Drowsiness Detection System

What is a Drowsiness Detection System ?

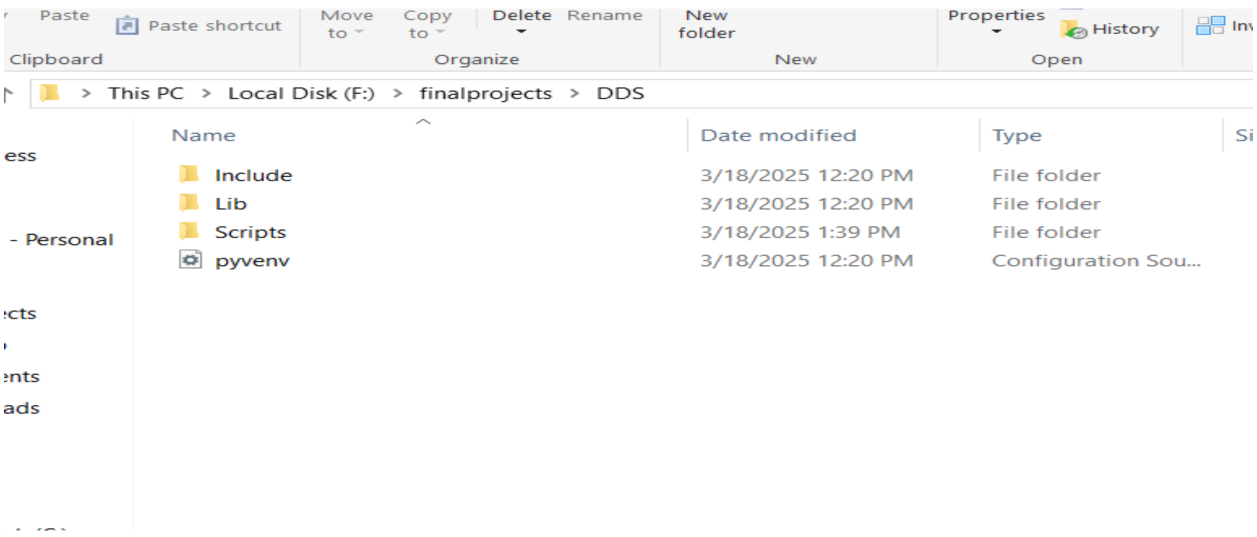
- It is used to monitor a person's alertness and provides alerts when signs of drowsiness are detected. It typically uses techniques from computer vision, machine learning, and sensor-based detection.
- This app can get data from various sources such as IP camera,webcam,online video
- This is a web app that can be accessed over lan
- The ip camera is a wireless surveillance camera which can send footage to the wifi network to which it is connected then if the computer which contains the app is also connected with the same wifi network then we will be able to work on footage data into our project to detect drowsiness.

Why?

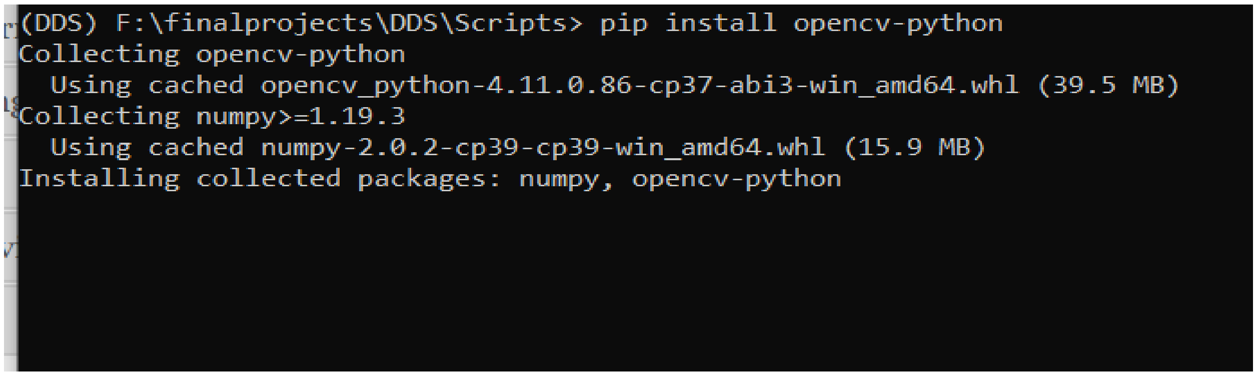
- This app can be used in enhancing Workplace Safety. In industries like construction or transportation, fatigue-related errors can be dangerous.Improves Productivity for monitoring alertness in long working hours (e.g., pilots, truck drivers) helps maintain efficiency.
- To enhance the ML knowledge,programming skill and web development skill and also for creating resume
- . It hows that you can solve real world problem with your knowledge

How?

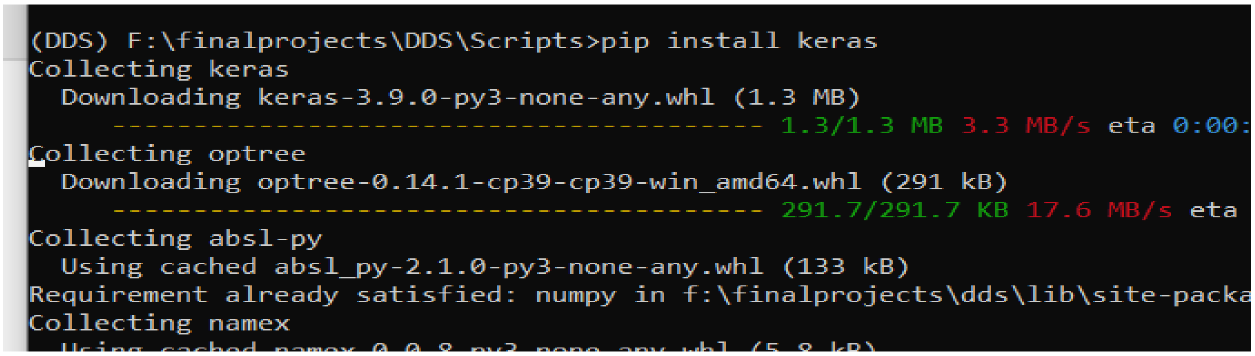
- OpenCV – pip install opencv-python (face and eye detection).(ip camera)
- TensorFlow – pip install tensorflow (keras uses TensorFlow as backend).
- Keras – pip install keras (to build our classification model).
- Frontend-Streamlit
- Pygame - alert sound



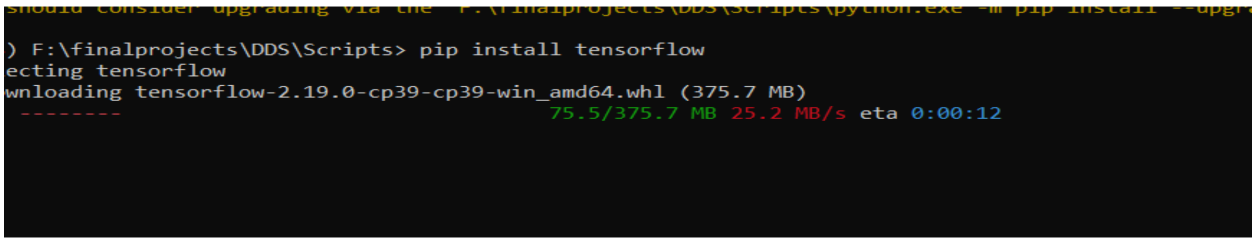
```
This PC > Local Disk (F:) > finalprojects > DDS

Name        Date modified        Type                   Si
Include     3/18/2025 12:20 PM   File folder
Lib         3/18/2025 12:20 PM   File folder
Scripts     3/18/2025 1:39 PM    File folder
pyvenv      3/18/2025 12:20 PM   Configuration Sou...
```



```
(DDS) F:\finalprojects\DDS\Scripts> pip install opencv-python
Collecting opencv-python
  Using cached opencv_python-4.11.0.86-cp37-abi3-win_amd64.whl (39.5 MB)
Collecting numpy>=1.19.3
  Using cached numpy-2.0.2-cp39-cp39-win_amd64.whl (15.9 MB)
Installing collected packages: numpy, opencv-python
```

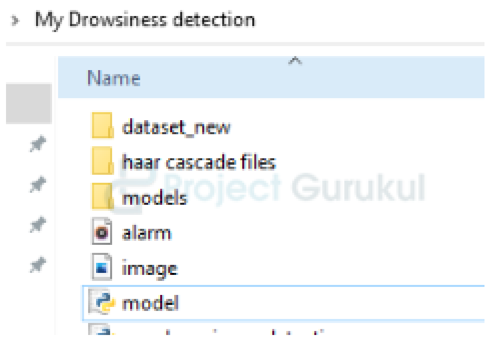


```
(DDS) F:\finalprojects\DDS\Scripts>pip install keras
Collecting keras
  Downloading keras-3.9.0-py3-none-any.whl (1.3 MB)
     ---------------------------------------- 1.3/1.3 MB 3.3 MB/s eta 0:00:
Collecting optree
  Downloading optree-0.14.1-cp39-cp39-win_amd64.whl (291 kB)
     ---------------------------------------- 291.7/291.7 KB 17.6 MB/s eta
Collecting absl-py
  Using cached absl_py-2.1.0-py3-none-any.whl (133 kB)
Requirement already satisfied: numpy in f:\finalprojects\dds\lib\site-packa
Collecting namex
```



```
) F:\finalprojects\DDS\Scripts> pip install tensorflow
ecting tensorflow
wnloading tensorflow-2.19.0-cp39-cp39-win_amd64.whl (375.7 MB)
 --------                                  75.5/375.7 MB 25.2 MB/s eta 0:00:12
```

Main file



dataset_new: In this folder, we have ur dataset

haar cascade files: This folder has files that are used to detect the face and eyes of a person, these files are xml files. The haar cascade files have many xml files that are required to detect objects in an image.

models: In this, we have our model ‘custmodel.h5’ file that we have created above.

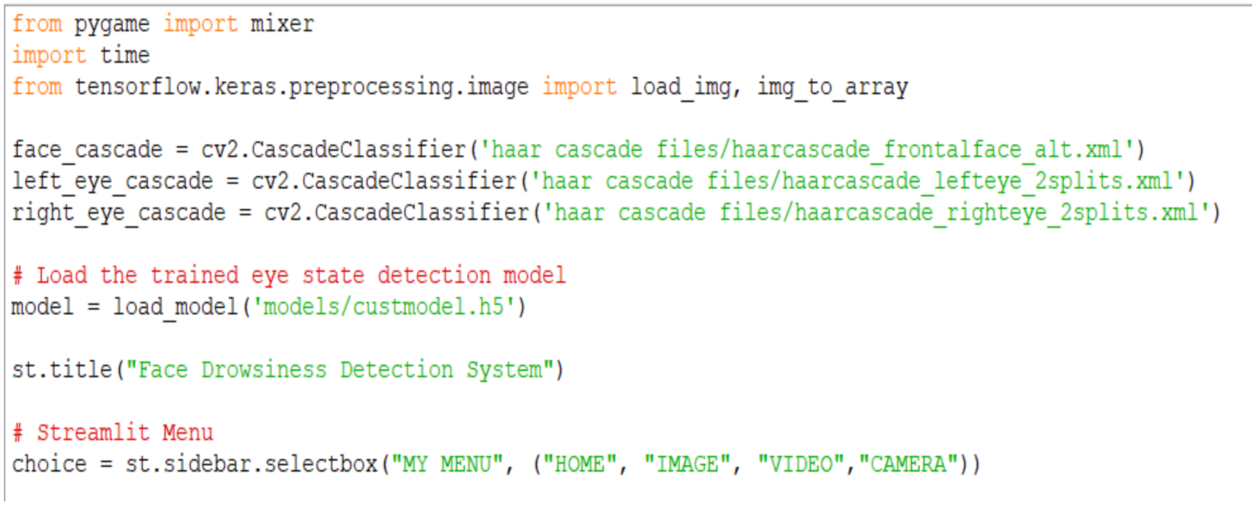
alarm: This file is used to play the alert sound when a person closes its eyes for a few seconds.

model.py: It is the python file in which we have created our classification model which is trained on our dataset. You can see the implementation of how we have created the model and how we trained it according to our dataset.

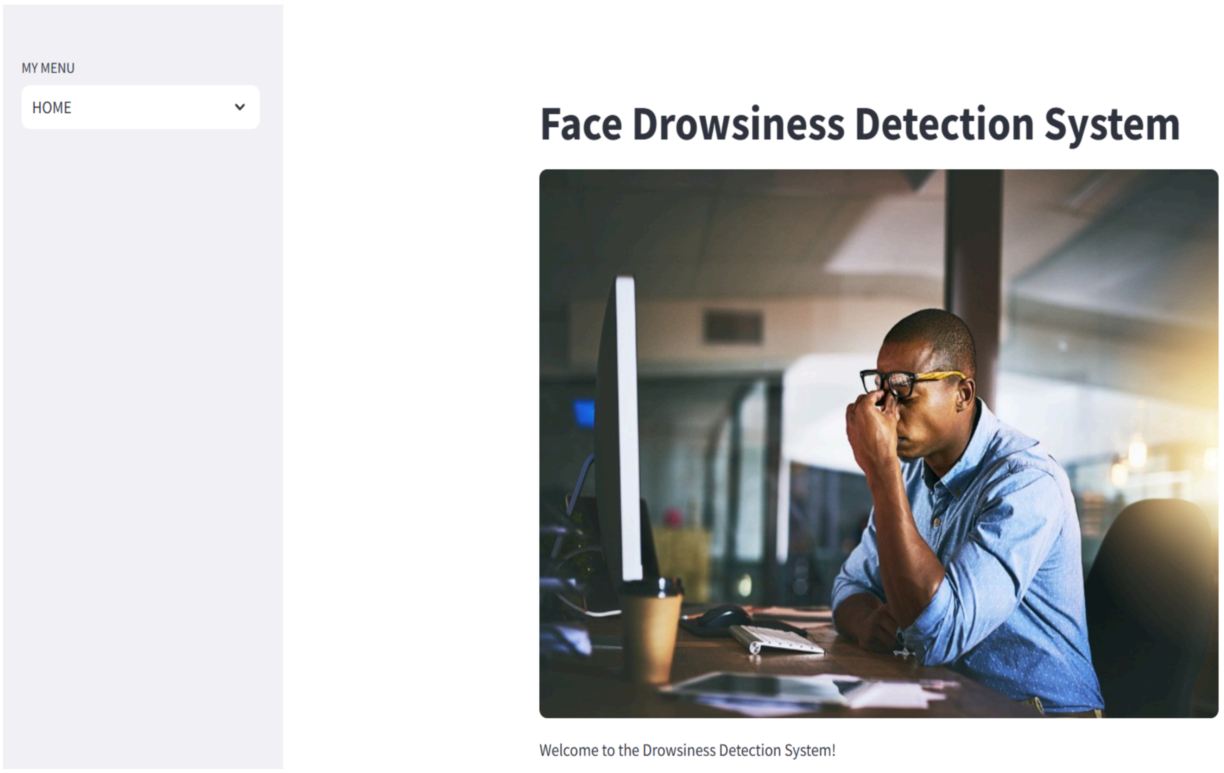
main.py: This file consists of full implementation of our project in which we have loaded the model(custmodel), and used it to alert the person whenever he/she will feel drowsy

Screenshot

Home section



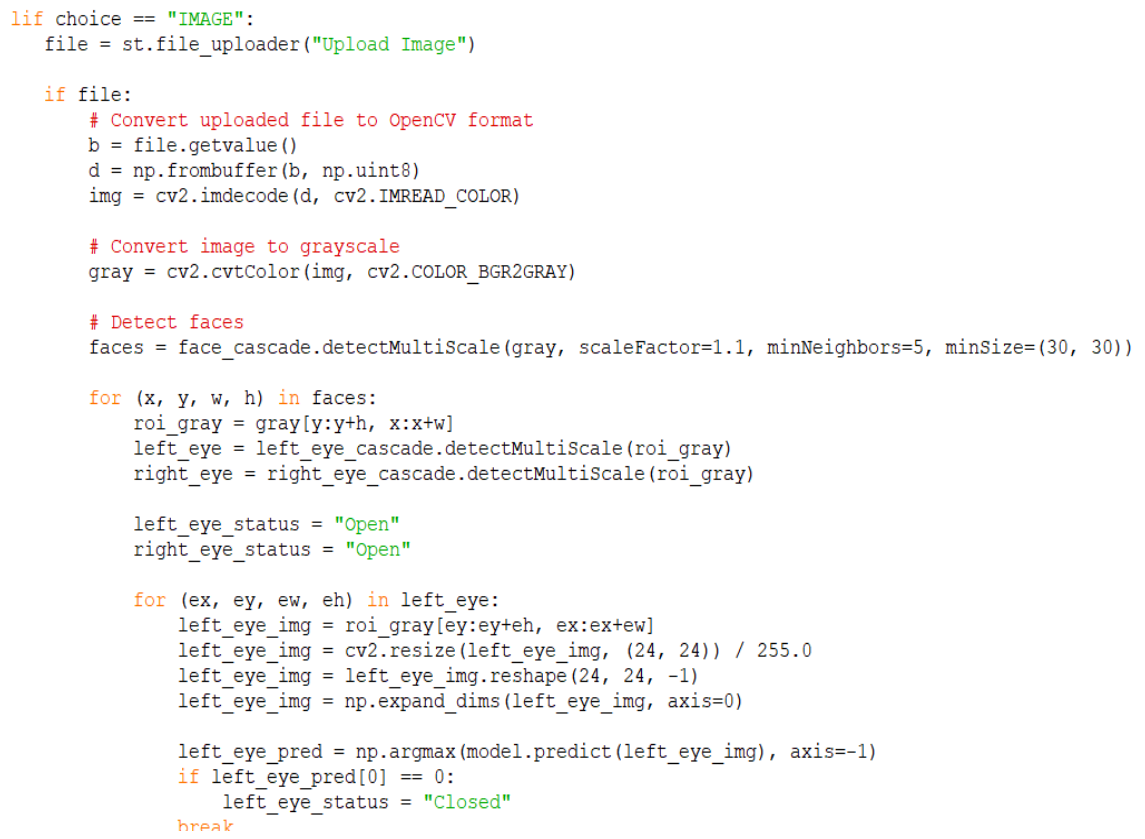
```python
from pygame import mixer
import time
from tensorflow.keras.preprocessing.image import load_img, img_to_array

face_cascade = cv2.CascadeClassifier('haar cascade files/haarcascade_frontalface_alt.xml')
left_eye_cascade = cv2.CascadeClassifier('haar cascade files/haarcascade_lefteye_2splits.xml')
right_eye_cascade = cv2.CascadeClassifier('haar cascade files/haarcascade_righteye_2splits.xml')

# Load the trained eye state detection model
model = load_model('models/custmodel.h5')

st.title("Face Drowsiness Detection System")

# Streamlit Menu
choice = st.sidebar.selectbox("MY MENU", ("HOME", "IMAGE", "VIDEO","CAMERA"))
```



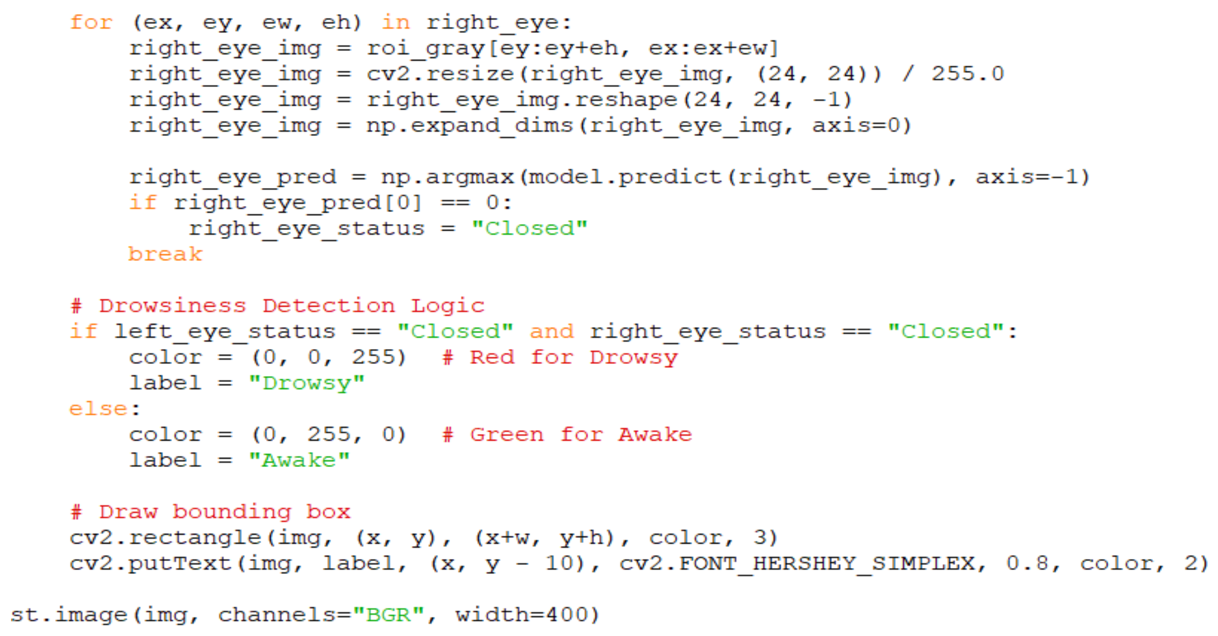
Face Drowsiness Detection System

Welcome to the Drowsiness Detection System!

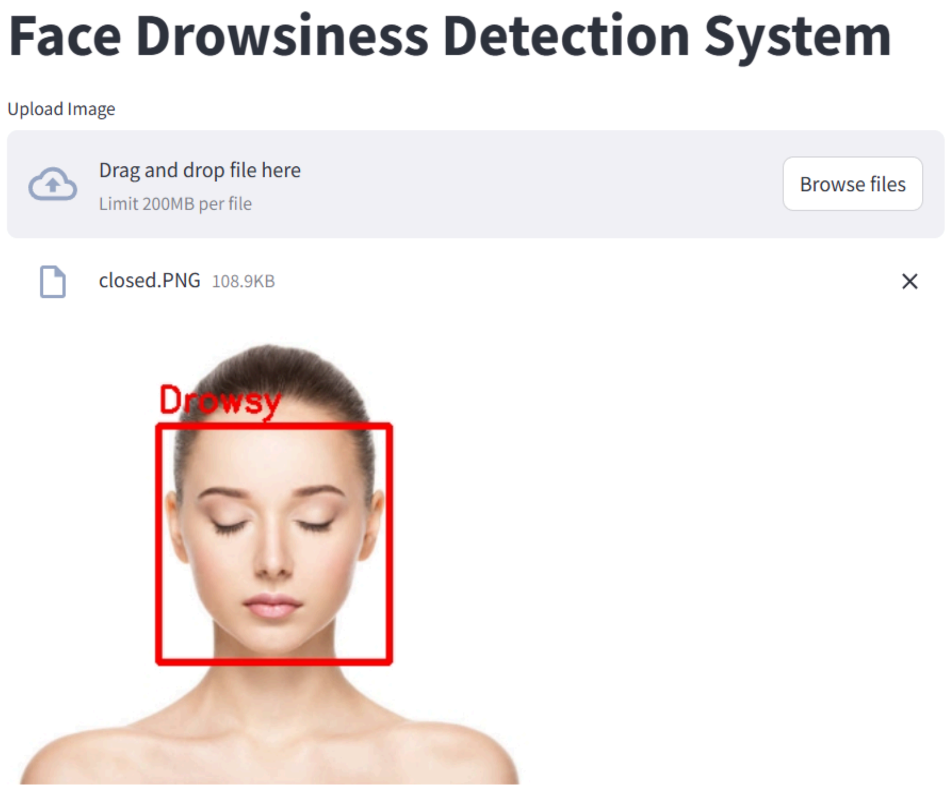
Image section



```
lif choice == "IMAGE":
    file = st.file_uploader("Upload Image")

    if file:
        # Convert uploaded file to OpenCV format
        b = file.getvalue()
        d = np.frombuffer(b, np.uint8)
        img = cv2.imdecode(d, cv2.IMREAD_COLOR)

        # Convert image to grayscale
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        # Detect faces
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

        for (x, y, w, h) in faces:
            roi_gray = gray[y:y+h, x:x+w]
            left_eye = left_eye_cascade.detectMultiScale(roi_gray)
            right_eye = right_eye_cascade.detectMultiScale(roi_gray)

            left_eye_status = "Open"
            right_eye_status = "Open"

            for (ex, ey, ew, eh) in left_eye:
                left_eye_img = roi_gray[ey:ey+eh, ex:ex+ew]
                left_eye_img = cv2.resize(left_eye_img, (24, 24)) / 255.0
                left_eye_img = left_eye_img.reshape(24, 24, -1)
                left_eye_img = np.expand_dims(left_eye_img, axis=0)

                left_eye_pred = np.argmax(model.predict(left_eye_img), axis=-1)
                if left_eye_pred[0] == 0:
                    left_eye_status = "Closed"
                break
```

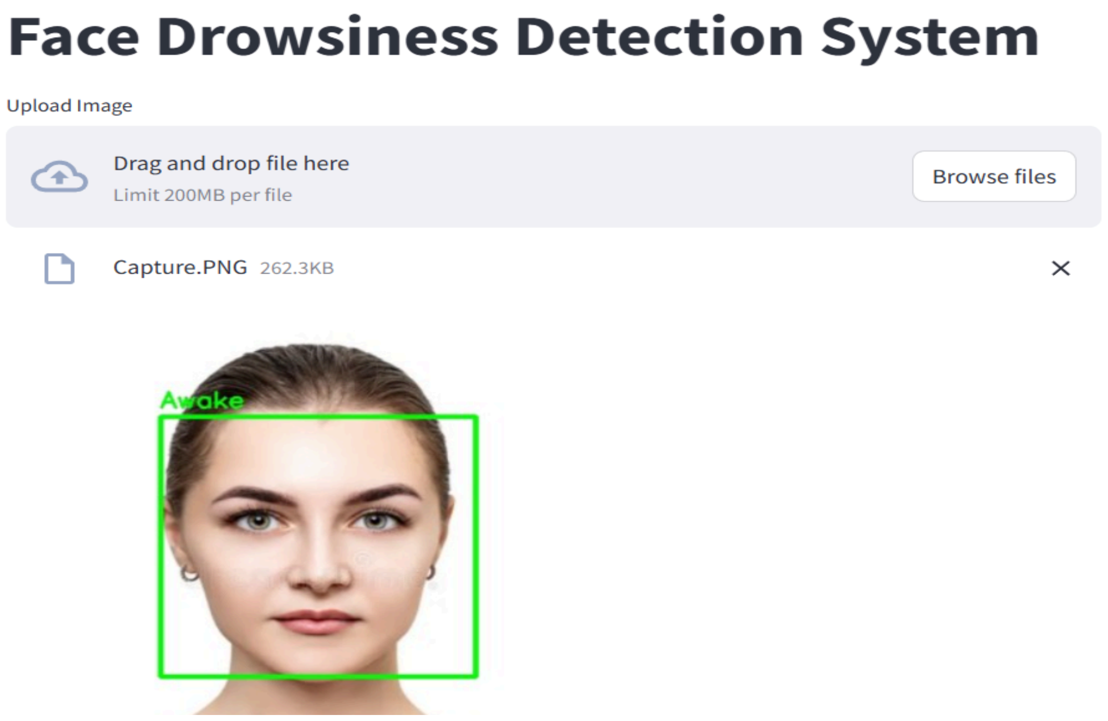


```
    for (ex, ey, ew, eh) in right_eye:
        right_eye_img = roi_gray[ey:ey+eh, ex:ex+ew]
        right_eye_img = cv2.resize(right_eye_img, (24, 24)) / 255.0
        right_eye_img = right_eye_img.reshape(24, 24, -1)
        right_eye_img = np.expand_dims(right_eye_img, axis=0)

        right_eye_pred = np.argmax(model.predict(right_eye_img), axis=-1)
        if right_eye_pred[0] == 0:
            right_eye_status = "Closed"
        break

    # Drowsiness Detection Logic
    if left_eye_status == "Closed" and right_eye_status == "Closed":
        color = (0, 0, 255)  # Red for Drowsy
        label = "Drowsy"
    else:
        color = (0, 255, 0)  # Green for Awake
        label = "Awake"

    # Draw bounding box
    cv2.rectangle(img, (x, y), (x+w, y+h), color, 3)
    cv2.putText(img, label, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, color, 2)

st.image(img, channels="BGR", width=400)
```



Face Drowsiness Detection System

Upload Image



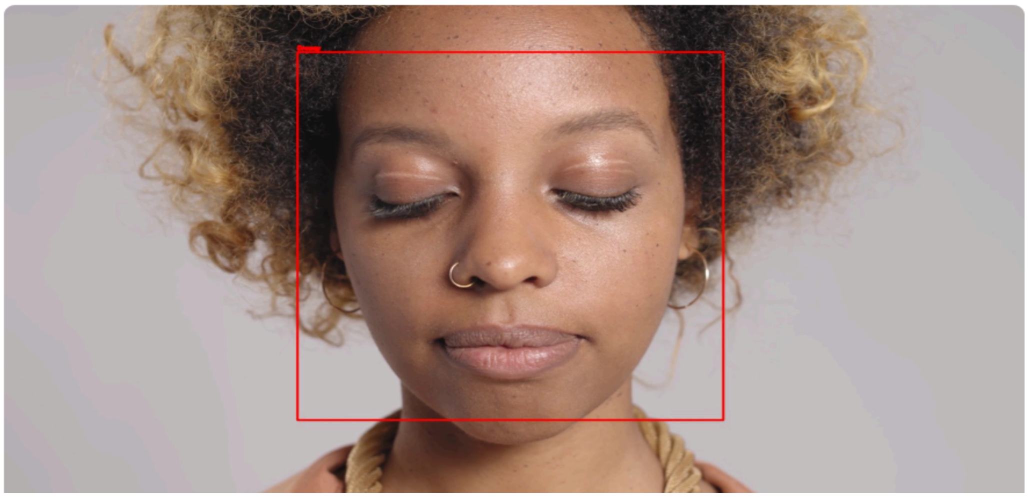
Face Drowsiness Detection System

Upload Image

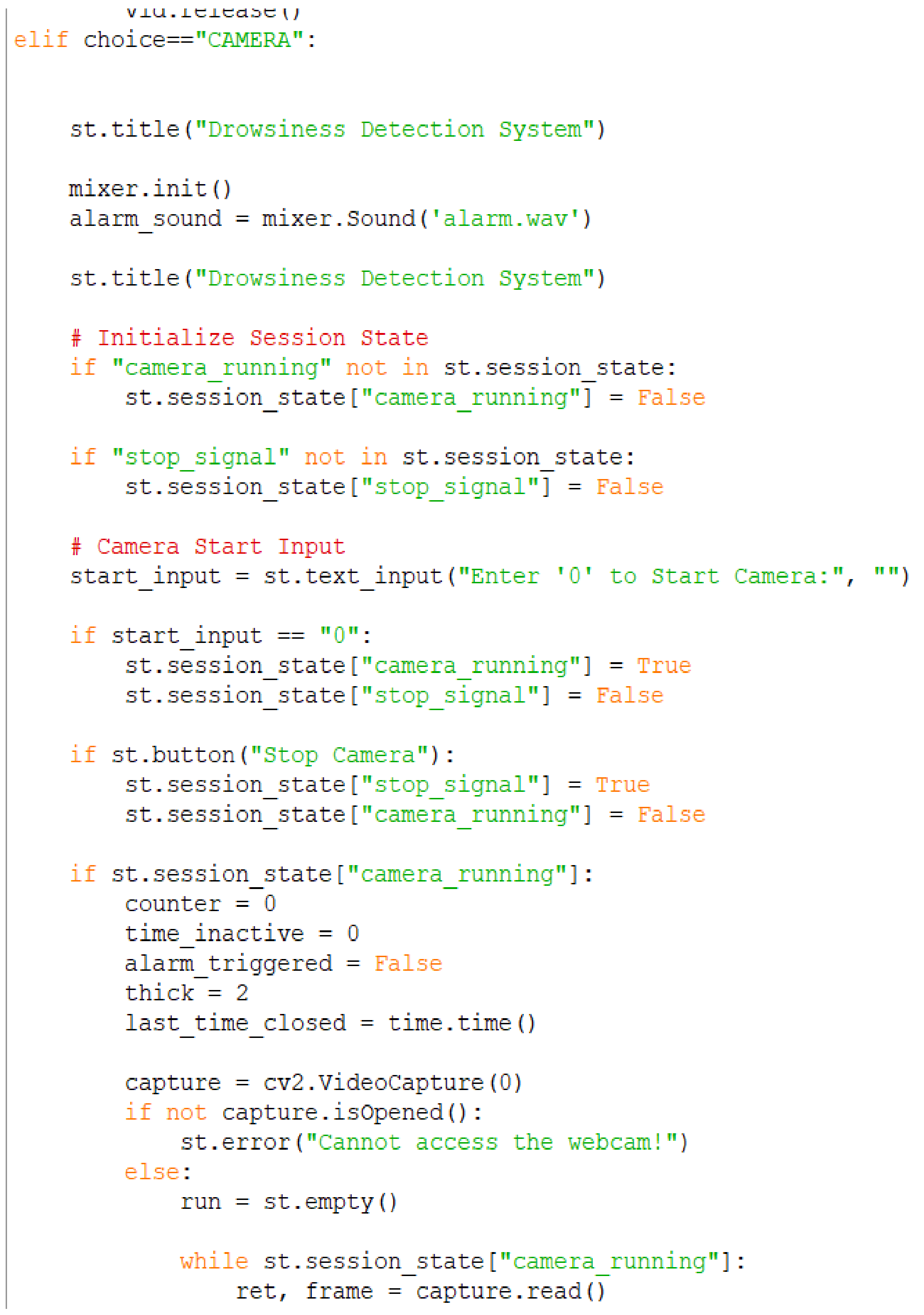
Video section

```
elif choice == "VIDEO":
    st.subheader("Upload a Video for Drowsiness Detection")
    file = st.file_uploader("Upload Video")
    window = st.empty()

    if file:
        tfile = tempfile.NamedTemporaryFile(delete=False)
        tfile.write(file.read())
        vid = cv2.VideoCapture(tfile.name)

        while vid.isOpened():
            flag, frame = vid.read()
            if not flag:
                break

            # Convert frame to grayscale
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,

            for (x, y, w, h) in faces:
                roi_gray = gray[y:y + h, x:x + w]
                left_eye = left_eye_cascade.detectMultiScale(roi_gray)
                right_eye = right_eye_cascade.detectMultiScale(roi_gray)

                left_eye_status, right_eye_status = "Open", "Open"

                # Check left eye status if eyes are detected
                if len(left_eye) > 0:
                    for (ex, ey, ew, eh) in left_eye:
                        left_eye_img = roi_gray[ey:ey + eh, ex:ex + ew]
                        left_eye_img = cv2.resize(left_eye_img, (24, 24)) / 255.0
                        left_eye_img = left_eye_img.reshape(24, 24, -1)
                        left_eye_img = np.expand_dims(left_eye_img, axis=0)

                        left_eye_pred = np.argmax(model.predict(left_eye_img), axis=-1)
                        if left_eye_pred[0] == 0:  # Closed
                            left_eye_status = "Closed"
                        break  # Exit loop once the first eye is processed
```

```
           break  # Exit loop once the first eye is processed

   # Check right eye status if eyes are detected
   if len(right_eye) > 0:
       for (ex, ey, ew, eh) in right_eye:
           right_eye_img = roi_gray[ey:ey + eh, ex:ex + ew]
           right_eye_img = cv2.resize(right_eye_img, (24, 24)) /
           right_eye_img = right_eye_img.reshape(24, 24, -1)
           right_eye_img = np.expand_dims(right_eye_img, axis=0)

           right_eye_pred = np.argmax(model.predict(right_eye_img
           if right_eye_pred[0] == 0:  # Closed
               right_eye_status = "Closed"
           break  # Exit loop once the first eye is processed

   # Determine the color based on eye status
   if left_eye_status == "Closed" or right_eye_status == "Closed"
       color = (0, 0, 255)  # Red for closed eyes
       label = "Drowsy"
   else:
       color = (0, 255, 0)  # Green for awake (eyes open)
       label = "Awake"

   # Draw bounding box and label on the frame
   cv2.rectangle(frame, (x, y), (x + w, y + h), color, 7)
   cv2.putText(frame, label, (x, y - 10), cv2.FONT_HERSHEY_SIMPLE

 Show frame in Streamlit
indow.image(frame, channels='BGR')
```

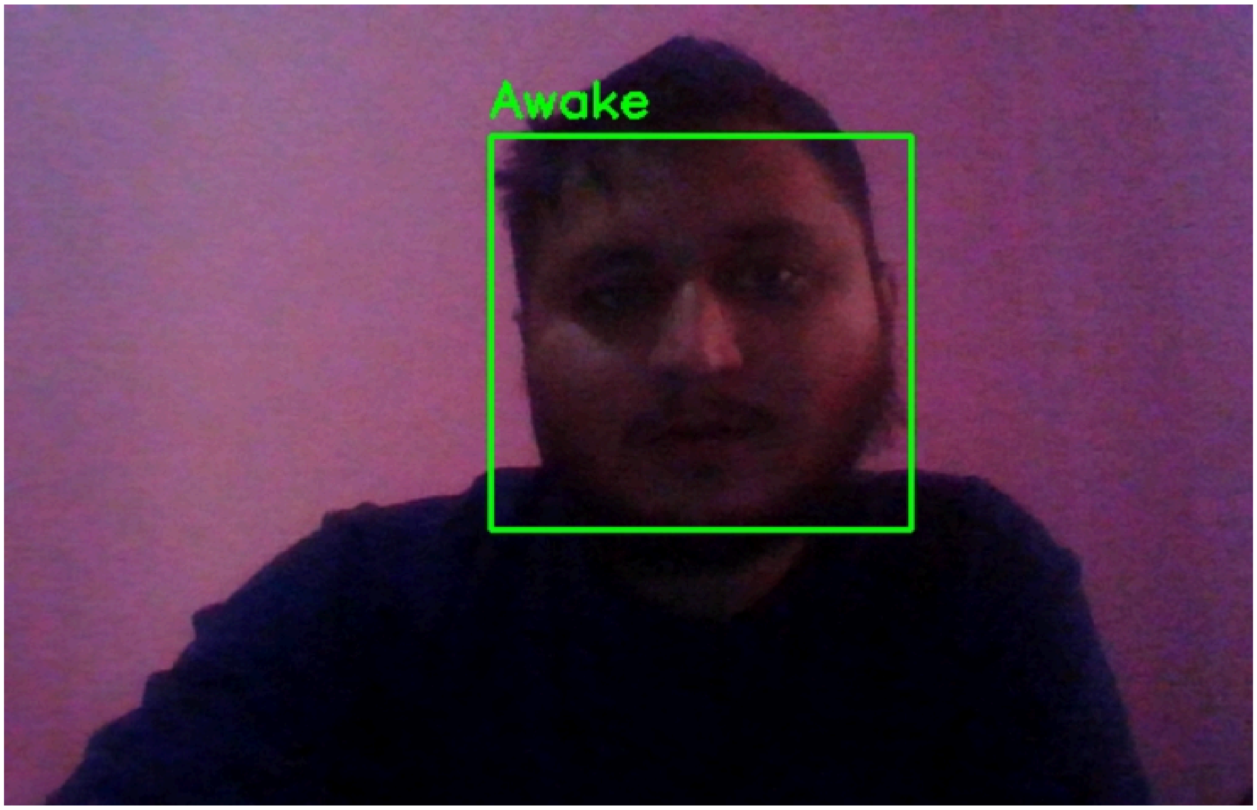


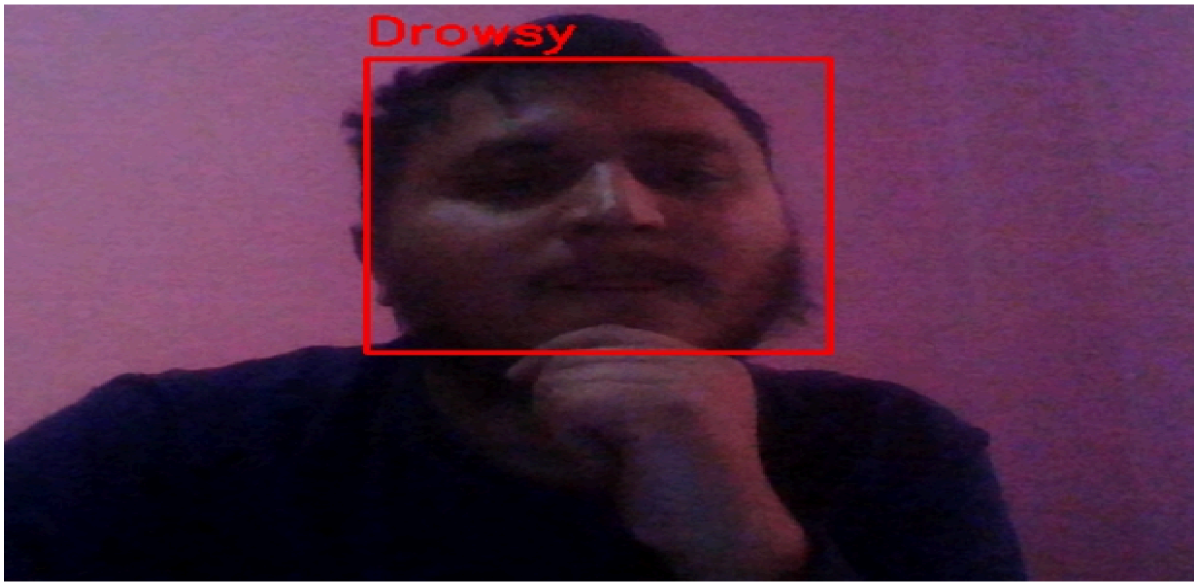
Camera section



```python
        vid.release()
elif choice=="CAMERA":


    st.title("Drowsiness Detection System")

    mixer.init()
    alarm_sound = mixer.Sound('alarm.wav')

    st.title("Drowsiness Detection System")

    # Initialize Session State
    if "camera_running" not in st.session_state:
        st.session_state["camera_running"] = False

    if "stop_signal" not in st.session_state:
        st.session_state["stop_signal"] = False

    # Camera Start Input
    start_input = st.text_input("Enter '0' to Start Camera:", "")

    if start_input == "0":
        st.session_state["camera_running"] = True
        st.session_state["stop_signal"] = False

    if st.button("Stop Camera"):
        st.session_state["stop_signal"] = True
        st.session_state["camera_running"] = False

    if st.session_state["camera_running"]:
        counter = 0
        time_inactive = 0
        alarm_triggered = False
        thick = 2
        last_time_closed = time.time()

        capture = cv2.VideoCapture(0)
        if not capture.isOpened():
            st.error("Cannot access the webcam!")
        else:
            run = st.empty()

            while st.session_state["camera_running"]:
                ret, frame = capture.read()
```

```
while st.session_state["camera_running"]:
    ret, frame = capture.read()
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30,

    left_eye_status, right_eye_status = "Open", "Open"
    drowsy_detected = False

    for (x, y, w, h) in faces:
        roi_gray = gray[y:y + h, x:x + w]
        left_eye = left_eye_cascade.detectMultiScale(roi_gray)
        right_eye = right_eye_cascade.detectMultiScale(roi_gray)

        # **Detect Eye Closure**
        if len(left_eye) == 0:
            left_eye_status = "Closed"
        if len(right_eye) == 0:
            right_eye_status = "Closed"

        # **Drowsiness Detection Logic**
        if left_eye_status == "Closed" and right_eye_status == "Closed":
            time_inactive += 1  # Increase counter
            drowsy_detected = True
            last_time_closed = time.time()
        else:
            time_inactive = max(0, time_inactive - 1)  # Decrease counter safely

        # **Alarm Logic**
        if time_inactive > 10:
            if not alarm_triggered:
                alarm_sound.play(-1)  # Play Alarm Continuously
                alarm_triggered = True
        else:
            if alarm_triggered:
                alarm_sound.stop()  # Stop Alarm
                alarm_triggered = False

        # **Draw Bounding Box**
        color = (0, 0, 255) if drowsy_detected else (0, 255, 0)
        label = "Drowsy" if drowsy_detected else "Awake"
        cv2.rectangle(frame, (x, y), (x + w, y + h), color, thick)
        cv2.putText(frame, label, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, color, 2)
```


Awake


Drowsy