

# Capital One Launchpad Hackathon: Synopsis

## 1. Team Details

**Team Name:** Byte Bandits

**Team Members:**

1. Anshu Sharma
  2. Ambarish Manna
  3. Hardik Kumar
  4. Amitrajeet Konch
- 

## 2. Theme Details

**Theme Name:** Exploring and Building Agentic AI Solutions for a High-Impact Area of Society: Agriculture – Actionable Hyperlocal Farming Guidance, optimized for real world.

**Theme Benefits:**

**1. Localized Context Aware Assistance:** The theme choice enables understanding of local languages, dialects, and cultural contexts, making the system truly accessible and usable by farmers in their native language and communication mode (chat, voice, photo). This hyper localisation is key to the solution.

**2. Efficient use of Bandwidth:** Reducing data transfer, optimising bandwidth and providing faster response times is crucial in rural areas with intermittent or low-bandwidth internet connectivity. For remote areas, it is paramount that the app work with minimal internet usage.

**3. Robust Multi-Modal Input Handling:** Ability to process voice, photos, typed queries, and sensor data means sellers can interact in whichever way suits them best. This also allows users to use a different method of communication with the app if one proves ineffective.

**4. Integrated Tool Orchestration for Holistic Advice:**

The system orchestrates calls to multiple specialized tools like Weather APIs, Market APIs, Soil API, and domain-specific knowledge bases (RAG), providing sellers with a 360-degree view of factors impacting their decisions.

**5. Seamless Action & Notification Pipeline:** Post-hooks and Action Agents automate follow-up steps: sending SMS reminders, scheduling irrigation, placing input orders, or escalating to human agronomists when uncertain.

---

## 3. Synopsis

**Solution Overview:**

Farmora is a smart assistant for farmers that understands natural questions about farming, finds the most relevant data, and gives clear, practical advice all while being localised and relevant to the region of the user.

Unlike typical AI chatbots that have no surrounding context or tools to help them build better answers to queries, Farmora is built in a way to prioritise hard data and context.

The basic crust of Farmora is:

“Data-driven, grounded, relevant and actionable advice to all farm queries”

Farmora takes a user query, uses relevant tools in its toolset to get the most relevant data to answer that query and based on that data, provides answers in the user’s language.

## **Components & Responsibilities**

### **1. Farmer Client -**

Platforms: Android (primary), web, basic feature phone (IVR/SMS).

Inputs: typed question, voice note, photo(s), simple forms (e.g., record harvest).

Local cache for profile, recent advisories, fallback messages.

### **2. Speech to Text -**

Functions: language detection, speech→text, image compression, quick profanity filter.

Tech: Whisper-Small

Output: transcript in English + Native Language

### **3. Local Classifier / Router (stateless microservice) -**

Multi-label intent detection: weather, market, pest, scheme, irrigation, accounting, order.

Tech: Llama/ Groq.

### **4. Tool Orchestrator -**

Calls relevant external/internal tools based on classifier:

Weather service (regional forecasts + alerts).

Market price DB (regional mandi).

Vector DB for RAG (PDFs, scheme docs, agronomy guides).

Pest/disease vision service (local model + RAG).

### **5. Middleware Hooks (pluggable) -**

Pre-tool hooks: profanity, PII scrub, quota check.

Tool output validation: schema checks, anomaly detection, confidence gating.

Context compression: token budget enforcement, pruning rules.

Term mapping: local names ↔ canonical names (bidirectional).

Post-LLM hooks: fact-check numbers, safety rules (no illegal pesticides), tone/localization.

## 6. Context Generator -

Builds minimal, structured prompt: variables + short bullet fact

Keeps fixed system prompt (template) + variable block with compact JSON -> human-readable bulleted summary.

Example structure (max ~400 tokens typical):

SYSTEM: You are a helpful agricultural advisor. Use only supplied context. No hallucinations.

CONTEXT:

- Farmer: Ramesh (ID: 123), Location: Mandya, Karnataka (12.5N,76.9E)

- Crops: Maize (main), Paddy (secondary)

- Stage: Maize - V6

- Weather (next 3 days): Rain 60% day3, Tmax 33C...

- Soil moisture: 28% (sensor id)

- Market: Local mandi price maize 1700₹/q

- RAG: ICAR doc: "blight treatment: apply X" (ref id: vdb:345)

USER QUERY: "Should I irrigate maize tomorrow?"

## 7. LLM (semantic understanding)-

Purpose: synthesize, reason, and produce farmer-facing actions/advice.

System instructions: "Only use supplied context. If missing info, say what you need. Provide 1–3 concrete actions, with rationale and confidence score (0–1)."

Keep the LLM call to one per query where possible.

## 8. Post Hooks & Action Agents -

Post-checks: numeric sanity, banned recommendations, pesticide safety check.

Action Agents (event-driven microservices): schedule reminders, send SMS/IVR, place orders via dealer APIs.

## 9. Storage & Indexes -

Farmer profile DB (encrypted): profile, farm geometry, equipment, crop calendar.

Vector DB for RAG: hashed docs, embeddings, metadata.

Time-series DB: market prices, weather history, sensor readings.

Audit logs: all prompts, responses, tool outputs for debugging & compliance.

**Technical Stack:**

Mobile App: Java, Kotlin, Jetpack Compose, Retrofit/ Ktor, Appwrite Auth SDKs, Gradle  
Server Development: Express JS, Flask, Fast API  
LLM APIs: Groq (lowest latency) - Llama  
Local Models: Whisper, IndicTrans  
Database: ChromaDB (RAG), MongoDB, Redis (Caching)  
APIs: OpenWeather API, geopy  
Datasets: AgMarkNet(Crop mandi prices), ICAR Reports (Soil health, crop advisories), PM-KISAN (Finance and scheme linking)

## Decision Rationale:

### 1. Efficient AI Model Usage

- **Constraint:** How to provide fast responses while maintaining quality of response?
- **Decision:**
  - **Offloaded On-Server Processing:**
  - **LLM Call Optimization:** Use a cloud-based LLM for the reasoned advice generation, and provide structured prompt to reduce unnece

### 2. Multi-Label Intent Detection & Classification

- **Constraint:** The system needs to accurately classify and route the queries to the right resources while handling ambiguity or low-confidence queries.
- **Decision:**
  - **Local Classifier:** Using **DistilBERT/MiniLM**, lightweight fine-tuned models are employed to classify user queries into multiple categories like weather, market prices, pests, schemes, and more. This helps in efficiently directing the query to relevant external/internal tools.
  - **Fallback Mechanism:** Low-confidence queries are routed to an LLM-based fallback or flagged for clarification, minimizing errors and ensuring high-quality responses.

### 3. Action Agents & Task Automation

- **Assumption:** Some queries might require follow-up actions (e.g., reminders, placing orders, or scheduling tasks with agronomists).
- **Constraint:** A certain set of actions need to be triggered automatically based on user queries.
- **Decision:**
  - **Event-Driven Action Agents:** If the query requires scheduling a reminder, sending an SMS, or placing an order with a local dealer, **action agents** trigger those actions automatically. This minimizes the farmer's need to manually track and execute tasks.
  - **Escalation Protocol:** If the system is uncertain about the response or needs further expert validation, it flags the query for escalation to a human expert, ensuring high-quality advice.

### 4. Scalable and Modular Tool Orchestration

- **Assumption:** The system will need to access a variety of external data sources and tools (weather, market prices, pest/disease detection, IoT sensor data).
- **Constraint:** The tools must be orchestrated efficiently to minimize latency, enforce rate limits, and ensure reliability.
- **Decision:**
  - **Modular Tool Orchestrator:** This component centralizes the decision-making for which external tools to call based on the classifier's output. For example, it calls **weather services**, **market price databases**, and **pest vision services** only when necessary, reducing unnecessary calls to external APIs and speeding up the response process.
  - **Caching:** Local caching mechanisms ensure that frequently requested data (e.g., weather forecasts) doesn't require redundant API calls, thereby saving on costs and reducing response time.

### Innovation Highlights:

Farmora is a resource-conscious AI agent ecosystem tailored for agricultural communities. It combines hyper-local environmental data, domain-specific knowledge, and multi-step reasoning to deliver actionable, context-aware insights while minimizing reliance on costly API calls and internet connectivity. Unlike generic AI assistants, it's purpose-built to integrate live weather, crop health, and local farming practices into decision-making. This makes it both affordable and practical for real-world rural use.

- Token-efficient → minimizes LLM calls by handcrafting prompts and using tool data
- Context-rich → blends real-time weather, soil, crop data with AI reasoning
- Domain-specific → built for agriculture, not general chat
- Agentic → it doesn't just answer, it acts, plans, and monitors over time

### Feasibility and User-Friendliness:

#### 1. Realism

**Feasibility:** The tech stack (vector DB, local preprocessing, minimal LLM calls) is realistic with current open-source tools (FAISS, Ollama, Groq API).

**Data availability:** Weather APIs (Open-Meteo, WeatherAPI) are free for basic use. Agricultural datasets exist (FAO, government open-data portals).

**Hardware constraints:** Since most services are run on-server and not device specific, hardware concern drop to almost none.

**Verdict:** Highly realistic for a prototype, with careful scope control.

#### 2. User-Friendliness

**Adoption barrier:** Farmers generally won't care about "agents" or "RAG", they'll care about "I ask, I get a useful answer". Simplicity is key.

**Interface design:** Is multilingual, possibly voice-first for inclusivity. Mobile-first approach. Reduces barrier of entry for farmers.

**Trust:** Agricultural advice affects livelihoods meaning the answers need clear, simple explanations ("why" this is the recommendation) to build trust. The solution goes through multi step reasoning and multiple checks and balances to reach the answer.

**Learning curve:** Any farmer that has been using apps like ChatGPT, Gemini etc already will have no problem adopting Farmora as it has similar interface. Even those who don't will not be completely lost due to the intuitive UI.

**Verdict:** With the right UX decisions (voice + local language + simple flows), adoption could be strong.

#### 3. Operational Efficiency

**Token efficiency:** Classification, preprocessing, and context assembly using traditional models will cut API costs.

**Speed:** Vector DB + lightweight models for first-pass analysis = fast responses even on low-end devices.

**Verdict:** Very efficient.

#### 4. Long-Term Success Potential

**Social impact:** Bridges digital divide in agriculture by providing relevant, timely, and localized help.

**Growth potential:** Can expand into other rural services like market prices, government scheme info, supply chain optimization

**Retention:** If users see improved yields, they'll keep using it; word-of-mouth adoption in rural communities is strong.

**Risk:** The biggest threat is data trustworthiness. Poor or wrong advice could ruin credibility fast making strong verification loops an important part of the architecture.

---

#### 4. Methodology/Architecture Diagram

- i. App Architecture

[https://drive.google.com/file/d/1ai5e60r7x-9l3450G\\_iEaA4Y5dOJWuLf/view?usp=sharing](https://drive.google.com/file/d/1ai5e60r7x-9l3450G_iEaA4Y5dOJWuLf/view?usp=sharing)

- ii. AI Pipeline Architecture

<https://drive.google.com/file/d/1pamc2Uld-Kgprqd4qGa-5kurbKONAuWN/view?usp=sharing>

---