CAL STATE FULLERTON

Department of Computer Science

This project has been satisfactorily demonstrated and is of suitable form.

This project report is acceptable in partial completion of the requirements for the Master of Science degree in Computer Science.

AI Meal Recommendation System

Project Title (type)

Divya Tanwar

Student Name (type)

Dr. Kanika Sood

Advisor's Name (type)

Advisor's signature Date

Dr. Anand Panangadan

Reviewer's Name (type)

Reviewer's signature Date

Abstract

The rise of generative Artificial Intelligence (AI) has created the possibility of presenting novel recipes, i.e., recipes that do not exactly match any known recipe and this has led to the creation of AI-based recipe recommendation systems. AI-based recipe recommendation has the possibility of accommodating a variety of preferences – including a person's current health (e.g., diabetes), health goals (e.g., weight loss), taste preferences, cultural or ethical needs (e.g., vegan diet). However, unlike recipes recommended or created by a human dietitian, recipes created by generative AI do not guarantee accuracy, i.e., the generated recipe may not meet the requirements specified by the user. This work quantitatively evaluates how closely recipes generated by OpenAI's GPT4 large language models, created in response to specific prompts, match known recipes in a collection of human-curated recipes. The prompts also include requests for a health condition, diabetes. The recipes are from the largest online community of home cooks sharing the Mayo Clinic's collection of diabetes meal plan recipes. Recipes from these sources are assumed to be authoritative and thus are used as ground truth for this evaluation. Quantitative evaluation using NLP techniques (Named Entity Recognition (NER) to extract each ingredient from the recipes and cosine similarity metrics) enable reporting the quality of the AI results along a continuum. The nutrient attributes of a recipe, such as its total calories, are also evaluated for accuracy. Each ingredient is looked up on USDA's FoodData Central API to retrieve its calories and other nutrient information to provide a true estimate. Our results show that the ingredients list in the AI-generated recipe matches 67-88% with the ingredients in the equivalent recipe in the ground truth database. The corresponding cooking directions match 64-86%. Ingredients in recipes generated by AI for diabetic's match those in known recipes in our ground truth datasets at widely varying levels: between 26-83%.

Table of Contents

1. Introduction

1.1 Background

Large Language Models (LLMs), exemplified by models like ChatGPT and GPT-4[1], are making a profound impact on the AI community, suggesting a closer realization of artificial general intelligence (AGI). They're transforming multiple research domains: in Natural Language Processing (NLP), they're becoming versatile task solvers; in Information Retrieval (IR), they're reshaping search through AI chatbots and enhancing platforms like Bing; and in Computer Vision (CV), they're paving the way for multimodal dialogues. The scaling of PLM (pre-trained language models) has significantly improved on these tasks. The training of LLMs requires the processing of large-scale and distributed parallel training on thousands of GPUs. Their integration in real-world applications is evident, with tools like Microsoft 365's Copilot. However, challenges loom. The reasons behind LLMs' emergent abilities remain elusive, their training demands are massive, and ensuring they produce safe content is critical. This survey delves deep into these facets of LLMs, aiming to offer a thorough examination of their advancements and hurdles.[2]

1.1.1 LLM Architecture

LLM is based on Transformer architecture. To improve the model's performance, the power of scaling is utilized, where billions of parameters are used to train on the large, massive text of data. In 2020, the Open AI team proposed the KM scaling law (Kaplan 2020-fd) to model the power of the relationship between model performance with respect to the three major factors: model size (N), dataset size (D), and the amount of training compute(C). Given a computed budget c, the empirical representation is:

$$\begin{aligned}
L(N) &= \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad \alpha_N \sim 0.076, N_c \sim 8.8 \times 10^{13} \\
L(D) &= \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad \alpha_D \sim 0.095, D_c \sim 5.4 \times 10^{13} \\
L(C) &= \left(\frac{C_c}{C}\right)^{\alpha_C}, \quad \alpha_C \sim 0.050, C_c \sim 3.1 \times 10^{8}
\end{aligned}$$

Figure 1: KM Scaling Law

Where L(.) denotes the cross-entropy loss in networks. Another law was proposed by the Google DeepMind team, the *Chinchilla scaling law*. These scaling laws provide an understanding of the scaling effects and help predict the performance of the models. The fundamental architecture for many state-of-the-art LLMs is - it comprises two main parts: an encoder and a decoder, both of which use self-attention mechanisms.

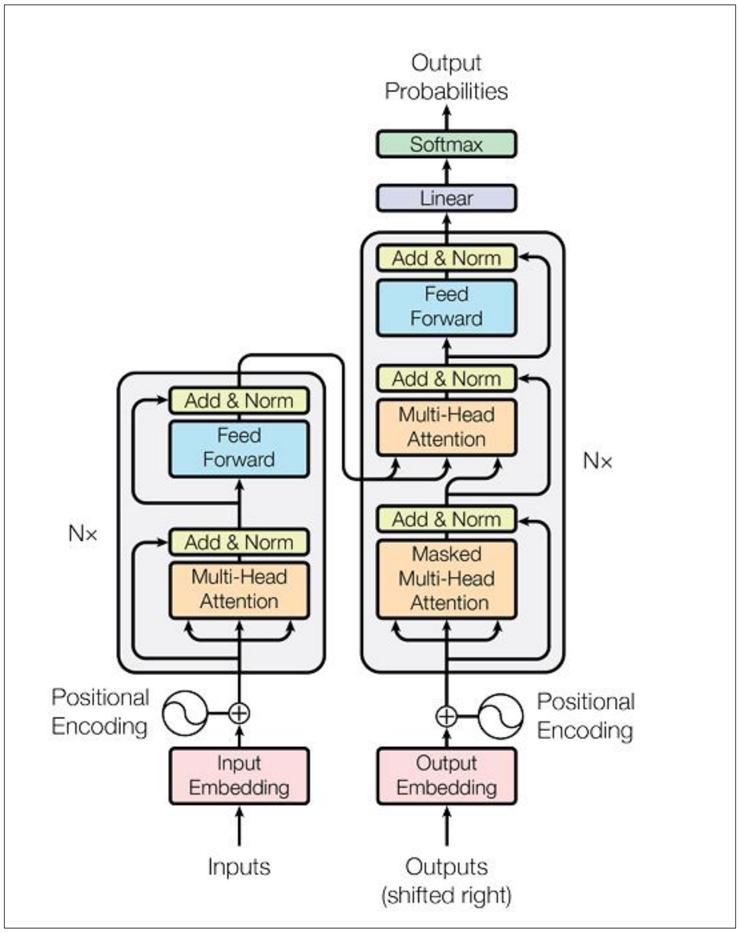


Figure 2: The Transformer - model architecture

Encoder: Takes input sequences and converts them into a continuous representation. Multiple layers where each layer has self-attention and feed-forward neural networks. The output retains the sequence length but transforms the content into a richer representation.

Self-Attention: Allows the model to focus on different words in the input when producing an output word, giving it context awareness. Uses query, key, and value weight matrices to compute attention scores.

Decoder: Uses the encoder's output to produce the final output sequence. It also has multiple layers of self-attention but adds another attention mechanism focused on the encoder's output. Final Layer: A linear layer is followed by softmax to produce the predicted probabilities over the target vocabulary.

Positional Encoding: Because transformers lack inherent sequence awareness, positional encodings are added to give a notion of word order.[3]

1.1.2 Evolution of LLMs

With the help of the feature known as in-context learning (ICL), which was added in the GPT-3 model, it is now able to provide anticipated results based on given instructions and examples without extra training. The GPT-1 and GPT-2 models lack the significant ICL capabilities displayed by the 175B GPT-3 variant. GPT-3 excels at arithmetic but struggles with Persian QA tests, showing how its skill varies with task. Large Language Models (LLMs) are tuned using natural language descriptions on multi-task datasets for instructions. In doing so, LLMs improve their ability to generalize by successfully completing new activities based solely on instructions. For models greater than 68B, for example, the instruction-tuned LaMDA-PT performed better than the untuned version. Finally, step-by-step thinking enables LLMs to work through challenging multi-step activities like math word puzzles. The chain-of-thought (CoT) approach is one of them. Over time, LLM has evolved[4] with various improvements as a general and capable learner.

The following key techniques led to the success of LLM:

1. **Scaling**: A larger model or data size improves LLM capabilities. Efficient compute allocation and quality data are critical. (30,34)
2. **Training**: Because of their scale, LLMs require distributed training using tools like DeepSpeed and Megatron-LM, as well as optimization tricks to ensure stability.
3. **Ability eliciting**: After training, LLM possesses hidden abilities. Techniques such as guided corrections and thought chain prompts will uncover them.
4. **Alignment tuning**: To minimize potential bias, LLM is fine-tuned to align with human values using methods such as InstructGPT.
5. **Tool manipulation**: LLM can leverage external tools, such as computers or search engines, to compensate for inherent limitations .

6. **Infrastructure**: Hardware and infrastructure upgrades, including methods for predicting the performance of large models using smaller models, facilitate LLM development.
7. **Human Interaction**: Incorporating human feedback and interaction, as with ChatGPT, ensures safer, more consistent model responses.

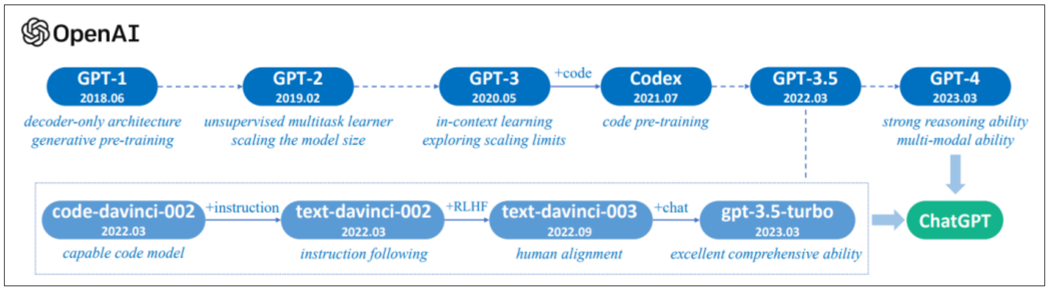


Figure 3: Technical Evolution of GPT-series Models (Reference: Zhao2023-bv)

1.1.3 Name Entity Recognition

Named entity recognition (NER) [5] is a subtask of natural language processing (NLP) that aims to identify and classify entities in a given text. Entities are words or phrases that refer to specific types of information, such as people, places, organizations, dates, numbers, etc. For example, in the sentence "Barack Obama was born in Hawaii on August 4, 1961", the entities are "Barack Obama" (person), "Hawaii" (location), and "August 4, 1961" (date). Following NER techniques are popular:

1. Rule Based Technique: It involves defining a set of rules or patterns that can match the entities in the text. The rules can be based on lexical, syntactic, semantic, or contextual features of the text. For example, a rule to identify person names can be based on capitalization, titles, or honorifics. A rule to identify locations can be based on prepositions, suffixes, or gazetteers. A rule to identify dates can be based on formats, separators, or keywords.
2. Machine Learning methods: To overcome the limitations of rule-based methods, machine learning methods have been developed to automatically learn the patterns and features that can identify and classify entities in text. Machine learning methods can be supervised,

semi-supervised, or unsupervised, depending on the availability and quality of annotated data. Supervised methods require large amounts of labeled data to train models that can generalize well to new data. Semi-supervised and unsupervised methods can leverage unlabeled data or external resources to improve the performance and robustness of the models.

3. Deep Learning method: Deep learning methods are a subset of machine learning methods that use neural networks to learn complex and non-linear representations of text. Deep learning methods can capture the semantic and syntactic information of words, phrases, and sentences, and handle the variability and ambiguity of natural languages. Deep learning methods can also benefit from transfer learning and multi-task learning, where the knowledge learned from one domain or task can be transferred or shared with another domain or task.

1.1.4 OpenAI Assistant API

The Assistants API is a service that lets you build custom AI assistants within your own applications. You can use the Assistants API to create assistants that can handle various types of user queries, such as executing code, searching files, or calling functions.

- Accessing multiple tools in parallel: The models can use different tools to perform different tasks simultaneously, such as searching the web, executing code, retrieving files, or calling functions. The models can also switch between tools as needed, depending on the user's request or the context of the conversation.
- Persistent Threads: The models can maintain long-term memory and context across multiple interactions with the same user or topic. This allows the models to resume previous conversations, recall previous facts or actions, and provide consistent and personalized responses.
- Files in various formats: The models can handle files in various formats, such as text, images, audio, video, or PDF. The models can read, write, edit, or search files, as well as convert them to different formats or extract information from them.

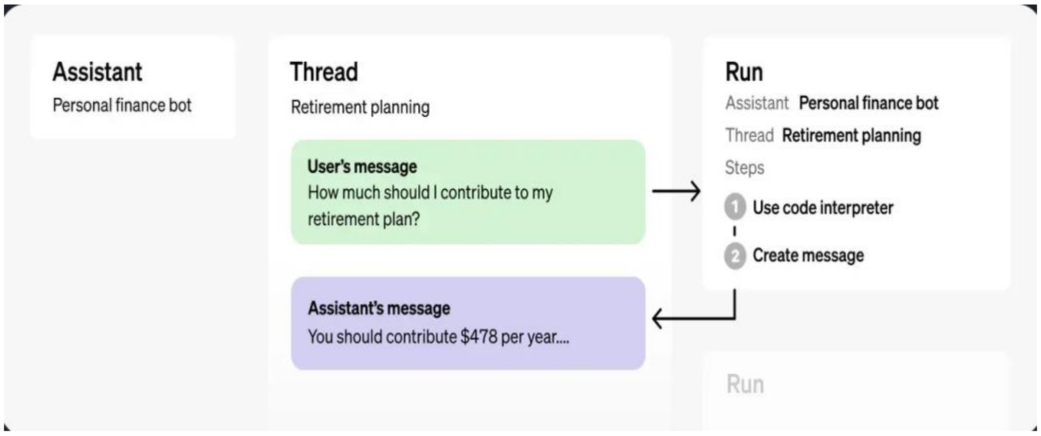

Assistant
Personal finance bot
Thread
Retirement planning
User's message
How much should I contribute to my retirement plan?
Assistant's message
You should contribute $478 per year....
Run
Assistant Personal finance bot
Thread Retirement planning
Steps
1 Use code interpreter
2 Create message
Run

Figure 4: How Assistant API works

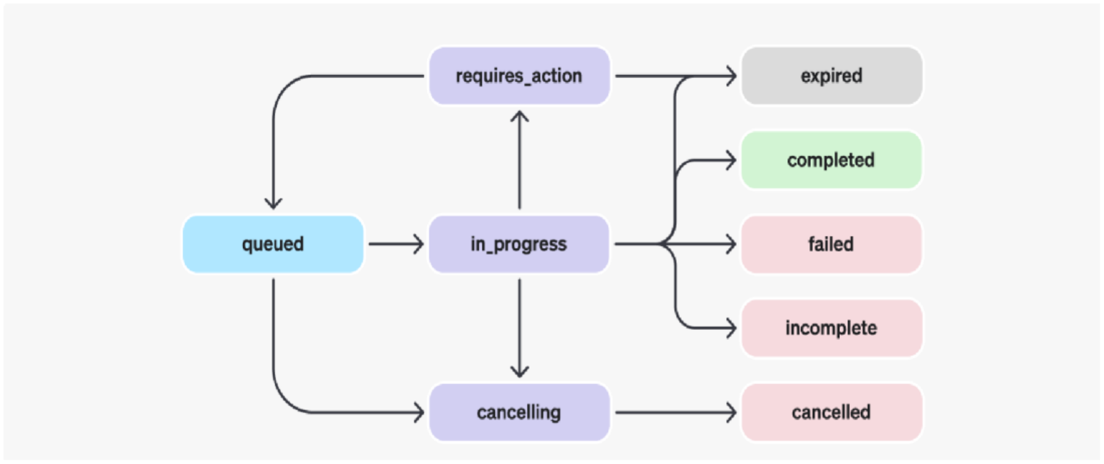


Figure 5: Run Life Cycle

1.1.5 TikToken

An open-source tool created by OpenAI is used for the process of tokenization, which involves breaking down a text string into individual tokens. These tokens can be letters, words, or groups of words, depending on the language of the text. LLMs used Byte-Pair Encoding (BPE) was initially developed as an algorithm to compress texts, and then used by OpenAI for tokenization when pretraining the GPT model. It's used by a lot of Transformer models, including GPT, GPT-2, RoBERTa, BART, and DeBERTa.

The algorithm compresses data by identifying and replacing the most frequently occurring pairs of adjacent bytes with unused bytes from the original data, creating a table of these substitutions that precedes the compressed data. This process repeats until no more pairs can be compressed or there are no unused bytes left. To handle streams and large files, the data is buffered in manageable blocks, each compressed separately with its own pair table, allowing for local adaptation and preventing memory overload. The algorithm is multi-pass, requiring all data to be in memory, and uses different pair tables for each buffer to enhance compression efficiency.[6]

1.1.6 Prompt Engineering

Prompt engineering helps the LLM model to comprehend and send respond to wide variety of queries. Developing effective prompts is essential for transforming raw queries into significant AI-generated responses. By carefully refining these prompts, engineers can greatly enhance the quality and pertinence of the outputs, addressing both specific and broad needs. This approach minimizes the necessity for manual oversight and post-generation modifications, thereby conserving time and resources while achieving intended results more efficiently.

Generative AI models rely on transformer architectures to understand language nuances and process extensive data via neural networks. Prompt engineering is crucial in shaping these models' outputs, ensuring they respond meaningfully and coherently. Techniques such as tokenization, model parameter tuning, and top-k sampling are used to refine AI responses. These models are underpinned by foundation models, which are large language models (LLMs) equipped with comprehensive data for AI operations. They use natural language processing (NLP) to interpret inputs and generate complex text or image outputs. Models like DALL-E and Midjourney integrate an LLM with stable diffusion techniques to create images from text descriptions effectively. This process requires a blend of technical skills and an in-depth understanding of language and context to optimize the outputs, minimizing the need for subsequent corrections. Strategies for better response from LLMs model [7]:

1. Write clear instructions
2. Provide reference text
3. Split complex Tasks into simpler subtask
4. Give the model to think
5. Use external tools to compensate for the weakness of model

6. Test changes systematically

1.1.7 USDA Food API

Several USDA food composition databases, such as the Food and Nutrient Database for Dietary Studies (FNDDS), the Standard Reference (SR) Legacy, and the USDA Branded Food Products Database, have been consolidated into FoodData Central. This new, unified system enhances the availability and quality of food and nutrient data, facilitating robust research and policy development. FoodData Central offers comprehensive nutrient content information and connects to a wide range of related data from agricultural, environmental, health, and dietary sectors. It supports advanced search functionalities, provides downloadable datasets, and features detailed documentation. Additionally, the system is equipped with a REST API, enabling developers to integrate this data into apps and websites.

The evolving complexity of the food supply poses a challenge for users of food and nutrient data. By amalgamating diverse data types within a single system, FoodData Central serves as a critical tool for researchers, policymakers, and other stakeholders focusing on key nutritional and health issues.

- It houses five distinct data types, each tailored for specific informational needs: Foundation Foods, Experimental Foods, Standard Reference, Food and Nutrient Database for Dietary Studies, and the USDA Global Branded Food Products Database.
- It provides a comprehensive overview of the nutrients and other components found in a broad array of foods and food products.
- It features data sourced from multiple origins, which are regularly updated with the latest available information.
- It employs a range of analytical and computational methods to derive values, ensuring the use of cutting-edge techniques and transparent data presentation.

1.2 Significance of the Problem

At present, numerous AI Meal Recommendation systems are available, offering recipe suggestions based on user preferences or provided prompts. These recommendation systems offer users a variety of options, such as selecting ingredients from existing databases, specifying expertise level, cooking time, meal type, dietary preferences, cuisine type, and more. However, despite these advancements, several challenges persist:

1.2.1 Authenticity Verification: When users receive recipe suggestions from AI systems, they may wonder about the credibility or reliability of those recipes. Currently, there isn't a standardized way for users to assess the authenticity of the suggested recipes. Metrics such as user ratings, reviews, or feedback mechanisms could be integrated into these systems to help users gauge the quality and accuracy of the recipes provided.

1.2.2 Health Condition Assessment: Many users have specific dietary requirements or health conditions that need to be taken into consideration when recommending recipes. For example, individuals with diabetes may need recipes that are low in sugar, while those with heart conditions might require low-sodium options. AI systems could enhance their capabilities by incorporating more comprehensive health assessment features, such as analyzing nutritional content and ensuring compatibility with dietary restrictions or medical conditions.

1.2.3 Accurate Calorie Estimation: For users who are conscious of their calorie intake, it's essential that AI systems provide accurate estimations of the total calories in recommended recipes. Inaccurate calorie calculations can lead to distrust in the suggested meals and may hinder users' ability to make informed decisions about their diet. Implementing robust algorithms and data sources for calculating nutritional information can help improve the reliability of calorie estimations.

1.2.4 Customization Options: While AI meal recommendation systems offer a range of features and options, users may still feel limited in their ability to customize recipes according to their preferences. Providing more extensive customization options, such as adjusting ingredient quantities, substituting ingredients, or selecting specific cooking methods, can empower users to tailor recipes to their individual tastes and dietary needs.

In this project, our main objective is to develop an intelligent agent capable of generating recipes in response to user prompts. The agent will engage in conversational interactions with users via a user interface, allowing them to express their preferences and receive tailored recipe recommendations.

1.3 Review of Significant Research

LLM output validation plays a critical role in shipping production ready application. This predictability helps protect the application from unexpected behaviors and ensures a consistent user experience. Below are some scenarios where validating outputs might be necessary.

Structure Compliance: Certain LLM applications require outputs that adhere to a specific structure with designated types of information. For instance, a text extraction task might need outputs in JSON format that conform to a defined structure. This requirement can extend to synthetic data generation, where the data created must satisfy specific conditions.

Safe Responses: Given their probabilistic nature, LLM applications need added safeguards to ensure their output is safe, ethical, and protect privacy. For instance, outputs should be checked to ensure they do not contain profanity or personally identifiable information (PII).

Semantic Similarity: For some uses, outputs must maintain sufficient similarity to a target. In text summarization, for example, it's important that the summary stays true to the original document. This can be achieved by using text embeddings to verify that the summary and the original document are semantically similar.

Valid Choices: In cases like creating a chess playing LLM, the output must only include valid moves based on the board's current state. Similarly, for code generation, the output must be syntactically correct according to the programming language's rules.

Quality Assurance: Generally, there might be a need to implement a validation step to ensure that an LLM output upholds a specific quality standard relevant to the application's use case, thus delivering real value to users.

1.4 Objective

Throughout the project, we aim to explore a variety of techniques to assess the quality and accuracy of the generated recipes. Here's a breakdown of the metrics we plan to utilize:

1.4.1 Metrics

1.4.1.1 RMSE for Total Calories Based on Ingredients: We'll calculate the Root Mean Square Error (RMSE) to evaluate the accuracy of the total calorie estimation in the generated recipes. This metric provides a quantitative measure of the deviation between the predicted and actual calorie values, helping us assess the reliability of the nutritional information provided.

1.4.1.2 Cosine Similarity for Ingredients and Directions: Cosine similarity will be employed to measure the similarity between the ingredients and cooking directions of the generated recipes compared to a reference dataset. This metric quantifies the degree of resemblance between two sets of textual data, allowing us to assess how closely the generated recipes align with existing recipes in terms of ingredients and cooking instructions.

1.4.1.3 Match percentage of ingredients Generated by LLM: Calculate the match percentage to determine the proportion of ingredients generated by our Language Model (LLM) that match those present in our dataset of known recipes. This metric helps us gauge the relevance and appropriateness of the ingredients suggested by the agent, providing insights into the effectiveness of our recipe generation approach.

1.4.1.3 Energy calculation for each ingredient in each recipe: Energy calculation involves estimating the total energy content (e.g., calories or kilojoules) contributed by each ingredient in each recipe. By summing up the energy values of all ingredients, we can obtain an overall measure of the recipe's energy content. This metric allows us to provide users with comprehensive nutritional information and helps them make informed decisions about their dietary choices.

1.5 Limitation

1.5.1 Hallucinations: This is one of the main challenges of this project to ensure the accuracy and reliability of the content generated by the AI agent. LLMs can sometimes produce hallucinations**,** which are inaccurate or irrelevant pieces of text that do not match the input or the task. Which can lead to producing the same recipe for a similar type of prompt semantically. Therefore, we need to devise methods to detect and prevent hallucinations and validate the quality of the recipes generated by the agent.

1.5.2 Incorrect Mathematical calculations: Another reason for the lack of numeric representations is that LLMs are designed to process and generate text, working with tokens rather than numerical values. While most text-based tasks can have several reasonable solutions, mathematical problems usually have only one correct answer.

2. Research Approach

2.1 Summary

This section will focus on the overall research approach, design methods, and implementation choices made during the project. The first section will focus on the dataset selection and cleaning process along with the approaches used to achieve the goal of checking the quality and validity of the recipes generated by the LLM by using the System message i.e. prompt engineering.

2.2 Dataset Selection and Cleaning

Dataset: Mayo Clinic Healthy Recipes

The Mayo-Clinic website provides a dataset of healthy recipes, complete with dietitian tips, ingredients, directions, serving sizes, and nutritional analysis per serving. The recipes are categorized with tags such as dash diet, low sodium, weight management, low fat, kidney diet, diabetes meal plan, plant-based, and heart-healthy, to cater to various dietary needs. It contains a total of 526 recipes for diabetic meal plans. This graph shows the top 100 ingredients count for the dataset and if we compare this graph with the previous dataset. In this graph, sugar is not even in the top 100 ingredients.

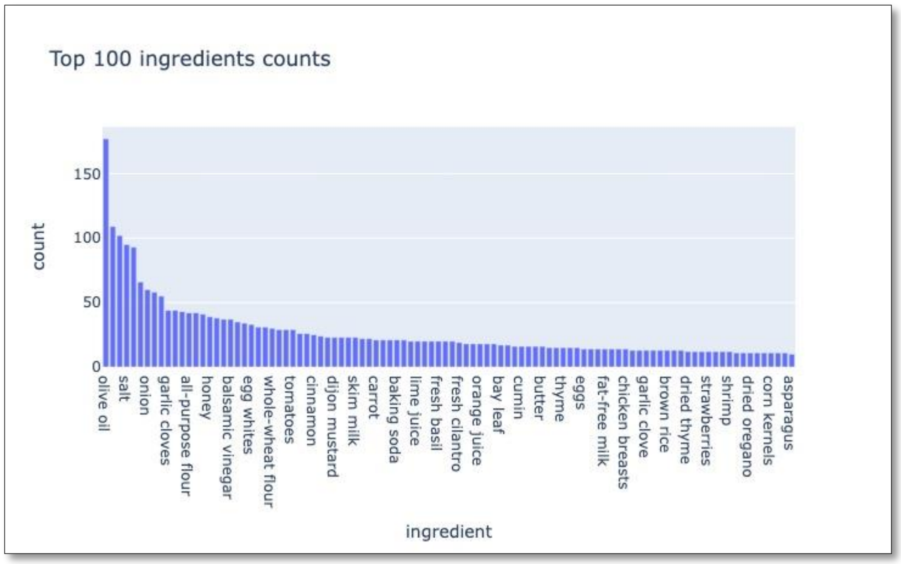


Figure 6: Graph for ingredient count in recipes

2.3 Implementation Approaches

This section will discuss various approaches for generating and validating the recipe generated by LLM [8,9,10]. We will briefly explain each technique and working methodology, code snippets and performance metrics.

Experiment 1:

In experiment 1, the recipe was created using OpenAI Assistant API. The model used is gpt-4-1106. The dataset used is from Kaggle titled 'Recipes Dataset' and has fields like recipe_name, preparation time for the recipe, cook time, total time required for the recipe, number of servings, ingredients, directions, rating, URL for the recipe, cuisine path, nutrition information, timing, and image source. The recipe matching is implemented using cosine similarity. Primarily, some random recipes are selected from the dataset, and the LLM generates recipes with the same name in XML format using Assistant Instructions. Parsing the XML format recipes using Beautiful Soup. These recipes are converted from XML to JSON format. From this JSON format, some nutrition information like protein, fat, and carbohydrate are extracted. The cosine similarity method then calculates the ingredient and direction similarity and compares the calories of the recipes. The calorie estimation for LLM generated and original recipes is performed. The embedding model is used to calculate the cosine similarity for the recipe's ingredients and directions. The final metrics from these embeddings and calorie estimation are compiled in JSON file format.

System instruction used to create an assistant:

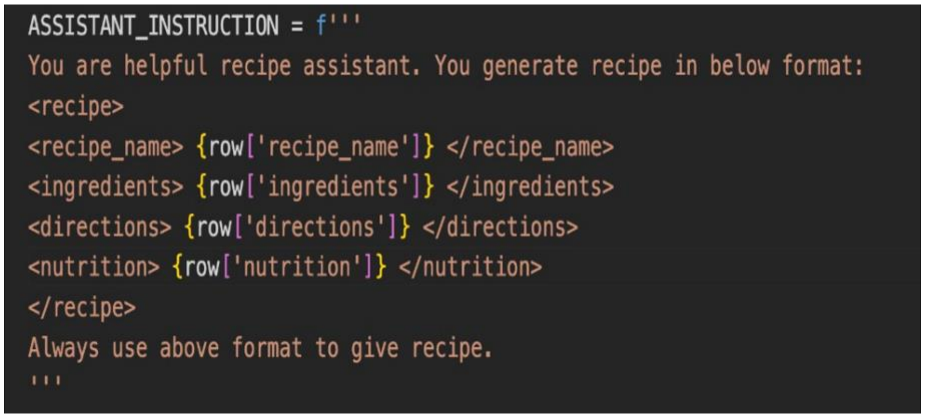

```
ASSISTANT_INSTRUCTION = f'''
You are helpful recipe assistant. You generate recipe in below format:
<recipe>
<recipe_name> {row['recipe_name']} </recipe_name>
<ingredients> {row['ingredients']} </ingredients>
<directions> {row['directions']} </directions>
<nutrition> {row['nutrition']} </nutrition>
</recipe>
Always use above format to give recipe.
'''
```

Figure 7: Assistant Instruction

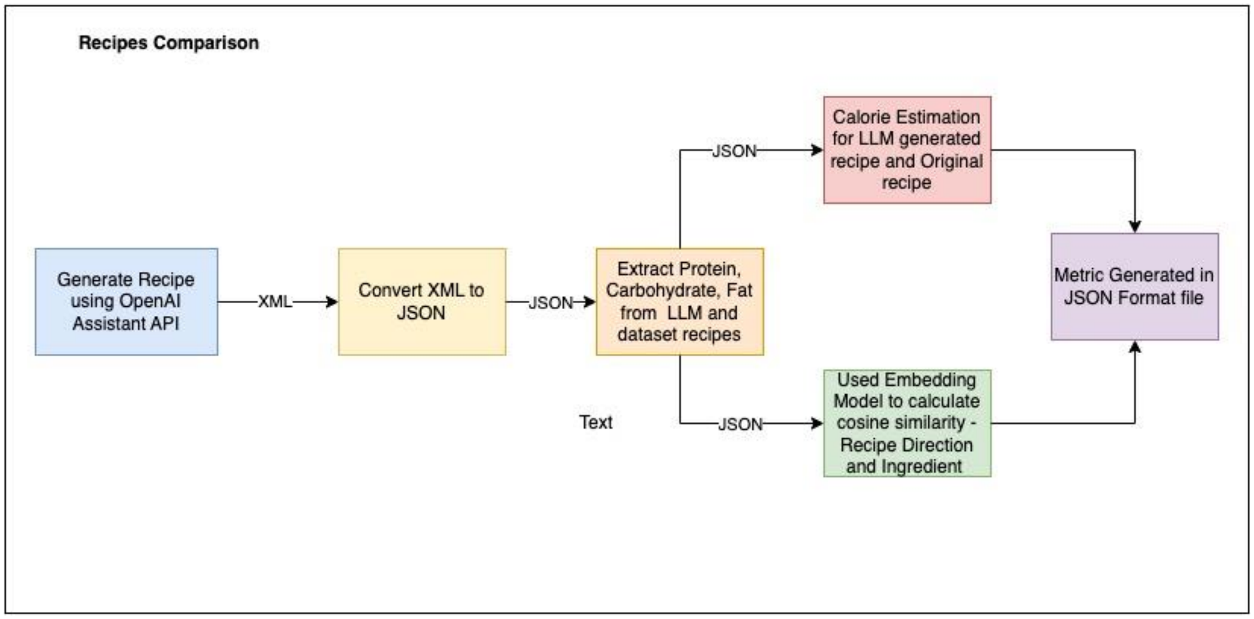

Recipes Comparison
Generate Recipe using OpenAI Assistant API
XML
Convert XML to JSON
JSON
Extract Protein, Carbohydrate, Fat from LLM and dataset recipes
JSON
Calorie Estimation for LLM generated recipe and Original recipe
Text
JSON
Used Embedding Model to calculate cosine similarity - Recipe Direction and Ingredient
Metric Generated in JSON Format file

Figure 8: Pipeline for Recipe Comparison

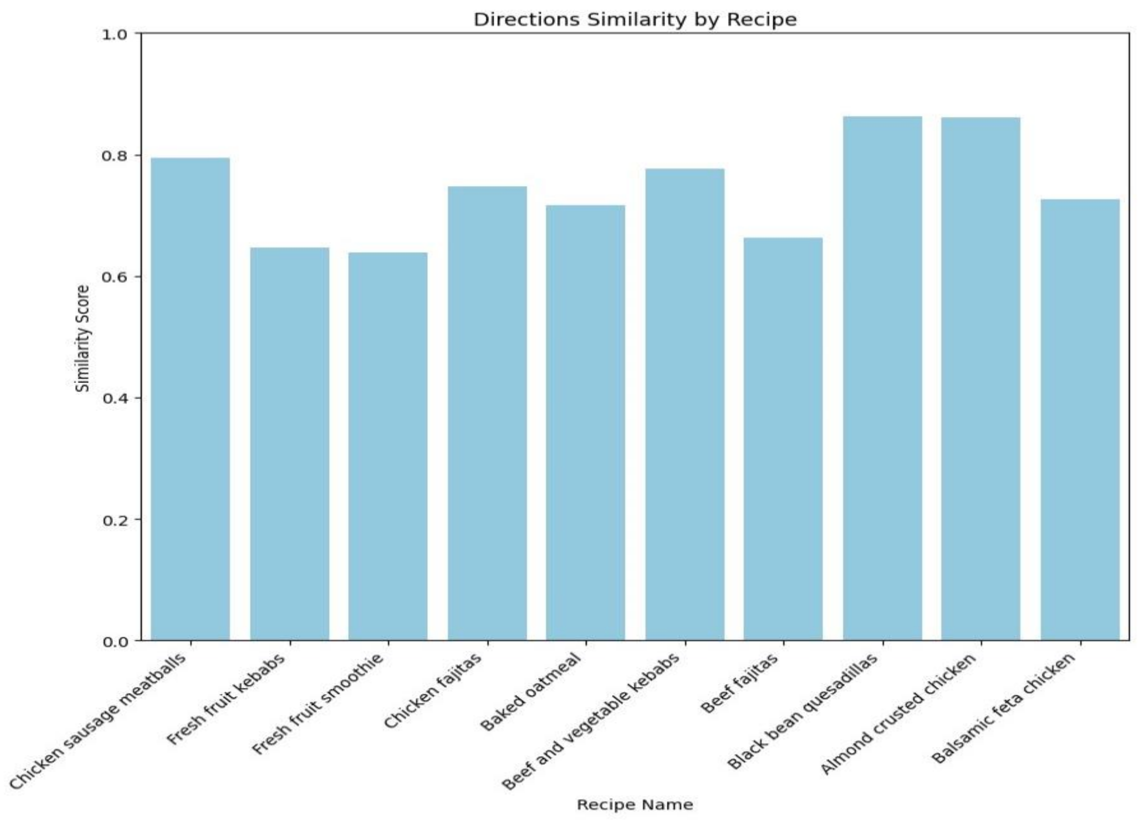


Figure 9: Graph for Direction Similarity of known recipe vs LLM generated recipe

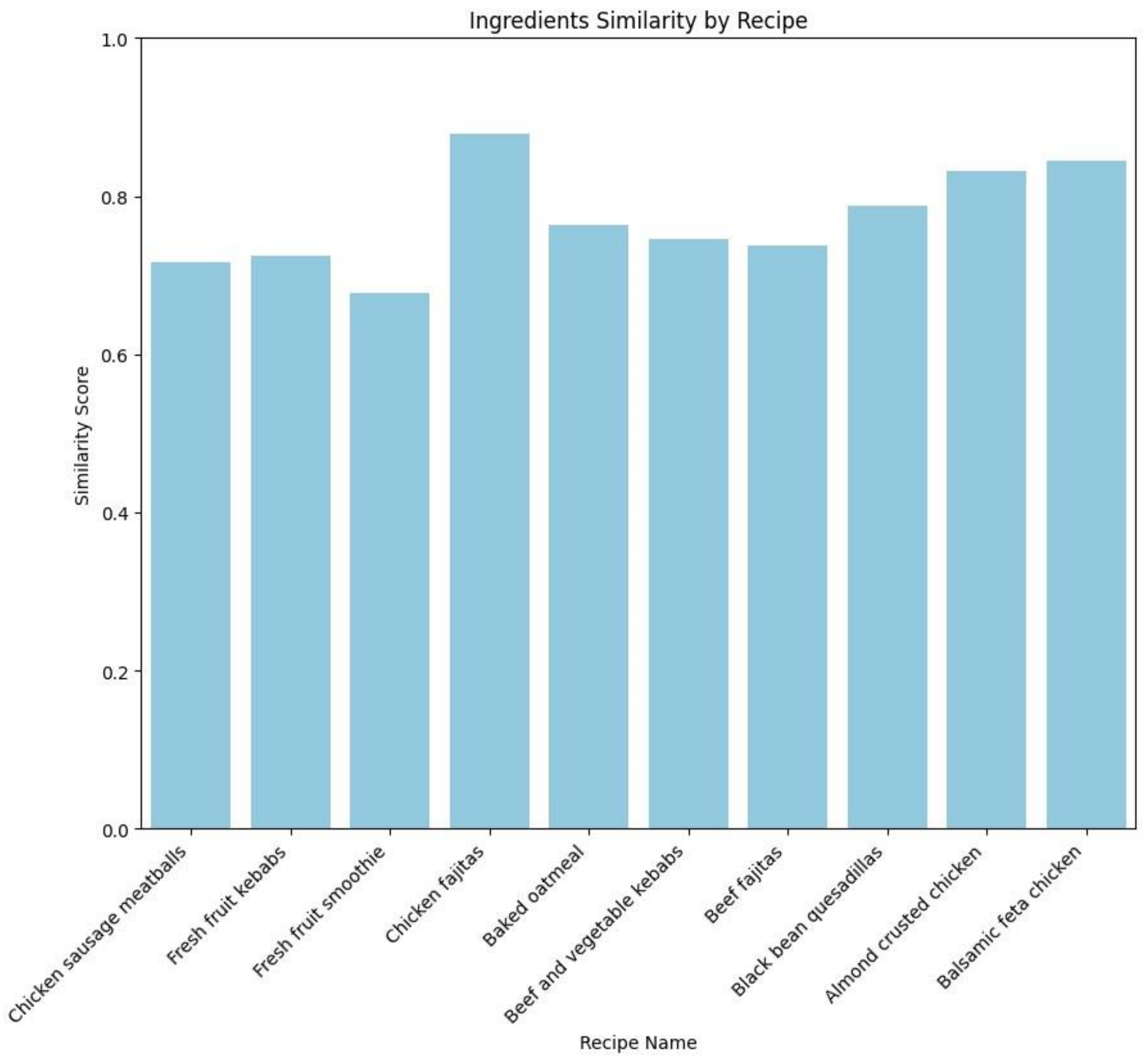


Figure 10: Graph for Ingredients Similarity of known recipe vs LLM generated recipe

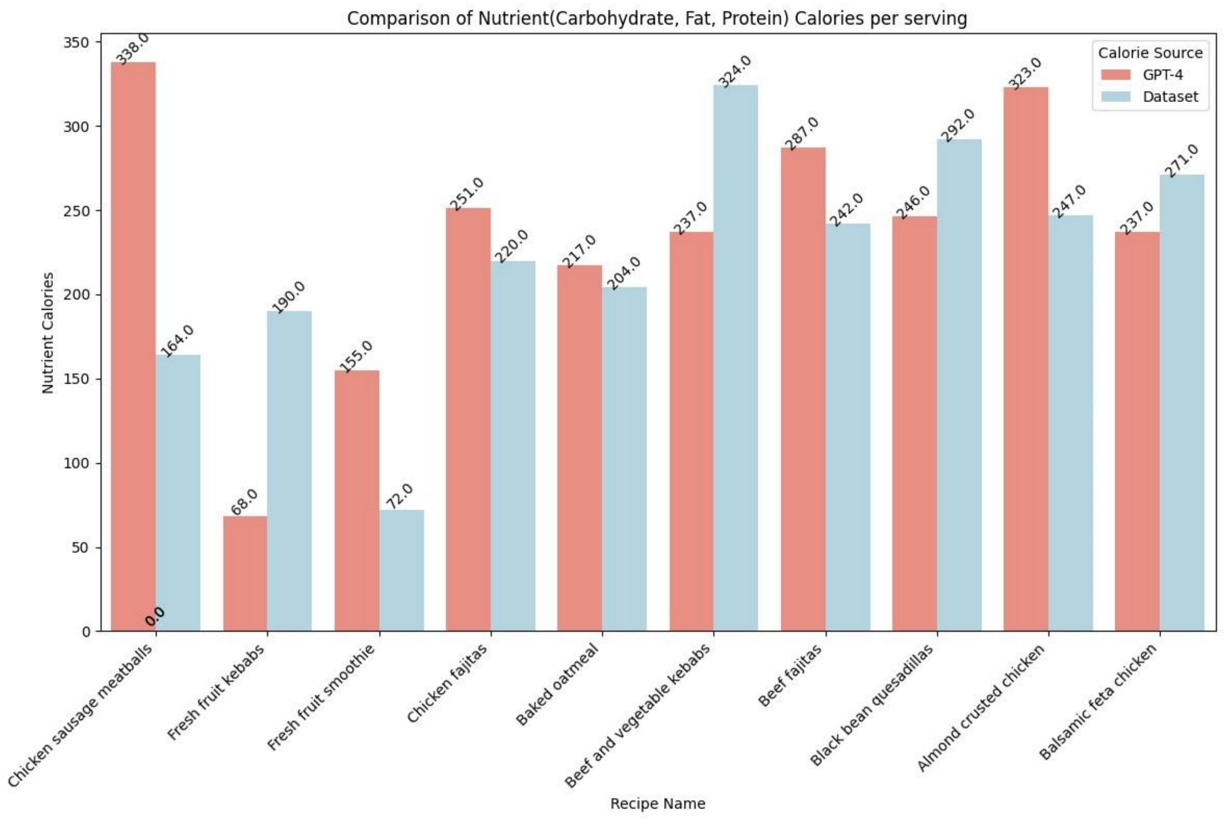


Figure 11: Graph for comparison of nutrients (carbohydrates, fat, protein) calories per serving

Experiment 2:

Experiment 2, titled "Ingredient Match Using Percentage for Diabetic-Friendly Recipes," commences with the generation of random diabetic-friendly recipes for breakfast, lunch, and dinner using the OpenAI Assistant API, formatted in XML. These XML files are subsequently parsed into JSON format using Beautiful Soup Library. The recipes are then compared to those in the 'Mayo Clinic Dataset', which is renowned for containing healthy recipe options tailored for specific dietary needs such as diabetes, cholesterol management, and low sodium diets. This dataset encompasses fields including the recipe name, dietitian's tips, number of servings, ingredients, dietary tags (e.g., diabetic-friendly), cooking directions, nutritional analysis per serving, and serving sizes. It uses the same instruction used in Experiment 1.

The comparative analysis involves searching for similar recipes within the dataset using string-matching techniques and extracting ingredients through Named Entity Recognition (NER). For instance, a recipe titled 'chicken salad' is compared with other chicken or salad recipes in the dataset. Post-ingredient extraction via NER, the similarity of the recipes is quantified by calculating the percentage of matching ingredients. This is achieved using the formula: (the number of matching ingredients from both recipes divided by the total number of ingredients in the dataset recipes) multiplied by 100. This final step serves to validate the authenticity of the recipes generated by LLM.

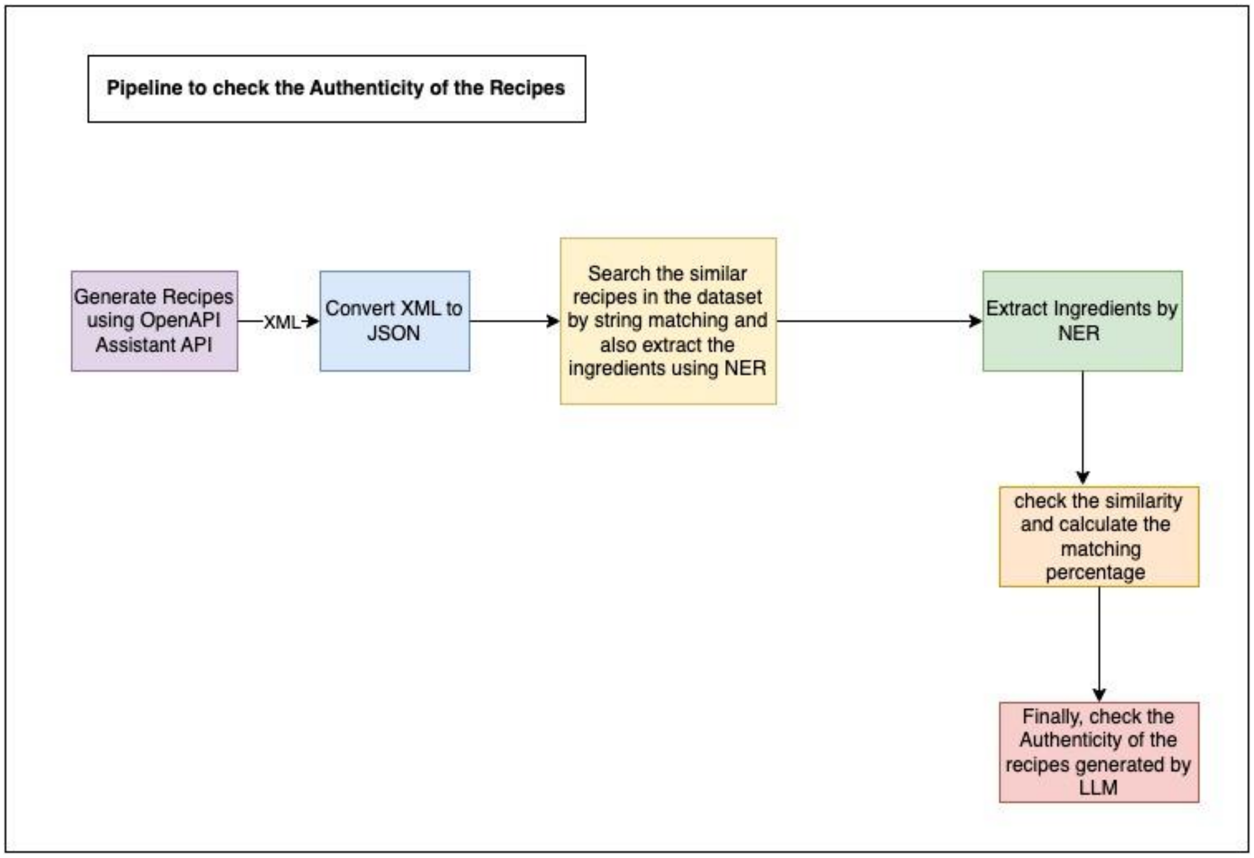

Pipeline to check the Authenticity of the Recipes
Generate Recipes using OpenAPI Assistant API
XML
Convert XML to JSON
Search the similar recipes in the dataset by string matching and also extract the ingredients using NER
Extract Ingredients by NER
check the similarity and calculate the matching percentage
Finally, check the Authenticity of the recipes generated by LLM

Figure 12: Pipeline to check authenticity of the recipes generated by LLM.

Experiment Results:

Dataset recipes → GPT-4 Recipes ↓	Pecan crusted tofu	Sesame-crusted tofu	Tofu triangles with peanut sauce	Tofu with bok choy	Vegetarian chili with tofu
Scrambled tofu with spinach and tomato	22.22	33.33	33.33	33.33	44.44

Table 1: Matching percentage table for Tofu

Dataset recipes → GPT-4 Recipes ↓	Fresh fruit smoothie	Green smoothie	Orange dream smoothie	Orange juice smoothie
Almond and blueberry smoothie	0.0	11.11	55.55	33.33

Table 2: Matching percentage table for Smoothie

Dataset recipes → GPT-4 Recipes ↓	Banana oatmeal pancakes	Buckwheat pancakes	Whole-grain pancakes	Whole-grain blueberry pancakes	Whole-grain pumpkin pancakes
Almond flour pancakes	45.45	45.45	36.36	36.36	81.81

Table 3: Matching percentage table for Pancakes

Dataset recipes → GPT-4 Recipes ↓	Smokey frittata	Southwestern frittata	Spinach and mushroom Frittata	Spinach Frittata
Spinach and mushroom egg Frittata	54.54	83.33	75.0	75.0
Spinach and mushroom Frittata	72.72	50.0	75	75
Veggie-packed frittata	72.72	66.66	66.66	66.66

Table 4: Matching percentage table for Frittata

Dataset recipes → GPT-4 Recipes ↓	Chickpea polenta with olives	Gazpacho with Chickpea
Chickpea and salad wraps	53.84	53.84
Mediterranean Chickpea salad	76.92	61.53
Quinoa Chickpea salad jars	53.84	76.92
Tuna and Chickpea salad	54.54	63.63

Table 5: Matching percentage table for Chickpea

Dataset recipes → GPT-4 Recipes ↓	Buffalo chicken salad wrap	Chicken salad with pineapple and balsamic vinaigrette	Chicken salad with Thai flavors	Grilled chicken salad with buttermilk dressing	Grilled chicken salad with olives and oranges
Grilled chicken salad with avocado dressing	20.0	33.33	66.66	33.33	46.66
Grilled lemon herb chicken salad	33.33	40.0	60.0	73.33	80.0

Table 6: Matching percentage table for Chicken Salad

Dataset recipes → GPT-4 Recipes ↓	Roasted red bell pepper pineapple salsa
Stuffed bell peppers	29.41
Stuffed bell peppers with quinoa and black beans	26.66

Table 7: Matching percentage table for Bell Pepper

Experiment 3:

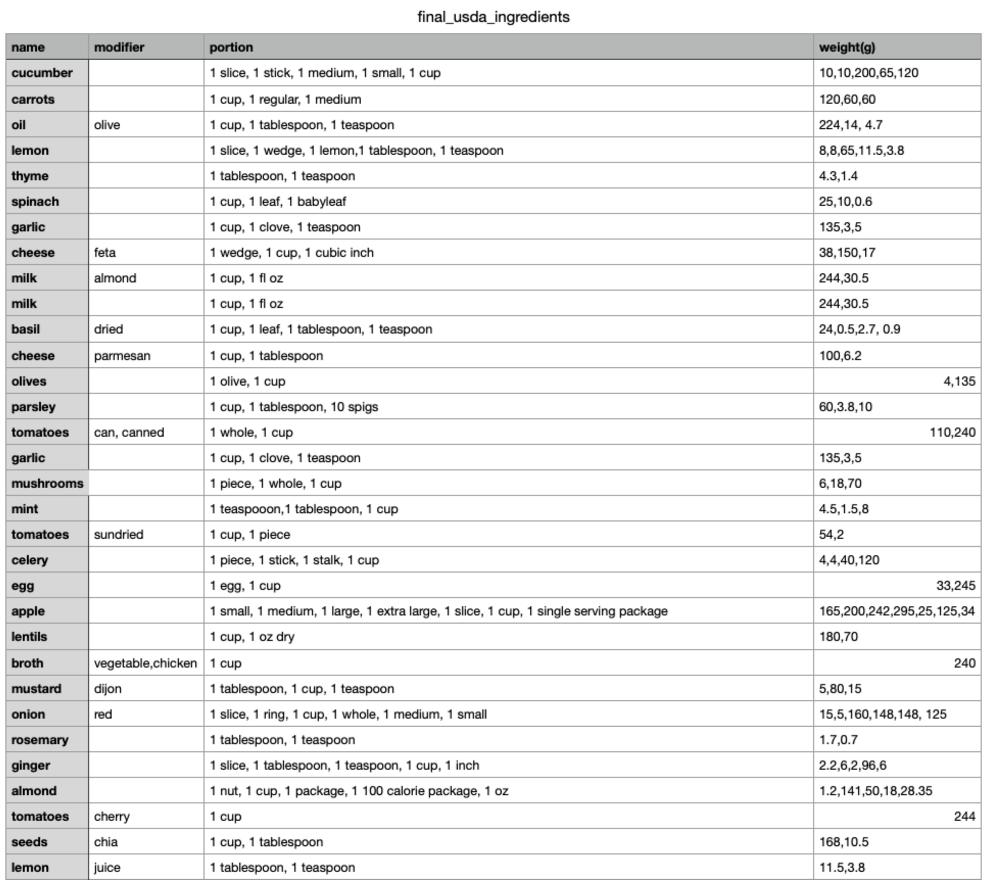
Experiment 3, titled "Checking the authenticity of calorie comparison for Diabetic-Friendly Recipes," commences with the generation of random diabetic-friendly recipes for breakfast, lunch, and dinner using the OpenAI Assistant API, formatted in XML. These XML files are subsequently parsed to JSON format using Beautiful Soup Library. The recipes are then compared to those in the 'Mayo Clinic Dataset', which is renowned for containing healthy recipe options tailored for specific dietary needs such as diabetes, cholesterol management, and low sodium diets. This dataset encompasses fields including the recipe name, dietitian's tips, number of servings, ingredients, dietary tags (e.g., diabetic-friendly), cooking directions, nutritional analysis per serving, and serving sizes. The assistant instruction used for the experiment:

```
You are a helpful diabetic-friendly recipe assistant who give different recipe each time when user ask. Recipe can
be for vegetarian, vegan, and non-vegeterian diets.You generate diabetic friendly recipe in the following format:
<recipe>
    <recipe_name> Classic Apple Pie </recipe_name>
    <ingredients>
- quinoa: 1 cup
- water: 2 cups
- black beans (rinsed and drained): 1 can (15 ounces)
- red bell pepper (diced): 1
- fresh cilantro (chopped): 1/4 cup
- lime juice: 1/4 cup
- olive oil: 2 tablespoons
- ground cumin: 1 teaspoon
- salt: 1/2 teaspoon
- black pepper: 1/4 teaspoon
- avocado (diced, optional): 1
- cherry tomatoes (halved, optional): 1/2 cup
 </ingredients>
    <directions>
1. Cook Quinoa: Rinse the quinoa under cold water. In a medium saucepan, bring 2 cups of water to a boil. Add q
uinoa, reduce heat to low, cover, and simmer for about 15 minutes, or until quinoa is tender and water is absorbed
. Remove from heat and let it cool.
```

```
2. Combine Ingredients: In a large bowl, combine the cooled quinoa, black beans, red bell pepper, and cilantro.
3. Make Dressing: In a small bowl, whisk together lime juice, olive oil, cumin, salt, and black pepper.
4. Mix Salad: Pour the dressing over the quinoa mixture and stir until well combined. If using, gently fold in the a
vocado and cherry tomatoes.
5. Chill and Serve: Refrigerate the salad for at least 30 minutes before serving. This allows the flavors to meld tog
ether.
 </directions>
    <nutrition> Total Carbohydrate: 50g, Dietary Fiber: 15g, Sodium: 300mg, Saturated Fat: 2g, Total Fat: 17g, Pr
otein: 12g, Added Sugars: 0g, Total Sugars: 5g </nutrition>
    <total_calories_estimation>
- Total Calories for entire recipe : Approximately 1561 calories for the entire recipe.
- Calories for each ingredient:
    Quinoa: 712 calories for 1 cup cooked (from 185g of dry quinoa)
    Black Beans: 227 calories per can
    Red Bell Pepper: 37 calories
    Cilantro: Negligible
    Lime Juice: 8 calories per 1/4 cup
    Olive Oil: 239 calories per 2 tablespoons
    Cumin, Salt, Black Pepper: Negligible
    Avocado: 322 calories
    Cherry Tomatoes: 16 calories per 1/2 cup
- Serving People: 4
</total_calories_estimation>
</recipe>
Always use above format to give recipe. Give total calories for recipe and refer USDA Food API for calories.
```

Table 8: Experiment 3 Assistant instruction

The comparative analysis involves entity recognition is applied for ingredients on the recipes generated by the LLM to extract the ingredients. To extract the quantity from the ingredients regular expression is applied. Further energy is calculated for each ingredient per 100 gram using USDA Food DataCentral API. After converting the quantity to weight conversion, results obtained from the quantity to weight conversion and results from per 100 grams are combined to perform the generated by the LLM in recipes is done by a regular expression. Finally, the total calories generated by LLM is compared with the results obtained from the USDA Food DataCentral API. A dataset has been created to accurately convert the volume of ingredients to their weight, to calculate the energy content of the ingredients used in recipes from the USDA API. This allows for precise measurement of the quantity of ingredients used.



final_usda_ingredients

name	modifier	portion	weight(g)
cucumber		1 slice, 1 stick, 1 medium, 1 small, 1 cup	10,10,200,65,120
carrots		1 cup, 1 regular, 1 medium	120,60,60
oil	olive	1 cup, 1 tablespoon, 1 teaspoon	224,14, 4.7
lemon		1 slice, 1 wedge, 1 lemon,1 tablespoon, 1 teaspoon	8,8,65,11.5,3.8
thyme		1 tablespoon, 1 teaspoon	4.3,1.4
spinach		1 cup, 1 leaf, 1 babyleaf	25,10,0.6
garlic		1 cup, 1 clove, 1 teaspoon	135,3,5
cheese	feta	1 wedge, 1 cup, 1 cubic inch	38,150,17
milk	almond	1 cup, 1 fl oz	244,30.5
milk		1 cup, 1 fl oz	244,30.5
basil	dried	1 cup, 1 leaf, 1 tablespoon, 1 teaspoon	24,0.5,2.7, 0.9
cheese	parmesan	1 cup, 1 tablespoon	100,6.2
olives		1 olive, 1 cup	4,135
parsley		1 cup, 1 tablespoon, 10 spigs	60,3.8,10
tomatoes	can, canned	1 whole, 1 cup	110,240
garlic		1 cup, 1 clove, 1 teaspoon	135,3,5
mushrooms		1 piece, 1 whole, 1 cup	6,18,70
mint		1 teaspooon,1 tablespoon, 1 cup	4.5,1.5,8
tomatoes	sundried	1 cup, 1 piece	54,2
celery		1 piece, 1 stick, 1 stalk, 1 cup	4,4,40,120
egg		1 egg, 1 cup	33,245
apple		1 small, 1 medium, 1 large, 1 extra large, 1 slice, 1 cup, 1 single serving package	165,200,242,295,25,125,34
lentils		1 cup, 1 oz dry	180,70
broth	vegetable,chicken	1 cup	240
mustard	dijon	1 tablespoon, 1 cup, 1 teaspoon	5,80,15
onion	red	1 slice, 1 ring, 1 cup, 1 whole, 1 medium, 1 small	15,5,160,148,148, 125
rosemary		1 tablespoon, 1 teaspoon	1.7,0.7
ginger		1 slice, 1 tablespoon, 1 teaspoon, 1 cup, 1 inch	2.2,6,2,96,6
almond		1 nut, 1 cup, 1 package, 1 100 calorie package, 1 oz	1.2,141,50,18,28.35
tomatoes	cherry	1 cup	244
seeds	chia	1 cup, 1 tablespoon	168,10.5
lemon	juice	1 tablespoon, 1 teaspoon	11.5,3.8

Figure 13: Volume to weight conversion

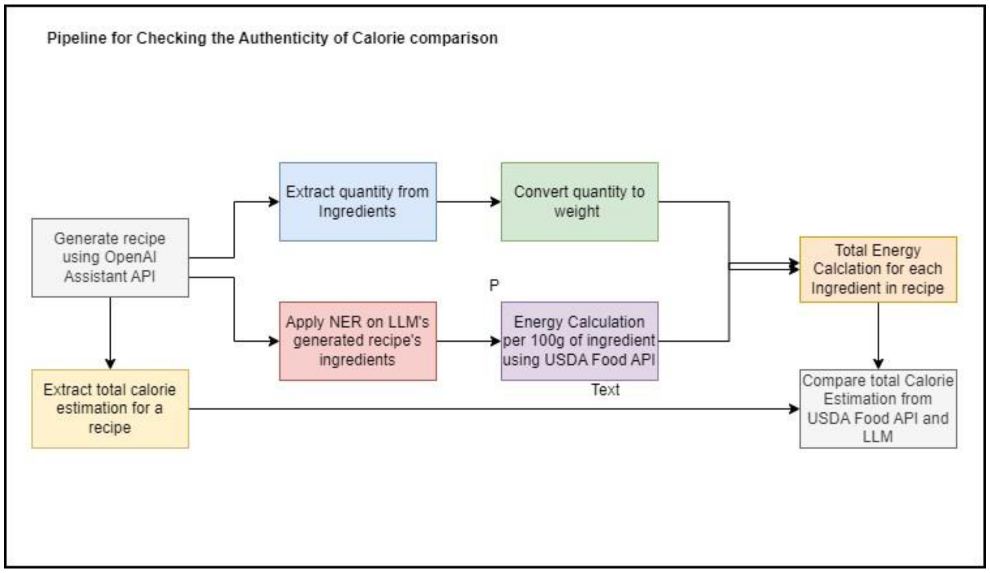

Pipeline for Checking the Authenticity of Calorie comparison
Generate recipe using OpenAI Assistant API
Extract quantity from Ingredients
Convert quantity to weight
Apply NER on LLM's generated recipe's ingredients
Energy Calculation per 100g of ingredient using USDA Food API
P
Text
Total Energy Calclation for each Ingredient in recipe
Extract total calorie estimation for a recipe
Compare total Calorie Estimation from USDA Food API and LLM

Figure 14: Pipeline for Checking the authenticity of calorie comparison

2.4 Web Application - Recipe Assistant

An end user interacts with the frontend, which has two components: UI and Client. The UI covers the pages like Register, Login and Chat playground. The role of the client is interacted with the Backend.

There are two types of routes in Backend: Users and Chats, which enables the frontend to perform Login, Register, Chat and Delete chat conversation operations. To achieve the other operations like token service. Token service issues a JSON web token which allows only authenticated users to do chat conversation. To generate recipes, the OpenAI client talks to Assistant API. The MongoDB client serves the purpose of storing user profile and chat metadata. To optimize the latency, we have added cache service which keeps the chat metadata for 60 minutes.

Moving from implementation to deployment. Azure Container Apps is a scalable container service to serve the user traffic. The frontend and backend application deployed in Azure container service.

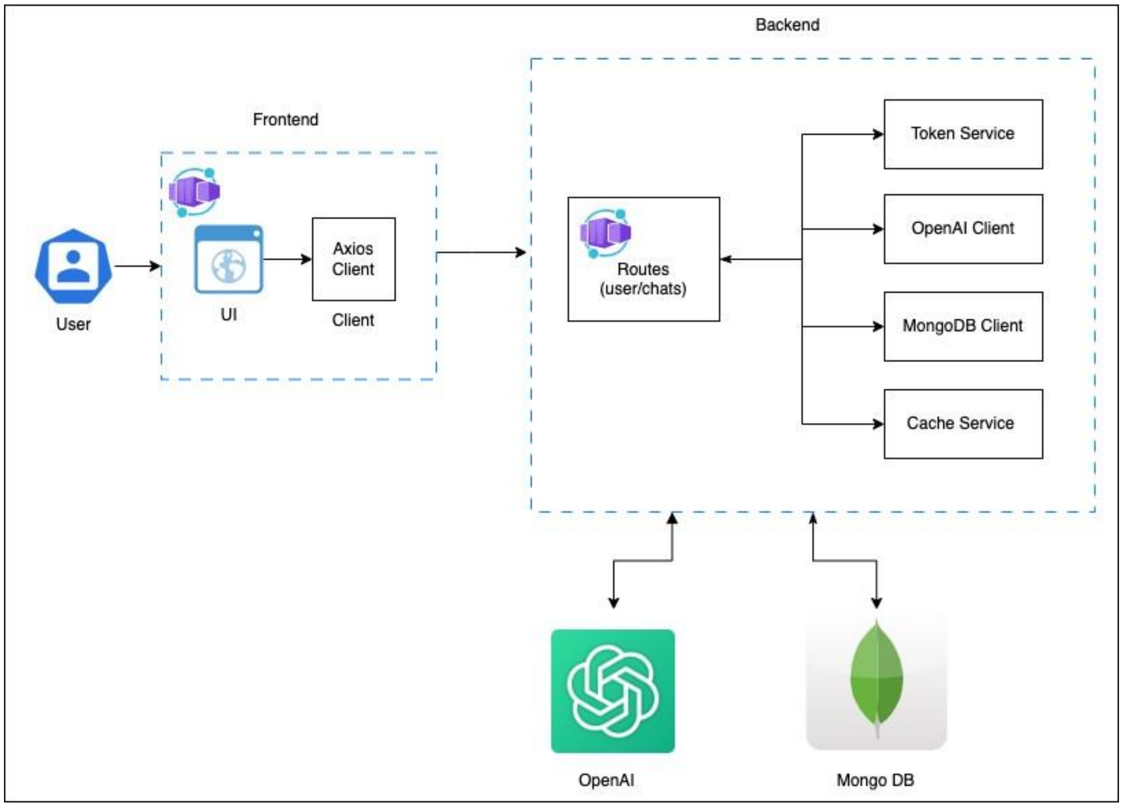


Figure 15: High Level Diagram of Recipe Assistant Application

2.5 Sequence Diagrams:

1. Register endpoint

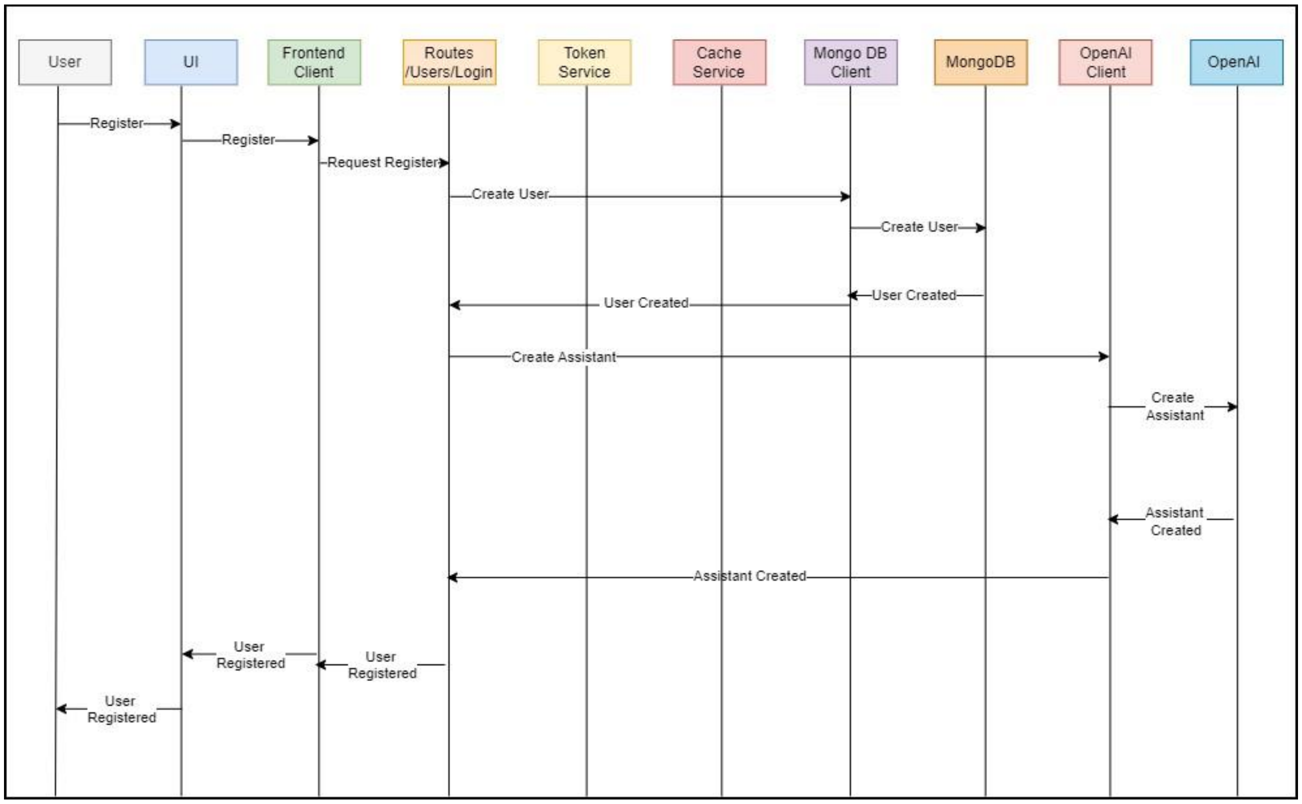


Figure 16: Register endpoint

2. Login endpoint

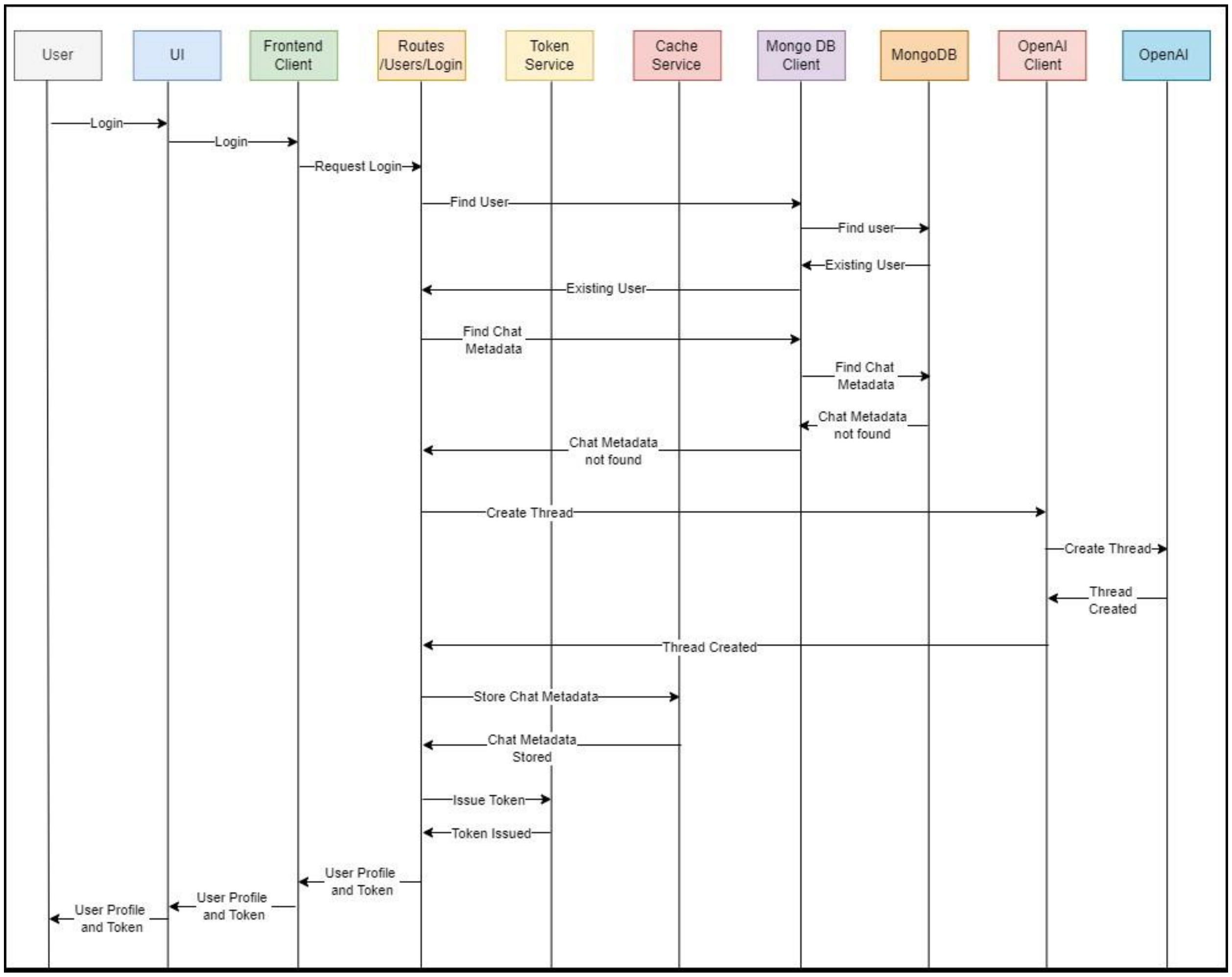


Figure 17: Login endpoint

3. Chat endpoint

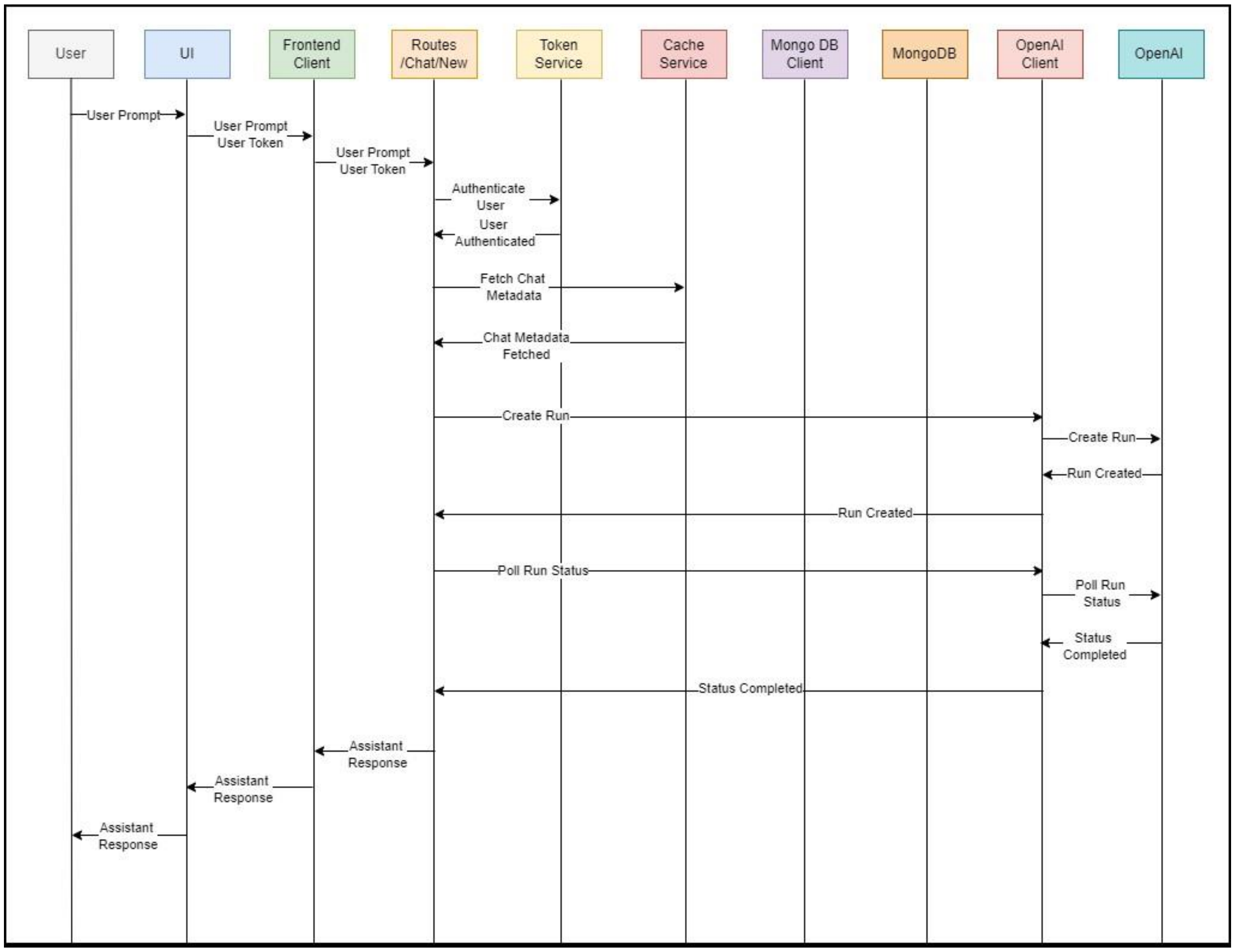


Figure 18: Chat endpoint

4. Delete/Logout endpoint

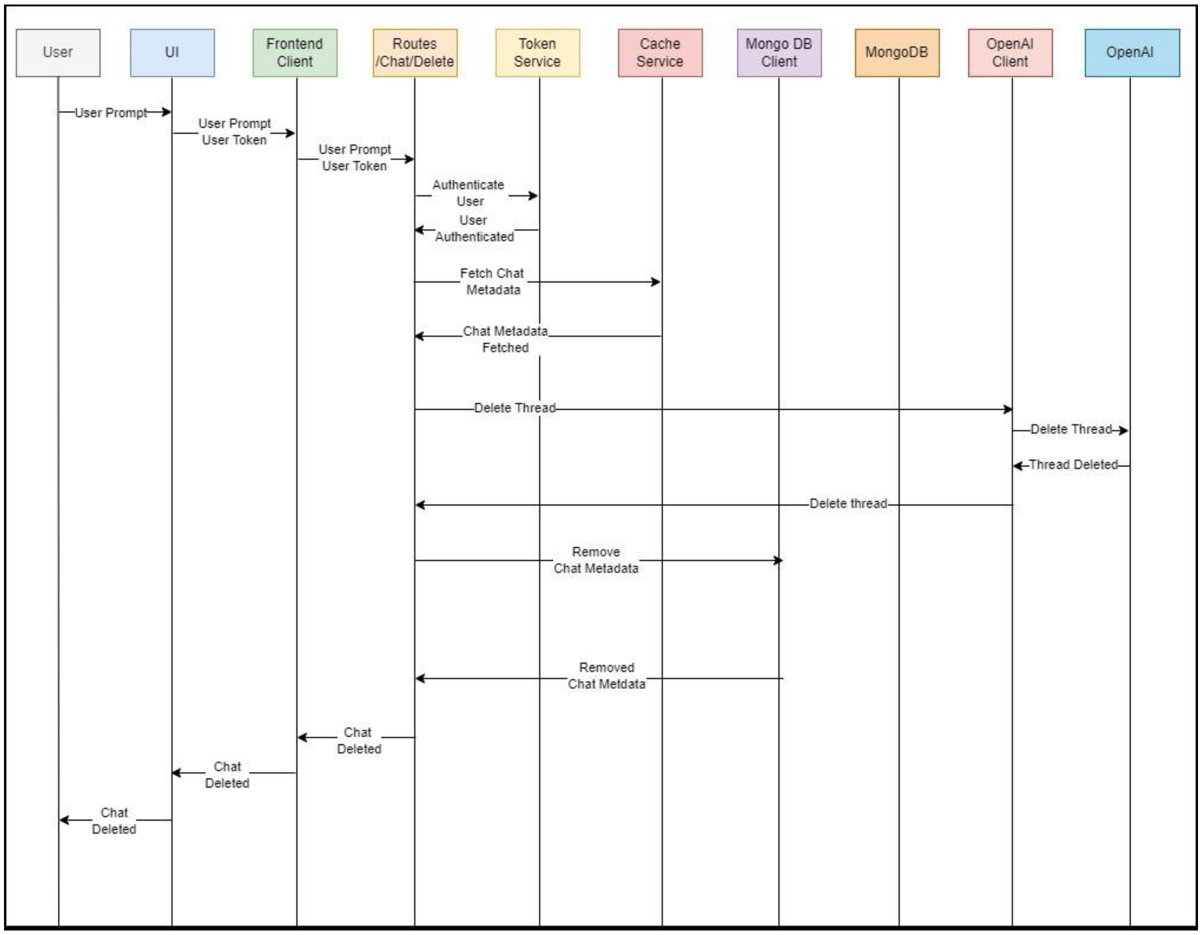


Figure 19: Delete/Logout endpoint

2.6 Database – MongoDB

MongoDB is a NoSQL database that uses a format like JSON (readable to humans and machines), which makes MongoDB different from other databases that organize data in table format. MongoDB is popular to store different kinds of data altogether without having a fixed rigid structure. MongoDB is flexible and scalable which makes it popular to store large amounts of diverse data. It is widely popular to manage a lot of complex information like big data applications and content management. The reason for the selection of MongoDB for this project is due to there is no relation between the users. On every user register a new assistant is created. On each user's login a new thread is created. This shows the application doesn't need any relational information to save. Each user is independent from each other, and data is stored in a document in MongoDB.

User Document Model
id: guid
firstname: str
lastname: str
password: str
email: str
location: str
dietary_preference: str
assistant_id: optional[str]

Table 9: User document model

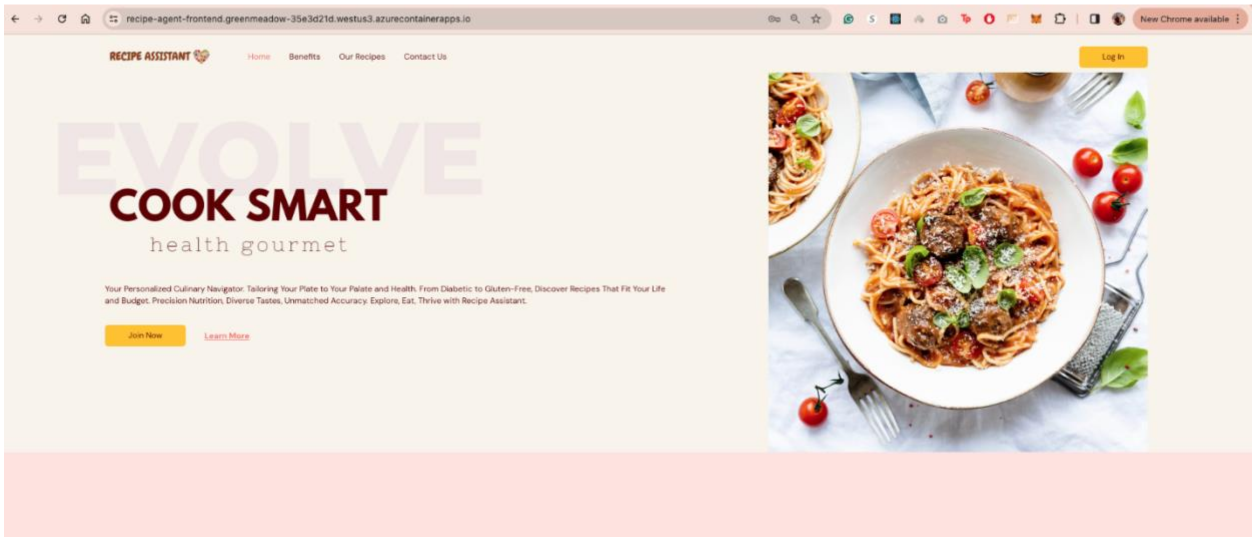
Chat Document Model
id: guid
assistant_id: str
thread_id: str

Table 10: Chat document model

2.7 Tech Stack

Languages	Python, NodeJS, TypeScript
Framework	Fast API, ReactJS, React Redux, Axios, Material UI, Tailwind CSS
Version control	GitHub
Tools	Visual Studio Code
API	USDA Food DataCentral, OpenAI Assistant
Database	MongoDB
Cloud	Azure Container Apps
Package tool	Docker, NPM, pip
Libraries	Pandas, Beautiful Soup, Numpy

2.8 Frontend

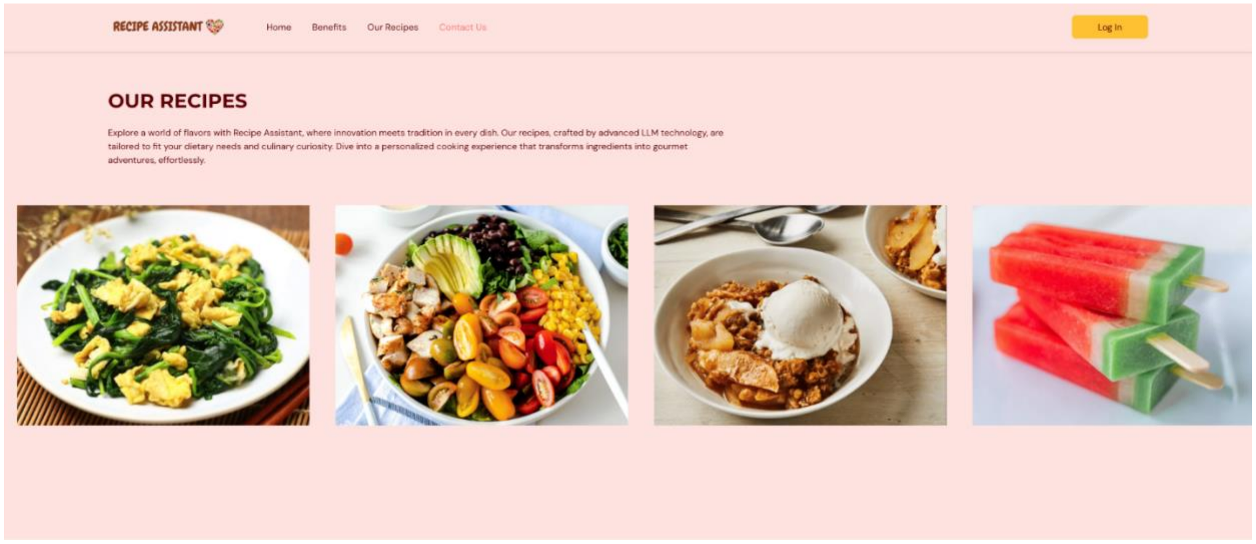


COOK SMART

health gourmet

Your Personalized Culinary Navigator. Tailoring Your Plate to Your Palate and Health. From Diabetic to Gluten-Free, Discover Recipes That Fit Your Life and Budget. Precision Nutrition, Diverse Tastes, Unmatched Accuracy. Explore, Eat, Thrive with Recipe Assistant.

Figure 20: Landing page



OUR RECIPES

Explore a world of flavors with Recipe Assistant, where innovation meets tradition in every dish. Our recipes, crafted by advanced LLM technology, are tailored to fit your dietary needs and culinary curiosity. Dive into a personalized cooking experience that transforms ingredients into gourmet adventures, effortlessly.

Figure 21: Our recipes in web application

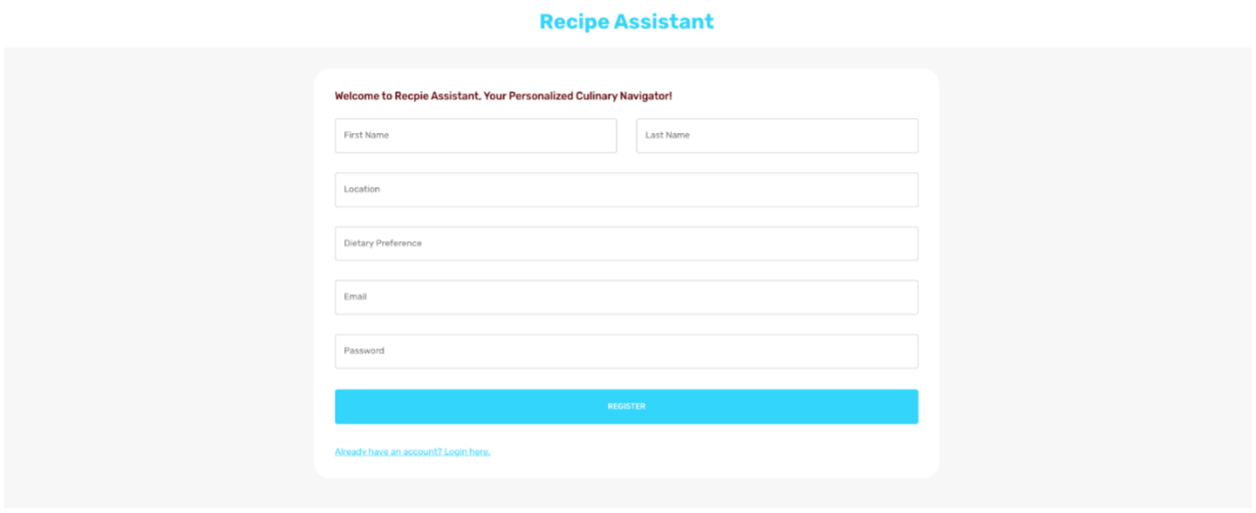

Recipe Assistant
Welcome to Recpie Assistant, Your Personalized Culinary Navigator!
First Name
Last Name
Location
Dietary Preference
Email
Password
REGISTER
Already have an account? Login here.

Figure 22: Register page

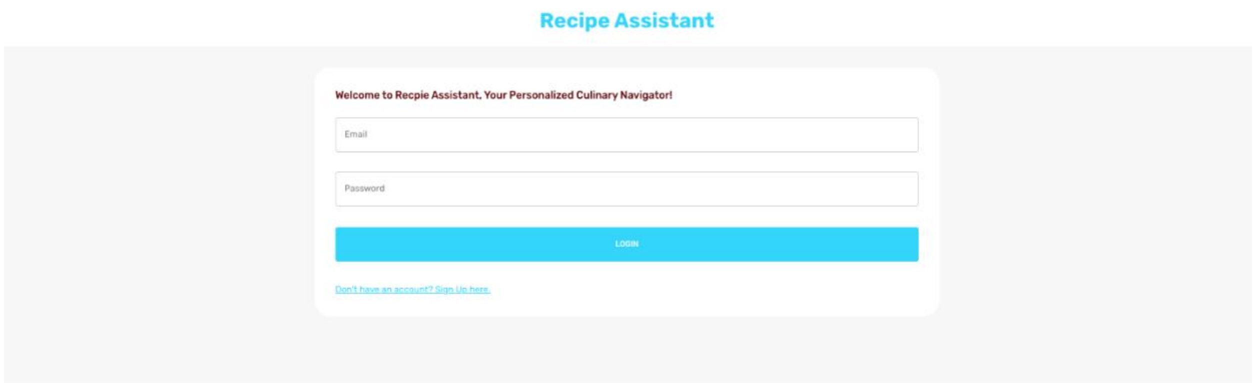
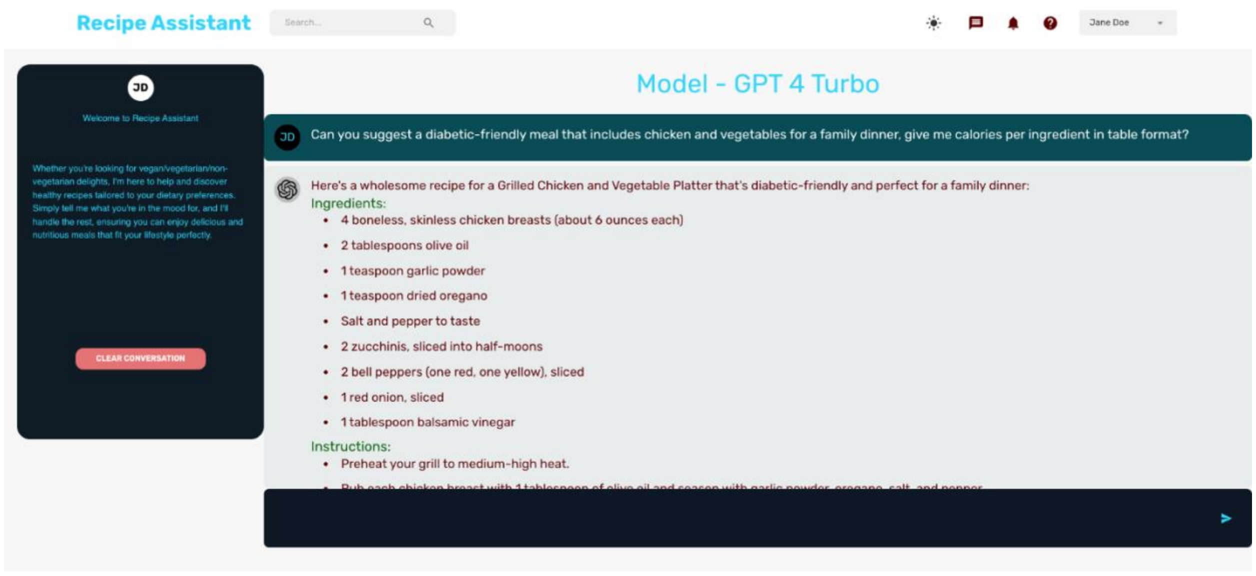

Recipe Assistant
Welcome to Recpie Assistant, Your Personalized Culinary Navigator!
Email
Password
LOGIN
Don't have an account? Sign Up here.

Figure 23: Login page



Welcome to Recipe Assistant

Whether you're looking for vegan/vegetarian/non-vegetarian delights, I'm here to help and discover healthy recipes tailored to your dietary preferences. Simply tell me what you're in the mood for, and I'll handle the rest, ensuring you can enjoy delicious and nutritious meals that fit your lifestyle perfectly.

Model - GPT 4 Turbo

Can you suggest a diabetic-friendly meal that includes chicken and vegetables for a family dinner, give me calories per ingredient in table format?

Here's a wholesome recipe for a Grilled Chicken and Vegetable Platter that's diabetic-friendly and perfect for a family dinner:

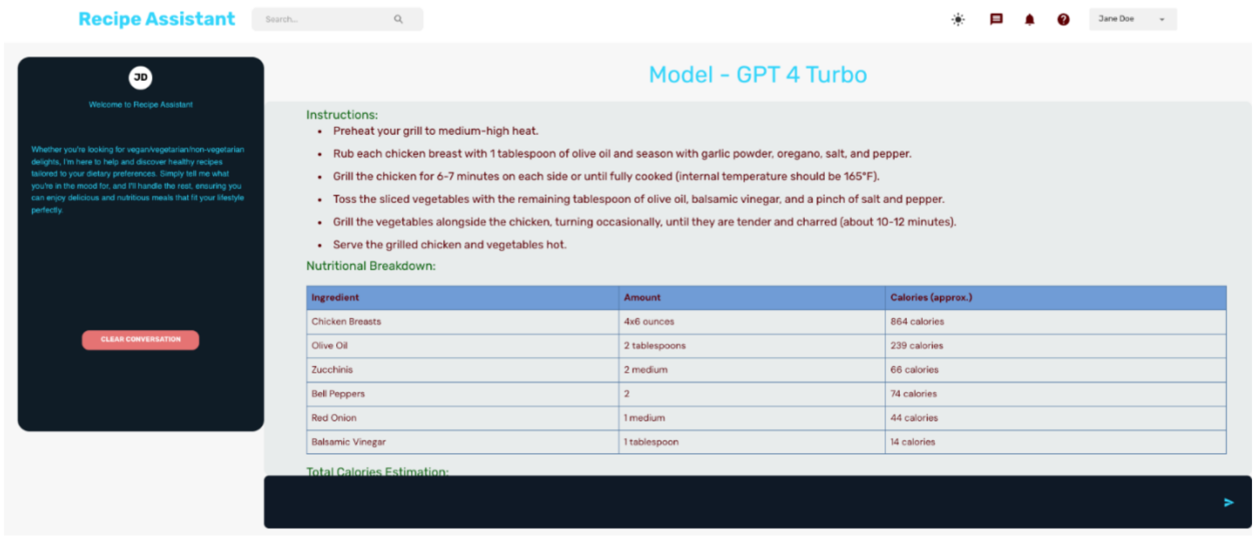
Ingredients:

- 4 boneless, skinless chicken breasts (about 6 ounces each)
- 2 tablespoons olive oil
- 1 teaspoon garlic powder
- 1 teaspoon dried oregano
- Salt and pepper to taste
- 2 zucchinis, sliced into half-moons
- 2 bell peppers (one red, one yellow), sliced
- 1 red onion, sliced
- 1 tablespoon balsamic vinegar

Instructions:

- Preheat your grill to medium-high heat.
- Rub each chicken breast with 1 tablespoon of olive oil and season with garlic powder, oregano, salt, and pepper

Figure 24: Chat Playground



Welcome to Recipe Assistant

Whether you're looking for vegan/vegetarian/non-vegetarian delights, I'm here to help and discover healthy recipes tailored to your dietary preferences. Simply tell me what you're in the mood for, and I'll handle the rest, ensuring you can enjoy delicious and nutritious meals that fit your lifestyle perfectly.

Model - GPT 4 Turbo

Instructions:

- Preheat your grill to medium-high heat.
- Rub each chicken breast with 1 tablespoon of olive oil and season with garlic powder, oregano, salt, and pepper.
- Grill the chicken for 6-7 minutes on each side or until fully cooked (internal temperature should be 165°F).
- Toss the sliced vegetables with the remaining tablespoon of olive oil, balsamic vinegar, and a pinch of salt and pepper.
- Grill the vegetables alongside the chicken, turning occasionally, until they are tender and charred (about 10-12 minutes).
- Serve the grilled chicken and vegetables hot.

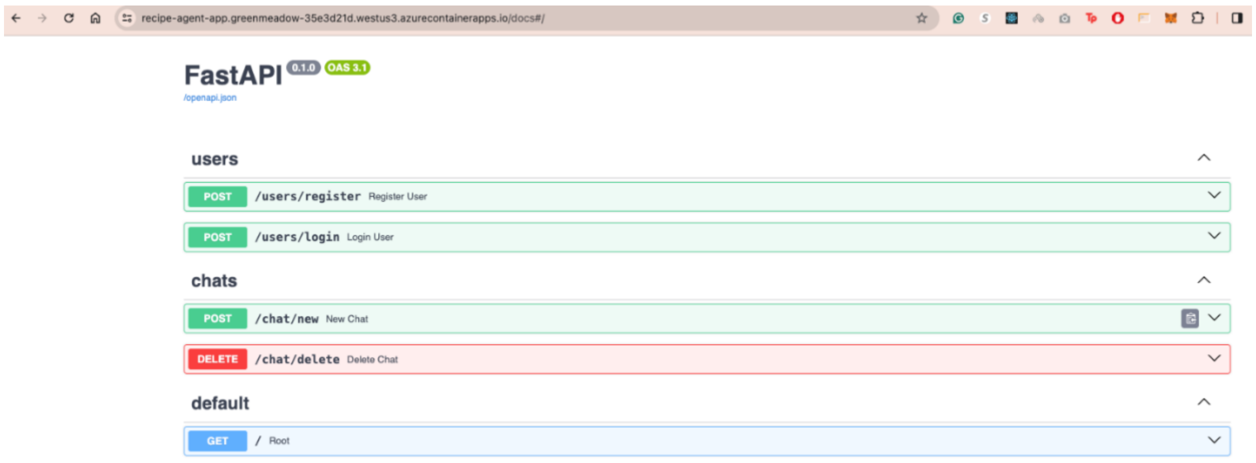
Nutritional Breakdown:

Ingredient	Amount	Calories (approx.)
Chicken Breasts	4x6 ounces	864 calories
Olive Oil	2 tablespoons	239 calories
Zucchinis	2 medium	66 calories
Bell Peppers	2	74 calories
Red Onion	1 medium	44 calories
Balsamic Vinegar	1 tablespoon	14 calories

Total Calories Estimation:

Figure 25: Chat Playground

2.9 Backend



recipe-agent-app.greenmeadow-35e3d21d.westus3.azurecontainerapps.io/docs#/

FastAPI 0.1.0 OAS 3.1

/openapi.json

users

POST /users/register Register User

POST /users/login Login User

chats

POST /chat/new New Chat

DELETE /chat/delete Delete Chat

default

GET / Root

Figure 26: Backend endpoints

3. Conclusion and future Work

We presented an approach to quantitatively evaluate the quality of AI-generated recipes by comparing it with a trusted collection of recipes. The data processing pipeline for this evaluation uses standard NLP techniques, including Named Entity Recognition (NER) to extract each ingredient from the recipes and cosine similarity to assign a match score on a continuum. We applied this data processing pipeline to evaluate the quality of GPT4 models for generating recipes for diabetics. The results show that the ingredients list in the AI-generated recipe matches 67-88% (average 77.1%) with the ingredients in the equivalent recipe in the ground truth database. The corresponding cooking directions match 64-86% (average 74.3%). However, the ingredients in recipes generated by AI for diabetic's match those in known recipes in our ground truth datasets at widely varying levels – between 26-83%. This indicates that AI-generated recipes can be improved by assigning greater weight to the more important ingredients. On the other hand, the mean absolute percentage error in calories reported by GPT4-generated recipes is relatively small (0.14%). For future work, we intend to develop more sophisticated prompts that account for the relative importance of ingredients in a recipe. We will also evaluate the other nutrient attributes of AI-generated recipes, such as carbohydrate, fat, cholesterol, and sodium levels.

4. References

[1] O., Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., . . . Zoph, B. (2023, March 15). GPT-4 Technical Report. arXiv.org. https://arxiv.org/abs/2303.08774

[2] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., . . . Wen, J. R. (2023, March 31). A Survey of Large Language Models. arXiv.org. https://arxiv.org/abs/2303.18223

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). Attention Is All You Need. arXiv.org. https://arxiv.org/abs/1706.03762

[4] Lange, R. T., Tian, Y., & Tang, Y. (2024, February 28). Large Language Models As Evolution Strategies. arXiv.org. https://arxiv.org/abs/2402.18381

[5] Roy, A. (2021, January 25). Recent trends in named Entity recognition (ner). arXiv.org. https://arxiv.org/abs/2101.11420v1

[6] Sennrich, R., Haddow, B., & Birch, A. (2015, August 31). Neural Machine Translation of Rare Words with Subword Units. arXiv.org. https://arxiv.org/abs/1508.07909

[7] Chen, B., Zhang, Z., Langrené, N., & Zhu, S. (2023, October 23). Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review. arXiv.org. https://arxiv.org/abs/2310.14735

[8] M. Chen, X. Jia, E. Gorbonos, C. T. Hoang, X. Yu, and Y. Liu, "Eating healthier: Exploring nutrition information for healthier recipe recommendation," Information Processing & Management, vol. 57, no. 6, p. 102051, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S030645731930161X

[9] H. H. Lee, K. Shu, P. Achananuparp, P. K. Prasetyo, Y. Liu, E.-P. Lim, and L. R. Varshney, "Recipegpt: Generative pre-training based cooking recipe generation and evaluation system," in Companion Proceedings of the Web Conference 2020, 2020, pp. 181–184.

[10] A. F. U. R. Khilji, R. Manna, S. R. Laskar, P. Pakray, D. Das, S. Bandyopadhyay, and A. Gelbukh, "Cookingqa: answering questions and recommending recipes based on ingredients," Arabian Journal for Science and Engineering, vol. 46, no. 4, pp. 3701–3712, 2021.

API Reference:

1. OpenAI API: https://platform.openai.com
2. USDA Food Data Central: FoodData Central (usda.gov)
3. FastAPI: https://fastapi.tiangolo.com/