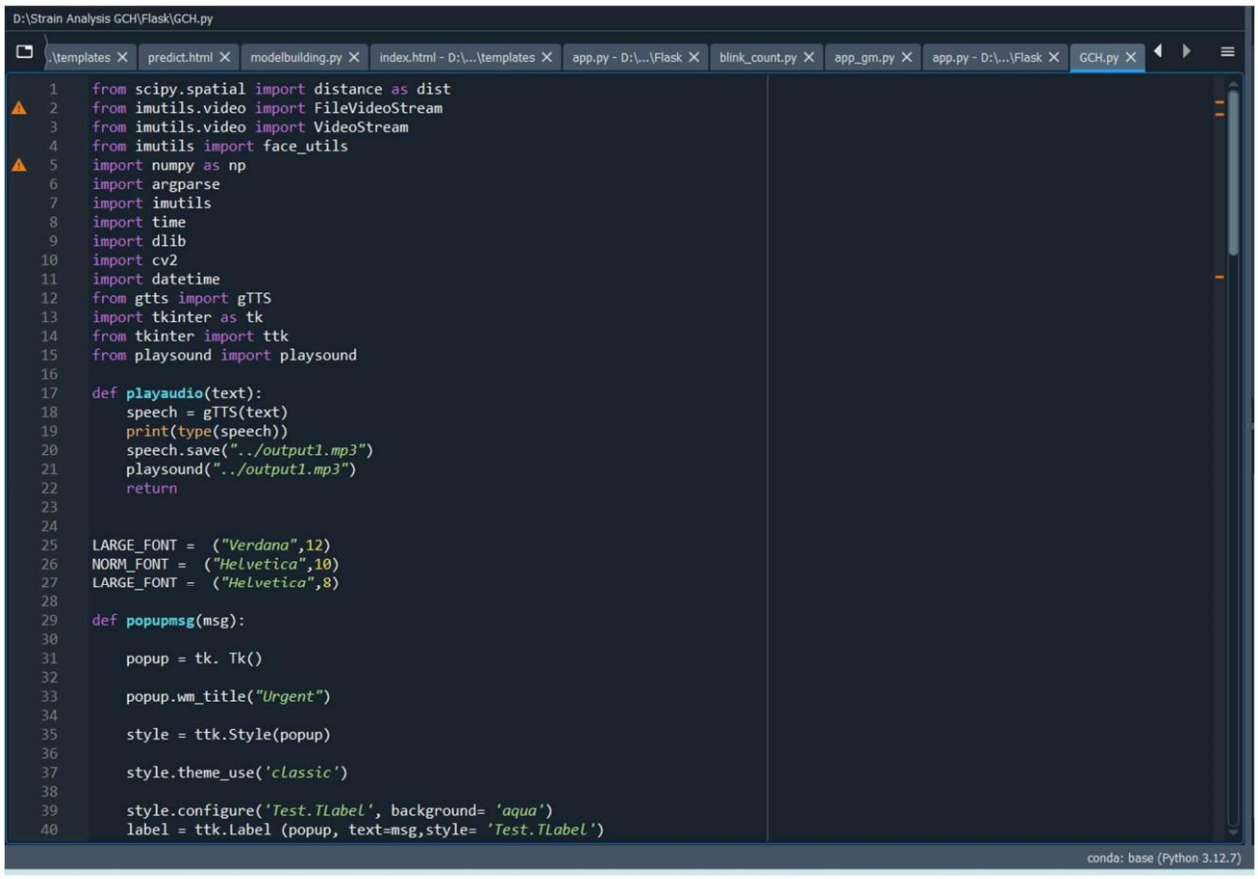
Model Development Phase Template

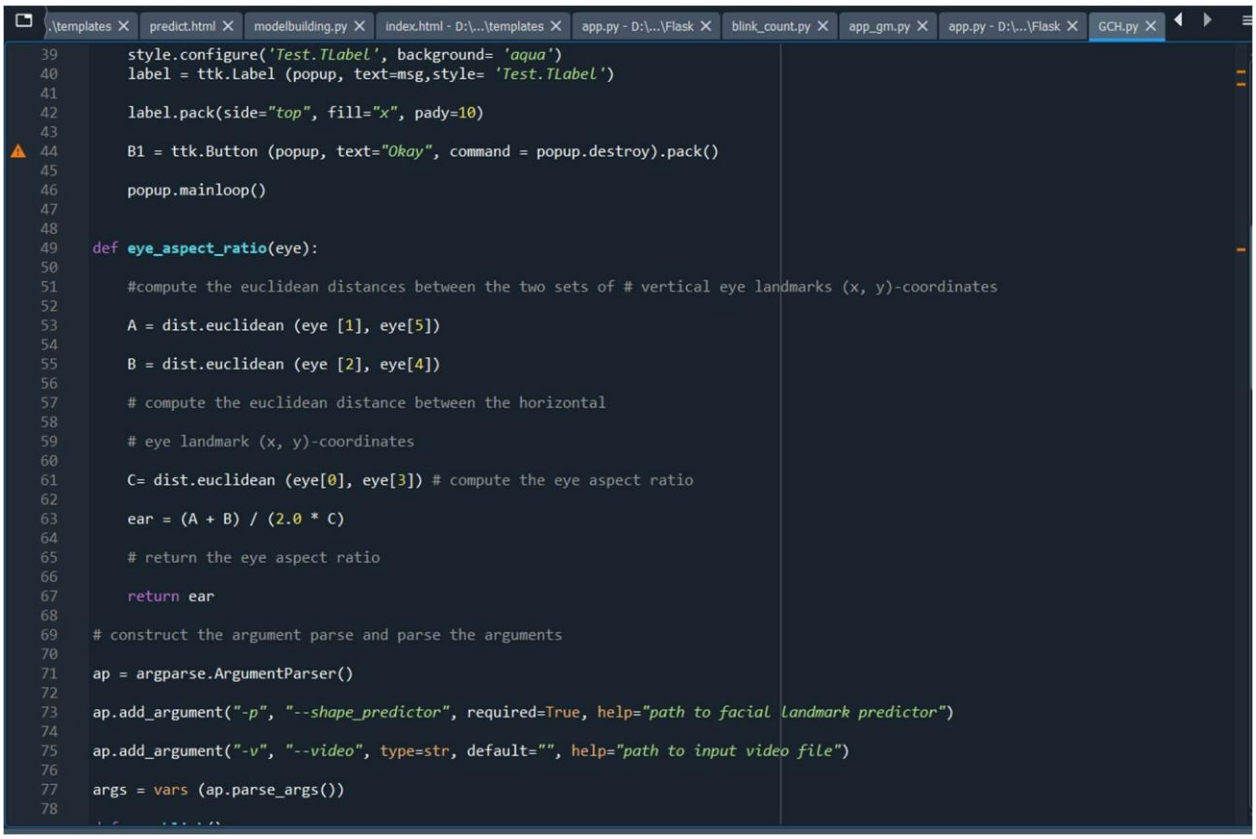
Date	25 September 2024
Team ID	739698
Project Title	Strain analysis based on eye blinking
Maximum Marks	10 Marks

Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):


D:\Strain Analysis GCH\Flask\GCH.py
..\templates
predict.html
modelbuilding.py
index.html - D:\...\templates
app.py - D:\...\Flask
blink_count.py
app_gm.py
app.py - D:\...\Flask
GCH.py
from scipy.spatial import distance as dist
from imutils.video import FileVideoStream
from imutils.video import VideoStream
from imutils import face_utils
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import datetime
from gtts import gTTS
import tkinter as tk
from tkinter import ttk
from playsound import playsound
def playaudio(text):
speech = gTTS(text)
print(type(speech))
speech.save("../output1.mp3")
playsound("../output1.mp3")
return
LARGE_FONT = ("Verdana",12)
NORM_FONT = ("Helvetica",10)
LARGE_FONT = ("Helvetica",8)
def popupmsg(msg):
popup = tk. Tk()
popup.wm_title("Urgent")
style = ttk.Style(popup)
style.theme_use('classic')
style.configure('Test.TLabel', background= 'aqua')
label = ttk.Label (popup, text=msg,style= 'Test.TLabel')
conda: base (Python 3.12.7)



```
    style.configure('Test.TLabel', background= 'aqua')
    label = ttk.Label (popup, text=msg,style= 'Test.TLabel')

    label.pack(side="top", fill="x", pady=10)

    B1 = ttk.Button (popup, text="Okay", command = popup.destroy).pack()

    popup.mainloop()


def eye_aspect_ratio(eye):

    #compute the euclidean distances between the two sets of # vertical eye landmarks (x, y)-coordinates

    A = dist.euclidean (eye [1], eye[5])

    B = dist.euclidean (eye [2], eye[4])

    # compute the euclidean distance between the horizontal

    # eye landmark (x, y)-coordinates

    C= dist.euclidean (eye[0], eye[3]) # compute the eye aspect ratio

    ear = (A + B) / (2.0 * C)

    # return the eye aspect ratio

    return ear

# construct the argument parse and parse the arguments

ap = argparse.ArgumentParser()

ap.add_argument("-p", "--shape_predictor", required=True, help="path to facial landmark predictor")

ap.add_argument("-v", "--video", type=str, default="", help="path to input video file")

args = vars (ap.parse_args())
```

D:\Strain Analysis GCH\Flask\GCH.py

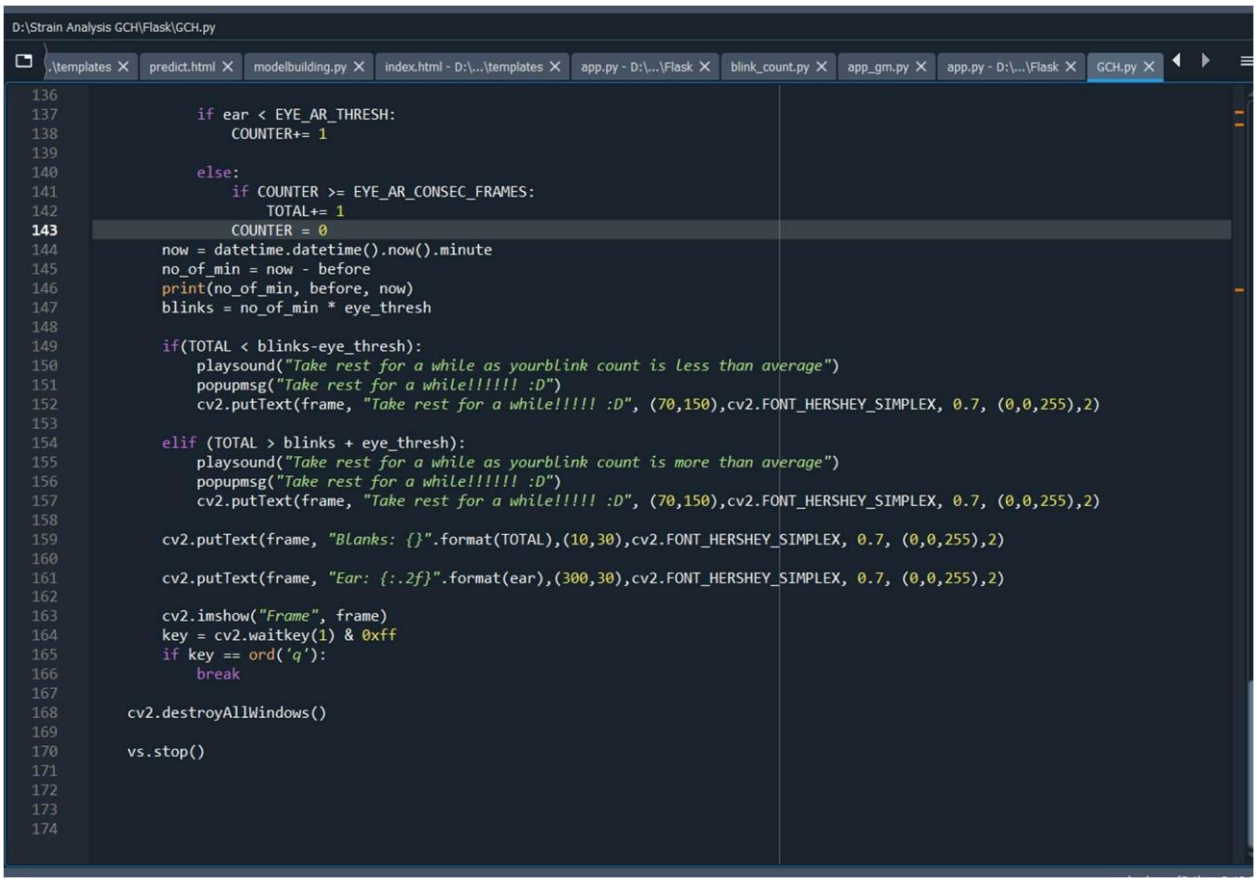
.\templates | predict.html | modelbuilding.py | index.html - D:\...\templates | app.py - D:\...\Flask | blink_count.py | app_gm.py | app.py - D:\...\Flask | GCH.py

```
79  def eye_blink():
80      EYE_AR_THRESH = 0.3
81      EYE_AR_CONSEC_FRAMES = 3
82
83
84      COUNTER = 0
85      TOTAL = 0
86      print("[INFO] loading facial landmark predictor...")
87      detector = dlib.get_frontal_face_detector()
88      predictor = dlib.shape_predictor(args['shape_predictor'])
89
90      #predictor =dlib.shape_predictor(args['shape_predictor'])
91      #predictor = dlib.shape-predictor(args['shape-predictor'])
92      print(type(predictor),predictor)
93
94      (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
95      (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
96
97
98      eye_thresh = 10
99      before = datetime.datetime.now().minute
100
101     if not args.get("video", False):
102         print("[INFO] starting video stream..")
103         vs = VideoStream(src = 0).start()
104         time.sleep(1.0)
105
106     else:
107         print("[INFO] Opening video file...")
108         vs = cv2.VideoCapture(args["video"])
109         time.sleep(1.0)
110
111     while True:
112         frame = vs.read()
113         if frame is None:
114             print("unable to capture")
115             break
116         frame = imutils.resize(frame, width=450)
117         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
118         rects = detector(gray, 0)
```

D:\Strain Analysis GCH\Flask\GCH.py

.\templates | predict.html | modelbuilding.py | index.html - D:\...\templates | app.py - D:\...\Flask | blink_count.py | app_gm.py | app.py - D:\...\Flask | GCH.py

```
116         frame = imutils.resize(frame, width=450)
117         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
118         rects = detector(gray, 0)
119         for rect in rects:
120             shape = predictor(gray,rect)
121             if shape is None:
122                 print("shape predictor returning none")
123                 continue
124             shape = face_utils.shape_to_np(shape)
125             leftEye = shape[lStart:lEnd]
126             rightEye = shape[rStart:rEnd]
127             leftEAR = eye_aspect_ratio(leftEye)
128             rightEAR = eye_aspect_ratio(rightEye)
129 
130             ear =(leftEAR + rightEAR) / 2.0
131 
132             leftEyeHull = cv2.convexHull(leftEye)
133             rightEyeHull = cv2.convexHull(rightEye)
134             cv2.drawContours(frame, [leftEyeHull], -1, (0,255,0),1)
135             cv2.drawContours(frame, [rightEyeHull], -1, (0,255,0),1)
136 
137             if ear < EYE_AR_THRESH:
138                 COUNTER+= 1
139 
140             else:
141                 if COUNTER >= EYE_AR_CONSEC_FRAMES:
142                     TOTAL+= 1
143                 COUNTER = 0
144         now = datetime.datetime().now().minute
145         no_of_min = now - before
146         print(no_of_min, before, now)
147         blinks = no_of_min * eye_thresh
148 
149         if(TOTAL < blinks-eye_thresh):
150             playsound("Take rest for a while as yourblink count is less than average")
151             popupmsg("Take rest for a while!!!!! :D")
152             cv2.putText(frame, "Take rest for a while!!!! :D", (70,150),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255),2)
153 
154         elif (TOTAL > blinks + eye_thresh):
155             playsound("Take rest for a while as yourblink count is more than average")
```



Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
Model 1	Screenshot of the neural network summary	-

Model 2	Screenshot of the neural network summary	-
…	…	…