Programming for AI

VISUAL TRYON GLASSES

PROJECT REPORT

BS in Artificial Intelligence

Department of Software Engineering

Faculty of Computer Science & Information Technology

The Superior University, Lahore

Sr.#	Reg.#	Student Name
1.	SU92-BSAIM-F23-107	Ayesha Faisal

Table of Contents

Table of Figures

VISUAL TRY-ON GLASSES:

1 INTRODUCTION:

1.1 Purpose:

The Visual Try-On Glasses application enables users to virtually try on different eyeglass frames in real time using their device’s webcam.

1.2 Technology Stack:

Backend: Python (Flask) web framework

Computer Vision: OpenCV for face and eye detection

Frontend: HTML, CSS, and JavaScript for a modern, responsive user interface

1.3 How It Works:

- The application captures live video from the user's webcam.
- OpenCV detects the user's face and eyes in each video frame.
- When a user selects a glasses frame from the gallery, the system overlays the chosen frame onto their face in the video stream.
- The overlay is dynamically positioned and resized based on the detected facial features for a realistic fit.

1.4 User Interface Features:

- Live video preview with real-time glasses overlay
- Clickable gallery of glasses frames for instant selection
- Responsive design that works on both desktop and mobile devices
- Visual feedback for selected frames (highlighted selection)

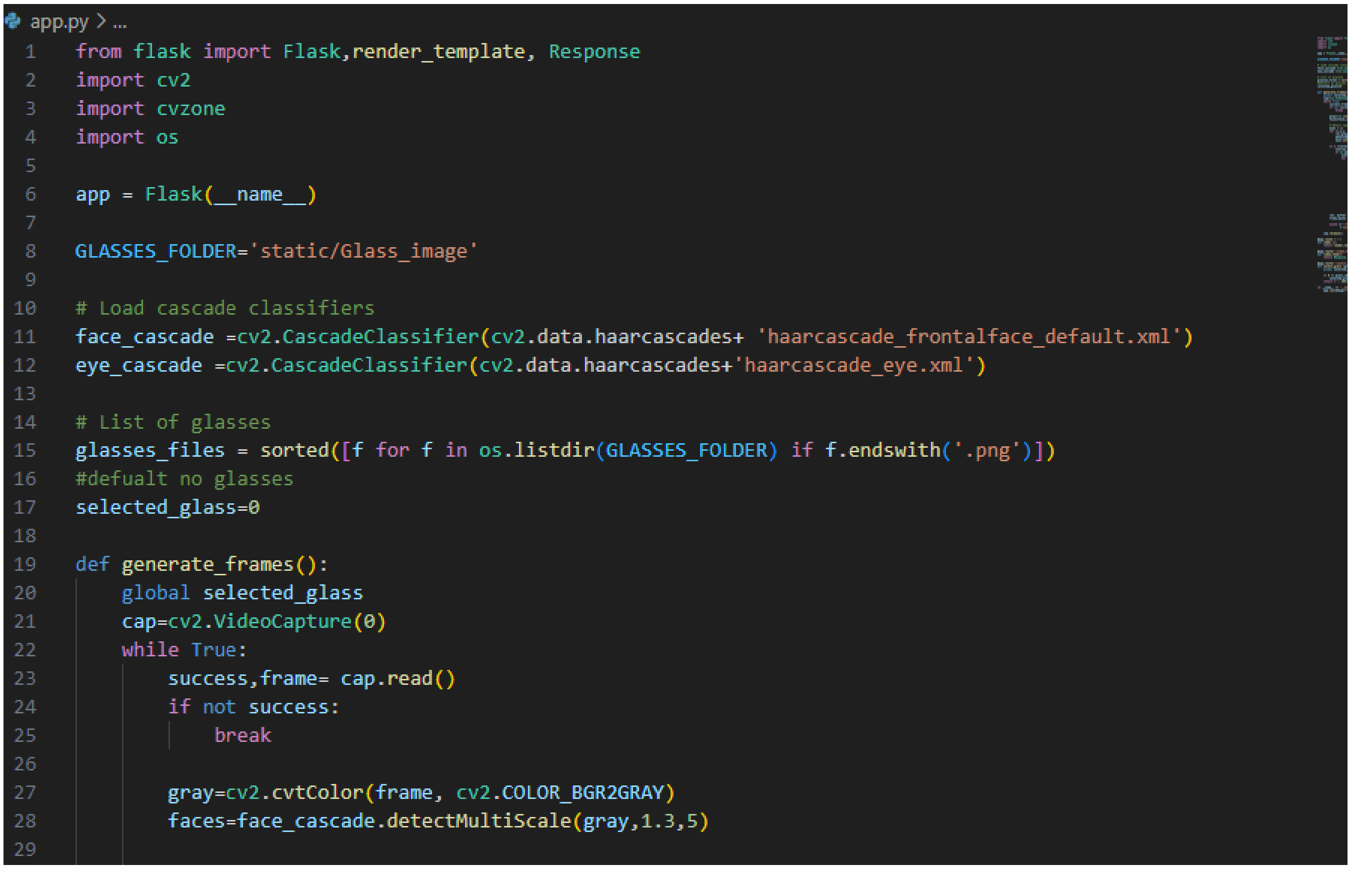
1.5 Key Functionalities:

- Accurate alignment of glasses using face and eye detection
- Smooth and immediate switching between different frames
- Option to view the video stream without any glasses overlay

1.6 Benefits:

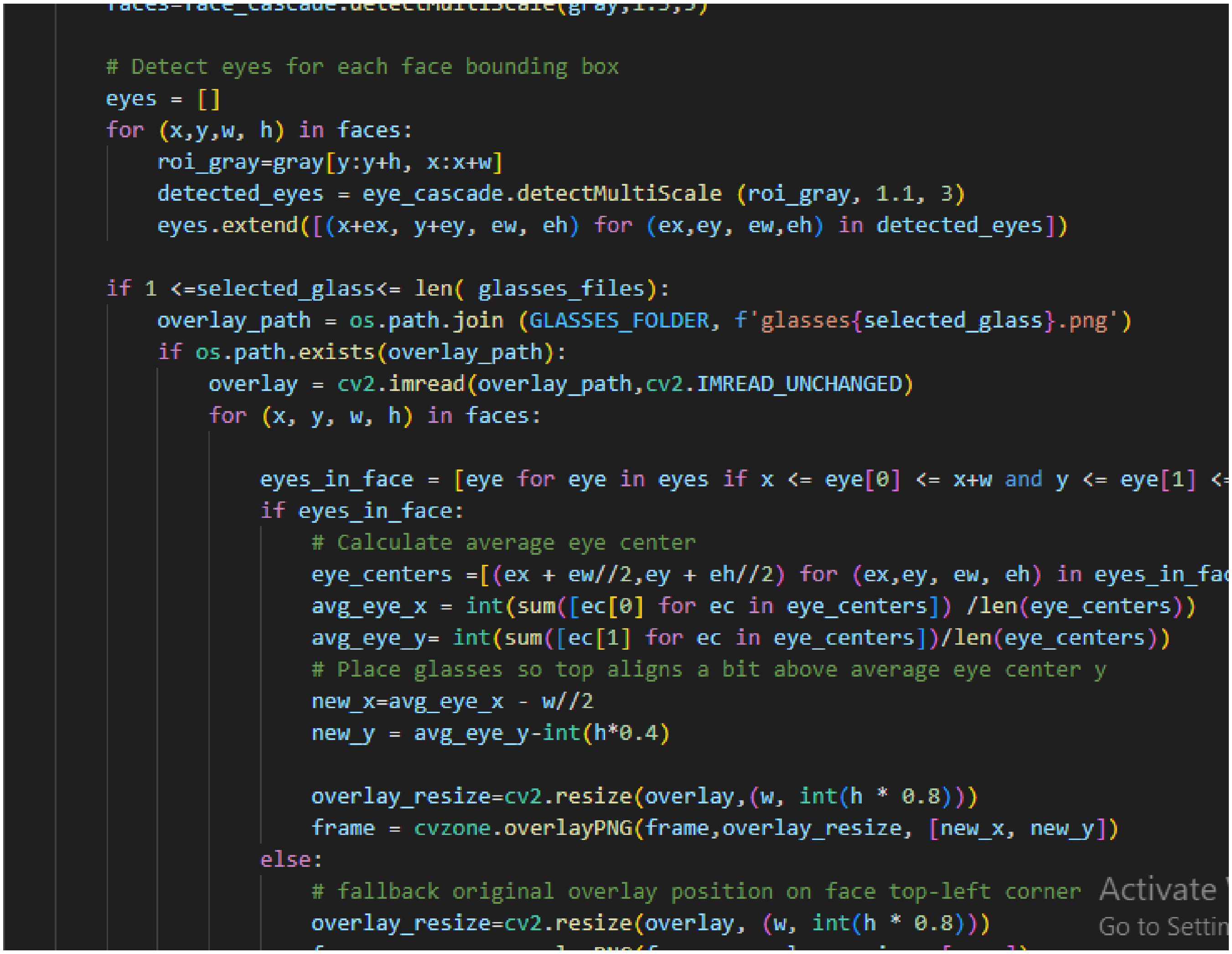
- Reduces uncertainty and increases confidence in frame selection
- Enhances the online eyewear shopping experience
- Allows users to experiment with different styles before making a purchase

2 CODE IMPLEMENTATION of APP.py:



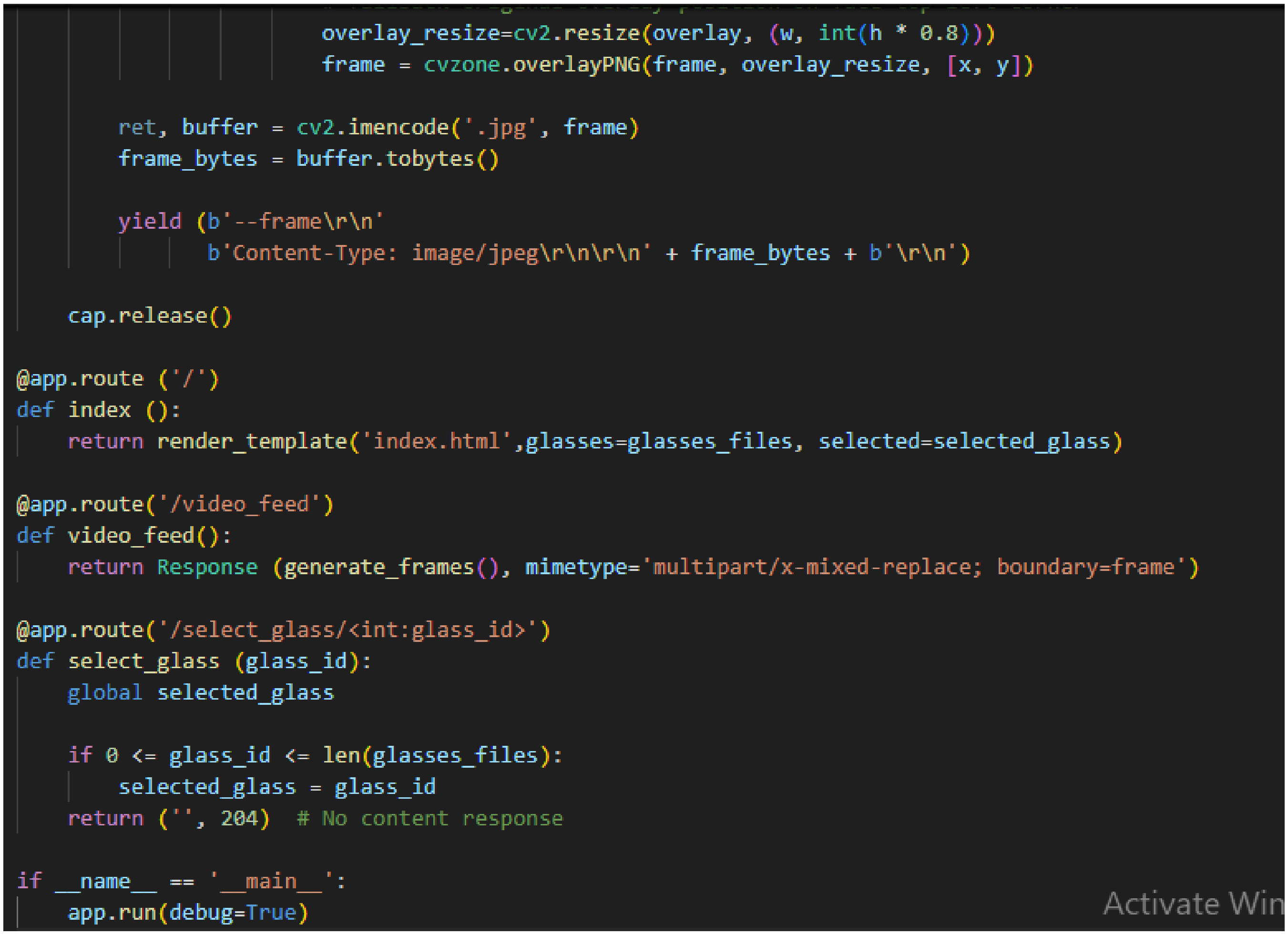
```python
from flask import Flask,render_template, Response
import cv2
import cvzone
import os

app = Flask(__name__)

GLASSES_FOLDER='static/Glass_image'

# Load cascade classifiers
face_cascade =cv2.CascadeClassifier(cv2.data.haarcascades+ 'haarcascade_frontalface_default.xml')
eye_cascade =cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_eye.xml')

# List of glasses
glasses_files = sorted([f for f in os.listdir(GLASSES_FOLDER) if f.endswith('.png')])
#defualt no glasses
selected_glass=0

def generate_frames():
    global selected_glass
    cap=cv2.VideoCapture(0)
    while True:
        success,frame= cap.read()
        if not success:
            break

        gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces=face_cascade.detectMultiScale(gray,1.3,5)

```

Figure a code-1



```
faces=face_cascade.detectMultiScale(gray,1.3,5)

# Detect eyes for each face bounding box
eyes = []
for (x,y,w, h) in faces:
    roi_gray=gray[y:y+h, x:x+w]
    detected_eyes = eye_cascade.detectMultiScale (roi_gray, 1.1, 3)
    eyes.extend([(x+ex, y+ey, ew, eh) for (ex,ey, ew,eh) in detected_eyes])

if 1 <=selected_glass<= len( glasses_files):
    overlay_path = os.path.join (GLASSES_FOLDER, f'glasses{selected_glass}.png')
    if os.path.exists(overlay_path):
        overlay = cv2.imread(overlay_path,cv2.IMREAD_UNCHANGED)
        for (x, y, w, h) in faces:

            eyes_in_face = [eye for eye in eyes if x <= eye[0] <= x+w and y <= eye[1] <=
            if eyes_in_face:
                # Calculate average eye center
                eye_centers =[(ex + ew//2,ey + eh//2) for (ex,ey, ew, eh) in eyes_in_fac
                avg_eye_x = int(sum([ec[0] for ec in eye_centers]) /len(eye_centers))
                avg_eye_y= int(sum([ec[1] for ec in eye_centers])/len(eye_centers))
                # Place glasses so top aligns a bit above average eye center y
                new_x=avg_eye_x - w//2
                new_y = avg_eye_y-int(h*0.4)

                overlay_resize=cv2.resize(overlay,(w, int(h * 0.8)))
                frame = cvzone.overlayPNG(frame,overlay_resize, [new_x, new_y])
            else:
                # fallback original overlay position on face top-left corner
                overlay_resize=cv2.resize(overlay, (w, int(h * 0.8)))
                frame = cvzone.overlayPNG(frame, overlay_resize, [x, y])
```

Figure b code-2



```
                        overlay_resize=cv2.resize(overlay, (w, int(h * 0.8)))
                        frame = cvzone.overlayPNG(frame, overlay_resize, [x, y])

        ret, buffer = cv2.imencode('.jpg', frame)
        frame_bytes = buffer.tobytes()

        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')

    cap.release()

@app.route ('/')
def index ():
    return render_template('index.html',glasses=glasses_files, selected=selected_glass)

@app.route('/video_feed')
def video_feed():
    return Response (generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/select_glass/<int:glass_id>')
def select_glass (glass_id):
    global selected_glass

    if 0 <= glass_id <= len(glasses_files):
        selected_glass = glass_id
    return ('', 204)  # No content response

if __name__ == '__main__':
    app.run(debug=True)
```

Figure c code-3

3 CONCLUSION:

- The Visual Try-On Glasses application successfully integrates computer vision and web technologies to provide a real-time, interactive eyewear try-on experience.
- By using Flask as the backend and OpenCV for face and eye detection, the system accurately overlays selected glasses frames on the user's face in a live video stream.
- The user interface is intuitive, allowing users to easily select from various glasses styles and instantly see the results.

---END---