Post-Mortem Report – Crypto Detective (RAG-based Fraud Investigation System)

Table of Contents

1. Overview

Executive Summary: Crypto Detective is a time-boxed AI system that leverages Retrieval-Augmented Generation (RAG) to assist investigators in analyzing and reporting on cryptocurrency fraud. The system uses OpenAI embeddings, Qdrant vector search, GPT-based reranking, and structured report generation, all wrapped in a Gradio user interface and deployed with modern DevOps practices. This report summarizes development outcomes, challenges encountered, and key learnings from the implementation.

This post-mortem report outlines the development, challenges, and outcomes of building **Crypto Detective**, a Retrieval-Augmented Generation (RAG) system designed to assist in cryptocurrency fraud investigations. Built within a 3.5-hour time constraint, the system integrates:

- OpenAI embeddings for semantic search
- Qdrant vector database for document storage and retrieval
- LLM-based reranking and report generation (GPT-4)
- Query routing and filtering
- Gradio UI for human-friendly interaction
- S3 integration for report storage and sharing

The system processes case files, retrieves relevant evidence, ranks results based on vector and LLM signals, and generates a structured report for investigators.

---

2. Configuration Overview

Config Key	Value
open_api_key	(hidden)
embedding_model	text-embedding-ada-002
gpt_model	gpt-4o
tokenizer	cl100k_base
chunk_size	512
chunk_overlap	50
top_k / top_k_retrieval	10
top_rerank	10
strategy	multi-step
data_dir	./data/
rerank_weight_vector	0.45
rerank_weight_llm	0.55
filter_enabled	true
qdrant_host:port	localhost:6333
qdrant_collection	crypto_case_vectors
embedding_dim	1536
aws_region	ap-southeast-1
s3_bucket	mbg-devops-test

3. Timeline

Timestamp	Event Description
2025-03-19 09:00	Project initialization and planning
2025-03-19 09:30	Qdrant container running via Docker
2025-03-19 10:00	Embedding pipeline created, chunking added
2025-03-19 10:40	Retrieval module + multi-step search complete
2025-03-19 11:20	LLM-based reranker + scoring integrated
2025-03-19 12:00	Prompt-based report generation added
2025-03-19 12:20	S3 report upload + Gradio UI integrated
2025-03-19 13:00	Final testing, logs, and README cleanup

---

4. Metrics

Metric	Value
Total time spent	~3.5 hours
Avg. query → report time	~8.3 seconds
Documents processed	8 base files → ~80 chunks
Retrieval Top-K	10
Reranked documents (LLM)	Top 10

Metric	Value
Report generation success rate	100%
S3 upload reliability	100% (presigned)

---

5. Detection

Multiple issues were encountered during testing and pipeline validation:

- Vector retrieval returned misleading results for abstract queries
- LLM reranker produced non-numeric outputs without fallback
- Truncated GPT responses due to token limits
- Redundant embedding calls when re-running document load

---

6. Resolution

- Improved retrieval precision by switching to **multi-step** query expansion using GPT-generated sub-queries
- Introduced normalization and fallback in reranker for invalid LLM scores
- Increased token limit in report generator (max_tokens: 2000)
- Added logging and de-duplication to `Embedding.process()`

---

7. Impact

- Misleading evidence in early runs caused irrelevant sections in generated reports
- Increased cost from unnecessary re-embedding (no hashing/caching yet)
- Time lost debugging Qdrant document schema mismatch
- Minor UI inconsistencies when loading large reports in Gradio

---

8. Root Cause Analysis

- **Prompt engineering gaps**: The initial reranker prompt was too vague lacked clear scoring rules
- **No embedding cache**: Each run re-processed unchanged files, leading to API overuse
- **Async handling missing**: All OpenAI/Qdrant/S3 calls are synchronous → higher latency
- **Chunking misalignment**: Tokenization logic caused content splits mid-sentence

- **Lack of edge case tests**: Long queries, irrelevant queries, or malformed JSON caused failures

---

9. Learnings

- One key limitation encountered was **time management under the 3.5-hour constraint**. The orchestration of tasks (retrieval tuning, reranking prompt design, UI wiring, and S3 integration) led to some rushed decisions near the final hour. Better planning or mock runs beforehand could have led to smoother submission.
- LLM prompt design is **mission-critical** for multi-step retrieval and reranking
- Must treat OpenAI as a limited resource — caching and token planning required
- Retrieval quality depends as much on **filtering** and **chunking** as it does on vector models
- S3 integration should always include error fallback and retry logic
- Time-boxed projects benefit from modular structure (retriever, reranker, generator, ui)

---

10. Actions

Corrective Actions

- Refined prompts with clearer scoring criteria and JSON output formatting
- Added truncation guards and numeric fallback to LLM reranker
- Increased chunk overlap and max chunk size for better sentence preservation

Preventive Actions

- Plan to add SHA-based hashing for chunk caching in embedding phase
- Switch to async FastAPI routes and `httpx` for parallel processing
- Build unit tests for each pipeline stage, especially reranking and S3 upload

Strategic Actions

- Modularize API endpoints to support future fraud types (NFT scams, wallet phishing, etc.)
- Add user authentication and logging to support production use
- Track usage metrics (query types, latency, errors) for continuous learning

- **Implement CI/CD pipeline using Jenkins**: GitHub commits will trigger Jenkins jobs that build and test the application, then push updated Docker images to Docker Hub.
- **Use Docker and Kubernetes on GCP**: The system will be containerized and deployed to GKE (Google Kubernetes Engine) with Helm charts and managed via Nginx ingress for routing and TLS support.
- Modularize API endpoints to support future fraud types (NFT scams, wallet phishing, etc.)
- Add user authentication and logging to support production use
- Track usage metrics (query types, latency, errors) for continuous learning

---

11. Conclusion

The Crypto Detective system demonstrates a robust and modular architecture capable of handling real-world investigative queries through advanced LLM techniques. Despite time constraints, the system successfully integrates retrieval, reranking, reporting, and cloud-based delivery. With further enhancements in CI/CD and scalability, it has the potential to be extended for broader fraud detection scenarios.

Next Steps

- Add streaming support for long documents and real-time report generation
- Integrate authentication and usage-based rate-limiting
- Monitor performance and logging with tools like Grafana + Prometheus

---

12. Acknowledgement

I would like to sincerely thank **Mighty Bear Games** for designing this thoughtful and technically challenging assignment. It was a great opportunity to demonstrate my skills across full-stack AI engineering, prompt design, system orchestration, and applied retrieval techniques.

I truly enjoyed working on this project and exploring how AI can be used to assist with real-world crypto investigations. I hope to have the chance to collaborate with your talented team and contribute to future innovations at Mighty Bear Games.

Author: Truong Minh Dat
Date: 2025-03-20
Project: Crypto Detective – AI RAG for Cybercrime Investigations