## The Craft: AI-Powered GitHub Issue Assistant

### Core Philosophy: "Agentic Thinking in a Box"

This craft is designed to be a microcosm of the actual work you'd do at Seedling. It's not just about building a model; it's about creating a small, useful AI agent that integrates different parts (LLMs, data, APIs) to solve a concrete business problem.

### Problem Statement:

At Seedling Labs, we move fast. To maintain quality and speed in our development workflow, efficiently understanding and prioritizing new GitHub issues is crucial.

Your task is to build a simple web application that takes a GitHub repository URL and an issue number as input. The application should then use AI to analyze the issue and provide a structured summary.

### Core Requirements:

1. **Input UI:** A simple user interface with fields for a public GitHub repository URL (e.g., https://github.com/facebook/react) and an Issue Number.
2. **Backend API:** A lightweight API endpoint (e.g., using FastAPI or Flask in Python) that triggers the analysis.
3. **AI Core:** The backend should:
   - Use the GitHub API to fetch the title, body, and comments of the specified issue.
   - Utilize a Large Language Model (LLM) to process this information.
   - Generate a structured output in a specific JSON format.
4. **Output Display:** The UI should display the generated analysis in a clean, readable format.

### Required AI-Generated JSON Output Format:

```
{
   "summary": "A one-sentence summary of the user's problem or request.",
   "type": "Classify the issue as one of the following: bug, feature_request,
documentation, question, or other.", "priority_score": "A score from 1 (low) to 5
(critical), with a brief justification for the score.",
   "suggested_labels": ["An array of 2-3 relevant GitHub labels (e.g., 'bug', 'UI',
'login-flow')."],
   "potential_impact": "A brief sentence on the potential impact on users if the issue is a
bug."
}
```

**Tech Stack Suggestions (Flexibility Encouraged):**

- **Backend:** Python (FastAPI, Flask)
- **Frontend:** Streamlit or a simple HTML/JS page that calls the API. (Streamlit is excellent for rapid prototyping).
- **LLM Interaction:** LangChain, LlamaIndex, or direct API calls (e.g., Google Gemini, OpenAI, or an open-source model via Hugging Face).

**Submission:**
The entire project should be submitted as a link to a public GitHub repository. Your repository **must** include a clear and comprehensive README.md file explaining how to set up and run your project in under 5 minutes. This demonstrates professionalism and empathy for other developers.

**Evaluation Framework (The Rubric)**

We will evaluate your submission based on the following criteria, reflecting Seedling Labs' core values of problem-solving, quality, speed, and communication:

1. **Problem Solving & AI Acumen (40%)**
   - **Prompt Engineering:** How effectively did you craft the prompt for the LLM? Is it robust and does it reliably produce the desired JSON format? Did you consider techniques like few-shot prompting?
   - **System Design:** How did you structure your code to fetch data, process it with the LLM, and return the result? Is the design logical and efficient? ○ **Handling Edge Cases:** Did you consider scenarios like issues with no comments, or very long issue bodies? How did your solution account for these?
2. **Code Quality & Engineering Practices (30%)**
   - **Clarity & Readability:** Is the code clean, well-commented, and easy to understand?
   - **Project Structure:** Is the project organized logically into files and folders?
   - **README:** Is your README.md clear and comprehensive, allowing for easy setup?
   - **Dependency Management:** Did you use a requirements.txt or pyproject.toml file?
3. **Speed & Efficiency (20%)**
   - **Tool Usage:** Did you leverage libraries (e.g., LangChain, FastAPI) effectively to avoid reinventing the wheel and accelerate development?
   - **Functionality:** Is the final product snappy, usable, and does it fully address the problem statement?
4. **Communication & Initiative (10%)**
   - **Git History:** Are your commit messages clear, descriptive, and do they tell a story of your development process?

- **Going the Extra Mile:** Did you add any small, thoughtful features not explicitly asked for (e.g., a button to copy the JSON, basic error pop-ups in the UI, caching results)?

**Good luck!**