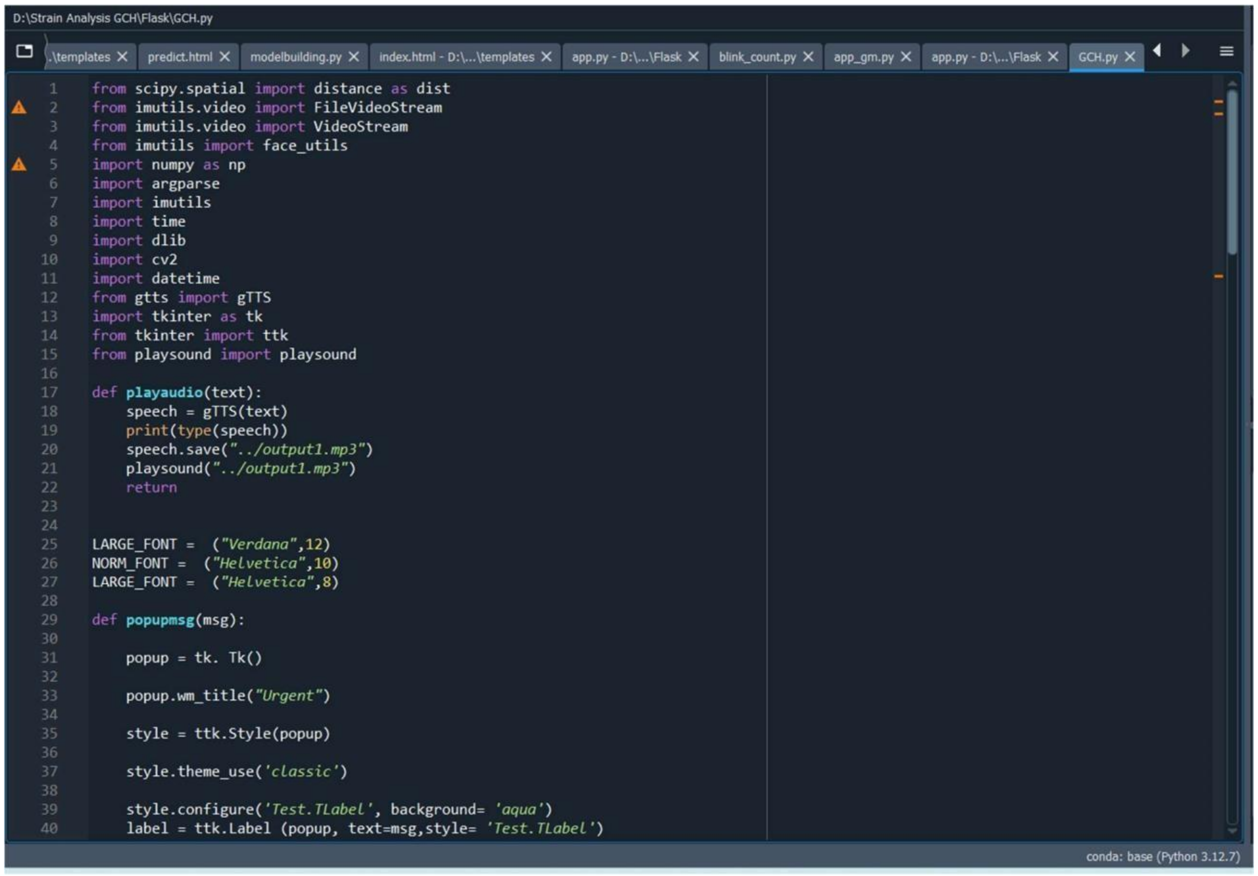
# Model Development Phase Template

| | |
|---|---|
| Date | 25 September 2024 |
| Team ID | 739753 |
| Project Title | Strain analysis based on eye blinking |
| Maximum Marks | 10 Marks |

Initial Model Training Code, Model Validation and Evaluation Report

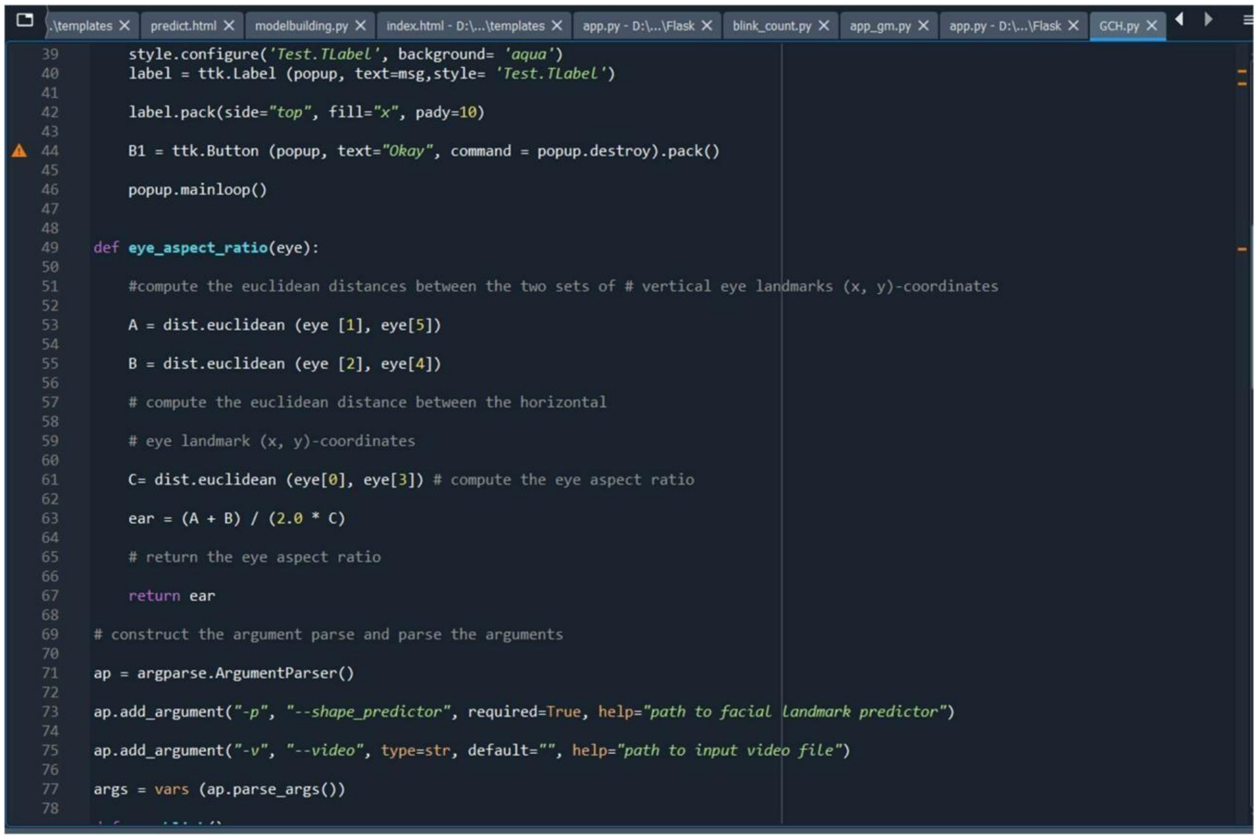
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

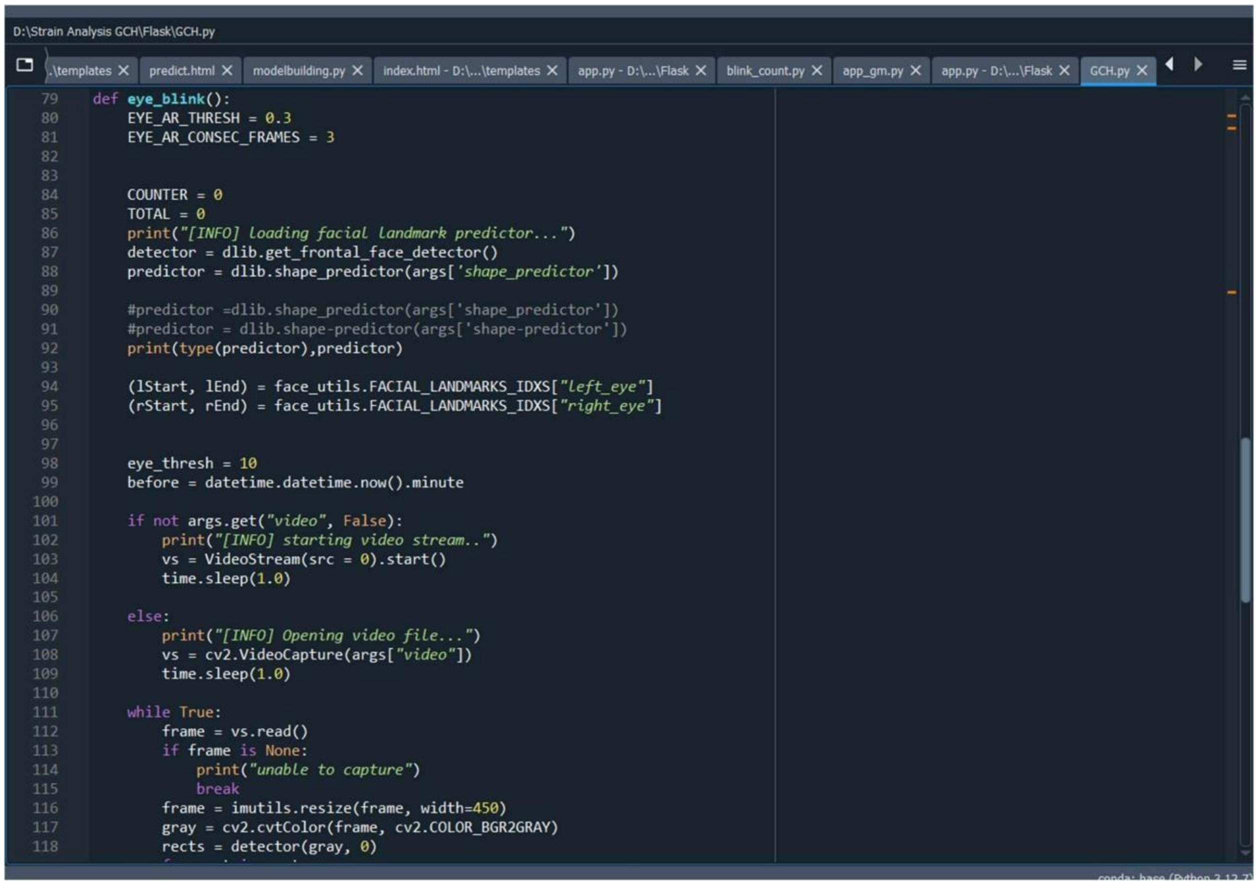

D:\Strain Analysis GCH\Flask\GCH.py

```python
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')
```

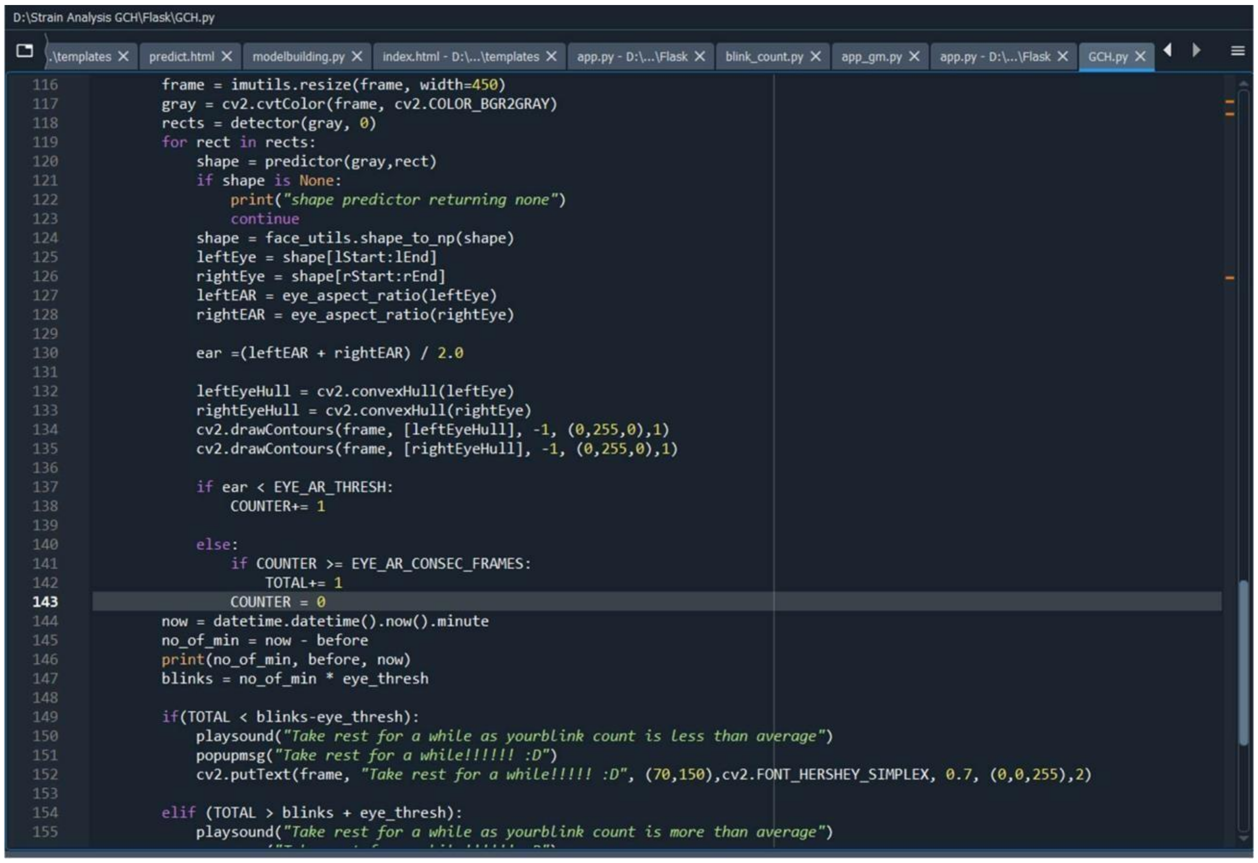
conda: base (Python 3.12.7)



```
    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())

```



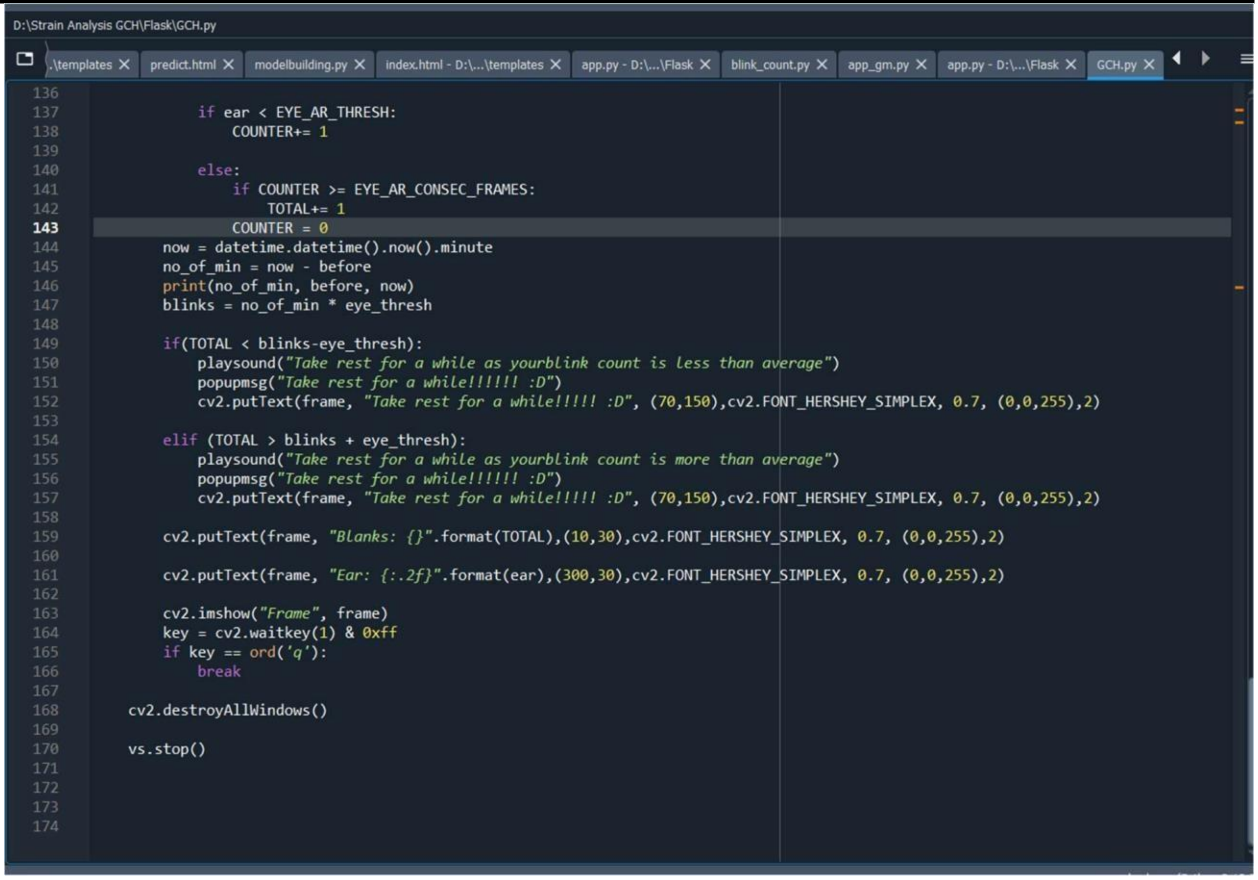
```python
def eye_blink():
    EYE_AR_THRESH = 0.3
    EYE_AR_CONSEC_FRAMES = 3


    COUNTER = 0
    TOTAL = 0
    print("[INFO] loading facial landmark predictor...")
    detector = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor(args['shape_predictor'])

    #predictor =dlib.shape_predictor(args['shape_predictor'])
    #predictor = dlib.shape-predictor(args['shape-predictor'])
    print(type(predictor),predictor)

    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]


    eye_thresh = 10
    before = datetime.datetime.now().minute

    if not args.get("video", False):
        print("[INFO] starting video stream..")
        vs = VideoStream(src = 0).start()
        time.sleep(1.0)

    else:
        print("[INFO] Opening video file...")
        vs = cv2.VideoCapture(args["video"])
        time.sleep(1.0)

    while True:
        frame = vs.read()
        if frame is None:
            print("unable to capture")
            break
        frame = imutils.resize(frame, width=450)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        rects = detector(gray, 0)
```



```python
        frame = imutils.resize(frame, width=450)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        rects = detector(gray, 0)
        for rect in rects:
            shape = predictor(gray,rect)
            if shape is None:
                print("shape predictor returning none")
                continue
            shape = face_utils.shape_to_np(shape)
            leftEye = shape[lStart:lEnd]
            rightEye = shape[rStart:rEnd]
            leftEAR = eye_aspect_ratio(leftEye)
            rightEAR = eye_aspect_ratio(rightEye)

            ear =(leftEAR + rightEAR) / 2.0

            leftEyeHull = cv2.convexHull(leftEye)
            rightEyeHull = cv2.convexHull(rightEye)
            cv2.drawContours(frame, [leftEyeHull], -1, (0,255,0),1)
            cv2.drawContours(frame, [rightEyeHull], -1, (0,255,0),1)

            if ear < EYE_AR_THRESH:
                COUNTER+= 1

            else:
                if COUNTER >= EYE_AR_CONSEC_FRAMES:
                    TOTAL+= 1
                COUNTER = 0
        now = datetime.datetime().now().minute
        no_of_min = now - before
        print(no_of_min, before, now)
        blinks = no_of_min * eye_thresh

        if(TOTAL < blinks-eye_thresh):
            playsound("Take rest for a while as yourblink count is less than average")
            popupmsg("Take rest for a while!!!!!! :D")
            cv2.putText(frame, "Take rest for a while!!!!! :D", (70,150),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255),2)

        elif (TOTAL > blinks + eye_thresh):
            playsound("Take rest for a while as yourblink count is more than average")
```

| Model | Summary | Training and Validation Performance Metrics |
|---|---|---|

| | | |
|---|---|---|
| Model 1 | Screenshot of the neural network summary | - |



```

            if ear < EYE_AR_THRESH:
                COUNTER+= 1

            else:
                if COUNTER >= EYE_AR_CONSEC_FRAMES:
                    TOTAL+= 1
                COUNTER = 0
        now = datetime.datetime().now().minute
        no_of_min = now - before
        print(no_of_min, before, now)
        blinks = no_of_min * eye_thresh

        if(TOTAL < blinks-eye_thresh):
            playsound("Take rest for a while as yourblink count is less than average")
            popupmsg("Take rest for a while!!!!!! :D")
            cv2.putText(frame, "Take rest for a while!!!!! :D", (70,150),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255),2)

        elif (TOTAL > blinks + eye_thresh):
            playsound("Take rest for a while as yourblink count is more than average")
            popupmsg("Take rest for a while!!!!!! :D")
            cv2.putText(frame, "Take rest for a while!!!!! :D", (70,150),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255),2)

        cv2.putText(frame, "Blanks: {}".format(TOTAL),(10,30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255),2)

        cv2.putText(frame, "Ear: {:.2f}".format(ear),(300,30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,255),2)

        cv2.imshow("Frame", frame)
        key = cv2.waitkey(1) & 0xff
        if key == ord('q'):
            break

    cv2.destroyAllWindows()

    vs.stop()
```

Model Validation and Evaluation Report (5 marks):

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