Database Final Project Report

Group 14

A. Main idea

This project aims to develop a personalized nutrition recommendation website to help users make dietary choices based on specific nutritional goals.

The two primary purposes of this website are to allow users to track the nutrients they consume daily and to recommend foods based on their dietary needs for each meal.

By inputting today's food intake, users will receive tailored food suggestions to help them make healthier meal choices.

B. Data

1. Datasets: All primary keys are food (which means “food name”)
   i. Calorie:

food	Caloric_Value

   ii. Carbon_Sugar:

food	Carbohydrates	Sugars

   iii. Dietary_Fiber_Cholesterol:

food	Dietary_Fiber	Cholesterol

   iv. Mineral:

food	Sodium	Calcium	Copper	Iron
Manganese	Phosphorus	Potassium	Selenium	Zinc

   v. Fat:

food	Fat	Saturated_Fats	Monounsaturated_Fats	Polyunsaturated_Fats

   vi. Nutrition Density:

food	Nutrition_Density

   vii. Vitamin:

food	Vitamin_A	Vitamin_B1	Vitamin_B11	Vitamin_B12	Vitamin_B2
Vitamin_B3	Vitamin_B5	Vitamin_B6	Vitamin_C	Vitamin_D	Vitamin_E

   viii. Protein:

food	Protein

   ix. Water:

food	Water

2. Database: We use MySQL in this project.
   i. customers:

user_id	name	password	workout_frequency	weight	height	age

   - user_id: Automatically increase.
   - workout_frequency, weight, height, age: To calculate the value of BMI, BMR, and TDEE. These attributes are also used in giving suggestions.

   ii. foods: Combine all tables in datasets

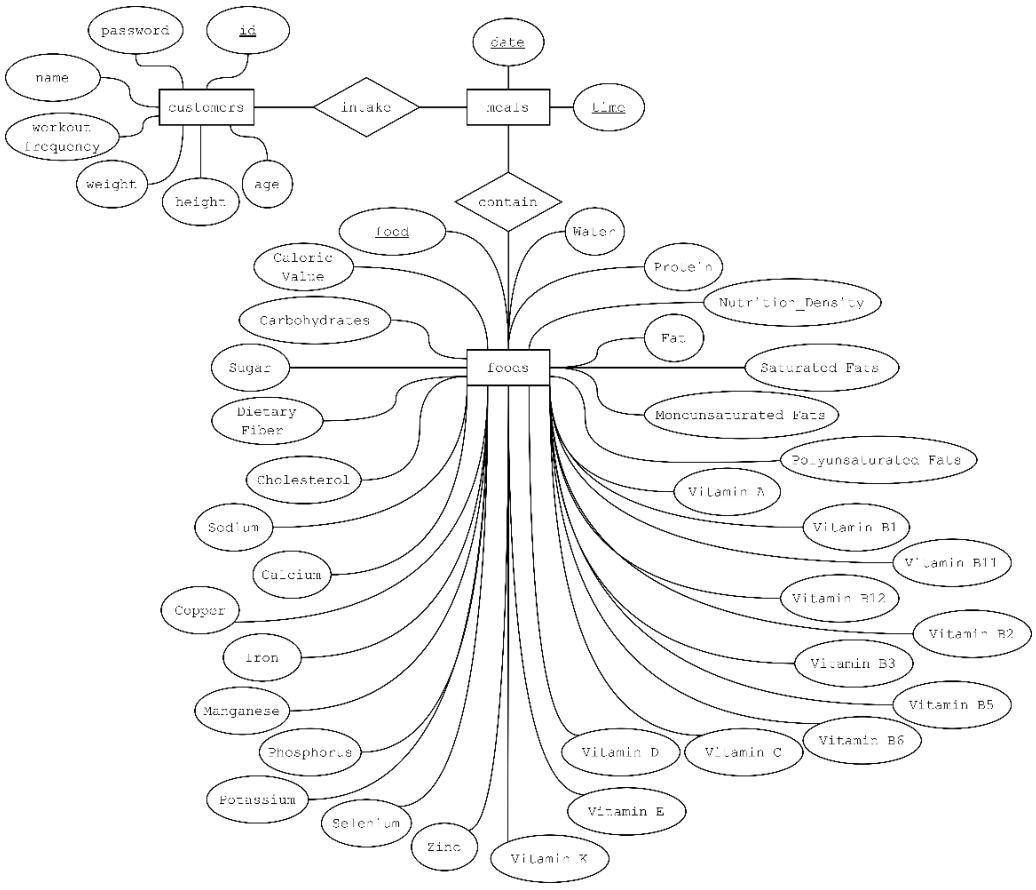
food	Caloric_Value	Carbohydrates	Sugars	Dietary_Fiber	Cholesterol
Fat	Saturated_Fats	Monounsaturated_Fats	Polyunsaturated_Fats		
Sodium	Calcium	Copper	Iron	Magnesium	Manganese	Phosphorus
Potassium	Selenium	Zinc	Nutrition_Density	Protein	Vitamin_A
Vitamin_B1	Vitamin_B2	Vitamin_B3	Vitamin_B5	Vitamin_B6	
Vitamin_B11	Vitamin_B12	Vitamin_C	Vitamin_D	Vitamin_E	Water

   iii. meals:

user_id	date	time	food

   - All attributes in this entity combine the primary key.

3. ER model:



C. Application

1. Sign up and Log in:

   This feature functions similarly to what was implemented in HW3, allowing users to create an account or log into an existing one.

   i. Log In:

   ```
   SELECT user_id, password FROM customers WHERE name = %s;
   ```

   'user_id' and 'password' are a set of keys. We achieve this function by confirming their correctness.

   ii. Sign Up: A CRUD function

   ```
   INSERT INTO customers (name, password) VALUES (%s, %s);
   ```

   Exception handling: If the user doesn't enter a name or password, the system will show the error message: "Please enter both username and password."

2. Your Profile:

   i. Displays User's Attributes:

   ```
   SELECT * FROM customers WHERE user_id = %s;
   ```

   Exception handling: If user doesn't enter some information, the default value is:

   ```
   "name": "", "weight": None, "height": None, "age": None,
   "workout_frequency": None, "bmi": "N/A", "bmr": "N/A",
   "tdee": "N/A"
   ```

   ii. Modify User's Attributes: A CRUD function

   ```
   INSERT INTO customers (user_id, name, weight, height, age,
   workout_frequency) VALUES (%s, %s, %s, %s, %s, %s)
   ON DUPLICATE KEY UPDATE
   name = VALUES (name), weight = VALUES (weight), height =
   VALUES (height), age = VALUES (age), workout_frequency =
   VALUES (workout_frequency);
   ```

   Because the value of the user's `user_id` must exist, we can use `ON DUPLICATE KEY UPDATE` to update the values in customers.

3. Food Information:

   i. Search for Foods:

   ```
   SELECT food FROM foods WHERE food LIKE %{%s}%;
   ```

   The system will list all food names that include the keyword, and the user can click on one of the options to look it up.

ii. Show Nutritional Information:

```
SELECT * FROM foods WHERE food = %s;
```

This function is just listing all attributes of the searched food.

4. Diet Record:
   i. Search for Foods: Same as the function in Food Information.
   ii. Log Food Intake: A CRUD function

```
INSERT INTO meals (user_id, date, time, food) VALUES (%s, %s, %s, %s);
```

Exception handling: If the user doesn't choose a date, the system will request the user to select one.

Exception handling: If the food has already been added, the system will display the error message: " 'error': 'Food already exists in the meal'."

   iii. Delete Food Intake: A CRUD function

```
DELETE FROM meals
WHERE user_id = %s AND date = %s AND time = %s AND food = %s;
```

Because the primary key of meals is (user_id, date, time, food), we should confirm all of the attributes before delete the data.

   iv. Show Records:

```
SELECT time, GROUP_CONCAT(food) AS foods FROM meals
WHERE user_id = %s AND date = %s
GROUP BY time;
```

We want to show the food names in one row, so we use `GROUP_CONCAT` to achieve this function.

   v. Show Caloric_value of added food:

```
SELECT Caloric_Value FROM foods WHERE food = %s
```

When the user hovers the cursor over an existing food item, its Caloric_value will be displayed.

5. Diet Suggestion:
   i. Provide Daily Statistics:

```
SELECT meals.time AS meal,
GROUP_CONCAT(meals.food) AS foods,
SUM(foods.Caloric_Value) AS calories,
SUM(foods.Protein) AS protein,
```

```
SUM(foods.Fat) AS fat,
SUM(foods.Carbohydrates) AS carbs
FROM meals
LEFT JOIN foods ON meals.food = foods.food
WHERE meals.user_id = %s AND meals.date = {date}
GROUP BY meals.time
ORDER BY FIELD (meals.time, 'Breakfast', 'Lunch', 'Snacks',
'Dinner', 'Late Night');
```

The table shows SUM of Caloric_Value, Protein, Fat, Carbohydrates every meals in one day. We use `SUM()` and `GROUP BY meals.time` to achieve this function.

ii. Provide Weekly Statistics:

```
SELECT DATE(meals.date) AS day,
SUM(foods.Caloric_Value) AS calories,
SUM(foods.Protein) AS protein,
SUM(foods.Fat) AS fat,
SUM(foods.Carbohydrates) AS carbs
FROM meals
LEFT JOIN foods ON meals.food = foods.food
WHERE meals.user_id = %s AND YEARWEEK(meals.date, 1) =
YEARWEEK(CURDATE(), 1)
GROUP BY DATE(meals.date);
```

The table shows SUM of Caloric_Value, Protein, Fat, Carbohydrates every meals in a week. We use `YEARWEEK(meals.date, 1) = YEARWEEK(CURDATE(), 1)` to achieve this. The rest are similar to "Provide Daily Statistics."

iii. Recommendation: No SQL.

Based on the existing data, provide some suggestions to the user, displaying text such as "Consider balancing your meals with more protein and reducing excessive carbs."

D. Progress and Problems:

- The order in which we completed the work is not far from what we had envisioned. After submitting the proposal, we assign tasks for all team members. First, we organized the data in the dataset and created a more detailed design for

the webpage, ensuring that the necessary data for each step is confirmed. Next, two groups were responsible for designing the ER model, writing SQL, and creating the webpage with Flask. Finally, we completed the process by preparing the report and video.

- The challenge we encountered was likely in the process of finding a suitable dataset. Datasets that met our requirements were very scarce, and even after finding one, coming up with a theme was another hurdle. This part troubled us for a long time, and even after submitting the proposal, we were still debating whether we needed to adjust the dataset.

E. Contribution:

Name	Build the system	Video	Report
曾歆喬	database and SQL	presentation slides	report
陳奕潔		presentation	
黃襄香	website and flask	demo	
陳宥臻			

F. Link:

- Repository: https://github.com/kellychen058/DBMS-final-project
- Video: https://youtu.be/3QupD0RrC7I