Hiring Task: Making an Open-Source Project "AI-Native"

Scenario

Welcome to Frequenz! As your first mini-project, we want you to help us make one of our open-source projects more understandable and accessible to AI systems. Your goal is to extract key information about the project, structure it using semantic web principles, and build a simple tool to query this new knowledge base.

For this task, you will focus on our primary open-source library: the **Frequenz SDK for Python**.

* **GitHub Repository:** `https://github.com/frequenz-floss/frequenz-sdk-python`

Objective

Your mission is to create a small, machine-readable knowledge graph for this project and demonstrate how it can be used to answer questions accurately.

Part 1: Knowledge Modeling & Extraction

Your first step is to define a "shape" for the knowledge and then populate it.

1. **Design a Schema (Ontology):**
    * Design a simple ontology or schema that captures the essential concepts of this software project. Think about what an engineer or an LLM would need to know.
    * Consider concepts like:
        * Project Name & Description
        * Purpose / Use Case (What problem does it solve?)
        * Installation Instructions
        * Key Features or Components
        * Code Examples / Usage Snippets
        * License
        * Dependencies
    * You are free to use, extend, or mix vocabularies like `schema.org` (e.g., `SoftwareApplication`, `HowTo`, `Question`, `Answer`) or define your own simple terms.

2. **Extract & Structure the Knowledge:**
    * Write a Python script that extracts this information from the project's GitHub repository. You can focus on the `README.md` file as your primary source. You may use the GitHub API or web scraping libraries (like `BeautifulSoup` or `Playwright`).

    * Structure the extracted information according to the schema you designed in the previous step.
    * The final output of this script should be a single file named `project_knowledge.jsonld` containing the structured data in **JSON-LD** format.

Part 2: Knowledge Retrieval Application

Now, build a simple tool to demonstrate the value of your structured data.

1. **Create a Query Script:**
    * Write a Python script (e.g., `query.py`) that loads your `project_knowledge.jsonld` file.
    * This script should accept a natural language question as a command-line argument.
    * Based on the question, your script should perform a simple semantic search over your structured data to find the most relevant "chunk" of knowledge and print it to the console.
    * **Note:** You are **not** required to use an LLM or a vector database for this. The goal is to simulate the *retrieval* part of a RAG system. Simple keyword matching, regex, or basic NLP logic is sufficient.

2. **Example Queries to Support:**
    * `python query.py "What is the Frequenz SDK for?"`
    * `python query.py "How do I install the sdk?"`
    * `python query.py "Show me an example of how to use it."`

Part 3: Explanation & Rationale

Create a `README.md` file to accompany your submission. In this file, please explain your work:

1. **Schema Design:** Briefly explain the choices you made in your schema. Why did you choose those specific properties and structures?
2. **Process:** Describe your extraction and retrieval process. What were the challenges? What assumptions did you make?
3. **Next Steps & Improvements:** If you had more time, how would you improve this system? Discuss how you would integrate this with LLMs. For example, you could mention:
    * Using embedding models and vector databases (e.g., Pinecone).
    * Strategies for chunking the content.
    * How this structured data could be used in a full RAG pipeline (e.g., using LangChain or LlamaIndex).
    * How you would automate this process to keep the knowledge graph synchronized with the source repository.

4. **Public Visibility:** How would publishing this JSON-LD on a webpage improve the project's visibility to systems like Google Search, Perplexity, and other generative AI tools?

Deliverables

Please submit a link to a Git repository (e.g., on GitHub or GitLab) containing:

1. Your Python script(s) for extraction and querying.
2. A `requirements.txt` file listing any dependencies.
3. The final `project_knowledge.jsonld` file generated by your script.
4. Your `README.md` file with the explanations.

Evaluation Criteria

We will be looking at:

* **Code Quality:** Is your Python code clean, well-commented, and easy to run?
* **Knowledge Modeling:** How logical and useful is your schema design? Does it effectively capture the domain?
* **Retrieval Logic:** How well does your query script retrieve relevant information for the sample questions?
* **Strategic Thinking:** How insightful are your written explanations, especially regarding next steps and AI visibility? This demonstrates your understanding of the broader ecosystem.

Good luck! We are excited to see how you approach the challenge.