Home Assignment for Senior Backend Engineer – AI Agent Systems & Orchestration

1. Problem Statement

Build a **mini "Research Assistant" agent** that answers natural-language questions about a local folder of documents.

- **All components can be free to use.**
    - Rely on open-source models (e.g., Mistral 7B, Llama 3-8B, Phi-3-mini) served through **Ollama** or **LocalGPT or any other API**.
    - Vector DB: **Chroma** (open-source, in-process) or **FAISS**.
- Must run end-to-end on a laptop with ≤ 8 GB RAM (Docker is fine).
- Expose one REST endpoint `/ask` that performs retrieval-augmented generation with short-term session memory.

2. Functional Requirements

Area	Must-Have (Free)	Nice-to-Have (optional)
Data ingestion	CLI/endpoint to load `.txt`/`.md` files, embed with sentence-transformers.	Incremental upsert.
Vector store	Chroma/FAISS, persisted to local disk.	Env-switchable back-ends.
LLM layer	Local model via Ollama or llama-cpp-python.	Switchable adapters for Bedrock/OpenAI
Agent orchestration	LangGraph (or any other alternative)	Tool node (e.g., Wikipedia fetch).
API	FastAPI (Python) or Express (Node). `/ask` returns `{answer, sources}`.	Streaming responses.

Observability	Pretty + JSON logs to stdout.	`/metrics` Prometheus endpoint.
Local run	`make dev` or `docker compose up` boots everything offline.	none
Deploy (nice)	Script for Heroku, Render, or AWS SAM **free tier**.	GitHub Actions CI.

3. Non-Functional Requirements

- **Latency goal**: ≤ 8 s on a single Q-A over 5 small docs.
- **Code quality**: typed, linted, ≥ 80 % unit-test coverage on core logic.
- **Security**: `.env.example` with placeholders; no secrets committed.
- **Docs**: README with quick-start, architecture diagram, and steps to record demo.

4. Deliverables

```
README.md        ← setup, design, free-only emphasis, video link
architecture.png ← diagram
/src             ← agent + API code
/tests           ← pytest / jest
docker-compose.yaml + Dockerfile
sample_data/     ← 3-5 tiny docs for demo
```

5. Demo Video (≤ 5 min) — Mandatory

1. Start the stack locally (`docker compose up`).
2. Ingest sample docs.
3. Call `/ask` twice to show memory.
4. Display logs and returned JSON.

5. (Optional) briefly show free-tier deploy script working.

6. Evaluation Rubric

Weight	Criterion
40 %	Correctness & completeness
25 %	Code / architecture quality
15 %	Performance & resource usage
10 %	Clarity of README & video
10 %	Extras (optional deploy, CI, dashboards)