KCC Query Assistant - Technical Documentation

# 1. Introduction

The KCC Query Assistant is a local-first, offline-capable AI-powered system designed to answer Indian agricultural queries using a large historical dataset from the Kisan Call Center (KCC). The system leverages Retrieval-Augmented Generation (RAG) to provide fact-based responses using a local language model (LLM) and falls back to a live web search (via SerpAPI) when no relevant local data is found.

# 2. Problem Statement & Objectives

The goal is to develop an application that processes queries from Indian farmers and provides accurate answers based on historical KCC data or a fallback online search. Key objectives include:  
- Download and clean the KCC dataset  
- Normalize and preprocess data into Q&A format  
- Use a sentence-transformer model to generate embeddings  
- Implement a RAG pipeline with a local LLM  
- Create a simple Streamlit-based web interface

# 3. Dataset Overview

Dataset Source: [KCC Chakshu Portal](https://kcc-chakshu.icar-web.com/6\_data\_extract.php)  
Initial Shape: 996,618 rows × 14 columns  
After Cleaning: 996,551 rows × 14 columns

Key Attributes:

* - StateName, DistrictName, BlockName, Crop
* - QueryType, QueryText, KccAns, Category
* - Sector, Year, Month, Day
* - Latitude, Longitude

# 4. Preprocessing Pipeline

The dataset was cleaned by removing NaN values and standardizing text fields (lowercased, whitespace trimmed). Categorical columns such as Crop, DistrictName, and Sector were normalized. Rows with empty or meaningless responses were also dropped.

# 5. Embedding Generation

We used the 'all-MiniLM-L6-v2' model from SentenceTransformers to create vector embeddings of Q&A pairs. 20,000 examples were used to build the prototype system.  
  
Due to the ChromaDB constraint of 5461 entries per batch, data was split into 4 chunks for ingestion.  
Embedding size: 384 dimensions

# 6. Local LLM Setup with Ollama

We deployed the Gemma 3B (1B variant) model locally using [Ollama](https://ollama.com). Gemma3:1B was selected due to limited hardware capabilities, avoiding the need for quantization.  
  
Custom Modelfile:  
```  
FROM gemma3:1b  
PARAMETER top\_p 0.7  
PARAMETER temperature 0.3  
SYSTEM You are the Kisan Call Center (KCC) Query Assistant, an expert AI specialized in agricultural advice for Indian farmers...```  
This customization helps the model answer in a direct, expert tone as expected from KCC advisors.

# 7. Retrieval-Augmented Generation (RAG)

We implemented a classic RAG pipeline using ChromaDB for vector storage and semantic search. The query pipeline:  
- Embed the user query  
- Retrieve top-k similar QA pairs  
- If context found, pass to LLM  
- If not, perform fallback search via SerpAPI  
  
Semantic similarity enables precise, context-aware responses.

# 8. Streamlit Frontend

The frontend is a lightweight local app built with Streamlit. Launch command:  
```  
streamlit run app.py  
```  
Local URL: http://localhost:8501

# 9. Model Parameters

- Temperature = 0.3: Limits creativity, encourages fact-based answers  
- Top\_p = 0.7: Narrows down output token diversity to improve relevance

# 10. Future Scope

- Refactor to avoid regenerating embeddings on each Streamlit run  
- Add persistent storage to reduce redundant computation  
- Optionally deploy on a cloud server for public access  
- Fine-tune Gemma3:1B using LoRA or PEFT on domain-specific Q&A  
- Support multilingual queries for broader farmer accessibility

# 11. References

- KCC Dataset: https://kcc-chakshu.icar-web.com/6\_data\_extract.php  
- Ollama: https://ollama.com/search  
- SerpAPI: https://serpapi.com/dashboard  
- GitHub Repo: https://github.com/Gireesh-Guntupalli/GireeshGuntupalli\_KCCQueryAssistant  
- Sentence Transformers: https://www.sbert.net/  
- ChromaDB: https://docs.trychroma.com/  
- Streamlit: https://streamlit.io/