**Object Detection - Implementation Guide**

**Table of Contents**

1. [Project Overview](#project-overview)
2. [Prerequisites](#prerequisites)
3. [Installation](#installation)
4. [Implementations](#implementations)
   * [Streamlit Version](#streamlit-version)
   * [FastAPI Version](#fastapi-version)
   * [Flask Version](#flask-version)
5. [API Documentation](#api-documentation)

**Project Overview**

This project provides three different implementations of a YOLOv8-based object detection system:

* Streamlit: For rapid prototyping and data apps
* FastAPI: For modern, fast API development
* Flask: For traditional web applications

Each implementation offers the same core functionality with different advantages.

**Key Features**

* Upload images (JPG, JPEG, PNG)
* Adjust confidence threshold
* Real-time object detection
* Download processed images and results
* JSON output of detections
* Visual bounding box display

**Prerequisites**

Required software and tools:

* Python 3.10 or higher
* Necessary packages
* git

**Installation**

1. Clone or create project directory
2. Create virtual environment
   * Conda
   * Python
3. Install common dependencies

**Implementations**

**Streamlit Version**:

**Additional Dependencies**

pip install streamlit

**Directory Structure**

streamlit/

├── streamlit\_app.py

├── uploads/

└── detections/

**Running the App**

streamlit run streamlit\_app.py

**Key Features**

* Single-file implementation
* Interactive UI components
* Real-time updates
* Built-in caching
* Automatic hot-reloading

**FastAPI Version**

**Additional Dependencies**

pip install fastapi uvicorn python-multipart jinja2

**Directory Structure**

fastapi\_app/

├── main.py

├── static/

│ ├── uploads/

│ └── detections/

└── templates/

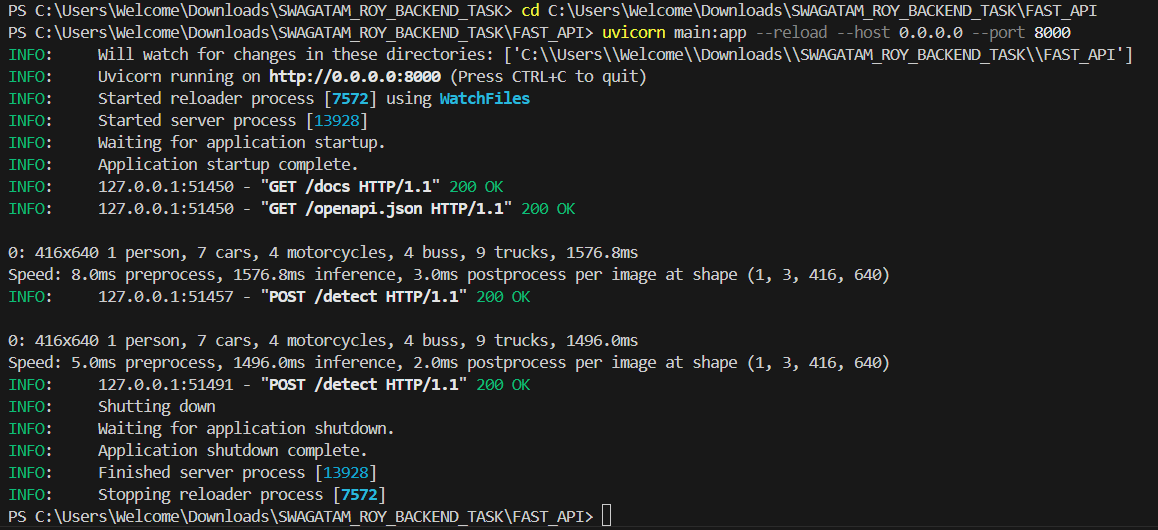
└── index.html

**Running the App**

uvicorn main:app --reload --host 0.0.0.0 --port 8000

**Key Features**

* Modern async/await syntax
* OpenAPI documentation
* High performance
* Type checking
* Automatic API documentation



**Flask Version**

**Additional Dependencies**

pip install flask python-dotenv

**Directory Structure**

flask\_app/

├── app.py

├── static/

│ ├── uploads/

│ └── detections/

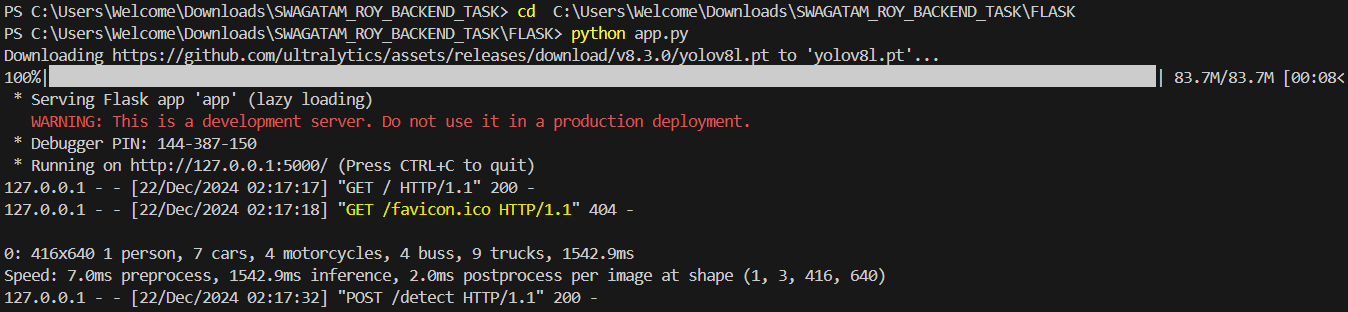
└── templates/

└── index.html

**Running the App**

flask run --host=0.0.0.0 --port=5000

python app.py



**Key Features**

* Traditional web framework
* Simple routing
* Extensive documentation
* Large ecosystem
* Easy to understand

**API Documentation**

**Endpoints (FastAPI & Flask)**

**GET /**

* Purpose: Serve main web interface
* Response: HTML page

**POST /detect**

* Purpose: Process uploaded image
* Parameters:
  + image: File (required)
  + confidence: float (optional, default=0.25)
* Response: JSON with detection results

**GET /download/{filename}**

* Purpose: Download processed files
* Parameters:
  + filename: string
* Response: File download