Action Plan: Implementing a Local LLM with RAG & Data Visualization for BEAR

Phase 1: Setting Up the Local LLM

Goal: Deploy an offline LLM capable of responding to BEAR queries.

Select and Deploy the Local LLM:

- Choose a model: Llama 2 (7B/13B), Mistral 7B, or Phi-2 (depending on available resources).

- Install the model using Hugging Face Transformers or Llama.cpp for CPU/GPU efficiency.

- Test basic inference: Ensure it understands BEAR-related prompts.

Deliverable: Local LLM deployed and responding to queries.

Phase 2: Implementing RAG for BEAR's Dataset

Goal: Enable the LLM to retrieve and generate responses based on BEAR's experiments dynamically.

Prepare the Dataset for RAG:

- Convert BEAR's experiment data into structured text embeddings (e.g., FAISS, ChromaDB).

- Store metadata: Experiment ID, material type, force-displacement data, failure/success, etc.

Set Up the Vector Database (FAISS or ChromaDB):

- Index experiment data using embedding models (BGE, InstructorXL, etc.).

- Test retrieval: Query 'Summarize the last 10 experiments' and check for relevant data.

Integrate RAG with the LLM:

1. User inputs a question.

2. RAG retrieves relevant experiments from the database.

3. The LLM generates a response based on retrieved data.

Deliverable: BEAR's LLM can fetch real-time experimental insights from its dataset.

Phase 3: Enabling Data Visualization

Goal: Allow the LLM to generate visual graphs based on user queries.

Implement Python Execution for Plot Generation:

- Use Matplotlib/Plotly/Seaborn for dynamic graph generation.

- Set up a sandboxed execution environment to securely run LLM-generated Python code.

Connect LLM to the Plotting System:

1. Understand user queries (e.g., 'Plot force vs. displacement for the last 5 TPU samples').

2. Generate a Python script for visualization.

3. Execute the script and return a graph output.

Validate Graph Quality & Accuracy:

- Ensure plots correctly represent BEAR's experimental data.

- Test edge cases (e.g., missing data, large datasets).

Deliverable: Users can request plots via the LLM, and it generates accurate visualizations.

Phase 4: Building a User Interface for BEAR's LLM

Goal: Create a GUI for easy interaction with the LLM and visualization system.

Choose the GUI Framework:

- Streamlit or Gradio for a lightweight web interface.

- Dash or Flask/FastAPI for a more advanced system.

Develop the GUI Features:

- Text box: Users can input queries (e.g., 'Compare TPU vs PLA').

- Dropdowns: Select materials, parameters, and time range.

- Visualization Panel: Display generated plots dynamically.

Test & Deploy the GUI:

- Ensure the UI correctly processes queries, retrieves data, and visualizes results.

- Deploy on a local server accessible within BEAR's lab network.

Deliverable: BEAR's LLM has an interactive UI for text queries & visualizations.

Phase 5: Deploying and Refining the System

Goal: Final testing, optimization, and deployment.

Optimize Model Performance:

- Use quantization (4-bit, 8-bit) to improve LLM efficiency.

- Optimize database retrieval speed for RAG queries.

Implement Real-Time Monitoring & Alerts:

- If integrated with Slack Bot, enable live experiment updates.

- Notify researchers when anomalies occur (e.g., printer failure, mass calibration issue).

Final Testing & Deployment:

- Conduct extensive testing with real BEAR experiment queries.

- Deploy on a secure, local server for lab use.

- Document the system for future improvements & troubleshooting.