Regulatory and Compliance Automation

Technical Report

Challenge CH-04: Regulatory and Compliance Automation Using **RAG Ai Agent**

Submission Date: February 07, 20

1. Executive Summary

This technical report documents the implementation of an automated regulatory compliance system designed for Challenge CH-04. The solution leverages Retrieval-Augmented Generation (RAG) architecture to analyze financial transactions against a comprehensive rule set covering Anti-Money Laundering (AML), Know Your Customer (KYC), Sanctions Screening, and Transaction Monitoring requirements.

The system processes 25 regulatory rules and successfully analyzes 20 test cases with an average processing time of 15-80 seconds per case. The solution operates entirely offline using local language models, ensuring data privacy and regulatory compliance.

2. System Architecture

2.1 High-Level Architecture

The system consists of two primary workflows implemented in n8n: the Indexing Workflow for rule storage and the Query Workflow for compliance analysis. Both workflows integrate seamlessly through a shared Pinecone vector database.

Component	Technology	Purpose
Orchestration Platform	n8n	Workflow automation and visual logic design
Vector Database	Pinecone	Semantic search across dual knowledge bases
Embedding Model	bge-m3 (Ollama)	Text vectorization (1024 dimensions)
Chat Model (Policies)	qwen2.5:3b (Ollama)	Local policy analysis
Chat Model (Regulations)	qwen2.5:3b (Ollama)	Local regulation analysis
Primary AI Model	Groq (gpt-oss-120b)	Main compliance decision engine
User Interface	Chat Trigger	Web-based case submission interface

2.2 Workflow Description

Indexing Workflow (VD): Retrieves rules from Google Drive, generates embeddings using **bge-m3**, and stores vectors in Pinecone with appropriate namespaces for efficient retrieval.

Query Workflow (RAG ChatBot): Accepts case data via chat interface, retrieves relevant rules through semantic search, analyzes compliance using the **qGroqwen3:4b model**, and generates structured reports.

3. Approach and Methodology

3.1 RAG-Based Compliance Engine

The solution employs Retrieval-Augmented Generation to combine the benefits of semantic search with large language model reasoning. This approach addresses three core requirements:

Automated Compliance Checks: The system retrieves the top 10 most relevant rules using cosine similarity between case embeddings and rule embeddings, ensuring comprehensive coverage without manual intervention.

Structured Evidence Extraction: Each violation includes specific field references, actual values, threshold comparisons, and percentage calculations, providing clear audit trails.

Audit Trail Generation: Every decision is accompanied by rule IDs, severity classifications, confidence scores, and actionable recommendations with time frames.

3.2 Multi-Stage Processing Pipeline

The analysis pipeline consists of five distinct stages to ensure accuracy and completeness:

Stage	Operation	Output
1. Ingestion	Parse case data from JSON	Structured case object
2. Embedding	Generate 1024-dim vector (bge-m3)	Case embedding vector
3. Retrieval	Query Pinecone with top-k=10	Relevant rule set
4. Analysis	LLM evaluation against rules	Compliance assessment
5. Reporting	Format structured output	Human-readable report

4. Design Choices

4.1 Technology Selection Rationale

n8n Workflow Platform:

Selected for visual workflow design, extensive integration capabilities, and rapid prototyping. **Enables non-developers to understand and modify the system logic.**

Ollama for Local LLMs:
Chosen for offline operation capability, data privacy compliance, and **zero API costs**. The qwen3:4b model provides optimal balance between performance and resource usage.

Groq (gpt-oss-120b):
Free and best model for this case

bge-m3 Embeddings:
Selected for multilingual support (important for Arabic/English cases), high-quality semantic representations, and efficient 1024-dimensional vectors.

Pinecone Vector Database:
Chosen for sub-second query times, managed infrastructure, and namespace-based organization enabling multi-tenant rule management.

5. Implementation Details

5.1 Data Flow Architecture

Input data flows from Google Drive through file download, JSON parsing, and field extraction. The Edit Fields node dynamically determines the namespace based on case characteristics. Cases proceed to the AI Agent which orchestrates retrieval and analysis. The vector store tool queries Pinecone, returning the top-k most relevant rules. The LLM analyzes case fields against retrieved rules, calculating risk scores and generating structured outputs.

5.2 Risk Scoring Algorithm

Risk scores are calculated using an additive model: Base score of 0 for compliant cases. Each violation adds points based on severity: Critical (+40), High (+25), Medium (+15), Low (+5). The total is multiplied by the confidence factor and capped at 100. This approach ensures that multiple medium violations can equal one critical violation in terms of overall risk.

5.3 Namespace Management

Rules are stored in Pinecone with namespaces matching their source filenames (e.g., 'rules.json'). The retrieval workflow uses dynamic namespace resolution through the Edit Fields1 node, which examines case content to determine the appropriate namespace. This enables efficient multi-tenant operation and rule versioning.

5.4 System Prompt Engineering

The AI Agent employs a carefully crafted system prompt that defines behavioral rules, output formatting requirements, severity classification criteria, and language handling. The prompt instructs the model to respond in the user's language, maintain professional tone, cite specific rule IDs, and provide actionable recommendations with time frames.

6. Limitations and Constraints

6.1 Technical Limitations

Model Capacity:

- The qwen3:4b model has a limited context window (8,192 tokens). Extremely complex cases with extensive free text may require truncation.
- Groq model supports 8,192 token contexts. Cases with extensive free-text descriptions may require truncation. Mitigation: Implement automatic summarization for descriptions exceeding 500 words before embedding.

Embedding Dimensionality: bge-m3 generates 1024-dimensional vectors. While sufficient for the current rule set, larger knowledge bases may benefit from higher-dimensional representations.

Real-time Processing: Current implementation processes one case at a time. Batch processing of hundreds of cases simultaneously would require workflow modifications.

Numerical Reasoning: LLM-based systems may occasionally struggle with precise numerical calculations. Critical thresholds use rule-based checks as fallback.

6.2 Operational Constraints

Network Dependency: While Ollama runs locally, Pinecone requires internet connectivity. Fully offline operation would require migration to FAISS or Qdrant.

Rule Updates: Adding new rules requires re-running the indexing workflow. Real-time rule updates would need incremental indexing capabilities.

Multilingual Support: Currently optimized for English and Arabic. Additional languages would require model fine-tuning or multilingual prompt engineering.

7. Failure Modes and Mitigation

Failure Mode	Impact	Mitigation Strategy
Response time	Taking more time	Divide Complex tasks into subtasks.
Empty vector retrieval	No rules found for analysis	Namespace verification, fallback to broader search
LLM hallucination	Fabricated rule violations	Strict prompt engineering, rule ID validation
Embedding service failure	Cannot process new cases	Retry logic, fallback to cached embeddings
JSON parsing errors	Case ingestion failure	Schema validation, error logging, graceful degradation
Pinecone rate limiting	Temporary service disruption	Request queuing, exponential backoff
Namespace mismatch	Incorrect rule retrieval	Explicit namespace configuration, validation checks

7.1 Handling Implementation

The system implements multi-level error handling: Input validation catches malformed JSON before processing. Retrieval failures trigger automatic retry with exponential backoff. LLM timeouts result in graceful degradation with partial results. All errors are logged with timestamps and context for debugging.

Conclusion

This solution successfully addresses all core requirements of Challenge CH-04 through a robust RAG-based architecture. The system automates compliance checks, extracts structured evidence, and maintains comprehensive audit trails. With documented limitations and clear mitigation strategies, the implementation provides a solid foundation for production deployment in financial compliance environments.

The modular design using n8n enables rapid iteration and customization, while local LLMs ensure data privacy and regulatory compliance. Performance characteristics meet operational requirements, and identified enhancements provide a clear roadmap for future development.