# ■■ IR-GPT Engineer’s Guide

By Adiwakar | Security+ Certified | M.S. Cybersecurity Risk Management, Indiana University

## 1. Quick Overview

IR-GPT is a Retrieval-Augmented Generation (RAG)-based assistant for incident response, built using Python, Streamlit, and ChromaDB. It operates locally with Ollama models, aligning recommendations with NIST SP 800-61 and NIST CSF. This guide covers setup, architecture, and best practices for accuracy and governance.

## 2. Codebase Structure

| app/ | Streamlit UI, pipeline orchestration, summarizer, prompts |
|---|---|
| data/playbooks/ | NIST/CSF aligned playbooks (.md/.txt/.pdf) |
| .chroma/ | Local ChromaDB vector cache (safe to delete) |
| prompts/system_ir.md | NIST-aligned rulebook and JSON output format |
| requirements.txt | Dependencies for RAG and UI |

## 3. Pipeline Workflow

1■■ User enters an incident or log description 2■■ ChromaDB retrieves relevant playbook chunks 3■■ Model prompt combines: system role + style guardrails + retrieved context + user input 4■■ Ollama model (e.g., Mistral, Phi-3) generates a structured response with JSON 5■■ Streamlit displays the summary, structured fields, and retrieved snippets

## 4. Core Components

- **app.py**: Streamlit interface and JSON extraction logic - **pipeline.py**: Retrieval + Generation logic using ChromaDB and Ollama - **summarizer.py**: Optional regex-based log summarizer - **system_ir.md**: Defines NIST mapping, tone, and JSON schema

## 5. Setup & Commands

```bash cd ~/ir-gpt-starter python3 -m venv .venv source .venv/bin/activate pip install -r requirements.txt ollama pull mistral export IRGPT_MODEL="mistral" cd app && streamlit run app.py ```

## 6. Cache & Edit Rules

- Edit **playbooks** → Delete `.chroma/` before rerunning - Edit **prompts** → No deletion needed, reloads automatically - ChromaDB rebuilds embeddings for all playbooks on launch

## 7. Deployment Options
```

- **Local Demo:** Run via Streamlit for privacy and cost-free testing - **GitHub Portfolio:** Push repo with README, screenshots, and docs - **Server/VM:** Deploy with nginx proxy or Docker for team access

## 8. Quality & Governance Controls

- **Retrieval grounding**: All responses cite playbook context - **Temperature:** Keep low (0.1–0.2) for accuracy - **Approval gate:** Encodes human-in-the-loop design - **Framework alignment:** NIST SP 800-61 + CSF phases mapped for audit readiness

## 9. Troubleshooting Quicklist

- No JSON parsed → tighten schema or lower temperature - Old content → delete `.chroma/` - Model not found → `ollama pull ` - “Port in use” → Ollama already running - Double page_config → keep only one call in app.py