Applied Human Factors and Ergonomics (AHFE2025), Vol. 199, 2025, 1166–1181
https://doi.org/10.54941/ahfe1006928

Toward Climate-Smart Agriculture: An Environmental, Social, and Governance (ESG)-Centric Edge AI Architecture for Smallholder Farmers

Chandrasekar Vuppalapati[1], Anitha Ilapakurti[2], Shruti Vuppalapati[2], koduru Rajasri[2], Sharat Kedari[3], and Jaya Vuppalapati[4]

[1]Computer Engineering *(CMPE Department)*, San Jose State University, San Jose, USA
[2]Hanumayamma Innovations and Technologies, Inc., Fremont, USA
[3]Hanumayamma Innovations and Technologies PVT Limited, Hyderbad, India

ABSTRACT

This paper introduces Edge Fluent, a novel Environmental, Social, and Governance (ESG)-oriented Internet of Things (IoT) edge architecture designed to empower smallholder farmers in the agri-dairy sector through inclusive artificial intelligence (AI). Central to the system is a fine-tuned, multi-label DistilBERT model deployed on low-power, resource-constrained edge devices, enabling real-time ESG classification, multilingual translation of regulatory content, and actionable intervention support. By addressing the pervasive barrier of language accessibility—particularly among non-English-speaking and low-literacy farming communities—the platform ensures equitable delivery of ESG intelligence and climate-resilient decision-making. Validated through live deployments on dairy farms equipped with Class 10 veterinary sensors and farmer interfaces in native dialects, the solution facilitates methane emission tracking, rumination-based health monitoring, feed optimization, and Scope 1–3 emissions traceability. Designed for offline inference and multimodal sensor inputs, the system reinforces ESG compliance, sustainable certification, and data harmonization across distributed farm networks. Ultimately, this architecture advances UN Sustainable Development Goals (SDGs) by embedding linguistic inclusion at the core of climate-smart agriculture and redefining sustainability through equitable technological integration. Persistent rural-to-urban migration and generational shifts in labor have precipitated a critical shortage of human capital across global agriculture and dairy sectors, disproportionately impacting smallholder farmers. This paper introduces a novel framework for deploying *Agentic AI*—autonomous, self-improving digital agents capable of learning farmer preferences, regional agronomic conditions, and value chain dynamics—to mitigate labor deficits, facilitate knowledge transfer, and enhance productivity in resource-constrained agricultural systems. The proposed solution utilizes modular, multilingual AI agents deployed on mobile and edge computing platforms, optimized for low-bandwidth rural environments. These agents integrate voice-driven interfaces to support semi-literate users and are tailored to local agronomic practices, ensuring cultural and contextual relevance. The economic implications are substantial. According to estimates from the World Bank and the Food and Agriculture Organization (FAO), labor shortages in agriculture result in approximately USD 56 billion in annual productivity losses, with smallholders constituting the majority of affected stakeholders. Our analysis demonstrates that Agentic AI can improve decision-making in key areas such as crop selection, irrigation scheduling, and input optimization, leading to yield increases of 5–15%. This translates to potential global economic gains of USD 12–20 billion annually. This work advocates for a paradigm shift in agricultural technology, wherein AI agents function not merely as tools but as cognitive collaborators—capable of learning, adapting, and augmenting the reach of conventional agricultural extension services. We position Agentic AI as foundational digital infrastructure for building resilient, inclusive, and future-ready food systems.

Keywords: Automated car washing, Vehicle defect detection, Lightweight RetNet, Dynamic channel attention, Multi-scale feature fusion, Knowledge distillation

INTRODUCTION

The global shortage of skilled labor in dairy and agriculture, particularly among smallholder farmers—has resulted in annual productivity losses exceeding USD 56 billion across developing economies (FAO, 2021; World Bank, 2022). This crisis is driven by a confluence of factors: aging workforces, limited access to agronomic expertise, rising labor costs, and fragmented supply chains. For instance, over 70% of dairy farms report difficulty in recruiting skilled workers, with labor costs rising by 15% in the past five years and turnover rates reaching 30% annually (FAO, 2021). In the UK alone, nearly 200 dairy farmers exited the industry in the year leading up to April 2025, citing labor shortages as a primary concern (McKinsey Global Institute, 2022).

Agentic artificial intelligence (AI)—defined as autonomous, context-aware agents—offers a transformative solution to these systemic challenges. Unlike traditional automation, Agentic AI systems can reason, adapt, and execute multistep tasks with minimal human intervention. When strategically deployed, these agents can augment the agricultural labor force, deliver personalized agronomic guidance, and bridge the rural knowledge divide. Real-world applications include autonomous irrigation scheduling, pest monitoring via smart traps, and dynamic task coordination with farm workers (Vuppalapati, 2025; Vuppalapati, 2021).

Empirical studies suggest that even a modest 10% improvement in crop yield enabled by intelligent advisory systems could generate USD 12–20 billion in additional global value annually (IFAD, 2023; CGIAR AI for Agriculture, 2022). Moreover, the Agentic AI market itself is projected to grow from USD 7.28 billion in 2025 to USD 41.32 billion by 2030, reflecting a compound annual growth rate (CAGR) of 41.48% (Vuppalapati, 2025). By learning from farmer preferences, local crop profiles, climate variables, and economic constraints, Agentic AI can foster inclusive, scalable rural development and counteract the erosion of agricultural expertise due to urban migration.

The remainder of this paper is organized as follows: Section II introduces the Agentic AI architectural framework. Section III details the machine learning models underpinning our Agentic AI system. Section IV discusses key design and implementation considerations. Section V presents a real-world case study. Finally, Section VI concludes the paper and outlines directions for future research.

AI SYSTEMS AND HIERARCHICAL ARCHITECTURES

AI Agents

The emergence of ChatGPT marked a pivotal moment in artificial intelligence (AI), showcasing the capabilities of large language models (LLMs) and accelerating the integration of AI into mainstream workflows. At its core, an AI agent is a goal-driven software entity capable of executing tasks autonomously or semi-autonomously. These agents leverage LLMs trained on vast corpora to interpret natural language, reason over context, and perform actions with minimal human intervention.

Modern agentic patterns include Retrieval-Augmented Generation (RAG) agents, SQL agents, and hierarchical agents, which may operate within

single-agent or multi-agentic frameworks (LangChain, 2023a). A copilot is a specialized form of AI agent designed to assist users through natural language interactions. Unlike fully autonomous systems, copilots function as collaborative companions—offering suggestions, summarizing content, generating code, and streamlining decision-making across domains such as software development, legal analysis, and enterprise operations.

Recent advancements have introduced agentic frameworks such as AutoGen, LangGraph, and Crew AI, which support modular agent design, tool integration, and human-in-the-loop workflows. These frameworks are rapidly evolving, with the upcoming AI Agents Framework SDK expected to standardize agent creation, reasoning, and action execution. The ReACT (Reasoning and Acting) paradigm is becoming foundational, enabling agents to interleave thought processes with tool use for dynamic decision-making.

Agentic AI systems are characterized by the following core capabilities (LangChain, 2023b):

- **Autonomy**: Agents initiate and complete tasks with minimal supervision.
- **Contextual Reasoning**: They evaluate trade-offs, make judgment calls, and adapt to changing inputs.
- **Adaptive Planning**: Agents dynamically adjust workflows based on environmental feedback.
- **Multi-modal Understanding**: They process inputs across text, images, APIs, and structured data.
- **Action Execution**: Agents interact with external systems via APIs, databases, and web services.

Enterprise-grade agentic systems increasingly incorporate design patterns such as:

- **Tool Use**: Agents invoke APIs and services to complete tasks end-to-end.
- **Reflection**: Agents self-evaluate and refine output for reliability.
- **Planning**: Agents decompose goals into actionable subtasks with dependencies.
- **Multi-agent Collaboration**: Specialized agents coordinate under orchestration layers to handle complex workflows (CGIAR AI for Agriculture, 2022; LangChain, 2023a; LangChain, 2023b).

Hierarchical Agent Framework and Agentic Architectures

The Hierarchical Agent Framework represents a structured approach to agentic AI, where agents are organized into layers of control and specialization. This architecture mirrors organizational hierarchies, with high-level agents responsible for strategic planning and low-level agents executing domain-specific tasks.

In agricultural applications, for example, a top-level agent may oversee farm-wide optimization, while mid-level agents manage irrigation, pest control, and crop scheduling. Low-level agents interface with IoT sensors, machinery, and weather APIs to execute granular actions. This layered decomposition enhances scalability, interpretability, and fault tolerance in dynamic environments (LangChain, 2023c).

Recent innovations such as **Agent Orchestra** demonstrate the power of hierarchical multi-agent systems. In this framework, a central planning agent decomposes complex objectives and delegates subtasks to specialized agents equipped with analytical tools, memory systems, and multimodal reasoning capabilities. These agents collaborate through explicit sub-goal formulation, inter-agent communication, and adaptive role allocation, outperforming flat-agent baselines in task success and generalization (FAO, 2021).

Key technical design principles of hierarchical agentic systems include:

- **Supervisor-Agent Model**: High-level agents define goals and allocate tasks to subordinate agents.
- **Role-Based Specialization**: Agents are designed around functional roles (e.g., planner, executor, validator) rather than isolated tasks (LangChain, 2023b).
- **Shared Memory and Context**: Agents access persistent state and shared knowledge bases to maintain coherence.
- **Dynamic Orchestration**: Agents adapt roles and responsibilities based on evolving task requirements.
- **Tool-Enabled Execution**: Agents integrate with external tools (e.g., databases, APIs, file systems) to perform real-world actions (LangChain, 2023a).

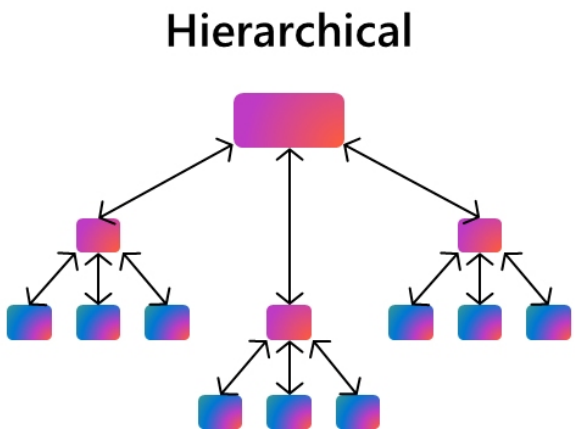

Figure 1: Hierarchical agentic architecture.

Frameworks such as AutoGen and Crew AI support hierarchical agent deployment by offering built-in coordination mechanisms, memory sharing, and error mitigation. These systems are increasingly used in domains like logistics, legal automation, and precision agriculture, where layered task management and domain-specific expertise are critical.

Supervisor Agent Framework Agentic Architectures

A Supervisor Agent (please see Figure 2) is a high-level autonomous entity within a multi-agent or hierarchical AI framework that oversees, coordinates, and regulates the behavior of subordinate agents to ensure alignment with overarching system goals. It typically manages task delegation, monitors

execution progress, handles exceptions, and resolves conflicts among lower-level agents. In agricultural or dairy applications, a Supervisor Agent might monitor weather-aware irrigation agents, crop health agents, and logistics agents, ensuring that their outputs collectively contribute to farm-level yield optimization or sustainability targets. It acts as the decision-making orchestrator, adapting strategies based on environmental changes, system feedback, or evolving objectives, thereby enhancing system resilience, coherence, and autonomy (Vuppalapati, 2025).

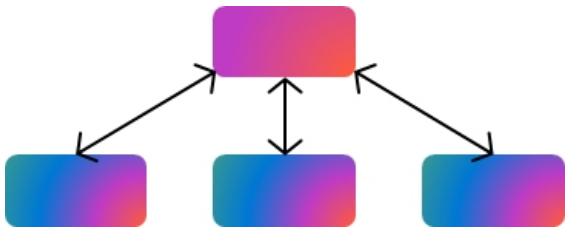

Figure 2: Supervisor agent.

In agentic AI architectures, different coordination patterns govern how agents collaborate to accomplish complex tasks. The ***Network pattern*** involves (please see figure) a decentralized configuration where multiple agents interact laterally, sharing information and decisions in a peer-to-peer fashion without a central controller. This setup is ideal for distributed environments where agents, such as crop monitors, irrigation optimizers, and weather predictors, need to coordinate autonomously. In contrast, the Hand-off pattern follows a sequential flow, where one agent completes its task and passes the result to the next agent in the chain. For instance, a soil analysis agent might trigger a seed selection agent, which then passes information to a planting schedule agent—creating a clear, stage-wise pipeline.

The ***Maker-Checker*** pattern introduces a quality control mechanism: one agent (the maker) produces an output, such as a recommendation or plan, while a second agent (the checker) reviews and validates it before execution. This is particularly useful in high-stakes or regulated domains like finance, healthcare, or precision agriculture (LangChain, 2023a; 2023b; 2023c; Vuppalapati, 2025).

Lastly, the ***Custom pattern*** allows developers to design bespoke agent workflows tailored to unique domain challenges. It may combine elements of hierarchy, recursion, or event-driven executions such as a supervisor agent dynamically orchestrating a team of context-specific agents in response to evolving farm conditions or supply chain disruptions. Each of these patterns offers distinct advantages in terms of scalability, auditability, and flexibility, depending on the use case. agentic architectures (please see Figure 3).

Agentic Frameworks

Emerging Agentic Frameworks

The agentic AI ecosystem is expanding rapidly to meet growing market demands and address the increasing complexity of intelligent system design. This section highlights four prominent frameworks—**LangChain, Semantic Kernel, Google Agent Development Kit (ADK)**, and **Model Context Protocol**

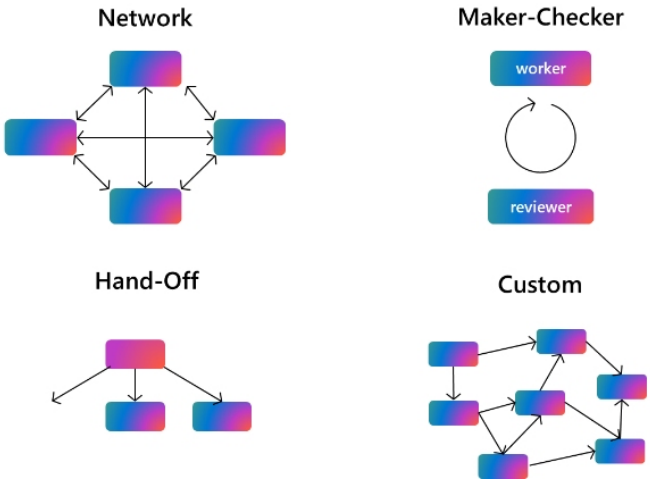

Figure 3: Agentic architectures.

(**MCP**)—each offering unique capabilities for building, orchestrating, and deploying autonomous AI agents.

LangChain

LangChain is a modular Python-based framework designed for building applications powered by large language models (LLMs). It emphasizes two core principles: **data-awareness**, enabling LLMs to connect with external data sources, and **agentic behavior**, allowing models to interact with their environment through tools and APIs (LangChain, 2023a).

Key features include:

- Reusable components such as chains, agents, memory, and tools
- Integration with vector stores, databases, and external APIs
- LangGraph support for stateful orchestration and streaming
- Compatibility with OpenAI, Anthropic, Hugging Face, and other providers.

LangChain is widely used for retrieval-augmented generation (RAG), autonomous task execution, and multi-agent coordination.

Semantic Kernel

Semantic Kernel (SK), developed by Microsoft, is an open-source SDK that enables developers to embed LLMs into enterprise applications using familiar programming languages like C#, Python, and Java (LangChain, 2023b). It abstracts prompt engineering, memory management, and orchestration into modular components.

Core capabilities include:

- **Plugins**: Semantic and native functions that encapsulate AI logic
- **Memory**: Persistent context for multi-turn reasoning
- **Planner**: Goal decomposition and execution strategy engine
- **Connectors**: Interfaces to Azure OpenAI, Hugging Face, and custom APIs.

SK empowers developers to build intelligent copilots and workflow agents with robust observability, security, and integration flexibility.

Google Agent Development Kit (ADK)

Google's Agent Development Kit (ADK) is a model-agnostic framework for building and deploying AI agents within the Gemini and Vertex AI ecosystems. ADK supports hierarchical agent composition, tool integration, and dynamic orchestration (FAO, 2021).

Highlights include:

- **Workflow Agents**: Sequential, parallel, and loop-based execution
- **Multi-Agent Collaboration**: Role-based delegation and coordination
- **Tool Ecosystem**: Built-in and custom tools for real-world actions
- **Deployment Options**: Local containers, Cloud Run, and Vertex AI Agent Engine
- **Integrated Evaluation**: Step-by-step tracing and performance metrics.

ADK is designed to make agent development feel like modern software engineering, with structured templates and CLI support.

Model Context Protocol (MCP)

The **Model Context Protocol (MCP)**, introduced by Anthropic in late 2024, is an open standard for enabling AI models to interact seamlessly with external tools, data sources, and APIs (World Bank, 2022; McKinsey Global Institute, 2022). MCP addresses the "N×M" integration problem by providing a universal interface for AI-to-system communication.

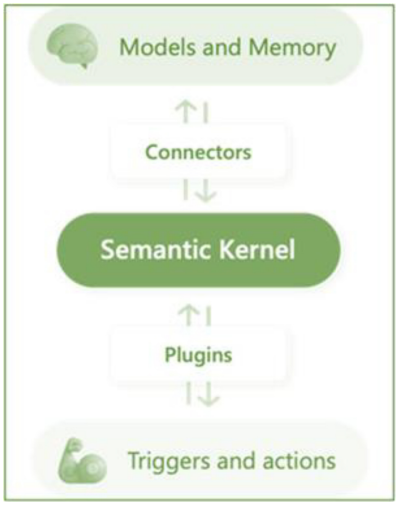

Figure 4: Semantic kernel.

Key architectural components:

- **MCP Host**: The AI model (e.g., Claude, GPT) initiating requests
- **MCP Client**: Middleware that routes requests to MCP servers
- **MCP Server**: Lightweight services exposing APIs, databases, or files
- **Transport Protocols**: Supports JSON-RPC 2.0 over stdio, HTTP, SSE, and WebSockets.

MCP enables secure, scalable, and bidirectional communication between AI agents and enterprise systems. It has been adopted by major providers including OpenAI, Google DeepMind, and Microsoft Azure OpenAI Services. Developers can build MCP clients and servers using official SDKs, allowing AI models to fetch real-time data, execute functions, and deliver context-rich responses (FAO, 2021; World Bank, 2022; McKinsey Global Institute, 2022).

REASONING-DRIVEN AGENTS AND RETRIEVAL-AUGMENTED ARCHITECTURES

Chain of Reasoning: ReACT (Reasoning and Acting)

The current generation of AI agents increasingly functions as business-driven task enablers, enhancing productivity and fostering innovation across domains. A pivotal advancement in this space is the **ReACT** paradigm—**Reasoning and Acting**—which synergizes structured reasoning with actionable execution. Introduced by Google Research, ReACT enables language models to alternate between generating reasoning traces and performing actions, thereby solving complex tasks through dynamic, interpretable trajectories (FAO, 2021).

While the concept of structured reasoning is not new—having long existed in agricultural practices such as crop calendars, irrigation schedules, and input management—ReACT formalizes this process within AI systems. It allows agents to:

- **Decompose goals** into actionable subtasks
- **Inject common sense knowledge** into decision-making
- **Track progress** and adjust plans dynamically
- **Interact with external environments** (e.g., APIs, search engines) to refine reasoning (FAO, 2021; World Bank, 2022).

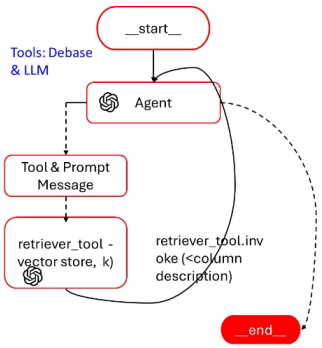

Figure 5: ReACT.

Unlike earlier architecture such as **RAG** (Retrieval-Augmented Generation), which focus primarily on information retrieval and static generation, ReACT agents maintain working memory and adapt actions based on evolving context. This marks a shift from passive generation to **goal-oriented, feedback-driven execution**.

Key frameworks and libraries advancing ReACT include:

- **AutoDev**: A foundational research effort that inspired GitHub Copilot
- **DyLAN and Amazon Q Developer**: Domain-specific agentic platforms
- **MetaGPT, LATS, and ReAct libraries**: Define inter-agent collaboration and reflection patterns
- **AutoGen and CrewAI**: Support multi-agent orchestration with memory and planning modules.

The decision to deploy AI agents depends on several factors:

- **Environmental complexity**: Dynamic, stochastic, or adversarial settings benefit from adaptive agents
- **Data availability**: Agents require rich, relevant datasets for training and inference
- **Human-AI interaction**: Agents may augment, assist, or replace human decision-making depending on autonomy levels
- **Ethical considerations**: Fairness, accountability, and transparency must guide deployment (Vuppalapati, 2019; Yang et al., 2019).

Use cases range from autonomous vehicles and contract negotiation to plant disease diagnosis and music composition. As agents evolve toward higher reasoning fidelity, they may not achieve true consciousness, but they increasingly emulate **goal-driven cognition**.

Retrieval-Augmented Generation (RAG) Pattern

Retrieval-Augmented Generation (**RAG**) is a pragmatic architecture that enhances LLMs by grounding their outputs in external, domain-specific data. Originally introduced by Facebook AI, RAG mitigates hallucinations and improves factual accuracy by integrating semantic search with generative modeling (CGIAR AI for Agriculture, 2022; LangChain, 2023a).

The RAG workflow typically involves four steps:

1. **Embedding Creation**: Internal documents are vectorized using embedding models and stored in a vector database (e.g., FAISS, Azure AI Search).
2. **Query Submission**: A user submits a natural language query.
3. **Context Retrieval**: An orchestrator performs similarity search and retrieves relevant chunks.
4. **Response Generation**: The LLM generates an answer using the retrieved context (IFAD, 2023; CGIAR AI for Agriculture, 2022).

This architecture is particularly effective for enterprise applications such as legal research, customer support, and product matching. It enables:

- **Domain grounding:** Responses are tied to proprietary content
- **Interpretability:** Retrieved sources can be cited and audited
- **Scalability:** Vector databases support real-time search across large corpora
- **Security and control:** Enterprise-grade systems like Azure AI Search offer robust access controls and reliability (IFAD, 2023).

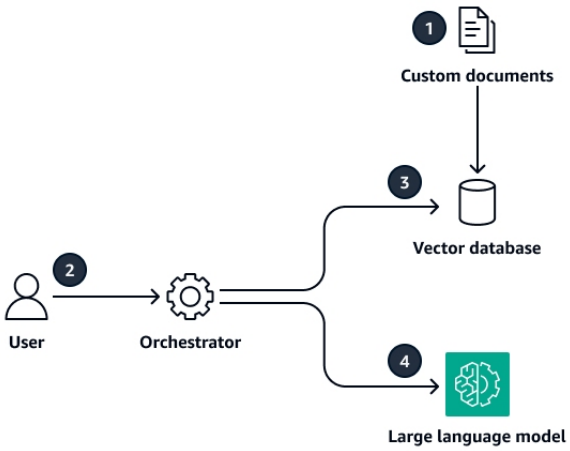

Figure 6: RAG.

Advanced RAG implementations now support **agentic retrieval**, where agents evaluate initial responses and autonomously seek better answers if the original output is incomplete. Frameworks like LangChain, Semantic Kernel, and LlamaIndex facilitate orchestration and integration with retrieval systems.

As generative AI agents become more prevalent, it is essential to approach their deployment with ethical rigor. Bias mitigation, transparency, and responsible design must underpin all implementations. The future of agentic AI promises a world where intelligent systems not only generate content but also **reason, act, and adapt** in real time—transforming industries and empowering users across the globe.

INTEGRATING MCP WITH REACT AND RAG WORKFLOWS

The integration of the **Model Context Protocol** (**MCP**) into agentic workflows such as **ReACT** and **RAG** marks a significant advancement in the design of intelligent, modular AI systems. MCP provides a standardized interface for connecting AI agents to external tools, APIs, and data sources, enabling seamless execution of reasoning-driven and retrieval-augmented tasks.

MCP + ReACT: Enabling Reasoning-Driven Action

The ReACT paradigm—Reasoning and Acting—relies on agents that interleave thought processes with tool use to solve complex tasks. MCP enhances this architecture by serving as the **execution layer** for ReACT agents,

allowing them to invoke tools, access structured resources, and maintain working memory across multi-step interactions.

Key benefits of MCP integration with ReACT include:

- **Tool Invocation**: Agents can call MCP-exposed tools (e.g., search, email, database updates) directly from reasoning traces.
- **Memory Persistence**: MCP servers support long-term memory, enabling agents to track goals, user preferences, and historical context.
- **Modular Composition**: ReACT agents can dynamically select tools and resources based on evolving reasoning paths, improving adaptability and interpretability.
- **Agentless Deployment**: MCP supports agentless architectures, allowing lightweight clients (e.g., React apps) to interact with intelligent backends without hosting full agents.

For example, a ReACT agent embedded in a farm management system could reason about irrigation schedules, retrieve weather data via MCP, and trigger automated watering actions—all within a single reasoning loop.

MCP + RAG: Enhancing Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) enhances LLMs by grounding their outputs in external data. MCP complements RAG by acting as a **retrieval orchestrator**, standardizing access to vector databases, APIs, and structured resources2.

Advantages of MCP-enhanced RAG workflows include:

- **Dynamic Retrieval**: MCP servers expose resources (e.g., documents, records, sensor data) that agents can query in real time, bypassing static embedding pipelines.
- **Tool-Driven Context Injection**: MCP tools can preprocess, filter, or annotate retrieved data before it is passed to the LLM, improving relevance and factual accuracy.
- **Composable Pipelines**: Developers can build modular RAG systems using MCP components such as FastMCP, RetrievalQA, and Chroma, enabling scalable and auditable retrieval flows.
- **Secure and Discoverable Resources**: MCP supports hierarchical data exposure with metadata annotations, allowing agents to select relevant context intelligently.

For instance, an MCP-powered RAG agent in a legal assistant could retrieve clauses from contracts stored in a vector database, summarize them using an LLM, and cite the source documents—all through standardized MCP calls.

Unified Agentic Loop With MCP

By integrating MCP with both ReACT and RAG, developers can build **fully agentic systems** that support:

- **Perception**: Accessing external data via MCP resources
- **Reasoning**: Structuring logic and decision-making via ReACT

- **Action:** Executing tasks through MCP tools
- **Memory:** Persisting context across sessions and agents.

This unified loop enables AI agents to operate autonomously in dynamic environments, bridging the gap between static generation and real-world execution.

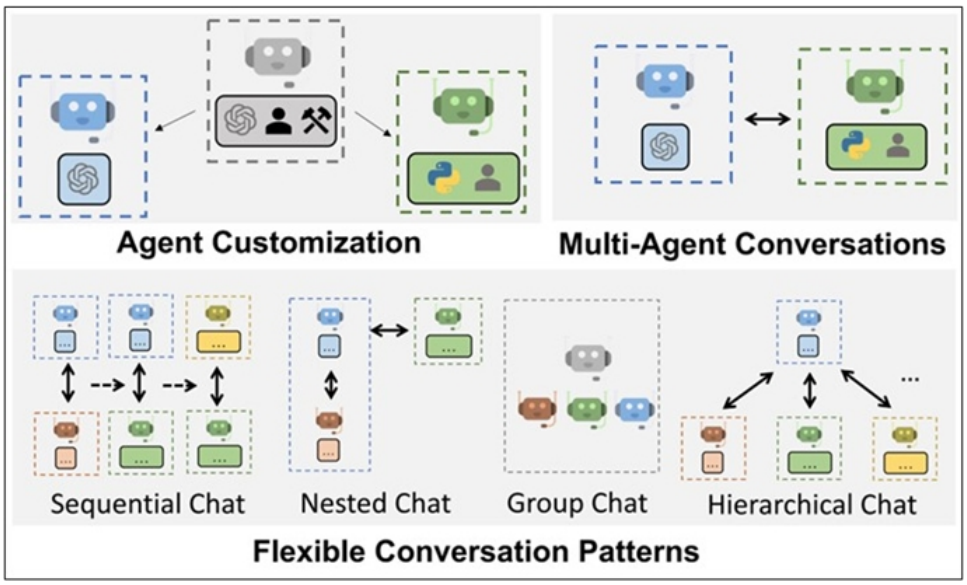

Figure 7: Flexible conversation patterns.

SYSTEM OVERVIEW

This architecture enables a multi-agent, context-aware system to support small farmers by combining: Natural language interfaces, Supervisor agent orchestration, Specialized domain agents and tools, Semantic search over domain-specific vector stores, and Personalized recommendations using farmer history and ESG data (please see Figure 8).

The Hierarchical Agentic AI Architecture system designed to serve small farmers by integrating user interaction, supervising intelligence, worker agents/tools, and backend data storage. Here's a detailed interpretation of each component:

The batch layer then feeds into a serving layer, which indexes the batch view for efficient window querying. In turn, the speed layer updates the serving layer with incremental updates based on the most recent data. By utilizing the lambda architecture, we can optimize data processing for rumination count detection, providing valuable insights into the health and wellbeing of cattle for farmers and industry professionals alike (CGIAR AI for Agriculture, 2022; LangChain, 2023c; Vuppalapati, 2025; Ilapakurti and Vuppalapati, 2015).

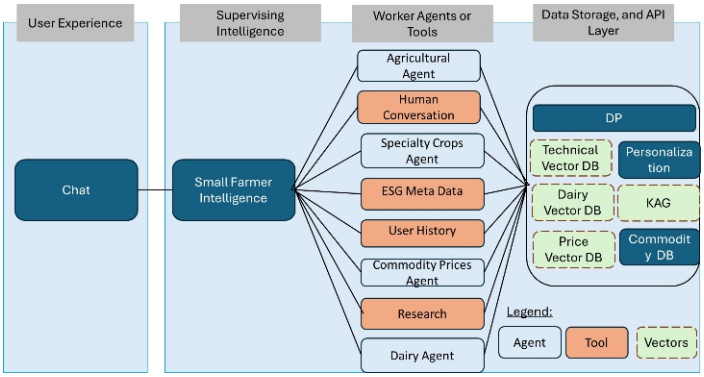

Figure 8: Agentic architecture.

User Experience

This is the entry point for farmers or users to interact with the system via natural language. It acts as the conversational interface, translating user queries into structured inputs for agentic processing. May support multilingual and voice interfaces, crucial for accessibility in rural regions.

To calculate the rumination count, we utilize both sensor data, with the exception of the accelerometer, which serves as an inhibitor or initiator trigger. By combining these data points, we can gain valuable insights into the health and wellbeing of the cattle, providing farmers with the necessary information to optimize their feed management practices and ensure the continued health and productivity of their herds.

Supervising Intelligence

This is the central Supervisor Agent responsible for orchestrating other agents. It interprets user intent, decides which worker agents to activate, and routes context across modules. Learns farmer preferences, history, and goals to enable personalized, context-aware responses.

SYSTEM DESIGN AND IMPLEMENTATION

As part of the system design, the next step would be application of Window functions to detect rumination and then once detected calculated rumination count (Vuppalapati, 2019).

Agents

Following are agents that helps supervising architecture:

- Agricultural Agent: Provides recommendations on crop planning, soil health, irrigation, and sustainable practices.
- Specialty Crops Agent: Focuses on region-specific or high-value crops like spices, floriculture, or organic produce.
- Commodity Prices Agent: Tracks and analyzes real-time market trends, price fluctuations, and forecasts.
- Dairy Agent: Advises on cattle health, milk productivity, feed optimization, and veterinary support.

Tools (Highlighted in Orange)

Following are tools:

- Human Conversation: Supports natural human-agent dialogue handling beyond structured queries.
- ESG Meta Data: Embeds Environmental, Social, and Governance context, such as carbon emissions or water usage.
- User History: Stores interaction logs and behavioral patterns for personalized recommendations.
- Research: Fetches domain-specific knowledge, including scientific papers, weather advisories, and agronomy updates.

Data Storage and API Layer

This layer underpins the intelligence system by storing embeddings, domain data, and delivering personalization.

Databases (Green - Vector Stores)

- Technical Vector DB: Stores embeddings from technical manuals, soil data, etc., used for semantic search.
- Dairy Vector DB: Contains vectorized content related to dairy health, nutrition, and productivity.
- Price Vector DB: Captures historical and real-time market price data for commodities.

A CASE STUDY

We have tested and deployed Agentic AI models (please see Figure 9) for Agriculture Analytics, Specialty Crops, and Value Chain Analytics (Hanumayamma, 2024).

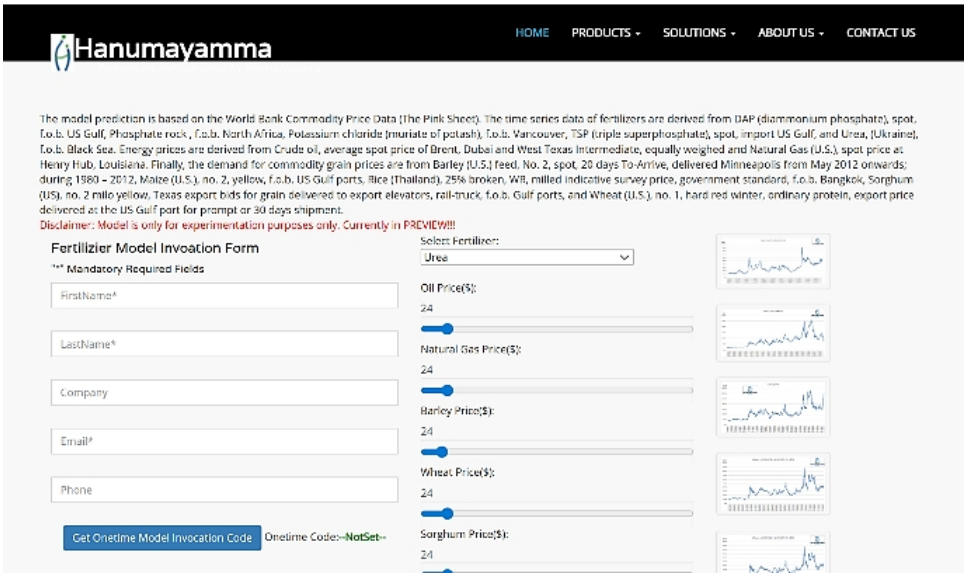

Figure 9: Fertilizer model agent.

CONCLUSION AND FUTURE WORK

The dairy and agriculture sectors are increasingly challenged by critical human resource issues that threaten their long-term viability and global food security. These include a shrinking rural workforce due to urban migration, demographic aging of farm laborers, declining interest in agricultural employment among youth, and significant labor shortages during peak farming cycles. Technological advancements—though necessary—have not yet offset the human capital deficit in many developing and developed regions. The result is a growing gap between agricultural production needs and workforce availability, directly impacting crop yields, dairy productivity, and supply chain resilience. Addressing these issues is central to achieving sustainable development goals (SDGs), particularly in food security, poverty reduction, and rural economic stability.

REFERENCES

A. Ilapakurti and C. Vuppalapati, "Building an IoT Framework for Connected Dairy," 2015 IEEE First International Conference on Big Data Computing Service and Applications, Redwood City, CA, USA, 2015, pp. 275–285, doi: 10.1109/BigDataService.2015.39.

A. Ramalingam, S. Kedari and C. Vuppalapati, "IEEE FEMH Voice Data Challenge 2018," 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 2018, pp. 5272–5276, doi: 10.1109/BigData.2018.8622164.

C. Vuppalapati, A. Ilapakurti, S. Kedari, J. Vuppalapati, S. Kedari and R. Vuppalapati, "Democratization of AI, Albeit Constrained IoT Devices & Tiny ML, for Creating a Sustainable Food Future," 2020 3rd International Conference on Information and Computer Technologies (ICICT), San Jose, CA, USA, 2020, pp. 525–530, doi: 10.1109/ICICT50521.2020.00089.

C. Vuppalapati, A. Ilapakurti, S. Kedari, R. Vuppalapati, J. Vuppalapati and S. Kedari, "The Role of Combinatorial Mathematical Optimization and Heuristics to improve Small Farmers to Veterinarian access and to create a Sustainable Food Future for the World," 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 2020, pp. 214–221, doi: 10.1109/WorldS450073.2020.9210339.

CGIAR AI for Agriculture (2022). Digital Extension Services: Toward Inclusive Agri-AI Deployment. https://aiforag.org.

Chandrasekar Vuppalapati (Author), Artificial Intelligence and Advanced Analytics for Food Security, Publisher: CRC Press, Publication date: July 17, 2023, ISBN-13: 978–1032346182.

Chandrasekar Vuppalapati (Author), Assessing Policy Effectiveness using AI and Language Models: Applications for Economic and Social Sustainability, Publisher: Springer, Publication date: May 31, 2024, ISBN-13: 978–3031560965.

Chandrasekar Vuppalapati, Building Enterprise IoT Applications, CRC Press; 1st edition (December 17, 2019), ISBN-13: 978–0367173852.

Chandrasekar Vuppalapati, Democratization of Artificial Intelligence for the Future of Humanity, Publisher: CRC Press; 1st edition (January 18, 2021), ISBN-13: 978–0367524098.

Chandrasekar Vuppalapati, Divine Feminine Energies in Vedic Anthologies: A Scientific Inquiry Using AI and Large Language Models, Publication Date: July 14, 2025, ISBN: 979–8-89530–583-6.

Chandrasekar Vuppalapati, Machine Learning and Artificial Intelligence for Agricultural Economics: Prognostic Data Analytics to Serve Small Scale Farmers, Publisher: Springer; 1st ed. 2021 edition (October 5, 2021), ISBN-13: 978–3030774844.

Editors: Yang, X.-S., Sherratt, S., Dey, N., Joshi, A, ICICT 2019, London, Volume 2, "Fourth International Congress on Information and Communication Technology", eBook ISBN 978–981-329–343-4 and Softcover ISBN 978–981-329–342-7.

FAO (2021). The State of Food and Agriculture 2021 – Making Agrifood Systems More Resilient. Food and Agriculture Organization of the United Nations. https://www.fao.org.

Hanumayamma, Try Our Fertilizer Model, 2024 https://www.hanuinnotech.com/agricultureanalytics.html, Access Date: July 14, 2025.

IFAD (2023). The Future of Smallholder Farming in the Digital Age. https://www.ifad.org.

LangChain, Build a Question/Answering system over SQL data, 2023, https://python.langchain.com/docs/tutorials/sql_qa/, Access Date: July 14, 2025.

LangChain, Build a Retrieval Augmented Generation (RAG) App: Part 1, 2023, https://python.langchain.com/docs/tutorials/rag/, Access Date: July 14, 2025.

LangChain, Build a Retrieval Augmented Generation (RAG) App: Part 2, 2023, https://python.langchain.com/docs/tutorials/qa_chat_history/, Access Date: July 14, 2025.

McKinsey Global Institute (2022). Agriculture's Connected Future: How Technology Can Yield New Growth. https://www.mckinsey.com.

World Bank (2022). Harvesting Prosperity: Technology and Productivity Growth in Agriculture. https://www.worldbank.org.