📘 AI Trainee Program – Phase 1

📖 Guidelines for Journey

- **Consistency:** Dedicate focused time daily to learning and coding.
- **Documentation:** Keep detailed notes and comments in your code to track your thought process.
- **Experimentation:** Try different approaches. Treat mistakes as learning opportunities.
- **Architecture:** Follow **MVC architecture** principles when structuring your code.
- **Version Control:** Save and push your code/scripts for each milestone using **Git/GitHub**. Each milestone should have clear commits and documentation.

Milestone 1: Learning Python

Learning Objectives

- Gain a solid understanding of Python basics.
- Write clean, well-structured Python code.

Activities

- Study Python fundamentals: variables, data types, control flow, functions, modules.
- Practice beginner exercises (loops, conditionals, list/dict operations).
- Apply **Google Python Style Guide** (type hints, docstrings, naming conventions).

Deliverables

- A short **discussion summary** of Python basics.
- A set of **Python scripts** showing basic functionalities (loops, functions, classes).

Milestone 2: Understanding RAG & Core Technologies

Learning Objectives

- Understand **Retrieval-Augmented Generation (RAG)** architecture.
- Familiarize with a **local LLM** (e.g., DeepSeek, GPT-OSS, LLaMA) and an **indexing library** (LlamaIndex or LangChain).

Activities

- Studying RAG architecture (retriever, generator, integration).
- Install and configure a local LLM.
- Write initial scripts to interact with the chosen LLM and index a few documents.

Deliverables

- A **discussion summary** of RAG concepts.
- A **Python script** demonstrating interaction with your chosen local LLM + a simple index build.

Milestone 3: Data Preparation & Indexing

Learning Objectives

- Preprocess text data, create embeddings, and index documents.
- Expose these operations via **FastAPI endpoints**.

Activities

- Prepare dataset (text files, articles, or documents).
- Generate embeddings using a vector store (FAISS, ChromaDB, Weaviate, etc.).
- Build a FastAPI service with endpoints:
  - POST /index → preprocess and index documents.
  - POST /search → accept a query, return relevant documents.

Deliverables

- A **FastAPI project** exposing endpoints to index and manage documents.
- Documentation explaining how to call the endpoints and what they return.

Milestone 4: Retrieval & LLM Integration

Learning Objectives

- Implement retrieval of relevant documents.
- Integrate retrieval results with the local LLM to generate responses.
- Expose functionality via FastAPI.

Activities

- Add a retrieval pipeline that fetches documents based on a query.
- Pass retrieved documents to LLM for response generation.
- Extend FastAPI with:
  - POST /ask → accept a query, retrieve documents, and generate an LLM response.

Deliverables

- A **FastAPI project** with working endpoints for document retrieval and LLM integration.
- Example **cURL or Postman requests** demonstrating usage.

📍 Milestone 5: Chat History, Prompt Engineering & Contextual RAG

🎯 Learning Objectives

- Understand the importance of **chat history and context** in conversational AI.
- Learn the basics of **prompt engineering** (instruction design, role prompting, few-shot examples).
- Design and integrate a **chat history storage system**.
- Enhance the existing RAG bot with **context-aware conversations**.

📌 Activities

1. **Chat History & ERD**
   - Design an **ER Diagram** for chat history.
   - Define your own **table names and structure** to store sessions, messages, and context.
   - Implement persistence (e.g., SQL DB) for storing user queries, bot responses, and retrieved context.
2. **Prompt Engineering**
   - Learn and apply key prompt engineering concepts:
     - **Instruction Prompting**: guide the model with clear instructions.
     - **Role Prompting**: set the assistant's persona.
     - **Few-shot Prompting**: show examples to improve consistency.
   - Experiment with rewriting prompts to improve response quality.
3. **Integration with RAG Bot**
   - Extend FastAPI endpoints:
     - POST /chat → accepts a new user message, stores it, retrieves context from history + RAG, then calls the LLM.

        - GET /history/{session_id} → returns the conversation history for a session.
    - Ensure the bot responds with **context-aware answers**, using both history and retrieved documents.

☑ Deliverables

- **ER Diagram** of the chat history database (with custom naming & design).
- **Database implementation** for storing chat history.
- Extended **FastAPI endpoints**:
    - POST/chat (context-aware chat with RAG + history).
    - GET/history/{session_id} (retrieve stored history).
- A **short demo or documentation** showing:
    - How prompts were engineered and improved.
    - How history + RAG improves the conversation quality.
- **GitHub Repository** containing milestone code, with clear commits, branches, and documentation.

Bonus (Optional for Milestone 5)

- Add **summarization of old chat history** (to keep the context short but relevant).

Milestone 6: Optimization & Finalization

Learning Objectives

- Optimize the system for performance, accuracy, and usability.
- Prepare the system for a final presentation/demo.

Activities

- Improve embedding/search performance.
- Conduct final testing with multiple datasets.
- Prepare a **short presentation/demo script** showing how the system works end-to-end.

Deliverables

- A **fully functional RAG system** running with FastAPI endpoints.
- A **demo presentation** explaining:
    - System architecture
    - Challenges and solutions
    - Example use cases

Bonus (Optional for Milestone 6)

- Create a **UI chat page** showing user messages, bot responses, user sessions and retrieved documents side by side.