A brief explanation of the chosen tech stack and AI approach.Tech

Stack:

For the web application, I utilized Next.js along with Tailwind CSS to create a responsive and modern user interface. Next.js provides server-side rendering capabilities and a robust routing system, while Tailwind CSS offers utility-first styling, enabling rapid design adjustments.

For the machine learning backend, I implemented FastAPI, which facilitates seamless integration with the Python-based machine learning libraries I am using, including YOLOv8 for object detection. This stack allows for efficient communication between the front end and the ML model, ensuring quick response times and a smooth user experience for inventory detection tasks.

Model Selection:

I Employed a pre-trained YOLOv8 model for real-time object detection, chosen for its speed and accuracy in identifying multiple objects within images. I chose YOLOv8 for inventory detection due to its excellent balance of speed and accuracy, making it ideal for real-time applications. Compared to other models like Faster R-CNN or SSD, YOLOv8 processes images quickly while maintaining high detection precision, which is crucial for managing inventory effectively. Its user-friendly implementation with pre-trained weights allows for easy integration and fine-tuning, making it adaptable to various environments.

Any challenges faced during development and how they were resolved?

One significant challenge I encountered was the inability to deploy the application on Vercel or other free hosting services due to the large size of the machine learning model, which exceeded their limits for free deployments. This could be resolved by opting for a paid tier, specifically deploying the application on an AWS EC2 instance, which provides the necessary resources to accommodate the large model.

Additionally, I faced accuracy issues with the model, particularly when detecting items that were too close together or overlapping, which led to misclassifications. If I had more time and access to GPU resources, I could further customize the model specifically for food detection, enhancing its accuracy and performance in these challenging scenarios. This would enable a more tailored solution that better meets the unique needs of the application.

Fridge Ingredient Detection with AI

Upload a picture of your fridge (or any image containing food), and the system will identify the visible ingredients in the image along with an estimated quantity of each item

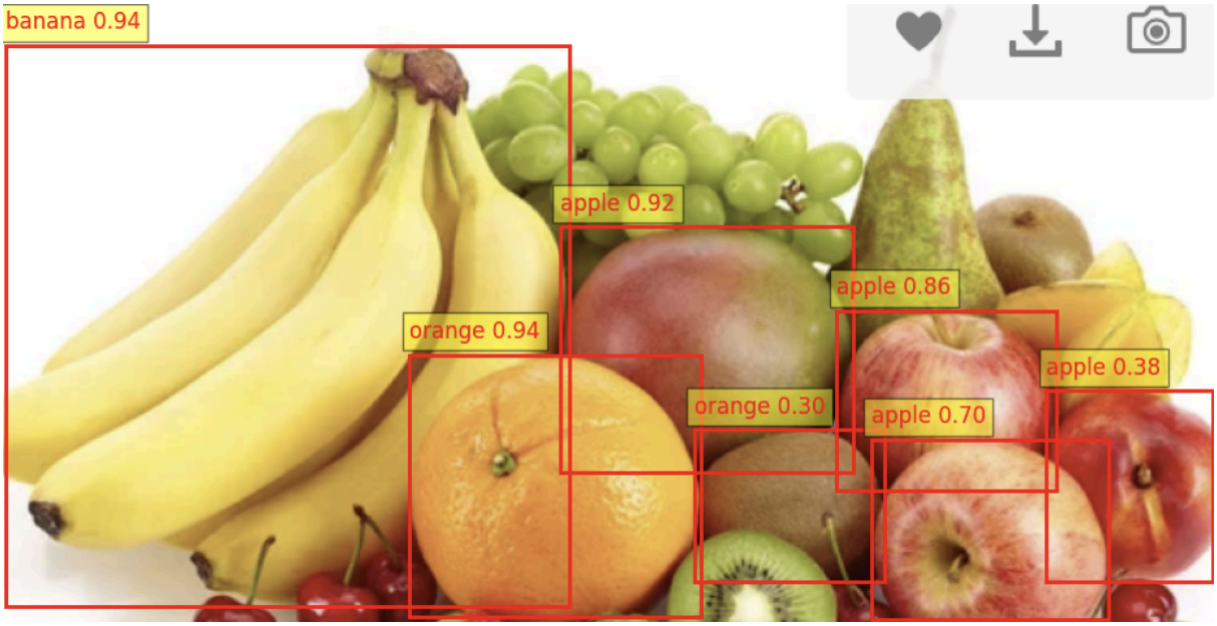
Choose File Screenshot …t 6.21.49 PM

Upload

Detected Items:

apple: 4
banana: 1
orange: 2

Detected Image:



Fridge Ingredient Detection with AI

Upload a picture of your fridge (or any image containing food), and the system will identify the visible ingredients in the image along with an estimated quantity of each item

Choose File Screenshot …t 2.08.45 PM

Upload

Detected Items:

apple: 2

Detected Image: