Dynafy Technologies

# 📝 Assessment: Mini Knowledge Assistant with LangGraph + RAG

## Objective

Build a small backend service that:  
  
- Ingests documents (e.g., PDFs or Markdown files).  
- Uses RAG to answer user questions based on those documents.  
- Uses LangGraph to structure the flow with:  
 • State management (store conversation context, retrieved chunks).  
 • Conditional branching (detect when the model cannot answer and fallback to a “no answer found” node).  
 • Parallel nodes (fetch from multiple retrievers or run multiple model calls simultaneously).  
- Exposes a REST API to interact with the assistant.  
- Provides clear documentation (README with setup, architecture, and API usage).

## Requirements

1. Backend (Python)  
  
- Use FastAPI or Flask.  
- Create endpoints for:  
 • /upload → Accepts documents and adds them to the vector store.  
 • /chat → Accepts a question, runs LangGraph pipeline, returns answer.  
- Clean, modular code (separate routes, LangGraph flow, and RAG logic).

2. LangGraph Flow  
  
- Must include:  
 • Conditional branching: Detect if retrieved context is empty → return fallback message.  
 • Global state: Persist conversation history and retrieved results.  
 • Parallel nodes: Try two different retrievers (e.g., vector store + keyword search) and merge results before sending to the LLM.

3. RAG Implementation  
  
- Use FAISS / Chroma / any vector database.  
- Allow multiple document uploads.  
- Support chunking, embedding, and similarity search.

4. Documentation  
  
- README must include:  
 • Architecture diagram (can be a simple image or ASCII diagram).  
 • Setup instructions (dependencies, how to run).  
 • API usage examples (curl or Postman).  
 • Short explanation of design choices (why they used certain retriever, embeddings, etc.).

## Deliverables

/GitHub repo or zip with:  
- /src folder containing code.  
- README.md with full documentation.  
- Example .env.example (for API keys).  
- Sample test documents + example queries.  
- Working API that can be tested locally.

## Evaluation Criteria

|  |  |  |
| --- | --- | --- |
| Area | Points | What to Look For |
| LangGraph Workflow | 30 | Proper use of nodes, conditional branching, global state, parallel execution. |
| RAG Implementation | 20 | Document ingestion, embedding, retrieval relevance. |
| Backend Quality | 20 | Clean code, modular design, error handling, well-structured endpoints. |
| LLM Usage | 10 | Correct API usage, prompt quality, fallback handling. |
| Documentation | 20 | Clear README, diagrams, usage examples, explanation of decisions. |