**Dangerous Object Detection**

**Introduction**

In recent years, the need for real-time dangerous object detection has become increasingly important in various domains, including public safety, security, and industrial environments. This project, "Dangerous Object Detection," addresses this need by developing a system that can identify hazardous objects in real-time using machine learning. The system is designed to be user-friendly and accessible, with both a backend for processing and a frontend for interaction.

**Project Objectives**

* **Real-Time Detection**: Enable the detection of dangerous objects in real-time using live camera feeds.
* **User Accessibility**: Provide an intuitive user interface for non-technical users.
* **Comprehensive Reporting**: Generate detailed reports of detected objects for further analysis.

**System Overview**

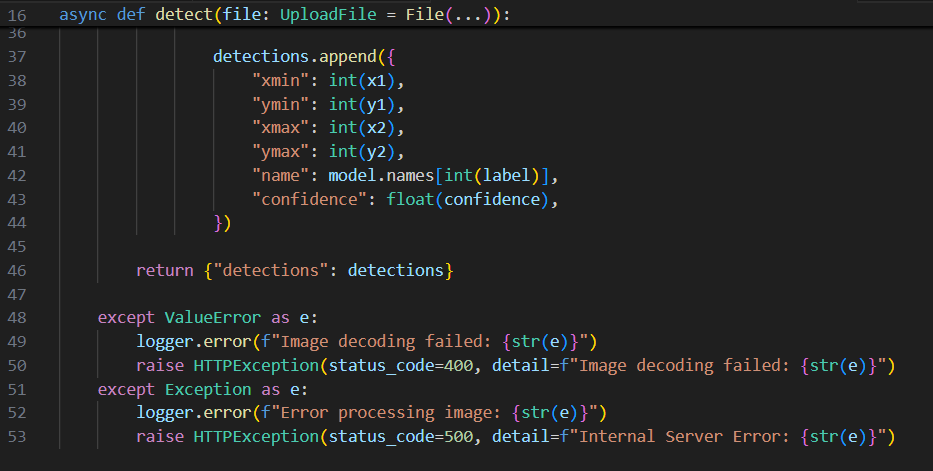
The Dangerous Object Detection system is composed of two primary components: the backend and the frontend. These components work together to provide a seamless experience for the user, from detecting dangerous objects to generating detailed reports.

**Backend**

The backend is responsible for handling the core functionality of the system, including object detection and data processing. It leverages a pre-trained YOLOv8 model to identify potentially hazardous objects in the camera feed. The backend is implemented using FastAPI, a modern web framework that allows for fast and efficient handling of HTTP requests.

**Main.py :-**

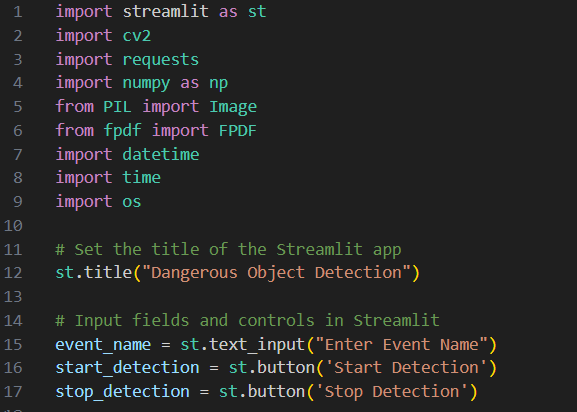
****

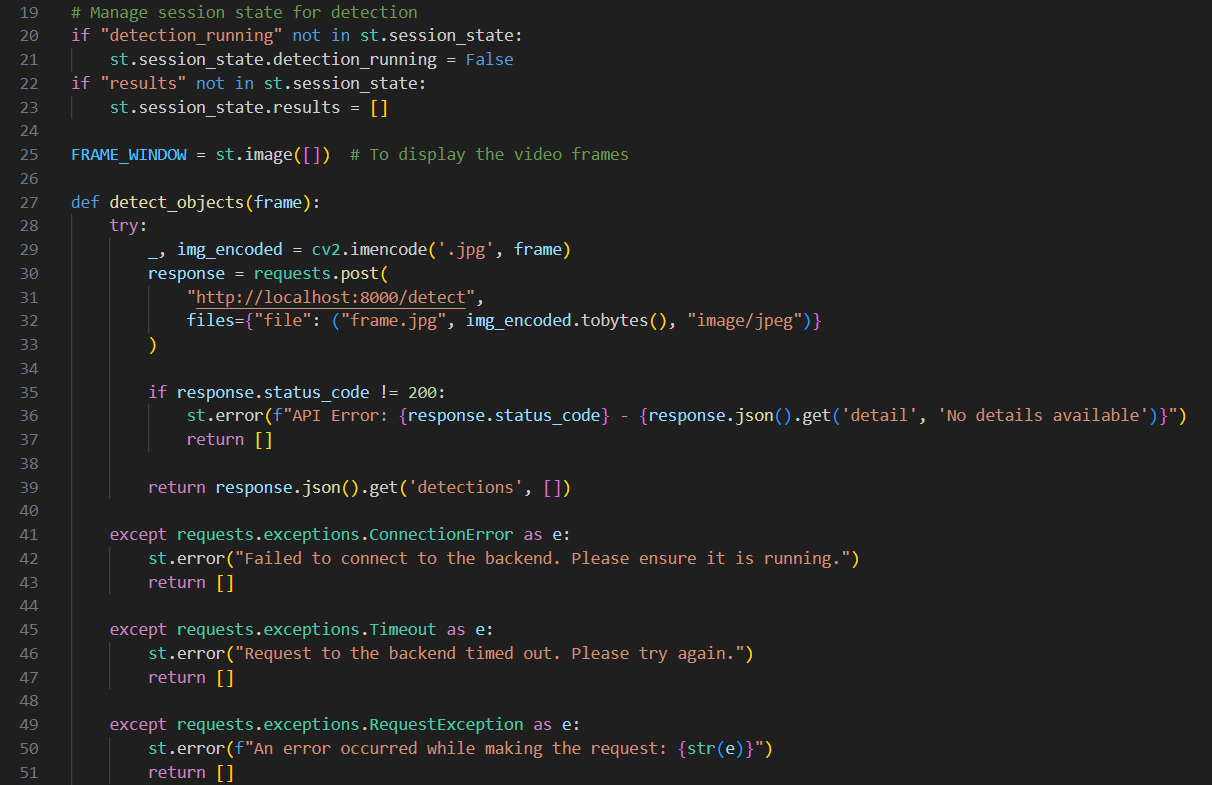
****

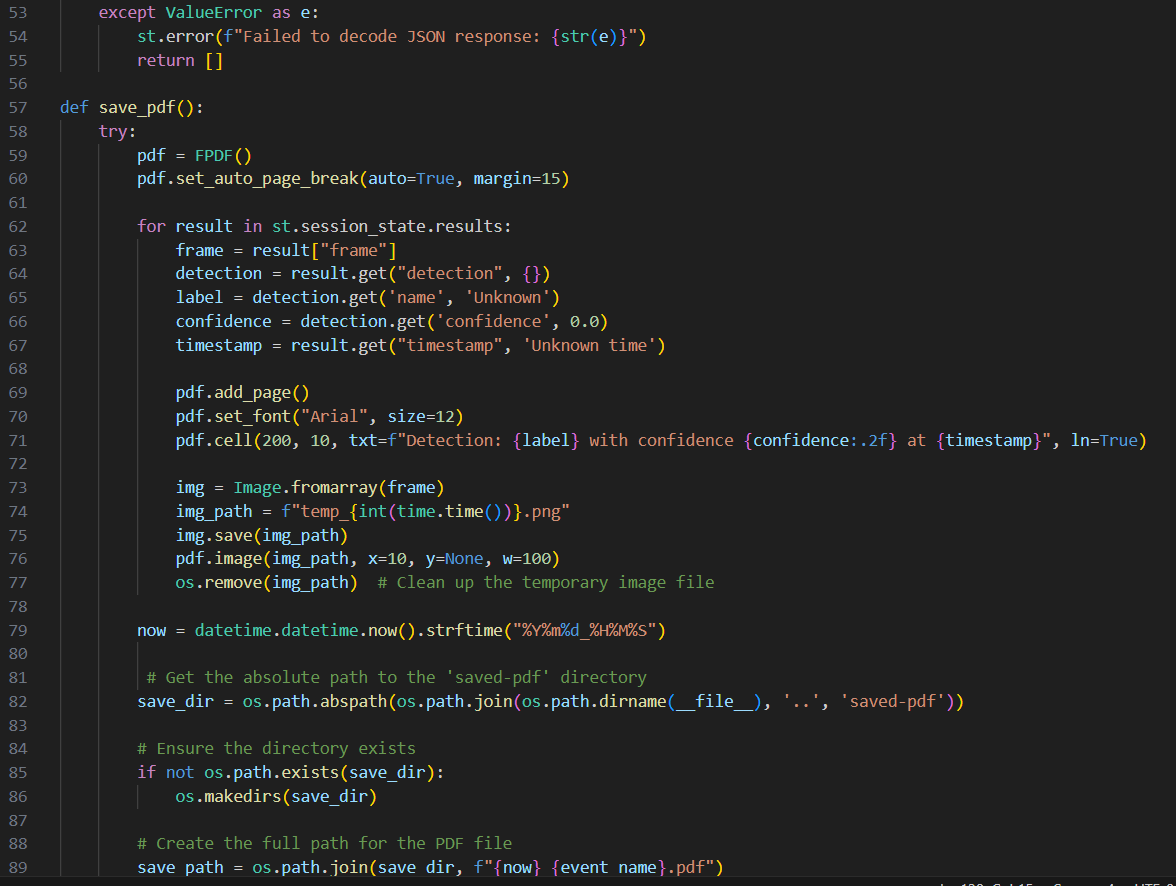
**Frontend**

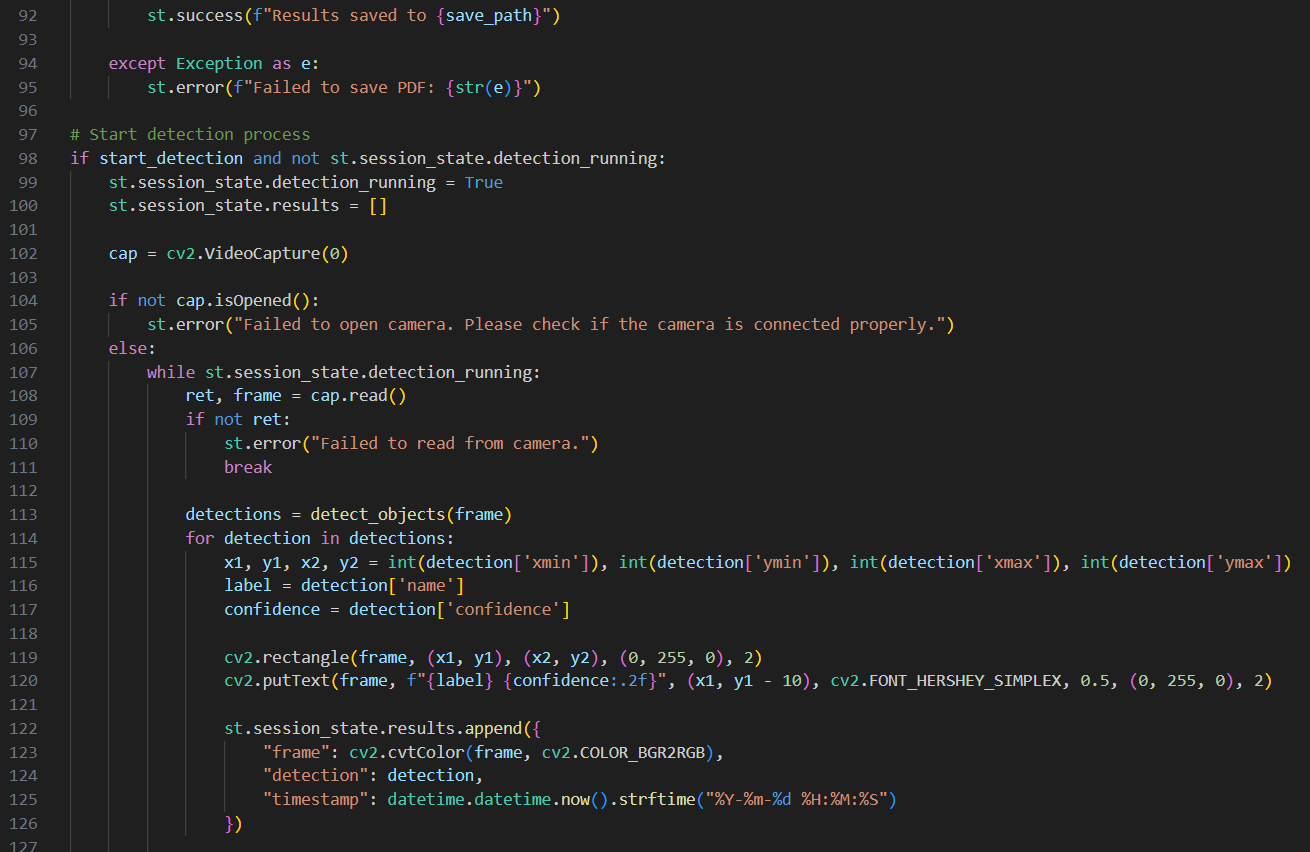
The frontend provides the user interface for the system, allowing users to interact with the detection process. It is built using Streamlit, a Python library that enables the creation of interactive web applications. The frontend allows users to start the detection process, view the live camera feed, and generate PDF reports of detected objects.

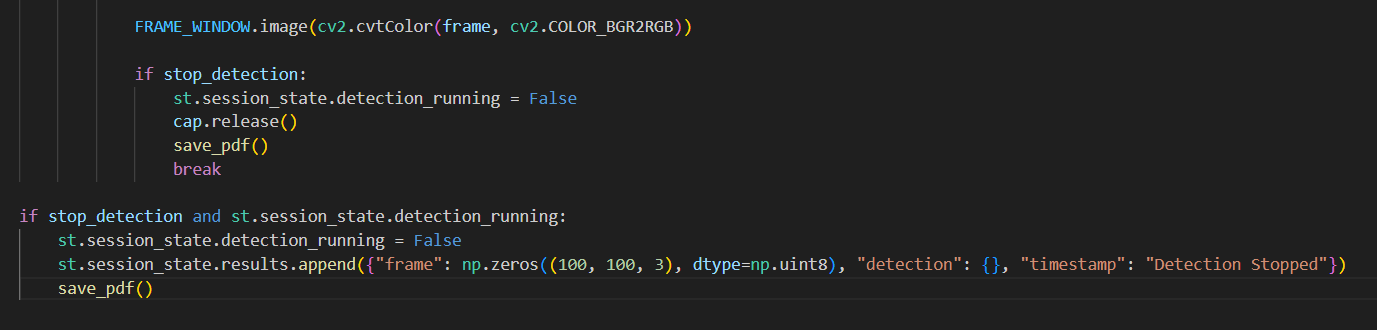
**app.py: -**

****









**PROJECT STRUCTURE**

The project directory is organized as follows:

**final\_project/**

**├frontend/**

**│ ├app.py # Streamlit frontend application**

**├backend/**

**│ ├main.py # FastAPI backend application**

**├datasets/ # Datasets for training, validation, and testing**

**│ ├test/**

**│ │ ├images/ # Test images**

**│ │ └── labels/ # Test labels**

**│ ├ val/**

**│ │ ├ images/ # Validation images**

**│ │ └── labels/ # Validation labels**

**│ ├train/**

**│ │ ├images/ # Training images**

**│ │ └── labels/ # Training labels**

**├saved-pdf/ # Directory for saving generated PDFs**

**├requirements.txt # Python dependencies**

**├README.md # Project documentation (this file)**

**├config.py # Configuration file for project settings**

**├test\_dangerous\_objects\_detection.py # Test file for object detection**

**├best.pt # Pre-trained model weights**

**├.gitignore # Git ignore file**

**├paths.json # JSON file containing resource paths**

**├Training\_Model/ # Scripts and resources for training the model**

**│ ├best.py # Best performing model script**

**│ ├dog.jpeg # Sample image used in training**

**│ ├Training.ipynb # Jupyter notebook for model training**

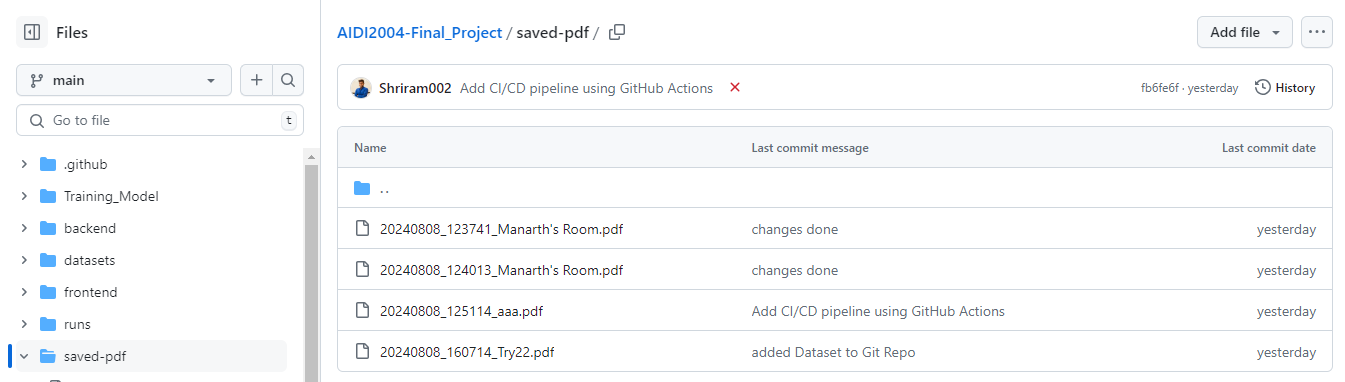
**│ ├yolov8n.pt # YOLOv8 model weights**

**├runs/ # Directory for storing resultsFeatures**

* **Real-time Object Detection:** Detects objects using a camera feed in real-time.



* **PDF Report Generation:** Creates PDF reports containing frames with detected dangerous objects and timestamps.



* **User-friendly Interface:** Streamlit-based interface for easy interaction.
* **Continuous Detection:** Continues detecting objects until manually stopped by the user.

**Installation**

* Python 3.9+
* Git (optional, for cloning the repository)
* A virtual environment (recommended)

**Steps to Install**

1. **Clone the Repository:**

git clone https://github.com/yourusername/dangerous-object-detection.git

cd dangerous-object-detection

1. **Set Up the Environment:**

Create a virtual environment (optional but recommended):

python -m venv venv

source venv/bin/activate # On Windows use `venv\Scripts\activate`

1. **Install the required dependencies**:

pip install --upgrade pip

pip install -r requirements.txt

**Usage**

Backend (Object Detection)

* **Run the FastAPI server:**

uvicorn backend.main:app –reload

Frontend (User Interface)

* Start the Streamlit application

streamlit run frontend/app.py

**Testing**

Tests are located within the backend and frontend directories. To execute the tests:

We used **PYTEST** ensure the necessary environment (e.g., a running fastapi server) is available if required by the tests.

**Deployment**

CI/CD Pipeline

The project includes a CI/CD pipeline configured using GitHub Actions. The pipeline covers:

1. **Testing:** Runs on every push or pull request to the main or develop branches.
2. **Build:** Builds the application post successful testing.
3. **Deployment:** Deploys the application on changes to the main branch.

Deployment Options

For production deployment, especially when GPU resources are necessary:

* **Google Colab:** Suitable for development, not for production.
* **AWS EC2 with GPU:** Recommended for production, though cost may be a factor.
* **Heroku (with limitations):** Suitable for small-scale deployment, but does not support GPU.