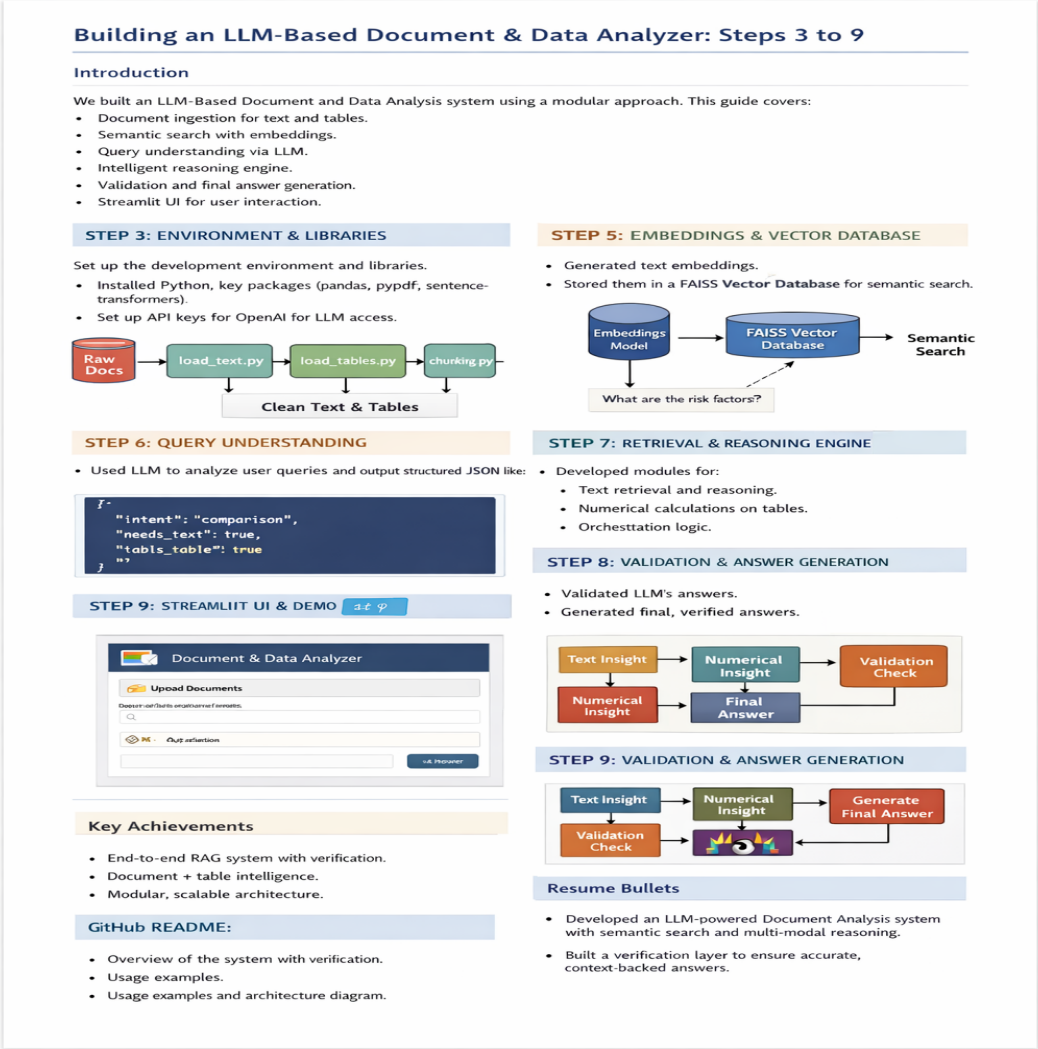
LLM-Based Document & Data Analyzer

Comprehensive Project Report: Step 3 to Step 9

System Architecture Diagram



STEP 3: Environment Setup
- Python 3.10+ virtual environment
- Install required packages (pandas, faiss, langchain, openai, sentence-transformers, streamlit, python-dotenv)
- Securely store API key in .env
- Virtual environment ensures reproducibility and dependency isolation

STEP 4: Document Ingestion
- Load PDFs, DOCX, TXT, CSV, Excel

- Chunk text into 500–800 tokens with 100 token overlap
- Add metadata (file name, page number, chunk ID)
- Code snippet:
from ingestion.load_text import load_text text = load_text('report.pdf') from ingestion.chunking import chunk_text chunks = chunk_text(text)

STEP 5: Embeddings & Vector Database

- Generate embeddings using sentence-transformers or OpenAI API
- Store embeddings in FAISS vector database
- Save/load vector index
- Metadata ensures traceable results
- Code snippet:
vs = VectorStore() vs.add_texts(chunks, metadatas) vs.save()

STEP 6: Query Understanding & Intent Detection

- LLM parses user query
- Determines intent: summary, numerical, comparison, explanation, mixed
- JSON output guides orchestrator
- Example JSON:
{ "intent": "comparison", "needs_text": true, "needs_table": true, "metrics": ["revenue", "growth"], "filters": {"time": "Q2"} }

STEP 7: Retrieval & Reasoning Engine

- Retrieve relevant text chunks
- Perform numerical computations on tables
- Orchestrator integrates text reasoning and table reasoning
- Sample snippet:
results = run_reasoning(parsed_query, question)

STEP 8: Answer Validation & Final Generation

- Validate answers against context to prevent hallucination
- Merge text and numerical insights
- Generate structured, verified answers
- Sample snippet:
final_answer = generate_final_answer(question, text_answer, numerical_data, sources)

STEP 9: Streamlit Application

- Upload multiple documents
- Build knowledge base
- Ask questions and get verified answers
- Sample Streamlit snippet:
question = st.text_input("Enter your question:") answer = run_full_pipeline(parsed_query, question) st.write(answer)

Project Outcome & Resume Value

- Multi-modal reasoning (text + numerical data)
- Enterprise-grade RAG architecture
- Anti-hallucination design
- Resume bullets for GitHub/interviews
- Example:
- "Built an LLM-based Document & Data Analyzer capable of reasoning over text and structured data with verification and source tracking."
- "Implemented multi-step RAG pipeline integrating embeddings, semantic search, numerical computation, and LLM reasoning."