

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



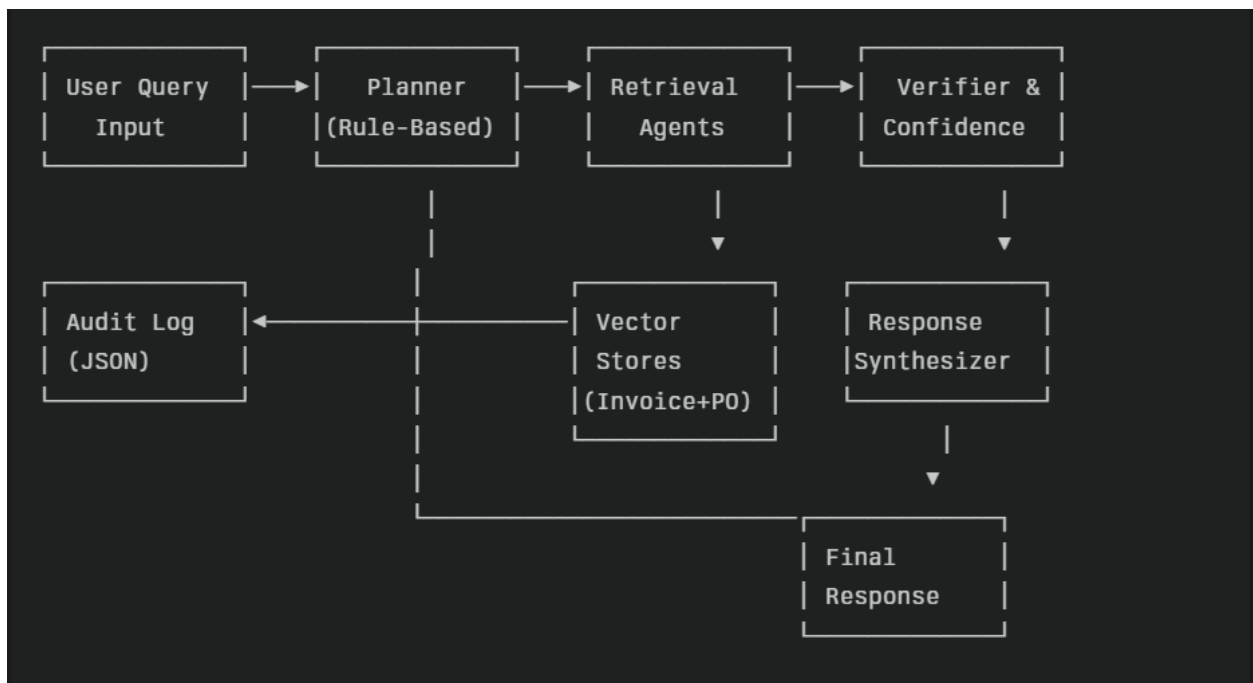
Retrieval Agents



Verifier & Audit Log



Response Synthesizer



Code & Setup / README Overview:

GitHub Repository: <https://github.com/ssk-2003/agentik-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
```

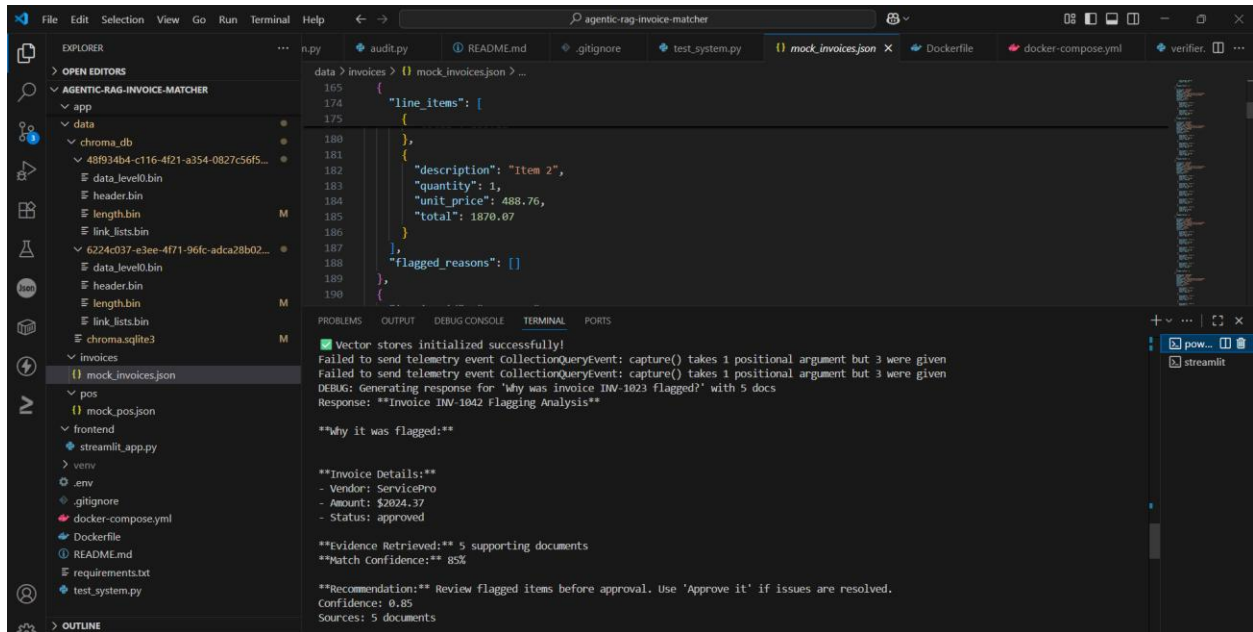
File/Folder	Purpose
app/agents/planner.py	Rule-based planner for user query interpretation
app/agents/rag_system.py	Orchestrates agents and produces answers
app/data/mock_invoices.py	Generates demo invoices & POs
app/data/vector_store.py	Builds vector search database
test_system.py	CLI testing/demo of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



The screenshot shows a VS Code editor with a project named 'agentic-rag-invoice-matcher'. The Explorer sidebar on the left shows a file tree with folders like 'app', 'data', 'invoices', 'pos', 'frontend', and 'venv'. The 'invoices' folder is expanded, showing 'mock_invoices.json'. The main editor displays the content of 'mock_invoices.json', which is a JSON array of invoice objects. The first object is for 'INV-1042' and is flagged. The terminal at the bottom shows the output of a query, including a debug message, the response, and a detailed analysis of why invoice INV-1042 was flagged.

```
data > invoices > {} mock_invoices.json > ...
165 {
174   "line_items": [
175     {
180       },
181     {
182       "description": "Item 2",
183       "quantity": 1,
184       "unit_price": 488.76,
185       "total": 1870.07
186     }
187   ],
188   "flagged_reasons": [
189     {
190       }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Vector stores initialized successfully!
Failed to send telemetry event collectionQueryEvent: capture() takes 1 positional argument but 3 were given
Failed to send telemetry event collectionQueryEvent: capture() takes 1 positional argument but 3 were given
DEBUG: Generating response for 'why was invoice INV-1023 flagged?' with 5 docs
Response: ****Invoice INV-1042 Flagging Analysis****

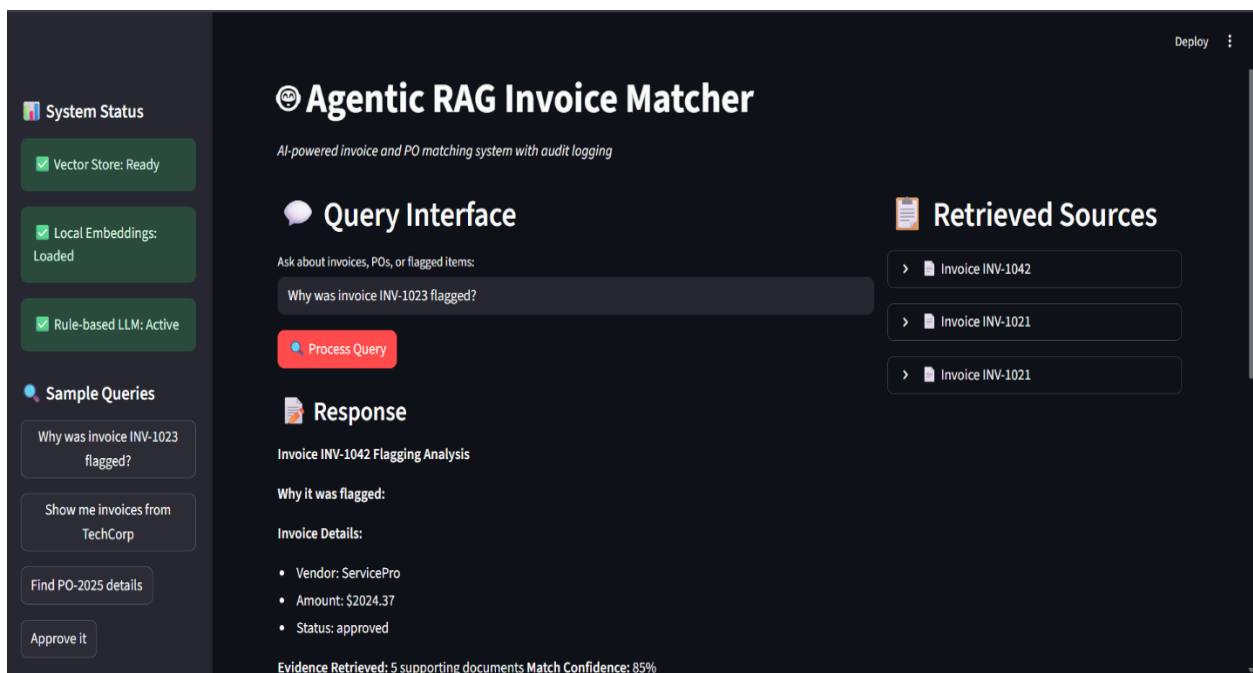
****why it was flagged:****

****Invoice Details:****

- Vendor: ServicePro
- Amount: \$2024.37
- Status: approved

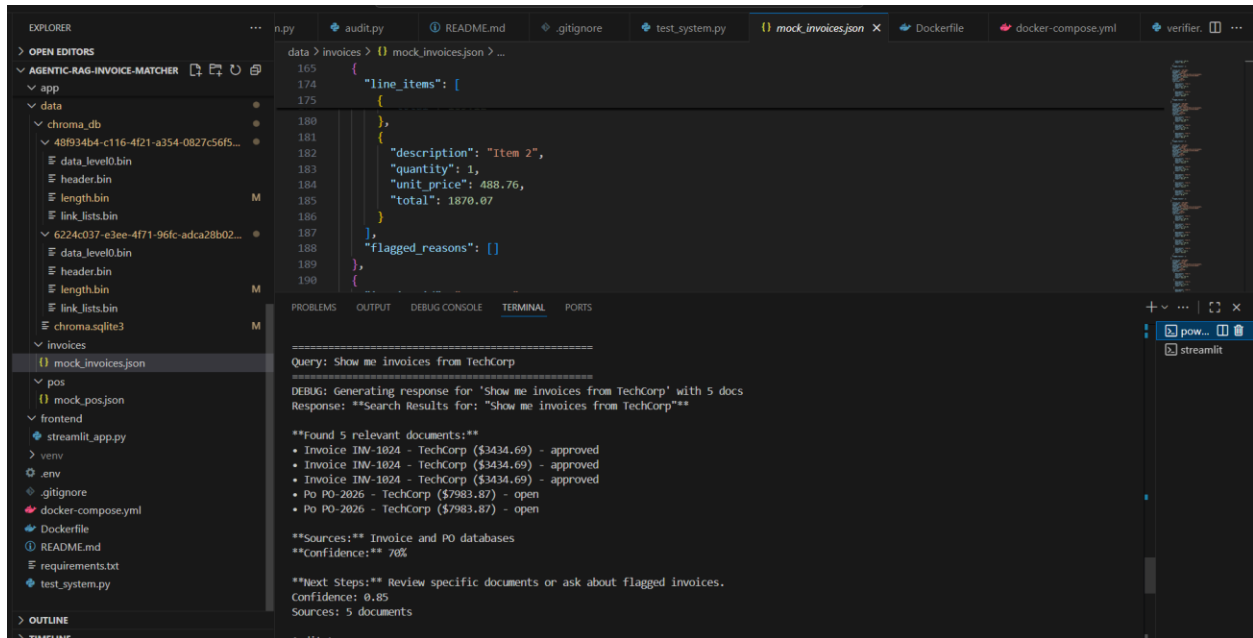
****Evidence Retrieved:**** 5 supporting documents
****Match Confidence:**** 85%

****Recommendation:**** Review flagged items before approval. Use 'Approve it' if issues are resolved.
Confidence: 0.85
Sources: 5 documents



Demo Screenshot 2

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



The screenshot shows a VS Code editor with a file explorer on the left and a terminal on the right. The file explorer shows a project structure with folders like 'app', 'data', 'invoices', and 'pos'. The 'invoices' folder is selected, showing a file named 'mock_invoices.json'. The terminal displays the output of a query: 'Query: Show me invoices from TechCorp'. The output includes a debug message, search results for 'Show me invoices from TechCorp', a list of 5 relevant documents (3 invoices and 2 purchase orders), sources, confidence, and next steps.

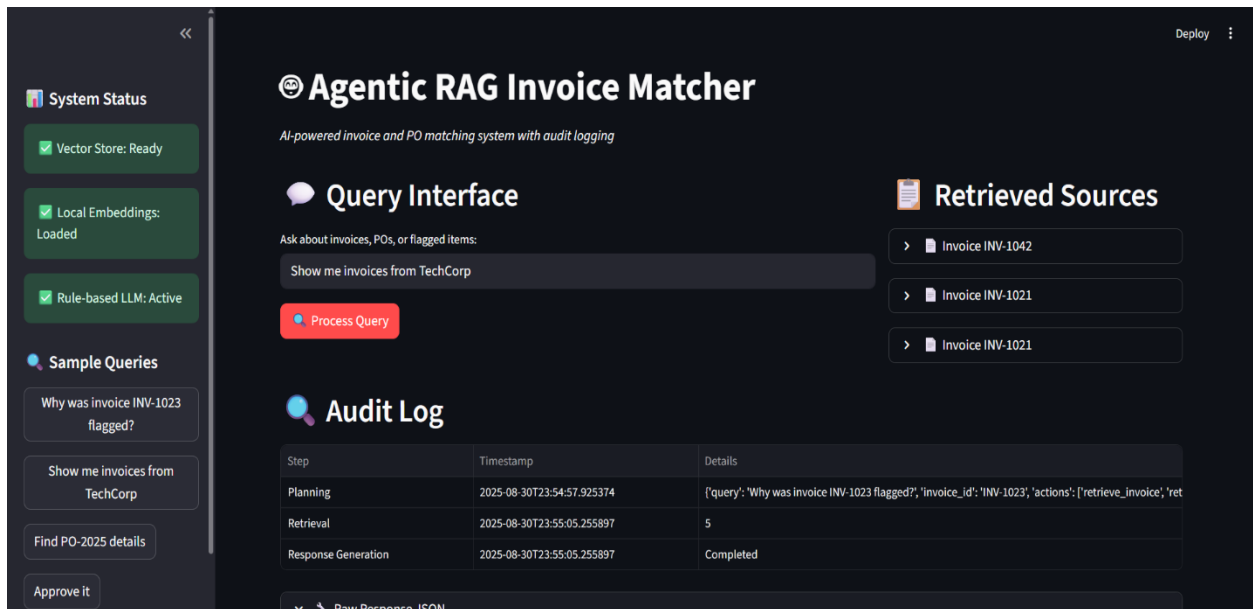
```
data > invoices > mock_invoices.json > ...
165 {
174   "line_items": [
175     {
180       },
181     {
182       "description": "Item 2",
183       "quantity": 1,
184       "unit_price": 488.76,
185       "total": 1870.07
186     }
187   ],
188   "flagged_reasons": []
189 },
190 {
```

```
Query: Show me invoices from TechCorp
=====
DEBUG: Generating response for 'Show me invoices from TechCorp' with 5 docs
Response: **Search Results for: "Show me invoices from TechCorp"**

**Found 5 relevant documents:**
• Invoice INV-1024 - TechCorp ($3434.69) - approved
• Invoice INV-1024 - TechCorp ($3434.69) - approved
• Invoice INV-1024 - TechCorp ($3434.69) - approved
• PO PO-2026 - TechCorp ($7983.87) - open
• PO PO-2026 - TechCorp ($7983.87) - open

**Sources:** Invoice and PO databases
**Confidence:** 70%

**Next Steps:** Review specific documents or ask about flagged invoices.
Confidence: 0.85
Sources: 5 documents
```



The screenshot shows the Agentic RAG Invoice Matcher web application. The interface includes a sidebar with system status, sample queries, and a main content area with a query interface, retrieved sources, and an audit log.

System Status

- Vector Store: Ready
- Local Embeddings: Loaded
- Rule-based LLM: Active

Sample Queries

- Why was invoice INV-1023 flagged?
- Show me invoices from TechCorp
- Find PO-2025 details
- Approve it

Agentic RAG Invoice Matcher

AI-powered invoice and PO matching system with audit logging

Query Interface

Ask about invoices, POs, or flagged items:

Show me invoices from TechCorp

Process Query

Retrieved Sources

- Invoice INV-1042
- Invoice INV-1021
- Invoice INV-1021

Audit Log

Step	Timestamp	Details
Planning	2025-08-30T23:54:57.925374	[{"query": "Why was invoice INV-1023 flagged?", "invoice_id": "INV-1023", "actions": ["retrieve_invoice", "ret"]
Retrieval	2025-08-30T23:55:05.255897	5
Response Generation	2025-08-30T23:55:05.255897	Completed

Raw Response: JSON

Demo Screenshot 3

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

The screenshot shows a VS Code editor with the file explorer on the left displaying the project structure. The main editor window shows the 'mock_invoices.json' file with the following content:

```
165 {
174   "line_items": [
175     {
180       "description": "Item 2",
181       "quantity": 1,
182       "unit_price": 488.76,
183       "total": 1870.07
184     },
185     {
186       "description": "Item 3",
187       "quantity": 1,
188       "unit_price": 488.76,
189       "total": 1870.07
190     }
191   ],
192   "flagged_reasons": []
193 }
```

The terminal window at the bottom shows the output of the 'Approve it' query:

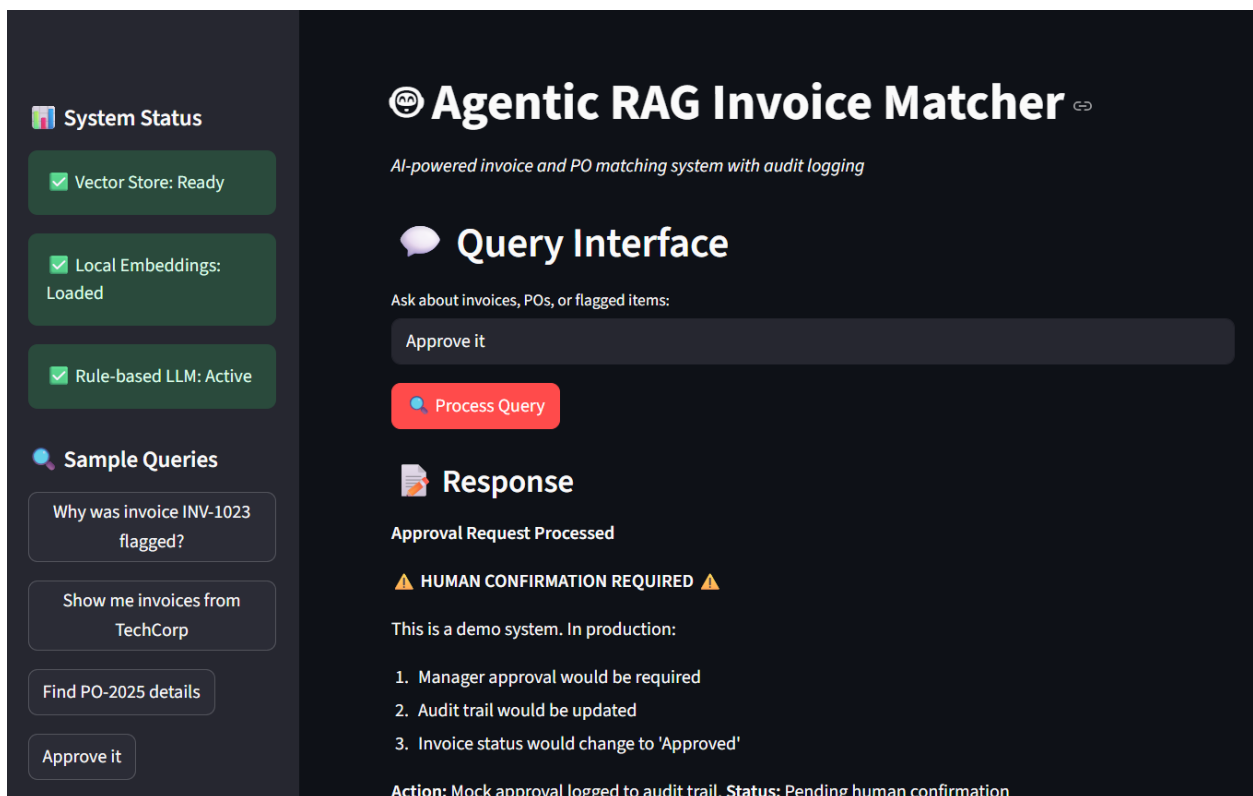
```
Query: Approve it
=====
DEBUG: Generating response for 'Approve it' with 0 docs
Response: **Approval Request Processed**

⚠️**HUMAN CONFIRMATION REQUIRED**⚠️

This is a demo system. In production:
1. Manager approval would be required
2. Audit trail would be updated
3. Invoice status would change to 'Approved'

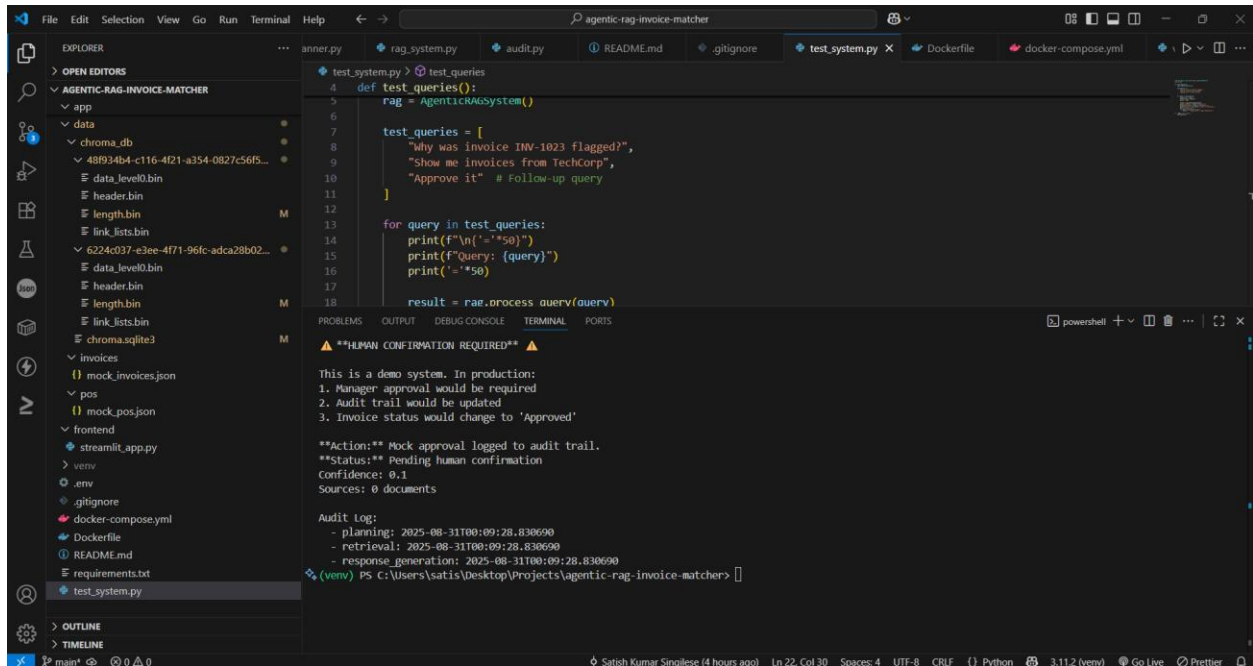
**Action:** Mock approval logged to audit trail.
**Status:** Pending human confirmation
Confidence: 0.1
Sources: 0 documents

Audit Log:
- planning: 2025-08-31T00:09:28.830690
- retrieval: 2025-08-31T00:09:28.830690
```



Demo Screenshot 4

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



The screenshot shows a VS Code editor with a file explorer on the left and a terminal on the right. The file explorer shows a project named 'AGENTIC-RAG-INVOICE-MATCHER' with various files and folders. The terminal shows the output of a Python script named 'test_system.py'. The script defines a function 'test_queries()' which calls 'rag = AgenticRAGSystem()' and then processes a list of test queries. The output in the terminal shows a confirmation message, a list of production requirements, and an audit log with timestamps for planning, retrieval, and response generation.

```
def test_queries():
    rag = AgenticRAGSystem()

    test_queries = [
        "why was invoice INV-1023 flagged?",
        "Show me invoices from TechCorp",
        "Approve it" # Follow-up query
    ]

    for query in test_queries:
        print(f"\n{'='*50}")
        print(f"Query: {query}")
        print(f"{'='*50}")

    result = rag.process_query(query)
```

```
***HUMAN CONFIRMATION REQUIRED***

This is a demo system. In production:
1. Manager approval would be required
2. Audit trail would be updated
3. Invoice status would change to 'Approved'

**Action:** Mock approval logged to audit trail.
**Status:** Pending human confirmation
Confidence: 0.1
Sources: 0 documents

Audit log:
- planning: 2025-08-31T00:09:28.830690
- retrieval: 2025-08-31T00:09:28.830690
- response_generation: 2025-08-31T00:09:28.830690
```

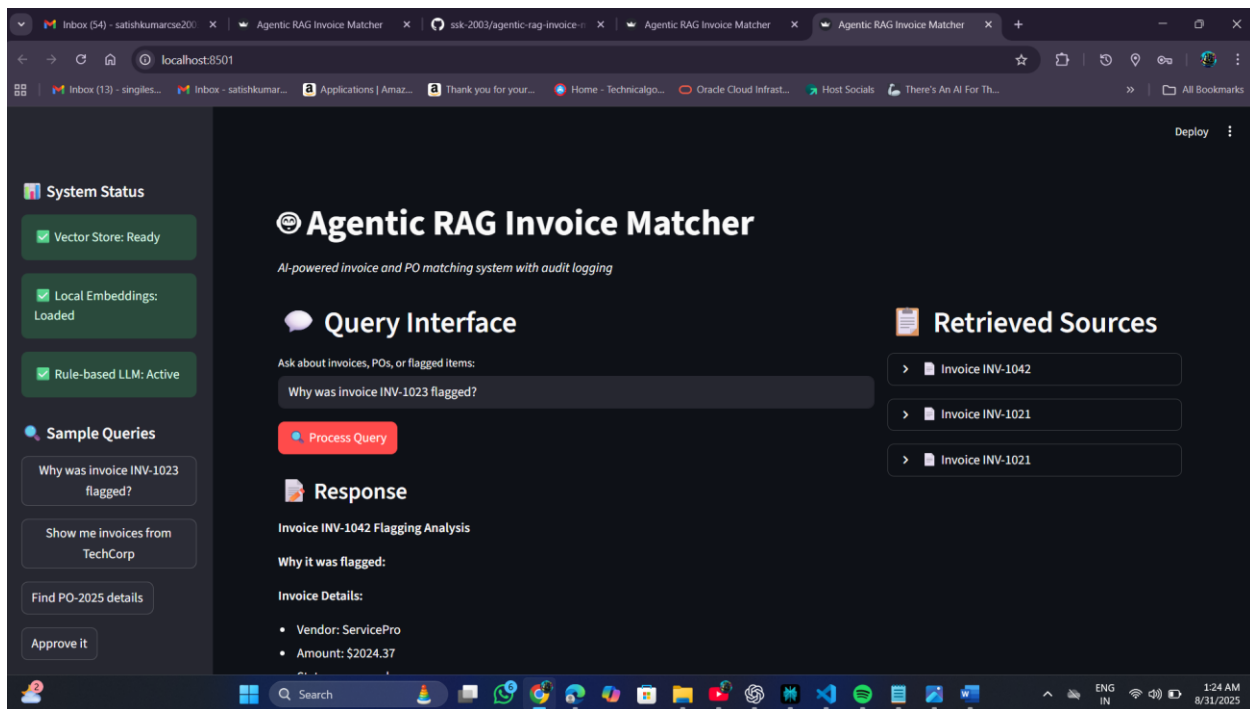
```
{"step": "planning", "timestamp": "2025-08-31T01:04:15.123456", "input": "why was invoice INV-1023 flagged?"}
{"step": "retrieval", "timestamp": "2025-08-31T01:04:20.654321", "sources": ["INV-1023", "PO-2045"]}
{"step": "response_generation", "timestamp": "2025-08-31T01:04:20.660000", "response_length":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