EECS3311

ECHO FITNESS APP

Software Design Document

Version	Date	Author(s)	Summary of Changes
1.0	13/10/23	Khoa Tran	Document created, general structure outlined, project title pending.
1.1	13/10/23	Adam Mokdad	Added charts for modules and interfaces.
1.2	14/10/23	Alex Valdez	Added test cases.
1.3	19/10/23	Khoa Tran	Added “Major Design Decisions”, some sequence diagrams, and meeting logs.
1.4	19/10/23	Alex Valdez	Added some sequence diagrams and UML diagrams.
1.5	20/10/23	Khoa Tran	Added GANTT diagram, link to GitHub, and use case 1 UML Class diagram.
1.6	20/10/23	Adam Mokdad	Added class diagram for use case 2 and 3.
1.7	20/10/10	Omer Omer	Added sequence diagram of use case 2 and 3, class diagram of the calculator class, added component diagram and class description of use case 3
1.8	07/11/2023	Alex Valdez	Updated test cases
1.9	22/11/2023	Khoa Tran	

Introduction:

Purpose: The goal of the project is to create an application that tracks and calculates a user's BMR level, BMI level, calorie intake, and nutrition goals.

Overview: The software must be able to handle the following use cases.

1. *As a user, I want to be able to create a profile in the application.*
2. *As a user, I want to be able to log my diet data in the application.*
3. *As a user, I want to be able to log my exercise in the application.*
4. *As a user, I want to be able to visualize my calory intake and my exercise over time.*
5. *As a user, I want to be able to visualize my daily nutrient intake.*
6. *As a user, I want to see how much weight in fat I will lose under my current diet and exercise pattern.*
7. *As a user, I want to know how well my diet aligns with the Canada Food Guide.*

Repository: https://github.com/adam5192/EECS3311-Echo

References:

- BMR Calculator Formula: https://www.calculator.io/bmr-calculator/#the-formula-of-katch-mcardle-3
- BMI Calculator Formula: https://www.calculator.io/bmi-calculator/
- Total Daily Energy Expenditure: https://www.verywellfit.com/what-is-energy-expenditure-3496103#toc-tdee-calculator

Major Design Decisions:

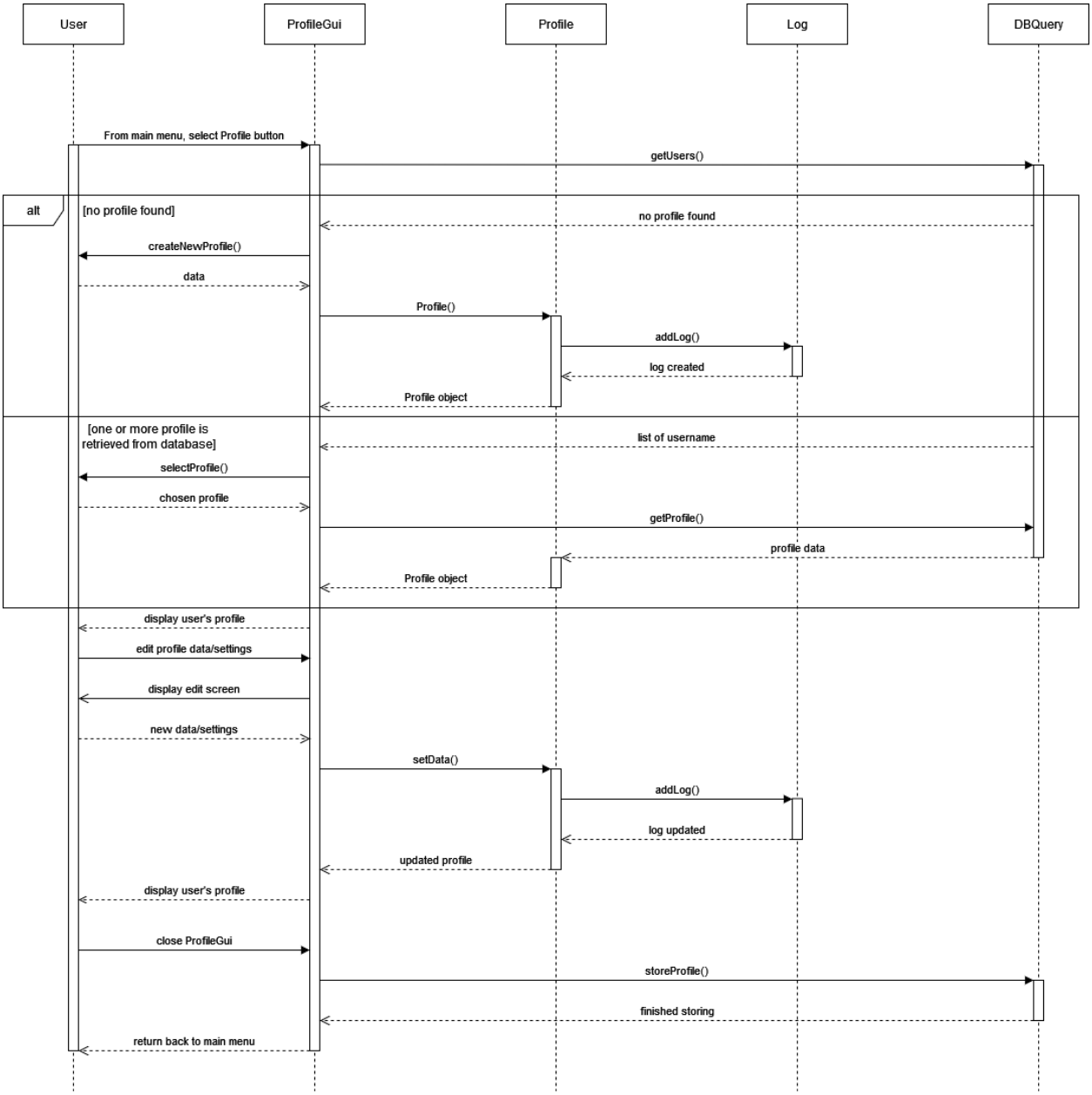
The development will be done in Java, as per the recommended programming language of the course. The database management system picked is MySQL and the 2007 Canada Food Guide was chosen as the main database as it is more detailed compared to the alternative option.

The architecture style of the application is MVC where the application is split into a Model (represented by the Log module), a Controller (further split into more specialized modules to handle the different use cases), and a View (which handles the communication between the user and the Controller).

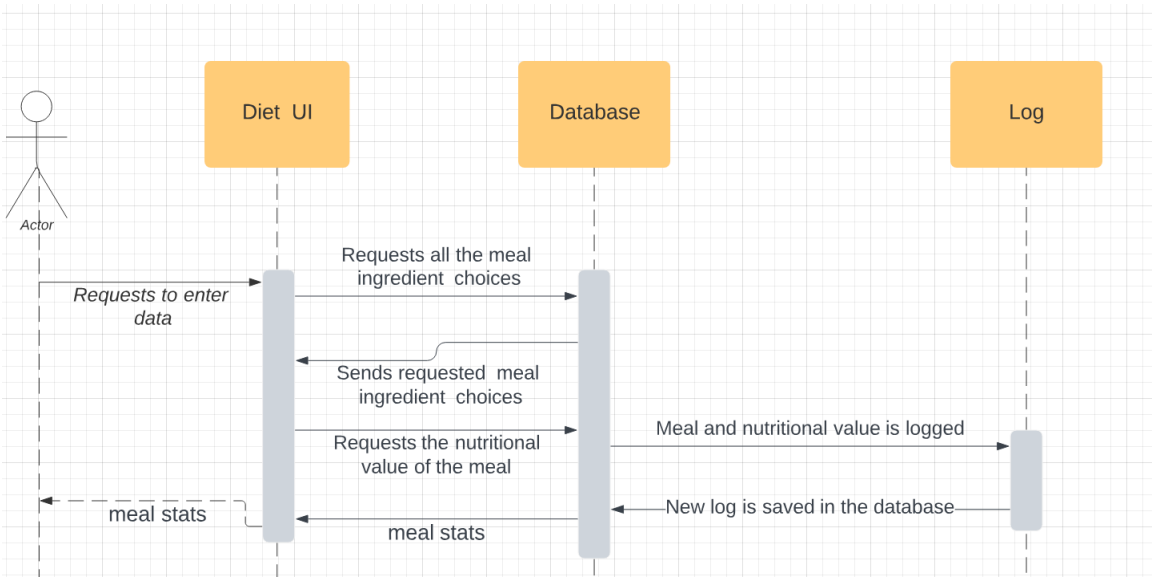
The modules are mostly unaffected by the implementations of the functions of other modules by limiting their interactions through predefined methods. Furthermore, the modules are only responsible for their specialized purposes, such as Profile is only responsible for storing and managing the data related directly to the user and information unrelated (or requires further manipulation to be related) is defined and store outside of its view. In addition, the View's requests will be pass through a façade class in order to reduce coupling with the modules.

Sequence Diagrams:

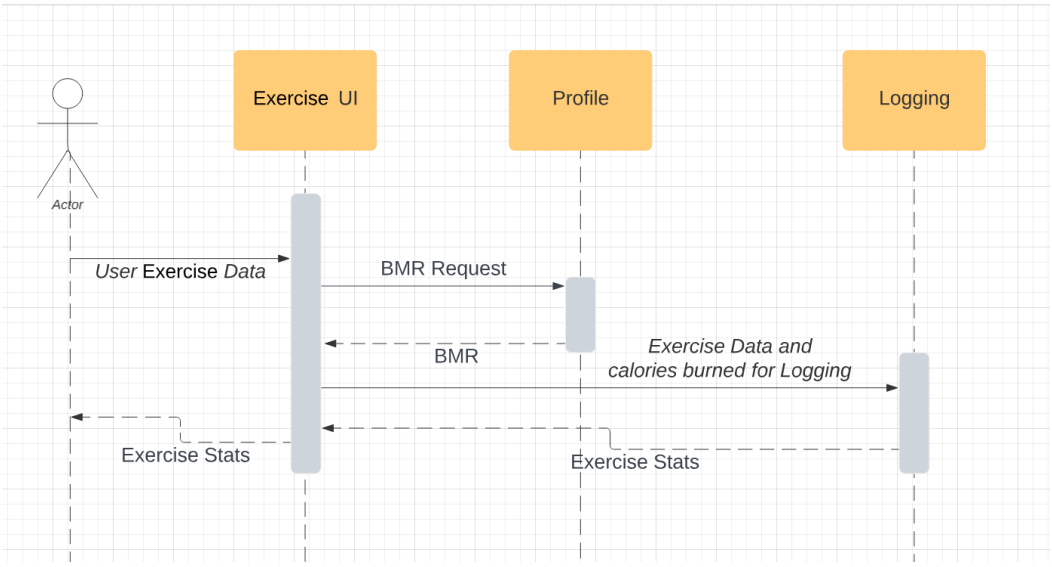
Use case 1:



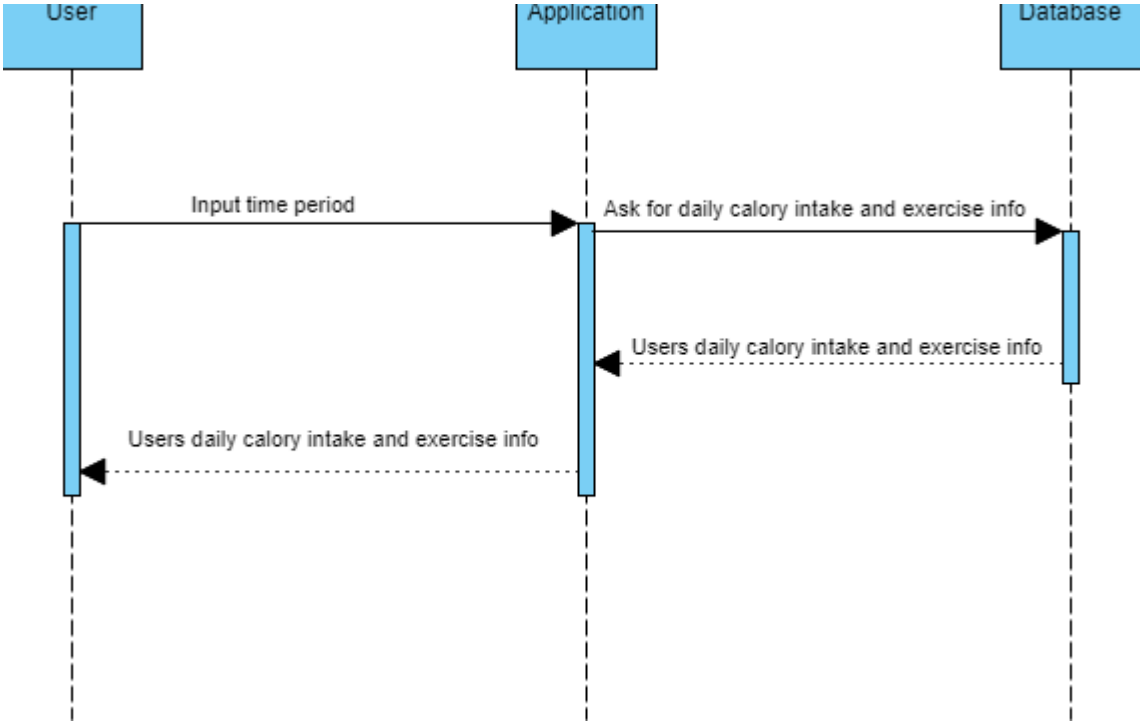
Use case 2:


Diet UI
Database
Log
Actor
Requests all the meal ingredient choices
Requests to enter data
Sends requested meal ingredient choices
Meal and nutritional value is logged
Requests the nutritional value of the meal
meal stats
New log is saved in the database
meal stats

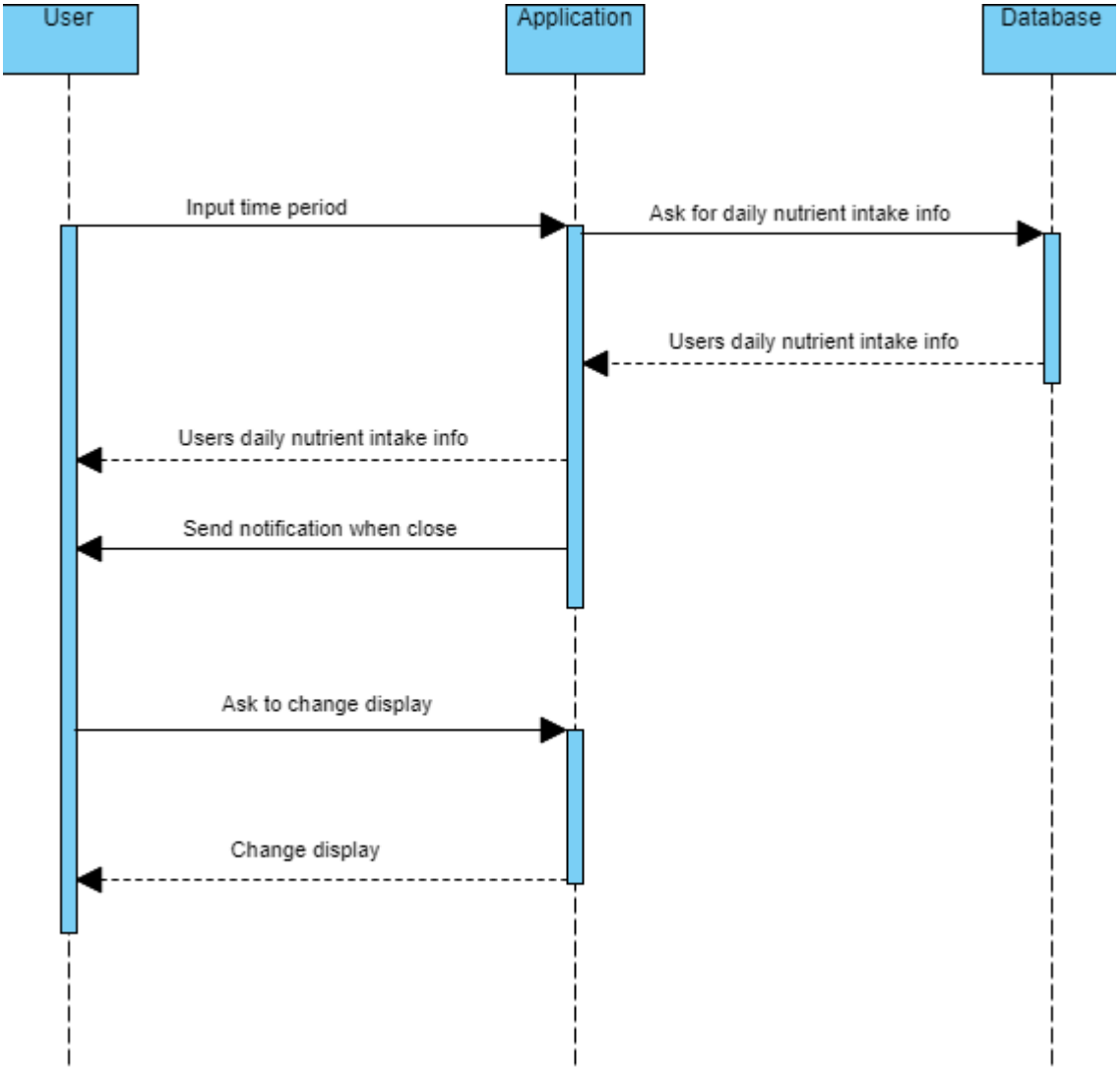
Use case 3:


Exercise UI
Profile
Logging
Actor
User Exercise Data
BMR Request
BMR
Exercise Data and calories burned for Logging
Exercise Stats
Exercise Stats

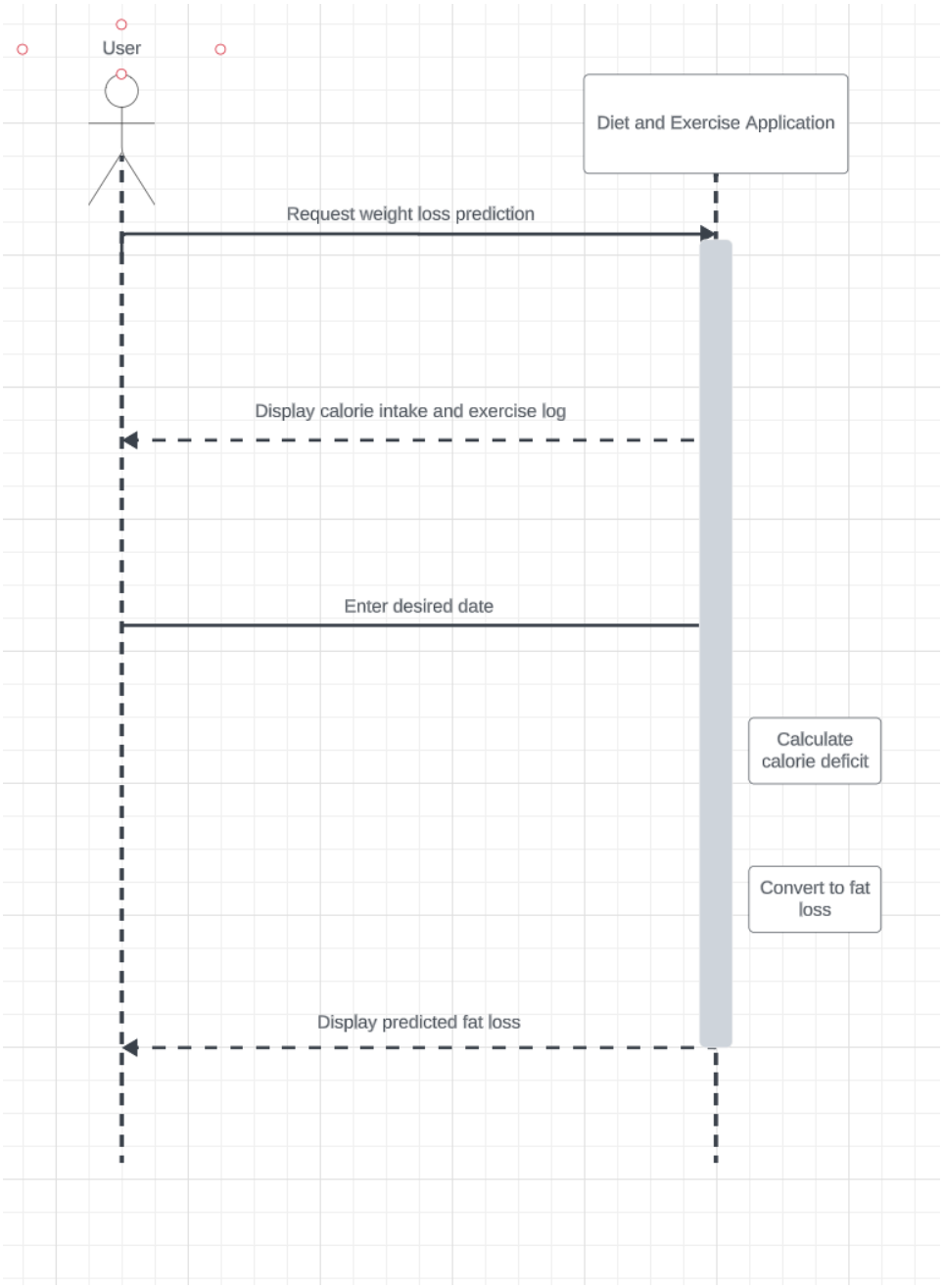
Use case 4:


User
Application
Database
Input time period
Ask for daily calory intake and exercise info
Users daily calory intake and exercise info
Users daily calory intake and exercise info

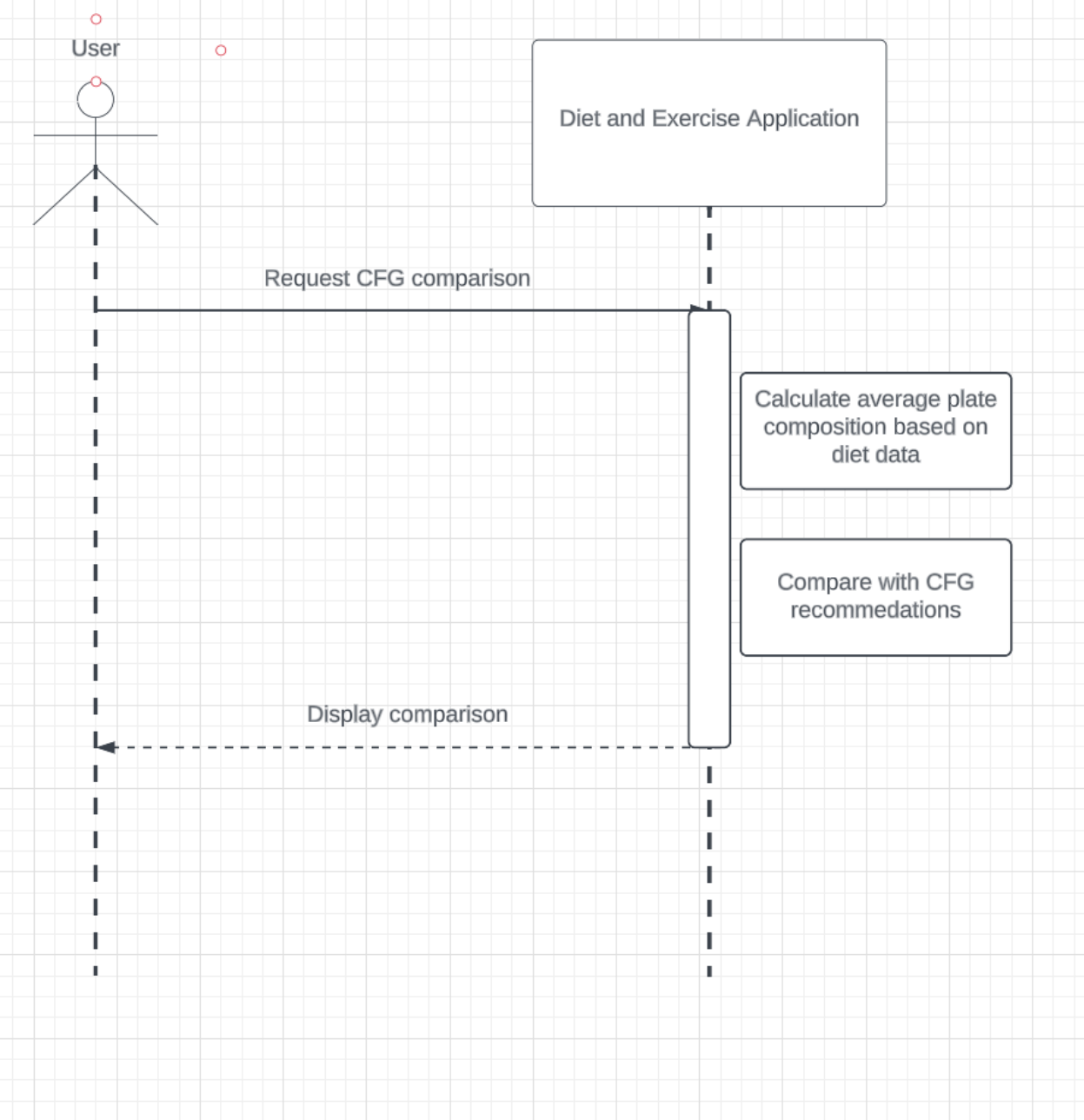
Use case 5:


User
Application
Database
Input time period
Ask for daily nutrient intake info
Users daily nutrient intake info
Users daily nutrient intake info
Send notification when close
Ask to change display
Change display

Use case 6:


User
Diet and Exercise Application
Request weight loss prediction
Display calorie intake and exercise log
Enter desired date
Calculate calorie deficit
Convert to fat loss
Display predicted fat loss

Use case 7:


User
Diet and Exercise Application
Request CFG comparison
Calculate average plate composition based on diet data
Compare with CFG recommedations
Display comparison

Architecture:

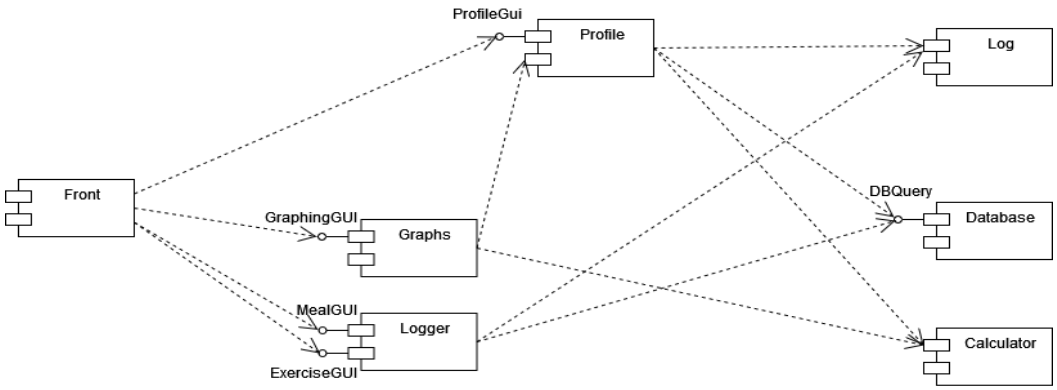
Modules

Module Name	Description	Exposed Interface Names	Interface Description
M1: User Profile	Manage user profile (creation, editing, deletion). Manage settings.	M1:I1, M1:I2	M1:I1: Interface to handle basic user data. M1:I2: Interface to manage user settings and preferences
M2: Dietary and Exercise Logging	Allows users to log dietary and exercise data, and calculate nutritional value and calories burned.	M2:I3, M2:I4	M2: I3: Interface for data input and nutrition calculation M2:I4: Interface for logging exercise and calculate calories burned
M3: Graphing Visualization	Generates visual data representations for caloric and nutritional data.	M3:I5	M3: I5: Interface to produce various visualizations
M4: Weight Prediction	Uses caloric data to predict potential weight loss	M4:I6	M4:I6: Predicts weight loss based on caloric data
M5: CFG Alignment	Compares dietary data with CFG recommendations	M5:I7	M5:I7: Evaluates and visualizes alignment with CFG
M6: Database	Handles storage, retrieval and management of all user data	M6:I8	M6:I8: Handles data storage and retrieval
M7: UI	Renders UI, handles user inputs, and manages frontend interaction.	M7:I9	M7:I9: Interfaces for displaying UI and managing user interactions.

Interfaces

Interface Name	Operations	Operation Desc
M1:I1	<void> I1:Op1() used by M7 <void> I1:Op2(int x) used by M7	Op1(): Handle creation and selection of user profiles. Op2(int x): Modify details in a user's profile.
M1:I2	<void> I2:Op3() used by M7	Op3(): Adjust user settings
M2:I3	<Nutrition Data> I3:Op4(String y) used by M7, M4, M5	Op4(String y): Input dietary data and calculate nutritional values.
M2:I4	<int> I4:Op5(int z) used by M7, M4	Op5(int z): Log exercise data and calculate calories burned.
M3:I5	<Chart> I5:Op6(Date a, Date b) used by M7	Op6(Date a, Date b): Generate a visualization for a specified date range.
M4:I6	<float> I6:Op7() used by M7	Op7(): Predict weight loss using current data.
M5:I7	<CFG Comparison> I7:Op8() used by M7	Op8(): Compare and visualize comparison with CFG recommendations.
M6:I8	<User Data> I7:Op8() used by M7, M2, M3, M4, M5	Op9(): Retrieve/store data in the database.
M7:I9	<void> I9:Op10() used by M1, M2, M3, M4, M5, M6	Op10(): Render specific UI elements based on user interaction.

Component Diagram

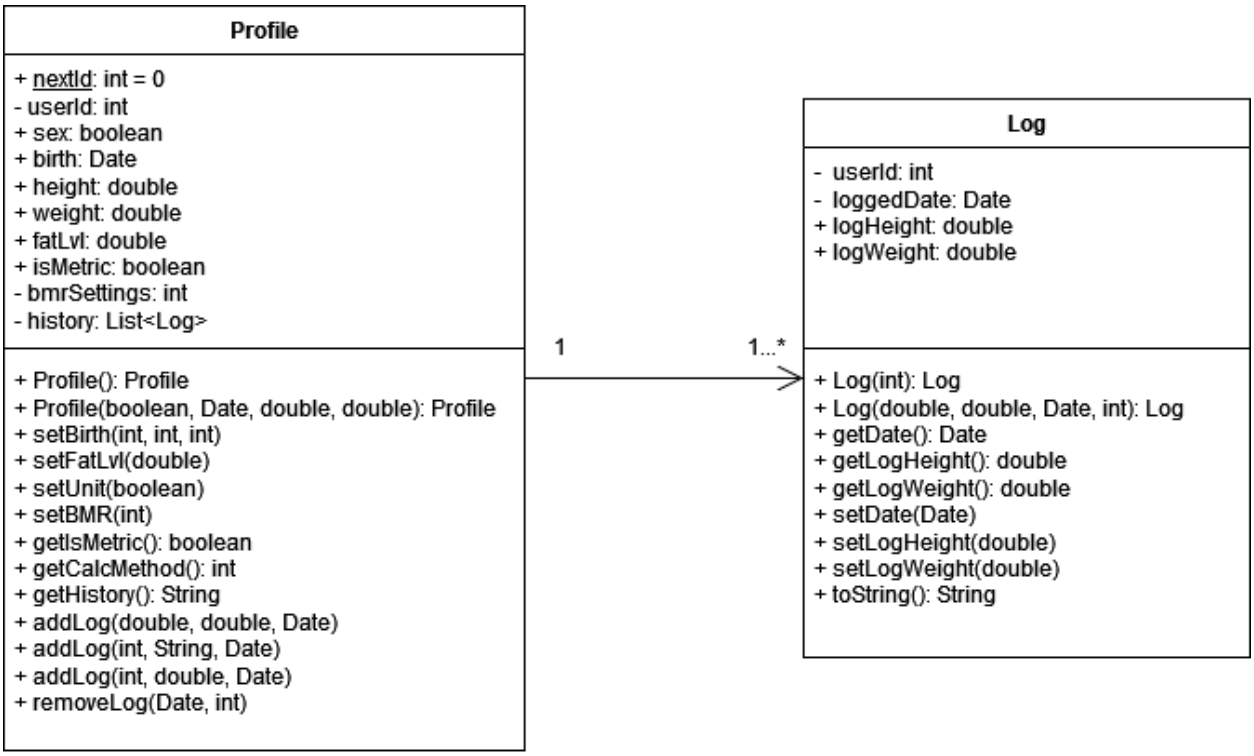


Class Diagrams and Initial Implementation:

Use case 1:

Class name	Attribute/Method name	Description
Profile	nextId: static int	For creating unique IDs for each profile instance.
	userId: int	Unique ID of a Profile instance.
	sex : boolean birth: Date height: double weight: double fatLvl: double	Fields representing the basic data each profile stores.
	isMetric: boolean bmrSettings: int	Fields representing the basic settings of each profile. Used to determine the inputs and outputs of some methods.
	history: List<Log>	Log instances associated with the profile.
	Profile Profile() Profile Profile(boolean, Date, double, double)	Constructors. Create a new profile with specific data. For the default constructor, Profile(true, null, 0.0, 0.0) is called.
	void setSex(boolean) void setBirth(Date) void setHeight(double) void setWeight(double) void setFatLvl(double) void setUnit(boolean) void setBMR(int)	Setter methods. Some might get removed if not used outside of Profile's self-calls/edits.
	boolean getSex() Date getBirth() double getHeight() double getWeight() double getFatLvl() boolean getIsMetric() int getCalcMethod()	Basic getter methods.
	String getHistory()	Returns all the logs associated with the profile as a String instance.
	void addLog(double, double, Date) void addLog(int, String, Date) void addLog(int, double, Date)	Overloaded methods for adding new logs to the profile, each is responsible for generating a specific type of log.
	Log removeLog(Date, int)	Find the first instance of Log in history that has the same Date and type specified, remove it, and return it to the user.
Log	userId: int loggedDate: Date logType: int	Fields associated with all logs. userId is currently unused (possibly to be used to store and access logs from a database structure).

	logHeight: double logWeight: double	logType is to differentiate between subclasses of Log instances (0 for Log, 1 for DataLog, 2 for MealLog, 3 for ExerciseLog) logHeight and logWeight stores the previously added changes to the user's Profile (including the current values)
	Log Log(int) Log Log(Date, int)	Basic constructors, create the basic values common to all logs. If no Date instance is provided, store a Date instance associated with the time the method was called.
	setDate(Date) setLogType(int)	Basic setter methods.
	Date getDate() int getLogType()	Basic getter methods.
	String toString()	Returns a String in the format "YY/MM/DD"
	DataLog DataLog(int) DataLog DataLog(double, double, Date, int)	Constructors. Call the superclass constructor and store the value given (if none, stores the value 0.0 for height and weight)
	setLogHeight(double) setLogWeight(double)	Basic setter methods. Only takes non-negative values. If a negative value is given, throw an InvalidAttributeException.
	double getLogHeight() double getLogWeight()	Basic getter methods.
	String toString()	Returns a String in the format "YY/MM/DD – Height, Weight"



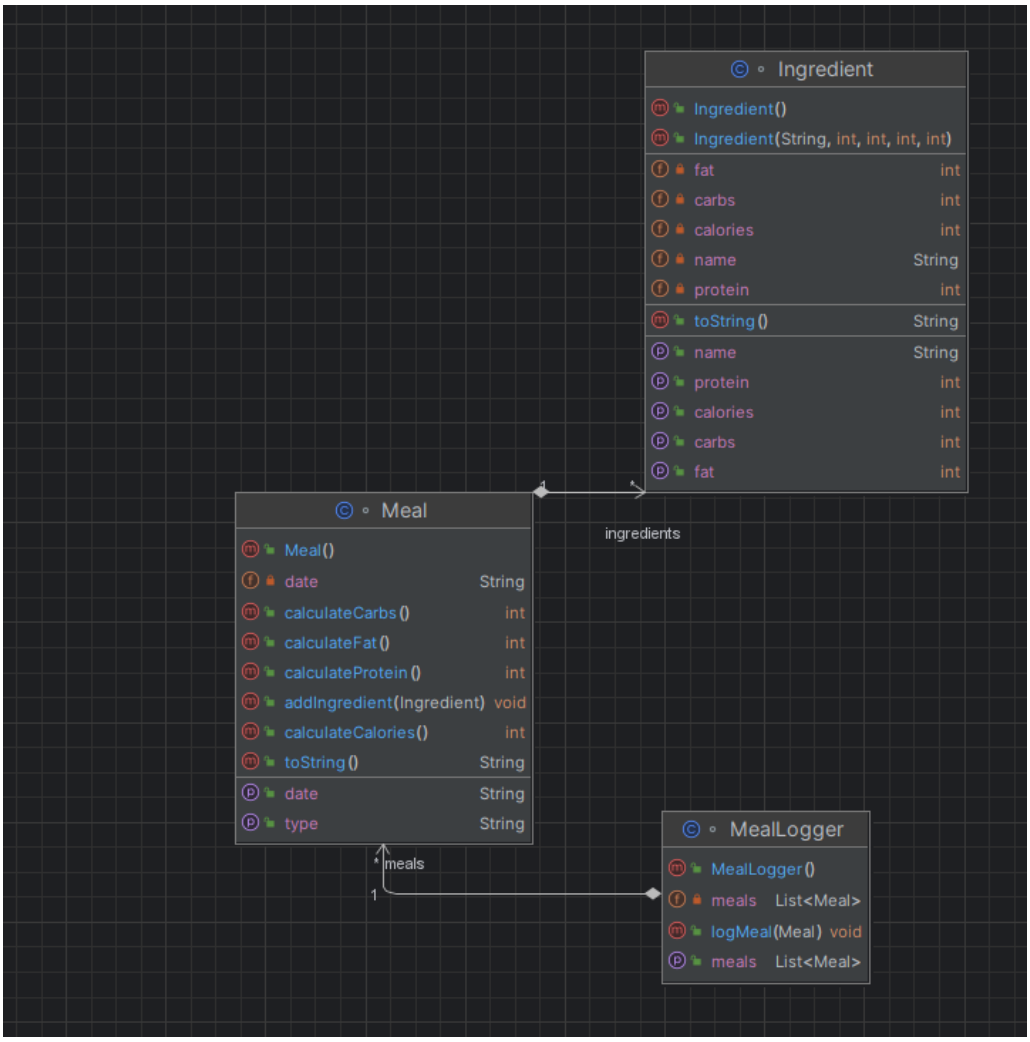
Use Case 2:

Class Name	Attribute/Method Name	Description
Meal	date: String	Holds date that meal was eaten
	mealType: String	Breakfast, lunch, dinner
	Ingredients: List<Ingredient>	Stores ingredients of meal
	<void> setType(String mealType) <void> setDate(String date)	Setters for mealType attribute and date
	<void> addIngredient(Ingredient ingredient)	Add ingredient to list
	<int> calculateProtein () <int> calculateCalories () <int> calculateFat () <int> calculateCarbs()	Calculate corresponding nutrition fact
	<String> toString()	Represent meal information in readable format

Class Name	Attribute/Method Name	Description
Ingredient	name: String	Name of ingredient
	calories: int fat: int protein: int carbs: int	Nutrient facts
	<String> getName() <String> setName()	Getter for name attribute Setter for name
	<int> getCalories() <void> setCalories(int calories) <int> getFat() <void> setFat(int fat) <int> getProtein() <void> setProtein(int protein) <int> getCarbs()	Setters and getters for nutritional information

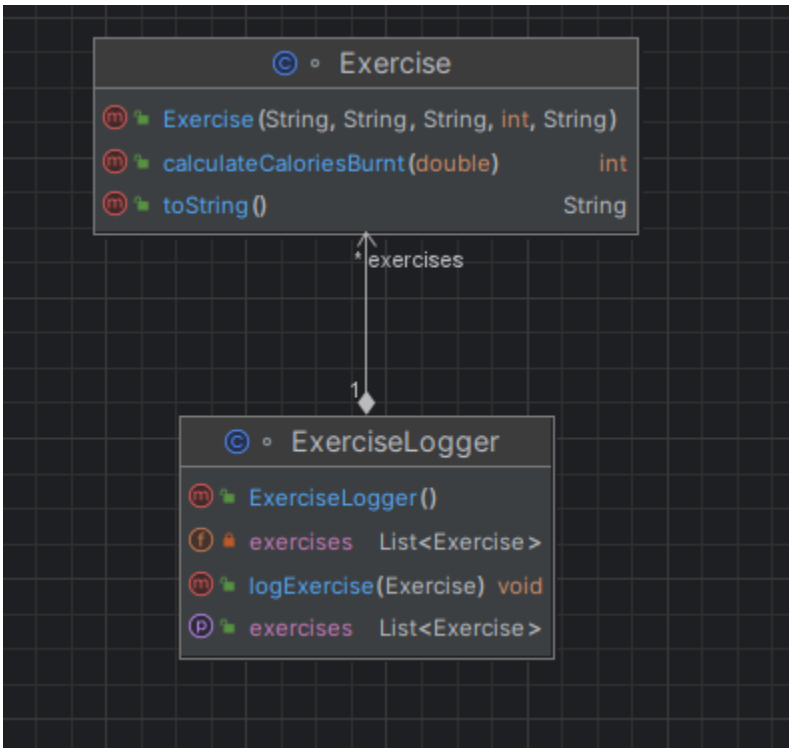
	<void> setCarbs(int carbs)	
	<String> toString()	Represent ingredient information in readable format

Class Name	Attribute/Method Name	Description
MealLogger	meals: List<Meal>	List that holds meals
	<void> logMeal (Meal meal)	Add a meal to list
	<List<Meal>> getMeals()	Return list of meals



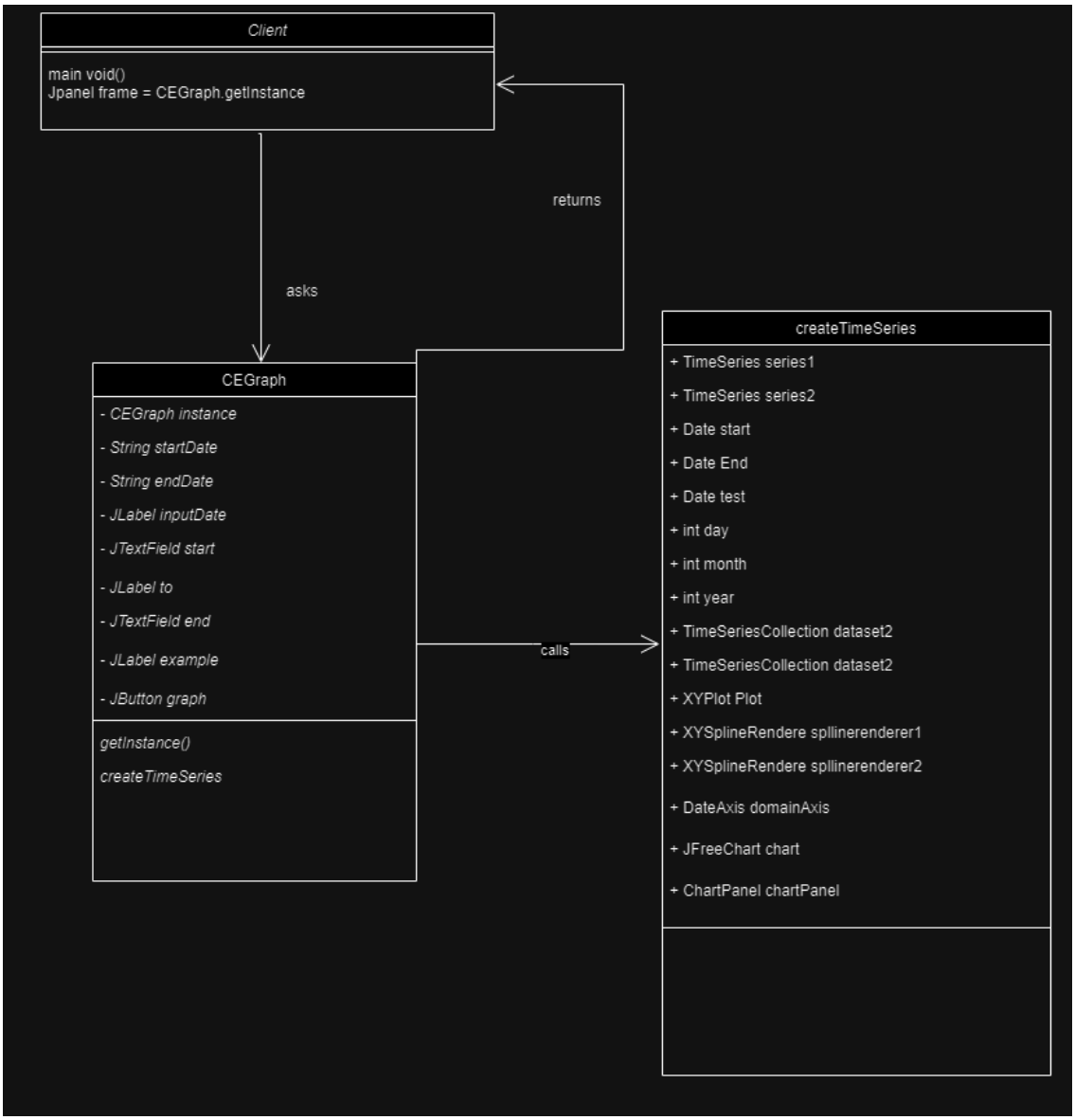
Use Case 3:

Class name	Attribute/Method name	Description
Exercise	Public Exercise(String, String, String, int, String) Public calculateCaloriesBurnt(double):int Public toString(): String	Creates an Exercise object and stores all the user exercise data. Also calculates calories burnt
ExerciseLogger	Public ExerciseLogger() Private exercises: List<Exercise> Public logExercise(Exercise): void Public exercises: List<Exercise>	Creates a list to hold Exercise objects and adds Exercises to the list per user input. Also has a method to return the list.



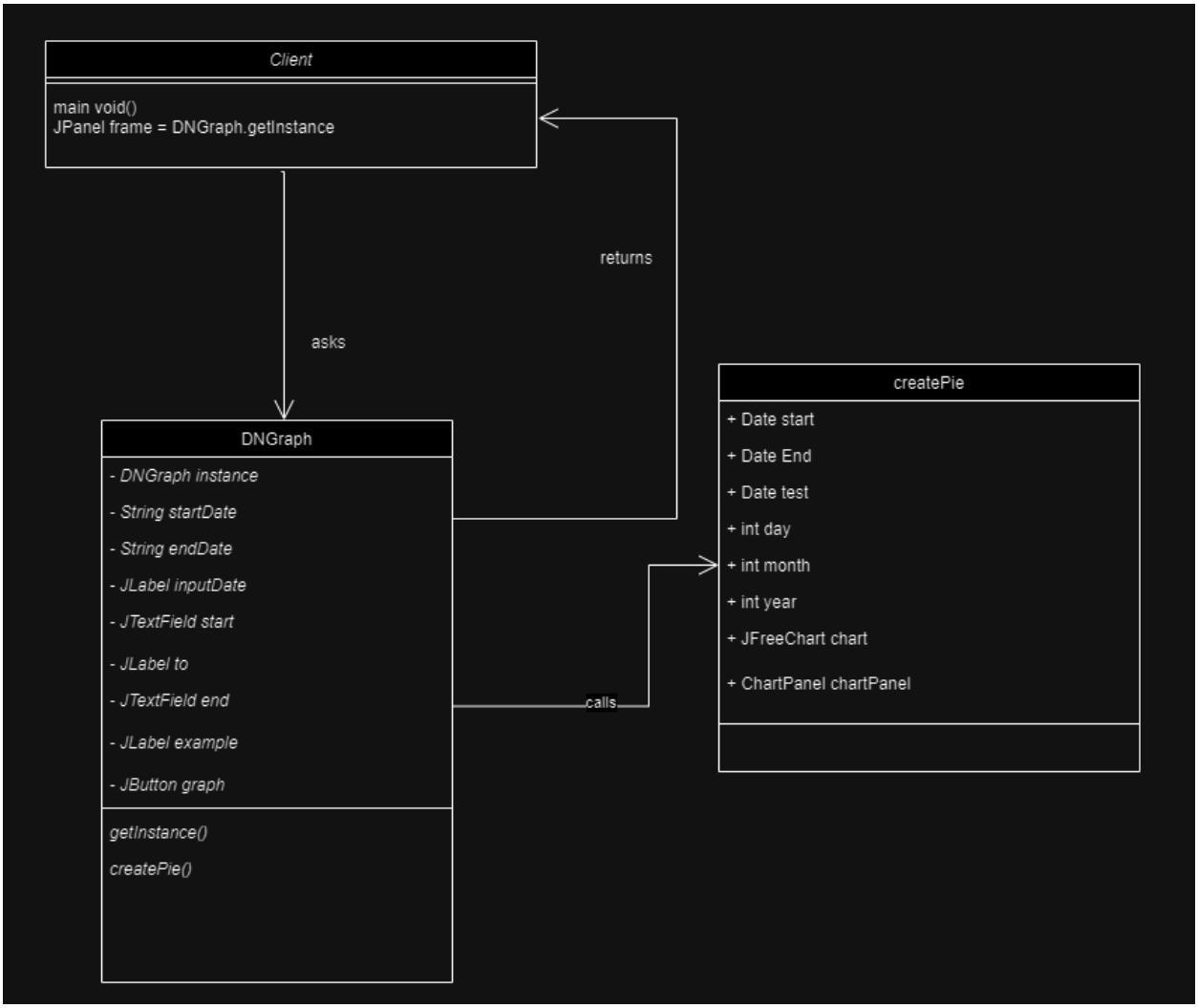
Class name	Attribute/Method name	Description
CEGraph	private static CEGraph instance private String startDate private String endDate private JLabel inputDate private JTextField start private JLabel to private JTextField end private JLabel example private JButton graph Public static getInstance() Public createTimeSeries(JPanel, startDate, endDate)	This class creates the GUI that asks the user the time period and uses another method to create the graph about their daily calory intake and daily exercise

Use case 5:



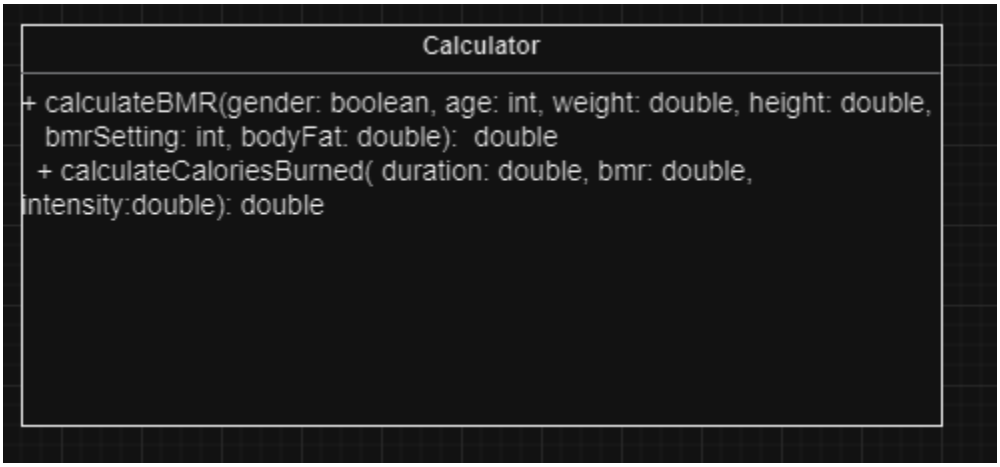
Class name	Attribute/Method name	Description
DNGraph	private static DNGraph instance private String startDate private String endDate private JLabel inputDate private JTextField start private JLabel to private JTextField end private JLabel example private JButton graph Public static getInstance() Public createPie(JPanel, startDate, endDate)	This class creates the GUI that asks the user the time period and uses another method to create the graph based on nutritional intake

Use case 6:



Calculator Class

Class name	Attribute/Method name	Description
Calculator	+calculateBMR() +calculateCaloriesBurned()	The class gets the all the variables to calculate BMR and the calories burned and then it returns that number.

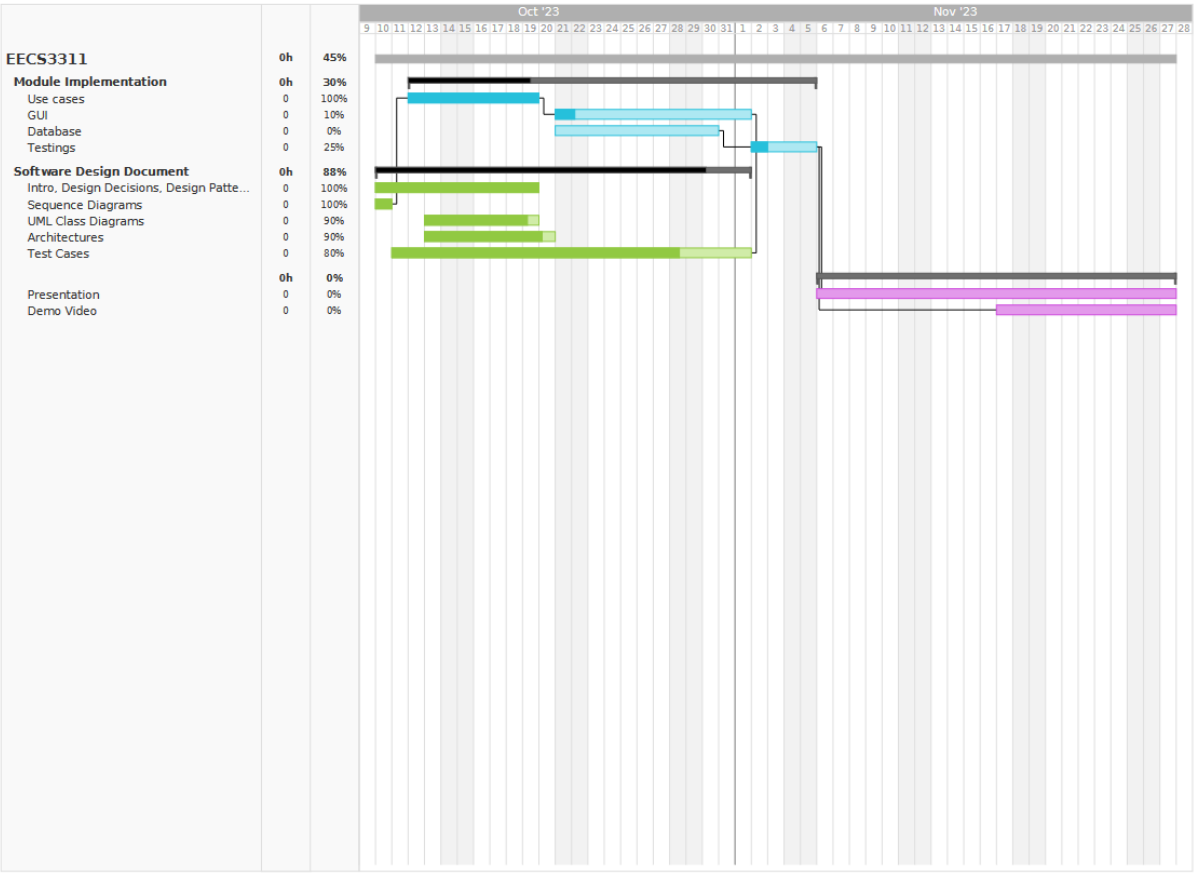


Design Patterns:

- Structural Patterns:
    - Façade: To decouple the logic modules from the view.
- Behavioral Patterns:
    - Chain of Responsibility: To deal with the possible need of requiring multiple modules to handle a request.
    - Strategy: Alternative to chain of responsibility.
    - Observer pattern: Implemented in the GUI classes through the use of action listeners on buttons, and other components.
    - Command pattern: When a button is pressed in GUI, the action listener acts as a command. The actionPerformed method is called, which encapsulates the command's logic.
- Creational Patterns:
    - Builder: To reduce the complexity of the Log constructors (to be implemented)
    - Singleton: To ensure that there is always only one connection to the database and only one Profile object loaded at any time.

Activities Plan, Product Backlog, and Sprint Backlog:

GANTT Diagrams:



Group Meeting Logs:

- Oct 10th, 6:30pm:
    - Time: 30min
    - Attendance: All
    - Discussed how to split use cases to design sequence diagrams.
    - Decisions:
        - Assignment:
            - Use case 1: Khoa Tran
            - Use case 2-3: Omer Omer
            - Use case 4-5: Alex Valdez
            - Use case 6-7: Adam Mokdad
        - Finish first draft of sequence diagrams by Oct 11th.
- Oct 11th, 6:30pm:
    - Time: 1h
    - Attendance: All
    - Showing of first draft of sequence diagrams.
    - Established the modules required for the application and their basic classes.

    - Decisions:
        - Further assignment of work:
            - Architecture Summary Table: Adam Mokdad
            - Component Diagram for architecture: Omer Omer
            - SDD – Introduction and basic structures: Khoa Tran
            - Database accessor: Khoa Tran
            - Basic test cases: Alex Valdez
        - Next meeting/progress report on Oct 13th.
- Oct 13th, 2:00pm:
    - Time: 45min
    - Attendance: All
    - Decision:
        - Major Design Decision:
            - Coding: Java
            - Database: MySQL
            - Canada Food Guide: 2007 version (more detailed)
        - Discussed the design patterns being used: Builder, Façade, Bridge
        - Next meeting/progress report on Oct 16th.
- Oct 16th, 9:00pm:
    - Time: 30min
    - Attendance: All
    - Discussed the goal for the deadline of the Deliverable 1.
    - Decision:
        - Implementations of main methods for all modules (for testing).
        - Major Design Decision:
            - MVC (to be changed)
        - Implementations of basic MET values.
        - Change in implementation of nutrient intake to better meet use case's need.
        - Final meeting before Deliverable 1 planned.
- Oct 19th, 10:00pm:
    - Time:
    - Attendance: All
    - Quick discussion final necessary components of Deliverable 1.
    - Decisions:
        - Creations of UML class diagrams specific to each module.
        - Completion of the SDD.
- Oct 31st, 7:00pm:
    - Time: 25min
    - Attendance: All
    - Discussed responsibilities for the next stretch of the project.
    - Decision:
        - GUIs temporarily split into 4 based on previous use-case assignments.
        - Exercise intensity chart: Omer Omer
        - Databases: Khoa Tran
        - Adjusting test cases to better fit requirements: Alex Valdez
- Nov 7th, 9:30pm:

    - Time: 20min
    - Attendance: All
    - Discussed required changes to implementation.
    - Quick demo of some GUI components.
    - Decision:
        - Database for profiles and logs are required.
        - Changes needed to MealLog.java to match the information stored in the given database.
        - Continue development of GUIs and databases.
- Nov 14th, 10pm:
    - Time: 20min
    - Attendance: All
    - Discussed the coupling of the modules and explained how to load and perform queries on the database.
    - Decision:
        - Continue developing and adjusting the GUIs and database queries.
        - Clarified usage of elements between modules.

Test Driven Development:

Test ID	T0001
Category	The profile creation tool works, the profile will show on the splash screen and allows for simple editing
Requirements Coverage	UC1-Successful-Data-Load
Initial Condition	System has been initiated and runs
Procedure	1. Remove all profiles from the database 2. Adds age. Input 23 3. Adds sex. Input male 4. Adds their date of birth. Input December 20th, 2005 5. Adds their height. Input 5'7ft 6. Adds their weight. Input 165 lbs 7. Change weight to 170 lbs
Expected Outcome	The users profile should be added ,with changes, and will be displayed in the splash screen User : Adam Smith Age - 23 Sex - male Date of Birth - December 20th 2005 Height - 5'7 ft Weight - 170 lbs
Notes	Assume the user inputs the right input, height, age and weight being over zero. Date of birth not being greater than the current date.

Test ID	T0002
Category	The log meal tool works and the user being able to view their meals nutrient value
Requirements Coverage	UC2-Successful-Data-Load
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information
Procedure	1. Input the date the meal was eaten. Input March 11th, 2021 2. Input whether the meal was breakfast, lunch, dinner or snack. Input snack 3. Input the food eaten. Input bread. 4. Input the amount of the food eaten, Input one slice of bread
Expected Outcome	The application must log the meal data, and the user should be able to see the food nutrient data. Food successfully logged! Snacks nutrient information Calories 32g Total fat 0.4g Saturated fat 0.1g

	Trans fat regulation 0g Cholesterol 0mg Sodium 58.9mg Potassium 13.8mg Total Carbohydrate 6g Protein 1.1g
Notes	Assume the user inputs the right input.

Test ID	T0003
Category	The log exercise tool works and the user being able to view their meals nutrient value
Requirements Coverage	UC3-Successful-Data-Load
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information
Procedure	1. Input date and time of the exercise. Input March 11th, 2021. 2. Input the type of exercise. Input running. 3. Input the duration. Input 60 mins. 4. Input the intensity. Input medium
Expected Outcome	The application must log the exercise data. Exercise successfully logged!
Notes	Assume the user inputs the right input.

Test ID	T0004
Category	BMR and calories burnt is successfully calculated
Requirements Coverage	UC3-Successful-Calculation
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information
Procedure	1. Input date and time of the exercise. Input March 11th, 2021. 2. Input the type of exercise. Input running. 3. Input the duration. Input 60 mins. 4. Input the intensity. Input medium
Expected Outcome	The application must log the exercise data. BMR and calories successfully calculated Amount of calories burnt - 600* BMR - 1600* calories *sample data not accurate, only used as example
Notes	Assume the user inputs the right input.

Test ID	T0005

Category	Graph calorie intake and exercise over a inputted time period
Requirements Coverage	UC4-Successful-Visualization
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information, their meal information and exercise information.
Procedure	1. Input start date. Input December 1st, 2021 2. Input end date. Input December 5th, 2021 3. Click graph button
Expected Outcome	Line graph created and accurately graphs the users calorie intake and exercise over December 1st to December 5th.
Notes	Assume the user inputs the right input. Start date being less than end date

Test ID	T0006
Category	Graph nutrient intake over a inputted time period
Requirements Coverage	UC5-Successful-Visualization
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information, their meal information and exercise information.
Procedure	1. Input start date. Input December 1st, 2021 2. Input end date. Input December 5th, 2021 3. Click graph button
Expected Outcome	Pie graph created and accurately graphs the users nutrient intake December 1st to December 5th.
Notes	Assume the user inputs the right input. Start date being less than end date

Test ID	T0007
Category	Display to the user the amount of weight lost from current date to inputted date
Requirements Coverage	UC6-Successful-Calculation
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information, their meal information and exercise information.
Procedure	1. Input future date. Input December 10th, 2021 2. Click calculate *Assume the current date is December 1st, 2021
Expected Outcome	Amount of weight lost is accurately calculated from current date to future date. Using the info from meal and exercise log. Amount of fat lost successfully calculated! Amount of fat lost from today to December 10th, 2021 is 8kg
Notes	Assume the user inputs the right input. Future date being greater than the current date

Test ID	T0008
Category	Display to the user how well their diet aligns with the Canadian food guide (CFG)

Requirements Coverage	UC7-Successful-Calculation
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information, their meal information and exercise information.
Procedure	1. Click button
Expected Outcome	Diet related data is successfully accessed and compared to the CFG
Notes	

Test ID	T0009
Category	Checking if the meal log application will catch when meal info is left out
Requirements Coverage	UC2-Successful-Error
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information.
Procedure	1. Input the date the meal was eaten. Input March 11th, 2021 2. Input whether the meal was breakfast, lunch, dinner or snack. Input snack 3. Input the food eaten. Input bread. 4. Purposefully leave the amount of the food blank
Expected Outcome	The application should throw an error exception and display to the user a message saying that they have left out important information The amount you have eaten is empty
Notes	

Test ID	T0010
Category	Checking if the exercise log application will catch when exercise info is left out
Requirements Coverage	UC3-Successful-Error
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information.
Procedure	1. Input date and time of the exercise. Input March 11th, 2021. 2. Input the type of exercise. Input running. 3. Input the duration. Input 60 mins. 4. Purposefully leave the intensity of the exercise blank
Expected Outcome	The application should throw an error exception and display to the user a message saying that they have left out important information The intensity of the exercise is empty
Notes	

Test ID	T0011

Category	Checking if use case five and six will catch when the user inputs an end date that is less than the start date
Requirements Coverage	UC5-Successful-Error
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information, their meal information and exercise information.
Procedure	1. Input start date. Input December 5st, 2021 2. Input end date. Input December 1th, 2021 3. Click graph button
Expected Outcome	The application should throw an error exception and display to the user that the inputted date is incorrect The inputted date does not make sense
Notes	

Test ID	T0012
Category	Checking if the meal log application will catch when user tries to input more than one breakfast/lunch/dinner
Requirements Coverage	UC2-Successful-Error
Initial Condition	System has been initiated and runs, the user has correctly logged their profile information.
Procedure	1. Input the date the meal was eaten. Input March 11th, 2021 2. Input whether the meal was breakfast, lunch, dinner or snack. Input breakfast. 3. Input the food eaten. Input bread. 4. Input the amount of the food eaten, Input one slice of bread. 5. Input the date the meal was eaten. Input March 11th, 2021 6. Input whether the meal was breakfast, lunch, dinner or snack. Input breakfast. 7. Input the food eaten. Input oatmeal. 8. Input the amount of the food eaten, Input sixty grams of oatmeal
Expected Outcome	The application should register the first meal as breakfast, but then the application should throw an error exception and display to the user that they can not input two breakfasts Food successfully logged! Snacks nutrient information Calories 32g Total fat 0.4g Saturated fat 0.1g Trans fat regulation 0g Cholesterol 0mg Sodium 58.9mg Potassium 13.8mg Total Carbohydrate 6g Protein 1.1g

	You cannot have more than one breakfast meal
Notes	