FLEX: AI-Powered Diet and Workout Plan Application

Jihoon Lee, Jaewook Shin, Mano Hong, Manu Gomez

Sungkyunkwan University

December 12, 2024

Abstract

Flex is a mobile application for recommending food and workout plan. Unlike common diet applications, FLEX not only reflects user's daily food preference, but also consider user's health. Using Aspect-Based Sentiment Analysis, we analyze users text about what kind of food they want, and recommend list of foods based on the analysis. Using user's muscle mass information, we recommend workout plan, separated by main part and additional part. Also, we limit user's one day calorie intake using user's food intake and weight change information. For development, we used Figma for UI/UX, React Native and Zustand for Frontend, Django and Docker for Backend, and PostreSQL for Database. For further enhancement, we aim to include more foods with details, and reflect user feedbacks for more precise food recommendation and personalized workout plans.

Keywords: Food Recommendation, Workout Plan, Calorie Bound, Aspect-Based Sentimental Analysis

1 Introduction

Our project focuses on delivering personalized recommendations for healthy eating and physical activity, tailored to individual tastes, goals, and preferences. Unlike generic solutions, our approach enhances personalization, making it easier to adopt and maintain healthy lifestyle habits.

The innovation lies in integrating nutrition and physical exercise by analyzing food preferences and aligning them with specific activity goals. This ensures that recommendations are effective, engaging, and customized to dietary restrictions and physical conditions.

The proposed solution is a mobile application that personalizes nutrition and activity plans, considering dietary preferences, restrictions, and weight management goals. By combining nutritional data, taste analysis algorithms, and personalized activity models, the system generates practical recommendations. The user-centric design ensures the app is intuitive, scalable, and impactful.

Empirical evaluations show the system effectively supports users in achieving weight loss, gain, or maintenance goals. Significant improvements in user satisfaction further validate its potential.

2 Design

2.1 Overall Architecture

2.1.1 Flow Chart

Link below shows our application's user flow. Users first have to create their own account for personalized food and exercise plan recommendation.

On Main Page, users can freely move to Food Recommendation Page, Food Select Page, and Exercise Page.

On the Food Recommendation Page, users can type what kind of food they want including tastes, cuisine, ingredients, or temperature of foods, and our application recommends a list of 5 foods.

On Food Select Page, users can type what food they ate and it is automatically saved.

On Exercise Page, users can type their weight and muscle mass information, and our application recommends their exercise plans about what body parts they should exercise and what types of exercise they should do based on those information.

Flow Chart link

2.1.2 Database Design

Link below shows our database design. We designed user related tables, food related tables, and exercise related tables.

Relational Schema Link

2.2 UI/UX

For the UI/UX design, we used Figma as our primary tool for creating and refining the app's design.

To reflect the concept of flexibility inherent in our app, FLEX, we implemented a linear gradient in the design. This gradient provided a dynamic and modern color scheme, aligning with the app's core theme of adaptability and movement.

We avoided using a traditional navbar since each function in the app is independent, ensuring users can directly access specific features from the main screen. This streamlined navigation approach reduces unnecessary steps and improves usability.

For surveys, we considered a seamless user experience by implementing an automatic popup modal upon login. This modal appears to prompt users for

input, eliminating the need for manual navigation while maintaining an intuitive flow.

By focusing on these design choices, we created a cohesive and user-centered interface that enhances engagement and usability.

2.3 Frontend

We used React Native for the frontend framework to build a cross-platform application with a single codebase. React Native allowed us to develop a consistent user experience on both Android and iOS devices, significantly reducing development time while maintaining performance.

For state management, we utilized Zustand, a lightweight and flexible library. Zustand simplified the management of complex states in our application by providing a clean and efficient API. This allowed us to create scalable and maintainable state management logic without adding unnecessary complexity.

By combining React Native and Zustand, we ensured a smooth and responsive user interface with robust state management, enhancing the overall user experience.

2.4 Backend

We used Django for the backend framework because we needed python's machine learning libraries for food recommendation. Also, we developed various apis very fast using Django's various features. With user management system, we used Django's User model and expanded it with some more fields. With user authentication, we simply developed apis that requires authentication. With Object Relational Mapping(ORM), we developed various features without SQL Queries.

We used PostgreSQL for our database to manage user data and our dataset. As a RDBMS, it adheres to the ACID (Atomicity, Consistency, Isolation, Durability) principles, ensuring reliability in data integrity and transaction management. Also it fits very well with Django ORM.

We used Docker for our server deployment. Docker ensures that an application runs consistently across different environments, such as development, testing, and production by packaging the application along with its dependencies into a container.

2.5 Challenge

2.5.1 Food Recommendation

At first, we planned to reflect individual's food preference scores for all foods, which makes our food recommendation more precise. By collecting food preferences for 100 different foods from 1000 people, we planned to design a AI model using AutoRec which predicts a new user's food preference scores for 100 different foods. However, it was impossible to collect food preference scores for

100 different foods from 1000 people. Thus, we changed our solution to analyze user's text about what kind of food they want for food recommendation.

2.5.2 Workout Plan

At first, we planned to design a AI model which recommends personalized workout plan. However, this required personal body information from many people, and it was impossible to collect. Thus, we changed our solution to use user's muscle mass percentage information, recommending workout plan based on user's muscle imbalance.

3 Implementation

3.1 Frontend

For the frontend implementation, we incorporated various tools and techniques to ensure a smooth, responsive, and user-friendly application. Below are the key components:

- **Async Storage and JWT Management**: We utilized Async Storage to securely store JSON Web Tokens (JWT) on the client side. This approach enables persistent authentication across sessions while maintaining a lightweight storage solution for tokens. To manage JWT efficiently, we implemented a JWT Manager. This handles token issuance, validation, and refreshing processes.
- **API Middleware with Axios**: Using a custom Axios configuration, we implemented an API middleware layer. This centralizes API requests, includes default headers like authorization tokens, and provides error-handling mechanisms for robust communication with the backend.
- **Global State Management with Zustand**: We used Zustand for managing user data across the application. Zustand's simple and lightweight API enabled us to efficiently manage global state while keeping the codebase clean and maintainable.
- **Animations and Blur Effects for Modals**: We implemented various animations and blur effects to enhance the user experience. For example, modal popups include smooth entry/exit animations combined with a background blur to create a visually appealing and interactive UI.
- **Componentized Codebase**: To improve maintainability and reusability, the frontend code was componentized. Each feature or UI element is modularized into independent components, adhering to the principles of separation of concerns and clean architecture.

These strategies and tools combined to deliver a robust and user-friendly frontend for our application.

3.2 Backend

According to Database Design in 2.1.2, we created database tables using Django's models, and by using Django ORM, we accessed database without SQL queries.

We used Django's User model for User Table, adding some required fields. User's password is automatically hashed by Django's hash algorithm, which is pbkdf2. This algorithm uses a unique salt which makes it more difficult for attackers to recover the original password, and the iterative process significantly increases the time required to crack the password. Thus this improves the overall security.

We used Django JWT authentication system for authentication. If user logins to our application, we issue user the access token and the refresh token. User can use all features in our application using access token. If the access token is expired, user can reissue it using the refresh token. If the refresh token is expired, user has to login again.

We used ChatGPT for predicting the calorie of the food which is not in our dataset. There are so many foods in the world, and we couldn't get all the information about them in advance.

3.3 Food Dataset

Our food dataset is designed to support the personalized recommendation system for food and nutrition analysis. This dataset contains multiple attributes such as food names, cuisine types, categories, ingredients, vegetarian status, allergy information, temperature preferences, and calorie counts. The dataset also includes descriptive keywords, which were extracted using the GPT-3.5 model with a customized keyword extraction pipeline.

Attributes Overview: The key attributes of the dataset include:

- **English Name:** The name of the dish in English (e.g., Gamjajeon, Kimchi Jjigae).
- **Characteristics:** Descriptive traits such as texture or flavor (e.g., "crispy," "spicy").
- **Cuisine:** The type of cuisine (e.g., Korean).
- **Category:** Food type such as "fried," "soup," or "stew."
- **Ingredients:** The main ingredients in English (e.g., potatoes, kimchi).
- **Vegetarian Status:** A binary indicator (1 for vegetarian, 0 for non-vegetarian).
- **Allergy Information:** Any allergy-related ingredients (e.g., shellfish, sesame).

- **Temperature:** The preferred serving temperature (e.g., warm).
- **Calories:** The calorie count of the dish.

Dataset Table: The table below displays the first five rows of the dataset, showcasing some key attributes:

Name	Char.	Cuisine	Cat.	Ingr.	Veg.	Cal.	Allergy
Gamjajeon	Crispy, nutty	Korean	Fried	Potatoes	1	400	None
Sundae Guk	Earthy	Korean	Soup	Pork belly	1	650	None
Suyuk Guk	Earthy	Korean	Soup	Pork spinach	0	700	None
Pork Bone	Spicy, nutty	Korean	Soup	Pork spine	0	700	None
Bean Sprout	Crunchy	Korean	Soup	Bean sprout	1	250	Shellfish

Table 1: Top 5 rows of the food dataset.

Future Expansion: Currently, the dataset contains a limited number of dishes. However, we plan to expand it significantly by incorporating additional dishes from various cuisines, including global and niche culinary options. This will enhance the diversity and adaptability of the recommendation system. Furthermore, we aim to add new attributes, such as preparation time and cooking methods, to make the dataset even more comprehensive.

3.4 Keywords extraction

We firstly open the dataset created by us manually and with help of chatGPT to get the foods' description. If we use an AI to generate descriptions of food, then we just need to get new food names and get the descriptions easily. This will make the program scalable.

Then we have made an algorithm to extract the keywords from food description. For that purpose we firstly tokenize each description and lemmatize it so we only get the base word. Words like *cooking* and *cooked* are equivalent to the word *cook*. Later, we got the nouns using POS tagging, which will only get the POS tags beginning with "NN". This POS tags are the words that tend to be entities or topics. We also got the entities by using NER. Finally, we merge the lists of nouns and entities we got from both, and we use that as food description keywords. There were some problems with this method. Firstly, we have to remove duplicates so we do not give double points for the same keywords. Secondly, we see that even with our algorithm, there were some words that should not be appearing in our keywords list. For that reason, we started creating a new list with keywords that should not appear in our list.

So after writing almost 100 words that should not appear, our word cloud changed a lot. We saw more important things such as rice, pork... and less non food related words such as hangover.

It is really important to have a good list of keywords on food description because these are the words that we will use for calculating similarities with

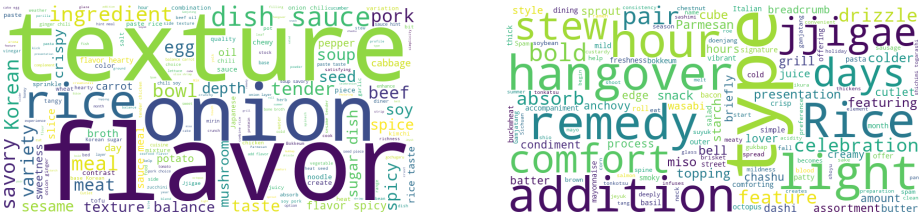

Figure 1: Word clouds generated using NER and POS tagging without cleaning useless words.

Figure 2: Word clouds generated using NER and POS tagging with manually cleaned useless words.

user's message keywords. Therefore, we can look for a good amount of quantity in user message keywords. Even if we got a lot of useless keywords in the user's message, when we calculate the similiraties with our food description keywords, if these are really great, the useless words should not be similar to anything. This will leading to not affecting the final score of each food.

So to be able to evaluate our user message keyword extraction algorithm, we asked to Chat GPT to create some messages and also to give us the positive and negative keywords from them. Our algorithm was based on the same as food description, but we also tried to search for exact words that are the same as the food description keywords list. Finally, we combine these two methods and we compare all three results. For comparison, we used coverage. We look up how many and also which keywords ChatGPT gave us as useful (positive or negative keywords) and we compare it to the keywords we got. The formula will be:

$$\text{Coverage} = \frac{|K_{\text{ChatGPT}} \cap K_{\text{Your}}|}{|K_{\text{ChatGPT}}|} \times 100$$

3.5 Sentiment Analysis

The food recommendation algorithm leverages a fine-tuned **DeBERT-a model** for *Aspect-Based Sentiment Analysis (ABSA)*. The model was fine-tuned using the **Amazon Food Reviews Dataset**, a dataset comprising approximately 500,000 reviews and ratings related to food and beverages. Each data

point in the dataset includes specific aspects (e.g., taste, texture, ingredients, presentation) and corresponding sentiment labels (positive, neutral, negative). The input text and each aspect were combined in a specific format, tokenized, and fed into the model for training.

Dataset Example:

- **Review Text:** "The chips were too salty and had a strange aftertaste."
- **Aspect:** "Taste"
- **Sentiment Label:** Negative

We tried using all the logits (three) and also eliminating the neutral one, so we can stick only with positive or negative. Due to the relatively small proportion of neutral sentiments, they were excluded from further analysis. This simplification allowed for a more focused and effective recommendation process.

The analysis results were then utilized to adjust the scores for each food item: scores were increased for items with positive sentiment and decreased for those with negative sentiment. This sentiment-based scoring adjustment significantly enhances the accuracy of food recommendations, ensuring alignment with user preferences.

3.6 Food Recommendation Algorithm

The food recommendation algorithm identifies the top 10 foods based on preference scores. Each food's description is preprocessed to extract key keywords, followed by sentiment analysis to remove negative keywords.

User-provided input (e.g., positive keywords, cuisine, tastes, ingredients, allergy information, vegan status) is compared with the food dataset. Overlapping attributes are counted, and scores are adjusted accordingly.

Duplicate and Negative Keywords

Duplicate matches were retained as they highlight significant characteristics of the food. For instance, if "spicy" is found in both taste and description, the food's preference score is increased. Conversely, negative keywords reduce scores, with stronger penalties reflecting their greater impact on decision-making.

Allergy Information and Past Consumption

Foods containing allergy-related ingredients were excluded entirely to ensure safety and preserve the food's identity. Additionally, previously consumed foods were assigned lower scores to promote the discovery of new options.

Final Scoring

The final preference score incorporates:

- Positive and negative keyword adjustments,
- Allergy information,
- Past consumption history.

The top 10 foods with the highest scores are then recommended to the user.

3.7 Exercise Recommendation Algorithm

We recommend two parts for exercise recommendation. One is main part which is recommending major muscle exercises, specifically chest, back, and legs. The other is additional part which is recommending minor muscle exercises, specifically arms, shoulders, and abs.

For main part, we compare user's body muscle mass percentage and legs muscle mass percentage. Suppose

$$\Delta main = \text{body muscle mass percentage - legs muscle mass percentage}$$

Then our main part recommendation is like below. 'Body' means to exercise chest and back alternatively.

$$\begin{gathered}
10 \leq \Delta main \leq 20\text{: body} \rightarrow \text{legs} \rightarrow \text{repeat} \\
\Delta main \geq 20\text{: legs} \rightarrow \text{body} \rightarrow \text{legs} \rightarrow \text{body} \rightarrow \text{legs} \rightarrow \text{repeat} \\
-20 \leq \Delta main \leq -10\text{: body} \rightarrow \text{body} \rightarrow \text{body} \rightarrow \text{legs} \rightarrow \text{repeat} \\
\Delta main \leq -20\text{: body} \rightarrow \text{body} \rightarrow \text{body} \rightarrow \text{body} \rightarrow \text{legs} \rightarrow \text{repeat} \\
otherwise\text{: body} \rightarrow \text{legs} \rightarrow \text{body} \rightarrow \text{repeat}
\end{gathered}$$

For additional part, we compare user's body muscle mass percentage and arms muscle mass percentage. Suppose

$$\Delta add = \text{body muscle mass percentage - arms muscle mass percentage}$$

Then our additional part recommendation is like below.

$$\begin{gathered}
10 \leq \Delta add \leq 20\text{: arms} \rightarrow \text{shoulders} \rightarrow \text{arms} \rightarrow \text{abs} \rightarrow \text{repeat} \\
otherwise\text{: arms} \rightarrow \text{shoulders} \rightarrow \text{abs} \rightarrow \text{repeat}
\end{gathered}$$

Everytime user updates their muscle mass percentage information, we immediately recommend the new exercise plan based on that information.

3.8 Food Calorie Bound Algorithm

User's food intake is received three times a day and every day, exercise data is received once a day and every day, and weight is received every 5 days starting from their initial weight. Every time the data on what user has eaten and

exercised for 5 days accumulates, we calculate the user's calorie bound for one day.

First, we calculate user's sum of calorie intake for 5 days and user's sum of calorie burn for 5 days, and calculate the difference of them which is calorie surplus for 5 days. We also calculate the weight change over 5 days.

Suppose we have datas for $5n$ days. Then there will be n calorie surpluses, suppose them from x_1 to x_n, and n weight changes, suppose them from y_1 to y_n. For two consecutive datas, we calculate

$$ans_k = x_k - \frac{y_{k+1} - y_k}{x_{k+1} - x_k} y_k \ (k = 1, 2, \ldots, n-1)$$

which means that if user's calorie intake for 5 days is ans_k, then user's weight change over 5 days is 0.

Second, we calculate the calorie bound for five days, the average of all ans_ks.

$$calorie\_bound\_for\_five\_days = \frac{1}{n-1} \sum_{k=1}^{n-1} ans_k$$

Finally, we calculate the calorie bound for each day by dividing $calorie\_bound\_for\_five\_days$ by 5.

$$calorie\_bound = \frac{calorie\_bound\_for\_five\_days}{5}$$

If user's goal is to lose weight, this becomes the upper bound, and if user's goal is to bulk up, this becomes the lower bound.

4 Evaluation

4.1 Keywords Extraction

4.1.1 Coverage results

As said before, we tried three different methods. The first one was using the same as food description algorithm. The second one was searching for the same words as food description keywords list. The third one was combining both. These were the coverage percentage of each one: So we stick with the third one

Method	**Coverage Percentage (%)**
Using food description algorithm	52.96
Searching for food description keywords	56.66
Combining both methods	70.00

Table 2: Coverage Percentage for the three methods tried in our experiments, which are based on keywords extraction

which got the best results.

4.1.2 Aspect-Based Sentiment Analysis

Performance Improvement: Fine-tuning improved the *Aspect-Level Accuracy* of the DeBERT-a model from 82.5% to 89.7%, reflecting a performance gain of approximately **7.2%**.

The fine-tuned model predicts sentiments (positive, neutral, or negative) for each aspect in the input text. The sentiment distribution in the dataset after analysis was as follows:

- Neutral: 23%
- Positive: 46%
- Negative: 31%

We tried two different methods. The first one was using all three logits and the second one using only positive and negative. Due to the relatively small proportion of neutral sentiments, they were excluded from further analysis. This simplification allowed for a more focused and effective recommendation process. The one who used all three logits get an accuracy of 52.25%. Instead the one who only used the positive and negative got 98.10%, which is a really great improvement compared to the other one.

5 Related Work

5.1 Delighting Palates with AI

The cited paper presents a method combining reinforcement learning and collaborative filtering to create personalized meal plans that meet users' nutritional needs and preferences, aiming to enhance long-term adherence. However, it does not address exercise planning, which limits its effectiveness for users seeking both dietary and physical activity recommendations to meet health goals.

5.2 AI Meal Planner App

This work generates meal plans suggestion based on a user's age,height, gender. But it does not consider about exercise and users taste.

5.3 Fitness Trackers with Diet Suggestions

Several fitness applications, such as Fitbit and MyFitnessPal, integrate exercise tracking and basic diet suggestions. However, they typically rely on predefined databases of meals and workouts, lacking the level of personalization needed to truly cater to individual user preferences. While effective at helping users track activities and caloric intake, they do not generate personalized recommendations based on taste, dietary restrictions, or the user's specific fitness goals. The focus on generic data limits the flexibility needed for sustained user engagement.

6 Conclusion

6.1 Result

With FLEX, we planned to reflect user's food preference while also considering user's health. As expected, the app provides both food recommendation and slightly personalized workout plan.

6.2 Limitations and Discussions

While our project shows promising results in providing personalized food and physical activity recommendations, several limitations must be addressed to enhance its effectiveness and scalability.

6.2.1 Dataset Limitations

The primary limitation lies in the dataset's size and diversity. As it is manually curated, the process requires significant time and resources, resulting in a limited range of foods. However, ongoing efforts to expand the dataset by including dishes from various cuisines and attributes will help align recommendations more closely with user preferences.

6.2.2 Dependency on GPT-Generated Data

The dataset relies on GPT-based models for generating descriptive attributes such as keywords and characteristics. While GPT accelerates the creation process, it may produce generic or incomplete outputs that require refinement. To mitigate this, integrating external data sources like culinary databases, nutritional APIs, and expert insights is recommended to complement GPT-generated content and improve accuracy.

6.2.3 Future Enhancements

Adding detailed attributes to the dataset, such as preparation time, nutritional breakdown, cooking methods, regional variations, and user feedback, will further enhance the system's performance. These expansions can make the recommendations more precise and user-aligned.

6.2.4 Scalability and Recommendations

Expanding the dataset iteratively will improve personalization and user satisfaction over time. Adopting a hybrid approach by combining GPT insights with verified external data and real-time user feedback will further enhance the system's scalability and accuracy.

In conclusion, addressing these limitations through dataset enrichment and hybrid methodologies will enable the system to deliver more accurate and effective recommendations, empowering users to adopt healthier lifestyle habits.

References

[1] Author, F.: Article title. Journal **2**(5), 99–110 (2016)

[2] Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890

[3] Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)

[4] Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)

[5] LNCS Homepage, `http://www.springer.com/lncs`, last accessed 2023/10/25