**📌 Project Approach & Workflow Overview**

**🔹 Project Overview**

This project aims to simulate a **drone-based smart surveillance system**.  
Instead of using a real drone, we use a **mobile phone** to simulate drone behavior.  
The phone captures **live video** and **live telemetry data** (such as GPS location, altitude, heading, etc.) and sends it to a **laptop**, where we process everything using computer vision and AI tools.

The system detects **what objects** are visible in the video, **describes** what is happening in each scene in simple English, **indexes** the events (logs them), and later allows a **user to ask questions** like "Where did the truck appear?" or "What was seen after 12 PM?".

The main goal is to create a **modular, real-time AI surveillance system** that could later be expanded into real drones or security cameras.

**🔹 Project Pipeline (Step-by-Step)**

**▶️ 1. Mobile Phone Setup (Pseudo-Drone)**

* The phone acts like a flying drone.
* It runs two apps:
  + **IP Webcam App**: Streams live video over WiFi.
  + **Phyphox App**: Streams live telemetry data (like GPS, altitude, and accelerometer readings).

This way, we simulate both **visual input** and **movement data** from a real drone.

**▶️ 2. Video & Telemetry Ingestion (Receiving Data)**

* On the laptop:
  + **Video Ingest Module** connects to the IP Webcam stream and captures video frames one-by-one.
  + **Telemetry Ingest Module** connects to Phyphox through HTTP, fetching GPS, heading, altitude, and sensor data.

Both modules are **synchronized** using timestamps, so each video frame matches with the correct drone position.

If either camera feed or telemetry connection breaks, the system automatically stops, showing a safety message like:

*"Drone returning to dock due to telemetry failure."*

**▶️ 3. Object Detection and Scene Description**

* **Object Detection Models**:
  + **YOLOv8** is used for real-time, fast object detection (like detecting "person", "car", "bottle").
  + **CLIP** is used as a secondary option for semantic object understanding (better when you want a lightweight model).
* **Scene Description**:
  + The list of detected objects is passed to an LLM (Language Model).
  + Originally, we planned to use **Phi-2** (an efficient LLM), but due to file corruption issues, we **reverted to using ChatGPT-2**.
  + ChatGPT-2 generates a short natural sentence like:

*"A person is standing next to a white truck in the parking area."*

This description makes it easy for humans and the AI agent to understand what's happening in each frame.

**▶️ 4. Event Indexing and Storage**

* Every frame, along with:
  + Frame ID
  + Timestamp
  + Latitude, Longitude, Altitude
  + Drone Heading
  + Description is stored in a **JSONL (JSON Lines) file**.

Two types of logs are maintained:

* **Session Log** (temporary: stores last few sessions)
* **Master Index** (permanent: stores all captured frames/events)

This makes it easy to later search across all recorded drone sessions.

**▶️ 5. LangChain Agent for Querying**

* A **LangChain-based agent** is used to answer human questions about the drone observations.
* It loads the **master log file** and can answer questions like:
  + "Show me all events where a refrigerator was detected."
  + "What objects were seen after 3 PM?"
* The agent can also support **follow-up questions** using memory, like:

"Where did the person go after being near the car?"

This makes the system **interactive**, almost like talking to the drone's brain!

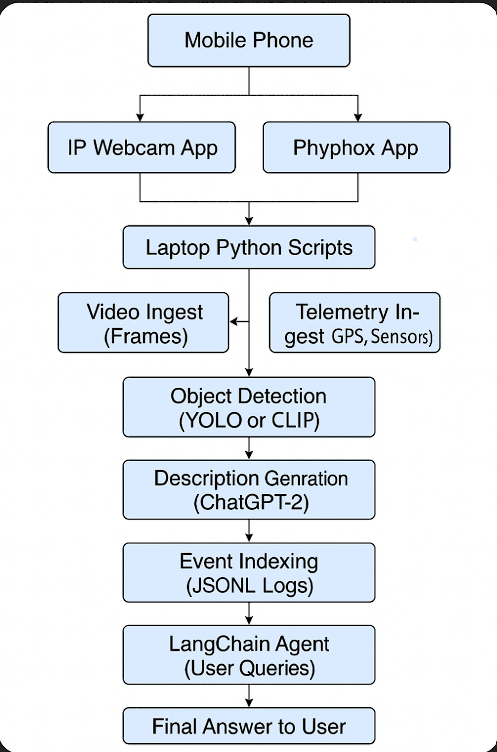
**▶️ 6. Emergency Handling System**

* To simulate real-world drone behavior, if there is a **problem** like:
  + Video feed lost
  + Telemetry data missing
  + Sensor gives bad data then the system **immediately stops** and prints a warning like:

*"Drone encountering issue, returning to dock for investigation."*

This is critical for safe operations in real drone deployments.

**▶️ 7. Flow Chart**

****

Img. Explaining the working of Complete project.