Name : Arnav Jamidar
Candidate Number : 7109
Centre Number : 25206

Project Title : FitMetrix

Analysis

The Problem

As per fitness enthusiasts and doctors, there is a growing number of people looking to improve their physical fitness, and achieve the physique they've always desired.

But most people have to do some preparation and planning before starting their fitness transformation, this requires calculations such as counting one's maintenance calories, body mass index, body analysis statistics, and amount of hours one should spend in the gym.

The common problem fitness enthusiasts have is that there is no website/software available for the public to access and gain these stats in one place.

My program will solve this problem, as it will have all the necessary information required for one to create and start their fitness program as well as the ability to track their progress via visual representation of data that is easy to read and understand.

Healthverse will provide the user with the ability to calculate their caloric requirements depending on their final goal, and they would be able to choose an appropriate workout and diet plan which will be tailored to their fitness goals.

Also, this program will include the option to modify one's diet plan based on their dietary requirements. All of these features will be accessible in the same program, which solves the problem mentioned above.

Interview

I held an interview with a friend, Pete, who is actively involved in fitness and is midway through his fitness transformation journey.

Q1) How was your experience while researching for a plan to begin your quest into fitness?
Ans: It wasn't a pleasant experience, I had to go through a lot of websites just to gather basic information regarding my diet/workout plan.

Q2) How could a beginner's experience while researching for a plan be made smoother and better?
Ans: In my opinion, a beginner's experience would be way smoother if all of the necessary information regarding diet/workout plans could be found in one app or website.

Q3) What is one of the biggest challenges while looking for a fitness-assistant app for a fitness enthusiast?
Ans: The biggest challenge is the cost, most apps/websites that provide good features cost a lot of money by charging for monthly subscriptions.

Q4) Are you satisfied with the website/app you use for fitness currently?
Ans: No, I am not satisfied with the app I use currently as it does not provide a way to compare my body's statistics with the average statistics, which indicates whether I am on the right track or not.

Q5) What is one feature that you would like to see in fitness apps in the future?
Ans: I would say the feature to look up the calorie contents for different food items must be included in fitness apps, as it is the feature that is required most frequently by fitness enthusiasts like me.

Analysis of Interview

The interview with Pete made my vision towards this program more firm, as many of the problems stated by Pete match the problems pointed out by me earlier on in the analysis. Also, the suggestions given by Pete are the same as some of the features my program offers which are mentioned in the earlier on in the analysis.

The first question shows how fitness enthusiasts who are seeking advice to begin their transformation journey are still struggling with problems related to the fitness programs currently being used by most people in the modern world. This tells us that there is a need for a fitness program which can provide a smooth user experience for fitness enthusiasts to start and maintain their fitness transformation.

The second question suggests how having information about diet plans and workout plans being included in one program would solve a lot of the problems existing fitness systems present. My program will solve this problem as information about diet plans and workout plans is included in the same program.

The third question highlights the fact that presently a lot of the fitness systems charge their users a lot of money to access their services, this can lead to many people quitting their fitness journey due to financial pressure and therefore also demotivating anyone who is looking for a fitness program to start working on their fitness.

The fourth question provides evidence that fitness enthusiasts also look to assure the progress they are making is in the right direction, by comparing their statistics by information such as national averages. Therefore, my program will solve this problem by providing the users an option to compare their fitness statistics with national

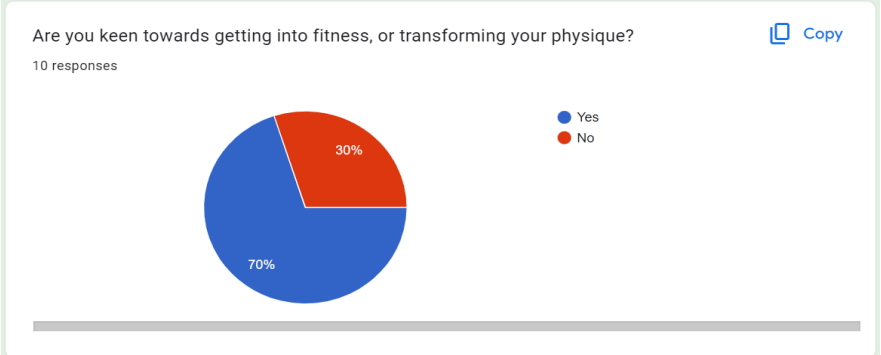
averages and the average of all the people in the database of the system using visual representation of data via bar graphs.

The fifth question gives a very useful suggestion towards designing my system, Pete's answer suggests that adding a calorie count for all the foods in a diet plan can be really useful of all the people using the system as reduces the effort put in one to search the calorie count of each item they eat as part of their diet plan. My program will include the calorie count for each food item in every diet plan as well as the nutritional information about all the diet plans such as total calories, protein, carbohydrates and fats.

Questionnaire

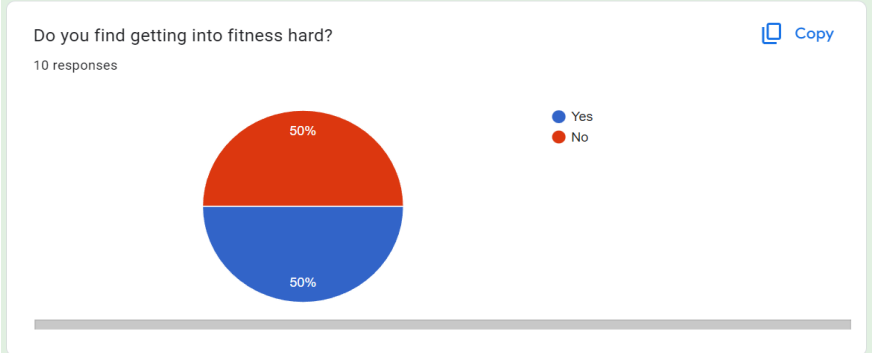
I held a survey in the form of a questionnaire, where all the students in my Computer Science class were invited to participate, the results are evaluated below.

Q1) Are you keen towards getting into fitness, or transforming your physique?



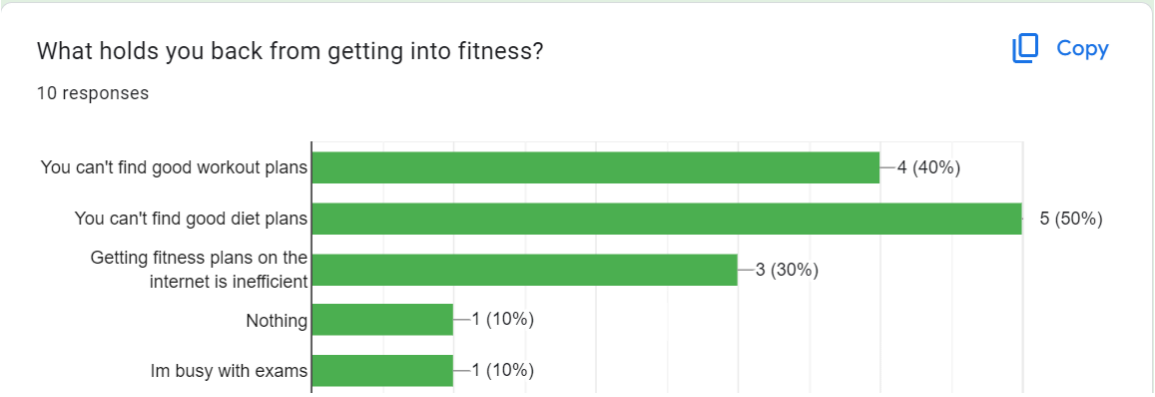
70% of the participants agreed that they are keen towards getting into fitness, this shows my program has many potential users.

Q2) Do you find getting into fitness hard?



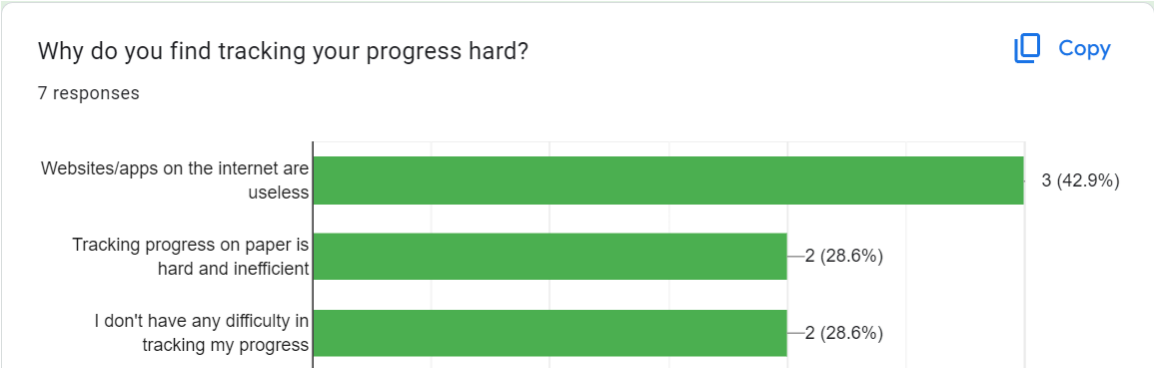
50% of the participants said yes, this shows that my program can help a lot of beginners looking to get into fitness.

Q3) What holds you back from getting into fitness?



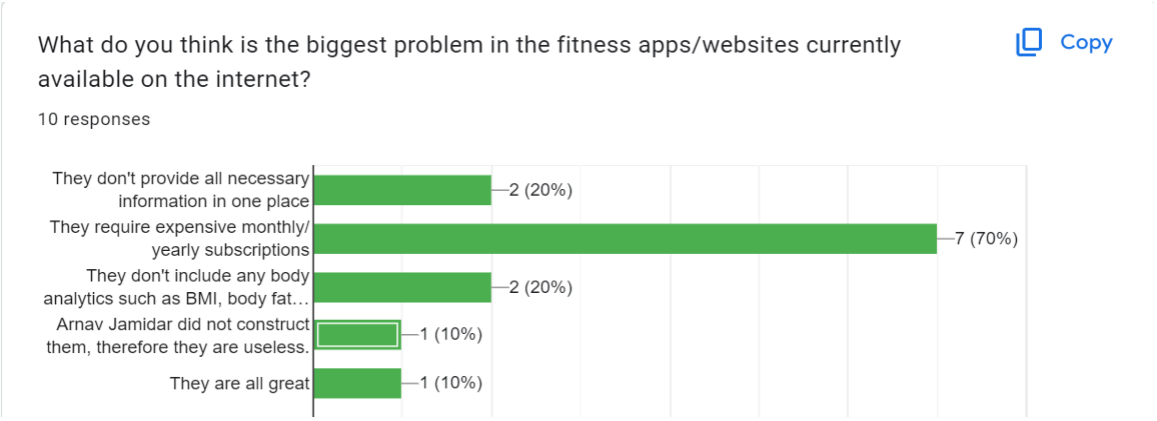
Here, the responses clearly indicate that people face problems without the existing systems for fitness-assistance on the internet, so this shows that my program will actually solve an existing problem for the users.

Q4) Why do you find tracking your progress hard?



These responses highlight the problem stated above, therefore meaning that my program will solve this problem, hence helping many fitness enthusiasts.

Q5) What do you think is the biggest problem in the fitness apps/websites currently available on the internet?



These responses provided by the participants further signify the need of a good fitness program that provides all the information to start and track your fitness journey in the fitness industry. My program will solve this problem for fitness enthusiasts and will encourage more people to get into fitness and a healthier lifestyle.

Analysis of Questionnaire

The questionnaire provided me with even more evidence that my program can truly revolutionise the fitness industry, by providing seamless efficiency in a fitness enthusiast's day-to-day life.

The first question shows that 70% of the participants are interested towards getting into fitness, this means my program can potentially become a way of attracting more people into fitness, benefitting levels of fitness in our society, as specially after the pandemic and its consequences such as several lockdowns, obesity levels in the general population have increased drastically therefore increasing the number of patients that get diagnosed by CHDs, diabetes and other lifestyle related diseases. Furthermore, this can show that there is a big group of people that are keen to become a user of this program. This question has made me decide to make a robust database for this program that can handle a high number of users, so that all of the user's data can be stored without any issues and can be used to improve a user's experience while using the app.

The second question tells us that 50% of the participants find it hard to get into fitness, this shows how people are struggling to improve their health, despite having the right intentions to move forward. This question highlights how the problem about existing systems is affecting the people of our society and restricting a lot of beginners to pursue their journey as a fitness enthusiast.

The third question is one of the most useful questions for my analysis before designing the program, as it highlights the specific problems the participants have that hold them back from getting into fitness. 40% of the participants agreed that they can't find good workout plans, this solidifies my plans to include the feature to suggest workout plans as per the user's fitness goals in my program; 50% of the participants agreed that they can't find good diet plans, this problem will be solved by this program as I plan to include personalised diet plan suggestion as one of the features of the app; 30% of the participants agreed that finding fitness plans on the internet is inefficient, my program is the solution to this problem as the main purpose to introduce this app is to make one's experience on the program as efficient as possible;

The fourth question signifies the problem that fitness enthusiasts find it hard to track their fitness progress, where participants seem to be frustrated with the current systems on the internet. A big number of the participants think that the websites/apps on the internet are useless as 42.9% of the responses agree with this claim. This question also proves that the old-school ways of tracking fitness progress on paper are now considered as outdated and inefficient as 28.5% of the participants agree with this statement.

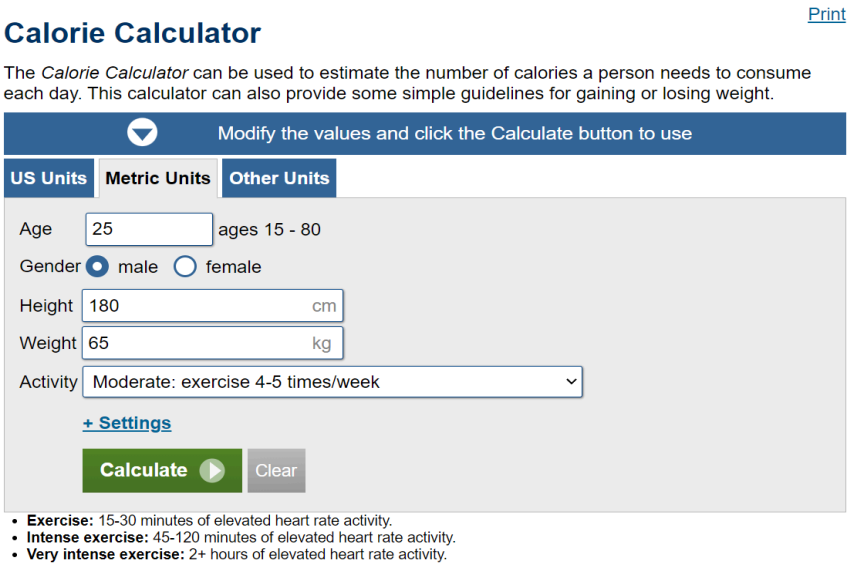
The fifth question shows the biggest problems with current fitness apps/websites currently available on the internet. 20% of the participants agree that the existing fitness systems don't provide all necessary information in one place. This issue will not be present in my program as one of the main purposes of the program is to provide all fitness related information in one program. 70% of the participants think that current fitness systems require expensive monthly/yearly

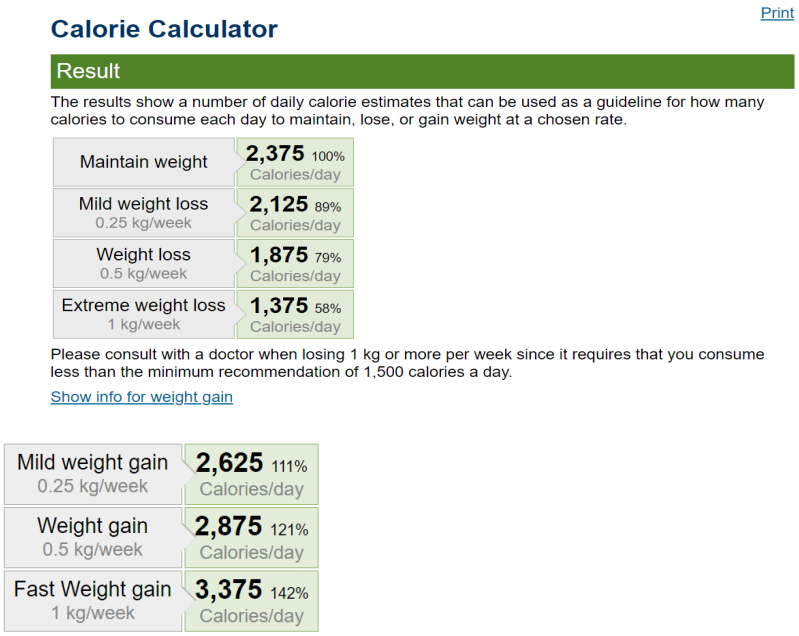
subscriptions, this is not a problem associated with my program as the program will be free to use for all users and there will be no hidden costs required for extra features. 20% of the participants have an issue with current fitness systems not including the ability to calculate any body analytics such as body fat percentage and BMI scores, this problem is solved as my program will include the feature to calculate one's body fat percentage or BMI score using a few body measurements of the user.

Existing Systems

There are several websites/softwares that are currently available on the internet, such as Mayo Clinic and Calculator.net's Calorie calculators, and PureGym's workout plan guide, but none of these solve our problem as these resources are efficient, and the user has to go to different websites to gain the necessary information.

1) Calculator.net





Print

Calorie Calculator

Result

The results show a number of daily calorie estimates that can be used as a guideline for how many calories to consume each day to maintain, lose, or gain weight at a chosen rate.

Maintain weight	**2,375** 100% Calories/day
Mild weight loss 0.25 kg/week	**2,125** 89% Calories/day
Weight loss 0.5 kg/week	**1,875** 79% Calories/day
Extreme weight loss 1 kg/week	**1,375** 58% Calories/day

Please consult with a doctor when losing 1 kg or more per week since it requires that you consume less than the minimum recommendation of 1,500 calories a day.

Show info for weight gain

Mild weight gain 0.25 kg/week	**2,625** 111% Calories/day
Weight gain 0.5 kg/week	**2,875** 121% Calories/day
Fast Weight gain 1 kg/week	**3,375** 142% Calories/day

“Calculator.net” is one of the most trusted websites on the internet when it comes to getting data about your calorie limits. The website provides the user their maintenance calories, and the data about their caloric deficit/surplus, using the measurements the user inputs - body weight, height, activity level, gender and age.

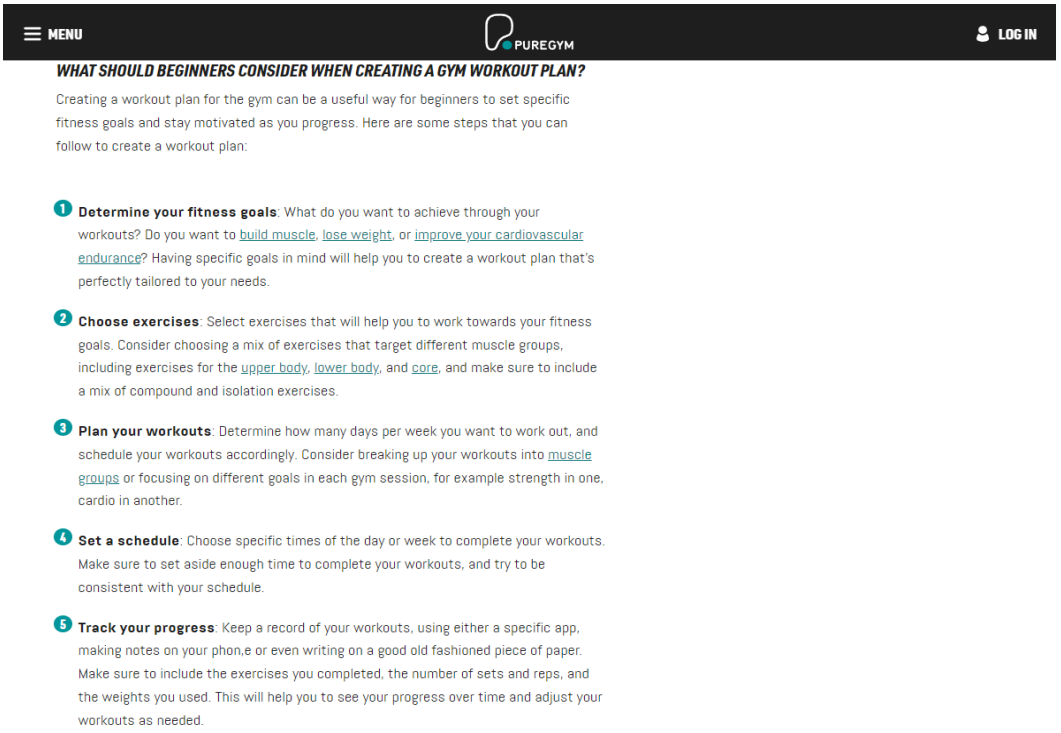
This website partially solves our problem by providing the user information about all the necessary calorie limits, so the user can either consider a caloric deficit or surplus or both as their diet plan.

However, the website does not give any suggestions about possible workout plans and you can’t track your progress. Also, it does not include possible diet plans for the users which is the most essential part of a fitness transformation.

I am planning on using the activity level table from this website as it clarifies the amount of exercise one needs to qualify to a certain activity level, which is an important part in calculating one’s caloric limits.

I would avoid the age input feature from Calculator.net as it restricts young teens(13-14 year olds), this prevents young teens from looking into improving their fitness level and perhaps this can be a serious issue as it can limit their growth during puberty. Instead my program will include modified diet & workout plans for people in their early teens and people of age.

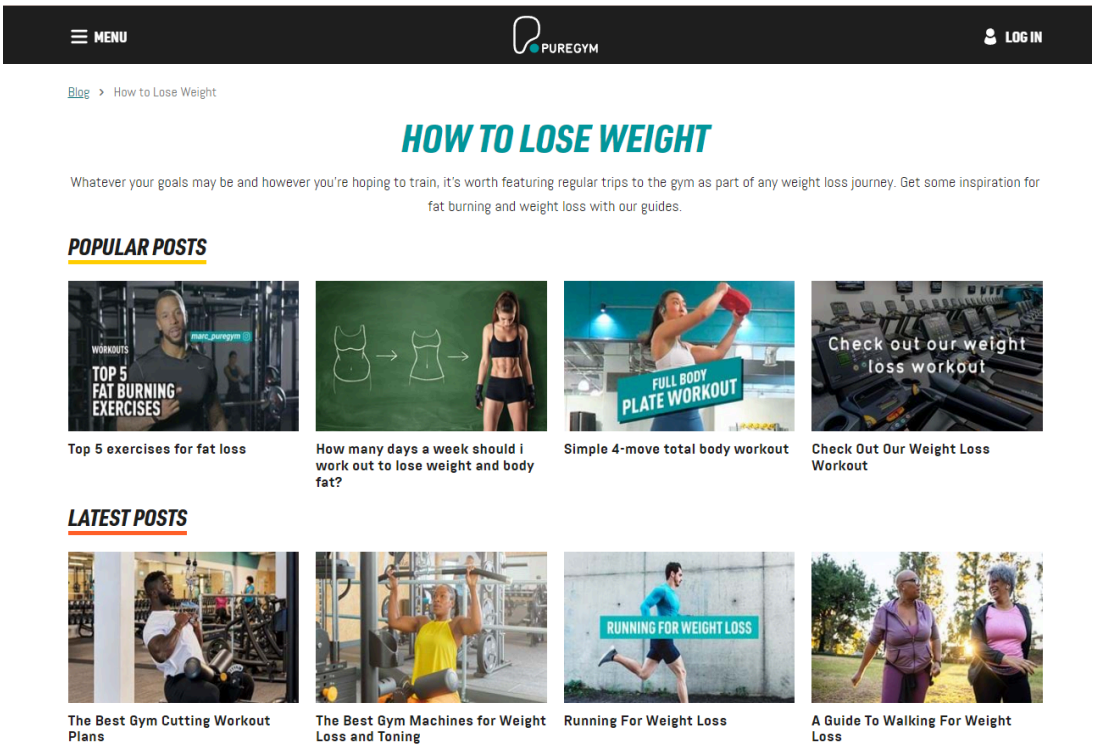
2) Puregym.com

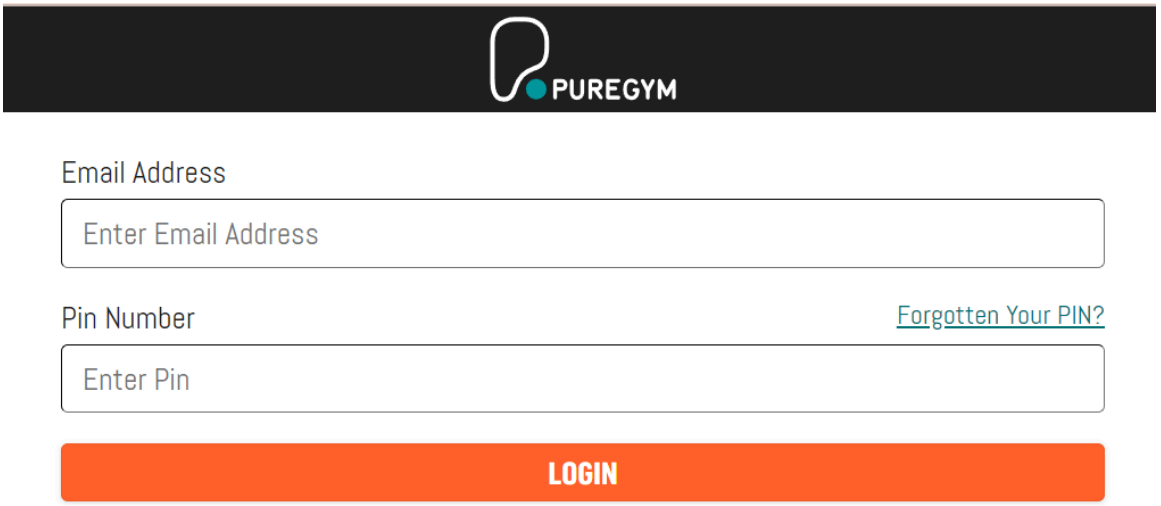

MENU PUREGYM LOG IN

WHAT SHOULD BEGINNERS CONSIDER WHEN CREATING A GYM WORKOUT PLAN?

Creating a workout plan for the gym can be a useful way for beginners to set specific fitness goals and stay motivated as you progress. Here are some steps that you can follow to create a workout plan:

1. **Determine your fitness goals**: What do you want to achieve through your workouts? Do you want to build muscle, lose weight, or improve your cardiovascular endurance? Having specific goals in mind will help you to create a workout plan that's perfectly tailored to your needs.
2. **Choose exercises**: Select exercises that will help you to work towards your fitness goals. Consider choosing a mix of exercises that target different muscle groups, including exercises for the upper body, lower body, and core, and make sure to include a mix of compound and isolation exercises.
3. **Plan your workouts**: Determine how many days per week you want to work out, and schedule your workouts accordingly. Consider breaking up your workouts into muscle groups or focusing on different goals in each gym session, for example strength in one, cardio in another.
4. **Set a schedule**: Choose specific times of the day or week to complete your workouts. Make sure to set aside enough time to complete your workouts, and try to be consistent with your schedule.
5. **Track your progress**: Keep a record of your workouts, using either a specific app, making notes on your phon,e or even writing on a good old fashioned piece of paper. Make sure to include the exercises you completed, the number of sets and reps, and the weights you used. This will help you to see your progress over time and adjust your workouts as needed.





“Puregym.com” is a well-established website/app which allows users to find and create an ideal workout plan according to their fitness goals, it also lets the users track their progress by making an account, hence allowing the user to set an appropriate schedule to assist their planning.

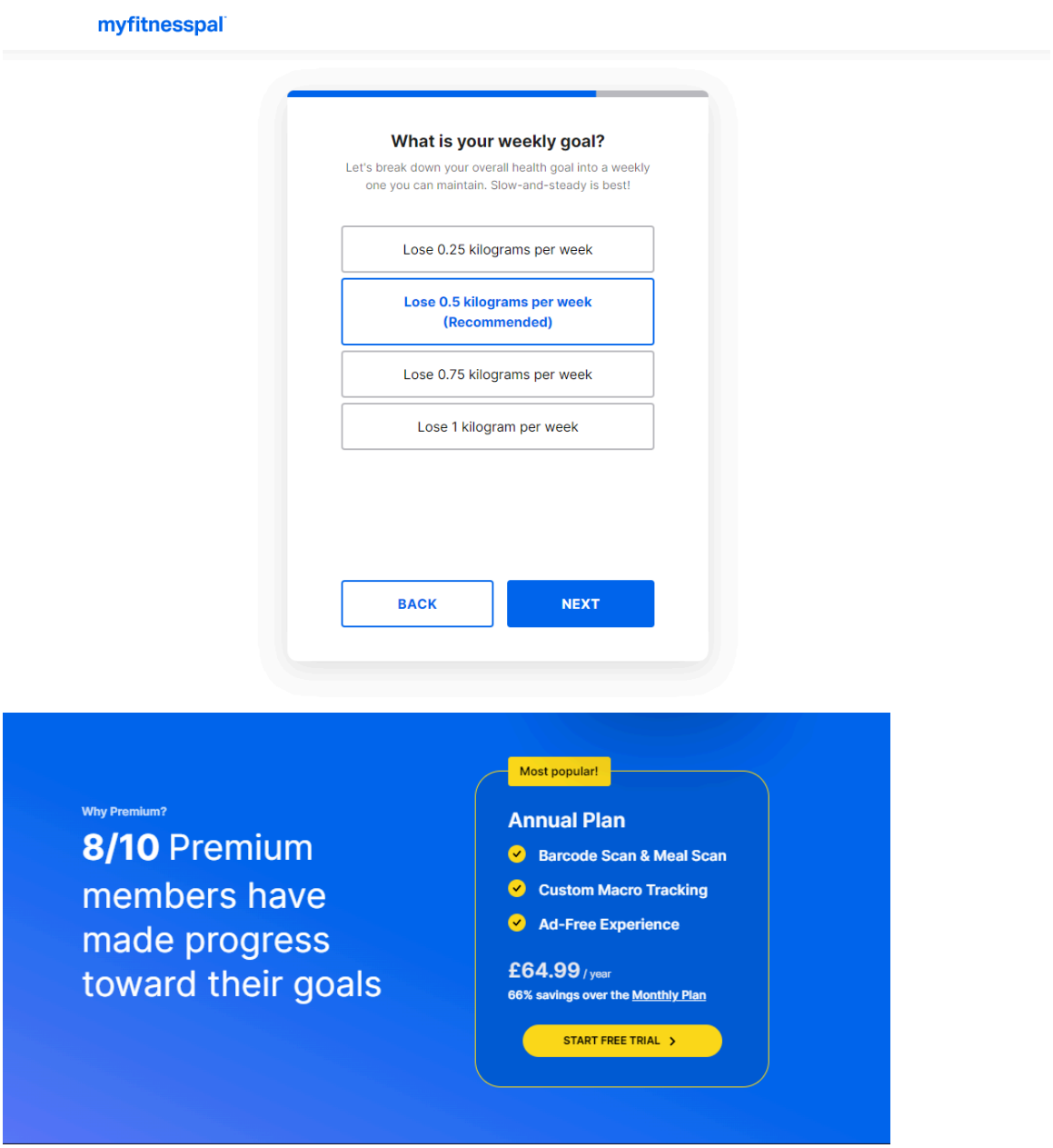
This website partially solves the problem as it provides the user with a wide range of possible workout plans according to their fitness goals and lets them track their progress on a regular basis.

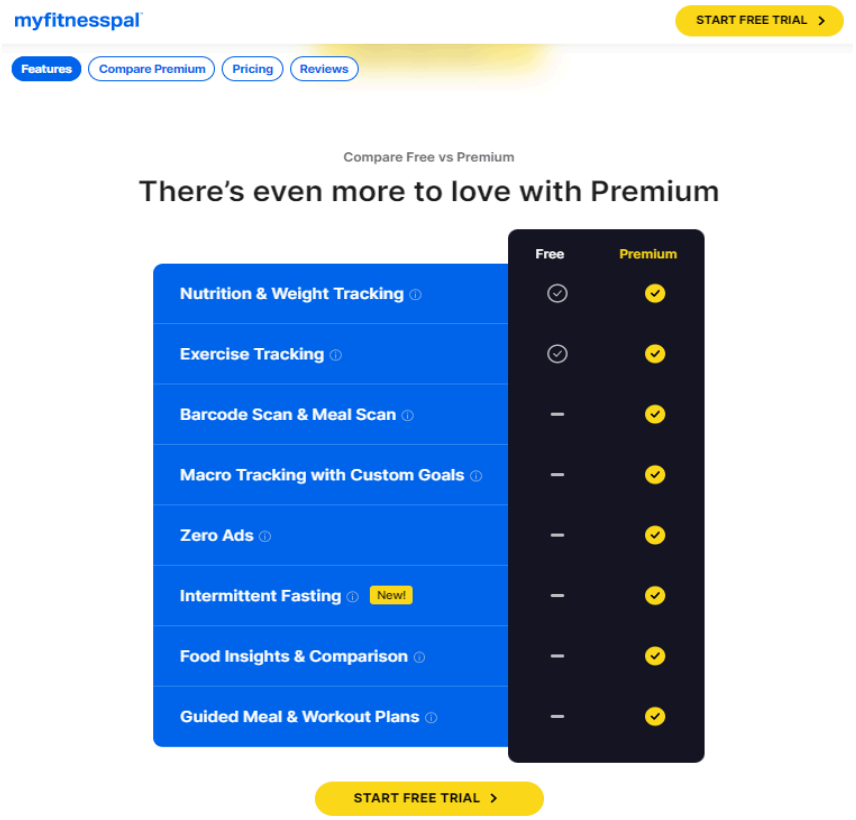
The issue with this website is that it provides no information regarding the nutritional needs for one's fitness transformation. Furthermore, the user has to look into various different posts to select a workout, which is not straight-forward and efficient, this can also seem boring to the user, which can affect their motivation to carry on creating their perfect fitness plan.

The website allows users to choose different exercises for specific body parts such as upper body, lower body, core etc. This is a nice way of simplifying the process of creating a workout plan. I plan to use this feature in my program as it makes the user experience much smoother and efficient.

On the other hand, the user might have to go through a high number of different posts to create a good workout plan for them, this can be time consuming and inefficient, I will avoid this feature as I want my program to be easy to use and less time consuming for the user, instead my program will provide the user with lists of workout plans directly, which means skipping reading through articles included in the posts on Puregym's website.

3) Myfitnesspal.com





Myfitnesspal is one of the most renowned apps which has a high number of users globally. It allows its users to create a personalised diet plan according to their fitness aspirations, it gives the users a lot of options to choose from to customise their diet plan, such as desired weight loss per week. In addition, it lets the user track their daily calorie intake by allowing them to take pictures of their meal and calculating the calories of the meal, making the website very user friendly. It also provides the option to link other apps such as Apple Health to your Myfitnesspal account.

This website partially solves our problem as it provides the user with guided diet and workout plans, as well as allowing the user to store and track their progress, all in one place.

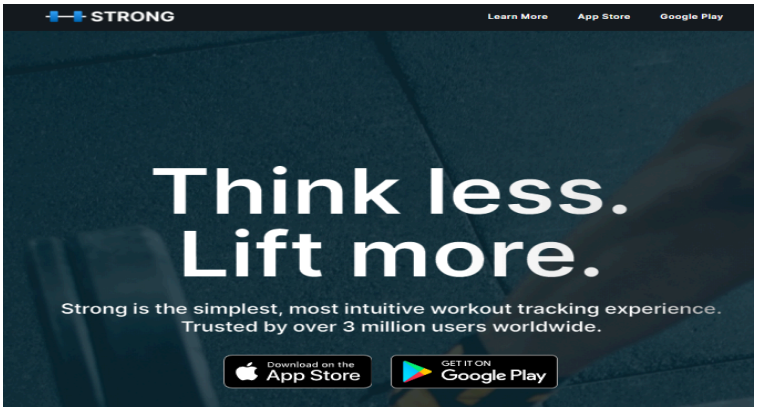
However, the major flaw of Myfitnesspal is that it requires a paid subscription after the 30 days trial period ends, which costs £64.99 per year, this can put off many people who are about to start their fitness journey as this might seem like an additional cost, also not all people will be financially capable to afford the

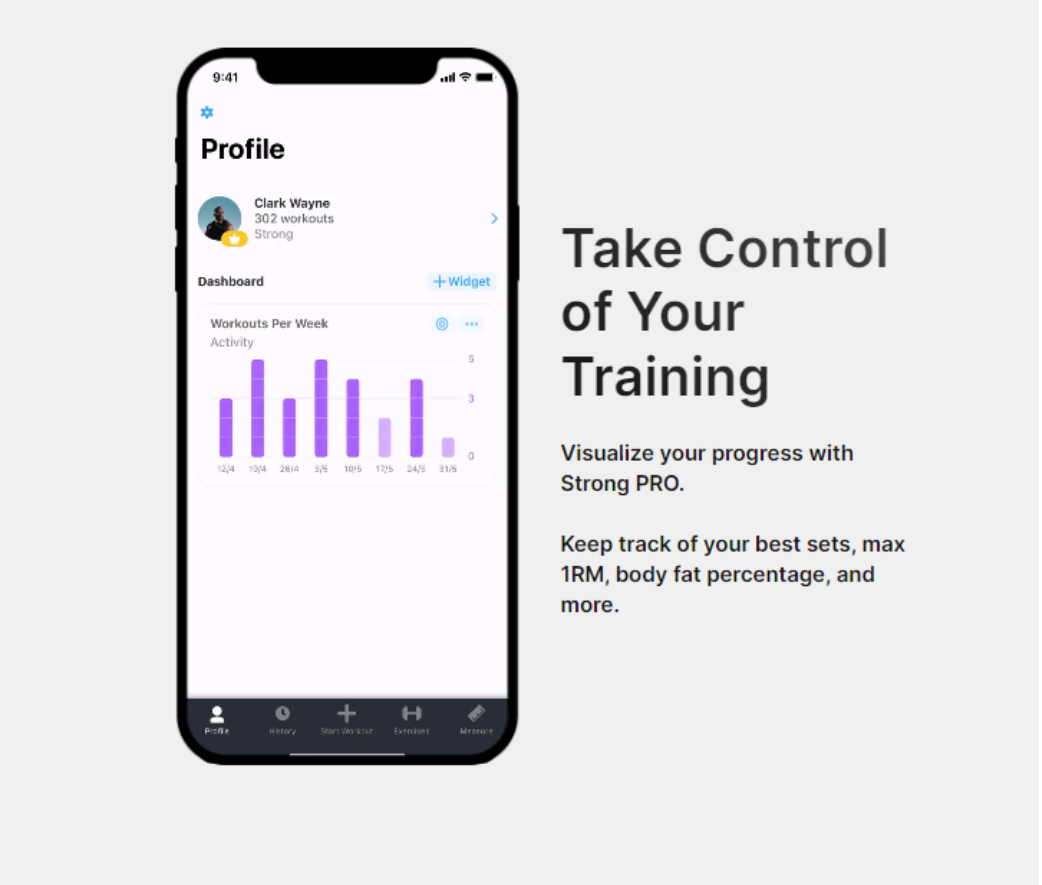
subscription, therefore reducing the number of possible users using the website.

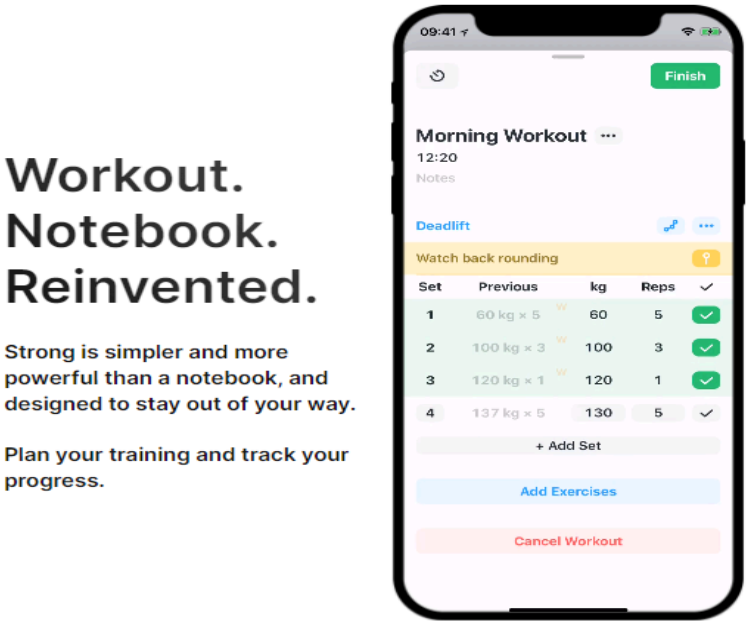
A feature from Myfitnesspal that I would like to include in my program will be the ability to choose the pace at which the user would like to lose/gain weight, this maximises a user's personalisation towards their fitness transformation.

A feature that I would like to alter would be the age input option as Myfitness does not allow anyone under the age of 18 to create an account, while I would prefer my program to let everyone above the age of 12 to make an account to assist them with their fitness goals, where the diet plans suggested to people in the age group of 12-15 years old will be modified to be less restrictive so that a good nutritional balance is consumed by the people about to hit puberty/during adolescence. Also, I will make my program free to use for everyone, ensuring that anyone regardless of their financial situation can access the program and begin their fitness journey.

4) <u>Strong.app</u>







The Strong app allows its users to create workouts, record their workout sessions and track their progress. It also makes the data easily understandable through visual representation of data such as bar graphs.

This app includes all the features required for one to track their fitness routines, body analysis statistics, personal bests etc. This means that the Strong app partially solves our problem. The main issue with this app is that it is incomplete, as it only focuses on the training related aspects of one's fitness transformation, and there is no help provided to the user related to diet plans, tracking their calorie intake etc.

This app specifically makes interpretation of data seem very simple through the use of visual methods of data representation such as bar graphs, pie charts, scatter plots etc. This saves the user a lot of time and makes the process of tracking progress very user friendly. This is a feature that I am planning to include in my program.

On the other hand, this app does not give the user any helpful suggestions, this makes it harder for a beginner user to make good use of this app, this is a feature of this app that I would like to alter. In my program, the user is suggested information which assists them make the right choices throughout their fitness plan.

Objectives

From observations about the existing systems on the internet, I have realised that the main issue in those systems is the lack of a simplistic approach in terms of the layout of the program, which results in the program being less user-friendly. Below is a list of objectives that I have thought would make an efficient and user-friendly system.

1. **Login details of the user must be stored in the database.**
    1.1. Inputs in the "Username" and "Password" are stored in the database if the "Register" button is clicked and no account pre-exists with that username.
        1.1.1. If a user is registering a new account and the username they enter already exists in the database

then an error message is displayed saying - “Username exists”.

1.1.1.1. This is repeated till a new unique username is entered.

1.1.2. If the user is successfully registered a page should be displayed where the user can input their health data which would be stored in the database as the user’s information.

1.1.2.1. If the data inputted is valid then the data should be stored in the database after pressing the Submit button.

1.1.2.2. If data inputted in this user information page is invalid, an error message should be displayed quoting “Invalid inputs”.

1.2. Input details are cross-checked with the login details in the database if the “Login” button is clicked and if the username and password match the user is let into the program with all of their progress stored.

1.2.1. If the username entered is incorrect, an error message is displayed which says - “Incorrect username”.

1.2.1.1. This is repeated until a correct username is entered.

1.2.2. If the username is correct but the password entered is incorrect, an error message is displayed which says - “Incorrect password”.

1.2.2.1. This is repeated until the correct password is inputted.

2. The main menu provides a hub for users to navigate through the program.

2.1. After logging in, three different sections are displayed.

2.1.1. The user can choose to navigate through the three different sections - Diet plans, Workout plans, Your progress.

2.1.2. If the user clicks on the diet plans button, they can select one of the two options - Find diet plans or View saved diet plan.

2.1.2.1. If they press the “Find diet plans” button, the find diet plans menu should appear with an information form with the required information.

2.1.2.1.1. The system then suggests two diet plans based on the provided information.

2.1.2.1.1.1. If the inputs are invalid, an error message should appear which states - “Invalid inputs”.

2.1.2.1.2. The user can then set one of these diet plans as their saved diet plan using the Save button.

2.1.2.1.3. The user can repeat this process at any time, and for unlimited times.

2.1.2.1.4. The “back” button takes the user back to the previous page.

2.1.2.2. If they press the “View saved diet plan” button, the system will show the diet plan currently saved by the user.

2.1.2.2.1. Here, the user will be able to delete their currently saved diet plan in order to save a new diet plan.

2.1.2.2.2. The “back” button takes the user back to the previous page.

2.1.2.3. The “back” button takes the user back to the previous page.

2.1.3. If the user clicks on the workout plans button, they can select one of the two options - Find workout plans or View saved workout plan.

2.1.3.1. If the user clicks the “Find workout plans” button, the find workouts menu should appear.

2.1.3.1.1. The system then suggests two workout plans based on the answers to the questions asked previously.

2.1.3.1.1.1. If the inputs are invalid, an error message should appear which states - “Invalid inputs”.

2.1.3.1.2. The user can then set one of these workouts as their saved workout using the save button.

2.1.3.1.3. The user can repeat this process at any time, and for unlimited times.

2.1.3.1.4. The “back” button takes the user back to the previous page.

2.1.3.2. If the user clicks the “View saved workout” button, the system will show the workout plan currently saved by the user.

2.1.3.2.1. Here, the user will be able to delete their currently saved workout plan in order to save a new workout plan.

2.1.3.2.2. The “back” button will take the user to the previous page.

2.1.3.3. The “back” button will take the user to the previous page

2.1.4. If the user clicks on the Your progress button, they can select to update their progress or view analytics.

2.1.4.1. By clicking on the “Update your progress” button, the user can update their weekly progress or carry out fitness related calculations.

2.1.4.1.1. To update their progress, the user has to enter details such as weight, body fat percentage etc. This option can be accessed by clicking the “Upload your progress” button.

2.1.4.1.1.1. The user can use the “Submit” button to save their progress in the database if no progress has been saved previously.

2.1.4.1.1.1.1. If the inputs are invalid, an error message must be served which states - “Invalid inputs”.

2.1.4.1.1.2. The user can use the “Update” button to update their progress in the database if progress is already saved but they want to update it.

2.1.4.1.1.2.1. If the inputs are invalid, an error message must be served which states - “Invalid inputs”.

2.1.4.1.1.3. By clicking the “Back” button the user can go to the previous page.

2.1.4.1.2. These calculations require the user to enter all the necessary information about the user such as height, gender, age and other body measurements. This option can be accessed by clicking the “Calculations” button.

2.1.4.1.2.1. Pressing the “Body fat percentage” button should take the user to the Body fat percentage calculator where the user can input the required data.

2.1.4.1.2.1.1. If any invalid data is inputted and the user presses the Calculate button, an error message appears stating “Invalid input”.

2.1.4.1.2.1.2. If the user’s inputs are valid, pressing the “calculate” button should execute the calculation and take the user to a results page which shows the result of the calculation.

2.1.4.1.2.1.3. Pressing the “Back” button takes the user to the previous page.

2.1.4.1.2.2. Pressing the “BMI Score” button should take the user to the BMI Score calculator where the user can input the required data.

2.1.4.1.2.2.1. If any invalid data is inputted and the user presses the Calculate button, an error message appears stating “Invalid input”.

2.1.4.1.2.2.2. If the user’s inputs are valid, pressing the “calculate” button should execute the calculation and take the user to a results page which shows the result of the calculation.

2.1.4.1.2.2.3. Clicking the “Back” button takes the user to the previous page.

2.1.4.1.3. The “Back” button should take the user to the previous page.

2.1.4.2. By clicking the “View analytics” button, the user can view the graphs which compare their BMI/Body fat% to the national average and to all the users’ average BMI/Body fat% in the database.

2.1.4.2.1. If the user presses the top “Graph it” button, the BMI comparison graph should appear.

2.1.4.2.1.1. If no data is saved in the database for the user then an error message is shown which says - “Please upload your progress first”.

2.1.4.2.2. If the user presses the bottom “Graph it” button, the Bodyfat comparison graph should appear.

2.1.4.2.2.1. If no data is saved in the database for the user then an error message is shown which says - “Please upload your progress first”.

2.1.4.2.3. The “back” button takes the user back to the previous page.

2.1.4.3. The “back” button takes the user to the previous page.

2.2. By pressing the “Logout” button in the main menu, the user can log out of the system and the login page will appear.

3. Users can update their details at any time.

3.1. Apart from the three main sections in the main menu, there will be another section called - “Your details”.

3.1.1. When the user clicks on the “Your details” button, they are presented with their currently stored details in the Your details page.

3.1.1.1. These details include - name, height, weight, gender, age.

3.1.2. The user can edit these details and then click the “Save” button.

3.1.2.1. The user can edit each detail separately.

3.1.2.2. This button will store the new details in the program’s database.

3.2. This process can be done unlimited times.

3.3. The “back” button takes the user to the previous page.

Potential Algorithms

To make sure my objectives are fulfilled, certain algorithms can be considered to be the best choices to create the optimal fitness program. Some of them are given below.

1)SQLite

To query the database I will use SQLite as it works with Python. Queries will be done to retrieve, add, update and remove data from the database. This is how the front end will interact with the back end of the database. I will use SQLite to create the database for the login system as well. This will provide a smooth experience for users to edit/update their account details.

I have done my research around SQLite through these websites : "https://compucademy.net/user-login-with-python-and-sqlite/?utm_content=cmp-true", "https://www.geeksforgeeks.org/create-mysql-database-login-page-in-python-using-tkinter/"

Data Definition Language(DDL)

The definition language defines the data structure for the storage of data on the database. Example:

```
CREATE TABLE loginCredentials(
    name TEXT,
    username TEXT,
    password TEXT)""")
```

Each SQL statement produces a basic table. Each cell within the table is set to default as Text input for demonstration purposes.

Establishing connection with database

To establish a connection with the database whilst using Python, the following commands will be used.

```
conn = sqlite3.connect("database.db")
c = conn.cursor()
```

The conn variable executes the sqlite command connect and saves the active connection. The variable c executes the sqlite command cursor through the conn variable. The cursor is used to navigate the database just like a mouse pointer is used on a computer.

SELECT

The select statement is used to select groups of data in a database. It can be used to search for individual records or to search for groups containing a certain value.

```
SELECT * FROM userDetails
```

This query selects all the records in the database from the table userDetails. In sql * means all

```
SELECT * FROM userDetails WHERE age = 21
```

This query selects all the records in the database from the table userDetails where the player is aged 21.

INSERT

The insert function is used to insert new records into the database.

```
INSERT INTO userDetails VALUES('Pete Temperton',31,
'07456992785')
```

This SQL query inserts the values in the brackets to a new record in the table playerDetails inside the database

```
INSERT INTO userDetails VALUES('Pete',31, '07456')
```

This SQL query inserts the values in the brackets to a new record in the table playerDetails inside the database

UPDATE

The update function is used to update existing records inside the database.

```
UPDATE userDetails SET age = 23 WHERE name = ‘Pete Temperton’
```

This SQL statement updates all records with the name of Pete Temperton by changing the age to 23.

DELETE

The delete function is used to delete existing records inside a database.

```
DELETE FROM playerDetails WHERE name = ‘Pete Temperton’
```

This SQL query deletes all the records in the table playerDetails from the database with the name Pete Temperton.

```
DELETE FROM fixtures WHERE name = ‘Pete’
```

This SQL statement deletes all the records in the table playerDetails from the database with the name Pete.

Commit

After a statement is executed, the database must be saved. To do this, the commit function is called

```
Data = database.db
data.commit()
```

Once commit is called, the computer saves the database.

Close

The close function is used to close the database when it is no longer in use. This allows the database to be accessed by other programs.

```
Data = database.db
data.close()
```

Once close is called, the computer closes the connection the system has with the database.

Advantages of SQLite

SQLite is a very flexible and user-friendly way to set up a database, as no extra configurations are required to use it, as it is not very processor specific, meaning it can run on most electronic devices, making it easy for users to access the program regardless of the device they are using.

Disadvantage of SQLite

The algorithms in SQLite are not very robust and maintainable, meaning that editing and modifying code could be hard after the program is made. Also, in most cases, SQLite only allows a database size of 2 GB which can be an issue to hold information of all the users in the database.

2)Hashing Algorithms

To encrypt the user's passwords I'll use a custom hashing algorithm to ensure that all the sensitive information about the user is secure and no one can understand and read the details without the key, which will only be accessible to authorised personnel. Also, this makes sure that I am adhering to the Data Protection Act.

An example of a Hashing algorithm is provided below:

```
def custom_hash(message):
    # Define the initial hash value
    hash_value = 0

    # Iterate over each character in the message
```

```
    for char in message:
        # Convert the character to its ASCII value
        char_value = ord(char)

        # Update the hash value
        hash_value = (hash_value + char_value)%256

    return hash_value

# Example usage
message = input("Input message: ")
hashed_value = custom_hash(message)
print("Message:", message)
print("Hashed value: ", hashed_value)
```

In this example, the custom_hash function iterates over each character in the message, converts it to its ASCII value using the ord() function, and updates the hash_value by adding the ASCII value and taking the modulo 256 to ensure it stays within the range of 0 to 255. The resulting hash_value is then returned as the hash for the given message.

I want to emphasise that this custom hashing algorithm is not secure and should not be used for any production or security-sensitive applications. It lacks the robustness and security properties provided by established hashing algorithms like SHA-256.

Advantage

The advantage of hashing algorithms is that they are very secure, as encrypted data cannot be understood without a key to decrypt, which can only be accessed by a few authorised personnel. Also, only a very bad brute force attack can revert the data that has been encrypted using a hashing algorithm. Furthermore, it ensures that I am not breaching the Data Protection Act.

3)Hash tables

Hash tables are a type of data structure in which the address or the index value of the data element is generated from a hash function.

That makes accessing the data faster as the index value behaves as a key for the data value. In other words Hash table stores key-value pairs but the key is generated through a hashing function.

So the search and insertion function of a data element becomes much faster as the key values themselves become the index of the array which stores the data.

The data dictionary types represent the implementation of hash tables. The keys in the dictionary satisfy the following requirements.

- The keys of the dictionary are hashable i.e. they are generated by a hashing function which generates a unique result for each unique value supplied to the hash function.
- The order of data elements in a dictionary is not fixed.

So we see the implementation of a hash table by using the dictionary data types as below.

Accessing values in a dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value.

```
# Declare a dictionary
dict={'Name':'Zara','Age':7,'Class':'First'}
# Accessing the dictionary with its key
print("dict['Name']: ",dict['Name'])
print("dict['Age']: ",dict['Age'])
```

Updating dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

```
# Declare a dictionary
dict={'Name':'Zara','Age':7,'Class':'First'}
dict['Age']=8
# update existing entry
dict['School']="DPS School"
# Add new entry
```

```
print("dict['Age']: ",dict['Age'])
print("dict['School']: ", dict['School'])
```

Deleting dictionary elements

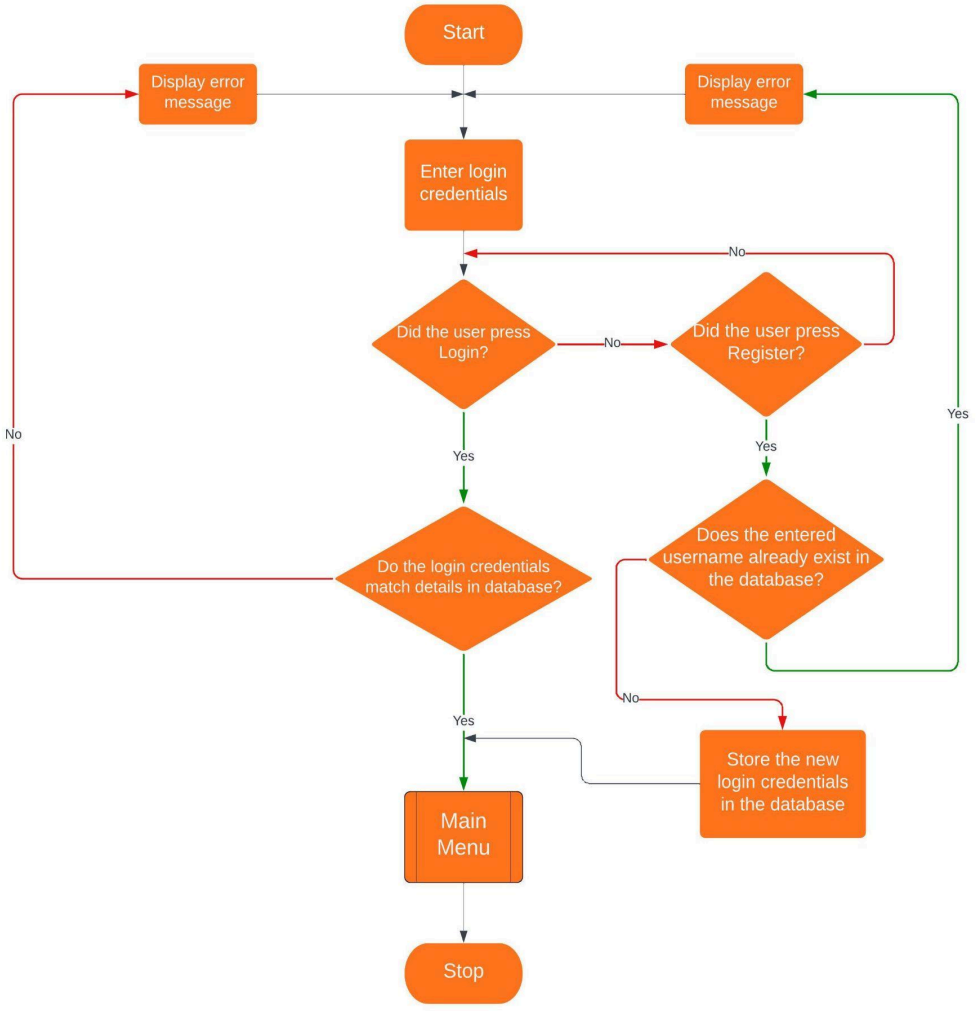
You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete an entire dictionary in a single operation.To explicitly remove an entire dictionary, just use the del statement.

```
dict={'Name':'Zara','Age':7,'Class':'First'}
del dict['Name']# remove entry with key 'Name'
dict.clear()# remove all entries in dict
del dict# delete entire dictionary
print("dict['Age']: ",dict['Age'])
print("dict['School']: ",dict['School'])
```

Flowcharts

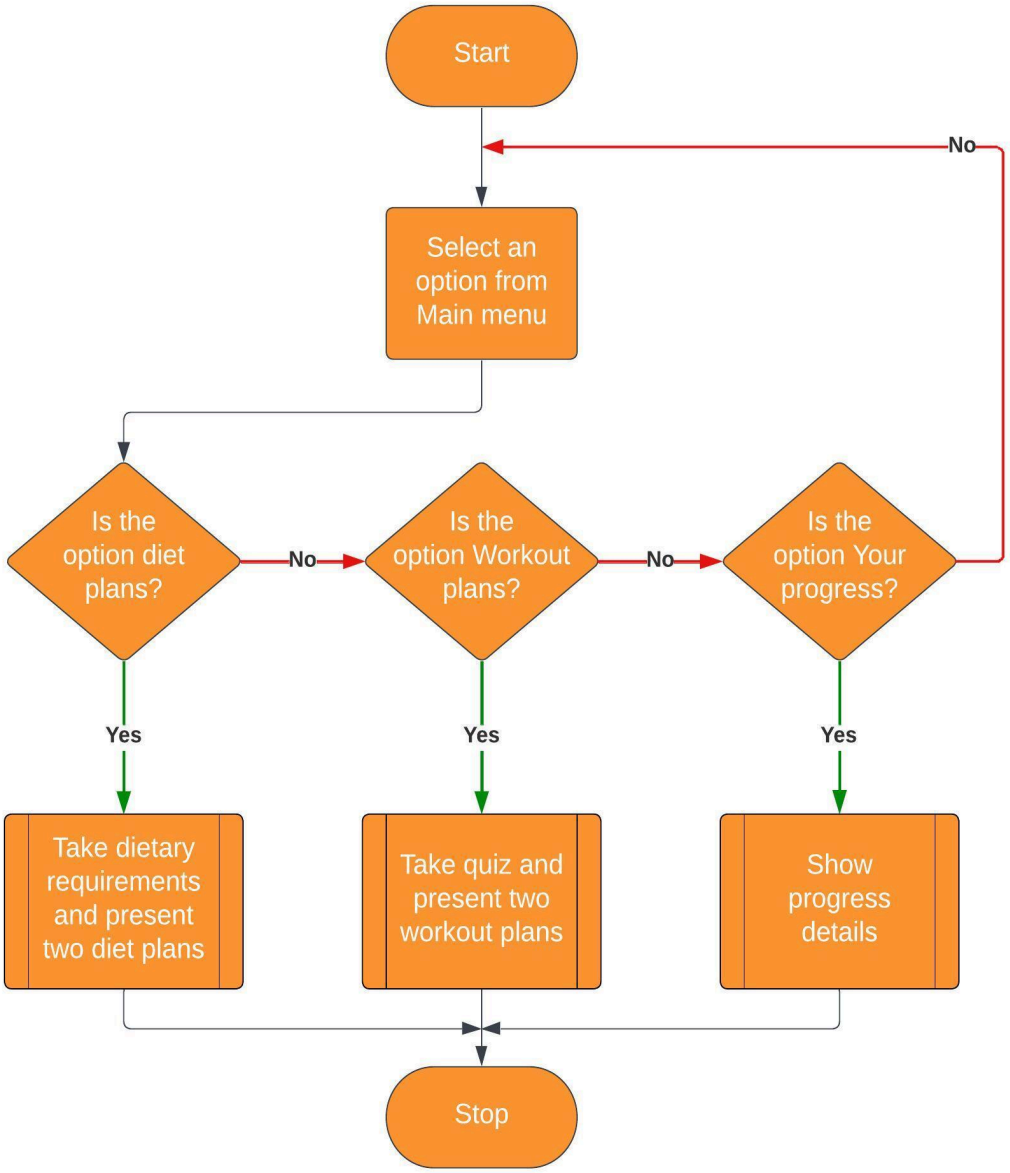
The functionality of the main modules of my program through the use of annotated flowcharts.

1)Login System



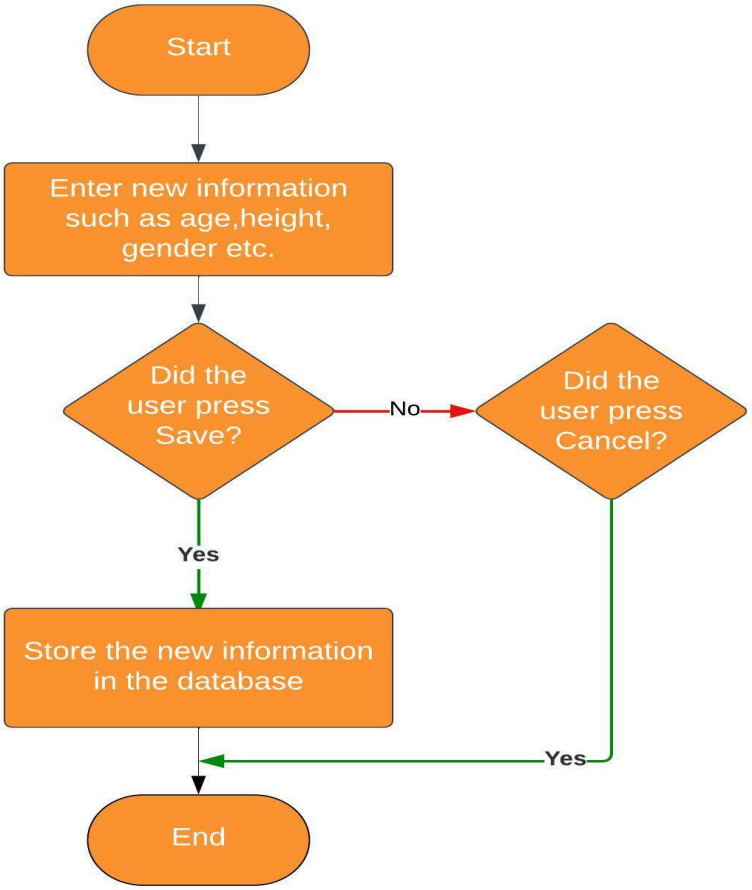
This flowchart will be used as one of the primary processes of gaining access to the system for a user, this is a visual representation of each stage of the login process, where the user will enter the login credentials, which will be checked if they match the credentials in the database or not, if correct the user gains access to the system and the main menu is displayed. If the user inputs incorrect login credentials that don't match the credentials in the database, an error message will be shown and the user will have to re-enter their login credentials.

2)Main menu



This flowchart describes the feature of the main menu interface, where the user is presented with many options to choose from, where pressing an option leads to various subprocesses which provide the user with the final result. This process is repeated until the user has either chosen an option and been presented a result, or has opted to logout.

3)Updating personal information



This flowchart is a visual representation of the process of updating personal information in the system, where the user can edit/update their personal information such as gender, height and age at any time and then proceed to save the new information by saving the edited details, however if the save button is not pressed, any changes made to the personal information will not be saved to the system's database.

Inputs, processes, storage and outputs

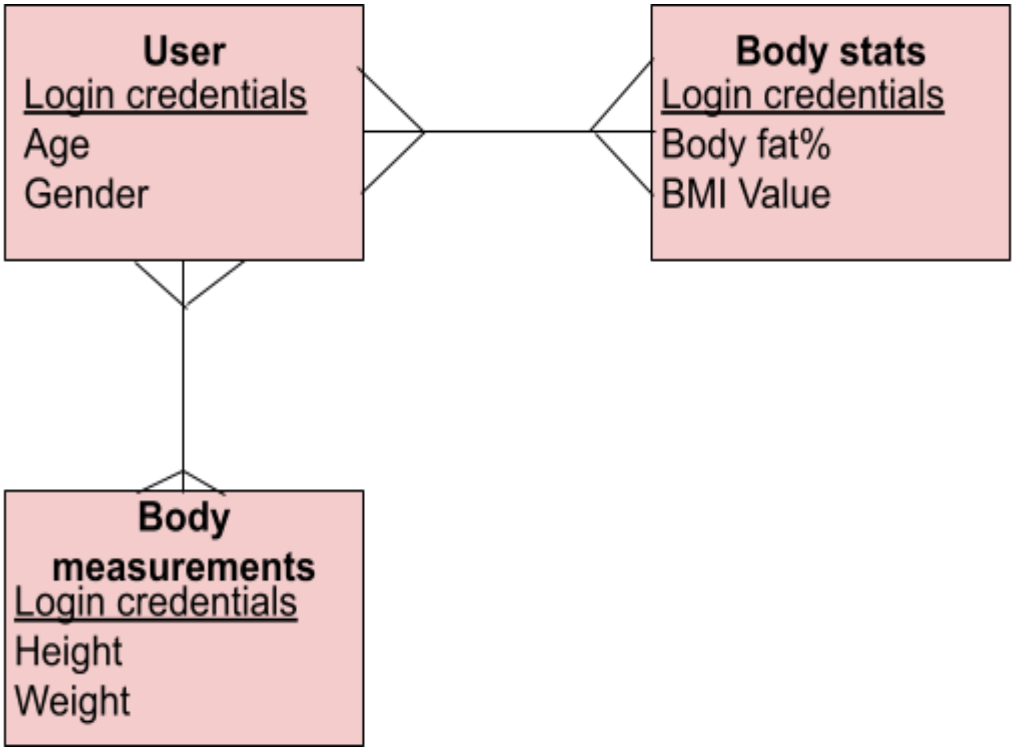
Input	Process	Storage	Output
Login: 1. Username 2. Password	Authenticate login credentials	Credentials stored to credentials table in system database	● Credentials correct ● Credentials incorrect
Selecting menu option: ● Mouse click on option	Call function binded to selected option		Selected screen
New user 1. Firstname 2. Surname 3. Username 4. Password 5. Age 6. Gender 7. Height 8. Dietary requirements 9. Activity Level 10. Weight	Create new user	1. Added to login credentials table in system database 2. Added to information table in system database	● User successfully created ● User creation unsuccessful
Update user details 1. Option to edit 2. Option to save 3. Option to cancel	Edit user	1. Added to login credentials table in system database 2. Added to information table in system database	● Update successful ● Update unsuccessful
Tracking goals: 1. Diet plan 2. Workout plan	Count streaks of consistency	Streak count added to the progress table in system database	● Streak started ● Streak ended ● Streak increased ● New streak record

Data dictionary

Data Item	Data Type	Validation	Sample data
User's full name	String		Pete Temperton
Username	String	<=8 characters	ptemp123
Password	String	<=10 characters, must include a combination to letters & numbers	temp789
Age	Integer	In the range 12-100	17
Gender	String	Must be male or female	Male
Height(cm)	Integer	In the range 100-220	175
Dietary requirements	String	Must be vegetarian, non-vegetarian or vegan	Non-vegetarian
Activity level	String	One of the presented options	3
Weight(kg)	Real	In the range 35-150	71.2
Body fat percentage(%)	Real	Calculated by the user's body stats	10.4
BMI value	Real	Calculated by the user's body stats	22.1
Progress streaks(days)	Integer	>0	19
Neck size	Integer	In the range 15-100	36
Waist size	Integer	In the range 30-200	100
Hip size	Integer		92
Workout plan	String		Day 1: Push Barbell bench press (3 sets of 8-12 reps) Barbell military press (3 sets of 8-12 reps) Dumbbell incline press (3 sets of 8-12 reps)

			Dumbbell lateral raises (3 sets of 8-12 reps) Dumbbell tricep extensions (3 sets of 8-12 reps) Day 2: Pull Barbell deadlifts (3 sets of 8-12 reps) Barbell bent over rows (3 sets of 8-12 reps) Lat pulldowns (3 sets of 8-12 reps) Dumbbell upright rows (3 sets of 8-12 reps) Dumbbell single arm bicep curls (3 sets of 8-12 reps) Day 3: Legs Barbell squats (3 sets of 8-12 reps) Bulgarian split squat (3 sets of 8-12 reps) Leg press (3 sets of 8-12 reps) Leg extensions (3 sets of 8-12 reps) Standing calf raises (3 sets of 8-12 reps)
Diet plan	String		Breakfast: ½ Grapefruit 1 Slice of Toast 2 Tablespoons of peanut butter 1 Cup coffee or tea Lunch: ½ Cup of Tuna 1 Slice of toast 1 Cup coffee or tea Dinner: 1 Cup of green beans ½ Banana 1 Small apple 1 Cup of vanilla ice cream

Entity relationship diagram of system database

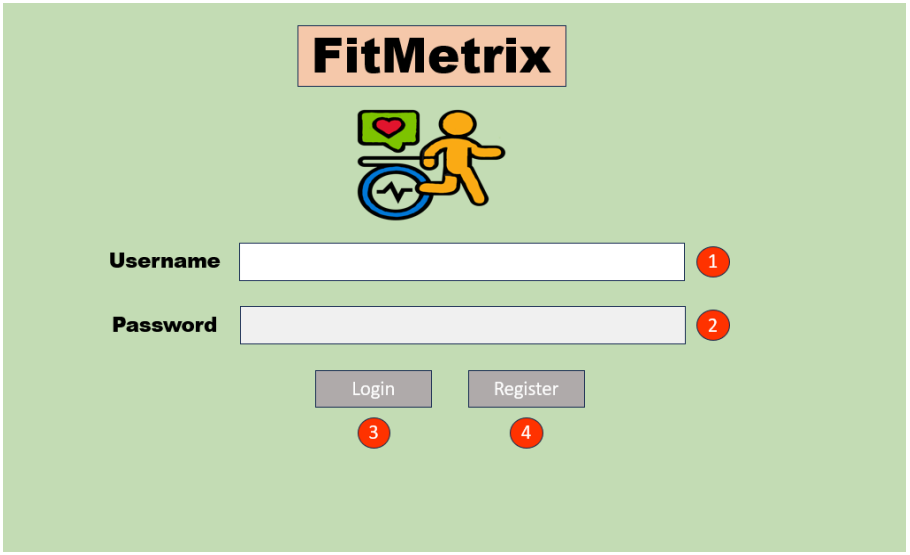


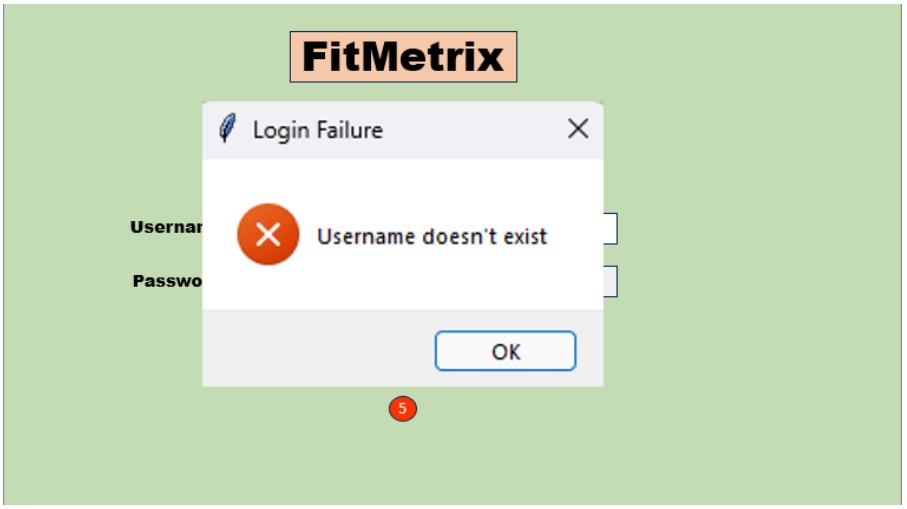
Design

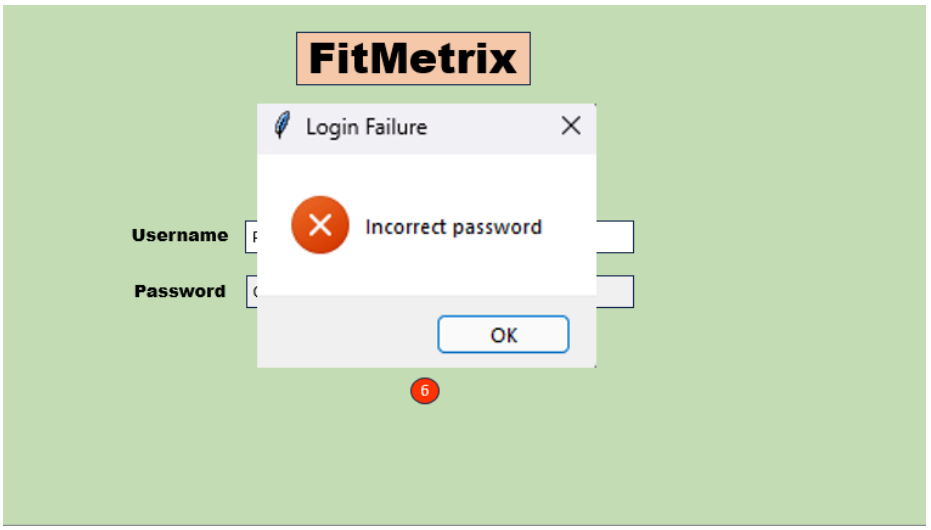
Interface designs

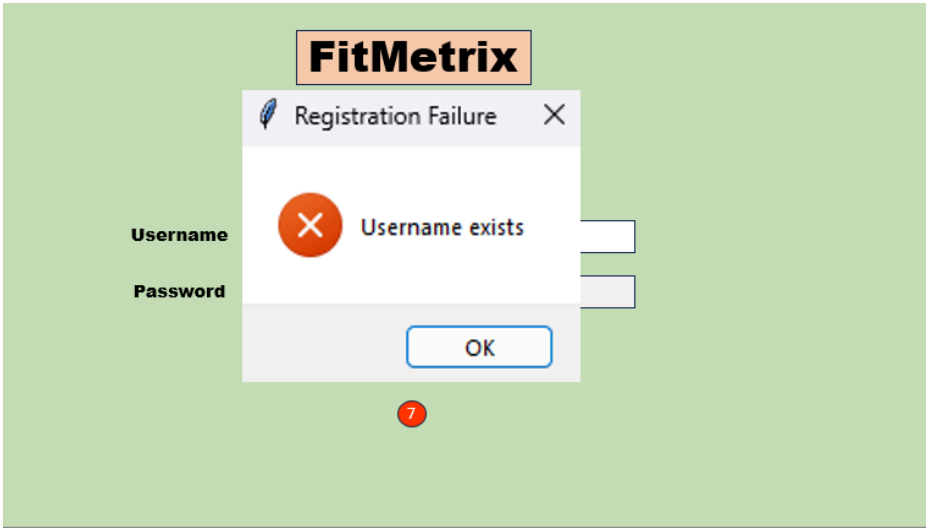
Interface designs are a great way to understand how the program will be set up and how its core functionality will be fulfilled, this ensures that in the development of the program there is a clear objective to accomplish.

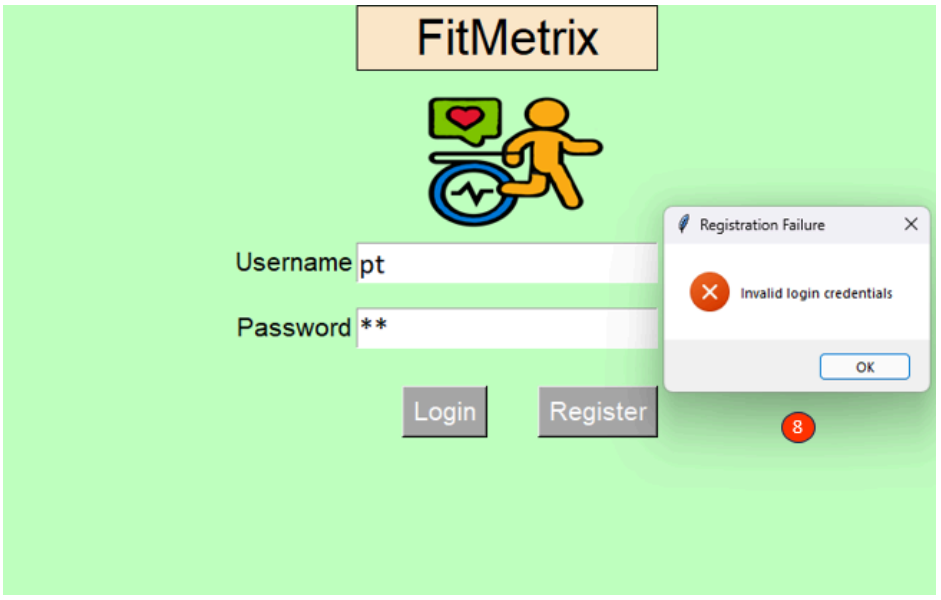
1)Login Interface











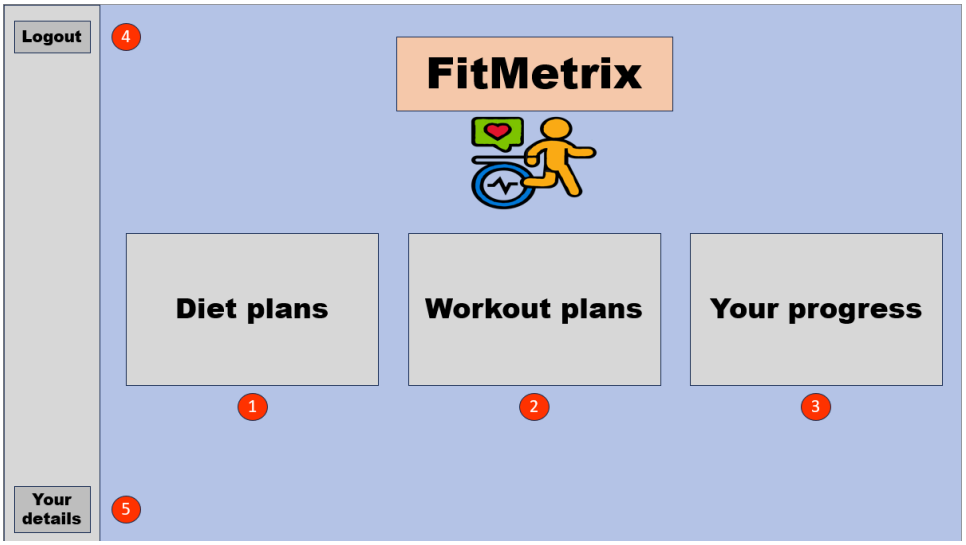
This is the login interface, which is the first thing a user will encounter when they open the program, without login credentials there is no

means to access the system. The description of separate sections of the interface is given below:

1. This is the text box where the user will enter their username to login to the system. The user can enter a combination of letters, numbers and symbols, but the username entered must be greater than 4 characters and less than 12 characters. Font: Calibri; Text colour: #000000; Font size: 18.
2. This is the text box where the user will enter their password to login to the system. The user can enter a combination of letters, numbers and symbols, but the password entered must be greater than 2 characters and less than 20 characters. Font: Calibri; Text colour: #000000; Font size: 18.
3. This is the login button, when a user clicks this button, their login credentials will be checked and matched from the database, if the login credentials are correct, the user will gain access to the system. Font:Arial; Text colour: #ffffff; Font size: 16.
4. This is the register button, when a user clicks this button, the credentials they have inputted will be checked if they already exist in the database or not. If the credentials are unique, they are stored in the database and a new account is created, therefore the user gains access to the system. Font:Arial; Text colour: #ffffff; Font size: 16.
5. This is the error message shown after pressing register, when the username inputted does not exist in the database. Font: Calibri; Text colour: #ffffff; Font size: 18.
6. This is the error message shown after pressing the login button if the username inputted is correct but the password is incorrect and doesn’t match the credentials in the database. Font: Calibri; Text colour: #ffffff; Font size: 18.
7. This is the error message shown when the user clicks the “Register” button, but the username entered already exists in the database. Font: Calibri; Text colour: #ffffff; Font size: 18.
8. This is the error message shown when the register button is pressed and the username entered is less than 5 or greater than 11 characters, and/or the password entered is less than 2

characters or greater than 20 characters. Font: Calibri; Text colour: #ffffff; Font size: 18.

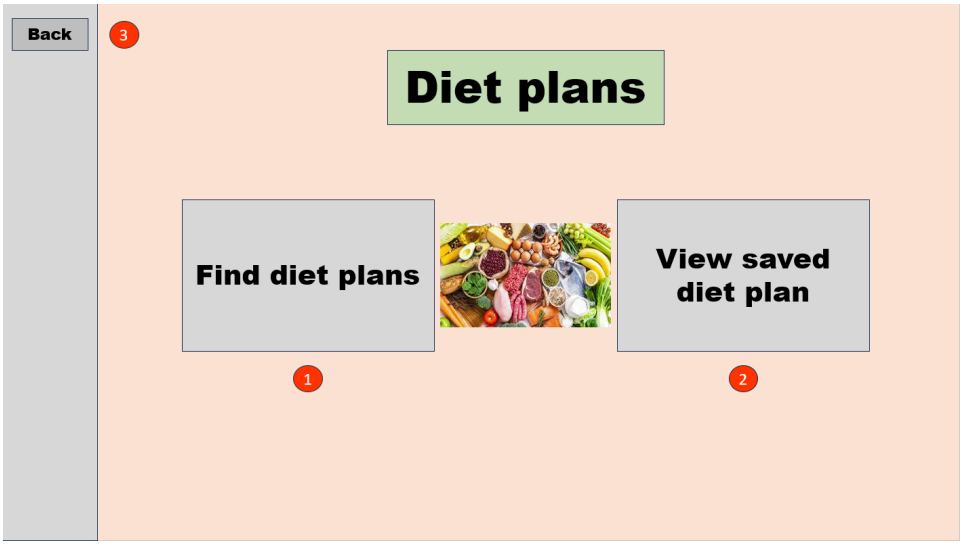
2)Main menu



After logging into the system using the correct login credentials, the user will be presented with this interface, where they can navigate through the main menu and choose a relevant option according to their need. The description of separate sections of the interface is given below:

1. Pressing this button will take the user to the diet plans section of the main menu where they can navigate through the options of the diet plans sections. Font: Arial Black; Text colour: #ffffff; Font size: 28.
2. Pressing this button will take the user to the workout plans section of the main menu, where the user can navigate through the options of the workout plans section. Font: Arial Black; Text colour: #ffffff; Font size: 28.
3. Pressing this button will take the user to the “Your progress” section of the main menu, where the user can choose one of the options to track/view their progress. Font: Arial Black; Text colour: #ffffff; Font size: 28.
4. Pressing this button will logout the user from the system. Therefore, the user is sent back to the login interface to login again. Font: Arial Black; Text colour: #ffffff; Font size: 14.
5. Pressing this button will take the user to the “Your details” section of the main menu, where the user can update or edit their personal details. Font: Arial Black; Text colour: #ffffff; Font size: 14.

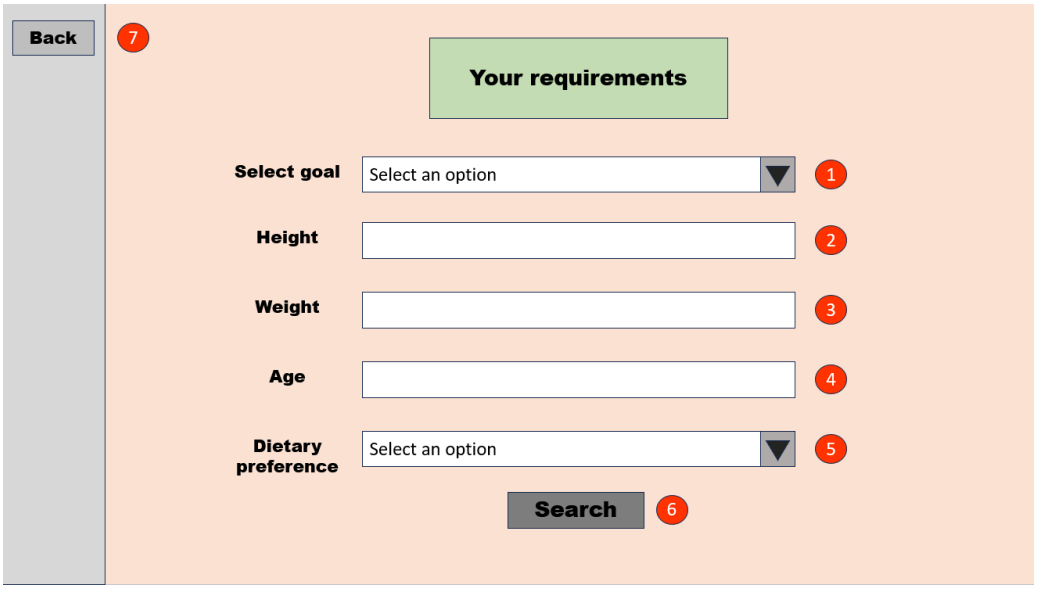
3)Diet plans menu

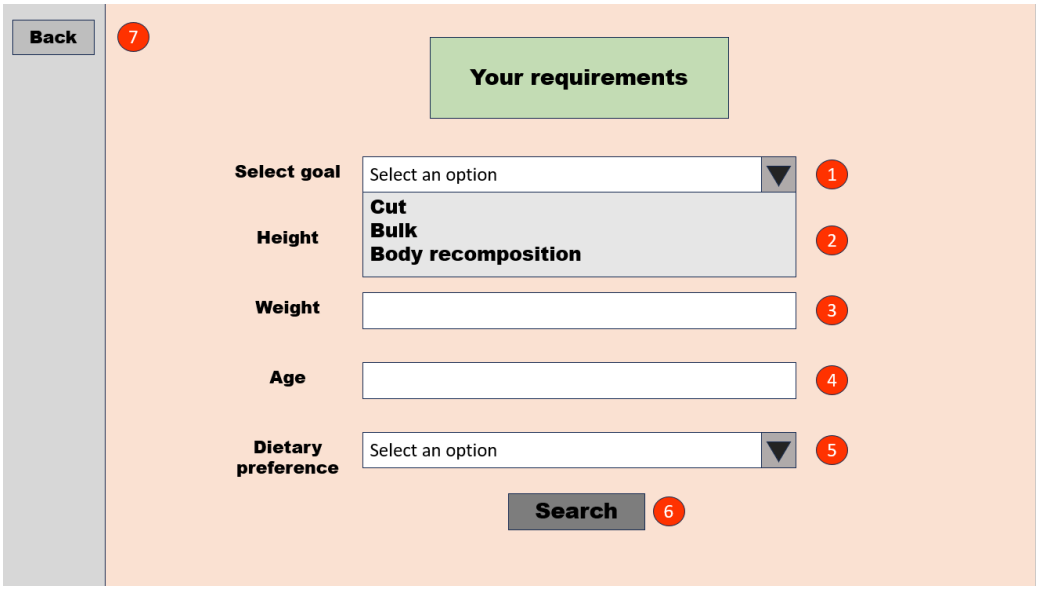


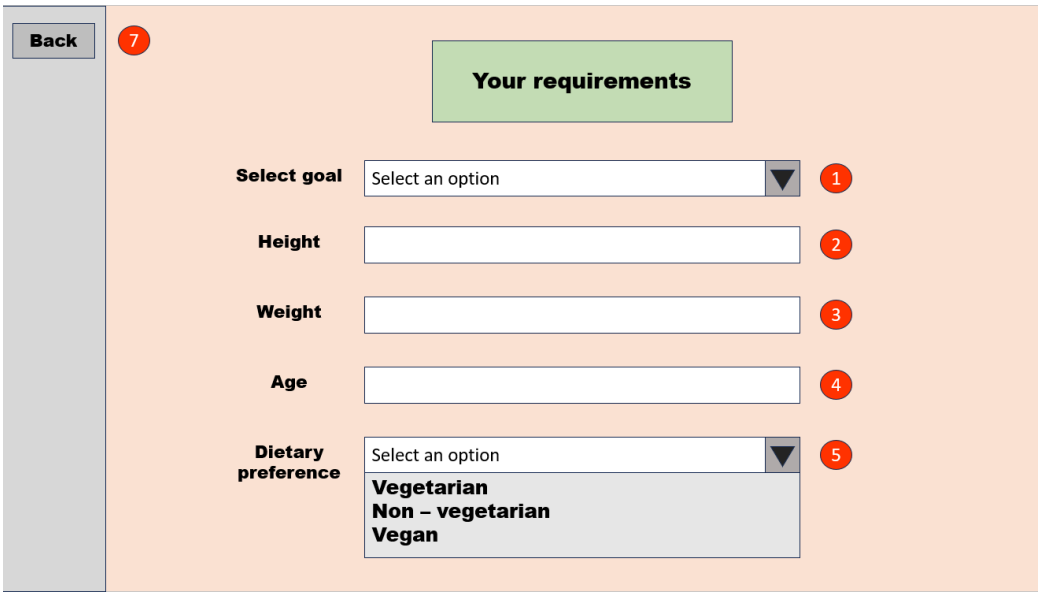
If the user clicks diets plan button in the main menu, this interface will appear, where the user has three options to choose from, the description of these options are as follows:

1. This button takes the user to the find diet plans section, where the user can enter their requirements to search suitable diet plans. Font: Arial Black; Text colour: #ffffff; Font size: 28.
2. This button presents the user with their currently saved diet plan. Font: Arial Black; Text colour: #ffffff; Font size: 28.
3. This button gives the user the option to go back to the main menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

4)Find diet plans search



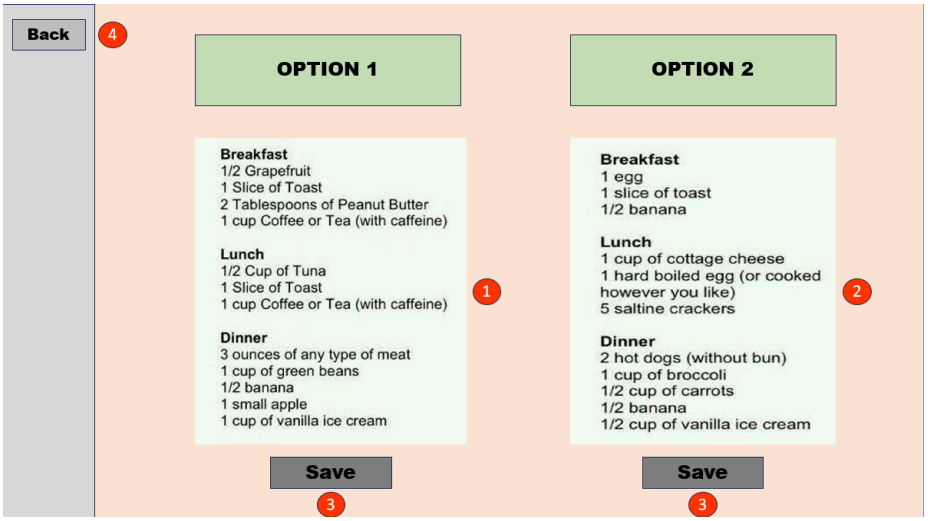




After clicking on the find diet plans button, this interface will appear where the user can fill out their requirements for a personalised diet plan. The description of each section of this interface is as follows:

1. This is a drop down list where the user can choose one of the three options to move forward with their search, clicking one of the three options will confirm their choice of their fitness goals, the list of options drops down by clicking on the down arrow. Font: Calibri; Text colour: #ffffff; Font size: 18.
2. This is a text box where the user can enter their height, in centimetres, the input has to be an integer in the range 100-200. Font: Calibri; Text colour: #ffffff; Font size: 18.
3. This is a text box where the user can enter their weight, in kilograms, the input must be an integer, in the range 30 - 200. Font: Calibri; Text colour: #ffffff; Font size: 18.
4. This is a text box where the user can enter their age, in years, the input must be an integer, in the range 12-100. Font: Calibri; Text colour: #ffffff; Font size: 18.
5. This is a drop down list where the user can choose one of the three options to move forward with their search, clicking one of the three options will confirm their choice of the dietary preference, the list of options drops down by clicking on the down arrow. Font: Calibri; Text colour: #ffffff; Font size: 18.
6. This button is used to execute the search command once the user has filled all the requirements. Font: Arial Black; Text colour: #ffffff; Font size: 20.
7. This button gives the user the option to go back to the diet plans menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

5)Find diet plans search results



This interface will appear when the user fills out their requirements and clicks on the search button in the previous interface. Here the user can save one of the two diet plans to their profile, otherwise, the user can also choose to go back to the main menu. The description of each separate section of the interface is given below:

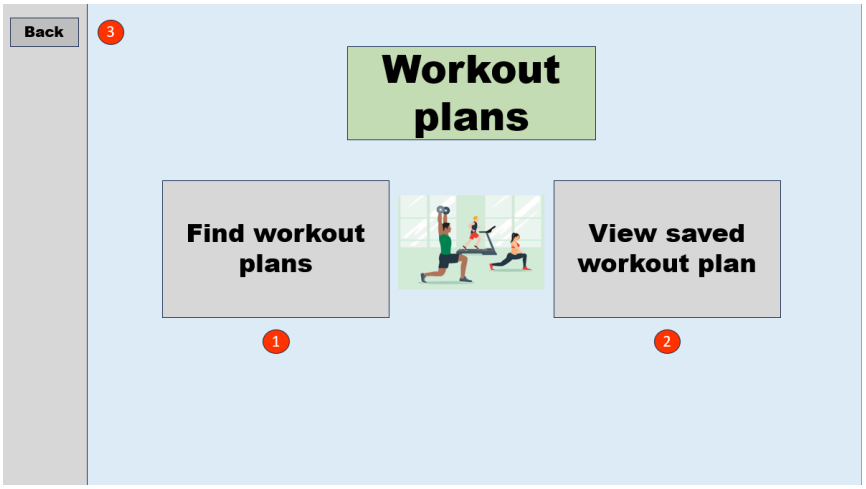
1. This is the first option that the user is presented with after they have completed the search, this is a textbox displaying the diet plan that the user can save to their profile. Font: Arial Black; Text colour: #ffffff; Font size: 16.
2. This is the second option that is presented to the user after they have completed the search, this is a textbox displaying the diet plan that the user can save to their profile. Font: Arial Black; Text colour: #ffffff; Font size: 16.
3. These are the buttons that can be used to save one of the two diet plans to their profile. Font: Arial Black; Text colour: #ffffff; Font size: 20.
4. This button takes the user back to the find diet plans search menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

6)Saved diet plans

After clicking the view saved plans button in the diet plans menu, this interface will appear, here the user can view their saved diet plan, delete the saved diet plan, or go back to the diet plans menu.The details of each separate section is as follows:

1. This is a textbox displaying the diet plan that the user has saved most recently. Font: Arial Black; Text colour: #ffffff; Font size: 16.
2. This button allows the user to delete their current diet plan. Once deleted, there is no possibility to recover any deleted diet plans. Font: Arial Black; Text colour: #ffffff; Font size: 20.
3. This button can be used by the user to go back to the diet plans menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

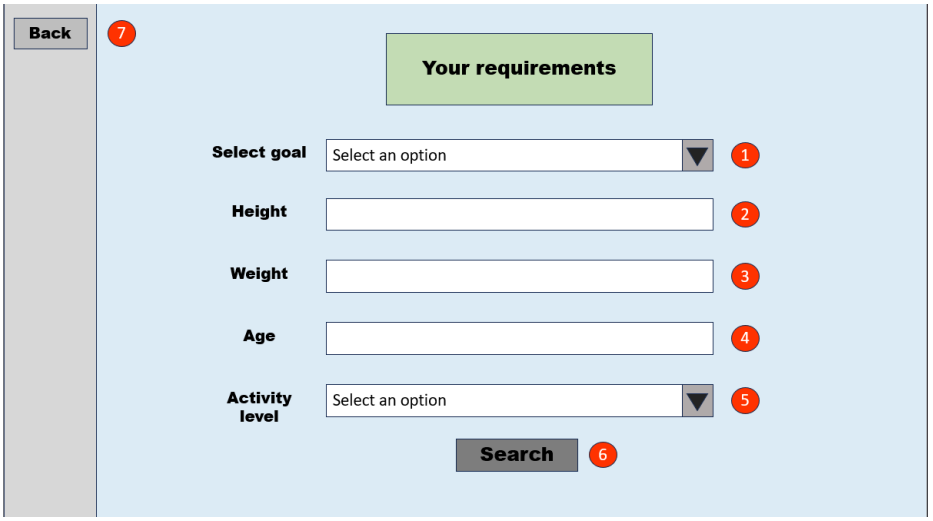
7)Workout plans menu

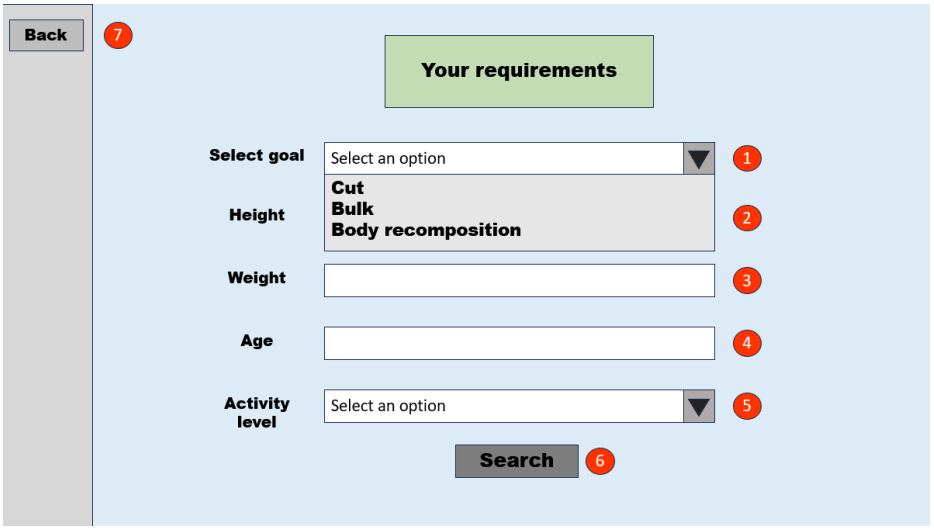


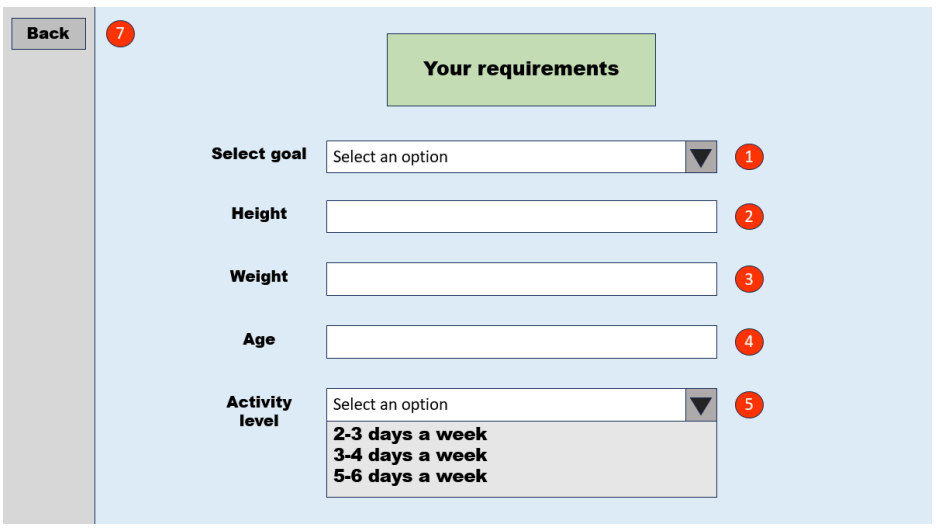
If the user clicks on the workout plans button in the main menu, this interface will appear, where the user can choose from two options - find workout plans or view saved workout plans. The description of the different sections of the interface are given below:

1. This button takes the user to the find workout plans section, where the user can enter their requirements to search for suitable workout plans. Font: Arial Black; Text colour: #ffffff; Font size: 28.
2. This button presents the user with their currently saved workout plan. Font: Arial Black; Text colour: #ffffff; Font size: 28.
3. This button gives the user the option to go back to the main menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

8)Find workout plans search



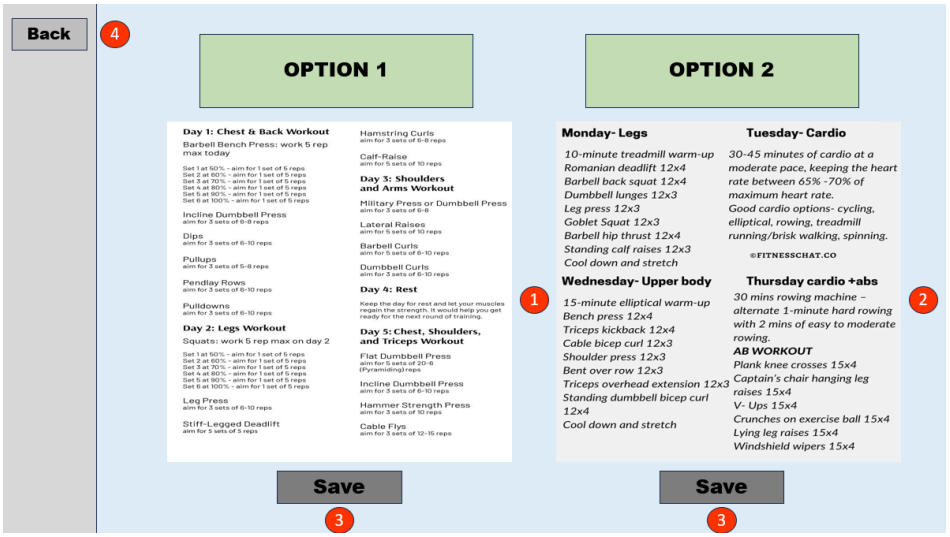




This interface shows up after the user clicks on the find workout plans button in the workout plans menu, where the user can fill out their requirements to execute their search for a suitable workout plan. These are the descriptions for each separate section in the interface:

1. This is a drop down list where the user can choose one of the three options to move forward with their search, clicking one of the three options will confirm their choice of their fitness goals, the list of options drops down by clicking on the down arrow. Font: Calibri; Text colour: #ffffff; Font size: 18.
2. This is a text box where the user can enter their height, in centimetres, the input has to be an integer in the range 100-200. Font: Calibri; Text colour: #ffffff; Font size: 18.
3. This is a text box where the user can enter their weight, in kilograms, the input must be an integer, in the range 30 - 150. Font: Calibri; Text colour: #ffffff; Font size: 18.
4. This is a text box where the user can enter their age, in years, the input must be an integer, in the range 12-100. Font: Calibri; Text colour: #ffffff; Font size: 18.
5. This is a drop down list where the user can choose one of the three options to move forward with their search, clicking one of the three options will confirm their choice of their activity level, the list of options drops down by clicking on the down arrow. Font: Calibri; Text colour: #ffffff; Font size: 18.
6. This button is used to execute the search command once the user has filled all the requirements. Font: Arial Black; Text colour: #ffffff; Font size: 20.
7. This button gives the user the option to go back to the workout plans menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

9)Find workout plans search results



This interface will appear when the user fills out their requirements and clicks on the search button in the previous interface. Here the user can save one of the two workout plans to their profile, otherwise, the user can also choose to go back to the main menu. The description of each separate section of the interface is given below:

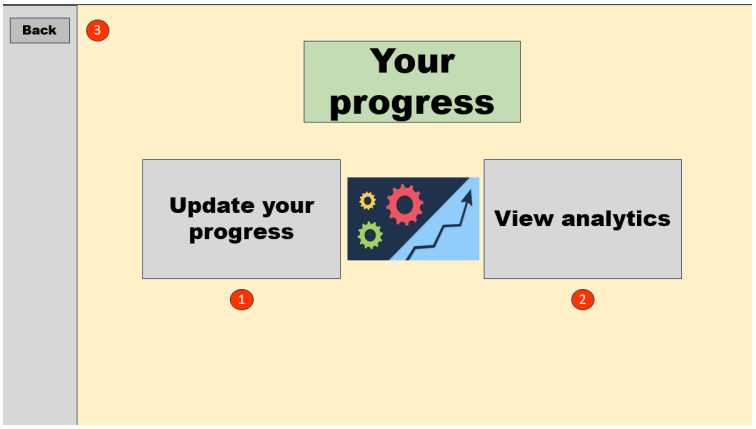
1. This is the first option that the user is presented with after they have completed the search, this is an image of the workout plan that they can save to their profile.
2. This is the second option that is presented to the user after they have completed the search, this is an image of the workout plan that they can save to their profile.
3. These are the buttons that can be used to save one of the two workout plans to their profile. Font: Arial Black; Text colour: #ffffff; Font size: 20.
4. This button takes the user back to the find diet plans search menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

10)Saved workout plans

After clicking the view saved workout plan button in the diet plans menu, this interface will appear, here the user can view their saved workout plan, delete the saved workout plan, or go back to the workout plans menu. The details of each separate section is as follows:

1. This is an image of the diet plan saved by the user most recently.
2. This button allows the user to delete their current workout plan. Once deleted, there is no possibility to recover any deleted workout plans. Font: Arial Black; Text colour: #ffffff; Font size: 20.
3. This button can be used by the user to go back to the workout plans menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

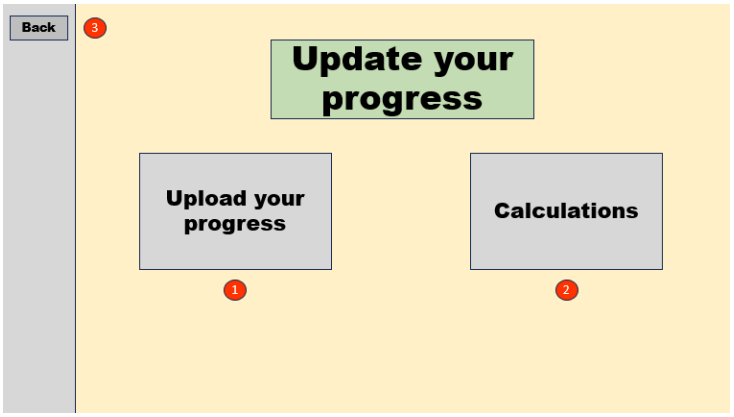
11)Your progress menu



This interface will appear if the user clicks on the “Your progress” button, here, the user can choose to update their progress or view their fitness analytics. The description for each separate section of this interface are as follows:

1. This button takes the user to the Update your progress menu, where the user is presented with two options to move forward. Font: Arial Black; Text colour: #ffffff; Font size: 28.
2. This button presents the user with their weight - time graph and workout & diet plan streaks. Font: Arial Black; Text colour: #ffffff; Font size: 28.
3. This button takes the user back to the main menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

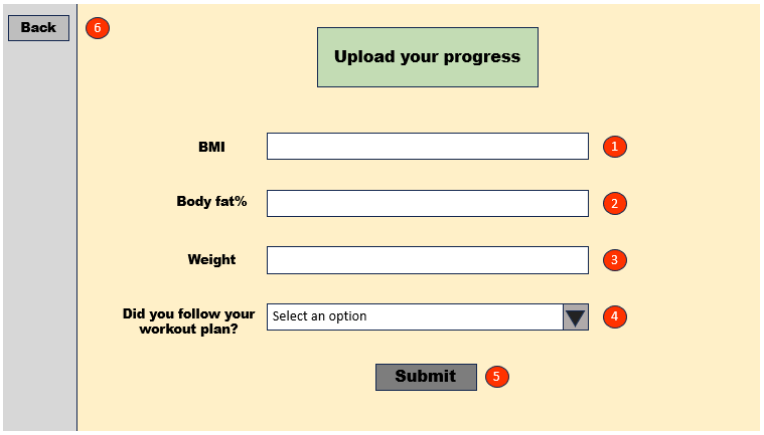
12)Update your progress menu

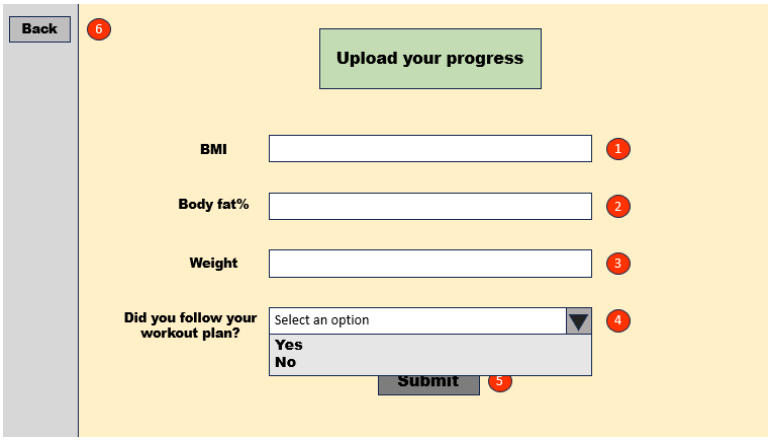


This interface will appear if the user clicks the “Update your progress” button in the Your progress menu. The user can now choose from two options - Update your weekly progress or carry out fitness related calculations. The details of each separate section of this interface is given below:

1. This button takes you to the update your weekly progress section, where the user can enter their fitness stats to save their weekly progress. Font: Arial; Text colour: #000000; Font size:40.
2. This button takes you to the calculations section where the user can calculate their body fat percentage or BMI score. Font: Arial Black; Text colour: #000000; Font size:18.
3. This button takes you back to the Your progress menu. Font: Arial Black; Text colour: #000000; Font size: 18.

13)Update your weekly progress



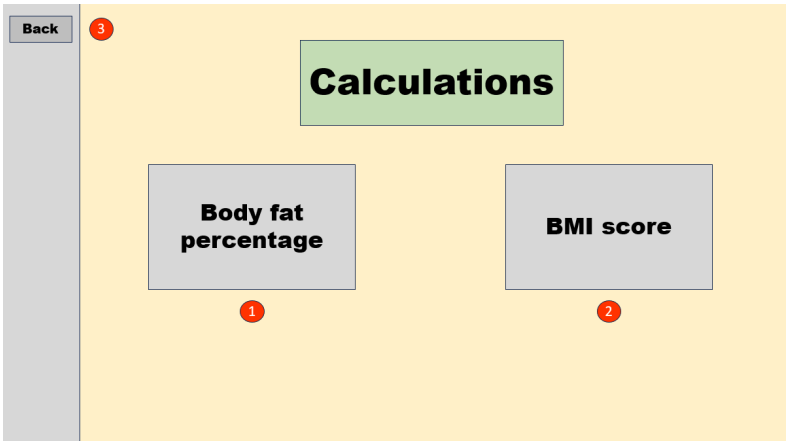


This interface lets the user enter their weekly fitness stats to save it to the system's database, this data is used to determine and update the user's workout and diet plan streaks. Each separate section is described below:

1. This is a text box where the user can enter their BMI score. Font: Calibri; Text colour: #000000; Font size: 18.
2. This is a text box where the user can enter their body fat percentage. The input must be an integer in the range 1-50. Font: Calibri; Text colour: #000000; Font size: 18.
3. This is a text box where the user can enter their weight in kilograms. The input must be an integer in the range 30-150.
4. This is a drop down list with two options - yes or no, choosing one of these two options allows the user to carry on with submitting their weekly progress. Font: Calibri; Text colour: #000000; Font size: 18.

5. This button saves the user's progress by saving the stats entered by the user into the system's database. Font: Arial Black; Text colour: #000000; Font size: 16.
6. This button takes the user back to the Update your progress menu. Font: Arial Black; Text colour: #000000; Font size: 16.

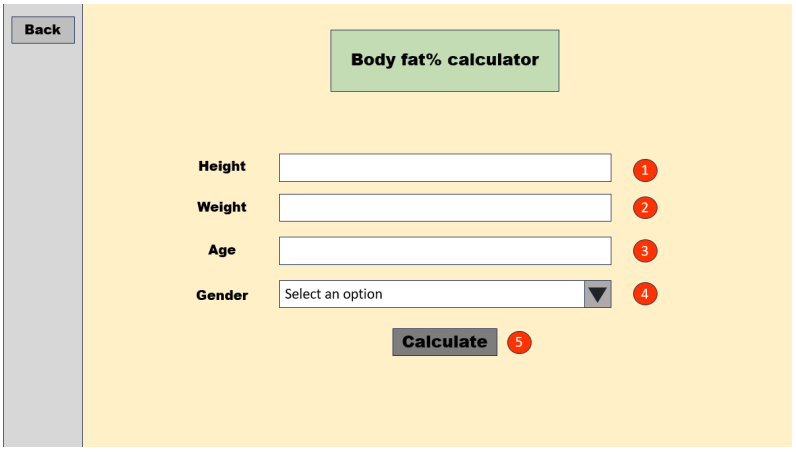
14)Calculations

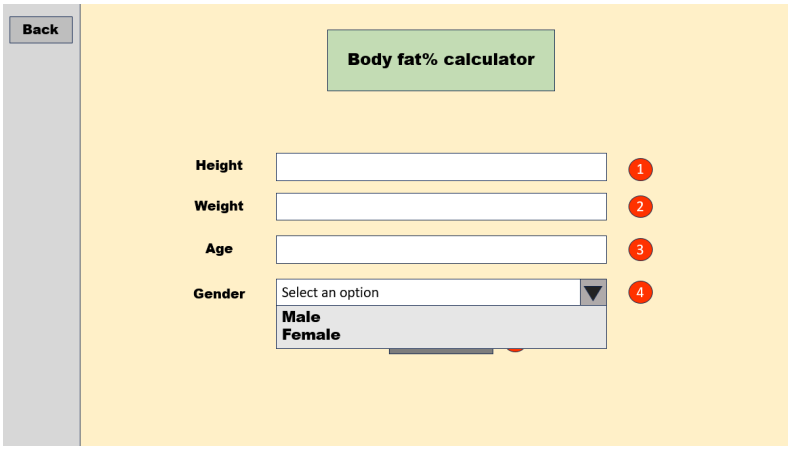


This is the calculations interface, here the user can use the body fat percentage calculator or the BMI score calculator. The details of each separate section of this interface is given below:

1. This button takes the user to the body fat percentage calculator. Font: Arial Black; Text colour: #ffffff; Font size: 28.
2. This button takes the user to the BMI score calculator. Font: Arial Black; Text colour: #ffffff; Font size: 28.
3. This button takes the user back to the Update your progress menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

15)Body fat percentage calculator



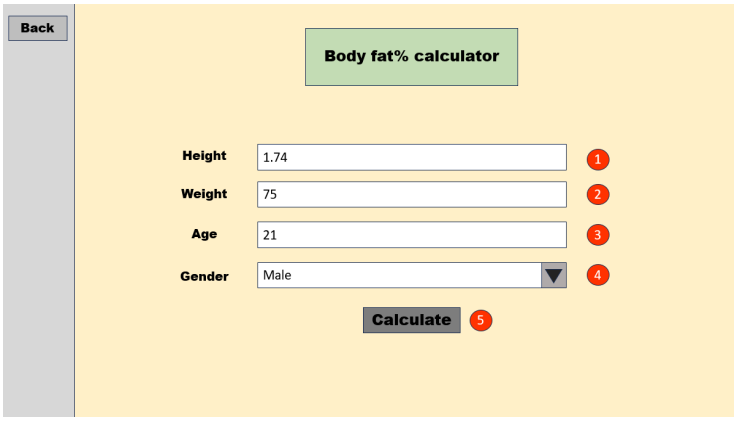


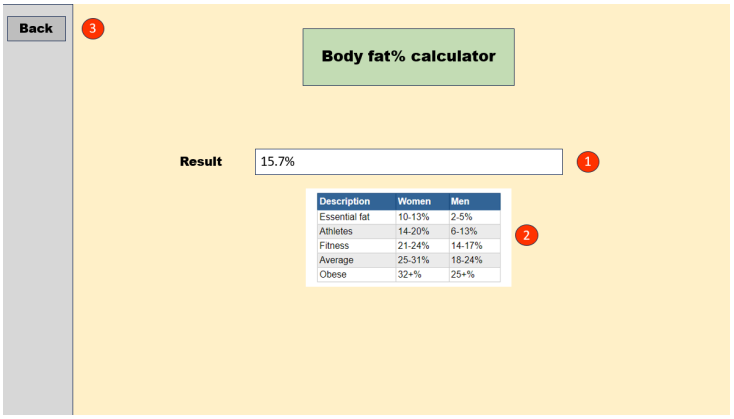
This interface takes all the inputs required to calculate the user's body fat percentage, the inputs can vary depending on the user's gender, where the calculation for a female user requires one extra data entry. The descriptions for each separate section in the interface are given below:

1. This is a text box where the user can enter their height in cm, this input must be an integer in the range 100-200. Font: Calibri; Text colour: #000000; Font size: 18.
2. This is a text box where the user can enter their weight, in kilograms, the input must be an integer, in the range 30 - 200. Font: Calibri; Text colour: #000000; Font size: 18.
3. This is a text box where the user can enter their age, in years, the input must be an integer, in the range 12-100. Font: Calibri; Text colour: #000000; Font size: 18.
4. This is a drop down list where the user can choose their gender to move forward with the calculation. The list of options drops

down by clicking on the down arrow. Font: Calibri; Text colour: #000000; Font size: 18.

16)Body fat percentage calculator results

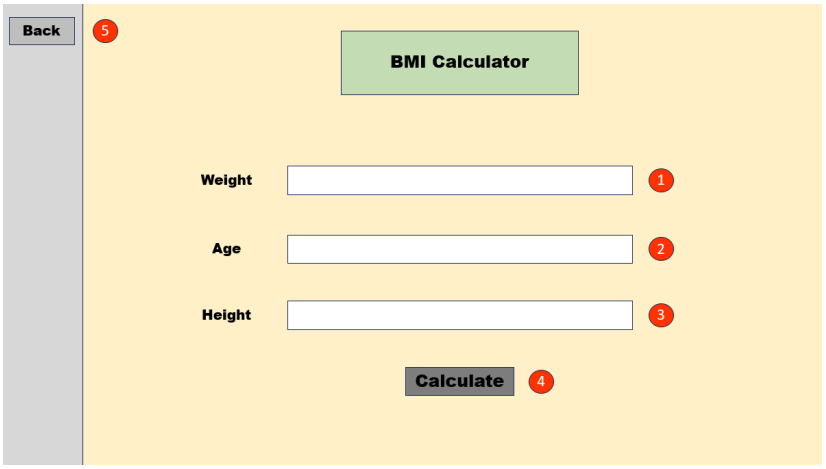




This is the interface which shows the user the results of their body fat percentage calculation. The example of entries used to produce this output is given above. These are the descriptions of each individual section of this interface:

1. This is a text box which tells the user the result of their body fat percentage calculation. Font: Calibri; Text colour: #ffffff; Font size: 18.
2. This is an image of a table which tells the user what their body fat percentage value indicates about their health.
3. This button takes the user back to the body fat percentage calculator. Font: Arial Black; Text colour: #ffffff; Font size: 16.

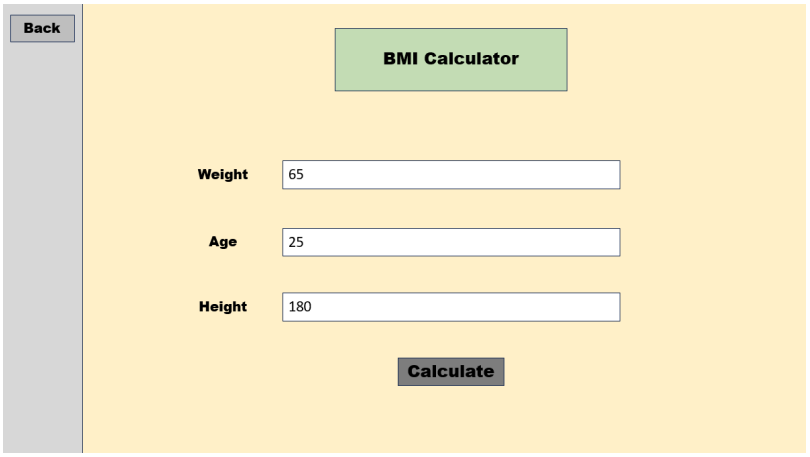
17)BMI score calculator

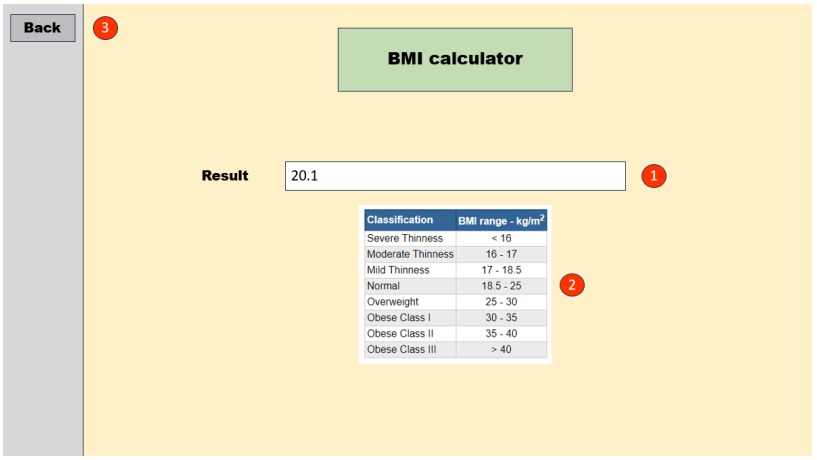


This is the BMI score calculator's interface, which takes all the necessary inputs to calculate a user's BMI score. The descriptions for each separate section in the interface are given below:

1. This is the text box where the user can input their weight in kilograms. The input must be an integer in the range 30-150. Font: Calibri; Text colour: #ffffff; Font size: 18.
2. This is the text box where the user can input their age in years. The input must be an integer in the range 12-100. Font: Calibri; Text colour: #ffffff; Font size: 18.
3. This is the text box where the user can input their height in cm. The input must be an integer in the range 100-200. Font: Calibri; Text colour: #ffffff; Font size: 18.
4. This is the button that executes the calculation once all the fields have been filled. Font: Arial Black; Text colour: #ffffff; Font size: 16.
5. This button takes the user back to the calculations menu. Font: Arial Black; Text colour: #ffffff; Font size: 16.

18)BMI score calculator results

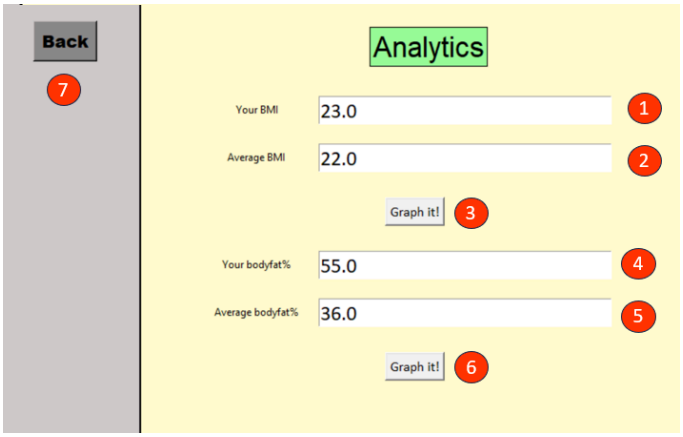


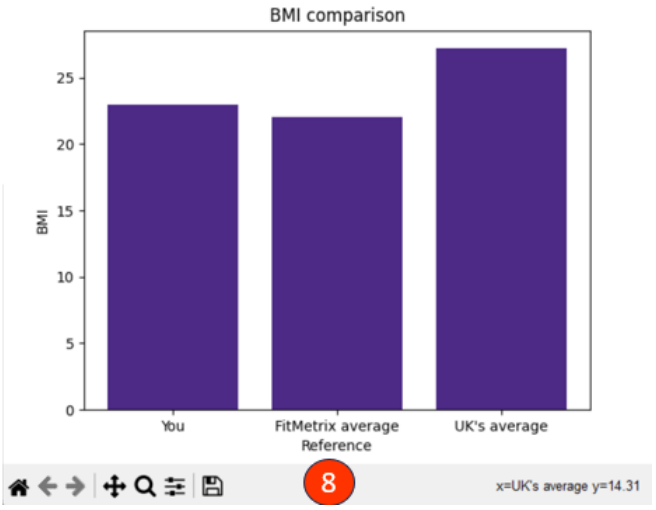


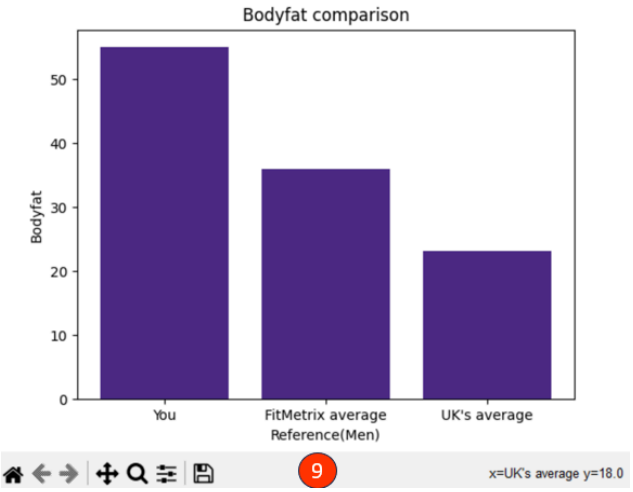
This is the interface which shows the user the results of their BMI score calculation. The example of entries used to produce this output is given above. These are the descriptions of each individual section of this interface:

1. This is a text box which tells the user the result of their BMI score calculation. Font: Calibri; Text colour: #ffffff; Font size: 18.
2. This is an image of a table which tells the user what their BMI score value indicates about their health.
3. This button takes the user back to the BMI score calculator. Font: Arial Black; Text colour: #ffffff; Font size: 16.

19)Analytics





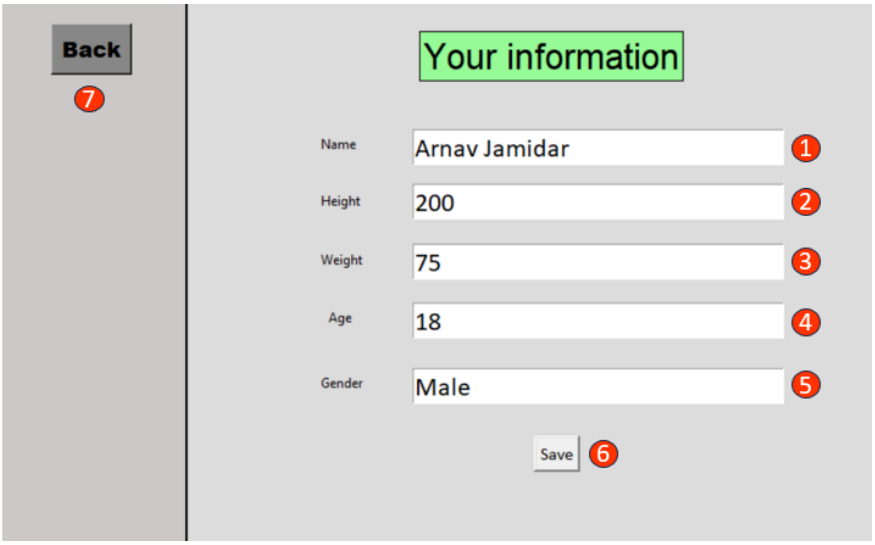


This interface will appear if the user clicks on the “Analytics” button in the Your progress menu. Here the user can choose to view the BMI comparison or/and the Bodyfat comparison graph.

The description of each individual section of this interface is as follows:

1. This is the user's BMI score, which is accessed from the system's database. Font: Calibri; Text colour: #000000; Font size: 18.
2. This is the average of all of the BMI scores stored in the system's database. Font: Calibri; Text colour: #000000; Font size: 18.
3. This is the button which when pressed opens the BMI comparison graph. Font: Calibri; Text colour: #000000; Font size:12.
4. This is the user's bodyfat percentage, which is accessed from the system's database. Font: Calibri; Text colour: #000000; Font size: 18.
5. This is the average of all of the bodyfat percentages stored in the system's database. Font: Calibri; Text colour: #000000; Font size: 18.
6. This is the button which when pressed opens the bodyfat comparison graph. Font: Calibri; Text colour: #000000; Font size:12.
7. This button takes the user back to the your progress menu. Font: Arial Black; Text colour: #000000; Font size:14.
8. This is an example of the BMI comparison graph.
9. This is an example of the bodyfat comparison graph.

20)Your details



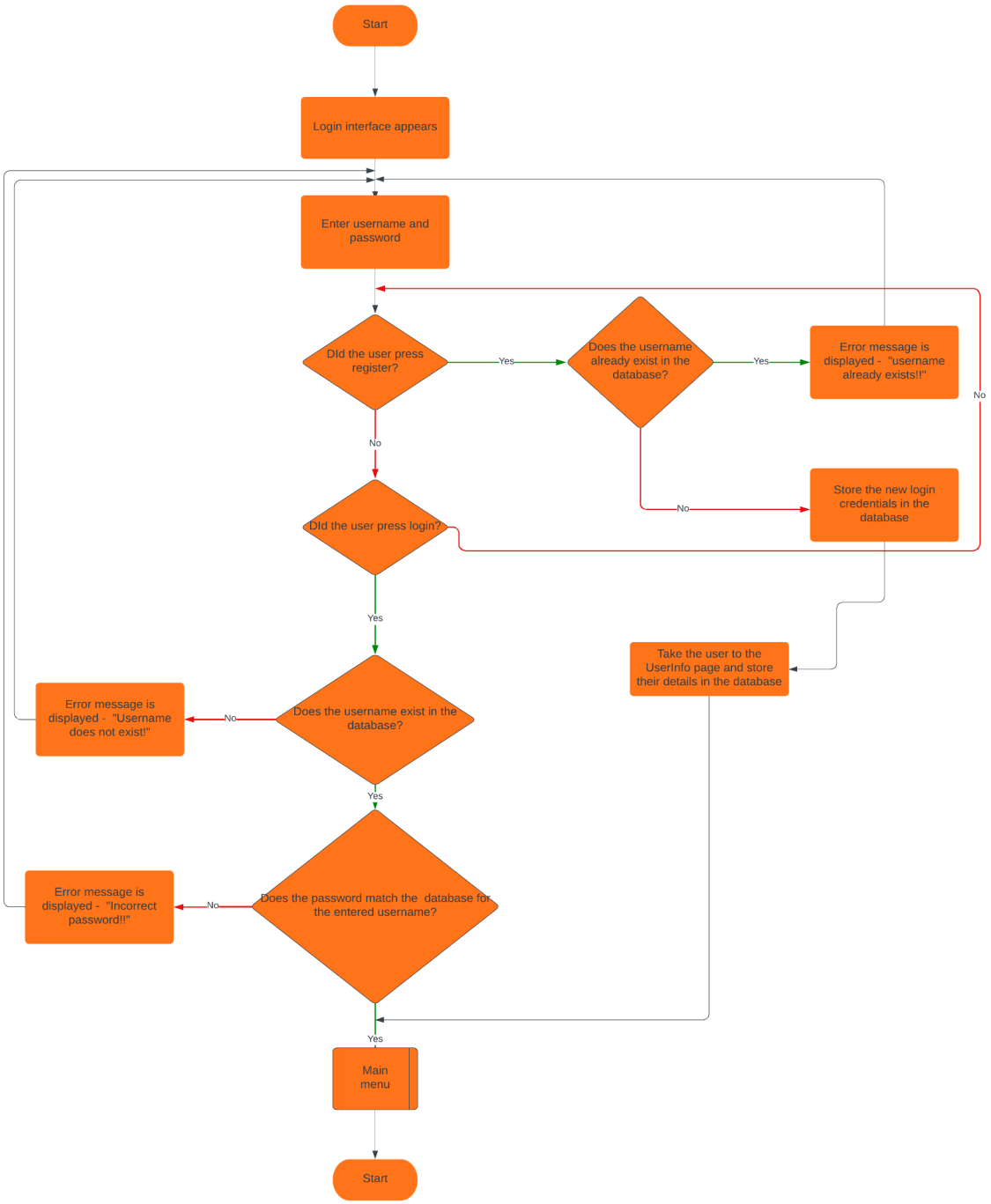
This interface will appear when the user clicks the “Your details” button in the main menu. Here the user can edit their personal details. The details for all the separate sections of this interface are given below:

1. This is the text box where the user can edit their name, the input must be a string with less than 30 characters. Font: Calibri; Text colour: #000000; Font size: 18.
2. This is the text box where the user can edit their height in centimetres. The input must be an integer in the range 100-200. Font: Calibri; Text colour: #000000; Font size: 18.
3. This is the text box where the user can edit their weight, where the input must be an integer greater than 29 and less than 200. Font: Calibri; Text colour: #000000; Font size: 18.
4. This button allows the user to edit the user’s age, where the input must be an integer in the range 12-100. Font: Arial Black; Text colour: #000000; Font size: 18.
5. This button allows the users to edit their gender, the input must be one of the two strings - “Male”, “Female”. Font: Calibri; Text colour: #000000; Font size: 18.
6. This button saves the edits made by the user into the system’s database if all the inputs are valid. Font: Calibri; Text colour: #000000; Font size: 12.
7. This button takes the user back to the main menu. Font: Arial Black; Text colour: #000000; Font size: 14.

Flowcharts

The detailed functionality of the main modules of my program are represented visually through the use of annotated flowcharts.

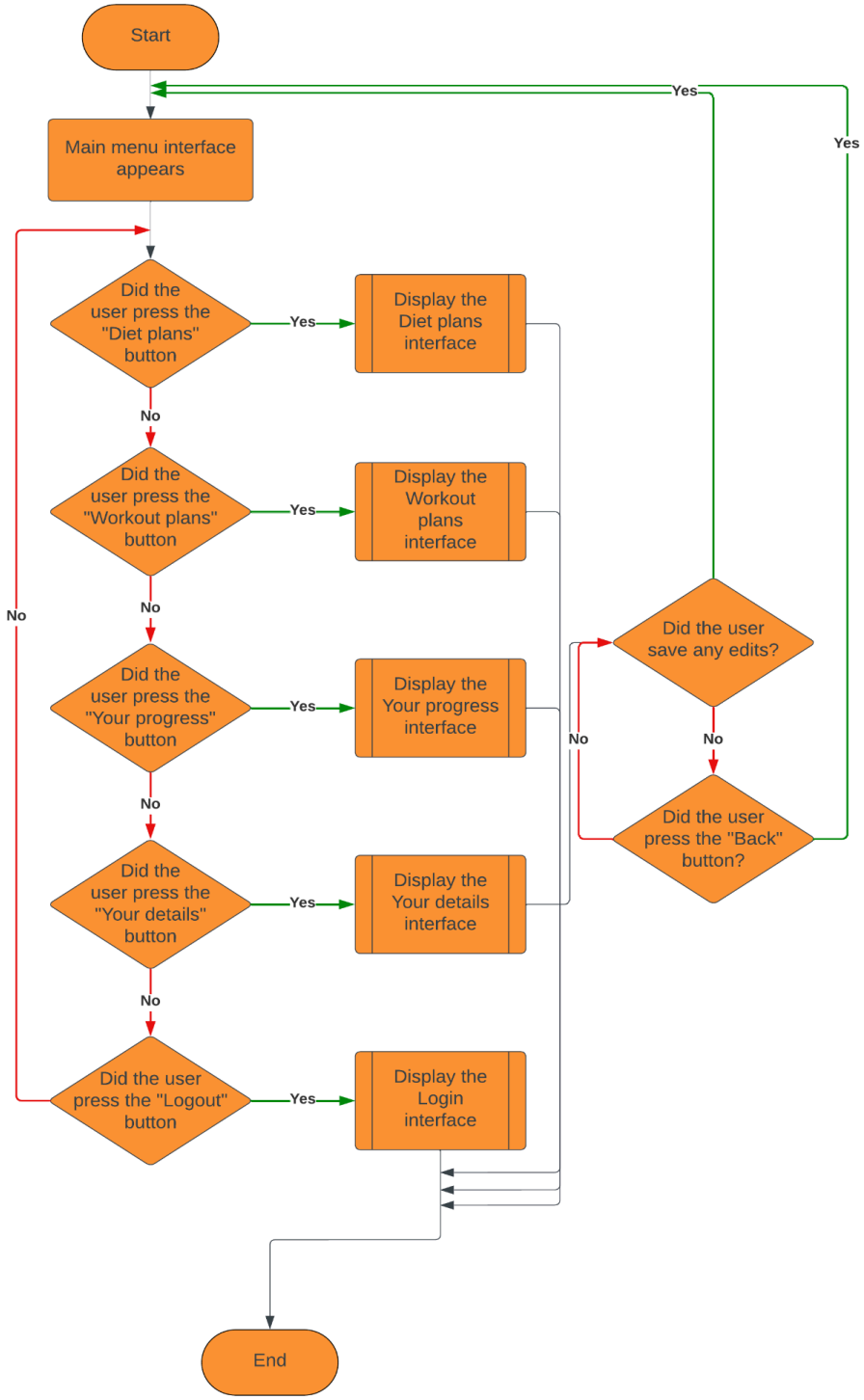
1)Login System



This flowchart will be used as the main process of gaining access to the system for a user, this is a visual representation of each stage of the login process, where the user will enter the login credentials once

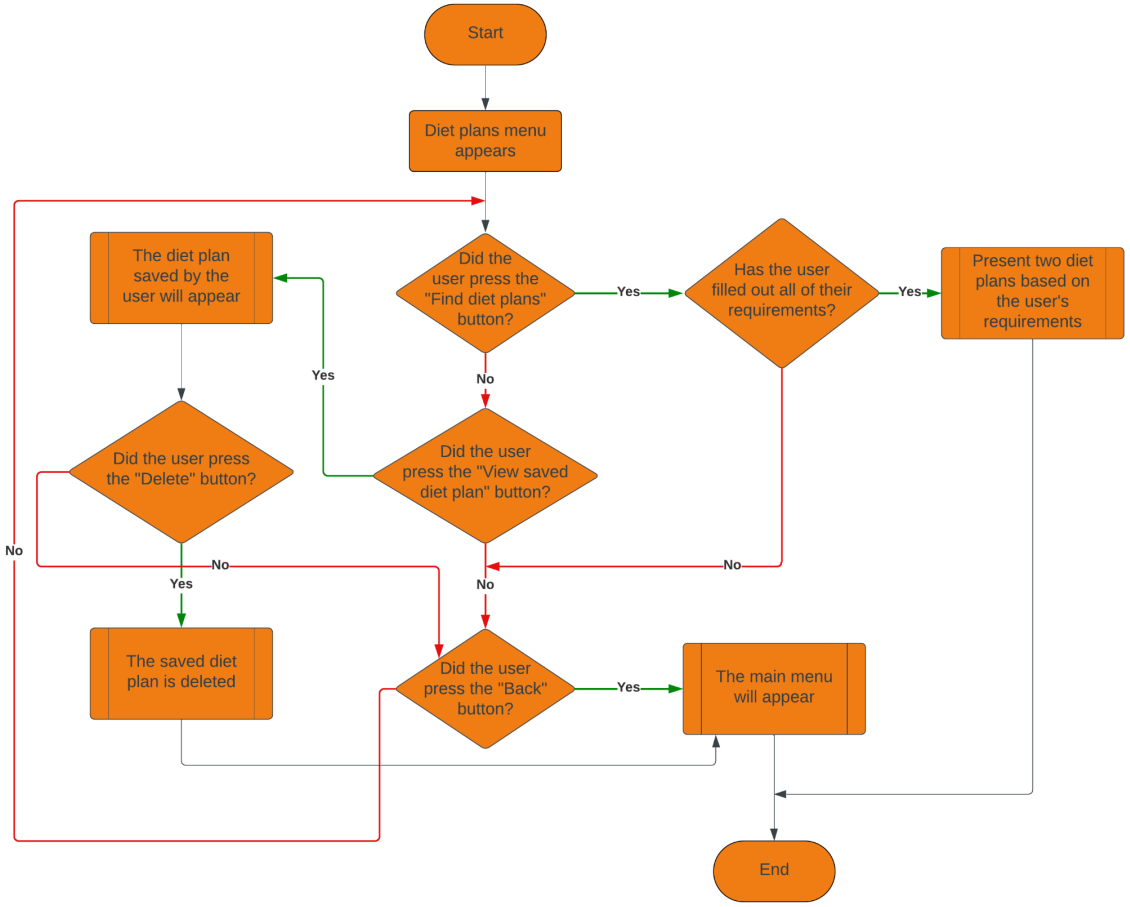
the login interface appears, then depending on whether the user chooses to register or login, different processes will be carried out to grant them access to the system. If the user chooses to register, the username entered is checked in the database on whether it already exists or not, if it doesn’t exist, the user can now register their account in the system database, therefore gaining access to the system after storing their user information through the UserInfo page and then the main menu appears. However, if the user chooses to login, the username and password are cross-checked in the systems database, if the username exists, and the password entered matches the password in the database for the corresponding username, the user gains access to the system and the main menu will appear.

2)Main menu



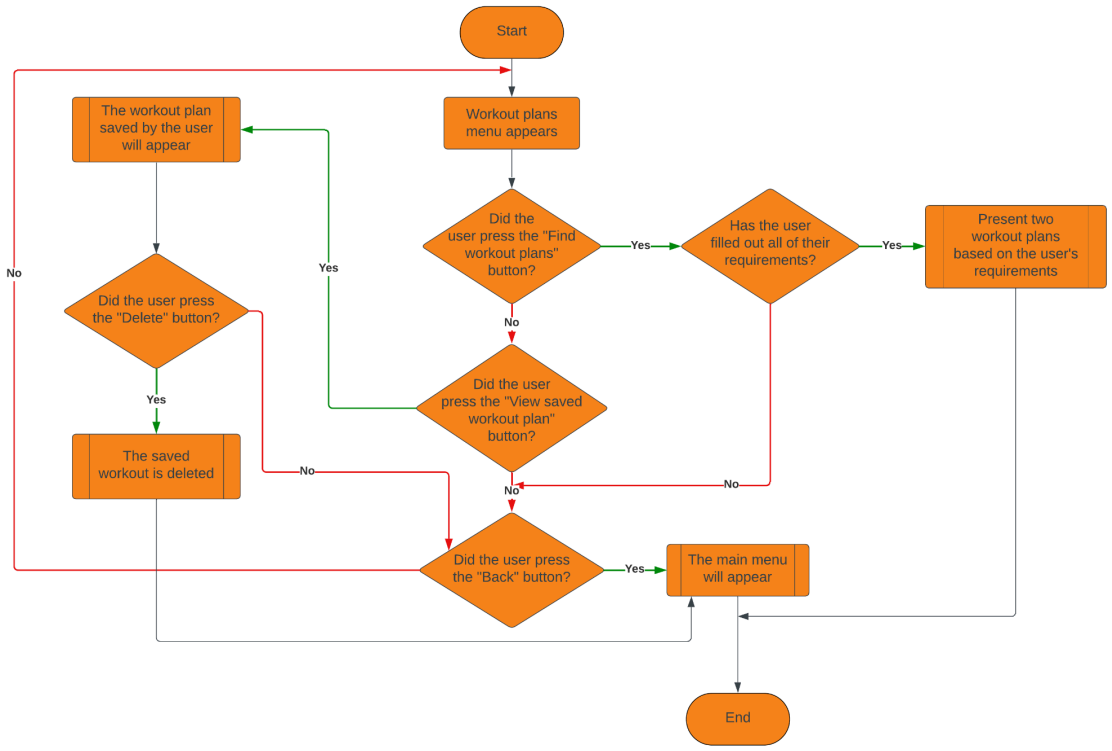
This flowchart represents the processes carried out in the main menu, where the user is presented with 5 options when they navigate through the main menu. After pressing any button in the main menu, a corresponding separate interface will appear, for example, if the user presses the workout plans button, the workout plans menu will appear.

3)Diet plans menu



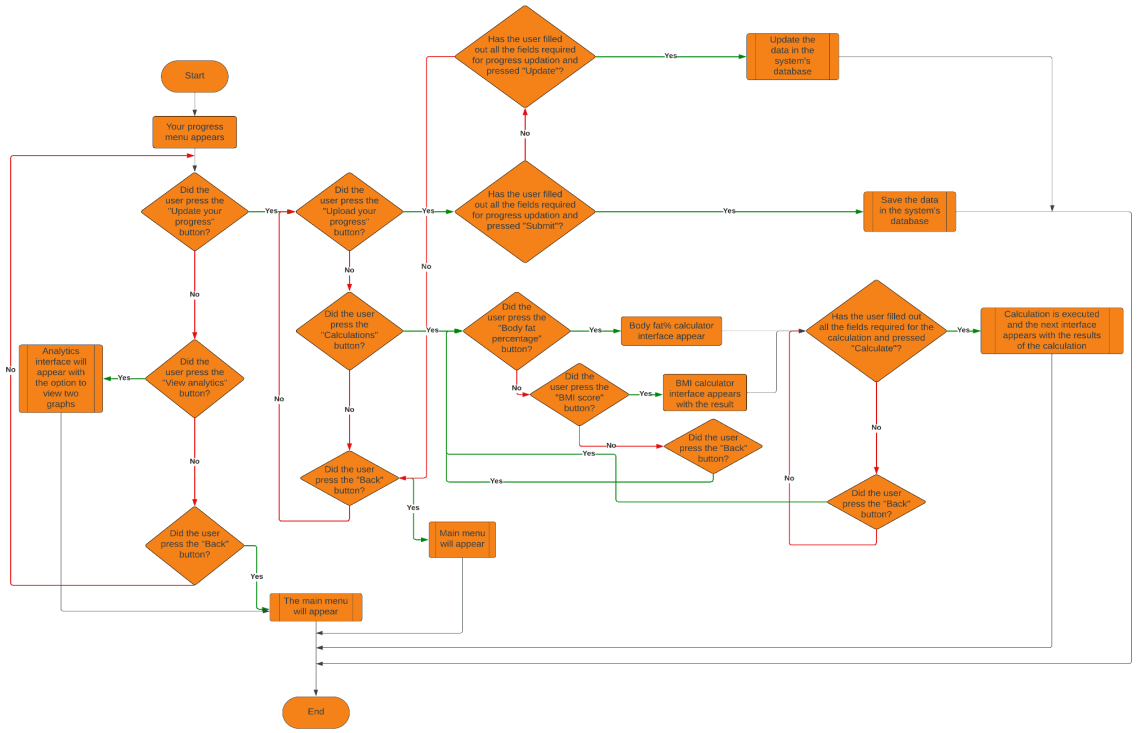
This flowchart shows how the user can interact with the diet plans menu interface. In this interface, the user is presented with three initial options to choose from - go back to the main menu, find diet plans or view saved diet plans. To find diet plans, the user has to fill out their requirements to search for personalised diet plans, after doing so, the system will present the user with two diet plans. The user can save one of these two diet plans which can then be later viewed by pressing the “View saved diet plan” button in the diet plans menu, the user can also delete the saved plan by pressing the “Delete” button in this interface.

4)Workout plans menu



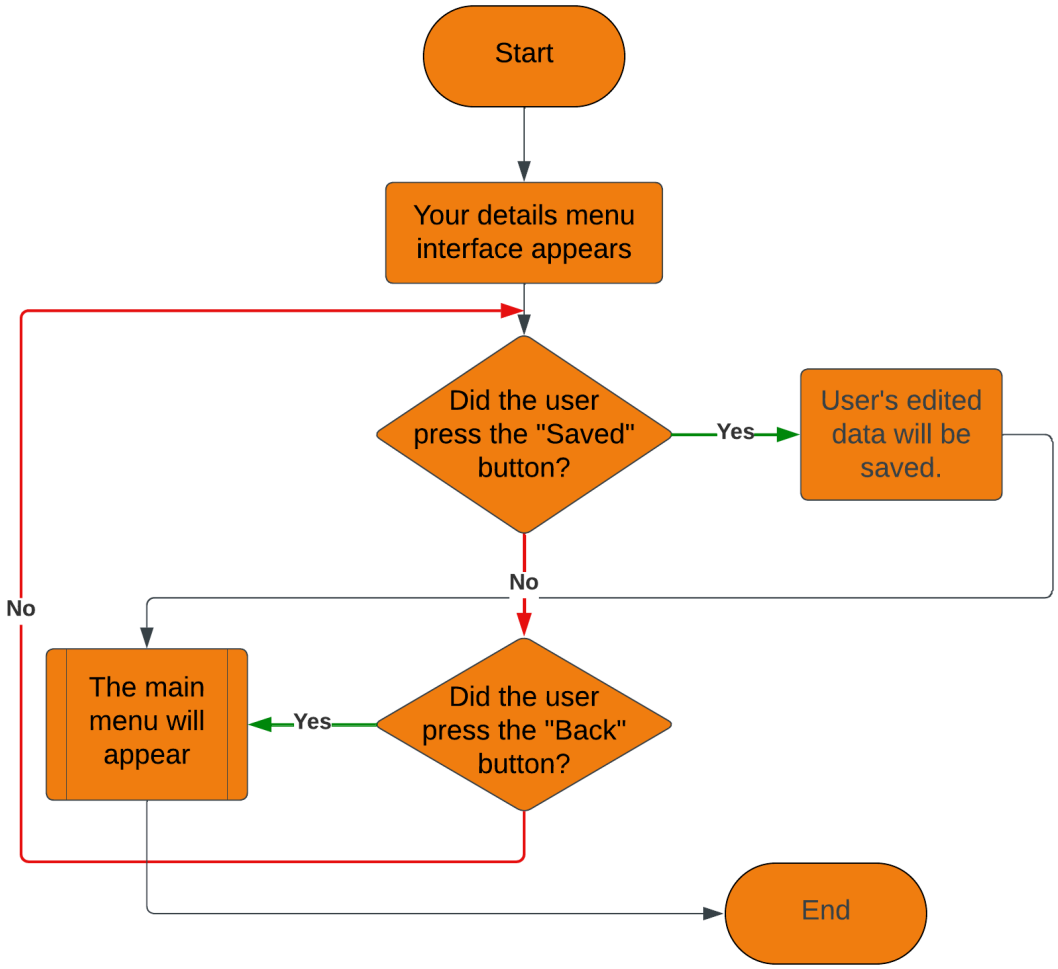
This flowchart represents the functionalities of the workout plans menu where the user is presented with three options to choose from in the menu - go back to the main menu, find workout plans or view saved workout plan. To find workout plans, the user has to fill out their requirements to search for personalised workout plans, after doing so, the system will present the user with two workout plans. The user can save one of these two workout plans which can then be later viewed by pressing the “View saved workout plan” button
in the workout plans menu, the user can also delete the saved plan by pressing the “Delete” button in this interface.

5)Your progress menu



This flowchart describes how the user can navigate through the Your progress menu, where the user is presented with three options in the first interface - “Update your progress” button or “View analytics” button or “Back”. If the user chooses to press the “Update your progress” button, they can either choose to update their progress by filling out all the fields required for progress updation and pressing submit afterwards, or they can choose to carry out one of the two calculations - Body fat percentage or BMI score, these calculations can be carried out by filling all the required details and information for the corresponding calculators. If the user chooses to press the “View analytics” button, they are presented with the analytics interface where they can choose to view either the BMI comparison or bodyfat% comparison graph. If the user presses the “Back” button, the main menu interface will appear.

6)Your details menu



This flowchart represents the functionalities of the Your details menu. In this interface the user can modify their details in the text boxes, then save the new details in the system's database by clicking the "Save" button. If the user clicks the "Back" button, the main menu interface will appear.

Table definitions

credentials					
Field Name	**Field Purpose**	**Field Type**	**Field Size**	**Example Data**	**Validation**
username	To store the user's password	String	8	Pete123	Field must not be blank
hashpassword	To store the user's hashed-password	String	10	CSbest4507	Field must not be left blank
name	To store the user's name	String		Arnav Jamidar	Field must not be left blank
age	To store the user's age	Integer	2	17	Field must not be left blank
gender	To store the user's gender	String	6	Male	Field must not be left blank
height	To store the user's height	Integer	3	175	Field must not be left blank
weight	To store the user's weight	Integer	3	75	Field must not be left blank

userDiet					
Field Name	**Field Purpose**	**Field Type**	**Field Size**	**Example Data**	**Validation**
username (Foreign key)	To store the username	String	8	Pete123	Field must not be left blank
dietID (Foreign key)	To store a diet plan's ID	Integer	2	2	Field must not be left blank

Diets					
Field Name	**Field Purpose**	**Field Type**	**Field Size**	**Example Data**	**Validation**
dietID	To store a diet plan's ID	Integer	2	2	Field must not be left blank

diet	To store diet plans	String		Breakfast: 2 Poached eggs, salmon and avocado - 550 calories Lunch: Double chicken breast, broccoli and rice - 700 calories Dinner: Tinned tuna, quinoa, avocado and broccoli - 500 calories Pre-workout snack: 2 scoops of whey protein with 400 ml of water Total macronutrients : 1970 calories 227 grams of protein 81 grams of carbohydrates 53 grams of fats	Field must not be left blank

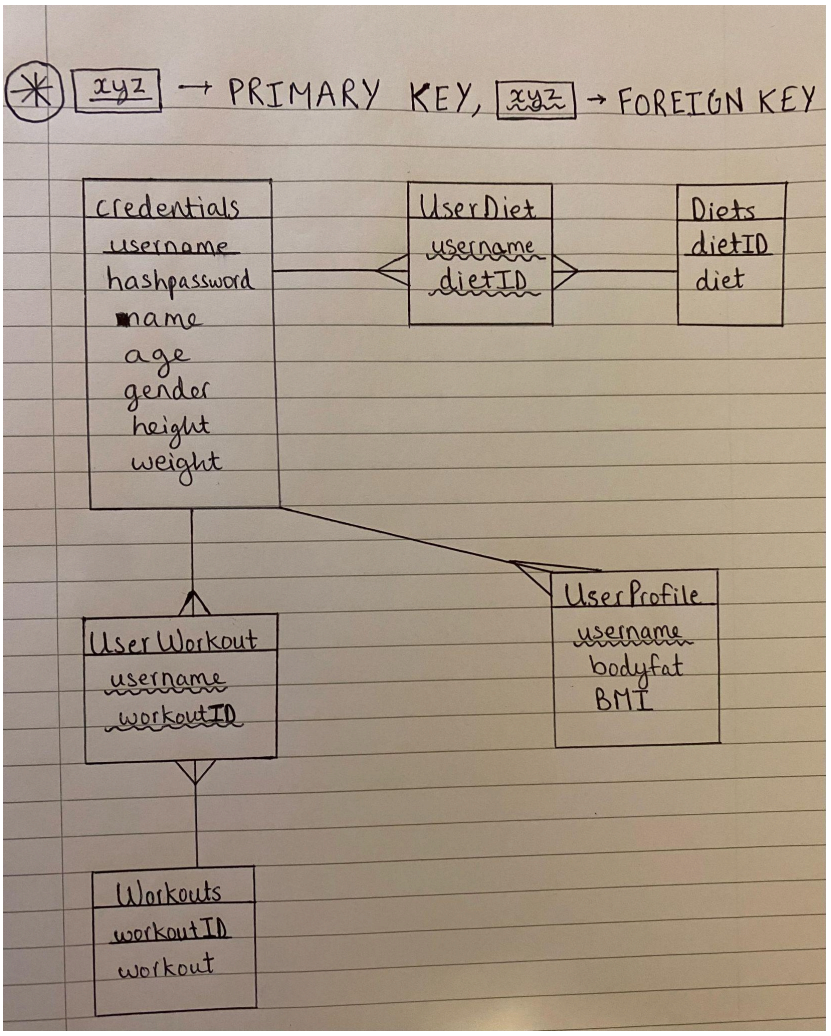
userWorkout					
Field Name	**Field Purpose**	**Field Type**	**Field Size**	**Example Data**	**Validation**
username (Foreign key)	To store the username	String	8	Pete123	Field must not be left blank
workoutID (Foreign key)	To store a workout plan's ID	Integer	2	2	Field must not be left blank

Workouts					
Field Name	**Field Purpose**	**Field Type**	**Field Size**	**Example Data**	**Validation**
workoutID	To store a workout plan's ID	Integer	2	2	Field must not be left blank
workout	To store workout plans	String		Day 1: Upper body\nChest: flat barbell bench press — 4 sets of 6-8 reps\nBack: bent-over barbell rows — 3 sets of 6-8 reps\nShoulders: seated dumbbell press — 3 sets of 8-10 reps\nChest/triceps: dips — 3 sets of 8-10 reps\nBack: pullups or lat pulldowns — 3 sets of 8-10 reps\nBiceps: incline dumbbell curls — 3 sets of 10-12 reps\n\nDay 2: Lower body\nLegs: barbell back squats — 4 sets of 6-8 reps\nLegs: leg press — 3 sets of 8-10 reps\nQuadriceps: seated leg extensions — 3 sets of 10-12 reps\nCalves: calf press on leg press — 4 sets of 12-15 reps\nAbs: decline crunches — 4 sets of 12-15 reps\n\nRepea	Field must not be left blank

				t this circuit two times	

userProfile					
Field Name	**Field Purpose**	**Field Type**	**Field Size**	**Example Data**	**Validation**
username (Foreign key)	To store the username	String	8	Pete123	Field must not be left blank
bodyfat	To store a user's bodyfat%	Float	2	15	Field must not be left blank
BMI	To store a user's BMI	Float	2	23.2	

Entity Relationship Diagram for the database



The ERD for my system's database has six tables - credentials has a one to many relationship with UserDiet, UserWorkout & UserProfile; Workouts has a one to many relationship with UserWorkout; Diets has a one to many relationship with UserDiet.

The table credentials is responsible for storing the user's details which are determined when they register into the system. The tables - credentials, UserDiet and Diets work in relation to fulfil the functionalities of the Diet menu in the program. The tables - credentials, UserWorkout and Workouts work in relation to fulfil the functionalities of the Workout menu in the program. The tables - credentials and UserProfile work in relation to fulfil the functionalities of the Your progress menu.

Potential algorithms

These are examples of potential SQL algorithms that I might use to create this system's database:

1)Establishing connection with database

To establish a connection with the database whilst using Python, the following commands will be used.

```
conn = sqlite3.connect("database.db")
c = conn.cursor()
```

The conn variable executes the sqlite command connect and saves the active connection. The variable c executes the sqlite command cursor through the conn variable. The cursor is used to navigate the database just like a mouse pointer is used on a computer.

2)SELECT

The select statement is used to select groups of data in a database. It can be used to search for individual records or to search for groups containing a certain value.

```
SELECT * FROM userDetails
```

This query selects all the records in the database from the table userDetails. In sql * means all

```
SELECT * FROM userDetails WHERE age = 21
```

This query selects all the records in the database from the table userDetails where the player is aged 21.

3)INSERT

The insert function is used to insert new records into the database.

```
INSERT INTO userDetails VALUES('Pete Temperton',31,
'07456992785')
```

This SQL query inserts the values in the brackets to a new record in the table playerDetails inside the database

```
INSERT INTO userDetails VALUES('Pete',31, '07456')
```

This SQL query inserts the values in the brackets to a new record in the table playerDetails inside the database

4)UPDATE

The update function is used to update existing records inside the database.

```
UPDATE userDetails SET age = 23 WHERE name = 'Pete
Temperton'
```

This SQL statement updates all records with the name of Pete Temperton by changing the age to 23.

5)DELETE

The delete function is used to delete existing records inside a database.

```
DELETE FROM playerDetails WHERE name = 'Pete Temperton'
```

This SQL query deletes all the records in the table playerDetails from the database with the name Pete Temperton.

```
DELETE FROM fixtures WHERE name = 'Pete'
```

This SQL statement deletes all the records in the table playerDetails from the database with the name Pete.

6)Commit

After a statement is executed, the database must be saved. To do this, the commit function is called

```
Data = database.db
data.commit()
```

Once commit is called, the computer saves the database.

7)Close

The close function is used to close the database when it is no longer in use. This allows the database to be accessed by other programs.

```
Data = database.db
data.close()
```

Once close is called, the computer closes the connection the system has with the database.

This is an example of the hashing algorithm I'll be using in my program:

```
def custom_hash(message):
    # Define the initial hash value
    hash_value = 0

    # Iterate over each character in the message
    for char in message:
        # Convert the character to its ASCII value
        char_value = ord(char)

        # Update the hash value
        hash_value = (hash_value + char_value)%256

    return hash_value

# Example usage
message = input("Input message: ")
hashed_value = custom_hash(message)
```

```
print("Message:", message)
print("Hashed value: ", hashed_value)
```

Hash tables

Hash tables are a type of data structure in which the address or the index value of the data element is generated from a hash function. That makes accessing the data faster as the index value behaves as a key for the data value. In other words Hash table stores key-value pairs but the key is generated through a hashing function.

So the search and insertion function of a data element becomes much faster as the key values themselves become the index of the array which stores the data.

The data dictionary types represent the implementation of hash tables. The keys in the dictionary satisfy the following requirements.

- The keys of the dictionary are hashable i.e. they are generated by a hashing function which generates a unique result for each unique value supplied to the hash function.
- The order of data elements in a dictionary is not fixed.

So we see the implementation of a hash table by using the dictionary data types as below.

Accessing values in a dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value.

```
# Declare a dictionary
dict={'Name':'Zara','Age':7,'Class':'First'}
# Accessing the dictionary with its key
print("dict['Name']: ",dict['Name'])
print("dict['Age']: ",dict['Age'])
```

Updating dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

```
# Declare a dictionary
dict={'Name':'Zara','Age':7,'Class':'First'}
dict['Age']=8
# update existing entry
dict['School']="DPS School"
# Add new entry
print("dict['Age']: ",dict['Age'])
print("dict['School']: ", dict['School'])
```

Deleting dictionary elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete an entire dictionary in a single operation.To explicitly remove an entire dictionary, just use the del statement.

```
dict={'Name':'Zara','Age':7,'Class':'First'}
del dict['Name']# remove entry with key 'Name'
dict.clear()# remove all entries in dict
del dict# delete entire dictionary
print("dict['Age']: ",dict['Age'])
print("dict['School']: ",dict['School'])
```

Implementation

Login_Page.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from HashingAlgorithm import *
import Main_Menu as MainMenu
import mySQL as SQL
import UserInfo_Page as UserInfo
import GlobalVariables

#Creating the window.
class Login(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Login page")
        self.geometry("800x500")
        self.configure(bg="#C1FFC1")

        self.main=Main(self)

        self.mainloop()

#Creating Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#C1FFC1")
        self.place(relx=.5, rely=.5,anchor= CENTER)
        self.pack()

        self.check_db_existense()
        self.create_widgets()

    def create_widgets(self):
        self.login_label = tk.Label(self, text = "FitMetrix",
bg="#FCE6C9", fg="#000000", font=("Arial", 30),
highlightbackground="black", highlightthickness=1,
highlightcolor="black")#fg = font colour ; font = font("fontType",
size)
        self.username_label = tk.Label(self, text = "Username",
bg="#C1FFC1", fg="#000000", font=("Arial", 16))
        self.password_label = tk.Label(self, text = "Password",
bg="#C1FFC1", fg="#000000", font=("Arial", 16))
        self.username_entry = tk.Entry(self, font=("Calibri", 18))
```

```
        self.password_entry = tk.Entry(self, show = "*",
font=("Calibri", 18))
        self.login_button = tk.Button(self, text = "Login",
bg="#A9A9A9", fg="#FFFFFF", font=("Arial", 16), command=self.login)
        self.register_button = tk.Button(self, text = "Register",
bg="#A9A9A9", fg="#FFFFFF", font=("Arial", 16), command=lambda:
self.register())
        self.logo=PhotoImage(file="C:/Users/iamar/OneDrive/Pictures/A
LEVELS/NEA/LOGO FITMETRIX.png")
        self.logo_label = tk.Label(self, image=self.logo,
bg="#C1FFC1")

        #Placing widgets on the screen
        self.login_label.grid(row=0, column=1, columnspan=1,
sticky="news", pady=10)
        #pad=padding ; pady = change in y-axis ; padx = change in
x-axis.
        #sticky - a property of the grid where u tell it to take as
much space as possible in direction (news=north,east,west,south)
        self.logo_label.grid(row=1, column=1, columnspan=1,
sticky="news")
        self.username_label.grid(row=2, column=0)
        self.username_entry.grid(row=2, column=1, pady=10)
        self.password_label.grid(row=3, column=0)
        self.password_entry.grid(row=3, column=1, pady=10)
        self.login_button.grid(row=4, column=0, columnspan=2, pady=20,
sticky="n")
        self.register_button.grid(row=4, column=0, columnspan=2,
pady=20, sticky="e")

    #Checking database exists or not. If yes, database is opened
otherwise database is created and data is added into the database.
    def check_db_existense(self):
        try:
            f = open("credentials_book.db")
        except:
            SQL.create_users()
            SQL.addDietPlan(1,GlobalVariables.diet1)
            SQL.addDietPlan(2,GlobalVariables.diet2)
            SQL.addDietPlan(3,GlobalVariables.diet3)
            SQL.addDietPlan(4,GlobalVariables.diet4)
            SQL.addWorkoutPlan(1,GlobalVariables.workout1)
            SQL.addWorkoutPlan(2,GlobalVariables.workout2)
            SQL.addWorkoutPlan(3,GlobalVariables.workout3)
            SQL.addWorkoutPlan(4,GlobalVariables.workout4)
            SQL.addWorkoutPlan(5,GlobalVariables.workout5)
            SQL.addWorkoutPlan(6,GlobalVariables.workout6)
            SQL.addWorkoutPlan(7,GlobalVariables.workout7)
            SQL.addWorkoutPlan(8,GlobalVariables.workout8)
            SQL.create_UserDiet()
            SQL.create_UserWorkout()
            SQL.create_UserProfile()
```

```
    #Adding login functionality to the interface.
    def login(self):
        hashedpass = SQL.getHashedPassword(self.username_entry.get())
        if
self.username_entry.get()==SQL.Username_get2(self.username_entry.get()
):
            if int(custom_hash(self.password_entry.get())) ==
int(hashedpass):
                messagebox.showinfo(title="Login Success",
message="Successfully logged in.")#messagebox creates a pop up
message.
                #showinfo - displays info(positive outcome of a
process)
                GlobalVariables.username = self.username_entry.get()

                self.master.destroy()
                MainMenu.App()
            else:
                messagebox.showerror(title="Login Failure",
message="Incorrect password")
                #showerror - displays error(negative outcome of a
process)
        else:
            messagebox.showerror(title="Login Failure",
message="Username doesn't exist")

    #Adding register functionality to the interface.
    def register(self):
        Username_count=[]
        Password_count=[]
        NewUN=self.username_entry.get()
        NewUN=[*NewUN]#The method - [*NewUN] separate the input to
list of letters, eg: input -> ["i","n","p","u","t"]
        NewPass=self.password_entry.get()
        NewPass=[*NewPass]
        for x in NewUN:
            Username_count.append(x)
        for i in NewPass:
            Password_count.append(i)
        if len(Username_count)>=5 and len(Username_count)<=11 and
len(Password_count)>2 and len(Password_count)<20:
            if SQL.Username_get(self.username_entry.get()) == []:
                messagebox.showinfo(title="Registration Success",
message="Successfully registered\nNow we just need to know a bit more
about you.")
                hashedpass=custom_hash(self.password_entry.get())

SQL.register_users(self.username_entry.get(),hashedpass,"","","","","
)
                GlobalVariables.username = self.username_entry.get()
```

```
                self.master.destroy()
                UserInfo.App()
            else:
                messagebox.showerror(title="Registration Failure",
message="Username exists")
                self.username_entry.delete(first=0, last=END)
                self.password_entry.delete(first=0, last=END)
        else:
            messagebox.showerror(title="Registration Failure",
message="Invalid login credentials")

if __name__ == "__main__":
    Login()
```

UserInfo_Page.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Main_Menu as MainMenu
import Login_Page as LoginPage
import mySQL as SQL
from HashingAlgorithm import *
import GlobalVariables

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("User Info")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options menu.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
```

```
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            MainMenu.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFE4E1", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_dropList()
        self.create_entries()
        self.create_button()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Your information",
bg="#98FB98", fg="#000000", font=("Arial", 25),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.53, rely=.1, anchor=CENTER)

        self.name_label=tk.Label(self, text="Name", bg="#FFE4E1")
        self.name_label.place(relx=0.19,rely=0.240)

        self.height_label=tk.Label(self, text="Height(cm)",
bg="#FFE4E1")
        self.height_label.place(relx=0.19,rely=0.345)

        self.weight_label=tk.Label(self, text="Weight(kg)",
```

```
bg="#FFE4E1")
        self.weight_label.place(relx=0.19,rely=0.453)

        self.age_label=tk.Label(self, text="Age", bg="#FFE4E1")
        self.age_label.place(relx=0.2,rely=0.56)

        self.gender_label=tk.Label(self, text="Gender", bg="#FFE4E1")
        self.gender_label.place(relx=0.19, rely=0.68)

#Creating the drop-down list.
    def create_dropList(self):

        self.gender_options=["Male","Female"]
        self.clicked_gender=StringVar()
        self.clicked_gender.set("Select gender")
        self.preference_drop=OptionMenu(self, self.clicked_gender,
*self.gender_options)
        self.preference_drop.config(width=50)
        self.preference_drop.place(relx=.6, rely=0.7, anchor=CENTER)


#Creating the entries.
    def create_entries(self):

        self.name=tk.Entry(self,font=("Calibri",18))
        self.name.config(width=28)
        self.name.place(relx=.328, rely=0.235)

        self.height=tk.Entry(self, font=("Calibri",18))
        self.height.config(width=28)
        self.height.place(relx=.328, rely=0.335)

        self.weight=tk.Entry(self, font=("Calibri",18))
        self.weight.config(width=28)
        self.weight.place(relx=.328, rely=0.445)

        self.age=tk.Entry(self, font=("Calibri",18))
        self.age.config(width=28)
        self.age.place(relx=.328, rely=0.555)

#Creating the submit button.
    def create_button(self):
        submit=tk.Button(self, text="Submit", font=("Calibri", 12),
command=self.info)
        submit.place(relx=0.5, rely=0.8)

#Implementing the info function which saves a user's details when they
register onto the program.
    def info(self):

        self.username = GlobalVariables.username
        try:
```

```
            x = SQL.Username_get2(self.username)
            if (len([*self.name.get()])<30) and
(int(self.height.get())>=100 and int(self.height.get())<=200) and
(int(self.age.get())>=12 and int(self.age.get())<=100) and
(int(self.weight.get())>=30 and int(self.weight.get())<=200):
                SQL.updateInfo(self.username, self.name.get(),
self.age.get(), self.clicked_gender.get(), self.height.get(),
self.weight.get())
                messagebox.showinfo(title="Registered", message="Thank
you for your co-operation\nYou are now a part of the FitMetrix
community")
                self.master.destroy()
                MainMenu.App()
            else:
                messagebox.showerror(title="Registration failure",
message="Invalid inputs")
        except:
            messagebox.showerror(title="Error", message="Invalid
inputs")
```

Main_Menu.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
import Login_Page as Login
import Workout_Menu as Workout
import Diet_Menu as DietMenu
import YourDetails_Page as YourDetails
import YourProgress_Menu as YourProgress


#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Main menu")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating Options Frame.
```

```
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.logout_button = tk.Button(self, bg="#8B8989",
text="Logout", font=("Arial Black", 14), command=self.logout)
        self.yourdetails_button = tk.Button(self, bg="#8B8989",
text="Your details", font=("Arial Black", 14),
command=self.yourdetails)

        #placing widgets
        self.logout_button.place(x=35, y=20)
        self.yourdetails_button.place(x=10, y=425)

    def logout(self):
        messagebox.showinfo(title="Logout", message="Successfully
logged out")
        try:
            self.master.destroy()
            Login.Login()
        except:
            pass

    def yourdetails(self):
        try:
            self.master.destroy()
            YourDetails.App()
        except:
            pass

#Creating Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#B9D3EE", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_buttons()
```

```
#Creating the labels.
    def create_labels(self):
        self.logo=PhotoImage(file="C:/Users/iamar/OneDrive/Pictures/A
LEVELS/NEA/LOGO FITMETRIX BLUE.png")
        self.main_label = tk.Label(self, text = "FitMetrix",
bg="#FCE6C9", fg="#000000", font=("Arial", 30),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.47, rely=.1, anchor=CENTER)
        self.logo_label = tk.Label(self, image=self.logo,
bg="#B9D3EE")
        self.logo_label.place(relx=.47, rely=.28,anchor=CENTER)

#Creating the buttons.
    def create_buttons(self):
        self.diet_button = tk.Button(self, text="Diet plans",
bg="#CDC9C9", font=("Arial black", 18), height=3,
command=self.dietplans)
        self.diet_button.place(relx=.15, rely=.55,anchor=CENTER)
        self.workout_button = tk.Button(self, text="Workout plans",
bg="#CDC9C9", font=("Arial black", 18), height=3,
command=self.workoutplans)
        self.workout_button.place(relx=.465, rely=.55,anchor=CENTER)
        self.progress_button = tk.Button(self, text="Your progress",
bg="#CDC9C9", font=("Arial black", 18), height=3,
command=self.progress)
        self.progress_button.place(relx=.82, rely=.55,anchor=CENTER)

#Opening diet menu.
    def dietplans(self):
        try:
            self.master.destroy()
            DietMenu.App()
        except:
            pass

#Opening workout menu.
    def workoutplans(self):
        try:
            self.master.destroy()
            Workout.App()
        except:
            pass

#Opening your progress menu.
    def progress(self):
        try:
            self.master.destroy()
            YourProgress.App()
        except:
            pass
```

YourDetails_Page.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Main_Menu as MainMenu
import Login_Page as LoginPage
import mySQL as SQL
from HashingAlgorithm import *
import GlobalVariables

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Your details")
        self.geometry("800x500")
        self.configure(bg="#FFFAFA")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
```

```
        try:
            self.master.destroy()
            MainMenu.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#DCDCDC", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_entries()
        self.create_button()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Your information",
bg="#98FB98", fg="#000000", font=("Arial", 25),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.53, rely=.1, anchor=CENTER)

        self.name_label=tk.Label(self, text="Name", bg="#DCDCDC")
        self.name_label.place(relx=0.19,rely=0.240)

        self.height_label=tk.Label(self, text="Height", bg="#DCDCDC")
        self.height_label.place(relx=0.19,rely=0.345)

        self.weight_label=tk.Label(self, text="Weight", bg="#DCDCDC")
        self.weight_label.place(relx=0.19,rely=0.453)

        self.age_label=tk.Label(self, text="Age", bg="#DCDCDC")
        self.age_label.place(relx=0.2,rely=0.56)

        self.gender_label=tk.Label(self, text="Gender", bg="#DCDCDC")
        self.gender_label.place(relx=0.19, rely=0.68)

#Creating the entries.
    def create_entries(self):

        self.username = GlobalVariables.username

        self.Name = str(SQL.getName(self.username))
        self.name=tk.Entry(self,font=("Calibri",18))
        self.name.insert(0,self.Name)
        self.name.config(width=28)
        self.name.place(relx=.328, rely=0.235)
```

```
        self.Height = SQL.getHeight(self.username)
        self.height=tk.Entry(self, font=("Calibri",18))
        self.height.insert(0,self.Height)
        self.height.config(width=28)
        self.height.place(relx=.328, rely=0.335)

        self.Weight = SQL.getWeight(self.username)
        self.weight=tk.Entry(self, font=("Calibri",18))
        self.weight.insert(0,self.Weight)
        self.weight.config(width=28)
        self.weight.place(relx=.328, rely=0.445)

        self.Age = SQL.getAge(self.username)
        self.age=tk.Entry(self, font=("Calibri",18))
        self.age.insert(0,self.Age)
        self.age.config(width=28)
        self.age.place(relx=.328, rely=0.555)

        self.Gender = SQL.getGender(self.username)
        self.gender=tk.Entry(self, font=("Calibri",18))
        self.gender.insert(0,self.Gender)
        self.gender.config(width=28)
        self.gender.place(relx=.328, rely=0.675)

#Creating the save button.
    def create_button(self):
        self.save=tk.Button(self, text="Save", font=("Calibri", 12),
command=self.save)
        self.save.place(relx=0.505, rely=0.8)

#Implementing the save function which updates the user's details in
the program's database.
    def save(self):
        username = self.username
        if (len([*self.name.get()])<30) and
(int(self.height.get())>=100 and int(self.height.get())<=200) and
(int(self.age.get())>=12 and int(self.age.get())<=100) and
(int(self.weight.get())>=30 and int(self.weight.get())<=200):
            SQL.updateInfo(username, self.name.get(), self.age.get(),
self.gender.get(), self.height.get(), self.weight.get())
            messagebox.showinfo(title="Update success", message="Your
information has been updated")
            self.master.destroy()
            MainMenu.App()
        else:
            messagebox.showerror(title="Update failure",
message="Invalid inputs")
```

Diet_Menu.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Main_Menu as MainMenu
import FindDietMenu as FindDietMenu
import SavedDietPage as SavedDiet

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Diet menu")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
```

```
            MainMenu.App()
        except:
            pass


#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFE4E1", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_buttons()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Diet plans",
bg="#98FB98", fg="#000000", font=("Arial", 40),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.50, rely=.15, anchor=CENTER)

        self.diet_photo=Image.open("C:/Users/iamar/OneDrive/Pictures/A
LEVELS/NEA/DIET PHOTO NEA.png")
        self.diet_photo=self.diet_photo.resize((120,120))
        self.diet_photo=ImageTk.PhotoImage(self.diet_photo)
        self.diet_photo_label=tk.Label(self,image=self.diet_photo)
        self.diet_photo_label.place(relx=.50, rely=.50,anchor=CENTER)

#Creating the buttons.
    def create_buttons(self):
        self.find_diet_button = tk.Button(self, text="Find diet
plans", bg="#CDC9C9", font=("Arial black", 18), height=3, width=12,
command=self.finddiet)
        self.find_diet_button.place(relx=.20, rely=.50,anchor=CENTER)
        self.view_dietplans_button = tk.Button(self, text="View
saved\ndiet plan", bg="#CDC9C9", font=("Arial black", 18), height=3,
width=12, command=self.saveddiet)
        self.view_dietplans_button.place(relx=.80,
rely=.50,anchor=CENTER)

#Opening the find diets menu.
    def finddiet(self):
        try:
            self.master.destroy()
            FindDietMenu.App()
        except:
            pass
```

```
#Opening the saved diet menu.
    def saveddiet(self):
        try:
            self.master.destroy()
            SavedDiet.App()
        except:
            pass
```

Workout_Menu.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Main_Menu as MainMenu
import FindWorkoutMenu as FindWorkoutMenu
import SavedWorkoutPage as SavedWorkoutPage

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Workout menu")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
```

```
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            MainMenu.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#98F5FF", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_buttons()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Workout plans",
bg="#98FB98", fg="#000000", font=("Arial", 40),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.50, rely=.15, anchor=CENTER)


self.workout_photo=Image.open("C:/Users/iamar/OneDrive/Pictures/A
LEVELS/NEA/WORKOUT PHOTO NEA.png")
        self.workout_photo=self.workout_photo.resize((120,100))
        self.workout_photo=ImageTk.PhotoImage(self.workout_photo)

self.workout_photo_label=tk.Label(self,image=self.workout_photo)
        self.workout_photo_label.place(relx=.50,
rely=.50,anchor=CENTER)

#Creating the buttons.
    def create_buttons(self):
        self.find_workout_button = tk.Button(self, text="Find
workout\nplans", bg="#CDC9C9", font=("Arial black", 18), height=3,
width=12, command=self.findworkout)
        self.find_workout_button.place(relx=.20,
rely=.50,anchor=CENTER)
        self.view_workoutplans_button = tk.Button(self, text="View
saved\nworkout plan", bg="#CDC9C9", font=("Arial black", 18),
```

```
height=3, width=12, command=self.savedWorkout)
        self.view_workoutplans_button.place(relx=.80,
rely=.50,anchor=CENTER)

#Opening the find workout menu.
    def findworkout(self):
        try:
            self.master.destroy()
            FindWorkoutMenu.App()
        except:
            pass

#Opening the saved workout page.
    def savedWorkout(self):
        try:
            self.master.destroy()
            SavedWorkoutPage.App()
        except:
            pass
```

YourProgress_Menu.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Main_Menu as MainMenu
import UpdateProgress_Menu as UpdateProgress
import Analytics as Analytics

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Your progress menu")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
```

```
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            MainMenu.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFFACD", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_buttons()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Your progress",
bg="#98FB98", fg="#000000", font=("Arial", 40),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.50, rely=.15, anchor=CENTER)


self.progress_photo=Image.open("C:/Users/iamar/OneDrive/Pictures/A
LEVELS/NEA/PROGRESS PHOTO NEA.png")
        self.progress_photo=self.progress_photo.resize((140,120))
        self.progress_photo=ImageTk.PhotoImage(self.progress_photo)

self.progress_photo_label=tk.Label(self,image=self.progress_photo)
```

```
        self.progress_photo_label.place(relx=.50,
rely=.50,anchor=CENTER)

#Creating the buttons.
    def create_buttons(self):
        self.update_progress_button = tk.Button(self, text="Update
your\nprogress", bg="#CDC9C9", font=("Arial black", 18), height=3,
width=12, command=self.updateProgress)
        self.update_progress_button.place(relx=.20,
rely=.50,anchor=CENTER)
        self.view_analytics_button = tk.Button(self, text="View
analytics", bg="#CDC9C9", font=("Arial black", 18), height=3,
width=12, command=self.viewAnalytics)
        self.view_analytics_button.place(relx=.80,
rely=.50,anchor=CENTER)

#Opening the update progress page.
    def updateProgress(self):
        try:
            self.master.destroy()
            UpdateProgress.App()
        except:
            pass

#Opening the analytics page.
    def viewAnalytics(self):
        try:
            self.master.destroy()
            Analytics.App()
        except:
            pass
```

FindDietMenu.py

```
import tkinter as tk
from tkinter import *
import Diet_Menu as DietMenu
import GlobalVariables
from tkinter import messagebox
import ShowDietsPage as ShowDiets

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Find diets menu")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")
```

```
        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)
#Creating the back buttons.
    def back(self):
        try:
            self.master.destroy()
            DietMenu.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFE4E1", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_dropList()
        self.create_entries()
        self.create_button()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Your requirements",
```

```
bg="#98FB98", fg="#000000", font=("Arial", 25),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.59, rely=.1, anchor=CENTER)

        self.goal_label=tk.Label(self, text="Select goal",
bg="#FFE4E1")
        self.goal_label.place(relx=.17, rely=0.225)

        self.height_label=tk.Label(self, text="Height(cm)",
bg="#FFE4E1")
        self.height_label.place(relx=0.19,rely=0.345)

        self.weight_label=tk.Label(self, text="Weight(kg)",
bg="#FFE4E1")
        self.weight_label.place(relx=0.19,rely=0.453)

        self.age_label=tk.Label(self, text="Age", bg="#FFE4E1")
        self.age_label.place(relx=0.2,rely=0.56)

        self.preference_label=tk.Label(self,
text="Dietary\npreference", bg="#FFE4E1")
        self.preference_label.place(relx=0.17, rely=0.66)

#Creating the drop-down list.
    def create_dropList(self):
        self.goal_options=["Cut","Bulk","Body recomposition"]
        self.clicked_goal=StringVar()
        self.clicked_goal.set("Select a goal")
        self.goal_drop=OptionMenu(self, self.clicked_goal,
*self.goal_options)
        self.goal_drop.config(width=50)
        self.goal_drop.place(relx=.6, rely=0.25, anchor=CENTER)


self.preference_options=["Vegetarian","Non-vegetarian","Vegan"]
        self.clicked_preference=StringVar()
        self.clicked_preference.set("Select a goal")
        self.preference_drop=OptionMenu(self, self.clicked_preference,
*self.preference_options)
        self.preference_drop.config(width=50)
        self.preference_drop.place(relx=.6, rely=0.7, anchor=CENTER)

#Creating the entries.
    def create_entries(self):
        self.height=tk.Entry(self, font=("Calibri",18))
        self.height.config(width=28)
        self.height.place(relx=.328, rely=0.335)

        self.weight=tk.Entry(self, font=("Calibri",18))
        self.weight.config(width=28)
        self.weight.place(relx=.328, rely=0.445)
```

```
        self.age=tk.Entry(self, font=("Calibri",18))
        self.age.config(width=28)
        self.age.place(relx=.328, rely=0.555)

#Creating the submit button.
    def create_button(self):
        self.submit=tk.Button(self, text="Submit", font=("Calibri",
12), command=self.info)
        self.submit.place(relx=0.5, rely=0.8)

#Implementing the info function to set diet levels based on the user's
inputs.
    def info(self):
        try:
            if self.clicked_goal.get() == "Cut":
                if self.clicked_preference.get() == "Non-vegetarian":
                    if int(self.height.get())>=170 and
int(self.weight.get())>=70 and int(self.age.get())>=15:
                        GlobalVariables.dietLevel = 18
                    else:
                        GlobalVariables.dietLevel = 17
                elif self.clicked_preference.get() == "Vegetarian":
                    if int(self.height.get())>=170 and
int(self.weight.get())>=70 and int(self.age.get())>=15:
                        GlobalVariables.dietLevel = 16
                    else:
                        GlobalVariables.dietLevel = 15
                elif self.clicked_preference.get() == "Vegan":
                    if int(self.height.get())>=170 and
int(self.weight.get())>=70 and int(self.age.get())>=15:
                        GlobalVariables.dietLevel = 14
                    else:
                        GlobalVariables.dietLevel = 13

            elif self.clicked_goal.get() == "Bulk":
                if self.clicked_preference.get() == "Non-vegetarian":
                    if int(self.height.get())>=170 and
int(self.weight.get())>=65 and int(self.age.get())>=15:
                        GlobalVariables.dietLevel = 12
                    else:
                        GlobalVariables.dietLevel = 12
                elif self.clicked_preference.get() == "Vegetarian":
                    if int(self.height.get())>=170 and
int(self.weight.get())>=70 and int(self.age.get())>=15:
                        GlobalVariables.dietLevel = 10
                    else:
                        GlobalVariables.dietLevel = 9
                elif self.clicked_preference.get() == "Vegan":
                    if int(self.height.get())>=170 and
int(self.weight.get())>=70 and int(self.age.get())>=15:
                        GlobalVariables.dietLevel = 8
                    else:
```

```
                        GlobalVariables.dietLevel = 7

            elif self.clicked_goal.get() == "Body recomposition":
                if self.clicked_preference.get() == "Non-vegetarian":
                    if int(self.height.get())>=170 and
int(self.weight.get())>=70 and int(self.age.get())>=15:
                        GlobalVariables.dietLevel = 6
                    else:
                        GlobalVariables.dietLevel = 5
                elif self.clicked_preference.get() == "Vegetarian":
                    if int(self.height.get())>=170 and
int(self.weight.get())>=70 and int(self.age.get())>=15:
                        GlobalVariables.dietLevel = 4
                    else:
                        GlobalVariables.dietLevel = 3
                elif self.clicked_preference.get() == "Vegan":
                    if int(self.height.get())>=170 and
int(self.weight.get())>=70 and int(self.age.get())>=15:
                        GlobalVariables.dietLevel = 2
                    else:
                        GlobalVariables.dietLevel = 1
            self.master.destroy()
            ShowDiets.App()
        except:
            messagebox.showerror(title="Invalid",message="Invalid
inputs")
```

ShowDietsPage.py

```
import tkinter as tk
from tkinter import *
from tkinter import messagebox
import FindDietMenu as FindDietMenu
import GlobalVariables
import mySQL as SQL
import SavedDietPage as SavedDietPage

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Diets page")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)
```

```
        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            FindDietMenu.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFE4E1", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_buttons()

#Creating the labels.
    def create_labels(self):

        if GlobalVariables.dietLevel == 18:
            GlobalVariables.diet1Text = SQL.getDietPlan(1)
            GlobalVariables.diet1ID = 1
            GlobalVariables.diet2Text = SQL.getDietPlan(2)
            GlobalVariables.diet2ID = 2
```

```
        elif GlobalVariables.dietLevel == 17:
            GlobalVariables.diet1Text = SQL.getDietPlan(1)
            GlobalVariables.diet1ID = 1
            GlobalVariables.diet2Text = SQL.getDietPlan(2)
            GlobalVariables.diet2ID = 2
        elif GlobalVariables.dietLevel == 12:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 16:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 15:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 14:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 13:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 11:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 10:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 9:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 8:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 7:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
```

```
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 6:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 5:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 4:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 3:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 2:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4
        elif GlobalVariables.dietLevel == 1:
            GlobalVariables.diet1Text = SQL.getDietPlan(3)
            GlobalVariables.diet1ID = 3
            GlobalVariables.diet2Text = SQL.getDietPlan(4)
            GlobalVariables.diet2ID = 4


        self.opt1_label = tk.Label(self, text = "Option 1",
bg="#98FB98", fg="#000000", font=("Arial", 35),
highlightbackground="black", highlightthickness=1)
        self.opt1_label.place(relx=.275, rely=.100, anchor=CENTER)

        self.opt2_label = tk.Label(self, text = "Option 2",
bg="#98FB98", fg="#000000", font=("Arial", 35),
highlightbackground="black", highlightthickness=1)
        self.opt2_label.place(relx=.715, rely=.100, anchor=CENTER)

        self.diet1_label = tk.Label(self,
text=GlobalVariables.diet1Text, bg="#FFFFFF", font=("Arial black", 6),
height=20, width=45,highlightbackground="black", highlightthickness=1)
        self.diet1_label.place(relx=.260, rely=.540,anchor=CENTER)
        self.diet2_label = tk.Label(self,
text=GlobalVariables.diet2Text, bg="#FFFFFF", font=("Arial black", 6),
height=20, width=45,highlightbackground="black", highlightthickness=1)
```

```
        self.diet2_label.place(relx=.730, rely=.540,anchor=CENTER)

#Creating the buttons.
    def create_buttons(self):
        self.save1_button = tk.Button(self, text = "Save",
bg="#C0C0C0",font=("Arial black", 10), height=1,
width=5,highlightbackground="black",
highlightthickness=1,command=self.saveDiet1)
        self.save1_button.place(relx=.260, rely= .935, anchor=CENTER)

        self.save2_button = tk.Button(self, text = "Save",
bg="#C0C0C0",font=("Arial black", 10), height=1,
width=5,highlightbackground="black",
highlightthickness=1,command=self.saveDiet2)
        self.save2_button.place(relx=.730, rely= .935, anchor=CENTER)

#Saving the diet presented as Option 1.
    def saveDiet1(self):

SQL.addDietID(GlobalVariables.username,GlobalVariables.diet1ID)
        messagebox.showinfo(title="Diet Saved", message="Your diet has
been saved\nTo save a different diet\nmake sure to delete the saved
diet first.")
        self.master.destroy()
        SavedDietPage.App()

#Saving the diet presented as Option 2.
    def saveDiet2(self):

SQL.addDietID(GlobalVariables.username,GlobalVariables.diet2ID)
        messagebox.showinfo(title="Diet Saved", message="Your diet has
been saved\nTo save a different diet\nmake sure to delete the saved
diet first.")
        self.master.destroy()
        SavedDietPage.App()
```

SavedDietPage.py

```
import tkinter as tk
from tkinter import *
import Diet_Menu as DietMenu
import GlobalVariables
import mySQL as SQL

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Saved diet")
```

```
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            DietMenu.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFE4E1", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_buttons()

#Creating the labels.
```

```
    def create_labels(self):

        GlobalVariables.SavedDiet =
SQL.getSavedDietPlan(GlobalVariables.username)

        self.main_label = tk.Label(self, text = "Saved diet plan",
bg="#98FB98", fg="#000000", font=("Arial", 30),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.5, rely=.100, anchor=CENTER)

        self.diet_label = tk.Label(self,
text=GlobalVariables.SavedDiet, bg="#FFFFFF", font=("Arial black", 6),
height=20, width=45,highlightbackground="black", highlightthickness=1)
        self.diet_label.place(relx=.5, rely=.500,anchor=CENTER)

#Creating the delete button.
    def create_buttons(self):
        self.delete_button = tk.Button(self, text = "Delete",
bg="#FF0000",font=("Arial black", 10), height=1,
width=5,highlightbackground="black", highlightthickness=1,
command=self.deleteDiet)
        self.delete_button.place(relx=.5, rely= .825, anchor=CENTER)

#Deleting the saved diet.
    def deleteDiet(self):
        SQL.deleteSavedDiet(GlobalVariables.username)
        self.diet_label.destroy()
```

FindWorkoutMenu.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Workout_Menu as WorkoutMenu
import GlobalVariables
import ShowWorkoutsPage as ShowWorkoutsPage

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Find workouts menu")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
```

```
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            WorkoutMenu.App()
        except:
             pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#98F5FF", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_dropList()
        self.create_entries()
        self.create_button()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Your requirements",
bg="#98FB98", fg="#000000", font=("Arial", 25),
```

```
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.59, rely=.1, anchor=CENTER)

        self.goal_label=tk.Label(self, text="Select goal",
bg="#98F5FF")
        self.goal_label.place(relx=.17, rely=0.225)

        self.height_label=tk.Label(self, text="Height(cm)",
bg="#98F5FF")
        self.height_label.place(relx=0.19,rely=0.345)

        self.weight_label=tk.Label(self, text="Weight(kg)",
bg="#98F5FF")
        self.weight_label.place(relx=0.19,rely=0.453)

        self.age_label=tk.Label(self, text="Age", bg="#98F5FF")
        self.age_label.place(relx=0.2,rely=0.56)

        self.preference_label=tk.Label(self, text="Activity\nlevel",
bg="#98F5FF")
        self.preference_label.place(relx=0.19, rely=0.66)

#Creating the drop down lists.
    def create_dropList(self):
        self.goal_options=["Cut","Bulk","Body recomposition"]
        self.clicked_goal=StringVar()
        self.clicked_goal.set("Select a goal")
        self.goal_drop=OptionMenu(self, self.clicked_goal,
*self.goal_options)
        self.goal_drop.config(width=50)
        self.goal_drop.place(relx=.6, rely=0.25, anchor=CENTER)

        self.preference_options=["2-3 days a week","3-4 days a
week","5-6 days a week"]
        self.clicked_preference=StringVar()
        self.clicked_preference.set("Select a goal")
        self.activity_drop=OptionMenu(self, self.clicked_preference,
*self.preference_options)
        self.activity_drop.config(width=50)
        self.activity_drop.place(relx=.6, rely=0.7, anchor=CENTER)

#Creating the entries.
    def create_entries(self):
        self.height=tk.Entry(self, font=("Calibri",18))
        self.height.config(width=28)
        self.height.place(relx=.328, rely=0.335)

        self.weight=tk.Entry(self, font=("Calibri",18))
        self.weight.config(width=28)
        self.weight.place(relx=.328, rely=0.445)

        self.age=tk.Entry(self, font=("Calibri",18))
```

```
        self.age.config(width=28)
        self.age.place(relx=.328, rely=0.555)

#Creating the submit button.
    def create_button(self):
        submit=tk.Button(self, text="Submit", font=("Calibri", 12),
command=self.info)
        submit.place(relx=0.5, rely=0.8)

#Implementing the info function which sets workout level based on the
user's input.
    def info(self):
        try:
            if int(self.age.get())>15 and int(self.age.get())<60:
                if self.clicked_goal.get() == "Cut":
                    if self.clicked_preference.get() == "2-3 days a
week":

                            GlobalVariables.workoutLevel = 18

                    elif self.clicked_preference.get() == "3-4 days a
week":

                            GlobalVariables.workoutLevel = 16

                    elif self.clicked_preference.get() == "5-6 days a
week":

                            GlobalVariables.workoutLevel = 14

                elif self.clicked_goal.get() == "Bulk":
                    if self.clicked_preference.get() == "2-3 days a
week":

                            GlobalVariables.workoutLevel = 12

                    elif self.clicked_preference.get() == "3-4 days a
week":

                            GlobalVariables.workoutLevel = 10

                    elif self.clicked_preference.get() == "5-6 days a
week":

                            GlobalVariables.workoutLevel = 8

                elif self.clicked_goal.get() == "Body recomposition":
                    if self.clicked_preference.get() == "2-3 days a
week":

                            GlobalVariables.workoutLevel = 6
```

```
                    elif self.clicked_preference.get() == "3-4 days a
week":

                            GlobalVariables.workoutLevel = 4

                    elif self.clicked_preference.get() == "5-6 days a
week":

                            GlobalVariables.workoutLevel = 2
            elif int(self.age.get())<15 and int(self.age.get())>60:
                GlobalVariables.workoutLevel = 20
                messagebox.showwarning(title="Notice", message="As you
fall into the sensitive age group of our guidelines,\nwe advise you to
only follow our plan for a maximum of 3 days.")

            self.master.destroy()
            ShowWorkoutsPage.App()

        except:
             messagebox.showerror(title="Error",message="Invalid
inputs")
```

ShowWorkoutsPage.py

```
import tkinter as tk
from tkinter import *
from tkinter import messagebox
import FindWorkoutMenu as FindWorkoutMenu
import GlobalVariables
import mySQL as SQL
import SavedWorkoutPage as SavedWorkoutPage

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Workouts page")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
```

```
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            FindWorkoutMenu.App()
        except:
            pass


#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#98F5FF", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_buttons()

#Creating the labels.
    def create_labels(self):

        if GlobalVariables.workoutLevel == 20:
            GlobalVariables.workout1Text = SQL.getWorkoutPlan(5)
            GlobalVariables.workout1ID = 5
            GlobalVariables.workout2Text = SQL.getWorkoutPlan(6)
            GlobalVariables.workout2ID = 6
        elif GlobalVariables.workoutLevel == 18:
            GlobalVariables.workout1Text = SQL.getWorkoutPlan(3)
            GlobalVariables.workout1ID = 3
            GlobalVariables.workout2Text = SQL.getWorkoutPlan(4)
```

```
            GlobalVariables.workout2ID = 4
        elif GlobalVariables.workoutLevel == 16:
            GlobalVariables.workout1Text = SQL.getWorkoutPlan(1)
            GlobalVariables.workout1ID = 1
            GlobalVariables.workout2Text = SQL.getWorkoutPlan(7)
            GlobalVariables.workout2ID = 7
        elif GlobalVariables.workoutLevel == 14:
            GlobalVariables.workout1Text = SQL.getWorkoutPlan(2)
            GlobalVariables.workout1ID = 2
            GlobalVariables.workout2Text = SQL.getWorkoutPlan(8)
            GlobalVariables.workout2ID = 8
        elif GlobalVariables.workoutLevel == 12:
            GlobalVariables.workout1Text = SQL.getWorkoutPlan(2)
            GlobalVariables.workout1ID = 2
            GlobalVariables.workout2Text = SQL.getWorkoutPlan(8)
            GlobalVariables.workout2ID = 8
        elif GlobalVariables.workoutLevel == 10:
            GlobalVariables.workout1Text = SQL.getWorkoutPlan(2)
            GlobalVariables.workout1ID = 2
            GlobalVariables.workout2Text = SQL.getWorkoutPlan(8)
            GlobalVariables.workout2ID = 8
        elif GlobalVariables.workoutLevel == 8:
            GlobalVariables.workout1Text = SQL.getWorkoutPlan(2)
            GlobalVariables.workout1ID = 2
            GlobalVariables.workout2Text = SQL.getWorkoutPlan(8)
            GlobalVariables.workout2ID = 8
        elif GlobalVariables.workoutLevel == 6:
            GlobalVariables.workout1Text = SQL.getWorkoutPlan(2)
            GlobalVariables.workout1ID = 2
            GlobalVariables.workout2Text = SQL.getWorkoutPlan(8)
            GlobalVariables.workout2ID = 8
        elif GlobalVariables.workoutLevel == 4:
            GlobalVariables.workout1Text = SQL.getWorkoutPlan(2)
            GlobalVariables.workout1ID = 2
            GlobalVariables.workout2Text = SQL.getWorkoutPlan(8)
            GlobalVariables.workout2ID = 8
        elif GlobalVariables.workoutLevel == 2:
            GlobalVariables.workout1Text = SQL.getWorkoutPlan(2)
            GlobalVariables.workout1ID = 2
            GlobalVariables.workout2Text = SQL.getWorkoutPlan(8)
            GlobalVariables.workout2ID = 8

        self.opt1_label = tk.Label(self, text = "Option 1",
bg="#98FB98", fg="#000000", font=("Arial", 35),
highlightbackground="black", highlightthickness=1)
        self.opt1_label.place(relx=.275, rely=.100, anchor=CENTER)

        self.opt2_label = tk.Label(self, text = "Option 2",
bg="#98FB98", fg="#000000", font=("Arial", 35),
highlightbackground="black", highlightthickness=1)
        self.opt2_label.place(relx=.715, rely=.100, anchor=CENTER)
```

```
        self.workout1_label = tk.Label(self,
text=GlobalVariables.workout1Text, bg="#FFFFFF", font=("Arial black",
6), height=30, width=47,highlightbackground="black",
highlightthickness=1)
        self.workout1_label.place(relx=.260, rely=.540,anchor=CENTER)
        self.workout2_label = tk.Label(self,
text=GlobalVariables.workout2Text, bg="#FFFFFF", font=("Arial black",
6), height=30, width=47,highlightbackground="black",
highlightthickness=1)
        self.workout2_label.place(relx=.730, rely=.540,anchor=CENTER)

#Creating the buttons.
    def create_buttons(self):
        self.save1_button = tk.Button(self, text = "Save",
bg="#C0C0C0",font=("Arial black", 10), height=1,
width=5,highlightbackground="black",
highlightthickness=1,command=self.saveworkout1)
        self.save1_button.place(relx=.260, rely= .935, anchor=CENTER)

        self.save2_button = tk.Button(self, text = "Save",
bg="#C0C0C0",font=("Arial black", 10), height=1,
width=5,highlightbackground="black",
highlightthickness=1,command=self.saveworkout2)
        self.save2_button.place(relx=.730, rely= .935, anchor=CENTER)

#Saving the workout presented as Option 1.
    def saveworkout1(self):
        messagebox.showinfo(title="Workout Saved", message="Your
workout has been saved\nTo save a different workout\nmake sure to
delete the saved workout first.")

SQL.addWorkoutID(GlobalVariables.username,GlobalVariables.workout1ID)
        self.master.destroy()
        SavedWorkoutPage.App()

#Saving the workout presented as Option 2.
    def saveworkout2(self):
        messagebox.showinfo(title="Workout Saved", message="Your
workout has been saved\nTo save a different workout\nmake sure to
delete the saved workout first.")

SQL.addWorkoutID(GlobalVariables.username,GlobalVariables.workout2ID)
        self.master.destroy()
        SavedWorkoutPage.App()
```

SavedWorkoutPage.py

```
import tkinter as tk
from tkinter import *
import Workout_Menu as WorkoutMenu
import GlobalVariables
```

```
import mySQL as SQL


class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Saved workout")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            WorkoutMenu.App()
        except:
            pass


#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#98F5FF", highlightbackground="black",
```

```
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_buttons()

#Creating the labels.
    def create_labels(self):

        GlobalVariables.SavedWorkout =
SQL.getSavedWorkoutPlan(GlobalVariables.username)

        self.main_label = tk.Label(self, text = "Saved workout plan",
bg="#98FB98", fg="#000000", font=("Arial", 30),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.5, rely=.100, anchor=CENTER)

        self.workout_label = tk.Label(self,
text=GlobalVariables.SavedWorkout, bg="#FFFFFF", font=("Arial black",
6), height=30, width=50,highlightbackground="black",
highlightthickness=1)
        self.workout_label.place(relx=.5, rely=.500,anchor=CENTER)

#Creating the delete button.
    def create_buttons(self):
        self.delete_button = tk.Button(self, text = "Delete",
bg="#FF0000",font=("Arial black", 10), height=1,
width=5,highlightbackground="black", highlightthickness=1,
command=self.deleteDiet)
        self.delete_button.place(relx=.5, rely= .9, anchor=CENTER)

#Deleting the saved workout.
    def deleteDiet(self):
        SQL.deleteSavedWorkout(GlobalVariables.username)
        self.workout_label.destroy()
```

UpdateProgress_Menu.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Main_Menu as MainMenu
import YourProgress_Menu as YourProgress
import UpdateProgress as UpdateYourProgress
import Calculations_menu as Calculations

#Creating the window.
```

```
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Update progress menu")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            YourProgress.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFFACD", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
```

```
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_buttons()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Update
your\nprogress", bg="#98FB98", fg="#000000", font=("Arial", 40),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.50, rely=.15, anchor=CENTER)

#Creating the buttons.
    def create_buttons(self):
        self.update_progress_button = tk.Button(self, text="Upload
your\nprogress", bg="#CDC9C9", font=("Arial black", 18), height=3,
width=12, command=self.updateProgress)
        self.update_progress_button.place(relx=.25,
rely=.50,anchor=CENTER)
        self.calculations_button = tk.Button(self,
text="Calculations", bg="#CDC9C9", font=("Arial black", 18), height=3,
width=12, command=self.Calculations)
        self.calculations_button.place(relx=.75,
rely=.50,anchor=CENTER)

#Opening the update progress page.
    def updateProgress(self):
        try:
            self.master.destroy()
            UpdateYourProgress.App()
        except:
            pass

#Opening the calculations page.
    def Calculations(self):
        try:
            self.master.destroy()
            Calculations.App()
        except:
            pass
```

UpdateProgress.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import GlobalVariables
import UpdateProgress_Menu as UpdateProgress
```

```
import YourProgress_Menu as YourProgress
import mySQL as SQL

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Update progress")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options menu.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            UpdateProgress.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
```

```
        self.configure(bg="#FFFACD", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_dropList()
        self.create_entries()
        self.create_button()

#Creating the labels
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Upload your
progress", bg="#98FB98", fg="#000000", font=("Arial", 25),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.59, rely=.1, anchor=CENTER)

        self.BMI_label=tk.Label(self, text="BMI", bg="#FFFACD")
        self.BMI_label.place(relx=.20, rely=0.26)

        self.bodyfat_label=tk.Label(self, text="Body fat%",
bg="#FFFACD")
        self.bodyfat_label.place(relx=0.18,rely=0.38)

        self.weight_label=tk.Label(self, text="Weight", bg="#FFFACD")
        self.weight_label.place(relx=0.19,rely=0.51)

        self.follow_label=tk.Label(self, text="Did you follow
your\nsaved plans?", bg="#FFFACD")
        self.follow_label.place(relx=0.13, rely=0.62)

#Creating the drop-down list.
    def create_dropList(self):
        self.follow_options=["Yes","No"]
        self.follow=StringVar()
        self.follow.set("Select an option")
        self.follow_drop=OptionMenu(self, self.follow,
*self.follow_options)
        self.follow_drop.config(width=50)
        self.follow_drop.place(relx=.6, rely=0.66, anchor=CENTER)

#Creating the entries.
    def create_entries(self):
        self.BMI=tk.Entry(self, font=("Calibri",18))
        self.BMI.config(width=28)
        self.BMI.place(relx=.328, rely=0.25)

        self.bodyfat=tk.Entry(self, font=("Calibri",18))
        self.bodyfat.config(width=28)
        self.bodyfat.place(relx=.328, rely=0.37)
```

```
        self.weight=tk.Entry(self, font=("Calibri",18))
        self.weight.config(width=28)
        self.weight.place(relx=.328, rely=0.50)

#Creating the buttons.
    def create_button(self):
        self.save=tk.Button(self, text="Submit", font=("Calibri", 12),
command=self.Save)
        self.save.place(relx=0.4, rely=0.8)

        self.update=tk.Button(self, text="Update", font=("Calibri",
12), command=self.Update)
        self.update.place(relx=0.6, rely=0.8)

#Saving the user's progress in the system's database.
    def Save(self):
        try:
            if int(self.BMI.get())<=40 and
int(self.bodyfat.get())<=55:

SQL.addtoUserProfile(GlobalVariables.username,self.bodyfat.get(),self.
BMI.get())
                SQL.updateWeight(GlobalVariables.username,
self.weight.get())
                messagebox.showinfo(title="Saved", message="Your
progress has been saved.")
                self.master.destroy()
                YourProgress.App()
            else:
                messagebox.showerror(title="Error",message="Invalid
inputs")
        except:
            messagebox.showerror(title="Error",message="Invalid
inputs")

#Updating the user's progress in the system's database.
    def Update(self):
        try:
            if int(self.BMI.get())<=40 and
int(self.bodyfat.get())<=55:

SQL.UpdateUserProfile(GlobalVariables.username,self.bodyfat.get(),self
.BMI.get())

SQL.updateWeight(GlobalVariables.username,self.weight.get())
                messagebox.showinfo(title="Updated", message="Your
progress has been updated.")
                self.master.destroy()
                YourProgress.App()
            else:
                messagebox.showerror(title="Error",message="Invalid
inputs")
```

```
        except:
            messagebox.showerror(title="Error",message="Invalid
inputs")
```

Calculations_menu.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Main_Menu as MainMenu
import UpdateProgress_Menu as UpdateProgress
import Calculate_Bodyfat as Calculate_Bodyfat
import Calculate_BMI as Calculate_BMI


#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Calculations")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
```

```
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

    def back(self):
        try:
            self.master.destroy()
            UpdateProgress.App()
        except:
            pass


#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFFACD", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_buttons()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Calculations",
bg="#98FB98", fg="#000000", font=("Arial", 40),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.50, rely=.15, anchor=CENTER)


#Creating the buttons.
    def create_buttons(self):
        self.bodyfat_button = tk.Button(self, text="Body
fat\npercentage", bg="#CDC9C9", font=("Arial black", 18), height=3,
width=12, command=self.Bodyfat)
        self.bodyfat_button.place(relx=.25, rely=.50,anchor=CENTER)
        self.BMI_button = tk.Button(self, text="BMI Score",
bg="#CDC9C9", font=("Arial black", 18), height=3, width=12,
command=self.BMI)
        self.BMI_button.place(relx=.75, rely=.50,anchor=CENTER)

#Opening the bodyfat calculator.
    def Bodyfat(self):
        try:
            self.master.destroy()
            Calculate_Bodyfat.App()
        except:
```

```
            pass

#Opening the BMI calculator.
    def BMI(self):
        try:
            self.master.destroy()
            Calculate_BMI.App()
        except:
            pass
```

Calculate_BMI.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Calculations_menu as Calculations_menu
import Show_BMI as Show_BMI
import GlobalVariables

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("BMI")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
```

```
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

    #Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            Calculations_menu.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFFACD", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_entries()
        self.create_button()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "BMI calculator",
bg="#98FB98", fg="#000000", font=("Arial", 25),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.53, rely=.1, anchor=CENTER)

        self.height_label=tk.Label(self, text="Height(m)",
bg="#FFFACD")
        self.height_label.place(relx=0.18,rely=0.345)

        self.weight_label=tk.Label(self, text="Weight(kg)",
bg="#FFFACD")
        self.weight_label.place(relx=0.1775,rely=0.453)

        self.age_label=tk.Label(self, text="Age", bg="#FFFACD")
        self.age_label.place(relx=0.2,rely=0.56)


#Creating the entries.
    def create_entries(self):

        self.height=tk.Entry(self, font=("Calibri",18))
```

```
        self.height.config(width=28)
        self.height.place(relx=.328, rely=0.335)

        self.weight=tk.Entry(self, font=("Calibri",18))
        self.weight.config(width=28)
        self.weight.place(relx=.328, rely=0.445)

        self.age=tk.Entry(self, font=("Calibri",18))
        self.age.config(width=28)
        self.age.place(relx=.328, rely=0.555)

#Creating the buttons.
    def create_button(self):
        calculate=tk.Button(self, text="Calculate", font=("Calibri",
12), command=self.calculation)
        calculate.place(relx=0.5, rely=0.7)

#Implementing the BMI calculation.
    def calculation(self):
        try:
            self.Height=float(self.height.get())
            self.Age=int(self.age.get())
            self.Weight=float(self.weight.get())

            self.BMI = (self.Weight)/(self.Height**2)
            self.BMI = ("{:.1f}".format(self.BMI))

            GlobalVariables.BMIscore = self.BMI

            self.master.destroy()
            Show_BMI.App()
        except:
            messagebox.showerror(title="Error",message="Invalid
inputs")
```

Show_BMI.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Calculate_BMI as Calculate_BMI
import GlobalVariables

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
```

```
        super().__init__()
        self.title("BMI")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            Calculate_BMI.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFFACD", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_entries()
```

```
#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "BMI calculator",
bg="#98FB98", fg="#000000", font=("Arial", 25),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.53, rely=.1, anchor=CENTER)

        self.bodyfat_label=tk.Label(self, text="Result", bg="#FFFACD")
        self.bodyfat_label.place(relx=0.18,rely=0.2)


self.chart_photo=Image.open("C:/Users/iamar/OneDrive/Pictures/A
LEVELS/NEA/BMI NEA.png")
        self.chart_photo=self.chart_photo.resize((600,360))
        self.chart_photo=ImageTk.PhotoImage(self.chart_photo)
        self.chart_photo_label=tk.Label(self,image=self.chart_photo)
        self.chart_photo_label.place(relx=.5, rely=.63,anchor=CENTER)

#Creating the entries.
    def create_entries(self):

        self.bodyfat=tk.Entry(self, font=("Calibri",18))
        self.bodyfat.config(width=28)
        self.bodyfat.place(relx=.275, rely=0.195)
        self.bodyfat.insert(0,GlobalVariables.BMIscore)
```

Calculate_Bodyfat.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Calculations_menu as Calculations_menu
import Show_Bodyfat as Show_Bodyfat
import GlobalVariables

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Bodyfat")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
```

```
        self.main=Main(self)

        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)
    #Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            Calculations_menu.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFFACD", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_entries()
        self.create_button()
        self.create_dropList()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Body fat%
calculator", bg="#98FB98", fg="#000000", font=("Arial", 25),
highlightbackground="black", highlightthickness=1)
```

```
        self.main_label.place(relx=.53, rely=.1, anchor=CENTER)

        self.height_label=tk.Label(self, text="Height(m)",
bg="#FFFACD")
        self.height_label.place(relx=0.18,rely=0.345)

        self.weight_label=tk.Label(self, text="Weight(kg)",
bg="#FFFACD")
        self.weight_label.place(relx=0.1775,rely=0.453)

        self.age_label=tk.Label(self, text="Age", bg="#FFFACD")
        self.age_label.place(relx=0.2,rely=0.56)

        self.gender_label=tk.Label(self, text="Gender", bg="#FFFACD")
        self.gender_label.place(relx=0.19,rely=0.67)

#Creating the entries.
    def create_entries(self):

        self.height=tk.Entry(self, font=("Calibri",18))
        self.height.config(width=28)
        self.height.place(relx=.328, rely=0.335)

        self.weight=tk.Entry(self, font=("Calibri",18))
        self.weight.config(width=28)
        self.weight.place(relx=.328, rely=0.445)

        self.age=tk.Entry(self, font=("Calibri",18))
        self.age.config(width=28)
        self.age.place(relx=.328, rely=0.555)

#Creating the drop-down list.
    def create_dropList(self):

        self.gender_options=["Male","Female"]
        self.clicked_gender=StringVar()
        self.clicked_gender.set("Select gender")
        self.preference_drop=OptionMenu(self, self.clicked_gender,
*self.gender_options)
        self.preference_drop.config(width=50)
        self.preference_drop.place(relx=.6, rely=0.69, anchor=CENTER)

#Creating the calculate button.
    def create_button(self):
        calculate=tk.Button(self, text="Calculate", font=("Calibri",
12), command=self.calculation)
        calculate.place(relx=0.5, rely=0.775)


#Implementing the bodyfat calculations.
    def calculation(self):
        try:
```

```
            self.Height=float(self.height.get())
            self.Age=int(self.age.get())
            self.Weight=float(self.weight.get())

            self.BMI = (self.Weight)/(self.Height**2)
            self.BMI = ("{:.1f}".format(self.BMI))

            if self.clicked_gender.get() == "Male":
                self.Bodyfat =
("{:.1f}".format((float(1.20)*float(self.BMI))+(0.23*self.Age)-(16.2))
)
                self.Bodyfat = str(self.Bodyfat) + "%"

            elif self.clicked_gender.get() == "Female":
                self.Bodyfat =
("{:.1f}".format((float(1.20)*float(self.BMI))+(0.23*self.Age)-(5.4)))
                self.Bodyfat = str(self.Bodyfat) + "%"


            GlobalVariables.bodyfat = self.Bodyfat

            self.master.destroy()
            Show_Bodyfat.App()
        except:
            messagebox.showerror(title="Error",message="Invalid
inputs")
```

Show_Bodyfat.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import Calculate_Bodyfat as Calculate_Bodyfat
import GlobalVariables

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Bodyfat")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")

        #widgets
        self.options=Options(self)
        self.main=Main(self)
```

```
        #run
        self.mainloop()

#Creating the Options frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

#Creating the back button.
    def back(self):
        try:
            self.master.destroy()
            Calculate_Bodyfat.App()
        except:
            pass

#Creating the Main frame.
class Main(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFFACD", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_entries()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Body fat%
calculator", bg="#98FB98", fg="#000000", font=("Arial", 25),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.53, rely=.1, anchor=CENTER)
```

```
        self.bodyfat_label=tk.Label(self, text="Result", bg="#FFFACD")
        self.bodyfat_label.place(relx=0.18,rely=0.32)


self.chart_photo=Image.open("C:/Users/iamar/OneDrive/Pictures/A
LEVELS/NEA/BODYFAT NEA.png")
        self.chart_photo=self.chart_photo.resize((250,225))
        self.chart_photo=ImageTk.PhotoImage(self.chart_photo)
        self.chart_photo_label=tk.Label(self,image=self.chart_photo)
        self.chart_photo_label.place(relx=.55, rely=.65,anchor=CENTER)

#Creating the entries.
    def create_entries(self):

        self.bodyfat=tk.Entry(self, font=("Calibri",18))
        self.bodyfat.config(width=28)
        self.bodyfat.place(relx=.275, rely=0.315)
        self.bodyfat.insert(0,GlobalVariables.bodyfat)
```

HashingAlgorithm.py

```
def custom_hash(input_string):
    # Initialise the hash value to 0
    hash_value = 0

    # Iterate through each character in the input string
    for char in input_string:
        # Update the hash value using the ASCII value of the character
        hash_value = (hash_value * 31 + ord(char)) % (2**32)

    return hash_value
```

GlobalVariables.py

```
username = ""

diet1 = "Breakfast:\n4 boiled eggs(3 whites + 1 whole) - 140
calories\n1 slice of bread(multigrain) - 137 calories\nLunch:\n1
serving of Sheet-Pan Chicken Fajita Bowls\nwith 1/3 cooked brown rice
- 450 calories\n2 scoops of whey protein with 300 ml water and\n100 ml
of semi skimmed milk - 300 calories\nDinner:\n1 serving Veggie &
Hummus Sandwich - 350 calories\nPre-workout snacks:\n1 banana - 105
calories\n1 apple - 95 calories \nTotal macronutrients:\n1587
calories\n187 grams of protein\n145 grams of carbohydrates\n20 grams
of fats"
diet2 = "Breakfast:\n2 Poached eggs, salmon and avocado - 550
calories\nLunch:\nDouble chicken breast, broccoli and rice - 700
calories\nDinner:\nTinned tuna, quinoa, avocado and broccoli - 500
calories\nPre-workout snack:\n2 scoops of whey protein with 400 ml of
```

```
water\nTotal macronutrients:\n1970 calories\n227 grams of protein\n81
grams of carbohydrates\n53 grams of fats"
diet3 = "Breakfast:\n6 egg omelette with spinach - 564 calories\n1
serving of Avocados on toast - 330 calories\nLunch:\nSalmon, sweet
potatoes and sesame seeds - 700 calories\n2 scoops of whey protein
with\n300 ml of whole milk - 400 calories\nDinner:\nBurger with lean
beef, fries\nwhite bread roll, cup of green beans - 1450
calories\nPre-workout snack:\n1 Protein flapjack - 224 calories\nTotal
Macronutrients:\n3668 calories\n247 grams of protein\n201 grams of
carbohydrates\n96 grams of fats"
diet4 = "Breakfast:\nSalmon, sweet potatoes and sesame seeds - 700
calories\n2 scoops of whey protein with\n300 ml of whole milk - 400
calories\nLunch:\n250 grams chicken breast and mixed salad,\nwith 1
whole avocado - 850 calories\nDinner:\n 500 grams steak with\n220
grams baked potato - 1400 calories\n Pre-workout snack:\n 1 Protein
flapjack - 224 calories\nTotal Macronutrients:\n3574 calories\n261
grams of protein\n178 grams of carbohydrates\n121 grams of fats"

dietLevel=""

diet1Text=""
diet2Text=""

diet1ID=""

diet2ID=""

SavedDiet=""

#normal 3-4 days
workout1="Day 1: Upper body\nChest: flat barbell bench press – 4 sets
of 6-8 reps\nBack: bent-over barbell rows – 3 sets of 6-8
reps\nShoulders: seated dumbbell press – 3 sets of 8-10
reps\nChest/triceps: dips – 3 sets of 8-10 reps\nBack: pullups or lat
pulldowns – 3 sets of 8-10 reps\nBiceps: incline dumbbell curls – 3
sets of 10-12 reps\n\nDay 2: Lower body\nLegs: barbell back squats – 4
sets of 6-8 reps\nLegs: leg press – 3 sets of 8-10 reps\nQuadriceps:
seated leg extensions – 3 sets of 10-12 reps\nCalves: calf press on
leg press – 4 sets of 12-15 reps\nAbs: decline crunches – 4 sets of
12-15 reps\n\nRepeat this circuit two times"
#normal 5-6 days
workout2="Day 1: Legs, shoulders, and abs\nLegs: dumbbell squats – 3
sets of 6-8 reps\nShoulders: standing shoulder press – 3 sets of 6-8
reps\nLegs: dumbbell lunge – 2 sets of 8-10 reps per leg\nShoulders:
dumbbell upright rows – 2 sets of 8-10 reps\nHamstrings: Romanian
dumbbell deadlift – 2 sets of 6-8 reps\nShoulders: lateral raises – 3
sets of 8-10 reps\nCalves: seated calf raises – 4 sets of 10-12
reps\nAbs: crunches with legs elevated – 3 sets of 10-12 reps\n\nDay
2: Chest and back\nChest: dumbbell bench press or floor press – 3 sets
of 6-8 reps\nBack: dumbbell bent-over rows – 3 sets of 6-8
reps\nChest: dumbbell fly – 3 sets of 8-10 reps\nBack: one-arm
dumbbell rows – 3 sets of 6-8 reps\nChest: pushups – 3 sets of 10-12
```

```
reps\nBack/chest: dumbbell pullovers – 3 sets of 10-12 reps\n\nDay 3:
Arms and abs\nBiceps: alternating biceps curls – 3 sets of 8-10 reps
per arm\nTriceps: overhead triceps extensions – 3 sets of 8-10
reps\nBiceps: seated dumbbell curls – 2 sets of 10-12 reps per
arm\nTriceps: bench dips – 2 sets of 10-12 reps\nBiceps: concentration
curls – 3 sets of 10-12 reps\nTriceps: dumbbell kickbacks – 3 sets of
8-10 reps per arm\nAbs: planks – 3 sets of 30-second holds\n\nRepeat
this circuit once more."
#normal 2-3 days
workout3="Day 1: Legs, shoulders, and abs\nLegs: dumbbell squats – 3
sets of 6-8 reps\nShoulders: standing shoulder press – 3 sets of 6-8
reps\nLegs: dumbbell lunge – 2 sets of 8-10 reps per leg\nShoulders:
dumbbell upright rows – 2 sets of 8-10 reps\nHamstrings: Romanian
dumbbell deadlift – 2 sets of 6-8 reps\nShoulders: lateral raises – 3
sets of 8-10 reps\nCalves: seated calf raises – 4 sets of 10-12
reps\nAbs: crunches with legs elevated – 3 sets of 10-12 reps\n\nDay
2: Chest and back\nChest: dumbbell bench press or floor press – 3 sets
of 6-8 reps\nBack: dumbbell bent-over rows – 3 sets of 6-8
reps\nChest: dumbbell fly – 3 sets of 8-10 reps\nBack: one-arm
dumbbell rows – 3 sets of 6-8 reps\nChest: pushups – 3 sets of 10-12
reps\nBack/chest: dumbbell pullovers – 3 sets of 10-12 reps\n\nDay 3:
Arms and abs\nBiceps: alternating biceps curls – 3 sets of 8-10 reps
per arm\nTriceps: overhead triceps extensions – 3 sets of 8-10
reps\nBiceps: seated dumbbell curls – 2 sets of 10-12 reps per
arm\nTriceps: bench dips – 2 sets of 10-12 reps\nBiceps: concentration
curls – 3 sets of 10-12 reps\nTriceps: dumbbell kickbacks – 3 sets of
8-10 reps per arm\nAbs: planks – 3 sets of 30-second holds"
#another 2-3 days
workout4="Day 1: Legs, shoulders\nLegs: dumbbell squats – 3 sets of
6-8 reps\nShoulders: standing shoulder press – 3 sets of 6-8
reps\nLegs: dumbbell lunge – 2 sets of 8-10 reps per leg\nShoulders:
dumbbell upright rows – 2 sets of 8-10 reps\nHamstrings: Romanian
dumbbell deadlift – 2 sets of 6-8 reps\nShoulders: lateral raises – 3
sets of 8-10 reps\nCalves: seated calf raises – 4 sets of 10-12
reps\n\nDay 2: Chest and back\nChest: dumbbell bench press or floor
press – 3 sets of 6-8 reps\nBack: dumbbell bent-over rows – 3 sets of
6-8 reps\nChest: dumbbell fly – 3 sets of 8-10 reps\nBack: one-arm
dumbbell rows – 3 sets of 6-8 reps\nChest: pushups – 3 sets of 10-12
reps\nBack/chest: dumbbell pullovers – 3 sets of 10-12 reps\n\nDay 3:
Arms and abs\nBiceps: alternating biceps curls – 3 sets of 8-10 reps
per arm\nTriceps: overhead triceps extensions – 3 sets of 8-10
reps\nBiceps: seated dumbbell curls – 2 sets of 10-12 reps per
arm\nTriceps: bench dips – 2 sets of 10-12 reps\nBiceps: concentration
curls – 3 sets of 10-12 reps\nTriceps: dumbbell kickbacks – 3 sets of
8-10 reps per arm\nAbs: planks – 3 sets of 30-second holds"
#kids workout
workout5="Day 1: Full body\nLegs: barbell back squats – 3 sets of 5
reps\nChest: flat barbell bench press – 3 set of 5 reps\nBack: seated
cable rows – 3 sets of 6-8 reps\nShoulders: seated dumbbell shoulder
press – 3 sets of 6-8 reps\nTriceps: cable rope triceps pushdowns – 3
sets of 8-10 reps\nShoulders: lateral raises – 3 sets of 10-12
reps\nCalves: seated calf raises – 3 sets of 10-12 reps\nAbs: planks –
```

```
3 sets of 30-second holds\n\nDay 2: Full body\nBack/hamstrings:
barbell or trap bar deadlifts – 3 sets of 5 reps\nBack: pullups or lat
pulldowns – 3 sets of 6-8 reps\nChest: barbell or dumbbell incline
press – 3 sets of 6-8 reps\nShoulders: machine shoulder press – 3 sets
of 6-8 reps\nBiceps: barbell or dumbbell biceps curls – 3 sets of 8-10
reps\nShoulders: reverse machine fly – 3 sets of 10-12 reps\nCalves:
standing calf raises – 3 sets of 10-12 reps\n\nDay 3: Full body\nLegs:
leg press – 3 sets of 5 reps\nBack: T-bar rows – 3 sets of 6-8
reps\nChest: machine or dumbbell chest fly – 3 sets of 6-8
reps\nShoulders: one-arm dumbbell shoulder press – 3 sets of 6-8
reps\nTriceps: dumbbell or machine triceps extensions – 3 sets of 8-10
reps\nShoulders: cable or dumbbell front raises – 3 sets of 10-12
reps\nCalves: seated calf raises – 3 sets of 10-12 reps\nAbs: decline
crunches – 3 sets of 10-12 reps"
#another kids
workout6="Day 1: Full body\nLegs: barbell back squats – 3 sets of 5
reps\nChest: flat barbell bench press – 3 set of 5 reps\nBack: seated
cable rows – 3 sets of 6-8 reps\nShoulders: seated dumbbell shoulder
press – 3 sets of 6-8 reps\nTriceps: cable rope triceps pushdowns – 3
sets of 8-10 reps\nShoulders: lateral raises – 3 sets of 10-12
reps\nCalves: seated calf raises – 3 sets of 10-12 reps\nAbs: planks –
3 sets of 30-second holds\n\nDay 2: Full body\nBack/hamstrings:
barbell or trap bar deadlifts – 3 sets of 5 reps\nBack: pullups or lat
pulldowns – 3 sets of 6-8 reps\nChest: barbell or dumbbell incline
press – 3 sets of 6-8 reps\nShoulders: machine shoulder press – 3 sets
of 6-8 reps\nBiceps: barbell or dumbbell biceps curls – 3 sets of 8-10
reps\nShoulders: reverse machine fly – 3 sets of 10-12 reps\nCalves:
standing calf raises – 3 sets of 10-12 reps\n\nDay 3: 2 hours of
playing any sport you like"
#another 3-4 days
workout7="Day 1: Legs, shoulders, and abs\nLegs: dumbbell squats – 3
sets of 6-8 reps\nShoulders: standing shoulder press – 3 sets of 6-8
reps\nLegs: dumbbell lunge – 2 sets of 8-10 reps per leg\nShoulders:
dumbbell upright rows – 2 sets of 8-10 reps\nHamstrings: Romanian
dumbbell deadlift – 2 sets of 6-8 reps\nShoulders: lateral raises – 3
sets of 8-10 reps\nCalves: seated calf raises – 4 sets of 10-12
reps\nAbs: crunches with legs elevated – 3 sets of 10-12 reps\n\nDay
2: Chest and back\nChest: dumbbell bench press or floor press – 3 sets
of 6-8 reps\nBack: dumbbell bent-over rows – 3 sets of 6-8
reps\nChest: dumbbell fly – 3 sets of 8-10 reps\nBack: one-arm
dumbbell rows – 3 sets of 6-8 reps\nChest: pushups – 3 sets of 10-12
reps\nBack/chest: dumbbell pullovers – 3 sets of 10-12 reps\n\nDay 3:
Arms and abs\nBiceps: alternating biceps curls – 3 sets of 8-10 reps
per arm\nTriceps: overhead triceps extensions – 3 sets of 8-10
reps\nBiceps: seated dumbbell curls – 2 sets of 10-12 reps per
arm\nTriceps: bench dips – 2 sets of 10-12 reps\nBiceps: concentration
curls – 3 sets of 10-12 reps\nTriceps: dumbbell kickbacks – 3 sets of
8-10 reps per arm\nAbs: planks – 3 sets of 30-second holds\n\nDay 4:
Play any sport for 2 hours."
#another 5-6 days
workout8="Day 1: Upper body\nChest: flat barbell bench press – 4 sets
of 6-8 reps\nBack: bent-over barbell rows – 3 sets of 6-8
```

```
reps\nShoulders: seated dumbbell press – 3 sets of 8-10
reps\nChest/triceps: dips – 3 sets of 8-10 reps\nBack: pullups or lat
pulldowns – 3 sets of 8-10 reps\nBiceps: incline dumbbell curls – 3
sets of 10-12 reps\n\nDay 2: Lower body\nLegs: barbell back squats – 4
sets of 6-8 reps\nLegs: leg press – 3 sets of 8-10 reps\nQuadriceps:
seated leg extensions – 3 sets of 10-12 reps\nCalves: calf press on
leg press – 4 sets of 12-15 reps\nAbs: decline crunches – 4 sets of
12-15 reps\n\nRepeat this circuit three times"

workoutLevel=""

workout1Text=""
workout2Text=""

workout1ID=""
workout2ID=""

SavedWorkout=""

UKAvgBMI = 27.2

UKAvgBodyfat_Male = 23.2
UKAvgBodyfat_Female = 33.7

bodyfat=""
BMIscore=""
```

Analytics.py

```
import tkinter as tk
from tkinter import PhotoImage
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk
import YourProgress_Menu as YourProgress_Menu
import GlobalVariables
import mySQL as SQL
import matplotlib.pyplot as plt
from abc import ABC, abstractmethod

#Creating the window.
class App(tk.Tk):
    def __init__(self):

        #main setup
        super().__init__()
        self.title("Analytics")
        self.geometry("800x500")
        self.configure(bg="#B9D3EE")
```

```
        #widgets
        self.options=Options(self)
        self.main=Main(self)

        #run
        self.mainloop()

#Creating Options Frame.
class Options(tk.Frame):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#CDC9C9", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.pack_propagate(False)
        self.configure(width=170, height=500)

        self.create_widgets()

    def create_widgets(self):
        #creating widgets
        self.back_button = tk.Button(self, bg="#8B8989", text="Back",
font=("Arial Black", 14), command=self.back)

        #placing widgets
        self.back_button.place(x=45, y=20)

    #Back button's function.
    def back(self):
        try:
            self.master.destroy()
            YourProgress_Menu.App()
        except:
            pass


#Creating Abstract class - this gives a structure for the main frame.
class MainStructure(ABC):
    def __init__(self):
        pass

    @abstractmethod
    def create_labels():
        pass

    @abstractmethod
    def create_entries():
        pass

    @abstractmethod
    def create_buttons():
        pass
```

```
    @abstractmethod
    def Graph1():
        pass

    @abstractmethod
    def Graph2():
        pass

    @abstractmethod
    def check():
        pass

#Creating Main frame.
class Main(tk.Frame, MainStructure):
    def __init__(self,master):
        super().__init__(master)
        self.configure(bg="#FFFACD", highlightbackground="black",
highlightthickness=1)
        self.pack(side=tk.LEFT)
        self.propagate(False)
        self.configure(width=800, height=500)

        self.create_labels()
        self.create_entries()
        self.create_buttons()

#Creating the labels.
    def create_labels(self):
        self.main_label = tk.Label(self, text = "Analytics",
bg="#98FB98", fg="#000000", font=("Arial", 25),
highlightbackground="black", highlightthickness=1)
        self.main_label.place(relx=.53, rely=.1, anchor=CENTER)

        self.BMI_label=tk.Label(self, text="Your BMI", bg="#FFFACD")
        self.BMI_label.place(relx=.17, rely=0.225)

        self.AvgBMI_label=tk.Label(self, text="Average BMI",
bg="#FFFACD")
        self.AvgBMI_label.place(relx=.155, rely=0.335)

        self.bodyfat_label=tk.Label(self, text="Your bodyfat%",
bg="#FFFACD")
        self.bodyfat_label.place(relx=.145, rely=0.585)

        self.AvgBodyfat_label=tk.Label(self, text="Average bodyfat%",
bg="#FFFACD")
        self.AvgBodyfat_label.place(relx=.130, rely=0.695)

#Creating the entries.
    def create_entries(self):
```

```
        self.BMI=tk.Entry(self, font=("Calibri",18))
        self.BMI.config(width=28)
        self.BMI.place(relx=.328, rely=0.215)
        self.BMI.insert(0,str(SQL.getBMI(GlobalVariables.username)))
        self.check()

        self.AvgBMI=tk.Entry(self, font=("Calibri",18))
        self.AvgBMI.config(width=28)
        self.AvgBMI.place(relx=.328, rely=0.325)
        self.AvgBMI.insert(0,str("{:.1f}".format(SQL.getAvgBMI())))

        self.Bodyfat=tk.Entry(self, font=("Calibri",18))
        self.Bodyfat.config(width=28)
        self.Bodyfat.place(relx=.328, rely=0.575)

self.Bodyfat.insert(0,str(SQL.getBodyfat(GlobalVariables.username)))

        self.AvgBodyfat=tk.Entry(self, font=("Calibri",18))
        self.AvgBodyfat.config(width=28)
        self.AvgBodyfat.place(relx=.328, rely=0.685)

self.AvgBodyfat.insert(0,str("{:.1f}".format(SQL.getAvgBodyfat())))

#Creating the buttons.
    def create_buttons(self):
        self.graph1=tk.Button(self, text="Graph it!", font=("Calibri",
12), command=self.Graph1)
        self.graph1.place(relx=0.45, rely=0.45)

        self.graph2=tk.Button(self, text="Graph it!", font=("Calibri",
12), command=self.Graph2)
        self.graph2.place(relx=0.45, rely=0.81)

#Setting up the 1st graph.
    def Graph1(self):
        YourBMI = int(SQL.getBMI(GlobalVariables.username))
        AvgBMI = int(SQL.getAvgBMI())
        UKAvgBMI = GlobalVariables.UKAvgBMI
        BMI_data = {"You":YourBMI,
                    "FitMetrix average":AvgBMI,
                    "UK's average":UKAvgBMI}

        plt.rcParams["axes.prop_cycle"] = plt.cycler(
            color =
["#4C2A85","#BE96FF","#957DAD","#5E366E","#A98CCC"])

        fig1, ax1 = plt.subplots()
        ax1.bar(BMI_data.keys(), BMI_data.values())
        ax1.set_title("BMI comparison")
        ax1.set_xlabel("Reference")
        ax1.set_ylabel("BMI")
        plt.show()
```

```
#Setting up the 2nd graph.
    def Graph2(self):
        if SQL.getGender(GlobalVariables.username) == "Male":
            YourBodyfat =
int(SQL.getBodyfat(GlobalVariables.username))
            AvgBodyfat = int(SQL.getAvgBodyfat())
            UKAvgBodyfat = GlobalVariables.UKAvgBodyfat_Male

            Bodyfat_data = {"You":YourBodyfat,
                            "FitMetrix average":AvgBodyfat,
                            "UK's average":UKAvgBodyfat}

            plt.rcParams["axes.prop_cycle"] = plt.cycler(
            color =
["#4C2A85","#BE96FF","#957DAD","#5E366E","#A98CCC"])

            fig1, ax1 = plt.subplots()
            ax1.bar(Bodyfat_data.keys(), Bodyfat_data.values())
            ax1.set_title("Bodyfat comparison")
            ax1.set_xlabel("Reference(Men)")
            ax1.set_ylabel("Bodyfat")
            plt.show()

        elif SQL.getGender(GlobalVariables.username) == "Female":
            YourBodyfat =
int(SQL.getBodyfat(GlobalVariables.username))
            AvgBodyfat = int(SQL.getAvgBMI())
            UKAvgBodyfat = GlobalVariables.UKAvgBodyfat_Female

            Bodyfat_data = {"You":YourBodyfat,
                            "FitMetrix average":AvgBodyfat,
                            "UK's average":UKAvgBodyfat}

            plt.rcParams["axes.prop_cycle"] = plt.cycler(
            color =
["#4C2A85","#BE96FF","#957DAD","#5E366E","#A98CCC"])

            fig1, ax1 = plt.subplots()
            ax1.bar(Bodyfat_data.keys(), Bodyfat_data.values())
            ax1.set_title("Bodyfat guide")
            ax1.set_xlabel("Reference(Women)")
            ax1.set_ylabel("Bodyfat")
            plt.show()

#Check for existence of data in the database.
    def check(self):
        if self.BMI.get() == "None":
            messagebox.showerror(title="Error", message="Please upload
your progress first.")
```

mySQL.py

```
import sqlite3
from Login_Page import *
from Diet_Menu import *
from Main_Menu import *
from FindDietMenu import *
from FindWorkoutMenu import *
from Workout_Menu import *
from ShowDietsPage import *

#Creating the main three tables in the database.
def create_users():
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""CREATE TABLE credentials (
              username text PRIMARY KEY,
              hashpassword int,
              name text,
              age int,
              gender text,
              height int,
              weight int)""")

    c.execute("""CREATE TABLE Diets(
              dietID int PRIMARY KEY,
              diet text)""")

    c.execute("""CREATE TABLE Workouts(
              workoutID int PRIMARY KEY,
              workout text)""")

    conn.commit()
    conn.close()

#Creating the UserDiet table.
def create_UserDiet():
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""CREATE TABLE UserDiet(
              username text,
              dietID int,
              FOREIGN KEY(dietID)
              REFERENCES Diets(dietID),
              FOREIGN KEY(username)
              REFERENCES credentials(username))""")

    conn.commit()
    conn.close()
```

```
#Creating the UserWorkout table.
def create_UserWorkout():
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""CREATE TABLE UserWorkout(
              username text,
              workoutID int,
              FOREIGN KEY(workoutID)
              REFERENCES Workouts(workoutID),
              FOREIGN KEY(username)
              REFERENCES credentials(username))""")

    conn.commit()
    conn.close()

#Creating the UserProfile table.
def create_UserProfile():
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""CREATE TABLE UserProfile(
              username text,
              bodyfat FLOAT,
              BMI float,
              FOREIGN KEY(username)
              REFERENCES credentials(username))""")

    conn.commit()
    conn.close()

#Registering a new user.
def register_users(username_entry,
password_entry,name,age,gender,height,weight):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""INSERT INTO credentials
VALUES(:username,:password,:name,:age,:gender,:height,:weight)""",
              {"username":username_entry,
              "password":password_entry,
              "name":name,
              "age":age,
              "gender":gender,
              "height":height,
              "weight":weight})

    conn.commit()
    conn.close()

#Used to get username from credentials.
def Username_get(data):
```

```
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT username
                    FROM credentials
                    WHERE username=(?)""",
                    (data,))
        f = c.fetchall()
        return f
    except:
        pass

    conn.commit()
    conn.close()

#Used to get hashedpass from credentials.
def getHashedPassword(username):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT hashpassword
                    FROM credentials
                    WHERE username=(?)""",
                    (username,))
        data = c.fetchall()

        return data[0][0]
    except:
        pass

#Used to get username as a string from credentials.
def Username_get2(data):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT username
                    FROM credentials
                    WHERE username=(?)""",
                    (data,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Used to get name from credentials.
def getName(data):
```

```
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT name
                    FROM credentials
                    WHERE username=(?)""",
                    (data,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Used to get height from credentials.
def getHeight(data):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT height
                    FROM credentials
                    WHERE username=(?)""",
                    (data,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Used to get weight from credentials.
def getWeight(data):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT weight
                    FROM credentials
                    WHERE username=(?)""",
                    (data,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()
```

```
#Used to get age from credentials.
def getAge(data):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT age
                    FROM credentials
                    WHERE username=(?)""",
                    (data,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Used to get gender from credentials.
def getGender(data):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT gender
                    FROM credentials
                    WHERE username=(?)""",
                    (data,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Updating info into credentials.
def updateInfo(username, name, age, gender, height, weight):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""UPDATE credentials SET
                 name = :name,
                 height = :height,
                 weight = :weight,
                 age = :age,
                 gender = :gender

                WHERE username = :username""",

                {"name":name,
                 "height":height,
```

```
                "weight":weight,
                "age":age,
                "gender":gender,
                "username":username})

    conn.commit()
    conn.close()

#Updating UserProfile.
def UpdateUserProfile(username,bodyfat,BMI):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""UPDATE userProfile SET
                BMI = :BMI,
                bodyfat = :bodyfat

                WHERE username = :username""",

                {"username":username,
                 "BMI":BMI,
                 "bodyfat":bodyfat})

    conn.commit()
    conn.close()

#Adding diet plans into the Diets table.
def addDietPlan(dietID, diet):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""INSERT OR IGNORE INTO Diets VALUES(:dietID,
:diet)""",
              {"dietID":dietID,
               "diet":diet})

    conn.commit()
    conn.close()

#Getting diet plan from Diets.
def getDietPlan(data):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT diet
                     FROM Diets
                     WHERE dietID=(?)""",
                  (data,))
        f = c.fetchall()
        return f[0][0]
    except:
```

```
        pass

    conn.commit()
    conn.close()

#Adding dietID to UserDiet
def addDietID(username,dietID):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""INSERT INTO userDiet VALUES(:username, :dietID)""",
              {"username":username,
               "dietID":dietID})

    conn.commit()
    conn.close()

#Deleting diet from UserDiet for a specific given username.
def deleteSavedDiet(username):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""DELETE FROM UserDiet WHERE username=(?)""",
              (username,))

    conn.commit()
    conn.close()

#Getting dietID from UserDiet for a specific given username.
def getDietID(username):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT dietID
                     FROM UserDiet
                     WHERE username=(?)""",
                  (username,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Adding workout to the Workouts table.
def addWorkoutPlan(workoutID, workout):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""INSERT OR IGNORE INTO Workouts VALUES(:workoutID,
```

```
:workout)""",
              {"workoutID":workoutID,
               "workout":workout})

    conn.commit()
    conn.close()

#Getting workout from Workouts.
def getWorkoutPlan(data):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT workout
                     FROM Workouts
                     WHERE workoutID=(?)""",
                  (data,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Adding WorkoutID to UserWorkout.
def addWorkoutID(username,workoutID):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""INSERT INTO userWorkout VALUES(:username,
:workoutID)""",
              {"username":username,
               "workoutID":workoutID})

    conn.commit()
    conn.close()

#Deleting workout for UserWorkout for a specific given username.
def deleteSavedWorkout(username):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""DELETE FROM UserWorkout WHERE username=(?)""",
              (username,))

    conn.commit()
    conn.close()

#Getting workout ID from UserWorkout for a specific given username.
def getWorkoutID(username):
    conn = sqlite3.connect("credentials_book.db")
```

```
    c = conn.cursor()

    try:
        c.execute("""SELECT workoutID
                     FROM UserWorkout
                     WHERE username=(?)""",
                  (username,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Adding details to UserProfile for a specific given username.
def addtoUserProfile(username,bodyfat,BMI):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""INSERT INTO userProfile
VALUES(:username,:bodyfat,:BMI)""",
              {"username":username,
               "BMI":BMI,
               "bodyfat":bodyfat})

    conn.commit()
    conn.close()

#Updating weight for a user in the table credentials.
def updateWeight(username, weight):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""UPDATE credentials SET
                 weight = :weight

                 WHERE username = :username""",
                 {"username":username,
                  "weight":weight})

    conn.commit()
    conn.close()

#Deleting UserProfile.
def deleteUserProfile(username):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    c.execute("""DELETE FROM UserProfile WHERE username=(?)""",
              (username,))
```

```
    conn.commit()
    conn.close()

#Getting saved diet for a specific given username. Uses INNER JOIN.
def getSavedDietPlan(username):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT Diets.diet
                     FROM credentials
                     INNER JOIN UserDiet ON credentials.username =
UserDiet.username
                     INNER JOIN Diets ON UserDiet.dietID =
Diets.dietID
                     WHERE credentials.username=(?)""",
                  (username,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Getting saved workout for a specific given username. Uses INNER JOIN.
def getSavedWorkoutPlan(username):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT Workouts.workout
                     FROM credentials
                     INNER JOIN UserWorkout ON credentials.username =
UserWorkout.username
                     INNER JOIN Workouts ON UserWorkout.workoutID =
Workouts.workoutID
                     WHERE credentials.username=(?)""",
                  (username,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Getting average BMI from all of the users in the UserProfile table.
Uses Aggregate SQL.
def getAvgBMI():
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()
```

```
    try:
        c.execute("""SELECT avg(BMI)
                     FROM userProfile""")
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Getting average bodyfat from all of the users in the UserProfile
table. Uses Aggregate SQL.
def getAvgBodyfat():
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT avg(bodyfat)
                     FROM userProfile""")
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Getting BMI for a specific given username.
def getBMI(username):
    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT BMI
                     FROM userProfile
                     WHERE username = (?)""",
                     (username,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()

#Getting bodyfat for a specific given username.
def getBodyfat(username):

    conn = sqlite3.connect("credentials_book.db")
    c = conn.cursor()
```

```
    try:
        c.execute("""SELECT bodyfat
                     FROM userProfile
                     WHERE username = (?)""",
                     (username,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()
```

Testing :

https://drive.google.com/drive/folders/1c-X2Otk98hmg9oBtKp4tzDvud4LY6CE-?usp=sharing

Test No	1
Objective Ref.	1.1
Test type	Valid
Test Data	Username: ajamidar Password: arnav
Expected Outcome	Username & password should be stored in the database and the user is registered. User info page should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_1
Improvements	N/A

Test No	2
Objective Ref.	1.1
Test type	Invalid
Test Data	Username: hellothereimhere Password: aj45072605
Expected Outcome	Error message should pop up - “Invalid login credentials”
Actual outcome	Jamidar_Arnav_NEA_Test_2
Improvements	N/A

Test No	3
Objective Ref.	1.1
Test type	Boundary
Test Data	Username: Ajack Password: aj1
Expected Outcome	Username & password should be stored in the database and the user is registered. User info page should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_3
Improvements	N/A

Test No	4
Objective Ref.	1.1.1
Test type	Invalid
Test Data	Username: ajamidar Password: aj4507
Expected Outcome	Error message should pop up - “Username exists”
Actual outcome	Jamidar_Arnav_NEA_Test_4
Improvements	N/A

Test No	5
Objective Ref.	1.1.2
Test type	Valid
Test Data	N/A
Expected Outcome	User Info page should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_5
Improvements	N/A

Test No	6
Objective Ref.	1.1.2.1
Test type	Valid
Test Data	Name: Arnav Jamidar, Height:177, Weight: 74, Age: 18, Gender:Male
Expected Outcome	The inputted data should be stored in the database and the Main menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_6
Improvements	N/A

Test No	7
Objective Ref.	1.1.2.2
Test type	Invalid
Test Data	Name: Alyssia Jackson, Height:hello, Weight: 57, Age: 27, Gender:Female
Expected Outcome	Error message should pop up - “Invalid inputs”
Actual outcome	Jamidar_Arnav_NEA_Test_7
Improvements	N/A

Test No	8
Objective Ref.	1.1.2.1
Test type	Boundary
Test Data	Name: Alyssia Jackson, Height:200, Weight: 57, Age: 27, Gender:Female
Expected Outcome	The inputted data should be stored in the database and the Main menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_8
Improvements	N/A

Test No	9
Objective Ref.	1.2
Test type	Valid
Test Data	Username: ajamidar Password: arnav
Expected Outcome	Access into the program should be granted, and the main menu will appear.
Actual outcome	Jamidar_Arnav_NEA_Test_9
Improvements	N/A

Test No	10
Objective Ref.	1.2.1
Test type	Invalid
Test Data	Username: football Password: arnav
Expected Outcome	Error message should pop up - “Username does not exist”
Actual outcome	Jamidar_Arnav_NEA_Test_10
Improvements	N/A

Test No	11
Objective Ref.	1.2.2
Test type	Invalid
Test Data	Username: ajamidar Password: hello
Expected Outcome	Error message should pop up - “Incorrect password”
Actual outcome	Jamidar_Arnav_NEA_Test_11
Improvements	N/A

Test No	12
Objective Ref.	2.1
Test type	Valid
Test Data	N/A
Expected Outcome	Main menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_12
Improvements	N/A

Test No	13
Objective Ref.	2.1.2
Test type	Valid
Test Data	N/A
Expected Outcome	Diet menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_13
Improvements	N/A

Test No	14
Objective Ref.	2.1.2.1.4
Test type	Valid
Test Data	N/A
Expected Outcome	Main menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_14
Improvements	N/A

Test No	15
Objective Ref.	2.1.2.1
Test type	Valid
Test Data	N/A
Expected Outcome	Find diet plans menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_15
Improvements	N/A

Test No	16
Objective Ref.	2.1.2.1.1
Test type	Valid
Test Data	Goal: Cut, Height: 177, Weight: 74, Age: 18, Dietary preference: Non-vegetarian
Expected Outcome	Two diet plans should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_16
Improvements	N/A

Test No	17
Objective Ref.	2.1.2.1.1.1
Test type	Invalid
Test Data	Goal: Cut, Height: hola, Weight: 72, Age: 18, Dietary preference: Non-vegetarian
Expected Outcome	An error message should pop up which states - “Invalid inputs”.
Actual outcome	Jamidar_Arnav_NEA_Test_17
Improvements	N/A

Test No	18
Objective Ref.	2.1.2.1.1
Test type	Boundary
Test Data	Goal: Cut, Height: 177, Weight: 74, Age: 15, Dietary preference: Non-vegetarian
Expected Outcome	Two diet plans should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_18
Improvements	N/A

Test No	19
Objective Ref.	2.1.2.1.2
Test type	Valid
Test Data	N/A
Expected Outcome	Saved diet page should appear with the saved diet plan, and the saved diet plan should be saved in the database.
Actual outcome	Jamidar_Arnav_NEA_Test_19
Improvements	N/A

Test No	20
Objective Ref.	2.1.2.1.4
Test type	Valid
Test Data	N/A
Expected Outcome	Diet menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_20
Improvements	N/A

Test No	21
Objective Ref.	2.1.2.2
Test type	Valid
Test Data	N/A
Expected Outcome	Saved diet page should appear with the saved diet plan.
Actual outcome	Jamidar_Arnav_NEA_Test_21
Improvements	N/A

Test No	22
Objective Ref.	2.1.2.2.1
Test type	Valid
Test Data	N/A
Expected Outcome	Currently saved diet plan should be deleted from the page and the database.
Actual outcome	Jamidar_Arnav_NEA_Test_22
Improvements	N/A

Test No	23
Objective Ref.	2.1.2.2.2
Test type	Valid
Test Data	N/A
Expected Outcome	Diet menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_23
Improvements	N/A

Test No	24
Objective Ref.	2.1.3
Test type	Valid
Test Data	N/A
Expected Outcome	Workout menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_24
Improvements	N/A

Test No	25
Objective Ref.	2.1.3.3
Test type	Valid
Test Data	N/A
Expected Outcome	Main menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_25
Improvements	N/A

Test No	26
Objective Ref.	2.1.3.1
Test type	Valid
Test Data	N/A
Expected Outcome	Find workout menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_26
Improvements	N/A

Test No	27
Objective Ref.	2.1.3.1.1
Test type	Valid
Test Data	Goal: Cut, Height: 177, Weight: 74, Age: 18, Activity level: 2-3 days a week
Expected Outcome	Two workout plans should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_27
Improvements	N/A

Test No	28
Objective Ref.	2.1.3.1.1.1
Test type	Invalid
Test Data	Goal: Cut, Height: hola, Weight: dias, Age: rtr, Activity level: 2-3 days a week
Expected Outcome	An error message should pop up which states - “Invalid inputs”.
Actual outcome	Jamidar_Arnav_NEA_Test_28
Improvements	N/A

Test No	29
Objective Ref.	2.1.3.1.1
Test type	Boundary
Test Data	Goal: Cut, Height: 177, Weight: 74, Age: 15, Activity level: 2-3 days a week
Expected Outcome	Two workout plans should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_29
Improvements	N/A

Test No	30
Objective Ref.	2.1.3.1.2
Test type	Valid
Test Data	N/A
Expected Outcome	Saved workout page should appear with the saved workout plan, and the saved workout plan should be saved in the database.
Actual outcome	Jamidar_Arnav_NEA_Test_30
Improvements	N/A

Test No	31
Objective Ref.	2.1.3.1.4
Test type	Valid
Test Data	N/A
Expected Outcome	Workout page should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_31
Improvements	N/A

Test No	32
Objective Ref.	2.1.3.2
Test type	Valid
Test Data	N/A
Expected Outcome	Saved workout page should appear with the saved workout plan.
Actual outcome	Jamidar_Arnav_NEA_Test_32
Improvements	N/A

Test No	33
Objective Ref.	2.1.3.2.1
Test type	Valid
Test Data	N/A
Expected Outcome	The saved workout plan should be deleted from the page and the database.
Actual outcome	Jamidar_Arnav_NEA_Test_33
Improvements	N/A

Test No	34
Objective Ref.	2.1.3.2.2
Test type	Valid
Test Data	N/A
Expected Outcome	Workout Menu appears.
Actual outcome	Jamidar_Arnav_NEA_Test_34
Improvements	N/A

Test No	35
Objective Ref.	2.1.4
Test type	Valid
Test Data	N/A
Expected Outcome	Your progress menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_35
Improvements	N/A

Test No	36
Objective Ref.	2.1.4.1
Test type	Valid
Test Data	N/A
Expected Outcome	Update progress menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_36
Improvements	N/A

Test No	37
Objective Ref.	2.1.4.1.1
Test type	Valid
Test Data	N/A
Expected Outcome	Update progress page should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_37
Improvements	N/A

Test No	38
Objective Ref.	2.1.4.1.1.1
Test type	Valid
Test Data	BMI: 21, Body fat%: 17, Weight: 54, Plan check: Yes
Expected Outcome	The progress should be saved in the database and the Your progress menu appears.
Actual outcome	Jamidar_Arnav_NEA_Test_38
Improvements	N/A

Test No	39
Objective Ref.	2.1.4.1.1.1.1
Test type	Invalid
Test Data	BMI: wrong, Body fat%: 17, Weight: 54, Plan check: Yes
Expected Outcome	Error message should appear which states - “Invalid inputs”
Actual outcome	Jamidar_Arnav_NEA_Test_39
Improvements	N/A

Test No	40
Objective Ref.	2.1.4.1.1.1
Test type	Boundary
Test Data	BMI: 40, Body fat%: 17, Weight: 74, Plan check: Yes
Expected Outcome	The progress should be saved in the database and the Your progress menu appears.
Actual outcome	Jamidar_Arnav_NEA_Test_40
Improvements	N/A

Test No	41
Objective Ref.	2.1.4.1.1.2
Test type	Valid
Test Data	BMI: 23, Body fat%: 18, Weight: 74, Plan check: Yes
Expected Outcome	The progress should be updated in the database and the Your progress menu appears.
Actual outcome	Jamidar_Arnav_NEA_Test_41
Improvements	N/A

Test No	42
Objective Ref.	2.1.4.1.1.2.1
Test type	BMI: lose, Body fat%: 18, Weight: 74, Plan check: Yes
Test Data	N/A
Expected Outcome	An error message should appear which states - “Invalid inputs”.
Actual outcome	Jamidar_Arnav_NEA_Test_42
Improvements	N/A

Test No	43
Objective Ref.	2.1.4.1.1.2
Test type	Boundary
Test Data	BMI: 23, Body fat%:55, Weight: 75, Plan check: Yes
Expected Outcome	The progress should be updated in the database and the Your progress menu appears.
Actual outcome	Jamidar_Arnav_NEA_Test_43
Improvements	N/A

Test No	44
Objective Ref.	2.1.4.1.1.3
Test type	Valid
Test Data	N/A
Expected Outcome	Update progress menu appears.
Actual outcome	Jamidar_Arnav_NEA_Test_44
Improvements	N/A

Test No	45
Objective Ref.	2.1.4.1.2
Test type	Valid
Test Data	N/A
Expected Outcome	The calculations menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_45
Improvements	N/A

Test No	46
Objective Ref.	2.1.4.1.2.1
Test type	Valid
Test Data	N/A
Expected Outcome	The body fat percentage calculator page should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_46
Improvements	N/A

Test No	47
Objective Ref.	2.1.4.1.2.1.1
Test type	Invalid
Test Data	Height: 1.77, Weight: 75, Age: alnassr, Gender: Male
Expected Outcome	An error message should appear which says - “Invalid inputs”.
Actual outcome	Jamidar_Arnav_NEA_Test_47
Improvements	N/A

Test No	48
Objective Ref.	2.1.4.1.2.1.2
Test type	Valid
Test Data	Height: 1.77, Weight: 75, Age:18, Gender: Male
Expected Outcome	Result page should appear with the output of the calculation.
Actual outcome	Jamidar_Arnav_NEA_Test_48
Improvements	N/A

Test No	49
Objective Ref.	2.1.4.1.2.1.2
Test type	Boundary
Test Data	Height: 2.00, Weight: 76, Age:19, Gender: Male
Expected Outcome	Result page should appear with the output of the calculation.
Actual outcome	Jamidar_Arnav_NEA_Test_49
Improvements	N/A

Test No	50
Objective Ref.	2.1.4.1.2.1.3
Test type	Valid
Test Data	N/A
Expected Outcome	Calculations menu appears.
Actual outcome	Jamidar_Arnav_NEA_Test_50
Improvements	N/A

Test No	51
Objective Ref.	2.1.4.1.2.2
Test type	Valid
Test Data	N/A
Expected Outcome	The BMI Score calculator page should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_51
Improvements	N/A

Test No	52
Objective Ref.	2.1.4.1.2.2.1
Test type	Invalid
Test Data	Height: nope, Weight: 71, Age:18
Expected Outcome	An error message should appear which says - “Invalid inputs”.
Actual outcome	Jamidar_Arnav_NEA_Test_52
Improvements	N/A

Test No	53
Objective Ref.	2.1.4.1.2.2.2
Test type	Valid
Test Data	Height: 1.81, Weight: 72, Age:18
Expected Outcome	Result page should appear with the output of the calculation.
Actual outcome	Jamidar_Arnav_NEA_Test_53
Improvements	N/A

Test No	54
Objective Ref.	2.1.4.1.2.2.2
Test type	Boundary
Test Data	Height: 2.00, Weight: 72, Age:18
Expected Outcome	Result page should appear with the output of the calculation.
Actual outcome	Jamidar_Arnav_NEA_Test_54
Improvements	N/A

Test No	55
Objective Ref.	2.1.4.1.2.2.3
Test type	Valid
Test Data	N/A
Expected Outcome	Calculations menu appears.
Actual outcome	Jamidar_Arnav_NEA_Test_55
Improvements	N/A

Test No	56
Objective Ref.	2.1.4.1.3
Test type	Valid
Test Data	N/A
Expected Outcome	Update progress menu appears.
Actual outcome	Jamidar_Arnav_NEA_Test_56
Improvements	N/A

Test No	57
Objective Ref.	2.1.4.2
Test type	Valid
Test Data	N/A
Expected Outcome	The analytics page should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_57
Improvements	N/A

Test No	58
Objective Ref.	2.1.4.2.1
Test type	Valid
Test Data	N/A
Expected Outcome	The BMI comparison graph should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_58
Improvements	N/A

Test No	59
Objective Ref.	2.1.4.2.1.1
Test type	Invalid
Test Data	N/A
Expected Outcome	An error message is shown which says - “Please upload your progress first”.
Actual outcome	Jamidar_Arnav_NEA_Test_59
Improvements	N/A

Test No	60
Objective Ref.	2.1.4.2.2
Test type	Valid
Test Data	N/A
Expected Outcome	The Bodyfat comparison graph should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_60
Improvements	N/A

Test No	61
Objective Ref.	2.1.4.2.2.1
Test type	Invalid
Test Data	N/A
Expected Outcome	An error message is shown which says - “Please upload your progress first”.
Actual outcome	Jamidar_Arnav_NEA_Test_61
Improvements	N/A

Test No	62
Objective Ref.	2.1.4.2.3
Test type	Valid
Test Data	N/A
Expected Outcome	Your progress menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_62
Improvements	N/A

Test No	63
Objective Ref.	2.1.4.3
Test type	Valid
Test Data	N/A
Expected Outcome	Main menu should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_63
Improvements	N/A

Test No	64
Objective Ref.	2.2
Test type	Valid
Test Data	N/A
Expected Outcome	The login page should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_64
Improvements	N/A

Test No	65
Objective Ref.	3.1.1
Test type	Valid
Test Data	N/A
Expected Outcome	Your details page should appear.
Actual outcome	Jamidar_Arnav_NEA_Test_65
Improvements	N/A

Test No	66
Objective Ref.	3.1.2
Test type	Valid
Test Data	Height: 176, Weight: 74
Expected Outcome	Edits are updated into the program's database. Main menu should appear
Actual outcome	Jamidar_Arnav_NEA_Test_66
Improvements	N/A

Test No	67

Objective Ref.	3.1.2
Test type	Invalid
Test Data	Height: 201, Weight: 75
Expected Outcome	Edits are updated into the program's database. Main menu should appear
Actual outcome	Jamidar_Arnav_NEA_Test_67
Improvements	N/A

Test No	68
Objective Ref.	3.1.2
Test type	Valid
Test Data	Height: 200, Weight: 75
Expected Outcome	Edits are updated into the program's database. Main menu should appear
Actual outcome	Jamidar_Arnav_NEA_Test_68
Improvements	N/A

Test No	69
Objective Ref.	3.3
Test type	Valid
Test Data	N/A
Expected Outcome	Main menu appears.
Actual outcome	Jamidar_Arnav_NEA_Test_69
Improvements	N/A

Evaluation

This section is the evaluation of the objectives I set in the analysis section. After implementing the design of my program, I have come across many potential improvements and I am keen to receive vital feedback from the evaluation interview.

Objectives

This is the evaluation of all of the objectives I set earlier during the analysis for this system.

1. Login details of the user must be stored in the database.
I was very successful in accomplishing this objective for my system, the final login system is very user friendly, where one can login or register to the system on the same page, making it a hassle-free experience for any new users. Furthermore, the login system is very secure too, as a hashing algorithm is used to encrypt the password inputted by new users when registering, so the passwords stored in the database are in a hashed form meaning no one can view and understand any user's password, even if they have direct access to the database's tables. All the passwords for accessing this system are very secure too with these conditions implemented when asking a new user for their password : must be greater than 2 characters and less than 20 characters; must be a combination of letters & numbers.

2. The main menu provides a hub for users to navigate through the program.
The main menu appears after logging in to the system or for new users, after registering successfully to the system. The main menu successfully makes navigating through the program seamless with four sections for four features of different categories, whilst also providing the option to logout of the program quickly at any point. The "Diet plans" section gives the user an opportunity to find diet plans based on their requirements, with the option to save a diet plan to their profile in the database, which can then be viewed/deleted from a different page. The "Workout plans" section gives the user an

opportunity to find workout plans based on their requirements, with the option to save a workout plan to their profile in the database, which can then be viewed/deleted from a different page. The “Your progress” section provides the user an option to upload their fitness data, which can be calculated using the two different calculators in this section, after which the user can view their Body Fat%/BMI comparison graphs to learn more about their body composition. The “Your details” section allows the user to edit their personal data which is stored in the database of the system. Also, the “Logout” button allows the user to logout of their account and return to the login page.

3. Users can update their details at any time.
This objective is successfully achieved through the “Your details” section of the program, where the user can update their details such as name, height, weight, gender and age at any time. Moreover, this feature is made to be more robust through the conditions set when the user updates their data, for example, height must be between 100-200 cm. This reduces the chance of invalid data being stored in the database, hence reducing the chances of errors being raised when creating graphs in the “Analytics” section of the “Your progress” menu.

Feedback

As a part of collecting feedback for my program, I interviewed my program’s end user - Pete Temperton. This interview provided a lot of crucial information around my program, highlighting successfully implemented features and a few that can be improved in the future.

Interview

Q1) Were you able to register into the system without any problems?
Yes, I must say that the login/register system is very easy to use and user friendly. The feature to login and register on the same page makes it easy for any new user to find the correct option to choose.

Q2) Do you think the requirements for a valid username and password presented by the program makes login credentials more secure?
I certainly believe that the system's validation requirements make each user's login credentials very secure and less vulnerable to any threats such as brute force attacks, whilst not being too harsh on the user's memory.

Q3) In your opinion, are there any improvements that could be made to the program's login system?
Yes, in my opinion, there could be an option to change your password if a user forgets their current password, this can be done by sending an email to the user with a new password or authentication link.

Q4) Is the main menu easy to navigate?
I think the main menu is well set out, where the placements of all of the buttons makes it easy to look for the desired option, making navigation through the program a hassle free experience.

Q5) How was your experience updating any personal information in the "Your details section"?
My experience whilst updating my personal information was very smooth, the option to change all of the major details on the same page at once really makes the task seem very simple!

Q6) Did you face any problems whilst updating your personal information?
Yes, once whilst updating my weight, I was facing an error, but there were no details given about the error leading to a bit of time wasted in trying to find the error in my input.

Q7) How did you find the "Find diet plans" feature?
This feature was the most important in starting my fitness journey, I was able to get a very good diet plan based on my requirements. The diet plan really sped up by physique transformation.

Q8) Would you like to see any improvements with the “Diet plans” section of the program?
I would like to see an option to swap out and replace individual meals in the saved diet plan based on personal preferences, so that I can swap out the dishes that don't fit my taste with other alternatives.

Q9) How did you find the “Find workout plans” feature?
I found this feature really useful, the workout plans suggested by the programs were really effective, I was also able to get a safe workout plan for my kids thanks to the specialised workout plan for kids.

Q10) Would you like to see any improvements with the “Workout plans” section of the program?
Yes, I would recommend adding a feature to find workout plans without having to input your data every time if you're searching one for the user itself.

Q11) How user friendly did you find the “Your progress” section?
This section of the program is very user friendly with clear navigation through the easy to understand terminology used to present the options.

Q12) How useful were the calculators throughout your fitness journey?
The BMI and Body Fat % calculators were an essential part of tracking my fitness journey, I used them every week to validate the progress I was making. The charts presented with the calculations’ results are very helpful too.

Q13) Would you like to see any improvements with the “Calculations” section of the program?
No, in my opinion, this section provides all the information required to make the progress you desire and with the visual representation of important information such as the BMI/Body fat chart.

Q14) Were you able to benefit from the “Analytics” section of the programs?
The ability to view the BMI & Body fat % comparison graph is a really useful feature, I was able to see my progress whilst competing with other members of the program at the same time. This feature also motivates me to keep going further into my fitness journey.

Q15) Would you like to see any improvements with the “Analytics” section of the program?
I am very satisfied with the current feature provided in the section, in my opinion, there are no improvements required.

Q16) Is it easy to logout of the program?
Yes, the logout button in the main menu makes it very convenient to logout quickly.

Analysis of the Interview

This interview further solidified my success throughout various parts of my program, whilst giving an opportunity to vital feedback including suggestions for improvement. The interview has made the need of a “Forgot password” option clear; being a crucial part of the login process, it will make sure that no user has to worry about losing their progress saved on the system. Another potential improvement would be to add a way to further customise the diet & workout plans, meaning each meal/exercise can be swapped with another appropriate alternative. This interview highlights the successful implementation of fitness analytics such as the BMI/Body Fat % graphs and their respective calculators. Furthermore, I am now sure that the program is user friendly and easy to navigate which was one of the main targets to fulfil at the time of designing of the program.

Improvements

I have listed some of the improvements that I could have made if I had the opportunity to. These will ensure the users have a smoother experience while using the program.

Improvement - 1

Aim	To allow password changes in case a user forgets their current password
Description	This can be done by sending an email verification link and then allowing the user to create a new password; and the implementation can be done by python's "smtplib" library.
Example code	

```
import smtplib
from email.mime.multipart import
MIMEMultipart
from email.mime.text import MIMEText

def send_verification_email(email,
verification_link):
    # Email configuration
    sender_email = "your_email@example.com"
    sender_password = "your_password"
    smtp_server = "smtp.example.com"
    smtp_port = 587

    # Create message container
    msg = MIMEMultipart()
    msg['From'] = sender_email
    msg['To'] = email
    msg['Subject'] = "Email Verification"

    # Email body
    body = f"Please click the following
link to verify your email address:
{verification_link}"
    msg.attach(MIMEText(body, 'plain'))

    # Connect to SMTP server and send email
    try:
        server = smtplib.SMTP(smtp_server,
smtp_port)
        server.starttls()
        server.login(sender_email,
sender_password)
        server.sendmail(sender_email,
email, msg.as_string())
        server.quit()
        print("Verification email sent
successfully.")
    except Exception as e:
        print(f"Error: Unable to send
verification email. {e}")

# Example usage
```

	`email = "recipient@example.com"`
	`verification_link = "https://example.com/verify?token=123456"`
	`send_verification_email(email, verification_link)`

Improvement - 2

Aim	To allow users to translate programs into different languages.
Description	This feature will help to grow our community as people can access the program regardless of their language and location. The implementation will be made possible through the use of python's "gettext" library.

Example code	`import gettext`
	`import os`
	
	`# Set the directory containing the language translation files`
	`locale_directory = os.path.join(os.path.abspath(os.path.dirname(__file__)), 'locales')`
	`gettext.bindtextdomain('messages', locale_directory)`
	`gettext.textdomain('messages')`
	`_ = gettext.gettext`
	
	`def greet_user(language='en'):`
	`  # Set the language based on the input parameter`
	`  if language == 'fr':`
	`  gettext.translation('messages', localedir=locale_directory, languages=['fr']).install()`
	`  elif language == 'es':`
	`  gettext.translation('messages', localedir=locale_directory, languages=['es']).install()`
	`  else:`
	`  gettext.translation('messages', localedir=locale_directory, languages=['en']).install()`
	
	`  # Print the greeting message based on the current language`
	`  print(_("Hello, welcome to our program!"))`
	
	`# Example usage`
	`greet_user()  # Default language is English`
	`greet_user('fr')  # French language`
	`greet_user('es')  # Spanish language`

Improvement - 3

Aim	To allow users to edit specific meals and exercises.

Description	This feature will add more customisation to the program, increasing the satisfaction levels of a lot of users. This can be implemented by retrieving data from the database using and inserting them into entries placed in a label, therefore allowing changes to be made.
Example code	

```
#Creating the entries.
    def create_entries(self):

        self.bodyfat=tk.Entry(self,
font=("Calibri",18))
        self.bodyfat.config(width=28)
        self.bodyfat.place(relx=.275,
rely=0.315)

self.bodyfat.insert(0,GlobalVariables.bodyf
at)

#Getting bodyfat for a specific given
username.
def getBodyfat(username):

    conn =
sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT bodyfat
                     FROM userProfile
                     WHERE username =
(?)""",
                     (username,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()
```

Improvement - 4

Aim	To have separate menus in the calculators for user mode and guest mode.
Description	This feature will make the user's experience smoother by avoiding unnecessary inputs if they are using the calculator for themselves, this can be done with separate windows, retrieving data from the user's profile in the database using SQL, inserting them into pre-existing entries. Alternatively, if they are using the calculator for a friend/family member they'll have to input the data manually.

Example code	`#Creating the window.`
	`class App(tk.Tk):`
	`  def __init__(self):`
	
	`  #main setup`
	`  super().__init__()`
	`  self.title("BMI Calculator")`
	`  self.geometry("800x500")`
	`  self.configure(bg="#B9D3EE")`
	
	`  #widgets`
	`  self.user=User(self)`
	`  self.guest=Guest(self)`
	
	`  #run`
	`  self.mainloop()`
	
	`#Creating User mode Frame.`
	`class User(tk.Frame):`
	`  def __init__(self,master):`
	`  super().__init__(master)`
	`  self.configure(bg="#CDC9C9",`
	`highlightbackground="black",`
	`highlightthickness=1)`
	`  self.pack(side=tk.LEFT)`
	`  self.pack_propagate(False)`
	`  self.configure(width=170,`
	`height=500)`
	
	`  self.create_widgets()`
	
	`#Creating a Guest mode frame.`
	`class Guest(tk.Frame):`
	`  def __init__(self,master):`
	`  super().__init__(master)`
	`  self.configure(bg="#B9D3EE",`
	`highlightbackground="black",`
	`highlightthickness=1)`
	`  self.pack(side=tk.LEFT)`
	`  self.propagate(False)`
	`  self.configure(width=800,`
	`height=500)`
	
	`  self.create_labels()`
	`  self.create_buttons()`
	
	`#Creating the entries.`

```
    def create_entries(self):

        self.bodyfat=tk.Entry(self,
font=("Calibri",18))
        self.bodyfat.config(width=28)
        self.bodyfat.place(relx=.275,
rely=0.315)

self.bodyfat.insert(0,GlobalVariables.bodyf
at)

#Used to get weight from credentials.
def getWeight(data):
    conn =
sqlite3.connect("credentials_book.db")
    c = conn.cursor()

    try:
        c.execute("""SELECT weight
                    FROM credentials
                    WHERE username=(?)""",
                    (data,))
        f = c.fetchall()
        return f[0][0]
    except:
        pass

    conn.commit()
    conn.close()
```

Conclusion

Overall, I am happy with how my program turned out. All of my objectives have been accomplished, all of my tests have been successful and my end user is very satisfied with the program and uses it on a regular basis.