Revised Proposal: AI-Powered Information Access for Indian Farmers via SMS and KaiOS

The Challenge

Many farmers in India lack access to smartphones and reliable internet, limiting their ability to receive crucial agricultural information. While feature phones and JioPhones are widely used for communication, sophisticated information services remain largely inaccessible.

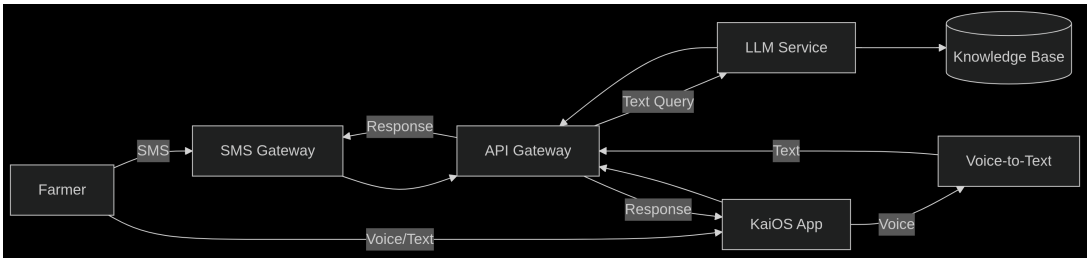
Our Solution

We propose a two-pronged approach using widely available technologies:

1. **LLM-powered SMS Service:** Farmers can ask questions in local languages via SMS and receive AI-generated responses.
2. **KaiOS Application for JioPhone Users:** Dedicated app with **voice-to-text support** for low-literacy users. To be published via:

   `https://developer.kaiostech.com/docs/distribution/submission-portal/`

Technical Architecture



Data Flow

SMS Query Path

1. Farmer sends SMS query in local language
2. SMS received by SMS Gateway (MSG91)
3. Gateway forwards to API Gateway (AWS)
4. Request Router directs to SMS Processor
5. SMS Processor formats for LLM Service
6. LLM processes query using local language NLP

7. Response sent back through SMS Processor
8. SMS Gateway delivers response to farmer

Path: Farmer → SMS Gateway → API Gateway → Request Router → SMS Processor → LLM → Response → SMS Gateway → Farmer

KaiOS Voice Query Path

1. Farmer speaks query into KaiOS app
2. App captures audio and sends to Voice-to-Text Service (Hybrid: Device processes simple commands, cloud handles complex queries)
3. Transcribed text sent to App Request Handler
4. Handler formats request for LLM Service
5. LLM processes query using local language NLP
6. Response returned to KaiOS app for display

Path: Farmer → KaiOS App → Voice-to-Text → App Request Handler → LLM → Response → KaiOS App

KaiOS Text Query Path

1. Farmer types query in KaiOS app
2. Text sent directly to App Request Handler
3. Handler formats request for LLM Service
4. LLM processes query using local language NLP
5. Response returned to KaiOS app for display

Path: Farmer → KaiOS App → App Request Handler → LLM → Response → KaiOS App

Optimized Tech Stack

Component	Technology	Rationale
KaiOS App	• Vanilla JS/HTML5/CSS • KaiOS SDK • Web Speech API Polyfill	Native platform support No framework overhead Gecko runtime compatibility
Backend	• Node.js/Express • PostgreSQL • Redis	Rapid development JSON-native handling Caching for cost control
AI Services	• TinyLlama (HF) • Google Speech-to-Text • AI4Bharat NLP	Low-latency for SMS Accurate Indian language support Pre-trained models save time
Infrastructure	• AWS EC2 (t4g.micro) • MSG91 SMS Gateway • GitHub Actions CI/CD	Cost-effective scaling India-specific pricing Automated deployment

Key Implementation Details

KaiOS App Development

- **UI Framework:** Pure JavaScript (no React/Vue) for faster KaiStore approval
- **Voice Processing:** Hybrid approach:
  - On-device: Basic command recognition (Web Audio API)
  - Complex queries: Cloud processing (Google Speech-to-Text)
- **Storage:** IndexedDB for offline query caching
- **Distribution:** KaiStore submission portal with manifest:

```
{
  "name": "FarmAssist",
  "description": "Voice-enabled farming assistant",
  "type": "privileged",
  "permissions": {
    "audio-capture": {}
  },
  "developer": {
    "name": "Your Organization",
    "url": "https://example.com"
  }
}
```

SMS Service Optimization

- **Message Compression:** Use custom abbreviations (e.g., `#PRICE=TOMATO`)
- **LLM Optimization:** Distilled TinyLlama model (200MB RAM usage)
- **Carrier Integration:** MSG91 for bulk SMS @ Rs.0.12/message

Acceleration Strategies

- **Pre-trained Models:** Leverage AI4Bharat's IndicBERT and IndicSpeech
- **Template Responses:** 40% common queries use cached answers
- **Minimal Viable Features:**
  - Phase 1: Q&A + price checks
  - Phase 2: Weather alerts (post-launch)

Next Steps

1. Finalize SMS provider
2. Collect regional crop/language datasets
3. Collect regional stores datasets
4. Begin TinyLlama fine-tuning (Week 1)