Role-Aware Financial Insights Assistant

An Agentic RAG System with Role-Based Access Control

Project Links

- **GitHub Repository:** https://github.com/Mohib1402/role-aware-financial-assistant
- **Hugging Face Demo:** https://huggingface.co/spaces/mohibkhan949/role-aware-financial-assistant
- **Video Presentation:** https://youtu.be/JLQCUWa6SnM
- **PowerPoint Deck:** View Slides

Team & Contributions

- **Aishly Manglani:** LangGraph workflow design, ReAct pattern implementation, agent architecture.
- **Syeda Nida Khader:** Role-Based Access Control (RBAC) logic, retriever filtering, security design.
- **MohibKhan Pathan:** GPT-4o-mini integration, Vision RAG module, Hugging Face deployment.
- **Siri Batchu:** UI testing, sample data review.
- **All Team Members:** Documentation, testing, and project report.

Abstract

This project presents a Role-Aware Financial Insights Assistant that combines Retrieval-Augmented Generation (RAG) with Role-Based Access Control (RBAC). Built with LangChain, LangGraph, and GPT-4o-mini, the system delivers personalized financial analysis while enforcing strict data sensitivity boundaries. The assistant supports three distinct roles: Analyst, Product Manager, and Executive, each with specific data access permissions. The architecture includes guardrails for PII detection, hallucination prevention, audit logging, and a Vision RAG module for chart analysis. The system achieves 100% accuracy on RBAC verification tests and is currently deployed on Hugging Face Spaces.

Keywords: Agentic RAG, RBAC, LangGraph, GPT-4o-mini, Financial AI, Guardrails.

1. Introduction

Problem Statement

Financial organizations need AI-powered insights to improve efficiency but must maintain strict data access controls. The core challenges include:

- **Data Sensitivity:** Financial data exists at multiple levels, including public, internal product data, and insider information.
- **Role-Based Access:** Different organizational roles require different data permissions.
- **Compliance:** All system access must be auditable to meet regulatory requirements.
- **AI Safety:** Responses must be guarded against Personally Identifiable Information (PII) leakage and hallucination.

Importance

Solving this problem is critical for enterprise adoption. Regulatory compliance standards such as SOX, SEC, and GDPR require robust audit trails. Improper access to insider information can create significant legal liability. Consequently, security concerns remain the primary barrier to enterprise AI adoption in the financial sector.

Results Overview

- **RBAC Accuracy:** 100% (23/23 tests passed)
- **PII Detection:** 100% effectiveness
- **Response Time:** Under 3 seconds per query
- **Deployment:** Live on Hugging Face Spaces

2. Related Work

Our work builds upon several foundational AI methodologies:

1. **RAG (Lewis et al., 2020):** Combines retrieval with generation. Our work extends standard RAG by implementing role-based document filtering prior to generation.
2. **Agentic AI (LangGraph):** We utilize stateful graph-based workflows with conditional routing to manage complex user interactions.
3. **ReAct Pattern (Yao et al., 2022):** Synergizing Reasoning and Acting for tool use. We implement this specifically for financial calculations.

Comparison with Standard Systems

Feature	**Standard RAG**	**Our System**
Document Retrieval	✓	✓
Role-Based Filtering	✗	✓
Dynamic Response Style	✗	✓
Tool Use	✗	✓
PII Guardrails	✗	✓
Vision/Multimodal	✗	✓

3. Data

Dataset Overview

We curated a specific dataset representing mixed financial data sensitivities.

Sensitivity	Documents	Content Examples	Access Level
Public	4	10-Q filings, press releases	All roles
Product	3	Roadmaps, feature plans	PM, Executive
Insider	4	Confidential memos	Executive only
Total	**11**		

Preprocessing

The data pipeline involved the following steps:

1. **Curation:** 11 documents were created containing realistic financial content.
2. **Tagging:** Each document was tagged with metadata indicating its sensitivity level (public, product, or insider).
3. **Embedding:** Documents were embedded using the Hugging Face all-MiniLM-L6-v2 model (384 dimensions).
4. **Indexing:** The embeddings were indexed in a FAISS vector store for efficient retrieval.

4. Methods

System Components

1. Role-Based Access Control (RBAC)

We implemented a 3-tier system: Analyst (public access only), Product Manager (public and product access), and Executive (all access). This is enforced via post-retrieval filtering where documents are checked against the user's role before context construction.

2. LangGraph Workflow

The application uses a stateful graph with specific nodes for Retrieve, Generate, and Tool Executor. It employs conditional routing to implement the ReAct pattern. The state tracks messages, user role, context, and guardrail triggers.

3. ReAct Calculator Tool

This component detects calculation requests within user queries. It executes Python code safely in a sandboxed environment and returns the results for the LLM to incorporate into the final answer.

4. Dynamic Prompting

Prompts are dynamically adjusted based on the user's role:

- **Executive:** Concise bullet points, focus on risks.
- **Product Manager:** Timeline-focused, feature impact analysis.
- **Analyst:** Detailed analysis with specific source citations.

5. Guardrails

- **PII Detection:** Regex patterns identify SSN, credit card, email, and phone formats.
- **Hallucination Prevention:** The system blocks confident answers if the retrieved context is insufficient.
- All violations are logged and the response is blocked.

6. Vision RAG

We integrated GPT-4o-mini Vision to analyze financial charts. The module extracts values, trends, and specific insights from visual data.

Technology Stack

- **LLM:** GPT-4o-mini
- **Embeddings:** all-MiniLM-L6-v2
- **Vector Store:** FAISS
- **Workflow:** LangGraph
- **UI:** Gradio
- **Vision:** GPT-4o-mini Vision

5. Experiments and Results

Test Suite Summary

We performed 23 automated tests to verify system integrity.

Category	**Tests**	**Passed**	**Rate**
RBAC Access Control	5	5	100%
Retriever Filtering	3	3	100%
Guardrails	6	6	100%
Calculator Tool	3	3	100%
Dynamic Prompts	2	2	100%
Audit Logging	2	2	100%

Vision Module	2	2	100%
Total	**23**	**23**	**100%**

RBAC Test Scenarios

Query	**Role**	**Expected Behavior**	**Result**
"Project Blackwell status?"	Analyst	No access (Blocked)	✓ Pass
"Project Blackwell status?"	Executive	Full details provided	✓ Pass
"Product roadmap?"	PM	Roadmap info provided	✓ Pass
"Q3 revenue?"	Analyst	Public data provided	✓ Pass

Performance Metrics

- **Average Response Time:** 2.3 seconds
- **Vector Search Latency:** < 50ms
- **Cost per Query:** ~$0.00045
- **Monthly Cost (est. 1K queries):** ~$0.45

6. MLOps Artifacts

Project Structure

The repository follows standard engineering practices:

- `app.py`: Hugging Face Spaces entry point.
- `main.py`: CLI entry point.
- `requirements.txt`: Pinned dependencies for reproducibility.
- `src/`: Core modules (config, data, retriever, guardrails, agent, ui, vision).
- `tests/`: Suite of 23 unit tests.
- `audit_log.jsonl`: Immutable log for compliance.

Deployment Pipeline

The system utilizes a CI/CD-like flow: Local Development → GitHub Repository → Hugging Face Spaces (utilizing Secrets Management for API keys).

Audit Logging

To ensure compliance, every query generates a log entry containing the timestamp, user role, query content, documents accessed, sensitivity levels, response length, and guardrail status.

7. Key Metrics Visualization

RBAC Access Matrix

The following matrix illustrates the permission structure enforced by the model.

Role	**Public Data**	**Product Data**	**Insider Data**
Analyst	✓	✗	✗
Product Manager	✓	✓	✗
Executive	✓	✓	✓

Response Characteristics by Role

We measured the output style to ensure dynamic prompting was effective.

Role	Avg Length	Style Description
Executive	~150 tokens	Concise bullets
Product Manager	~250 tokens	Timeline-focused
Analyst	~350 tokens	Detailed citations

8. Conclusion

Key Achievements

We successfully delivered a secure financial assistant with:

1. 100% RBAC enforcement accuracy.
2. Comprehensive PII and hallucination guardrails.
3. Role-appropriate dynamic responses.
4. A functional Calculator tool via the ReAct pattern.
5. Multimodal chart analysis capabilities.
6. Full audit logging for compliance.

Future Work

Future iterations will focus on:

- Multi-turn conversation memory.
- User authentication system integration (SSO).
- Additional tools (web search, SQL database querying).
- Domain-specific fine-tuned embeddings.

References

1. Lewis, P., et al. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *NeurIPS*.
2. Yao, S., et al. (2022). "ReAct: Synergizing Reasoning and Acting in Language Models." *ICLR*.
3. LangChain Documentation. python.langchain.com
4. OpenAI API Documentation. platform.openai.com/docs

Supplementary Materials

- **Source Code:** GitHub Repository
- **Interactive Demo:** Hugging Face Space

Acknowledgments

We thank OpenAI, LangChain, and Hugging Face for the tools and infrastructure that made this project possible.

- **Gemini (Google):** Document framing and formatting assistance.