

Mobile Application For Diet Recall

Group ID: 20

Refer: <https://sih.gov.in/sih2022PS>

Group Member: Amay Jaiswal

Semester : 6th

Department: CS

Group Member: Shreya Bhadauriya

Semester : 6th

Department: CS

Group Member: Anshu Tomar

Semester : 6th

Department: CS

Group Member: Abhay Solanki

Semester : 6th

Department: CS

Name of Guide : Dr. Harsh Khatter

Date of Presentation : 2 May 2024

Project Objectives

Our 8 Major Objectives of our project are:-

1. **Diet Tracking:** Enable users to record and track their daily food intake, including meals, snacks, and beverages, in a user-friendly and efficient manner.
2. **Nutrient Analysis:** Provide users with detailed information about the nutritional content of their diet, including calories, macronutrients (carbohydrates, proteins, and fats), and micronutrients (vitamins and minerals).
3. **Reminder and Notification System:** Implement reminders and notifications to encourage users to log their meals regularly, stay on track with their goals, and provide feedback or suggestions based on their recorded data.
4. **Goal Setting and Tracking:** Allow users to set specific fitness goals (e.g., weight loss, muscle gain, improved cardiovascular endurance) and track their progress over time

Project Objectives

- 5. Workout Planning:** Facilitate the creation of customized workout plans by allowing users to schedule exercises, set rest intervals, and design routines tailored to their goals and time constraints.
- 6. User Support and Guidance:** Offer access to educational resources, workout tips, and guidance on proper nutrition, recovery, and injury prevention to support users in their fitness journey.
- 7. Meal Recommendations:** Provide users with meal recommendations based on their fitness goals, nutritional needs, and dietary preferences. These recommendations can take into account the user's workout schedule and the specific nutrient requirements for pre-workout and post-workout meals.
- 8. Recipe Database:** Develop a robust and diverse recipe database that includes healthy meal options suitable for various dietary preferences (e.g., vegetarian, vegan, gluten-free) and fitness goals (e.g., muscle building, weight loss).

METHODOLOGY

We are making a **Mobile application for diet recall** In which we will ask the user about his Age, Weight, Height and Activity level.

On the basis of that we will calculate the user daily maintenance calorie which will be done by a formula available by **Harris-Benedict**

Calories count formula BMR(Body to mass ratio)= $13.397W + 4.799H - 5.677A + 88.362$

Total maintenance calories = $BMR * 1.55$

Like if I suppose my weight is 70kg Height is 171cm and Age is 20 years then

$BMR = 13.397 \times 70 = 937 + 4.799 \times 171 = 820 - 5.677 \times 20 = 113 + 88.362 = 1732$

Total calories needed = $1732 \times 1.55 = 2684$ calories

Methodology- Using NumPy and Pandas

1.Importing Necessary Libraries:

- Pandas and NumPy libraries are imported to facilitate data manipulation and numerical operations.

2.Defining the **getfood** Function:

- A custom function named **getfood** is defined to generate food recommendations based on user inputs.

3.Reading the CSV File:

- The system reads a CSV file ("finalfoodis.csv") containing food data. This file serves as the primary data source for food recommendations.

4.Processing User Inputs:

- User inputs such as time of the day, body type, and disease list are processed. Time is standardized to numerical values for easier handling.

5.Filtering Food Recommendations:

- Based on the user inputs, the system filters food recommendations from the dataset. It considers factors like body type and any specified health conditions (e.g., diabetes, hypertension, alcoholism, smoking).

6.Returning Final Recommendations:

- After filtering, the system compiles a list of recommended foods meeting the user's criteria. This list is then returned as the final output of the **getfood** function.

KNN

1.Dataset Loading:

- The dataset ("finalfoodis.csv") containing food-related information is loaded into memory using the Pandas library. This dataset serves as the primary data source for food recommendations.

2.Data Filtering:

- Based on the user inputs (time, body type, and disease list), the dataset is filtered to include only relevant entries. Foods are filtered based on their suitability for the specified time, body type, and absence of any listed diseases.

3.Feature Preparation:

- Features for the K-Nearest Neighbors (KNN) algorithm are prepared from the filtered dataset. Non-feature columns such as 'food', 'time', and 'label' are excluded from the feature set.

4.Model Initialization and Fitting:

- A KNN model is initialized with a specified number of neighbors (in this case, 5). The model is then trained on the prepared feature set to learn the underlying patterns in the data.

5.Sampling a Food Item:

- To generate recommendations, a single food item is randomly sampled from the filtered dataset. This food item serves as the reference point for finding similar foods using the KNN algorithm.

6.Finding Nearest Neighbors:

- Using the trained KNN model, the nearest neighbors (foods) to the sampled food item are identified based on their feature similarity. The distances and indices of these nearest neighbors are computed.

7.Generating Recommendations:

- The food items corresponding to the indices of the nearest neighbors are retrieved from the filtered dataset. These foods are considered as the recommended options for the user, based on their similarity to the sampled food item.

8.Output:

- The recommended foods are returned as the output of the **get_food_recommendations** function, ready for presentation to the user.

METHODOLOGY

- After allowing the user to choose between Fat Loss or Muscle Gain, with these being the only two options available, let's say the user opts for Muscle Gain.
In that case, we will guide them to consume an additional 300 calories above their maintenance calorie level, which is set at 2500. This would mean a daily intake of 2800 calories.
- Following this, we aim to provide the user with a comprehensive diet plan tailored for muscle gain over a week, inclusive of detailed information on calories and macronutrients.
The plan will feature recipes for various dishes, accompanied by integrated YouTube videos demonstrating the preparation process.
Additionally, we will furnish the user with a customized workout plan for the entire week.
- To enhance user engagement, our app will send timely notifications, such as reminders to stay hydrated or dietary advice. This approach mirrors the notification system employed by popular applications like Swiggy and Zomato.

Machine Learning Algorithm for recommendation

The machine learning algorithm which we will use in our project is - **K nearest neighbor**

Content-based diet recommendation by K nearest neighbor-

- 1. User Profile Creation:** Collect user preferences, dietary restrictions, and goals. Consider factors like age, gender, activity level, and any specific health conditions.
- 2. Food Item Representation:** Represent each food item in a feature vector. Features can include nutritional content (calories, proteins, fats, carbohydrates, vitamins, etc.). Normalize the features to ensure equal weightage.
- 3. User Profile Representation:** Create a user profile vector based on their preferences and dietary requirements.
- 4. Similarity Calculation:** Use a similarity metric (cosine similarity, Euclidean distance) to measure the similarity between the user profile vector and the vectors of different food items.
- 5. K-Nearest Neighbors:** Identify the k-nearest neighbors with the highest similarity scores. These neighbors represent food items that are most similar to the user's preferences.
- 6. Recommendation Generation:** Aggregate the recommendations from the k-nearest neighbors. Rank the recommendations based on similarity scores.

Content-based recommendation system

In a content-based recommendation system for a diet app, the idea is to recommend food items to users based on the characteristics or "content" of the foods and the user's preferences.

- 1. Food Information:** Each food item is described by its nutritional content, such as calories, proteins, fats, and carbohydrates.
- 2. User Preferences:** Users provide information about their dietary preferences, restrictions, and goals. For example, they might indicate if they want low-calorie meals or are avoiding certain ingredients.
- 3. Matching Preferences:** The system compares the nutritional content of each food item with the user's preferences.
- 4. Similarity Calculation:** A similarity score is calculated to determine how closely the nutritional content of a food item aligns with the user's preferences.
- 5. Recommendation:** Food items with the highest similarity scores are recommended to the user. These are considered the closest match to what the user is looking for in terms of nutrition and dietary goals. For example, if a user prefers low-calorie meals and avoids high-fat foods, the system would recommend foods that are low in calories and have lower fat content based on the nutritional information. This approach focuses on the "content" or characteristics of the food items and tailors recommendations to match the user's individual preferences

Flow of the project

- 1. Data collection** - Data is collected by web scraping (projects work on demo data). Consider using libraries like http or web_scraper in Flutter.
- 2. Data processing** - Data is processed and required attributes are added to make demo datasets.
- 3. User's profile generation** - by taking input from them .
- 4. Initial recommendation** - on the basis of user's profile (Content-based, implemented by k-nearest neighbors).
- 5. Recommendation** - based on similar time bodytype and dislist.
- 6. Recommendation** - on the basis of user's past/recent activity (Collaborative Memory based approach, implemented by K-nearest neighbors).

USP Of Our Project

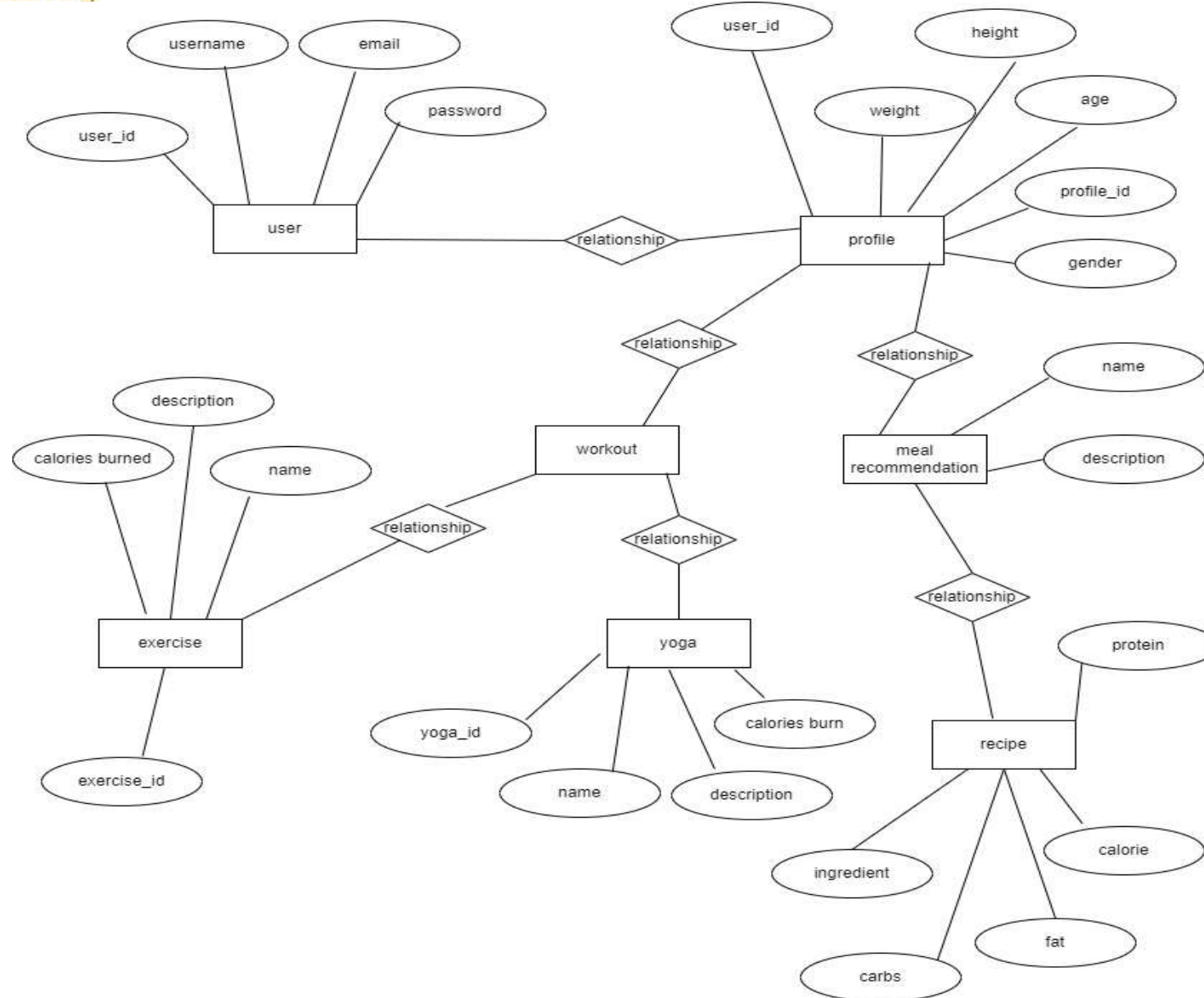
Integrating **personalized meal recommendations** based on the user's preferences, considering their cultural background and dietary restrictions.

Community Building - Users to connect with friends or a community. Users could share meal plans, exchange healthy recipes, and provide mutual support. This social engagement can foster a sense of accountability and motivation, making the app more than just a tracking tool.

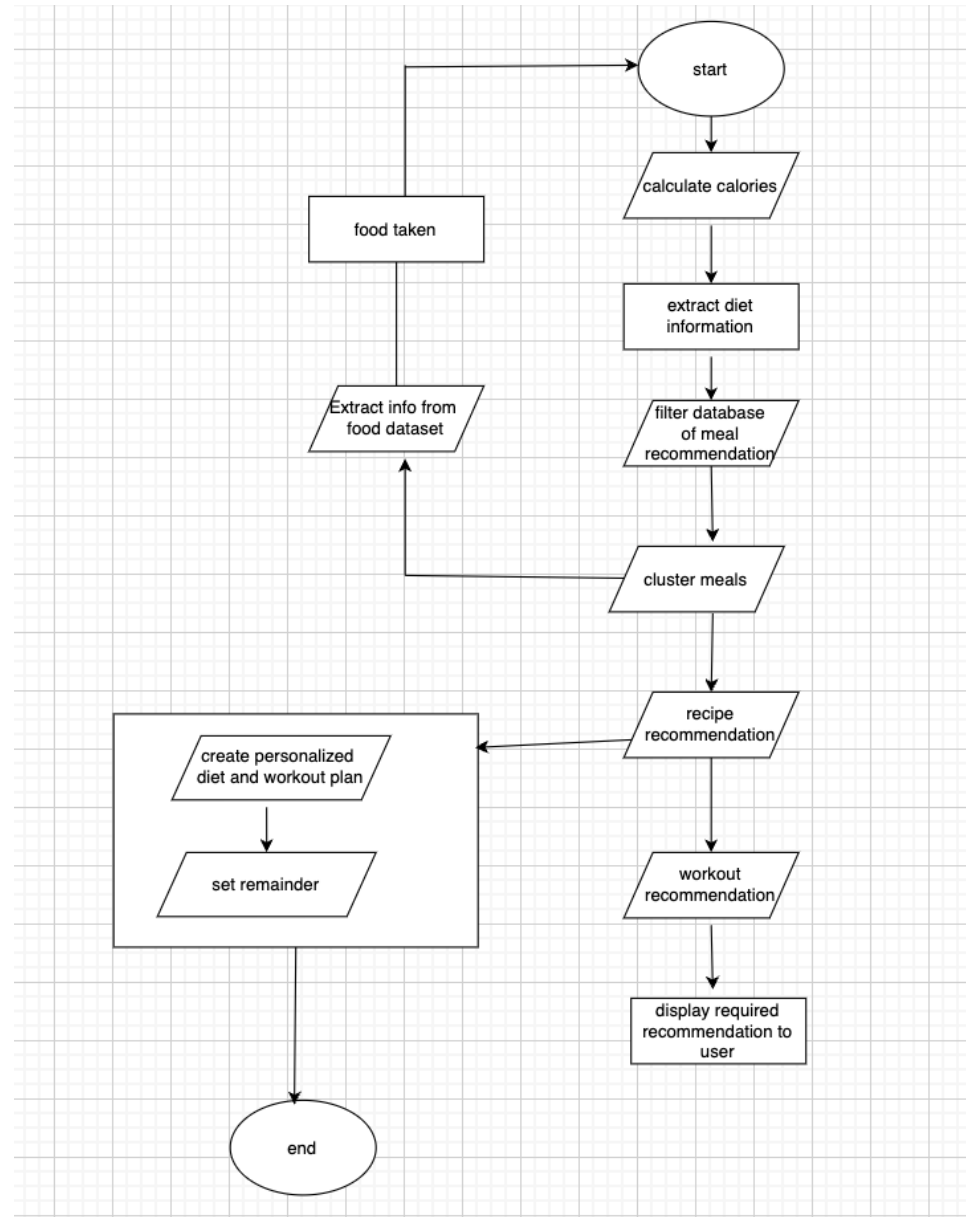
Integrating Gamification elements- Create challenges and achievements within the app to encourage users to consistently log their meals and follow their dietary goals. Rewarding users with badges, points, or even virtual gifts can make the experience more enjoyable and boost user engagement.

Trending recipes - Additionally, we intend to introduce a "**Trending Recipes**" section in our app, featuring the latest and most popular recipes. Users can explore and discover new culinary trends through this dedicated section.

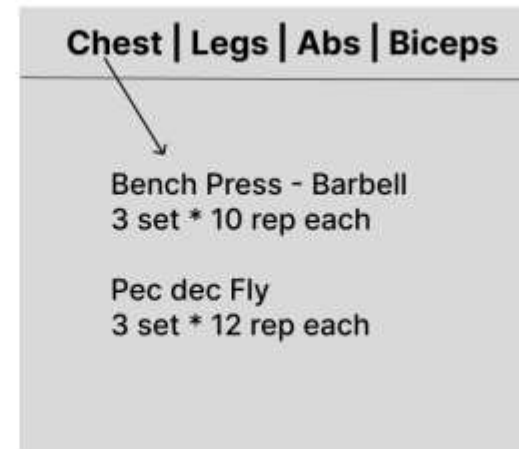
Entity Relationship Diagram



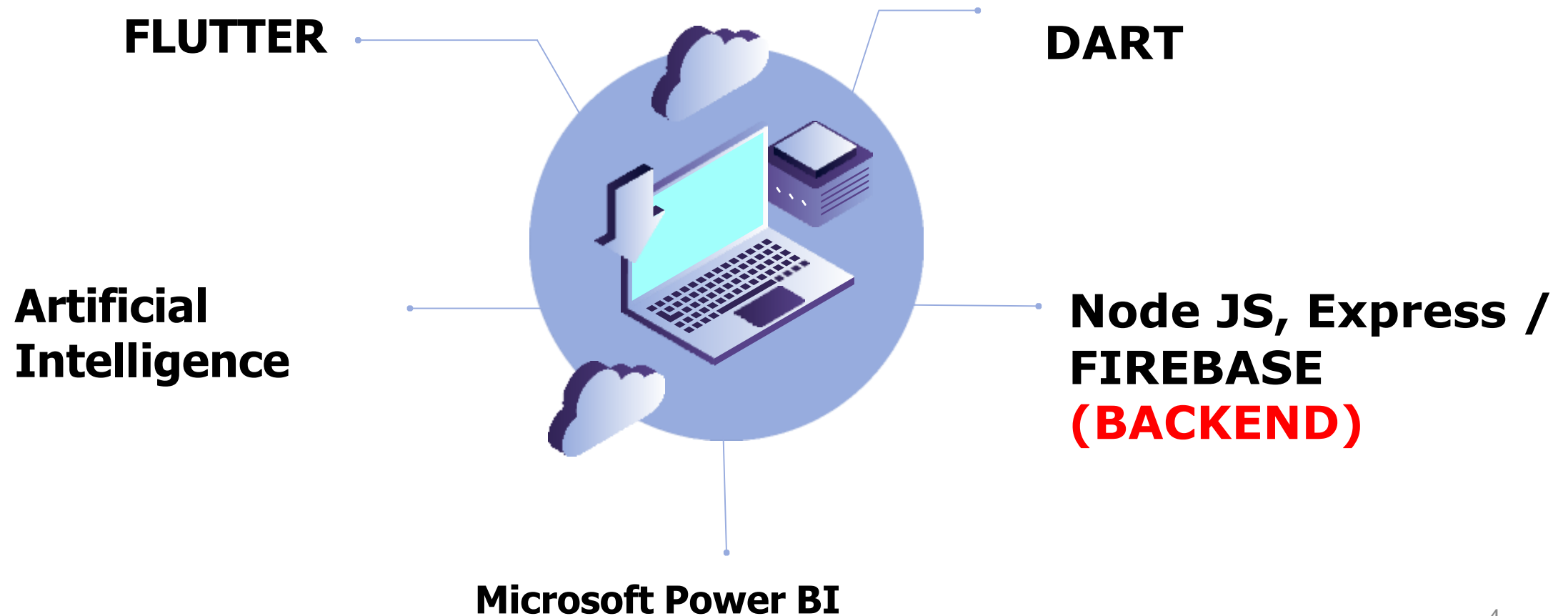
Flow Chart



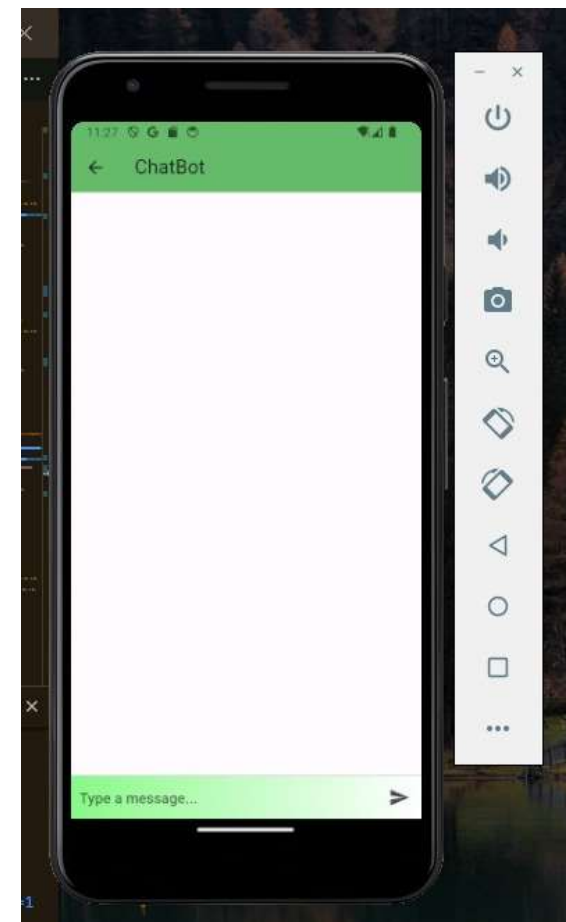
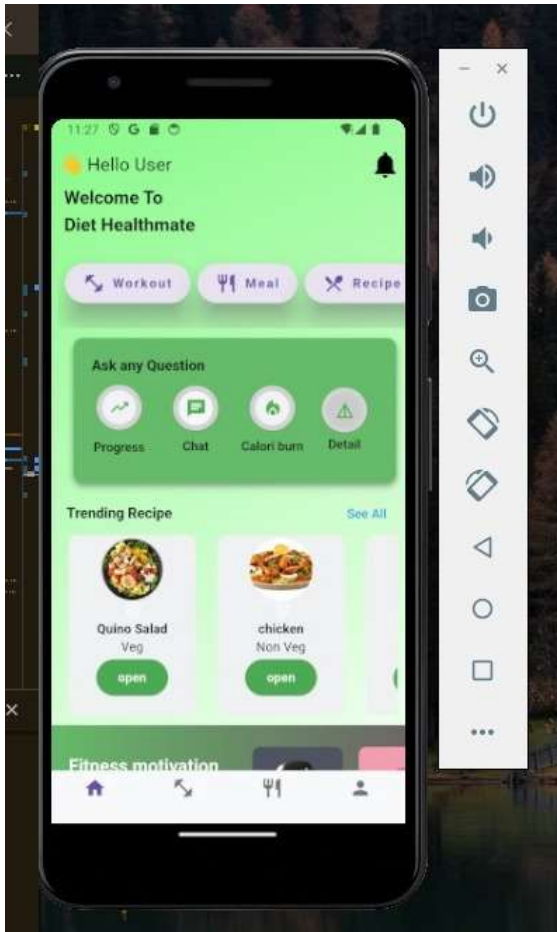
Layout of the Application



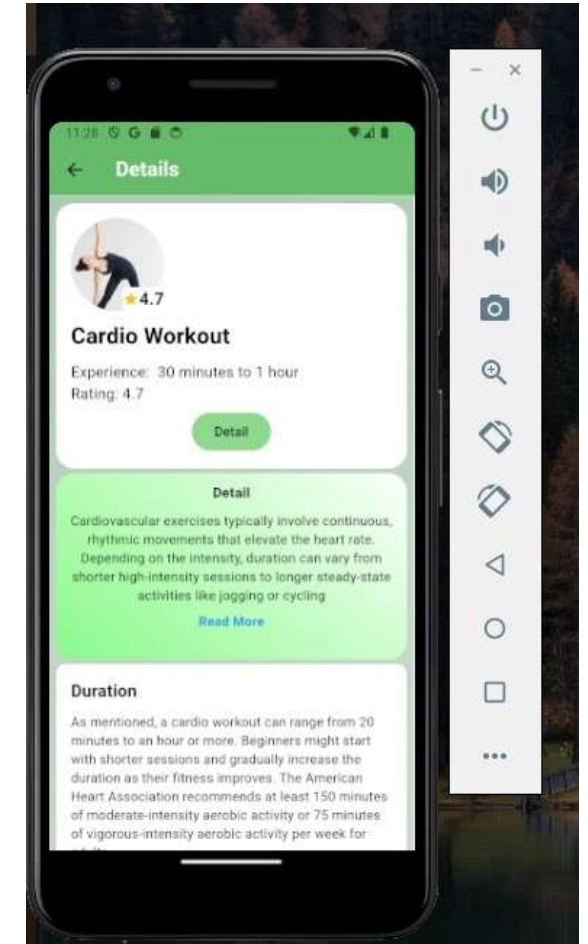
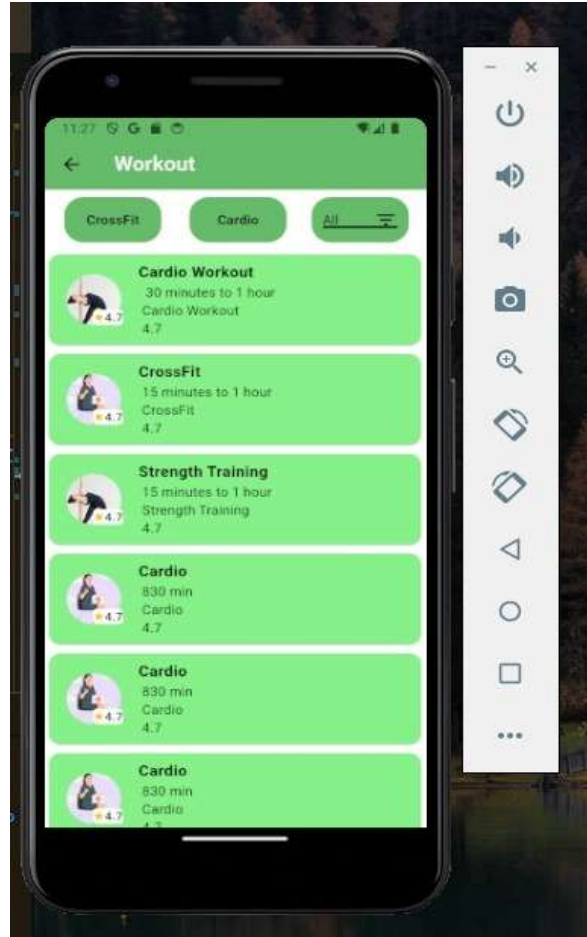
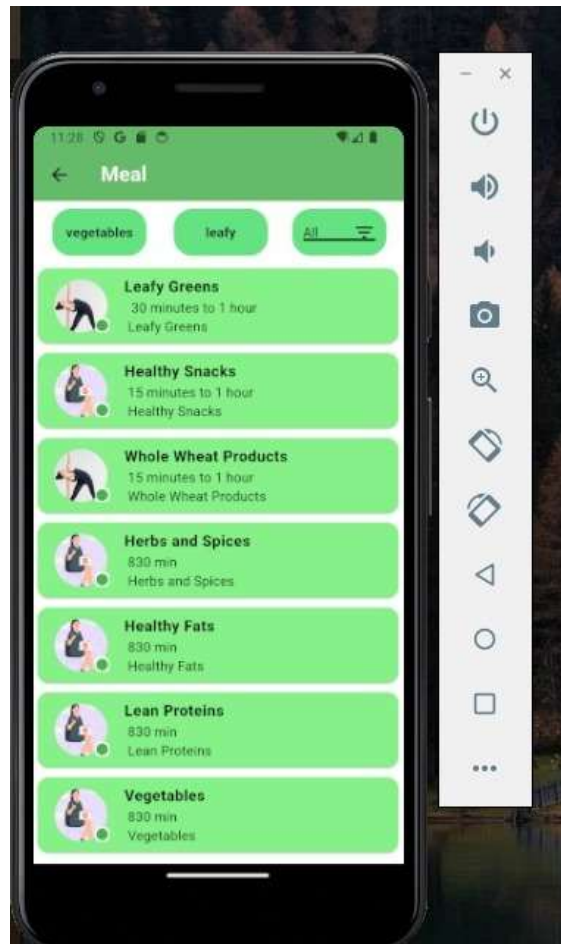
TECH STACK



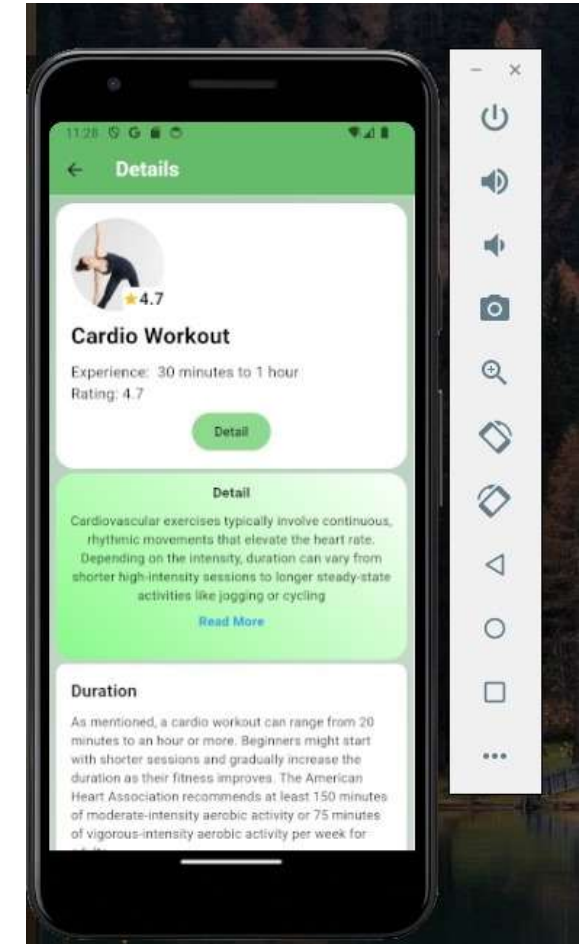
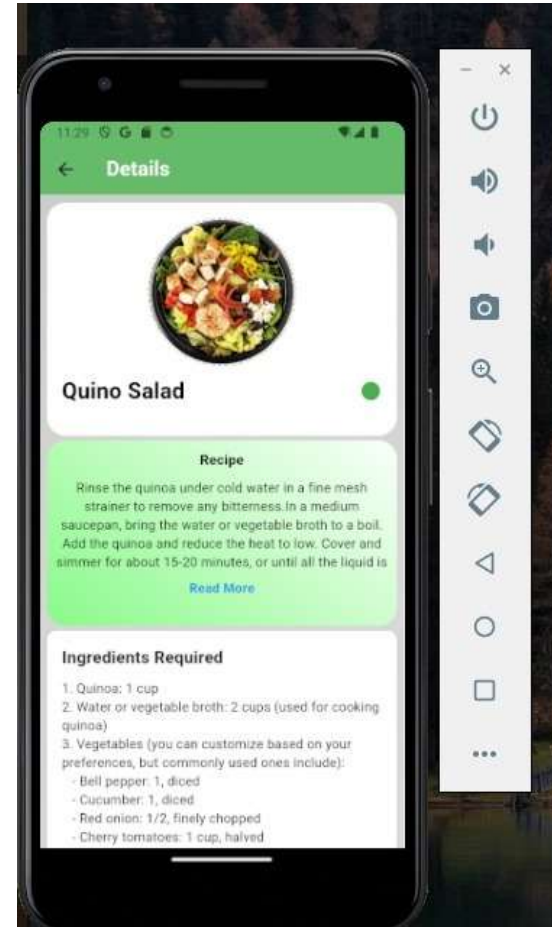
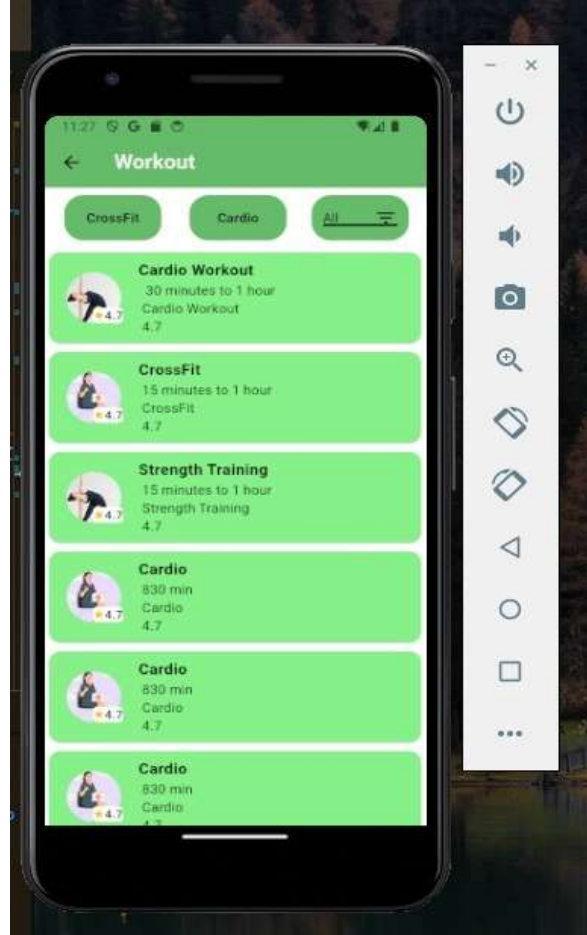
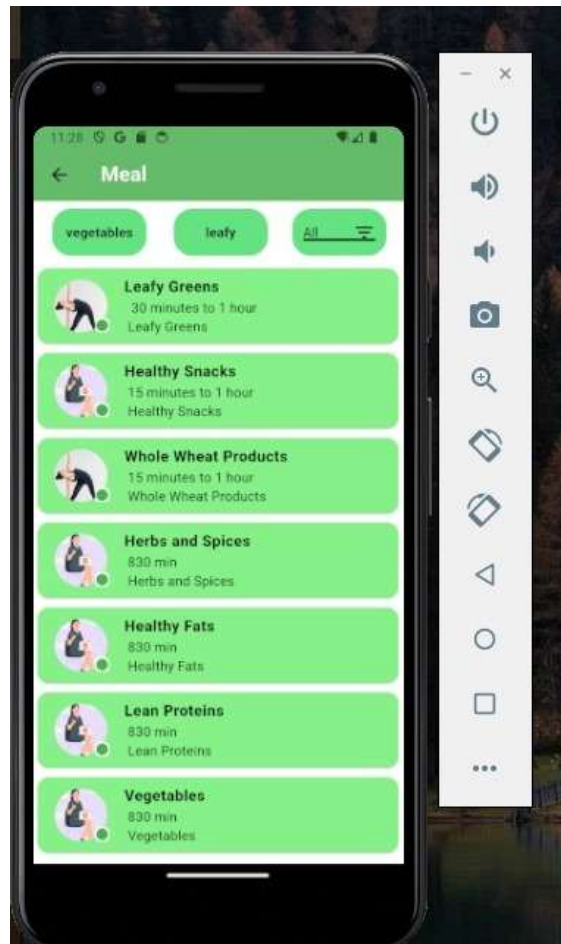
Frontend Implementation



Frontend Implementation



Frontend Implementation



Dataset for breakfast

```

1 // ignore: camel_case_types
2 class Breakfast_dataset {
3   getbreakfast() {
4     var _foodbreakfastlist = [
5       1: "soy dosa",
6       2: "ragi dosa",
7       3: "fenugreek dosa",
8       4: "herbal dosa",
9       5: "dosa",
10      6: "wheat upma",
11      7: "rice upma",
12      8: "kicokiri",
13      9: "egg whites",
14      10: "wheat dosa",
15      11: "brinjal dosa",
16      12: "ragi roti",
17      13: "barley roti",
18      14: "oats idli",
19      15: "oats upma",
20      16: "aloo paratha",
21      17: "dosa",
22      18: "sali par-eedu",
23      19: "appam",
24      20: "namkeen sevdyaan",
25      21: "indian spiced omelette",
26      22: "poha",
27      23: "aloo puri",
28      24: "poori bhajl",
29      25: "bread upma",
30      26: "rava idli",
31      27: "vada",
32      28: "rava pongal",
33      29: "oats uttapam",
34      30: "scrambled eggs",
35      31: "oats kheer",
36      32: "oats khokla",
37      33: "paneer",
38      34: "oats cheela",
39      35: "oats dosa"
    ];
  }
}
  
```

```

1 // ignore: camel_case_types
2 class Breakfast_dataset {
3   getbreakfastcal() {
4     var _foodbreakfastcalolist = [
5       1: 228,
6       2: 264,
7       3: 280,
8       4: 240,
9       5: 133,
10      6: 109,
11      7: 192,
12      8: 203,
13      9: 51,
14      10: 340,
15      11: 300,
16      12: 285,
17      13: 150,
18      14: 149.5,
19      15: 84.2,
20      16: 231.1,
21      17: 170.5,
22      18: 139,
23      19: 203.2,
24      20: 167,
25      21: 146,
26      22: 113.8,
27      23: 330.2,
28      24: 324.1,
29      25: 345.8,
30      26: 153.9,
31      27: 186.8,
32      28: 156.4,
33      29: 386.8,
34      30: 150.2,
35      31: 115.2,
36      32: 254.4,
37      33: 360,
38      34: 360,
39      35: 360
    ];
  }
}
  
```

```

1 // ignore: camel_case_types
2 class Breakfast_dataset {
3   getbreakfastpro() {
4     var _foodbreakfastprolist = [
5       1: 9,
6       2: 6.4,
7       3: 8.1,
8       4: 7.3,
9       5: 5.4,
10      6: 2.6,
11      7: 3,
12      8: 4.08,
13      9: 33,
14      10: 7.8,
15      11: 3,
16      12: 5.1,
17      13: 61.5,
18      14: 17.6,
19      15: 15.5,
20      16: 4.7,
21      17: 4,
22      18: 0,
23      19: 3.2,
24      20: 3,
25      21: 11,
26      22: 2.1,
27      23: 5.2,
28      24: 5.2,
29      25: 11,
30      26: 4.3,
31      27: 6.2,
32      28: 3.5,
33      29: 13.3,
34      30: 10.1,
35      31: 17.0,
36      32: 23.1,
37      33: 6,
38      34: 6,
39      35: 6
    ];
  }
}
  
```

```

1 // ignore: camel_case_types
2 class Breakfast_dataset {
3   getbreakfastcarb() {
4     var _foodbreakfastcarbolist = [
5       1: 27.3,
6       2: 36.2,
7       3: 6.48,
8       4: 18.1,
9       5: 35.6,
10      6: 18.8,
11      7: 30.7,
12      8: 58.1,
13      9: 6,
14      10: 18.4,
15      11: 141,
16      12: 66.16,
17      13: 111.6,
18      14: 98.4,
19      15: 81.6,
20      16: 34.6,
21      17: 30,
22      18: 35,
23      19: 36.1,
24      20: 10,
25      21: 6,
26      22: 24.9,
27      23: 28.6,
28      24: 28.2,
29      25: 63.8,
30      26: 29.6,
31      27: 16,
32      28: 26.4,
33      29: 68,
34      30: 1.6,
35      31: 83.1,
36      32: 95.5,
37      33: 20,
38      34: 23,
39      35: 70.2,
40      36: 70.2
    ];
  }
}
  
```

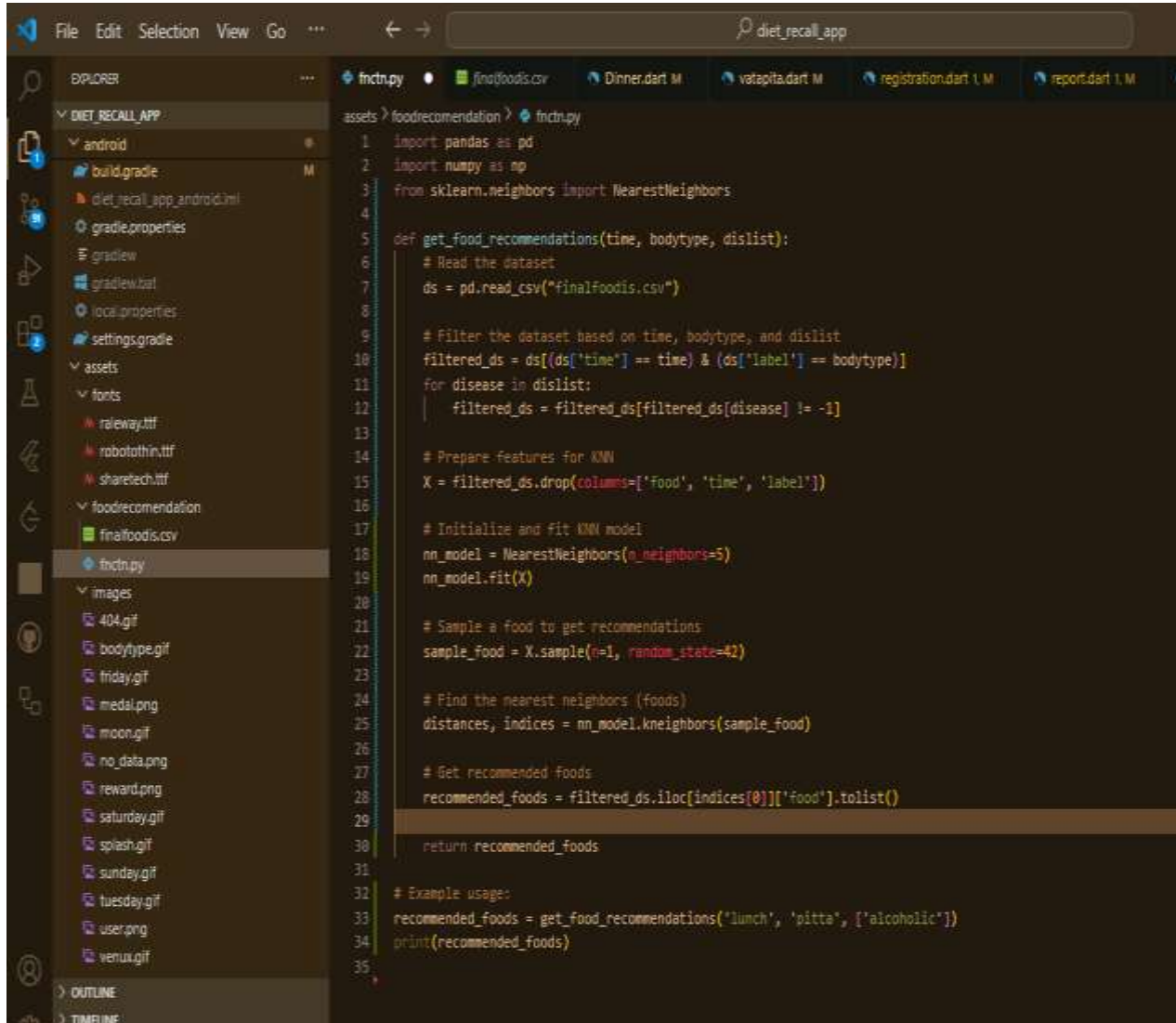

Recommendation System

```
def getFood(time, bodytype, dislist):
    j=0
    ds=pd.read_csv("finalfoodis.csv")
    foods=np.zeros((len(ds)),dtype='object')
    new_food=[]
    if(time=="breakfast"):
        time=0
    elif(time=="lunch"):
        time=1
    elif(time=="dinner"):
        time=2
    elif(time=="snack"):
        time=3
    if (dislist.count==0):
        for i in range(1,(len(ds))):
            if ds.loc[i,"label"]==bodytype and ds.loc[i,"time"]==time:
                foods[i]=str(ds.loc[i,"food"])
                #print(ds.loc[i,"food"])
    elif(dislist.count("Diabetes")!=0):
        for i in range(1,(len(ds))):
            if (ds.loc[i,"label"]==bodytype and ds.loc[i,"time"]==time and ds.loc[i,"Diabetes"]!=1):
                foods[i]=str(ds.loc[i,"food"])
    elif(dislist.count("hypertension")!=0):
        for i in range(1,(len(ds))):
            if ds.loc[i,"label"]==bodytype and ds.loc[i,"time"]==time and ds.loc[i,"hypertension"]!=1:
                foods[i]=str(ds.loc[i,"food"])
    elif(dislist.count("alcoholic")!=0):
        np.append(foods,(['coffee']))
        for i in range(1,(len(ds))):
            if ds.loc[i,"label"]==bodytype and ds.loc[i,"time"]==time and ds.loc[i,"fat"]>25:
                foods[i]=str(ds.loc[i,"food"])
    elif(dislist.count("smoker")):
```

```
        foods[i]=str(ds.loc[i,"food"])
    elif(dislist.count("alcoholic")!=0):
        np.append(foods,(['coffee']))
        for i in range(1,(len(ds))):
            if ds.loc[i,"label"]==bodytype and ds.loc[i,"time"]==time and ds.loc[i,"fat"]>25:
                foods[i]=str(ds.loc[i,"food"])
    elif(dislist.count("smoker")):
        if(time==3):
            np.append(foods,(['drumstick leaves spinach','radish leaves spinach','spinach','amaranthus leaves','fenugreek leav
            for i in range(1,(len(ds))):
                if ds.loc[i,"label"]==bodytype and ds.loc[i,"time"]==time:
                    foods[i]=str(ds.loc[i,"food"])
    print(len(foods))
    for i in range(0,(len(foods))):
        if foods[i]!=0:
            new_food.append(str(foods[i]))
    return new_food
print(getFood('lunch','pitta',(['alcoholic'])))
```

The method utilizes the pandas library, often used for data manipulation and analysis. The particular model being used here is a machine learning model developed by pandas for data manipulation

KNN



```
File Edit Selection View Go ... diet_recall_app
EXPLORER
DIET_RECALL_APP
  android
  build.gradle
  diet_recall_app_android.xml
  gradle.properties
  gradlew
  gradlew.bat
  local.properties
  settings.gradle
  assets
  fonts
  raweay.ttf
  robotothin.ttf
  sharetech.ttf
  foodrecommendation
  finalfoods.csv
  fnctn.py
  images
  404.gif
  bodytype.gif
  friday.gif
  medal.png
  moon.gif
  no_data.png
  reward.png
  saturday.gif
  splash.gif
  sunday.gif
  tuesday.gif
  user.png
  venue.gif
  OUTLINE
  TIMELINE
  assets > foodrecommendation > fnctn.py
1 import pandas as pd
2 import numpy as np
3 from sklearn.neighbors import NearestNeighbors
4
5 def get_food_recommendations(time, bodytype, dislist):
6     # Read the dataset
7     ds = pd.read_csv("finalfoods.csv")
8
9     # Filter the dataset based on time, bodytype, and dislist
10    filtered_ds = ds[(ds['time'] == time) & (ds['label'] == bodytype)]
11    for disease in dislist:
12        filtered_ds = filtered_ds[filtered_ds[disease] != -1]
13
14    # Prepare features for KNN
15    X = filtered_ds.drop(columns=['food', 'time', 'label'])
16
17    # Initialize and fit KNN model
18    nn_model = NearestNeighbors(n_neighbors=5)
19    nn_model.fit(X)
20
21    # Sample a food to get recommendations
22    sample_food = X.sample(n=1, random_state=42)
23
24    # Find the nearest neighbors (foods)
25    distances, indices = nn_model.kneighbors(sample_food)
26
27    # Get recommended foods
28    recommended_foods = filtered_ds.iloc[indices[0]]['food'].tolist()
29
30    return recommended_foods
31
32 # Example usage:
33 recommended_foods = get_food_recommendations('lunch', 'pitta', ['alcoholic'])
34 print(recommended_foods)
35
```

We filter the dataset based on the input parameters **time**, **bodytype**, and **dislist**. We use the features from the filtered dataset to train a KNN model.

We then sample a food from the filtered dataset and find its nearest neighbors using the KNN model. Finally, we return the recommended foods based on the nearest neighbors.

Conclusion With Result

In conclusion, our project is a success as it achieves its key objectives.

Users can easily track their daily food intake, get detailed insights into nutrition, and receive timely reminders for consistency.

Setting and tracking fitness goals are made simple, and customized workout plans cater to individual needs.

The support and guidance feature, along with a diverse recipe database, enhances the user experience.

It provides valuable information on nutrition, workouts, and offers meal recommendations, making it easier for users to adopt a healthier lifestyle.

Overall, our project bridges technology with personal well-being, empowering users to make informed choices for a healthier and more balanced life.

As we continue refining and improving these features, we aim to have a lasting positive impact on users' health and fitness journeys.

REFERENCES

1. <https://www.researchgate.net/figure>
2. <https://www.geeksforgeeks.org/>
3. Quora- <https://www.ncbi.nlm.nih.gov>
4. <https://nutritionalassessment.mumc.nl/en/nutritional-assessment>

**Thank
You**