# Project Instructions – LLM Agent Workflow

# Objective

Build a small, business-focused LLM system that chains **at least two agents** to handle a single user request end-to-end (example requests: “What’s our refund policy for annual plans?” or “Summarize the onboarding steps and draft an email to a new customer”).
The pipeline must accept a user prompt, route it through agents in sequence, and return a clear, grounded response.

# Scope of Work

## 1) Agent Flow

Design a **three-step loop** that runs until a final answer is produced:

1. **Decide (Planner/Router Agent)**
   Classifies intent (Q&A vs summary vs draft message), chooses tools, sets a plan, and defines what “done” means.
2. **Retrieve (RAG Agent)**
   Fetches supporting context from a **Vector DB and/or Graph DB**, returns top sources + short evidence snippets.
3. **Respond (Writer Agent)**
   Produces the final answer **using only retrieved context**, plus optional artifact generation (email/checklist/ticket).

**Loop condition:** If evidence is insufficient, the Decide agent triggers another Retrieve pass (max 2 retries) or returns a “need more info” response.

## 2) Tool Choice

Candidate can use **any** stack that achieves the workflow, such as:

- **Python**: LangGraph, LangChain Agents, OpenAI tool calling, plain Python
- **Data layer**: Chroma / FAISS / Pinecone / Weaviate (vector), Neo4j (graph), or hybrid
- **Models**: OpenAI / Gemini / Anthropic / local models (if feasible)

### 3) Safety and Guardrails

Add at least **one** guardrail:

- PII masking/redaction (email, phone, credit card patterns)
- Prompt-injection detection (block/flag attempts like “ignore previous instructions”)
- Output validation (must include citations OR “insufficient evidence”)

### 4) Demo Requirements

Process **at least three sample prompts** and show:

- Planner decision (intent + plan)
- Retrieval output (top sources)
- Final response (with citations or source references)

Example prompts (feel free to invent your own):

1. “What is the escalation path for high-priority incidents?”
2. “Summarize the onboarding steps and create a checklist for a new hire.”
3. “Draft a customer reply about a billing dispute based on our policy.”

# Deliverables

1. **Working code or flow export**
    - Notebook, Python repo, or exported Flowise/LangFlow/Botpress project
2. **Logs or screenshots of 3 test runs**
    - Must include intermediate agent outputs
3. **Short project report (≤ 2 pages)**
    - Architecture, tools used, key decisions, lessons learned
4. **Five-minute final presentation**
    - Problem → Architecture → Demo → Learnings

# Proposal Structure

Your proposal (1–2 pages) should include:

1. **Introduction**
    - Problem statement + target users + example user prompts
2. **Scope of Work**
    - Agents, tools, guardrails, demo plan
3. **Approach and Tools**
    - Framework + model + storage choice, why it fits
4. **System Architecture**

    - Diagram of the agent pipeline (simple is fine)
5. **Timeline and Work Split**
    - Weekend milestones + who does what (if a team)

# Evaluation Criteria

- Clear mapping of user prompts through **all agents**
- Correct, reproducible outputs on the 3 sample prompts
- Good use of tools (RAG quality, correct chaining, clean state handling)
- Guardrail present and demonstrably working
- Documentation quality and presentation clarity
- Meets submission deadlines

# Timeline and Deadlines

## Proposal Phase

- **Sat 20 Dec 2025** – Proposal document submission
- **Sun 21 Dec 2025** – Proposal presentation + feedback

## Build Phase

- **Sat 27 Dec 2025** – Working skeleton (agents wired + basic retrieval)
- **Sun 04 Jan 2026** – Progress presentation (demo 1 prompt end-to-end)
- **Sat 10 Jan 2026** – Final submission (3 prompts + guardrail + cleaned logs)
- **Sun 11 Jan 2026** – Final Demo(code + report) and final demo