Problem Statement

You are tasked with building an offline-capable, local-first AI application—KCC Query Assistant—that allows users to query agricultural advice from the Kisan Call Center (KCC) dataset.

Objectives

1. Data Integration & Preprocessing
    - Download the public KCC dataset.
    - Clean, normalize, and split into logical “document” chunks or Q&A pairs.
    - Export both raw and preprocessed formats, preserving metadata fields.
2. Local LLM Deployment
    - Use an open-source model via the Ollama API (e.g., Gemma 3, Deepseek).
    - Quantize as needed for CPU/GPU efficiency (e.g., GGUF/GPTQ).
    - Ensure the model runs entirely offline.
3. Retrieval-Augmented Generation (RAG)
    - Generate embeddings for each document chunk using a sentence-transformer (e.g., MPNet, bge-large-en).
    - Store embeddings in a lightweight vector database (ChromaDB, FAISS, or MongoDB).
    - Implement semantic search to retrieve top-k relevant chunks for any query.
    - If no context meets a relevance threshold, invoke a live Internet search and clearly notify the user.
4. User Interface
    - Build a simple local web app (Streamlit or React.js).
    - Allow natural-language queries.
    - Display structured answers, highlighting which advice came from KCC versus fallback search.

Flow Summary

1. Data Ingestion
   → Load raw KCC CSV

2. Preprocessing
   → Clean, normalize, chunk Q&A pairs
3. Embedding Generation
   → Encode chunks into vectors
4. Vector Store Ingestion
   → Index embeddings in FAISS/ChromaDB/MongoDB
5. Query Handling (UI Layer)
   - User submits query
   - Retrieve top-k context via semantic search
   - If context found:
     – Pass context + query to LLM → display response
   - Else:
     – Notify “No local context found”
     – Perform live Internet search → display fallback results

Example Use Cases

- “What pest-control methods are recommended for paddy in Tamil Nadu?”
- “How to manage drought stress in groundnut cultivation?”
- “What issues do sugarcane farmers in Maharashtra commonly face?”

You may create additional queries reflective of your processed data. Ensure your UI, documentation, and demonstrations cover these core scenarios.

Submission Guidelines

Please follow these instructions carefully when submitting your KCC Query Assistant project.

1. Repository Setup
   - Create a private Git repository named `YourName_KCCQueryAssistant`.
   - Grant our review team collaborator access.
   - Submit only the repo link and your Google Drive demo-video link via the form.
2. Deliverable Checklist
   Ensure your repository contains:
   - Raw KCC Dataset plus comprehensive technical documentation covering every step workflow.
   - Preprocessed Dataset files and scripts.
   - Vector Database Artifacts or build scripts (ChromaDB, FAISS, or MongoDB).
   - Source Code for:
     1. Data ingestion & preprocessing
     2. Embedding generation & vector store ingestion
     3. RAG pipeline & LLM integration
     4. Web UI (Streamlit or React.js)
   - Sample Queries relevant to data to test atleast 10.
   - README.md covering installation, dependencies, overview, and launch instructions.
3. Demonstration Video
   - Record a 3–5 min screencast showing:
     1. Local startup of the LLM and vector store
     2. At Least 3-5 Queries returning KCC-based answers
     3. Fallback to live Internet search when no local context is found for at least 2-3 queries.
   - Upload to Google Drive (view-only) and include the link.