Tribhuvan University

Academia International College

Final Year Project Report

On

Diet-Food Recommendation System[CSC 412]

Under the supervision of

"Er.Rabin Maharjan"

Submitted by

Akriti Chhetri(T.U. Exam Roll No. 26476/077)

Dipesh Kumar Mandal(T.U. Exam Roll No. 26488/077)

Submitted to

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

January, 2025

Tribhuvan University

Academia International College

Final Year Project Report
On

Diet- Food Recommendation System

[CSC 412]

A final year project submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

Submitted by

Akriti Chhetri(T.U. Exam Roll No. 26476/077)

Dipesh Kumar Mandal(T.U. Exam Roll No. 26488/077)

Submitted to

Department of Computer Science and Information Technology Academia International College

Institute of Science and Technology

Tribhuvan University

January, 2025

Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Email: mail@academiacollege.edu.np

Supervisor's Recommendation

I hereby recommend that the project work report prepared under my supervision by Miss Akriti Chhetri (26476/077), and Dipesh Kumar mandal (26488/077) entitled "Diet-Food Recommendation System" be accepted as fulfilling in partial requirements for the degree of Bachelors of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology.

.....

Er. Rabin Maharjan

Project Supervisor

Department of Computer Science and Information Technology

Academia International College

Gwarko, Lalitpur

Tribhuvan University

Department of Computer Science and Information Technology

Academia International College

Letter of Approval

This is to certify that this project prepared by Miss.Akriti Chhetri, and Mr.Dipesh Kumar Mandal entitled "Diet-Food Recommendation System" in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

…….. Er.Rabin Maharjan Project Supervisor Department of Computer Science and IT Academia International College	…….. Mr. Bishwas Mathema HOD/Program Coordinator Department of Computer Science and IT Academia International College
…….. Internal Examiner Academia International College	…….. External Examiner Central Department of CSIT Tribhuvan University

Acknowledgement

We want to express our sincere appreciation to Academia International College for giving us the chance to complete this project as a requirement for our degree.

We are very grateful to Er. Rabin Maharjan, our supervisor from Academia International College, for his unwavering leadership, encouragement, and insightful criticism during the project. We are incredibly appreciative of him for providing this fantastic chance to improve our knowledge and abilities, which has greatly influenced the direction of our educational path.

We also wish to sincerely thank our professors, friends, coworkers, and families for their constant encouragement and support. Their helpful advice and support were invaluable in enabling us to finish this project in the allotted time.

Thanking You,

Akriti Chhetri(T.U. Exam Roll No. 26476/077)

Dipesh Kumar Mandal (T.U. Exam Roll No. 26488/077)

Abstract

The Diet-Food Recommendation System is a web-based program that generates individualized meal plans based on users' specific health objectives and dietary preferences. The system achieves great performance, scalability, and user-friendliness by utilizing contemporary technologies such as Streamlit for the frontend, FastAPI for the backend, and MongoDB for data storage. Users provide important information such as their age, gender, height, weight, weight reduction objectives, and preferred number of daily meals. The system generates tailored breakfast, lunch, and supper suggestions using the Nearest Neighbor algorithm and cosine similarity. Pie charts are used to visually display nutritional information, hence improving user comprehension and engagement.

With a library of over 20,000 recipes, the system provides broad and reliable recommendations while remaining simple and interactive. This project seeks to solve issues such as poor customization, restricted user interfaces, and static meal databases seen in previous solutions. The Diet-Food Recommendation System enables users to make educated dietary choices and accomplish long-term health goals by combining smart algorithms with an accessible design.

Keywords Diet Recommendation System, Streamlit, Python, FastApi, Nearest Neighbour Algorithmn, Nutritional Visualization

Table of Content

Supervisor's Recommendation ..... **III**
Letter of Approval ..... **IV**
Acknowledgement ..... **V**
Abstract ..... **VI**
Table of Content ..... **VII**
List of Figures ..... **IX**
List of Table ..... **X**
List of Abbreviation ..... **XI**
1. Chapter 1: Introduction **1**
1.1. Introduction 1
1.2. Problem Statement 1
1.3. Objectives 2
1.4. Scope and Limitations 2
1.5. Development Methodology 3
1.6. Report Organization 5
Chapter 2: Background Study and Literature Review **7**
2.1 Background Study 7
2.2 Literature Review 8
3. Chapter:3 System Analysis **9**
3.1. System Analysis 9
3.2 EDA 14
Chapter 4: System Design **16**
4.1. Design 16
4.2. Algorithm Details 20
Chapter 5: Implementation and Testing **22**
5.1 Implementation 22
5.2 Testing 23
5.3 Result Analysis 28
Chapter6: Conclusion and Future Recommendation **34**
6.1 Conclusion 34
6.2 Future Recommendation 34

References **36**
Appendix **37**
#Display of Automatic Recommendation Code 37
Display of Custom Recommendation Code: 39
#Sign Up Register: 40
#Login Register: 40

List of Figures

Figure 1 : Agile development 5

Figure 2 :Project Schedule Activites 11

Figure 3 :ER diagram 12

Figure 4 :DFD Level0 13

Figure 5 :DFD Level1 14

Figure 6 : Data cleaning 15

Figure 7 :Data preprocessing using StandardScaler 16

Figure 8 :Data Transformation using Algorithm 16

Figure 9 :Architectural Design 17

Figure 10 :Database Design 18

Figure 11 :Login Form 18

Figure 12 :Signup Form 19

Figure 13 :Automatic Recommendation System 20

Figure 14 :Custom Recommendation System 20

Figure 15 : Cosine Similarity 21

Figure 18 : Automatic Recommendation Output 28

Figure 19 : Custom Recommendation Output 29

Figure 20 : Silhouette Score 29

Figure 21 : Silhouette Score of Agglomerative Clustering 30

Figure 22 : Dendrogram for Agglomerative Clustering 31

Figure 23 :Silhouette Score of DBSCAN 31

Figure 24 : DBSCAN Clustering esp and sample figure 32

Figure 25 : Silhouette Score of Kmeans 32

Figure 26 : Kmeans Clustering 33

List of Table

Table 1 :Test case of Signup Functionality 23
Table 2 :Test case of login functionality 25
Table 3 Test case of system responsiveness 26
Table 4 : Test case of Usability Testsing 27

List of Abbreviation

API	Application Programming Interface
CPM	Critical Path Method
DBN	Deep Belief Network
DFD	Data Flow Diagram
ER	Entity Relationship
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
NN	Nearest Neighbor
MBO	Monarch Butterfly Optimization
DRS	Diet Recommendation System

1. Chapter 1: Introduction

1.1. Introduction

The Diet Recommendation System is a tool designed to help users make healthier meal choices based on their personal health goals. The system uses Streamlit for the frontend, FastAPI and Python for the backend, and MongoDB for storing recipe data. Users can enter their age, height, weight, gender, weight loss plan, and the number of meals they want per day using slider controller, making it easy to input data.

Once the user provides their details, the system suggests meals for breakfast, lunch, and dinner based on their information. The system also shows nutritional values like calories, fat, and other nutrients in a pie chart for each recommended meal. The database contains over 20,000 recipes, which are used to find the best meal options for the user.

In the Custom Diet Recommendation feature, users can choose specific nutrition goals, like calorie or fat limits. The system will show a list of recipes that meet these goals, complete with pictures and nutritional details in pie charts.

This system helps users easily plan their meals and stay on track with their health and fitness goals.

1.2. Problem Statement

In today's fast-paced world, maintaining a healthy diet is difficult due to busy schedules, limited nutritional knowledge, and the lack of personalized diet plans.Unhealthy eating patterns and health problems can result from basic meal recommendations that frequently fail to take into consideration specific health goals like weight loss as well as individual needs like age, height, weight, and gender.

Furthermore, a lot of diet suggestion systems are difficult to use and don't provide visual representations of nutritional values, which makes it more difficult for users to stick to the plans. There is a need for a simple, data-driven solution that provides personalized meal suggestions while ensuring nutritional balance.

By developing a diet recommendation system that provides personalized meal plans based on user input, this project seeks to address these issues. In order to deliver precise and understandable recommendations—including visual representations such as pie charts for nutritional values—the system will make use of technologies such as Streamlit, FastAPI, MongoDB, and the Nearest Neighbor algorithm.The goal is to help users in

reaching their fitness and health objectives in an easy and long-lasting manner.

1.3. Objectives

The goal of this project is to create a personalized diet suggestion system that gives customers meal plans that are specific to their weight loss objectives and health information.Key objectives include:

- Allow users to enter their gender, age, height, weight, preferred number of meals per day, and weight loss plan.
- Create individualized breakfast, lunch, and supper meal suggestions using this data.
- Make that the suggested meals meet the dietary requirements of the user.
- For a better understanding, use a pie chart to visually show the nutritional information.
- Provide a user-friendly, interactive platform that helps people control their weight by assisting them in making educated food choices.

1.4. Scope and Limitations

❖ **Scope:**

The scope of the project are follows:

- **Personalized Suggestions:** The system will generate customized meal plans according to the user's health data, including age, height, weight, gender, weight loss strategy, and daily food intake.
- **User-Friendly Interface:** The system will have an intuitive, interactive frontend that uses Streamlit to let users enter their information and get meal recommendations without any problems.
- **Backend Logic:** Python and FastAPI will manage the backend logic, guaranteeing effective data processing and suggestions for meal plans.
- **Nutrition Visualization:** To help users comprehend the nutritional balance of their meals, the system will use pie charts to visually show the nutritional values of the suggested meals.
- **Database Integration:** User information, meal suggestions, and nutritional data will be safely stored in MongoDB.

❖ **Limitations:**

The limitations of the proposed system are as follows:

- **Data Accuracy:** The accuracy of the user input determines the recommendations. faulty meal recommendations could result from any missing or faulty information.
- **Static Meal Database:** A predetermined list of meals and their nutritional values serve as the basis for the meal suggestions. Dynamic or extensive dietary preferences might not be taken into consideration by the system.
- **Health disorders:** Certain allergies and medical disorders are not taken into account by the system. When it comes to diet regimens, users with medical concerns might need to speak with a healthcare provider.
- **Restricted Meal Variety:** Because of the limitations of the meal database and nutritional recommendations, the system may only suggest breakfast, lunch, and dinner.
- **No Real-time Updates:** The system does not track users' progress or provide real-time meal plan modifications.

1.5. Development Methodology

Agile development will be used for this project, guaranteeing adaptability and iterative development. The following stages will comprise the development:

Requirement Gathering and Analysis:

Determining the system's features and comprehending user requirements.
determining user requirements for meal planning, dietary advice, and nutritional visualization.
Gathering technical requirements for the technologies (Streamlit, FastAPI, Python, MongoDB) to be used.

System Design:

creating the system architecture, which includes the database (MongoDB), backend (FastAPI and Python), and frontend (Streamlit interface).To guarantee an intuitive and user-friendly design, wireframes for the user interface must be created.creating the database structure to hold nutritional information, meal information, and user data.

Frontend Development:

creating a responsive and interactive frontend using Streamlit that lets users enter their information (weight, height, age, etc.) and see the suggested meals.including data validation checks and input forms to guarantee accurate user input.

Backend Development

The backend functionality for processing user inputs and creating customised meal plans based on nutritional data is developed using Python and FastAPI by putting in place algorithms that, depending on the user's profile (weight loss goal, daily meals, etc.), suggest breakfast, lunch, and dinner where integrating APIs to manage front-end and back-end data flow.

Databases Development:

Using MongoDB to hold nutritional data, food planning, and user information.

putting in place safe and effective procedures for data retrieval and storage.

Testing and Integration:

Integrating the database, frontend, and backend to provide smooth communication between the various parts of the system.Carrying out unit and integration testing to guarantee the accuracy and effectiveness of the user interface, nutritional computations, and meal recommendations.

To get input and make changes, conduct user acceptability testing (UAT).

Deployment:

making certain that the system is scalable and functions effectively for numerous users.

Maintenance and Updates:

After deployment, continuing maintenance and support are given to fix bugs, performance problems, and upcoming updates.

gathering user input to enhance system performance and, if necessary, add new features.

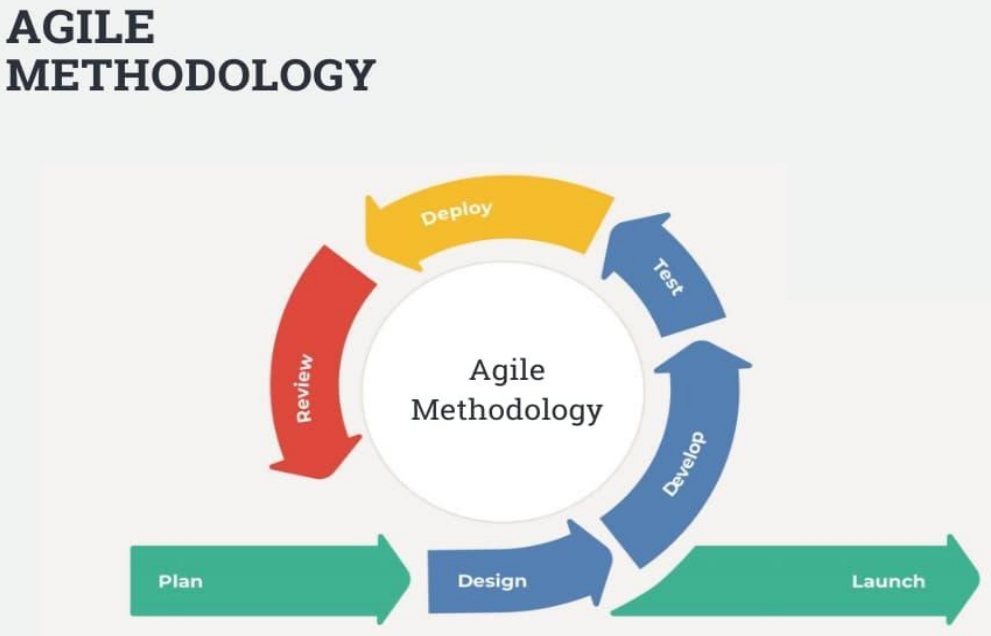


Figure 1: Agile development

1.6. Report Organization

Chapter 1: Introduction

Overview: An introduction of the project that explains its significance and goal.

Problem Statement: The issue or gap that the project aims to address.

Objectives: certain objectives that the initiative seeks to fulfill.

Scope: The boundaries of the project, including its functionalities and target audience.

Chapter 2: Background Study and Literature Review

Background Study: An examination of the project's larger background, encompassing developments in diet guidance systems and their significance for fitness and health.

Literature Review: A detailed analysis of existing works, related systems, and technologies. This section lists the drawbacks of the existing solutions and describes how the suggested method overcomes them.

Chapter 3: System Analysis

Requirements Analysis: A thorough justification of the system's functional and non-functional requirements based on user demands.

Feasibility Study: Analysis of technical, economic, and operational feasibility.

System Objectives: Specific goals for the performance, usability, and usefulness of the system.

Use Case Diagrams: a visual representation of how a user interacts with a system.

Chapter 4: System Design

System architecture: An explanation of the system's general composition and component interactions.

Data flow diagrams (DFD): Show the flow of data across a system.

Database Design: MongoDB's schema and table relationships.

UI Design: Wireframes or mockups of the user interface, detailing how users interact with the system.

Chapter 5: Implementation and Testing

Implementation: Describes how the system was created with Python for data processing, FastAPI for backend functionality, Streamlit for the frontend, and MongoDB for data storage.

An explanation of the methods used to provide individualized meal recommendations and the pie chart visualization of nutritional data.

Testing:

Testing Methodology: An explanation of user acceptability, integration, and unit testing.

Test Cases: Specific scenarios used to test the functionality and reliability of the system.

Results: An overview of the test's findings that highlights problems found and how they were fixed.

Chapter 6: Conclusion and Future Recommendation

Conclusion: A synopsis of the project's accomplishments, highlighting the effective creation and operation of the dietary advice system.

evaluation of the project's compliance with the introduction's goals.

Future Recommendations:

Ideas for enhancing the system include expanding the menu, including AI for advanced modification, or allowing for dietary restrictions.

Possible improvements include monitoring user progress, providing real-time information, or branching out to mobile platforms.

Chapter 2: Background Study and Literature Review

2.1 Background Study

A recommendation system is a machine learning-based approach that leverages data to predict, filter, and identify what users are searching for among a vast array of options. Its ability to accurately anticipate user preferences and choices has led to its widespread adoption across various sectors. Systems can employ methods such as content-based filtering, collaborative filtering, hybrid filtering, and user-based collaborative filtering.

Types of recommender system:

- **Content based filtering**

  This technique recommends meals based on their nutritional content and user preferences. For instance, if a user prefers low-calorie meals, the system will suggest meals with fewer calories, focusing on the ingredients and nutritional values of each dish. It analyzes the features of the recipes and matches them to the user's specified needs, like calories, fat content, or protein.

- **Collaborative Filtering**

This method relies on user behavior and preferences to suggest meals that similar users have enjoyed. The system looks at the choices of users who have similar tastes and health goals to recommend meals based on their preferences, even if those meals don't match the user's exact specifications. **[2]**

- **Hybrid Filtering**

Combining content-based and collaborative filtering techniques, hybrid filtering enhances recommendations by leveraging both user preferences and behaviors. This method offers more accurate suggestions by considering both the features of the meals and the collective input from similar users. [2] [3]

- **User based collaborative filtering**

A specific type of collaborative filtering, this method suggests meals based on the preferences of users who have similar characteristics, such as age, gender, or fitness goals. The system finds users with comparable tastes and recommends meals they have liked or found beneficial, ensuring personalized recommendations.

Personalized diet-food recommendation systems have developed as effective tools for

encouraging healthy eating habits and reaching personal dietary objectives. These systems use complex algorithms including content-based filtering, collaborative filtering, and hybrid approaches to customize meal suggestions based on user information such as age, gender, weight, dietary preferences, and health goals. The efficacy of such systems is heavily dependent on the quality of the recipe database, the accuracy of nutritional data, and the user interface's usability.

2.2 Literature Review

Personalized diet suggestion systems have gained popularity as a result of the increased need for health-conscious solutions that cater to specific dietary needs. These systems develop personalized meal plans using powerful algorithms, user input data, and massive recipe libraries. Research has demonstrated that using artificial intelligence and machine learning techniques improves the accuracy and usefulness of such systems. [4]

Recommender systems are vital for making specialized suggestions. Several approaches have been used, including content-based filtering, collaborative filtering, and hybrid methods. Content-based filtering suggests meals based on their nutritional value and the user's choices, such as calorie limitations or macronutrient distribution. In contrast, collaborative filtering takes into account user behavior and preferences, making suggestions based on data from users with similar dietary objectives. Hybrid filtering combines these strategies to improve suggestion accuracy and customisation by using content-specific data as well as user input. [5]

Food recommendation systems use modern technology like artificial intelligence and multimodal data fusion to encourage healthy eating habits. These systems provide individualized meal ideas by combining user-specific circumstances, such as health measures and preferences, with a variety of data sources such as food imagery, nutritional profiles, and data from wearable sensors. Frameworks such as those developed by Min et al. emphasize the need of combining domain knowledge and building personal models to balance individual preferences and health requirements. However, issues remain in expressing changing dietary preferences, integrating disparate data, and creating complete knowledge graphs. Future improvements, such as explainable systems, large-scale datasets, and reinforcement learning, promise to improve these systems' accuracy, scalability, and influence on public health. [6]

Visualization methods, such as pie charts and graphs, have been useful in helping users grasp the nutritional value of advised meals. These technologies assist users in efficiently

interpreting data, resulting in improved comprehension and adherence to dietary recommendations. Khan et al. (2021) found that such visual aids increase user engagement, ensuring that dietary recommendations are more matched with individual health objectives. Visualization approaches improve the entire user experience by presenting information in an understandable and visually appealing style, making complicated nutritional data more accessible. [7]

Despite substantial advances, diet advice systems have limits, most notably their capacity to adjust flexibly to user demands. Many systems lack the ability to track progress or alter suggestions based on real-time input. Gordon et al. (2020) underline the significance of using adaptive learning models to enhance responsiveness. Expanding recipe databases to encompass a wide range of dietary preferences, as well as using advanced filtering techniques like hybrid filtering and ingredient exclusion, can help to solve user-specific difficulties. These modifications can help to improve system accuracy and usability, opening the way for more inclusive and successful nutritional solutions. [8]

3. Chapter:3 System Analysis

3.1. System Analysis

The system analysis phase is critical for understanding the requirements, identifying potential constraints, and ensuring that the project is feasible in terms of technology, operations, cost, and time. The following sections outline the key aspects of the system analysis for the diet recommendation system.

3.1.1. Requirement Analysis

i. Functional Requirements

- Functional requirements define the core features and behaviors the system must support. These include:
- User Registration and Login: Users must be able to create an account and log in to access personalized features.
- Data Input: The system should allow users to input their personal information (age, height, weight, gender, and meal preferences).
- Meal Recommendation: Based on the user data, the system generates personalized meal plans for breakfast, lunch, and dinner.
- Nutritional Visualization: The system should display nutritional values of recommended meals in the form of pie charts.

ii. Non-functional

Non-functional requirements describe the system's performance and constraints. Key non-functional requirements for this project include:

- **Usability:** The system should have a simple, user-friendly interface, allowing users to interact with it without technical expertise.
- **Performance:** The system must provide meal recommendations in real-time, with minimal latency.
- **Scalability:** The system should handle a growing number of users without performance degradation.
- **Security:** User data must be securely stored and protected against unauthorized access.
- **Reliability:** The system should function smoothly with minimal downtime or errors.

3.1.2. Feasibility Analysis

Feasibility analysis helps assess whether the project is viable from a technical, operational, economic, and schedule perspective.

i. Technical Feasibility

The project leverages technologies like Streamlit for the frontend, FastAPI for the backend, Python for processing logic, and MongoDB for storing user data. These technologies are widely supported, easy to integrate, and well-suited for the project's requirements. The technical feasibility is high, as these technologies ensure scalability, security, and performance.

ii. Operational Feasibility

The system is designed to be intuitive and easy to use for non-technical users, making it operationally feasible. The project's operational success depends on ensuring the system functions properly in a real-world environment, with users able to access it via a web interface without significant issues.

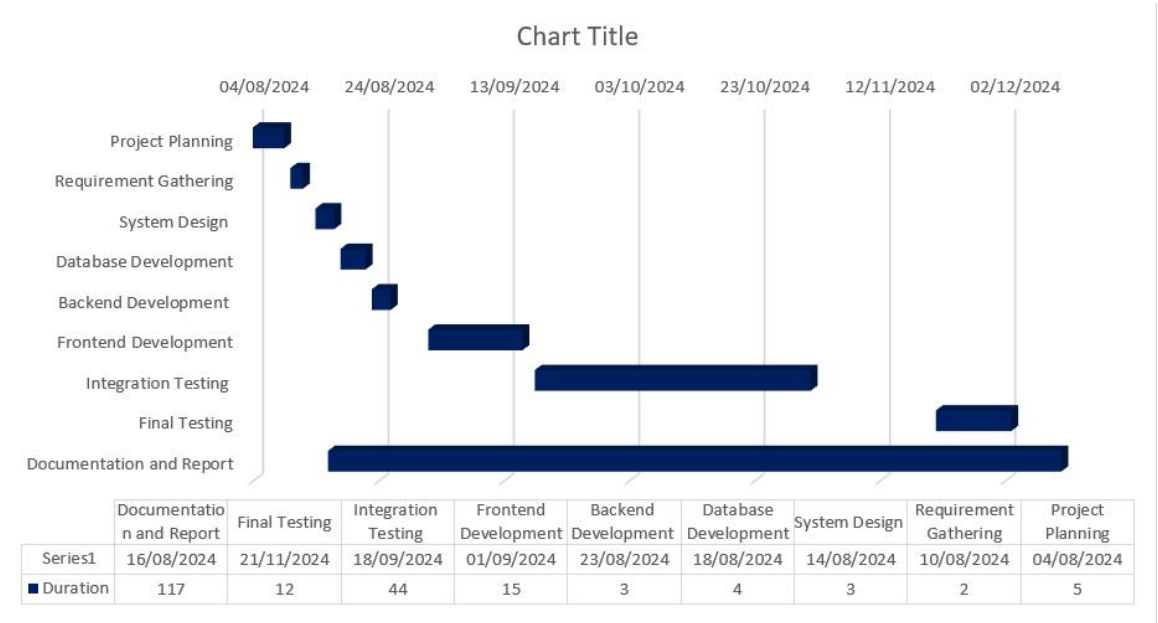
iii. Economic Feasibility

The project is economically feasible as it uses open-source technologies (Streamlit, FastAPI, Python, MongoDB) that do not incur licensing costs. The primary cost will be the time and effort required for development, testing, and deployment, which is manageable within the project's scope and budget. The project is financially viable because the datasets are free available datasets from Kaggle /Food.

.iv. Schedule Feasibility

The project is expected to be completed within a reasonable timeline, assuming adequate resources are allocated. The project phases—design, development, testing, and deployment—are well-defined and achievable within a typical software development lifecycle (e.g., 3-4 months). There is sufficient time for testing and feedback before the final deployment. This feasibility analysis ensures that the project is technically, operationally, economically, and schedule-wise viable for successful implementation.

	Task Name	Start Date	End Date	Duration
2	Project Planning	04/08/2024	09/08/2024	5
3	Requirement Gathering	10/08/2024	12/08/2024	2
4	System Design	14/08/2024	17/08/2024	3
5	Database Development	18/08/2024	22/08/2024	4
6	Backend Development	23/08/2024	26/08/2024	3
7	Frontend Development	01/09/2024	16/09/2024	15
8	Integration Testing	18/09/2024	01/11/2024	44
9	Final Testing	21/11/2024	03/12/2024	12
10	Documentation and Report	16/08/2024	11/12/2024	117



	Documentation and Report	Final Testing	Integration Testing	Frontend Development	Backend Development	Database Development	System Design	Requirement Gathering	Project Planning
Series1	16/08/2024	21/11/2024	18/09/2024	01/09/2024	23/08/2024	18/08/2024	14/08/2024	10/08/2024	04/08/2024
Duration	117	12	44	15	3	4	3	2	5

Figure 2:Project Schedule Activites

3.1.3. Analysis

The data and procedures of the system have been modeled for this project using an organized methodology. While the Data Flow Diagram (DFD) is used for process modeling, the Entity-Relationship (ER) diagram is used for data modeling.

i. ER-Diagram

The ER diagram will illustrate how the data entities relate to each other in the database.

Key entities could include:

User: Stores user details like age, height, weight, gender, etc.

MealPlan: Contains meal suggestions (breakfast, lunch, and dinner).

Meal: Specific meal details such as ingredients, nutrition values, etc.

Nutrition: Contains nutritional values (calories, carbs, proteins, fats).

WeightLossPlan: Different plans available based on user goals (e.g., lose weight, maintain weight).

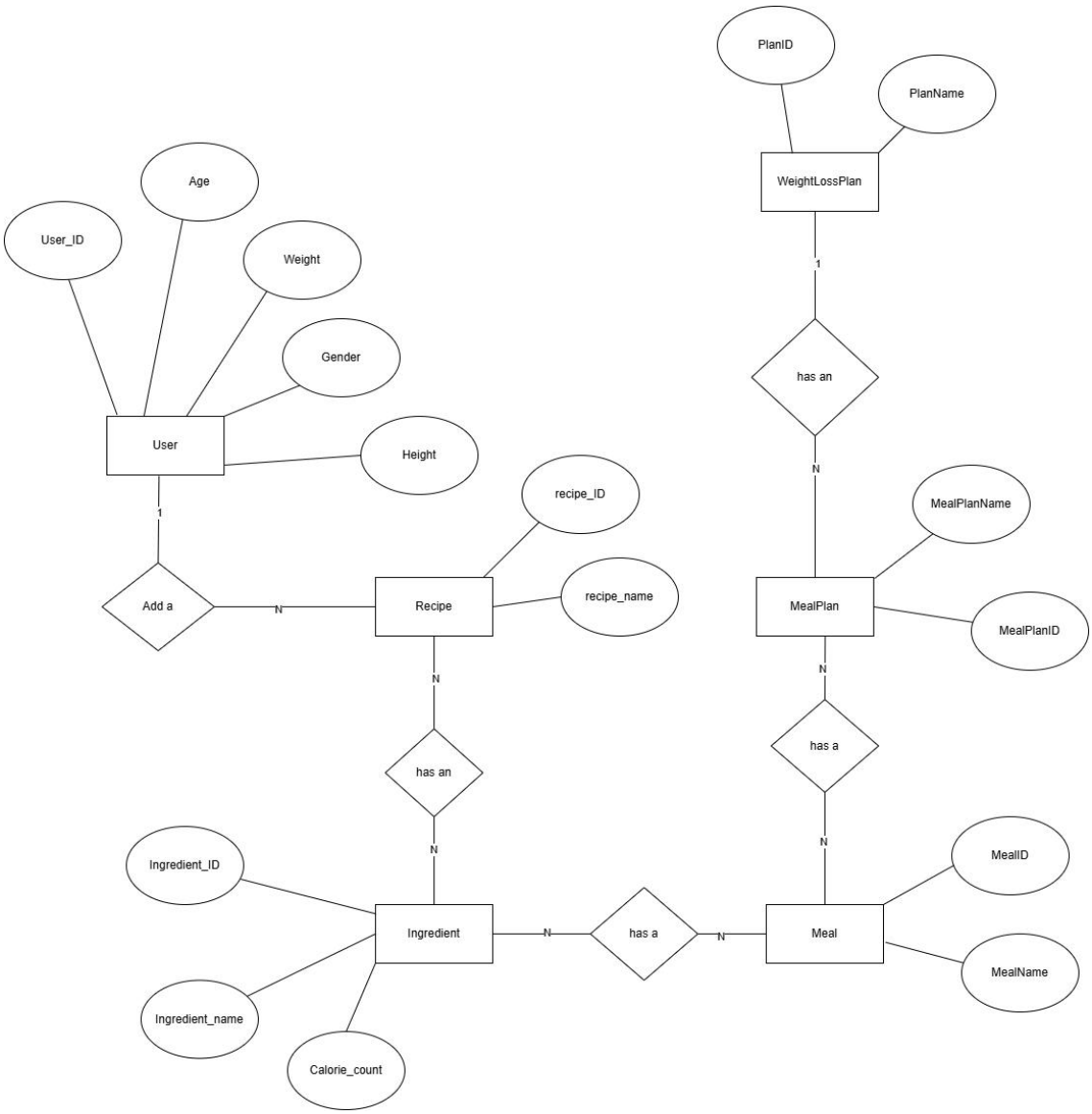


Figure 3:ER diagram

ii. DFD Diagram

The DFD shows how the data flows through the system. It can be represented at different levels (Level 0, Level 1, etc.):

Level 0 (Context Diagram):

User Input: The user enters their age, height, weight, gender, weight loss goal, and meals per day. **System:** Based on user input, the system generates recommended meals and nutritional data.

Output: The system provides the recommended meals and shows the nutrition values as a pie chart.

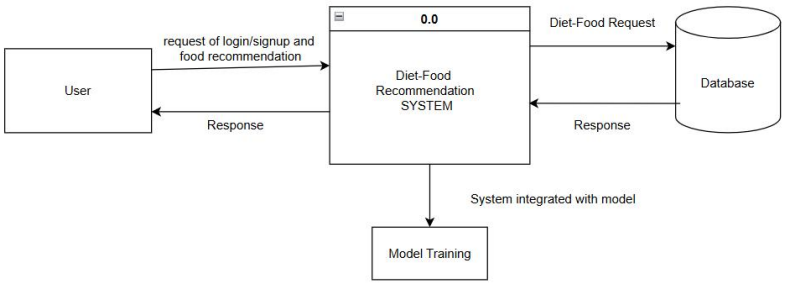


Figure 4:DFD Level0

Level 1:

User: Provides input data (age, height, weight, etc.).

Frontend (Streamlit): Collects the user data and sends it to the backend. Backend **(FastAPI/Python):** Processes the data, applies logic based on the weight loss plan, and generates meal recommendations.

Database (MongoDB): Stores user data, meal plans, and nutritional data. **Output (Streamlit):** Displays meal recommendations and the nutrition pie chart.

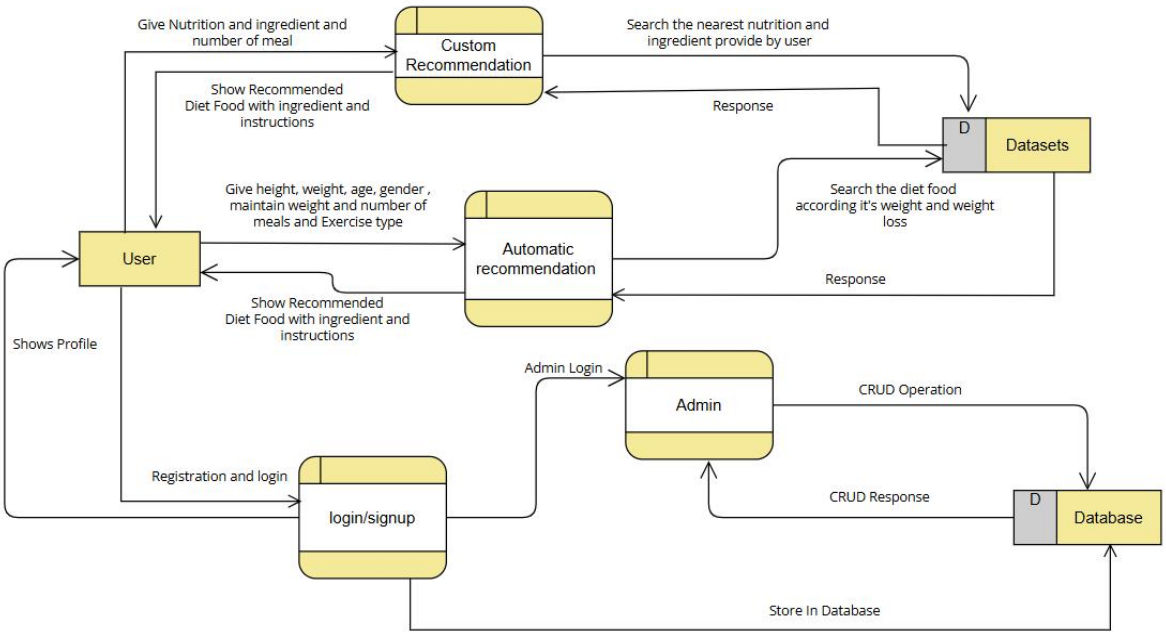


Figure 5:DFD Level1

3.2 EDA

Exploratory Data Analysis (EDA) is an important step in the data science and machine learning workflows. Before moving on to system development or modeling, the dataset must be extensively examined to find trends, patterns, detect irregularities, and test hypotheses.

For this experiment, we used a dataset from Kaggle's Food Dataset, which contains information about over 500,000 recipes, including ingredients, calorie values, and instructions. Given the recommendation system's computational and performance limitations, we chose a subset of 20,000 records from this dataset. This choice offers the best system performance while striking a balance between dataset diversity and processing speed.

Dataset Overview

Initial dataset size: 500,000 records.

Selected dataset size is 20,000 records.

Key features:

Recipe IDs are unique identifiers for each recipe.

items: A list of the items used in the recipe.

Calorie count for the recipe.

Instructions: Step-by-step cooking directions.

The dataset comprises both numerical and textual variables, which play an important role in developing the recommendation system employing approaches content-based filtering.

Key EDA Activities:

Data Cleaning: Fixed missing values in features like Ingredients and Calories by adding or eliminating inadequate information.Duplicate recipes were removed to ensure data quality.

```
n [7]:
dataset=data.copy()
columns=['RecipeId','Name','CookTime','PrepTime','TotalTime','RecipeIngredientParts','Calories','FatContent','SaturatedFatC
dataset=dataset[columns]

n [8]:
max_Calories=2000
max_daily_fat=100
max_daily_Saturatedfat=13
max_daily_Cholesterol=300
max_daily_Sodium=2300
max_daily_Carbohydrate=325
max_daily_Fiber=40
max_daily_Sugar=40
max_daily_Protein=200
max_list=[max_Calories,max_daily_fat,max_daily_Saturatedfat,max_daily_Cholesterol,max_daily_Sodium,max_daily_Carbohydrate,ma

n [9]:
extracted_data=dataset.copy()
for column,maximum in zip(extracted_data.columns[6:15],max_list):
    extracted_data=extracted_data[extracted_data[column]<maximum]

[10]:
extracted_data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 375703 entries, 0 to 522515
Data columns (total 16 columns):
```

Figure 6: Data cleaning

Preprocessing and Transformation

Data pretreatment and transformation are critical components of a diet-food recommendation system. Initially, nutritional parameters such as calories, fat, protein, and sugar are normalized using StandardScaler. This guarantees that all characteristics have a mean of zero and a standard deviation of one, making the data similar across scales. Next, a Pipeline is formed by merging StandardScaler and NearestNeighbors with FunctionTransformer. This configuration converts the data and uses cosine distance to obtain similarity ratings between food items. The NearestNeighbors model is built with n_neighbors=10 to identify the ten most comparable food products based on nutritional characteristics.This preprocessing stage prepares the data for personalised meal suggestions by assuring consistency and maximizing performance.

```
2]: from sklearn.preprocessing import StandardScaler
    scaler=StandardScaler()
    prep_data=scaler.fit_transform(extracted_data.iloc[:,6:15].to_numpy())

3]: prep_data

3]: array([[-0.55093359, -0.91281917, -0.77924852, ...,  0.15672078,
            2.35502102, -0.68338127],
           [ 1.47428542,  1.13139595, -0.0647135 , ...,  3.91055068,
            2.56324444,  1.25158691],
           [-0.92414618, -1.11248669, -1.12222533, ...,  0.4855234 ,
            0.98513013, -0.60183088],
           ...,
           [ 0.49162165,  0.73206091,  1.85024037, ..., -0.61048534,
            1.76322815, -0.56476253],
           [ 0.25704672,  0.03797856,  1.02137974, ..., -0.61048534,
            1.54404561, -0.63148557],
           [-1.40937801, -1.09347074, -1.12222533, ..., -0.82968708,
            -0.94367625, -0.74269064]])
```

Figure 7:Data preprocessing using StandardScaler

```
[15]: from sklearn.pipeline import Pipeline
      from sklearn.preprocessing import FunctionTransformer
      transformer = FunctionTransformer(neigh.kneighbors,kw_args={'return_distance':False})
      pipeline=Pipeline([('std_scaler',scaler),('NN',transformer)])

[16]: params={'n_neighbors':10,'return_distance':False}
      pipeline.get_params()
      pipeline.set_params(NN__kw_args=params)

[16]: Pipeline(steps=[('std_scaler', StandardScaler()),
                      ('NN',
                       FunctionTransformer(func=<bound method KNeighborsMixin.kneighbors of NearestNeighbors(algorithm='brute', me
      tric='cosine')>,
                                           kw_args={'n_neighbors': 10,
                                                    'return_distance': False}))])

[17]: pipeline.transform(extracted_data.iloc[0:1,6:15].to_numpy())[0]

[17]: array([     0, 333440, 349044, 109248,  19679, 156831, 144322, 301119,
             262699, 332342])
```

Figure 8:Data Transformation using Algorithm

Chapter 4: System Design

The system design of the Diet Recommendation System consists of several key components, including Database Design, Form Design, and Interface Design. Each of these components is essential to creating an efficient, user-friendly, and scalable system.

4.1. Design

The Diet-Food Recommendation System is designed to help individuals achieve weight

loss, maintain their ideal weight, and improve their overall health. The system is user-friendly and highly accurate, tailoring diet recommendations to individual needs based on their height, weight, age, and the exercises they perform.

4.1.1. Architectural Design

The Diet-Food Recommendation System utilizes the three-tier architecture, implementing client-server pattern. Here the presentation layer provides a user-friendly interface to client, and the application layer ensures the accuracy of diet-food recommended and retrieves all the important data related to the detect result. Together, these layers work harmoniously to provide users with fast, accurate results and valuable insights all in one easy-to-use system.

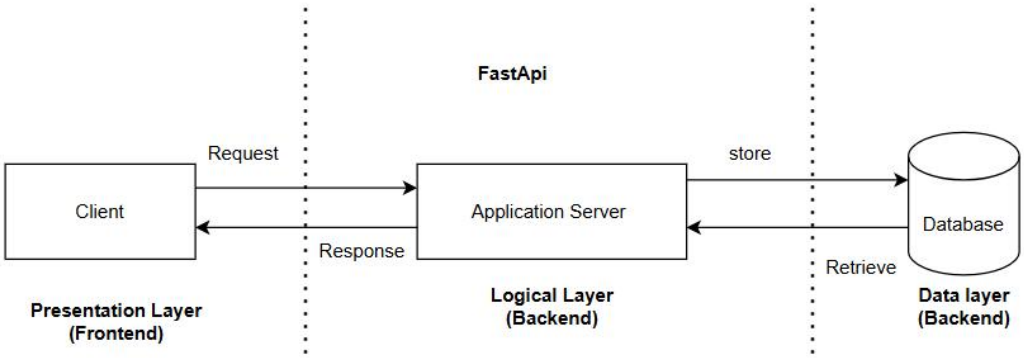


Figure 9:Architectural Design

4.1.2. Database Design

The Diet-Food Recommendation System strongly relies on the database designed for the login/Signup and Security of the application and it is mainly required for the storing the user information. MongoDB Compass a NoSQL database is used for the proper management related to this project.

Figure 10:Database Design

Form Design

The Login and Signup forms are crucial components for user authentication in the diet recommendation system. These forms ensure that only authorized users can access the system, and new users can create accounts to use the system's features.

Figure 11:Login Form

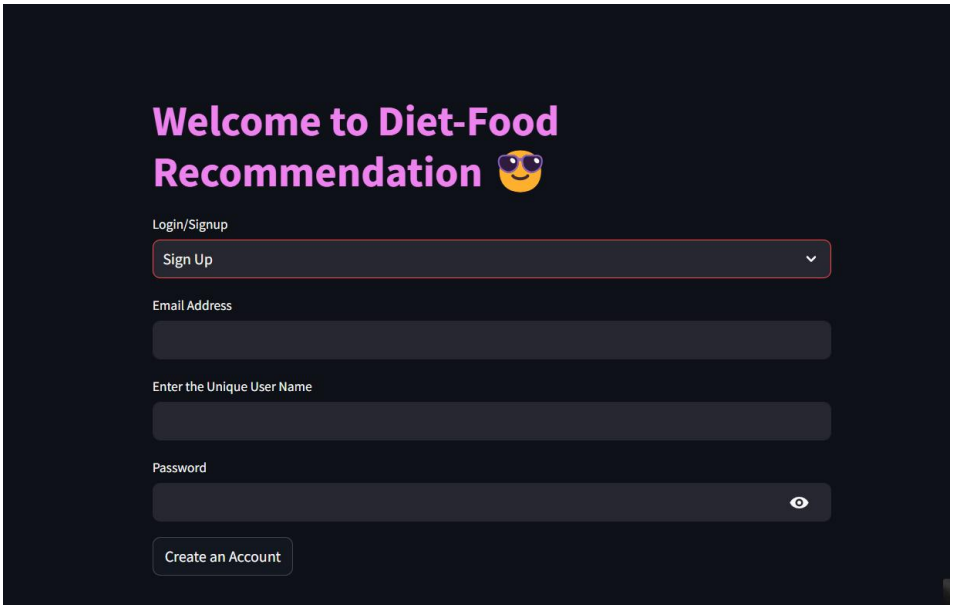


Figure 12:Signup Form

Interface Design

Users enter data using sliders for age, height, weight, gender, weight loss plan, and daily meals. The algorithm then recommends personalized breakfast, lunch, and dinner options together with all the necessary information, including ingredients, cooking instructions, and a pie chart showing the nutritional values. Users can switch between light and dark modes for comfort, and a settings area allows them to tailor the settings. The responsive design ensures a smooth experience across devices, enabling the user-friendly and fun meal planning tool.The interface of our system will be shown in the following example.

.

Automatic Recommendation System Page

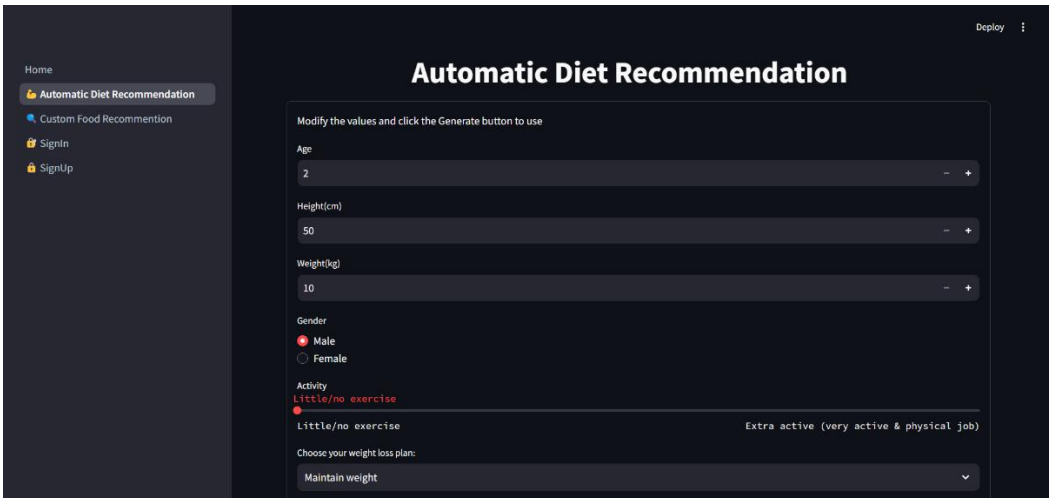


Figure 13:Automatic Recommendation System

Custom Recommendation System Page

Figure 14:Custom Recommendation System

4.2. Algorithm Details

Nearest Neighbor Algorithm:

The Nearest Neighbors (k-NN) algorithm is indeed an unsupervised learning algorithm used for neighbor searches, and it's often used in recommendation systems, anomaly detection, and clustering tasks. It doesn’t require labeled data and works by finding the closest neighbors in the feature space to a given query. It is a simple, instance-based machine learning algorithm. It doesn't involve a training phase in the traditional sense.

Instead, it stores the training data and uses it to find similar examples when making predictions. This method is often used in recommendation systems, classification, and regression tasks.

- Training Phase:

The algorithm doesn't learn parameters or create a model in the typical sense. It only stores the training data.

- Prediction Phase:

When a new data point (query) is provided, the algorithm calculates the distance between the query point and all the data points in the training set.

It then returns the closest (or k nearest) data points based on the distance metric used.

Distance Metric: Cosine Similarity:

The distance metric used is cosine similarity, which is widely used for text and document similarity as it measures the cosine of the angle between two vectors.

Cosine Similarity Formula:

For two vectors A and B, cosine similarity is computed as:

$$\text{Cosine similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

Figure 15: Cosine Similarity

Cosine similarity ranges from:

1: Perfectly similar (the vectors are pointing in the same direction),

0: No similarity (the vectors are orthogonal),

-1: Perfectly dissimilar (the vectors are pointing in opposite directions).

Algorithm: Brute Force:

One of the current areas of machine learning research is the quick computation of nearest neighbors. The most basic neighbor search implementation uses a brute-force method to calculate the distances between every pair of points in the dataset; this method scales as O[DN2] for N samples in D dimensions. For tiny data samples, effective brute-force neighbor searches can be highly competitive. However, the brute-force method soon becomes unfeasible as the number of samples N increases.

Chapter 5: Implementation and Testing

5.1 Implementation

5.1.1 Tools Used

- CASE Tools:

Draw.io: Used for creating UML diagrams.

MS Excel: Used for project scheduling and generating Gantt charts.

- Programming Languages:

Frontend: Streamlit for creating an interactive and user-friendly interface.

Backend: FastAPI and Python for scalable and efficient backend processing.

- Database Platform:

MongoDB: Stores 20,000 recipe entries with nutritional information.

- Development Tools:

Visual Studio Code: For coding.

Git: For version control.

Postman: For API testing.

- Libraries/Frameworks:

Streamlit, FastAPI, Uvicorn.

Scikit-learn, Pandas, BeautifulSoup, Streamlit-echarts, Numpy: For data processing and additional functionality.

4.2.1. Implementation Details

The Profile Manager is the system's gateway for user login, registration, and profile maintenance. It guarantees that users can safely log in, sign up, and change their settings. User registration needs a unique email address and username, and passwords are securely hashed using techniques such as bcrypt to prevent unwanted access. Login feature checks credentials and creates access tokens for session management, ensuring that sensitive data is kept safe. MongoDB is used to maintain user profiles, which contain personal information like age, height, weight, gender, and dietary preferences. The Profile Manager also allows users to change their profiles, ensuring that the recommendation system is responsive to changing dietary or health goals.

Data Processing Module

The Data Processing Module is in charge of preparing the dataset so that suggestions may be made efficiently and accurately. Initially, the system cleans the dataset to handle missing values, delete duplicate entries, and verify its integrity. Calories, protein, fat, and carbs are standardized using techniques like StandardScaler to ensure uniformity between scales. The processed dataset, which was produced from a curated subset of 20,000 recipes, contains information such as ingredients, nutritional values, and cooking directions. Advanced preprocessing methods tokenize and vectorize ingredients, allowing for ingredient-based filtering. This module also includes a pretreatment pipeline that uses normalization and Nearest Neighbor modeling to speed the recommendation process.

Recommendation Module

The Recommendation Module uses the Nearest Neighbor Algorithm with cosine similarity to provide individualized meal suggestions based on user inputs such as age, gender, weight, dietary objectives, and favorite meals per day. It uses the user's information to find dishes that best meet their nutritional needs from a collection of over 20,000 recipes. The module offers two modes: automatic recommendations, which provide personalized breakfast, lunch, and supper alternatives, and custom recommendations, which allow users to specify dietary limitations such as calorie limits or item exclusions. Each advice is complemented with nutritional information, which is shown as interactive pie charts for easier comprehension. The module provides accuracy, scalability, and user engagement by integrating seamlessly with Streamlit, FastAPI, and MongoDB.

5.3. Testing

5.3.1. Unit Testing

Table 1:Test case of Signup Functionality

Test Case ID	Test Description	Input Test Data	Expected Result	Actual Result	Status

TC001	Verify successful signup	Select "Signup" from the dropdown Username: dipeshmandal601 Password: Password123! Email: dipeshmandal601@gmail.com	User is successfully registered, and confirmation email is sent.	User successfully registered, and confirmation email sent.	Pass
TC002	Verify signup with an existing email	Select "Signup" from the dropdown Username: user1 Password: Password123! Email: dipeshmandal601@gmail.com	Error message: "Email already in use. Please use a different email."	Error message: "Email already in use. Please use a different email."	Pass
TC003	Verify signup with existing username	Select "Signup" from the dropdown Username: dipeshmandal601 Password: Password123! Email: dipeshmandal601@gmail.com Email: newemail@gmail.com	Error message: "Username already taken. Please choose a different username."	Error message: "Username already taken. Please choose a different username."	Pass
TC004	Verify invalid email format rejection	Select "Signup" from the dropdown Username: aakriti1 Password: Password123! Email: aakritidc1gmail.com	Error message: "Please enter a valid email address."	Error message: "Please enter a valid email address."	Pass

TC005	Verify missing password error	Select "Signup" from the dropdown Username: dipeshmandal601 Password: (empty) Email: dipeshmandal601@gmail.com	Error message: "Password is required."	Error message: "Password is required."	Pass

Table 2:Test case of login functionality

Test Case ID	Test Description	Input Test Data	Expected Result	Actual Result	Status
TC001	Verify successful login after selecting "Login" from the dropdown	Select "Login" from the dropdown Email: dipeshmandal601@gmail.com Password: Password123!	User successfully logged in and redirected to the dashboard.	User successfully logged in and redirected to the dashboard.	Pass
TC002	Verify login with incorrect email	Select "Login" from the dropdown Email: dip@example.com Password: Password123!	Error message: "Invalid email or password."	Error message: "Invalid email or password."	Pass

TC003	Verify login with incorrect password	Select "Login" from the dropdown Email: dipeshmandal601@gmail.com Password: Password123! Email: dipeshmandal601@gmail.com	Error message: "Invalid email or password."	Error message: "Invalid email or password."	Pass
TC004	Verify login with blank email address	Select "Login" from the dropdown Email: (empty) Password: Password123!	Error message: "Email is required."	Error message: "Email is required."	Pass
TC005	Verify login with blank password	Select "Login" from the dropdown Email: dipeshmandal601@gmail.com Password: (empty)	Error message: "Password is required."	Error message: "Password is required."	Pass

5.3.2. System Testing

Table 3:Test case of system responsiveness

Test Case ID	Test Case Description	Steps to Perform	Expected Outcome	Actual Outcome	Status
TC-01	Validate connection between backend and model	Send data to model → Check response	Prediction is accurate	As expected	Pass
TC-02	Test data flow to result display	Submit data → Process → Fetch results →	Results display correctly	As expected	Pass

		Check display			
TC-03	Verify data flow from form to backend	Fill form → Submit → Check backend processing	Data is processed and stored correctly	As expected	Pass
TC-04	Test error handling for invalid data	Submit invalid data → Check error message	Appropriate error message displayed	As expected	Pass
TC-05	Test prediction time	Submit data → Measure time for result	Result returned within expected time	As expected	Pass

Table 4: Test case of Usability Testsing

Test Case ID	Test Case Description	Steps to Perform	Expected Outcome	Actual Outcome	Status
UT-01	Verify easy navigation	Open the website → Navigate through the main menu	The user can easily navigate through the menu and sections	User navigated without difficulty	Pass
UT-02	Check form submission	Open the "Sign Up" form → Fill valid details → Submit	The form submits without errors, and a confirmation message appears	Form submitted successfully	Pass

UT-03	Validate error message clarity	Enter invalid data in the login form → Submit	A clear error message appears	Error message displayed as expected	Pass

Result Analysis

The diet recommendation system meets the expectations. Users can sign up and log in to access the system and input their details such as age, height, weight, gender, weight loss plan, and meals per day through slider controls. Validation is implemented in the signup/login functionality to ensure security. The system automatically generates breakfast, lunch, and dinner recommendations based on user inputs, with nutritional values represented in a pie chart. For custom recommendations, users can specify calorie, fat, and nutrient preferences, and the system provides tailored meal suggestions along with pictures of recipes. The recommendations are accurate and user-friendly, meeting the dietary goals set by the users. It provides Diet-Food recommendations with Silhouette Score of 85%.

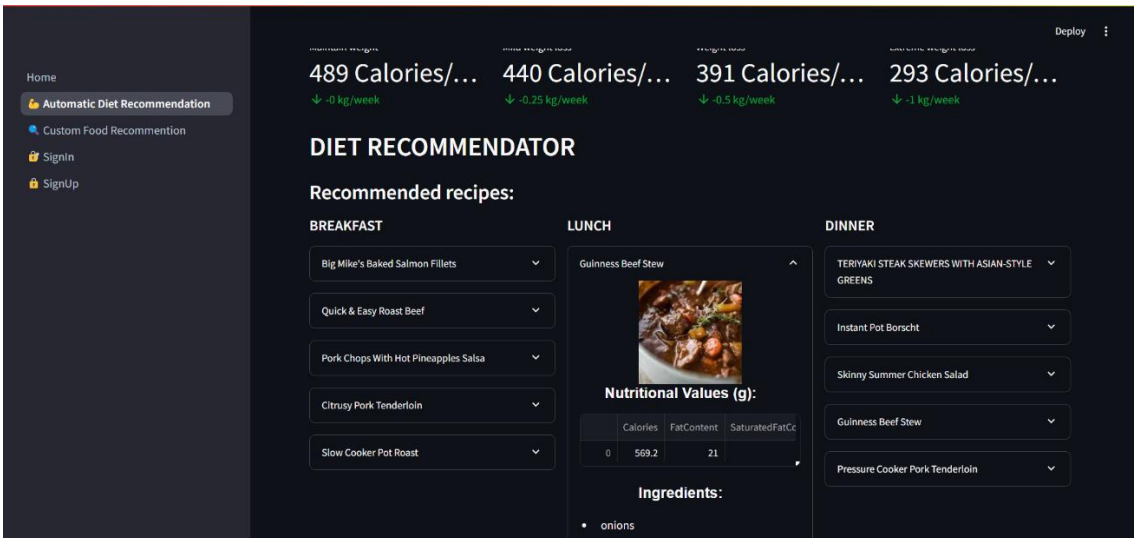


Figure 18: Automatic Recommendation Output

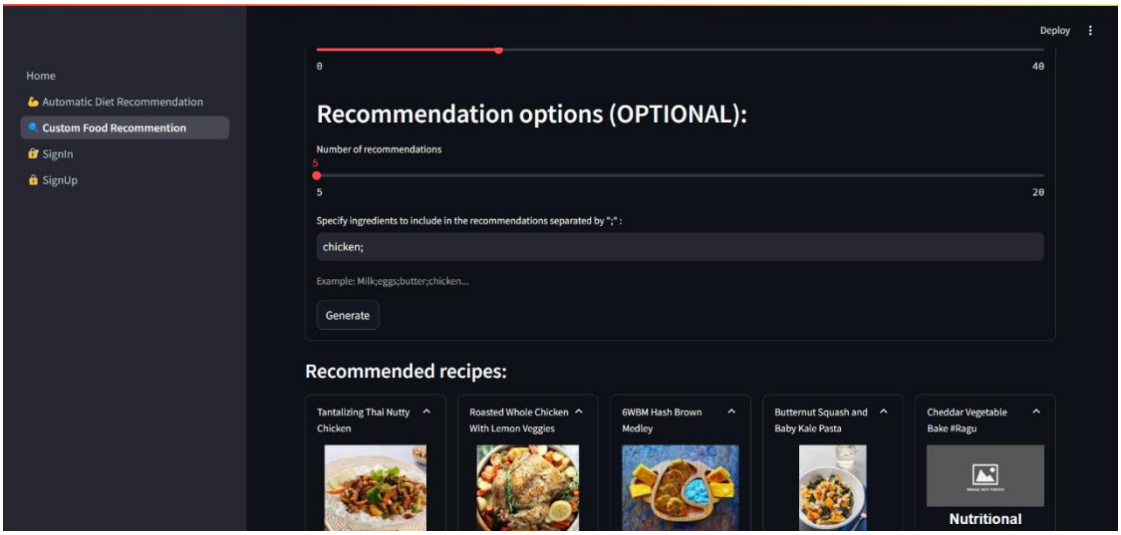


Figure 19: Custom Recommendation Output

The Silhouette Score is a measure of how well each data point fits within its assigned cluster compared to other clusters. It provides an evaluation of the quality of clustering by combining both cohesion (how close the points in a cluster are) and separation (how distinct the clusters are).

Formula:

For each data point i, the silhouette score $s(i)$ is calculated using the following formula:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

Figure 20: Silhouette Score

Where:

- a(i): The average distance between data point i and all other points in the same cluster(cohesion).
- b(i): The average distance between data point i and all points in the nearest cluster that i is not a part of (separation).

Interpretation:

- +1: A silhouette score near +1 indicates that the data point is well clustered (i.e., it is very close to other points in the same cluster and far from points in other clusters).

- 0: A score near 0 indicates that the data point is on or very near the decision boundary between two clusters.
- -1: A score near -1 indicates that the data point might have been assigned to the wrong cluster (i.e., it is closer to points in another cluster than to points in its own cluster).

We have done the testing of Agglomerative, DBSCAN, and KMeans and compare the datasets and scores are given.

1. **Agglomerative Clustering:**

Agglomerative Clustering is hierarchical, which means it builds the cluster tree by merging clusters based on a certain linkage criterion (like Ward, Complete, or Average).

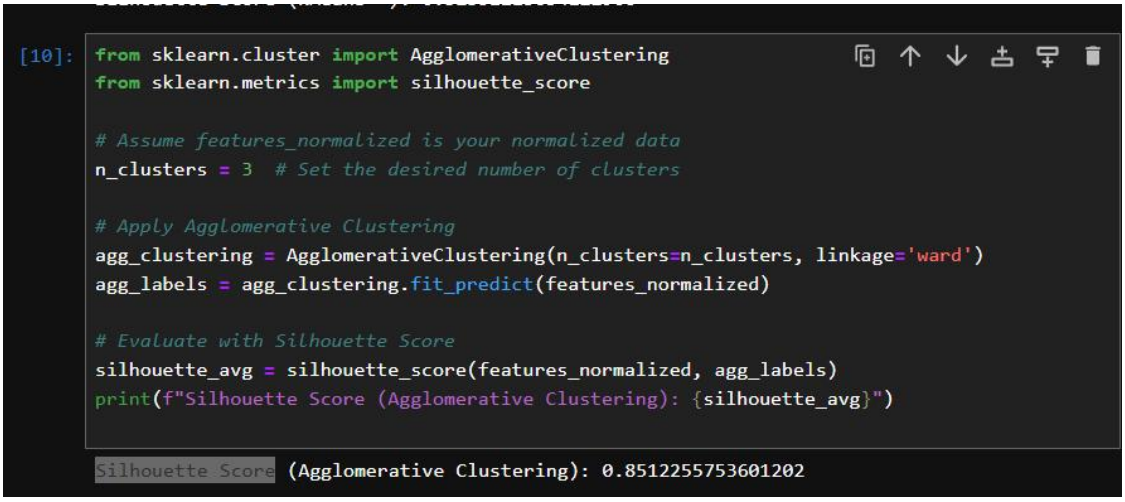


Figure 21: Silhouette Score of Agglomerative Clustering

A **Silhouette Score of 0.8512** for **Agglomerative Clustering** indicates that the clusters formed are quite well-separated and distinct.

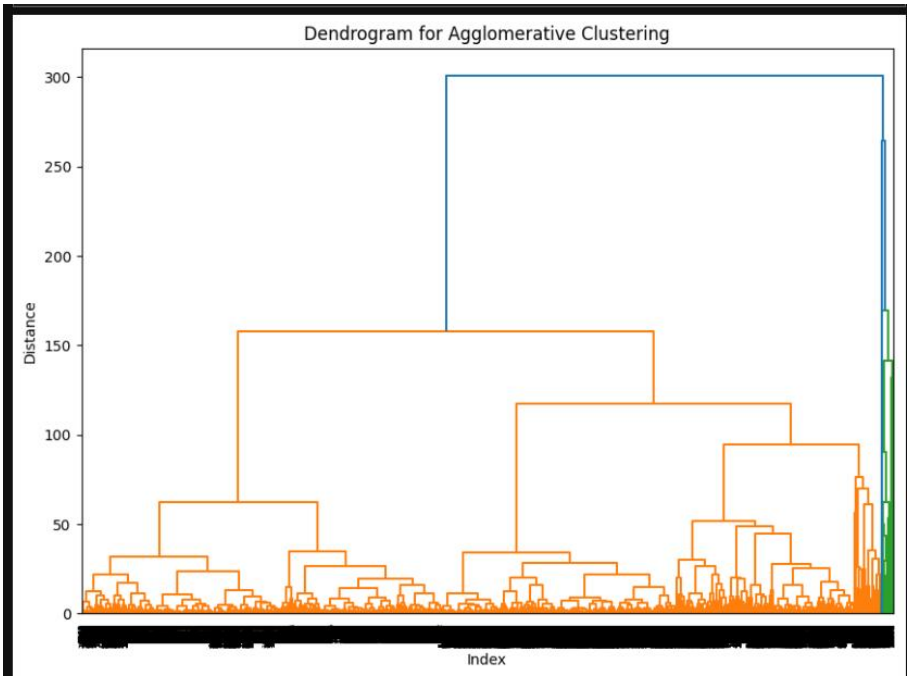


Figure 22: Dendrogram for Agglomerative Clustering

2. DBSCAN :

DBSCAN groups points that are close to each other based on a distance measurement and a minimum number of points in a neighborhood. DBSCAN can find arbitrarily shaped clusters and handle noise (outliers) better than other algorithms.

```
eps_values = [0.6, 0.8, 0.9]
min_samples_values = [4, 6]

for eps_value in eps_values:
    for min_samples_value in min_samples_values:
        dbscan = DBSCAN(eps=eps_value, min_samples=min_samples_value)
        labels = dbscan.fit_predict(features_normalized)
        non_noise_labels = labels != -1
        features_non_noise = features_normalized[non_noise_labels]
        labels_non_noise = labels[non_noise_labels]

        silhouette_avg = silhouette_score(features_non_noise, labels_non_noise)
        print(f"eps: {eps_value}, min_samples: {min_samples_value} -> Silhouette Score: {si
```

```
eps: 0.6, min_samples: 4 -> Silhouette Score: 0.18924998827717268
eps: 0.6, min_samples: 6 -> Silhouette Score: 0.256698811828446
eps: 0.8, min_samples: 4 -> Silhouette Score: 0.36890165311763884
eps: 0.8, min_samples: 6 -> Silhouette Score: 0.47356317547427823
eps: 0.9, min_samples: 4 -> Silhouette Score: 0.348007832016462
eps: 0.9, min_samples: 6 -> Silhouette Score: 0.5943348363764417
```

Figure 23:Silhouette Score of DBSCAN

A Silhouette Score of 0.594 for DBSCAN with parameters eps=0.9 and min_samples=6 suggests a relatively decent clustering performance, though not as strong as the

Agglomerative Clustering result of 0.8512.

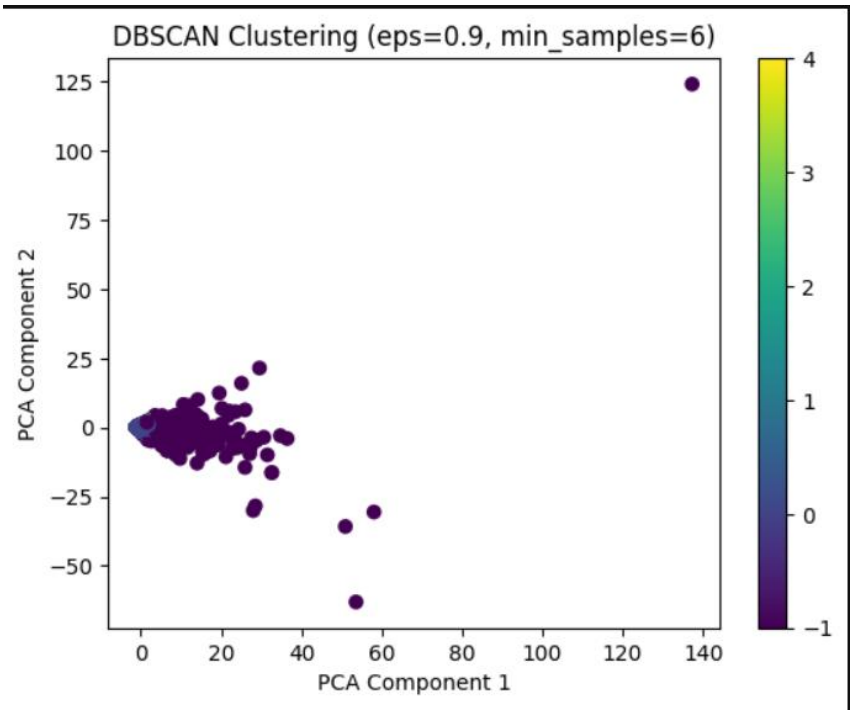


Figure 24: DBSCAN Clustering esp and sample figure

3. KMeans:

By minimizing the sum of squared distances between data points and the centroids of each cluster, KMeans clustering divides the data into kkk clusters.

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

# Define the number of clusters
k = 3  # You can change this based on your choice

# Apply KMeans clustering
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans_labels = kmeans.fit_predict(features_normalized)

# Calculate Silhouette Score to evaluate clustering quality
silhouette_avg = silhouette_score(features_normalized, kmeans_labels)
print(f"Silhouette Score (KMeans): {silhouette_avg}")
```

```
Silhouette Score (KMeans): 0.32351213634122966
```

Figure 25: Silhouette Score of Kmeans

A Silhouette Score of 0.32 for KMeans clustering suggests that the clustering performance is somewhat weak, as it indicates the clusters are not well-separated.

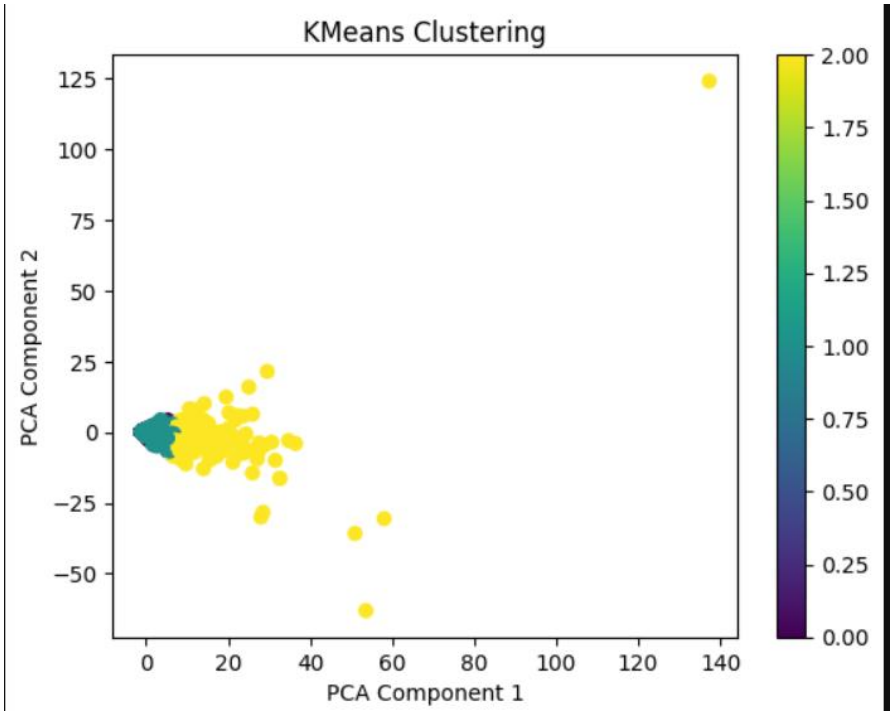


Figure 26: Kmeans Clustering

we compared the performance of four clustering algorithms—Agglomerative Clustering, DBSCAN, and KMeans—on a given dataset, using Silhouette Score as the evaluation metric.

- Agglomerative Clustering performed the best with a Silhouette Score of 0.8512, indicating well-separated and distinct clusters. This suggests that Agglomerative Clustering is highly suitable for this dataset, producing clear groupings with minimal overlap.
- DBSCAN with parameters eps=0.9 and min_samples=6 achieved a Silhouette Score of 0.594, which is a decent result. While the clusters formed are reasonably well-separated, there is still some room for improvement. The result suggests that DBSCAN is able to handle noise and varying densities but might need further parameter tuning to achieve optimal performance.
- KMeans underperformed with a Silhouette Score of 0.32, indicating weak cluster separation. This score suggests that the clusters formed by KMeans are not well-defined and there may be significant overlap between them. It highlights the importance of selecting the correct number of clusters (k) or considering alternative algorithms.

Chapter6: Conclusion and Future Recommendation

6.1 Conclusion

In conclusion, our diet recommendation system successfully addresses the need for personalized and automated meal planning. By allowing users to input data such as age, height, weight, gender, weight loss plan, and meals per day, the system generates tailored breakfast, lunch, and dinner suggestions. Additionally, custom diet recommendations based on specific nutritional requirements provide users with flexibility. The use of Streamlit for a user-friendly interface, FastAPI for efficient backend processing, and MongoDB for storing a comprehensive recipe database ensures system reliability and accuracy. With features like interactive slider controls and pie chart visualizations, the system effectively promotes healthy eating habits, helping users achieve their dietary goals conveniently.

6.2 Future Recommendation

Recommender systems in the field of diet planning have immense potential for growth. By leveraging advancements in technology and data science, we can further improve the system.

- **Use hybrid filtering recommendation:** To enhance the recommendation accuracy, the system can integrate hybrid techniques that combine collaborative filtering with content-based approaches. This would enable better recommendations by using user preferences along with the nutritional content of recipes.
- **Adding a User Dislike List:** Future improvements can include a feature where users can specify disliked foods or ingredients. This data can be used to refine recommendations, ensuring they align with user preferences.
- **Introducing Machine Learning:** Machine learning algorithms can be used to dynamically adjust recommendations by analyzing user feedback and identifying patterns. This would improve the relevance of suggestions over time.
- **Expanding the Recipe Database:** Increasing the diversity of recipes by incorporating more dietary options, cuisines, and regional specialties will make the system more inclusive.

References

[1] NVIDIA, "Recommendation System," NVIDIA, [Online]. Available: https://www.nvidia.com/en-us/glossary/recommendation-system/.

[2] IBM, "What is collaborative filtering?," [Online]. Available: https://www.ibm.com/think/topics/collaborative-filtering.

[3] G. P. &. S. Devi, "Hybrid Recommendation System Based on Collaborative and Content-Based Filtering," 2023. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/01969722.2022.2062544.

[4] O. K. Faisal Rehman, "Diet-Right: A Smart Food Recommendation System," Research Gate, 2017. [Online]. Available: https://www.researchgate.net/publication/313723698_Diet-Right_A_Smart_Food_Recommendation_System.

[5] V. S. T. M. R. Reema Golagana, "DIET RECOMMENDATION SYSTEM USING MACHINE LEARNING," *UGC Care Group I Journal,* Vols. Vol-13, 2023.

[6] M. I. S. J. ,. S. M. I. a. R. J. F. I. Weiqing Min, "Food Recommendation: Framework, Existing," Vols. IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 22, NO. 10, , 2020.

[7] A. A. Z. &. M. R. (. Khan, "The role of visualization tools in enhancing dietary adherence: A user-centric approach," *Journal of Nutritional Informatics,* 2020.

[8] T. P. M. &. H. L. (. Gordon, "Adaptive models in diet recommendation systems: Current trends and future directions.," *nternational Journal of Health Informatics,* 2020.

Appendix

#Index Page

```
st.set_page_config(
   page_title="Hello",
   page_icon="👋",
)

st.write("# Welcome to Diet Recommendation System! 👋")

st.sidebar.success("Select a recommendation app.")

st.markdown(
   """
   A diet recommendation web application using content-based approach with Scikit-Learn, FastAPI and Streamlit.
   You can find more details and the whole project on my
   """
)

# Check if user is logged in
if "token" in st.session_state:
   st.write("### Welcome to the Diet Recommendation System!")
   st.write("You are logged in!")
else:
   st.write("### Welcome to the Diet Recommendation System!")
   st.write("Please log in or sign up.")
```

#Display of Automatic Recommendation Code

```
def display_recommendation(self,person,recommendations):
        st.header('DIET RECOMMENDATOR')
        with st.spinner('Generating recommendations...'):
            meals=person.meals_calories_perc
            st.subheader('Recommended recipes:')
```

```
        formeal_name,column,recommendation in
zip(meals,st.columns(len(meals)),recommendations):
            with column:
                #st.markdown(f'<div
style="text-align: center;">{meal_name.upper()}</div>', unsafe_allow_html=True)
                st.markdown(f'##### {meal_name.upper()}')
                for recipe in recommendation:

                    recipe_name=recipe['Name']
                    expander = st.expander(recipe_name)
                    recipe_link=recipe['image_link']
                    recipe_img=f'<div><center><img
src={recipe_link} alt={recipe_name}></center></div>'
                    nutritions_df=pd.DataFrame({value:[recipe[value]]for value in
nutritions_values})

                    expander.markdown(recipe_img,unsafe_allow_html=True)
                    expander.markdown(f'<h5 style="text-align: center;font-family:sans-
serif;">Nutritional Values (g):</h5>', unsafe_allow_html=True)
                    expander.dataframe(nutritions_df)
                    expander.markdown(f'<h5 style="text-align:center;font-family:sans-
serif;">Ingredients:</h5>', unsafe_allow_html=True)
                    for ingredient in recipe['RecipeIngredientParts']:
                        expander.markdown(f"""
                                - {ingredient}
                        """)
                    expander.markdown(f'<h5style="text-align:center;font-family:sans-
serif;">Recipe Instructions:</h5>', unsafe_allow_html=True)
                    for instruction in recipe['RecipeInstructions']:
                        expander.markdown(f"""
                                - {instruction}
                        """)
                    expander.markdown(f'<h5style="text-align:center;font-family:sans-
```

```
serif;">Cooking and Preparation Time:</h5>', unsafe_allow_html=True)
                expander.markdown(f"""
                    - Cook Time      : {recipe['CookTime']}min
                    - Preparation Time: {recipe['PrepTime']}min
                    - Total Time     : {recipe['TotalTime']}min
                  """)
```

Display of Custom Recommendation Code:

```
def display_recommendation(self, recommendations):
    st.subheader('Recommended recipes:')
    if recommendations:
        rows = len(recommendations) // 5
        for column, row in zip(st.columns(5), range(5)):
            with column:
                for recipe in recommendations[rows * row : rows * (row + 1)]:
                    recipe_name = recipe['Name']
                    expander = st.expander(recipe_name)
                    recipe_link = recipe['image_link']
                    recipe_img = f'<div><center><img src={recipe_link} alt={recipe_name}></center></div>'
                    nutritions_df = pd.DataFrame({value: [recipe[value]] for value in nutrition_values})

                    expander.markdown(recipe_img, unsafe_allow_html=True)
                    expander.markdown(f'<h5 style="text-align: center;font-family:sans-serif;">Nutritional Values (g):</h5>', unsafe_allow_html=True)
                    expander.dataframe(nutritions_df)
                    expander.markdown(f'<h5 style="text-align: center;font-family:sans-serif;">Ingredients:</h5>', unsafe_allow_html=True)
                    for ingredient in recipe['RecipeIngredientParts']:
                        expander.markdown(f"- {ingredient}")
                    expander.markdown(f'<h5 style="text-align: center;font-family:sans-serif;">Recipe Instructions:</h5>', unsafe_allow_html=True)
                    for instruction in recipe['RecipeInstructions']:
```

```
                expander.markdown(f"- {instruction}")
            expander.markdown(f'<h5 style="text-align: center;font-family:sans-serif;">Cooking and Preparation Time:</h5>', unsafe_allow_html=True)
            expander.markdown(f"""
                - Cook Time      : {recipe['CookTime']} min
                - Preparation Time: {recipe['PrepTime']} min
                - Total Time     : {recipe['TotalTime']} min
            """)
    else:
        st.info("Couldn't find any recipes with the specified ingredients", icon="😕")
```

#Sign Up Register:

```
async def signup(user: UserSignup):
    existing_user = await db.users.find_one({"email": user.email})
    if existing_user:
        raise HTTPException(status_code=400, detail="Email already exists.")
    hashed_password = hash_password(user.password)
    user_data = {
        "email": user.email,
        "username": user.username,
        "password": hashed_password,
        "role": user.role,  # Store the role in the database
    }
    await db.users.insert_one(user_data)
    return {"message": "Account created successfully!"}
```

#Login Register:

```
async def login(form_data: OAuth2PasswordRequestForm = Depends()):
    user = await db.users.find_one({"email": form_data.username})
    if not user or not verify_password(form_data.password, user["password"]):
        raise HTTPException(status_code=401, detail="Invalid credentials")

    access_token = create_access_token(data={"sub": user["email"]}, expires_delta=timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES))
    return {"access_token": access_token, "token_type": "bearer"}
```