**Index of Contents**

**Chapter 1 : Introduction**

**1.1 Background**

**1.2 Problem definition**

**1.3 Scope**

**1.4 Need to define the problem**

**Chapter 2 : Literature Survey**

**2.1 Links referred**

**Chapter 3: Project Design**

**3.1 Software development tools**

**3.2 Module wise flow diagram**

**3.3 System Architecture**

**3.4 External modules**

**Chapter 4 : Implementation and Testing of project work/ topic**

**4.1 Code**

**4.2 Function implementation**

**4.3 Testing Phases**

**4.4 Responsibilities**

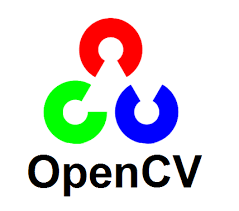
**4.5 Output/Interface**

**4.6 Github implementation**

**Chapter 5 : Conclusions and further work**

**Chapter 1**

* 1. **Background**



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.

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today’s systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

**1.2 Problem Definition:**

## To make a virtual application menu that opens different apps based on the location and configuration of our fingers. The menu must open different apps when we point at different places in the webcam with our index finger and middle finger raised.

**1.3 Scope**

* On running the code, the webcam output will be shown through a new window.
* On detection of hands, the hand landmarks are continuously highlighted on the window
* On raising of the middle and index fingers, boxes are made on the tip of the fingers to indicate Selection mode
* When user guides hands to different parts of the window, different apps are opened

**1.4 Need to define this problem**

The desktop computing paradigm limits the users' flexibility by forcing them to interact using a 2-Degree-Of-Freedom device (the mouse), while they are used to interacting with the physical world in much more differentiated ways (Bellucci, Malizia & Aedo, 2014). Gestures allow the user to handle multiple points of input and even define several parameters at once. They are, therefore, a more natural form of communication.

### Immediate and powerful interaction

Unlike traditional buttons and menus, gestures do not interrupt the user's activity by forcing him to move his hand to the location of a command. Instead, they can be performed directly from the current cursor position. (Bau & Mackay, 2008)

Also, they do not require any additional devices: the command and even its parameters can be specified by a simple hand movement (Baudel & Beaudouin-Lafon, 1993). Input devices narrow down the user's possibilities of interaction, for example a pen or a mouse limiting the potential forms of input to single-touch interaction. Gestures that are performed with the user's hands however, can be versatile and do not have these constraints. As Wobbrock et al. put it: "almost anything one can do with one's hands could be a potential gesture" (Wobbrock, Morris & Wilson, 2009). This includes not only the movement or the followed path of the hand, but the movement and position of every finger as well as the general hand posture. (Brandl, Forlines, Wigdor, Haller & Shen, 2008)

### Intuitiveness and enjoyability

Gestures feel very natural to perform since they mirror our experiences in the real world.

Maybe that is the reason a study by Watson et al. showed that participants using touch-input for a task were enjoying themselves more and also felt more competent compared to participants using a mouse. They systematically favoured direct touch input over mouse input and also performed better regarding speed and accuracy. (Watson, Hancock, Mandryk & Birk, 2013)

In addition, Cao, Ofek and Vronay found that gesture-controlled presentations were not only perceived as more enjoyable by the presenter but also as more attractive by the audience. The presenters were able to make eye contact more often and to use their body language to convey information. (2005)

**Chapter 2 Literature survey**

**2.1 Links referred:**

[[2006.10214] MediaPipe Hands: On-device Real-time Hand Tracking (arxiv.org)](https://arxiv.org/abs/2006.10214)

Abstract:-

We present a real-time on-device hand tracking solution that predicts a hand skeleton of a human from a single RGB camera for AR/VR applications. Our pipeline consists of two models: 1) a palm detector, that is providing a bounding box of a hand to, 2) a hand landmark model, that is predicting the hand skeleton. It is implemented via MediaPipe[12], a framework for building cross-platform ML solutions. The proposed model and pipeline architecture demonstrate real-time inference speed on mobile GPUs with high prediction quality. MediaPipe Hands is open sourced at https://mediapipe.dev.

We will be working on hand detection using OpenCv and

Some more links that were helpful to us in the making of this project:

[OpenCV: OpenCV modules](https://docs.opencv.org/4.x/)

[Hands - mediapipe (google.github.io)](https://google.github.io/mediapipe/solutions/hands.html)

[OpenCV - Overview - GeeksforGeeks](https://www.geeksforgeeks.org/opencv-overview/)

**Chapter 3 - Project Design**

## 3.1 *Programming languages*

Programming language – Python

Here is a list of the advantages of using Python for Computer Vision:

* Ease of coding
  + [“Code as plain English”](https://en.wikipedia.org/wiki/Python_(programming_language)) is Python’s primary goal. This allows programmers to focus on the design and not on coding. This is perfect for those who are just getting started with machine learning or basic programming. This advantage is very beneficial, especially when faced with complex scenarios.
* Fast prototyping
  + Since you can focus more on the design, you can now experiment with more design ideas. Python is well-suited for implementing new features. Libraries like OpenCV are written in C++ and make Python have slower runtime as it will still call C/C++ libraries. This means you will have the development advantage from Python while you can have performance optimization from C++.
* Vast libraries for machine learning
  + Python is commonly used for machine learning. Data scientists invest their time contributing since it’s easy to code, and it’s free. CV developers don’t have to worry much about projects that they’re working on since most of their cases are already covered by Python libraries.
* It is open source
  + Python is free, unlike MATLAB, which also specializes in data analysis, exploration, visualization, etc. Needless to say, for Python, all you need is a computer, and you are good to go. You can even deploy your work for free on sites like [python anywhere](https://www.pythonanywhere.com/).
* It can be directly integrated with web frameworks
  + Python has mature web frameworks like [Django](https://www.djangoproject.com/). It aims for fast development time, neat and realistic designs. Also, Python has micro frameworks that are just as functional as their larger counterparts.
* [Most commonly used](https://www.quora.com/What-is-the-best-programming-language-to-use-for-algorithms-machine-learning-computer-vision#__w2_wEYHkwQF9_answer_content)
  + This means it has a bigger community. There are a lot of blog posts and online resources regarding Python + OpenCV, so you can always get help most of the time trying to fix a problem.

## Software development tools:

1. Microsoft Edge
2. Microsoft Word
3. Draw.io
4. Pycharm IDE

## Features of PyCharm

### Intelligent Code Editor

PyCharm comes with a smart code editor that facilitates writing high-quality Python code. It offers an enhanced level of code comprehension and readability by means of distinct color schemes for keywords, classes, and functions, i.e., syntax and error highlighting.

In addition to offering the smart code completion feature, the code editor generates instructions for completing the current code. Identifying errors and issues is much more comfortable, along with linter integration and quick fixes.

### Data Science and Machine Learning [Professional Edition Only]

PyCharm comes with support for scientific libraries, such as Matplotlib and SciPy, to help Python developers accomplish data science and machine learning projects.

### Integrated Debugging and Testing

An IDE comes with support for debugging and testing programs. To accomplish the same, PyCharm features an integrated Python debugger and integrated unit testing with line-by-line code coverage.

### Multi-technology Development [Professional Edition Only]

Python developers can also use PyCharm for creating web applications. As such, the Python IDE provides support for popular web technologies, including CoffeeScript, CSS, HTML, JavaScript, TypeScript. Additionally, it also includes support for Cython, template languages, and SQL.

Live editing is also available in PyCharm, i.e., developers can create/modify a web page while pushing it live simultaneously. Hence, changes can be followed directly on a web browser. Building [web applications](https://hackr.io/blog/web-application-architecture-definition-models-types-and-more) using AngularJS or NodeJS is also available.

### Project and Code Navigation

The code navigation feature makes it much easier for developers to navigate to a class, function, or file. It also helps in significantly cutting-down effort and time required to edit and enhance the Python code. File structure views and specialized project views are readily available.

The lens mode allows a developer to inspect and debug the entire Python source code thoroughly. With code navigation, locating an element, variable, etc. is done in almost no time. Developers can quickly jump between classes, files, and methods.

### Refactoring

The refactoring feature in PyCharm helps in improving the internal structure of a Python program without affecting the external performance of the same. Making changes to both local and global variables is efficient and fast.

The extract method is also there to split up extended classes and functions. Other useful code refactoring features include:

* Introduce constant
* Introduce variable
* Pull up
* Push down
* Rename

### 

### Support for Popular Python Web Frameworks [Professional Edition Only]

PyCharm lets developers leverage Django in their Python development projects. The Python IDE offers the autocomplete feature and generates suggestions for Django.

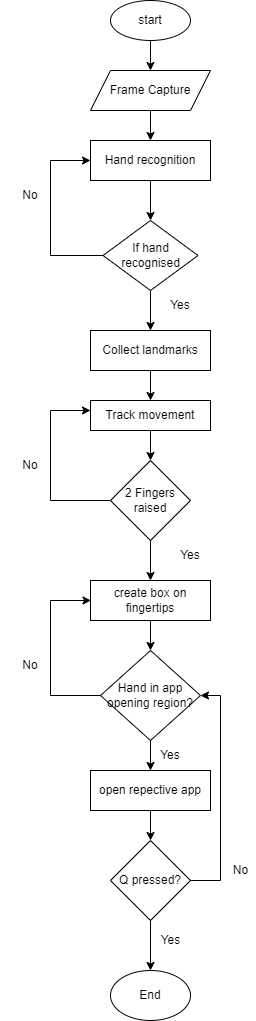
Debugging code written using Django is also available. PyCharm also provides support for other popular Python frameworks, namely Flask, Pyramid, and web2py.

### Version Control Systems (VCSs) Integration

In its simplicity, a version control system (VCS) keeps track of the changes made to files, applications, and other sources of information. It can be considered as a database of changes.

PyCharm provides a unified user interface for CVS, Git, Mercurial, Perforce, and Subversion.

**3.2 Flow Diagram:**

****

**3.3 System Architecture**



**3.4 External modules:**

Python: Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small- and large-scale projects.

Opencv: OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more. It supports multiple languages including python, java, C++

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.

Keras: It is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML

Numpy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays

Tensorflow: TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural network

**Chapter 4 - Implementation and Testing of project**

**4.1 Code:**

**HandTrackingModule.py**

**import cv2**

**import mediapipe as mp**

**import time**

**class handDetector():**

**def \_\_init\_\_(self, mode=False, maxHands=2,modelComplexity=1,detectionCon=0.5, trackCon=0.5):**

**self.mode = mode**

**self.maxHands = maxHands**

**self.detectionCon = detectionCon**

**self.trackCon = trackCon**

**self.modelComplex=modelComplexity**

**self.mpHands = mp.solutions.hands**

**self.hands = self.mpHands.Hands(self.mode, self.maxHands,self.modelComplex, self.detectionCon, self.trackCon)**

**self.mpDraw = mp.solutions.drawing\_utils**

**self.tipIds = [4, 8, 12, 16, 20]**

**def findHands(self, img, draw=True):**

**imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)**

**self.results = self.hands.process(imgRGB)**

**# print(results.multi\_hand\_landmarks)**

**if self.results.multi\_hand\_landmarks:**

**for handLms in self.results.multi\_hand\_landmarks:**

**if draw:**

**self.mpDraw.draw\_landmarks(img, handLms, self.mpHands.HAND\_CONNECTIONS)**

**return img**

**def findPosition(self, img, handNo=0, draw=True):**

**xList = []**

**yList = []**

**bbox = []**

**self.lmList = []**

**if self.results.multi\_hand\_landmarks:**

**myHand = self.results.multi\_hand\_landmarks[handNo]**

**for id, lm in enumerate(myHand.landmark):**

**# print(id, lm)**

**h, w, c = img.shape**

**cx, cy = int(lm.x \* w), int(lm.y \* h)**

**xList.append(cx)**

**yList.append(cy)**

**print(id, cx, cy)**

**self.lmList.append([id, cx, cy])**

**if draw:**

**cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)**

**xmin, xmax = min(xList,default='0'), max(xList,default='0')**

**ymin, ymax = min(yList,default='0'), max(yList,default='0')**

**bbox = xmin, ymin, xmax, ymax**

**if draw:**

**cv2.rectangle(img, (int(xmin) - 20, int(ymin) - 20), (int(xmax) + 20, int(ymax) + 20),**

**(0, 255, 0), 2)**

**return self.lmList, bbox**

**def fingersUp(self):**

**fingers = []**

**if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:**

**fingers.append(1)**

**else:**

**fingers.append(0)**

**for id in range(1, 5):**

**if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:**

**fingers.append(1)**

**else:**

**fingers.append(0)**

**return fingers**

**def main():**

**pTime = 0**

**cTime = 0**

**cap = cv2.VideoCapture(0)**

**detector = handDetector()**

**while True:**

**success, img = cap.read()**

**img = detector.findHands(img)**

**lmlist,bbox = detector.findPosition(img)**

**if len(lmlist) != 0:**

**print(lmlist[4])**

**cTime = time.time()**

**fps = 1 / (cTime - pTime)**

**pTime = cTime**

**cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT\_HERSHEY\_PLAIN, 3, (255, 0, 255), 3)**

**cv2.imshow("Image", img)**

**cv2.waitKey(1)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**app-opener.py**

import cv2

import numpy as np

import time

import os

import HandTrackingModule as htm

import webbrowser

brushThickness = 10

eraserThickness = 100

drawColor = (255, 0, 255)

cap = cv2.VideoCapture(0)

cap.set(3, 1280)

cap.set(4, 720)

detector = htm.handDetector(detectionCon=0.65, maxHands=1)

xp, yp = 0, 0

imgCanvas = np.zeros((720, 1280, 3), np.uint8)

while True:

success, img = cap.read()

img = cv2.flip(img, 1)

cv2.putText(img, 'Spotify', (800, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 1, cv2.LINE\_AA, False)

cv2.putText(img, 'Telegram', (550 ,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 1, cv2.LINE\_AA, False)

cv2.putText(img, 'Discord', (1050, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 1, cv2.LINE\_AA, False)

cv2.putText(img, 'Teams', (250, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 1, cv2.LINE\_AA, False)

img = detector.findHands(img)

lmList, bbox = detector.findPosition(img, draw=False)

if len(lmList) != 0:

print(len(lmList))

x1, y1 = lmList[8][1:]

x2, y2 = lmList[12][1:]

fingers = detector.fingersUp()

# print(fingers)

if fingers[1] and fingers[2]:

print("Selection Mode")

if y1 < 125:

if 250 < x1 < 450:

os.startfile('"C:/Users/kushl/OneDrive/Desktop/Microsoft Teams (work or school).lnk"')

print("Opening Teams...")

elif 550 < x1 < 750:

os.startfile('"C:/Users/kushl/OneDrive/Desktop/Telegram.lnk"')

print("Opening Telegram...")

elif 800 < x1 < 950:

os.startfile('C:/Users/kushl/AppData/Roaming/Spotify/Spotify.exe')

print("Opening Spotify...")

elif 1050 < x1 < 1200:

flag=0

if(flag==0):

os.startfile('"C:/Users/kushl/OneDrive/Desktop/Discord.lnk"')

print("Opening Discord...")

flag=1

cv2.rectangle(img, (x1, y1 - 25), (x2, y2 + 25), drawColor, cv2.FILLED)

imgGray = cv2.cvtColor(imgCanvas, cv2.COLOR\_BGR2GRAY)

\_, imgInv = cv2.threshold(imgGray, 50, 255, cv2.THRESH\_BINARY\_INV)

imgInv = cv2.cvtColor(imgInv, cv2.COLOR\_GRAY2BGR)

img = cv2.bitwise\_and(img, imgInv)

img = cv2.bitwise\_or(img, imgCanvas)

cv2.imshow("Image", img)

if (cv2.waitKey(30) == ord('q')):

file\_name\_path = 'C:/Users/kushl/PycharmProjects/MIniProject/' + '.jpg'

cv2.imwrite(file\_name\_path, imgInv)

break

cap.release()

**4.2 Functions Implemented :**

Major functions:

findPosition(): This function takes the coordinates of the hand landmarks. It is used to detect where our hand currently is with respect to the webcam output window coordinates. This function decides which app to open up

findHands(): This function takes in the webcam video and finds if there are any hands present, and if yes, draws landmarks on the hand as per the mediapipe hand tracking module, which has 17 different landmarks on different parts of the hand

FingersUp(): This function detects how many fingers are held up at a time. This is used to toggle selection mode which is turned on when we hold up the middle and index finger together.

Minor functions:

cv2.putText(frame, Text, org, font, color, thickness)

Parameters:

frame: current running frame of the video.

Text: The text string to be inserted.

org: bottom-left corner of the text string

font: the type of font to be used.

color: the colour of the font.

thickness: the thickness of the font

cv2.rectangle(image, start\_point, end\_point, color, thickness)

Parameters:

image: It is the image on which rectangle is to be drawn.

start\_point: It is the starting coordinates of rectangle. The coordinates are represented as tuples of two values i.e. (X coordinate value, Y coordinate value).

end\_point: It is the ending coordinates of rectangle. The coordinates are represented as tuples of two values i.e. (X coordinate value, Y coordinate value).

color: It is the color of border line of rectangle to be drawn. For BGR, we pass a tuple. eg: (255, 0, 0) for blue color.

thickness: It is the thickness of the rectangle border line in px. Thickness of -1 px will fill the rectangle shape by the specified color.

Return Value: It returns an image.

cv2.cvtColor(src, code[, dst[, dstCn]])

Parameters:

src: It is the image whose color space is to be changed.

code: It is the color space conversion code.

dst: It is the output image of the same size and depth as src image. It is an optional parameter.

dstCn: It is the number of channels in the destination image. If the parameter is 0 then the number of the channels is derived automatically from src and code. It is an optional parameter.

Return Value: It returns an image.

Cv2.threshold: The first argument is the source image, which **should be a grayscale image**. The second argument is the threshold value which is used to classify the pixel values. The third argument is the maximum value which is assigned to pixel values exceeding the threshold. OpenCV provides different types of thresholding which is given by the fourth parameter of the function.

**4.3 Test plan :**

| **Test case** | **Description** | **Intended result** | **Actual result** | **Completed by** |
| --- | --- | --- | --- | --- |
| 1 | Font size of menu | Should have been smaller | App names were overlapping | Kaushik Iyer |
| 2 | Multiple Edge tabs | Opening of one Edge window | Opening of too many edge windows for each frame detected | Vinod Ghanchi |
| 3 | Multiple Hands | Should detect hand nearest to webcam | Detected random hand in the background | Vinod Ghanchi |
| 4 | Opening discord | Discord should have opened directly | Discord had an update check earlier and started updating first instead of opening | Kaushik Iyer |
| 5 | Quick launch of applications | Apps should launch instantly when already in background | Apps launched instantly when already in background | Kaushik Iyer |
| 6 | Apps Menu | Proper spacing between application choices | Proper spacing between application choices | Vinod Ghanchi |

**Major problems :**

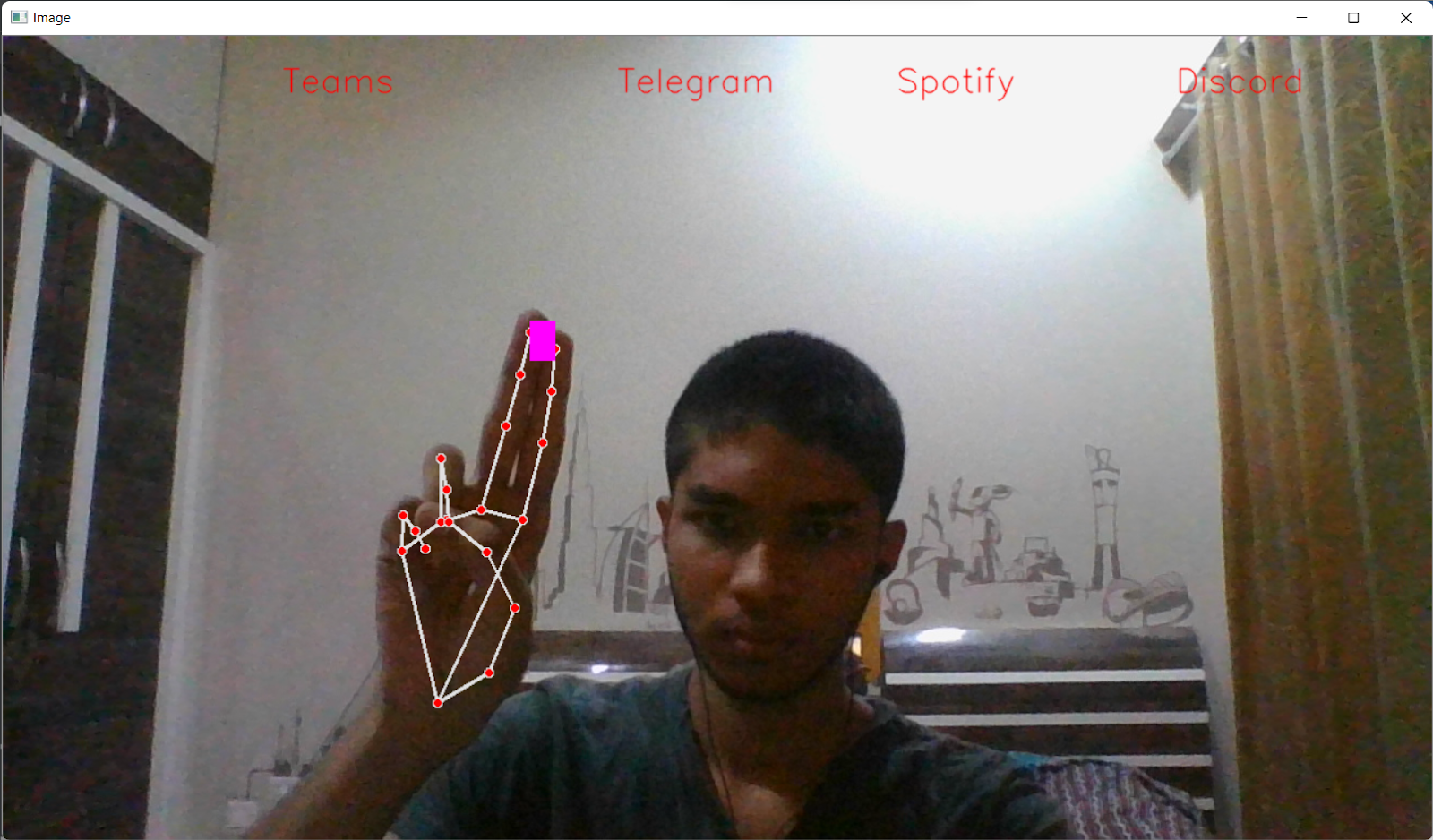
* Tackling the problem of opening multiple instances of the same application
* Detecting of unneeded hands during multiple occasions
* Hands not being detected when shown at an angle

**4.4 Responsibilities :**

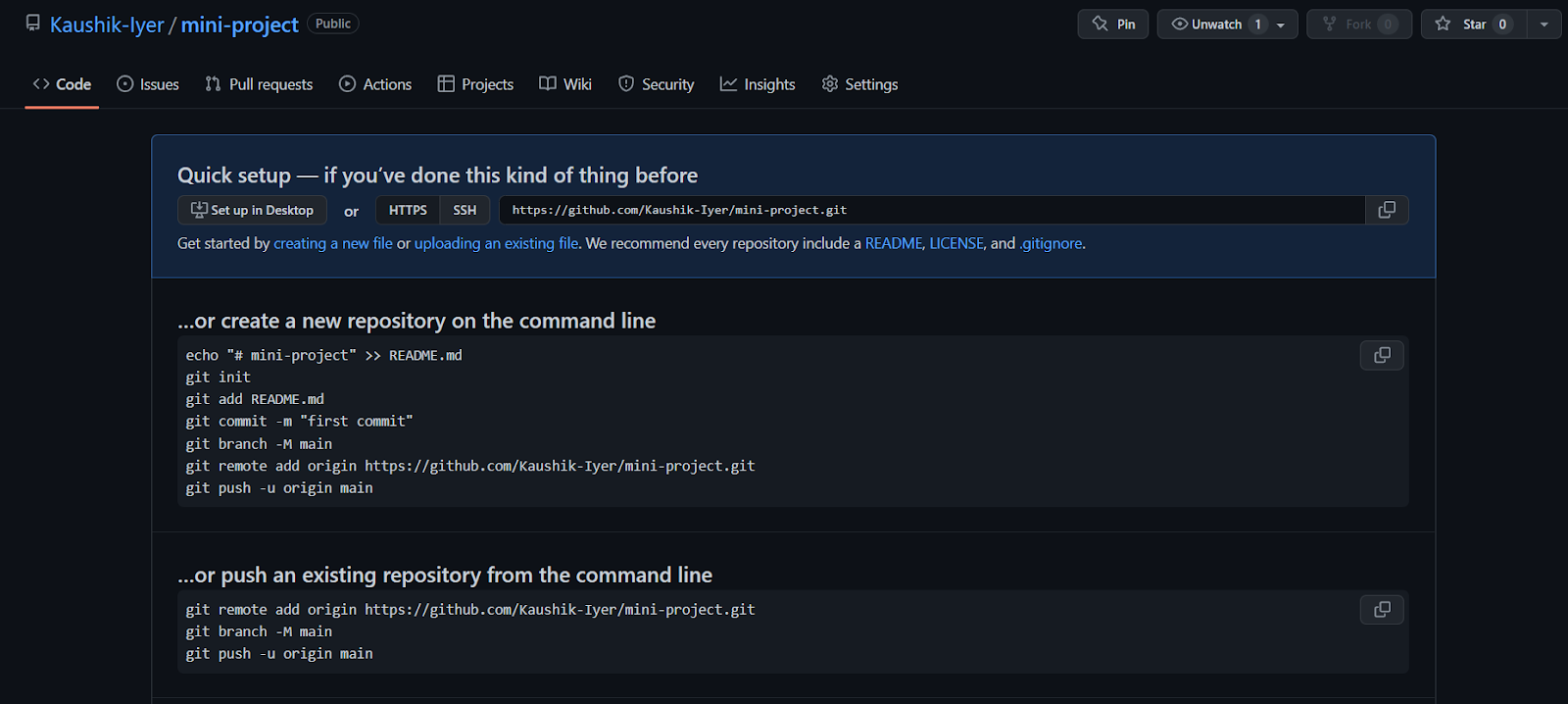
| **Task** | **Kaushik** | **Vinod** |
| --- | --- | --- |
| **UI :** |  |  |
| **Design** | X |  |
| **Coding** | X | X |
| **Menu Creation** |  | X |
| **Program:** |  |  |
| **Application Choice** |  | X |
| **Application Testing** | X |  |
| **Testing :** |  |  |
| **Testing approach** |  | X |
| **Test Cases** | X |  |
| **Presentation:** |  |  |
| **Powerpoint** | X | X |
| **Report** | X | X |

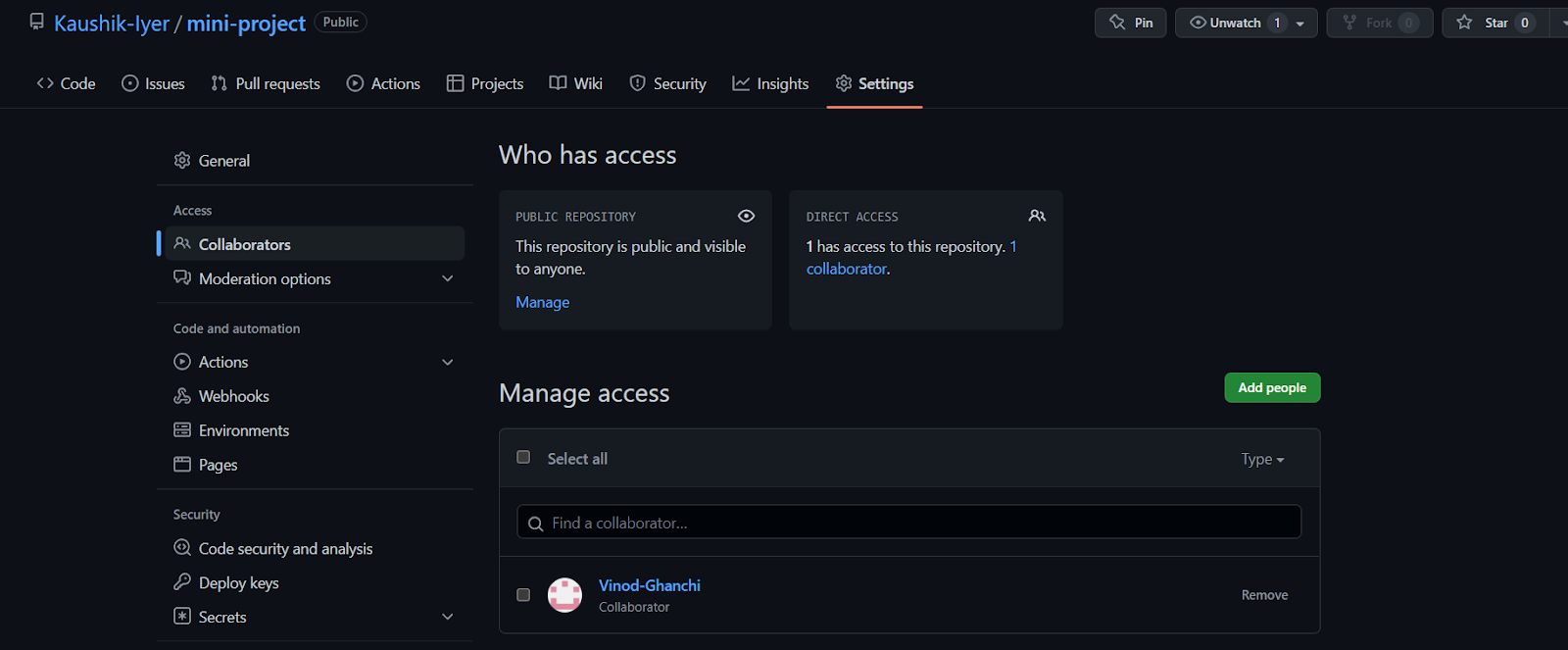
**4.5 Output/ Interface:**



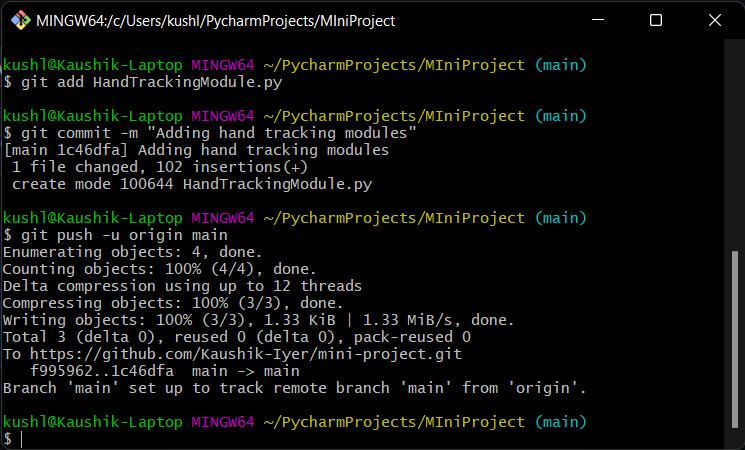


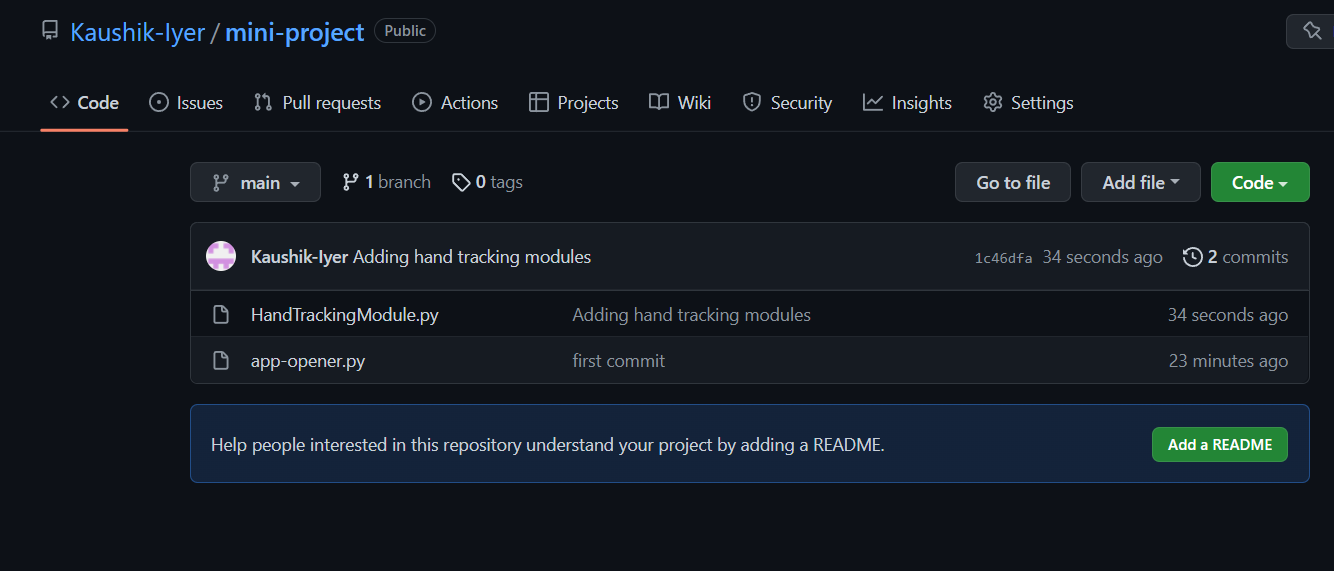
**4.6 Github Implementation details:**

Created a github repository:  


Added project members as collaborators: 

Pushing files to the repository:





By performing this experiment we have successfully understood and implemented the features of git and github

# Result

The final result is a window that gets opened after running the code. The window shows a continuous output of the webcam. In the window itself there are indicators to show the places where the hands need to be pointed to open a specific app. This is the virtual menu. When the required two hands are raised. A box is made on the fingertips to show that the program is now in ‘Selection Mode’. This is to avoid accidental opening of apps.

# Conclusion:

Hence we have implemented a virtual application menu that opens apps based on hand tracking gestures with the help of mediapipe, OpenCV, and python.

# Future Scope:

There are a lot of ways in which we can improve this project and add new capabilities and abilities in the future. Some of the ideas we have thought of are:

* Trying to implement face recognition into the project
* Trying to add functionality of opening more applications
* Improving the interface to make it look smoother and better
* Adding more functionalities with different hand gestures and combinations

# Acknowledgement:

We would like to thank Dr.Ninad Mehendale for providing us his guidance and for giving us the opportunity to learn and implement new technologies such as OpenCV, Numpy, Tensorflow, Mediapipe, and Keras. Through the topic of mini-project, we were exposed to these modules and it proved to be a very interesting project for us to make. We would also like to thank Pradnya Bhangale Ma’am for providing us with suggestions on how to improve our project.

# Bibliography:

[OpenCV: OpenCV modules](https://docs.opencv.org/4.x/)

[Hands - mediapipe (google.github.io)](https://google.github.io/mediapipe/solutions/hands.html)

[OpenCV - Overview - GeeksforGeeks](https://www.geeksforgeeks.org/opencv-overview/)

[[2006.10214] MediaPipe Hands: On-device Real-time Hand Tracking (arxiv.org)](https://arxiv.org/abs/2006.10214)

**University of Mumbai**

**Virtual Application Opener using Hand Gesture Recognition**

Submitted at the end of semester IV in partial fulfillment of requirements

**Of Bachelors in Technology in Computer Engineering**

by

**Kaushik Iyer**

**Roll No: 16010120017**

**Vinod Ghanchi**

**Roll No: 16010120016**

Guide

**Dr. Ninad Mehendale**



**Department of Computer Engineering**

**K. J. Somaiya College of Engineering, Mumbai-77**

**(Autonomous College Affiliated to University of Mumbai)**

**Batch 2018 -2021**

**K. J. Somaiya College of Engineering, Mumbai-77**

(Autonomous College Affiliated to University of Mumbai)

**Certificate**

This is to certify that the MINI PROJECT report entitled **Virtual App Opener using Hand Gesture Recognition** submitted by Vinod Ghanchi, Kuashik Iyer at the end of semester IV of SY A. Tech are bona fide record for partial fulfillment of requirements for the degree of Bachelors in Computer Engineering of University of Mumbai

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Guide Head of the Department

Date: 13-5-2022

Place: Mumbai-77

**K. J. Somaiya College of Engineering, Mumbai-77**

(Autonomous College Affiliated to University of Mumbai)

**Certificate of Approval of Examiners**

We certify that this Mini Project report entitled **Virtual App Opener using Hand Gesture Recognition** is bonafide record of Mini project work done by Vinod Ghanchi, Kaushik Iyer during semester IV.

This Mini project work is submitted at the end of semester IV in partial fulfillment of requirements for the degree of Bachelors in Technology in Computer Engineering of University of Mumbai.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner 1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Internal Examiner 2

Date: 13-5-2022

Place: Mumbai-77

**K. J. Somaiya College of Engineering, Mumbai-77**

(Autonomous College Affiliated to University of Mumbai)

**DECLARATION**

We declare that this written report submission represents the work done based on our and / or others’ ideas with adequately cited and referenced the original source. We also declare that we have adhered to all principles of intellectual property, academic honesty and integrity as we have not misinterpreted or fabricated or falsified any idea/data/fact/source/original work/ matter in my submission.

We understand that any violation of the above will be cause for disciplinary action by the college and may evoke the penal action from the sources which have not been properly cited or from whom proper permission is not sought.

| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Signature of the Student**  **16010120016**  **Roll No.** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Signature of the Student**  **16010120017**  **Roll No.** |
| --- | --- |

**Date: 13-5-2022**

**Place: Mumbai-77**