# Problem Statement: Build a Secure AI Chatbot for Data Processing

**Objective:**
Create a deployable langraph based AI chatbot that can accept csv data from users, use Gemini to generate code for plotting and fitting the data, and securely execute the generated code in a sandbox environment like REPL or Docker. It should analyze the data plotted and give a summary. Upon asking a follow up question the agent should have memory of the last conversation and respond accordingly.

Perform as much of the agent building as possible. All the requirements need not be met to proceed with the interview. But a basic functioning chatbot is required.

---

**Requirements:**

1. **Chatbot Interface:**
    - The chat ot should greet the user in a friendly manner and ask how it can be of assistance.
    - The chatbot should allow users to upload a data file (e.g., CSV) provided.
    - Gemini should take a decision on what plot needs to plotted based on user prompt and provided data, and Gemini provided code should run in the sandbox and the final results should be shown on the interface.

2. **AI Code Generation:**
    - Use the Gemini API (or a similar large language model) to generate the code needed to process and plot the uploaded data.
    - The generated code should be optimized for secure execution, including data parsing, error handling, and fitting.

3. **Secure Execution Environment:**
    - The code generated by Gemini should be executed in a secure sandbox, REPL or Docker, to prevent unauthorized access or system vulnerabilities.
    - The sandbox should have limited permissions and restricted access to ensure data security.

4. **Deployment Readiness:**

    - Aim to make the chatbot as close to a deployable product as possible.

---

**Evaluation Criteria:**

- **Integration Skills:** Ability to connect the chatbot to the Gemini API and the secure sandbox.
- **Security Awareness:** Understanding of containerization, isolation, and secure code execution.
- **Code Quality:** Clean, readable, and well-documented code.
- **Problem-Solving:** Creative approaches to handling edge cases and user inputs.
- **Scalability:** Potential to expand the chatbot's functionality for future Batlabs AI needs.

---

**Bonus Points:**

- Implement real-time status updates for the user (e.g., data upload progress)
- Use a modern chatbot framework like FastAPI, Flask, or Node.js with a React or Next.js frontend.
- Add data visualization options like Plotly or Matplotlib for a more polished user experience.