Spotting Spectacles: A YOLOv10 Approach

Introduction:

In the age of artificial intelligence and computer vision, object detection has become a pivotal technology across various industries. Our project, "Visionary Insight: Spectacle Detection Using YOLOv10," aims to harness the power of YOLOv10 (You Only Look Once, version 10) to accurately identify and classify spectacles in images and videos. This advanced deep learning model provides real-time object detection with high precision and speed, making it an ideal solution for applications where quick and accurate identification is crucial. By focusing on spectacle detection, this project has the potential to enhance various fields such as healthcare, security, retail, and augmented reality.

Scenario 1:

Assistance for Visually Impaired Individuals:

Spectacle detection technology can be integrated into assistive devices for visually impaired individuals. These devices, equipped with cameras and YOLOv10, can detect spectacles and provide audio feedback to the user. For example, if the device detects that the user is not wearing their spectacles, it can remind them to put them on. Additionally, it can help users locate their spectacles if they are misplaced. The ability of the model to detect spectacles in various scenarios, as demonstrated in the sample outputs, is vital for developing reliable assistive technologies.

Scenario 2:

Social Media and Photo Tagging:

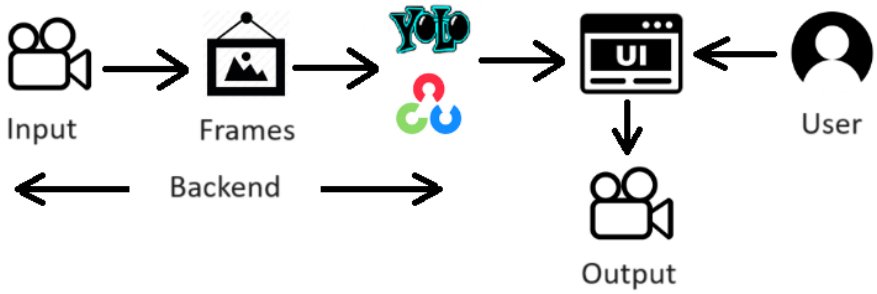
On social media platforms, automated spectacle detection can be used for photo tagging and content moderation. When users upload photos, the system can identify individuals wearing spectacles and tag them accordingly. This feature enhances the user experience by making it easier to organize and search for photos. Additionally, it can be used in content moderation to detect and flag images that meet certain criteria involving spectacles. The high-confidence detection of spectacles on a person in the sample output showcases the potential for accurate and efficient photo tagging.

Scenario 3 :

Security and Surveillance:

In security and surveillance systems, identifying individuals who are wearing spectacles can be crucial. For example, in high-security areas such as airports or government buildings, automated detection of spectacles can be integrated into facial recognition systems to improve accuracy and reliability. By distinguishing between individuals with and without spectacles, security personnel can maintain better records and enhance the overall effectiveness of surveillance operations. This capability can also be useful in tracking persons of interest in public spaces, aiding law enforcement in their efforts to ensure public safety. The sample output with the person wearing spectacles demonstrates the potential for real-time identification in surveillance footage, which is critical for maintaining security in sensitive areas.

Technical Architecture:



Pre-requisites:
To complete this project, you must require the following software, concepts, and packages.

1. **IDE Installation**:

   Spyder/ PyCharm IDE is Ideal to complete this project

   To install **Spyder IDE**, please refer to Spyder IDE Installation Steps

   To install **PyCharm IDE**, please refer to the PyCharm IDE Installation Steps

2. **Python Packages**

   If you are using the **Anaconda navigator**, follow the below steps to download the required packages:

   Open the Anaconda prompt.

   - Type “pip install opencv-python==4.9.0.80” and click enter.
   - Type “pip install numpy ==1.26.3” and click enter
   - Type “pip install flask” and click enter.

Prior Knowledge:
You must have prior knowledge of the following topics to complete this project.

- OpenCV - https://www.youtube.com/watch?v=WQeoO7MI0Bs
- Flask - https://www.youtube.com/watch?v=lj4I_CvBnt0
- Yolov10 - https://youtu.be/Dmv4EVBuCTQ?si=sc5mbfLky08_Ru96

Project Objectives:
By the end of this project you will:

- Know fundamental concepts and techniques used for computer vision.
- Gain knowledge in the pre-trained model yolov8.
- Gain knowledge of OpenCV.
- Gain knowledge of Web Integration.

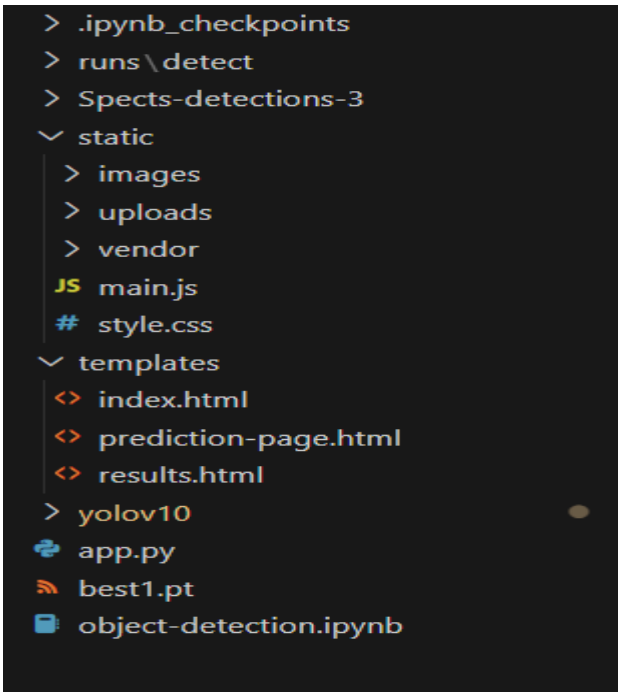
Project Flow:
The user interacts with the UI (User Interface) to choose the image.
The chosen image is analyzed by the model which is integrated with the flask application.

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
- Create Train and Test Folders.
- Create data.yaml file
- Training and testing the model
- Save the Model
- Application Building
- Create an HTML file
- Build Python Code

Project Structure:

Create a Project folder which contains files as shown below

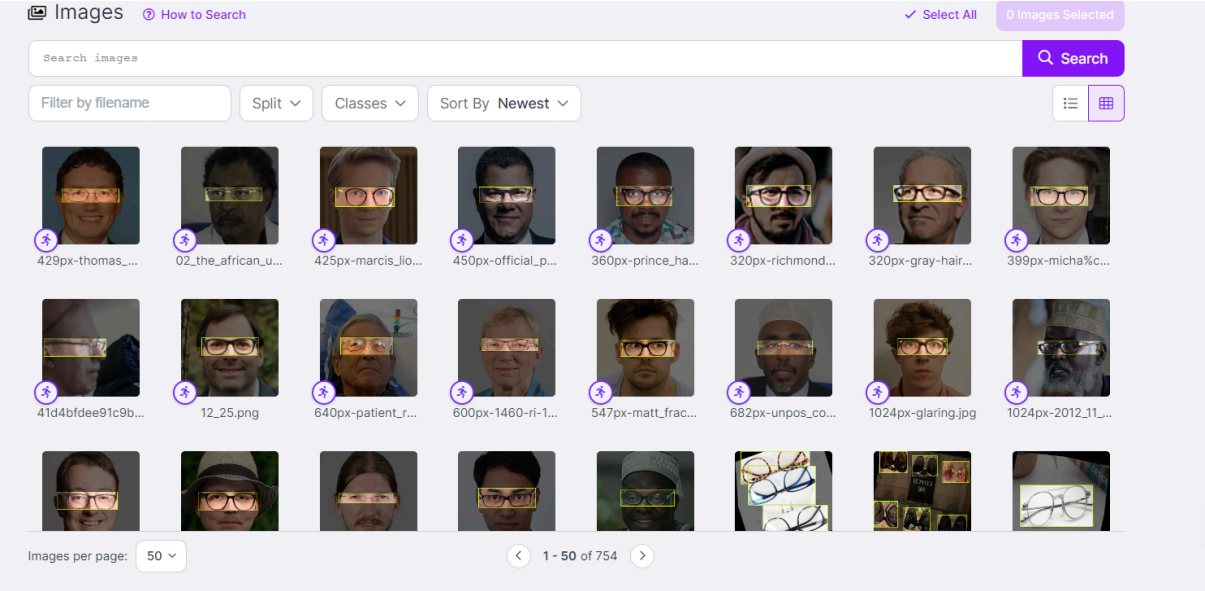


- The Dataset folder contains the training, testing, and val images for training our model.
- We are building a Flask Application that needs HTML pages stored in the templates folder and a python script app.py for server-side scripting
- We need the model which is saved and the saved model in this content there is a templates folder containing index.html and inner-page.html pages.

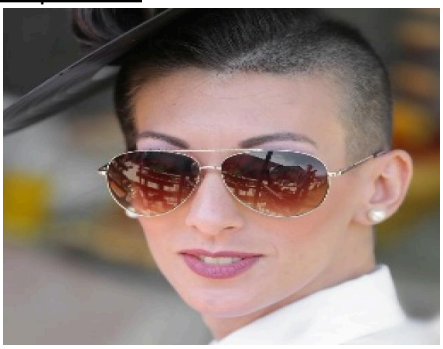
Milestone 1: Collection of Data

Dataset has 3 classes Which are crab, Lobster and shrimp

Download the Dataset-
https://app.roboflow.com/ds/m9krH7E3ql?key=8K9cw6gmig



Sample data:

:  

Milestone 2: Image Pre-processing

In this milestone, we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although performing some geometric transformations of images like rotation, scaling, translation, etc.

Activity 1: Import the required libraries

We need to download yoloV10 from the github

```
[ ] !git clone https://github.com/THU-MIG/yolov10.git
```

```
Cloning into 'yolov10'...
remote: Enumerating objects: 20294, done.
remote: Counting objects: 100% (373/373), done.
remote: Compressing objects: 100% (176/176), done.
remote: Total 20294 (delta 250), reused 306 (delta 197), pack-reused 19921
Receiving objects: 100% (20294/20294), 11.37 MiB | 14.39 MiB/s, done.
Resolving deltas: 100% (14229/14229), done.
```

```
cd yolov10
```

```
/content/yolov10
```

```
[ ] pip install -r requirements.txt
```

```
Show hidden output
```

```
[ ] pip install .
```

Activity 2: Download the pretrained weights

loading pre-trained model yolov10 weights from github

```python
import os
import urllib.request

# Create a directory for the weights in the current working directory
weights_dir = os.path.join(os.getcwd(), "weights")
os.makedirs(weights_dir, exist_ok=True)

# URLs of the weight files
urls = [
    "https://github.com/jameslahm/yolov10/releases/download/v1.0/yolov10n.pt",
    "https://github.com/jameslahm/yolov10/releases/download/v1.0/yolov10s.pt",
    "https://github.com/jameslahm/yolov10/releases/download/v1.0/yolov10m.pt",
    "https://github.com/jameslahm/yolov10/releases/download/v1.0/yolov10b.pt",
    "https://github.com/jameslahm/yolov10/releases/download/v1.0/yolov10x.pt",
    "https://github.com/jameslahm/yolov10/releases/download/v1.0/yolov10l.pt"
]

# Download each file
for url in urls:
    file_name = os.path.join(weights_dir, os.path.basename(url))
    urllib.request.urlretrieve(url, file_name)
    print(f"Downloaded {file_name}")
```

```
Downloaded /content/yolov10/weights/yolov10n.pt
Downloaded /content/yolov10/weights/yolov10s.pt
Downloaded /content/yolov10/weights/yolov10m.pt
Downloaded /content/yolov10/weights/yolov10b.pt
Downloaded /content/yolov10/weights/yolov10x.pt
Downloaded /content/yolov10/weights/yolov10l.pt
```

Activity 3: Load the Dataset:

installing roboflow and connect our dataset from roboflow

```
from roboflow import Roboflow
rf = Roboflow(api_key="fYs9yoBlOL5kubBGu2QY")
project = rf.workspace("spects").project("spects-detections")
version = project.version(3)
dataset = version.download("yolov9")
```

```
loading Roboflow workspace...
loading Roboflow project...
Downloading Dataset Version Zip in Spects-detections-3 to yolov9:: 100%|| 29018/29018 [00:03<00:00, 9282.18it/s]

Extracting Dataset Version Zip to Spects-detections-3 in yolov9:: 100%|| 1520/1520 [00:00<00:00, 6904.88it/s]
```

Milestone 3: training

Now it's time to train our Yolo model:

We have to create **data.yaml:**

```
names:
- spectacles
nc: 1
roboflow:
  license: CC BY 4.0
  project: spects-detections
  url: https://universe.roboflow.com/spects/spects-detections/dataset/3
  version: 3
  workspace: spects
test: test/images
train: train/images
val: valid/images
```

Training yolo v10 model on a custom dataset.

```
!yolo detect train epochs=50 batch=16 model=../weights/yolov10m.pt data=/content/yolov10/Spects-detections-3/data.yaml
```

```
New https://pypi.org/project/ultralytics/8.2.31 available 😃 Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.1.34 🚀 Python-3.10.12 torch-2.0.1+cu117 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=detect, mode=train, model=../weights/yolov10m.pt, data=/content/yolov10/Spects-detections-3/data.yaml, epochs=50, time=None, patience=100, batch=16, imgsz=640,
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/yolov10/Arial.ttf'...
100% 755k/755k [00:00<00:00, 74.4MB/s]
2024-06-13 06:46:23.976794: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when c
2024-06-13 06:46:23.976855: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when or
2024-06-13 06:46:24.096433: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS whe
Overriding model.yaml nc=80 with nc=1

                   from  n    params  module                                       arguments
  0                  -1  1      1392  ultralytics.nn.modules.conv.Conv             [3, 48, 3, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv             [48, 96, 3, 2]
  2                  -1  2    111360  ultralytics.nn.modules.block.C2f             [96, 96, 2, True]
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv             [96, 192, 3, 2]
  4                  -1  4    813312  ultralytics.nn.modules.block.C2f             [192, 192, 4, True]
  5                  -1  1     78720  ultralytics.nn.modules.block.SCDown          [192, 384, 3, 2]
  6                  -1  4   3248640  ultralytics.nn.modules.block.C2f             [384, 384, 4, True]
  7                  -1  1    228672  ultralytics.nn.modules.block.SCDown          [384, 576, 3, 2]
  8                  -1  2   1689984  ultralytics.nn.modules.block.C2fCIB          [576, 576, 2, True]
  9                  -1  1    831168  ultralytics.nn.modules.block.SPPF            [576, 576, 5]
```

```
      Epoch    GPU_mem     box_om     cls_om     dfl_om     box_oo     cls_oo     dfl_oo  Instances       Size
      49/50      9.05G     0.5052     0.3671      1.033     0.5343     0.3395       1.06         12        640: 100% 34/34 [00:20<00:00,  1.69it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 5/5 [00:05<00:00,  1.02s/it]
                   all        152        152      0.973      0.952      0.987      0.773

      Epoch    GPU_mem     box_om     cls_om     dfl_om     box_oo     cls_oo     dfl_oo  Instances       Size
      50/50      9.06G      0.495     0.3625      1.026     0.5425     0.3443      1.062         13        640: 100% 34/34 [00:20<00:00,  1.67it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 5/5 [00:05<00:00,  1.00s/it]
                   all        152        152      0.949      0.941      0.986      0.769

50 epochs completed in 0.426 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 33.5MB
Optimizer stripped from runs/detect/train/weights/best.pt, 33.5MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.1.34 🚀 Python-3.10.12 torch-2.0.1+cu117 CUDA:0 (Tesla T4, 15102MiB)
YOLOv10m summary (fused): 369 layers, 16451542 parameters, 0 gradients, 63.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 5/5 [00:06<00:00,  1.35s/it]
                   all        152        152      0.973      0.952      0.987      0.773
Speed: 5.8ms preprocess, 28.3ms inference, 0.0ms loss, 0.1ms postprocess per image
Results saved to runs/detect/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

Testing

Testing our model with a random image from the test folder. Also, we have saved our best.pt

```
!yolo task=detect mode=predict conf=0.25 save=True model=runs/detect/train/weights/best.pt source=/content/yolov10/Spects-detections-3/test/images

Ultralytics YOLOv8.1.34 🚀 Python-3.10.12 torch-2.0.1+cu117 CUDA:0 (Tesla T4, 15102MiB)
YOLOv10m summary (fused): 369 layers, 16451542 parameters, 0 gradients, 63.4 GFLOPs

image 1/63 /content/yolov10/Spects-detections-3/test/images/080717-f-9963e-014_jpg.rf.73a58c20ff8429deaeb3c3be84522b71.jpg: 640x640 1 spectacles, 35.4ms
image 2/63 /content/yolov10/Spects-detections-3/test/images/090401-o-2222i-008_jpg.rf.61b286b8f39a2e570076629860f0d1ec.jpg: 640x640 2 spectacless, 35.4ms
image 3/63 /content/yolov10/Spects-detections-3/test/images/1024px-dieter-bohlen-o-sole-mio-cala-ratjada_jpg.rf.0e9094e213a8c998d3ab5aa52f677f3d.jpg: 640x640
image 4/63 /content/yolov10/Spects-detections-3/test/images/111003-f-bs505-231_jpg.rf.1806135cff7a2c7f3044df7d2b513a8c.jpg: 640x640 (no detections), 29.8ms
image 5/63 /content/yolov10/Spects-detections-3/test/images/120622-f-zy202-031_jpg.rf.ccc9ec6c8fff9de03df5c2c2ad0a7a74.jpg: 640x640 1 spectacles, 25.7ms
image 6/63 /content/yolov10/Spects-detections-3/test/images/120804-f-eb935-121_jpg.rf.ffe7a9e4de69af160dce82f81b2ddc05.jpg: 640x640 1 spectacles, 25.7ms
image 7/63 /content/yolov10/Spects-detections-3/test/images/140408-f-gj308-021_jpg.rf.d7e219a9c7cd2cfec80abeabeb1e380e.jpg: 640x640 1 spectacles, 25.7ms
image 8/63 /content/yolov10/Spects-detections-3/test/images/140501-f-lx370-003_jpg.rf.1e9ec2c4324aa8e400d5664557cc12a4.jpg: 640x640 1 spectacles, 25.7ms
image 9/63 /content/yolov10/Spects-detections-3/test/images/140720-m-ms007-039_jpg.rf.88db5ba16e7c910837d274f05cc3f4ef.jpg: 640x640 1 spectacles, 20.5ms
image 10/63 /content/yolov10/Spects-detections-3/test/images/150206-f-hk400-001_jpg.rf.009b89f2710d48e72ab7d9668e71e29a.jpg: 640x640 1 spectacles, 17.9ms
image 11/63 /content/yolov10/Spects-detections-3/test/images/150505-f-ip635-024_jpg.rf.ce8555437a3e2dfe31e4d3395adb16b0.jpg: 640x640 1 spectacles, 17.9ms
image 12/63 /content/yolov10/Spects-detections-3/test/images/150919-f-ar004-003_jpg.rf.6386490761fad4902a9957636244600a.jpg: 640x640 1 spectacles, 17.8ms
image 13/63 /content/yolov10/Spects-detections-3/test/images/160511-f-cu844-002_jpg.rf.1c039f95a569d57982b22c5fedc7de99.jpg: 640x640 1 spectacles, 18.0ms
image 14/63 /content/yolov10/Spects-detections-3/test/images/160628-f-zz999-666_jpg.rf.65e39245ba743da38f4bcfa50f7f9780.jpg: 640x640 1 spectacles, 17.9ms
image 15/63 /content/yolov10/Spects-detections-3/test/images/160802-f-fe339-162_jpg.rf.417d2fe35826e1f7799ed2d4fb1120bc.jpg: 640x640 1 spectacles, 16.7ms
image 16/63 /content/yolov10/Spects-detections-3/test/images/161111-f-pj024-005_jpg.rf.c84c2f2ee6698fda1cd0aa900302cf35.jpg: 640x640 1 spectacles, 17.3ms
image 17/63 /content/yolov10/Spects-detections-3/test/images/180418-z-nb545-1181_jpg.rf.e2431151b4c4716a0b405637b969aa0f.jpg: 640x640 1 spectacles, 17.0ms
image 18/63 /content/yolov10/Spects-detections-3/test/images/180426-z-nb545-1186_jpg.rf.49a366d0118350e3779eefa0653186e0.jpg: 640x640 1 spectacles, 17.1ms
image 19/63 /content/yolov10/Spects-detections-3/test/images/180602-d-db155-001_jpg.rf.2e17203dcf8692a5b92bf8370e2e5f7c.jpg: 640x640 2 spectacless, 17.6ms
```

Displaying detected image in training notebook:

displaying a detected image from the saved folder runs/detect/predict4

```
from PIL import Image
im = Image.open('runs/detect/predict4/692.jpg')
display(im)
```

Milestone 4: Application Building

Now that we have trained our model, let us build our flask application which will be running in our local browser with a user interface.

In the flask application, the input parameters are taken from the HTML page These factors are then given to the model to know to predict the type of Garbage and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the “Image” button, the next page is opened where the user chooses the image and predicts the output.

Activity 1: Create HTML Pages

- We use HTML to create the front-end part of the web page.
- Here, we have created 3 HTML pages- home.html, intro.html, and upload.html
- home.html displays the home page.
- Intro.html displays an introduction about the project
- upload.html gives the emergency alert For more information regarding HTML https://www.w3schools.com/html/
- We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.
- Link:CSS, JS

Create app.py (Python Flask) file: -

Write the below code in Flask app.py python file script to run the Object Detection Project.

```
.py > predict
from flask import Flask, request, render_template, redirect, url_for, send_from_directory
import os
from ultralytics import YOLOv10
from PIL import Image
from werkzeug.utils import secure_filename

app = Flask(__name__)

model = YOLOv10('best1.pt')

UPLOAD_FOLDER = 'static/uploads'
ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png', 'mp4', 'avi', 'mkv'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/')
def index():
    return render_template('index.html')
```

To upload image in UI:

```
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        if 'file' not in request.files:
            return 'No file part'

        file = request.files['file']

        if file.filename == '':
            return "No selected file"

        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(filepath)

            if filename.lower().endswith(('.jpg', '.jpeg', '.png')):
                img = Image.open(filepath)
                model(img, save=True)

            elif filename.lower().endswith(('.mp4', '.avi', '.mkv')):
                model(filepath, save=True)

            return redirect(url_for('result', original_filename=filename))

    return render_template('prediction-page.html')
```

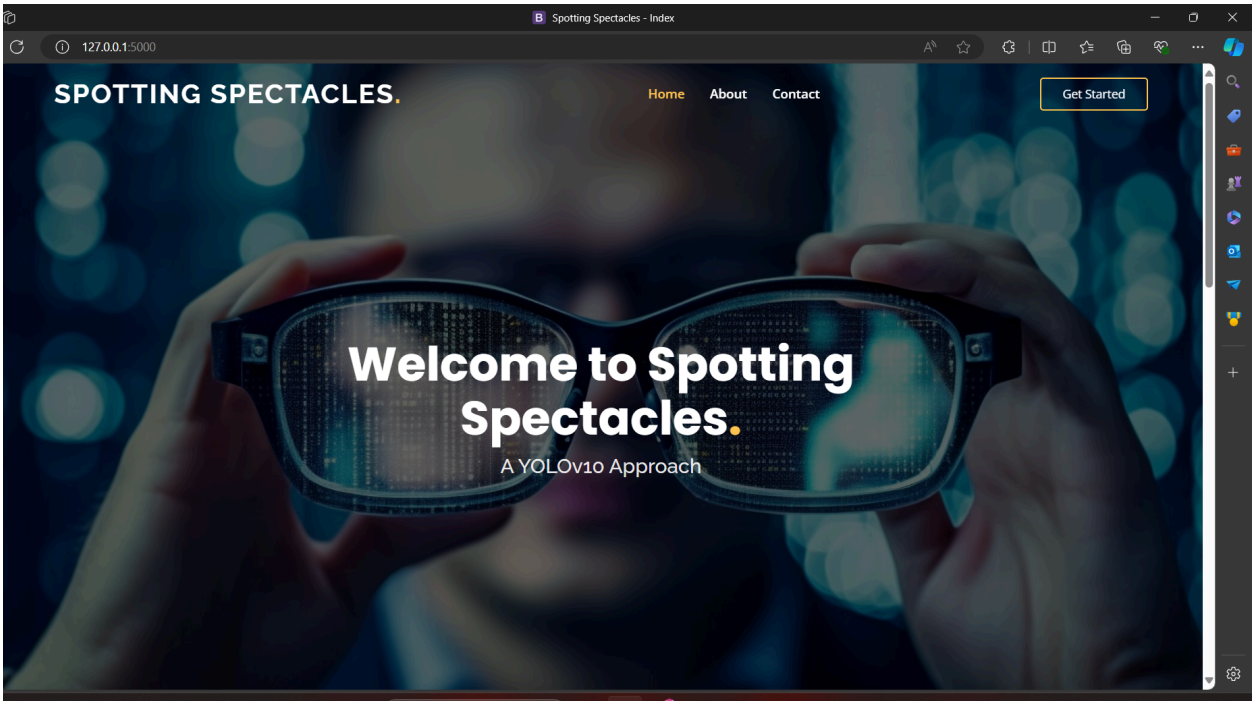
To display the image in UI:

```python
@app.route('/result/<original_filename>')
def result(original_filename):
    (variable) latest_subfolder: str
    (folder_path) if os.path.isdir(os.path.join(folder_path, f))]
    latest_subfolder = max(subfolders, key=lambda x: os.path.getctime(os.path.join(folder_path, x)))
    directory = folder_path+'/'+latest_subfolder
    print("printing directory: ",directory)
    files = os.listdir(directory)
    latest_file = files[0]

    print(latest_file)

    filename = os.path.join(folder_path, latest_subfolder, latest_file)

    file_extension = filename.rsplit('.', 1)[1].lower()

    environ = request.environ
    if file_extension == 'jpg':
        return send_from_directory(directory,latest_file) #shows the result in seperate tab

    else:
        return "Invalid file format"

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

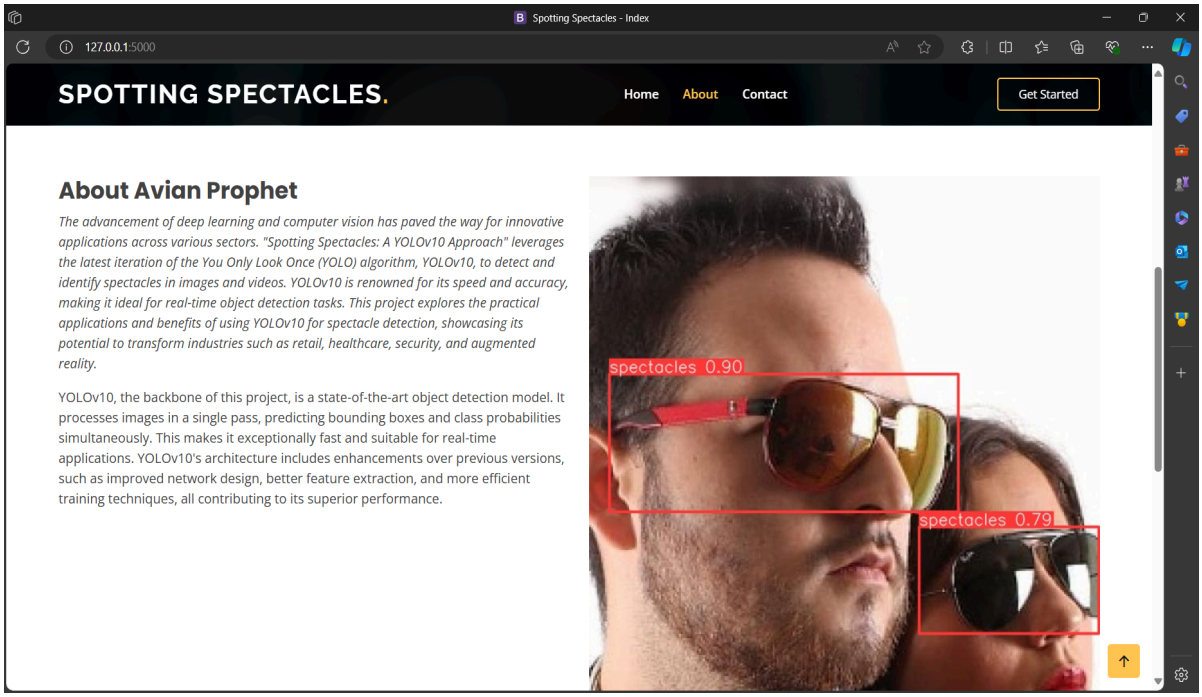
Getting local host in the terminal while running app.py:

```
(base) D:\>conda activate yoloV10

(yoloV10) D:\>cd yolo

(yoloV10) D:\yolo>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 168-769-906
127.0.0.1 - - [19/Jun/2024 10:32:38] "GET /predict HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2024 10:32:38] "GET / HTTP/1.1" 200 -
```

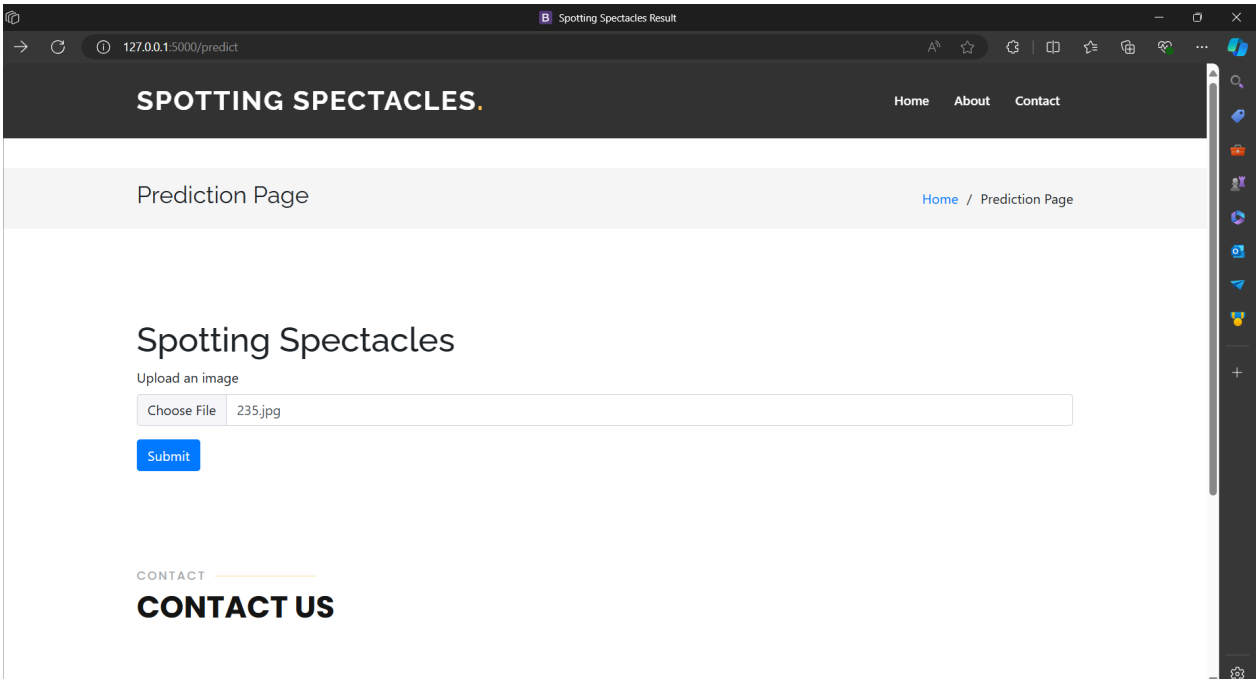
Index.html is displayed below:



About Section is displayed below:

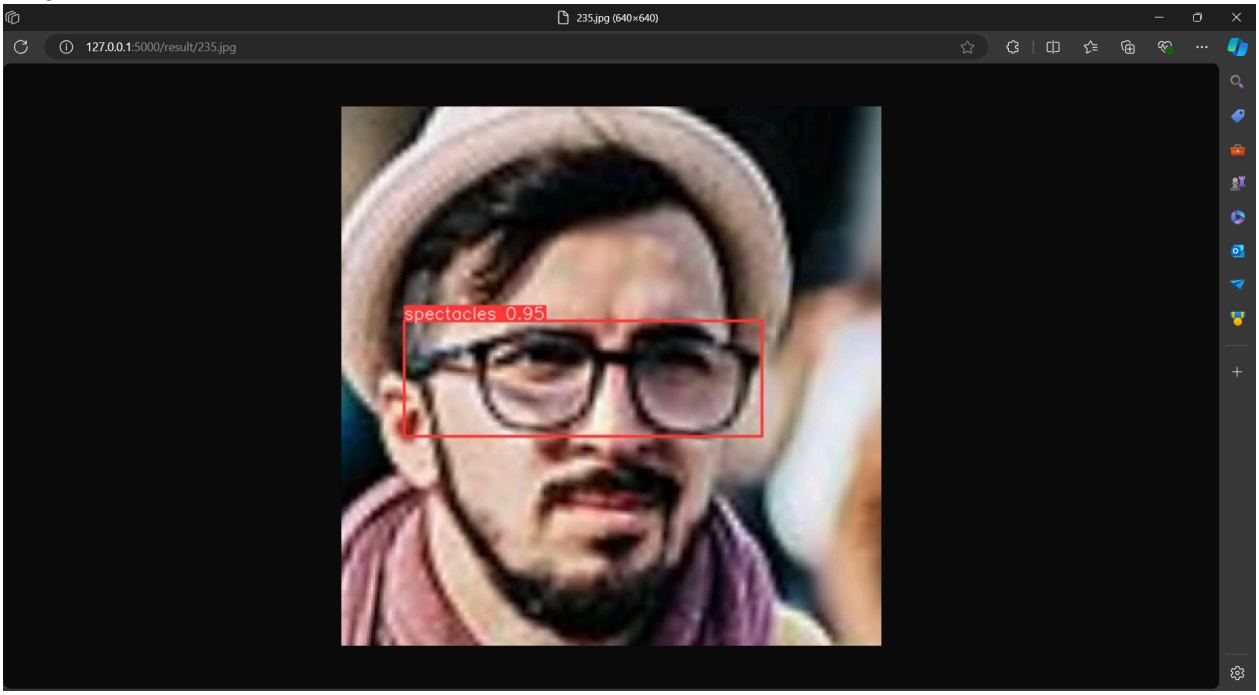


Upload and Output Page is displayed below Input:1

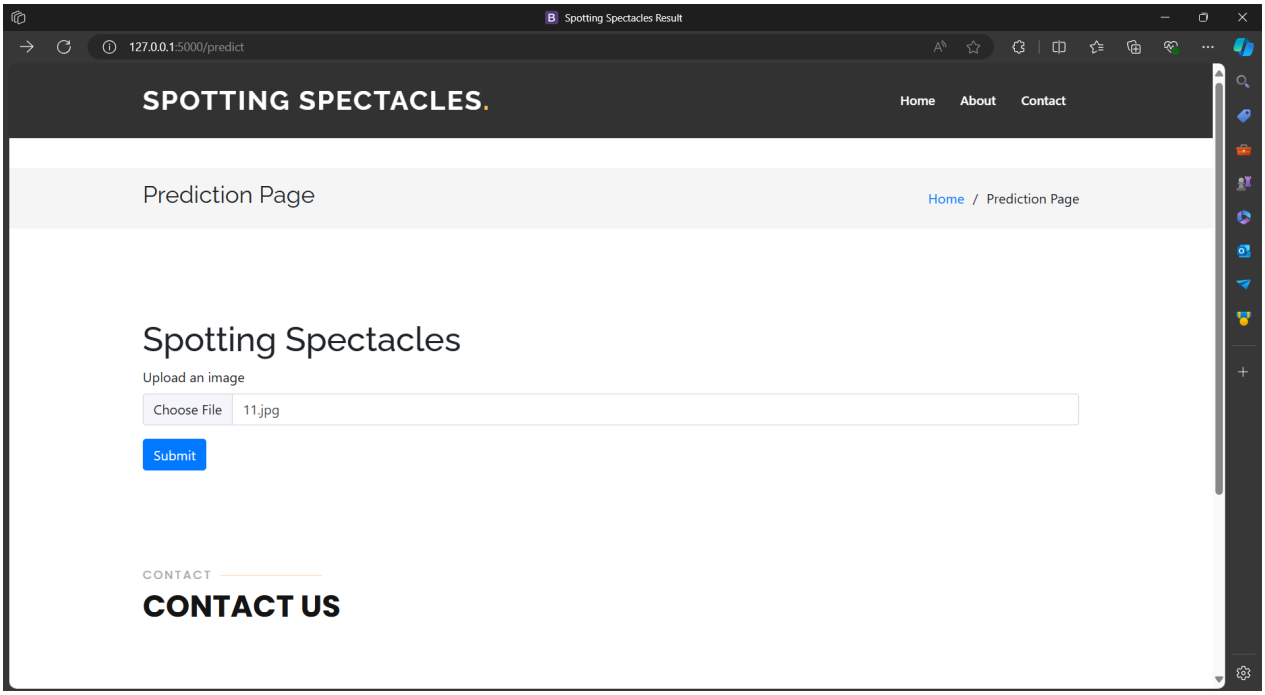


Final Output (after you click on Upload) is displayed as follows:

Output 1



Input:2



Output 2