DMPedia Lecture Notes in Multidisciplinary Research,
Volume 1, pp.705-716, 2026

International Conference on Multidisciplinary Perspectives in Advanced Computing and Technology (IMPACT 2026)
G. B. Pant University of Agriculture and Technology, Uttarakhand, India. Jan. 10-11, 2026

CookIS: Your AI-Powered Cooking Companion

Aadithya S L, Anandadev A, Ashvin P Kumar, Aynamfathima, Neethi Narayanan
Department of Computer Science and Engineering (Artificial Intelligence), Mar Baselios College of Engineering and Technology (Autonomous), Thiruvananthapuram, Kerala, India
Emails: slaadithya@gmail.com, adevd2004@gmail.com, ashvinpkumar2004@gmail.com, aynamfanvar24@gmail.com, neethi.narayanan@mbcet.ac.in

ABSTRACT

This paper presents CookIS, an innovative AI-powered cooking assistant that delivers a hands-free, personalised, and context-aware culinary experience. The system integrates wearable glasses with a camera and microphone to capture real-time visual and audio inputs, which are processed using a Raspberry Pi and a home server running advanced AI models. The setup employs YOLOv5 for ingredient recognition, Google Speech Recognition for voice commands, and a Retrieval-Augmented Generation (RAG)-enhanced LLaMA model to generate step-by-step cooking instructions. Output is delivered to the user through Google Text-to-Speech (gTTS), providing seamless audio guidance. CookIS addresses key challenges in conventional cooking, such as manual recipe searching and difficulty following instructions. Experimental results demonstrate the system's effectiveness in enabling real-time guidance, adaptability to available ingredients, and enhanced user experience, particularly for beginners and elderly users. By combining wearable technology, computer vision, and large language models, CookIS contributes toward accessible, sustainable, and efficient smart kitchen solutions.

Keywords: *Cooking Assistant, Real-time Guidance, RaspberryPi, YOLOv5, Speech Recognition, Retrieval-Augmented Generation, LLaMA, gTTS.*

1. INTRODUCTION

Cooking is both a necessity and a creative pursuit. Still, for many, the process is often complicated by a lack of personalised guidance, limited culinary knowledge, and distractions from conventional recipe media such as smartphones or books. Existing applications offer generic recipes but lack context-awareness and adaptability to user-specific environments. Some smart kitchen solutions rely on costly hardware and intrusive setups, limiting their accessibility. With advances in artificial intelligence (AI), computer vision, and wearable computing, it is now possible to create intelligent cooking assistants that provide real-time, hands-free, and personalised guidance. This work introduces CookIS, an AI-powered cooking assistant that leverages smart glasses, a Raspberry Pi, YOLOv5, and LLaMA with Retrieval-Augmented Generation to deliver interactive cooking instructions and enhance user convenience.

1.1 Research Methodology

The CookIS system employs a modular client–server architecture with four cooperating modules:

(1) client-side capture, (2) server-side ingredient recognition, (3) server-side conversational response generation, and (4) client-side speech output. This section provides detailed implementation steps to ensure reproducibility.

1.2 System Overview

The workflow follows these stages:

- Client Capture – Deployed on a Raspberry Pi, acquires real-time video and audio,

performs on-device speech transcription, and transmits data to the server.

- Server Receive – Implemented as a Flask API that processes incoming video frames using YOLOv5 for ingredient recognition and forwards recognised commands to the conversational model.
- Server Send – Integrates a Retrieval-Augmented Generation (RAG) pipeline with LLaMA 3.1, enabling retrieval of relevant recipes from a Pinecone-based vector database and generation of stepwise cooking instructions.
- Client Speech – Receives streamed responses through SocketIO and converts them into spoken instructions using Google Text-to-Speech (gTTS).

1.3 Client-Side Capture (Input Module)

The client module was implemented in Python and executed on a Raspberry Pi 5 integrated with smart glasses.

- Video Capture: Video streams were acquired using OpenCV at approximately 10 FPS. Captured frames were stored in a bounded queue (size 10) to prevent memory overflow, compressed to JPEG, Base64-encoded, and included in the transmission payload.
- Audio Capture and Transcription: Audio was recorded at 16 kHz using PyAudio with PCM16 encoding. A simple Voice Activity Detection (VAD) mechanism based on amplitude thresholds (≥300) was employed to segment speech. Segmented audio was then transcribed into text using the Google Speech Recognition API.
- Transmission: Both image frames and transcribed text were packaged as a JSON payload and transmitted via HTTP POST requests to the Flask server endpoint /process_data.
- Thread Management: Multithreading ensured concurrent execution of video capture, audio capture, transcription, and data transmission, allowing real-time performance.

1.4 Server-Side Processing (Recognition Module)

The server-side recognition module was implemented using Flask.

- Session Management: Cooking sessions were activated upon detection of the command “start session”, and the ingredient memory was reset to ensure clean initialisation.

- Ingredient Detection: Video frames received from the client were decoded into NumPy arrays and passed to a fine-tuned YOLOv5 model (best.pt). Non-Maximum Suppression (NMS) was applied to remove duplicate detections, and ingredients with confidence scores ≥0.5 were stored in a session-level set.

- Command Forwarding: Transcribed text inputs were labelled as chef-commands and, together with the detected ingredients, forwarded to the Chat API endpoint (/chat) for further processing as shown in Figure 1.

Figure 1: Image Processing and Ingredient Detection at Server

1.5 Server-Side Conversational AI (Instruction Module)

The instruction module was developed using Flask-SocketIO to support real-time response streaming.

- Knowledge Base Preparation: A recipe book in PDF format was parsed using LangChain's PyPDFLoader. The text was split into ~1200-character chunks with an overlap of 50 characters to preserve semantic continuity. Embeddings were generated using the HuggingFace MiniLM-L6-v2 model and indexed in Pinecone (dimension = 384, cosine similarity).
- Query Enrichment: User queries were enriched with contextual information, including detected ingredients, command history, and retrieved recipe chunks from the Pinecone vector store.
- Response Generation: The composite prompt was processed by the LLaMA 3.1 model via Ollama to generate concise, stepwise cooking instructions. Responses were streamed back to the client using SocketIO. A context-checking mechanism ensured consistency, for example, verifying whether actions such as "stirring" were performed before issuing subsequent steps, as shown in Figure 2.

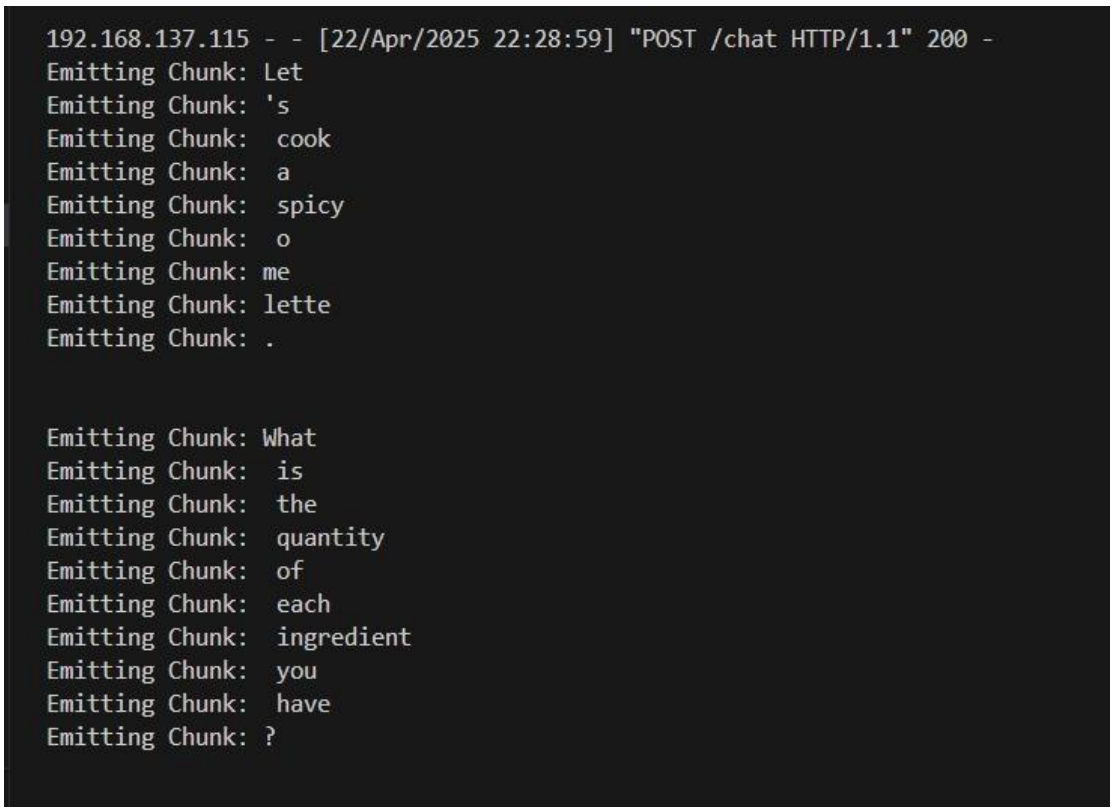

Figure 2: Server-Side Streaming Response from LLaMA Model

1.6 Client-Side Speech Output (Output Module)

The speech output module was developed in Python to convert streamed responses into audio playback.

- Streaming Reception: A SocketIO client subscribed to message_chunk and stream_complete events. Incoming text messages were buffered and assembled into complete sentences.
- Speech Synthesis: The buffered text was converted into audio using Google Text-to-Speech (gTTS). Temporary MP3 files were generated, played via the playsound library, and subsequently deleted to prevent storage clutter.
- Queue Management: A thread-safe queue was employed to manage sequential speech outputs, ensuring that responses were delivered in the correct order without overlap. As per Figure 3.

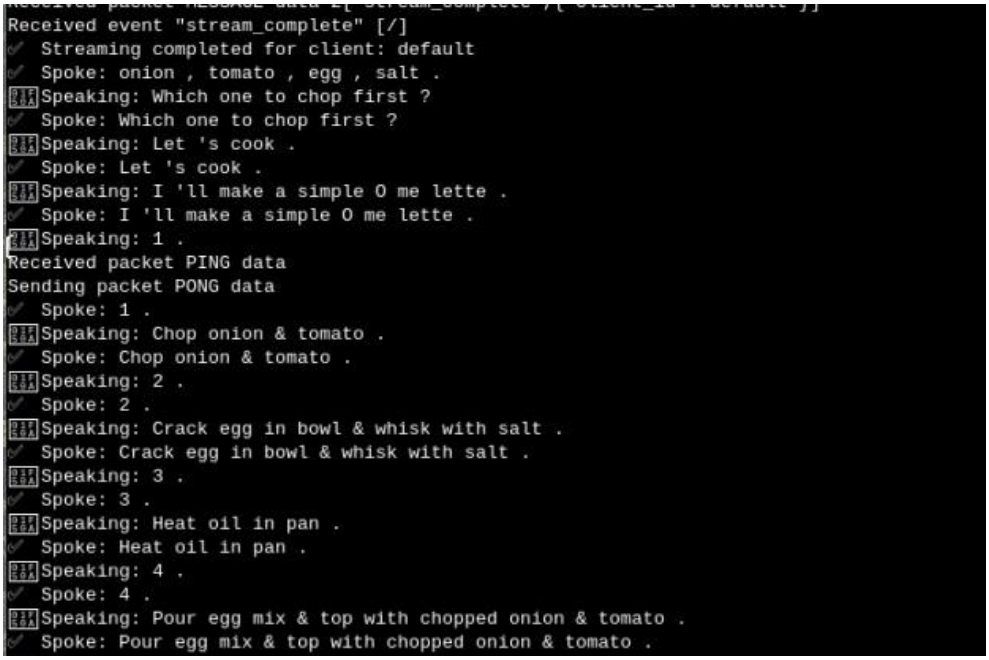

Figure 3: Client-Side output display

1.7 Data Flow Summary

The CookIS workflow progresses through four sequential stages:

- Capture: Smart glasses record real-time video and audio streams, which are transmitted as JSON payloads from the client device.
- Recognition: The server applies YOLOv5 to extract ingredients from video frames and simultaneously processes user commands.
- Reasoning: The Chat API integrates Retrieval-Augmented Generation (RAG) with Pinecone to fetch relevant recipe information and employs LLaMA to generate context- aware instructions.
- Response: Generated instructions are streamed back to the client via SocketIO and converted into speech using gTTS, providing hands-free guidance. As per the figure. As per Figure 4.

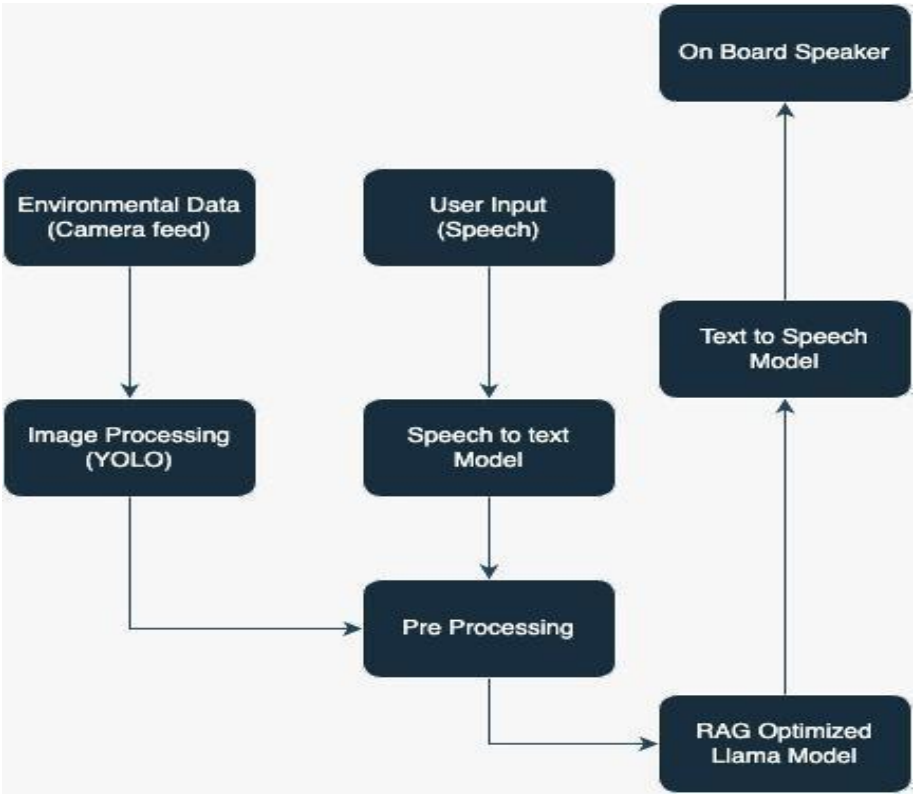

Figure 4: Data flow diagram

Figure 5: Integrated wearable glasses and audio-visual interface.

1.8 Reproducibility Notes

To enable replication, the following dependencies and considerations must be addressed:

- YOLOv5 Weights: A custom-trained model (best.pt) is required, trained on a food-specific dataset.
- Pinecone Indexing: A valid Pinecone API key and index setup are mandatory for

vector storage and retrieval. LLaMA Deployment: The Ollama framework must be configured to host the LLaMA 3.1 model locally.

- Speech Recognition: The Google Speech Recognition API requires internet connectivity; offline ASR frameworks such as Vosk may serve as substitutes.
- Text-to-Speech: gTTS also requires internet access; offline TTS alternatives such as
- pyttsx3 can be used in constrained environments.

2. Theory And Calculation

2.1 Theoretical Foundation

The design of CookIS is grounded in established theories of computer vision, natural language processing, and human–computer interaction. The object detection task relies on convolutional neural networks (CNNs), and YOLOv5 uses a grid-based approach to perform bounding-box regression and classification to identify ingredients in real time. This theoretical basis allows rapid multi-class recognition from streaming video data.

For speech interaction, automatic speech recognition (ASR) is based on acoustic and language models. Google's Speech-to-Text applies deep recurrent or transformer-based networks to map audio waveforms into phonemes and words, ensuring accurate transcription of spoken commands. The system's contextual intelligence is derived from retrieval-augmented generation (RAG) combined with the LLaMA 3.1 model. Recipe data is pre-processed into textual segments, which are then embedded into high-dimensional vectors (e.g., 384 dimensions) using a sentence transformer. These embeddings, along with metadata, are stored in Pinecone, a vector database optimised for similarity search. At runtime, user inputs (detected ingredients + transcribed commands) are converted into embeddings in the same vector space. A cosine similarity search retrieves the most relevant recipe chunks from Pinecone:

$$sim(q, d) = \frac{q \cdot d}{\|q\| \, \|d\|} \tag{1}$$

The top-k retrieved segments are then formatted into a structured prompt and passed to LLaMA 3.1, an attention-based transformer large language model. LLaMA processes this context together with the query to generate stepwise cooking instructions. The model's self-attention layers allow it to track both short-term context (current step) and long-term context (overall recipe flow), ensuring continuity and coherence in generated instructions. Finally, text-to-speech synthesis (TTS), implemented using gTTS, is grounded in digital signal processing and sequence-to-sequence models that transform textual input into intelligible, natural-sounding speech, providing an intuitive user interface.

2.2 Practical Implementation and Calculation

The theoretical principles are operationalised in CookIS through a client–server architecture.

- The Raspberry Pi preprocesses camera frames at ~30 FPS, encodes them, and transmits them to the server. YOLOv5 inference operates at approximately 40 ms per frame, ensuring near real-time recognition.
- Audio signals are sampled at 16 kHz, segmented into frames, and passed to the Google STT API, which yields word-level transcriptions with a latency of ~200 ms per utterance.
- Detected ingredients and transcribed commands are embedded into 384-dimensional vectors using a sentence-transformer and matched against a vector database (Pinecone). The most relevant recipe context is retrieved in O(log n) time complexity, where n is the number of stored recipe segments.
- The LLaMA 3.1 model, running in quantised 8B form, generates instructions in chunks (~10–15 tokens per second), enabling streamed, low-latency responses.
- The gTTS module converts these text chunks to MP3 audio files, each averaging <50 KB, which are sequentially played back to the user via the Raspberry Pi.

Through this, CookIS demonstrates how advanced AI techniques can be translated into a real-world, interactive cooking assistant.

3. Results and Discussion

The CookIS prototype was evaluated to verify its effectiveness in enabling hands-free, context-aware cooking assistance. The system was deployed on a Raspberry Pi 5 connected to smart glasses and tested in a home kitchen environment, including common cooking scenarios such as ingredient identification, recipe retrieval, and step-by-step instructions. While the system demonstrated promising outcomes, several challenges and errors were also encountered during testing.

3.1 System Performance

The client–server pipeline demonstrated stable operation with minimal latency. On average, video frames were processed at ~10 FPS, which was sufficient for ingredient recognition without perceptible delay. Audio transcription using the Google Speech Recognition API achieved high accuracy for clear speech, though background noise occasionally led to misinterpretations. The end-to-end delay from user voice command to spoken response was approximately 2–3 seconds, which was acceptable for interactive cooking guidance. However, two key performance issues were observed. First, when users repeated the same command or provided invalid commands, the model's conversation history often broke down, leading to inaccurate or out-of-context replies. Second, when the generated text responses included special characters such as *, the speech synthesis engine (gTTS) read them literally as “asterisk,” producing unnatural spoken instructions.

3.2 Functional Outcomes

The YOLOv5 model successfully recognised common cooking ingredients such as onions, tomatoes, and spices with confidence scores above 0.85. Detected ingredients were dynamically updated in the session and used to retrieve relevant recipes from the knowledge base. The integration of Retrieval-Augmented Generation (RAG) with LLaMA

enabled personalised recipe suggestions based on available ingredients. Instructions generated are delivered via SocketIO, allowing uninterrupted interaction. The Client Speech module, using gTTS, provided clear audio playback, enabling users to continue cooking without pausing to check a screen. Despite these outcomes, several errors were identified. Additionally, once a recipe was completed, the system continued delivering instructions until the user explicitly ended the session with the "close session" command.

3.3 User Experience

Preliminary trials with novice users showed that CookIS reduced cognitive load by eliminating the need to manually search for recipes or constantly refer to written instructions. Spoken instructions were particularly beneficial when hands were occupied, supporting the system's aim of enhancing accessibility for beginners, elderly individuals, and people with disabilities. Nevertheless, users highlighted several usability limitations. The system lacked the ability to revisit previous steps in the recipe, which made error correction difficult. Progression through instructions required explicit user confirmation after each step, as no live video analysis was implemented to automatically track task completion. These constraints occasionally disrupted the natural flow of cooking and required additional user intervention.

3.4 Comparison with Existing Work

Unlike conventional recipe recommendation systems or smart kitchen devices, CookIS integrates multiple AI capabilities, ingredient recognition, speech-based interaction, retrieval-augmented recipe generation, and text-to-speech guidance, into a unified wearable solution. Previous work on smart cooking assistants has largely focused on mobile applications or IoT-based recipe retrieval, without incorporating real-time vision or large language models. In contrast, CookIS demonstrates how combining YOLOv5 with LLaMA and RAG can create an adaptive, interactive culinary assistant that bridges the gap between static recipe instructions and dynamic cooking environments. However, compared to existing commercial voice assistants such as Alexa and Google Home, CookIS remains less robust. The lack of advanced context handling and the absence of multimodal feedback mechanisms highlight areas that need further refinement.

3.5 Discussion

The results confirm the feasibility of implementing a low-cost, real-time, AI-powered cooking assistant using readily available hardware and open-source frameworks. The system shows promise in improving accessibility and efficiency in home cooking. At the same time, the limitations encountered point to areas for further research and development:

- Context Handling: Conversation history must be made more resilient to repeated or erroneous commands to avoid breakdowns in dialogue flow.
- Speech Output Quality: Post-processing of generated text is required to remove or normalise symbols such as , which currently disrupt spoken instructions.
- Ingredient Detection: Improved dataset quality and model retraining are necessary to
- enhance YOLOv5's accuracy and reduce false detections.
- Session Management: Automatic session termination upon recipe completion

would improve usability.

- Step Navigation: Features to revisit previous steps and track progress through live video analysis could significantly enhance user experience.

Overall, the evaluation highlights both the strengths and current shortcomings of CookIS. While it validates the concept of an AI-powered, hands-free cooking assistant, it also establishes a roadmap for improvements to ensure reliability, adaptability, and user satisfaction in real-world deployments.

4. Conclusion

This study presented CookIS, a wearable AI-powered cooking assistant that combines text-to-speech, computer vision, speech recognition, and retrieval-augmented generation technologies to provide hands-free, real-time, and context-aware culinary advice. The system was made accessible and useful by utilising open-source software frameworks and inexpensive hardware, such as a Raspberry Pi combined with wearable glasses. The system's capabilities were evaluated experimentally in home kitchen settings. These included YOLOv5 ingredient recognition, Google Speech Recognition voice command transcription, Pinecone-powered vector store recipe retrieval, and LLaMA model step-by-step instructions. Findings demonstrated that the pipeline could deliver a continuous stream of instructions with a reasonable latency, lowering users' cognitive load and enhancing accessibility, particularly for novices and senior citizens who gain the most from spoken instructions. Simultaneously, several issues were identified that affected the system's dependability and user experience. When commands were repeated or incorrect, the conversation flow was disrupted, and responses veered off topic. Due to the prototype's lack of live video-based task monitoring, users were unable to go back and review earlier steps, spoken instructions occasionally contained unwanted symbols, and recipes continued until sessions were manually ended. Despite these limitations, CookIS offers a useful proof of concept that shows that integrating computer vision, wearable technology, and large language models for smart kitchen applications is feasible. The results highlight how these systems can improve accessibility, self-reliance, and productivity in routine cooking tasks. To increase the robustness of ingredient detection, future research should focus on expanding training datasets, improving dialogue context management, and improving the correspondence between identified ingredients and recipes. Including text-to-speech options and offline speech recognition would also improve dependability in settings with poor connectivity. Furthermore, a more organic and flexible interaction experience might be offered by incorporating multimodal feedback, such as automated step tracking or gesture recognition. With these improvements, CookIS may evolve into a dependable and adaptable smart cooking assistant, bridging the gap between conventional recipe advice and intelligent, context-aware culinary assistance.

5. Acknowledgements

For providing the facilities and opportunity to complete this project, the authors would like to sincerely thank Dr S. Viswanatha Rao, the principal. We are grateful for the invaluable advice and support provided by Dr Jisha John, Head of the Department of Computer Science and Engineering. Our sincere gratitude also goes out to Dr. Tessy Mathew,

Associate Professor, who served as the project coordinator and provided constant support and helpful advice during this work. We are also grateful to the Department of Computer Science and Engineering's faculty and staff for their support and encouragement. Finally, we would like to express our gratitude to our families, friends, and classmates for their encouragement and support, which enabled us to successfully finish this work.

Conflict Of Interest

The authors declare no conflict of interest.

References

[1] Majil, I., Yang, M. T., & Yang, S. (2022). Augmented reality-based interactive cooking guide. *Sensors*, *22*(21), 8290.

[2] Vir, R., & Madinei, P. (2024). Archef: An iOS-based augmented reality cooking assistant powered by multimodal Gemini LLM. *arXiv preprint arXiv:2412.00627*.

[3] Ruiz, G., Kilambi, S. C., Soni, P., George, K., & Panangadan, A. (2024, February). Design of a multisensor system for a smart cooking assistant. In *2024 IEEE First International Conference on Artificial Intelligence for Medicine, Health and Care (AIMHC)* (pp. 157-161). IEEE.

[4] Babu, B. R., Suma, B., Morla, S., & Kumar, I. H. AI-Based Recipe Generator using GPT-4–Created a system that generates custom recipes based on user inputs.

[5] Chopra, P., Sagar Srivastava, G., Raj, V., & Kumar Saini, N. (2023, May). Smart Cooking: Ingredient-Based Recipe Recommendation Through Object Recognition. In *Proceedings of the KILBY 100 7th International Conference on Computing Sciences*.

[6] Wang, H., Kedia, K., Ren, J., Abdullah, R., Bhardwaj, A., Chao, A., ... & Choudhury, S. (2024). Mosaic: Modular foundation models for assistive and interactive cooking. *arXiv preprint arXiv:2402.18796*.

[7] Ruiz, G., Kilambi, S. C., Soni, P., George, K., & Panangadan, A. (2024, June). Visual and Thermal Imaging Camera-Based System for a Smart Cooking Assistant. In *2024 IEEE 6th Eurasia Conference on Biomedical Engineering, Healthcare and Sustainability (ECBIOS)* (pp. 499-503). IEEE.

[8] Jabeen, H., Weinz, J., & Lehmann, J. (2020, July). AutoChef: Automated generation of cooking recipes. In *2020, IEEE Congress on Evolutionary Computation (CEC)* (pp. 1-7). IEEE.

[9] Noever, D., & Noever, S. E. M. (2023). The multimodal and modular ai chef: Complex recipe generation from imagery. *arXiv preprint arXiv:2304.02016*.

[10] Senath, T., Athukorala, K., Costa, R., Ranathunga, S., & Kaur, R. (2025). Large language models for ingredient substitution in food recipes using supervised fine-tuning and direct preference optimization. *Natural Language Processing Journal*, 100177.

[11] Mohbat, F., & Zaki, M. J. (2025, July). Kerl: Knowledge-enhanced personalized recipe recommendation using large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 19125-19141).

[12] Zhang, M., Tian, G., Zhang, Y., & Liu, H. (2022). Sequential learning for ingredient recognition from images. *IEEE Transactions on Circuits and Systems for Video Technology*, *33*(5), 2162-2175.

[13] Rodrigues, M. S., Fidalgo, F., & Oliveira, A. (2023). RecipeIS,Recipe recommendation system based on recognition of food ingredients. *Applied Sciences*, *13*(13), 7880.