DataBases Fitness Tool

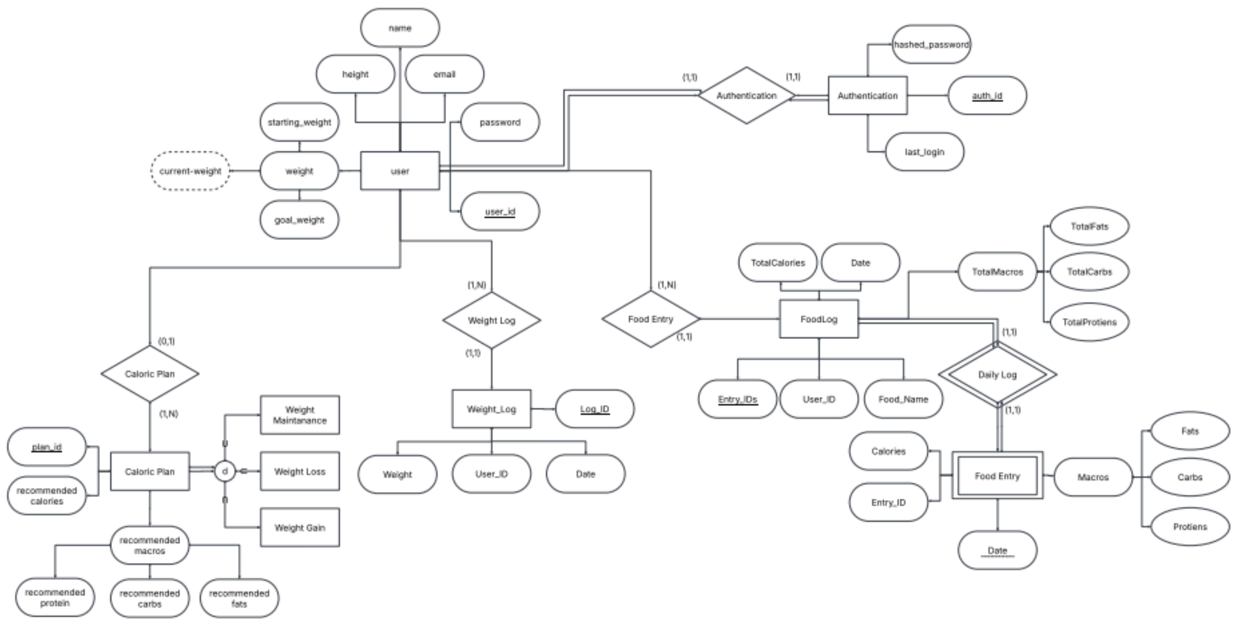
Group Members: Christopher Lu, Manas Maddi, Preetham Reddy Dodda

Problem Statement

America is home to many different people struggling to balance their lives with proper nutrition and weight maintenance. Some people fail to eat enough to keep a healthy weight while others eat far too much. Many people are used to eating junk food and fail to eat healthy nutrient-dense foods vital to the human body. While it is easy to tell people these facts about their lifestyle, it is very hard for them to make any meaningful changes as a healthy body is not just a one-time event, it is a lifetime of work and commitment. To help these people consistently recognize their eating habits and the effects on their health and weight, a form of accountability and a way to help them see the changes over time.

Our project idea is a calorie, weight, and macro counting tool. The purpose of this is to help those who are struggling to maintain a consistent eating lifestyle by providing them a way to keep them accountable and actually see what they are putting into their bodies daily as well as their weight effects. The app will be tracking weight, daily calories, and macros to be used to make weight trend graphs, calculate weekly calorie expenditures, and macro calculations. We will be using the Python framework, Flask, and we will be using SQL.

Conceptual Database Design



Assumptions:

Food entry: Each FoodEntry represents a single instance of a food being consumed. The calories of all foods entered compute the Total Calories in the Food Log.

Weak entity(Food Entry): Food Entries cannot exist without the Food Log entity, it is dependent on the amount of macros and calories.

Caloric Plan Assignment: Every user may have a caloric plan, and many different users can choose the same caloric plan. A caloric plan must be maintaining, gaining, or losing weight.

Macros entry: Macros can be recorded directly within the complex Macros attribute when entering a food item. Macros will then be added to the Food log total.

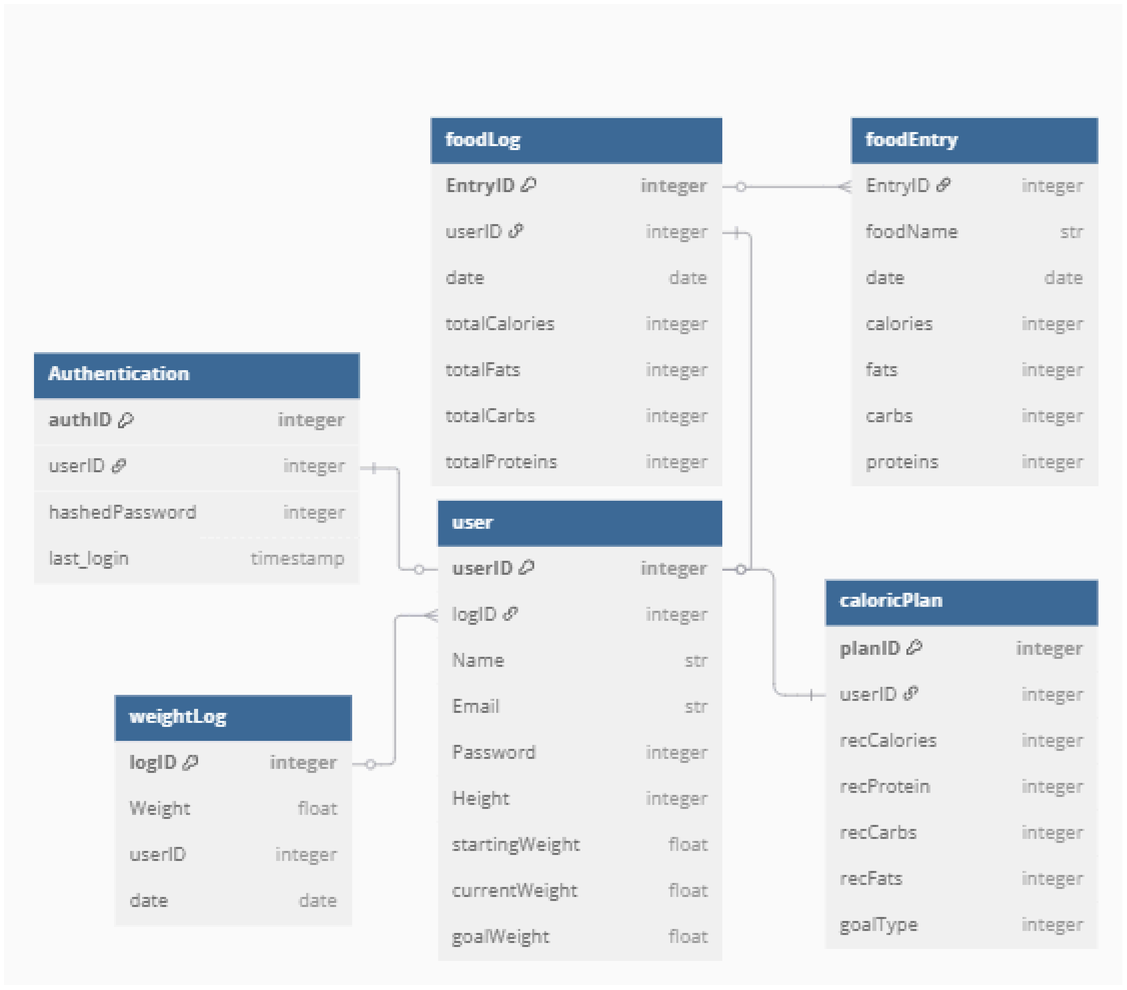
Authentication: Each user's authentication details are stored in only the Authentication entity. This also implies that each user has exactly one authentication record.

Derived Attribute(current-weight): When Weight is logged through Weight_Log, the derived attribute, current weight, will be updated.

Functional Requirements

1. Adding Foods: Users should be able to add foods to their day which will include calories and macros like protein, carbohydrates, and fats.
2. Editing and deleting: Users should be able to edit and delete foods from their day.
3. Graphs: The user should be able to view their weight trends and caloric intake through detailed graphs.
4. User profiles: Users should be able to create and manage their profiles, including information like their name, email, starting weight, weight change, caloric plan, and other statistical data.
5. Caloric plan: Users should be able to choose a plan whether it is to lose weight, gain weight, or maintain. This will change the calorie intake and macros suggested to them based on their strategy and targeted goal.
6. User Authentication: Users should be able to login and authenticate themselves.

Logical Database Design



Summary Table of Data Types

Table	Attribute	Type	Constraint
User	user_id	Int	Primary Key
User	LogID	Int	Foreign key
User	Name	Str	
User	Email	Str	
User	Height	Int	
User	Starting_Weight	Float	
User	Current_weight	Float	
User	Goal_Weight	Float	
WeightLog	LogID	Int	Primary Key
WeightLog	Weight	Float	
WeightLog	UserID	Float	
WeightLog	Date	date	
Authentication	AuthID	Int	Primary Key
Authentication	UserID	Int	Foreign Key
Authentication	HashedPassword	Int	
Authentication	Last_Login	timestamp	
FoodLog	EntryID	Int	Primary Key
FoodLog	UserId	int	Foreign Key
FoodLog	Date	date	
FoodLog	totalCalories	Int	
FoodLog	totalFats	Int	
FoodLog	totalCarbs	Int	
FoodLog	totalProtiens	Int	
FoodEntry	EntryID	Int	Foreign Key

Table	Attribute	Type	Constraint
FoodEntry	FoodName	Str	
FoodEntry	Date	date	
FoodEntry	Calories	Int	
FoodEntry	Fats	Int	
FoodEntry	Carbs	Int	
FoodEntry	Proteins	Int	
CaloricPlan	PlanID	Int	Primary Key
CaloricPlan	UserId	Int	Foreign Key
CaloricPlan	RecCalories	Int	
CaloricPlan	RecCarbs	Int	
CaloricPlan	RecFats	Int	
CaloricPlan	GoalType	Int	

Application Program Design

signup()

```
User = prompt for username
Password = prompt for password
confirmPassword = prompt for password

if(username already exists)
   Display “username already exists, choose a different one”
else
Display “Username taken”
password = hash(password) //Security for storing passwords
//write a query to write to store credentials in the User table
Display “Account created successfully.”
```

Login()

```
        User = prompt for user
        Password = prompt for password
        If (query authentication match in database):
                Logged_in = True
                display “Login successful”
        Else
                display “Invalid username/Password”
```

update_Profile

```
        if(logged_in = True)
                Name = prompt for name
                Email = prompt for email
                Starting_weight = prompt for starting weight
                Height = Prompt for Height
                Gender = prompt for gender
        Execute query to store values in UserProfile Table
        Display “Profile saved successfully.”
```

Set_Caloric_Plan

```
        if (logged_in = True)
                goal = prompt for goal(Lose , Gain , Maintain)
                goal_Weight = prompt for value(int)
```

```
            weight_change_per_week = set the value from the scrolling bar
        if (goal == "Lose")
        recommended_calories = Calculate_Deficit_Calories(weight_change_per_week)
      else if (goal == "Gain")
        recommended_calories = Calculate_Surplus_Calories(weight_change_per_week)
      else
        recommended_calories = Maintenance_Calories(user_name)
        Execute query to update CaloricPlan Table with goal and recommended calories
        Display “Caloric Plan updated)
```

Maintenance_calories(user_name)

```
            Execute query to fetch user details from the User Tablec
        bmr = 0
        If gender == "Male"
        bmr = 88.362 + (13.397 * weight) + (4.799 * height) - (5.677 * age)
        Else If gender == "Female"
        bmr = 447.593 + (9.247 * weight) + (3.098 * height) - (4.330 * age)
        /Return maintenance_calories
```

Calculate_Deficient_Calories(weight_change_per_week)

```
         deficient_calories = (weight_change_per_week * 3500) / 7
// 1 pound of fat is 3500 calories
        recommended_calories = Maintenance_Calories(user_name) - deficient_calories
        return recommended_calories
```

Calculate_Surplus_Calories(weight_change_per_week)

```
         surplus_calories = (weight_change_per_week * 3500) / 7
         recommended_calories = Maintenance_Calories(user_name) + surplus_calories
        return recommended_calories
```

Food Entry

Add_Food()

```
            Food = prompt for food name
            Calories = prompt for calories
            Protein = prompt for protein
            Fats = prompt for fats
            Carbs = prompt for carbs
            Date = prompt for date
        Execute query to insert food details into Food entry table
        Display “ Food added successfully.”
```

Edit_Food()

```
            if (logged_in == true)
```

```
                new_calories = prompt for new calories
                new_protein = prompt for new protein
                new_fats = prompt for new fats
                new_carbs = prompt for new carbs
        Execute query to update FoodEntry Table with new values
        display "Food entry updated"
```

Delete_Food()

```
    if (logged_in == true)
     Execute query to delete entry from FoodEntry Table
 display "Food entry deleted"
```

Get_Daily_Log()

```
  if (logged_in == true)
    date = prompt for date
    food_entries = Execute query to fetch all food entries for given date
    total_calories = SUM(food_entries.calories)
    total_protein = SUM(food_entries.protein)
    total_fats = SUM(food_entries.fats)
    total_carbs = SUM(food_entries.carbs)
    Display food log
```

Log_Weight()

```
  if (logged_in == true)
    date = prompt for date
    weight = prompt for weight
    Execute query to insert weight log into WeightLog Table
    Execute query to store current weight and date
    display "Weight logged successfully"
```

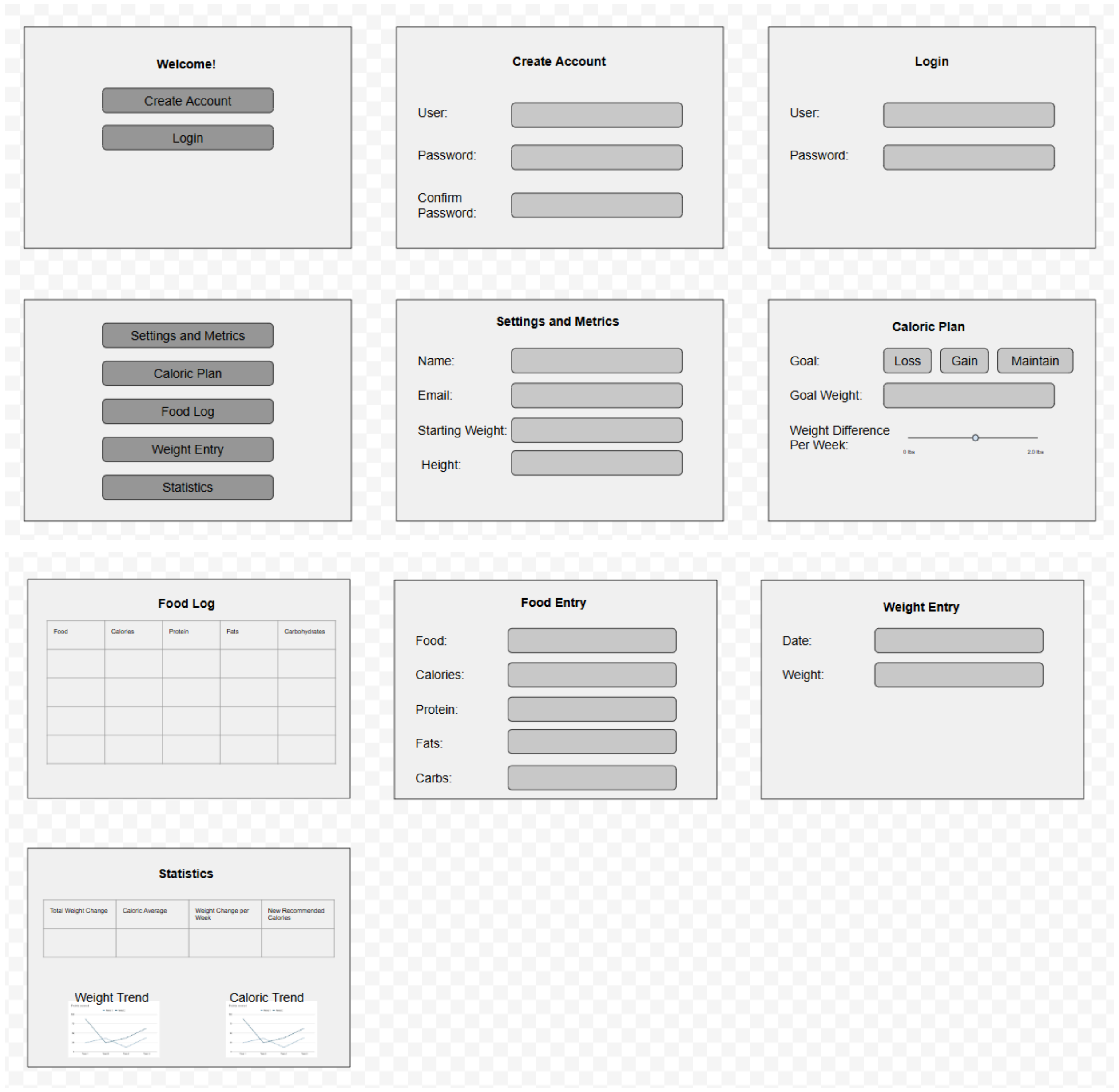
//Statistics

Weight_trends()

```
    starting_weight = Execute query to fetch user's starting weight
    current_weight = Execute query to fetch user's most recent weight in the log_weight
table
              total_weight_change = current_weight - starting_weight
              Display total_weight_change

    total_calories = Execute query to fetch total calories in over all logins
    total_days = Execute query to fetch total number of logged days
              caloric_average = total_calories / total_days
              Display  caloric_average

    first_log_date = Execute query to fetch first weight log date
              total_weeks = (current_date - first_log_date) / 7
              weight_change_per_week = total_weight_change / total_weeks
              Display weight_change_per_week

              if (weight_change_per_week > 0)
                new_recommended_calorie =
            Calculate_Surplus_Calories(weight_change_per_week)
              else
                new_recommended_calories = Maintenance_Calories(user_name)

              Execute query to update CaloricPlan Table with new_recommended_calories
              Display  new_recommended_calories
```

```
Weight_Graph()
    if(logged_in =true)
        Weight_data = Execute query to fetch weight logged in every day
        Use matplotlib and plot graph using Weight_data
        Display graph

Caloric_Trend_Graph()
    if(logged_in = True)
        Calorie_data = Execute Query to fetch daily calories from the table.
        Use matplotlib and plot graph using Calorie_data
        Display graph
```

Graphical User Design



Installation Guide

Intended Systems: Windows 10, 11; Mac OS

1. Clone this repository: https://github.com/manasmaddi/CS2300-Project.git

2. Create a virtual environment

   ```
   python3 -m venv venv
   source venv/bin/activate
   ```

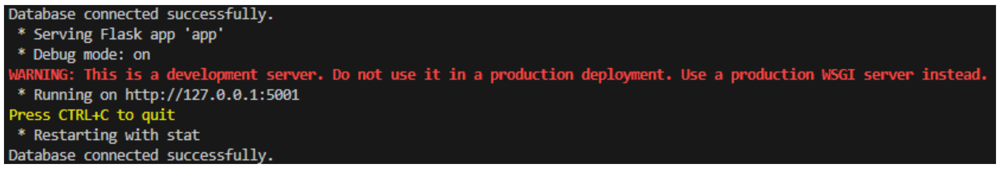
3. Within the cloned repository, run the following command in the terminal:

   ```
   pip install -r requirements.txt
   ```

4. Now we have the dependencies downloaded and the files to run the Fitness Application.

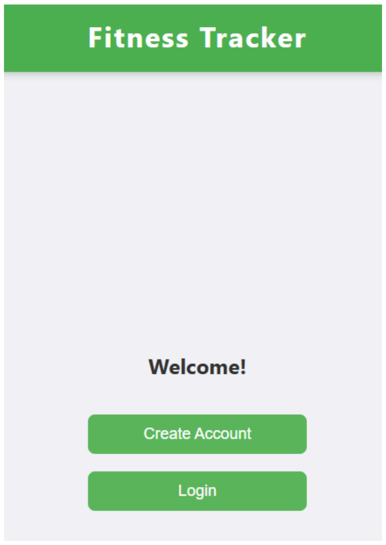
   In the terminal, run “python app.py”

5. The program will give you a local connection that will allow you to run the application through a web browser.



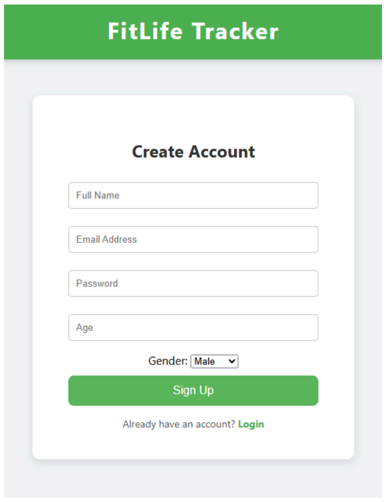
```
Database connected successfully.
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5001
Press CTRL+C to quit
 * Restarting with stat
Database connected successfully.
```

User Guide



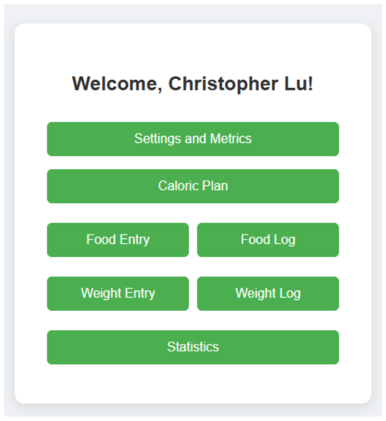
1. When you start the application you will want to create an account.

   Click “Create Account”

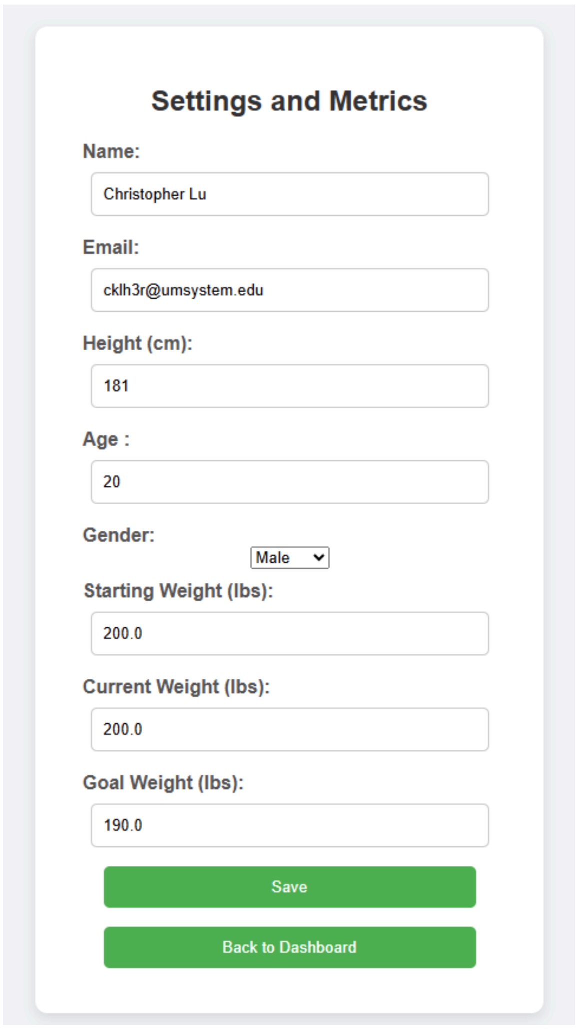


2. Now you will enter your name, email, password, age, and gender.
   If you have an account already, you can click login at the bottom.

   Click “Sign Up” when done



3. Once you have made an account, you will see your dashboard where you can access all of the Fitness applications features.



Settings and Metrics

Name:
Christopher Lu

Email:
cklh3r@umsystem.edu

Height (cm):
181

Age :
20

Gender:
Male

Starting Weight (lbs):
200.0

Current Weight (lbs):
200.0

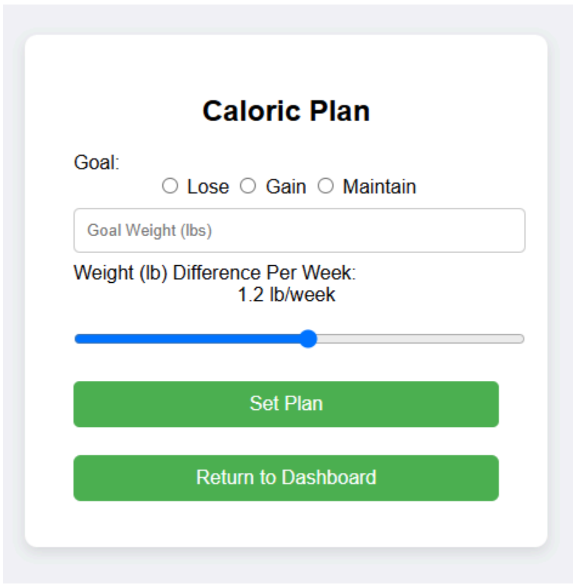
Goal Weight (lbs):
190.0

Settings and Metrics

This is where you change your account information and your body metrics.

The following features will require you to have accurate Metrics of height, age, gender, and weights in order to properly work.

Click “Save” when done updating



Caloric Plan

Goal:
○ Lose ○ Gain ○ Maintain

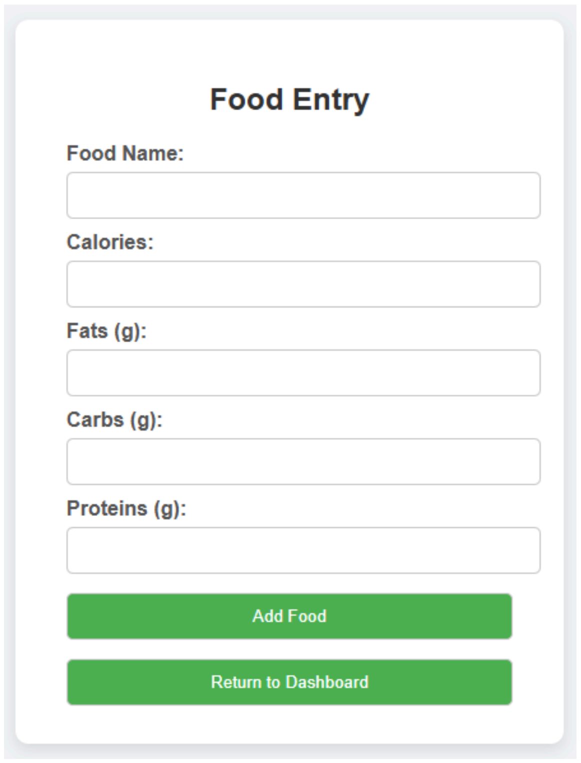
Goal Weight (lbs)

Weight (lb) Difference Per Week:
1.2 lb/week

Caloric Plan

Now we have set our metrics, we can set our goals with a caloric plan. You can select 3 general goals as well as a goal weight to reach, then you will select the rate at which you would like to change.

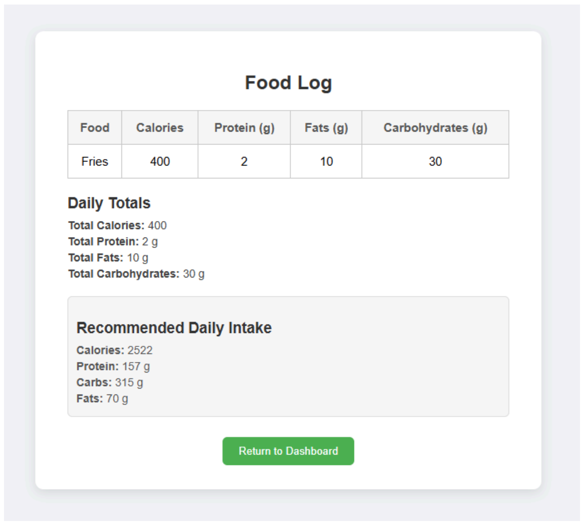
Click “Set Plan” when finished



Food Entry

Here you can add food you eat throughout the day. A food name and calories are required, but macros are not required to be inputted.

When done click “Add Food”



Food Log

Food	Calories	Protein (g)	Fats (g)	Carbohydrates (g)
Fries	400	2	10	30

Daily Totals

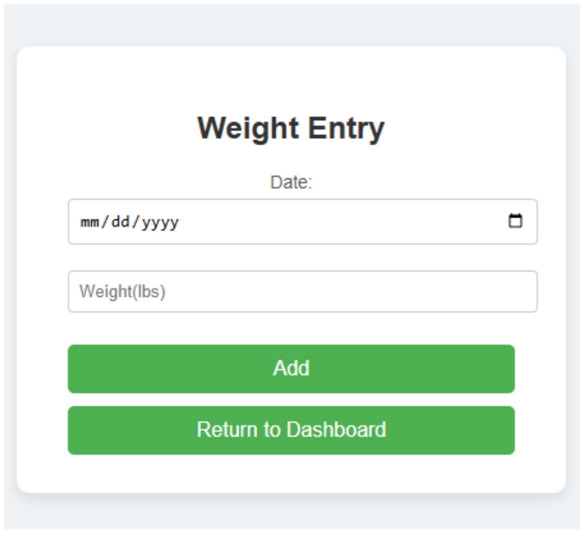
Total Calories: 400
Total Protein: 2 g
Total Fats: 10 g
Total Carbohydrates: 30 g

Recommended Daily Intake

Calories: 2522
Protein: 157 g
Carbs: 315 g
Fats: 70 g

Food Log

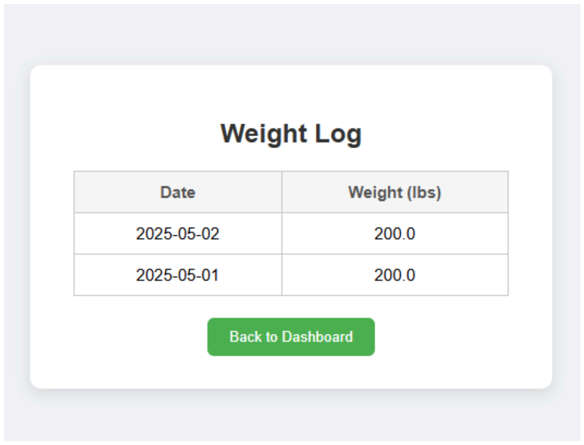
Now when you click on the food log within the dashboard, you will see your food for the day along with the total calories and macros. You will also see the recommended daily intake based on your caloric plan.



Weight Entry

This is where you will enter your weight which will allow the app to give you statistics and better calculate your daily intake to fit your goals.

Click “Add” when done



Weight Log

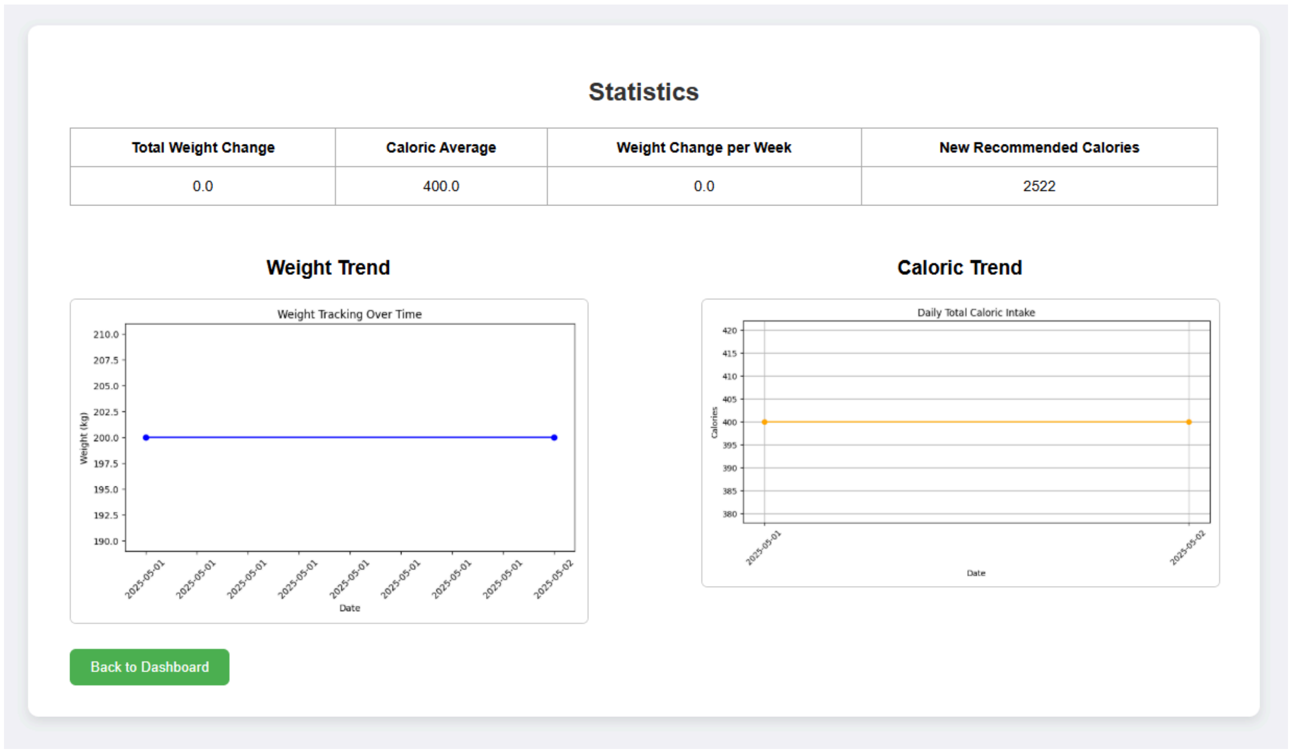
Date	Weight (lbs)
2025-05-02	200.0
2025-05-01	200.0

Weight Log

This is where you can see all of your weight entries along with the date of each entry.

Statistics

Now that we have put entries in, we can see them in nice graphs as well as statistics that can help us see our progress.



Statistics

Total Weight Change	Caloric Average	Weight Change per Week	New Recommended Calories
0.0	400.0	0.0	2522

Weight Trend

Caloric Trend