

Analysis	5
Introduction	5
Problem Identification	5
Target Audience	6
Plan For Research	6
Existing Fitness apps	6
MyFitnessPal	6
Strong app	7
Interviews	7
Experienced user	8
Inexperienced user	8
User needs	9
Success Criteria	9
Potential softwares to develop the app	11
Database entity relationship diagram	12
Design	13
Software Choice	13
IPSO chart	13
Database design	15
Data dictionary	15
SQL queries I will use	16
Plan for object-oriented programming	17
User class:	17
Plan for class methods	18
System flowcharts	21
Logging in	21
Adding meal to treeview and database	22
Searching database for workout	23
Page Navigation	23
GUI Design	24
Login	24
GUI design	24
Data Required	24
Register Menu	25
GUI design	25
Nutrition Page	26
GUI design	26
Data required	27
Calorie intake target page	27

Gui Design	27
Workout page	28
GUI design	28
Data required	29
Search meals page	30
GUI design	30
Data required	30
Algorithm designs in pseudocode	31
'Check_date_is_real'	31
'Sort' and 'merge' (Recursive merge sort)	32
'Find_workout'	33
'Enter_workout'	34
'Createaccount'	35
Technical solution	36
Implementation	65
Subroutines	65
1 hash	65
2 login	65
3 login_submit	65
4 newuser_submit	65
5 createaccount	65
6 calculate_age	65
7 check_date_is_real	65
8 calorie_target_page	66
9 inputcalstarget	66
10 nutriton_and_workout_page	66
11 nutrition_page	66
12 add_meal	66
13 bar_chart	66
14 find_meal_page	66
15 find_meal	67
16 workout_page	67
17 add_workout	67
18 find_workout	67
19 sort	67
20 merge	67
Examples of skills I have used	67
Testing	69
Test plan	69
Test results	84
Test evidence	102
Test 1	102

Test 2	103
Test 3	103
Test 4	104
Test 5	104
Test 6	105
Test 7	105
Test 8	106
Test 9	106
Test 10	106
Test 11	107
Test 12	107
Test 13	107
Test 14	108
Test 15	109
Test 16	110
Test 17	110
Test 18	111
Test 19 & 20	112
Test 21	113
Test 22	114
Test 23	115
Test 24	116
Test 25	117
Test 26	118
Test 27	119
Test 28	120
Test 29	121
Test 30	122
Test 31	123
Test 32 & 33	124
Test 34	125
Test 35	126
Test 36	126
Test 37	126
Test 38	127
Test 39	128
Test 40	129
Test 41	130
Test 42	131
Test 43	131
Test 44	132
Test 45	132

Test 46	133
Test 47	133
Test 48	134
Test 49	134
Test 50	135
Test 51	135
Test 52	136
Test 53	136
Test 54	137
Test 55	138
Test 56	139
Evaluation	140
Final product vs success criteria	140
1.1 A GUI for user to sign in and create an account	140
1.2 Different windows to separate parts of the application	140
1.3 Different tabs for nutrition and resistance training	140
2.1 User sign in feature	140
2.2 Create account option for new users	140
2.3 Log out feature	140
3.1 Allow the user to enter the exercises they have performed into a table, this data will be stored in a database	140
3.2 Allow users to search the database for any exercises they performed on a chosen date	141
4.1 Calculate what number of calories user would need to gain/lose	141
4.2 Recommend users different calorie intakes	141
5.1 Allow users to record all their meals and track those meals.	141
5.2 Record user total calorie intake over a day	141
5.3 Allow users to look back at this data and see what they ate on a specific day in the past	141
6.1 At the end of each day, the users calorie surplus/deficit will be calculated and plotted onto a graph	141
6.2 The graph will show the past 5 days, and will state how high this deficit/surplus was	142
User feedback	142
Feedback from experienced user	142
Feedback from inexperienced user	143
Analysis of feedback	143
Room for improvement	144
Allowing the user to make default workouts	144
Recommended/default workouts	144
Specifying dates on bar chart	144
Including a smaller bar chart when searching for a meal	144
Tutorial/guide upon creating an account	144

Visual appeal	144
Using usernames instead of user IDs	144
Conclusion	145

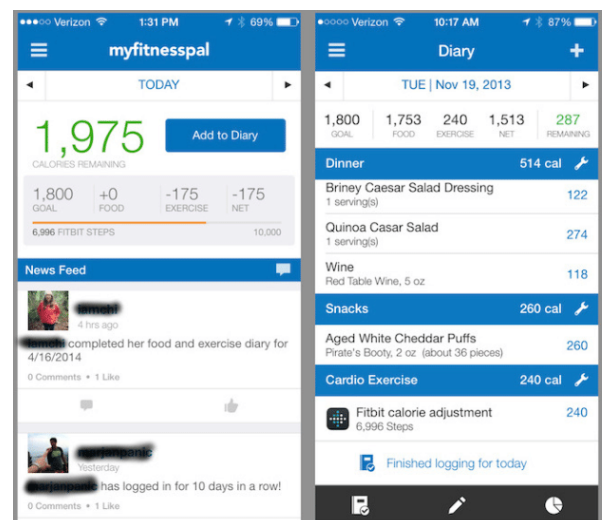
Analysis

Introduction

I am going to make a fitness tracking app for my project. The app will help users stay fit and track their weight, diet, and exercise. There will be a section to enter the meals you have eaten and how many calories they contain, a section to record exercise sessions and calculate the calories that were burned during the session, a section to track weight loss, and a section to track and record your training sessions to show progress. This project contains challenging elements which I hope will force me to expand my programming skill set, for example; the work I will have to do with databases in my project, and plotting graphs to track bodyweight and training progress.

Background To The Project

There are many pre-existing apps that track fitness and nutrition (for example; myfitnesspal), however the majority cost money and either only track nutrition and cardio or resistance training.



Problem Identification

I would like to create my own fitness tracker, which allows users to easily and effectively track nutrition, fitness, resistance training, and weight loss/gain. The importance of keeping healthy and consistently exercising is paramount, and in the modern world it becomes harder and harder to do both due to jobs requiring little physical activity, food becoming more unhealthy and popular pastimes involving less and less exercise (e.g. watching television). Many beginners also struggle with a lack of knowledge on the matter, and would benefit massively from some help. Many people attempt to track their fitness using physical pen and paper, which is very difficult. By using a computer instead, users do not have to make calculations for things such as their maintenance calorie intake themselves. I aim to develop an app which will make both getting and staying healthy much easier. To do this the app will assist users in maintaining a good diet by tracking calories consumed, and assist users in building strength and muscle by tracking resistance training.

Target Audience

The target audience for this game will be people who are beginners to fitness, and therefore do not want to spend money on an app. The app will therefore have to be accessible, and not use complex terminology which beginners will not understand. At the same time, it will still be advanced enough for people more experienced in fitness, but who do not consider the cost of paid apps to be worth it.

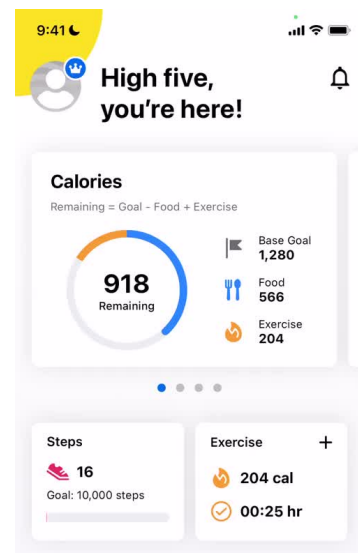
Plan For Research

I will look at fitness trackers that already exist to deduce possible features for my fitness assistant. I will look for correlations between the apps features to help me decide what features I should include. I will also do an interview with prospective users to gauge people's interests in certain features. There are a few potential softwares I can use to make my app, so I will research these options and come to a decision after finding which software is best.

Existing Fitness apps

MyFitnessPal

MyFitnessPal (shown on the right) is an incredibly popular fitness app with over 200 million users worldwide. It has an incredibly simple and attractive GUI which makes it easy to navigate the app and understand the information being shown. It allows users to track both nutrition and exercise and therefore see their net calorie intake each day. It also has a workout tracker, however this is not as detailed as other apps.

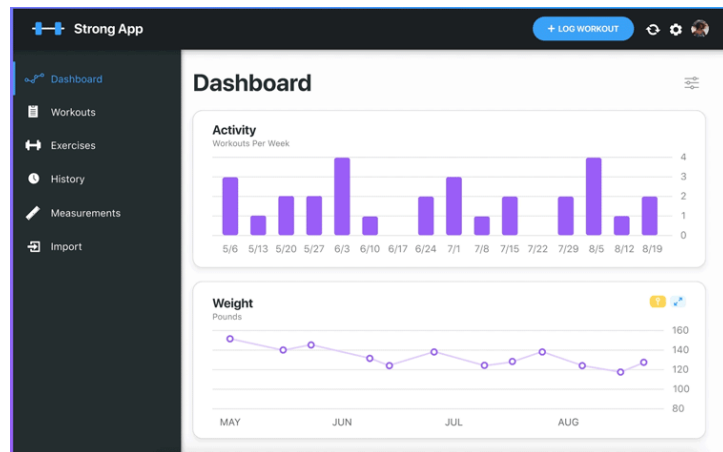


Notable Features

- Very aesthetically pleasing and simple GUI
- A circular progress bar to show progression towards both nutrition and exercise goals.
- Asks users for weight, height, activity level ect when creating an account and calculates BMI, calories burned daily ect.
- Lets users set weight loss/gain goals and recommends diets based on this goal
- Paid premium

Strong app

The strong app (shown on the right) is a very popular app tailored more towards tracking the users actual workouts. It offers incredibly detailed tracking for resistance training. However, there is very little in the way of nutrition and cardiovascular exercise.



Notable features

- Has pre-made workout routines for users. Also allows users to create their own and download ones from the internet.
- Allows users to track progress on individual lifts.
- Offers various graphs to show progress e.g. workout frequency, strength progression on different lifts.
- 1 rep max calculator
- Allows users to mark sets as warm ups, drop sets, or failures.
- Allows users to record body measurements.
- Paid premium

Common features

- Paid premium to get full experience
- Stores data so users can retroactively see progress
- Various strategies to make progress easier to see and comprehend
- Users are able to set goals
- Asks users for info such as weight, height ect upon install

Features not in common

- Strong is resistance training focused, whereas myfitnesspal is nutrition focused

Interviews

I selected two possible users to interview. One was someone very experienced in the world of research, who had spent a long time learning and researching. Another was a beginner who had never measured their exercise, but was interested in beginning.

Experienced user

Question 1:

What are the bare minimum features you would expect from a fitness assistant for both nutrition and training?

Answer:

The user would expect basic features such as:

- A place to record calories throughout the day, and track net intake
- A place to track resistance training

Question 2:

Are there any more advanced features you would like to see?

Answer:

- Store data from workouts and meals, so the user can look at them later and see progression
- Recommended nutrition plans based on the information they give (weight, height etc)
- Allow users to pick calorie intake targets

Question 3:

How important is a graphical user interface to you in this app?

Answer:

The user felt a GUI is incredibly important as it will allow them to easily use the app. They also felt including a GUI would allow for many of the features they requested such as graphs.

Inexperienced user

Question 1:

Do you know the basics of fitness, nutrition, and resistance training?

Answer:

The user had a small range of knowledge about these things.

Question 2:

Are there any features that could make the app more accessible for you?

Answer:

The user thought the app should calculate different calorie intakes for different weight loss/gain speeds for them, as they did not know how to do this themselves.

Question 3:

How important is a graphical user interface for you in this app?

Answer:

The user felt a GUI was incredibly important and should be easy to use so they can quickly gain an understanding of the app.

User needs

After doing these interviews, I have created a list of user needs that must be in the app.

- A simple, easy to use GUI
- The app should save data by default so the user can look back on progress
- The ability to track your own workouts
- The app should represent user progress in graphs to better visualise progress
- The app should let the user track nutrition and resistance training
- The graph should allow the user to select a target calorie intake for different weight loss/gain goals e.g. 0.5 kg per week

Success Criteria

Objective 1:

A simple, easy to use GUI

- 1.1 A GUI for users to sign in and create an account
- 1.2 Different windows to separate different part of the application
- 1.3 Separate tabs for nutrition and resistance training

Objective 2:

Save data so users can look back on progress

- 2.1 User sign in feature
 - Allow the user to enter credentials and log in if they are valid
 - This will be done in a separate menu
 - After this happens, account options will appear
- 2.2 Create account option for new users
 - The user can enter a new username and password and make an account
 - If the username is unique and the password meets requirements, then an account will be created
 - Upon account creation, the user will be asked to enter details to calculate maintenance calories, and pick a calorie target based on their goals
 - User info will be tracked on accounts
- 2.3 Log out feature

Objective 3:

Allow users to create and track workouts

- 3.1 Allow the user to enter the exercises they have performed into a table, this data will be stored in a database
 - The user will be able to enter in an exercise, and the sets and reps performed as well as the weight
 - Upon being submitted, this information will be displayed in a treeview and saved to the database
- 3.2 Allow users to search the database for any exercises they performed on a chosen date
 - Users will be able to enter a date into the app, after submitting, all exercises done on this date will be taken from the database and displayed in a treeview

Objective 4:

Recommend different calorie intakes for users based on their information and needs

- 4.1 Calculate what number of calories users must need to lose/gain weight
 - Upon making an account, users will be asked to, input height, age and weight
 - Based on this information, the app will calculate the users maintenance calories
- 4.2 Recommend users calorie intakes
 - The user will be offered different calories to lose/gain weight at different speeds

Objective 5:

Allow the user to track how many calories they eat each day

- 5.1 Allow users to record all their meals and calories in those meals
 - Users will be able to enter a meal, the time it was eaten, and the calories it contained into the app
 - Upon being entered, this information will be displayed in a treeview and added to a database
- 5.2 Record users total calorie intake over a day
 - When the user enters a meal, the calories in it will be added to a calorie counter
- 5.3 Allow users to look back at this data and see what they ate on a specific day in the past
 - Users will be able to enter a date into the app, all meals entered into the app on this date will be pulled from the database and added to a treeview

Objective 6:

Plot the users calorie deficits and surpluses onto a graph to help visualise their progress

- 6.1 At the end of each day, the users calorie surplus/deficit will be calculated and plotted onto a graph
 - The graph will be a bar chart
 - There will be a central axis to show the users maintenance intake, and bars will be above or below this depending on whether they were more or less than in
- 6.2 The graph will show the past 5 days, and will state how high this deficit/surplus was

Potential softwares to develop the app

Python

Python is a very popular programming language which I myself have some experience with. It is known for being very easy to learn and understand.

Python is an interpreted language which means when an error is detected, it halts the process and allows the user to deal with it, rather than continuing and showing all errors in the code at once. This makes it much easier to debug than compiled language, and is especially helpful to me as I am relatively new to programming and therefore may have to do lots of trial and error.

Furthermore, python has a vast amount of libraries I can use to make developing my app easier. For example, Tkinter can be used to easily create GUIs, which do not require live input, and SQLite can be used to query databases without the need for a server.

Python also has a large collection of guides and tutorials on the internet due to its popularity, which I can use to help me develop this app as I will need to use many techniques and commands which I have not used before.

Tkinter features:

- Creates graphic output
- Can be imported without any downloads
- Huge selection of widgets including:
 - Buttons
 - Labels
 - Combo boxes and dropdowns
 - Entry boxes
 - Scroll bars
 - Tabs
- Very easy to use with grid layouts

- Event based, the user has to click on things
- Popular - many guides and tutorials exist

Pygame features:

- Creates graphic output
- Can be imported without any downloads
- Has sound libraries
- Live input, will update without the user
- Popular - many guides and tutorials exist

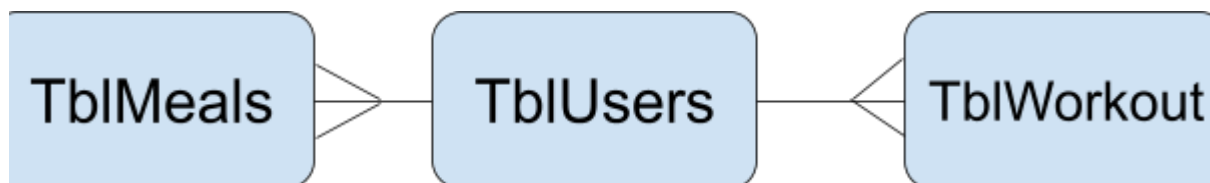
SQLite features:

- Allows the user to use language SQL to create and query databases, this language is quite easy to use
- Does not require a server
- Can be imported with no downloads
- Can be used in combination with DB browser to manage databases very easily

DB browser for SQLite:

DB browser allows you to see your SQLite databases as tables, rather than just typing SQL code to query and create them. This makes coding with SQL much much easier, which will be incredibly helpful for me as I lack any experience with SQL.

Database entity relationship diagram



Design

Software Choice

I have chosen to use python, Tkinter, and SQLite alongside DB browser to develop my application.

I have chosen python as I have an abundance of experience using it. I also have experience with Tkinter, which is the most appropriate choice for this app as it does not require live input and contains all the features I will need while developing the app (buttons, treeviews, combo and entry boxes etc). Being a portable language is a big advantage for this application, as having it on a mobile device would allow users to track resistance training in the gym, and not have to go to a computer when they want to use the app.

I will use SQLite to create and manage my database which will contain every user's information. I will use DB browser with SQLite to better visualise my database and to save time creating it.

IPSO chart

Input	Process	Storage	Output
Valid user ID and password Login button pressed	Password is hashed. Database is searched for accounts with matching user IDs and hashed passwords, the user is logged in and all data about the user is returned	Users data is pulled from TblUsers in database All of the users data is stored as attributes in the user_logged_in_object	Nutrition page opens, login page closes
User's details are entered Submit button pressed	All details are checked to see if they are valid. New record is created in database, password is hashed, all information is entered	All data is entered into and stored in database All data is added to user_logged_in object as attributes	New user page closes, calorie target page opens

One of the calorie target buttons is pressed	Calorie target is entered into database	Selected target is stored in database and stored as an attribute in user_logged_in class	Calorie target page closes, nutrition page opens
Meal details are entered, add meal button is pressed	Checks if time and calories are numeric, adds info to database and treeview	Information about the meal is stored in the database and the treeview	Meal appears in treeview
Search meals button is pressed	Find meals page is loaded		Find meals window opens on screen
Date is entered into find meals page Search button pressed	Date is checked to make sure it's real. Database is searched for entries with matching dates, entries on entered date are inserted into treeview	Storage is searched for entries with matching dates	All meals entered on date appear in treeview
Calorie target button pressed	Target calories window is opened, calorie surplus and deficits are all calculated	Maintenance calories is found from user_logged_in object	Target calories page opens
Workout tracker tab pressed	Treeviews, labels, buttons and entries are all loaded into workout tracker frame	Any workouts entered today are pulled from database and inserted into treeview	Workout page opens
Workout details entered into workout page Add button pressed	Data taken from entries and entered into treeview and database	New record is made in TblWorkout, and data entered by user is inserted	Data is entered into treeview and entries are cleared
Date entered into find workout section Search button pressed	Database searched for entries with matching dates Results sorted by sets	TblWorkout searched for entries with matching date, matching entries pulled from database	All entries entered on date entered by user are displayed in treeview, sorted by sets performed

	Results displayed in treeview		
--	-------------------------------	--	--

Database design

Data dictionary

Table	Field	Data type	Validation	Example
TblUsers	userID	INTEGER	Not null, Auto increments, Unique, Primary Key	1
TblUsers	firstname	TEXT	Not null	Benjamin
TblUsers	lastname	TEXT	Not null	Franklin
TblUsers	weight	INTEGER	Not null	50
TblUsers	height	INTEGER	Not null	187
TblUsers	password	TEXT	Not Null	Ge0rg3Washy32
TblUsers	gender	TEXT	Not null	Male
TblUsers	yearborn	INTEGER	Not null	1706
TblUsers	monthborn	INTEGER	Not null	1
TblUsers	dayborn	INTEGER	Not null	17
TblUsers	age	INTEGER		19
TblUsers	targetintake	INTEGER		2700
TblUsers	exerciselvl	INTEGER		3
TblUsers	calorieseaten	INTEGER		0
TblMeals	userID	INTEGER	Not null	3
TblMeals	mealID	INTEGER	Not null, primary key, auto increment, unique	1
TblMeals	mealdesc	TEXT		pasta

TblMeals	dateeaten	TEXT	Not null	2023-02-24
TblMeals	timeeaten	TEXT		04:25
TblMeals	calories	INTEGER	Not null	1200
TblWorkout	userID	INTEGER	Not null	3
TblWorkout	workoutID	INTEGER	Not null, primary key, unique, auto increment	2
TblWorkout	date	TEXT	Not null	2023-04-26
TblWorkout	exercise	TEXT	Not null	Bench press
TblWorkout	sets	INTEGER	Not null	3
TblWorkout	reps	INTEGER	Not null	10
TblWorkout	weight	TEXT	Not null	60kg
TblWorkout	comments	TEXT		Pain in tendon

SQL queries I will use

This query will be used to search TblUsers for records with matching User IDs and passwords, and take all data from these records when someone logs in

```
SELECT * FROM TblUsers WHERE userID = "(userID entered by user)" AND Password =
"(password entered by user)"
```

This query will update the users age when they log in, incase they have turned a year older

```
UPDATE TblUsers SET age = ("users age, which is calculated using date of birth") WHERE
userID = ("current users userID")
```

This query will be used to create a new account by adding a new record into TblUsers

```
"INSERT INTO TblUsers (firstname, lastname, gender, weight, height, yearborn, monthborn,
dayborn, password, exerciselvl) VALUES ('%s', '%s', '%s', %s, %s, %s, %s, %s, '%s', %s)"
% (first name entered by user, last name entered by user, users gender, users weight, users
height, year user was born, month user was born, day user was born, password entered by
user, users exercise level)
```

This query will find a user's userID after they create an account by finding the last added userID

```
SELECT MAX(userID) FROM TblUsers
```


This query will set a users target calorie intake

```
UPDATE TblUsers SET targetintake = ("Intake selected by user") WHERE userID = ("current users userID")
```

This query will find all meals eaten by a user today

```
SELECT * FROM TblMeals WHERE dateeaten = ("todays date") and userID = ("current users userID")
```

This query will find the sum of all calories in all meals eaten today

```
SELECT SUM(calories) FROM TblMeals WHERE dateeaten = ("todays date") AND userID = ("current users userID")
```

This statement will add a meal to TblMeals

```
INSERT INTO TblMeals (UserID, Mealdesc, Dateeaten, Timeeaten, Calories) VALUES (%s, '%s', '%s', '%s', %s)" % (user_logged_in.get_id(), mealbox.get(), today_formatted_as_string, time_eaten, calseatenbox.get())
```

This query will add a workout to TblWorkout

```
INSERT INTO TblWorkout (UserID, date, exercise, sets, reps, weight, comments) VALUES (%s, '%s', '%s', %s, %s, '%s', '%s')" % (user_logged_in.get_id(), today_formatted_as_string, exercise_box.get(), sets_box.get(), reps_box.get(), weight_box.get(), comments_box.get())
```

This query will select all exercises done on a given day from TblWorkouts

```
SELECT * FROM TblWorkout WHERE userID = ("current users UserID") AND date = ("Date entered by user")
```

Plan for object-oriented programming

User class:

This class will hold information about the user who is logged in. When a user logs in, all necessary information about the user will be pulled for the database and put into the attributes of an object from this class, and there will be various functions allowing this data to be found or changed. This will allow information about the user logged in (e.g. their) age to be found much more easily than querying the database.

Plan for class methods

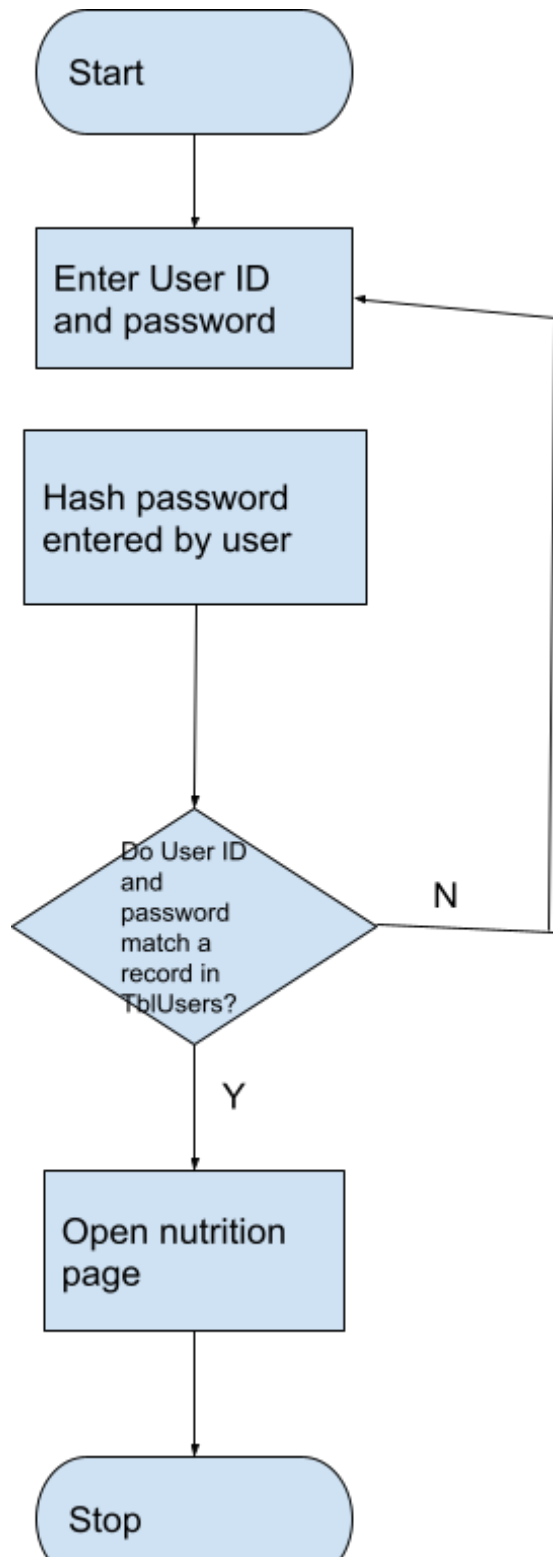
<u>Class</u>	<u>Method</u>	<u>Description</u>	<u>Parameters</u>	<u>Return values</u>
User	get_id	Returns user ID of the user logged in		id
User	set_id	Sets user ID for user logged in	id	
User	get_firstname	Returns first name of user logged in		firstname
User	set_firstname	Sets first name for user logged in	firstname	
User	get_lastname	Returns last name of user logged in		lastname
User	set_lastname	Sets last name for user logged in	lastname	
User	get_weight	Returns weight of user logged in		weight
User	set_weight	Sets weight of user logged in	weight	
User	get_height	Returns height of user logged in		height
User	set_height	Sets height of user logged in	height	
User	get_gender	Returns gender of user logged in		gender
User	set_gender	Sets gender of user logged in	gender	
User	get_yearborn	Returns year born of user		yearborn

		logged in		
User	set_yearborn	Sets year born of user logged in	yearborn	
User	get_monthborn	Returns what month the user logged in was born in		monthborn
User	set_monthborn	Sets month born of user logged in	monthborn	
User	get_dayborn	Returns the day the user logged in was born		dayborn
User	set_dayborn	Sets the day the user logged in was born	dayborn	
User	get_age	Returns age of the user logged in		age
User	set_age	Sets age of the user logged in	age	
User	get_password	Returns password of the user logged in		password
User	set_password	Sets password of the user logged in	password	
User	get_target_intake	Returns target calorie intake of user logged in		target_intake
User	set_target_intake	Sets target intake of user logged in	target_intake	
User	get_exercise_level	Returns activity level of user logged in		exercise_level
User	set_exercise_level	Sets exercise level of users	exercise_level	

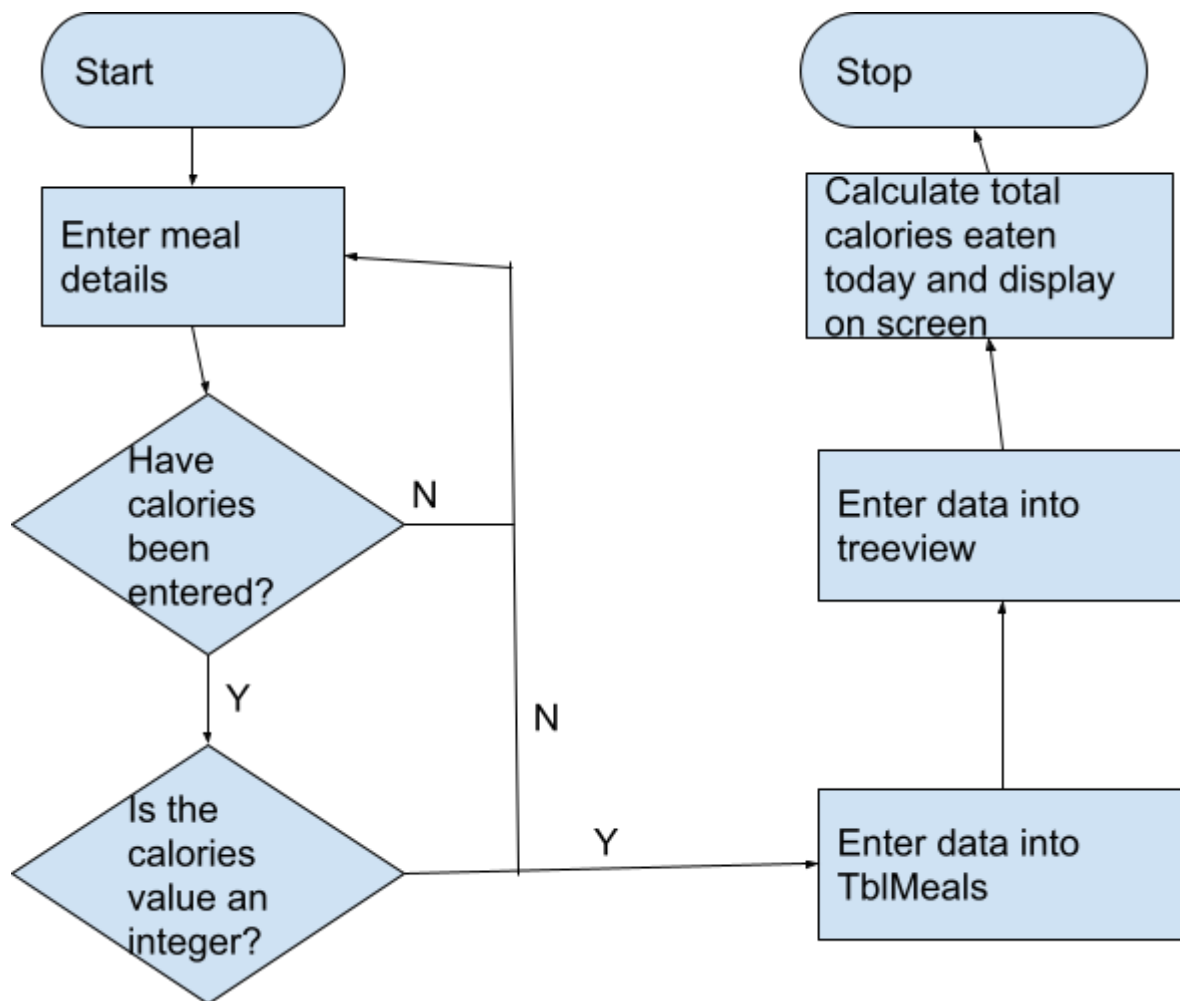
		logged in		
--	--	-----------	--	--

System flowcharts

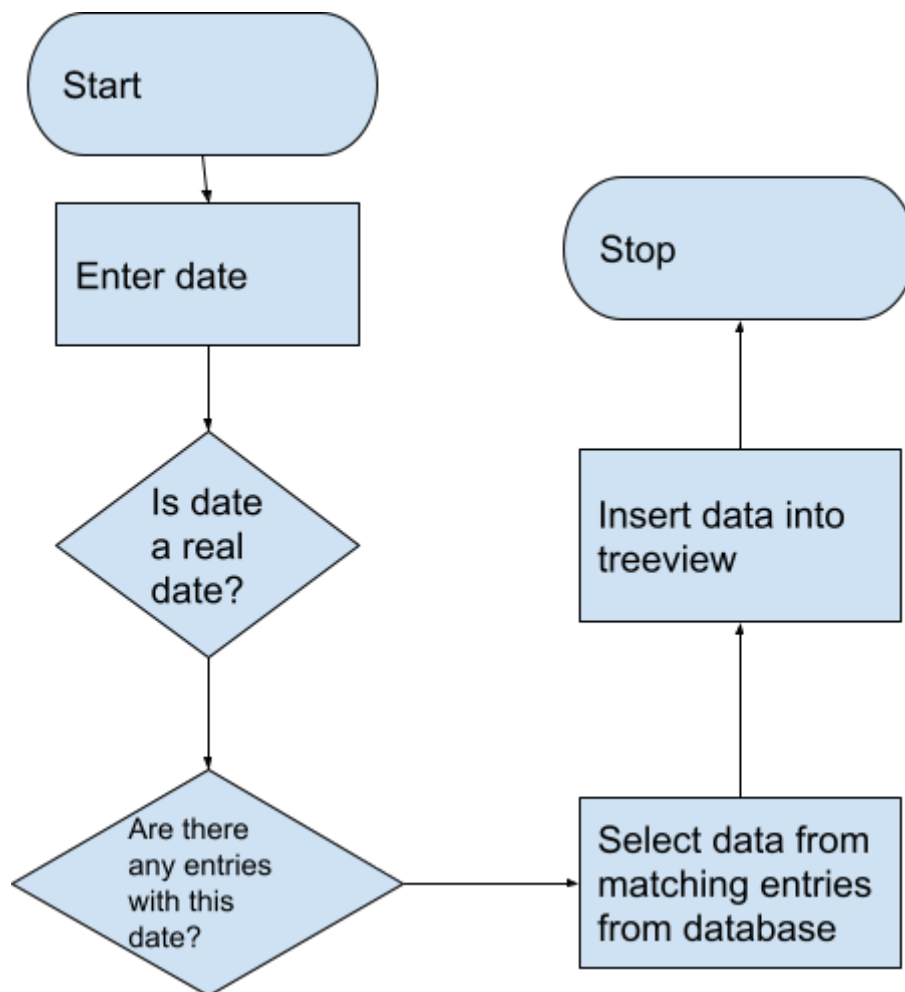
Logging in



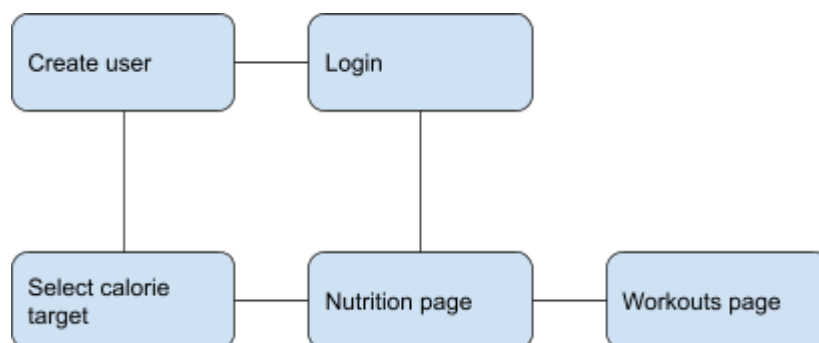
Adding meal to treeview and database



Searching database for workout



Page Navigation



GUI Design

Login

GUI design

Login Menu
User ID:
Password

Data Required

<u>Data Item</u>	<u>Data Type</u>	<u>Validation / Restrictions</u>
User ID	Integer	Must be a number, and there must be an account which has this User ID
Password	Text	Must match the password of the account with the user ID entered

Register Menu

GUI design

New User

New Password:

First name:

Last name:

Gender:

↓

Date of Birth:

Height (cm):

Weight (kg):

Exercise level:

↓

Submit

Data required

<u>Data Item</u>	<u>Data Type</u>	<u>Validation / Restrictions</u>
First Name	Text	Cannot be blank
Last Name	Text	Cannot be blank
Password	Text	Cannot be blank
Gender	Text	Must be one of either male of female so that necessary data can be gathered
Date of Birth	Text	Cannot be blank
Height	Integer	Must be an integer

Weight	Integer	Must be an integer
Exercise level	Text	Must be one of 4 options, user cannot type in this box

Nutrition Page

GUI design

Calories consumed:

None

Meal	Time	Calories

Add a meal:

Meal: Time: Calories:

Maintenance calories

Search meals

Data required

<u>Data Item</u>	<u>Data Type</u>	<u>Validation / Restrictions</u>
Meal	Text	No restrictions
Time (minutes)	Integer	Must be an integer
Time (hours)	Integer	Must be an integer
Calories	Integer	Must be an integer

Calorie intake target page

Gui Design

Select Calorie Intake

Rapid weight Loss (-0.5kg Per week)	Mild weight Loss (-0.25kg Per week)	Maintain weight	Rapid weight gain(+0.25kg Per week)	Rapid weight Gain (+0.5kg Per week)
2000	2250	2500	2750	3000

Workout page

GUI design

Nutrition Tracker		Workout Tracker		
Exercise	Sets	Reps	Weight	Comments
Add exercise:				
Exercise	Sets	Reps	Weight	Comments
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Add"/>				
Find a workout:				
Day	Month	Year		
<input type="text"/>	<input type="text"/>	<input type="text"/>		
Exercise	Sets	Reps	Weight	Comments

Data required

<u>Data Item</u>	<u>Data Type</u>	<u>Validation / Restrictions</u>
Day	Integer	Must be less than the maximum number of days in given month (e.g. if month entered is "1", then day can't be higher than 31), must be an integer
Month	Integer	Must be an integer between 1 and 12
Year	Integer	Must be an integer
Exercise	Text	Cannot be blank
Sets	Integer	Must be an integer
Reps	Integer	Must be an integer
Weight	Text	Cannot be blank
Comments	Text	No requirements, can be left blank

Search meals page

GUI design

Meal	Time	Calories

Day

Month

Year

Search

Data required

<u>Data item</u>	<u>Data type</u>	<u>Validations/restrictions</u>
Day	Integer	Must be a number, cannot be blank
Month	Integer	Must be a number, cannot be blank
Year	Integer	Must be a number, cannot be blank

Algorithm designs in pseudocode

‘Check_date_is_real’

This algorithm checks if a date entered by a user is a real, valid date. For example, if a user enters 30/2/2023, this date is not valid, as January (the second month) has a maximum of 29 days.

If day, month and year are numeric:

 If month > 12 or month = 0:
 Return false

 If month = 1, 3, 5, 7, 8, 10, or 12:
 If day < 32 and day ≥ 1:
 Return true
 Else:
 Return false

 If month = 4, 6, 9, or 11:
 If day < 31 and day ≥ 1:
 Return false
 Else:
 Return true

 If year % 4 = 0 and (year % 100 ≠ 0 or year % 400 = 0):
 If day < 30 and day ≥ 1:
 Return true
 Else:
 Return false

```

Else:
    If day < 29 and day ≥ 1:
        Return true
    Else:
        Return false

```

```

Else:
    Return false

```

‘Sort’ and ‘merge’ (Recursive merge sort)

This algorithm will sort a list of lists by a specific index in each list

```

Def sort(data, index):
    If data length ≤ 1:
        Return data

    Left_array = left half of array
    Right_array = right half of array

    Right_array_sorted = sort(right_array)
    Left_array_sorted = sort(left_array)

    Return merge(left_array_sorted, right_array_sorted, index)

#a separate merge subroutine is needed as this algorithm sorts a list of lists
Def merge(left_array, right_array, index)
    Create result list
    Left_index and right_index = 0

    While left_index < length of left_array and right_index < length of right array:
        If left_array[left_index] [index] ≤ right_array[right_index] [index]:
            add left_array[left_index] to result list
            Add 1 to left_index
        Else:
            Add right_array[right_index] to result list
            Add 1 to right_index

    Join result and left_array[left_index:]
    Join result and right_array[right_index:]

    Return result

```


‘Find_workout’

This algorithm searches TblWorkout for exercises entered on a specified date

Def find_workout(day, month, year):

 If day length = 1:
 day = “0” + day

 If month length = 1:
 Month = “0” + month

 Date = year - month - day

 If date is real:

 Select all meals in TblMeals entered by current user where date entered =
date

 Matching_meals = result of SQL query

 If matching_meals length = 0:
 Display error box saying “No meals found on this date”

 Else:
 matching _meals = sort(matching_meals, time eaten)
 Matching_meals.reverse()

 For matching_meals length:
 Add matching_meals to old_meal_record treeview

 Else:
 Display error box saying “Please enter a valid date”

'Enter_workout'

This subroutine enters an exercise into TblWorkout and the treeview on the workout page

```
Def enter_workout(workout_record)
```

```
  If exercise or weight or sets or reps = "":
```

```
    Show error message saying box is empty
```

```
  If sets and reps isnumeric:
```

```
    Insert exercise, sets, reps, weight, and comments into workout_record
```

```
    Insert userID, todays date, exercise, sets, reps, weight, and comments into  
    TblWorkout
```

```
    Clear all entry boxes
```

```
  Else:
```

```
    Show error box saying "Make sure sets and reps are numbers!"
```

'Createaccount'

Def createaccount(newuser_window, password, firstname, lastname, gender, dayborn, monthborn, yearborn, height, weight, exerciselvl):

 If password, firstname, lastname, gender, dayborn, monthborn, yearborn, height, weight or exerciselvl == ""

 Show error message asking user to enter field

 Elif date is not a real date:

 Show error message asking user to enter a valid date of birth

 elif height or weight is not numeric:

 Show error message asking user to enter a valid height/weight

 Else:

 Create new record in tblusers, and insert all details entered by the user

 Select highest user ID from Tblusers

 Set all user_logged_in attributes equal to details entered by user, set user ID attribute equal to highest userID in TblUsers

 Destroy newuser_window

 calorie_target_page()

Technical solution

```
from tkinter import * #import tkinter to make GUI
from tkinter import ttk #import ttk tkinter library to get extra features
from tkinter import messagebox #import messageboxes module to create message boxes so
users can be informed about errors
import datetime #import datetime module to find current date and time
import sqlite3 #import sqlite3 so database can be accessed and queried
from datetime import timedelta #import timedelta so I can subtract days from today's date
```

#user class, allows object to be made holding all of a user's details, which can be retrieved and modified. This stops me having to query the database every time I want user information
class User: #create User class to store info about users

```
    def __init__(self, id, firstname, lastname, weight, height, gender, yearborn, monthborn,
dayborn, password, target_intake, exercise_level, calories_eaten): #initialise class and set
attributes
```

```
    #set all attributes equal to different variables
```

```
    self.__id = id
    self.__firstname = firstname
    self.__lastname = lastname
    self.__weight = weight
    self.__height = height
    self.__gender = gender
    self.__yearborn = yearborn
    self.__monthborn = monthborn
    self.__dayborn = dayborn
    self.__password = password
    self.__target_intake = target_intake
    self.__exercise_level = exercise_level
    self.__calories_eaten = calories_eaten
```

```
    def get_id(self): #create function to return user's ID
        return self.__id
```

```
    def get_firstname(self): #create function to return user's first name
        return self.__firstname
```

```
    def get_lastname(self): #create function to return user's last name
        return self.__lastname
```

```
    def get_weight(self): #create function to return user's weight
```

```

    return self.__weight

def get_height(self): #create function to return user's height
    return self.__height

def get_gender(self): #create function to return user's gender
    return self.__gender

def get_yearborn(self): #create function to return the year a user was born
    return self.__yearborn

def get_monthborn(self): #create function to return the month a user was born
    return self.__monthborn

def get_dayborn(self): #create function to return the day a user was born
    return self.__dayborn

def get_age(self): #create function to return user's age
    return self.__age

def get_password(self): #create function to return user's password
    return self.__password

def get_target_intake(self): #create function to return user's target intake
    return self.__target_intake

def get_exercise_level(self): #create function to return user's exercise level
    return self.__exercise_level

def set_id(self, id): #create function to set user's ID
    self.__id = id

def set_firstname(self, firstname): #create function to set user's first name
    self.__firstname = firstname

def set_lastname(self, lastname): #create function to set user's last name
    self.__lastname = lastname

def set_weight(self, weight): #create function to set user's weight
    self.__weight = weight

def set_height(self, height): #create function to set user's height
    self.__height = height

def set_password(self, password): #create function to set user's password
    self.__password = password

```

```

def set_gender(self, gender): #create function to set user's gender
    self.__gender = gender

def set_dayborn(self, dayborn): #create function to set user's day born
    self.__dayborn = dayborn

def set_monthborn(self, monthborn): #create function to set user's month born
    self.__monthborn = monthborn

def set_yearborn(self, yearborn): #create function to set user's year born
    self.__yearborn = yearborn

def set_age(self, age): #create function to set user's age
    self.__age = age

def set_target_intake(self, target_intake): #create function to set user's target calorie intake
    self.__target_intake = target_intake

def set_exercise_level(self, exercise_level): #create function to set user's exercise level
    self.__exercise_level = exercise_level

user_logged_in = User(0,"", "",0,0, "", 0, 0, 0, "", 0, 0, 0) #create user_logged_in object to
store details of user logged in, all attributes are currently set to 0 or " " as no user is logged
in at the start of the program


database_connection = sqlite3.connect('fitness_assistant.db') #connect database to code
database_cursor = database_connection.cursor()

#create iid variables which automatically goes up every time something is added to treeview,
so that things can be added.
iidnumb = 0
iidnumb2 = 0
iidnumb3 = 0
iidnumb4 = 0

def hash(password):
    unhashed_password = "" #create empty hashed password variable to put hashed password
into
    for letter in password:
        unhashed_password = unhashed_password + str(ord(letter)) #go through every letter in
password and turn it into unicode value, append this value to the unshahed_password string

```

```

password_int = int(unhashed_password) #turn password into an integer so that I can use
mod function on it
hashed_password = str(password_int % 118147) #mod value by prime number, remainder
is the hash value
return hashed_password

```

```

#login window
def login():
    login_window = Tk() #create login window
    login_window.title("Login") #set window title
    login_window.geometry("300x100") #set window dimensions
    login_window.configure(bg="white") #set window background

    #create "login" label so user knows they are in login page
    title_label = Label(login_window, text="Login", bg="white")
    title_label.grid(row=0, column=0, padx=30, columnspan=2)

    #create user ID entry
    user_label = Label(login_window, text="User ID:", bg="white")
    user_label.grid(row=1, column=0, padx=30)
    user_entry = Entry(login_window, width=15)
    user_entry.grid(row=1, column=1)

    #create password entry
    password_label = Label(login_window, text="Password:", bg="white", )
    password_label.grid(row=2, column=0, padx=30)
    password_entry = Entry(login_window, width=15, show="*")
    password_entry.grid(row=2, column=1)

    #create button to submit user ID and password
    submit_button = Button(login_window, text="Submit", command=lambda:
login_submit(login_window, user_entry.get(), password_entry.get()))
    submit_button.grid(row=3, column=1)

    #create button to create a new account
    new_user_button = Button(login_window, text="New User", command=lambda:
newuser_submit(login_window))
    new_user_button.grid(row=3, column=0)

    login_window.mainloop()

#create account window
def newuser_submit(login_window):

```

```

login_window.destroy() #close login window
newuser_window = Tk() #create window
newuser_window.title("Create Account") #set window title
newuser_window.geometry("380x215") #set window dimensions
newuser_window.configure(bg="white") #set window background

newuser_label = Label(newuser_window, text="New User", bg="white")
newuser_label.grid(row=0, column=0, columnspan=4, padx=50)

#create password
password_label = Label(newuser_window, text="New Password:", bg="white")
password_label.grid(row=1, column=0, padx=30)
password_entry = Entry(newuser_window, width=27)
password_entry.grid(row=1, column=1, columnspan=3)

#enter first name
firstname_label = Label(newuser_window, text="First Name:", bg="white")
firstname_label.grid(row=2, column=0, padx=30)
firstname_entry = Entry(newuser_window, width=27)
firstname_entry.grid(row=2, column=1, columnspan=3)

#enter last name
lastname_label = Label(newuser_window, text="Last Name:", bg="white")
lastname_label.grid(row=3, column=0, padx=30)
lastname_entry = Entry(newuser_window, width=27)
lastname_entry.grid(row=3, column=1, columnspan=3)

#enter gender
gender_label = Label(newuser_window, text="Gender:", bg="white")
gender_label.grid(row=4, column=0)
genders = ["Male", "Female"] #set values for gender dropdown
gender_entry = ttk.Combobox(newuser_window, values=genders, width=26,
state="readonly") #make combobox read only so only "male" or "female" can be entered
gender_entry.grid(row=4, column=1, columnspan=3)

#enter date of birth
dob_label = Label(newuser_window, text="Date of Birth:", bg="white")
dob_label.grid(row=5, column=0, padx=30)

#year, month, and day born must be entered individually
dob_year_entry = Entry(newuser_window, width=8)
dob_month_entry = Entry(newuser_window, width=8)
dob_day_entry = Entry(newuser_window, width=8)
dob_year_entry.grid(row=5, column=3)
dob_month_entry.grid(row=5, column=2)
dob_day_entry.grid(row=5, column=1)

```



```

#enter height
height_label = Label(newuser_window, text="Height (cm):", bg="white")
height_label.grid(row=7, column=0, padx=30)
height_entry = Entry(newuser_window, width=27)
height_entry.grid(row=7, column=1, columnspan=3)

#enter weight
weight_label = Label(newuser_window, text="weight (kg):", bg="white")
weight_label.grid(row=8, column=0, padx=30)
weight_entry = Entry(newuser_window, width=27)
weight_entry.grid(row=8, column=1, columnspan=3)

#enter exercise level
exercise_level_label = Label(newuser_window, text="Exercise level:", bg="white")
exercise_level_label.grid(row=9, column=0, padx=30)
exercise_levels = ["couch potato", "moderately active (1hr daily)", "vigorously active (2
hours per day)", "extremely active (>2 hours per day)"] #set values for activity level
dropdown
exercise_level_entry = ttk.Combobox(newuser_window, values=exercise_levels, width=26,
state="readonly") #dropdown to select activity level
exercise_level_entry.grid(row=9, column=1, columnspan=3)

#button to submit details and create and account
submitaccount_button = Button(newuser_window, text="Submit",
command=lambda:createaccount (newuser_window, password_entry.get(),
firstname_entry.get(), lastname_entry.get(), gender_entry.get(), dob_day_entry.get(),
dob_month_entry.get(), dob_year_entry.get(), height_entry.get(), weight_entry.get(),
exercise_level_entry.get()))
submitaccount_button.grid(row=10,column=0, columnspan=4)

#calculate the users age from date of birth
def calculate_age():
    #turn year, month and day born into integers so they can be used in an equation to find
age
    yearborn = int(user_logged_in.get_yearborn())
    monthborn = int(user_logged_in.get_monthborn())
    dayborn = int(user_logged_in.get_dayborn())
    birthDate = datetime.datetime(yearborn, monthborn, dayborn) #combine day, month and
day born to make one date
    today = datetime.datetime.now() #user datetime module to get todays date
    age = today.year - birthDate.year - ((today.month, today.day) < (birthDate.month,
birthDate.day)) #subtract birth date from current date to find users age
    user_logged_in.set_age(age) #add age to user_logged_in object
    return(age)

```

```

#check a date entered by user is real
def check_date_is_real(day, month, year):
    if day.isnumeric() and month.isnumeric() and year.isnumeric(): #checks values entered are
        numbers

        #make day, month, and year variables integers, so that functions such as > can be used
        on them
        day = int(day)
        month = int(month)
        year = int(year)

        if month > 12 or month == 0: #checks if month is larger than twelve, returns false if is
            return False

        if month == 1 or month == 3 or month == 5 or month == 7 or month == 8 or month == 10
        or month == 12: #checks if month entered has 31 days
            if day <= 31 and day >= 1: #checks day entered is 31 or less
                return True
            else: #if month has more than 31 or less than 1 days, return false
                return False

        if month == 4 or month == 6 or month == 9 or month == 11: #checks if month is one with
        30 days
            if day <= 30 and day >= 1: #if month is one with 30 days, checks day is less than or
            equal to 30
                return True
            else: #if month has more than 30 or less than 1 days, return false
                return False

        if year % 4 == 0 and (year % 100 != 0 or year % 400 == 0): #calculates wether or not year
        given is a leap year
            if day <= 29 and day >= 1: #if year is a leap year, checks if Feruary has 29 or less days
                return True
            else: #if month has more than 29 or less than 1 days, return false
                return False
            else:
                if day <= 28 and day >= 1: #if year isn't a leap year, checks if Febrary has 28 or less
                days
                    return True
                else: #if month has more than 28 or less than 1 days, return false
                    return False

        else: #if values given are non numeric, return false
            return False

```

```

#enter details of a new user into database
def createaccount(newuser_window, password, firstname, lastname, gender, dayborn,
monthborn, yearborn, height, weight, exerciselvl):

    #turn exercise level into an integer so it can go in the database
    if exerciselvl == "couch potato":
        exerciselvl = 1
    elif exerciselvl == "moderately active (1hr daily)":
        exerciselvl = 2
    elif exerciselvl == "vigorously active (2 hours per day)":
        exerciselvl = 3
    elif exerciselvl == "extremely active (>2 hours per day)":
        exerciselvl = 4
    else:
        messagebox.showerror(title="Error", message="Please enter a valid exercise level") #if
exercise level given is not one of 4 options, show error

    #check data has been entered into all entries
    if password == "": #check data has been entered
        messagebox.showerror(title="Error", message="Please enter a password")
        return None

    elif firstname == "": #check data has been entered
        messagebox.showerror(title="Error", message="Please enter your first name")
        return None

    elif lastname == "": #check data has been entered
        messagebox.showerror(title="Error", message="Please enter your last name")
        return None

    elif gender == "": #check data has been entered
        messagebox.showerror(title="Error", message="Please select your gender")
        return None

    elif dayborn == "": #check data has been entered
        messagebox.showerror(title="Error", message="Please enter your date of birth")
        return None

    elif monthborn == "": #check data has been entered
        messagebox.showerror(title="Error", message="Please enter your date of birth")
        return None

    elif yearborn == "": #check data has been entered
        messagebox.showerror(title="Error", message="Please enter your date of birth")

```

```

return None

#check date of birth entered is an actual, real date
elif check_date_is_real(dayborn, monthborn, yearborn) == False:
    messagebox.showerror(title="Error", message="Please enter a valid date of birth
(dd/mm/yyyy)")
    return None

elif height == "": #check data has been entered
    messagebox.showerror(title="Error", message="Please enter your height")
    return None

#check height is a integer
elif height.isdigit() == False:
    messagebox.showerror(title="Error", message="Please enter a valid height")
    return None

elif weight == "": #check data has been entered
    messagebox.showerror(title="Error", message="Please enter your weight")
    return None

#check weight is an integer
elif weight.isnumeric() == False:
    messagebox.showerror(title="Error", message="Please enter a valid weight")
    return None

else:
    with database_connection: #make connection to database
        sql_statement = "INSERT INTO TblUsers (firstname, lastname, gender, weight, height,
yearborn, monthborn, dayborn, password, exerciselvl) VALUES ('%s', '%s', '%s', %s, %s,
%s, %s, %s, '%s', %s)" % (firstname, lastname, gender, weight, height, yearborn,
monthborn, dayborn, hash(password), exerciselvl) #insert data into new record in TblUsers
(password is hashed)
        database_cursor.execute(sql_statement) #execute statement
        database_connection.commit() #commit changes to database

        sql_statement_2 = "SELECT MAX(userID) FROM TblUsers" #find most recent user ID
entered into table (from the account that has just been created) so it can be entered into
user_logged_in object
        database_cursor.execute(sql_statement_2)
        matching_users = database_cursor.fetchall()

#insert details entered for account creation into user_logged_in object
user_logged_in.set_id(matching_users[0][0])
user_logged_in.set_firstname(firstname)
user_logged_in.set_lastname(lastname)

```

```

user_logged_in.set_weight(weight)
user_logged_in.set_height(height)
user_logged_in.set_password(password)
user_logged_in.set_gender(gender)
user_logged_in.set_yearborn(yearborn)
user_logged_in.set_monthborn(monthborn)
user_logged_in.set_dayborn(dayborn)
user_logged_in.set_exercise_level(exercise_lvl)

```

```

calorie_target_page() #load calorie target page so user can choose calorie intake target
newuser_window.destroy() #close new user window

```

```

#create calorie target page
def calorie_target_page():

```

```

    #set physical activity ratio based on activity level of user so maintenance calories can be
    calculated

```

```

    if user_logged_in.get_exercise_level() == 1:
        physical_activity_ratio = 1.55
    elif user_logged_in.get_exercise_level() == 2:
        physical_activity_ratio = 1.85
    elif user_logged_in.get_exercise_level() == 3:
        physical_activity_ratio = 2.2
    elif user_logged_in.get_exercise_level() == 4:
        physical_activity_ratio = 2.4

```

```

    maintenance_cals = ((10 * int(user_logged_in.get_weight())) + (6.25 *
int(user_logged_in.get_height())) - (5 * int(calculate_age())) * int(physical_activity_ratio)
    #calculate maintenance calorie intake

```

```

    #subtract 166 from maintenance calorie intake if user is female
    if user_logged_in.get_gender() == "Female":
        maintenance_cals = maintenance_cals - 166

```

```

    #create different intake targets for different weight gain/loss goals
    rapidgaincals = int(round(maintenance_cals + 500))
    mildgaincals = int(round(maintenance_cals + 250))
    mildlosscals = int(round(maintenance_cals - 250))
    rapidlosscals = int(round(maintenance_cals - 500))
    maintenance_cals = int(round(maintenance_cals))

```

```

#create window
calorie_target_window = Tk() #create window for calorie target page
calorie_target_window.title("User ID: " + str(user_logged_in.get_id())) #set title for window

```

```

calorie_target_window.geometry("365x100") #set dimensions of window
calorie_target_window.configure(bg="white") #set background of window

choose_cals = Label(calorie_target_window, height = 1, width = 21, text = "Select target
calorie intake", bg="white", font = ("Imperial", 15))
choose_cals.grid(column=0, row=0, columnspan=5)

#place buttons and labels on screen for user to select different calorie intakes
rapid_gain_label = Label(calorie_target_window, text = ""+0.5kg
per month", bg="white")
rapid_gain_label.grid(column=0, row=1)
rapid_gain_button = Button(calorie_target_window, height=1, width=4, text=rapidgaincals,
command=lambda:inputcalstarget(rapidgaincals, calorie_target_window))
rapid_gain_button.grid(column=0, row=2)

mild_gain_label = Label(calorie_target_window, text = ""+0.25kg
per month", bg="white")
mild_gain_label.grid(column=1, row=1)
mild_gain_button = Button(calorie_target_window, height=1, width=4, text = mildgaincals,
command=lambda:inputcalstarget(mildgaincals, calorie_target_window))
mild_gain_button.grid(column=1, row=2)

maintain_label = Label(calorie_target_window, text = ""maintain
weight", bg="white")
maintain_label.grid(column=2, row=1)
maintain_button = Button(calorie_target_window, height=1, width=4, text =
maintenance_cals, command=lambda:inputcalstarget(maintenance_cals,
calorie_target_window))
maintain_button.grid(column=2, row=2)

mild_loss_label = Label(calorie_target_window, text = ""-0.25kg
per month", bg="white")
mild_loss_label.grid(column=3, row=1)
mild_loss_button = Button(calorie_target_window, height=1, width=4, text = mildlosscals,
command=lambda:inputcalstarget(mildlosscals, calorie_target_window))
mild_loss_button.grid(column=3, row=2)

rapid_loss_label = Label(calorie_target_window, text = ""-0.5kg
per month", bg="white")
rapid_loss_label.grid(column=4, row=1)
rapid_loss_button = Button(calorie_target_window, height=1, width=4, text = rapidlosscals,
command=lambda:inputcalstarget(rapidlosscals, calorie_target_window))
rapid_loss_button.grid(column=4, row=2)

```

```

#enter calories target to database
def inputcalstarget(calories, calorie_target_window):
    with database_connection:

        user_id = user_logged_in.get_id()
        print(calories)
        print(user_id)
        print(type(calories))
        print(type(user_id))
        sql_statement = "UPDATE TblUsers SET targetintake = %s WHERE userID = %s" %
(calories, user_id) #insert target calories into user record
        database_cursor.execute(sql_statement)
        database_connection.commit() #commit changes
        user_logged_in.set_target_intake(calories) #add target intake to user_logged_in object
        calorie_target_window.destroy() #destroy calorie target window as it is no longer needed
        nutrition_and_workout_page() #load nutrition page as calorie target has been chosen

```

```

#submit login details, check database to see if they are valid, pull user info from database
and put into user_logged_in object, and continue to nutrition page
def login_submit(login_window, userID, password):

```

```

    if userID == "": #check userID has been entered
        messagebox.showerror(title="Error", message="Please enter a user ID")

    if userID.isnumeric():
        if userID == "": #check userID has been entered
            messagebox.showerror(title="Error", message="Please enter a user ID")
        elif password == "": #check password has been entered
            messagebox.showerror(title="Error", message="Please enter a password")
        else:
            with database_connection:
                sql_statement = "SELECT * FROM TblUsers WHERE userID = %s AND Password =
'%s'" % (userID, hash(password)) #select record from TblUsers with matching username and
password (password is hashed)
                database_cursor.execute(sql_statement)
                matching_users = database_cursor.fetchall()

```

```

    if len(matching_users) == 1: #check there is only 1 record with a matching userID and
password
        #take all data from matching results in database and add to user_logged_in object
        user_logged_in.set_id(matching_users[0][0])
        user_logged_in.set_firstname(matching_users[0][1])
        user_logged_in.set_lastname(matching_users[0][2])
        user_logged_in.set_weight(matching_users[0][3])
        user_logged_in.set_height(matching_users[0][4])
        user_logged_in.set_password(matching_users[0][5])
        user_logged_in.set_gender(matching_users[0][6])
        user_logged_in.set_yearborn(matching_users[0][7])
        user_logged_in.set_monthborn(matching_users[0][8])
        user_logged_in.set_dayborn(matching_users[0][9])
        user_logged_in.set_target_intake(matching_users[0][11])
        user_logged_in.set_exercise_level(matching_users[0][12])
        sql_statement = "UPDATE TblUsers SET age = %s WHERE userID = %s" %
(calculate_age(), user_logged_in.get_id()) #recalculate and update age of user incase they
have gotten a year older
        database_cursor.execute(sql_statement)
        database_connection.commit()
        login_window.destroy() #close login window as login process is now complete
        nutrition_and_workout_page() #open nutrition page
    else:
        messagebox.showerror(title="Error", message="Username or password is incorrect") #if
credentials entered do not match to and database records, ask the user to reenter username
and/or password
    else:
        messagebox.showerror(title="Error", message="User ID must be a number.") #show error
message is user ID given is not a number

#this subroutine logs the user out, and resets all user_logged_in values
def logout(main_window):

    #clear all attributes in user_logged_in and set them to 0
    user_logged_in.set_id(0)
    user_logged_in.set_firstname("")
    user_logged_in.set_lastname("")
    user_logged_in.set_weight(0)
    user_logged_in.set_height(0)
    user_logged_in.set_password("")
    user_logged_in.set_gender("")
    user_logged_in.set_yearborn(0)
    user_logged_in.set_monthborn(0)
    user_logged_in.set_dayborn(0)
    user_logged_in.set_target_intake(0)
    user_logged_in.set_exercise_level(0)

```



```

main_window.destroy() #close main window
login() #open login window

#create main window with nutrition and workout page in
def nutrition_and_workout_page():
    #make window
    main_window = Tk()
    main_window.title("User ID: " + str(user_logged_in.get_id())) #display the users User ID in
the window title
    main_window.geometry("365x690")

    logout_button = Button(main_window, width = 3, height =1, text = "Log out",
command=lambda: logout(main_window))
    logout_button.grid(column=0, row=0)

    #add notebook to make different tabs
    notebook = ttk.Notebook(main_window)
    notebook.grid(column=0, row=1)

    #nutrition tab
    nutrition = Frame(notebook, width=1280, height=705, bg="white")
    nutrition.pack(fill="both", expand=1)

    #workout training tab
    workout = Frame(notebook, width=1280, height=705, bg="white")
    workout.pack(expand=1)

    #add tabs to top of screen
    notebook.add(nutrition, text="Nutrition Tracker")
    notebook.add(workout, text="Workout Tracker")

    #load nutrition page in the nutrition tab in the notebook
    nutrition_page(nutrition)

    #load workout page in the workout tab in the notebook
    workout_page(workout)

    main_window.mainloop()

#create nutrition page
def nutrition_page(nutrition):
    #make calories consumed title
    calorie_intake = Label(nutrition, height = 1, width=20, text="Calories Consumed:",
bg="white", font=("Imperial", 20))

```

```

calorie_intake.grid(row=0, column=0, pady=20)

#make counter for calories consumed
calorie_intake_counter = Label(nutrition, height = 1, width=10, text="", bg="white",
font=("Imperial", 30))

#make calories consumed progress bar
calorie_intake_counter.grid(row=1, column=0)
calorie_intake_progress_bar = ttk.Progressbar(nutrition, orient=HORIZONTAL, length =
250, mode="determinate")
calorie_intake_progress_bar.grid(row=3, column=0)

#display calorie intake target
target_label = Button(nutrition, height = 1, width= 13, bg="white", font=("Imperial", 20),
command=lambda: calorie_target_page())
target_label.grid(row=4, column=0)

#make the treeview for calories consumed
meal_record = ttk.Treeview(nutrition, height=5)
meal_record["columns"] = ("#1", "#2", "#3")

meal_record.column("#0", anchor=CENTER, stretch=NO, width=0)
meal_record.heading("#0", text = "")

meal_record.column("#1", anchor=CENTER, stretch=NO, width=100)
meal_record.heading("#1", text = "Meal")

meal_record.column("#2", anchor=E, stretch=NO, width=50)
meal_record.heading("#2", text="Time")

meal_record.column("#3", anchor=CENTER, stretch=NO, width=100)
meal_record.heading("#3", text="Calories")
meal_record.grid(row=5, column=0, pady=10)

#add all meals eaten today in database to meals treeview
with database_connection:
    global iidnumb2 #bring iidnumb2 into function, so it can be used and updated when
adding to treeview
    todays_date = datetime.datetime.now() #find todays date
    today_formatted_as_string = todays_date.strftime("%Y-%m-%d") #format todays date as a
string

    sql_statement = "SELECT targetintake, Mealdesc, Timeeaten, Calories FROM TblUsers,
TblMeals WHERE TblUsers.userID = TblMeals.userID AND TblUsers.userID = %s AND
dateeaten = '%s'" % (user_logged_in.get_id(), today_formatted_as_string) #sql statement to
find the details of all meals eaten today by current user, and their calorie target

```

```

database_cursor.execute(sql_statement) #execute sql command
matching_results = database_cursor.fetchall() #set matching_users variable equal to
result of SQL query
if len(matching_results) == 0: #check if there are any results of the above SQL query
    target_label.config(text = "Target: " + str(user_logged_in.get_target_intake()) + "kcal")
#set the text on target_label equal to the users target calorie intake
else:
    target_label.config(text = "Target: " + str(matching_results[0][0]) + "kcal") #set the text on
target_label equal to the users target calorie intake

#this loop inputs all meals into treeview
for x in range (len(matching_results)):
    meal_record.insert(parent="", index="end", iid = iidnumb2, text="parent",
values=(matching_results[x][1], matching_results[x][2], matching_results[x][3])) #insert meals
eaten today into treeview
    iidnumb2 = iidnumb2 + 1 #update iid number so things can be added to treeview in
future
    sql_statement = "SELECT SUM(calories) FROM TbIMeals WHERE dateeaten = '%s' AND
userID = %s" % (today_formatted_as_string, user_logged_in.get_id()) #SQL statement to
find the sum of all calories eaten today
    database_cursor.execute(sql_statement) #execute SQL statement
    total_calories = database_cursor.fetchall() #set total_calories variable equal to result of
SQL statement

#if the user has not inputted any meals today, set counter to 0
if total_calories[0][0] == "None":
    total_calories = 0
    calorie_intake_counter.config(text=total_calories) #add total calories to total calorie
counter

addmealf = Frame(nutrition, bg="white") #create frame to put add meals menu in
addmealf.grid(row=6, column=0)

addm = Label(addmealf, text="Add a Meal", bg="white") #create a label to show where to
enter a meal into database and treeview
addm.grid(row=0, column=0, columnspan=4)

addmeal = Label(addmealf, text="Meal", bg="white") #create label to show where to enter
meal description
addmeal.grid(row=1, column=0, padx=30)
mealbox = Entry(addmealf, width=15) #create an entry to enter meal description
mealbox.grid(row=2, column=0)

addtime = Label(addmealf, text="Time", bg="white") #create a label to show where to enter
the time a meal was eaten

```

```

addtime.grid(row=1, column=1, padx=38, columnspan=2)

minutes = ["00", "05", "10", "15", "20", "25", "30", "35", "40", "45", "50", "55"] #set values for
minutes
hours = ["00", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13", "14",
"15", "16", "17", "18", "19", "20", "21", "22", "23"] #set values for hours

hourselect = ttk.Combobox(addmealf, values=hours, width=3) #create dropdown to enter
what hour a meal was eaten
hourselect.current(0) #set default value in hourselect dropdown to the first data in the array,
"00"
hourselect.grid(row=2, column=1)
minuteselect = ttk.Combobox(addmealf, values=minutes, width=3) #create dropdown to
enter what minute a meal was eaten
minuteselect.current(0) #set default value in minuteselect dropdown to the first data in the
array, "00"
minuteselect.grid(row=2, column=2)

addcals = Label(addmealf, text="Calories", bg="white") #create a label to show users
where calories are entered
addcals.grid(row=1, column=3, padx=30) #place add calories label in screen
calseatenbox = Entry(addmealf, width=15) #create entry to add how many calories were in
a meal
calseatenbox.grid(row=2, column=3) #place calseaten entry on screen

add_data = Button(addmealf, text="Add", command=lambda:
add_meal(calorie_intake_progress_bar, meal_record, mealbox, calseatenbox, hourselect,
minuteselect, calorie_intake_counter)) #create button to add data to treeview & database
add_data.grid(row=3, column=1, columnspan=2, pady=5) #place add data button on
screen

#create calorie deficit/surplus bar chart
bar_chart_frame = Frame(nutrition, bg="white")
bar_chart_frame.grid(row=7, column=0)

#call bar chart function to plot the chart
bar_chart(bar_chart_frame)

##### WORKOUT PAGE #####
def workout_page(workout):
    #create treeview to enter and show workouts
    workout_record = ttk.Treeview(workout, height=10)

```

```

workout_record["columns"] = ("#1", "#2", "#3", "#4", "#5")

workout_record.column("#0", anchor=CENTER, stretch=NO, width=0)
workout_record.heading("#0", text = "")

workout_record.column("#1", anchor=CENTER, stretch=NO, width=100)
workout_record.heading("#1", text = "Exercise")

workout_record.column("#2", anchor=E, stretch=NO, width=50)
workout_record.heading("#2", text="Sets")

workout_record.column("#3", anchor=CENTER, stretch=NO, width=50)
workout_record.heading("#3", text="Reps")

workout_record.column("#4", anchor=CENTER, stretch=NO, width=55)
workout_record.heading("#4", text="Weight")

workout_record.column("#5", anchor=CENTER, stretch=NO, width=100)
workout_record.heading("#5", text="Comments")
workout_record.grid(row=0, column=0, pady=10, columnspan=5) #place treeview on
screen

with database_connection:
    global iidnumb #import iidnumb so that data can be added to the treeview in the correct
spot
    todays_date = datetime.datetime.now() #create todays_date variable equal to todays date
using datetime module
    today_formatted_as_string = todays_date.strftime("%Y-%m-%d") #format todays_date into
string

    sql_statement = "SELECT * FROM TblWorkout WHERE date = '%s' AND userID = %s" %
(today_formatted_as_string, user_logged_in.get_id()) #sql statement to find all exercises
done today by current user
    database_cursor.execute(sql_statement) #execute SQL statement
    matching_results = database_cursor.fetchall() #set matching_results variable equal to
result of SQL query
    for x in range (len(matching_results)): #create for loop so that all exercises are added to
treeview
        workout_record.insert(parent="", index="end", iid = iidnumb, text="parent",
values=(matching_results[x][3], matching_results[x][4], matching_results[x][5],
matching_results[x][6], matching_results[x][7])) #insert exercises performed today into
treeview
        iidnumb = iidnumb + 1 #update iid number so things can be added to treeview in future

    enter_exercise_label = Label(workout, text="Add an exercise:", bg="white") #make label
telling user to enter exercise

```

```

enter_exercise_label.grid(row=1, column=0, columnspan=5)

add_exercise = Label(workout, text="Exercise", bg="white") #create label to show where to
enter exercise
add_exercise.grid(row=2, column=0, padx=1)
exercise_box = Entry(workout, width=10) #create an entry to enter exercise
exercise_box.grid(row=3, column=0)

add_sets = Label(workout, text="Sets", bg="white") #create label to show where to enter
sets
add_sets.grid(row=2, column=1, padx=1)
sets_box = Entry(workout, width=5) #create an entry to enter sets performed
sets_box.grid(row=3, column=1)

add_reps = Label(workout, text="Reps", bg="white") #create label to show where to enter
reps
add_reps.grid(row=2, column=2, padx=1)
reps_box = Entry(workout, width=5) #create an entry to enter reps
reps_box.grid(row=3, column=2)

add_weight = Label(workout, text="Weight", bg="white") #create label to show where to
enter weight
add_weight.grid(row=2, column=3, padx=1)
weight_box = Entry(workout, width=5) #create an entry to enter weight
weight_box.grid(row=3, column=3)

add_comments = Label(workout, text="Comments", bg="white") #create label to show
where to enter any comments about workout
add_comments.grid(row=2, column=4, padx=1)
comments_box = Entry(workout, width=10) #create an entry to enter any comments about
workout
comments_box.grid(row=3, column=4)

add_workout = Button(workout, text="Add", command=lambda:
enter_workout(workout_record, exercise_box, sets_box, reps_box, weight_box,
comments_box)) #create button to add info in entries to treeview and database
add_workout.grid(row=4, column=0, columnspan=5, pady=10)

find_workout_label = Label(workout, text="Find a workout:", bg="white") #create label to
show users where they can search for a workout
find_workout_label.grid(row=5, column=0, columnspan=5)

day_label = Label(workout, text="Day", bg="white") #add label to show where to enter day
day_label.grid(row=6, column=1)

```

```

month_label = Label(workout, text="Month", bg="white") #add label to show where to enter
month
month_label.grid(row=6, column=2)
year_label = Label(workout, text="year", bg="white") #add label to show where to enter
year
year_label.grid(row=6, column=3)

find_workout_day = Entry(workout, width = 5) #make entry box for day
find_workout_day.grid(row=7, column=1)

find_workout_month = Entry(workout, width = 5) #make entry box for month
find_workout_month.grid(row=7, column=2)

find_workout_year = Entry(workout, width = 5) #make entry box for year
find_workout_year.grid(row=7, column=3)

find_workout_button = Button(workout, text="Search", command=lambda:
find_workout(old_workout_record, find_workout_day, find_workout_month,
find_workout_year)) #create button to find workout from date entered by user
find_workout_button.grid(row=7, column=4)

#create treeview to show past workouts
old_workout_record = ttk.Treeview(workout, height=10)
old_workout_record["columns"] = ("#1", "#2", "#3", "#4", "#5")

old_workout_record.column("#0", anchor=CENTER, stretch=NO, width=0)
old_workout_record.heading("#0", text = "")

old_workout_record.column("#1", anchor=CENTER, stretch=NO, width=100)
old_workout_record.heading("#1", text = "Exercise")

old_workout_record.column("#2", anchor=CENTER, stretch=NO, width=50)
old_workout_record.heading("#2", text="Sets")

old_workout_record.column("#3", anchor=CENTER, stretch=NO, width=50)
old_workout_record.heading("#3", text="Reps")

old_workout_record.column("#4", anchor=CENTER, stretch=NO, width=55)
old_workout_record.heading("#4", text="Weight")

old_workout_record.column("#5", anchor=CENTER, stretch=NO, width=100)
old_workout_record.heading("#5", text="Comments")
old_workout_record.grid(row=8, column=0, pady=10, columnspan=5) #place treeview on
screen

```

```

#give the option to add a meal
def add_meal(calorie_intake_progress_bar, meal_record, mealbox, calseatenbox,
hourselect, minuteselect, calorie_intake_counter):
    global iidnumb2 #bring iidnumb2 into subroutine so that the correct one can be used when
adding to treeview

    #check time values entered are numbers, not letters. Show error if there are any letters
    if hourselect.get().isnumeric() == False or minuteselect.get().isnumeric() == False:
        messagebox.showerror(title="Error", message="Make sure both hours and minutes are
numbers")
        return None

    if calseatenbox.get() == "": #check calories box is not empty
        messagebox.showerror(title="Error", message="Please enter calories")
        return None #if box is empty, show error

    if calseatenbox.get().isnumeric(): #check if calories eaten is a number
        meal_record.insert(parent="", index="end", iid = iidnumb2, text="parent",
values=(mealbox.get(), hourselect.get() + ":" + minuteselect.get(), calseatenbox.get()))
    #insert meal into treeview
    iidnumb2 = iidnumb2 + 1 #updt e iid so future items can be added to treeview

    time_eaten = hourselect.get() + ":" + minuteselect.get() #add hour and minute
comboboxes to make time
    todays_date = datetime.datetime.now()
    today_formatted_as_string = todays_date.strftime("%Y-%m-%d") #calculate todays date
using datetime module
    with database_connection: #make connection to database
        sql_statement = "INSERT INTO TblMeals (UserID, MealDesc, Dateeaten, Timeeaten,
Calories) VALUES (%s, '%s', '%s', '%s', %s)" % (user_logged_in.get_id(), mealbox.get(),
today_formatted_as_string, time_eaten, calseatenbox.get()) #insert meal into TblMeals
        database_cursor.execute(sql_statement) #executes the query
        database_connection.commit() #commits the changes to the database
        sql_statement = "SELECT SUM(calories) FROM TblMeals WHERE dateeaten = '%s'
AND userID = %s" % (today_formatted_as_string, user_logged_in.get_id()) #select the total
calories eaten today
        database_cursor.execute(sql_statement) #executes the query
        total_calories = database_cursor.fetchall() #sets result of query equal to total_calories
        calorie_intake_counter.config(text=total_calories) #add calories to total calorie counter

    #update progress bar

```



```

calorie_intake_progress_bar['value'] = int((int(total_calories[0][0]) /
user_logged_in.get_target_intake()) * 100) #update progress abr to be filled in by the
percentage of the users calorie target that they have eaten

```

```

#clear boxes
mealbox.delete(0, END)
calseatenbox.delete(0, END)

```

```

else:
    messagebox.showerror(title="Error", message="Calories consumed must be a whole
number") #make user reenter calories if it is not a number

```

```

#clear calories box as it was entered incorrectly
calseatenbox.delete(0, END)

```

```

def find_meal_page():
    find_meal_window = Tk() #create find_meal_window
    find_meal_window.title("User ID: " + str(user_logged_in.get_id())) #display the users User
ID in the window title
    find_meal_window.geometry("252x210") #set window size
    find_meal_window.config(bg="white") #set window background to white

```

```

#make the treeview for meals
old_meal_record = ttk.Treeview(find_meal_window, height=5)
old_meal_record["columns"] = ("#1", "#2", "#3")

```

```

old_meal_record.column("#0", anchor=CENTER, stretch=NO, width=0)
old_meal_record.heading("#0", text = "")

```

```

old_meal_record.column("#1", anchor=CENTER, stretch=NO, width=100)
old_meal_record.heading("#1", text = "Meal")

```

```

old_meal_record.column("#2", anchor=E, stretch=NO, width=50)
old_meal_record.heading("#2", text="Time")

```

```

old_meal_record.column("#3", anchor=CENTER, stretch=NO, width=100)
old_meal_record.heading("#3", text="Calories")
old_meal_record.grid(row=0, column=0, columnspan=3)

```

```

#create label to show where users can enter a date to find meals eaten on that day
find_meal_label = Label(find_meal_window, text="Find all meals eaten in a day:",
bg="white")
find_meal_label.grid(row=1, column=0, columnspan=3)

```

```

    day_label = Label(find_meal_window, text="Day", bg="white") #add label to show where to
enter day
    day_label.grid(row=2, column=0)
    month_label = Label(find_meal_window, text="Month", bg="white") #add label to show
where to enter month
    month_label.grid(row=2, column=1)
    year_label = Label(find_meal_window, text="year", bg="white") #add label to show where to
enter year
    year_label.grid(row=2, column=2)

    find_meal_day = Entry(find_meal_window, width = 5) #make entry box for day
    find_meal_day.grid(row=3, column=0)

    find_meal_month = Entry(find_meal_window, width = 5) #make entry box for month
    find_meal_month.grid(row=3, column=1)

    find_meal_year = Entry(find_meal_window, width = 5) #make entry box for year
    find_meal_year.grid(row=3, column=2)

    find_meal_button = Button(find_meal_window, text="Search", command=lambda:
find_meal(old_meal_record, find_meal_day, find_meal_month, find_meal_year)) #create
buton to find meals from date entered by user
    find_meal_button.grid(row=4, column=1)

def find_meal(old_meal_record, find_meal_day, find_meal_month, find_meal_year):
    global iidnumb4 #import iidnumb4 so it can be user to add to treeview

    #set month and day equal to those entered by the user
    month = find_meal_month.get()
    day = find_meal_day.get()

    if len(day) == 1:
        day = "0" + day #adds 0 to start of day if only 1 digit is entered so that it can be found in
database

    if len(month) == 1:
        month = "0" + month #adds 0 to start of month if only 1 digit is entered so that it can be
found in database

    if check_date_is_real(day, month, find_meal_year.get()): #check date entered is a real date
        date = str(find_meal_year.get() + "-" + month + "-" + day) #combine values entered into 1
date string, so that it can be used to search the database

    with database_connection: #create connection to database

```

```

    sql_statement = "SELECT * FROM TblMeals WHERE UserID = %s AND Dateeaten = '%s'" % (user_logged_in.get_id(), date) #select all data from TblUsers where userID and date match those entered
    database_cursor.execute(sql_statement) #executes query
    matching_meals = database_cursor.fetchall() #sets results of query as matching_exercises variable

    if len(matching_meals) == 0: #checks if there are any results at all, and returns error box if there are none
        messagebox.showerror(title="Error", message="No meals found on this date")
    else:
        #sort results by sets before displaying
        matching_meals = sort(matching_meals, 4)
        matching_meals.reverse() #reverse order of matching exercises so exercises with most sets appear at the top
        for x in range (len(matching_meals)): #create for loop so that all exercises are added to treeview
            old_meal_record.insert(parent="", index="end", iid = iidnumb4, text="parent", values=(matching_meals[x][2], matching_meals[x][4], matching_meals[x][5])) #insert exercises performed today into treeview
            iidnumb4 = iidnumb4 + 1 #update iid number so things can be added to treeview in future

        else:
            messagebox.showerror(title="Error", message="Please enter a valid date") #if date is not real, tell user to enter a valid date

#give option to add workout
def enter_workout(workout_record, exercise_box, sets_box, reps_box, weight_box, comments_box):
    global iidnumb #import iidnumb so the correct iid can be used to insert data into treeview

    #make sure an exercise name has been entered
    if exercise_box.get() == "":
        messagebox.showerror(title="Error", message="Enter exercise name")
        return None #if no exercise name is entered, end subroutine

    #make sure weight has been entered
    if weight_box.get() == "":
        messagebox.showerror(title="Error", message="Enter weight")
        return None #if no weight is entered, end subroutine

    #check sets box is not empty
    if sets_box.get() == "":
        messagebox.showerror(title="Error", message="Enter sets performed")

```

```

return None #if no sets are entered, end subroutine

#check reps box is not empty
if reps_box.get() == "":
    messagebox.showerror(title="Error", message="Enter reps performed")
    return None #if no reps are entered, end subroutine

if sets_box.get().isnumeric() and reps_box.get().isnumeric(): #check sets and reps entered
by user are numeric values

    todays_date = datetime.datetime.now() #calculate todays date using datetime module
    today_formatted_as_string = todays_date.strftime("%Y-%m-%d") #calculate todays date
using datetime module

    workout_record.insert(parent="", index="end", iid = iidnumb, text="parent",
values=(exercise_box.get(), sets_box.get(), reps_box.get(), weight_box.get(),
comments_box.get())) #insert data into treeview
    iidnumb = iidnumb + 1 #add 1 to iidnumb so that data can be added to next slot in future

#add information to TblWorkout in database
with database_connection: #make connection to database
    sql_statement = "INSERT INTO TblWorkout (UserID, date, exercise, sets, reps, weight,
comments) VALUES (%s, '%s', '%s', %s, %s, '%s', '%s')" % (user_logged_in.get_id(),
today_formatted_as_string, exercise_box.get(), sets_box.get(), reps_box.get(),
weight_box.get(), comments_box.get()) #insert workout into TblWorkout
    database_cursor.execute(sql_statement) #executes the query
    database_connection.commit() #commits the changes to the database

#clear entry boxes
exercise_box.delete(0, END)
sets_box.delete(0, END)
reps_box.delete(0, END)
weight_box.delete(0, END)
comments_box.delete(0, END)

else:
    messagebox.showerror(title="Error", message="Make sure sets and reps are numbers!")
#show error box is sets and reps entered by user are non numeric

#find a workout from a date entered by the user
def find_workout(old_workout_record, find_workout_day, find_workout_month,
find_workout_year):
    global iidnumb3 #import iidnumb3 so it can be user to add to treeview

    #set month and day variables equal to those entered by the user

```

```

month = find_workout_month.get()
day = find_workout_day.get()

if len(day) == 1:
    day = "0" + day #adds 0 to start of day if only 1 digit is entered so that it will match the
dates in the database, which all have 2 digit months and days

if len(month) == 1:
    month = "0" + month #adds 0 to start of month if only 1 digit is entered so that it will match
the dates in the database, which all have 2 digit months and days

if check_date_is_real(day, month, find_workout_year.get()): #check date entered is a real
date
    date = str(find_workout_year.get() + "-" + month + "-" + day) #combine values entered into
1 date string, so that it can be used to search the database

    with database_connection: #create connection to database
        sql_statement = "SELECT * FROM TblWorkout WHERE userID = %s AND date = '%s'"
% (user_logged_in.get_id(), date) #select all data from TblUsers where userID and date
match those entered
        database_cursor.execute(sql_statement) #executes query
        matching_exercises = database_cursor.fetchall() #sets results of query as
matching_exercises variable

    if len(matching_exercises) == 0: #checks if there are any results at all, and returns error
box if there are none
        messagebox.showerror(title="Error", message="No workout found on this date")
    else:
        #sort results by sets before displaying
        matching_exercises = sort(matching_exercises, 4)
        matching_exercises.reverse() #reverse order of matching exercise so exercises with
most sets appear at the top
        for x in range (len(matching_exercises)): #create for loop so that all exercises are added
to treeview
            old_workout_record.insert(parent="", index="end", iid = iidnumb3, text="parent",
values=(matching_exercises[x][3], matching_exercises[x][4], matching_exercises[x][5],
matching_exercises[x][6], matching_exercises[x][7])) #insert exercises performed today into
treeview
            iidnumb3 = iidnumb3 + 1 #update iid number so things can be added to treeview in
future

        else:
            messagebox.showerror(title="Error", message="Please enter a valid date") #if date is not
real, tell user to enter a valid date

```

```

#Draw bar chart to show calorie intake
def bar_chart(bar_chart_frame):
    today = datetime.datetime.now() #set "today" variable equal to todays date

    #create label to show that line represents calorie maintenance level
    maintenance_label = Label(bar_chart_frame, text = """"
maintenance
calories""", bg="white") #create label for central maintenance calories axis on bar chart
    maintenance_label.grid(row=0, column=0)

    find_meal_button = Button(bar_chart_frame, text = """"search
meals""", command=lambda: find_meal_page()) #create button to open find meals window
    find_meal_button.grid(row=1, column=0)

    bar_chart_canvas = Canvas(bar_chart_frame, width=270, height=203, bg="white") #create
canvas to draw bar chart onto
    bar_chart_canvas.grid(row=0, column=1, rowspan=2) #place bar chart on screen

    bar_width = 45 #set width of bars
    x_position = 17 #set x position, where the first bar will be placed

    for x in range (1, 6): #make loop repeat 5 times, so it covers the last 5 days
        day = today - timedelta(days=x)#set day variable equal to todays date minus x days
        day = str(day.strftime("%Y-%m-%d")) #make the date a string, I have formatted the daye
backwards (yy-mm-dd) as this is how it appears in the database
        with database_connection: #make connection to database
            sql_statement = "SELECT SUM(calories) FROM TbIMeals WHERE dateeaten = '%s'
AND userID = %s" % (day, user_logged_in.get_id()) #this sql statement finds the total
calories eaten on day x
            database_cursor.execute(sql_statement) #execute sql statement
            result = database_cursor.fetchall() #set result of query equal to "result" variable

        if result[0][0] == None: #check if no calories have been recorded on this day
            bar_chart_canvas.create_text(x_position, 87, text= """"          No meals
            recorded""", fill = "black", font = "Helvetica 6") #if no meals are foud in the database,
put text in the slot saying no meals have been recorded on this day

            elif int(str(result[0][0])) >= user_logged_in.get_target_intake(): #check if result is bigger
than maintenance calories

```

```

        bar_chart_canvas.create_rectangle(x_position, 100, x_position + bar_width, 100 -
((int(result[0][0]) - user_logged_in.get_target_intake()) / 10), fill = "grey") #create bar above
axis showing calorie surplus

```

```

        bar_chart_canvas.create_text(x_position, 108, text= "          +" + str(int(result[0][0]) -
user_logged_in.get_target_intake()) + "kcal", fill = "black", font = "Helvetica 6") #place text
below the bar to show the exact calorie surplus

```

```

    else:

```

```

        bar_chart_canvas.create_rectangle(x_position, 100 - ((int(result[0][0]) -
user_logged_in.get_target_intake()) / 10), x_position + bar_width, 100, fill = "grey") #if they
have not eaten more than their maintenance calories, put bar below axis to show calorie
deficit

```

```

        bar_chart_canvas.create_text(x_position, 92, text= "          -" +
str(user_logged_in.get_target_intake() - int(result[0][0])) + "kcal", fill = "black", font =
"Helvetica 6") #place text above bar to show the exact calorie deficit

```

```

    x_position += 50 #add 55 to x position so bars are well spaced out

```

```

    bar_chart_canvas.create_line(0, 100, 270, 100, width = 3) #create axis for 0/maintenance
calories

```

```

#sort workouts found from query by sets performed from highest to lowest

```

```

def sort(data, sets):

```

```

    if len(data) <= 1: #check array is larger than 1 index, if it is no sorting has to be done
        return data #return list if there is 1 or less items in it

```

```

    midpoint = len(data) // 2 #create midpoint to split array in 2

```

```

    left_array = data[:midpoint] #create 2 different arrays from left and right halves of list

```

```

    right_array = data[midpoint:]

```

```

    #merge sort both the left and right arrays

```

```

    left_sorted = sort(left_array, sets)

```

```

    right_sorted = sort(right_array, sets)

```

```

    return merge(left_sorted, right_sorted, sets) #return the merged version of the 2 arrays

```

```

#this subroutine carries out the merge part of a merge sort

```

```

def merge(left_array, right_array, sets):

```

```

    result = [] #create empty array for result of merge

```

```

left_index, right_index = 0, 0 #create values for the index in each list that is being compared

while left_index < len(left_array) and right_index < len(right_array): #check indexes are not
larger than their lists, if they are the whole list has been compared

    #check which number is larger
    if left_array[left_index][sets] <= right_array[right_index][sets]:
        result.append(left_array[left_index]) #add left number to result array first if it is smaller
        left_index = left_index + 1 #add 1 to left index so that the next number will be compared
    else:
        result.append(right_array[right_index]) #add right number to result array first if it is
smaller
        right_index = right_index + 1 #add 1 to right index so that the next number will be
compared

#add the last item in either list to result list
result += left_array[left_index:]
result += right_array[right_index:]

return result #return merged list

```

```

login() #call login page

```

Implementation

Subroutines

1 hash

This subroutine hashes passwords by converting each letter in the string to its unicode value, and then divides it by 118147. The remainder of this calculation is the hashed value of the password.

2 login

This subroutine creates the login page, all entries to enter data into, and all buttons to call different subroutines available from this window.

3 login_submit

This subroutine takes the data entered into the user ID and password entries from the login window, and searches TblUsers for records with matching user IDs and passwords. If there are matching records, the login window is closed, all the data is pulled from the database and stored in the user_logged_in class as attributes, and the nutrition page is opened.

4 newuser_submit

This creates the new user window and all the entries to enter data, and all the buttons to call different subroutines

5 createaccount

This takes all data from the entries in the new user window, and makes a new record in the TblUsers table with all this data in it

6 calculate_age

This calculates how many years old the user from their date of birth

7 check_date_is_real

This checks if a date entered by the user is real, e.g. if the user enters 30/02/2023 as a date, it will return false as this is not a real date.

8 calorie_target_page

This turns exercise level into a physical activity ratio, calculates the users maintenance calories, and the amount of calories that they would have to eat to lose/gain 0.25 and 0.5kg per week. It then creates the calorie target window and all the buttons to select the users target intake.

9 inputcalstarget

This inserts the selected calorie target from the calorie target window into the users record in TblUsers

10 nutriton_and_workout_page

This creates the main window, and a notebook with the nutrition and workout frames in it, allowing the user to switch between the 2 pages like tabs

11 nutrition_page

This creates the nutrition frame in the notebook and creates all buttons that call other subroutines, entries, the treeview, and calls the function to draw the bar chart

12 add_meal

This takes the data entered by the user, stores it in a new record in TblMeals, inserts it into the treeview, adds the calories onto the calorie count, and updates the progress bar

13 bar_chart

This draws the bar chart to show calorie surplus/deficits using Tkinter canvas on the nutrition page

14 find_meal_page

This creates the find meals window, its treeview, all entries to enter dates into, and the search button to call find_meal

15 find_meal

This queries TblMeals to find meals entered on a selected date by the user logged in, and inserts them into the treeview on the find meals page

16 workout_page

This creates the workout frame in the notebook, and the treeviews, entries, and buttons to call other subroutines

17 add_workout

This takes data entered into the workout page by the user and inserts it into TblWorkouts and the treeview

18 find_workout

This searches TblWorkouts for exercises entered by the user logged in on a specified date and inserts them into a treeview on the workout page

19 sort

This takes the exercises found from the find_workout subroutine and recursively splits it into 2 arrays, then calls merge() to merge them

20 merge

This merges the arrays made by sort()

Examples of skills I have used

<u>Group</u>	<u>Model</u>	<u>Algorithm</u>	<u>Uses</u>
A	Complex data model in database (eg several interlinked tables)	Aggregate SQL queries	4 - createaccount 10 - nutrition_page 11 - add_meal 12 - bar chart
A	Complex data model in database (eg several interlinked tables)	Parameterised SQL queries	2 - login_submit 4 - createaccount 10 - nutrition_page 11 - add_meal 12 - bar_chart 14 - find_meal 15 - workout_page 16 - add_worout 17 - find_workout
A	Complex data model in database (eg several interlinked tables)	Cross-table SQL queries	10 - nutrition_page
A	Complex data model in database (eg several interlinked	Cross-table parameterized SQL queries	10 - nutrition_page

	tables)		
A	Complex scientific/mathematical/robotics/control/business model	Recursive algorithms	18 - sort
A	Complex scientific/mathematical/robotics/control/business model	Merge sort or similarly efficient sort	18 - sort 19 - merge
A	Hash tables, lists, stacks, queues, graphs, trees or structures of equivalent standard	Hashing	1 - hash This subroutine is called in: 3 - login_submit 5 - create_account
B	Simple data model in database (eg two or three interlinked tables)	Single table or non-parameterised SQL	I use 3 tables in my database: TblUsers TblMeals TblWorkout
B	Simple OOP model	Generation of objects based on simple OOP model	I use the user class and user_logged_in object to hold data about the user logged in
B	Simple scientific/mathematical /robotics/control/business model	Simple user defined algorithms (eg a range of mathematical/statistical calculations)	7 - calorie_target_page 6 - check_date_is_real

Testing

Test plan

<u>Test Number</u>	<u>Description</u>	<u>Data required</u>	<u>Related objective</u>	<u>Expected result</u>
1	Check a user cannot log in	User ID: 3	2.1	Error message will be

	without a password	Password: *blank*		displayed asking the user to enter a valid user ID and password
2	Check a user cannot log in without entering a user ID	User ID: *blank*	2.1	Error message will be displayed asking user to enter a valid user ID and password
3	Check a user cannot log in with an incorrect password	Password: a		
		User ID: 3	2.1	Error message will be displayed asking user to enter a valid user ID and password
		Password: b		
		Login button needs to be pressed		
4	Check that a user cannot log in with a fake user ID that has no account attached	User ID: 99	2.1	Error message will be displayed asking user to enter a valid user ID and password
		Password: a		
		Login button must be pressed		
5	Check a user can log in with correct credentials	User ID: 3	2.1	Login window will close, and nutrition page will open
		Password: a		
		Login button needs to be pressed		
6	Check you can open the create account page from the login	New user button in login window must be pressed	2.2	Login window will be destroyed and new user page

	menu			will open
7	Check user cannot create an account without entering a password	<p>New password: *blank*</p> <p>First name: Benjamin</p> <p>Last name: Franklin</p> <p>Gender: Male</p> <p>Date of birth: 06, 01, 1705</p> <p>Height (cm): 175</p> <p>Weight (kg): 100</p> <p>Exercise level: Couch potato/1</p>	2.2	Error message should be displayed asking user to enter a password
8	Check user cannot create an account without entering a First name	<p>New password: password</p> <p>First name: *blank*</p> <p>Last name: Franklin</p> <p>Gender: Male</p> <p>Date of birth: 06, 01, 1705</p> <p>Height (cm): 175</p> <p>Weight (kg): 100</p> <p>Exercise level: Couch potato/1</p>	2.2	Error message should be displayed asking user to enter a first name

9	Check user cannot create an account without entering a last name	<p>New password: password</p> <p>First name: Benjamin</p> <p>Last name: *blank*</p> <p>Gender: Male</p> <p>Date of birth: 06, 01, 1705</p> <p>Height (cm): 175</p> <p>Weight (kg): 100</p> <p>Exercise level: Couch potato/1</p>	2.2	Error message should be displayed asking user to enter a last name
10	Check user cannot create an account without selecting gender	<p>New password: password</p> <p>First name: Benjamin</p> <p>Last name: Franklin</p> <p>Gender: *blank*</p> <p>Date of birth: 06, 01, 1705</p> <p>Height (cm): 175</p> <p>Weight (kg): 100</p> <p>Exercise level: Couch potato/1</p>	2.2	Error message should be displayed asking user to select a gender
11	Check user cannot create an account	<p>New password: password</p>	2.2	Error message should be displayed

	without entering a date of birth	First name: Benjamin Last name: Franklin Gender: Male Date of birth: 06, 01, *blank* Height (cm): 175 Weight (kg): 100 Exercise level: Couch potato/1		asking user to enter a valid date of birth
12	Check user cannot create an account without entering a height	New password: password First name: Benjamin Last name: Franklin Gender: Male Date of birth: 06, 01, 1705 Height (cm): *blank* Weight (kg): 100 Exercise level: Couch potato/1	2.2	Error message should be displayed asking user to enter their height
13	Check user cannot create an account without entering their weight	New password: password First name: Benjamin	2.2	Error message should be displayed asking user to enter their weight

		Last name: Franklin Gender: Male Date of birth: 06, 01, 1705 Height (cm): 175 Weight (kg): *blank* Exercise level: Couch potato/1		
14	Check user cannot create an account without selecting an exercise level	New password: password First name: Benjamin Last name: Franklin Gender: Male Date of birth: 06, 01, 1705 Height (cm): 175 Weight (kg): 100 Exercise level: *blank*	2.2	Error message should be displayed asking user to select an exercise level
15	Check the user cannot create an account if they enter height as a non-numeric value	New password: password First name: Benjamin Last name: Franklin	2.2	Error box should appear asking user to enter a valid height

		Gender: Male Date of birth: 06, 01, 1705 Height (cm): A Weight (kg): 100 Exercise level: Couch potato/1		
16	Check the user cannot create an account if they enter weight as a non-numeric value	New password: password First name: Benjamin Last name: Franklin Gender: Male Date of birth: 06, 01, 1705 Height (cm): 175 Weight (kg): A Exercise level: Couch potato/1	2.2	Error box should appear asking user to enter a valid weight
17	Check whether the user can register if they enter details correctly	New password: Georg3Washy1732 First name: Benjamin Last name: Franklin Gender: Male	2.2	New user page should be destroyed and calorie target page should load

		<p>Date of birth: 06, 01, 2005</p> <p>Height (cm): 175</p> <p>Weight (kg): 100</p> <p>Exercise level: Couch potato/1</p>		
18	Check calorie target page displays the correct calorie numbers for each weight loss goal on all 5 buttons	Users weight (100kg), height (175cm), age (17), and physical activity ratio (1) are all needed to calculate calorie target for each weight loss/gain goal	4.1 4.2	Each button on calorie target page should display the correct amount of calories required per day to achieve the displayed goals (+0.5kg per week, +0.25kg per week, maintain weight, -0.25kg per week, -0.5 kg per week)
19	Check users can choose a calorie target	Button for desired calorie intake must be pressed	4.2	After pressing the button with desired calorie intake on, the user should progress to the nutrition page, where their calorie target is displayed.
20	Check nutrition page displays calorie target	Calorie target	4.2	Nutrition page should say "Target: x" (x being the target) below calorie intake for today
21	Check nutrition page displays	Total calories eaten today by	5.2	Nutrition page should correctly

	total calories eaten today in treeview	one user		display the total sum of the calories in all meals eaten by a user in 1 day
22	Check progress bar in nutrition page correctly displays how close to reaching their calorie intake goal a user is	Users calorie intake goal Users total calories consumed in a day	5.2	The amount of progress on the progress bar should match the proportion of their calorie goal a user has consumed in the past day. E.g. if a users goal is 2500 calories, and they have eaten 2000, the bar should be 80% full
23	Check all meals the user has entered into the app in the past day are correctly displayed in the treeview on nutrition page	All meals eaten by a user within a day	5.5	When a user logs in and the nutrition page loads, all meals they have entered in the past day should be displayed in the treeview along with the time it was eaten and the calories it contained.
24	Check the user cannot enter a meal into the app without entering any calories	Meal description: Toast Time: 21:00 Calories *blank* Add button must be pressed	5.1	When the user tries to enter a meal into the app without entering calories, a message should appear asking the user to enter calories

25	Check the user cannot enter a meal into the app if the calories box contains non numeric characters	Meal: toast Time: 21:00 Calories A Add button must be pressed	5.1	An error message should appear telling the user "calories" must be a whole number
26	Check the user can enter a meal into the app	Meal desc: test Time: 23:00 Calories: 1000 Add button must be pressed	5.1	After pressing add button, the details entered of the meal should appear on the treeview, the calories consumed counter should update and add the calories of the meal just entered, and the meal should appear in the database
27	Check the user can access the workout page	The "Workout tracker" tab will have to be pressed	3	Upon pressing the tab, the workout tracker page should open
28	Check all exercises entered into the app today appear in the treeview on workout page		3.1	When the workout page opens, the treeview on the screen should contain all exercises the user has entered in the past day
29	Check the user cannot add an exercise if the name of the	Exercise: *blank* Sets:	3.1	A message should appear asking the user to enter the

	exercise is left blank	<p>5</p> <p>Reps: 5</p> <p>Weight: 50kg</p> <p>Comments: *blank* (this is not an essential field)</p>		exercise name
30	Check the user cannot add an exercise if sets performed has not been entered	<p>Exercise: Pull ups</p> <p>Sets: *blank*</p> <p>Reps: 17</p> <p>Weight: body</p> <p>Comments: *blank* (this is not an essential field)</p>	3.1	A message box should appear asking the user to enter sets performed
31	Check the user cannot add an exercise if reps performed is left blank	<p>Exercise: Pull ups</p> <p>Sets: 3</p> <p>Reps: *blank*</p> <p>Weight: body</p> <p>Comments: *blank* (this is not an essential field)</p>	3.1	A message should appear asking the user to enter reps performed
32	Check the user cannot add an exercise if sets performed is	<p>Exercise: Pull ups</p> <p>Sets:</p>	3.1	An error message should appear telling the user

	non numeric	A Reps: 10 Weight: body Comments: *blank* (this is not an essential field)		to ensure sets and reps are whole numbers
33	Check the user cannot add an exercise if reps performed is a non numeric value	Exercise: Pull ups Sets: 5 Reps: A Weight: body Comments: *blank* (this is not an essential field)	3.1	An error message should appear telling the user to ensure sets and reps are whole numbers
34	Check the user can add an exercise to the treeview and database	Exercise: Pull ups Sets: 5 Reps: 10 Weight: Body Comments: *blank* (this is not an essential field)	3.1	The details of the exercise should appear in the treeview after the add button is pressed
35	Check the bar chart on nutrition page shows calories	UserID: 3 Total calories	6.2	The bar chart should show when the user had a calorie

	deficits and surpluses over the last 5 days correctly	from last 5 days for user		surplus or deficit over the last 5 days, with the bar being below the maintenance calorie line for a deficit, and above for a surplus
36	Check the bar chart displays the size of the calorie deficit/surplus on each day	UserID: 3 Total calories from last 5 days for user	6	Underneath/above each bar on the bar chart, there should be a number displaying how much of a deficit or surplus the user was in each day
37	Check the bar chart shows when no meals have been entered in a day	UserID: 3 Total calories from last 5 days for user	6	When a user has not entered any meals in the one of the past 5 days, it should say "no meals recorded" in place of this days bar on the bar chart
38	Check the users user ID is displayed at the top of the window when logged in	UserID: 3	2	At the top of the window, it should say: "User ID:", and then the ID of the user logged in
39	Check a colon is put between hours and minutes when entering a meal into the treeview and database	Meal: dinner Time hours: 19 Time minutes: 00 Calories:	5.1	A colon should appear between the hours and minutes when entered, making it "19:00"

		1200		
40	Check users can change their calorie target after setting it by pressing the target button on nutrition page	Calorie target button will need to be pressed	4	After pressing button, calorie target page should open
41	Check users can load the find meals page	Search meals button must be pressed	5.3	After pressing the button, a separate window should load to search for meals
42	Check users cannot search meals with an invalid date	Day: 30 Month: 02 Year: 2023 Search button must be pressed	5.3	When this invalid date is entered, an error box should appear asking the user to enter a valid date
43	Check users cannot search for a meal if day is blank	Day: *blank* Month: 02 Year: 2023 Search button must be pressed	5.3	An error box should appear asking the user to enter a valid date
44	Check users cannot search for a meal if month is blank	Day: 01 Month: *blank* Year:	5.3	An error box should appear asking the user to enter a valid date

		2023 Search button must be pressed		
45	Check users cannot search for a meal if year is blank	Day: 01 Month: 02 Year: *blank* Search button must be pressed	5.3	An error box should appear asking the user to enter a valid date
46	Check users cannot search for meal if non numeric characters have been entered for day	Day: A Month: 02 Year: 2023 Search button must be pressed	5.3	An error box should appear asking the user to enter a valid date
47	Check users cannot search for meal if non numeric characters have been entered for month	Day: 01 Month: A Year: 2023 Search button must be pressed	5.3	An error box should appear asking the user to enter a valid date
48	Check users cannot search for meal if non numeric characters have been entered	Day: 01 Month: 02	5.3	An error box should appear asking the user to enter a valid date

	for year	Year: A Search button must be pressed		
49	Check users can enter a date on find meals page and see all meals entered into the app that day	Search button must be pressed	5.3	The treeview should fill with all meals eaten that day, the time they were eaten, and the calories each meal contained
50	Check a user cannot create an account if the date of birth entered isn't a real date	New password: password First name: Benjamin Last name: Franklin Gender: Male Date of birth: 30, 02, 2000 Height (cm): 175 Weight (kg): A Exercise level: Couch potato/1	2.2	An error box should appear asking the user to enter a valid date
51	Check a user cannot search for a workout if date entered isn't a real date	Day: 30 Month: 02 Year: 2023	3.2	An error box should appear asking the user to enter a valid date

Test results

<u>Test Number</u>	<u>Description</u>	<u>Data required</u>	<u>Related objective</u>	<u>Expected result</u>	<u>Result</u>
1	Check a user cannot log in without a password	User ID: Test_userID Password: *blank* Login button must be pressed	2.1	Error message will be displayed asking the user to enter a password	An error message was displayed saying "Please enter a password"
2	Check a user cannot log in without entering a user ID	User ID: *blank* Password: Test_password	2.1	Error message will be displayed asking user to enter a user ID	An error message appeared saying "please enter a user ID"
3	Check a user cannot log in with an incorrect password	User ID: Test_userID Password: Wrong_password Login button needs to be pressed	2.1	Error message will be displayed saying username or password is incorrect	An error message was displayed saying "Username or password is incorrect"
4	Check that a user cannot log in with a fake user ID that has no account attached	User ID: Fake_userID Password: Password Login button must be pressed	2.1	Error message will be displayed saying username or password is incorrect	An error message was displayed saying "Username or password is incorrect"
5	Check a user can log in with correct credentials	User ID: Test_userID Password: Password	2.1	Login window will close, and nutrition page will	The login window closed as the nutrition page

		Login button needs to be pressed		open	opened, as expected
6	Check you can open the create account page from the login menu	New user button in login window must be pressed	2.2	Login window will be destroyed and new user page will open	The login window closed and was replaced by the new user window as expected
7	Check user cannot create an account without entering a password	New password: *blank* First name: Benjamin Last name: Franklin Gender: Male Date of birth: 06, 01, 1705 Height (cm): 175 Weight (kg): 100 Exercise level: Couch potato/1	2.2	Error message should be displayed asking user to enter a password	As expected, an error box was displayed saying "Please enter a password"
8	Check user cannot create an account without entering a First name	New password: password First name: *blank* Last name: Franklin	2.2	Error message should be displayed asking user to enter a first name	An error message was correctly displayed saying "Please enter your first name"

		Gender: Male Date of birth: 06, 01, 1705 Height (cm): 175 Weight (kg): 100 Exercise level: Couch potato/1			
9	Check user cannot create an account without entering a last name	New password: password First name: Benjamin Last name: *blank* Gender: Male Date of birth: 06, 01, 1705 Height (cm): 175 Weight (kg): 100 Exercise level: Couch potato/1	2.2	Error message should be displayed asking user to enter a last name	An error message was correctly displayed saying "Please enter your last name"
10	Check user cannot create an account without selecting gender	New password: password First name: Benjamin	2.2	Error message should be displayed asking user to select a gender	An error box was correctly displayed saying "please select your gender"

		Last name: Franklin Gender: *blank* Date of birth: 06, 01, 1705 Height (cm): 175 Weight (kg): 100 Exercise level: Couch potato/1			
11	Check user cannot create an account without entering a date of birth	New password: password First name: Benjamin Last name: Franklin Gender: Male Date of birth: 06, 01, *blank* Height (cm): 175 Weight (kg): 100 Exercise level: Couch potato/1	2.2	Error message should be displayed asking user to enter a date of birth	An error box was correctly displayed saying "Please enter your date of birth"

12	Check user cannot create an account without entering a height	<p>New password: password</p> <p>First name: Benjamin</p> <p>Last name: Franklin</p> <p>Gender: Male</p> <p>Date of birth: 06, 01, 1705</p> <p>Height (cm): *blank*</p> <p>Weight (kg): 100</p> <p>Exercise level: Couch potato/1</p>	2.2	Error message should be displayed asking user to enter their height	An error box correctly appeared saying "please enter your height"
13	Check user cannot create an account without entering their weight	<p>New password: password</p> <p>First name: Benjamin</p> <p>Last name: Franklin</p> <p>Gender: Male</p> <p>Date of birth: 06, 01, 1705</p> <p>Height (cm): 175</p> <p>Weight (kg): *blank*</p> <p>Exercise</p>	2.2	Error message should be displayed asking user to enter their weight	An error box was correctly displayed saying "Please enter your weight"

		level: Couch potato/1			
14	Check user cannot create an account without selecting an exercise level	New password: password First name: Benjamin Last name: Franklin Gender: Male Date of birth: 06, 01, 1705 Height (cm): 175 Weight (kg): 100 Exercise level: *blank*	2.2	Error message should be displayed asking user to select an exercise level	An error box was correctly displayed saying "Please enter a valid exercise level"
15	Check the user cannot create an account if they enter height as a non-numeric value	New password: password First name: Benjamin Last name: Franklin Gender: Male Date of birth: 06, 01, 1705 Height (cm): A Weight (kg): 100	2.2	Error box should appear asking user to enter a valid height	An error box was correctly displayed saying "Please enter a valid height"

		Exercise level: Couch potato/1			
16	Check the user cannot create an account if they enter weight as a non-numeric value	<p>New password: password</p> <p>First name: Benjamin</p> <p>Last name: Franklin</p> <p>Gender: Male</p> <p>Date of birth: 06, 01, 1705</p> <p>Height (cm): 175</p> <p>Weight (kg): A</p> <p>Exercise level: Couch potato/1</p>	2.2	Error box should appear asking user to enter a valid weight	An error box was correctly displayed saying "please enter a valid weight"
17	Check whether the user can register if they enter details correctly	<p>New password: password</p> <p>First name: Benjamin</p> <p>Last name: Franklin</p> <p>Gender: Male</p> <p>Date of birth: 06, 01, 2005</p> <p>Height (cm): 175</p>	2.2	New user page should be destroyed and calorie target page should load	Calorie target page loaded and new user page was destroyed as expected

		Weight (kg): 100 Exercise level: Couch potato/1			
18	Check calorie target page displays the correct calorie numbers for each weight loss goal on all 5 buttons	Users weight (100kg), height (175cm), age (17), and physical activity ratio (1) are all needed to calculate calorie target for each weight loss/gain goal	4.1 4.2	Each button on calorie target page should display the correct amount of calories required per day to achieve the displayed goals (+0.5kg per week, +0.25kg per week, maintain weight, -0.25kg per week, -0.5 kg per week)	The calorie target page displayed the right amount of calories for each option
19	Check users can choose a calorie target	Button for desired calorie intake must be pressed	4.2	After pressing the button with desired calorie intake on, the user should progress to the nutrition page, where their calorie target is displayed.	
20	Check nutrition page displays calorie target	Calorie target	4.2	Nutrition page should say "Target: x" (x being the target)	The target was correctly displayed as 3500 kcal, which is the

				below calorie intake for today	target intake of the user logged in
21	Check nutrition page displays total calories eaten today in treeview	Total calories eaten today by one user	5.2	Nutrition page should correctly display the total sum of the calories in all meals eaten by a user in 1 day	The nutrition page correctly displayed that the user had eaten 3200 kcal, which was the sum of all meals entered today.
22	Check progress bar in nutrition page correctly displays how close to reaching their calorie intake goal a user is	Users calorie intake goal Users total calories consumed in a day	5.2	The amount of progress on the progress bar should match the proportion of their calorie goal a user has consumed in the past day. E.g. if a users goal is 2500 calories, and they have eaten 2000, the bar should be 80% full	The total calories eaten today in this example is 3200, this is around 91% of the users target (3500), the bar is 91% full, as expected.
23	Check all meals the user has entered into the app in the past day are correctly displayed in the treeview on nutrition page	All meals eaten by a user within a day	5.5	When a user logs in and the nutrition page loads, all meals they have entered in the past day should be displayed in the treeview along with	All meals and their information entered by the user were correctly displayed in the treeview on nutrition page

				the time it was eaten and the calories it contained.	
24	Check the user cannot enter a meal into the app without entering any calories	Meal description: Test_meal Time: Test_time Calories *blank* Add button must be pressed	5.1	When the user tries to enter a meal into the app without entering calories, a message should appear asking the user to enter calories	As expected, an error box was displayed saying "Please enter calories"
25	Check the user cannot enter a meal into the app if the calories box contains non numeric characters	Meal: Test_meal Time: Test_time Calories False_calories Add button must be pressed	5.1	An error message should appear telling the user "calories consumed must be a whole number"	An error box was correctly displayed saying "Calories consumed must be a whole number"
26	Check the user can enter a meal into the app	Meal desc: Test_meal Time: Test_time Calories: Test_calories Add button must be pressed	5.1	After pressing add button, the details entered of the meal should appear on the treeview, the calories consumed counter should update and add the calories of the meal just	As expected, the meal and its details appeared in the treeview and TblMeals after pressing the add button.

				entered, and the meal should appear in the database	
27	Check the user can access the workout page	The "Workout tracker" tab will have to be pressed	3	Upon pressing the tab, the workout tracker page should open	As expected, the workout page opened
28	Check all exercises entered into the app today appear in the treeview on workout page		3.1	When the workout page opens, the treeview on the screen should contain all exercises the user has entered in the past day	After opening the workout page, all exercises done today were correctly displayed in the treeview
29	Check the user cannot add an exercise if the name of the exercise is left blank	Exercise: *blank* Sets: Test_sets Reps: Test_reps Weight: Test weight Comments: *blank* (this is not an essential field)	3.1	A message should appear asking the user to enter the exercise name	An error box was correctly displayed saying "Enter exercise name"
30	Check the user cannot add an exercise if sets performed has not been	Exercise: Test_exercise Sets: *blank*	3.1	A message box should appear asking the user to enter sets performed	An error box was correctly displayed saying "Enter sets performed"

	entered	Reps: Test_reps Weight: Test weight Comments: *blank* (this is not an essential field)			
31	Check the user cannot add an exercise if reps performed is left blank	Exercise: Test_exercise Sets: Test_sets Reps: *blank* Weight: Test weight Comments: *blank* (this is not an essential field)	3.1	A message should appear asking the user to enter reps performed	An error box was correctly displayed saying "Enter reps performed"
32	Check the user cannot add an exercise if sets performed is non numeric	Exercise: Test_exercise Sets: Non_numeric_sets Reps: Test_reps Weight: Test weight Comments: *blank* (this is not an essential field)	3.1	An error message should appear telling the user to ensure sets and reps are numbers	An error box was correctly displayed saying "make sure sets and reps are numbers!"

33	Check the user cannot add an exercise if reps performed is a non numeric value	<p>Exercise: Test_exercise</p> <p>Sets: Test_sets</p> <p>Reps: Non_numeric_reps</p> <p>Weight: Test weight</p> <p>Comments: *blank* (this is not an essential field)</p>	3.1	An error message should appear telling the user to ensure sets and reps are numbers	An error box was correctly displayed saying "make sure sets and reps are numbers!"
34	Check the user can add an exercise to the treeview and database	<p>Exercise: Test_exercise</p> <p>Sets: Test_sets</p> <p>Reps: Test_reps</p> <p>Weight: Test weight</p> <p>Comments: *blank* (this is not an essential field)</p>	3.1	The details of the exercise should appear in the treeview after the add button is pressed	As expected, after pressing the add button, the details of the exercise appeared in the treeview
35	Check the bar chart on nutrition page shows calories deficits and surpluses over the last 5 days correctly	<p>UserID: Test_userID</p> <p>Total calories from last 5 days for user</p>	6.2	The bar chart should show when the user had a calorie surplus or deficit over the last 5 days, with the bar being below the maintenance	When I logged in and opened the nutrition page, the bar chart was drawn and correctly showed all of the calorie surpluses and deficits I

				calorie line for a deficit, and above for a surplus	had over the past 5 days.
36	Check the bar chart displays the size of the calorie deficit/surplus on each day	UserID: Test_userID Total calories from last 5 days for user	6	Underneath/above each bar on the bar chart, there should be a number displaying how much of a deficit or surplus the user was in each day	The exact number of calories over/under maintenance the user has been in the past 5 days are correctly displayed on the bar chart
37	Check the bar chart shows when no meals have been entered in a day	UserID: Test_userID Total calories from last 5 days for user	6	When a user has not entered any meals in the one of the past 5 days, it should say "no meals recorded" in place of this days bar on the bar chart	As expected, the bar chart correctly says "no meals recorded" for days when no meals were recorded by the user
38	Check the users user ID is displayed at the top of the window when logged in	UserID: Test_userID	2	At the top of the window, it should say: "User ID:", and then the ID of the user logged in	After logging in, the user ID of the user logged in was correctly displayed at the top of the screen as the window title
39	Check a colon is put between hours and minutes when entering a meal into the	Meal: Test_meal Time hours: 00 Time minutes:	5.1	A colon should appear between the hours and minutes when entered,	As expected, a colon was put between the hours and minutes in the time eaten when a meal was

	treeview and database	00 Calories: 0		making it "00:00"	entered into the app.
40	Check users can change their calorie target after setting it by pressing the target button on nutrition page	Calorie target button will need to be pressed	4	After pressing button, calorie target page should open	As expected, the calorie target page was displayed upon pressing the target button on nutrition page
41	Check users can load the find meals page	Search meals button must be pressed	5.3	After pressing the button, a separate window should load to search for meals	After pressing the search meals button, the find meals page opened as expected
42	Check users cannot search meals with an invalid date	Day: 30 Month: 02 Year: 2023 Search button must be pressed	5.3	When this invalid date is entered, an error box should appear asking the user to enter a valid date	As expected, an error box was displayed saying "Please enter a valid date"
43	Check users cannot search for a meal if day is blank	Day: *blank* Month: 02 Year: 2023 Search button must be pressed	5.3	An error box should appear asking the user to enter a valid date	As expected, an error box was displayed saying "Please enter a valid date"

44	Check users cannot search for a meal if month is blank	Day: 01 Month: *blank* Year: 2023 Search button must be pressed	5.3	An error box should appear asking the user to enter a valid date	As expected, an error box was displayed saying "Please enter a valid date"
45	Check users cannot search for a meal if year is blank	Day: 01 Month: 02 Year: *blank* Search button must be pressed	5.3	An error box should appear asking the user to enter a valid date	As expected, an error box was displayed saying "Please enter a valid date"
46	Check users cannot search for meal if non numeric characters have been entered for day	Day: A Month: 02 Year: 2023 Search button must be pressed	5.3	An error box should appear asking the user to enter a valid date	As expected, an error box was displayed saying "Please enter a valid date"
47	Check users cannot search for meal if non numeric characters have been entered for month	Day: 01 Month: A Year: 2023 Search button must be pressed	5.3	An error box should appear asking the user to enter a valid date	As expected, an error box was displayed saying "Please enter a valid date"

48	Check users cannot search for meal if non numeric characters have been entered for year	Day: 01 Month: 02 Year: A Search button must be pressed	5.3	An error box should appear asking the user to enter a valid date	As expected, an error box was displayed saying "Please enter a valid date"
49	Check users can enter a date on find meals page and see all meals entered into the app that day	Search button must be pressed	5.3	The treeview should fill with all meals eaten that day, the time they were eaten, and the calories each meal contained	After entering a date, all meals entered into the app on this date appear in the treeview
50	Check a user cannot create an account if the date of birth entered isn't a real date	New password: password First name: Benjamin Last name: Franklin Gender: Male Date of birth: 30, 02, 2000 Height (cm): 175 Weight (kg): A Exercise level: Couch potato/1	2.2	An error box should appear asking the user to enter a valid date	As expected, an error box appeared asking the user to enter a valid date

51	Check a user cannot search for a workout if date entered isn't a real date	Day: 30 Month: 02 Year: 2023	3.2	An error box should appear asking the user to enter a valid date	As expected, an error box was displayed asking the user to enter a valid date
52	Check a user can log out	Log out button must be pressed	2.3	After pressing the log out button, the main window should close and the login window should load	As expected, the login window loaded after pressing the log out button
53	Check a user cannot search for a workout without entering day	Day: *blank* Month: 02 Year: 2023	3.2	After pressing the search button, an error box should appear telling the user to enter a valid date	As expected, an error box was displayed asking the user to enter a valid date
54	Check a user cannot search for a workout without entering month	Day: 28 Month: *blank* Year: 2023	3.2	After pressing the search button, an error box should appear telling the user to enter a valid date	As expected, an error box was displayed asking the user to enter a valid date
55	Check a user cannot search for a workout without entering year	Day: 28 Month: 02 Year: *blank*	3.2	After pressing the search button, an error box should appear telling the user to enter a valid date	As expected, an error box was displayed asking the user to enter a valid date

56	Check a user can search for a workout when entering a valid date	Day: 10 Month: 5 Year: 2023	3.2	After pressing the search button, a record should be inserted into the database called "test exercise"	As expected, an exercise called "test exercise" was inserted into the database
----	--	--	-----	--	--

Test evidence

Test 1

As expected, an error box appears asking the user to enter a password



Test 2

As expected, an error box appears asking the user to enter a user ID



Test 3

As expected, an error box appears telling the user that the username or password is incorrect



Test 4

As expected, an error box appears telling the user that the username or password is incorrect



Test 5

As expected, after pressing submit the login window closed and the nutrition page opened.



Test 6

As can be seen, the login window closed and was replaced by the new user window as expected when I pressed the new user button, as expected

The screenshot shows a 'Create Account' dialog box with a title bar containing a minimize button, a maximize button, and a close button. The dialog is titled 'Create Account' and contains a section titled 'New User'. The form includes the following fields: 'New Password:' (empty), 'First Name:' (empty), 'Last Name:' (empty), 'Gender:' (a dropdown menu), 'Date of Birth:' (three separate input boxes), 'Height (cm):' (empty), 'weight (kg):' (empty), and 'Exercise level:' (a dropdown menu). A 'Submit' button is located at the bottom right of the form.

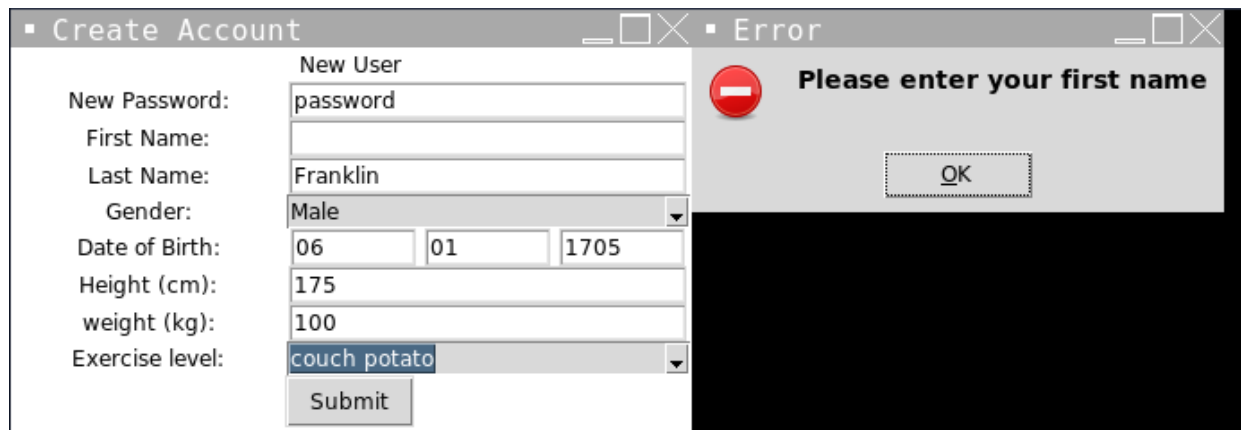
Test 7

As can be seen, an error box appears asking the user to enter a password after the submit button is pressed, as expected

This screenshot shows the 'Create Account' dialog box with the 'New User' section filled out. The 'New Password' field is empty. The other fields are: 'First Name:' Benjamin, 'Last Name:' Franklin, 'Gender:' Male, 'Date of Birth:' 06/01/1705, 'Height (cm):' 175, 'weight (kg):' 100, and 'Exercise level:' couch potato. The 'Submit' button is visible. An error dialog box is overlaid on the right side of the 'Create Account' dialog. The error dialog has a title bar with a close button and is titled 'Error'. It contains a red circle with a white minus sign icon and the text 'Please enter a password'. An 'OK' button is at the bottom of the error dialog.

Test 8

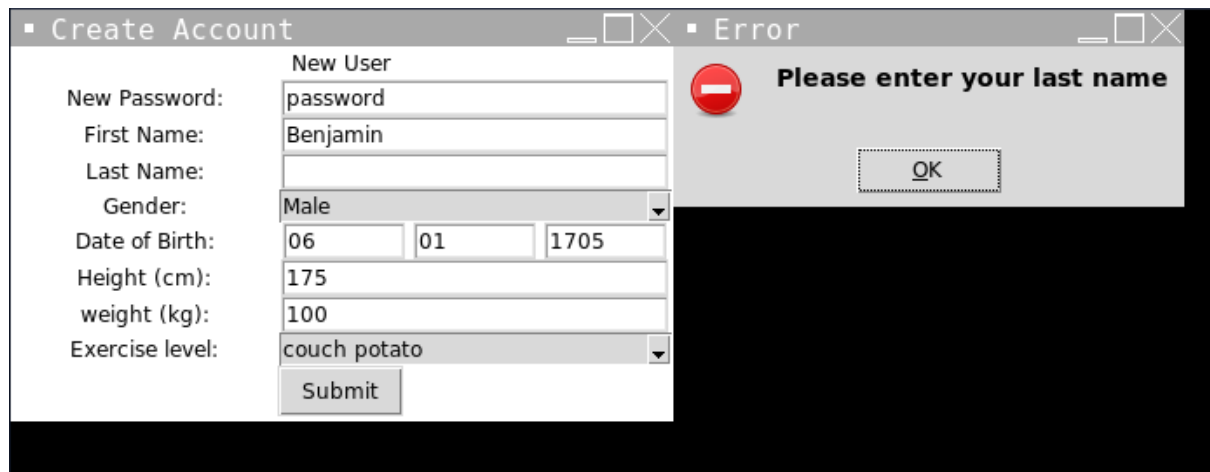
As can be seen, an error box appears asking the user to enter a first name after the submit button is pressed, as expected



The screenshot shows a 'Create Account' window with a 'New User' form and an 'Error' dialog box. The form fields are: New Password (password), First Name (empty), Last Name (Franklin), Gender (Male), Date of Birth (06/01/1705), Height (cm) (175), weight (kg) (100), and Exercise level (couch potato). The Submit button is visible. The error dialog box displays a red circle with a white minus sign and the text 'Please enter your first name' with an OK button.

Test 9

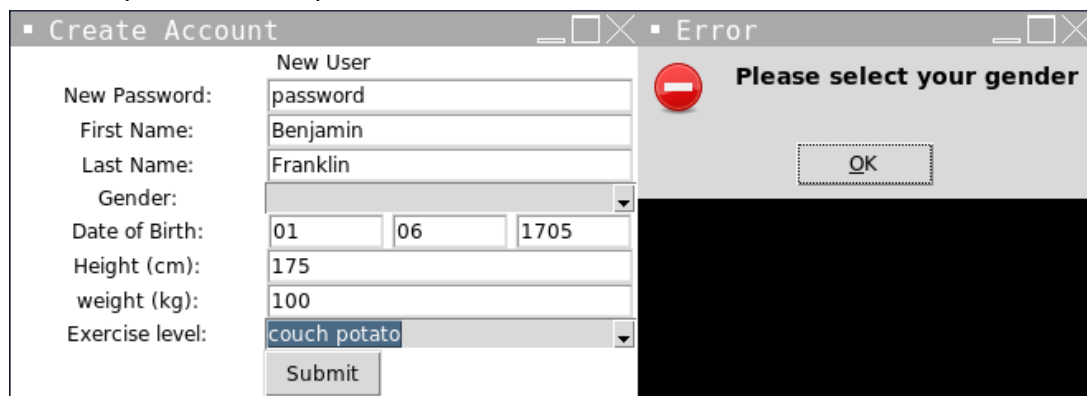
As can be seen, an error box appears asking the user to enter a last name after the submit button is pressed, as expected



The screenshot shows a 'Create Account' window with a 'New User' form and an 'Error' dialog box. The form fields are: New Password (password), First Name (Benjamin), Last Name (empty), Gender (Male), Date of Birth (06/01/1705), Height (cm) (175), weight (kg) (100), and Exercise level (couch potato). The Submit button is visible. The error dialog box displays a red circle with a white minus sign and the text 'Please enter your last name' with an OK button.

Test 10

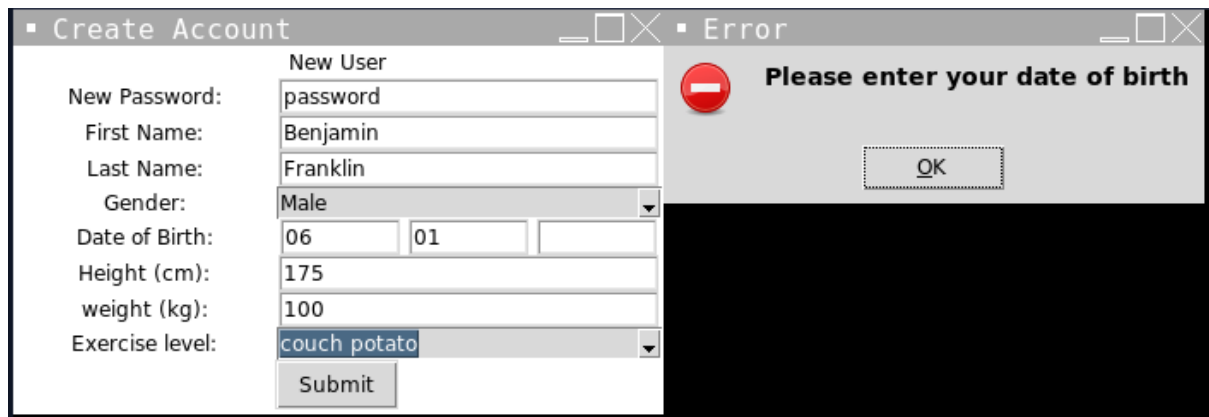
As can be seen, an error box appears asking the user to select their gender after the submit button is pressed, as expected



The screenshot shows a 'Create Account' window with a 'New User' form and an 'Error' dialog box. The form fields are: New Password (password), First Name (Benjamin), Last Name (Franklin), Gender (empty), Date of Birth (01/06/1705), Height (cm) (175), weight (kg) (100), and Exercise level (couch potato). The Submit button is visible. The error dialog box displays a red circle with a white minus sign and the text 'Please select your gender' with an OK button.

Test 11

As can be seen, an error box appears asking the user to enter their date of birth after the submit button is pressed, as expected



The screenshot shows a 'Create Account' window with a 'New User' form and an 'Error' dialog box. The form fields are: New Password: password, First Name: Benjamin, Last Name: Franklin, Gender: Male, Date of Birth: 06 / 01 / , Height (cm): 175, weight (kg): 100, Exercise level: couch potato. The Submit button is visible. The Error dialog box has a red circle icon and the text 'Please enter your date of birth' with an OK button.

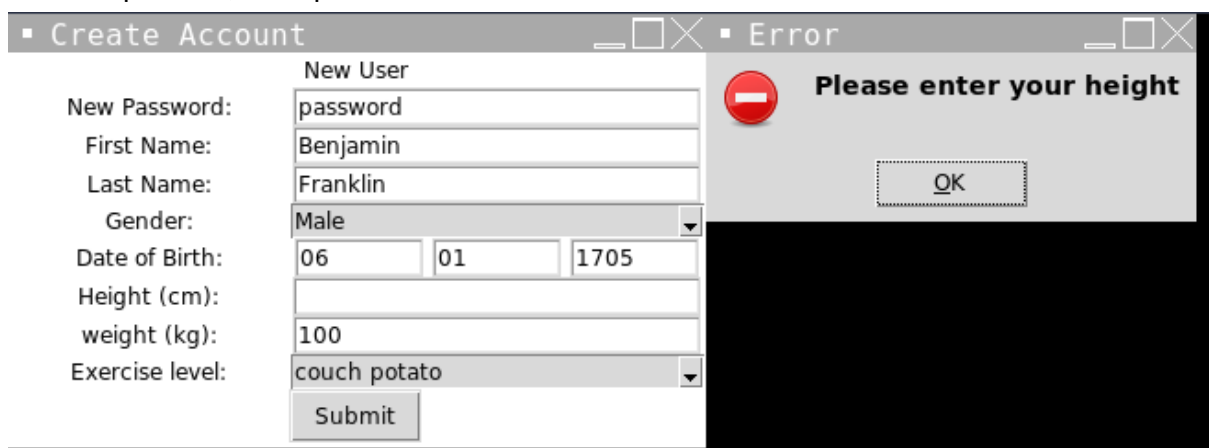
New User	
New Password:	password
First Name:	Benjamin
Last Name:	Franklin
Gender:	Male
Date of Birth:	06 / 01 /
Height (cm):	175
weight (kg):	100
Exercise level:	couch potato

Submit

Error
Please enter your date of birth
OK

Test 12

As can be seen, an error box appears asking the user to enter their height after the submit button is pressed, as expected



The screenshot shows a 'Create Account' window with a 'New User' form and an 'Error' dialog box. The form fields are: New Password: password, First Name: Benjamin, Last Name: Franklin, Gender: Male, Date of Birth: 06 / 01 / 1705, Height (cm): , weight (kg): 100, Exercise level: couch potato. The Submit button is visible. The Error dialog box has a red circle icon and the text 'Please enter your height' with an OK button.

New User	
New Password:	password
First Name:	Benjamin
Last Name:	Franklin
Gender:	Male
Date of Birth:	06 / 01 / 1705
Height (cm):	
weight (kg):	100
Exercise level:	couch potato

Submit

Error
Please enter your height
OK

Test 13

As can be seen, an error box appears asking the user to enter their weight after the submit button is pressed, as expected.

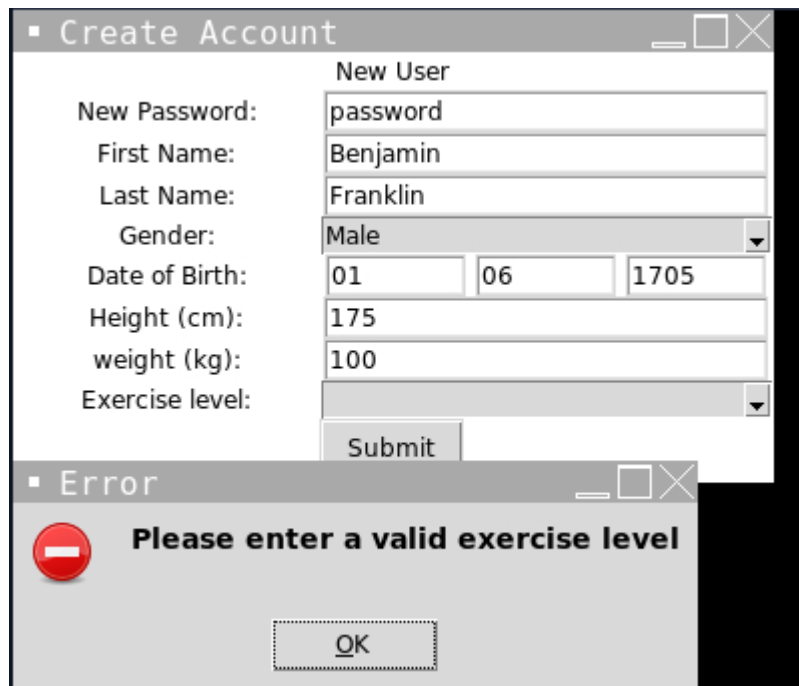
The image shows a 'Create Account' window and an 'Error' dialog box. The 'Create Account' window has a title bar with a minus, maximize, and close button. It contains the following fields: 'New Password:' with value 'password', 'First Name:' with value 'Benjamin', 'Last Name:' with value 'Franklin', 'Gender:' with a dropdown menu showing 'Male', 'Date of Birth:' with three input boxes containing '06', '01', and '1705', 'Height (cm):' with value '175', 'weight (kg):' with an empty input box, and 'Exercise level:' with a dropdown menu showing 'couch potato'. A 'Submit' button is at the bottom. The 'Error' dialog box has a title bar with a minus, maximize, and close button. It features a red circle with a white minus sign icon, the text 'Please enter your weight', and an 'OK' button.

Create Account	
New Password:	password
First Name:	Benjamin
Last Name:	Franklin
Gender:	Male
Date of Birth:	06 01 1705
Height (cm):	175
weight (kg):	
Exercise level:	couch potato
<input type="button" value="Submit"/>	

Please enter your weight

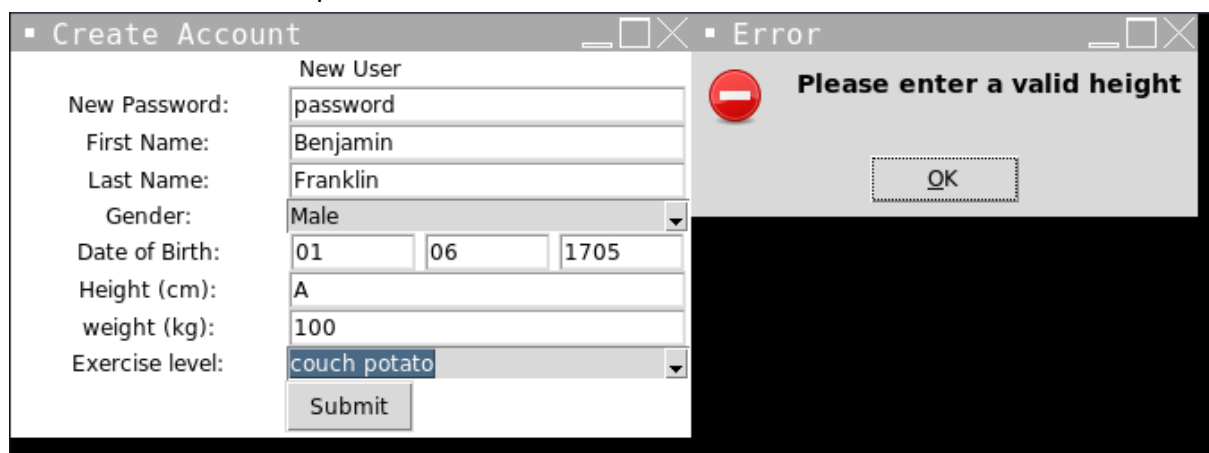
Test 14

As can be seen, an error box appears asking the user to enter a valid exercise level after pressing the submit button, as expected.



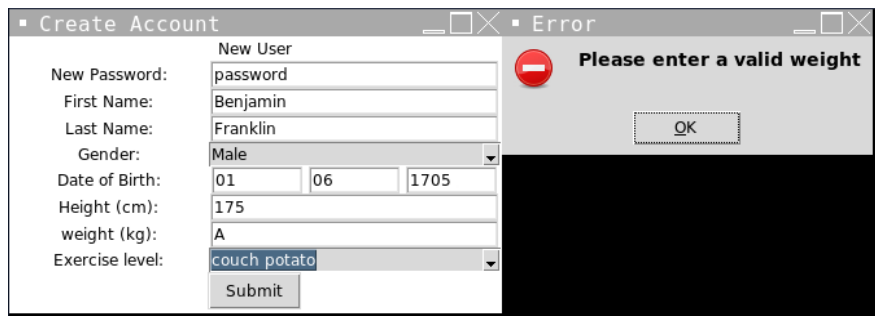
Test 15

As can be seen, an error box appears asking the user to enter a valid height after pressing the submit button, as expected



Test 16

As can be seen an error box appears asking the user to enter a valid weight after the pressing the submit button, as expected



Test 17

As can be seen, after entering correct details and pressing the submit button, the new user page was destroyed and the calorie target page loaded as expected



Test 18

As can be seen, the calorie target page displays the right amount of calories for each option for this user

■ User ID: 20

Select target calorie intake

+0.5kg per month	+0.25kg per month	maintain weight	-0.25kg per month	-0.5kg per month
2504	2254	2004	1754	1504

Test 19 & 20

As can be seen, after choosing an option, the nutrition page opened displaying my previously chosen option (2004 kcal) as the target intake

User ID: 20

Nutrition TrackerWorkout TrackerSocial

Calories Consumed:

None

Target: 2004kcal

Meal	Time	Calories
------	------	----------

Add a Meal

MealTimeCalories

0000

Add

maintenance calories

search meals

No meals recorded

No meals recorded

No meals recorded

No meals recorded

No meals recorded

Test 21

As can be seen, the total calories consumed is displayed as 3200 kcal, this is correct as the sum of all meals in the treeview is 3200.

User ID: 3

Nutrition TrackerWorkout Tracker

Calories Consumed:

3200

Target: 3500kcal

Meal	Time	Calories
cereal	09:00	600
lunch	12:45	1100
crisps	15:00	300
dinner	19:00	1200

Add a Meal

Meal

Time

Calories

1900

Add

maintenance calories

No meals recorded

No meals recorded

No meals recorded

No meals recorded

+400kcal

search meals

Test 22

As can be seen, the progress bar correctly displays how close the user is to reaching their calorie intake goal (91%).

User ID: 3

Nutrition TrackerWorkout Tracker

Calories Consumed:

3200

Target: 3500kcal

Meal	Time	Calories
cereal	09:00	600
lunch	12:45	1100
crisps	15:00	300
dinner	19:00	1200

Add a Meal

Meal

Time

Calories

1900

Add

maintenance calories

No meals recorded

No meals recorded

No meals recorded

No meals recorded

+400kcal

search meals

Test 23

As can be seen, the treeview displays all meals entered by the user today and their details.

User ID: 3

Nutrition TrackerWorkout Tracker

Calories Consumed:

3200

Target: 3500kcal

Meal	Time	Calories
cereal	09:00	600
lunch	12:45	1100
crisps	15:00	300
dinner	19:00	1200

Add a Meal

Meal

Time

Calories

1900

Add

maintenance calories

No meals recorded

No meals recorded

No meals recorded

No meals recorded

+400kcal

search meals

Test 24

As can be seen, an error box was correctly displayed asking the user to enter the calories in the meal, as expected

The screenshot shows a web application window titled "User ID: 3" with two tabs: "Nutrition Tracker" (active) and "Workout Tracker". The main content area displays "Calories Consumed: 3200" with a progress bar. Below this, a box shows "Target: 3500kcal". An "Error" dialog box is overlaid, containing a red circle with a white minus sign, the text "Please enter calories", and an "OK" button. Below the dialog, there is an "Add a Meal" section with input fields for "Meal" (containing "toast"), "Time" (21:00), and "Calories" (empty), followed by an "Add" button. At the bottom, a "search meals" button is on the left, and a "maintenance calories" section on the right shows a bar chart with four bars labeled "No meals recorded" and one bar labeled "+400kcal".

User ID: 3

Nutrition Tracker Workout Tracker

Calories Consumed:

3200

Target: 3500kcal

Error

Please enter calories

OK

Add a Meal

Meal Time Calories

toast 21 00

Add

search meals

maintenance calories

No meals recorded No meals recorded No meals recorded No meals recorded

+400kcal

Test 25

As can be seen, an error box is correctly displayed telling the user that calories consumed must be a whole number. This was the expected result.

The screenshot shows a web application window titled "User ID: 3" with two tabs: "Nutrition Tracker" (active) and "Workout Tracker". The main content area displays "Calories Consumed: 3200" with a progress bar. Below the progress bar, a box indicates "Target: 3500kcal". An "Error" dialog box is overlaid on the screen, displaying a red circle with a white minus sign and the message "Calories consumed must be a whole number". The dialog has an "OK" button. Below the error message, there is a form to "Add a Meal" with fields for "Meal" (containing "toast"), "Time" (with dropdowns for "21" and "00"), and "Calories" (containing "A"). An "Add" button is below the form. At the bottom, there is a "search meals" button and a table showing "maintenance calories". The table has four columns, each with the text "No meals recorded", and a final column with a bar chart and the text "+400kcal".

User ID: 3

Nutrition Tracker Workout Tracker

Calories Consumed:

3200

Target: 3500kcal

Error

Calories consumed must be a whole number

OK

Add a Meal

Meal Time Calories

toast 21 00 A

Add

search meals

maintenance calories

No meals recorded	No meals recorded	No meals recorded	No meals recorded	+400kcal
-------------------	-------------------	-------------------	-------------------	----------

Test 26

As can be seen, after entering meal details and pressing the add button, the meal appeared in the treeview and database as expected.

Meal being added

Add a Meal

Meal

Time

Calories

test

23 ▾ 00 ▾

1000

Add

Meal in treeview

lunch	12:45	1100
crisps	15:00	300
dinner	19:00	1200
test	23:00	1000

Meal in TbIMeals

	UserID	MealID	MealDesc	Dateeaten	Timeeaten	Calories
	Filter	Filter	Filter	Filter	Filter	Filter
12	3	12	dskhgv	2023-04-21	00 :00	600
13	3	13	ghjhf	2023-04-21	00 :00	700
14	3	14	dbd	2023-04-21	00 :00	600

Test 27

As expected, the workout tracker opened after clicking the tab at the top.

User ID: 3

Nutrition Tracker

Workout Tracker

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

Add an exercise:

Exercise

Sets

Reps

Weight

Comments

Add

Find a workout:

Day

Month

year

Search

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

Test 28

As can be seen, all exercises entered today have been displayed in the treeview when the workout page is opened

User ID: 3

Nutrition Tracker

Workout Tracker

Exercise	Sets	Reps	Weight	Comments
Bench press	5	5	60kg	
Shoulder press	3	10	20kg	left shoulder pa
Tricep pushdov	3	12	22.5kg	
Dips	3	8	body	

Add an exercise:

Exercise

Sets

Reps

Weight

Comments

Add

Find a workout:

Day

Month

year

Search

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

Test 29

As can be seen, an error box appears asking me to enter the exercise name after I press the add button without entering the name of the exercise

User ID: 3

Nutrition TrackerWorkout Tracker

Exercise	Sets	Reps	Weight	Comments
Bench press	5	5	60kg	
Shoulder press	3	10	20kg	left shoulder pa
Tricep pushdov	3	12	22.5kg	
Dips	3	8	body	

Add an exercise:

Exercise

Comments

Enter exercise name

OK

DayMonthyear

Search

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

Test 30

As can be seen, when I tried to enter an exercise without entering the sets performed, an error box appeared asking me to enter sets performed, as expected.

User ID: 3

Nutrition TrackerWorkout Tracker

Exercise	Sets	Reps	Weight	Comments
Bench press	5	5	60kg	
Shoulder press	3	10	20kg	left shoulder pæ
Tricep pushdov	3	12	22.5kg	
Dip				

ErrorEnter sets performedOK

Add an exercise:

Exercise	Sets	Reps	Weight	Comments
pull ups		17	body	

Add

Find a workout:

Day	Month	year

Search

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

Test 31


As can be seen, when I tried to enter an exercise without entering reps performed, an error box appeared asking me to enter reps performed, as expected.

User ID: 3

Nutrition TrackerWorkout Tracker

Exercise	Sets	Reps	Weight	Comments
Bench press	5	5	60kg	
Shoulder press	3	10	20kg	left shoulder pa
Tricep pushdov	3	12	22.5kg	
Dips	3	8	body	

Error

Enter reps performed

OK

Add an exercise:

Exercise	Sets	Reps	Weight	Comments
<input type="text" value="pull ups"/>	<input type="text" value="3"/>	<input type="text"/>	<input type="text" value="body"/>	<input type="text"/>

Add

Find a workout:

Day	Month	year
<input type="text"/>	<input type="text"/>	<input type="text"/>

Search

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

Test 32 & 33

As can be seen, when I try to enter a workout with sets or reps as non-numeric values, an error box appears asking the user to ensure they are numbers.


User ID: 3

Nutrition Tracker

Workout Tracker

Exercise	Sets	Reps	Weight	Comments
Bench press	5	5	60kg	
Shoulder press	3	10	20kg	left shoulder pa
Tricep pushdov	3	12	22.5kg	
Dips	3	8	body	

Error



Make sure sets and reps are numbers!

OK

Add an exercise:

Exercise

Sets

Reps

Weight

Comments

pull ups

A

A

body

Add

Find a workout:

Day

Month

year

Search

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

Test 34

As can be seen, after entered an exercise with the correct details, it was displayed in the treeview

User ID: 3

Nutrition Tracker

Workout Tracker

Exercise	Sets	Reps	Weight	Comments
Bench press	5	5	60kg	
Shoulder press	3	10	20kg	left shoulder pa
Tricep pushdow	3	12	22.5kg	
Dips	3	8	body	
Tricep pushdow	5	5	30kg	
pull ups	5	10	body	

Add an exercise:

Exercise

Sets

Reps

Weight

Comments

Add

Find a workout:

Day

Month

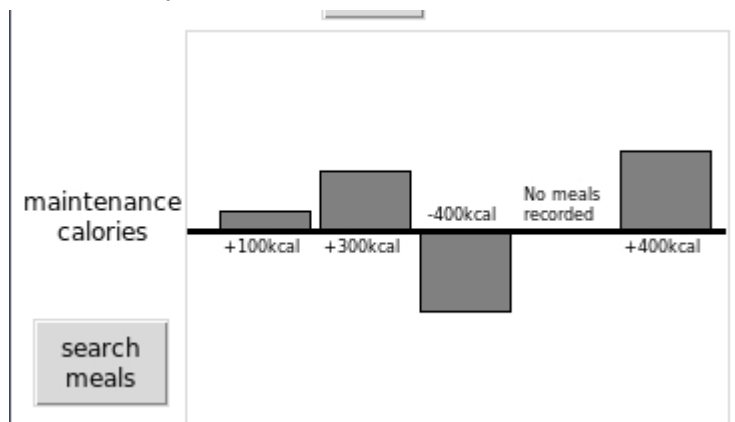
year

Search

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

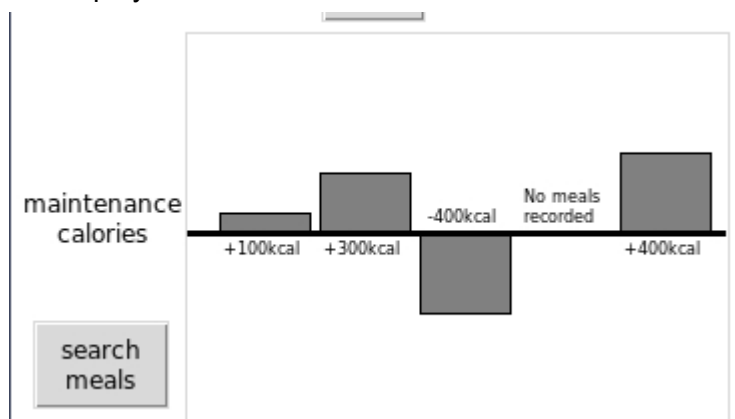
Test 35

As can be seen, the bar chart correctly shows the users calorie surpluses and deficits over the last 5 days.



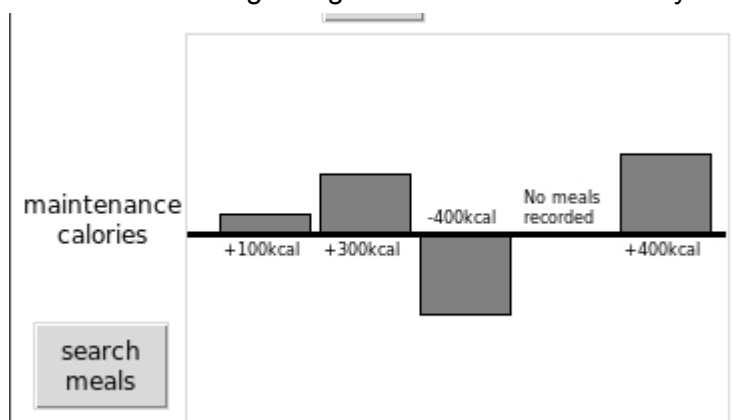
Test 36

As can be seen, the exact calorie surplus/deficit the user has been in over the past 5 days are displayed on the bar chart



Test 37

As can be seen, the bar chart correctly says when no meals have been recorded in a day, rather than showing a huge calorie deficit with a very long bar



Test 38

As can be seen, the users user ID is displayed at the top of the window as the window title

User ID: 3

Nutrition TrackerWorkout Tracker

Calories Consumed:

3200

Target: 3500kcal

Meal	Time	Calories
cereal	09:00	600
lunch	12:45	1100
crisps	15:00	300
dinner	19:00	1200

Add a Meal

Meal

Time

Calories

1900

Add

maintenance calories

No meals recorded

No meals recorded

No meals recorded

No meals recorded

+400kcal

search meals

Test 39

As can be seen, a colon was correctly put between the hours and minutes of when a meal was eaten when entered into the app

User ID: 3

Nutrition TrackerWorkout Tracker

Calories Consumed:

3200

Target: 3500kcal

Meal	Time	Calories
cereal	09:00	600
lunch	12:45	1100
crisps	15:00	300
dinner	19:00	1200

Add a Meal

Meal

Time

Calories

1900

Add

maintenance calories

No meals recorded

No meals recorded

No meals recorded

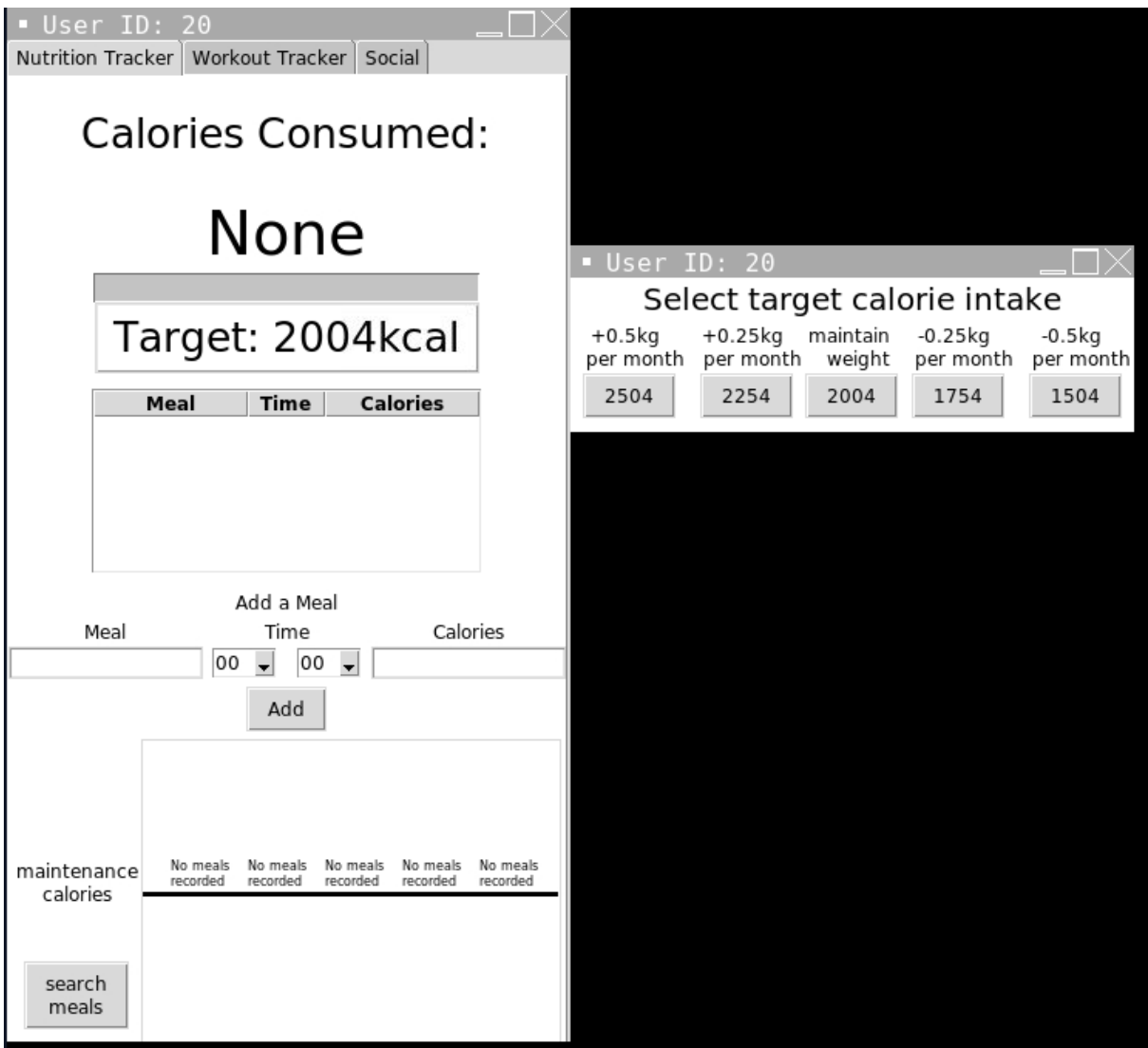
No meals recorded

+400kcal

search meals

Test 40

As can be seen, after pressing the calorie target button, the calorie target page was displayed.

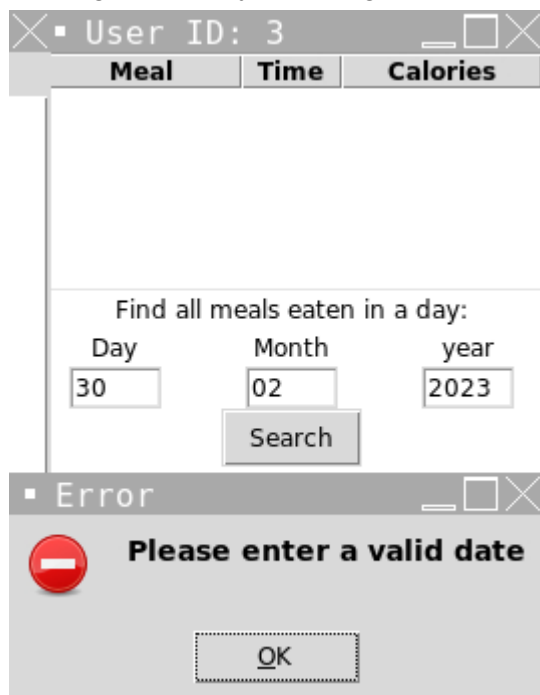


As can be seen, after pressing the search meals button, the find meals window opened as expected.

1. *Journal of the American Medical Association*, 277: 1025-1026, 1997.

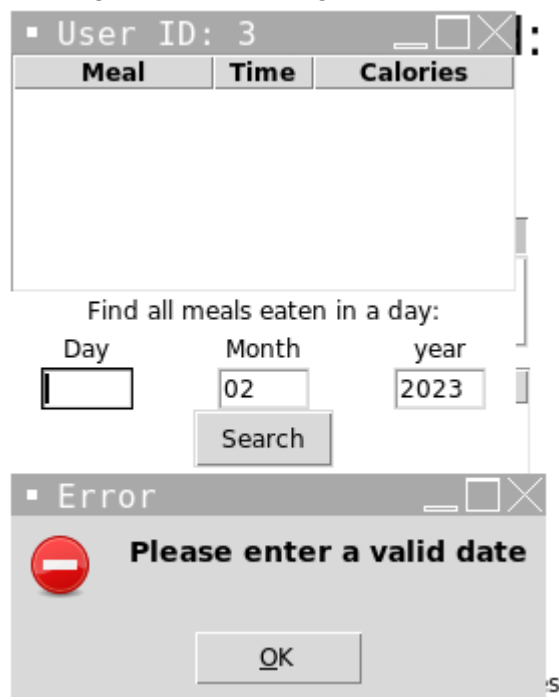
Test 42

As can be seen, when a user enters a fake date into the search meals window, an error message is displayed asking the user to enter a valid date



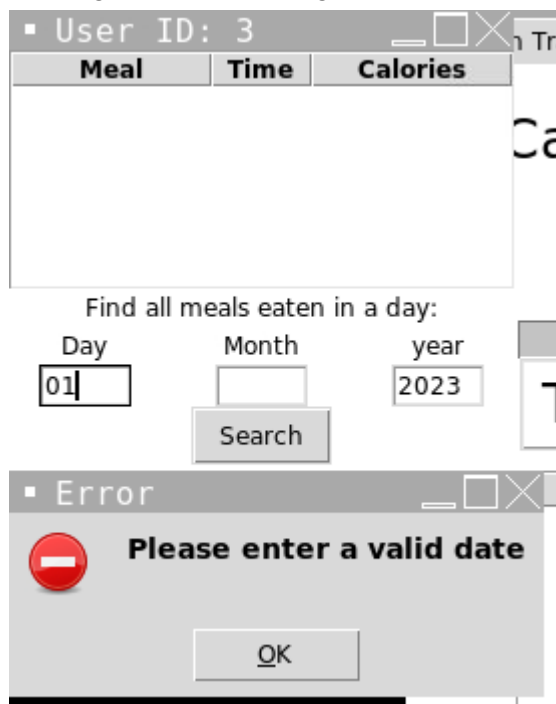
Test 43

As can be seen, if a user leaves the day box blank when searching for meals, an error message appears asking them to enter a valid date



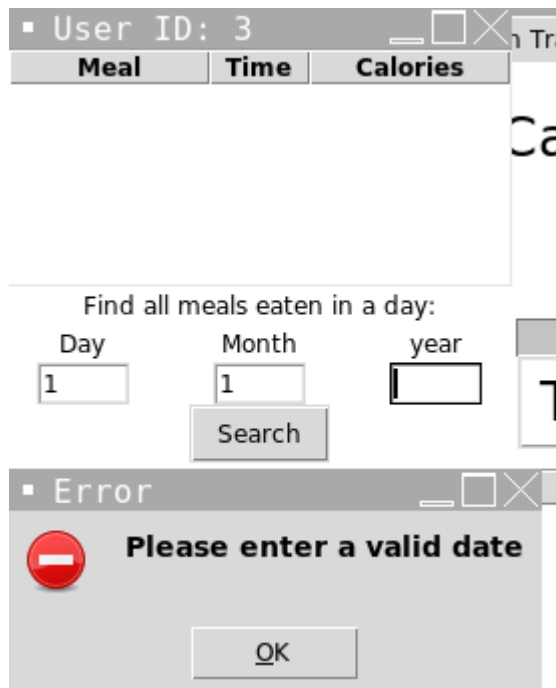
Test 44

As can be seen, if a user leaves the month box blank when searching for meals, an error message appears asking them to enter a valid date



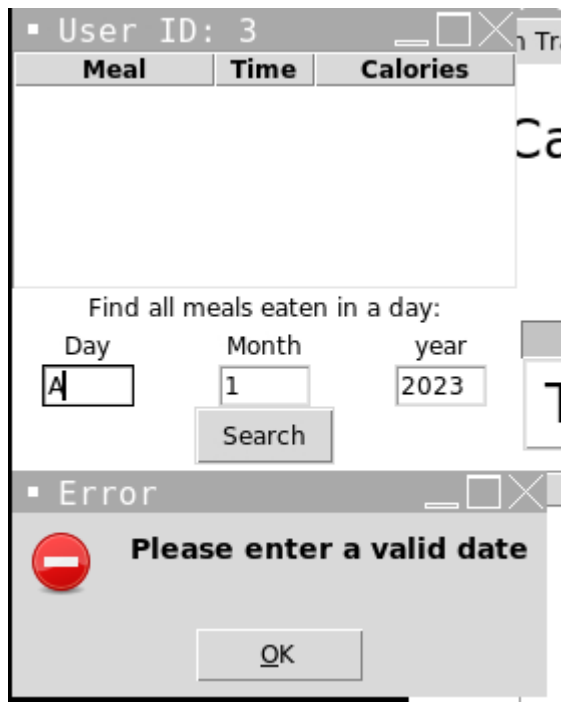
Test 45

As can be seen, if a user leaves the year box blank when searching for meals, an error message appears asking them to enter a valid date



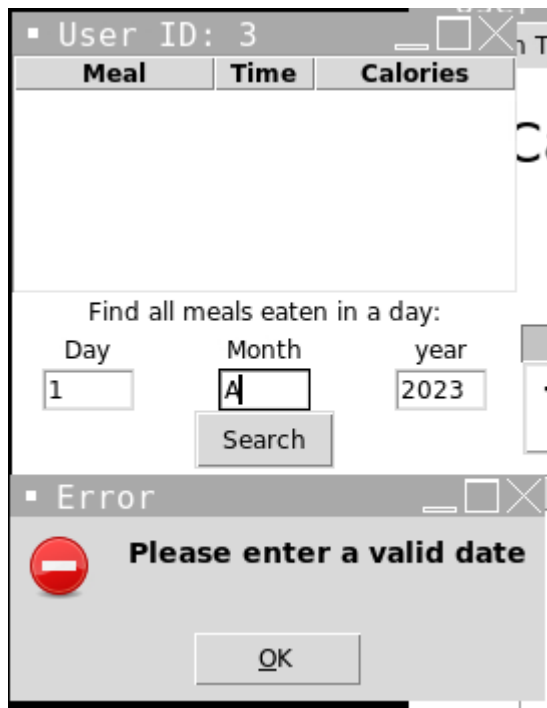
Test 46

As can be seen, when I tried to enter a date where the day was non numeric, an error box was displayed asking me to enter a valid date, as expected.



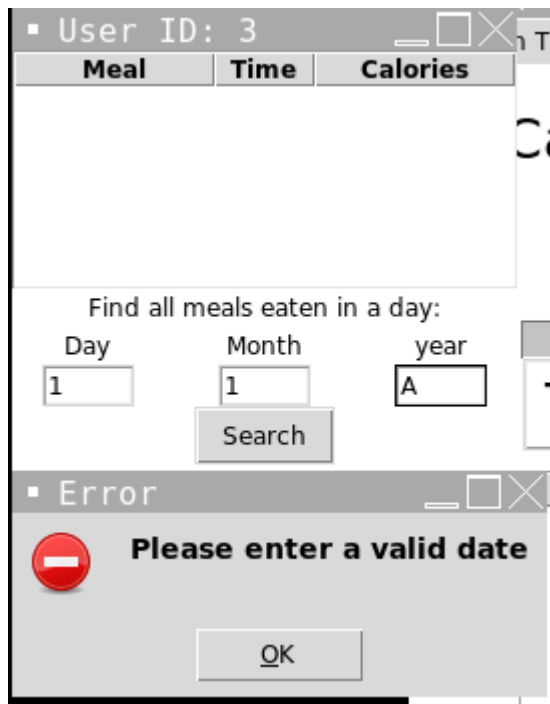
Test 47

As can be seen, when I tried to enter a date where the month was non numeric, an error box was displayed asking me to enter a valid date, as expected.



Test 48

As can be seen, when I tried to enter a date where the year was non-numeric, an error box appeared asking me to enter a valid date, as expected.



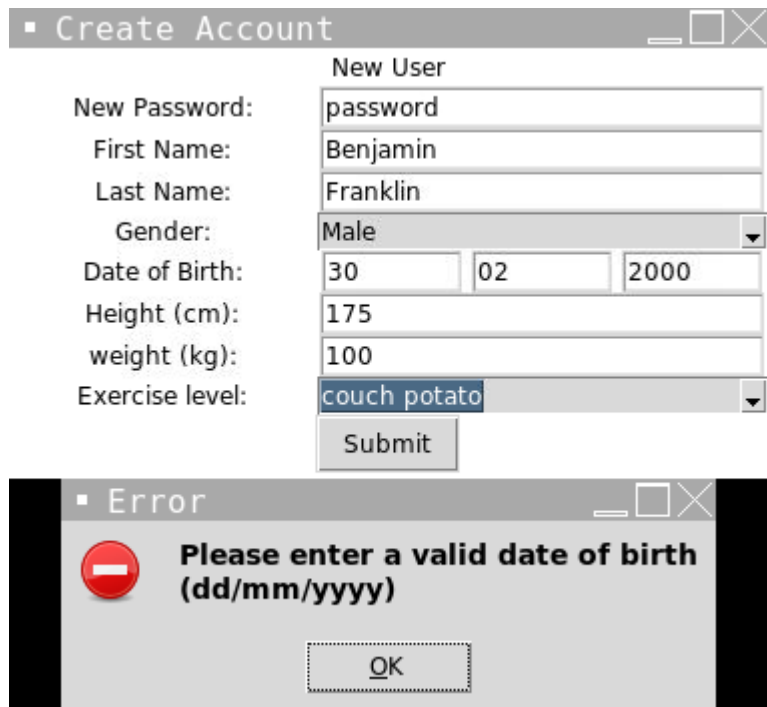
Test 49

As can be seen, when I entered a date, all meals entered into the app by the account logged in on this day appeared in the treeview as expected.



Test 50

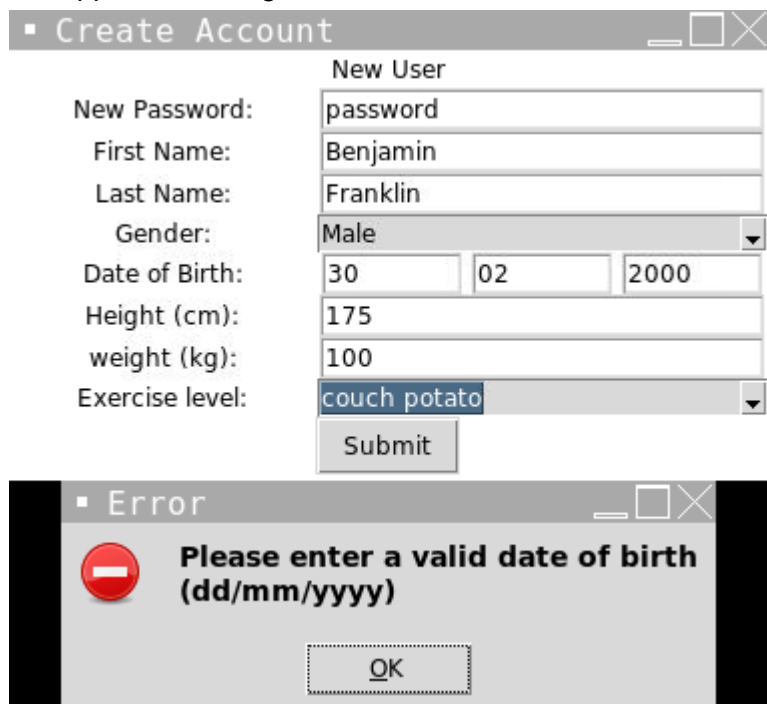
As can be seen, when I attempted to log in with a date which was not real, an error box appeared asking me to enter a valid date as expected.



The image shows two overlapping windows. The top window is titled 'Create Account' and contains a 'New User' form. The form fields are: 'New Password:' with value 'password', 'First Name:' with value 'Benjamin', 'Last Name:' with value 'Franklin', 'Gender:' with a dropdown menu showing 'Male', 'Date of Birth:' with three input boxes containing '30', '02', and '2000', 'Height (cm):' with value '175', 'weight (kg):' with value '100', and 'Exercise level:' with a dropdown menu showing 'couch potato'. A 'Submit' button is at the bottom. The bottom window is titled 'Error' and displays a red circular icon with a white horizontal bar. To the right of the icon, the text reads 'Please enter a valid date of birth (dd/mm/yyyy)'. An 'OK' button is at the bottom of the error dialog.

Test 51

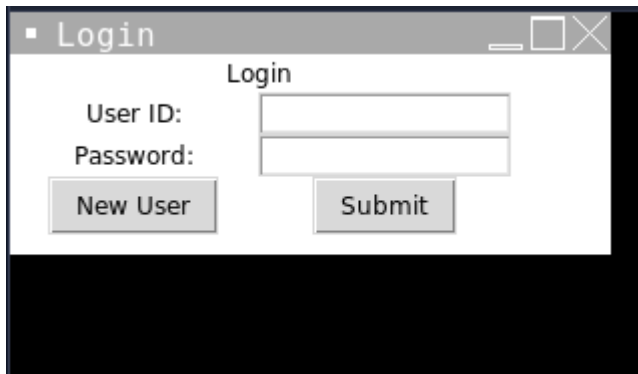
As can be seen, when I tried to search for a workout on a date which does not exist, an error box appeared asking me to enter a valid date



This screenshot is identical to the one for Test 50, showing the 'Create Account' form and the 'Error' dialog box. The form fields are: 'New Password:' with value 'password', 'First Name:' with value 'Benjamin', 'Last Name:' with value 'Franklin', 'Gender:' with a dropdown menu showing 'Male', 'Date of Birth:' with three input boxes containing '30', '02', and '2000', 'Height (cm):' with value '175', 'weight (kg):' with value '100', and 'Exercise level:' with a dropdown menu showing 'couch potato'. A 'Submit' button is at the bottom. The bottom window is titled 'Error' and displays a red circular icon with a white horizontal bar. To the right of the icon, the text reads 'Please enter a valid date of birth (dd/mm/yyyy)'. An 'OK' button is at the bottom of the error dialog.

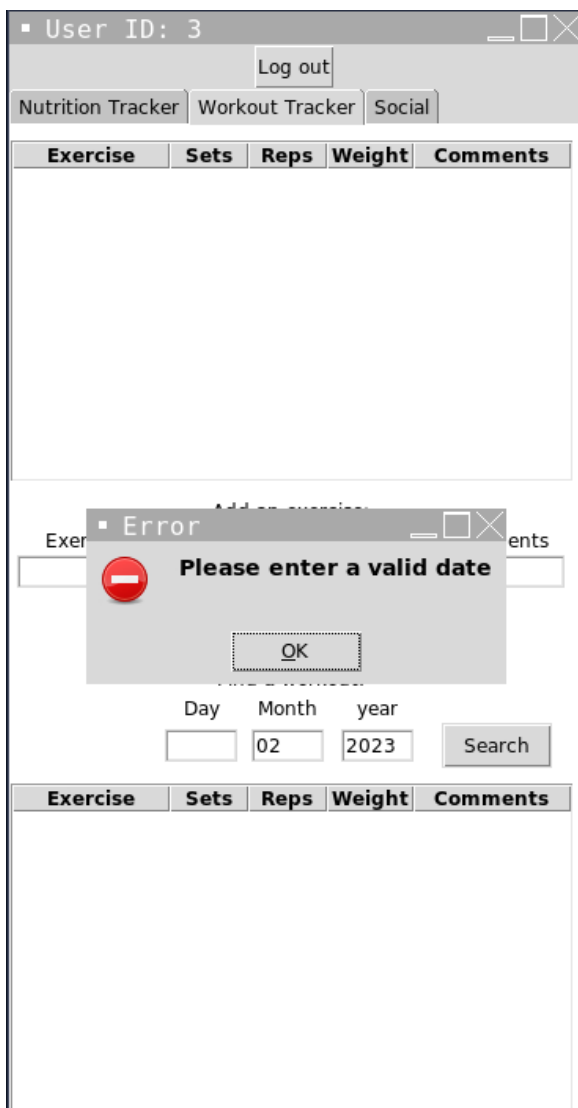
Test 52

As can be seen, after I pressed the log out button, the login page opened, as expected.



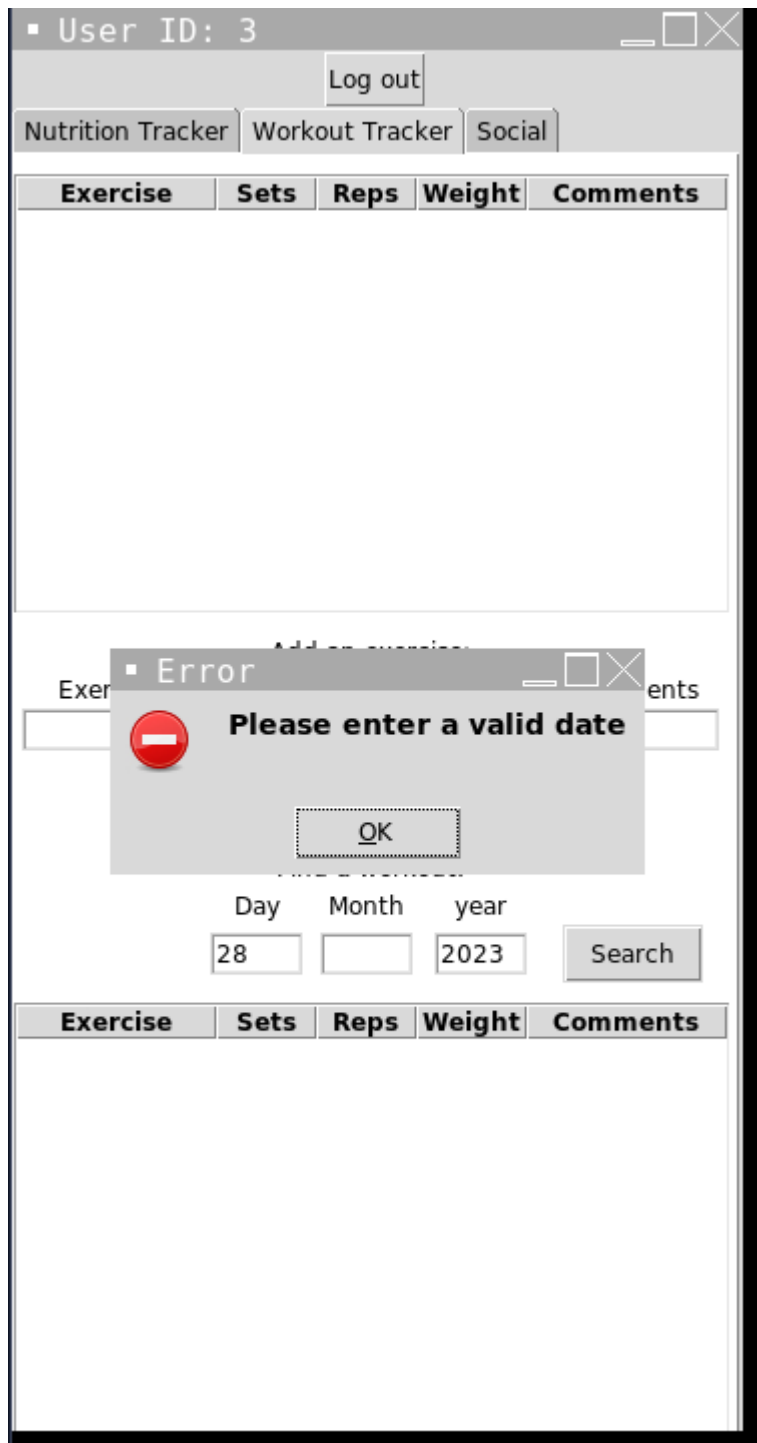
Test 53

As can be seen, an error box appeared asking me to enter a valid date after I tried to search for a workout without entering the day.



Test 54

As can be seen, an error box is displayed asking me to enter a valid date when I try to search for a workout without entering the month, as expected



Test 55

As can be seen, an error message is displayed asking me to enter a valid date when I try to search for a workout without entering the year

The screenshot shows a web application interface for a user with ID 3. The user is logged in, as indicated by the 'Log out' button. The application has three main tabs: 'Nutrition Tracker', 'Workout Tracker', and 'Social'. The 'Workout Tracker' tab is currently selected. Below the tabs is a table with the following headers: 'Exercise', 'Sets', 'Reps', 'Weight', and 'Comments'. The table is currently empty. An 'Error' dialog box is displayed in the center of the screen, with the message 'Please enter a valid date'. The dialog box has a red 'X' icon and an 'OK' button. Below the dialog box, there is a date selection form with three input fields labeled 'Day', 'Month', and 'year'. The 'Day' field contains the value '28' and the 'Month' field contains the value '02'. The 'year' field is empty. A 'Search' button is located to the right of the date fields. Below the date fields is another table with the same headers as the one above: 'Exercise', 'Sets', 'Reps', 'Weight', and 'Comments'. This table is also empty.

User ID: 3

Log out

Nutrition Tracker Workout Tracker Social

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

Exercise Add-on exercises

Exercise

Day Month year

28 02

Search

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

Test 56

As can be seen, an exercise was inserted into the treeview called “test exercise” after pressing the search button, as expected

User ID: 3

Log out

Nutrition Tracker

Workout Tracker

Social

Exercise	Sets	Reps	Weight	Comments
----------	------	------	--------	----------

Add an exercise:

Exercise

Sets

Reps

Weight

Comments

Add

Find a workout:

Day

Month

year

10

5

2023

Search

Exercise	Sets	Reps	Weight	Comments
test exercise	5	5	50kg	None

Evaluation

Final product vs success criteria

1.1 A GUI for user to sign in and create an account

Upon opening the app, users are greeted with a login window, from which they can go to an create account menu or simply login tests 1-5 clearly show the login page exists and functions perfectly. Test 5-17 clearly shows the create account window exists and functions perfectly. This criteria has been met.

1.2 Different windows to separate parts of the application

Logging in, creating an account, choosing a calorie target, and searching for a meal all use different windows from the main nutrition and workout one. This objective has been met.

1.3 Different tabs for nutrition and resistance training

The nutrition and workout pages use different frames/tabs in the main window notebook. Test 27 clearly shows a user can easily switch between these tabs. This objective has been met.

2.1 User sign in feature

When the app is run, the first thing to load is the login window. Tests 1-5 prove that this login feature exists and works perfectly. This objective has been met.

2.2 Create account option for new users

Users are able to access the create account page from the login window. Tests 5-17 show this feature exists and works perfectly. This objective has been met.

2.3 Log out feature

When in the main window, there is a logout button displayed at the top. Test 52 proves this works perfectly. This objective has been met.

3.1 Allow the user to enter the exercises they have performed into a table, this data will be stored in a database

The user is able to add a workout into the app via the workout page. This data is then added to TblWorkout and the treeview. Tests 28-34 show this feature exists and works perfectly. This objective has been met.

3.2 Allow users to search the database for any exercises they performed on a chosen date

The user is able to enter a date into the workout page, and all exercises they entered on this date are displayed on a treeview. Tests 53-56 show this feature exists and works perfectly. This objective has been met

4.1 Calculate what number of calories user would need to gain/lose

After creating an account and entering necessary details, the user is directed to the calorie target page, which calculates their maintenance calories. Test 18 proves this feature exists and is working perfectly. This objective has been met. This page can also be accessed from the nutrition page, as shown in test 40.

4.2 Recommend users different calorie intakes

After calculating their maintenance calorie intake, the app calculates different calorie intakes the user need to meet different weight loss goals e.g. +0.5kg per week. Test 18 shows different intake levels are calculated correctly, and test 19 shows the user is able to choose an intake, and this is added to the database and displayed on the nutrition page.

5.1 Allow users to record all their meals and track those meals.

Users are able to enter a meal into the app. It is then displayed in the treeview and recorded in the database. Tests 24-26 show this system works perfectly. This objective has been met.

5.2 Record user total calorie intake over a day

When users add a meal into the app, the total amount of calories they have eaten throughout the day updates. The counter is wiped every 24 hours at midnight. Test 21 proves this system works perfectly. This objective has been met.

5.3 Allow users to look back at this data and see what they ate on a specific day in the past

Users can enter a date into the find meals page and see all of the meals they ate on this day displayed in a treeview. Tests 41-49 show that this system works perfectly

6.1 At the end of each day, the users calorie surplus/deficit will be calculated and plotted onto a graph

At the end of each day, the total calorie intake for this day is plotted onto the bar chart on the nutrition page. Tests 35-37 show this system works perfectly. This objective has been met.

6.2 The graph will show the past 5 days, and will state how high this deficit/surplus was

The bar chart has 5 columns, for the past 5 days. If no meals have been entered in a day, it says “No meals recorded” in place of the bar. The exact deficit/surplus is also shown. This objective has been met.

User feedback

Feedback from experienced user

The user felt that the app fulfilled their minimum expectations (ability to record calorie intake and resistance training) very well, and had absolutely no issues in these areas. Recording their calorie intake over the day was easy and simple, and recording the resistance training exercises they did was also very easy. The fact that python is a portable language meant they could use the app on their phone, which helped as they had no access to a computer in the gym, and preferred using a mobile device to track nutrition while eating.

The user was glad the app saved his data and allowed him to look back at it, as it helped him track his progression in strength training, and see what he had eaten on days where his calorie intake was very high or low, so he could identify the reason for this.

The user felt that having a workout tracker which already had the exercises, sets and reps they would perform would be beneficial, as they would not have to search what exercises they last performed if they forgot, and they would save time entering all this data every workout.

They felt that the GUI was simple enough and made their experience more enjoyable and easy. They were not bothered by the aesthetic of the app, so long as the GUI was not over complicated. They thought the calorie intake bar chart was very easy to understand, and was effective in showing their calorie deficits and surpluses.

The user would have found it helpful if the bar chart specified the dates for each bar, and also would have liked if the find meals page showed a 1 day version of this chart alongside the treeview when searching for meals.

Feedback from inexperienced user

The inexperienced found the app very helpful in starting them on their fitness journey, they felt the app did a lot of work for them they would otherwise not be able to do due to a lack of knowledge, such as calculating a calorie target for different weight goals for them.

The user liked the create account process, as it was very streamlined and gave very little room for error due to splitting the process up into different windows, making it less complex. The user appreciated the calorie target page, and liked that the app said how rapidly they would lose or gain weight with each daily intake.

The user appreciated the bar chart on the nutrition page as this made it easier to understand how calorie deficits and surpluses worked.

The user found the workout tracker easy to use, but would have appreciated pre-made workouts as they weren't sure what exercises they should do.

The user found the app relatively easy to use and gain an understanding of, however felt that a quick tutorial or guide upon creating an account would help them understand the app more quickly than they otherwise did.

The user felt the GUI was adequately simple, and there weren't any issues when using the app. However they placed a much greater importance on the app's visual appeal than the experienced user, and thought the app could have looked more aesthetically pleasing.

The user found logging in very easy, but would have liked the option to use a username rather than a user ID, as it was slightly more memorable and was also just more personalised to the user.

Analysis of feedback

Overall, the users feedback was very positive, and they both felt the app fulfilled their needs.

Both users found the GUI suitable for the app, but the less experienced user ideally would have liked a slightly more visually appealing interface.

Both users felt the app did a very good job of delivering their minimum expectations. They had a few extra suggestions, but felt these were not necessities and therefore were not urgent.

Both users were happy with the way the app assisted them in the task of tracking nutrition and resistance training. They felt the way that data was inserted into a treeview when entered into the app was very helpful, and were also happy data was saved in the database, so they could look back on it later with the find workout and find meals features.

Room for improvement

Allowing the user to make default workouts

There could be an option to automatically fill the workout treeview with details (exercise name, sets, possibly reps) of a previously made workout which the user does regularly. This would save users the time and effort of typing in this information every workout.

Recommended/default workouts

Recommending users' default workouts which they do not have to make themselves could be a huge benefit to beginner users who do not want to create their own routine. An option to automatically fill out parts of the workout treeview, so the user was told what exercise to do, and for how many sets and reps, could be added to the app.

Specifying dates on bar chart

The date that each bar on the bar chart represents could be displayed on the bar chart, this would help the user understand the bar chart slightly more easily, as they could get mixed up about what order the days are in.

Including a smaller bar chart when searching for a meal

When searching for all meals eaten in a specific day, a bar chart showing the calorie surplus/deficit the user was in on this day (like a smaller version of the one on the nutrition page) could be displayed alongside the treeview.

Tutorial/guide upon creating an account

Adding a tutorial showing the user how to use the app would be helpful as some beginner users may need help understanding the app and some of the technical terminology it uses.

Visual appeal

The GUI could be more visually appealing, I could add a few more colours, as the app is almost exclusively black and white, I could also add a minimalist background as there is essentially no detail in the app, this would help attract prospective users.

Using usernames instead of user IDs

Instead of using auto-incrementing user IDs to log in, I could allow users to make a unique username to log in with when making an account. This would probably be more memorable, and just allow users to have some personalisation, which people usually appreciate.

Conclusion

After reflecting on the feedback, I think it is clear that my project fulfils the needs of my target audience. There are certainly changes I could make to improve the user experience even more so, however these are relatively small quality-of-life improvements which do not involve the core functions of the application. These changes would be very quick and easy to implement.