As young adults begin to pay more attention to the foods they consume and seek to make informed decisions about their health and diet, there is a growing need for access to accurate and personalized nutritional information. However, navigating large nutrition datasets or searching for suitable recipes consisting of random ingredients can be unnecessarily time-consuming and confusing. Therefore, our group designed a web-based chatbot using Flask that provides precise nutritional facts and recipe suggestions using the question prompts that the user provides, such as *"How much protein is in chicken breast?"* or *"What can I cook with tofu?"*. The goal of this project was to simplify meal planning and encourage nutritional awareness. By combining time-specific datasets from USDA and the Spoonacular live API, the chatbot is able to deliver reliable and appropriate responses from both static and dynamic data sources. The application is deployed on a Google Cloud Platform virtual machine, ensuring public accessibility and constant uptime.

A key foundation of the project was the ETL (Extract, Transform, Load) pipeline built using data from USDA FoodData Central (FDC) as the source for nutritional information. This source was chosen for its variety and government-maintained standardization. The USDA FDC provides reliable, authoritative information on nutrients for both raw and processed foods that are consistently regulated. Using the *Foundation Foods* dataset from April 2025, we extracted `food.csv`, `nutrient.csv`, and `food_nutrient.csv` from the ZIP file. We then renamed the columns to match between the data frames created for each file using `pandas`. The datasets were filtered based on `NUTRIENTS_OF_INTEREST`, consisting of calories, protein, carbohydrates, and fat. These values were then consolidated into one cleaned dataset, `cleaned_nutrition_data.csv`, which was then loaded into the Flask app.

In addition to the USDA dataset, we also integrated the Spoonacular API as a live data source for recipe suggestions. This API was chosen specifically for its ability to return ingredient-based recipe results, making it perfect for questions like *"What can I cook with chicken?"*. The user's input is parsed to extract a list of ingredient keywords, which are then passed to the API as query parameters. In response, the API provides JSON text containing recipe titles, images, links, and the count of matched ingredients, which are then formatted into a recipe grid with clickable links and thumbnails on the results interface, coming from `results.html`. This use of the API allows the chatbot to offer personalized meal ideas that are catered to the user's inputs.

Developing the chatbot represented technical challenges that required careful design decisions. One of the most significant difficulties involved handling the variability of user input when referencing food items. Because users might refer to foods in various ways, whether using plural/singular forms, shortened phrases, or simple misspellings, exact string matching proved to be troublesome. To combat this, we used "fuzzy" string matching using the `rapidfuzz` library. This allowed the chatbot to match the user's inputs to the closest available entry in the USDA dataset. A related challenge was distinguishing between food and nutrient names. This was resolved by building a filtering layer that separates known nutrient keywords (fat, protein, etc.) from the remaining input before applying the matching logic.

Working on the chatbot offered a series of learning experiences, including the opportunity to build a full-stack application and deploy it in a public environment. Although we had previous experience with building web applications and using Flask, deploying a Flask application on a publicly accessible server introduced new challenges. We learned to configure `gunicorn` as a server and use systemd, which ensured that the application would restart automatically on reboot and run continuously in the background. We also gained experience with securing sensitive information, such as API keys, by using environment variables.

In the case that we were to work on this project further, there are several extensions that could expand the chatbot's functionality and impact. One enhancement would be to implement nutrition logging, allowing users to track their dietary intake over time and receive personalized insights based on their consumption history. This could then evolve into a meal planning assistant which could recommend certain diets or meals to users based on their nutritional goals or dietary restrictions. Another enhancement could be to implement a filter to account for certain dietary restrictions such as vegetarian, gluten-free, vegan, and more. Spoontacular already supports such filters, so this would be a simple implementation.

Overall, this project allowed us to create a practical and accessible nutrition chatbot that combines structured USDA data with live recipe suggestions from the Spoonacular API. Through building an ETL pipeline, integrating fuzzy matching for user inputs, and deploying the app using Flask and GCP, we gained valuable experience in full-stack development and real-world problem solving. The project highlighted the importance of clean data, responsive design, and user-centered features. With more time, we would expand its capabilities to include personalized meal planning, dietary filters, and nutrition tracking to further support healthy decision-making.