Main Goal of the Project

The main goal of this project is to create a simple web application that uses a computer's camera to **automatically detect unsafe actions** in a workplace. It pretends to be an AI that watches for problems, like a worker not wearing a helmet, and then shows **immediate warnings** on a dashboard.

How It Works

The system works in a continuous loop between your web browser and the backend server.

1. **See:** You open the **index.html** file in your browser. It asks for permission and turns on your webcam.
2. **Send:** Your browser captures frames (pictures) from the video and sends them to the backend server (**app.py**).
3. **Think (AI Simulation):** The server receives the picture. It runs a function that **pretends to be an AI** and decides if there's a safety problem in the image.
4. **Act:** If a problem is found:
    - The server saves the details into the database file (**safety_monitor.db**).
    - It instantly sends a message back to your browser with the alert information.
5. **Show:** Your browser receives the message and updates the webpage **in real-time**, showing a red box on the video and adding the warning to the "Recent Alerts" list.

This see-send-think-act-show cycle repeats every half-second, creating a live monitoring system.

Technology Used (Tech Stack)

We use a few key technologies to make this work.

Frontend (The Website You See)

- **HTML:** Provides the basic structure and layout of the page.
- **CSS:** Adds all the styling—colors, fonts, and positioning—to make it look good.
- **JavaScript:** Makes the website interactive. It handles turning on the camera, sending video to the server, and displaying the alerts it receives.

Backend (The Server "Brain")

- **Python:** The main programming language used to build the server.
- **Flask:** A simple Python framework that helps run the web server and handle requests from the browser.
- **Gevent-WebSocket:** A special tool that allows for real-time, two-way communication. It's how the server can instantly "push" alerts to the website without waiting for the website to ask.

Database (Where Data is Stored)

- **SQLite:** A very simple, file-based database. It's used to keep a history of all the safety alerts that have been generated.

What Each File Does

Each file in the project has a specific job.

- index.html **The Website.** This single file is the entire user interface. It contains the HTML for the page layout, the CSS for styling, and the JavaScript code that runs in your browser to make everything work.
- app.py **The Server Brain.** This is the most important backend file. It runs the Flask server, simulates the AI analysis, handles the WebSocket connection for real-time alerts, and saves information to the database.
- database_setup.py **The Database Creator.** This is a simple utility script. You run it **only once** to create the safety_monitor.db file with the correct tables inside, ready to store alerts.
- safety_monitor.db **The Alert Logbook.** This file *is* the database. It is created automatically when you run the setup script. All information about every alert is stored in this file.
- captures/ (Folder) **The Photo Album.** This folder is created automatically. When you click the "Capture Image" button on the website, the snapshot is saved here.