**Ship Detection Using YOLOv8**

**Introduction**

This project is about detecting ships in water using a deep learning model called YOLOv8. We built a web application where users can upload an image, and the system will identify ships in the image. The application is developed using **HTML, CSS, JavaScript, Python, and Flask**.

**Objectives**

* Allow users to upload an image.
* Process the image using the **YOLOv8** model to detect ships.
* Display the processed image with bounding boxes around detected ships.
* Provide a user-friendly web interface.

**Technologies Used**

* **Frontend:** HTML, CSS, JavaScript
* **Backend:** Python, Flask
* **Machine Learning Model:** YOLOv8
* **Libraries:** OpenCV, Flask, Ultralytics YOLO

**How It Works**

1. The user uploads an image through the web interface.
2. The image is sent to the **Flask backend**.
3. The YOLOv8 model processes the image and detects ships.
4. The processed image (with detected ships) is sent back to the frontend.
5. The frontend displays the result to the user.

**Code Overview**

**1. Frontend (HTML, CSS, JavaScript)**

* The user interface allows image upload.
* JavaScript handles the image preview and submission.
* The **Submit** button sends the image to the Flask backend.
* The detected image is displayed on the webpage.

**2. Backend (Flask, Python)**

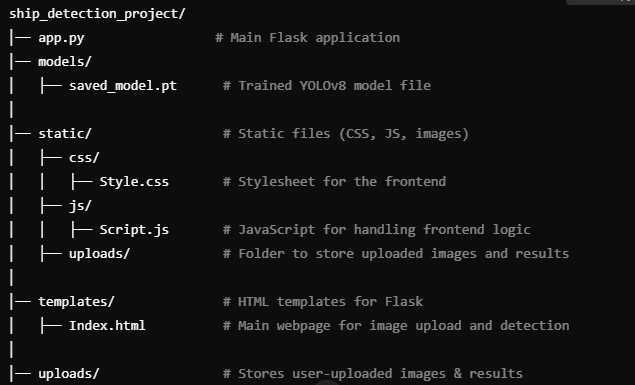
* Flask receives the uploaded image and saves it.
* The YOLOv8 model is loaded to process the image.
* The model detects ships and draws bounding boxes.
* The processed image is returned to the frontend.

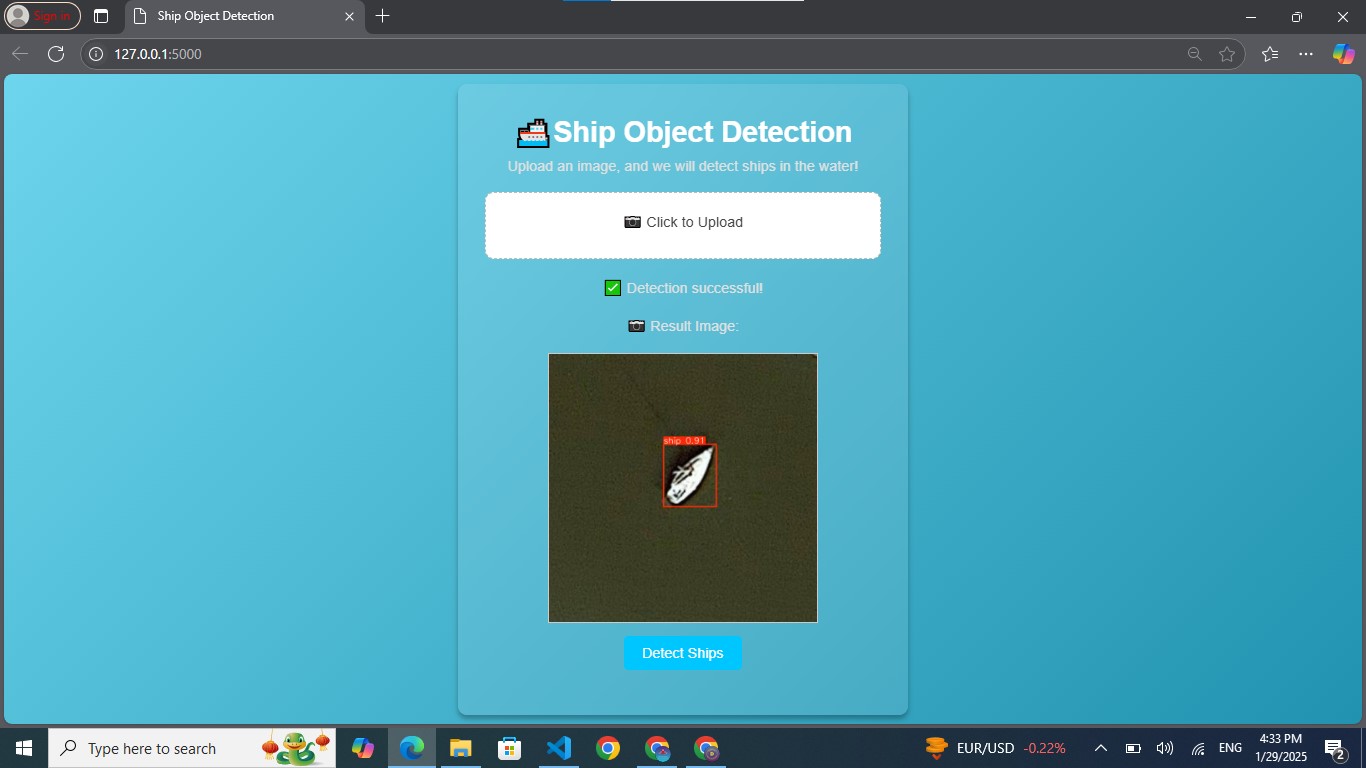
**Conclusion**

This project successfully demonstrates **real-time ship detection** using deep learning. By combining **YOLOv8, Flask, and a simple web interface**, users can easily detect ships in an image.

This report explains the project in simple words.

**Structure of Files**

****

**Screenshot of Front-end**

**THE END**