AGRIGURU: AN AI-POWERED MULTILINGUAL MULTI-AGENT FARMER ASSISTANT FOR END-TO-END AGRICULTURAL DECISION SUPPORT

[1]Prathan N, [2]Praveen Kumar R, [3]Suriyaa D, [4]V. Nagamalini
[123]Student, [4]Assistant Professor
Department of Computer Science and Engineering (Artificial Intelligence and Data Science)
Sri Venkateswara College of Engineering, Chennai, India

Abstract: *Agriculture in India remains deeply dependent on timely, contextual decision support across the full cultivation lifecycle. Yet, farmers, particularly smallholders, face a fragmented digital landscape of single-purpose tools that are largely text-centric, English-dependent, and poorly integrated with real-time data. This paper presents AgriGuru, an AI-powered, voice-first, multilingual farmer assistant that unifies crop recommendation, irrigation planning, crop disease support, government scheme identification, and mandi-based market advisory into a single platform. AgriGuru employs a Google Agent Development Kit (ADK) multi-agent backend orchestrated by a root agent that delegates to seven specialized agricultural sub-agents. A Flutter-based mobile client provides voice, text, and image interaction backed by Firebase authentication and Firestore profile management. A machine learning crop recommendation microservice using Naive Bayes on a 2,200-sample dataset of soil and climate features achieves an accuracy of 99.55%. Market intelligence is delivered through a custom Agmarknet data pipeline that ingests daily mandi prices from the national open-data API, covering 36 states and union territories, up to 117 commodity types, and approximately 2,000 price records per day, and exposes nine structured query tools via a Model Context Protocol (MCP) server. Location-aware services are provided through a Google Maps MCP integration. The entire backend is deployed on Google Cloud Run with Gemini 2.5 Pro as the underlying language model. Experimental results and system analysis demonstrate that AgriGuru provides comprehensive, accessible, and scalable agricultural intelligence, making it a viable platform for digital farmer empowerment at scale across India.*

IndexTerms *- precision agriculture, multi-agent systems, large language models, crop recommendation, voice-first interface, Naive Bayes, Google ADK, Gemini, MCP, farmer decision support.*

I. INTRODUCTION

Agriculture is a cornerstone of the Indian economy and supports the livelihoods of a large share of the population. Small and marginal farmers routinely navigate complex decisions related to crop selection, fertiliser management, disease identification, irrigation scheduling, and post-harvest market timing, often without adequate or timely expert support. Generative artificial intelligence is increasingly being explored to bridge this advisory gap in Agriculture 4.0 settings [3]. However, existing digital tools either address a single task in isolation or require proficiency in English, limiting their reach among low-literacy and regional-language speaking farmers.

The rapid maturation of large language models (LLMs) [10], cloud-native services, and mobile computing creates an opportunity to close this gap. Multimodal LLMs now support text, voice, and image inputs, enabling natural interaction patterns that align with field conditions where voice is the preferred modality [1]. Multi-agent architectures [11] allow specialized intelligence to be modularly composed behind a unified conversational interface.

This paper presents AgriGuru, a full-stack agricultural assistant that addresses these requirements. The key contributions of this work are: (i) a seven-agent hierarchical architecture covering the complete agricultural lifecycle, orchestrated by a root agent using the Google ADK framework; (ii) a voice-first Flutter client with multilingual support, Firebase-backed authentication, and profile-aware session management; (iii) a crop recommendation microservice achieving 99.55% accuracy using Naive Bayes on standardized soil and climate features; (iv) a modular market intelligence pipeline that ingests daily Agmarknet data and exposes it through a custom MCP server; and (v) a location-aware advisory layer using a custom Google Maps MCP integration.

II. RELATED WORK

Sapkota et al. [1] present a comprehensive review of multimodal large language models (MM-LLMs) in agriculture, establishing the necessity of MM-LLMs for complex agricultural decisions while identifying persistent gaps in domain-specific data quality, integration with existing systems, and robust training datasets. AgriGuru addresses this integration gap by unifying multiple agricultural decision functions behind a single multimodal assistant.

In crop disease detection, Mohanty et al. [6] demonstrated deep convolutional neural networks on the PlantVillage dataset with over 99% accuracy, but performed classification only without remedy guidance. Zhu et al. [2] combined multimodal AI with a large language model for potato disease detection and prevention, reaching 98.43% accuracy while generating preventive recommendations. AgriGuru adopts this detection-plus-remedy paradigm within a disease sub-agent.

Sai et al. [3] review generative AI applications in Agriculture 4.0, while Fang et al. [5] propose Agri-LLM for emission data analytics, demonstrating LLM capability over agricultural time-series data. Wyckhuys et al. [4] critically evaluate web-grounded

generative AI and report hallucination and data fabrication without human oversight, motivating AgriGuru's design choice to ground agents in structured tools and verified data. For crop recommendation, Doshi et al. [7] and Pudumalar et al. [8] established machine learning and ensemble approaches on soil and climate features. Multi-agent LLM frameworks such as MetaGPT [9] and the broader survey of LLM-based agents [11] inform the orchestration design realised here through the Google ADK [13].

Table 1 Comparison of related works and research gap

Work	Focus	Limitation	Relation to AgriGuru
Sapkota et al. [1]	MM-LLM review	Review only; integration gap	Realises integrated MM-LLM assistant
Zhu et al. [2]	Multimodal disease	Single crop family	Used in disease sub-agent
Sai et al. [3]	GenAI in agriculture	Conceptual; no system	Deploys GenAI across lifecycle
Wyckhuys et al. [4]	GenAI evaluation	Warns of hallucination	Grounds agents in tools/RAG
Mohanty et al. [6]	CNN disease detect	Classification only	Adds LLM remedy narrative
Doshi/Pudumalar [7][8]	Crop recommendation	No deployment/profile	Live API + profile context
Hong et al. [9]	Multi-agent LLM	Not agriculture-specific	Applied to agri domain

III. SYSTEM ARCHITECTURE

A. Overview

AgriGuru is structured as a layered, cloud-native system comprising a mobile client, a multi-agent backend, supporting microservices, and external data integrations (Fig. 1). The farmer interacts with the Flutter app through voice, text, or image inputs. The app authenticates users via Firebase and persists farmer profiles in Firestore. All conversational queries are routed to the ADK backend hosted on Google Cloud Run, which orchestrates specialized agents and returns streamed responses.

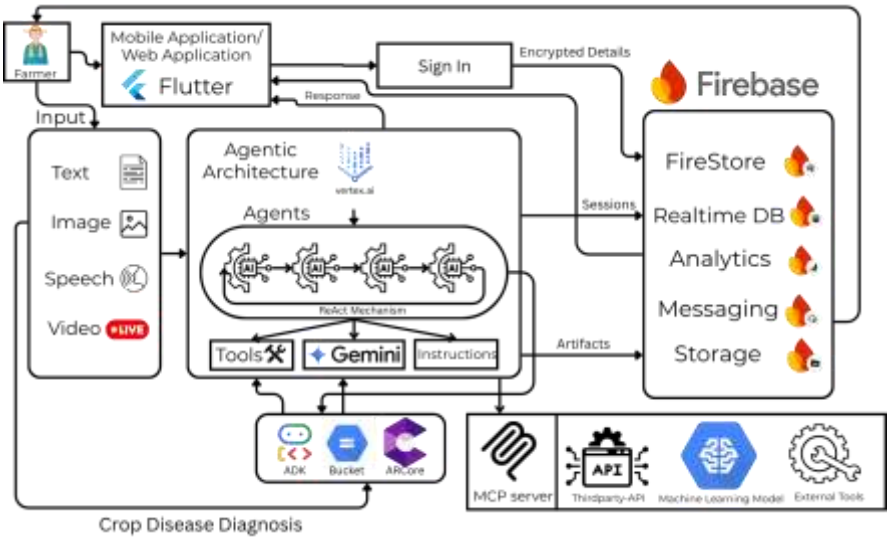

fig. 1 Agriguru overall system architecture

B. Multi-Agent Architecture

The backend uses a hierarchical multi-agent design. A root AgriGuru agent receives user queries enriched with the farmer's profile state (name, location, land size, language, coordinates, previous crops) and delegates to one or more of seven specialized sub-agents through the ADK agent-tool mechanism. The roles of these agents are summarised in Table 2.

Table 2 Agent roles in the agriguru multi-agent system

Agent	Responsibility
Root Agent	Query routing, tool coordination, session context
Extract Soil Data	Soil health data extraction and interpretation
Crop Identifier	Crop recommendation via ML service and agronomic context
Seed Identifier	Seed variety discovery and nearby shop guidance
Scheme Identifier	Matching government schemes to profile via RAG
Irrigation Planner	Crop-stage and weather-aware scheduling
Disease Detector	Image-grounded disease interpretation and remedy
Market Advisor	Mandi price comparison and location-aware advisory

C. Technology Stack

The client is built with Flutter using the Provider state-management pattern, with Firebase Authentication and Cloud Firestore. The backend is built on the Google ADK with FastAPI and reasons using Gemini 2.5 Pro, with Cloud SQL for session persistence, Google Cloud Storage for artifacts, and Vertex AI for memory and retrieval-augmented scheme search. Market and location intelligence are provided by custom Agmarknet and Google Maps MCP services, crop recommendation by a Python machine learning microservice, and all server components are deployed on Google Cloud Run.

D. Farmer Profile and Session Context

Upon authentication, the farmer's Firestore profile is serialized into the ADK session state. This allows all sub-agents to access personalized context without requiring the user to re-specify details on every query, enabling genuinely session-aware advisory responses.

IV. IMPLEMENTATION

A. Flutter Mobile Client

The Flutter client manages the full user journey: language selection, OTP-based registration, profile creation, home dashboard, conversational chat, scheme browsing, and irrigation plan visualization. Text, voice, and image inputs are supported, and Server-Sent Events (SSE) streaming provides real-time response delivery. Voice-origin responses are automatically synthesized back to speech via TTS.

B. ADK Backend and Orchestration

The ADK backend is implemented in Python. The root agent is configured with the Gemini 2.5 Pro model, MCP tool connections, artifact management tooling, and sub-agents as callable agent-tools. On receiving a query, the root agent determines which sub-agents and tools to invoke and streams the response back to the client.

C. Agmarknet Market Data Pipeline

Market intelligence relies on a two-part pipeline. The Agmarknet Invoker paginates the api.data.gov.in Agmarknet API in batches of 1,000 records with exponential-backoff retry logic and uploads a date-stamped CSV to a Google Cloud Storage bucket; a sample run produced 2,000 records spanning 18 states and 117 commodity types. The Agmarknet MCP server, deployed on Cloud Run, reads these files and exposes nine structured query tools, with a three-day fallback window ensuring continuous advisory when the latest file is unavailable.

D. Google Maps MCP Integration

A custom Google Maps MCP server wraps the Google Maps Platform SDK and exposes driving directions, geocoding, nearby place search, and distance computation across four transport modes. The Crop Market Advisor agent combines the get_best_price Agmarknet tool with the get_directions Maps tool to rank mandis by both modal price and travel distance, enabling profit-optimized sell advisory not found in existing systems.

E. Scheme, Irrigation, and Disease Modules

The Scheme Identifier sub-agent uses a Vertex AI RAG knowledge base of government scheme documents ranked by relevance to the farmer's profile. The Irrigation Planner generates crop-stage-aware schedules retrieved and visualised in the app. For disease support, an uploaded crop image is stored as an artifact and interpreted by the Disease Detector sub-agent using multimodal Gemini analysis, returning a diagnosis with remedies.

V. CROP RECOMMENDATION MODEL

A. Dataset and Preprocessing

The crop recommendation subsystem is trained on a dataset of 2,200 records balanced across 22 crop classes, each described by seven features: soil nitrogen (N), phosphorus (P), potassium (K), temperature, humidity, soil pH, and rainfall. The data has no missing values or duplicates. Preprocessing label-encodes the crop target, standardizes features using StandardScaler, and splits the data 80:20 into training and evaluation sets. Standardization is defined as:

$$z = (x - \mu) / \sigma$$

where x is the original feature value, μ is the feature mean, and σ is the standard deviation.

B. Model and Formulation

Ten classifiers were evaluated, and Gaussian Naive Bayes was selected for deployment on the basis of its accuracy and very low inference latency. The classifier predicts the crop class C_k that maximizes the posterior probability:

$$\hat{C} = \text{argmax}_k \, P(C_k) \prod_i P(x_i \mid C_k)$$

where the per-feature likelihood follows a Gaussian distribution estimated from the training data. Model performance is reported using accuracy, precision, recall, and F1-score:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

$$\text{Precision} = TP / (TP + FP), \quad \text{Recall} = TP / (TP + FN)$$

$$F1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

C. Evaluation Results

The trained Gaussian Naive Bayes model achieves 99.55% accuracy on the 440-sample held-out split, with a macro-averaged F1-score of 1.00. Only two misclassifications were observed, both rice samples predicted as jute, which is agronomically meaningful

as both crops grow in humid, high-rainfall environments with similar nitrogen requirements. Key metrics are summarised in Table 3.

Table 3 crop recommendation model evaluation summary

Metric	Value
Model	Gaussian Naive Bayes
Accuracy	99.55%
Macro Avg. F1-Score	1.00
Train / Test Split	80 : 20
Input Features	7
Target Classes	22
Total Misclassifications	2

The saved artifacts (naive_bayes_model.pkl, scaler.pkl, label_encoder.pkl) are packaged into a FastAPI microservice deployed on Cloud Run, exposing a POST /predict endpoint that accepts the seven features and returns the predicted crop in a suitable_crop field.

VI. RESULTS AND DISCUSSION

The system demonstrates end-to-end integration of OTP authentication, multilingual voice interaction, profile-aware sessions, multi-agent orchestration, crop recommendation, irrigation planning, scheme advisory, and market advisory. Representative scenarios were validated, including farmer login, chat session creation, voice query with spoken reply, crop image diagnosis, irrigation plan retrieval, scheme matching, crop recommendation API calls, and a market advisory query that identified a high-price mandi with distance awareness.

The modular multi-agent architecture provides clear advantages in extensibility and maintainability, since each sub-agent can be updated independently. MCP-based tool integration decouples data sourcing from AI reasoning, and SSE streaming delivers responsive perceived latency. Table 4 contrasts AgriGuru with existing agricultural tools.

Table 4 comparison of agriguru with existing agricultural tools

Feature	AgriGuru	Portal	Standalone ML	Chatbot
Voice-first	Yes	No	No	Partial
Multilingual	Yes	Partial	No	Partial
Crop Rec.	Yes	No	Yes	No
Disease Support	Yes	No	Partial	No
Market Advisory	Yes	Partial	No	No
Location-aware	Yes	No	No	No
End-to-end unified	Yes	No	No	No

AgriGuru supports the United Nations Sustainable Development Goals, notably SDG 8 (Decent Work and Economic Growth) through improved farmer income, SDG 9 (Industry, Innovation and Infrastructure) through cloud-native agri-tech, and SDG 13 (Climate Action) through data-informed irrigation and planting decisions.

VII. CONCLUSION

This paper presented AgriGuru, a cloud-native, voice-first, multilingual agricultural assistant built on a seven-agent ADK hierarchy backed by Gemini 2.5 Pro. The system unifies crop recommendation, disease detection, irrigation planning, government scheme identification, and market advisory into a single personalized platform. The crop recommendation subsystem achieves 99.55% accuracy using Gaussian Naive Bayes, and the Agmarknet and Google Maps MCP integrations provide real-time, location-aware advisory not found in existing systems. Future work includes full offline support, broader Indian language coverage, IoT soil-sensor integration, buyer-connect workflows, and field validation with farmers.

ACKNOWLEDGMENT

The authors thank the faculty and management of Sri Venkateswara College of Engineering, Chennai, for their guidance and institutional support, and acknowledge the open-data infrastructure provided by the Government of India through api.data.gov.in (Agmarknet).

REFERENCES

[1] R. Sapkota et al., "Multi-modal LLMs in agriculture: A comprehensive review," IEEE Transactions on Automation Science and Engineering, 2025, doi: 10.1109/TASE.2025.3612154.

[2] H. Zhu, W. Shi, X. Guo, S. Lyu, R. Yang, and Z. Han, "Potato disease detection and prevention using multimodal AI and large language model," Computers and Electronics in Agriculture, vol. 229, art. no. 109824, Feb. 2025.

[3] S. Sai, S. Kumar, A. Gaur, S. Goyal, V. Chamola, and A. Hussain, "Unleashing the power of generative AI in agriculture 4.0 for smart and sustainable farming," Cognitive Computation, vol. 17, no. 1, art. no. 63, Feb. 2025.

[4] K. A. G. Wyckhuys et al., "Human versus machine: Can generative AI anticipate insect biological control outcomes?," Computers and Electronics in Agriculture, vol. 242, art. no. 111317, Feb. 2026.

[5] L. Fang et al., "Agri-LLM: Prompt-based large language model for emission data analytics in smart agriculture," IEEE Internet of Things Journal, vol. 12, no. 23, pp. 49186-49197, Dec. 2025.

[6] S. P. Mohanty, D. P. Hughes, and M. Salathe, "Using deep learning for image-based plant disease detection," Frontiers in Plant Science, vol. 7, art. no. 1419, Sep. 2016.

[7] Z. Doshi, S. Nadkarni, R. Agrawal, and N. Shah, "AgroConsultant: Intelligent crop recommendation system using machine learning algorithms," in Proc. ICCUBEA, Pune, India, 2018, pp. 1-6.

[8] S. Pudumalar, E. Ramanujam, R. H. Rajashree, C. Kavya, T. Kiruthika, and J. Nisha, "Crop recommendation system for precision agriculture," in Proc. ICoAC, Chennai, India, 2017, pp. 32-36.

[9] S. Hong et al., "MetaGPT: Meta programming for a multi-agent collaborative framework," in Proc. ICLR, Vienna, Austria, 2024.

[10] T. Brown et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 1877-1901.

[11] L. Wang et al., "A survey on large language model based autonomous agents," Frontiers of Computer Science, vol. 18, no. 6, art. no. 186345, 2024.

[12] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[13] Google LLC, "Agent Development Kit (ADK) documentation," 2024. [Online]. Available: https://google.github.io/adk-docs/