Journal of Informatics and Web Engineering

Vol. 4 No. 3 (October 2025) eISSN: 2821-370X

Hybrid Filtering for Personalized and Health-Conscious Recipe Recommendations in UniEats

Hui Lek Liew[1], XinYing Chew[2*], Khai Wah Khaw[3], Shiuh Tong Lim[4]

[1,2]School of Computer Science, Universiti Sains Malaysia, 11800, Gelugor, Penang, Malaysia

[3,4]School of Management, Universiti Sains Malaysia, 11800, Gelugor, Penang, Malaysia

**corresponding author: (xinying@usm.my; ORCiD: 0000-0001-5539-1959)*

Abstract - This research paper introduces UniEats, a recipe recommendation website designed to help university students better organize their meals and adopt healthier eating habits. The system aims to address common challenges students face, including time constraints, limited cooking skills, and insufficient nutritional awareness, which often lead to unhealthy food choices. At the core of UniEats is its recommendation engine, which employs a hybrid filtering approach, combining content-based and collaborative filtering techniques to provide personalized recipe suggestions based on users' dietary preferences, rating history, and recipe attributes. UniEats offers a range of features, including recipe search, weekly meal planning, nutritional analysis through dashboards, and automatic grocery list generation. By enabling students to explore diverse culinary options, create balanced meal plans, and understand the nutritional content of their meals, UniEats empowers them to make informed dietary decisions. This research paper discusses the project's background, motivation, and objectives, emphasizing the importance of addressing students' dietary challenges. It also reviews existing recommendation systems and algorithms, justifying the choice of hybrid filtering for personalized meal suggestions. Additionally, the research paper details the system's design, implementation, and testing procedures, highlighting the development process. UniEats is a practical solution that leverages machine learning and data-driven methods to enhance students' culinary experiences, support skill development, and promote nutritional awareness. By tackling key challenges in meal planning and healthy eating, UniEats aims to improve the overall well-being of university students.

Keywords— Collaborative Filtering, Content-Based Filtering, Hybrid Filtering, Machine Learning, Nutritional Awareness, Personalized Meal Planning, Recipe Recommendation.

Received: 05 February 2025; Accepted: 25 May 2025; Published: 16 October 2025

1. INTRODUCTION

Nowadays, university students are often busy with classes, extracurricular activities, and assignments, leaving them with limited time for meal planning. According to a study from California State University, students today are increasingly making poor health decisions [1]. Many young adults also lack sufficient knowledge about nutrition to make informed food choices, which may lead to long-term health issues [2].

Many existing applications fail to fully consider users' dietary preferences, resulting in generalized recommendations that do not align with individual needs. Personalized recipe recommendations tailored to each user's specific dietary and nutritional requirements are crucial for improving university students' overall well-being. Additionally, traditional meal-planning apps primarily focus on recipe ideas without providing clear insights into their nutritional composition. Users often lack knowledge of specific nutritional values, making it difficult to align food choices with their personal dietary needs.

To address these challenges, a technical solution is needed to recommend nutritious recipes based on students' dietary preferences. UniEats is a website designed to help students make informed dietary decisions for meal planning. It not only saves time but also provides a detailed breakdown of meal nutritional content. By offering an easy-to-use meal-planning tool, students can focus on their studies and extracurricular activities without the added stress of food preparation. This, in turn, enhances their overall well-being and productivity.

Studies have found that one of the most common reasons adults, particularly young adults, struggle to maintain a healthy diet is lack of time [3]. Research indicates that time-related behaviours, such as multitasking while eating and frequent fast-food consumption, are linked to poor dietary intake. In other words, students experiencing higher levels of stress tend to engage in less healthy eating habits [4]. This application, with its user-tailored recipe recommendations and meal planning functions, provides a resource-efficient solution to help busy university students better manage their meals.

Furthermore, frequently eating out increases the risk of weight gain and obesity [5]. According to the 2020-2025 Dietary Guidelines for Americans, culinary skills are essential for meeting dietary recommendations [6]. Research has shown that culinary skills are linked to healthier eating habits [7]. However, many university students lack adequate cooking skills, making them more prone to unhealthy eating behaviours. By incorporating recipes and cooking instructions, UniEats has the potential to enhance students' culinary abilities, ultimately fostering healthier eating habits.

Moreover, many university students lack nutritional knowledge, especially those living away from home for the first time. This limited understanding of nutrition can lead to poor dietary choices, resulting in unbalanced diets that may negatively impact their overall health. Studies have demonstrated a strong correlation between nutritional knowledge and healthy eating habits [8]. A lack of understanding about nutrition is often a key factor behind unhealthy eating pattern. Some research suggests that improving nutritional awareness can help students make better food choices [9]. Thus, the nutritional analysis feature in UniEats is essential for providing students with detailed insights into their meals, helping them select healthier options while still meeting their dietary needs.

UniEats, short form of University Student Recipe Recommender System for Meal Planning, is a web-based recipe recommendation system designed to support university students with meal planning through four key modules:

- **Recipe Recommendations** – Uses content-based filtering to analyse recipe characteristics based on users' dietary preferences. Hybrid filtering further refines recommendations by incorporating user ratings to suggest recipes preferred by others with similar tastes.
- **Weekly Meal Planning** – Allows users to create customized meal plans from personalized recommendations.
- **Recipe and Nutritional Analysis Dashboard** – Utilizes Exploratory Data Analysis (EDA) to provide insights into recipe ratings, popularity, and nutritional content.
- **Grocery Shopping List** – Automatically generates organized shopping lists, adding ingredients from planned meals to streamline grocery shopping, save time, and encourage healthier eating habits.

Together, these modules create a comprehensive and user-friendly meal-planning experience, tailored to the unique needs of university students.

2. BACKGROUND AND RELATED WORKS

2.1 Existing Systems

There are a few existing systems for recipe recommendation and meal planning which are compatible with UniEats. First is the Samsung Food, which is a mobile application that enables users to store, create, and share meal plans, and get meal nutritional details. Similarly, as UniEats, this application also creates shopping lists based on the recipe chosen. The advantage of Samsung Food is that users may interact with a community to share recipe ideas and reviews. The rating system of Samsung Food is different from UniEats, whereby Samsung Food utilises a like/dislike method, while UniEats employs a rating system from 1 to 5. Besides, Samsung Food is not equipped with a recipe data summaries dashboard, which differentiates UniEats as a more data-driven meal planning app that enables users to view the data summaries easily.

The second system is the Meal Manager, which is a free meal planner tool that allows users to plan daily meals, as well as view meal calendars from previous weeks. The advantages of Meal Manager are that it allows users to add their own recipe and share it with family or friends, establishing a sense of community. However, unlike UniEats, there is no automated grocery list in Meal Manager, whereby users need to insert the ingredients needed into the grocery list manually. Another disadvantage of Meal Manager is that it does not provide nutritional analysis, making it difficult for users to make informed dietary decisions based on nutritional content. Other than that, due to the absence of a recipe rating function in Meal Manager, users cannot look for famous or well-rated recipes in this application. So, UniEats outperforms Meal Manager in functionality with its automated grocery list, nutritional analysis, recipe rating tool, and recipe data summaries dashboard.

Thirdly, Meal Planner & Nutrition Coach (Feastr) is also a personalised meal planner and nutrition coach that focuses on making weight loss easier by eliminating the necessity for calorie tracking. By using this application, users may easily switch between diet categories and watch cooking tutorials. One special feature of Feastr is that it allows users to interact with an experienced nutrition coach in the app, who offers personalized advice and answers nutrition-related questions. Feastr also enables users to adjust macronutrients, aligning the recipe suggestions to their unique dietary requirements. However, similarly as Meal Manager, Feastr does not offer a comprehensive recipe data summaries dashboard and lacks a recipe rating feature. Hence, with the recipe data summaries dashboard and recipe rating features, UniEats provides a more data-centric approach, increasing the entire meal planning experience. Table 1 presents a comparison of existing systems with UniEats.

Table 1. Comparison of Existing Systems with UniEats

System / **Features**	**Samsung Food**	**Meal Manager**	**Feastr**	**UniEats**
Recipe Data Summaries Dashboard	No	No	No	Yes
Meal Planning	Yes	Yes	Yes	Yes
Cooking Instructions	Yes	Yes	Yes	Yes
Add Own Recipe	No	Yes	No	No
Automated Shopping List	Yes	No	Yes	Yes
Nutritional Analysis	Yes	No	Yes	Yes
Share Recipe with Friends and Family	Yes	Yes	Yes	No
Recipe Rating Feature	Like/Dislike	No	No	1-5
Chat with Nutrition Coach	No	No	Yes	No
Platform	Mobile App	Mobile App	Mobile App	Website

2.2. Existing Algorithms/Theories/Models

Recommendation systems are one of the most widely used approaches in machine learning, with applications in a wide range of fields. The commonly used algorithms are collaborative filtering, content-based filtering, hybrid filtering, clustering algorithms, and the deep learning-based recommender system.

Each recommendation system approach has its own set of advantages and disadvantages. Collaborative filtering benefits from user engagement but struggles with new users and items. While content-based filtering provides user independence, it may lead to over-specialization. Hybrid filtering combines the strengths of both methods but increases complexity. Clustering algorithms effectively address data sparsity but have difficulty adapting to dynamic changes. Deep learning-based systems can uncover complex patterns, yet they require large datasets and are challenging to interpret.

A comprehensive review of these algorithms is essential for the development of UniEats. As UniEats aims to provide a seamless and personalized meal-planning experience, selecting the right recommendation algorithm will significantly impact the application's performance and user satisfaction. Table 2 summarizes the advantages and disadvantages of the recommendation systems.

Table 2. Summary of Advantages and Disadvantages of the Recommendation Systems

Algorithms	Advantages	Disadvantages
Collaborative Filtering	• Relies purely on user behavior data, no extra user/item attributes are required. • In general, it makes accurate and relevant recommendations based on wisdom. • Easy to continually update with fresh data, and quality improves over time.	• Has a cold start issue for new users and items, causing difficulty in making suitable recommendations for new entities. • Performance is limited by the extreme sparsity of user-item matrix data.
Content-Based Filtering	• Recommendations based on the interests/profile of the specific user. • Cold start problem for new items is addressed by depending on item characteristics. • Able to provide explanations by relating recommendations to item features.	• Overspecialization causes inability to broaden the user's exploration beyond recognized interests. • Domain item knowledge is required to create useful item attributes. • Cold start issue persists for new users.
Hybrid Filtering	• Improves accuracy by leveraging the advantages of multiple approaches. • Cold starts issue and data sparsity are addressed.	• Increased system complexity to combine various components.
Clustering Algorithms	• Users are grouped to solve scalability and sparsity problems. • Reduce computational complexity by focusing on cluster members.	• Difficult to adapt to changing user preferences over time. • Clusters are difficult to update incrementally over time.
Deep-Learning Based Recommender System (Auto Encoder-Based Collaborative Filtering)	• Learn complicated patterns for meaningful representations in large datasets. • Performs in both unsupervised and supervised settings.	• Effective training needs an enormous amount of interaction data. • Lack of interpretability for individual recommendations due to complexity of neural networks.

Based on the comparison of various recommendation systems methodologies, content-based filtering and the hybrid filtering method (content-based filtering and collaborative filtering) appear to be the best fit for meeting UniEats' primary needs and desired capabilities. In implementing hybrid filtering for the recipe recommendation module, users begin by entering dietary preferences, including cuisine and meal category (e.g., main dish, dessert, soup, appetizers, and snacks). Content-based filtering initially recommends recipes based on these inputs, addressing the cold start issue since users have yet to provide ratings. Additionally, when a user searches for or clicks on a recipe, similar recipes are suggested on the recipe details page. Content-based filtering effectively captures specific user preferences, creating a tailored experience. Users can also rate recipes on UniEats, enabling hybrid filtering to enhance recommendations by combining user preferences with collaborative filtering, which considers the choices of users with similar tastes. As users interact more, personalized recommendations improve, better aligning with their evolving interests.

2.2.1 Content-based Filtering

Content-based recommendation systems function differently from collaborative filtering methods. Instead of relying solely on past user interactions, content-based systems explicitly define features to characterize both items and user preferences. While collaborative filtering derives recommendations from population patterns, content-based filtering generates suggestions by comparing items to a user's specific characteristics.

The algorithm consists of three major steps [10]:

a. Creating informative features to represent items within the domain.
b. Building user profiles that capture individual preferences based on these features.
c. Measuring item similarity against previous user selections to generate personalized recommendations.

A key advantage of content-based systems is their user independence—recommendations are tailored based on an individual's preferences rather than crowd behaviour. This approach enhances scalability and improves explainability, as recommendations can be directly linked to specific attributes a user prefers. However, a potential drawback is overspecialization. Since content-based filtering only analyses similarities to past choices, it may restrict users to their existing interests, limiting exposure to new or diverse recommendations.

2.2.2 Collaborative Filtering

Collaborative filtering works by analysing past interactions between users and items to identify patterns and similarities that can be used to generate recommendations. These interactions—such as purchases, ratings, clicks, and likes—are stored in a user-item interaction matrix, a large and sparse database that quantifies user activity across the catalogue. While the matrix captures the entire interaction history, it is highly sparse since most users engage with only a small fraction of available items, leaving many cells with missing values. To address this challenge, collaborative filtering relies on two main approaches: memory-based and model-based filtering.

Memory-based collaborative filtering generates recommendations directly from the raw interaction matrix by identifying commonalities among users or items. Meanwhile, model-based collaborative filtering mitigates scalability issues by training models on interaction data. It maps users and items to a lower-dimensional latent factor space, with matrix factorization being a popular technique for decomposing the large interaction matrix into compact user and item representations.

A key advantage of collaborative filtering is that it does not require explicit attribute data about users or items; recommendations are purely based on interaction patterns. However, it struggles with the "cold start" problem, where new users or items lack sufficient interaction data, limiting the system's ability to generate relevant recommendations.

2.2.3 Hybrid Filtering

Hybrid recommendation systems combine different methodologies to enhance accuracy. Integration techniques include generating collaborative filtering and content-based predictions separately and merging them, embedding content-based methods into a collaborative filtering model, or unifying both into a single model [11], [12], [13].

Empirical studies show that hybrid techniques outperform standalone approaches, offering more accurate and well-rounded recommendations. They also mitigate issues like cold starts for new users and items or sparse interaction data, as one method's strengths can compensate for another's weaknesses [14], [15], [16].

3. RESEARCH METHODOLOGY

Figure 1 illustrates the system architecture of UniEats, which follows a client-server model with a clear separation between frontend and backend components [17], [18], [19]. The frontend consists of a Svelte-based website, a modern JavaScript framework known for its efficiency and simplicity. This serves as the primary interface for user interactions, primarily through a web application. Additionally, the frontend integrates Power BI, a powerful data visualization tool, to present insights related to recipes.

The backend is built around an API server, which manages requests from the frontend and coordinates interactions with other system components. Supabase, an open-source backend-as-a-service platform, provides a PostgreSQL database for data storage [20], [21]. At the core of the system is a machine learning model designed to process recipe data and user preferences to generate personalized recommendations. This model is deployed using FastAPI, a high-performance web framework for building APIs in Python. A Python script preprocesses the data to ensure it is clean and well-structured for training and inference.

In summary, this architecture integrates modern technologies and frameworks to create a robust and flexible recipe recommendation system. By leveraging machine learning and cloud services, UniEats aims to deliver a personalized and engaging user experience.

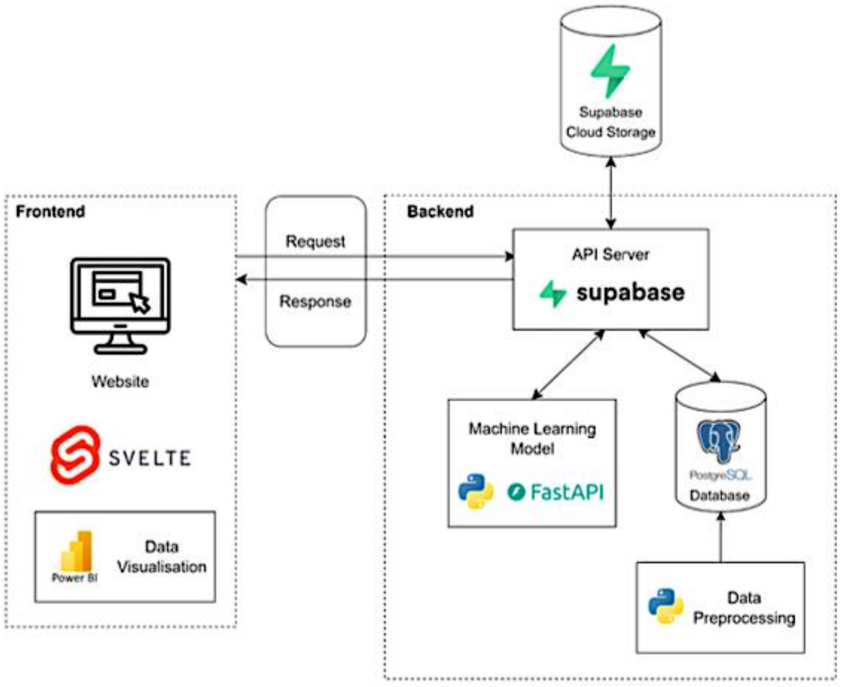

Figure 1. System Architecture Diagram of UniEats

Meanwhile, Figure 2 presents the process flow diagram of the UniEats recipe recommendation system, which leverages user interactions and preferences to generate personalized recipe suggestions. For first-time users, the system prompts them to select their preferred cuisine and category (e.g., main dish, dessert, soup, etc.). These preferences are then used in a content-based filtering approach, executed by the *recommend_by_cuisine_category()* function, to generate initial recommendations. The suggested recipes appear on the 'Browse Recipe' page, allowing users to explore recipes that align with their tastes.

If a user has not submitted any ratings, the system monitors their interactions with recipes. When a user searches for or clicks on a recipe, the *recommend_by_recipe()* function is triggered. This function applies content-based filtering to identify similar recipes, which are then displayed on the 'Recipe Details' page. This approach encourages users to discover related recipes and expand their culinary interests. For users who have provided ratings, a hybrid filtering method is applied through the *hybrid_recommender()* function. This method combines user preferences and past rating data to generate more tailored recommendations, which are displayed on the 'Browse Recipe' page. If a user has not submitted any ratings and does not interact with specific recipes, the system provides recommendations solely based on their initial cuisine and category preferences.

Overall, by integrating content-based and hybrid filtering methods, the UniEats system delivers a personalized recipe discovery experience, helping users find recipes that match their tastes and preferences.

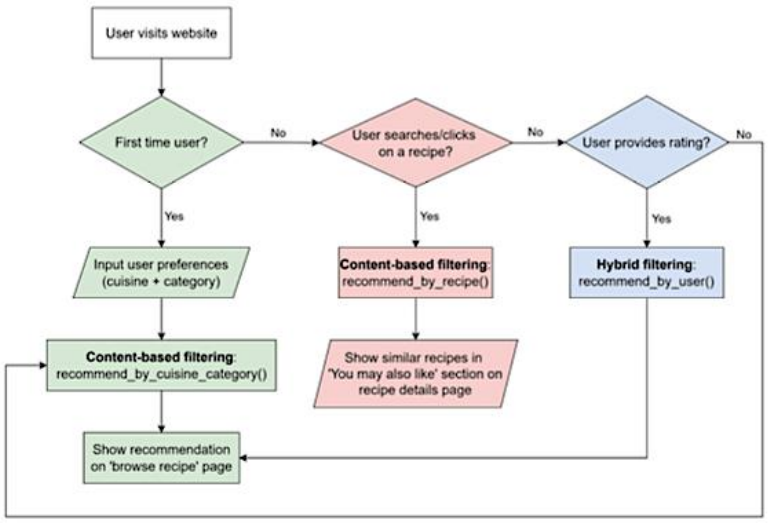
User visits website
First time user?
No
User searches/clicks on a recipe?
No
User provides rating?
No
Yes
Yes
Yes
Input user preferences (cuisine + category)
Content-based filtering: recommend_by_recipe()
Hybrid filtering: recommend_by_user()
Content-based filtering: recommend_by_cuisine_category()
Show similar recipes in 'You may also like' section on recipe details page
Show recommendation on 'browse recipe' page

Figure 2. Process Flow Diagram of Recipe Recommendation System

3.1 Content-based Filtering

In UniEats, content-based filtering is utilised in two main situations: suggesting similar recipes when a user search for/clicks on a specific recipe and offering recommendations when a user has not provided any rating yet (referred to as the cold start issue).

3.1.1 Recommending Similar Recipes

The *recommend_by_recipe* function employs content-based filtering to suggest recipes like a specific recipe searched or viewed by the user. The recommendation process follows these steps:

a. The function takes a *RecipeRecommendationRequest* object, which includes the recipe ID (*recipe_id*) of the selected recipe and the desired number of recommendations (*num_recipes*).
b. The content-based features of the selected recipe—such as its TF-IDF vector representation of the textual description and one-hot encoded cuisine and category—are extracted from the *content_based_features* matrix.
c. Using cosine similarity, the system calculates similarity scores between the selected recipe and all other recipes in the dataset.
d. The top N most similar recipes (excluding the input recipe) are selected based on the highest similarity scores, where N is determined by the *num_recipes* parameter.
e. The function returns details of the recommended recipes, including recipe ID, name, cuisine, and category.

3.1.2 Offering Recommendations

The *recommend_by_cuisine_category* function employs content-based filtering to generate recommendations for users who have not yet provided ratings, addressing the cold start problem. Since no user rating data is available, collaborative filtering and hybrid filtering cannot be applied in this scenario. The recommendation process follows these steps:

a. The function takes a *CuisineCategoryRecommendationRequest* object, which includes:
   - A list of the user's preferred cuisines (e.g., [“American”, “Italian”]).
   - A list of preferred categories (e.g., [“Main dish”, “Dessert”]).
   - The desired number of recommendations (*num_recipes*).

b. The system filters the recipe dataset to include only recipes that match at least one of the specified cuisines or categories.
c. It then computes content-based similarity scores using the *get_content_based_scores* function, which applies cosine similarity between the content-based features of the filtered recipes and the entire dataset.
d. The top N recipes with the highest similarity scores are selected, where N is determined by the *num_recipes* parameter.
e. The response includes details of the recommended recipes, such as recipe ID, name, cuisine, and category.

In these cases, the recommendation system provides personalized suggestions without relying on explicit user ratings. By analysing textual descriptions, cuisines, and categories, the system identifies similar recipes, ensuring a diverse and engaging recommendation experience. Additionally, by utilizing structured request objects like *RecipeRecommendationRequest* and *CuisineCategoryRecommendationRequest*, the system efficiently processes incoming requests in a standardized manner. These objects ensure that all essential parameters are included, allowing the system to generate accurate recommendations based on the appropriate input data.

3.2 Hybrid Filtering

The recipe recommendation system also employs a hybrid filtering method, combining the strengths of content-based filtering and collaborative filtering. This approach generates recommendations based on user-provided ratings, ensuring more personalized and accurate suggestions.

3.2.1 Collaborative Filtering

This approach leverages the Surprise library, specifically the Singular Value Decomposition (SVD) algorithm, to perform collaborative filtering. The process follows these steps:

a. **Retrieve and preprocess data** – User rating data is retrieved from the database and transformed into a format suitable for the Surprise library.
b. **Define the rating scale** – The Reader class is used to define the rating scale, which ranges from 1 to 5.
c. **Load the dataset** – The Dataset class imports the rating data, converting it into a structure that can be used by the Surprise library.
d. **Split the data** – The dataset is divided into a training set and a test set using the *train_test_split* function.
e. **Train the SVD model** – The SVD algorithm is trained on the training set using the SVD class from the Surprise library.

Once trained, the SVD model can predict ratings for specific user-recipe combinations, enabling collaborative filtering. This method leverages similar users' preferences to generate recommendations based on their rating behaviors.

3.2.2 Hybrid Filtering

The hybrid filtering approach combines content-based filtering and collaborative filtering to deliver more accurate and personalized recommendations. The hybrid_recommender function implements this combined approach, following these steps:

a. The function takes a user ID and the desired number of recommendations as input.
b. It retrieves the user's ratings from the *ratings* DataFrame.
c. If the user has not provided any ratings, it returns an empty list (no recommendations).
d. Otherwise, it identifies the valid recipe IDs the user has interacted with.
e. The *get_content_based_scores* function calculates content-based scores by computing cosine similarity between the valid recipes and the entire dataset.
f. The *get_collaborative_scores* function predicts collaborative scores for the user using the trained SVD model.
g. The hybrid score is computed as the average of the content-based and collaborative scores.
h. The top N recipes with the highest hybrid scores are selected as recommendations, where N is determined by the *num_recipes* parameter.

i. The function returns details of the recommended recipes, including recipe ID, name, cuisine, and category.

The recommend_by_user endpoint utilizes the hybrid filtering approach to generate personalized recipe recommendations based on a user's preferences and interactions. The process follows these steps:

a. The endpoint receives a *UserRecommendationRequest* object containing the User ID (*user_id*) and the Number of recommendations *(num_recipes*)
b. It calls the *hybrid_recommender* function with the provided user ID and recommendation count.
c. If the *hybrid_recommender* function returns a non-empty list of recommended recipe IDs, the system retrieves the corresponding recipe details (ID, name, cuisine, and category) from the *recipes* DataFrame.
d. The system returns a "success" status along with the top N recommended recipes for the specified user ID, including their details.

By integrating content-based filtering (which considers recipe features) and collaborative filtering (which analyses similar user preferences), this hybrid approach overcomes the limitations of individual filtering methods. It enhances recommendation accuracy and ensures a more personalized and engaging user experience.

3.3 Application Setup and Configuration

This section includes the technical details, which includes the pseudocode of the steps performed to setup and configure the recipe recommendation system. These actions guarantee that our application is initialised correctly and able to retrieve and handle the required data for offering recipe recommendations.

```
Initialize Environment Variables
    Load .env file
    Set SUPABASE_URL and SUPABASE_KEY from environment variables
Initialize FastAPI
    Create FastAPI app instance
    Configure CORS middleware with allowed origins
Supabase Configuration
    Create Supabase client with SUPABASE_URL and SUPABASE_KEY
Function: load_data()
    Load recipes from Supabase table "RECIPE"
    Convert recipes to pandas DataFrame
    Return recipes DataFrame
```

3.3.1 Content-based Filtering

```
Function: preprocess_for_content_based()
    Call load_data() to get recipes DataFrame
    Content-Based Filtering
        Initialize TfidfVectorizer with English stop words
        Fit and transform recipes['description'] with TfidfVectorizer
        Initialize OneHotEncoder for cuisines and categories
        Fit and transform recipes['cuisine'] with OneHotEncoder
        Fit and transform recipes['category'] with OneHotEncoder
        Combine TF-IDF matrix, cuisine encoded, and category encoded features
    Return recipes, content-based features
Function: get_content_based_scores(recipe_ids)
        Find valid indices that intersect with recipe_ids and content-based features
        If no valid indices, return zero scores
        Calculate cosine similarity between valid indices and all content-based features
        Return average content-based scores
API Endpoint: recommend_by_recipe(request)
    Extract recipe_id and num_recipes from request
    Find index of recipe_id in recipes DataFrame
```

```
If recipe_id found:
        Calculate content-based scores for recipe_id
        Sort and get top num_recipes indices excluding input recipe_id
        Prepare recipe details from recommended recipe_ids
        Return success response with recipe details
    Else:
        Return not found response with appropriate message
API Endpoint: recommend_by_cuisine_category(request)
    Extract cuisines, categories, and num_recipes from request
    Filter recipes DataFrame by cuisines and categories
    If filtered recipes found:
        Get indices of filtered recipes
        Calculate content-based scores for filtered indices
        Sort and get top num_recipes indices
        Prepare recipe details from recommended recipe_ids
        Return success response with recipe details
    Else:
        Return not found response with appropriate message
```

3.3.2 Hybrid Filtering

```
Function: preprocess_and_train()
    Call load_data() to get recipes and ratings DataFrames
    Perform content-based filtering (as detailed in the previous section)
        Collaborative Filtering
        Initialize Reader with rating scale (1, 5)
        Load data from ratings DataFrame using Dataset.load_from_df
        Split data into train and test sets using train_test_split
        Initialize and train SVD model with train set
        Return recipes, ratings, content-based features, content-based model, collaborative model
Function: get_collaborative_scores(user_id)
    Initialise empty list for collaborative scores
    For each recipe_id in recipes DataFrame:
        Predict rating for user_id and recipe_id using collaborative model
        Append predicted rating to collaborative scores list
    Return collaborative scores
Function: hybrid_recommender(user_id, num_recipes=10)
    Get user interactions from ratings DataFrame where user_id matches
    If user interactions are empty, return empty list
        Find valid recipe_ids that intersect with user interactions and recipes DataFrame
    If no valid recipe_ids, return empty list
        Get indices of valid recipe_ids in recipes DataFrame
Calculate Scores
    Get content-based scores for valid recipe indices using
    get_content_based_scores
        Get collaborative scores for user_id using get_collaborative_scores
        Calculate hybrid scores by averaging content-based and collaborative scores
Generate Recommendations
    Sort hybrid scores and get top num_recipes indices
    Extract top recipes based on sorted indices
    Return list of top recipe_ids
API Endpoint: recommend_by_user(request)
    Extract user_id and num_recipes from request
    Get recommended recipes using hybrid_recommender(user_id, num_recipes)
    If recommended recipes are found:
```

```
    Prepare recipe details from recommended recipe_ids
    Return success response with recipe details
Else:
    Check if the user exists in the ratings DataFrame
    If user does not exist, return not found response with appropriate message
    If user exists but no recommendations found, return not found response with appropriate message
```

4. RESULTS AND DISCUSSIONS

The fundamental dataset for UniEats is derived from Kaggle, containing essential recipe data. This dataset includes key attributes such as recipe names, ingredients, and directions, along with user interaction data, specifically user ratings for each meal. The raw dataset comprises approximately 180,000 recipes and 700,000 recipe reviews sourced from Food.com.

To evaluate the performance of the recipe recommender system, various subsets of the dataset were tested, ranging from 20% to 100% of the total dataset size. The evaluation primarily focused on two key metrics: Precision and F1-score. Precision measures the accuracy of the recommended recipes by assessing how well they align with a user's preferences. It calculates the proportion of recommended recipes that are relevant to the user. Recall, in contrast, evaluates the system's ability to identify all relevant recipes that a user might enjoy, regardless of whether they were included in the recommendations. F1-score is a harmonic mean of precision and recall, offering a balanced measure of both. It is particularly useful when both accuracy and completeness of recommendations are important.

In this study, precision was prioritized as it reflects the percentage of recommended recipes that truly match user preferences. Recall was omitted since capturing all possible relevant items was considered less critical than ensuring that the recommended items were highly relevant to the user. The evaluation results of the recipe recommendation system are presented in Table 3.

Table 3. Evaluation of Recipe Recommendation System with Different Dataset Sizes

Portion of Dataset	Recipe Dataset Size (Rows)	Precision	F1-score
20%	2318	0.8742	0.8776
50%	5938	0.9087	0.9034
75%	8908	0.9135	0.9056
100%	11877	0.9143	0.9076

With an increase in dataset size, both Precision and F1-score show noticeable improvements. This suggests that the hybrid filtering method becomes more effective as it gains access to more data, allowing it to provide more accurate and relevant recommendations.

The final version of the website was implemented using only 20% (2,318 rows) of the original dataset. Despite the smaller dataset size, the evaluation yielded a Precision of 0.8742 and an F1-score of 0.8776, which are considered strong performance metrics. The decision to use a reduced dataset was driven by the need to optimize data loading speed, ensuring a smooth user experience with efficient website rendering and functionality.

It is important to note that the evaluation results presented in Table 3 are based solely on the collaborative filtering model. The table does not compare performance metrics across content-based filtering, collaborative filtering, and hybrid filtering. This omission is due to the lack of predefined recommended recipes in the dataset, which are essential for establishing ground truth recommendations. Access to ground truth data is necessary to accurately compare model predictions and compute evaluation metrics for content-based filtering.

Although content-based filtering lacks formal evaluation results, it was still integrated into the recipe recommendation system. This technique plays a crucial role in addressing the cold start problem, which occurs when new users have not provided any ratings. Since collaborative filtering relies on user ratings, it becomes ineffective in such cases, making content-based filtering indispensable.

Ultimately, the hybrid filtering approach, which combines content-based and collaborative filtering, was chosen overusing collaborative filtering alone. This strategy leverages the strengths of both methods, ensuring more personalized and precise recommendations even in the presence of the cold start issue. By incorporating user preferences, past interactions, and recipe attributes, the hybrid model delivers a comprehensive and well-rounded recommendation experience.

5. CONCLUSION

In conclusion, this work provides an in-depth exploration of recommendation systems, analysing various algorithms and methodologies to establish the foundation for UniEats. The project background outlined the current landscape of meal recommendation platforms, comparing UniEats to similar systems such as Samsung Food, Meal Manager, and Feastr. A comprehensive review of collaborative filtering, content-based filtering, hybrid filtering, clustering algorithms, and deep learning-based approaches offered valuable insights into their strengths and limitations. To enhance the accuracy and personalization of meal recommendations, UniEats employs a hybrid filtering method, combining content-based and collaborative filtering techniques. The system is designed using a client-server model, where the front-end is built with the Svelte framework, while the back-end is powered by Python's FastAPI and Supabase database infrastructure. The evaluation results demonstrated that the hybrid filtering method performs effectively, achieving high precision and F1-score. In essence, UniEats is a web-based system developed to assist university students in meal planning and promoting nutritional awareness. By leveraging a hybrid filtering approach, it delivers personalized recipe recommendations based on users' dietary preferences and rating history. UniEats aims to help students overcome the challenges of maintaining a healthy diet by providing a user-friendly interface with comprehensive features. Through tailored recipe suggestions, weekly meal planning, and nutritional analysis, the system empowers students to make well-informed dietary choices, contributing to their overall well-being.

While UniEats has achieved its primary objectives, there are several opportunities for future enhancement. Firstly, incorporating real-time collaboration among users could enable shared meal planning, fostering a community-driven experience. Additionally, allowing users to upload and share their recipes could enrich the platform's content and encourage greater user engagement. Furthermore, integrating a feature that provides personalized nutritional recommendations based on individual health goals or dietary restrictions could further enhance the system's utility. By exploring these potential advancements, UniEats can continue to evolve, addressing the needs of university students while promoting healthy eating habits, culinary exploration, and overall wellness. Finally, while collaborative filtering offers improved personalization, it inherently depends on user interaction data such as ratings and clicks. As such, it is important to ensure user privacy through anonymization techniques and to obtain explicit user consent for data collection and use. Future implementations of UniEats will incorporate mechanisms to inform users and allow them to manage their data-sharing preferences.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable comments.

FUNDING STATEMENT

This work is funded by Ministry of Higher Education Malaysia, Fundamental Research Grant Scheme [Grant Number: FRGS/1/2022/STG06/USM/02/4], for the Project entitled "Efficient Joint Process Monitoring using a New Robust Variable Sample Size and Sampling Interval Run Sum Scheme".

AUTHOR CONTRIBUTIONS

Hui Lek Liew: Conceptualization, Data Curation;
XinYing Chew: Methodology, Validation;
Khai Wah Khaw: Writing – Original Draft Preparation;
Shiuh Tong Lim: Writing – Review & Editing; Project Administration, Supervision, Writing – Review & Editing.

CONFLICT OF INTERESTS

No conflict of interests.

ETHICS STATEMENTS

Our publication ethics follow The Committee of Publication Ethics (COPE) guideline. https://publicationethics.org/

REFERENCES

[1] P.A. Chavan, "FeedMeRight-Recipe Recommendation System," M.S. thesis, California State Univ., Monterey Bay, CA, USA, 2020. [Online]. Available: https://scholarworks.calstate.edu/concern/theses/pg15bg98f.

[2] J. Garcia, "The Lack of Nutrition Education among Student Athletes," Capstone Project, California State Univ., Monterey Bay, CA, USA, 2018. [Online]. Available: https://digitalcommons.csumb.edu/caps_thes_all.

[3] J.E. Pelletier, and M.N. Laska, "Balancing healthy meals and busy lives: Associations between work, school, and family responsibilities and perceived time constraints among young adults," *J. Nutr. Educ. Behav.*, vol. 44, no. 6, pp. 481–489, Nov. 2012, doi: 10.1016/j.jneb.2012.04.001.

[4] M. Telfer, "An analysis of stress and dietary intake among university students," *J. Amer. Coll. Health*, vol. 69, no. 5, pp. 556–560, 2021, doi: 10.1080/07448481.2019.1686005.

[5] News-Medical.net, "Are inadequate culinary skills among undergraduate students associated with obesity and overweight?," May 26, 2023. [Online]. Available: https://www.news-medical.net/news/20230526/Are-inadequate-culinary-skills-among-undergraduate-students-associated-with-obesity-and-overweight.aspx. [Accessed: Aug. 25, 2025].

[6] L.G. Snetselaar *et al.*, "Dietary guidelines for Americans, 2020–2025," *Nutr. Today*, vol. 56, no. 6, pp. 287–295, Nov. 2021, doi: 10.1097/NT.0000000000000512.

[7] S. Mills, S. Brown, W. Wrieden, M. White, and J. Adams, "Frequency of eating home cooked meals and potential benefits for diet and health: Cross-sectional analysis of a population-based cohort study," *Int. J. Behav. Nutr. Phys. Act.*, vol. 14, no. 1, p. 109, Aug. 2017, doi: 10.1186/s12966-017-0567-y.

[8] J. Wardle, K. Parmenter, and J. Waller, "Nutrition knowledge and food intake," *Appetite*, vol. 34, no. 3, pp. 269–275, Jun. 2000, doi: 10.1006/appe.1999.0311.

[9] W. Husain, F. Ashkanani, and M.A. Al-Dwairji, "Nutrition knowledge among college of basic education students in Kuwait: A cross-sectional study," *J. Nutr. Metab*, Art. no. 5560714, 2021, doi: 10.1155/2021/5560714.

[10] I.B. Bobadilla, and M.K. Krasnorylov, "A review of clustering-based recommender systems," *IEEE Access*, vol. 9, pp. 145, 2021, doi: 10.1109/ACCESS.2021.3131265.

[11] G.M.P, "Research on recommendation systems using deep learning models," *Int. J. Rec. Technol. Eng.*, vol. 8, no. 4, pp. 456–460, Jan. 2022, doi: 10.35940/ijrte.D4609.118419.

[12] J.A. Wolfson, and S.N. Bleich, "Is cooking at home associated with better diet quality or weight-loss intention?," *Public Health Nutr.*, vol. 18, no. 8, pp. 1397–1406, Jun. 2015, doi: 10.1017/S1368980014001943.

[13] F. Ricci, L. Rokach, and B. Shapira, “Introduction to recommender systems handbook,” in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Boston, MA, USA: Springer, 2011, pp. 1–35.

[14] M.J. Thornton, G.E. Adam, D.R. Black, and C.L. Adams, “The influence of time, motivation, and planning on meal preparation among university students,” *J. Nutr. Educ. Behav.*, vol. 49, no. 7, p. S1–S2, Jul. 2017, doi: 10.1016/j.jneb.2017.05.340.

[15] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using collaborative filtering to weave an information tapestry,” *Commun. ACM*, vol. 35, no. 12, pp. 61–70, Dec. 1992, doi: 10.1145/138859.138867.

[16] M.D. Abel, “A hybrid approach to recipe recommendation,” M.S. thesis, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, USA, 2018. [Online]. Available: https://scholar.colorado.edu/concern/graduate_thesis_or_dissertations/2n49t4443

[17] M.F. Harper, and J.A. Konstan, “The MovieLens datasets: History and context,” *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Dec. 2015, doi: 10.1145/2827872.

[18] A.K.M.N. Islam, M.M. Rahman, and S.S. Islam, “A survey on food recommendation systems,” in *Proc. 4th Int. Conf. Trends Electron. Informat. (ICOEI)*, Tirunelveli, India, pp. 728–733, 2020, doi: 10.1109/ICOEI48184.2020.9142988.

[19] L. Hong, Y. Fu, and J. Liu, “Health-aware food recommendation system,” in *Companion Proc. Web Conf. 2021*, Ljubljana, Slovenia, pp. 25–26, 2021, doi: 10.1145/3442442.3452050.

[20] A. Musto, C. Basile, P. Lops, and G. Semeraro, “Health-aware food recommendation: A model and a platform for the recommendation of recipes for a balanced diet,” *IEEE Trans. Emerg. Top. Comput.*, vol. 10, no. 2, pp. 1054–1067, Apr.-Jun. 2022, doi: 10.1109/TETC.2021.3139364.

[21] A. A. Freitas, T. R. Li, and J. T. Wang, “Recipe recommendation for a balanced diet: A multi-objective optimization approach,” *J. Amer. Med. Inform. Assoc.*, vol. 30, no. 4, pp. 718–728, Mar. 2023, doi: 10.1093/jamia/ocad015.

BIOGRAPHIES OF AUTHORS

	Hui Lek Liew is a Computer Science graduate with first-class honours (from the Universiti Sains Malaysia), specializing in Artificial Intelligence. Passionate about transforming complex data into actionable insights that drive both business success and sustainability. Hui Lek is particularly interested in leveraging data to support sustainable decision-making and drive meaningful impact. She is always eager to learn and innovate and aim to contribute to organizations that value responsible data practices and forward-thinking solutions. She can be contacted at email: huilekliew@student.usm.my.
	XinYing Chew is an Associate Professor at the School of Computer Sciences, Universiti Sains Malaysia. She holds a PhD. in statistical quality control from University Sains Malaysia. She is a certified trainer with Human Resources Development Fund (HRDF). She is also the trainer for MDEC-Intel AI Academy Program and SAP Next-Gen Program. She is the Professional Technologist with the Malaysia Board of Technologist (MBOT). Prior to her academic career, she worked in the Advanced Analytics research team of a renown U.S. multinational company. Her areas of research are in advanced analytics and statistical quality / process control. She can be contacted at email: xinying@usm.my.

	Khai Wah Khaw is an Associate Professor at the School of Management, Universiti Sains Malaysia. He holds a Ph.D. in statistical quality control from Universiti Sains Malaysia. He is a program chairperson of the Operations Management, Business Analytics and Marketing Sections in the School of Management, USM. His areas of research are in advanced analytics and statistical quality/process control. He has featured in prominent international publications. His efforts and excellence have been acknowledged and awarded at several dignified platforms. He is actively involved in conducting training in analytics, machine learning, statistics and data visualization. Prior to his academic career, he worked in a renowned U.S. multinational company as a Data Analytics Team Leader. He can be contacted at email: khaiwah@usm.my.
	Shiuh Tong Lim is pursuing her PhD Degree in the School of Management, Universiti Sains Malaysia. She has her Bachelor's focusing on Industrial Chemistry and Mathematics; and her Master's in Business Analytics. She is the research assistant with the lecturer in the School of Management, Universiti Sains Malaysia. Additionally, she was also a form 6 (pre-university) Chemistry teacher. She can be contacted at email: shiuhtong1997@gmail.com.