Abdulhafeed Alarifi

21343930  Thematic Project

Calorie counter

Contents

[How the program would ideally work: 1](#_Toc129787618)

[How the database would ideally work: 1](#_Toc129787619)

[How the front-end would be developed: 1](#_Toc129787620)

[UML: 2](#_Toc129787621)

[Spec 3](#_Toc129787622)

[Calculating the BMR: 3](#_Toc129787623)

[Tasks up for development: 5](#_Toc129787624)

# How the program would ideally work:

1. User is asked if they would like to lose or gain weight (calorie deficit/ calorie surplus).
2. User is asked to input BMI calculations (height, age, weight, sex)
3. Program prints out into console how many calories they’d have to consume to lose or gain weight according to their BMI.
4. User is asked to input food name and calories (food information is stored into the database and subtracted from the total calories).
5. User is asked to either continue (loop) or exit.
6. On exit program shows how many calories are left for the user to reach the calorie intake today.

# How the database would ideally work:

There would be two tables in the database. One table would store the users calculated calorie intake if they’d either want to lose or gain weight according to their BMI calculation. Every time the program runs the ID of the calorie value would update +1 (move down the table to a new row).

The second table would store the user’s inputted food information that would include the name of the food and its calories. Each food has a PK (unique ID) in the table. The calories are subtracted from the calorie intake every time a row (food) is created.

# How the front-end would be developed:

To create a front-end for a Java web service that allows users to input data that is then stored in a database, you would typically follow these steps:

1. Design the user interface (UI): This involves creating a visual representation of the interface that users will interact with. This can be done using a variety of tools such as HTML, CSS, and JavaScript.
2. Integrate the UI with the Java web service: You will need to connect the UI with the backend Java web service. This can be done using an API or other methods of communication.
3. Create a form to input data: This involves creating a form that allows users to input the data that will be stored in the database. The form should have fields that correspond to the database table columns.
4. Validate user input: You should validate user input to ensure that it conforms to the required format and meets any other criteria. This can be done using JavaScript or other validation frameworks.
5. Store data in the database: Once the user inputs data and submits the form, the Java web service should store the data in the database. This can be done using a database connection and SQL queries.
6. Display data: After the data is stored in the database, you may want to display it back to the user. This can be done by retrieving the data from the database and displaying it in a table or other format.

Overall, creating a front-end for a Java web service that allows users to input data into a database requires a combination of UI design, web development, and database programming skills. It can be challenging, but there are many resources available to help you get started, including online tutorials, sample code, and forums.

# UML:

Graphical user interface

Description automatically generated

# Spec

Graphical user interface, text, application

Description automatically generated

# Calculating the BMR:

Text

Description automatically generated

Text

Description automatically generated

Source: <https://www.verywellfit.com/how-many-calories-do-i-need-each-day-2506873>

Graphical user interface, application

Description automatically generated

# Tasks up for development:

1. Adding an about page to the website.
2. Making the website neater and prettier.
3. Updating the current UML to reflect the program correctly.
4. Fix the issue with the creation of data overwriting other rows in the database.
5. Figuring out how the user would be remembered through the pages of the website (session token, using user ID)
6. Adding code that deletes and creates the tables of the database every time the program runs.
7. Adding text telling the user that data may be inaccurate…
8. Adding comments to functions in code.
9. Adding more tasks to Trello.
10. Changing how the user interface appears on the website.