Agentic RAG Invoice Matcher

An AI-powered pipeline for invoice flagging, evidence retrieval, and approval workflows

Internship Project Submission – AI Engineer Intern

6th Sense AI

Submitted by:

Satish Kumar Singilese

B.Tech – Artificial Intelligence & Data Science (2025)

Email: satishkumarcse2003@gmail.com

GitHub Repository: https://github.com/ssk-2003/agentic-rag-invoice-matcher

Submitted to:

6th Sense AI – Internship Evaluation Committee

Role: AI Engineer Intern

Date of Submission: August 31, 2025

System Architecture:

System architecture of the Agentic RAG Invoice Matcher showing the pipeline from

User Query

↓

Planner

↓

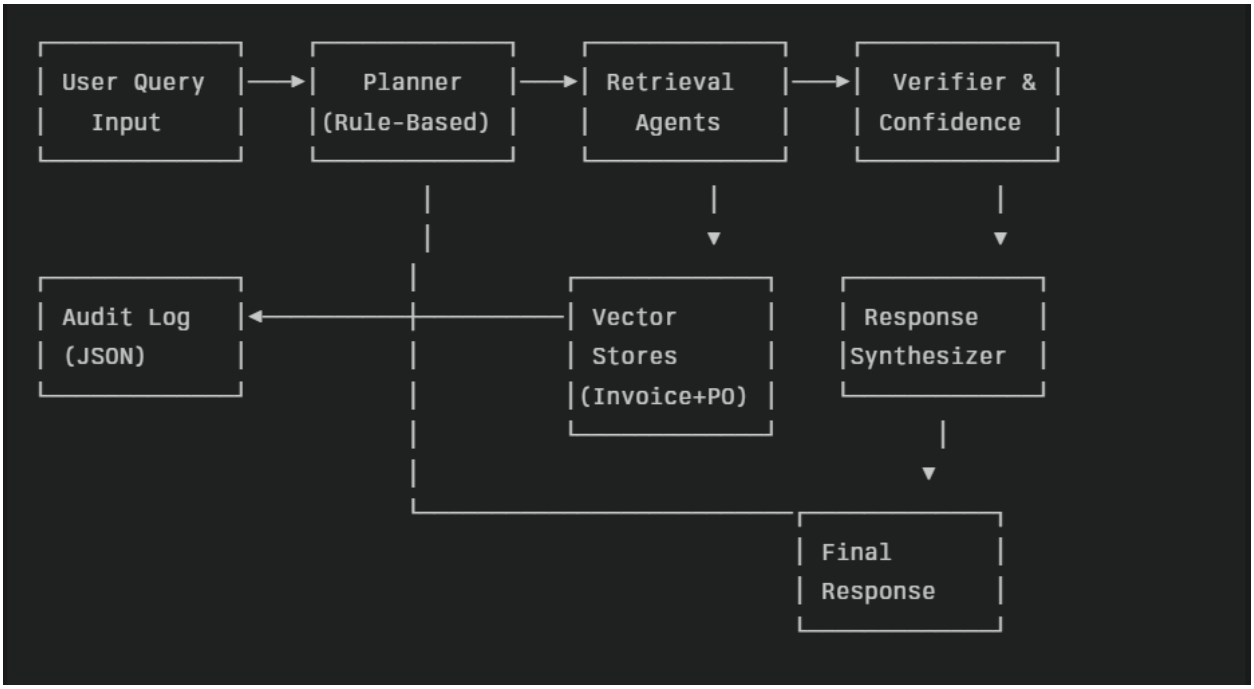
Retrieval Agents

↓

Verifier & Audit Log

↓

Response Synthesizer



Code & Setup / README Overview:

GitHub Repository: https://github.com/ssk-2003/agentic-rag-invoice-matcher

This section explains how to run the project, maps components to code files, and lists known limitations.

Setup Environment

1. Create and activate a virtual environment:
    - Windows:
    - python -m venv venv
    - venv\Scripts\activate
    - Linux/Mac:
    - python -m venv venv
    - source venv/bin/activate
2. Install dependencies:

pip install -r requirements.txt

Prepare Data & Vector Store

python app/data/mock_invoices.py

python app/data/vector_store.py

Run Tests / Demo

python test_system.py

(Optional) Start Web Dashboard

pip install streamlit

streamlit run app/frontend/streamlit_app.py

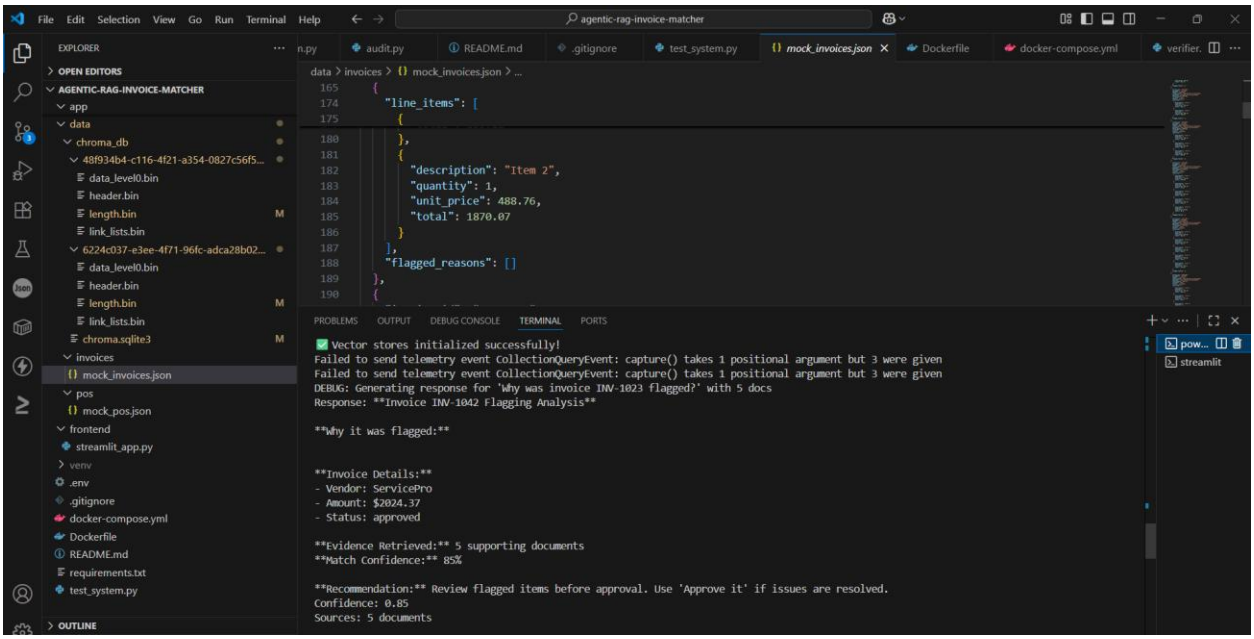
File/Folder	Purpose
app/agents/planner.py	Rule-based planner for user query interpretation
app/agents/rag_system.py	Orchestrates agents and produces answers
app/data/mock_invoices.py	Generates demo invoices & POs
app/data/vector_store.py	Builds vector search database
test_system.py	CLI testing/demo of the system
requirements.txt	Project dependencies
README.md	Project documentation

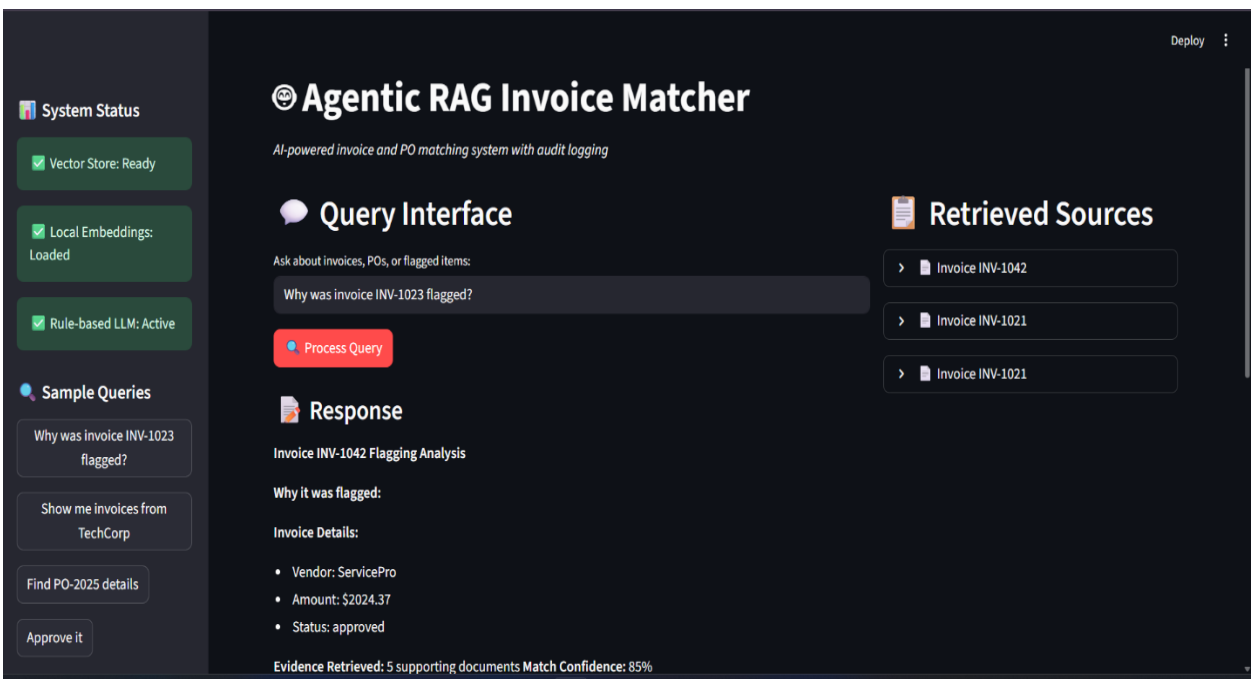
Known Limitations

- Fully local demo; no connection to real APIs or LLMs.
- Mock invoices and POs; not connected to real ERP/financial systems.
- Simplified planner, retriever, and audit trail logic.

Demo Screenshot 1:

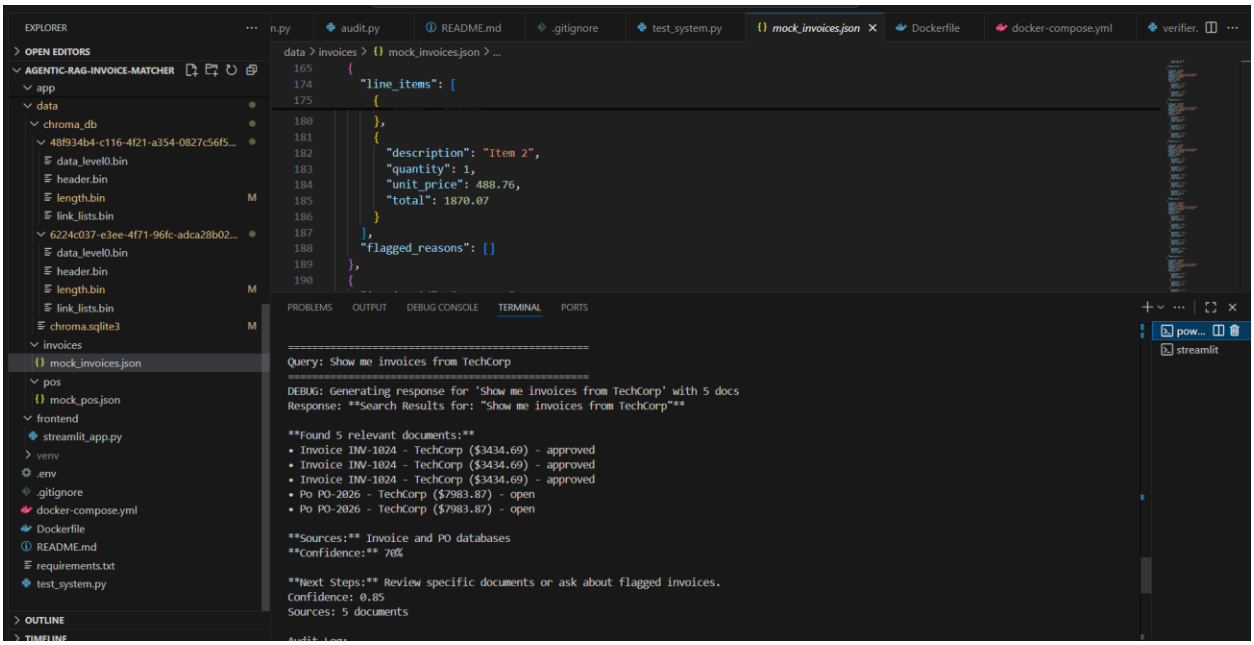
System response to query "Why was invoice INV-1023 flagged?" showing explanation, retrieved evidence, and confidence score.

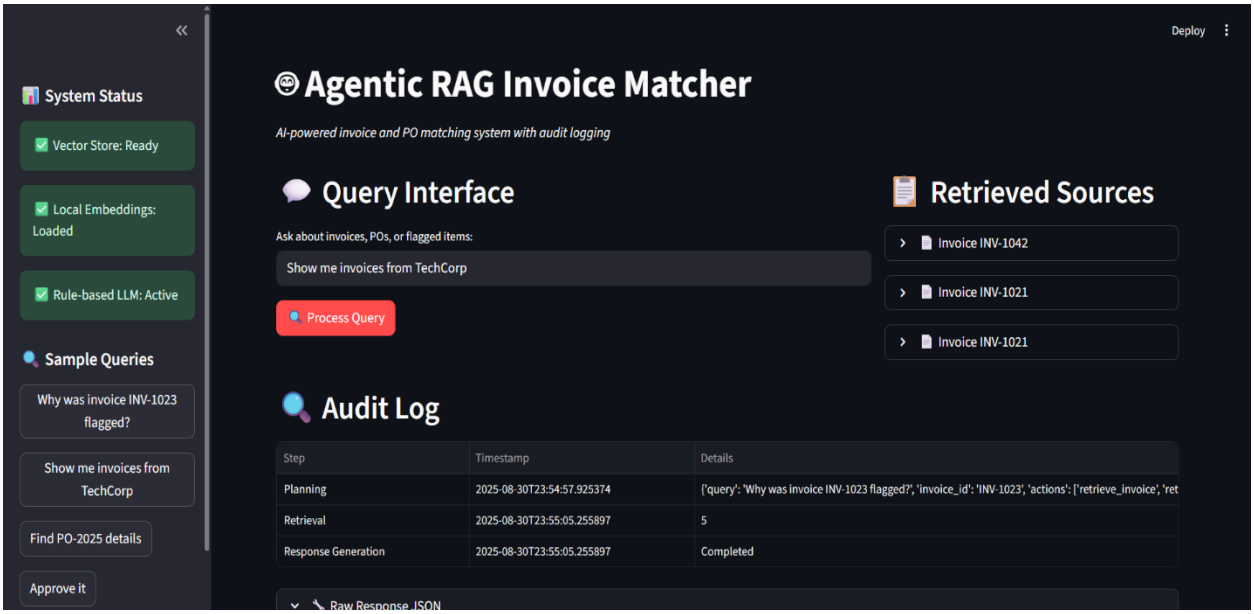




Demo Screenshot 2

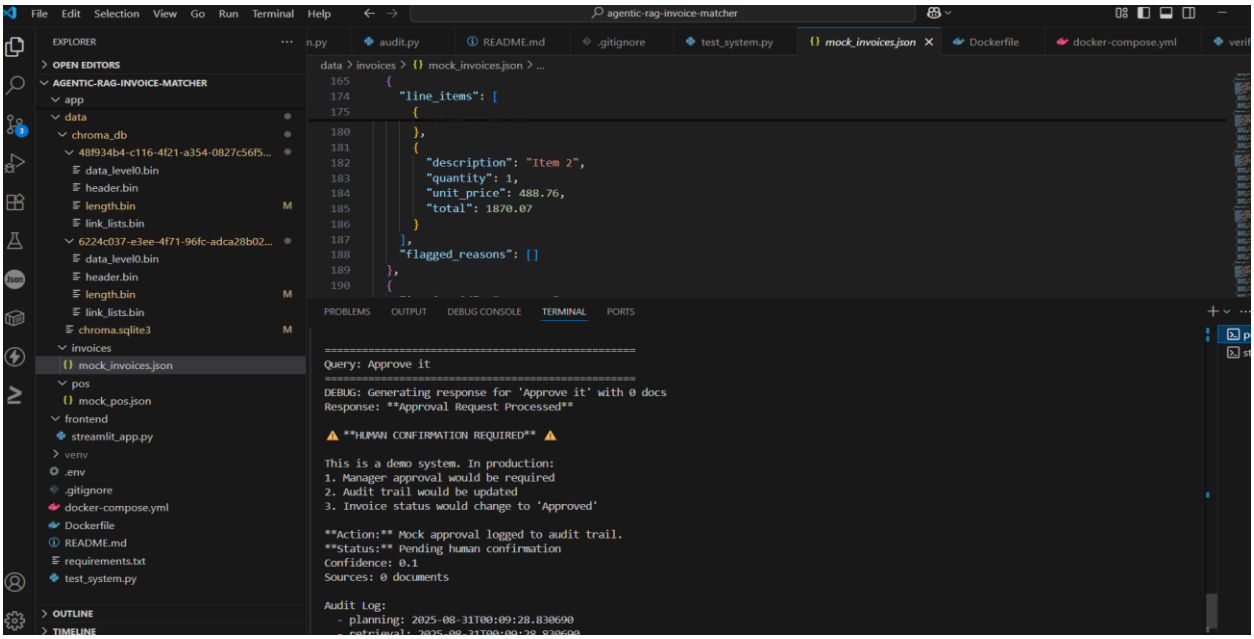
System response to query "Show me invoices from TechCorp" with retrieved invoices and related purchase orders.

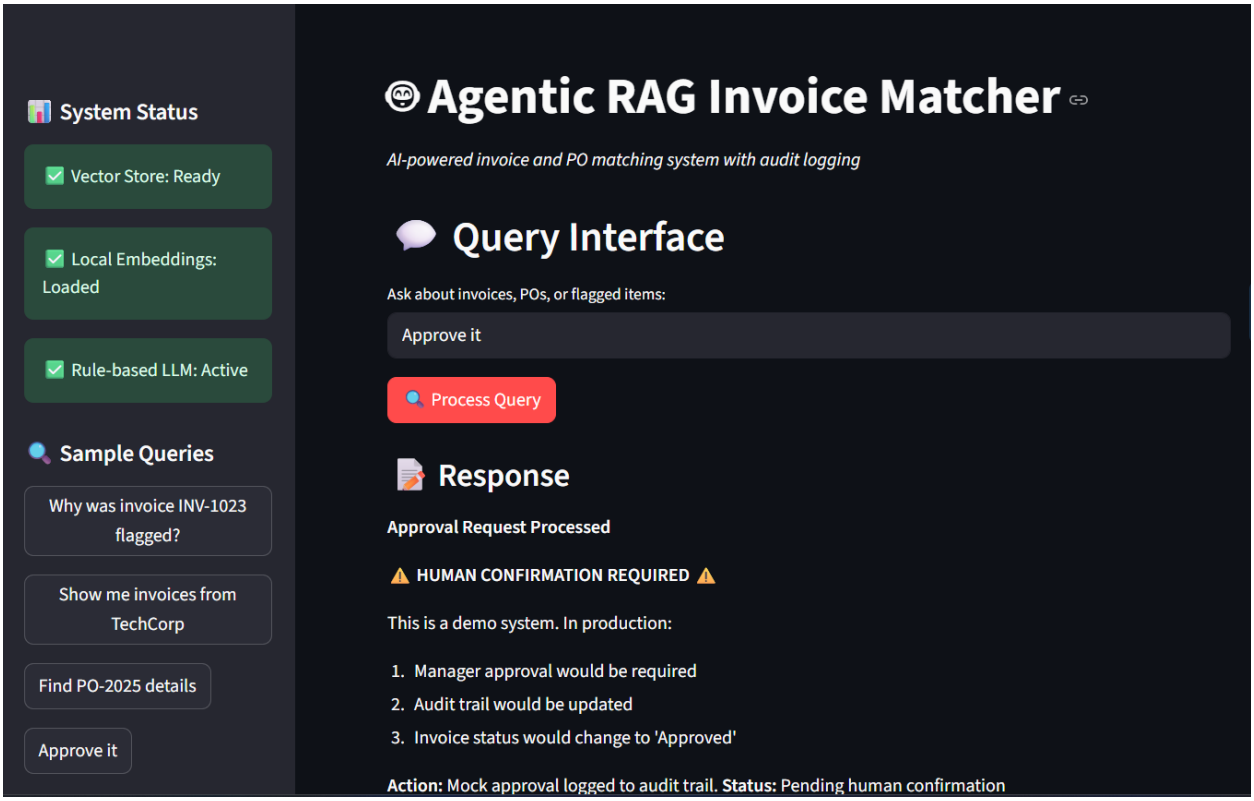




Demo Screenshot 3

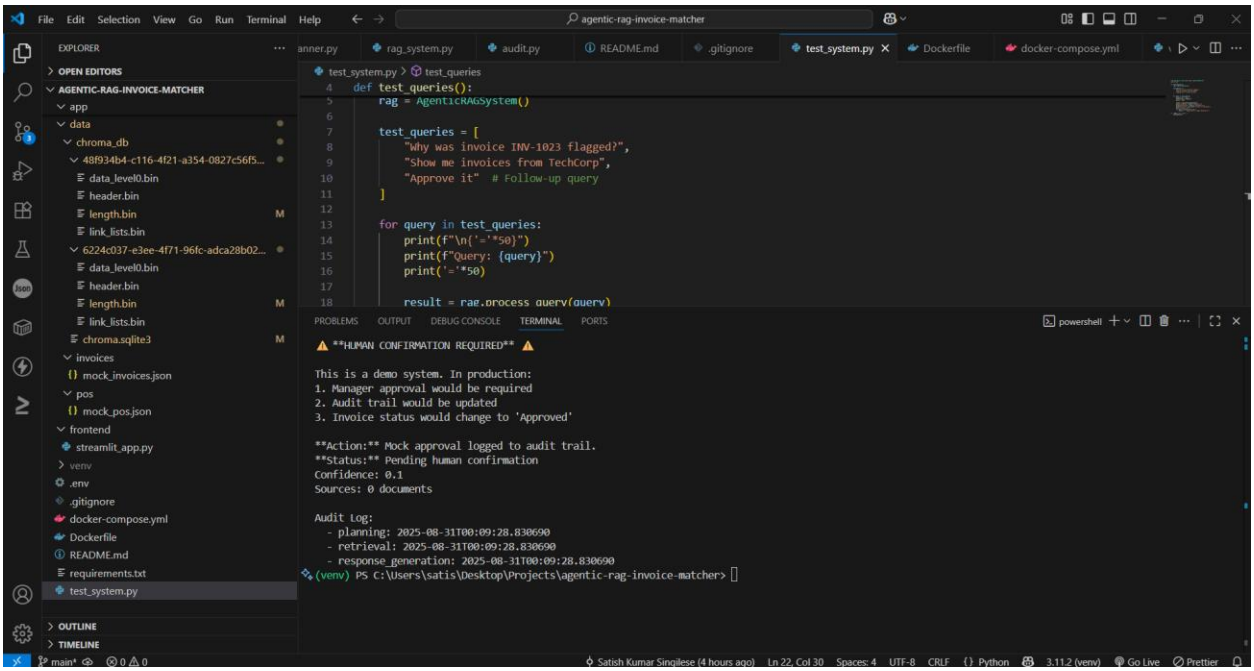
System handling query "Approve it" – approval workflow initiated with human-in-the-loop confirmation.





Demo Screenshot 4

JSON audit log snippet capturing planner decision, retrieval results, and response generation for traceability.



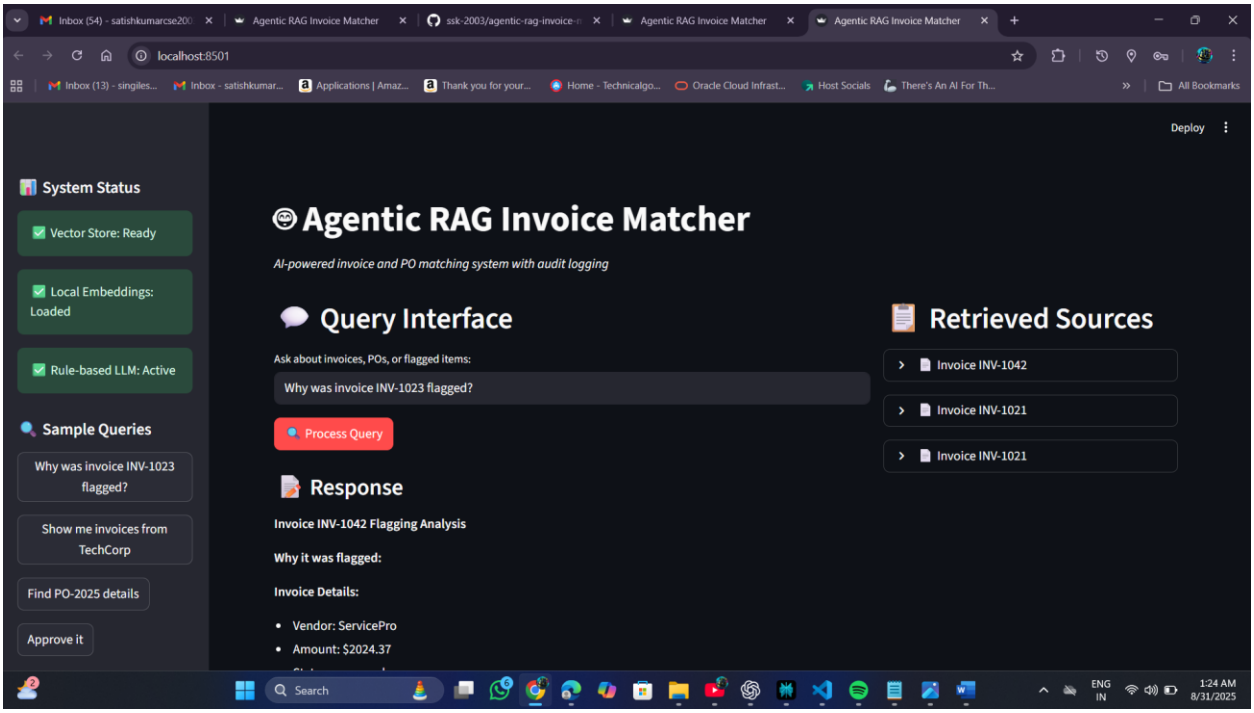

```
{"step":"planning","timestamp":"2025-08-31T01:04:15.123456","input":"Why was invoice INV-1023 flagged?"}
{"step":"retrieval","timestamp":"2025-08-31T01:04:20.654321","sources":["INV-1023","PO-2045"]}
{"step":"response_generation","timestamp":"2025-08-31T01:04:20.660000","response_length":192}
```


Demo Screenshot 5

Streamlit Evidence Panel

Streamlit UI showing evidence panel for query --

"Why was invoice INV-1023 flagged?", with vendor details, amount, and status.



Executive Summary

- This project demonstrates an **Agentic Retrieval-Augmented Generation (RAG) pipeline** for **invoice and purchase order (PO) matching**.
- The system explains **why invoices are flagged**, retrieves **supporting evidence**, and supports **follow-up queries** such as approvals, while maintaining a **complete audit log**.
- The demo includes both a **Command Line Interface (CLI)** and a **Streamlit Web UI** for interactive exploration.

Implemented Features:

- **Planner:** Rule-based logic to interpret user intent.
- **Retrieval Agents:** Vector search over invoices and purchase orders.
- **Verifier & Audit Log:** Confidence scoring with JSON-based logs.
- **Response Synthesizer:** Rule-based summaries**.**
- **CLI Demo:** test_system.py for command-line testing.
- **Streamlit Demo**: Web UI for interactive queries and evidence inspection.

Omissions:

- No integration with real ERP/financial systems (mock dataset used).
- No external web-search agents (local retrieval only).
- Limited follow-up memory (only one-step supported).
- No adaptive ML-based planner (rule-based only).

Limitations:

- Works only with the provided demo dataset.
- Basic confidence thresholds (not tuned).
- Limited handling of noisy or missing invoice fields.

Recommended Next Steps:

1. Expand dataset with **real-world invoices and POs**.
2. Add **external retrieval/web agents** for incomplete invoice data.
3. Implement **ML-based adaptive planner**.
4. Enhance verifier with **anomaly detection**.
5. Integrate with **human-in-the-loop approval workflows**.

Process of Building:

1. Defined invoice–PO matching as the use-case.
2. Designed the pipeline (**Planner → Retrieval → Verifier → Synthesizer**).
3. Built a mock dataset (JSON invoices & POs).
4. Implemented both **CLI** and **Streamlit UI**.
5. Added **audit logging** for explainability.
6. Captured **annotated screenshots** to validate the system.

Key Learnings

- I learned how to design and implement an **Agentic Retrieval-Augmented Generation (RAG) pipeline** for financial document matching.
- I gained practical skills in **vector search, retrieval pipelines, and rule-based query planning**.
- I implemented **audit logging** to ensure traceability and explainability of AI workflows.
- I built and tested both a **Command Line Interface (CLI)** and a **Streamlit Web UI** for real-time interaction.
- I improved my ability to create **mock datasets** to simulate real-world invoice and PO scenarios.
- I developed stronger **problem-solving skills** by debugging pipeline errors and optimizing retrieval accuracy.
- I enhanced my **documentation and reporting skills**, ensuring clarity and professionalism in project submission.
- I understood the importance of **human-in-the-loop verification** in sensitive financial use-cases.

Conclusion / Reflection

- The project successfully demonstrated an Agentic Retrieval-Augmented Generation (RAG) pipeline for invoice–purchase order (PO) matching.
- It provided hands-on experience in building a structured AI workflow combining rule-based logic and vector search.
- I gained practical knowledge of query planning, evidence retrieval, verification, and response synthesis.
- The addition of an audit log highlighted the importance of traceability and explainability in AI systems.
- Implementing both CLI and Streamlit demos improved my understanding of creating user-friendly interfaces for technical systems.
- I learned the value of human-in-the-loop validation, especially in financial workflows where decisions have real-world impact.
- The project gave me exposure to end-to-end AI system design – from data preparation and pipeline building to visualization and documentation.
- Overall, this internship enhanced my skills in applied AI engineering and prepared me to contribute effectively to real-world AI solutions.