NEA 2021

FITNESS APP

Vishal Pittala

Year 13

**Table of Contents**

1. Analysis …………………………………… 2
   1. Background to the problem.…………………………......2
      1. General Background ………………………………………............... 2
      2. Client Background …………………………………………………. 2
   2. Questionnaire Results and Analysis …………………… 3
      1. Question 1…………………………………………………………… 3
      2. Question 2…………………………………………………………… 4
      3. Question 3…………………………………………………………… 5
      4. Question 4…………………………………………………………… 5
      5. Question 5…………………………………………………………… 6
   3. Interview and Analysis…………………………….…… 7
      1. Interview with Client ………………………………………………. 7
      2. Interview Analysis …………………………………………………. 8
   4. Observation of Existing System ………….…………... 10
      1. App No.1 – “Home Workouts” …………………………………… 10
      2. App No.2 – “MyFitnessPal” ………………………………………. 13
   5. Data Flow Diagrams ………...………………………... 15
   6. System Flowchart …………………………………….. 17
   7. Prospective Users and Acceptable Limitations ………. 18
      1. Prospective Users ………………………………………………… 18
      2. Acceptable Limitations …………………………........................... 18
   8. Storyboard …..…………………………………………. 1
   9. Smart Objectives ……….……………………………… 2

**1 Analysis**

* 1. **Background to the Problem**
     1. **General Background**

Fitness and health are one of the most important things in live. No matter how successful you are or how much money you have, there are no shortcuts to fitness except simple hard work. To maintain a long, healthy life one must remain fit and active, to do so there are a lot of options, such as going to a gym, going on runs or doing home workouts.

Home workouts are in high demand as a lot people have very busy lives and do not have enough time to dedicate hours each day at the gym but still want to stay fit and healthy. Others may want to keep fit but can never stick to a single gym since they may be moving around a lot. Even under-18s that want to stay fit but can’t afford a gym membership are looking for new alternatives. The common factor between these groups of people is that they all want to stay fit but want a cheap, less time consuming and portable option.

Recently, I have researched into a few home workout apps that are on the market and I have found that each have them have excellent features such as graphs of progress, personalised workouts for different muscle groups and different modes such as beginner, intermediate and advanced etc. However, not many apps have all the features that the audience wants, some apps may include one of the features but not another, which is why some people have never really liked home workout app.

**1.1.2 Client Background**

Primary Client: Kedhar Pydimarry

Target Audience: Anyone looking to get fit and change their lives

Kedhar is an 18-year-old Year 13 student, currently studying at King Edward VI Grammar School, Chelmsford. He enjoys activities such as gaming, coding, playing his guitar and socialising. However, he feels that recently with all the stress he has he feels that he has put on weight and would like to lose it.  Unfortunately, due to the stressful period that year 13s have with revising for exams and preparing for university he is unable to travel to the gym regularly or even have motivation to go. Also, especially during recent events with COVID-19 causing a lockdown throughout the UK and also closing all gyms, he feels that his fitness levels are receding, and he wants to put his life back on track and change his body for the better

* 1. **Questionnaire Results Analysis**

I created a questionnaire for my target audience (which you can see on the left) to gather audience-specific information about how I can create a program that is tailored towards the census.

I collected data from 10 different individuals by the means of opportunity sampling (Including my client) and generated a variety of visual graphs and diagrams to represent the results of the survey. By using a mix of volunteer and opportunity sampling I have found the individuals that were most suitable for my project who showed great interest and wanted to be a part of it.

**1.2.1 Question 1**

* In the bar chart above I have depicted the data accumulated from the results of question 1 in the questionnaire, with a total sample size of 10 people.
* From the graph you can see that 40% of candidates opted for option f which is “I don’t work out”, therefore since the majority of the audience chose this I must consider incorporating a program for beginners to learn the basics and how to exercise safely and progress gradually without burning out straight away.
* Furthermore, you can see from the graph that the rest of the 60% of the sample’s choices ranged from option A to D, which meant there was a large range of responses from very little exercise to a large amount of exercise daily.
* Therefore, by taking this into consideration I must create programs that are tailored to both people are currently not very active and people that are very active, maybe by incorporating different sets of difficulty for different people with much more intense workouts for people that are more used it and the people that are used to less intense workouts, over time, will be able to progress to a higher intensity.

**1.2.2 Question 2**

* In the diagram above I have constructed a pie chart using the data from the results of each sampling unit’s choice.
* As you can see, the majority of the population opted for option A, meaning they think they aren’t doing enough exercise.
* Therefore, by taking this into consideration I should make sure that the program is designed in the way that is more inclined towards motivating the users to try and become more active and to maintain consistency.
* The rest of the candidates chose between options B and C, meaning they think they are doing enough or too much exercise.
* Therefore I must include a section to help people maintain their current regime and fitness level or maybe to relax it down for maximum results.

**1.2.3 Question 3**

* Using the results from the candidate's choices for Question 3 I have formulated a pie chart, as seen above, to represent the data.
* As you can see, around 60% of people have used fitness apps before however 40% of people have not.
* Therefore I must consider creating a system that is easy to operate for brand-new users and also include common systems as well so that the people that have used fitness apps before don’t have to get used to a completely different system so that they can get straight into it without having to waste time learning to use a new system.

**1.2.4 Question 4**

Responses:

*“I would usually look for a program that has a large variety of exercises to allow me to train all parts of my body”*

*“A program that can track my progress and data to show me how well I am doing and how I can progress would be great”*

*“I need to have an app that gives me daily reminders to keep me motivated and make me stay in shape”*

**Conclusion:**

These are some of the most popular responses received from question 4 on the questionnaire. A lot of the audience really wanted a feature where they can track their progress and data to motivate them. I think this is a great idea to implement as it is one of the key requirements by the audience and by including a page to store progress and data this will allow the user to visually see progress and by doing so this will provide them an incentive to continue to exercise as they can see the results actually taking place. Another response was to have daily reminders to workout, this will make sure that the user is always reminded to exercise and stay in shape as they may be busy and forget. Finally, the clients also requested a variety of different exercises routines. This is a good idea as it allows the user to train all the muscles in their body and with a large range of exercises this stops the workout from being boring and repetitive, it keeps the user engaged with fresh new workouts.

**1.2.4 Question 5**

* In the bar chart above I have formulated a bar chart from the results of question 5 in the questionnaire, with a total sample size of 10 people.
* From the graph you can see that 50% of candidates opted for option b which is that they want to “lose some weight / fat”, therefore since the majority of the audience chose this I must consider incorporating a HIIT fat burning program into the solution and make sure that the user is constantly reminded of their progress so that they will continout to exercise and reach their goal.
* Furthermore, you can see from the chart that the rest of the 30% of the sample’s choices were option C where they want to “gain muscle”. To accommodate for this I must create a program that makes the user use body weight to train their muscles, rest days are also crucial to allow for muscle recovery.
* Also, from the bar chart you can see that 20% of the sample opted to “tone their current body”, to incorporate this into the solution I must add a section for calisthenics with muscle training but also HIIT sections to burn fat to allow them to become leaner and tone their muscles.
* Therefore, by taking this into consideration I must create programs that are tailored to people with all types of fitness goals, this can be done by splitting the app into sections for which goal you wish to complete and provide them with workouts to help them with their specific goals.
  1. **Interview and Analysis**
     1. **Interview with Client**

ME: When you worked out, how would you choose what workouts you did?

KEDHAR: I chose randomly, whatever I felt like on the day, sometimes maybe 2 days in a row of arm day if I felt like it. Sometimes just some running, whatever my mood is in that day.

ME: What do you think are the benefits of how you currently do your workouts?

KEDHAR: It suits my mood and my lifestyle and is very lenient so it’s easy for me

ME: What do you think are the drawbacks of how you currently workout?

KEDHAR: It makes me very lazy and since it’s not very consistent I am unable to have any effective results.

ME: What is currently limiting you from reaching your end goal of losing fat?

KEDHAR: I am really lazy at the moment.

ME: You said you were lazy? What do you think the reason for this is? Stress? No facilities such as the gym to work out.

KEDHAR: I feel unmotivated at the moment and feel like I need equipment like in the gym to actually have an effective workout.

ME: If a new app for home workouts were to be developed, which would you rather prefer? A selection of workouts for different muscle groups like, Abs only or Chest only or Legs only, or would you like a full body workout?

KEDHAR: I would prefer a full body workout as I feel that my whole body will be utilised and if it was a muscle group per day then I feel like some muscles won’t really be used for quite a few days. Obviously, I will also need a rest day so that my muscles can rest as well. I also feel that in terms of burning fat a full-body workout will be much more efficient.

ME: What about cardiovascular exercise? Do you think you will want to do that as well? Also, how often? What is your preferred for of cardiovascular exercise?

KEDHAR: I feel that cardio is extremely important for fat burning and I would love to incorporate that into my routine. Maybe 3 or 4 times a week. My preferred cardio workout has got to be running/ treadmill because I feel like I am working myself powerfully throughout the exercise.

ME: How do you keep track of your improvements in fitness?

KEDHAR: I usually don’t, sometimes I might check my weight

ME: So, would you prefer a system that shows your progress regularly to motivate you?

KEDHAR: Definitely! I feel like that would help a lot!

ME: That is now the end of the interview, thank you for answering my questions and I hope to design my program tailored to your responses to provide a much more bespoke solution!

**1.3.2 Interview Analysis**

After the interview I was able to pinpoint key areas where the client requires more help within his routine. From the first question about how he chose different workouts he responded by saying he “chose randomly, whatever I felt like on the day” and this shows how his current plan is not strict and is too flexible at the point where it's not at all effective and he is not seeing the results he wants to. To target this key flaw, I must design a system where it keeps the client in a strict program with a balanced workout plan where it is easy to follow and extremely effective. To make sure that the client indeed follows the program I must think about implementing a reminder system where the client will receive daily notifications telling him to workout. He also says how he completes “maybe 2 days in a row of arm day” which is not very effective as research show from the article by pure gym1suggests that rest days are crucial for the development of muscles since the whole process of building muscles is about breaking down muscles and growing them back stronger. Therefore, training the same muscle group 2 days in a row will prevent muscle growth instead of helping it grow.

One of the main drawbacks my client feels stops him from working out is "feeling lazy" and tired and unmotivated. Therefore, because of this he doesn't stick to a regular plan which in the long run can be detrimental. To counteract this, I must implement a system to motivate the user, such as scheduling a "cheat day" once a week where the user can relax and eat what he/her wants. Therefore, this will make sure that the user puts in maximum amount of effort throughout the week as they can see a rest period coming up, they will work harder. If there were no "cheat days" the user will feel like they are constantly working and will feel extremely unmotivated.

My client also feels that a full-body workout plan would be much more effective for him compared to a workout that is tailored for a different muscle group. This is because he feels that by working his whole body some of his muscles are not used for a lot of days and he feels like it is useless. This shows that he is clearly misinformed since rest days are crucial for optimal muscle growth. Therefore, I must also ensure that the program includes an information section to provide beginners with key information that is important to know such as making sure that they are well rested between workouts. Also, since the client enjoys full-body workouts, I should make sure that the program provides a variety of different options from specialised workouts for specific muscle groups to general full-body workouts and also cardio workouts such as HIIT workouts.

In terms of tracking progress, my client says that he doesn't really check his progress and that the main form of checking progress for him is to check his weight occasionally, not in regular intervals. This is not effective and can also be the reason why my client feels unmotivated a lot. From an article by myprotein2, the best way to track weight loss or change in physique is to do periodic weight checks and to beware of water weight loss which is a drastic weight loss seen after starting a diet which is not fat loss but just losing water weight. So, by taking regular weight measurements you can see the pattern in weight loss, and this will provide a boost in motivation for you to continue. Therefore, in my program I will include a weight progression tracker which will create a graph using the user's data to represent their weight loss/gain. Also, another great way to measure weight loss is for the user to measure waist size as well, so this can also be implemented into the program and also show a graphical progression of waist size decrease.

1 = <https://www.puregym.com/blog/why-rest-days-are-important/>

2= https://www.myprotein.com/thezone/nutrition/best-ways-measure-fat-loss-progression/

* 1. **Observation of existing systems**

I observed some of the home workout applications that the client previously used to get a better understanding of how the users interact with it on a daily basis and identify the problems they face while doing this. I navigated through the app, trying to work out each feature and how efficient it is and analysed the advantages and drawbacks of certain aspects.

* + 1. **App No.1 – “Home Workouts”**

The current process is as follows:

1. There are multiple different sections which are labelled as: “Beginner”, “Intermediate” and “Advanced”. Each section has workouts for Abs, Chest, Arm, Leg and Shoulder & Back. Each workout tells you how long it’s meant to last and the exercises included in them. However, if the user doesn’t want to do, for example, Russian twists but instead wants to do more crunches then that is not possible. So, there for it is not able to be personalised for the user.

A screenshot of a cell phone

Description automatically generated with medium confidenceA screenshot of a cell phone

Description automatically generated with medium confidenceA screenshot of a cell phone

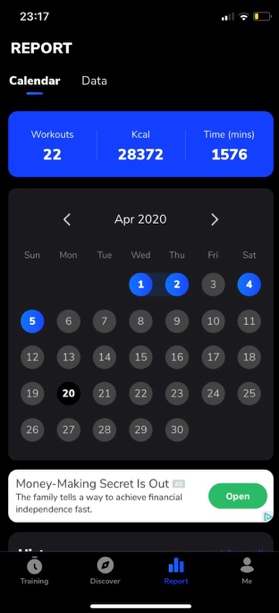
Description automatically generated with medium confidence

1. However it does include a “Discover” section which has an extremely large variety of workouts that are different from the normal such as “Get rid of man boobs HIIT” or “Build wider shoulder” which may interest the user as its unique and new and very specific to certain goals so inexperienced users can easily find what they are looking for in their goal and work on it.

A screenshot of a cell phone

Description automatically generated with medium confidence

1. It also includes a “Report” section which shows the user a calendar with what days the user worked out on and how many the user has done with the total calories burnt and total minutes the user has worked out for. Also, a bar graph of how any calories have been burn on each day. However, the data section would be more effective if it told the user roughly how long it would take to get to their desired weight if they continue with the same routine, I believe that this would make the user much more motivated to continue to work out.

A screenshot of a computer

Description automatically generated with medium confidence

*Conclusion:*

In conclusion I think that this app includes some great features such as having different difficulty modes so that users can tailor their workout to their needs. However there is very minimal personalisation available so the user may feel unmotivated to do a workout that isn’t really tailored to them. Also, the data section is great, showing how active the user is and showing their data such as calories and time. However, it could be better if it could show the user roughly how long it would take to reach their desired weight so that the user will feel more motivated. Also the app doesn’t include anything about cardio such as running with distance and time which could be very useful. Overall it is a very good app and is very user-friendly and I feel like I can incorporate some of these amazing features it has into my product such as the feature to choose your difficulty level.

* + 1. **App No.2 – “MyFitnessPal”**

The current process is as follows:

1. When you open the app there are multiple sections such as a Diary, Progress and settings. The Diary Section consists of Breakfast, Lunch and Dinner section where you can input your daily meals and it will tally up the calories in total. This is is really important when trying to keep track of your calories when staying in a calorie deficit or a calorie surplus. However, could be improved to show the different types of suggest foods for your goal.
2. The Progress section consists of regular weight entries and shows a graph of weight

change over a month. This is extremely useful as it allows the user to understand how they have progressed over a period of time and provides them with motivation to do better. However, to improve this, the graph could show a line of best fit that can show the user how long it will take to reach a certain goal if they continue along their current path.

1. The Settings sections is really useful as it consists of premade plans, recipes, routines and challenges for the user to do so that it will help the user achieve their goal. To improve this the user should be able to setup custom plans and recipes themselves to be used frequently so that it is more personalised for them.

Graphical user interface, application

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedGraphical user interface, application, Teams

Description automatically generated

*Conclusion:*

In conclusion, the app has some great features such as allowing the user to log their meals so that everything is easy to keep track off and easy to work out what kind of calories go into each meal. It also shows the users progress in the form of a line graph which is extremely useful as this will motivate the user to continue their path. However, to make this better it would be useful to have a line of best fit and even calculate the standard deviation to forecast the user’s future results if they continue the current path, this is useful as it gives the user incentive to carry on their path because it shows and achievable goal.

* 1. **System Flowchart**

**Diagram

Description automatically generated**

The flowchart depicts the current system and it allows me to identify the inefficient or problematic parts of the process. For example, the user can only select pre-made workouts and isn’t able to customise a workout routine for themselves. Also, when presenting the users data, the user can only see a list of all their data but it is hard to track their progress that way, therefore there requires a graph to represent how the user has progressed in terms of calories burnt and the duration, using the graph to calculate the mean timings and the inter-quartile range and standard deviation allows the user to notice how consistent they are and if they’re results are reliable or if it’s just a one off.

* 1. **Prospective Users and Acceptable Limitations**

**1.6.1 Prospective Users**

As of right now, there is 1 prospective user, my primary client. Since my primary client is an 18-year-old student he is very computer literate which means his skills shouldn’t suppress the complexity of the new system. However, it is still important to maintain ease of use throughout the system so that he won’t need to waste time navigating the app and also allows for newer users to easily understand the system.

**1.6.2 Acceptable Limitations**

The limitations to the system are as follows:

* Hardware and software constraints – since it is a fitness program it must be accessible wherever you are easily, so ideally it should be on mobile and there is a limited amount of software for me to design the system.
* My skills – The program cannot be too complex for me to solve using the programs and resources I have, since I will be using Python for this program there are many different paradigms I can work with.
* Time constraints – I have a limited amount of time to work on this and it must be completed by Easter 2021

**1.7 Data Flow Diagrams**

After taking a look at the variety of current systems available I have decided to create a DFD for the “Home Workouts” app to allow me to understand how data is transferred and processed throughout the system.

* + 1. **Initial Setup**

Showing how data is passed when a user sets up a new account.

Diagram

Description automatically generatedLevel 0:

Diagram

Description automatically generatedProcess 1 Level 1

* + 1. **Selecting Workouts**

When selecting a new workout, the user must first select the difficulty of the workout and then proceed to select which specific workout they want to do. This data is then stored in the database where it can then be retrieved to show users history with the app.

Level 0:

Diagram

Description automatically generated

Process1 Level 1

Diagram

Description automatically generated

* + 1. **Storing and representing user data**

Once a user completes a workout, their data accumulated from the workout is stored in a database and can be shown on the progress page so the user can see their data and their progress.

Level 0:

Diagram

Description automatically generated

Process 1 Level 1

Diagram

Description automatically generated

**1.8 Storyboard**

**Menu Screen**

**Login / Register**

A picture containing text, person, indoor, electronics

Description automatically generatedGraphical user interface

Description automatically generated with medium confidence

A picture containing text, sky, people, gear

Description automatically generatedLogo

Description automatically generatedA picture containing person

Description automatically generated

**User Settings**

Users can access and change their settings at any time, changing their personal info as well as changing protected data such as login details.

Users can check their progress by the graphs produced from collecting user’s workout data over a period of time.

User can select a new workout by difficulty and what muscles they want to target.

**New Workout**

**Progress Page**

User opens menu screen and can navigate the app from there from starting a new workout to checking their progress.

User will log in to the app with premade login details or register with new details

**1.9 Smart Objectives**

**1.9.1 Objective 1**

* Allow user to enter and regularly update their details such as age, weight and height and set a goal weight. This allows the user to be able to regularly keep record of their progress.
* Once the workout has been done, data such as calories burnt, duration, date etc. are all recorded into a database
  + 1. **Objective 2**
* Create a menu section where the user can select what they want to do
* They can choose to either use a pre-made workout routine for different types of muscles.
* If they choose pre-made, they can then select the difficulty e.g. beginner, intermediate, advanced, etc.
  + 1. **Objective 3**
* Once the difficulty has been chosen, they can select the type of workout they wish to do e.g. abs, chest, arms etc.
* Each workout type will have different exercises and different calories per each exercise, but will all have same number of exercises
  + 1. **Objective 4**
* Make sure the exercises fit the requirements of people with all types of goals. Such as people trying to gain muscle and people trying to lose fat.
* Create programs that are tailored to much more experienced fitness users and even accommodating beginners. Such as having high intensity workouts for experienced users and lower intensity for beginners.
* Have different levels of difficulty for workouts
* Thus, allowing the program to become more diverse and when people progress, they can easily just move to the next difficulty.
  + 1. **Objective 5**
* Sequentially go through each exercise and print the workout and after a set duration, start the next exercise
* Show the no. of reps and calories for each exercise
* Show the exercise with the most calories
  + 1. **Objective 6**
* Formulate a graph of user’s daily calories burnt each day so the user can see how well they are doing each day
* From the graph calculate the mean and standard deviation to show the user what they are averaging and how they can stay on track
* This will provide the user with motivation to keep going.

1. **Design**
   1. **Descriptive Overview of the Project**

By using the existing system, I have learnt a lot about its function and analysing its properties I have been able to determine which features worked well, which ones that were unnecessary or inefficient for the user and even features that the system lacked altogether. Therefore, these are the objectives I want to include for my new system.

1. Allow user to enter and regularly update their details such as age, weight and height and set a goal weight
   1. Overtime plot a graph of user’s weight and heights changes so users can easily see the progression
   2. Calculate the mean, standard deviation and variance from the graph to show the users progression
   3. Show the users regression line to portray how they will improve if they continue to follow their current path.
2. Create a menu section where the user can select what they want to do
   1. They can choose to either use a pre-made workout routine or customise their own
   2. If they choose pre-made, they can then select the difficulty e.g. beginner, intermediate, advanced, etc.
   3. Once the difficulty has been chosen, they can select the type of workout they wish to do e.g. abs, chest, arms etc.
   4. Once the workout has been done, data such as calories burnt, duration, date etc. are all recorded into a text file
3. Daily reminder to do a workout
4. At the end of the week a graph is shown of how many calories were burnt each day of the week along with how long you worked out each day and for the total of the week.
5. At the end of the month it will show a graph of how the user’s weight and height changed throughout the month.
6. After a week, the week’s data in the text-file is copied to a database and is then cleared, fresh for the next week
7. Shows personal best times and gives daily motivational quotes to boost the use.
8. Use the weekly data to create a graph projecting how long it would take to reach their goal weight by continuing the same routine.
9. Workout the mean calories burnt and form a linear graph, calculate the standard deviation and the inter quartile range to show the user how consistent they are.

**2.2 Structure Chart**

**Diagram, schematic

Description automatically generated**

This structure chart models how data is passed around in my new proposed system. This allows me to clearly identify the whole flow of information and processing and pinpoint key areas where I need to develop solutions to solve the problems in data flow. From the structure chart you can see that most of the data flows in and out of the database so the database must be secure and be able to handle fast data transfers and changes of data, so to ensure this the database must be normalised in 3NF to provide optimal efficiency. In some parts, data is seen to be flowing in both directions, therefore there must be a system to avoid any confusion with data and allow data to be inputted and checked correctly.

* 1. **GUI Design**
     1. **Home Page**

This is the Home Page. The first page the user sees when they open the application. For an app such as this the home page must be simplistic and straight to the point so that it allows the user to focus on their main target, to work out.

Once the username and password have been inputted the user will then click the Login button and the validity of the details are confirmed with the information in the database.

There are input sections for both the username and password that the user has premade, the username must be unique, and the password must fit a requirement.

Graphical user interface, application

Description automatically generated

LionCarrot89

VishalUser12

If the user doesn't already have an account, they can click the Register Account button which is clearly visible to create a new account, which is show on the next page.

* + 1. **Register Page**

This is the page seen when the user chooses to register a new account. To create a new account the user must have a unique username and also a password that meets the requirements of being 8 characters long and including both uppercase and lower-case letters to provide optimal security.

Once the username and password have been inputted the user will then click the Login button and the validity of the details are confirmed with the information in the database.

There are input sections for both the username and password. The username must be unique, and a notification will appear if a certain username has already been taken.

**Graphical user interface

Description automatically generated**

LionCarrot89

LionCarrot89

VishalUser12

If the user doesn't already have an account, they can click the Register Account button which is clearly visible to create a new account, which is show on the next page.

* + 1. **Menu Page**

This is the page is the Menu page. This is the central hub for the whole program and is where the user can decide which area of the program they wish to access. The design is simple and straight to the point to allow the user to get straight into their workout.

The different pages can be selected by physical icons which the user can select. This provides a much more efficient system as user doesn't need to manually type out a destination or select a value from a drop-down menu as this will take up a lot of time.

**A picture containing bird

Description automatically generated**

An arrow indicating to go back to the previous page. If the user clicks this the page will return to the one it was on previously. This is useful since the user doesn't have to close the entire program and open it again.

* + 1. **New Workout Page**

This page is the New Workout page and is where the user can select the difficulty and type of workout they wish to do. The difficulties incldue: Beginner, Intermediate and Advanced. The workout types include: Abs, Chest, Legs, Arms, Back and full body. The page for loading a previous workout also follows the same structure as this.

The arrows signify drop down menus. Since there are only a few options and the user doesn't know them it wouldn't be usefull to ask the user to manually input them. Drop down selection allows for a much easier selection process.

The user must select a difficulty out of Beginner, Intermediate and Advanced and each difficulty has a different number of reps as it gradually increases throughout to provide much of a harder challenge.

Graphical user interface, application

Description automatically generated

Chest

Beginner

The user must also select a workout type from Abs, Chest, Legs, Arms, Back and full body. This combined with the value for the difficulty will provide the user with the perfect workout routine for them to follow.

* + 1. **Current Workout Page**

This page the user sees when they select a workout. This page follows the same structure for all workout types such as Chest and Back. This page consists of a list of exercises the user must follow along with the reps and also images of how to carry them out. There is also a timer for how long they have. With the combination of visual aids and colour, this provides the user with much more motivation as it seems more fun and simplistic.

The different exercises are listed and each one also describes the number of reps to be done. The exercises are in different colours which represents which ones have been completed and which ones haven't. The ones in green have been completed and the ones in yellow are yet to be completed.

The exercises also contain diagrams of how to carry them out next to them to allow for beginners or users that have never done a particular exercise to understand how to actually do it properly with correct form and technique so as to provide optimal results.

Text

Description automatically generated

The page also contains a timer that tells the user how long they have left to complete the workout. A set time is required as being able to finish a workout in the shortest time possible will increase the users heart rate much higher so that they can burn a lot more calories.

* + 1. **Current Workout Page**

This is the progress page. This page shows the users progress over the period they have used the app. Showing trends in progression using calories burnt each day. By using statistical calculations learnt from A-Level mathematics I can calculate the standard deviation and variance of the client’s values to allow them understand how consistent they are.

From the data shown in the graph, the mean, standard deviation and variance are calculated and displayed to show the user they're regular values and how much of a range there is between values so that in the future they can improve to stay more consistent

From the data, by creating a line of best fit, values could be calculated to project where the user will be in 3 months, 6 months etc. This will provide motivation to the user as they will understand that if they keep their current routine up for a set period of time, they will achieve their goal.

Chart, line chart

Description automatically generated

20kg

16kg

10kg

8kg

6kg

2kg

1031.69

32.12

432.54

By using data stored in databases about the user such as calories burnt on each day, a graph is created for each week of workouts. This shows how many calories are burnt in total each day. This allows the user to understand if they are staying consistent, gradually increasing or even regressing. Preferably the user should remain consistent to provide certain results as a constant increase of calories burnt every day is not sustainable in the long run.  A visual graph like this is extremely useful as it is clear and straight to the point and anyone can easily understand it.

* 1. **IOPS Chart**

The chart seen below depicts what happens to data in the new system at a very basic level, labelled in terms of input/output, processing and storage.

Table, treemap chart

Description automatically generated with medium confidence

* 1. **System Flowchart**

The new flow chart below portrays the different changes and improvements I have made to the previous flowchart and the system regarding aspects such as using a database and drawing graphs.

It shows how the user would normally navigate through the system and also shows the transfer of data throughout the system.

Diagram

Description automatically generated

* 1. **Class Design**

By using the OOP paradigm, they are what some my class systems would look like by modelling the different exercises as subclasses.

Diagram

Description automatically generated

The above class diagram shows the relationship between the classes with “WORKOUT” as the super class and “LEGS”, “CHEST” and “ABS” being the sub classes. All the Subclasses carry the attributes of the parent class but have attributes of their own relating to their specific exercise. By using private attributes, this keeps the attributes secure in each subclass.

By representing the data by using classes this allows for a much more efficient and productive approach to viewing the code.

* 1. **Security and Integrity of Data**

Since the system will be using client’s personal data a certain level of security is require such as a authentication system such as user logins to allow only the user to access their personal data. To further secure the data, a layer of encryption could be implemented to secure the data fully. Furthermore, I must make sure that the data in the database is not public and remains private to make sure it is not available to the public and remains exclusive to the respective client.

To protect the integrity of the data, all the data entries will be met with strict validation rules. When selecting the difficulty or exercise a drop-down menu or tick-box could be used to reduce the risk of typing the incorrect result and crashing the system and also minimises free text input so doesn’t accidently crash the system and delete any data.

* 1. **Database Design**

Diagram

Description automatically generated**2.8.1 ER Diagram**

PK = Primary Key

FK = Foreign Key

To manage the quantity of data needed in this project I require a database to easily allow me to have links between different aspects of data through a relational database.

The main tables are:

* CUSTOMER = which contains all the customers unique data
* LOGIN = which contains the unique login details of each customer
* WORKOUTS = which contains the data for each workout a customer completes such as the exact routine, and the calories burnt

Overall these 3 tables are linked by a central table called USER\_WORKOUTS which links each customer with their login data and assigns each workout to each customer that completed it. The tables CUSTOMER and LOGIN are also separately linked by the USERID table that provides an interface between login details and customer details; this is especially crucial if the user hasn’t

done aa workout yet meaning the USER\_WORKOUTS table will be empty but there still needs to be a link between LOGIN and CUSTOMER and that is why the table USERID is necessary.

By using SQLite, I can create and update database tables and data with Python, which allows me to indirectly execute SQL queries and retrieve data. By using primary keys which can act as foreign keys this allows to access data across multiple tables to allow for a much more efficient way of storing and accessing data.

* + 1. **SQL Queries**

|  |
| --- |
| **Displaying users first name and surname when given the user login** |
| SELECT USER\_WORKOUTS.customer\_firstname , USER\_WORKOUTS.customer\_surname FROM USER\_WORKOUTS WHERE LoginID = ? ; |

|  |
| --- |
| **Adding a new workout to the database** |
| INSERT INTO WORKOUTS VALUES (?, ?, ?, ?) ; |

|  |
| --- |
| **Updating password** |
| UPDATE Login SET (?) WHERE password = ? ; |

|  |
| --- |
| **Deleting a user** |
| DELETE FROM Login WHERE LoginID = ‘?’  DELETE FROM CUSTOMER WHERE CustomerID = ‘?’ ; |

|  |
| --- |
| **Display all workouts that last 600 seconds** |
| SELECT workout\_routine FROM WORKOUTS WHERE duration = 600 ; |

|  |
| --- |
| **Adding a new user** |
| INSERT INTO CUSTOMER VALUES (? , ?, ?, ?, ?) ; |

|  |
| --- |
| **Displaying all existing workouts that burn 400 or more calories** |
| SELECT workout\_routine FROM WORKOUTS WHERE calories > 400 ; |

|  |
| --- |
| **Displaying all existing workouts that burn 200 or less calories** |
| SELECT workout\_routine FROM WORKOUTS WHERE calories <200 ; |

|  |
| --- |
| **Displaying all workouts done on a specific date** |
| SELECT UserWorkoutID from UserWorkouts WHERE workout\_date = /; |

|  |
| --- |
| **Displaying the user’s first name through tables UserID and Customer with a given CustomerID which is used as a foreign key** |
| SELECT Customer.customer\_firstname FROM UserID INNER JOIN Customer ON UserID.CustomerID = Customer.CustomerID WHERE Customer.CustomerID = ? |

|  |
| --- |
| **Displaying the user’s username through tables UserID and Login with a given LoginID which is used as a foreign key** |
| SELECT Login.username FROM UserID  INNER JOIN Login ON UserID.LoginID = Login.LoginID WHERE Login.LoginID = ? |

* 1. **Algorithm Design**

To create the program, I will need to plan the overall algorithms for it, describing the goals, draw backs, solutions and examples of inputs and outputs.

**2.9.1 Complex A-Level Mathematics (Standard Deviation Algorithm)**

Description:

* The algorithm will take values of the different weights each day as x and the total number of days as N and will calculate the standard deviation to show the amount of which every value within the dataset varies from the mean.
* The reason I have decided to use this algorithm is because it is the most efficient to calculate the standard deviation. By using functions this allows me to reuse the code anywhere within the program

Why is it needed?

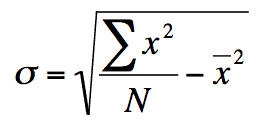
* I learnt about this algorithm from A-Level Mathematics course in the Statistic modules. The standard deviation represents the mean amount of which each value within the dataset varies from the mean.
* In this situation it is useful because it allows the user to understand whether they are on track or not and how far away from their regular mean they are so that they can assess that and get back on track.

Inputs & Outputs:

* Mean (input)
* Total of x^2 (input)
* Total number of days (input)
* Standard Deviation (output)

Problems:

* If an invalid number such as N as 0 is inputted, it may result in an error
* If a string is inputted it, there may be an error.



**Pseudo-code:**

list = [4,5,6,7,8,9]

N= list\_length

Def mean\_squared(N, list):

Z=0

for y in range N:

x = Z + list[y]

z= x

return z

P = mean\_squared(N, list)

mean = z / N

Def xsquared\_mean(N, list):

K= 0

squared\_list = []

For y in range N:

total\_xsquared = K + (N\*N)

K = total\_xsquared

return K

Q = xsquared\_mean(N, list)

Variance = Q – P

Standard\_deviation = square roott (Variance)

**2.9.2 Selecting a Workout**

Description:

- The algorithm will take the users input for a workout and load up the corresponding workout routines, total calories and duration. I have chosen to use this algorithm using functions to allow me to reuse this code anywhere within the program efficiently.

Inputs & Outputs:

* Workout difficulty (input)
* Workout type (input)
* Workout plan (output)

Problems:

* The user may input an option that isn’t available, therefore repeat the question and telling the user to choose from the available options.
* The user may input the wrong workout and will want to choose a different one so need to be able to go back and try again.

**Pseudo-code:**

DEF Menu ():

         valid = False

         WHILE valid = False:

                        OUTPUT “MENU, Press 1 for Beginner, 2 for Intermediate and 3 for advanced”)

                        IF INPUT = 1:

                             beginner\_workouttype()

                             valid = True

                       ELIF INPUT = 2:

                                 intermediate\_workouttype()

                                 valid = True

                       ELIF INPUT = 3:

                                 advanced\_workouttype()

                                 valid = True

                       ELSE:

                                    valid = False

DEF beginner\_workouttype():

         valid = False

         WHILE valid = False:

                 OUTPUT “Press 1 for Ab workout, 2 for chest workout, 3 for legs, 4 for triceps, 5 to go to menu”

                       IF INPUT = 1:

                            beg\_ab\_workout()

                            valid = True

                       IF INPUT = 2:

                            beg\_chest\_workout()

                            valid = True

                       IF INPUT = 3:

                            beg\_leg\_workout()

                             valid = True

                       IF INPUT = 4:

                            beg\_tricep\_workout()

                            valid = True

                       IF INPUT = 5:

                            Menu()

                             valid = True

**2.9.3 Plot User Data on a Graph**

Description:

- The algorithm will take the users data over the month, such as calories burnt each day and plot a line graph to show the trends.

Inputs & Outputs:

* Day of the month (input)
* -calories burnt on that day (input)
* Graph (output)

Problems:

* Somedays the user may not have done anything, so will causes a gap in the graph, to solve this, draw a line of best fit.

**Pseudo-code:**

Data = [424, 432, 46, 74, 345]

Date = [1 , 2 , 3 , 4 , 5 , 6 , 7]

Def graph(x):

y = (gradient\*x) + intercept

return y

Plt.plot(Date, data)

plt.plot(dateNo, data)

plt.xlabel('date')

plt.xlabel('calories')

plt.title('Week 1')

**2.9.4 Recursively Calculating the sum of all values and calculating the mean**

Description:

- The algorithm will take the users data over the week and add them all together to find the total and calculate the mean value. I have chosen to use recursion for this as it provides optimal efficiency and allows the problem to be solved with the minimal number of lines.

Inputs & Outputs:

* calories burnt on each day (input)
* Sum of all calories burnt (output)
* Mean value (output)

Problems:

* Somedays the user may not have done anything, so will causes an error whilst checking for value

**Pseudo-code:**

def total(data):

x = len(data)

if x = 1:

return data[0]

else:

return data[0] + totalX(data[1 to end])

def Mean(totalvalue, length, data ):

mean = total value/ length

return mean

print(mean)

print(total)

**2.9.6 Exercises Queue and Queue Operations**

Description:

* This algorithm will create a queue of a set length as an attribute. Then, it contains methods such as enqueue which will provide an interface for adding values to the end of the queue.
* It also contains a method called dequeue which will remove the value from the front of the queue and return the value
* It also contains the methods isEmpty and isFull to check whether the queue is full or not
* Finally, the method show, is used to return the current state of the queue.
* Contains a front and rear pointer

Why is it needed?

* This is needed when adding new exercises to a queue and sequentially removing them once a set time has elapsed.
* The Queue follows a First In First Out System meaning the first item in the queue is the first one to be removed
* This is important because the order of the exercises need to be maintained and provides the workouts to the user one at a time
* Since a queue is an abstract data structure and is dynamic this also means it is a much more memory efficient method of storing data.

Inputs & Outputs:

* Max size of the queue (input)
* Queue (output)

**Pseudo-code:**

DEFINE CLASS workoutQueue:

DEFINE FUNCTION \_\_init\_\_(self, max\_size, size=0, front=0, rear=0):

SET self.queue TO [[] FOR i IN range(5)]

SET self.max\_size TO max\_size

SET self.size TO size

SET self.front TO front

SET self.rear TO rear

DEFINE FUNCTION enqueue(self, data):

IF not self.isFull():

SET self.queue[self.rear] TO data

SET self.rear TO int((self.rear + 1) % self.max\_size)

self.size += 1

ELSE:

OUTPUT('Queue is full')

DEFINE FUNCTION dequeue(self):

IF not self.isEmpty():

SET x TO self.queue[self.front]

SET self.front TO int((self.front + 1) % self.max\_size)

self.size -= 1

ELSE:

OUTPUT('Queue is empty')

RETURN x

DEFINE FUNCTION isEmpty(self):

RETURN self.size EQUALS 0

DEFINE FUNCTION isFull(self):

RETURN self.size EQUALS self.max\_size

DEFINE FUNCTION show(self):

RETURN self.queue

ELSE:

OUTPUT('Queue is full')

DEFINE FUNCTION dequeue(self):

IF not self.isEmpty():

SET x TO self.queue[self.front]

SET self.front TO int((self.front + 1) % self.max\_size)

self.size -= 1

ELSE:

OUTPUT('Queue is empty')

RETURN x

DEFINE FUNCTION isEmpty(self):

RETURN self.size EQUALS 0

DEFINE FUNCTION isFull(self):

RETURN self.size EQUALS self.max\_size

DEFINE FUNCTION show(self):

RETURN self.queue

**2.9.7 Complex A-Level Mathematics & Hashing (Trigonometric Hashing Algorithm)**

Description:

* This algorithm makes use of a formula derived from the Trigonometry section of the Edexcel A-Level Mathematics subject. The formula is shown below

**Text

Description automatically generated with medium confidence**

* This formula represents a function d with parameter t, it makes use of trigonometric attributes like cosine and sine to produce a unique hash given an integer input

Why is it needed?

* The use of a hashing algorithm is crucial to make sure that the user’s login details are secure and are inaccessible.
* The use of a complex hashing algorithm makes the hash more secure and more unique and using knowledge from my other A-Level subject allowed me to do this.

Inputs & Outputs:

* User password (input)
* Hash index (output)

**Pseudo-code:**

DEFINE FUNCTION hashing(value):

SET passw TO []

FOR x IN range (len(value)):

SET passw TO ord(value[x])

SET total TO 12 + (5.6 \* math.cos((360\*passw) / 365)) + (1.4 \* math.sin((360 \* passw) /365 ))

RETURN int(total)

* + 1. **Recursive Validation Algorithms**

Description:

* Validation is required when navigating the program
* Whenever a user wishes to access a different section of a program, they are required to input an integer that corresponds to the appropriate section, however a user may accidently input a value that is not an integer or not within the range of acceptable values.
* This could cause the program to crash, to stop this a method of validation is required.
* The function intValid uses recursion to check whether the input is actually an integer or not, if not it calls the function again until it is acceptable.
* The function rangeValid uses recursion to check whether the input is within the range of acceptable inputs, if it is not, it will call itself again until the condition is met.

Why is it needed?

* This is necessary because if the user input isn’t validated this could cause the program to crash which results in a poorly designed system and the user not liking the program.
* Also, a recursive technique is the best choice to do this, although an iterative solution is easier to program, overall a recursive program is much more efficient in terms of space and memory.

Inputs & Outputs:

* User choice (input)
* Valid user input (output)

**Pseudo-code:**

# recursively checking to see IF user choice is an integer and IF it isn't, asks again

DEFINE FUNCTION intValid(response):

'''

SET check TO True

WHILE check EQUALS True:

'''

TRY:

SET integer\_check TO int(INPUT(response))

except ValueError:

OUTPUT("Please select an integer value option, try agian")

intValid(response)

ELSE:

RETURN integer\_check

#recursively checking to see IF user choice is within the options and IF it not, it asks again

DEFINE FUNCTION rangeValid(choice, lowest, highest):

'''

SET range\_check TO True

WHILE range\_check EQUALS True:

'''

IF choice < lowest or choice > highest:

OUTPUT("please select an option that is available")

SET choice TO int(INPUT("Please select the integer value of what you want to access: "))

rangeValid(choice, lowest, highest)

ELSE:

RETURN choice

**2.9.10 Merge Sort**

Description:

* A merge sort is a method of ordering an array by splitting it into smaller lists, and then reforming them. It is quicker than bubbles sort and has a time complexity of O(nlogn)
* The merge sort here takes an array of calories for each exercise and sorts them in order of smallest to largest in order to find out the exercises that burns the most and the least calories.

Why is it needed?

* A merge sort is necessary because the list of calories is unordered, therefore, to find the exercises that burn the most and least calories, the list needs to be sorted.
* The best way to sort the list is through a merge sort, although it is harder to code than bubble sort it is much more efficient in terms of time taken to sort a list, especially for larger lists.

Inputs & Outputs:

* Unsorted Array (input)
* Sorted Array (output)

**Pseudo-code:**

## divide the arrays into halves and sort them recursively UNTIL we get arrays with 1 element

DEFINE FUNCTION mergeSort(array, first\_index, second\_index):

IF first\_index >= second\_index:

RETURN

SET middle TO (first\_index + second\_index)//2

mergeSort(array, first\_index, middle)

mergeSort(array, middle + 1, second\_index)

mergeArray(array, first\_index, second\_index, middle)

**Pseudo-code:**

## merging the sub arrays into a ordered array

DEFINE FUNCTION mergeArray(data, first\_index, second\_index, middle):

SET first\_ver TO data[first\_index:middle + 1]

SET second\_ver TO data[middle+1:second\_index+1]

SET first\_ver\_index TO 0

SET second\_ver\_index TO 0

SET sorted\_index TO first\_index

WHILE first\_ver\_index < len(first\_ver) and second\_ver\_index < len(second\_ver):

IF first\_ver[first\_ver\_index] <= second\_ver[second\_ver\_index]:

SET data[sorted\_index] TO first\_ver[first\_ver\_index]

SET first\_ver\_index TO first\_ver\_index + 1

ELSE:

SET data[sorted\_index] TO second\_ver[second\_ver\_index]

SET second\_ver\_index TO second\_ver\_index + 1

SET sorted\_index TO sorted\_index + 1

WHILE first\_ver\_index < len(first\_ver):

SET data[sorted\_index] TO first\_ver[first\_ver\_index]

SET first\_ver\_index TO first\_ver\_index + 1

SET sorted\_index TO sorted\_index + 1

WHILE second\_ver\_index < len(second\_ver):

SET data[sorted\_index] TO second\_ver[second\_ver\_index]

SET second\_ver\_index TO second\_ver\_index + 1

SET sorted\_index TO sorted\_index + 1

RETURN data

* 1. **Data Types and Structures**

2D arrays:

* I will be using 2D arrays to store the user’s temporary data for the runtime and also for the workout routine lists, using a mix of 1D and 2D arrays to create easily identifiable tables.
* I could also use it to represent the length of all the values when calculating the mean or the standard deviation

Hash Table:

* I will be using hash tables to hash the user’s passwords and encrypt it to make sure that they are secure and only accessible to them. Hashing will allow me to easily locate the different users’ passwords and details to use every time the user logs in.
* Also, I will be using to store the data for the different workout routines so that an easily identifiable array is formed and I can easily find and get the values I need.

Queue:

* I will be using a queue when I collect the workout routines and form an array then it will follow a FIFO routine where whichever item entered first will be the first one out so the user will do the workout routines in that order all the way until the list is complete.

A hashing algorithm will also need to be created using complex mathematical functions to securely hash the key, making sure there aren’t any collisions.

**2.11 Library Software**

For this program I will be needing to use a handful of different library software to make coding the solution slightly simpler.

Math:

* I will be needing to use the math library function to have access to values such as trigonometrical and statistical values to both create a comp
* lex hashing algorithm and also calculate the standard deviation for the user’s continuous data.

Random

* I will be needing the random library to assign random integer values to the keys for data to hashed
* Also, for the primary keys for all the different database tables.

Matplotlib and SciPy

* I will be needing this library so that I can plot a scatter graph of the user’s data over the week and also a line of regression. I also need it to use trigonometry values such as cosine and sine.
* SciPy will be used to calculate the statistical values needed to plot a regression lin

SQLite3

* I will be using the sqlite3 library because it provides me with an interface to create and make changes to a data base where I store data
* It allows me to directly make changes to the database using SQL queries within Python

Datetime

* Allows me to find todays date in order to store alongside workout information so the user will know when they took a certain workout.

1. **Technical Solution**
   1. **General Solutions**
      1. **Login / Register Screen**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | *#This subroutine runs the opening page that the user first sees when the program is run*  **def** Opening():  **print**("Fitness App")  **print**("-----")  **print**("1. Login")  **print**("2. Register")  **print**()  *#intValid recursively checks whether the input is an integer or not and makes sure it is*  value\_integer\_check = intValid ("Please select the integer value of what you want to access from the menu: ")  *#rangeValid recursively checks whether the input fits the given length required and makes sure it is*  value\_userInput = rangeValid (value\_integer\_check, 1, 2)  OpeningChoice(c, value\_integer\_check)  **def** OpeningChoice (c, value):  **if** value == 1:  Login(c)  **elif** value == 2:  Register(c) |

* The code above is what is used to produce the opening screen the user will see when they first run the program.
* When the procedure Opening () is called they will be asked whether they wish to Login with a premade account or Register a new account.
* Their inputs are then validated and passed into the subroutine OpeningChoice, which given the user’s choice to Login or Register will then call upon the respective function.
  + 1. **Login Screen**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | **def** Login(c):  **print**()  **print**("Login")  **print**("-----")  *#username and password input*  username = input("Username: ")  password = input("Password: ")  *# Apply hashing algorithm and return the hash produced and store in the variable ID*  ID = hashing(password)  *#Select values from the database using functions that contain the SQL Queries to make the code more efficient and reusable.*  val = selectUsernameFromUserID(c, ID)  w = selectCustID(c, ID)  y = selectNameFromUserID(c, w)  **if** val == username:  **print**()  **print**("Welcome",y,"!")  time.sleep(1)  Menu(w)  **else**:  **print**()  **print**("incorrect login, try again!")  *#Recursion used to call the function again if the login details do not match*  Login(c) |

* Once a user selects to Login to their account, they are presented with a Login page where they can input their Username and Password.
* Once all the details are inputted, a hashing algorithm is applied to the password which was applied when they registered the account, the hash produces relates to the index where the username and password are stored in the database.
* Once the password has been hashed, the hash is then used to select the corresponding username from the database, by calling the function “selectUsernameFromUserID”.
* Next, the customer ID is also selected from the database with the function “selectCustID”, which is passed throughout the whole system so that the program knows which user is currently using the program.
* Once that is done, the program checks the inputted username with the corresponding username in the database. If they both matches, the Login is accepted, and the “Menu” function is run as the program allows the user to enter.
* If the values don’t match, then and error is presented, and the user will have to re-enter the correct details, which is done by recursion.
  + 1. **Register Screen**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | **def** Register(c):  **print**("Register")  **print**("--------")  **print**("1. Your username and password must be between 6 to 60 characters and contain both letters and numbers and atleast 1 uppercase letter")  **print**()    *# input new username and password*  user = input("Username: ")  passw = input("Password: ")  *#Check validity of username and password using classes, both objects make use of the same methods but are treated differently due to their classes (polymorphism)*  userVal = UserValid(user)  passVal = PassValid(passw)    *# Loop through the methods of each class checking the validity of the inputs*  **for** choice **in** (userVal, userVal):  user = choice.len\_valid()  user = choice.dig\_valid()  user = choice.upper\_valid()  user = choice.lower\_valid()    **for** choice **in** (passVal, passVal):  passw = choice.len\_valid()  passw = choice.dig\_valid()  passw = choice.upper\_valid()  passw = choice.lower\_valid()  ID = hashing(passw)  *#Insert username, password and LoginID into the DB using a function*  insertLogin(c, ID, user, passw)  user\_info(c, ID) |

* If they user decides to Register a new account, then the subroutine above is then called.
* It will ask the user to input their chosen Username and Password
* Once inputted, a validation processes using classes UserValid and PassValid will occur where each input is checked to see if it contains lowercase and uppercase letters along with at least one integer and must of a minimum length.
* Once all checks are complete, a hashing algorithm is applied to the password to produce a hash that represents the LoginID of the user which is also the primary key and index of which the values are stored into a database.
* Then finally, the LoginID, username and password are all inserted into the database using the subroutine insertLogin.
* Once that is done, since this is a new account the sub-routine user\_info is called to setup the user’s personal information.
  + 1. **User Info Setup**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | **def** user\_info(c, LoginID):  *# generate unique pseudorandom values for CustomerID and UserID*  CustID = random.randint(1, 100000)  UserID = random.randint(1, 100000)  **print**()  **print**("Loading user info page...")  time.sleep(2)  **print**("USER INFO SETUP")  **print**("---------------")  **print**()  *#Ask user to enter their details*  Fname = input("Firstname: ")  Sname = input("Surname: ")  weight = input("Weight(kg): ")  height = input("Height(cm): ")  dob = input("Date of birth(DD/MM/YYYY): ")  *#Insert user details into database tables Customer and UserID using functions*  insertCustomer(c, CustID, Fname, Sname, weight, height, dob)  insertUser(c, UserID, CustID, LoginID)  *#Given the UserID select the users name by the means of relational database tables and primary and foreign keys*  y = selectNameFromUserID(c, CustID)  **print**()  **print**("Welcome",y,"!")  time.sleep(1)  Menu(CustID) |

* Once a user registers a new account or decides to change user info from the menu section, the “user\_info” subroutine is then called.
* The subroutine, provides the user with a CustomerID and a UserID, which are used as primary keys in their respective database tables.
* Next, the program asks the user to input their personal details such as name, weight, and date of birth so that in the future it can be used to track the user’s progress.
* Once the user inputs all required data, each piece of data is then inserted into the corresponding database table using functions insertCustomer and insertUser
* Then the function selectNameFromUserID is called which returns the first name of the user through multiple relational database tables.
  + 1. **Menu Screen**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | **def** Menu(CustID):  **print**()  **print**("Menu")  **print**("----")  **print**("1. New Workout")  **print**("2. Review ")  **print**("3. Change user info")  **print**()  *#menu user inputs and validation*  menu\_integer\_check = intValid("Please select the integer value of what you want to access from the menu: ")  menu\_userInput = rangeValid(menu\_integer\_check, 1, 3)  menuDestination(menu\_userInput, CustID) |

* Once the user completes all their user details after Registering or after Logging in, they will be presented with a Menu screen, represented by the code above.
* The user will be able to navigate to 3 different sections:
* New Workout = where they select and begin a new workout
* Review = where they can look at a graph of their previous workout progress and see the mean calories burnt per day as well as standard deviation which allows them to understand if they are on track to reach their goal, what they need to improve on if they aren’t and also if they are then what they need to maintain to keep on track.
* Change user info = where they change their personal info such as weight and height so they can keep track of their progress.
* The input is then validated to make sure they choose and option that is available and that it’s an integer then the action is carried out.
  + 1. **Select Workout Difficulty**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | *#difficulty menu*  **def** difficultyList(CustID):  **print**()  **print**("Loading progress page...")  time.sleep(2)  **print**()  **print**("Difficulty")  **print**("----------")  **print**("1. Beginner")  **print**("2. Intermediate")  **print**("3. Expert")  **print**("---------")  **print**("4. Go Back")  **print**()  *#difficulty user inputs and validation*  diff\_integer\_check = intValid("Please select the integer value of the difficulty: ")  diff\_userInput = rangeValid(diff\_integer\_check, 1, 4)  **return**(diffDestination(diff\_userInput, CustID))  *# Depending on the user’s choice of difficulty a set value of reps and duration are passed into the workoutList function*  **def** diffDestination(userInput, CustID):  **if** userInput == 1:  workoutList(10, 600, CustID)  **elif** userInput == 2:  workoutList(15, 900, CustID)  **elif** userInput == 3:  workoutList(20, 1200, CustID)  **elif** userInput == 4:  Menu() |

* If the user decides to start a new workout they are met with a screen where they need to select the difficulty of their workout. That screen is represented by the code above.
* The user can select between 3 options: Beginner, Intermediate and Expert
* The validity of the input is checked and then the depending on the user’s decision a subroutine is called.
* The difficulty represents the reps and duration of a workout, Expert difficulty workouts will have higher reps per exercise as well as a longer duration than a Beginner difficulty workout.
* For example, if the user decides to choose beginner the reps are set at 10 reps per exercise which is passed into the workout List subroutine, whereas if they choose expert it will be 20.
  + 1. **Select Workout Type**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | *#workout menu list*  **def** workoutList(reps, duration, CustID):  *#2D array for each workout type allong with the correspoding exercises and calories for 1 rep of each exercise*  Abs = [["Crunches", "Mountain Climbers", "Sit Ups","Russian Twists", "Leg Raise"],[0.159, 0.327, 0.327, 0.234, 0.342]]  Chest = [["Push Ups", "Knee Push Ups", "Wide Arm Push Ups", "Incline Push Ups","Leg Raise"],[0.423, 0.342, 0.450, 0.123, 0.234]]  Arms = [["Push Ups", "Tricep Dips", "Bicep Curls", "Front Arm Raises", "Side Arm Raises"],[0.423, 0.334, 0.234, 0.145, 0.145]]  Leg = [["Squat", "Lunge", "Wall Calf Raises", "Side Lying Leg Lift Left", "Side Lying Leg Lift Right"],[0.324, 0.214, 0.134, 0.234, 0.234]]  Shoulder\_Back = [["Front Arm Raises", "Side Arm Raises", "Push Ups", "Pull Ups", "Knee Push Ups"],[0.145, 0.145, 0.423, 0.450, 0.342]]    **print**()  **print**("Workouts")  **print**("--------")  **print**("1. Abs")  **print**("2. Chest")  **print**("3. Arms")  **print**("4. Leg")  **print**("5. Shoulder & Back")  **print**("------------")  **print**("7. Go Back")  **print**()  *#integere and range validation*  workout\_integer\_check = intValid("Please select the integer value of what workout you want to access: ",)  workout\_userInput = rangeValid(workout\_integer\_check, 1 , 6)  *#since user’s inputs are in the range 1-7, a list is formed to get the user Input to accommodate for the users input, leaving the first index as 0 since that is not a viable input*  typeList = ['0','Abs','Chest','Arms','Leg','Shoulder & Back']  Type = typeList[workout\_userInput]  *#depending on the user Input, the parameters, reps, duration, the 2D array, name of workout and customerID are passed into the subroutine new\_workout*  **if** workout\_userInput == 1:  new\_workout(reps, duration, Abs, Type, CustID)  **if** workout\_userInput == 2:  new\_workout(reps, duration, Chest, Type, CustID)  **if** workout\_userInput == 3:  new\_workout(reps, duration, Arms, Type, CustID)  **if** workout\_userInput == 4:  new\_workout(reps, duration, Leg, Type, CustID)  **if** workout\_userInput == 5:  new\_workout(reps, duration, Shoulder\_Back, Type, CustID)  **if** workout\_userInput == 6:  new\_workout(reps, duration, Abs, Type, CustID)  **if** workout\_userInput == 7:  difficultyList()   * Once the user selects their difficulty for their workout the above code is then called to allow them to select their chosen workout type. * From lines 1 – 10 in the above section of code you can see a series of 2D arrays for each workout type which contain each exercise for that workout type with the corresponding calories that burnt for 1 rep of that exercise. * Once the user selects their workout, the input is validated and is passed as a parameter along with the 2D array corresponding to their chosen workout and the reps and duration which were determined from the difficulty they chose, are all passed as parameters into the function new workout. |

* + 1. **New Workout Display and Confirmation**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | **def** new\_workout(reps, duration, exercise, Type, CustID):  **print**()  **print**("Loading new",Type,"workout...")  time.sleep(2)  **print**()  **print**("---------------")  **print**(Type,"Workout")  **print**("---------------")    dataSet = []  array =[]    *#loop through a definite iteration which uses the class Workouts and the method start() to output the list of exercises along with the calories for each exericse*  **for** i **in** range (0, 5):  x = Workouts(duration, reps, exercise, 0, 1, i)  x.start()  y = reps \* (exercise[1][i])  array.append(round(y, 2))  *#loop through a definite iteration which adds all the values for calories onto an array*  **for** i **in** range (0, 5):  y = reps \* (exercise[1][i])  dataSet.append(round(y, 2))  *# merge sort sorts out the array in order of lowest calories to highest*  mergeSort(array, 0, len(dataSet)-1)  highest\_val = array[len(array)-1]    *#use the highest value in the sorted array to find the exercise that corresponds with that value with a binary search*  largest = binary\_search(dataSet, highest\_val, 0, len(dataSet)-1)    **print**()  **print**("The exercise that burns the most calories is",exercise[0][largest],"with",array[len(array)-1],"calories")    **print**()  **print**("Duration(seconds):",duration)  **print**()  *# ask user for confirmation to begin workout*  userInput = intValid("Press 1 to begin, Press 2 to go back to menu: ")  rangeV = rangeValid(userInput, 1, 2)  **if** rangeV == 1:  startWorkout(reps, duration, exercise, Type, CustID, array)  **elif** rangeV == 2:  Menu() |

* Once the user has selected what workout they wish to do, the code above lists all the exercises within the workout, the amount reps they need to do for each exercise (by using a definite iteration by the means of a for loop and makes use of the Workouts class), how many calories are burnt for each workout, the total duration and even tells you which workout has the most calories by the means of a merge sort and binary search algorithms.
* Once presenting the user with all the information for the workout, the programs asks for confirmation of whether the user wants to start thew workout or not. If the user accepts,

then the subroutine startWorkout is run.

* + 1. **Starting New Workout and Saving Data**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | **def** startWorkout(reps, duration, exercise, Type, CustID, array):  *# for loop adds all the calories of each workout to a queue*  q = workoutQueue(5)  **for** i **in** range (0, 5):  q.enqueue(exercise[0][i])  queue = q.show()  *# Output the workout to the user using a for loop for each workout, after a set time, start the next loop*  **print**(Type,"Workout")  **print**("---------------")  **for** i **in** range (0, 5):  **print**("Start doing",q.dequeue(),"for",reps,"reps")  time\_per\_exercise = duration / 5  time.sleep(time\_per\_exercise)  *# get todays date for workout date*  today = date.today()  d1 = today.strftime("**%d**/%m/%Y")    calories = totalCal(array)    *# generate user’s workout ID*  WorkoutID = random.randint(1, 100000)  ID = random.randint(1, 100000)    *# Insert workout details into database tables*  insertWorkouts(c, WorkoutID, Type, calories, duration)  insertUserWorkouts(c, ID, CustID, WorkoutID, d1)  **print**()  **print**("Workout Complete! Well done!")  input("Going back to Menu now....")  menu(CustID) |

* Once the user confirms that they wish to start their workout, the above subroutine startWorkout is run which starts the workout routine.
* Each exercise in the workout is firstly enqueued in lines 6-8 in the above code.
* A definite for loop from lines 14-18 in the code above dequeues each value from the queue in a “first in first out” system and outputs each one with its corresponding reps, then calculates the time needed for each exercise from the total duration
* Once the duration for each exercise has elapsed it moves onto the next exercise
* Once all exercises are complete, the date and total calories of the workout are calculated and are inserted into the database using functions “inserWorkouts” and “insertUserWorkouts”
* Finally, it returns the user back to the Menu

**3.3.10 Progress Page and Representing Data on a Graph**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | data = [453, 954, 564, 783, 234, 522, 342]  dateNo = [1, 2, 3, 4, 5, 6, 7]  length = len(data)  **def** progress(data, dateNo, length, CustID):  *# get the dates for workouts*  x = selectWorkoutDate(c, CustID)  **print**(x)  data = []  date = []  date.append(x)  **print**(date)  *# plot a graph of user data*  **print**()  **print**("Loading progress page...")  time.sleep(2)  mymodel = list(map(linearGraph, dateNo))  plt.plot(dateNo, data)  plt.xlabel('date')  plt.xlabel('calories')  plt.title('Week 1')  plt.scatter(dateNo, data)  plt.plot(dateNo, mymodel)  **print**()  *# output mean and standard deviation*  **print**("The mean calories burnt is:", mean)  **print**("The standard deviation is:", sd)  **print**()  back = int(input("Press 1 to go back to menu: "))  **if** back == 1:  Menu(CustID)  **else**:  **print**()  plt.show() |

* From the home screen, if the user selects to access the “Review” page, the subroutine above is then called.
* This subroutine will select the user’s daily calories total from each day of the past week from the database and will plot those values on a graph to show the user the trend.
* A line of best fit will also be plotted to show the user where they will be if they continue their current rate of workouts.
* The program then calculates the mean using the subroutine calcMean and the standard deviation using the subroutine standardDev and will output the results to show the user their average results as well as showing them, on average, how far each score lies from the mean.
  1. **Band A Algorithms**
     1. **OOP Classes and Polymorphism**

**3.4.1.1 UserValid and PassValid Classes Polymorphism**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92 | **class** **UserValid**:  **def** \_\_init\_\_(self, username):  self.\_username = username  *# validate length of username*  **def** len\_valid(self):  check = False  **while** check == False:  **if** len(self.\_username) <= 5 **or** len(self.\_username) >= 61:  **print**("Your username must be between 6 to 60 characters long")  **print**()  self.\_username = input("Username: ")  **else**:  **return** self.\_username    *# make sure username contains a number*  **def** dig\_valid(self):  check = False  **while** check == False:  **if** **not** any(char.isdigit() **for** char **in** self.\_username):  **print**("Your username must contain a number")  **print**()  self.\_username = input("Username:")  **else**:  **return** self.\_username    *# make sure there is an upper case letter*  **def** upper\_valid(self):  check = False  **while** check == False:  **if** **not** any(char.isupper() **for** char **in** self.\_username):  **print**("Your username must contain an uppercase letter")  **print**()  self.\_username = input("Username:")  **else**:  **return** self.\_username  *# make sure there is a lower case letter*  **def** lower\_valid(self):  check = False  **while** check == False:  **if** **not** any(char.islower() **for** char **in** self.\_username):  **print**("Your username must contain a lowercase letter")  **print**()  self.\_username = input("Username:")  **else**:  **return** self.\_username      **class** **PassValid**:  **def** \_\_init\_\_(self, password):  self.\_password = password  *# make sure the password length is valid*  **def** len\_valid(self):  check = True  **while** check == True:  **if** len(self.\_password) <= 5 **or** len(self.\_password) >= 61:  **print**("Your password must be between 6 to 60 characters long")  **print**()  self.\_password = input("Password: ")  **else**:  **return** self.\_password  *# make sure the password includes a digit*  **def** dig\_valid(self):  check = False  **while** check == False:  **if** **not** any(char.isdigit() **for** char **in** self.\_password):  **print**("Your password must contain a number")  **print**()  self.\_password = input("Password:")  **else**:  **return** self.\_password  *# make sure the password has an uppercase letter*  **def** upper\_valid(self):  check = False  **while** check == False:  **if** **not** any(char.isupper() **for** char **in** self.\_password):  **print**("Your password must contain an uppercase letter")  **print**()  self.\_password = input("Password:")  **else**:  **return** self.\_password  *# make sure the password has a lowercase letter*  **def** lower\_valid(self):  check = False  **while** check == False:  **if** **not** any(char.islower() **for** char **in** self.\_password):  **print**("Your password must contain a lowercase letter")  **print**()  self.\_password = input("Password:")  **else**:  **return** self.\_password |

* The above code is used to validate the user’s username and password when they register a new account
* The two classes PassValid and UserValid make use of polymorphism, the objects that are created from the respective classes have the same methods but are each treated differently depending on their class
* Polymorphism is useful in this scenario as it increases reusability and makes the program easy to understand.
  + - 1. **Workouts Class**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | **class** **Workouts**:    **def** \_\_init\_\_(self, duration, reps, exercise, workout, Type, calories):  self.\_\_duration = duration  self.\_\_reps = reps  self.\_\_exercise = exercise  self.\_\_Type = Type  self.\_\_calories = calories  self.\_\_workout = workout  *# calculate the total calories burnt with a given set of reps and workout type*  **def** calc\_calories(self):  **return** self.\_\_reps \* (self.\_\_exercise[self.\_\_Type][self.\_\_calories])  **def** get\_workout(self):  **return** self.\_\_exercise[self.\_\_workout][self.\_\_calories]  *#display the exercise along with the number of reps and total calories burnt*  **def** start(self):  **print**()  **print**((self.\_\_calories + 1),". ", self.get\_workout(), "x",self.\_\_reps,"reps of", round(self.calc\_calories(), 2) ,"calories.") |

* When starting a new workout the above class is called to setup the workout data and list it.
* The class contains properties such as duration, reps, exercise, Type, calories and workout
* The method calc\_calories is used to calculate and return the number of calories burnt per each workout routine by multiplying the number of reps by the calories for 1 rep which is located in the 2D array with the name exercise.
* The method get\_workout is used to name of the specific exercise using the 2D array exercise.
* The method start is used to output to the user the name of the exercise long with the number of reps required and also the calories burn by doing that exercise for a set amount of reps determined by the method calc\_calories.
  + 1. **Workout Queue and Queue Operations**

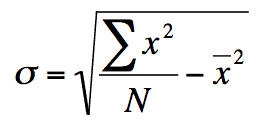
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | **class** **workoutQueue**:  **def** \_\_init\_\_(self, max\_size, size=0, front=0, rear=0):  self.queue = [[] **for** i **in** range(5)]  self.max\_size = max\_size  self.size = size  self.front = front  self.rear = rear    *# adding values to the end of the queue*  **def** enqueue(self, data):  **if** **not** self.isFull():  self.queue[self.rear] = data  self.rear = int((self.rear + 1) % self.max\_size)  self.size += 1  **else**:  **print**('Queue is full')  *# removing values from the front of the queue*  **def** dequeue(self):  **if** **not** self.isEmpty():  x = self.queue[self.front]  self.front = int((self.front + 1) % self.max\_size)  self.size -= 1  **else**:  **print**('Queue is empty')  **return** x  *#outputting the queue*  **def** show(self):  **return** self.queue |

* The above queue is used to add each workout to queue to allow to be dequeued one at a time meaning it happens sequentially.
* This is useful in my program because each workout needs to appear to the user one after another.
* The methods used in this class represent the main operations that are used on a queue: enqueue, dequeue and show.
* The enqueue method is used to add new values to the end of the queue
* The dequeue method is used to remove values from the front of the queue.
  + 1. **Complex A-Level Mathematics**

**3.4.3.1 Standard Deviation Algorithm**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | *#calculating the sum of all values in the list using recursion*  **def** totalX(data):  x = len(data)  **if** x == 1:  **return** data[0]  **else**:  **return** data[0] + totalX(data[1:])  *#calculating the mean of x*  **def** calcMean(total\_x, length, data ):  mean = total\_x / length  **return** mean    *#calculating the sum of the squares of all list values*  **def** totalXsqrd(length, data, y):  Z= 0  **for** i **in** range (0, length):  x = Z + (data[i] \* data[i])  Z = x  **return** Z  *#calculating the standard deviation*  **def** standardDev(Xsqrd, meanX, length):  y = Xsqrd / length  m = meanX \* meanX  variance = y - m  standard\_dev = math.sqrt(variance)  **return** standard\_dev |

* This algorithm makes use of the Standard Deviation formula I learnt in A Level Mathematics under the statistics module
* It is used to calculate how far on average do values vary from the mean which is useful in my program to keep track of users progress



* The formula above is the standard deviation formula, N stands for the total sum of all the data in the set, x with a bar is the mean value and signme x^2 represents the sum of the squares of all x values.
* To use this formula, I have split each section into a different function
* The totalX function uses recursion to calculate the sum of all values in the list
* The totalXsqrd function calculates the sum of the squares of all list values.
* The calcMean function calculates the mean value in the set of data
* The standardDev function combines all the functions mentioned above into a formula to return the standard deviation of the dataset.
  + 1. **Hashing**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | *#calculate a unique hash given a user input*  **def** hashing(value):  passw = []  *# convert string value of password to character code*  **for** x **in** range (len(value)):  passw = ord(value[x])  total = 12 + (5.6 \* math.cos((360\*passw) / 365)) + (1.4 \* math.sin((360 \* passw) /365 ))  **return** int(total) |

Text

Description automatically generated with medium confidence

* The function above is used to calculate a unique hash using the user’s password as an input. A hash is required since the login details of a user is something that needs to be kept secure and only accessible by the user themselves.
* The hash calculated will be stored as the index and primary key for the user login details.
* The algorithm follows a formula that I learnt in A level Mathematics under the Trigonometry module. The formula can be seen below.
* Since the formula makes use of complex A level mathematics this allows the hash to be more secure and complicated.
* The algorithm uses the trigonometric functions cosine and sin to calculate values given an input.
  + 1. **Database**
       1. **Relational Database**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | *#creating database and tables*  **def** createDatabase():  **if** **not** os.path.isfile("fitness\_db.db"):  fitness\_db = sqlite3.connect("fitness\_db.db")  c=fitness\_db.cursor()  *# create customer table, storing customers details*  c.execute('''CREATE TABLE Customer  (CustomerID integer PRIMARY KEY,  customer\_firstname text,  customer\_surname text,  customer\_weight\_kg integer,  customer\_height\_cm integer,  customer\_dob date)  ''')  fitness\_db.commit()  *# create workouts table, storing data on different workouts with a corresponding workoutID*  c.execute('''CREATE TABLE Workouts  (WorkoutID integer PRIMARY KEY,  workout\_routine text,  calories integer,  duration\_secs integer)  ''')  fitness\_db.commit()  *# create login table, storing user’s login details with a loginID calculated by a hashing algorithm with the password as the input*  c.execute('''CREATE TABLE Login  (LoginID integer PRIMARY KEY,  username text,  password text)  ''')  fitness\_db.commit()  *# create a UserWorkouts table that acts as a link table between the user’s customer data and the workout they completed, also including the date in which this was completed.*  c.execute('''CREATE TABLE UserWorkouts  (UserWorkoutID integer PRIMARY KEY,  CustomerID integer,  WorkoutID integer,  workout\_date date)  ''')  fitness\_db.commit()  *# create a UserID table, which stores the CustomerID along with the LoginID and acts as a link table, providing and interface between the two tables and pairing a users login details with their personal details.*  c.execute('''CREATE TABLE UserID  (UserID integer PRIMARY KEY,  CustomerID integer,  LoginID integer,  FOREIGN KEY(CustomerID) REFERENCES Customer(CustomerID),  FOREIGN KEY(LoginID) REFERENCES Customer(CustomerID))  ''')  fitness\_db.commit()  fitness\_db.close() |

**3.4.5.2 SQL queries and Cross-table parameterised SQL**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69 | *#inserting data into the database table for customer*  **def** insertCustomer(c, ID, firstname, surname, weight, height, dob):  c.execute('''INSERT OR REPLACE INTO Customer  VALUES (?, ?, ?, ?, ?, ?)''', (ID, firstname, surname, weight, height, dob))  fitness\_db.commit()  *#inserting data into the database table for workouts list*  **def** insertWorkouts(c, ID, routine, calories, duration):  c.execute('''INSERT OR REPLACE INTO Workouts  VALUES (?, ?, ?, ?)''', (ID, routine, calories, duration))  fitness\_db.commit()  *#inserting data into the database table for logins*  **def** insertLogin(c, ID, username, password):  c.execute('''INSERT OR REPLACE INTO Login  VALUES (?, ?, ?)''', (ID, username, password))  fitness\_db.commit()  *#inserting data into the database table for customer workouts done*  **def** insertUserWorkouts(c, ID, UserID, WorkoutID, workout\_date):  c.execute('''INSERT OR REPLACE INTO UserWorkouts  VALUES (?, ?, ?, ?)''', (ID, UserID, WorkoutID, workout\_date))  fitness\_db.commit()  *#inserting data into the database table for the link table UserID*  **def** insertUser(c, ID, CustomerID, LoginID):  c.execute('''INSERT OR REPLACE INTO UserID  VALUES (?, ?, ?)''', (ID, CustomerID, LoginID))  fitness\_db.commit()  *#selecting user’s username from Login table given LoginID*  **def** selectLogin(c, LoginID):  c.execute('''SELECT username FROM UserID.Login  WHERE LoginID = ? ''', (LoginID,))  val = c.fetchone()  val = val[0]  fitness\_db.commit()  **return** val  *#selecting user’s CustomerID from the UserID link table given a LoginID*  **def** selectCustID(c, LoginID):  c.execute('''SELECT CustomerID FROM UserID  WHERE LoginID = ? ''', (LoginID,))  val = c.fetchone()  val = val[0]  fitness\_db.commit()  **return** val  *#select user’s first name given a CustomerID through relational database and cross-table parametrised SQL functions such as INNER JOIN*  **def** selectNameFromUserID(c, CustID):  c.execute('''SELECT Customer.customer\_firstname FROM UserID INNER JOIN Customer ON UserID.CustomerID = Customer.CustomerID WHERE Customer.CustomerID = ? ''', (CustID,))  fitness\_db.commit()  val = c.fetchone()  val = val[0]  **return** val  *#select user’s first name given a LoginID through relational database and cross-table parametrised SQL functions such as INNER JOIN*  **def** selectUsernameFromUserID(c, LoginID):  c.execute('''SELECT Login.username FROM UserID  INNER JOIN Login ON UserID.LoginID = Login.LoginID WHERE Login.LoginID = ? ''', (LoginID,))  fitness\_db.commit()  val = c.fetchone()  val = val[0]  **return** val  *#select the workoutID from UserWorkouts given a CustomerID*  **def** selectWorkoutID(c, CustID):  c.execute('''SELECT WorkoutID FROM UserWorkouts  WHERE CustomerID = ? ''', (CustID,))  val = c.fetchall()  val = val[0]  fitness\_db.commit()  **return** val |

* The functions selectNameFromUserID and selectUsernameFromUserID are both examples of cross-table parameterised SQL functions
* They both select data through multiple SQL tables by the means of Primary and Foreign keys.
* This is done with the use of an INNER JOIN SQL function.
  + 1. **Recursive Algorithms**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | *# recursively checking to see if user choice is an integer and if it isn't, asks again*  **def** intValid(response):    **try**:  integer\_check = int(input(response))  **except** **ValueError**:  **print**("Please select an integer value option, try agian")  intValid(response)  **else**:  **return** integer\_check  *#recursively checking to see if user choice is within the options and if it not, it asks again*  **def** rangeValid(choice, lowest, highest):  **if** choice < lowest **or** choice > highest:  **print**("please select an option that is available")  choice = int(input("Please select the integer value of what you want to access: "))  rangeValid(choice, lowest, highest)  **else**:  **return** choice |

* Although recursion is used in other parts of the program that I have described in other sections, this is one of the most frequently used and fundamental use of a recursive technique in the program
* This algorithm above consists of 2 functions intValid and rangeValid
* The intValid function takes a parameter response and checks whether the value is an integer or not, if it is not, it tells the user that it is not a valid input and so calls it self again until the user input is valid.
* The rangeValid functions parameters choice, lowest and highest
* The parameter choice stores the user input, th parameter lowest and highest stores the lowest and highest acceptable rangle of values
* The function then checks whether the input fits within that range, if not then it returns and error then calls upon itself
* The base case of the recursion is when the user input fits the range of valid inputs
  + 1. **Merge Sort**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | *## divide the arrays into halves and sort them recursively until we get arrays with 1 element*  **def** mergeSort(array, first\_index, second\_index):  **if** first\_index >= second\_index:  **return**  middle = (first\_index + second\_index)//2  mergeSort(array, first\_index, middle)  mergeSort(array, middle + 1, second\_index)  mergeArray(array, first\_index, second\_index, middle)    *## mergining the sub arrays into a ordered array*  **def** mergeArray(data, first\_index, second\_index, middle):  *#make copy of both halfs of array that are merging*  first\_ver = data[first\_index:middle + 1]  second\_ver = data[middle+1:second\_index+1]  first\_ver\_index = 0  second\_ver\_index = 0  sorted\_index = first\_index  *#go through both copies of ararys until the it runs out of elements*  **while** first\_ver\_index < len(first\_ver) **and** second\_ver\_index < len(second\_ver):  *#if the left copy has the smaller element then put it in the in the sorted array and move forward in the pointer*  **if** first\_ver[first\_ver\_index] <= second\_ver[second\_ver\_index]:  data[sorted\_index] = first\_ver[first\_ver\_index]  first\_ver\_index = first\_ver\_index + 1    **else**:  data[sorted\_index] = second\_ver[second\_ver\_index]  second\_ver\_index = second\_ver\_index + 1  sorted\_index = sorted\_index + 1    *# when you run out of elements in either the first copy or the second copy, go through the remaining elements and add them*  **while** first\_ver\_index < len(first\_ver):  data[sorted\_index] = first\_ver[first\_ver\_index]  first\_ver\_index = first\_ver\_index + 1  sorted\_index = sorted\_index + 1  **while** second\_ver\_index < len(second\_ver):  data[sorted\_index] = second\_ver[second\_ver\_index]  second\_ver\_index = second\_ver\_index + 1  sorted\_index = sorted\_index + 1    **return** data |

* The above algorithm is merge sort, used to sort the list of calories per exercise in order to provide and order of calories from lowest to highest so that the program can show the user which workout burns the most calories and which workout burns the least.
* Recursion is used in the mergeSort function to split up an array to be sorted.
* This algorithm useful because mergeSort has a time complexity of O(nlogn) which is better than a bubble sort.

1. **Testing**

**4.1 Register Screen**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No. | Test Desc. | Data type | Data value | Expected  result | Pass/Fail | Proof | Explanation |
| 1.a | Check username Is valid | Normal | Vishal314 | Pass | Pass | https://imgur.com/a/quVelUM | Once the data value once inputted, the program opened the user info page and asked the user for their first name meaning they inputted a valid username |
|  | Check username is within the range required. | Erroneous | Vis | Fail | Fail | https://imgur.com/a/VpQv0X9 | Once the data value has been inputted the program returns an error and asks the user to input a value within a valid range |
|  | Check username is within the range required. | Boundary | Visha4 | Pass | Pass | https://imgur.com/a/dnjanf8 | Once the data value has been inputted, the program opened the user info page and asked the user for their first name meaning the input is valid. |
| 2.a | Check username contains an uppercase and lowercase letter | Erroneous | carrot123 | Fail | Fail | https://imgur.com/a/69Bwkqz | Once the data value has been inputted the program returns an error and asks the user to input a value that has an uppercase letter |
|  | Check username contains an integer | Erroneous | carrot | Fail | Fail | https://imgur.com/a/D0OGQgh | Once the data value has been inputted the program returns an error and asks the user to input a value that has an integere |

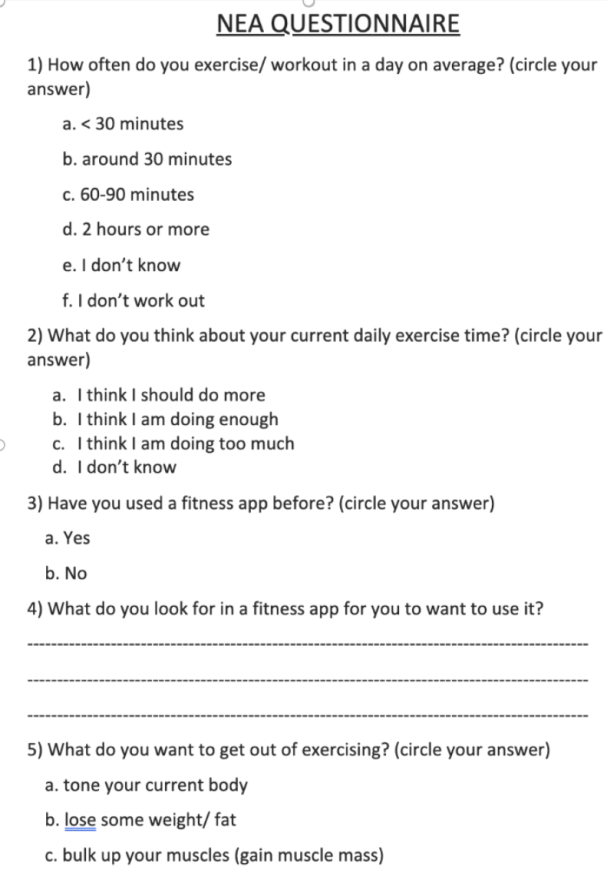
**4.1 Menu Screen (Objective 2)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No. | Test Desc. | Data type | Data value | Expected  result | Pass/Fail | Proof | Explanation |
| 3.a | Check if user’s menu input is in a valid range | Normal | 1 | Pass | Pass | https://www.youtube.com/watch?v=2fFtt9p5K-I | Once the data value once inputted, the program opened the new workout page meaning the value has been accepted |
|  | Check if user’s menu input is in a valid range | Erroneous | 8 | Fail | Fail | https://youtu.be/QaAes9WFea8 | Once the data value has been inputted the program returns an error and asks the user to re-enter their value within the range. |
| 3.b | Check if user’s menu input is an integer or not | Erroneous | ljooooooooooop | Fail | Fail | https://youtu.be/iNUG7cxu23s | Once the data value has been inputted the program returns an error and asks the user to re-enter a valid integer value |

**4.3 Workout Program (Objective 5)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test No. | Test Desc. | Data type | Data value | Expected  result | Pass/Fail | Proof | Explanation |
| 4.a | Check if workouts come one at a time in intervals determined by duration divided per exercise | Normal |  | Pass | Pass | https://youtu.be/Q1oE6vPtFUM | Once selected difficulty and workout type, the program will run the workout, it will show the first workout and ask the user to start, after the time has elapsed, in this case its 120 seconds per exercise since total duration is 600 seconds and there are 5 workouts, it will end the workout and upload details to the database. |

**Appendix**

**Figure 1:**