Lab 3: The Reasoning Loop (Powered by LangGraph)

Item	Details
Orchestration	**LangGraph**
Prerequisites	Lab 1 Architecture, Lab 2 Vector Store
Focus	State Management, Nodes, and Conditional Edges

Objective

The objective of this lab is to move your project from static retrieval to autonomous reasoning. You will implement a ReAct (Reason + Act) loop using the LangGraph framework. Your agent will no longer simply "respond"; it will "think," decide which of your project-specific tools to use, and update its internal state based on the results.

The Project-Specific Toolset

In this lab, you are building the functional capabilities of your agent. These tools must be the ones you identified in your Lab 1 Inventory.

- **The "Grounding" Tool:** A tool that queries the Vector DB you built in Lab 2.
- **The "Action" Tools:** Python functions that perform calculations, API calls, or database lookups specific to your use case (e.g., calculate_risk_score, fetch_live_inventory).

Mandatory Tasks

- ✓ **Task 1: Tool Engineering with Pydantic**

  Develop the Python functions required for your project.

  - **Requirement:** Every tool must use the @tool decorator from langchain_core.tools.
  - **Strict Validation:** Use Pydantic to define the input schema for each tool. This ensures the LLM does not pass "garbage" data to your functions.
  - **Docstrings:** Write descriptive docstrings. The LLM uses these as instructions to understand *when* to invoke the tool.

- ✓ **Task 2: Defining the Graph State & Nodes**

  Define your LangGraph structure to manage the conversation flow.

  - **The State:** Define a TypedDict that stores the messages list (history of thoughts and actions).
  - **The Agent Node:** A function that takes the current State, calls the LLM, and returns the next step.
  - **The Tool Node:** A specialized node that executes the tool calls identified by the LLM.

- ✓ **Task 3: The Conditional Router**

  Implement the logic gate that controls the loop.

  - Write a function (the "router") that checks the LLM's last message.
  - Logic: If the LLM generated "Tool Calls," the graph routes to the Tool Node. If the LLM generated a "Final Answer," the graph routes to END.

Expected Outcomes (Submission Checklist)

1. **tools.py**: Your project-specific tools with Pydantic validation and @tool decorators.
2. **graph.py**: The compiled LangGraph code containing your StateGraph, Nodes, and Edges.

Assessment Rubric

Criteria	Weightage	Full Marks Requirements
Requirements	3	
Working	3	
Viva	4	