CONSUMERWISE: A GENAI-POWERED NUTRIENT ANALYSIS TOOL

Harshith M.R
Department of Mechanical Engineering
Indian Institute of Technology, Madras
Chennai, Tamil Nadu 600036
me23b049@smail.iitm.ac.in

Aadarsh Ramachandran
Department of Electrical Engineering
Indian Institute of Technology, Madras
Chennai, Tamil Nadu 600036
ee23b001@smail.iitm.ac.in

Krishna Murari Chivukula
Department of Mechanical Engineering
Indian Institute of Technology, Madras
Chennai, Tamil Nadu 600036
me23b233@smail.iitm.ac.in

Sriprakash T
Department of Metallurgical and Materials Engineering
Indian Institute of Technology, Madras
Chennai , Tamil Nadu, 600036
mm23b066@smail.iitm.ac.in

October 2, 2024

ABSTRACT

This report serves as a project proposal for the Google GenAI Exchange Hackathon , detailing our review on ConsumerWise, a GenAI powered nutrient analysis tool/chatbot that allows users to validate their food choices through informed decisions. We attempt to propose a workflow of our implementation and offer users an insight into the nutritional aspects of the food products they wish to consume. The implementation involves an end-to-end chatbot , supported with Context Awareness and API based techniques. We further attempt to guide the user to appropriate nutritional choices , taking into account their past medical history , current lifestyle and other parameters.

1 Introduction - Addressing the Problem

The imperative for a simplified nutritional information tool stems from the multifaceted challenges consumers face in understanding and utilizing complex food labels. Modern labels often contain dense, technical information that can be difficult to comprehend, leading to confusion and misinterpretations. In today's fast-paced lifestyle, spending time deciphering nutritional labels can be inconvenient and time-consuming, particularly for individuals with busy schedules.

Moreover, consumers have diverse dietary needs and goals, and navigating complex label information to ensure their choices align with these requirements can be daunting. The prevalence of misinformation and misconceptions about nutrition further exacerbates the issue, as consumers may overlook crucial details or make unintended dietary choices. Finally, comparing similar products based solely on label information can be challenging, as consumers often struggle to identify the healthiest option.

By addressing these challenges, our tool aims to empower individuals to make informed and healthier food choices, ultimately promoting overall well-being By scanning a product's label, our tool quickly analyzes the nutritional data and presents it in a clear, concise, and easy-to-understand format.

This empowers users to make informed choices about their diet without the hassle of deciphering complex labels. Additionally, our chatbot attempts to provide personalized guidance, offering suggestions for healthier alternatives based on individual needs and preferences.

2 Architectural Workflow:

2.1 Pipeline Overview:

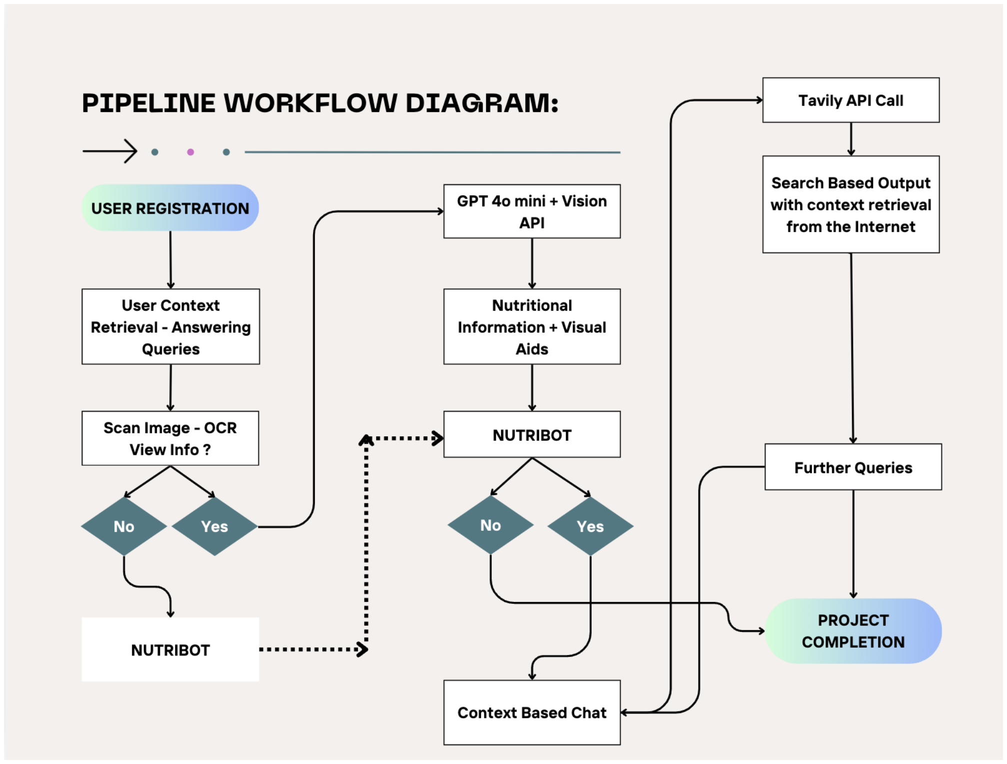

PIPELINE WORKFLOW DIAGRAM:
USER REGISTRATION
User Context Retrieval - Answering Queries
Scan Image - OCR View Info ?
No
Yes
NUTRIBOT
GPT 4o mini + Vision API
Nutritional Information + Visual Aids
NUTRIBOT
No
Yes
Context Based Chat
Tavily API Call
Search Based Output with context retrieval from the Internet
Further Queries
PROJECT COMPLETION

Figure 1: Pipeline Architecture: The chart above provides a high-level schematic representation of the input, output, and middle level flow of information.

2.1.1 Task modeling:

We approach this task as a Generative AI problem. For a given input image pertaining to the nutrients on a food product, we attempt to provide a nutrient analysis and answer further queries based of personalised user context. The project implementation consisted of 5 stages -

1. Ideation
2. Research and Planning
3. Initial Implementation
4. Iterative Testing and Refinement
5. Project Closure

These are detailed in the graph below.

2.1.2 Points of Interest:

User Readability: The main attempt is to provide nutritional information in a concise manner that is user readable , further assisted by visual aids such as pie-charts and histograms. This is to give the user a starting point.

Personalisation: This involves Context Awareness , answering user-level queries based of the information uploaded by them. It attempts to interlink the current analysis with these queries for personalised answers.

Data Interpretation and Visualization: Interpreting extracted data and provide insights, such as analyzing nutritional content or suggesting diet alternatives, and visualizing the data to enhance user comprehension.

Integration with User Data: Connecting nutritional recommendations with user habits (e.g., current exercise routine, stress level) for a holistic health approach.

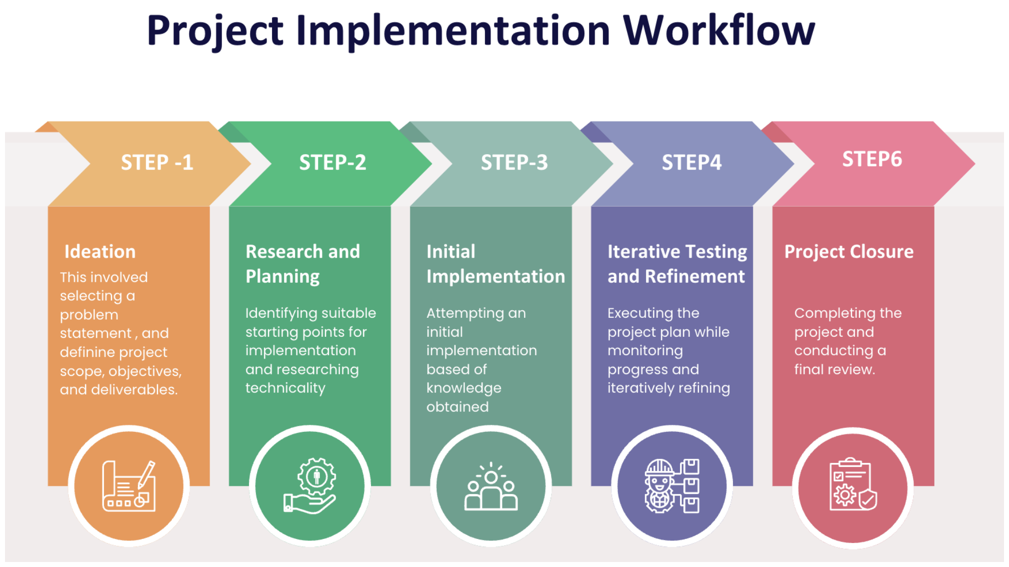

Project Implementation Workflow
STEP -1
STEP-2
STEP-3
STEP4
STEP6
Ideation
This involved selecting a problem statement , and definine project scope, objectives, and deliverables.
Research and Planning
Identifying suitable starting points for implementation and researching technicality
Initial Implementation
Attempting an initial implementation based of knowledge obtained
Iterative Testing and Refinement
Executing the project plan while monitoring progress and iteratively refining
Project Closure
Completing the project and conducting a final review.

Figure 2: A General Overview at our implementation

2.2 Optical Character Recognition - OCR:

The foundational phase of the pipeline centers on the task of parsing the product image furnished by the user. This process, orchestrated by **Optical Character Recognition (OCR)**, involves the meticulous extraction of structured information pertaining to macro and micronutrients, vitamins, minerals, and allergens. The OCR, powered by a **GPT-4o Mini** LLM, leverages the **Vision API** as a wrapper to adeptly process the uploaded image and generate a JSON output, encapsulating the extracted data in a standardized and machine-readable format. Our OCR implementation is engineered to extract nutritional information from real world product labels, and adverse conditions, such as suboptimal image quality or labels affixed to curved surfaces, like those found on bottles, attempting to improve accuracy of processing.

2.3 Rendering Visual Aids:

To enhance the visual comprehension of nutritional data, a form multi-modality was incorporated by rendering pie charts that graphically represent the distribution of macro and micronutrients. These visual aids, generated using **Matplotlib**, offer a more intuitive understanding of nutrient composition. Additionally, we have implemented a feature that allows users to add custom annotations to the charts, further tailoring the visualization to their specific needs and preferences.

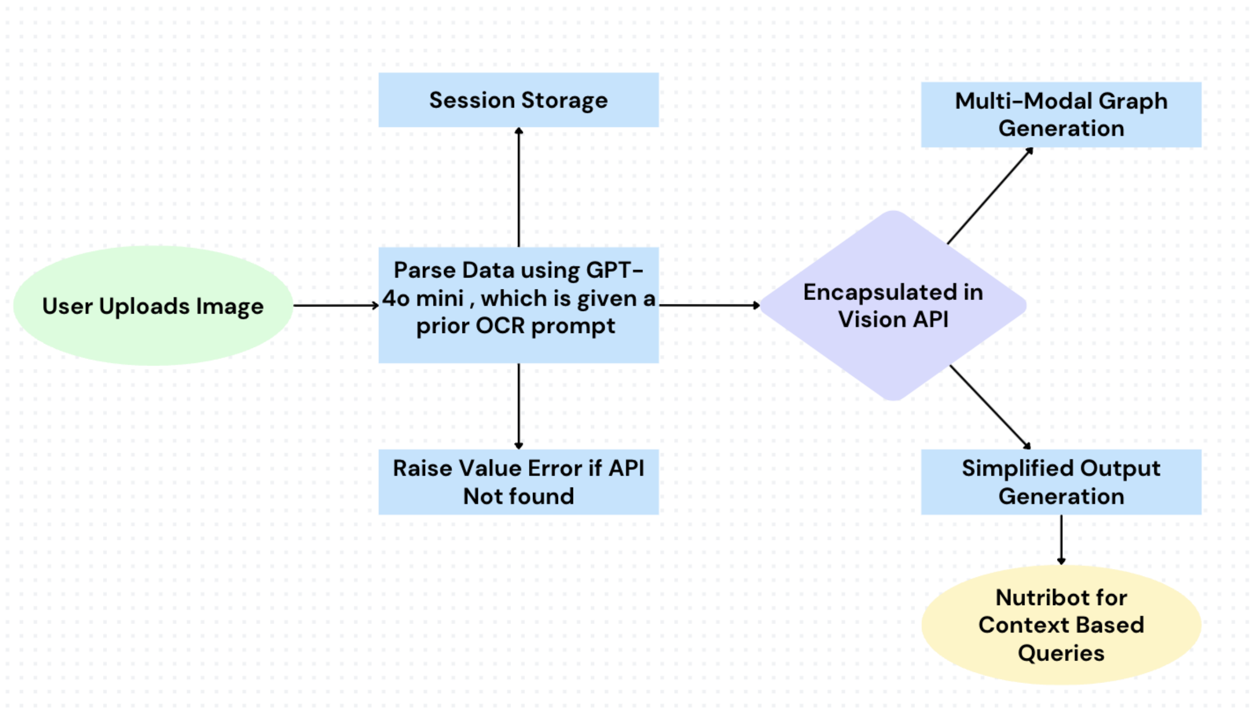

Session Storage
Multi-Modal Graph Generation
User Uploads Image
Parse Data using GPT-4o mini , which is given a prior OCR prompt
Encapsulated in Vision API
Raise Value Error if API Not found
Simplified Output Generation
Nutribot for Context Based Queries

Figure 3: Rudimentary Workflow for Optical Character Recognition

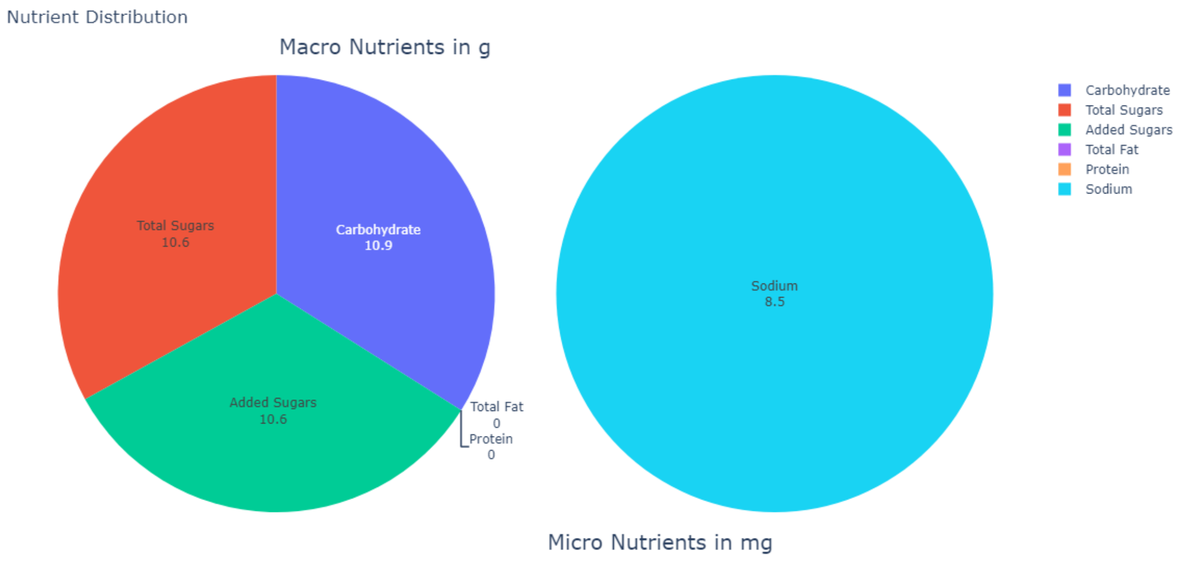


Figure 4: Visualisation and Graphical Analysis

2.4 Frontend UI and Backend:

The application is constructed with a **Flask** backend, utilizing **FastAPI** to wrap all large language model (LLM) calls for the API endpoints involved in the Flask environment. This was chosen to enhance the speed of the responses for prompts.

The frontend is developed using standard tools, namely **HTML, CSS, JavaScript, and Bootstrap**. **Bootstrap** was chosen for its ability to simplify responsive web design and ensure consistent styling across different devices, making the website adaptable to various screen sizes.

User authentication is handled using the **Werkzeug** library, providing a personalized experience. However, the app also allows access without login, albeit with limited flexibility compared to authenticated usage. **SQLAlchemy** is employed for session storage of user data, which serves well for the current scope; if the application scales up, transitioning to a more robust solution may be necessary.

Ajax is integrated within the **JavaScript** environment to enhance the app's functionality by enabling asynchronous operations, providing a smooth and interactive user experience. Overall, the technology stack ensures a flexible, scalable, and responsive application that offers personalized services to users.

2.5 Context Awareness:

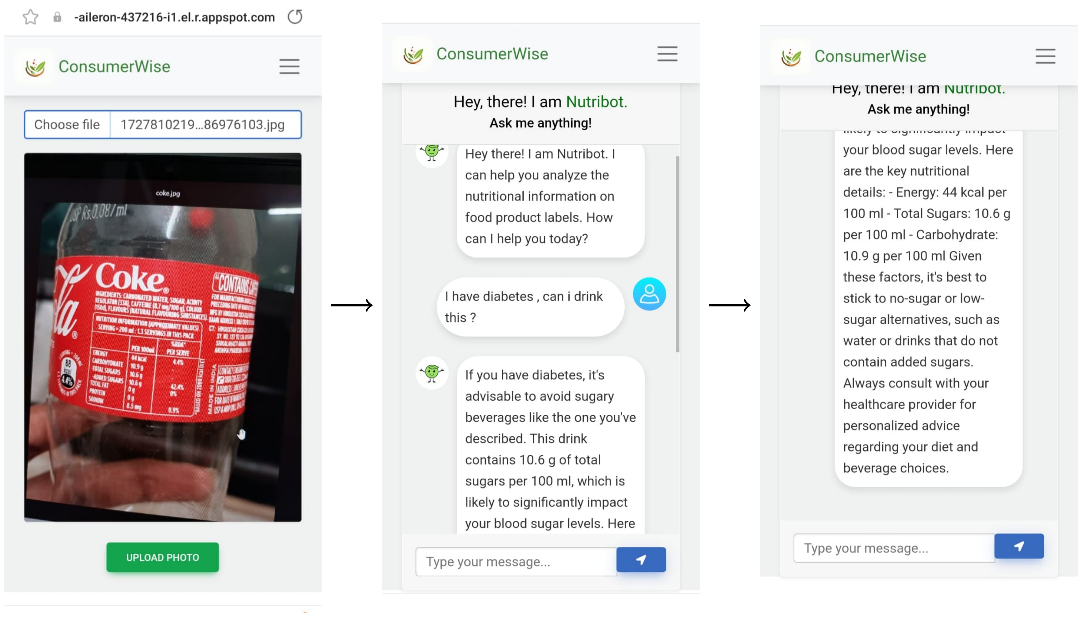
Prior to uploading image , during user authentication , some standard questions are used for obtaining general user context , in order to provide user - personalised outputs.

The provided questions are designed to gather comprehensive personal information to understand an individual's health, lifestyle, and dietary habits. They cover various aspects, including demographic details (age, gender), physical attributes (height, weight), activity level, and health-related factors (medical conditions, allergies, intolerances). The questions also explore dietary preferences and restrictions, current fitness and nutritional goals, lifestyle habits such as sleep patterns, occupation, and stress levels, as well as exercise routines.

Overall, the questions aim to create a profile of the individual, which can be used for personalized health, nutrition, or fitness recommendations.In no fashion is this data stored/accessed by us .

Open AI's **GPT4o mini** was used as a standard option for the for all the LLM queries involved.

This displays recommendations about the nature of the product personalised to your lifestyle , what you would gain in specific as well as the alternative choices you could possibly choose.



Hey, there! I am Nutribot.
Ask me anything!

Hey there! I am Nutribot. I can help you analyze the nutritional information on food product labels. How can I help you today?

I have diabetes , can i drink this ?

If you have diabetes, it's advisable to avoid sugary beverages like the one you've described. This drink contains 10.6 g of total sugars per 100 ml, which is likely to significantly impact your blood sugar levels. Here

your blood sugar levels. Here are the key nutritional details: - Energy: 44 kcal per 100 ml - Total Sugars: 10.6 g per 100 ml - Carbohydrate: 10.9 g per 100 ml Given these factors, it's best to stick to no-sugar or low-sugar alternatives, such as water or drinks that do not contain added sugars. Always consult with your healthcare provider for personalized advice regarding your diet and beverage choices.

Figure 5: Sample Context Awareness Chat

2.6 Technicalities of API Calls Used:

A tool based system is employed for deciding whether or not to call **Tavily** - a search API , which scrapes the net to obtain information related to the user query, for additional context.This decision is based of whether or not the user provides login details. An idea for implementation of Retrieval Augmented Generation(RAG) was also considered and benchmarked with **Tavily** , but was not used in the final implementaion. It is still on the table for future scaling.

2.6.1 Tavily API:

Tavily API is a search API that is involved in obtaining information from the internet, given the user query. It is a search engine optimized for LLMs (our use case), aimed at efficient, quick, and persistent search results.

2.6.2 FastAPI:

FastAPI is a simple wrapper around Python functions required to serve API endpoints. Used due to it's short functionality that minimized code duplication Multiple features from each parameter declaration could be extracted. Robust and fast , on par with **NodeJS**.

2.7 Benchmarking Retrieval Augmented Generation(RAG)with Tavily API:

An ideation was considered for the implementation of Agentic RAG for Context Awareness, however this approach was implemented in the final outcome. Nonetheless, a comprehensive benchmarking was performed between RAG and **Tavily API**.

Agentic RAG combines an agent(utilizing **LangChain**) with the traditional RAG approach to enhance response accuracy and relevance. The agent uses **OpenAI's** model through **LangChain's** `ZERO_SHOT_REACT_DESCRIPTION`, serving as a decision-making mechanism to determine whether RAG is needed based on the query content. For retrieval, a FAISS index is employed to store pre-computed document embeddings, enabling fast similarity searches. When a query requires RAG, the system uses the '**all-MiniLM-L6-v2**' Sentence Transformer model to encode queries into vectors, which are then matched against the FAISS index to retrieve relevant documents. These documents are formatted and appended as context to the conversation history, creating an augmented context for the response.

Finally, the **OpenAI** API generates the response, taking into account the complete augmented context to ensure a coherent and contextually appropriate answer. This implementation leverages **LangChain's** `initialize_agent` with `AgentType.ZERO_SHOT_REACT_DESCRIPTION` to manage the decision-making process and integrates a scalable retrieval mechanism for enhanced, context-aware responses.

On the other hand, the **Tavily API** offers several advantages that align well with our needs. It allows for faster implementation and lower operational complexity, as there is no requirement for maintaining a separate index or embedding model. It is also a more cost-effective option compared to maintaining an in-house RAG system. Given these benefits, it provides a streamlined and efficient solution, making it the better choice for our use case, considering the scale of our attempt.

A study published in the journal **"Artificial Intelligence Review"**(2023),[1] on various RAG implementations found that the average query processing time ranged from 2 to 5 seconds, with an accuracy improvement of 15-30% compared to non-RAG language models on domain-specific tasks. An industry report on an AgenticRAG-like system showed a 40% reduction in hallucinations compared to standard language models and a 25% improvement in answer relevance for complex, multi-step queries. Additionally, a preprint on arXiv(2024)[2] discussing agent-based RAG systems reported an average response time of 3.2 seconds for queries requiring multi-hop reasoning and an 85% accuracy rate on a benchmark of complex, domain-specific questions. Meanwhile , the below data was a benchmark for response time of **Tavily API**.

The above advantages highlighted above are our reasons for using **Tavily API** in comparision to RAG. However , cards are still on the table and an implementation using Agentic RAG is still under consideration for later phases of development.

[1]Artificial Intelligence Review"(2023)
[2]arXiv(2024)

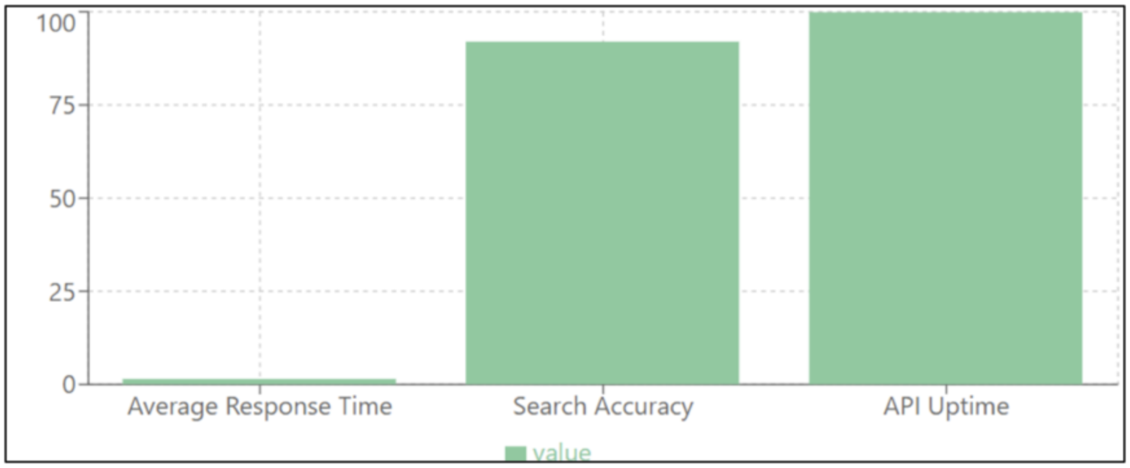
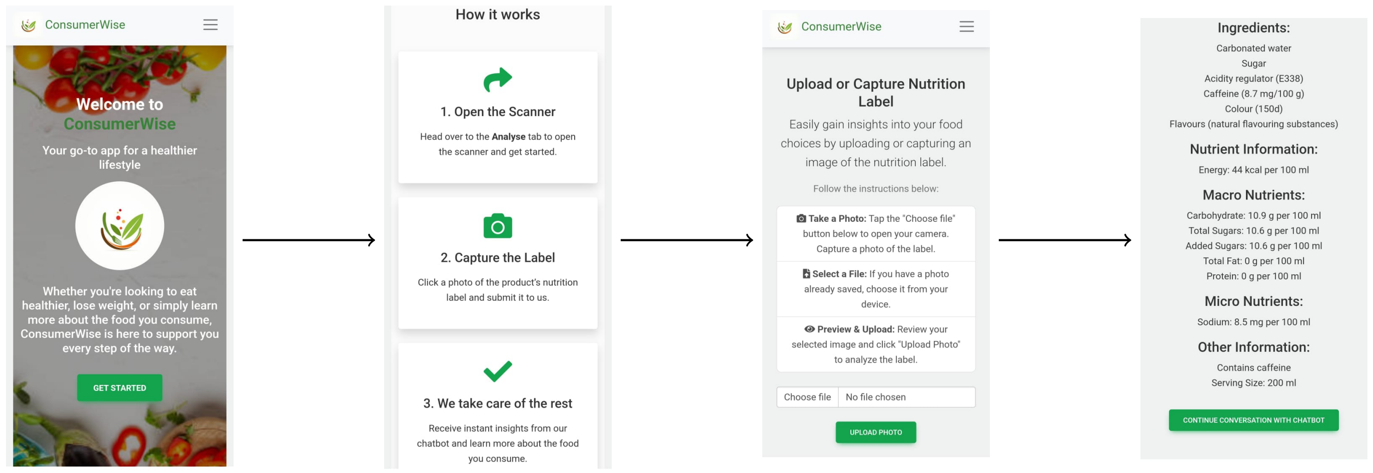


Figure 6: Tavily's Internal Benchmark Metrics

Aspect	**Tavily API**	**Agentic RAG**
Ease of Use	Simple API calls, minimal setup required	Requires complex setup (index, embeddings, agent configuration)
Speed	Generally faster, as it is a dedicated search service	Slower due to multiple steps (decision-making, retrieval, formatting)
Accuracy	Likely more accurate for general queries due to its vast, up-to-date knowledge base	Potentially more accurate for domain-specific queries if the index is well-curated
Customization	Limited customization options	Highly customizable (can modify decision logic, retrieval process, etc.)
Maintenance	Minimal maintenance required	Requires regular updates to the . index and potential model fine-tuning

Table 1: Comparison between Tavily API and Agentic RAG

3 User Workflow:



Welcome to ConsumerWise

Your go-to app for a healthier lifestyle

Whether you're looking to eat healthier, lose weight, or simply learn more about the food you consume, ConsumerWise is here to support you every step of the way.

How it works

1. Open the Scanner

Head over to the **Analyse** tab to open the scanner and get started.

2. Capture the Label

Click a photo of the product's nutrition label and submit it to us.

3. We take care of the rest

Receive instant insights from our chatbot and learn more about the food you consume.

Upload or Capture Nutrition Label

Easily gain insights into your food choices by uploading or capturing an image of the nutrition label.

Follow the instructions below:

Take a Photo: Tap the "Choose file" button below to open your camera. Capture a photo of the label.

Select a File: If you have a photo already saved, choose it from your device.

Preview & Upload: Review your selected image and click "Upload Photo" to analyze the label.

Ingredients:

Carbonated water
Sugar
Acidity regulator (E338)
Caffeine (8.7 mg/100 g)
Colour (150d)
Flavours (natural flavouring substances)

Nutrient Information:

Energy: 44 kcal per 100 ml

Macro Nutrients:

Carbohydrate: 10.9 g per 100 ml
Total Sugars: 10.6 g per 100 ml
Added Sugars: 10.6 g per 100 ml
Total Fat: 0 g per 100 ml
Protein: 0 g per 100 ml

Micro Nutrients:

Sodium: 8.5 mg per 100 ml

Other Information:

Contains caffeine
Serving Size: 200 ml

Figure 7: User Workflow Diagram

4 Challenges Faced:

As a group of college students attempting to build our first end-to-end chatbot and generative AI application, we faced numerous challenges throughout the process. From the outset, we had to navigate the vast landscape of LLM'a, which required extensive research. Understanding how LLMs function, selecting the appropriate models, and integrating them into our application was far from straightforward.

In addition to the complexities of working with AI, we also had to tackle the intricacies of web development, using technologies like **Flask, HTML, JavaScript, and CSS**. None of us had prior experience with these frameworks, and learning how to effectively build a cohesive web interface was a significant challenge.

In addition to these challenges, we started with an implementation of Agentic RAG, which turned out to be longer than we had anticipated. Furthermore, it produced unsatisfactory results. This discrepancy created complications in our integration efforts as a result we researched on **Tavily API** as an appropriate alternative which used in our final implementation.

Finally, deploying the application to **Google Cloud App Engine(GAE)** was an entirely new domain for us. Setting up hosting, configuring servers, and ensuring the application was accessible online involved many trials and errors, with obstacles such as troubleshooting deployment errors, configuring environments, and maintaining performance under different conditions. Transitioning from a local development environment to a cloud platform involved a steep learning curve, as we had to familiarize ourselves with various hosting options, server configurations, and deployment processes. Ensuring that our application ran smoothly in the cloud environment, with proper routing and data handling, added layers of complexity to our project.

Despite the various setbacks, each roadblock was a learning opportunity that ultimately helped us grow our skills in AI, web development, and cloud deployment, resulting in a fully functional end-to-end chatbot application.

5 Prospective Implementation Ideas:

Being our first outlook into this domain , we attempt to build up on our current state of idea and scale it. We plan to build:

1. **Native Mobile Application:** No internet connection required and an on-the-go form of service.
2. **Web Extension:**To work directly with delivery sites, resulting in a drag and drop fashion of the selected products , to get a fly-buy analysis.
3. **Agentic RAG:**Attempt to scale the functionality of RAG and succesfully implement integrate it with our chatbot. This would result in a greater degree of context based answers , perceivably resulting in a higher degree of personalisation.

6 Github and Video Submission:

Github Link to our repository can be found here. It also includes our video submission here.