A Mini Project report on

AEROGESTURE Eye Gesture Controlled Virtual Mouse

A documentation submitted in partial fulfillment of the academic requirement for the award
of degree of

BACHELOR OF ENGINEERING

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

by
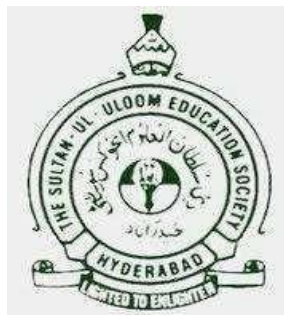
AHMED ABDUL FAHEEM (1604-21-748-015)

MANZOOR MOHD(1604-21-748-025)

SYED ABDUL KAREEM AHMED(1604-21-748-048)

Under the guidance of

Dr. UMA N. DULHARE

Professor and Head, CS&AI Dept., MJCET

**COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE
DEPARTMENT**

MUFFAKHAM JAH COLLEGE OF ENGINEERING AND
TECHNOLOGY

(Affiliated to Osmania University) Hyderabad.

2023 – 2024

MUFFAKHAM JAH
COLLEGE OF ENGINEERING AND TECHNOLOGY

Mount Pleasant, 8-2-249 to 267, Road No. 3, Banjara Hills,
Hyderabad - 500 034, Telangana State, India

CERTIFICATE

This is to certify that the mini project report on **"AEROGESTURE"** is a bonafide work carried out

by **Ahmed Abdul Faheem (1604-21-748-015), Manzoor Mohd(1604-21-748-025)**

and **Syed Abdul Kareem Ahmed(1604-21-748-048)** in the partial fulfillment of the

requirements for the award of the B.E. CSE(AI&ML) in MUFFAKHAM JAH COLLEGE OF

ENGINEERING AND TECHNOLOGY, Hyderabad for the academic year 2023-2024.

Internal Guide   **Head of the Dept**
Dr. Uma N. Dulhare   **Dr. Uma N. Dulhare**
Professor & Head,   **Professor & Head,**
CS &AI Dept, MJCET,   **CS&AI Dept, MJCET,**
Hyderabad.   **Hyderabad.**

External Examiner

Date: **College Seal**

ABSTRACT

AEROGESTURE is the application, where one can handle the mouse action by just using their two eyes and face movement, this application will figure out where the face is moving by predicting face movement in real-time by getting video frame feed from your camera no infrared camera is needed for this actions it will detect your eyelid and when it closes the blinking event triggers.

The Aim of making this software is to simplify the use of the hand in just one place i.e. keyboard, keyboard-centric people don't like to flip their hands to use a mouse when handling the keyboard. It can also be used by people with severe disabilities who can use their eyes to communicate to the computers

One can see the results of what eye-tracking technology can work when we using it with an interface of a handy computer.

<u>DECLARATION</u>

We hereby declare that the work entitled "**AeroGesture"** developed under the supervision of **Dr. UMA N. DULHARE, Professor & Head, CS&AI Department** and submitted to **MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY** in original and has not been submitted in part or while for under graduation degree to any other university.

AHMED ABDUL FAHEEM
(1604-21-748-015)

MANZOOR MOHD
(1604-21-748-025)

SYED ABDUL KAREEM AHMED
(1604-21-748-048)

ACKNOWLWDGEMENT

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

Who says that we need one mouse/eye-tracking device when you have two of your eyes, that's the idea behind the eyemousecontroller and when it works it works amazingly, just look/gaze at the screen and move your face to drag the mouse cursor.

You don't have to worry about natural blinks it will only consider the intentional blinks, mouse movement will gradually increase if the distance of the face middle point increased from the camera midpoint.

1.1   PURPOSE

This is an application made for keyboard-centric users and lazy users who don't want to use lift their hands for mouse movements or clicks. This application can also be used for the physically handicapped person who wanted to use a computer to communicate with others or just to use computers.

So this application is a medium to handle mouse events easily from the use of eyes and face movement. One can alter settings as per their needs, to save time to input camera number and start camera one of the very useful features is added which starts the working of a program on the start-up of application, the user will not have to start an application and perform setting to turn eye-tracking on.

1.2   SCOPE

This is one of the kind application where users easily move their face and cursor teleport to the place user desires. Since it is an application that can be used in daily normal life, there is just no end to it! It is in the hands of the users to keep using and it is an easy alternative for an eye-tracking device like Tobii's eye tracker.[1]

This application is to be and will be used for the benefit of, especially disabled users. Users can change the aspect ratio based on their camera ratio for accurate tracking, users can also change the illumination in the video feed if the camera is not in good lighting conditions, and illumination settings come with a slider that helps to choose the correct illumination for the real-time lighting conditions. Users can also invert the camera this feature is added for the cameras which flip the camera feed automatically.

1.3   OBJECTIVE

The main objective of this application is to make it easy to handle the mouse without any interference. Eye-tracking is aimed to keep track of the human eye and perform basic mouse functionalities. This is achieved by employing computer vision and image processing algorithms. Conventionally, we use a mouse to make the interaction between the system and the user as an input device. This application gives us a proposal to control the cursor of the computer with the help of the user's eye. Making software for mouse movements will elimina te users wearing an incontinent device to take specific actions.

An eye-tracking system is one of the best applications for a handicapped person. To develop this application various algorithm methods and techniques of image processing are used. These methods and techniques of image processing give a well-designed model for an eye-tracking system.

1.4   SOURCE OF IDEA

The idea of creating this application came suddenly during writing an essay where the use of a keyboard is more than a mouse, and using a mouse when writing was a drag. At the same time, I was working with some image processing programs in **OpenCV**[2], then I searched across the internet to find software that can perform mouse events with eyes and found out it is possible with external eye-tracking hardware's, then idea about making a piece of software struck to head which can make it easy to control mouse events. And after some thought, I came to think that it can help some physically handicapped people too who may be wanted to use computers as a normal human.

CHAPTER 2. SURVEY OF TECHNOLOGIES

2.1 WHY OPENCV?

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 the handwriting of a human. When it is integrated with various libraries, such as Numpy, python is capable of processing the OpenCV array structure for analysis. To identify image patterns and their various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both **academic** and **commercial** use. It has C++, C, Python, and Java interfaces and supports Windows, Linux, Mac OS, iOS, and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

2.1.1 ADVANTAGES OF OPENCV

2.1.1.1 ITS FAST AND SIMPLE

In image processing, since you are dealing with a large number of operations per second, it is mandatory that your code is not only providing the correct solution but that it is also providing it in the fastest manner. The OpenCV function is nearly 25x faster than the Numpy function.[3]

Convolution is the basis of many computer vision algorithms and straightforward algorithm to implement in C, but in the comparison of various implementations Opencv comes out as the winner.

If you are a python programmer, using OpenCV (Python) would be very easy. Python is an easy language to learn (especially compared to C++). It is also an excellent first language to learn.

2.1.1.2 ITS WELL ESTABLISHED

One of the great benefits of an open-source library is your ability to modify them to suit your needs. If you want to modify OpenCV, you have to modify the C/C++ source.

2.1.2 LIMITATIONS OF OPENCV

2.1.2.1 LESS CONVENIENT

Well, MATLAB is more convenient in developing and data presentation, however, OpenCV is much faster in execution. In the case of OpenCV, the speed ratio reaches more than 80 in some cases.[4]

However, OpenCV is comparatively harder to learn due to a lack of documentation and error handling codes. This disadvantage is what makes novice compute r vision users lean towards MATLAB more often. But once gained expertise with OpenCV, some professionals suggest sticking with it as it is the most comprehensive open-source libra ry for computer vision and has a large user community.

Some professionals also suggest MATLAB as it is useful for rapid prototyping and its code is very easy to debug. Moreover, it has good documentation and support.

2.2 WHY PYQT5?

There are so many options provided by Python to develop GUI applications and PyQt5 is one of them. PyQt5 is a cross-platform GUI toolkit, a set of python bindings for Qt v5. One can develop an interactive desktop application with so much ease because of the tools and simplicity provided by this library.

A GUI application consists of Front-end and Back-end. PyQt5 has provided a tool called 'QtDesigner' to design the front-end by drag and drop method so that development can become faster and one can give more time on back-end stuff.

2.2.1 ADVANTAGES OF USING PYQT

Coding versatility –GUI programming with Qt is built around the idea of signals and slots for creating contact between objects. This allows versatility in dealing with GUI incide nts which results in a smoother code base.

More than a framework: Qt uses a broad variety of native platform APIs for networking, database development, and more. It provides primary access to them through a special API.

Various UI components: Qt provides multiple widgets, such as buttons or menus, all designed with a basic interface for all compatible platforms.

Various learning resources: As PyQt is one of the most commonly used UI systems for Python, you can conveniently access a broad variety of documentation.

2.2.2 LIMITATION OF USING PYQT

Lack of Python-specific documentation for classes in PyQt5

It takes a lot of time to grasp al the specifics of PyQt, meaning it's a pretty steep learning curve.

If the application is not open-source, you must pay for a commercial license.

2.2.3 PYQT VS TKINTER

Coding flexibility – GUI programming with Qt is designed around the concept of signals and slots for establishing communication amongst objects. That permits flexibility when dealing with GUI events and results in a smoother codebase.

More than a framework – Qt uses a wide array of native platform APIs for *networking, database creation, and many more*. It offers primary access to them via a unique API.

Various UI components – Qt offers several widgets, such as buttons or menus, all designed with a basic appearance across all supported platforms.

Various learning resources – because PyQt is one of the most used UI frameworks for Python, you can get easy access to a wide array of documentation.

Easy to master – PyQt comes with a user-friendly, straightforward API functionality, along with specific classes linked to Qt C++. This allows the user to use previous knowledge from either Qt or C++, making PyQt easy to understand.

Tkinter does not include advanced widgets. It has no similar tool as Qt Designer for Tkinter.

It doesn't have a native look and feel

CHAPTER 3. REQUIREMENTANALYSIS

3.1 REQUIREMENT SPECIFICATIONS

3.1.1 FEASIBILITY STUDY

Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called a feasibility study.

This type of study determines if a project can and should be taken. Once it has been determined that a project is feasible, the analyst can go ahead and prepare the project specification which finalizes project requirements.

3.1.2 TECHNICAL FEASIBILITY

Technical issues involved are the necessary technology exists, technical guarantees of accuracy, reliability, ease of access, data security, and aspects of future expansion. The technology exists to develop a system. The proposed system is capable of holding data to be used. The proposed system is capable of providing adequate response and regardless of the number of users. The proposed system being modular to the administrator, if he/she wants can add more features in the future and as well as be able to expand the system.

3.1.3 OPERATIONALFEASIBILITY

If the system meets the requirements of the customers and the administrator we can say that the system is operationally feasible. The proposed system will be beneficial only if it can be turned into a system that will meet the requirements of the store when it is developed and installed, and there is sufficient support from the users. The proposed system will improve the total performance. Customers here are the most important part of the system and the proposed system will provide them with a convenient mode of operation for them.

3.1.4 ECONOMIC FEASIBILITY

Economic Feasibility is the most frequently used method for evaluating the effectiveness of the proposed system if the benefit of the proposed system outweighs the cost then the decision is made to design and implement the system. The cost of hardware and software is affordable.

3.2 SOFTWARE AND HARDWARE REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS

- Processor   : Intel Core 2 Duo or above
- Ram   : 2GB or above
- Hard Drive   : 250GB or above
- System Bus   : 64 bit
- Clock Speed   : 1.8 GHz
- Webcam   : Quality Camera for

3.2.2 SOFTWARE REQUIREMENTS
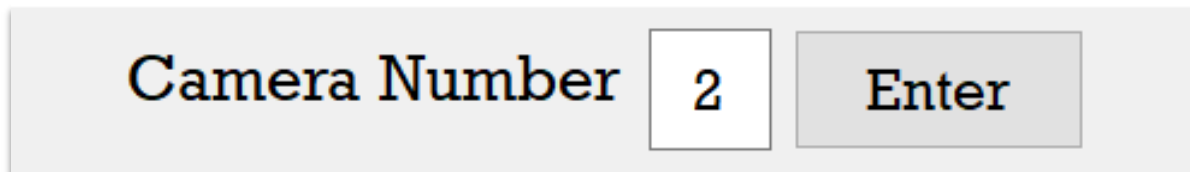
- Operating System   : Windows XP/7/Vista/8/10
- Programming Language   : Python
- Major Libraries   : Opencv, Dlib, Pynput, PyQt5

CHAPTER 4. SYSTEM DESIGN DETAILS

4.1 SOFTWARE DESIGN

4.1.1 INTERFACE
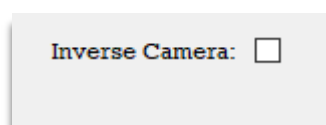
The interface of the eye mouse controller is basically to take input and show the webcam's video feed, start tracking, and detecting blinks.
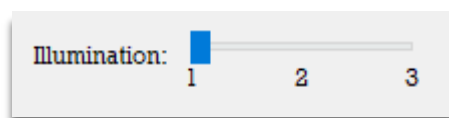
After starting the eye mouse controller application first user needs to input the camera number, if it a laptop camera number default value is 1 and the user had attached an external camera then the user will input number 2 as a value if multiple cameras are attached and then user need to input correct camera number to run the application. After keying the correct camera number Enter button needs to be pressed.

After adding the camera number user need to specify the aspect ratio of the camera which is used as a webcam. Most of the webcam comes with 4:3 as an aspect ratio so our application comes compatible with 4:3. If the camera-input aspect ratio is 16:9 then the user needs to input selection in the aspect ratio checkbox.

After choosing the correct aspect ratio user check whether a camera is flipped by default. If a webcam is flipped then check the inverse camera button. It will reverse the flipping and tracking will happen in exact order.

After choosing correct alternatives user need to balance out the lighting with help of illumination slider which helps in low light condition. Illumination slider comes with 3 values 1/2/3. The default value is one of the value increase by 1 then the gamma value increases in a video frame.

And here is the full view of the interface of the **AeroGesture**
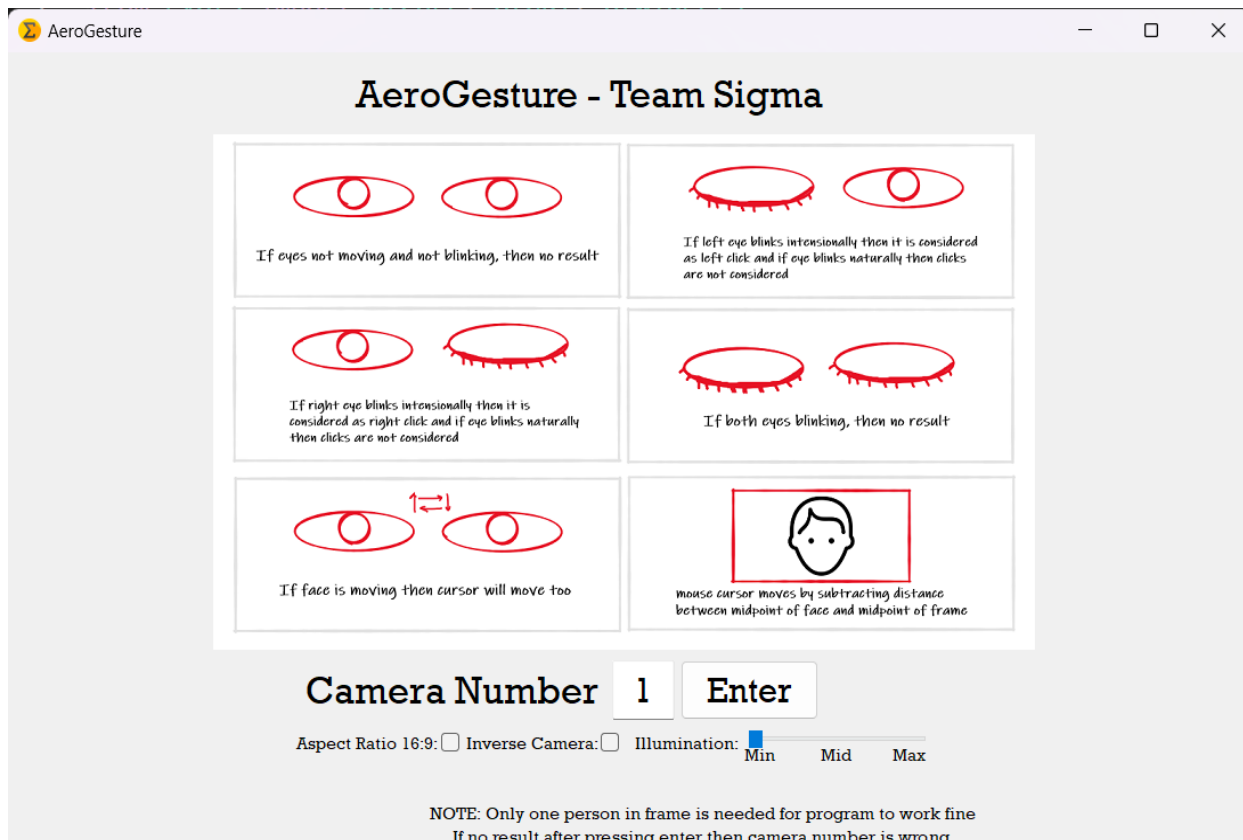
In this application for easy use and no repetitive assignment of camera number, the camera number is saved in a file where it stores the last used camera number which is used to run OpenCV frame with previously used frame.

4.1.2 OPENCV FRAME

After configuring all the settings, a user selects enter to run a program and gets a new window started of OPENCV video capture. Which illustrates how a face is being recognized and shows the distance between the face midpoint and the camera midpoint.

It also shows if a user is far from a frame or not, user can different by looking at the video frame how gradually cursor speed is increasing.
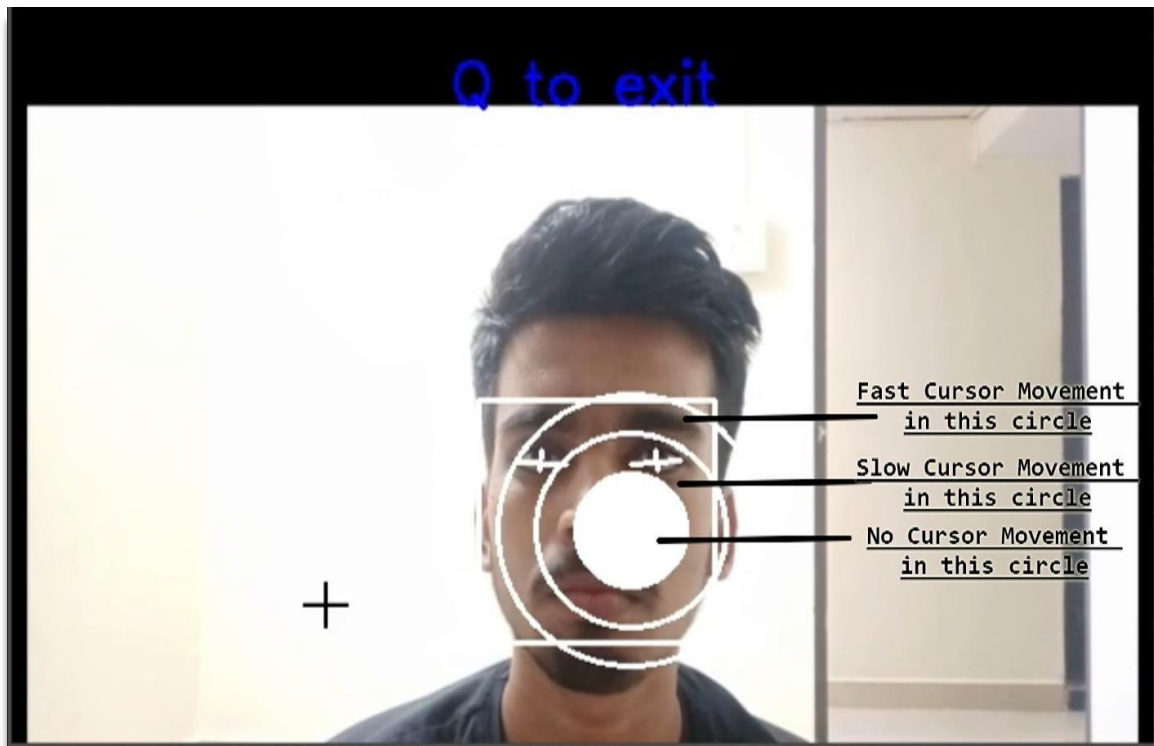
Here if the user's face is in the middle circle then the cursor not moving, and if a face is in the second circle then the cursor moves slowly, and if the face crosses 2nd circle then the cursor moves at a faster speed.

Users can click the letter 'Q' to exit. By clicking 'Q' the user just exits out of the OpenCV video frame.

To exit the full program user need to close the interface too.

4.2 FLOW CHART.

A **flowchart** is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task.

The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting, or managing a process or program in various fields.
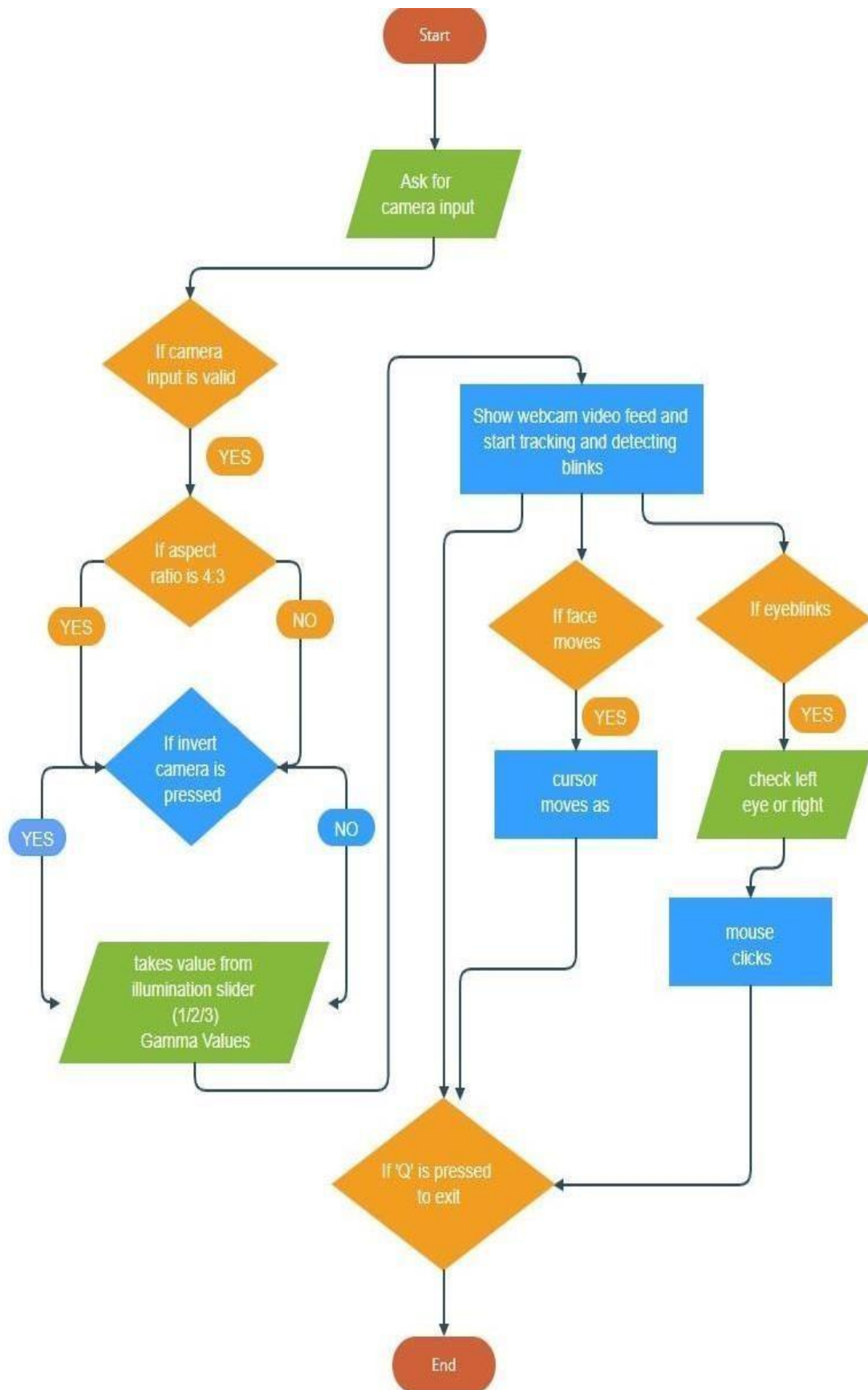
4.3 GANTT CHART

A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The width of the horizontal bars in the graph shows the duration of each activity. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. As we have already discussed, Gantt charts are used for project management purposes. To use Gantt charts in a project, there are a few initial requirements fulfilled by the project.

First of all, the project should have a sufficiently detailed Work Breakdown Structure (WBS). Secondly, the project should have identified its milestones and deliveries.

In some instances, project managers try to define the work breakdown structure while creating a Gantt chart. This is one of the frequently practiced errors in using Gantt charts. Gantt charts are not designed to assist the WBS process; rather Gantt charts are for task progress tracking.

This method maximizes the float time available for all tasks. Gantt chart of this project is as follows:

TASKS			START DATE	DAYS TOOK
PROJECT PLANNING AND RESEARCH			6 Nov, 2023	10
SURVEY OF TECHNOLOGY			16 Nov, 2023	5
REQUIREMENT ANALYSIS			21 Nov, 2023	9
VIDEO OUTPUT ANALYSIS			30 Nov, 2023	1
CURSOR MOVEMENT			1 Dec, 2023	5
EYE BLINKING DETECTION OPTIMIZATION OF VIDEO FEED			6 Dec, 2023	5
RESCALING FRAME			11 Dec, 2023	2
IMPROVED CURSOR MOVEMENT			13 Dec, 2023	7
IMPROVED EYE BLINK			20 Dec, 2024	20
INTERFACE DESIGN			9 Jan, 2024	2
ASPECT RATIO WORK			11 Jan, 2024	2
INVERT CAMERA FEATURE			13 Jan, 2024	4
ADDED ILLUMINATION SLIDER			17 Jan, 2024	5
WORKED ON BUGS			23 Jan, 2024	5
COMPATIBILTY FOR ALL DEVICE SIZES			28 Jan, 2024	5
INTERFACE IMPROVEMENT			2 Feb, 2024	2
TESTING			4 Feb, 2024	2
DELIVERY			6 Feb, 2024	1

GANTT CHART OF EYE MOUSE CONTROLLER
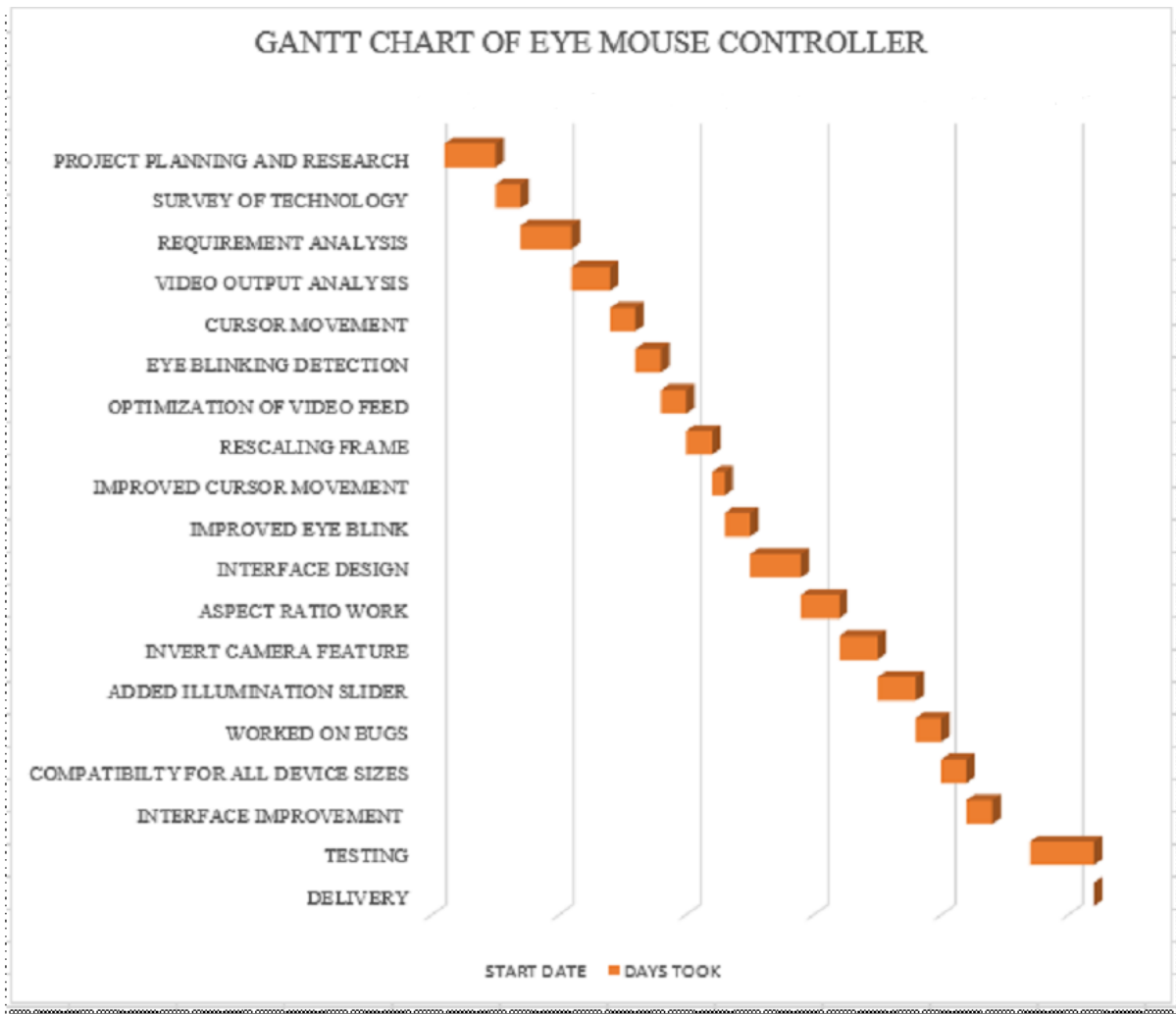
4.4 ER (Entity-Relation Diagram)

An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.
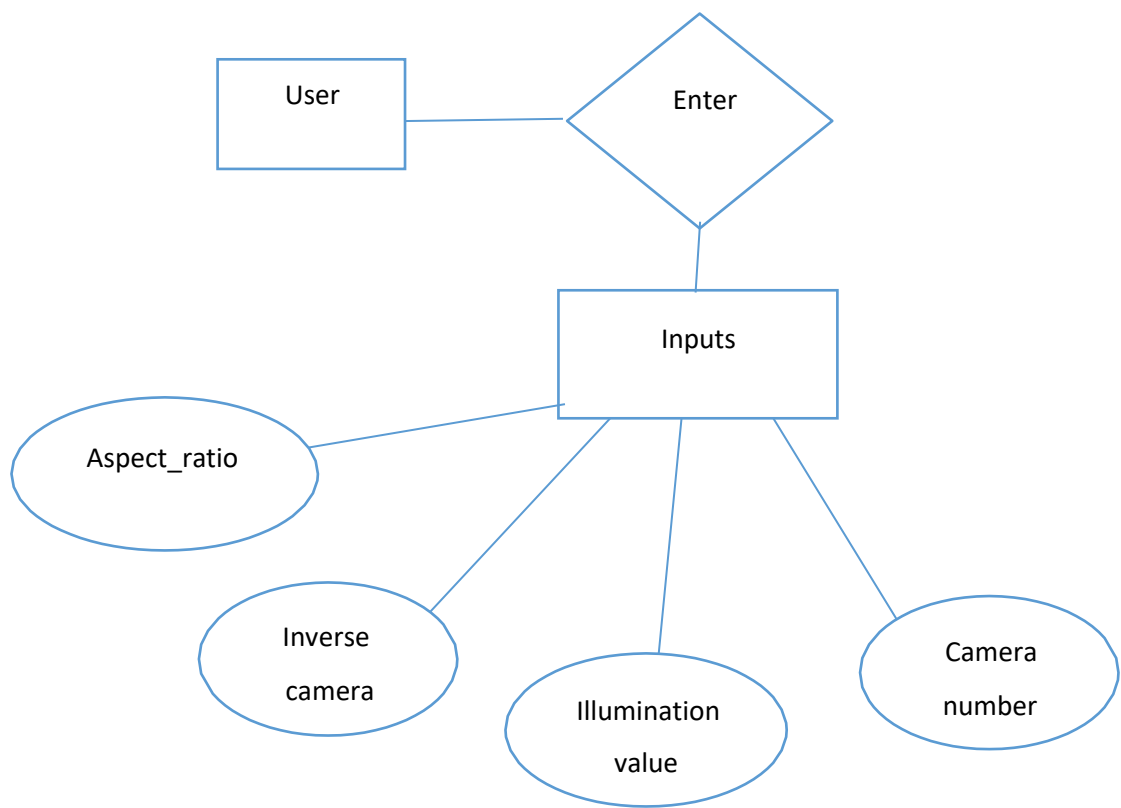
CHAPTER 5. SYSTEM IMPLEMENTATION

5.1 CODE

Eyemouseinterface_fastpc.py:

Code:

```python
from PyQt5 import QtCore, QtGui, QtWidgets
import eyetracking_fastpc
from PyQt5 import sip
import cv2 as cv
cv2=cv
import dlib
import mousecontrol_eye
from win32.win32api import GetSystemMetrics
from pynput.mouse import Listener,Button,Controller
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import QCoreApplication
import time
import webbrowser
from imutils.video import WebcamVideoStream
class Ui_MainWindow(object):
    def __init__(self):
        self.f=open("cameranumbersaved.txt","r+")
        self.cameranumbersaved=self.f.read()
        if(self.cameranumbersaved==""):
            open("cameranumbersaved.txt","r+").write("1")
            self.cameranumbersaved=self.f.read()
        self.f.close()
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.setEnabled(True)
        MainWindow.resize(1092, 684)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())
        MainWindow.setSizePolicy(sizePolicy)
        MainWindow.setContextMenuPolicy(QtCore.Qt.NoContextMenu)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("icon.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        MainWindow.setWindowIcon(icon)
        MainWindow.setToolTip("")
        MainWindow.setStatusTip("")
        MainWindow.setWhatsThis("")
        MainWindow.setAccessibleName("")
```

```
MainWindow.setAccessibleDescription("")
MainWindow.setAutoFillBackground(False)
MainWindow.setWindowFilePath("")
MainWindow.setDocumentMode(False)
MainWindow.setTabShape(QtWidgets.Q TabWidget.Rounded)
MainWindow.setDockNestingEnabled(False)
self.centralwidget = QtWidgets.QWidget(MainWindow)
self.centralwidget.setObjectName("centralwidget")
self.EnterButton = QtWidgets.QPushButton(self.centralwidget)
self.EnterButton.setGeometry(QtCore.QRect(590, 520, 121, 51))
font = QtGui.QFont()
font.setFamily("Rockwell")
font.setPointSize(18)
self.EnterButton.setFont(font)
self.EnterButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
self.EnterButton.setIconSize(Q tCore.QSize(16, 16))
self.EnterButton.setObjectName("EnterButton")
self.EnterButton.clicked.connect(self.Enterbuttonclicked)
self.Eyemouselabel = QtWidgets.QLabel(self.centralwidget)
self.Eyemouselabel.setGeometry(QtCore.QRect(130, -10, 761, 91))
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.Eyemouselabel.sizePolicy().hasHeightForWidth())
self.Eyemouselabel.setSizePolicy(sizePolicy)
font = QtGui.QFont()
font.setFamily("Rockwell")
font.setPointSize(20)
font.setBold(False)
font.setWeight(50)
self.Eyemouselabel.setFont(font)
self.Eyemouselabel.setTextFormat(QtCore.Qt.AutoText)
self.Eyemouselabel.setAlignment(QtCore.Qt.AlignCenter)
self.Eyemouselabel.setWordWrap(False)
self.Eyemouselabel.setObjectName("Eyemouselabel")
self.instructionimagelabel = QtWidgets.QLabel(self.centralwidget)
self.instructionimagelabel.setGeometry(QtCore.QRect(180, 70, 721, 441))
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.instructionimagelabel.sizePolicy().hasHeightForWidth())
self.instructionimagelabel.setSizePolicy(sizePolicy)
self.instructionimagelabel.setText("")
self.instructionimagelabel.setPixmap(QtGui.QPixmap("instruction.png"))
self.instructionimagelabel.setAlignment(QtCore.Qt.AlignCenter)
self.instructionimagelabel.setObjectName("instructionimagelabel")
self.cameralabel  = QtWidgets.QLabel(self.centralwidget)
```

```python
        self.cameralabel.setGeometry(QtCore.QRect(310, 510, 211, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel.sizePolicy().hasHeightForWidth())
        self.cameralabel.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(20)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel.setFont(font)
        self.cameralabel.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel.setWordWrap(False)
        self.cameralabel.setObjectName("cameralabel")
        self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(530, 520, 51, 51))
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(18)
        self.lineEdit.setFont(font)
        self.lineEdit.setMaxLength(1)
        self.lineEdit.setAlignment(QtCore.Qt.AlignCenter)
        self.lineEdit.setObjectName("lineEdit")
        self.cameralabel_3 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_3.setGeometry(QtCore.QRect(310, 620, 421, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_3.sizePolicy().hasHeightForWidth())
        self.cameralabel_3.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_3.setFont(font)
        self.cameralabel_3.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_3.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_3.setWordWrap(False)
        self.cameralabel_3.setObjectName("cameralabel_3")
        self.cameralabel_4 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_4.setGeometry(QtCore.QRect(310, 640, 421, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
```

```python
        sizePolicy.setHeightForWidth(self.cameralabel_4.sizePolicy().hasHeightForWidth())
        self.cameralabel_4.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_4.setFont(font)
        self.cameralabel_4.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_4.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_4.setWordWrap(False)
        self.cameralabel_4.setObjectName("cameralabel_4")
        self.cameralabel_5 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_5.setGeometry(QtCore.QRect(410, 560, 111, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_5.sizePolicy().hasHeightForWidth())
        self.cameralabel_5.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_5.setFont(font)
        self.cameralabel_5.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_5.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_5.setWordWrap(False)
        self.cameralabel_5.setObjectName("cameralabel_5")
        self.inversecameracheck = QtWidgets.QCheckBox(self.centralwidget)
        self.inversecameracheck.setGeometry(QtCore.QRect(520, 580, 20, 21))
        self.inversecameracheck.setAutoFillBackground(False)
        self.inversecameracheck.setText("")
        self.inversecameracheck.setIconSize(QtCore.QSize(16, 16))
        self.inversecameracheck.setObjectName("inversecameracheck")
        self.GithubLogo = QtWidgets.QLabel(self.centralwidget)
        self.GithubLogo.setGeometry(QtCore.QRect(660, 0, 71, 71))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.GithubLogo.sizePolicy().hasHeightForWidth())
        self.GithubLogo.setSizePolicy(sizePolicy)
        self.GithubLogo.setText("")
        self.GithubLogo.setPixmap(QtGui.QPixmap("githublogo.png"))
        self.GithubLogo.setAlignment(QtCore.Qt.AlignCenter)
        self.GithubLogo.setObjectName("GithubLogo")
        self.GithubButton = QtWidgets.QPushButton(self.centralwidget)
        self.GithubButton.setGeometry(QtCore.QRect(660, 0, 71, 71))
```

```python
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(18)
        self.GithubButton.setFont(font)
        self.GithubButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.GithubButton.setStyleSheet("\n"
"background-color: rgba(255, 255, 255, 0);")
        self.GithubButton.setText("")
        self.GithubButton.setIconSize(QtCore.QSize(16, 16))
        self.GithubButton.setObjectName("GithubButton")
        url="https://github.com/HARSHSINGH0/EYE_MOUSE"
        self.GithubButton.clicked.connect(lambda:self.linktogithub(url))
        self.cameralabel_6 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_6.setGeometry(QtCore.QRect(260, 560, 111, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_6.sizePolicy().hasHeightForWidth())
        self.cameralabel_6.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_6.setFont(font)
        self.cameralabel_6.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_6.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_6.setWordWrap(False)
        self.cameralabel_6.setObjectName("cameralabel_6")
        self.aspectratio169 = QtWidgets.QCheckBox(self.centralwidget)
        self.aspectratio169.setGeometry(QtCore.QRect(380, 580, 20, 21))
        self.aspectratio169.setAutoFillBackground(False)
        self.aspectratio169.setText("")
        self.aspectratio169.setIconSize(QtCore.QSize(16, 16))
        self.aspectratio169.setObjectName("aspectratio169")
        self.cameralabel_7 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_7.setGeometry(QtCore.QRect(540, 560, 111, 61))
        sizePolicy  = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_7.sizePolicy().hasHeightForWidth())
        self.cameralabel_7.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_7.setFont(font)
```

```python
        self.cameralabel_7.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_7.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_7.setWordWrap(False)
        self.cameralabel_7.setObjectName("cameralabel_7")
        self.cameralabel_8 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_8.setGeometry(QtCore.QRect(610, 570, 171, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_8.sizePolicy().hasHeightForWidth())
        self.cameralabel_8.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_8.setFont(font)
        self.cameralabel_8.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_8.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_8.setWordWrap(False)
        self.cameralabel_8.setObjectName("cameralabel_8")
        self.horizontalSlider = QtWidgets.QSlider(self.centralwidget)
        self.horizontalSlider.setGeometry(QtCore.QRect(640, 580, 111, 16))
        self.horizontalSlider.setMinimum(1)
        self.horizontalSlider.setMaximum(3)
        self.horizontalSlider.setPageStep(1)
        self.horizontalSlider.setProperty("value", 1)
        self.horizontalSlider.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider.setObjectName("horizontalSlider")
        MainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Eye Mouse Controller"))
        self.EnterButton.setText(_translate("MainWindow", "Enter"))
        self.EnterButton.setShortcut(_translate("MainWindow", "Return"))
        self.Eyemouselabel.setText(_translate("MainWindow", "Eye Mouse Controller"))
        self.cameralabel.setText(_translate("MainWindow", "Camera Number"))
        self.lineEdit.setText(_translate("MainWindow", "1"))
        self.cameralabel_3.setText(_translate("MainWindow", "NOTE: Only one person in
frame is needed for program to work fine"))
        self.cameralabel_4.setText(_translate("MainWindow", "If no result after pressing enter
then camera number is wrong "))
        self.cameralabel_5.setText(_translate("MainWindow", "Inverse Camera:"))
        self.GithubButton.setShortcut(_translate("MainWindow", "Return"))
        self.cameralabel_6.setText(_translate("MainWindow", "Aspect Ratio 16:9:"))
```

```python
            self.cameralabel_7.setText(_translate("MainWindow", "Illumination:"))
            self.cameralabel_8.setText(_translate("MainWindow", "1           2           3"))
    def Enterbuttonclicked(self,savedornot):
        try:
            if savedornot==False:
                camerainput=int(self.lineEdit.text())
                cameracheck=self.inversecameracheck.isChecked()
                aspectratio169check=self.aspectratio169.isChecked()
                open("cameranumbersaved.txt","r+").truncate()
                if camerainput==None:
                    camerainput=1

                open("cameranumbersaved.txt","r+").write(str(camerainput))
                illumnation=int(self.horizontalSlider.value())

eyemouse=eyetracking_fastpc.eye_mouse(camerainput,cameracheck,aspectratio169check,illu
mnation)
                eyemouse.eyetrack()
            else:
                self.lineEdit.setText(self.cameranumbersaved)
                camerainput=int(self.lineEdit.text())
                cameracheck=self.inversecameracheck.isChecked()
                aspectratio169check=self.aspectratio169.isChecked()
                illumnation=int(self.horizontalSlider.value())

eyemouse=eyetracking_fastpc.eye_mouse(camerainput,cameracheck,aspectratio169check,illu
mnation)
                eyemouse.eyetrack()
        except(Exception):
            pass
    def linktogithub(self, link):
        self.link=link
        webbrowser.open(self.link, new=0)

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    ui.Enterbuttonclicked(True)
    sys.exit(app.exec_())
```

Eyetracking_fastpc.py:
Code:

```python
import cv2 as cv
cv2=cv
import dlib
import mousecontrol_eye
from win32.win32api import GetSystemMetrics
import time
from pynput.mouse import Listener,Button,Controller
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from imutils.video import WebcamVideoStream
import numpy as np
class eye_mouse:
    def __init__(self,camerainput,cameracheck,aspectratio169,illumination):
        self.camerainput=int(camerainput)
        self.blinking_frames=0
        self.mousecontrol=mousecontrol_eye.mousecontrol()
        self.cameracheck=cameracheck
        self.aspectratio169=aspectratio169
        width = GetSystemMetrics(0)
        height = GetSystemMetrics(1)
        middlepoint1=width/2
        middlepoint2=height/2
        self.mousecontrol.firstpos(middlepoint1,middlepoint2)
        self.illumination=illumination
        if self.illumination==None:
            self.illumination=1
    def rescaleFrame(self,frame):
        dimension=(600,450)
        return cv.resize(frame,dimension,interpolation=cv.INTER_AREA)
    def midlinepoint(self,p1,p2):
        return int((p1.x+p2.x)/2),int((p1.y+p2.y)/2)
    def aspectratiochanger(self,ratio,frame):
        if ratio=="16by9":
            dimension=(640,360)
            return cv.resize(frame,dimension,interpolation=cv.INTER_AREA)
    def adjust_gamma(self,frame, gamma=1.0):
        if gamma==1:
            return frame
        else:
            invGamma = 1.0 / gamma
            table = np.array([((i / 255.0) ** invGamma) * 255
                for i in np.arange(0, 256)]).astype("uint8")
            return cv2.LUT(frame, table)
    def eyetrack(self):
        blinking_frames=self.blinking_frames
        blinking_frames1=self.blinking_frames
        self.cap=WebcamVideoStream(src=self.camerainput-1).start()
        cameracheck=self.cameracheck
```

22

```python
        self.detector=dlib.get_frontal_face_detector()
        self.predictor=dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
        mouse=Controller()
        while True:
            try:
                errornumber=0
                if cameracheck==False:
                    frame=self.cap.read()
                    frame=self.adjust_gamma(frame,self.illumination)
                    gray=cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
                    if self.aspectratio169==True:
                        gray=self.aspectratiochanger("16by9",gray)
                        frame=self.aspectratiochanger("16by9",frame)
                    else:
                        gray=self.rescaleFrame(gray)
                        frame=self.rescaleFrame(frame)
                    faces=self.detector(gray)
                else:
                    frame=self.cap.read()
                    frame=self.adjust_gamma(frame,self.illumination)
                    gray=cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
                    if self.aspectratio169==True:
                        gray=cv.flip(self.aspectratiochanger("16by9",gray),1)
                        frame=cv.flip(self.aspectratiochanger("16by9",frame),1)
                    else:
                        gray=cv.flip(self.rescaleFrame(gray),1)
                        frame=cv.flip(self.rescaleFrame(frame),1)
                    faces=self.detector(gray)
                cv.putText(frame,"Q to
exit",(230,50),cv.FONT_HERSHEY_DUPLEX,1,(255,0,0),1)
                for face in faces:
                    x,y=face.left(),face.right()
                    x1,y1=face.top(),face.bottom()
                    facerect=cv2.rectangle(frame,(x,x1),(y,y1),(255,255,255),2)
                    landmarks=self.predictor(gray,face)
                    noselandmark=(landmarks.part(30).x,landmarks.part(30).y)
                    xvaluerectsmall=275
                    yvaluerectsmall=300
                    xvaluerectsmall_r=325
                    yvaluerectsmall_r=350
                    eyestonosepointx,eyestonosepointy=landmarks.part(30).x,landmarks.part(30).y
                    nose_to_cursorx=325
                    nose_to_cursory=270
cv.rectangle(frame,(eyestonosepointx,eyestonosepointy),(eyestonosepointx,eyestonosepointy
),(255,255,255),thickness=4)
cv.rectangle(frame,(nose_to_cursorx,nose_to_cursory),(nose_to_cursorx,nose_to_cursory),(0
,0,0),thickness=5)
cv.line(frame,(eyestonosepointx,eyestonosepointy),(nose_to_cursorx,nose_to_cursory),(255,
255,255),thickness=2,lineType=cv.FILLED)
                    center_coordinates  = (325, 270)
```

23

```python
radius = 30
color = (220,220,220)
cv2.circle(frame, center_coordinates, radius, color, thickness=-1,lineType=cv.FILLED)
cv2.circle(frame, center_coordinates,50, color, thickness=2,lineType=cv.FILLED)
cv2.circle(frame, center_coordinates,70, color, thickness=2,lineType=cv.FILLED)
positivecursorvalue=15
negativesursorvalue=-15
if((eyestonosepointx-nose_to_cursorx)>positivecursorvalue):
    if((eyestonosepointx-nose_to_cursorx)>70):
        mouse.move(12,0)
    elif((eyestonosepointx- nose_to_cursorx)>70):
        mouse.move(8,0)
    elif((eyestonosepointx- nose_to_cursorx)>50):
        mouse.move(4,0)
    elif((eyestonosepointx- nose_to_cursorx)>30):
        mouse.move(2,0)
if((eyestonosepointx-nose_to_cursorx)<negativesursorvalue):
    if((eyestonosepointx-nose_to_cursorx)<-70):
        mouse.move(-12,0)
    elif((eyestonosepointx- nose_to_cursorx)<-70):
        mouse.move(-8,0)
    elif((eyestonosepointx- nose_to_cursorx)<-50):
        mouse.move(-4,0)
    elif((eyestonosepointx- nose_to_cursorx)<-30):
        mouse.move(-2,0)
if(eyestonosepointy- nose_to_cursory)<positivecursorvalue:
    if(eyestonosepointy-nose_to_cursory)<50:
        mouse.move(0,-6)
    elif(eyestonosepointy-nose_to_cursory)<30:
        mouse.move(0,-2)
if(eyestonosepointy- nose_to_cursory)>negativesursorvalue:
    if(eyestonosepointy-nose_to_cursory)>-50:
        mouse.move(0,6)
    elif(eyestonosepointy-nose_to_cursory)>-30:
        mouse.move(0,2)
left_point=(landmarks.part(36).x,landmarks.part(36).y)
right_point=(landmarks.part(39).x,landmarks.part(39).y)
hor_line=cv.line(frame,left_point,right_point,(255,255,255),2)
up_point=(self.midlinepoint(landmarks.part(37),landmarks.part(38)))
down_point=(self.midlinepoint(landmarks.part(41),landmarks.part(40)))
ver_line=cv.line(frame,up_point,down_point,(255,255,255),2)
left_point_r=(landmarks.part(42).x,landmarks.part(42).y)
right_point_r=(landmarks.part(45).x,landmarks.part(45).y)
hor_line_r=cv.line(frame,left_point_r,right_point_r,(255,255,255),2)
up_point_r=(self.midlinepoint(landmarks.part(43),landmarks.part(44)))
down_point_r=(self.midlinepoint(landmarks.part(47),landmarks.part(46)))
ver_line_r=cv.line(frame,up_point_r,down_point_r,(255,255,255),2)
```

```python
                value_of_blink=-3
                if((y1-x1)>170):
                    value_of_blink=-7
                elif((y1-x1)>140):
                    value_of_blink=-6
                elif((y1-x1)>130):
                    value_of_blink=-5
                elif((y1-x1)>120):
                    value_of_blink=-4
                elif((y1-x1)<105):
                    cv.putText(frame,"come close to the
camera",(80,150),cv.FONT_HERSHEY_DUPLEX,1,(0,0,0),1)
                    value_of_blink=10
                if((up_point[1]-down_point[1])>=value_of_blink):
                    blinking_frames+=1
                    if (blinking_frames>3):
                        blinking_frames=0
                        cv.putText(frame,"Left
click",(250,150),cv.FONT_HERSHEY_DUPLEX,1,(0,0,0),1)
                        self.mousecontrol.left_click()
                else:
                    while blinking_frames!=0:
                        blinking_frames-=1
                if((up_point_r[1]-down_point_r[1])>=value_of_blink):
                    blinking_frames1+=1
                    if (blinking_frames1>3):
                        blinking_frames1=0
                        cv.putText(frame,"Right
click",(250,150),cv.FONT_HERSHEY_DUPLEX,1,(0,0,0),1)
                        self.mousecontrol.right_click()
                else:
                    while blinking_frames1!=0:
                        blinking_frames1-=1
            cv.imshow("frame",frame)
            if cv2.waitKey(1) & 0xFF == ord('q') :
                cv2.destroyAllWindows()
                self.cap.stop()
                break
        except :
            cv2.destroyAllWindows()
            self.cap.stop()
            break
cv2.destroyAllWindows()
```

Mousecontrol_eye.py:
Code:

```python
from win32.win32api import GetSystemMetrics
from pynput.mouse import Listener,Button,Controller
import time
class mousecontrol:
    def __init_(self):
        self.mouse=Controller()
    def firstpos(self,x,y):
        self.mouse.position=(x,y)
    def left_click(self):
        self.mouse.click(Button.left,1)
        time.sleep(0.2)
    def right_click(self):
        self.mouse.click(Button.right,1)
        time.sleep(0.2)
```

Eyemouseinterface_slowpc.py
Code:

```python
from PyQt5 import QtCore, QtGui, QtWidgets
import eyetracking_slowpc
from PyQt5 import sip
import cv2 as cv
cv2=cv
import dlib
import mousecontrol_eye
from win32.win32api import GetSystemMetrics
from pynput.mouse import Listener,Button,Controller
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import QCoreApplication
import time
import webbrowser
from imutils.video import WebcamVideoStream
import numpy as np
class Ui_MainWindow(object):
    def _init_(self):
        self.f=open("cameranumbersaved.txt","r+")
        self.cameranumbersaved=self.f.read()
        if(self.cameranumbersaved==""):
            open("cameranumbersaved.txt","r+").write("1")
            self.cameranumbersaved=self.f.read()
        self.f.close()
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.setEnabled(True)
        MainWindow.resize(1092, 684)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
```

```python
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())
        MainWindow.setSizePolicy(sizePolicy)
        MainWindow.setContextMenuPolicy(QtCore.Qt.NoContextMenu)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("icon.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        MainWindow.setWindowIcon(icon)
        MainWindow.setToolTip("")
        MainWindow.setStatusTip("")
        MainWindow.setWhatsThis("")
        MainWindow.setAccessibleName("")
        MainWindow.setAccessibleDescription("")
        MainWindow.setAutoFillBackground(False)
        MainWindow.setWindowFilePath("")
        MainWindow.setDocumentMode(False)
        MainWindow.setTabShape(QtWidgets.Q TabWidget.Rounded)
        MainWindow.setDockNestingEnabled(False)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.EnterButton = QtWidgets.QPushButton(self.centralwidget)
        self.EnterButton.setGeometry(QtCore.QRect(590, 520, 121, 51))
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(18)
        self.EnterButton.setFont(font)
        self.EnterButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.EnterButton.setIconSize(Q tCore.QSize(16, 16))
        self.EnterButton.setObjectName("EnterButton")
        self.EnterButton.clicked.connect(self.Enterbuttonclicked)
        self.Eyemouselabel = QtWidgets.QLabel(self.centralwidget)
        self.Eyemouselabel.setGeometry(QtCore.QRect(130, -10, 761, 91))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
    QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.Eyemouselabel.sizePolicy().hasHeightForWidth())
        self.Eyemouselabel.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(20)
        font.setBold(False)
        font.setWeight(50)
        self.Eyemouselabel.setFont(font)
        self.Eyemouselabel.setTextFormat(QtCore.Qt.AutoText)
        self.Eyemouselabel.setAlignment(QtCore.Qt.AlignCenter)
        self.Eyemouselabel.setWordWrap(False)
        self.Eyemouselabel.setObjectName("Eyemouselabel")
        self.instructionimagelabel = QtWidgets.QLabel(self.centralwidget)
        self.instructionimagelabel.setGeometry(QtCore.QRect(180,  70, 721, 441))
```

```python
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.instructionimagelabel.sizePolicy().hasHeightForWidth())
        self.instructionimagelabel.setSizePolicy(sizePolicy)
        self.instructionimagelabel.setText("")
        self.instructionimagelabel.setPixmap(QtGui.QPixmap("instruction.png"))
        self.instructionimagelabel.setAlignment(QtCore.Qt.AlignCenter)
        self.instructionimagelabel.setObjectName("instructionimagelabel")
        self.cameralabel = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel.setGeometry(QtCore.QRect(310, 510, 211, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel.sizePolicy().hasHeightForWidth())
        self.cameralabel.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(20)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel.setFont(font)
        self.cameralabel.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel.setWordWrap(False)
        self.cameralabel.setObjectName("cameralabel")
        self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(530, 520, 51, 51))
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(18)
        self.lineEdit.setFont(font)
        self.lineEdit.setMaxLength(1)
        self.lineEdit.setAlignment(QtCore.Qt.AlignCenter)
        self.lineEdit.setObjectName("lineEdit")
        self.cameralabel_3 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_3.setGeometry(QtCore.QRect(310, 620, 421, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_3.sizePolicy().hasHeightForWidth())
        self.cameralabel_3.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
```

```python
        font.setWeight(50)
        self.cameralabel_3.setFont(font)
        self.cameralabel_3.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_3.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_3.setWordWrap(False)
        self.cameralabel_3.setObjectName("cameralabel_3")
        self.cameralabel_4 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_4.setGeometry(QtCore.QRect(310, 640, 421, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_4.sizePolicy().hasHeightForWidth())
        self.cameralabel_4.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_4.setFont(font)
        self.cameralabel_4.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_4.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_4.setWordWrap(False)
        self.cameralabel_4.setObjectName("cameralabel_4")
        self.cameralabel_5 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_5.setGeometry(QtCore.QRect(410, 560, 111, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_5.sizePolicy().hasHeightForWidth())
        self.cameralabel_5.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_5.setFont(font)
        self.cameralabel_5.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_5.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_5.setWordWrap(False)
        self.cameralabel_5.setObjectName("cameralabel_5")
        self.inversecameracheck = QtWidgets.QCheckBox(self.centralwidget)
        self.inversecameracheck.setGeometry(QtCore.QRect(520, 580, 20, 21))
        self.inversecameracheck.setAutoFillBackground(False)
        self.inversecameracheck.setText("")
        self.inversecameracheck.setIconSize(QtCore.QSize(16, 16))
        self.inversecameracheck.setObjectName("inversecameracheck")
        self.GithubLogo = QtWidgets.QLabel(self.centralwidget)
        self.GithubLogo.setGeometry(QtCore.QRect(660, 0, 71, 71))
```

```python
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.GithubLogo.sizePolicy().hasHeightForWidth())
        self.GithubLogo.setSizePolicy(sizePolicy)
        self.GithubLogo.setText("")
        self.GithubLogo.setPixmap(QtGui.QPixmap("githublogo.png"))
        self.GithubLogo.setAlignment(QtCore.Qt.AlignCenter)
        self.GithubLogo.setObjectName("GithubLogo")
        self.GithubButton = QtWidgets.QPushButton(self.centralwidget)
        self.GithubButton.setGeometry(QtCore.Q Rect(660, 0, 71, 71))
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(18)
        self.GithubButton.setFont(font)
        self.GithubButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.GithubButton.setStyleSheet("\n"
"background-color: rgba(255, 255, 255, 0);")
        self.GithubButton.setText("")
        self.GithubButton.setIconSize(QtCore.QS ize(16, 16))
        self.GithubButton.setObjectName("GithubButton")
        url="https://github.com/HARSHSINGH0/EYE_MOUSE"
        self.GithubButton.clicked.connect(lambda:self.linktogithub(url))
        self.cameralabel_6 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_6.setGeometry(QtCore.QRect(260, 560, 111, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_6.sizePolicy().hasHeightForWidth())
        self.cameralabel_6.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_6.setFont(font)
        self.cameralabel_6.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_6.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_6.setWordWrap(False)
        self.cameralabel_6.setObjectName("cameralabel_6")
        self.aspectratio169 = QtWidgets.QCheckBox(self.centralwidget)
        self.aspectratio169.setGeometry(QtCore.QRect(380, 580, 20, 21))
        self.aspectratio169.setAutoFillBackground(False)
        self.aspectratio169.setText("")
        self.aspectratio169.setIconSize(QtCore.QSize(16, 16))
        self.aspectratio169.setObjectName("aspectratio169")
        self.cameralabel_7 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_7.setGeometry(QtCore.QRect(540, 560, 111, 61))
```

```python
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_7.sizePolicy().hasHeightForWidth())
        self.cameralabel_7.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_7.setFont(font)
        self.cameralabel_7.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_7.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_7.setWordWrap(False)
        self.cameralabel_7.setObjectName("cameralabel_7")
        self.cameralabel_8 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_8.setGeometry(QtCore.QRect(610, 570, 171, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_8.sizePolicy().hasHeightForWidth())
        self.cameralabel_8.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_8.setFont(font)
        self.cameralabel_8.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_8.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_8.setWordWrap(False)
        self.cameralabel_8.setObjectName("cameralabel_8")
        self.horizontalSlider = QtWidgets.QSlider(self.centralwidget)
        self.horizontalSlider.setGeometry(QtCore.QRect(640, 580, 111, 16))
        self.horizontalSlider.setMinimum(1)
        self.horizontalSlider.setMaximum(3)
        self.horizontalSlider.setPageStep(1)
        self.horizontalSlider.setProperty("value", 1)
        self.horizontalSlider.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider.setObjectName("horizontalSlider")
        MainWindow.setCentralWidget(self.centralwidget)
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Eye Mouse Controller"))
        self.EnterButton.setText(_translate("MainWindow", "Enter"))
        self.EnterButton.setShortcut(_translate("MainWindow", "Return"))
```

```python
        self.Eyemouselabel.setText(_translate("MainWindow", "Eye Mouse Controller"))
        self.cameralabel.setText(_translate("MainWindow", "Camera Number"))
        self.lineEdit.setText(_translate("MainWindow", "1"))
        self.cameralabel_3.setText(_translate("MainWindow", "NOTE: Only one person in
frame is needed for program to work fine"))
        self.cameralabel_4.setText(_translate("MainWindow", "If no result after pressing enter
then camera number is wrong "))
        self.cameralabel_5.setText(_translate("MainWindow", "Inverse Camera:"))
        self.GithubButton.setShortcut(_translate("MainWindow", "Return"))
        self.cameralabel_6.setText(_translate("MainWindow", "Aspect Ratio 16:9:"))
        self.cameralabel_7.setText(_translate("MainWindow", "Illumination:"))
        self.cameralabel_8.setText(_translate("MainWindow", "1              2              3"))
    def Enterbuttonclicked(self,savedornot):
        try:
            if savedornot==False:
                camerainput=int(self.lineEdit.text())
                cameracheck=self.inversecameracheck.isChecked()
                aspectratio169check=self.aspectratio169.isChecked()
                open("cameranumbersaved.txt","r+").truncate()
                if camerainput==None:
                    camerainput=1
                open("cameranumbersaved.txt","r+").write(str(camerainput))

illumination=int(self.horizontalSlider.value())eyemouse=eyetracking_slowpc.eye_mouse(ca
merainput,cameracheck,aspectratio169check,illumination)
                eyemouse.eyetrack()
            else:
                self.lineEdit.setText(self.cameranumbersaved)
                camerainput=int(self.lineEdit.text())
                cameracheck=self.inversecameracheck.isChecked()
                aspectratio169check=self.aspectratio169.isChecked()
                illumination=int(self.horizontalSlider.value())
eyemouse=eyetracking_slowpc.eye_mouse(camerainput,cameracheck,aspectratio169check,ill
umination)
                eyemouse.eyetrack()
        except(Exception):
            pass
    def linktogithub(self, link):
        self.link=link
        webbrowser.open(self.link, new=0)

if__name_== "_main_":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    ui.Enterbuttonclicked(True)
    sys.exit(app.exec_())
```

Eyetracking_slowpc.py:
Code:

```python
from PyQt5 import QtCore, QtGui, QtWidgets
import eyetracking_slowpc
from PyQt5 import sip
import cv2 as cv
cv2=cv
import dlib
import mousecontrol_eye
from win32.win32api import GetSystemMetrics
from pynput.mouse import Listener,Button,Controller
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import QCoreApplication
import time
import webbrowser
from imutils.video import WebcamVideoStream
import numpy as np
class Ui_MainWindow(object):
    def_init_(self):
        self.f=open("cameranumbersaved.txt","r+")
        self.cameranumbersaved=self.f.read()
        if(self.cameranumbersaved==""):
            open("cameranumbersaved.txt","r+").write("1")
            self.cameranumbersaved=self.f.read()
        self.f.close()
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.setEnabled(True)
        MainWindow.resize(1092, 684)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())
        MainWindow.setSizePolicy(sizePolicy)
        MainWindow.setContextMenuPolicy(QtCore.Qt.NoContextMenu)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("icon.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        MainWindow.setWindowIcon(icon)
        MainWindow.setToolTip("")
        MainWindow.setStatusTip("")
        MainWindow.setWhatsThis("")
        MainWindow.setAccessibleName("")
        MainWindow.setAccessibleDescription("")
        MainWindow.setAutoFillBackground(False)
        MainWindow.setWindowFilePath("")
        MainWindow.setDocumentMode(False)
        MainWindow.setTabShape(QtWidgets.Q TabWidget.Rounded)
        MainWindow.setDockNestingEnabled(False)
```

33

```python
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.EnterButton = QtWidgets.QPushButton(self.centralwidget)
        self.EnterButton.setGeometry(QtCore.QRect(590, 520, 121, 51))
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(18)
        self.EnterButton.setFont(font)
        self.EnterButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.EnterButton.setIconSize(Q tCore.QSize(16, 16))
        self.EnterButton.setObjectName("EnterButton")
        self.EnterButton.clicked.connect(self.Enterbuttonclicked)
        self.Eyemouselabel = QtWidgets.QLabel(self.centralwidget)
        self.Eyemouselabel.setGeometry(QtCore.QRect(130, -10, 761, 91))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.Eyemouselabel.sizePolicy().hasHe ightForWidth())
        self.Eyemouselabel.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(20)
        font.setBold(False)
        font.setWeight(50)
        self.Eyemouselabel.setFont(font)
        self.Eyemouselabel.setTextFormat(QtCore.Qt.AutoText)
        self.Eyemouselabel.setAlignment(QtCore.Qt.AlignCenter)
        self.Eyemouselabel.setWordWrap(False)
        self.Eyemouselabel.setObjectName("Eyemouselabel")
        self.instructionimagelabel = QtWidgets.QLabel(self.centralwidget)
        self.instructionimagelabel.setGeometry(QtCore.QRect(180, 70, 721, 441))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.instructionimagelabel.sizePolicy().hasHeightForWidth())
        self.instructionimagelabel.setSizePolicy(sizePolicy)
        self.instructionimagelabel.setText("")
        self.instructionimagelabel.setPixmap(QtGui.QPixmap("instruction.png"))
        self.instructionimagelabel.setAlignment(QtCore.Qt.AlignCenter)
        self.instructionimagelabel.setObjectName("instructionimagelabel")
        self.cameralabel = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel.setGeometry(QtCore.QRect(310, 510, 211, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel.sizePolicy().hasHeightForWidth())
        self.cameralabel.setSizePolicy(sizePolicy)
```

```python
font = QtGui.QFont()
font.setFamily("Rockwell")
font.setPointSize(20)
font.setBold(False)
font.setWeight(50)
self.cameralabel.setFont(font)
self.cameralabel.setTextFormat(QtCore.Qt.AutoText)
self.cameralabel.setAlignment(QtCore.Qt.AlignCenter)
self.cameralabel.setWordWrap(False)
self.cameralabel.setObjectName("cameralabel")
self.lineEdit = QtWidgets.QLineEdit(self.centralwidget)
self.lineEdit.setGeometry(QtCore.QRect(530, 520, 51, 51))
font = QtGui.QFont()
font.setFamily("Rockwell")
font.setPointSize(18)
self.lineEdit.setFont(font)
self.lineEdit.setMaxLength(1)
self.lineEdit.setAlignment(QtCore.Qt.AlignCenter)
self.lineEdit.setObjectName("lineEdit")
self.cameralabel_3 = QtWidgets.QLabel(self.centralwidget)
self.cameralabel_3.setGeometry(QtCore.QRect(310, 620, 421, 61))
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.cameralabel_3.sizePolicy().hasHeightForWidth())
self.cameralabel_3.setSizePolicy(sizePolicy)
font = QtGui.QFont()
font.setFamily("Rockwell")
font.setPointSize(9)
font.setBold(False)
font.setWeight(50)
self.cameralabel_3.setFont(font)
self.cameralabel_3.setTextFormat(QtCore.Qt.AutoText)
self.cameralabel_3.setAlignment(QtCore.Qt.AlignCenter)
self.cameralabel_3.setWordWrap(False)
self.cameralabel_3.setObjectName("cameralabel_3")
self.cameralabel_4 = QtWidgets.QLabel(self.centralwidget)
self.cameralabel_4.setGeometry(QtCore.QRect(310, 640, 421, 61))
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.cameralabel_4.sizePolicy().hasHeightForWidth())
self.cameralabel_4.setSizePolicy(sizePolicy)
font = QtGui.QFont()
font.setFamily("Rockwell")
font.setPointSize(9)
font.setBold(False)
font.setWeight(50)
```

```
        self.cameralabel_4.setFont(font)
        self.cameralabel_4.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_4.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_4.setWordWrap(False)
        self.cameralabel_4.setObjectName("cameralabel_4")
        self.cameralabel_5 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_5.setGeometry(QtCore.QRect(410, 560, 111, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_5.sizePolicy().hasHeightForWidth())
        self.cameralabel_5.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_5.setFont(font)
        self.cameralabel_5.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_5.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_5.setWordWrap(False)
        self.cameralabel_5.setObjectName("cameralabel_5")
        self.inversecameracheck = QtWidgets.QCheckBox(self.centralwidget)
        self.inversecameracheck.setGeometry(QtCore.QRect(520, 580, 20, 21))
        self.inversecameracheck.setAutoFillBackground(False)
        self.inversecameracheck.setText("")
        self.inversecameracheck.setIconSize(QtCore.QSize(16, 16))
        self.inversecameracheck.setObjectName("inversecameracheck")
        self.GithubLogo = QtWidgets.QLabel(self.centralwidget)
        self.GithubLogo.setGeometry(QtCore.QRect(660, 0, 71, 71))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.GithubLogo.sizePolicy().hasHeightForWidth())
        self.GithubLogo.setSizePolicy(sizePolicy)
        self.GithubLogo.setText("")
        self.GithubLogo.setPixmap(QtGui.QPixmap("githublogo.png"))
        self.GithubLogo.setAlignment(QtCore.Qt.AlignCenter)
        self.GithubLogo.setObjectName("GithubLogo")
        self.GithubButton = QtWidgets.QPushButton(self.centralwidget)
        self.GithubButton.setGeometry(QtCore.Q Rect(660, 0, 71, 71))
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(18)
        self.GithubButton.setFont(font)
        self.GithubButton.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))
        self.GithubButton.setStyleSheet("\n"
"background-color:  rgba(255, 255, 255, 0);")
```

```python
        self.GithubButton.setText("")
        self.GithubButton.setIconSize(QtCore.QS ize(16, 16))
        self.GithubButton.setObjectName("GithubButton")
        url="https://github.com/HARSHSINGH0/EYE_MOUSE"
        self.GithubButton.clicked.connect(lambda:self.linktogithub(url))
        self.cameralabel_6 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_6.setGeometry(QtCore.QRect(260, 560, 111, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_6.sizePolicy().hasHeightForWidth())
        self.cameralabel_6.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_6.setFont(font)
        self.cameralabel_6.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_6.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_6.setWordWrap(False)
        self.cameralabel_6.setObjectName("cameralabel_6")
        self.aspectratio169 = QtWidgets.QCheckBox(self.centralwidget)
        self.aspectratio169.setGeometry(QtCore.QRect(380, 580, 20, 21))
        self.aspectratio169.setAutoFillBackground(False)
        self.aspectratio169.setText("")
        self.aspectratio169.setIconSize(QtCore.QSize(16, 16))
        self.aspectratio169.setObjectName("aspectratio169")
        self.cameralabel_7 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_7.setGeometry(QtCore.QRect(540, 560, 111, 61))
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_7.sizePolicy().hasHeightForWidth())
        self.cameralabel_7.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_7.setFont(font)
        self.cameralabel_7.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_7.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_7.setWordWrap(False)
        self.cameralabel_7.setObjectName("cameralabel_7")
        self.cameralabel_8 = QtWidgets.QLabel(self.centralwidget)
        self.cameralabel_8.setGeometry(QtCore.QRect(610, 570, 171, 61))
```

```
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.Preferred)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.cameralabel_8.sizePolicy().hasHeightForWidth())
        self.cameralabel_8.setSizePolicy(sizePolicy)
        font = QtGui.QFont()
        font.setFamily("Rockwell")
        font.setPointSize(9)
        font.setBold(False)
        font.setWeight(50)
        self.cameralabel_8.setFont(font)
        self.cameralabel_8.setTextFormat(QtCore.Qt.AutoText)
        self.cameralabel_8.setAlignment(QtCore.Qt.AlignCenter)
        self.cameralabel_8.setWordWrap(False)
        self.cameralabel_8.setObjectName("cameralabel_8")
        self.horizontalSlider = QtWidgets.QSlider(self.centralwidget)
        self.horizontalSlider.setGeometry(QtCore.QRect(640, 580, 111, 16))
        self.horizontalSlider.setMinimum(1)
        self.horizontalSlider.setMaximum(3)
        self.horizontalSlider.setPageStep(1)
        self.horizontalSlider.setProperty("value", 1)
        self.horizontalSlider.setOrientation(QtCore.Qt.Horizontal)
        self.horizontalSlider.setObjectName("horizontalSlider")
        MainWindow.setCentralWidget(self.centralwidget)
self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Eye Mouse Controller"))
        self.EnterButton.setText(_translate("MainWindow", "Enter"))
        self.EnterButton.setShortcut(_translate("MainWindow", "Return"))
        self.Eyemouselabel.setText(_translate("MainWindow", "Eye Mouse Controller"))
        self.cameralabel.setText(_translate("MainWindow", "Camera Number"))
        self.lineEdit.setText(_translate("MainWindow", "1"))
        self.cameralabel_3.setText(_translate("MainWindow", "NOTE: Only one person in
frame is needed for program to work fine"))
        self.cameralabel_4.setText(_translate("MainWindow", "If no result after pressing enter
then camera number is wrong "))
        self.cameralabel_5.setText(_translate("MainWindow", "Inverse Camera:"))
        self.GithubButton.setShortcut(_translate("MainWindow", "Return"))
        self.cameralabel_6.setText(_translate("MainWindow", "Aspect Ratio 16:9:"))
        self.cameralabel_7.setText(_translate("MainWindow", "Illumination:"))
        self.cameralabel_8.setText(_translate("MainWindow", "1          2          3"))
    def Enterbuttonclicked(self,savedornot):
        try:
            if savedornot==False:
                camerainput=int(self.lineEdit.text())
                cameracheck=self.inversecameracheck.isChecked()
                aspectratio169check=self.aspectratio169.isChecked()
```

```
            open("cameranumbersaved.txt","r+").truncate()
            if camerainput==None:
                camerainput=1
            open("cameranumbersaved.txt","r+").write(str(camerainput))
illumination=int(self.horizontalSlider.value())eyemouse=eyetracking_slowpc.eye_mouse(ca
merainput,cameracheck,aspectratio169check,illumination)
            eyemouse.eyetrack()
        else:
            self.lineEdit.setText(self.cameranumbersaved)
            camerainput=int(self.lineEdit.text())
            cameracheck=self.inversecameracheck.isChecked()
            aspectratio169check=self.aspectratio169.isChecked()
            illumination=int(self.horizontalSlider.value())
eyemouse=eyetracking_slowpc.eye_mouse(camerainput,cameracheck,aspectratio169check,ill
umination)
            eyemouse.eyetrack()
    except(Exception):
        pass
  def linktogithub(self, link):
      self.link=link
      webbrowser.open(self.link, new=0)

if __name__ == "_main_":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    ui.Enterbuttonclicked(True)
    sys.exit(app.exec_())
```

Eyetracking_slowpc.py:
Code:

```python
import cv2 as cv
cv2=cv
import dlib
import mousecontrol_eye
from win32.win32api import GetSystemMetrics
import time
from pynput.mouse import Listener,Button,Controller
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
from imutils.video import WebcamVideoStream
import numpy as np
class eye_mouse:
    def __init__(self,camerainput,cameracheck,aspectratio169,illumination):
        self.camerainput=int(camerainput)
        self.blinking_frames=0
        self.mousecontrol=mousecontrol_eye.mousecontrol()
        self.cameracheck=cameracheck
        self.aspectratio169=aspectratio169
        width = GetSystemMetrics(0)
        height = GetSystemMetrics(1)
        middlepoint1=width/2
        middlepoint2=height/2
        self.mousecontrol.firstpos(middlepoint1,middlepoint2)
        self.illumination=illumination
        if self.illumination==None:
            self.illumination=1
    def rescaleFrame(self,frame):
        dimension=(320,240)
        return cv.resize(frame,dimension,interpolation=cv.IN TER_AREA)
    def midlinepoint(self,p1,p2):
        return int((p1.x+p2.x)/2),int((p1.y+p2.y)/2)
    def aspectratiochanger(self,ratio,frame):
        if ratio=="16by9":
            dimension=(426,240)
            return cv.resize(frame,dimension,interpolation=cv.INTER_AREA)
    def adjust_gamma(self,frame, gamma=1.0):
        if gamma==1:
            return frame
        else:
            invGamma = 1.0 / gamma
            table = np.array([((i / 255.0) ** invGamma) * 255
                for i in np.arange(0, 256)]).astype("uint8")
            return cv2.LUT(frame, table)
    def eyetrack(self):
        blinking_frames=self.blinking_frames
        blinking_frames1=self.blinking_frames
        self.cap=WebcamVideoStream(src=self.camerainput-1).start()
        cameracheck=self.cameracheck
```

```
        self.detector=dlib.get_frontal_face_detector()
        self.predictor=dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
        mouse=Controller()
        while True:
            try:
                errornumber=0
                if cameracheck==False:
                    frame=self.cap.read()
                    frame=self.adjust_gamma(frame,self.illumination)
                    gray=cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
                    self.aspectratioranned=False
                    if self.aspectratio169==True:
                        gray=self.aspectratiochanger("16by9",gray)
                        frame=self.aspectratiochanger("16by9",frame)
                        self.aspectratioranned=True
                    else:
                        gray=self.rescaleFrame(gray)
                        frame=self.rescaleFrame(frame)
                    faces=self.detector(gray)
                else:
                    frame=self.cap.read()
                    frame=self.adjust_gamma(frame,self.illumination)
                    gray=cv.cvtColor(frame,cv.COLOR_BGR2GRAY)
                    if self.aspectratio169==True:
                        gray=cv.flip(self.aspectratiochanger("16by9",gray),1)
                        frame=cv.flip(self.aspectratiochanger("16by9",frame),1)
                        self.aspectratioranned=True
                    else:
                        gray=cv.flip(self.rescaleFrame(gray),1)
                        frame=cv.flip(self.rescaleFrame(frame),1)
                    faces=self.detector(gray)
                cv.putText(frame,"Q to
exit",(120,50),cv.FONT_HERSHEY_DUPLEX,0.5,(255,255,255),1)
                for face in faces:
                    x,y=face.left(),face.right()
                    x1,y1=face.top(),face.bottom()
                    facerect=cv2.rectangle(frame,(x,x1),(y,y1),(255,255,255),1)
                    landmarks=self.predictor(gray,face)
                    noselandmark=(landmarks.part(30).x,landmarks.part(30)
                    eyestonosepointx,eyestonosepointy=landmarks.part(30).x,landmarks.part(30).y
                    if self.aspectratioranned==True:
                        nose_to_cursorx=213
                        nose_to_cursory=150
                    else:
                        nose_to_cursorx=160
nose_to_cursory=150cv.rectangle(frame,(eyestonosepointx,eyestonosepointy),(eyestonosepoi
ntx,eyestonosepointy),(255,255,255),thickness=4)
cv.rectangle(frame,(nose_to_cursorx,nose_to_cursory),(nose_to_cursorx,nose_to_cursory),(0
,0,0),thickness=5)
```

41

```python
cv.line(frame,(eyestonosepointx,eyestonosepointy),(nose_to_cursorx,nose_to_cursory),(255,
255,255),thickness=1)
          center_coordinates  = (325, 270)
          radius  = 30
          color = (220,220,220)
          thickness=2,lineType=cv.FILLED)
          positivecursorvalue=15
          negativesursorvalue=-15
          if((eyestonosepointx-nose_to_cursorx)>positivecursorvalue):
            if((eyestonosepointx-nose_to_cursorx)>60):
              mouse.move(10,0)
            elif((eyestonosepointx- nose_to_cursorx)>30):
              mouse.move(6,0)
            elif((eyestonosepointx- nose_to_cursorx)>15):
              mouse.move(2,0)
          if((eyestonosepointx-nose_to_cursorx)<negativesursorvalue):
            if((eyestonosepointx-nose_to_cursorx)<-60):
              mouse.move(-10,0)
            elif((eyestonosepointx- nose_to_cursorx)<-30):
              mouse.move(-6,0)
            elif((eyestonosepointx- nose_to_cursorx)<-15):
              mouse.move(-2,0)
          if(eyestonosepointy- nose_to_cursory)<positivecursorvalue:
            if(eyestonosepointy-nose_to_cursory)<30:
              mouse.move(0,-8)
            elif(eyestonosepointy-nose_to_cursory)<15:
              mouse.move(0,-4)
          if(eyestonosepointy- nose_to_cursory)>negativesursorvalue:
            if(eyestonosepointy-nose_to_cursory)>-30:
              mouse.move(0,8)
            elif(eyestonosepointy-nose_to_cursory)>-15:
              mouse.move(0,4)
          left_point=(landmarks.part(36).x,landmarks.part(36).y)
          right_point=(landmarks.part(39).x,landmarks.part(39).y)
          hor_line=cv.line(frame,left_point,right_point,(255,255,255),1)
          up_point=(self.midlinepoint(landmarks.part(37),landmarks.part(38)))
          down_point=(self.midlinepoint(landmarks.part(41),landmarks.part(40)))
          ver_line=cv.line(frame,up_point,down_point,(255,255,255),1)
          left_point_r=(landmarks.part(42).x,landmarks.part(42).y)
          right_point_r=(landmarks.part(45).x,landmarks.part(45).y)
          hor_line_r=cv.line(frame,left_point_r,right_point_r,(255,255,255),1)
          up_point_r=(self.midlinepoint(landmarks.part(43),landmarks.part(44)))
          down_point_r=(self.midlinepoint(landmarks.part(47),landmarks.part(46)))
          ver_line_r=cv.line(frame,up_point_r,down_point_r,(255,255,255),1)
          value_of_blink=-3
          if((y1-x1)>123):
            value_of_blink=-5
          elif((y1-x1)>102):
            value_of_blink=-4
          elif((y1-x1)>85):
```

```
                    value_of_blink=-3
              elif((y1-x1)>71):
                 value_of_blink=-2
                 cv.putText(frame,"dont go far than
this",(50,70),cv.FONT_HERSHEY_DUPLEX,0.5,(255,255,255),1)
              elif((y1-x1)<73):
                 cv.putText(frame,"come close to the
camera",(50,70),cv.FONT_HERSHEY_DUPLEX,0.5,(255,255,255),1)
                 value_of_blink=10
           if((up_point[1]-down_point[1])>=value_of_blink):
                 blinking_frames+=1
                 if (blinking_frames>3):
                    blinking_frames=0
                    cv.putText(frame,"Left
click",(125,65),cv.FONT_HERSHEY_DUPLEX,0.5,(0,0,0),1)
                    self.mousecontrol.left_click()
           else:
                 while blinking_frames!=0:
                    blinking_frames-=1
           if((up_point_r[1]-down_point_r[1])>=value_of_blink):
                 blinking_frames1+=1
                 if (blinking_frames1>3):
                    blinking_frames1=0
                    cv.putText(frame,"Right
click",(125,65),cv.FONT_HERSHEY_DUPLEX,0.5,(0,0,0),1)
                    self.mousecontrol.right_click()
           else:
                 while blinking_frames1!=0:
                    blinking_frames1-=1
        cv.imshow("frame",frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
           cv2.destroyAllWindows()
           self.cap.stop()
           break
     except:
        cv2.destroyAllWindows()
        self.cap.stop()
        break
cv2.destroyAllWindows()
```

Setup.py:
Code:

```python
import cx_Freeze
import os
build_options = {'packages': [], 'excludes': []}
import cv2 as cv
cv2=cv
import dlib
import mousecontrol_eye
from win32.win32api import GetSystemMetrics
import time
from pynput.mouse import Listener,Button,Controller
import sys
from PyQt5 import QtCore, QtGui, QtWidgets
import webbrowser
import numpy
from imutils.video import WebcamVideoStream
base = 'Win32GUI' if sys.platform=='win32' else None
executables = [
    cx_Freeze.Executable('eyemouseinterface_fastpc.py', base=base, target_name =
'eyemouseinterface_fastpc.py',icon='icon.ico')
]
PYTHON_INSTALL_DIR = os.path.dirname(os.path.dirname(os._file_))
os.environ['TCL_LIBRARY'] = os.path.join(PYTHON_INSTALL_DIR, 'tcl', 'tcl8.6')
os.environ['TK_LIBRARY'] = os.path.join(PYTHON_INSTALL_DIR, 'tcl', 'tk8.6')
include_files = [(os.path.join(PYTHON_INSTALL_DIR, 'DLLs', 'tk86t.dll'),
os.path.join('lib', 'tk86t.dll')),
            (os.path.join(PYTHON_INSTALL_DIR, 'DLLs', 'tcl86t.dll'), os.path.join('lib',
'tcl86t.dll'))]
cx_Freeze.setup(name='Eye Mouse Controller',
    version = '1',
    description = 'Handle Mouse Control with your face',
    options = {'build_exe':
{"packages":["webbrowser","zmq","cv2","PyQt5","dlib","win32","pynput","sys","tkinter","P
yQt5.QtCore","PyQt5.QtGui","PyQt5.QtWidgets","time","numpy"],"include_files":include_f
iles+["githublogo.png","icon.ico","eyemouseinterface_fastpc.py","eyetracking_fastpc.py","m
ousecontrol_eye.py","instruction.png","shape_predictor_68_face_landmarks.dat","cameranu
mbersaved.txt"]}},
    executables = executables)
```

5.2 TEST CASES

NO.	Test	Expected Results	Actual Result	Status
1	Running executable (application)	Interface opened	Interface opened	Pass
2	Checkbox test	Working and editable checkbox	Working and editable checkbox	Pass
3	Checkbox value test	Only takes an integer as camera number	Only takes an integer as camera number	Pass
4	Load Previous camera value	Successfully loads previously added value	Successfully Loads previously added value	Pass
5	Opencv frame check	Opencv frame runs on startup	Opencv frame runs on startup	Pass
6	Aspect ration check	Aspect ratio checkbox works	Aspect ratio checkbox works	Pass
7	Changing aspect ratio	Aspect ratio changes to 16:9	Aspect ratio changes to 16:9	Pass

8	Inverse camera check	Inverse camera checkbox button works	Inverse camera checkbox button works	Pass
9	Flipping video frame	Video gets inverted if the checkbox is pressed	Video gets inverted if the checkbox is pressed	Pass
10	Illumination slider	Illumination slider slides if dragged	Illumination slider slides if dragged	Pass
11	Illumination slider works forwards	Illumination slider increases gamma value of video	Illumination slider increases gamma value of video	Pass
12	Illumination slider works backward	Illumination slider decreases gamma value of video	Illumination slider decreases gamma value of video	Pass
13	Enter button check	Enter button take press and capture values	Enter button take press and capture values	Pass
14	Value implement in OpenCV frame	After Enterbutton pressed OpenCV runs with custom settings	After Enterbutton pressed OpenCV runs with custom settings	Pass
15	Opencv frame close check	Opencv frame closes when the letter 'Q' is pressed	Opencv frame closes when the letter 'Q' is pressed	Pass

16	Face detection	Face detects And draws a rectangle around the face	Face detects and draws a rectangle around the face	Pass
17	Cursor movement	The cursor moves on face movement	The cursor moves on face movement	Pass
18	Blink detection	Mouse clicks on blink detection	Mouse clicks on blink detection	Pass
19	Cursor speed check	Cursor speed gradually increases on fast face movement	Cursor speed gradually increases on fast face movement	Pass
20	Midpoint cursor movement	No movement around the midpoint of the camera circle	No movement around the midpoint of the camera circle	Pass

CHAPTER 6. CONCLUSION AND FUTURE WORK

6.1 Conclusion:

This application is to be and will be used for the benefit of an especially disabled individual who wants to achieve or use it in the same way as ordinary people. This application can be an essential part of the people who are lazy to use a mouse when writing with a keyboard.

The Goal and soul purpose of making this software is to simplify the use of the hand in just one place i.e. keyboard, keyboard-centric people don't like to flip their hands to use a mouse when handling the keyboard. It can also be used by people with severe disabilities who can use their eyes to communicate to the computers

One can see the results of what eye-tracking technology can work when we using it with an interface of a handy computer.

6.2 Advantages:

1. A one of a kind application where the cursor is controlled with a face.
2. Users can get creative and use it in-game for many game functions.
3. Any individual in distress can be helped with communication.
4. Camera input can be changed and use multiple webcams.
5. One can change illumination in video screens according to their background illumination.
6. Easy to control setting and one can change aspect ratio settings.
7. Stores previously added a camera number which is used in a future boot.

6.3 Limitations:

1. Need fast computing for fast video compiling and processing.
2. Few things might need corrections in the initial stages.

3. Requires good quality camera.
4. Currently lacks extensive updating of the user profile.
5. Slow in responsein low light conditions.

6.4 Future Work:

1. In the upcoming updates, the users will be able to update their profiles to a far greater extent.
2. Will make this application for the Linux environment too.
3. The user interface will be made more handy and attractive over time.

CHAPTER 7. REFERENCES

[1] Tobii Eye Tracker 5

The Next Generation of Head and Eye Tracking
https://gaming.tobii.com/product/eye-tracker-5/

[2] Eyeblink detection with OpenCV, Python, and dlib

https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/

[3] Measuring Performance in IPython
https://docs.opencv.org/master/dc/d71/tutorial_py_optimization.html

 [4] Matlab vs. OpenCV:

https://arxiv.org/abs/AlthoughMatlabismoreconvenient,thaninsomecases.