AI Engineer Hiring Task: Contextual AI Presentation Orchestrator

Issue Date : 16/08/2025 **Submission Date : 22/08/2025**

---

1. Introduction & Business Context

1.1. Executive Summary

This technical assessment is designed to evaluate your ability to architect and implement a sophisticated AI system that addresses real-world challenges in automated content generation. You are tasked with building a **"Contextual AI Presentation Orchestrator,"** a system that demonstrates mastery of advanced AI engineering concepts including **LangChain orchestration, RAG systems, context management, multi-model** optimization, and enterprise security patterns. The final system must be capable of generating comprehensive, factually-grounded presentations of 15-20 slides from complex source materials while maintaining contextual integrity across iterative generation cycles. This task simulates the exact type of challenges our engineering team tackles daily.

1.2. Business Context & Problem Statement

Our company operates in the enterprise AI automation space, developing sophisticated tools that transform dense, unstructured data sources into actionable business intelligence. Our clients, which include Fortune 500 companies and research institutions, require the automated generation of high-quality presentations from technical documents, research papers, and regulatory filings.

Current market solutions suffer from critical limitations such as context degradation, content hallucination, scalability constraints, and inadequate security. Your solution must demonstrate how to overcome these gaps by delivering a system characterized by factual accuracy, contextual coherence, enterprise-grade security, and operational excellence.

2. Task Requirements & Technical Specifications

You are required to design, build, and document a production-ready AI system based on the following detailed requirements.

2.1. Functional Requirements

- **FR-1: Advanced LangChain Orchestrator:** Implement a multi-agent LangChain system with specialized agents for document analysis, outline generation, content expansion, quality assurance, and format optimization.
- **FR-2: Enterprise-Grade RAG Architecture:** Build a production-ready RAG pipeline supporting multiple document formats (PDF, DOCX, TXT, MD) with intelligent segmentation, hierarchical chunking, and hybrid search.

- **FR-3: Advanced Context Management:** Ensure the system maintains perfect contextual coherence across multiple generation cycles using persistent session state and incremental context building.
- **FR-4: Role-Based Access Control (RBAC):** Implement a four-level role hierarchy (Executive, Senior Manager, Analyst, Junior Staff) with distinct permissions and content access restrictions.
- **FR-5: Multi-Model Optimization:** Integrate and optimize multiple models, including a primary LLM, a specialized model for technical content, and an embedding model.
- **FR-6: Advanced Query Interface:** Develop a primary RESTful API (OpenAPI 3.0) and a secondary CLI to support complex, multi-turn conversational queries.
- **FR-7: Production-Ready Monitoring:** Implement a comprehensive monitoring and observability stack, including structured logging, performance metrics, error tracking, and health checks.

2.2. Technical Specifications

- **TS-1: LangChain Implementation:** Utilize `AgentExecutor` with custom tools and memory systems. Implement dynamic agent routing and integrate with an open-source LLM (e.g., Llama 3, Mistral) via Ollama or Hugging Face.
- **TS-2: RAG Pipeline:** Implement custom embedding strategies, use vector search (FAISS/ChromaDB) with metadata filtering, and build a preprocessing pipeline with entity extraction. Include features for automatic source citation and contradiction detection.
- **TS-3: Context & Memory:** Implement sophisticated LangChain memory systems (`ConversationBufferWindowMemory`, `ConversationSummaryMemory`, custom vector-based memory). Develop a multi-phase batch generation strategy.
- **TS-4: Security:** Use JWT-based authentication. Build content filtering middleware, comprehensive audit logging, and data masking for sensitive information.
- **TS-5: Model Management:** Create a model abstraction layer, an intelligent routing system, and a multi-level caching strategy. Implement a performance monitoring dashboard.
- **TS-6: API & CLI:** Design comprehensive API endpoints for all core functionalities. The CLI must support interactive modes and batch processing.
- **TS-7: Observability:** Establish a benchmarking suite with standardized test scenarios, performance baselines, and load testing capabilities.

3. Technical Guidelines

3.1. Programming & Architecture

- **Primary Language:** Python
- **Frameworks:** LangChain , FastAPI.
- **Databases:** FAISS or ChromaDB (Vector), PostgreSQL or SQLite (Storage).
- **Architecture:** Modular, microservices-ready design following SOLID principles.
- **Testing:** pytest with >80% code coverage.

3.2. Infrastructure & Deployment

- **Local Development:** Full containerization via Docker and `docker-compose`.
- **Configuration:** Comprehensive environment variable management and secure secret handling.
- **Documentation:** Inline code documentation and auto-generated API documentation.

4. Evaluation Criteria

Submissions will be evaluated based on the following criteria:

- **AI/ML Engineering Excellence (40%):** Mastery of LangChain, robustness of the RAG implementation, effectiveness of context management, and thoughtful model integration.
- **Technical Implementation (35%):** Quality of code and architecture, comprehensiveness of the testing strategy, and clarity of documentation.
- **Problem-Solving & Innovation (15%):** Creative solutions to technical challenges, scalability considerations, and performance optimization.
- **Production Readiness (10%):** Implementation of monitoring, security controls, and a polished user experience.

5. Submission Guidelines

5.1. Required Deliverables

1. **Source Code Repository:** A public GitHub repository containing the complete, well-organized source code, dependencies, configuration files, and test suite.
2. **Documentation Package:** Detailed documentation including system architecture diagrams, a complete OpenAPI specification, user guides for different roles, and a developer setup guide.
3. **Demonstration Materials:** A 110 - 12 minute video walkthrough demonstrating key features, technical decisions, and performance benchmarks. Include sample data and defined test scenarios.
4. **Technical Report:** A brief report outlining architecture decisions, challenges faced, scalability analysis, and a roadmap for future enhancements.

5.2. Submission Process

- Share public GitHub repo link
- Send an email to the same address with the subject line: "AI Engineer Task Submission: Contextual AI Presentation Orchestrator - [Your Name]".
- The email body should contain an executive summary of your work, highlight key innovations, and confirm completion of all requirements.

6. Constraints & Important Notes

- **Resource Limitations:** All tools, services, and models used must be free, open-source, or have a free tier available. The system must be deployable and runnable on local infrastructure.
- **Ethical Considerations:** All test data must be publicly available and devoid of personal or sensitive information. All software licensing requirements must be adhered to.
- **Quality Standards:** Submissions are expected to meet high standards for code (PEP 8), testing (>80% coverage), and documentation.

Deadline

7 days after receiving this task assignment, i.e., lastly by **22/08/2025 EOD** . Submissions received after the deadline will not be considered.