```
!pip install opencv-python-headless dlib pillow
```

```
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.11/dist-packages (4.11.0.86)
Requirement already satisfied: dlib in /usr/local/lib/python3.11/dist-packages (19.24.6)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (11.1.0)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.11/dist-packages (from opencv-python-headless) (2.0.2)
```

```
import cv2
import numpy as np
import io
from PIL import Image
import matplotlib.pyplot as plt
from IPython.display import display, HTML
from google.colab import files
from ipywidgets import FileUpload, Button, VBox, HBox, Output, Label

# Load Haar Cascade
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye_tree_eyeglasses.xml')

# Create widgets
upload_btn = FileUpload(accept='image/*', multiple=False)
detect_btn = Button(description='✨ Run Face & Eye Detection ✨', button_style='success')
preview_out = Output()
result_out = Output()
status = Label(value="⬆️ Upload an image, then click 'Run Detection'")

# Theme Styling: Inject CSS for background
custom_css = """
<style>
  body {
    background-color: #1e1e2f;
    color: white;
  }
  .output_wrapper, .output {
    background: #29293d !important;
    border-radius: 10px;
    padding: 10px;
```

```
    border: 1px solid #444;
  }
  button {
    font-weight: bold !important;
  }
</style>
"""
display(HTML(custom_css))

# Detection Logic
def detect_and_display(change=None):
    if not upload_btn.value:
        status.value = "⚠️ Please upload an image first."
        return

    uploaded_file = next(iter(upload_btn.value.values()))
    img = Image.open(io.BytesIO(uploaded_file['content'])).convert('RGB')
    img_np = np.array(img)
    img_bgr = cv2.cvtColor(img_np, cv2.COLOR_RGB2BGR)
    gray = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)

    # Face and eye detection
    faces = face_cascade.detectMultiScale(gray, 1.1, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(img_bgr, (x, y), (x + w, y + h), (0, 255, 0), 2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = img_bgr[y:y+h, x:x+w]

        eyes = eye_cascade.detectMultiScale(roi_gray, 1.05, 7, minSize=(20, 20))
        for (ex, ey, ew, eh) in eyes[:2]:
            cv2.rectangle(roi_color, (ex, ey), (ex + ew, ey + eh), (255, 0, 0), 2)

    result_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)

    preview_out.clear_output()
    result_out.clear_output()
    with preview_out:
        plt.figure(figsize=(4, 4))
        plt.imshow(img)
        plt.title("🎯 Uploaded Image")
        plt.axis('off')
```

```
        plt.show()

    with result_out:
        plt.figure(figsize=(6, 6))
        plt.imshow(result_rgb)
        plt.title("✅ Face & Eyes Detected")
        plt.axis('off')
        plt.show()

    status.value = "✅ Detection complete with cool theme!"

# Bind button click
detect_btn.on_click(detect_and_display)

# Layout
gui = VBox([
    status,
    upload_btn,
    detect_btn,
    HBox([preview_out, result_out])
])
display(gui)
```

Detection complete with cool theme!

**Upload (2)**

✨ Run Face & Ey...

## Uploaded Image

## Face & Eyes Detected

Start coding or generate with AI.