Masala Mamu : Agentic AI Kitchen Assistant

Barani Ranjan S[a,b], Brijgopal Bharadwaj[a,b], M Chandan Kumar Rao[a,b], Shunmuga Janani A[a,b] and Siva S[a,b]

[a]Division of Interdisciplinary Sciences
[b]Indian Institute of Science, Bangalore

Abstract. Modern households struggle with kitchen management tasks like grocery tracking, meal planning, and dietary monitoring. We present Masala Mamu, an AI-powered kitchen assistant using a multi-agent system to integrate these functions. A central Router Agent delegates tasks to specialized agents: Inventory Manager (tracks ingredients), Price Comparison Agent (finds deals across vendors), Recipe Generator (suggests meals based on inventory), and Health & Diet Agent (monitors nutrition). Built on LangChain/LangGraph with multimodal capabilities, the system offers an integrated approach to meal planning and grocery shopping optimized for Indian dietary preferences and e-commerce landscape.

Project Code: https://github.com/chandanraoiisc/masala-mamu-agent-ai

1 Introduction

Maintaining a balanced diet while managing costs presents significant challenges. Existing solutions separately address nutrition tracking, recipe generation, or price comparison, creating fragmented experiences. Masala Mamu integrates these functions for the Indian context, where diverse dietary preferences and a complex e-commerce landscape complicate decision-making.

Key contributions include:

- A multi-agent routing architecture for specialized task delegation
- A recipe generation system that considers inventory and dietary preferences
- A nutrition analysis system with macro-nutrient breakdown capabilities
- A price comparison agent for major Indian e-commerce platforms
- Vision-based inventory management with semantic search
- Real-time nutrition tracking dashboard with personalized insights

2 System Architecture

2.1 Multi-Agent Architecture

Masala Mamu uses specialized agents coordinated by a central router:

- **Routing Agent:** Directs workflows based on query intent and task state
- **Recipe Agent:** Generates meal suggestions based on inventory, dietary, and cuisine preferences
- **Nutrition Agent:** Processes food queries and provides nutritional data
- **Price Agent:** Finds best grocery deals across e-commerce platforms

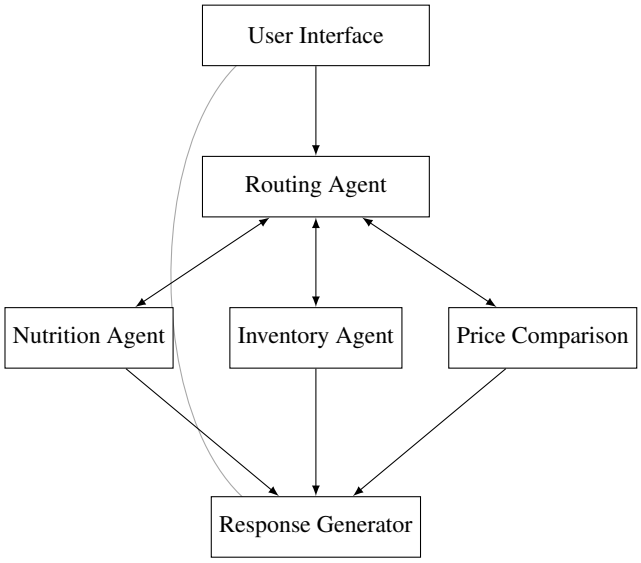

Figure 1. Multi-agent system architecture

- **Inventory Agent:** Manages kitchen ingredients database
- **Response Generator:** Creates cohesive responses from agent outputs

LangGraph orchestration enables complex multi-domain queries like "What's the most cost-effective high-protein vegetarian meal?"

2.2 Technology Stack

- **LLMs:** OpenAI, Gemini, GitHub Marketplace Models
- **Framework:** LangChain/LangGraph for orchestration
- **Frontend:** Streamlit for UI and visualizations
- **Storage:** SQLite (nutrition), MongoDB (inventory)
- **Vision:** GPT-4o for ingredient recognition
- **Embeddings:** Sentence-transformers for semantic search
- **Scraping:** Custom tools for e-commerce price comparison

3 Key Components

3.1 Routing Engine

The LangGraph router flow powers the system's execution logic. An IntentParser using GPT extracts query intents, key entities, and determines agent sequencing. The router, the entry point for queries, tracks required and completed agents to select the next one to activate.

Each agent functions as a node in the graph, processing its specialized task before updating the state and returning control to the router. This creates a conditional execution flow based on user intent that invokes the response generator when complete. This modular approach ensures goal-driven execution and easy extensibility.

3.2 Nutrition Analysis System

The nutrition analysis component provides macro-nutrient information through an LLM-powered approach with search augmentation for up-to-date data:

Algorithm 1 Nutrition Analysis Workflow

1: Parse query (recipe vs. ingredients)
2: Initialize LLM with nutrition prompt
3: Perform web search for nutrition data
4: Extract structured data with Pydantic
5: Store in SQLite database with timestamp
6: Generate visualizations of trends

Key technical features include:

- **Search Augmentation:** Custom DuckDuckGo wrapper for nutrition data retrieval
- **Source Tracking:** Attribution system for data provenance and transparency
- **Cooking Method Awareness:** Adjusts nutrition based on cooking techniques
- **Structured Data Models:** Pydantic models (MacroNutrient, IngredientNutrition)
- **Indian Cuisine Support:** Regional ingredient recognition and unit conversion

The relational SQLite database enables historical analysis with per-ingredient breakdowns, supporting comprehensive analytics through the dashboard interface.

3.3 Price Comparison Engine

The price comparison engine scrapes real-time pricing from Indian e-commerce platforms (BigBasket, BlinkIt, Zepto, JioMart) using custom tools and suggests alternatives based on price and nutritional similarity. Data is presented comparatively for informed purchasing decisions.

3.4 Kitchen Inventory Management

The inventory management module tracks groceries and quantities using:

- GPT-4o vision for identifying groceries from images and bill receipt
- Vector embeddings for semantic search capabilities

In MongoDB, Inventory collection has ItemNm and it's corresponding quantity with its stored on date as fields. Along with this, an embedding field is also added with the embedded ItemNm in 384 dim vector space. A vector search index is generated on this embedding field in MongoDB. The vector search index is used as the vector store in RAG pipeline.

Users can upload grocery photos or receipts for automatic item detection, with review options before database addition. The RAG-based query system enables natural language inventory questions like "What ingredients do I have for pasta?" or "Which vegetables will expire soon?"

3.5 Recipe Generation Service

The recipe generation service provides intelligent dish recommendations tailored to available ingredients and user preferences. Key features include:

- **User Preferences Support:** Accounts for dietary restrictions (e.g., vegan, gluten-free) and preferred cuisine types (e.g., Italian, Indian) to generate personalized recipes
- **Structured Response:** Returns recipes in a consistent JSON format, including ingredient quantities and step-by-step instructions
- **Inventory Integration:** Leverages available ingredient data to recommend practical, waste-reducing meal options
- **Missing Ingredient Identification:** Assists the router agent in detecting any missing ingredients and passes this information to the price comparison service, which then recommends the most cost-effective platforms for purchasing them

The service uses prompt engineering to ensure contextually relevant, structured recipe outputs. Integration with inventory and shopping services simplifies meal planning while supporting dietary needs and budget-conscious decisions.

3.6 Nutrition Analytics Dashboard

The Plotly-powered nutrition dashboard provides detailed nutritional insights through:

- **Macro-nutrient Tracking:**
  - Multi-panel time-series of calories, protein, carbs, and fat
  - Configurable date ranges (7-90 days) with target value indicators
- **Nutritional Balance:**
  - Macro-nutrient distribution pie charts with ratio analysis
  - Caloric source breakdown with percentage calculations
- **Interactive Features:**
  - Customizable nutritional targets and data filtering
  - Export options for reports and multi-user support
  - Consistency tracking to identify gaps in nutrition records

Built with Streamlit and connected directly to the SQLite database, the dashboard dynamically updates charts based on the latest nutrition data, providing immediate visual feedback on dietary patterns and trends.

4 Implementation Details

4.1 Nutrition Agent Implementation

The nutrition agent uses LangChain's framework with:

- **System Prompt:** Specialized prompt with structured output requirements and explicit guidance on nutrition analysis
- **Function-Calling Architecture:** OpenAI functions with custom NutritionSearchTool for ingredient and recipe queries
- **Data Models:** Pydantic models (MacroNutrient, IngredientNutrition, RecipeNutrition) for structured data handling
- **Processing Pipeline:** Query classification, web search, data consolidation, and unit normalization

Router integration is handled through a NutritionAgentRouter class that manages intent detection, conversation context, and structured responses compatible with the response generator.

Data extraction features include:

- Optimized search queries for nutrition databases
- LLM-based extraction from semi-structured content
- Unit conversion for international and Indian measurements

- Error handling with fallback mechanisms

The implementation stores both raw and processed data for auditability while providing accurate nutrition information with source citations.

4.2 Database Schema

The Nutrition-Agent SQLite database includes three key tables:

- **nutrition_inquiries:** Query metadata with timestamps and user identifiers
- **nutrition_records:** Recipe/ingredient macro-nutrient data with foreign keys to inquiries
- **ingredient_records:** Detailed per-ingredient nutrition with standardized units

This relational design enables time-series analysis of nutrition patterns, per-ingredient nutritional impact assessment, data integrity through foreign key constraints, and optimized queries for dashboard visualizations. Database utility functions abstract complex queries while maintaining performance with large datasets spanning months of nutrition records.

4.3 Routing Logic

The routing logic is implemented using LangGraph, starting at a router node that evaluates the state and determines the next agent to activate.

1. **Router Initialization:** Defined as a conditional edge that routes based on required_agents and completed_agents.
2. **Agent Registration:** Agents are added as nodes with async handlers to process input and return updated state.
3. **Looping Execution:** After each agent runs, control returns to the router to evaluate the next step.
4. **State Management:** Agents append themselves to completed_agents and add structured outputs (e.g., recipe_data, inventory_data).
5. **Extensibility:** New agents can be added without modifying router logic—only update the IntentParser.

4.4 Response Generator

The ResponseGeneratorAgent reads the current state and parsed intent to understand the user's query and the agents involved in processing it. Based on this, it selectively parses outputs such as recipe_data, inventory_data, shopping_data, and health_data, depending on which agents were used.

It constructs a structured response tailored to the user's request by including only the relevant sections—e.g., a recipe if the user asked for one, or nutritional data if health advice was sought. This ensures precision and relevance in the system's responses while maintaining a cohesive user experience across multiple agent interactions.

5 Evaluation

The system was evaluated on several dimensions:

5.1 OCR and Image Recognition Accuracy

Used OCR Readers for reading the grocery bills and handwritten texts, easyOCR as a standalone detected meaningless vegetable names by combining texts, easyOCR with an embedding pipeline to extract just vegetables and fruits, worked better, but still had issues with image noise and handwritten texts. Gpt-4O performed much better with very low WER and CER as shown in Table 1. A dataset of synthetic bills with handwritten text format and noisy images are generated and used for this comparison.

Model	WER (D1)	CER (D1)	WER (D2)	CER (D2)	WER (D3)	CER (D3)
GPT-4o	**0.1182**	**0.0745**	**0.1044**	**0.0583**	**0.0880**	**0.0577**
EasyOCR + Embedding Pipeline	0.5508	0.3751	0.5580	0.3839	0.5586	0.3819

Table 1. Comparison of OCR accuracy using Word Error Rate (WER) and Character Error Rate (CER) metrics

Similarly, a dataset of different vegetables is crawled from web for comparing different models for image detection. A clip model using a small patch and large patch of openAI Vit pretrained model is compared with gpt-4O. Clip model with large patch performed much better with higher accuracy but failed in distinguishing alike items eg: Peas vs Beans. Gpt-4o performed with full accuracy in identifying individual as well as group of vegetables which are declutterred. Table 2 shows the comparison of accuracy metrics for different models on these datasets.

Model	Dataset 1	Dataset 2	Dataset 3
CLIP (ViT-B/32, small patch)	0.72	0.82	0.81
CLIP (ViT-L/14, large patch)	0.92	0.93	0.94
GPT-4o (Vision + Multimodal)	**0.97**	**0.97**	**0.98**

Table 2. Comparison of image recognition accuracy across different models and datasets

5.2 Recipe Generation Accuracy

Recipe generation capabilities were evaluated across four large language models on three datasets of varying complexity and cuisine types. As shown in Table 3, smaller, well-optimized models outperformed larger ones in this specialized domain.

Model	Dataset 1	Dataset 2	Dataset 3
xai/grok-3-mini	**0.99**	0.95	**0.96**
mistral-ai/Ministral-3B	0.97	0.93	0.94
meta/Meta-Llama-3.1-70B-Instruct	0.68	0.64	0.70
microsoft/Phi-3.5-MoE-instruct	**0.99**	**0.97**	**0.96**

Table 3. Comparison of recipe generation accuracy across different LLMs and datasets

Both xai/grok-3-mini and microsoft/Phi-3.5-MoE-instruct demonstrated exceptional accuracy (above 0.95) across all datasets. The mistral-ai/Ministral-3B model also performed well with slightly lower but reliable scores. Surprisingly, the significantly larger meta/Meta-Llama-3.1-70B-Instruct model showed substantially lower accuracy (0.64–0.70), indicating that architecture specialization and training optimization are more important than model size for domain-specific tasks like recipe generation.

6 Conclusion & Future Work

Masala Mamu demonstrates the effectiveness of multi-agent kitchen assistance, combining nutrition awareness with price optimization in a unified interface. The modular architecture enables future expansion.

Future work:

- Enhanced image recognition for receipts/food photos
- Nutrition-goal-based recipe recommendations
- Long-term dietary pattern analysis
- Meal planning and grocery delivery integration

References

[1] LangChain Framework Documentation, 2023.
[2] LangGraph: Graph-based Multi-agent Orchestration, 2024.
[3] Streamlit: The fastest way to build data apps, 2022.
[4] Confident AI, "LLM Evaluation Metrics", https://documentation.confident-ai.com/llm-evaluation/metrics/create-locally, 2023.
[5] GitHub Marketplace Models, https://github.com/marketplace/models, 2024.

Individual Contributions

Barani Ranjan S

Master of Technology (Online) - DSBA

Led the overall architecture and system design of the Kitchen Assistant, adopting a modular, agent-based approach using LangGraph. Developed the core LangGraph orchestration layer to coordinate multiple specialized agents through a well-defined execution flow, including a custom workflow orchestrator, a robust router node for intelligent agent routing, and standardized agent and state models to ensure consistent communication across components.

Implemented the IntentParser for understanding user queries and dynamically constructing the agent workflow. Created the ResponseGeneratorAgent to synthesize outputs from multiple agents into a concise and context-aware final response. Beyond core orchestration, integrated teammates' independent agents into the unified workflow, adapting their implementations to fit seamlessly within the LangGraph structure. Built and deployed a scalable and reliable FastAPI backend, ensuring smooth interaction between users and the agentic pipeline.

Brijgopal Bharadwaj

Master of Technology (Online) - DSBA

Developed the nutrition-agent module as a cornerstone of the kitchen assistant system, creating an LLM-based agent that accurately breaks down nutritional content of recipes and ingredients. Designed a system that searches the web for up-to-date nutrition data using DuckDuckGo and structures this information using Pydantic models. Implemented a SQLite database to store past queries, enabling users to track nutritional habits over time, visualized through Matplotlib charts and a Streamlit dashboard.

Enhanced system flexibility to work with different LLM providers (supporting OpenAI, GitHub Copilot, and Groq). Implemented advanced features such as cooking method awareness in nutrition calculations, comprehensive error handling, logging, and a functional CLI. Created a router integration layer for smooth communication with other agents. Additionally, coordinated team efforts for the final LaTeX report, helping with formatting and condensing contributions within the page limit.

M Chandan Kumar Rao

Master of Technology (Online) - DSBA

Developed a Multi-Platform Price Comparison Agent that searches and compares prices for groceries and household items across major Indian e-commerce platforms including BigBasket, Blinkit, Zepto, and Swiggy Instamart, using LangChain technologies. Built a standalone Price Comparison MCP server architecture that integrates with the chatbot frontend, leveraging Streamlit, LangChain, and LangGraph technologies.

Created a feature-rich chatbot frontend with integrated text-to-speech capabilities using Google TTS, voice-to-text functionality, and dynamic chart visualization. These features enhanced the user experience by providing multiple interaction modalities and clear visual representation of price comparisons. Additionally, made significant contributions to the overall system design and project documentation.

Shunmuga Janani A

Master of Technology (Online) - DSBA

Developed the inventory agent by creating a MongoDB instance with vector search capabilities. Utilized Sentence Transformer for embedding the ItemNm key field and enabled vector search indexing in MongoDB for the embedding field. Created CRUD operation functions and implemented a RAG framework with an LLM agent for generating responses based on the RAG retrieved context.

Enhanced the system with image processing capabilities using GPT-4o to read images and receipts, extracting groceries with corresponding quantities. Implemented a Human-In-Loop confirmation process before updating inventory. Conducted comparative analysis of OCR and image recognition models using artificially generated bills and vegetable images, ensuring selection of the most effective model for the system's vision components.

Siva S

Master of Technology (Online) - DSBA

Developed a structured LLM-based cooking assistant by integrating open-source models from the GitHub Models Marketplace using LangChain. Engineered prompt templates capable of conditional reasoning and JSON-formatted responses, applying techniques such as few-shot prompting, chain-of-thought reasoning, and instruction tuning to improve generalization and reliability. Built an evaluation pipeline using DeepEval with a custom wrapper for GitHub-hosted models, enabling automated benchmarking on the RecipeDataset using answer relevancy metrics.

Explored retrieval-augmented generation with DuckDuckGo-based context injection, later deprioritizing it after observing sufficient generalization from the base LLM. Experimented with injecting real-time ingredient availability as contextual input, retrieved via semantic vector search over a MongoDB Atlas database. Initially fed this inventory context directly into the recipe generation chain, but later correctly rerouted it through a dedicated router agent to support modular, agentic flow orchestration. Encapsulated the evaluation workflow into a reusable framework for scalable testing across diverse models and domains.

Appendix

Project Images

Figure 2. System interface visualization

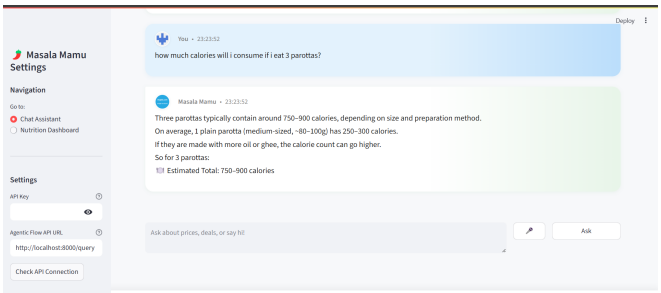

Figure 3. User interaction flow diagram

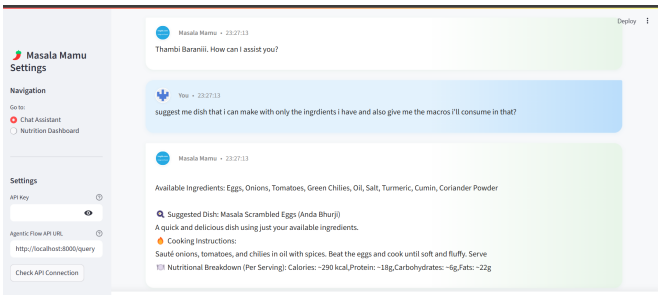

Figure 4. Agent communication network

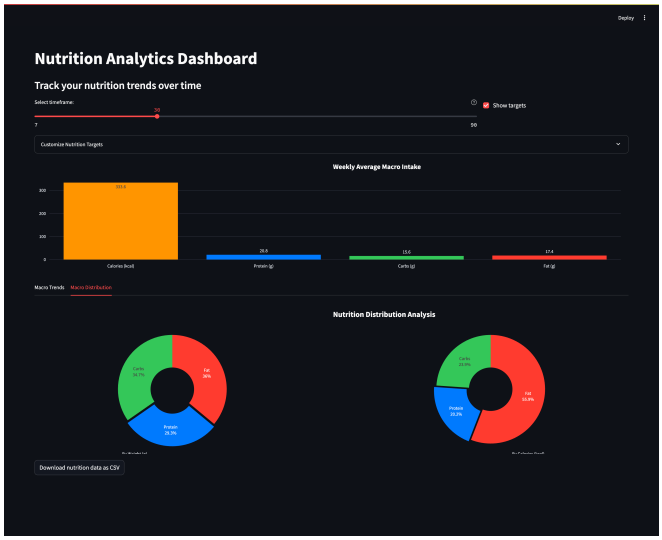

Figure 5. Macro-nutrient distribution visualization

Figure 6. Nutrition trends over time