Overview of the model:

This is an object detection model built using a pre-trained 'YOLO11n' model, in conjunction with a FastAPI back end and HTML+CSS+JS front end. It is containerized using Docker.

Users can upload their images through the UI, after which the original image will be shown alongside the processed image with the objects detected. There is also a table displaying the coordinates of the boxes and other text data sent by the model.

All the uploaded and processed images and the JSON data are saved in the 'app/results' folder.

Steps followed while building this app:

1. Researched the various open-source object detection models available. YOLO seemed the optimal choice with good performance, a well-built interface that makes integration quite easy. YOLO11n is the latest version, so I chose that.
2. Went through the Ultralytics YOLO11n model documentation (Link). Checked the way to process images through it, the functionalities it offers, and how to get back the data in required JSON format. Played around with the model, trying to understand its working. It offers a great interface and a wide range of functionalities, so it is quite easy to use.
3. Built the back end using FastAPI and UI using plain HTML. Took help of ChatGPT to generate the front end part, and for some debugging on the back end while integrating the model responses. The back end saves the uploaded image, the processed image after getting it back from the model, and the JSON response from it.
   I could have optimized and modularized the FastAPI code a bit more, but I decided to keep it in the current format as it is a short codebase which won't be used by many team members, and hopefully it still is very understandable.
4. Dockerized the application, with the required installations to ensure compatibility on all platforms.

5. Improved the UI to present the JSON data in a tabular format for easier comprehension.

References:

1. Google
2. https://docs.ultralytics.com/modes/predict/#probs
3. ChatGPT