**🌾 Project Overview: KCC Query Assistant (Offline + Fallback)**

**📌 Purpose**

The **KCC Query Assistant** is an intelligent question-answering system designed to help users—especially farmers and agricultural experts—get instant, natural language responses to farming-related queries. It leverages the **Kisan Call Center (KCC)** dataset for offline use and incorporates fallback internet search to ensure users always get helpful information, even when the local dataset lacks relevant answers.

**🎯 Objectives**

* Provide **offline question answering** using an open-source LLM (Gemma 2B) for low-connectivity environments.
* Support **semantic retrieval** of relevant information using FAISS and sentence embeddings.
* Implement a **fallback mechanism** (via SerpAPI and DuckDuckGo) to ensure high answer coverage.
* Enable users to query in **natural language** without needing technical knowledge.
* Offer a **user-friendly interface** through Streamlit, with clear feedback and guidance.

**🧠 Core Components**

| **Component** | **Description** |
| --- | --- |
| **KCC Dataset** | Real agricultural Q&A data from Kisan Call Center |
| **Preprocessing** | Converts CSV data into clean JSON with context |
| **Embedding + FAISS** | Transforms data into semantic vectors for fast similarity search |
| **Gemma 2B (via Ollama)** | Local LLM used to answer questions using retrieved context |
| **Fallback Search** | Live web search via SerpAPI or DuckDuckGo for unmatched queries |
| **Streamlit UI** | Simple web app interface for interacting with the system |

**🔍 How It Works**

1. **User enters a natural language question.**
2. System **searches the local KCC dataset** for the most relevant context using semantic similarity.
3. If a match is found:
   * Context is passed to **Gemma 2B LLM** to generate an answer.
4. If no match is found:
   * The query is routed to a **live internet search** engine for fallback results.
5. The result is displayed clearly, with source type (local or live) indicated.

**🛠️ Technologies Used**

* **Language Model:** Gemma 2B via Ollama (offline inference)
* **Vector Search:** FAISS + HuggingFace Sentence Transformers
* **Frontend:** Streamlit
* **Fallback APIs:** SerpAPI (Google), DuckDuckGo
* **Data Handling:** Pandas, JSON
* **Environment:** Python (.env support), cross-platform

**🧑‍🌾 Target Users**

* Farmers seeking real-time agricultural advice
* Agricultural officers and researchers
* NGOs and organizations building local advisory tools
* Developers creating rural tech solutions