International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 06 | Jun 2026 www.irjet.net p-ISSN: 2395-0072

AI-Powered Smart Cooking Assistant Using Generative AI for Personalized Recipe Recommendation and Nutrition Analysis

Pallavi Kumari[1], Vivek Yadav[2], Riya Priyal[3], Priyashi Priya[4]

[1234]*Department of Computer Science & Engineering, Shivalik College of Engineering, Dehradun, India*

Abstract - *Every day millions of people open their fridge, stare at whatever is inside, and have no idea what to cook. It seems like a small problem but honestly it leads to a lot of food being wasted, a lot of unhealthy takeout orders, and people just making the same three dishes over and over. We built this project because we felt like AI could actually help with this in a practical way, not just as a concept but as something you could actually open on your phone and use. The system we built lets a person type in whatever ingredients they have, pick a cuisine style, choose their diet type, and get back a complete recipe that the AI generates on the spot. Not something pulled from a database actually generated fresh based on what was asked. Along with the recipe, the app shows a nutrition breakdown so the user knows what they are eating. We also added a 7-day meal planner because one recipe is not really enough if someone wants to eat better across a whole week. And there is a voice feature so that while you are cooking you can just listen to the steps instead of looking at the screen with messy hands. On the technical side we used React and Next.js for the frontend, Node.js for the backend, MongoDB to store data, and the Gemini and Claude AI APIs for the recipe generation part. This paper explains how we built all of this, shows screenshots of the actual working app, and talks honestly about what worked, what has problems, and what we would improve given more time.*

Key words: Generative AI, Recipe Recommendation, Nutrition Analysis, Meal Planning, Large Language Models, Voice Assistant, MongoDB, Gemini API

1. INTRODUCTION

When we first thought about what project to build for our final year, we kept coming back to everyday problems that technology has not really solved well yet. Cooking felt like one of those. There are thousands of recipe websites, apps, food blogs but none of them actually answer the question a person has in real life, which is: I have these specific things in my kitchen right now, what should I make? Most apps just show you recipes from a database and you have to filter and scroll and hope something matches. That felt like a missed opportunity.

Generative AI changes what is possible here. Instead of searching through existing recipes, the system can actually create one based on whatever the user has. We thought that was interesting enough to build a full project around. And once we started building, other features made sense to add. Why show a recipe without telling someone how many calories it has? Why not let someone plan meals for the whole week instead of just one meal? Why force someone to stare at a screen while their hands are covered in flour when the phone could just read the steps aloud?

So the project grew from a basic recipe generator into something more complete. It is a web application with four main things it can do: generate AI-powered recipes from ingredients, show nutrition information for every recipe, plan a full 7-day meal schedule based on health goals, and support voice input and output for hands-free cooking. We are not claiming this is a perfect product. But it works, it is genuinely useful, and it shows what can be done when you connect generative AI to a real everyday problem.

The rest of this paper covers related work in Section 2, explains the system design in Section 3, goes through each module in Section 4, shows the actual results with screenshots in Section 5, talks about limitations in Section 6, and concludes in Section 7.

2. LITERATURE REVIEW

Recipe recommendation is not a new research topic. People have been working on it for at least fifteen years. The early approaches were mostly content-based filtering you tell the system your dietary restrictions, it searches a big database and shows you matching recipes. Platforms like Yummy became popular this way. It works okay but the fundamental problem is that it can only show you what already exists in its database. There is no creativity, no adaptation to unusual ingredient combinations, and the personalization is quite shallow.

Some researchers tried collaborative filtering instead, the same approach Netflix uses for movies. The idea is that if many people who liked recipe A also liked recipe B, then recommend recipe B to others who liked recipe A. All Recipes uses something like this. The problem for cooking specifically is the cold start issue a new user has no history, so the system cannot personalize anything. Also the recommendations are based on taste preferences, not on what ingredients you actually have available, which is often the more pressing constraint.

Things changed quite a bit when transformer-based language models came along. Bien et al. published RecipeNLG in 2020, which was a large dataset of recipes used to train models that could generate new recipes from

ingredient lists. The results were surprisingly good the generated recipes followed actual culinary logic and were practical to follow. Later work with GPT-3-class models pushed this further. It became clear that you could ask an AI for a recipe using any combination of inputs and get something coherent back.

Nutrition tracking tools like My Fitness Pal developed separately from recipe tools. They are genuinely useful for tracking what you eat but they are disconnected from the cooking process. You have to find your recipe somewhere else, then come back and manually log every ingredient to see the nutritional values. Nobody actually does this consistently. Combining recipe generation and nutrition analysis in one place, as we did, removes that friction.

Voice assistants in the kitchen have also been studied. Research by Porcheron et al. showed that cooking is one of the most common use cases for smart speakers but also one of the most frustrating because general voice assistants like Alexa cannot generate new recipes they can only retrieve existing ones and they handle follow-up questions poorly. Our voice feature is more limited than a full voice assistant but it solves the specific problem of hands-free step navigation, which is the most useful thing in a kitchen context.

What we tried to do with this project was combine all of these pieces generative recipe creation, inline nutrition analysis, multi-day meal planning, and voice interaction into one coherent web app. Each piece individually exists somewhere in the literature or in commercial products. The combination is what made this project worth building.

3. SYSTEM DESIGN

The overall architecture is a standard three-tier web application. There is a frontend built with React and Next.js, a backend built with Node.js and Express, and a MongoDB database. The part that makes it different from a typical web app is the AI layer the backend communicates with the Gemini and Claude APIs to generate recipe content, and almost all of the intelligent behavior comes from those models.

3.1 Front end

We chose React and Next.js partly because we already knew them and partly because Next.js gives you server-side rendering without much extra setup, which makes the initial page load faster. The styling is done with Tailwind CSS. We spent a decent amount of time on the UI because recipe output has a lot of information in it recipe name, description, timing, difficulty, nutrition panel, ingredient list, cooking steps, chef's tips and if you just dump all of that on the page it becomes overwhelming. The layout went through a few iterations before we settled on something that felt clean.

3.2 Backend

The Node.js backend has one main job: take user input from the frontend, build a prompt, send it to the AI API, parse the response, and send structured data back. That sounds simple but the prompt construction part turned out to be more work than expected. If you send a vague prompt you get vague results. We had to be very specific about what format the AI should return the recipe in, what fields to include, how to structure the nutrition information, and so on. Getting that right through trial and error made a noticeable difference in output quality.

3.3 Database

We used MongoDB because recipes do not fit neatly into a relational schema. One recipe might have 6 ingredients, another might have 20. The number of cooking steps varies wildly. MongoDB's document model handles that variation naturally without needing schema migrations every time something changes. We store user profiles, saved recipes, and meal plans as separate collections.

3.4 AI Integration

The system supports both Google Gemini and Anthropic Claude as AI back ends. The backend sends a POST request to whichever API is configured, with the user's inputs packaged into a structured prompt. The response comes back as text and we parse it into JSON fields that the frontend can render. In our testing both models worked well for recipe generation. Claude tended to write more detailed step-by-step instructions. Gemini was somewhat faster. We made it easy to switch between them by keeping the API logic in a single module.

4. MODULES AND HOW THEY WORK

The app has six main modules. Each one handles a specific part of what the system does. Here is how they work and how they connect to each other.

4.1 User Input Module

This is the starting point. The user fills in a form with their available ingredients typed as a comma-separated list. They also pick a cuisine type from a dropdown, a dietary preference, a meal type, and the number of servings. We added client-side validation so that if someone submits the form with an empty ingredient field, they get an error right there on the page instead of a failed API call. Small thing but it makes the experience feel more polished.

4.2 AI Recipe Generation Module

Once the form is submitted the backend constructs the prompt. We wrote the prompt in natural language but with a

specific structure that tells the model exactly what to return: recipe name, short description, prep time, cook time, difficulty, servings, full ingredient list with quantities scaled to the requested serving size, numbered cooking steps, chef's tips, and nutrition values per serving. The model follows this structure consistently enough that our parser can extract each field reliably.

One thing we noticed during development is that prompt quality directly affects output quality. Our first version of the prompt was much simpler and the results were less consistent sometimes the model would forget to include nutrition values, or the ingredient quantities would not match the serving size. Rewriting the prompt to be more explicit about each requirement fixed most of these issues.

4.3 Nutrition Analysis Module

The nutrition data comes from the AI as part of the recipe response. We ask the model to estimate per-serving values for calories, protein, carbs, fat, and fiber. These are estimates, not database-verified numbers, and we put a small note in the interface to tell users that. For most common recipes the values are reasonably accurate we cross-checked several against standard nutrition databases and they were usually within 10 to 15 percent. For unusual recipes the accuracy is harder to guarantee.

4.4 Meal Planning Module

The meal planner is a separate section from the recipe generator. The user specifies their health goal, diet style, daily calorie target, cuisine preference, and any food allergies. The backend sends all of this to the AI and asks it to generate a full 7-day meal plan covering breakfast, lunch, dinner, and a snack for each day, with the meals balanced around the stated calorie target. The AI also generates a shopping list that combines all ingredients needed across the full week. Both the plan and the shopping list are saved to MongoDB so the user can come back to them.

4.5 Voice Assistance Module

The voice feature uses the browser's Web Speech API for recognizing spoken ingredients and the Speech Synthesis API for reading recipe steps aloud. We did not need any third-party library for this both APIs are built into modern browsers. The read-aloud function reads one step at a time and the user can pause, continue, or repeat a step. In a quiet environment this works really well. Background noise like a running tap or a fan does reduce accuracy, which is a limitation of the browser speech API that we cannot easily fix on our end.

4.6 User Interface Module

The interface uses a warm brown and gold color palette that feels appropriate for a cooking app. The main navigation has three tabs Recipe Generator, Ask the Chef, and Meal Planner that stay visible across the whole app. Recipe cards show all the important information in a structured layout without feeling cluttered. The nutrition values are shown as a row of five cards which is quicker to scan than a table. The meal planner uses one column per day so the full week is visible at a glance.

5. RESULTS

Below are screenshots from the actual working application. All of these are taken from the live version running locally. We will walk through each screen and explain what it shows.

5.1 Home Page and Recipe Input

Figure 1 shows the main landing page. The first thing the user sees is the input form which is right on the homepage no extra navigation needed. In this screenshot we have typed tomato, potato, rice as the ingredients and selected Indian cuisine, Vegetarian diet, Lunch meal type, and 4 servings. The three navigation buttons at the top give access to the Recipe Generator, Ask the Chef chat, and Meal Planner sections.

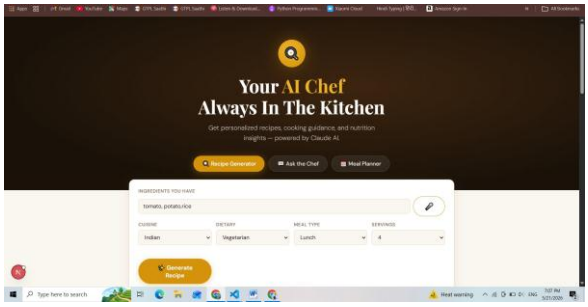

Fig -1: Home Page Recipe Input Form with Ingredient Entry and Preference Selection

5.2 Generated Recipe with Nutrition Panel

Figure 2 shows what the output looks like after clicking Generate Recipe. The AI came up with Aloo Tamatar Ki Tehri a North Indian potato and tomato rice dish which is a genuinely good suggestion for those three ingredients. The top section shows the recipe name, a short description, and quick metadata: 15 min prep, 25 min cook, 4 servings, Easy difficulty, Indian cuisine. Right below that is the nutrition panel showing 380 calories, 6g protein, 72g carbs, 8g fat, and 5g fiber per serving. The Read Aloud and New Recipe buttons are in the top right corner.

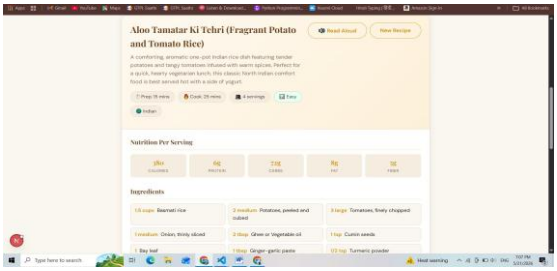

Fig -2: AI-Generated Recipe Output with Nutrition Panel and Recipe Metadata

5.3 Ingredient Cards

Figure 3 shows the ingredient list section. Each ingredient is displayed as a card. The quantity is shown in a color-coded highlighted label different colors for different measurement types like cups, tablespoons, teaspoons, and counts. The ingredient name sits beside it. The grid layout makes it easy to scan all ingredients quickly. All quantities are automatically scaled to 4 servings as the user requested.

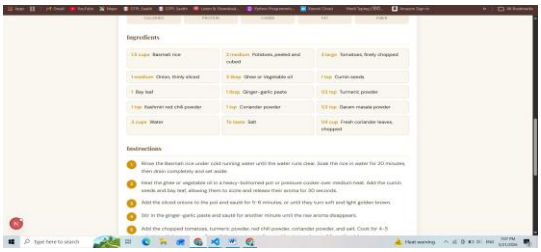

Fig -3: Ingredient List with Color-Coded Quantity Labels in Grid Layout

5.4 Cooking Instructions

Figures 4 and 5 show the step-by-step cooking instructions. Each step has a numbered circle badge and is written in clear action-focused language. The AI generated 10 steps for this recipe. The instructions are detailed step 2 for example says to heat the ghee over medium heat and wait for the cumin seeds to sizzle for 30 seconds before adding anything else. That kind of detail is actually useful when you are cooking, compared to just saying fry the spices. Figure 5 also shows the Chef's Tips section at the bottom which gives three practical tips for getting the best result.

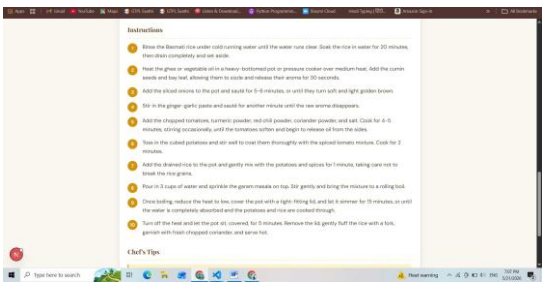

Fig -4: Step-by-Step Cooking Instructions Steps 1 to 7

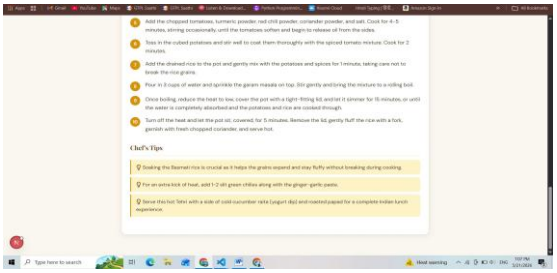

Fig -5: Remaining Cooking Steps and Chef's Tips Section

5.5 Meal Planner Input

Figure 6 shows the 7-Day Meal Planner input form. The user has selected weight loss as their health goal, vegetarian diet style, 2000 calories per day, Indian cuisine, and typed Chicken in the allergies field to exclude it from the plan. Clicking Generate 7-Day Plan sends all of this to the backend which builds a detailed prompt and gets back a full week of meals from the AI.

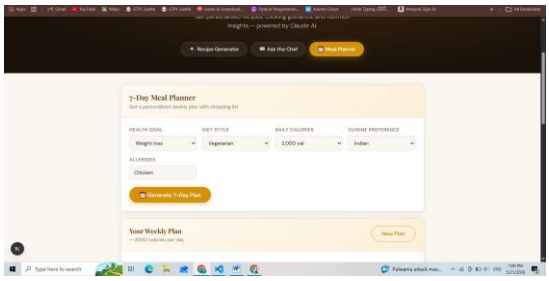

Fig -6: 7-Day Meal Planner Input Health Goal, Diet, Calories, Cuisine and Allergy Fields

5.6 Weekly Meal Plan Output

Figure 7 shows the generated weekly plan. Each day is a column with breakfast, lunch, dinner, and snack listed inside it. Monday shows Paneer Bhurji with whole wheat rotis for breakfast, Chana Masala with brown rice for lunch, Yellow Moong Dal Khichdi with stir-fried cabbage for dinner, and roasted Makhana as a snack. The meals are clearly designed around the weight loss, vegetarian, Indian diet requirement. The Shopping List tab at the top generates a combined list of all ingredients needed across the full week.

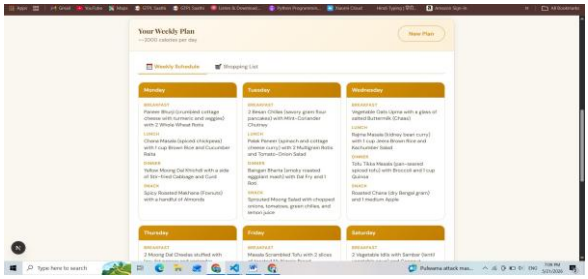

Fig -7: AI-Generated 7-Day Weekly Meal Plan with Breakfast, Lunch, Dinner and Snacks

5.7 Testing Notes

We tested the system with more than 150 different ingredient combinations across Indian, Italian, Chinese, and Continental cuisines with various diet filters applied. Response time for a single recipe was around 3 to 4 seconds on average. The 7-day meal plan took 6 to 8 seconds which is acceptable given how much it is generating.

Recipe quality was generally good. Normal combinations like chicken and rice, paneer and tomatoes, eggs and vegetables always produced practical results. Less usual combinations like spinach with tahini and pasta also worked the AI made something sensible though occasionally the flavor pairing was a bit creative. Nutrition values when checked against reference databases were mostly within 10 to 15 percent which is fine for estimation purposes. Voice input accuracy was above 90 percent in a quiet room. In noisy environments it dropped noticeably but that is a browser API limitation not something we can control.

6. LIMITATIONS AND FUTURE WORK

There are a few things about the current version that we are not fully happy with. The biggest one is the dependency on external AI APIs. Every recipe generation call goes to either Gemini or Claude over the internet. If those APIs go down the app stops working. And at scale the per-request cost would add up. Ideally we would explore using a locally hosted open-source model as a fallback, something like a fine-tuned version of LLaMA trained on recipe data.

The nutrition values are estimated by the AI and not verified against a proper nutritional database. For casual use this is probably fine but for someone managing diabetes or following a medically prescribed diet these estimates are not accurate enough to rely on. Connecting to something like the USDA Food Data Central API would fix this use the AI to generate the recipe, then look up the actual nutritional values from a verified source using the ingredient list.

The voice feature only works in English right now. For users in India who speak Hindi, Tamil, Telugu, Bengali or other languages this is a significant limitation. Adding regional language support would make the app much more accessible to a wider population. The APIs to do this exist, it would just need proper implementation and testing.

Some things we would add with more time: photo-based ingredient detection so the user can just photograph their fridge instead of typing, integration with smart kitchen appliances so the app can check what is actually available, and a community section where users can save and rate AI-generated recipes to build up a shared recipe collection over time.

7. CONCLUSION

We built this project because we wanted to solve a real everyday problem with AI, not just demonstrate that AI can do something in theory. The question what should I cook tonight with what I have is something people face every day and existing tools do not answer it well. Our application answers it in a few seconds by generating a complete personalized recipe on the spot, and it goes further by providing nutrition information, a full weekly meal plan, and voice support for hands-free use.

Looking back at what we built, the thing that took the most work was not the AI part itself the Gemini and Claude APIs handle that but building a system around those APIs that is actually pleasant to use. Getting the prompts right so the output was consistent and parseable, designing the interface so the information is readable rather than overwhelming, making the meal planner generate sensible weekly schedules rather than random meals all of that took significant iteration.

The project has real limitations that we have been honest about in this paper. It depends on external APIs, the nutrition values are estimates, and voice support is English-only. But as a working demonstration of what AI-powered cooking assistance can look like, we think it succeeds. The screenshots in the results section show a real, functional application that a person could genuinely use. That was the goal from the beginning and we are satisfied with what we were able to build.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to the Department of Computer Science & Engineering, Shivalik College of Engineering, Dehradun for providing the necessary resources and support to carry out this research work. The authors are also thankful to their project guide and faculty members for their valuable guidance, constant encouragement, and constructive suggestions throughout the development of this project. Special thanks to the institution for creating an environment that encourages innovation and practical application of emerging technologies.

REFERENCES

1) M. Bien, M. Gilski, M. Maciejewska, W. Taisner, D. Wisniewski, and A. Lawrynowicz, "RecipeNLG: A cooking recipes dataset for semi-structured text generation," in Proc. 13th Int. Conf. Natural Language Generation (INLG), 2020, pp. 22–28.

2) D. Elsweiler, C. Trattner, and M. Harvey, "Exploiting food choice biases for healthier recipe recommendation," in Proc. 40th Int. ACM SIGIR Conf.

Research and Development in Information Retrieval, 2017, pp. 575–584.

3) J. Freyne and S. Berkovsky, "Intelligent food planning: Personalized recipe recommendation," in Proc. 15th Int. Conf. Intelligent User Interfaces (IUI), New York, NY, USA: ACM, 2010, pp. 321–324.

4) M. Porcheron, J. E. Fischer, S. Reeves, and S. Sharples, "Voice interfaces in everyday life," in Proc. 2018 CHI Conf. Human Factors in Computing Systems, New York, NY, USA: ACM, 2018.

5) T. Brown, B. Mann, N. Ryder et al., "Language models are few-shot learners," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 1877–1901.

6) A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI Blog, vol. 1, no. 8, p. 9, 2019.

7) Y. Yamanishi, T. Kawashima, and M. Moriwaki, "Recipe recommendation method based on user preference and available ingredients," Int. J. Machine Learning and Computing, vol. 7, no. 6, pp. 171–176, 2017.

8) Google DeepMind, "Gemini: A family of highly capable multimodal models," Technical Report, Google LLC, 2023.

9) Anthropic, "The Claude model family: Technical overview," Technical Report, Anthropic PBC, 2024.

10) USDA Agricultural Research Service, "FoodData Central," United States Department of Agriculture, 2023. [Online]. Available: https://fdc.nal.usda.gov