Project Specification Document

Project Title: Smart Contract Summary & Q&A Assistant

Domain: LLM Pipelines, LangChain, Vector Stores, Gradio, LangServe

Type: Workshop Application Project (NVIDIA DLI Course Alignment)

1. Project Overview

The project is a small-scale web application that allows users to upload long documents (contracts, insurance policies, reports) and interact with them via a conversational assistant. Users upload PDF/DOCX, the system extracts, chunks, and embeds content, stores it in a vector store, and enables chat-based question answering with guard-rails and source citations.

2. Objectives

- Demonstrate understanding of LLM inference interfaces and microservices.
- Build an end-to-end RAG (Retrieval Augmented Generation) pipeline.
- Showcase strategies for long-form document processing.
- Apply embeddings and semantic similarity for guardrailing.
- Evaluate and validate retrieval/answer quality.

3. Scope

In Scope:

- File ingestion: PDF/DOCX.
- Chunking & embedding.
- Vector store setup (Chroma or FAISS).
- Retrieval + LLM answer generation.
- Chat interface with history.
- Guard-rails for safety and factuality.
- Document summarization (optional).
- Local deployment with FastAPI/LangServe.

Out of Scope:

- Production-scale deployment.
- Multi-language contracts (initially English only).
- Legal compliance beyond basic disclaimers.

4. Functional Requirements

- Upload & ingestion of contracts.
- Embedding creation & storage in vector DB.
- Semantic search-based retrieval.
- LLM-based question answering with source citations.

- Conversation state tracking.
- Optional summarization of contracts.
- Evaluation pipeline with metrics.

5. Non-Functional Requirements

- Performance: <5s response for medium contracts.
- Usability: Clean UI with upload & chat tabs.
- Security: Local storage only, no external sharing.
- Maintainability: Modular codebase for easy extension.

6. Architecture

Main Components:
- Frontend: Gradio UI (Upload, Chat).
- Backend: FastAPI + LangServe microservices.
- Pipelines: Ingestion, retrieval, summarization.
- Vector Store: Chroma / FAISS.
- LLM: OpenAI or local HuggingFace model.

7. Technology Stack

- LangChain, LangServe, FastAPI
- Gradio (UI)
- Chroma / FAISS (vector store)
- SentenceTransformers / OpenAI embeddings
- PyMuPDF, pdfplumber, python-docx (file parsing)

8. Deliverables

- Codebase (structured repo).
- Documentation (README, instructions).
- Demo UI (upload & chat).
- Evaluation Report (metrics, limitations).

9. Timeline (10 Days)

Day 1–2: Setup environment & repo
Day 3–4: Ingestion pipeline (extract, chunk, embed)
Day 5–6: Retrieval + answer pipeline
Day 7: Build Gradio UI
Day 8: Dialog state + summarization
Day 9: Evaluation tests
Day 10: Documentation & demo

10. Risks & Mitigations

- Hallucinations → enforce grounding & citations
- Large documents → configurable chunk size
- API limits → fallback to local LLM/embeddings

11. Future Enhancements

- Multi-doc search
- Domain-specific fine-tuned models
- Role-based access control
- Cloud deployment (Docker/Kubernetes)

12. System Flow Diagram

The following diagram illustrates the end-to-end flow of the Smart Contract Assistant system: