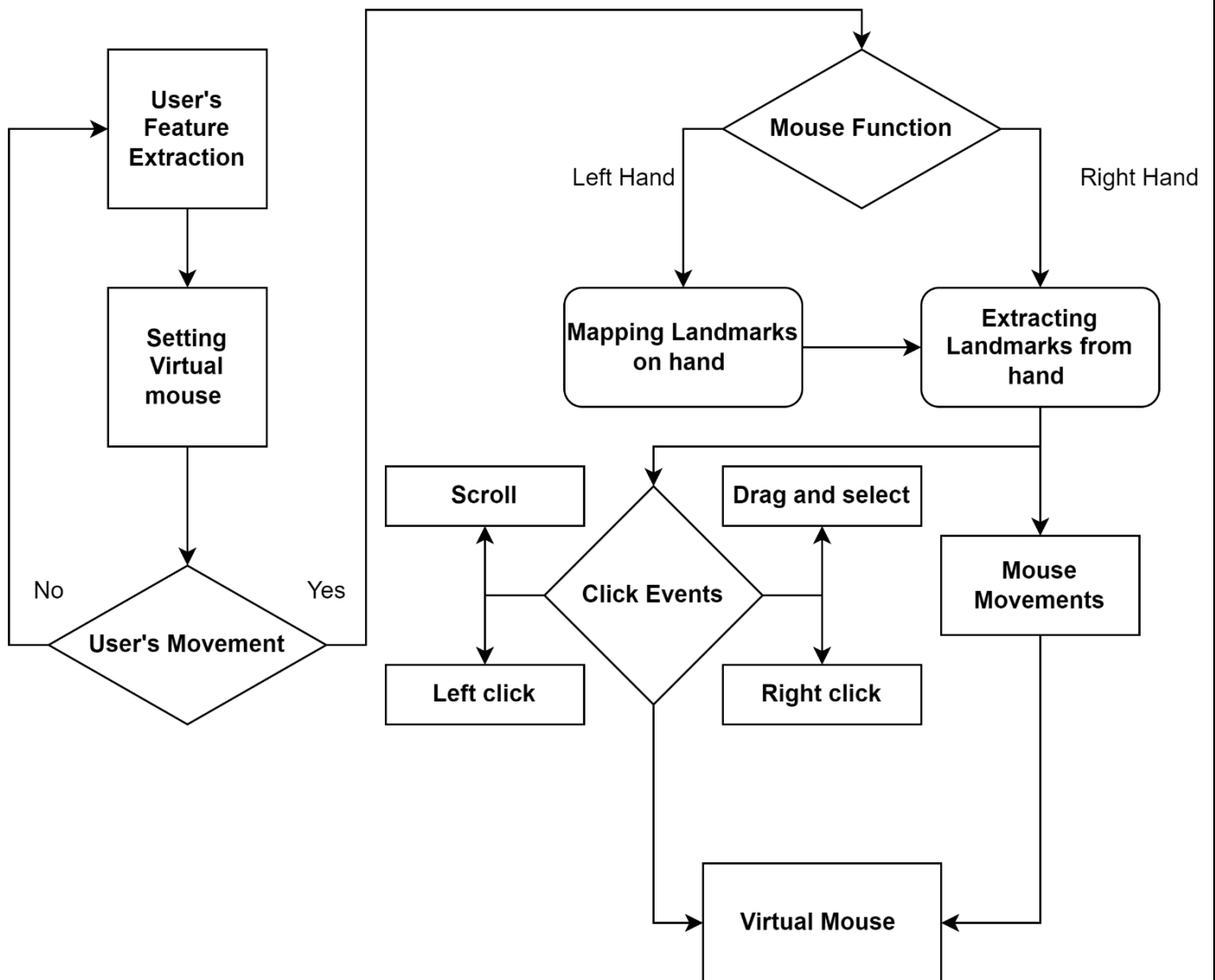


Fig 4:ER Diagram

### 3.3.3 Sequential Diagram:

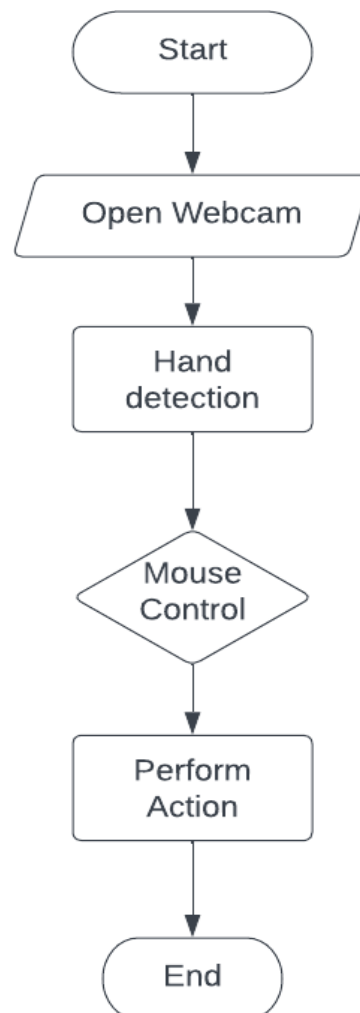


Fig 5: Sequential Diagram

## CHAPTER 4

### SOFTWARE DEVELOPMENT LIFE CYCLE

#### 4.1 SDLC:

SDLC stands for Software Development Life Cycle, which is a process used by software development teams to design, develop, test, and deploy software applications. The SDLC consists of several stages, including planning, analysis, design, implementation, testing, and maintenance.

#### 4.2 PHASES OF SDLC:

A system development life cycle or SDLC is essentially a project management model. It defines different stages that are necessary to bring a project from its initial idea or conception all the way to deployment and later maintenance.

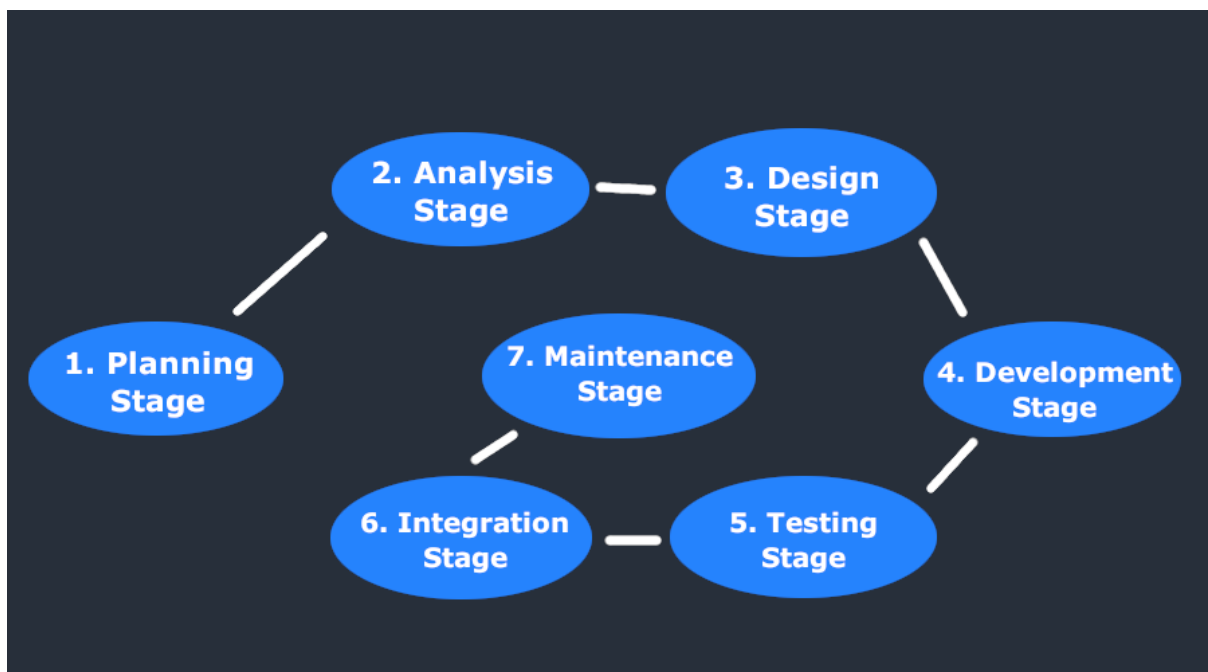


Fig 6: Phases of SDLC

## 7 Stages of the System Development Life Cycle:

There are seven primary stages of the modern system development life cycle. Here's a brief breakdown:

- Planning Stage
- Feasibility or Requirements of Analysis Stage
- Design and Prototyping Stage
- Software Development Stage
- Software Testing Stage
- Implementation and Integration
- Operations and Maintenance Stage

### Planning Stage:

Before we even begin with the planning stage, the best tip we can give you is to take time and acquire proper understanding of app development life cycle. The planning stage (also called the feasibility stage) is exactly what it sounds like: the phase in which developers will plan for the upcoming project. It helps to define the problem and scope of any existing systems, as well as determine the objectives for their new systems. By developing an effective outline for the upcoming development cycle, they'll theoretically catch problems before they affect development. And help to secure the funding and resources they need to make their plan happen. Perhaps most importantly, the planning stage sets the project schedule, which can be of key importance if development is for a commercial product that must be sent to market by a certain time.

### Analysis Stage:

The analysis stage includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes.

Developers may:

- Define any prototype system requirements
- Evaluate alternatives to existing prototypes
- Perform research and analysis to determine the needs of end-users

Furthermore, developers will often create a software requirement specification or SRS document. This includes all the specifications for software, hardware, and network requirements for the system they plan to build. This will prevent them from overdrawing funding or resources when working at the same place as other development teams.

#### Design Stage:

The design stage is a necessary precursor to the main developer stage. Developers will first outline the details for the overall application, alongside specific aspects, such as its:

- User interfaces
- System interfaces
- Network and network requirements
- Databases

They'll typically turn the SRS document they created into a more logical structure that can later be implemented in a programming language. Operation, training, and maintenance plans will all be drawn up so that developers know what they need to do throughout every stage of the cycle moving forward. Once complete, development managers will prepare a design document to be referenced throughout the next phases of the SDLC.

### Development Stage:

The development stage is the part where developers actually write code and build the application according to the earlier design documents and outlined specifications. This is where Static Application Security Testing or SAST tools come into play. Product program code is built per the design document specifications. In theory, all of the prior planning and outlined should make the actual development phase relatively straightforward. Developers will follow any coding guidelines as defined by the organization and utilize different tools such as compilers, debuggers, and interpreters. Programming languages can include staples such as C++, PHP, and more. Developers will choose the right programming code to use based on the project specifications and requirements.

### Testing Stage:

Building software is not the end. Now it must be tested to make sure that there aren't any bugs and that the end-user experience will not negatively be affected at any point. During the testing stage, developers will go over their software with a fine-tooth comb, noting any bugs or defects that need to be tracked, fixed, and later retested. It's important that the software overall ends up meeting the quality standards that were previously defined in the SRS document. Depending on the skill of the developers, the complexity of the software, and the requirements for the end-user, testing can either be an extremely short phase or take a very long time.

### Implementation and Integration Stage:

After testing, the overall design for the software will come together. Different modules or designs will be integrated into the primary source code through developer efforts, usually by leveraging training environments to detect further errors or defects. The information system will be integrated into its environment and eventually installed. After passing this stage, the software is theoretically ready for market and may be provided to any end-users.

### Maintenance Stage:

The SDLC doesn't end when software reaches the market. Developers must now move into a maintenance mode and begin practicing any activities required to handle issues reported by end-users. Furthermore, developers are responsible for implementing any changes that the software might need after deployment.

## CHAPTER 5

### IMPLEMENTATION

The major code for implementation:

```
1  import cv2
2  import mediapipe as mp
3  import pyautogui
4  import math
5  from enum import IntEnum
6  from ctypes import cast, POINTER
7  from comtypes import CLSCTX_ALL
8  from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
9  from google.protobuf.json_format import MessageToDict
10
11  pyautogui.FAILSAFE = False
12  mp_drawing = mp.solutions.drawing_utils
13  mp_hands = mp.solutions.hands
14
15  class Gest(IntEnum):
16      FIST = 0
17      PINKY = 1
18      RING = 2
19      MID = 4
20      LAST3 = 7
21      INDEX = 8
22      FIRST2 = 12
23      LAST4 = 15
24      THUMB = 16
25      PALM = 31
26
27      V_GEST = 33
28      TWO_FINGER_CLOSED = 34
29      PINCH_MAJOR = 35
30      PINCH_MINOR = 36
31
32  class HLabel(IntEnum):
33      MINOR = 0
34      MAJOR = 1
35
36  class HandRecog:
37      def __init__(self, hand_label):
38          self.finger = 0
39          self.ori_gesture = Gest.PALM
40          self.prev_gesture = Gest.PALM
```



```

39     self.curr_gesture = Gest.PALM
40     self.prev_gesture = Gest.PALM
41     self.frame_count = 0
42     self.hand_result = None
43     self.hand_label = hand_label
44
45     def update_hand_result(self, hand_result):
46         self.hand_result = hand_result
47
48     def get_signed_dist(self, point):
49         sign = -1
50         if self.hand_result.landmark[point[0]].y < self.hand_result.landmark[point[1]].y:
51             sign = 1
52         dist = (self.hand_result.landmark[point[0]].x - self.hand_result.landmark[point[1]].x) ** 2
53         dist += (self.hand_result.landmark[point[0]].y - self.hand_result.landmark[point[1]].y) ** 2
54         dist = math.sqrt(dist)
55         return dist * sign
56
57     def get_dist(self, point):
58         dist = (self.hand_result.landmark[point[0]].x - self.hand_result.landmark[point[1]].x) ** 2
59         dist += (self.hand_result.landmark[point[0]].y - self.hand_result.landmark[point[1]].y) ** 2
60         dist = math.sqrt(dist)
61         return dist
62
63     def get_dz(self, point):
64         return abs(self.hand_result.landmark[point[0]].z - self.hand_result.landmark[point[1]].z)
65
66     def set_finger_state(self, dist1=None):
67         if self.hand_result == None:
68             return
69
70         points = [[8, 5, 0], [12, 9, 0], [16, 13, 0], [20, 17, 0]]
71         self.finger = 0
72         self.finger = self.finger | 0 # thumb
73         for idx, point in enumerate(points):
74             dist = self.get_signed_dist(point[:2])
75             dist2 = self.get_signed_dist(point[1:])
76             try:
77                 ratio = round(dist / dist2, 1)
78             except:
79                 ratio = round(dist1 / (0.01 - 1))

```

```

78         except:
79             ratio = round(dist1 / 0.01, 1)
80             self.finger = self.finger << 1
81             if ratio > 0.5:
82                 self.finger = self.finger | 1
83
84     def get_gesture(self):
85         if self.hand_result == None:
86             return Gest.PALM
87         current_gesture = Gest.PALM
88         if self.finger in [Gest.LAST3, Gest.LAST4] and self.get_dist([8, 4]) < 0.05:
89             if self.hand_label == HLabel.MINOR:
90                 current_gesture = Gest.PINCH_MINOR
91             else:
92                 current_gesture = Gest.PINCH_MAJOR
93         elif Gest.FIRST2 == self.finger:
94             point = [[8, 12], [5, 9]]
95             dist1 = self.get_dist(point[0])
96             dist2 = self.get_dist(point[1])
97             ratio = dist1 / dist2
98             if ratio > 1.7:
99                 current_gesture = Gest.V_GEST
100             else:
101                 if self.get_dz([8, 12]) < 0.1:
102                     current_gesture = Gest.TWO_FINGER_CLOSED
103                 else:
104                     current_gesture = Gest.MID
105         else:
106             current_gesture = self.finger
107         if current_gesture == self.prev_gesture:
108             self.frame_count += 1
109         else:
110             self.frame_count = 0
111         self.prev_gesture = current_gesture
112         if self.frame_count > 4:
113             self.ori_gesture = current_gesture
114         return self.ori_gesture
115

```

```

115
116 class Controller:
117     tx_old = 0
118     ty_old = 0
119     trial = True
120     flag = False
121     grabflag = False
122     pinchmajorflag = False
123     pinchminorflag = False
124     pinchstartxcoord = None
125     pinchstartycoord = None
126     pinchdirectionflag = None
127     prevpinchlv = 0
128     pinchlv = 0
129     framecount = 0
130     prev_hand = None
131     pinch_threshold = 0.3
132
133     def getpinchylv(hand_result):
134         dist = round((Controller.pinchstartycoord - hand_result.landmark[8].y) * 10, 1)
135         return dist
136
137     def getpinchxlv(hand_result):
138         dist = round((hand_result.landmark[8].x - Controller.pinchstartxcoord) * 10, 1)
139         return dist
140
141     def changesystemvolume():
142         devices = AudioUtilities.GetSpeakers()
143         interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
144         volume = cast(interface, POINTER(IAudioEndpointVolume))
145         currentVolumeLv = volume.GetMasterVolumeLevelScalar()
146         currentVolumeLv += Controller.pinchlv / 50.0
147         if currentVolumeLv > 1.0:
148             currentVolumeLv = 1.0
149         elif currentVolumeLv < 0.0:
150             currentVolumeLv = 0.0
151         volume.SetMasterVolumeLevelScalar(currentVolumeLv, None)
152

```

```

257
258 class GestureController:
259     gc_mode = 0
260     cap = None
261     CAM_HEIGHT = None
262     CAM_WIDTH = None
263     hr_major = None # Right Hand by default
264     hr_minor = None # Left hand by default
265     dom_hand = True
266
267     def __init__(self):
268         GestureController.gc_mode = 1
269         GestureController.cap = cv2.VideoCapture(0)
270         GestureController.CAM_HEIGHT = GestureController.cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
271         GestureController.CAM_WIDTH = GestureController.cap.get(cv2.CAP_PROP_FRAME_WIDTH)
272
273     def classify_hands(results):
274         left, right = None, None
275         try:
276             handedness_dict = MessageToDict(results.multi_handedness[0])
277             if handedness_dict['classification'][0]['label'] == 'Right':
278                 right = results.multi_hand_landmarks[0]
279             else:
280                 left = results.multi_hand_landmarks[0]
281         except:
282             pass
283         try:
284             handedness_dict = MessageToDict(results.multi_handedness[1])
285             if handedness_dict['classification'][0]['label'] == 'Right':
286                 right = results.multi_hand_landmarks[1]
287             else:
288                 left = results.multi_hand_landmarks[1]
289         except:
290             pass
291         if GestureController.dom_hand == True:
292             GestureController.hr_major = right
293             GestureController.hr_minor = left
294         else:
295             GestureController.hr_major = left
296             GestureController.hr_minor = right

```

```

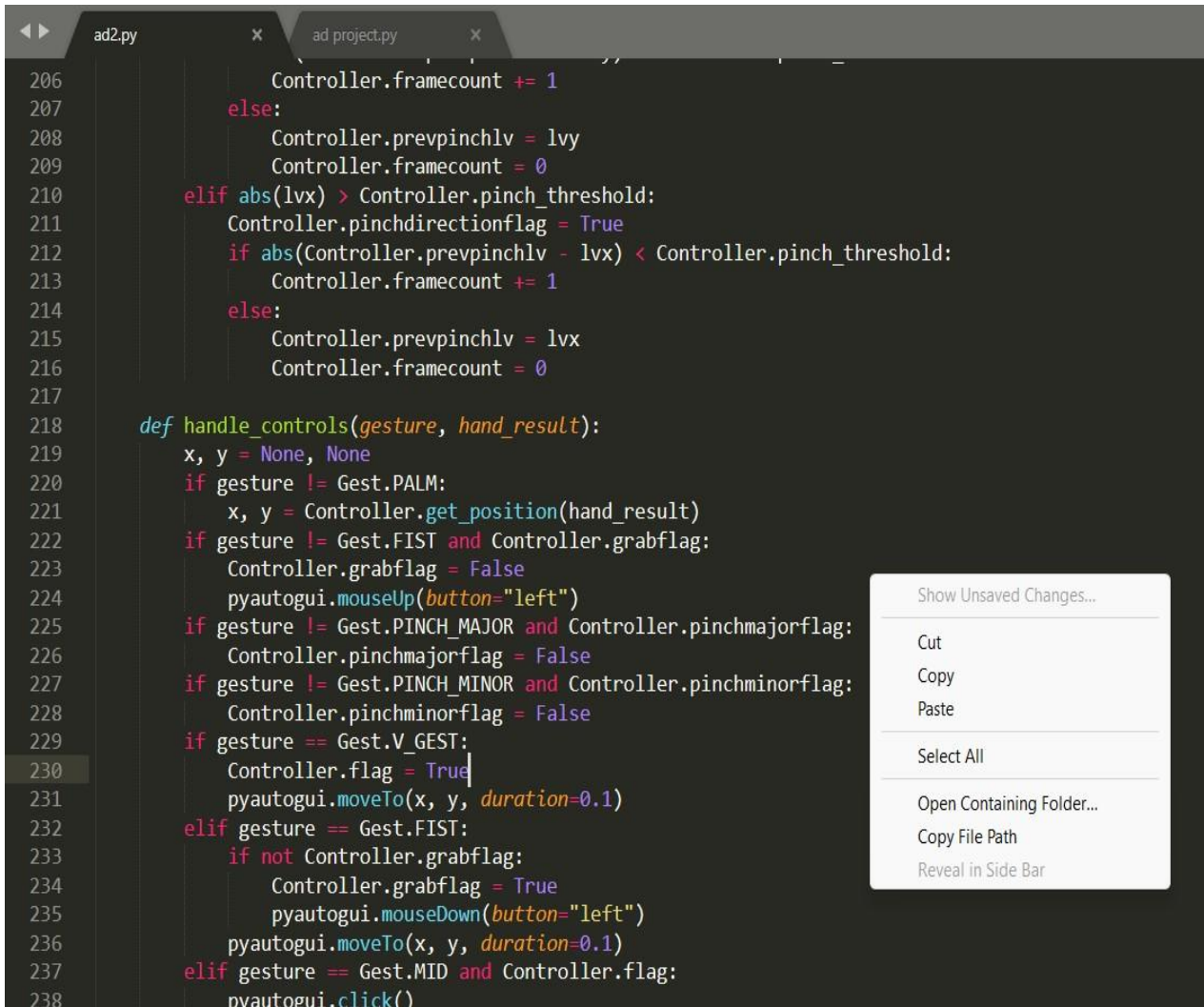
296         GestureController.hr_minor = right
297
298     def start(self):
299         handmajor = HandRecog(HLabel.MAJOR)
300         handminor = HandRecog(HLabel.MINOR)
301         with mp_hands.Hands(max_num_hands=2, min_detection_confidence=0.5, min_tracking_confidence=0.5) as hands:
302             while GestureController.cap.isOpened() and GestureController.gc_mode:
303                 success, image = GestureController.cap.read()
304                 if not success:
305                     print("Ignoring empty camera frame.")
306                     continue
307                 image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)
308                 image.flags.writeable = False
309                 results = hands.process(image)
310                 image.flags.writeable = True
311                 image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
312                 if results.multi_hand_landmarks:
313                     GestureController.classify_hands(results)
314                     handmajor.update_hand_result(GestureController.hr_major)
315                     handminor.update_hand_result(GestureController.hr_minor)
316                     handmajor.set_finger_state()
317                     handminor.set_finger_state()
318                     gest_name = handminor.get_gesture()
319                     if gest_name == Gest.PINCH_MINOR:
320                         Controller.handle_controls(gest_name, handminor.hand_result)
321                     else:
322                         gest_name = handmajor.get_gesture()
323                         Controller.handle_controls(gest_name, handmajor.hand_result)
324                     for hand_landmarks in results.multi_hand_landmarks:
325                         mp_drawing.draw_landmarks(image, hand_landmarks, mp_hands.HAND_CONNECTIONS)
326                 else:
327                     Controller.prev_hand = None
328                     cv2.imshow('Virtual Mouse', image)
329                     if cv2.waitKey(5) & 0xFF == 13:
330                         break
331             GestureController.cap.release()
332             cv2.destroyAllWindows()
333 gc1 = GestureController()
334 gc1.start()

```

## CHAPTER 6

### TESTING

#### 6.1 TEST CASE FOR RIGHT CLICK:



```
206         Controller.framecount += 1
207     else:
208         Controller.prevpinchlv = lvy
209         Controller.framecount = 0
210 elif abs(lvx) > Controller.pinch_threshold:
211     Controller.pinchdirectionflag = True
212     if abs(Controller.prevpinchlv - lvx) < Controller.pinch_threshold:
213         Controller.framecount += 1
214     else:
215         Controller.prevpinchlv = lvx
216         Controller.framecount = 0
217
218 def handle_controls(gesture, hand_result):
219     x, y = None, None
220     if gesture != Gest.PALM:
221         x, y = Controller.get_position(hand_result)
222     if gesture != Gest.FIST and Controller.grabflag:
223         Controller.grabflag = False
224         pyautogui.mouseUp(button="left")
225     if gesture != Gest.PINCH_MAJOR and Controller.pinchmajorflag:
226         Controller.pinchmajorflag = False
227     if gesture != Gest.PINCH_MINOR and Controller.pinchminorflag:
228         Controller.pinchminorflag = False
229     if gesture == Gest.V_GEST:
230         Controller.flag = True
231         pyautogui.moveTo(x, y, duration=0.1)
232     elif gesture == Gest.FIST:
233         if not Controller.grabflag:
234             Controller.grabflag = True
235             pyautogui.mouseDown(button="left")
236             pyautogui.moveTo(x, y, duration=0.1)
237     elif gesture == Gest.MID and Controller.flag:
238         pyautogui.click()
```

Context menu options:

- Show Unsaved Changes...
- Cut
- Copy
- Paste
- Select All
- Open Containing Folder...
- Copy File Path
- Reveal in Side Bar

Fig 7: Test case for right click

## 6.2 TEST CASE FOR SELECTION:

```
file  edit  selection  find  view  goto  tools  project  preferences  help
ad2.py  x  ad project.py  x
205     if abs(Controller.prevpinchlv - lvy) < Controller.pinch_threshold:
206         Controller.framecount += 1
207     else:
208         Controller.prevpinchlv = lvy
209         Controller.framecount = 0
210     elif abs(lvx) > Controller.pinch_threshold:
211         Controller.pinchdirectionflag = True
212         if abs(Controller.prevpinchlv - lvx) < Controller.pinch_threshold:
213             Controller.framecount += 1
214         else:
215             Controller.prevpinchlv = lvx
216             Controller.framecount = 0
217
218     def handle_controls(gesture, hand_result):
219         x, y = None, None
220         if gesture != Gest.PALM:
221             x, y = Controller.get_position(hand_result)
222         if gesture != Gest.FIST and Controller.grabflag:
223             Controller.grabflag = False
224             pyautogui.mouseUp(button="left")
225         if gesture != Gest.PINCH_MAJOR and Controller.pinchmajorflag:
226             Controller.pinchmajorflag = False
227         if gesture != Gest.PINCH_MINOR and Controller.pinchminorflag:
228             Controller.pinchminorflag = False
229         if gesture == Gest.V_GEST:
230             Controller.flag = True
231             pyautogui.moveTo(x, y, duration=0.1)
232         elif gesture == Gest.FIST:
233             if not Controller.grabflag:
234                 Controller.grabflag = True
235                 pyautogui.mouseDown(button="left")
236                 pyautogui.moveTo(x, y, duration=0.1)
237         elif gesture == Gest.MTD and Controller.flag:
```

Fig 8: Test case for selection



### **6.3 TEST CASE FOR DETECTION:**

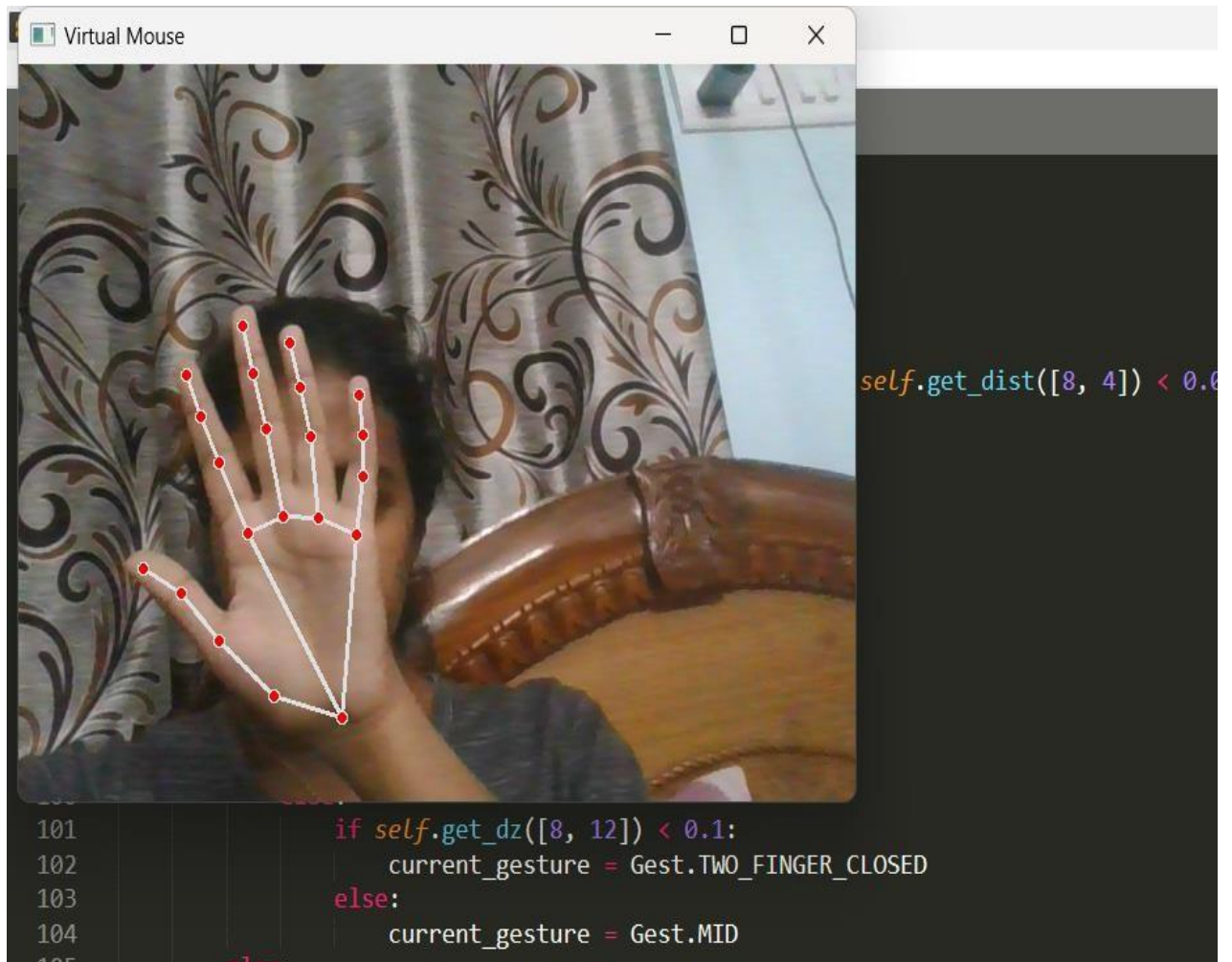


Fig 9: Test case for detection



#### **6.4 TEST CASE FOR OPENING DIALOGUE BOX:**

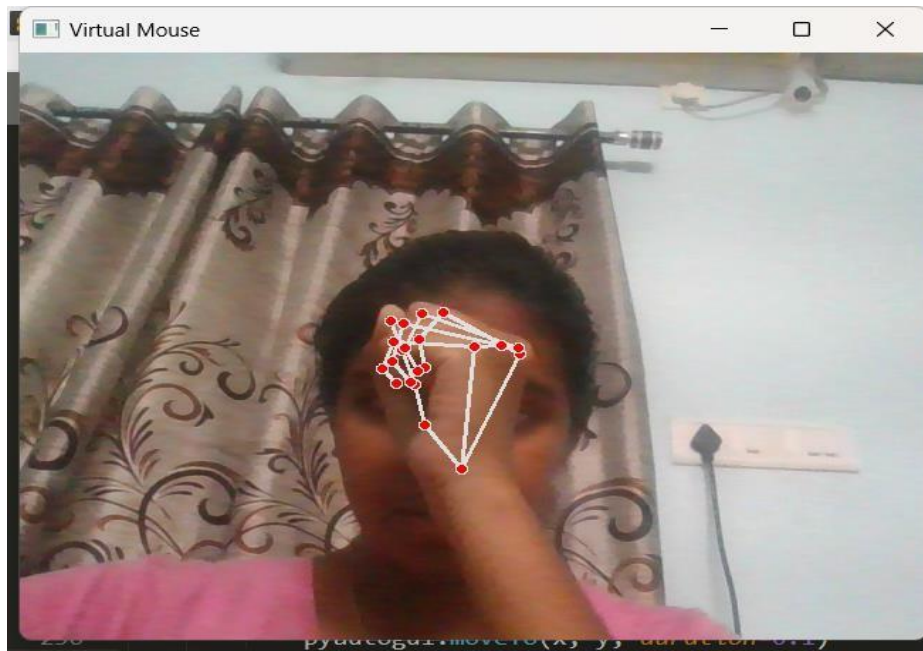


Fig 10: Test case for opening dialogue box

### **6.5 OVERALL TEST CASES:**

S No.	Functional Test Cases	Type-Negative or positive test case
1	Verify weather user can see the video capturing dialog box.	Positive
2	Verify weather web cam detects the hand.	Positive
3	Verify weather left click operation is preformed.	Positive
4	Verify weather right click operation is performed.	Positive
5	Verify weather any operation is performed when all fingers are up.	Negative
6	Verify weather scroll operation is performed.	Positive

Fig 11: Overall test cases

## **CHAPTER 7**

### **CONCLUSION AND FUTURE SCOPE**

#### **7.1 CONCLUSION**

- In this project, we made attempt to effectively introduce the concept of replacement of the physical mouse.
- We then described the proposed system and explained the features implemented by our proposed system .
- The proposed system architecture will completely change the way people would use the computer system.
- A few techniques had to be implemented because accuracy and efficiency play an important role in making the program.

#### **7.2 FUTURE SCOPE**

- With this gesture recognition system, we can use computers without any help from physical devices such as mouse.
- This technology can make work easy as just one big screen can be used by researchers to do all the work.

## **CHAPTER 8**

### **REFERENCES**

- A Simple Algorithm for using Face as a Pointing Device using OpenCV V. Pradeep International Journal of Advanced Research in Computer Science and Software Engineering Volume 2, Issue 2, February 2012 ISSN: 2277 128X.
- Rafiqul Zaman Khan, Noor Ibraheem, “Hand Gesture Recognition: A Literature Review”, International Journal of Artificial Intelligence & Applications,10.5121/ijaia.2012.3412 , August 2012.
- Kollipara Sai Varun, I. Puneeth, T. Prem Jacob, “Hand Gesture Recognition and Implementation for Disables using CNN’S”, 2019 International Conference on Communication and Signal Processing (ICCSP), 10.1109/ICCSP.2019.8697980,25 April 2019.