iFarmAssist – An Integrated Farm Assistant

Abhina Saseendran – 01
Devasena Sreelakshmi E V – 27
Prathyush M – 49
Aloka Manoj – 69
Arjun Saji Panichikkal – 75

Supervised by: Dr. Rafeeque P C

October 2025

Introduction

Background

- **Overview:** An AI-powered platform for farmers provides personalized agricultural advice via text, voice, or image queries in Malayalam or English, with complex issues escalated to local officers.
- **About the Project:** The AI gives farm advice using LLM + RAG with user and weather data, escalating low-confidence cases to experts.
- **Real-World Context:** The platform offers instant Malayalam advice and improves through a feedback-driven Learning Loop, reducing expert workload.

Relevance of the Topic

Why this topic?

- **Current Relevance:** The AI system empowers farmers with context-aware guidance via text, voice, or image queries.
- **Real-World Problem:** Farmers face delays in expert advice due to language, service gaps, and complexity, reducing productivity.
- **Positive Impact:** The platform gives instant Malayalam/English advice and improves through a feedback-driven Learning Loop.

Problem Statement

Problem Statement

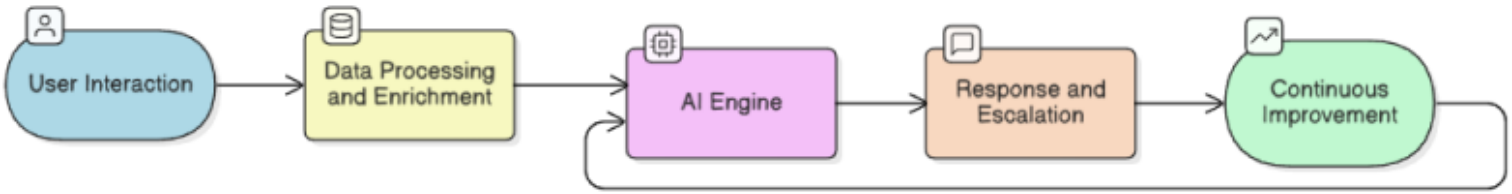
To develop an AI-powered, multimodal Farmer Query Support and Advisory System that leverages Large Language Models (LLMs) to provide farmers in Kerala with instant, reliable, and personalized agricultural advice in Malayalam through text, voice, and image-based queries.

Project Objectives

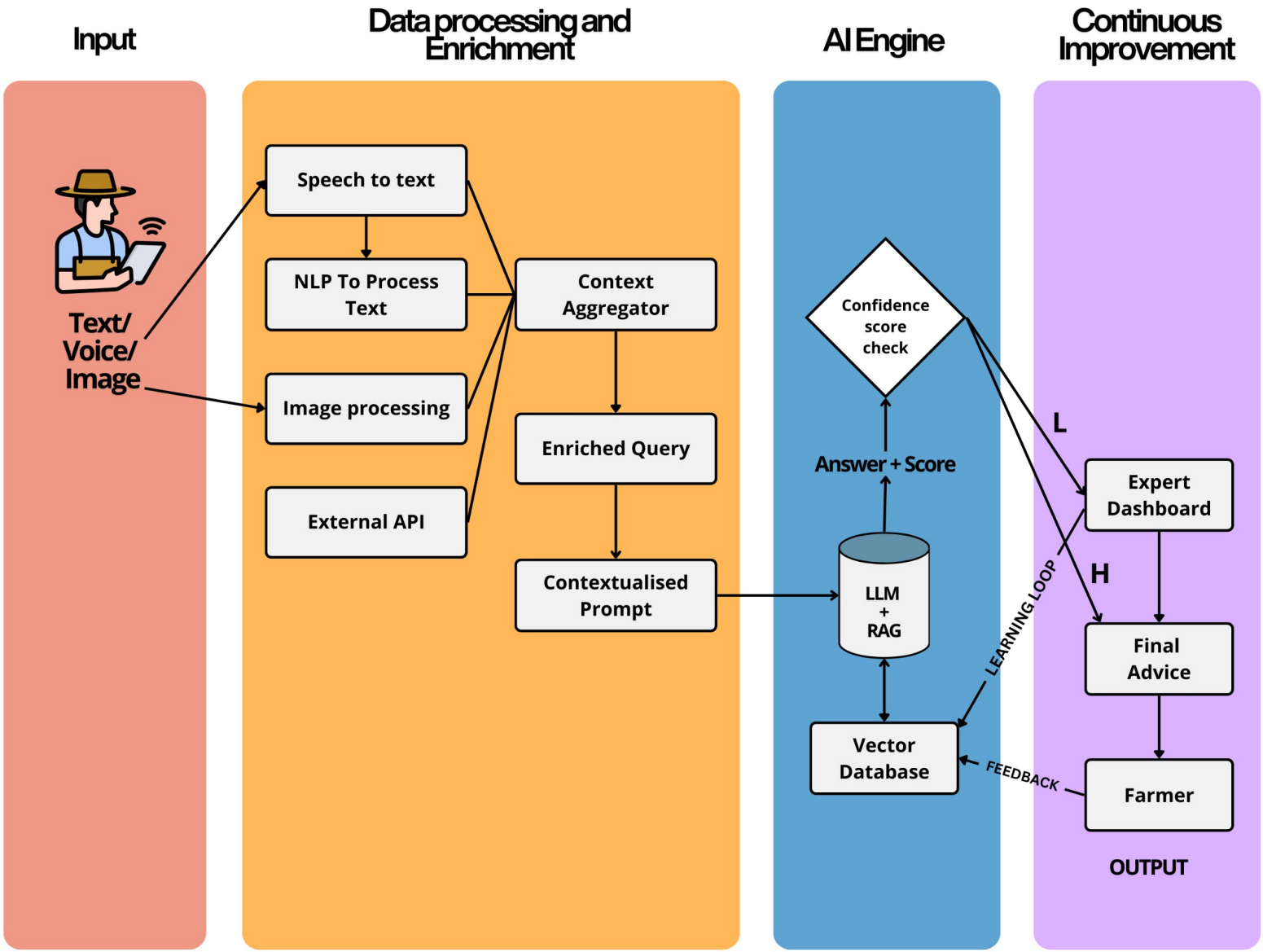
Key Goals

- To build a multimodal input module supporting Malayalam queries via text, voice, and images.
- To implement a context-aware AI engine using **LLM** + agriculture knowledge base for personalized advice.
- To create a **real-time advisory system** that gives instant, reliable, and simple guidance in Malayalam.
- Establish a **learning loop** that improves the knowledge base and **LLM** through farmer feedback and expert validation.
- To develop a mobile application that delivers the advisory system directly to farmers.

Proposed System Architecture



Proposed System Architecture



Methodology

Stage 1: User Interaction & Input Processing

- **Objective**:Build a front-end to capture farmer queries (text, voice, image).
- Implementation Steps
  1. Develop the Mobile Application Interface
     - **App Development**: Use Flutter/React Native for Android & iOS.
     - **UI/UX**: Simple interface with large "Text," "Voice," "Photo" icons in Malayalam.
  2. Integrate the Voice Input Module (Speech-to-Text)
     - **Service**: Integrate a powerful Automatic Speech Recognition (ASR) service with strong support for Malayalam (e.g., Google's Speech-to-Text API or OpenAI's Whisper).
  3. Implement the Image Input Module (Computer Vision)
     - **Service**: Integrate a CV model—start with a pre-trained plant disease model, then train a custom one for local crops and pests.
  4. Handle Direct Text Input
     - **Process**: Capture Malayalam/English text via input field and send to backend for processing.

Methodology

Stage 2: Data Processing & Context Aggregation

- **Objective**: Add personalized and real-time context to farmer queries.
- Implementation Steps
  1. Retrieve Farmer Profile & Historical Data
     - Fetch user details (location, crops, past queries) from PostgreSQL via user ID.
  2. Integrate External Data APIs
     - **Weather Data**: Integrate weather APIs (e.g., OpenWeatherMap/IMD) for real-time local weather conditions.
     - **Agronomic Data**: Build an internal API to provide region- and date-based crop season info.
  3. Construct the Enriched Prompt
     - **Logic**: Develop a backend service that aggregates all the collected data into a single, comprehensive prompt.
     - **Example**: "Farmer in Kerala reports banana leaf spot; humid weather, recent rain—give pesticide and organic control in Malayalam."

Methodology

Stage 3: Core AI Engine (Reasoning & Generation)

- **Objective**: Generate accurate, context-aware answers using AI.
- Implementation Steps
  1. Set Up and Populate the Vector Database
     - **Technology**: Implement a vector database like ChromaDB or Pinecone.
     - **Data Ingestion**: Convert all relevant agricultural documents into text chunks and generate vector embeddings for them.
  2. Implement the Retrieval-Augmented Generation (RAG) Flow
     - **Retrieval**: When the enriched prompt is received, its vector embedding is used to search the vector database for the most relevant chunks of information.
  3. Generate the Response
     - **LLM Integration**: Send augmented prompt to LLM (e.g., Google Gemini) to produce simple Malayalam advisory.
  4. Calculate a Confidence Score
     - **Logic**: Evaluate response reliability using document relevance or LLM self-assessment.

Methodology

Stage 4: Response Delivery & Escalation Logic

- **Objective**: To deliver high-quality answers to the farmer and seamlessly route complex queries to human experts.
- Implementation Steps
  1. Deliver High-Confidence Answers
     - If the confidence score is above a set threshold (e.g., 80%), the generated Malayalam response is sent directly to the farmer's mobile app and displayed in the chat interface.
  2. Develop the Escalation System
     - **Condition**: If the confidence score is below the threshold, the escalation protocol is triggered.
     - **Data Packaging**: The system will package the entire query context—original query (text/voice/image), farmer profile, location, weather, and the AI's low-confidence attempted answer—into a single case file.

Methodology

Stage 4: Response Delivery & Escalation Logic

3 Build the Agricultural Officer Dashboard

- **Interface**: Create a web-based dashboard for registered agricultural officers.
- **Functionality**: The dashboard will display a queue of escalated cases. Experts can review all the context, type their expert advice in Malayalam, and submit it.

4 Route Expert Response

- Once an expert submits their response, the system automatically routes it back to the farmer's app, notifying them that an expert has answered their query.

Methodology

Stage 5: Continuous Improvement (The Learning Loop)

- **Objective**: To create a feedback mechanism that allows the system to learn from both user interactions and expert input, continuously improving its accuracy and knowledge base.

1. Capture User Feedback
   - **Mechanism**: Add simple "Helpful" and "Not Helpful" buttons to each answer displayed in the app.
   - **Data Logging**: Log this feedback against the specific query-response pair for future analysis and model fine-tuning.
2. Integrate Expert Knowledge
   - **Pipeline**: Create an automated pipeline where the expert-provided answers from the escalation dashboard are treated as "gold-standard" data.
   - **Process**: Convert these question–answer pairs into vector embeddings and add them to the database.
   - **Fine-Tuning**: Periodically, this high-quality dataset will be used to fine-tune the LLM, improving its reasoning and generation capabilities for agricultural topics.

Conclusion

Summary

- **Recap:** iFarmAssist gives farmers personalized advice via text, voice, or images, using AI and escalating complex issues to experts.
- **Feasibility:**The project is feasible with a scalable five-stage modular design using Python, Flutter, and RAG.