TASK 1

Objective:

Create a **basic AI agent** using **LangGraph** or similar AI Agent framework that can answer questions from a small knowledge base using **RAG (Retrieval-Augmented Generation)**.
The goal is to test your understanding of AI agent workflows, RAG pipeline design, and LangGraph basics.

Project Description:

You are required to build a **Q&A AI Agent** that can:

1. Accept a user question (e.g., *"What are the benefits of renewable energy?"*)
2. Retrieve relevant information from a small text dataset (you can use a few .txt files or PDFs related with the questions you are expecting).
3. Use an **LLM (e.g., OpenAI, Gemini, Claude or Groq/HuggingFace)** to generate an answer using the retrieved context.
4. Show a simple reflection or validation step where the agent checks if the answer is relevant to the question.

The agent workflow should have at least **four LangGraph nodes**:

- **plan:** interpret the query and decide if retrieval is needed.
- **retrieve:** perform RAG using a local vector database (e.g., Chroma).
- **answer:** generate the final answer using the LLM and retrieved documents.
- **reflect:** evaluate the answer for relevance or completeness.

Important: If selected for the next round, you will be required to walk through the entire code line-by-line; please do **not** submit your solution if you are not willing to do this.

Requirements:

- Use **LangGraph or similar framework** to define the agent workflow.
- Use **LangChain** or **LlamaIndex or similar framework** for RAG (retrieval + generation).
- Use **ChromaDB** or any open-source **VectorDB** to store and query embeddings.

- Use **Hugging Face** or **OpenAI embeddings** for vector creation.
- Include minimal logging or print statements to show each step's output (plan → retrieve → answer → reflect).
- The code should run locally in a Jupyter notebook or Python script.
- Include requirements.txt file

Bonus Points:

- Add a simple **Streamlit** or **Gradio** UI to ask questions interactively.
- Use **LangSmith** or **TruLens** for trace and evaluation logging.
- Add a short report (1–2 paragraphs) describing how your agent works and what challenges you faced.
- Include an evaluation code to evaluate the agent output (RAGAs or any LLM as a Judge method or BLEU/ROUGE/BERTScore etc.)

Submission:

- Submit a **Jupyter Notebook (.ipynb)** or **Python script (.py)** file.
- Include a **README.md** describing setup steps and your approach.
- **Deadline:** ***7 PM, 05.11.2025***