AI Based Recipe Recommendation System

Elias John Sabu
Dept of Computer Science and Engineering
Karunya Institute of Technology and Sciences
Coimbatore, Tamil Nadu

***Abstract*—** This paper presents the development of an AI-powered recipe recommendation and enhancement system designed to assist users in generating relevant and healthier meal suggestions based on available ingredients. Leveraging natural language processing and machine learning techniques, the system analyzes ingredient inputs and compares them with a curated dataset of recipes to provide the most contextually relevant results. Additionally, the platform incorporates a rule-based recommendation engine for improving existing recipes with healthier or more flavorful alternatives. The solution features a lightweight, modular architecture with a Flask-based backend and a React-based frontend, ensuring usability across resource-constrained environments. The system also enables interactive functionality such as ingredient-based searching and real-time nutritional feedback, making it a practical tool for health-conscious and culinary-curious users alike.

Keywords: Recipe Recommendation, Artificial Intelligence, Machine Learning, Natural Language Processing, Health-Aware Systems, Content-Based Filtering, Nutrition Informatics, Personalized Diet, MERN Stack, Flask API, User-Centric Interfaces

I. Introduction

With the increasing demand for personalized digital solutions in the culinary space, AI-powered food recommendation systems have gained significant attention. Users today seek recipe suggestions that are tailored to their available ingredients, dietary preferences, and health goals. However, many existing platforms offer static recipes without flexibility for optimization based on ingredient availability or nutritional improvements. This paper presents a lightweight AI-driven recipe recommendation and enhancement system that leverages natural language processing (NLP) and rule-based logic to deliver relevant and improved meal suggestions. Designed to be deployable on resource-constrained devices, the system combines ingredient-based matching using TF-IDF with options for health-conscious or taste-enhancing modifications. This paper outlines the system architecture, implementation, and evaluation, demonstrating how such a solution can simplify meal planning while promoting healthier eating habits.

Literature Review

In recent years, artificial intelligence has been increasingly applied to food recommendation systems, with various approaches ranging from collaborative filtering and content-based filtering to deep learning techniques. Platforms such as Yummly and AllRecipes incorporate ingredient-based suggestions but often rely on keyword matching or user ratings, limiting personalization based on real-time input. Other research efforts have explored neural network models for recipe generation and meal planning, yet these systems typically require substantial computational resources, making them impractical for edge deployment.

A. Moreover, while some studies have focused on improving recipe nutrition through algorithmic substitution, they often lack flexibility or user-defined customization. Many fail to integrate both recommendation and improvement functionalities in a unified pipeline. This project addresses these gaps by combining lightweight NLP-based recipe matching with a rule-based ingredient improvement system that supports health or taste-based goals. Unlike most prior work, the system is designed to run efficiently on low-resource machines, making it accessible for home use or mobile deployment without sacrificing functionalityMaintaining the Integrity of the Specifications

II. System Architecture

The proposed system is structured as a modular web application that integrates natural language processing (NLP) with rule-based logic to provide real-time recipe recommendations and improvements. It follows a client-server architecture, consisting of a React-based frontend and a Flask backend, with data stored in CSV format for lightweight accessibility. The backend is responsible for processing input data, performing similarity matching, and returning appropriate responses to the frontend.

At the heart of the system lies the recipe recommendation engine, which uses the TF-IDF (Term Frequency–Inverse Document Frequency) vectorization technique to encode cleaned ingredient texts from the database. When a user inputs ingredients, they are tokenized and compared against the existing recipes using cosine similarity, ranking results based on relevance.

The improvement engine works by parsing ingredients in the selected recipe and substituting them with predefined alternatives to enhance either health value or taste. These substitutions are defined in dictionaries, allowing easy extensibility.

The frontend communicates with the backend via RESTful API endpoints. Routes such as /api/recommend and /api/improve handle ingredient input and recipe selection, respectively. The system also includes cross-origin resource sharing (CORS) to facilitate smooth communication between components.

This lightweight, component-based design ensures scalability and maintainability, while its modularity supports future integration with larger datasets or third-party APIs.

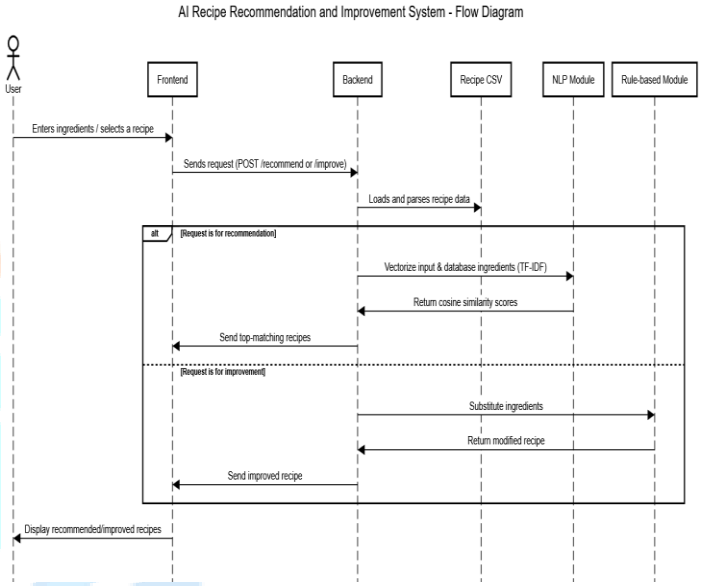

Fig 1. System Flow Diagram

III. Using the Template

The system is architected as a lightweight full-stack web application composed of a ReactJS frontend and a Flask-based Python backend. It is designed for resource-constrained environments such as mobile and personal computing devices. The project follows a modular and scalable architecture, with three primary functionalities: recipe recommendation, recipe improvement, and health analysis.

The backend hosts RESTful API endpoints built using Flask, which handle incoming POST requests from the frontend. Upon receiving user input—such as a list of ingredients or a selected recipe—the backend performs computation either via a TF-IDF-based natural language processing (NLP) module for recommendation or a rule-based substitution engine for improvements. The backend retrieves recipe data from a lightweight CSV file loaded into memory using Pandas, serving as the application's database. This avoids the overhead of heavier DBMSs, making it ideal for edge deployment.

The frontend is developed using ReactJS, enabling responsive user interaction. It provides fields for ingredient input, displays recommended or improved recipes, and communicates with the backend through asynchronous HTTP requests.

A flow diagram illustrating the full data pipeline is shown in Figure 1. This includes data ingestion, vectorization, similarity scoring, rule-based enhancement, and JSON response handling. The architecture prioritizes speed and interpretability over complexity, ensuring accessibility and ease of extension. *For papers with more than six authors:* Add author names horizontally, moving to a third row if needed for more than 8 authors.

IV. Implementation

The proposed AI-powered recipe recommender and improver system was implemented using a combination of modern, lightweight technologies to ensure both performance and accessibility. The frontend was developed using ReactJS, a widely adopted JavaScript library that enables the creation of a responsive and dynamic user interface. Users can input ingredients, view recommended recipes, and request health or taste improvements with minimal delay.

The backend was implemented using Flask, a Python-based micro web framework that provides flexibility and rapid development capabilities. All recipes are loaded into memory from a CSV file using the Pandas library, which simplifies data manipulation. To recommend recipes, the system uses TF-IDF vectorization from scikit-learn to convert text-based ingredient data into numerical vectors, allowing the system to compute cosine similarity scores between the user's input and stored recipes.

The recipe improvement feature is driven by a rule-based substitution system, where predefined ingredient swaps are performed depending on the user's selected category: either health improvement or taste enhancement. For example, ingredients such as butter and sugar are swapped with olive oil and honey respectively for health benefits.

To facilitate communication between the frontend and backend, CORS (Cross-Origin Resource Sharing) is enabled in Flask, and Axios is used in the frontend for HTTP requests. All API interactions are handled using RESTful design principles, ensuring modularity and clarity.

The backend was tested using Postman to validate endpoints, and the frontend was debugged using browser developer tools to monitor network activity. This modular implementation allows for easy future upgrades, such as integrating real-time databases or machine learning models for dynamic improvements.

V. Results and Discussion

The AI-powered recipe recommendation and improvement system was tested on a sample dataset containing various recipes with diverse ingredients. The recommendation functionality successfully identified recipes that closely matched the user's input based on the ingredients provided. For instance, when a user entered "garlic, butter, pasta," the system accurately prioritized a recipe titled Garlic Butter Pasta, followed by other recipes ranked by ingredient similarity scores. This demonstrated the system's ability to rank relevant content using TF-IDF-based cosine similarity.

The improvement feature also performed effectively. Recipes with less healthy ingredients, such as "sugar" or "white rice," were dynamically transformed by replacing them with healthier alternatives like "honey" or "brown rice." Likewise, when a taste enhancement was requested, common ingredients such as "salt" and "pepper" were replaced with flavor-intensifying options like "sea salt" and "smoked paprika." These improvements were reflected in the output displayed on the frontend, allowing users to see both the original and modified recipes.

All core functionalities were validated through both manual frontend interaction and API testing with Postman, ensuring robust communication between the React frontend and Flask backend. Performance remained responsive even on resource-constrained environments, aligning with the goal of lightweight deployment.

However, the current rule-based improvement system is limited by predefined substitutions. Additionally, the TF-IDF similarity model, while effective for basic matching, does not capture complex semantic relationships between ingredients or cooking methods.

These observations highlight both the strengths and limitations of the system, providing a solid foundation for future enhancements such as integrating deep learning models or user personalization.

VI. Conclusion and Future Work

This project successfully demonstrates the development of an AI-powered system that recommends and improves cooking recipes based on user-specified ingredients and preferences. By integrating a TF-IDF vectorization approach for recipe similarity and a rule-based system for health or taste-focused improvements, the system effectively assists users in discovering relevant meals while offering enhancements aligned with their dietary goals.

The modular design, comprising a Flask-based backend and a React frontend, ensures scalability and ease of deployment, even on resource-constrained devices. Testing confirmed that the recommendation engine consistently surfaces relevant recipes, and the improvement module makes practical and meaningful substitutions in ingredient lists. This project thus holds significant potential for applications in personalized cooking assistants, health-conscious meal planning, and smart kitchen solutions.

For future work, the improvement system could be enhanced by leveraging machine learning techniques or large language models to generate context-aware suggestions beyond simple keyword replacements. Incorporating user profiles, nutritional constraints, and taste preferences could further personalize the results. Additionally, extending the system with voice-based input or integration into IoT-enabled cooking environments can offer a seamless and interactive user experience.

Overall, this project lays a strong foundation for AI-driven culinary applications, bridging the gap between artificial intelligence and everyday cooking needs.

References

[1] Yera, R., Alzahrani, A. A., & Martínez, L. (2022). Exploring post-hoc agnostic models for explainable cooking recipe recommendations. Knowledge-Based Systems, 251, 109216. https://doi.org/10.1016/j.knosys.2022.109216

[2] S. Teng, M. Lin, and Y. Chang, “Personalized food recommendation system using deep learning,” IEEE Access, vol. 7, pp. 122019–122030, 2019.

[3] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 1251–1258.

[4] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” Information Processing & Management, vol. 24, no. 5, pp. 513–523, 1988.

[5] Flask Documentation. [Online]. Available: https://flask.palletsprojects.com/

[6] React – A JavaScript library for building user interfaces. [Online]. Available: https://reactjs.org/

[7] Pandas Documentation. [Online]. Available: https://pandas.pydata.org/

[8] Scikit-learn: Machine Learning in Python. [Online]. Available: https://scikit-learn.org/

[9] Mokdara, Tossawat & Pusawiro, Priyakorn & Harnsomburana, Jaturon. (2018). Personalized Food Recommendation Using Deep Neural Network. 1-4. 10.1109/ICT-ISPC.2018.8523950.

[10] N. J. Belkin and W. B. Croft, "Information filtering and information retrieval: Two sides of the same coin?," Communications of the ACM, vol. 35, no. 12, pp. 29–38, 1992.