**NEA:**

**Your Fitness**

By Gopala Bhamidipati

Contents

[**Introduction** 3](#_Toc60911683)

[**Section One – Analysis** 4](#_Toc60911684)

[1.1 - Problem Identification 4](#_Toc60911685)

[1.2 - Computation Methods: 5](#_Toc60911686)

[1.3 - Stakeholders: 6](#_Toc60911687)

[1.4 - Research of Existing Solutions 8](#_Toc60911688)

[1.5 - Essential Features 9](#_Toc60911689)

[1.6 - Limitations: 10](#_Toc60911690)

[1.7 - Hardware and Software Requirements 11](#_Toc60911691)

[1.8 - Success Criteria 12](#_Toc60911692)

[**Section Two - Documented Design** 14](#_Toc60911693)

[2.1 - Problem Decomposition 14](#_Toc60911694)

[2.2 - Structure of Solution 15](#_Toc60911695)

[2.3 – Algorithm Design 17](#_Toc60911696)

[2.4 – Usability Features 23](#_Toc60911697)

[2.5 – Key Variables and Validation 25](#_Toc60911698)

[**Section 3 – Iterative Development** 26](#_Toc60911699)

[Iteration 1 – Login and Registration 26](#_Toc60911700)

[Iteration 1.1 – Register 27](#_Toc60911701)

[Iteration 1.2 – Login 44](#_Toc60911702)

[Iteration 2 – Quiz 49](#_Toc60911703)

[Iteration 2.1 – Set\_User\_Details 50](#_Toc60911704)

[Iteration 2.2 – Store\_User\_Details 66](#_Toc60911705)

[Iteration 2.3 - Putting it all Together 68](#_Toc60911706)

# **Introduction**

This document outlines the analysis and design of the app Your Fitness. Section one is the analysis section that include problem explanation, computational methods, stakeholder, features, limitations etc. Section two documents high level and low-level design. Section three explains iterative development. As part of the development of this app, C# language will be used to develop Windows Forms using Visual Studio. Data, such as login details, will be stored in a SQL Server Express database.

# **Section One – Analysis**

## 1.1 - Problem Identification

I would like to set the scene in April 2020, when we are all quarantined. We are all at home and bored. Some of us may be quite efficient and using this time to be more productive, but I feel most of us may be feeling a little lonely and a little less productive. I would say it is more important now than ever before for us to be keeping healthy and exercising. Before, we would have the necessity of having to move around and get a little bit of exercise. Gyms and leisure centres were open, or you may meet up with friends to play a sport, such as basketball or badminton. Even if we play no sports, just walking to our next class or going outside provides us with a little bit of exercise. However, now we have none of that, and it poses a risk to us and our bodies. It’s dangerous to our wellbeing if we do not provide our bodies with some form of exercise, and keeping that in mind I believe this app will be very helpful to our physical and mental health.

Physical activity can provide immediate benefits. It helps to boost our mood, reduce stress, improve sleep and sharpen our focus. These benefits are extremely important in the testing and trying times that we are living in. However, according to CNBC, a study from “Evidation Health with 160,000 US participants” found that Americans are moving less and sleeping more. “Activity levels were down by 39% on Tuesday, March 24 compared to activity recorded on March 1”. However, time spent sleeping increased by 20%. I think this is a very unhealthy way to live, as this will increase laziness and, more importantly, mental health problems. The same health risks for sleeping too little, including heart diseases and cognitive issues including difficulty with memory, are associated with oversleeping. This has already been proven to be happening as results show that 49% of respondents have increased anxiety from March 13th.

## 1.2 - Computation Methods:

1. There will be many different features to the problem, and some of these will need to work separately. Therefore, decomposition will be at the centre of many parts of the problem. This will require breaking my problem down into smaller individual tasks that will be easier to accomplish. For example, my problem will require a main menu, and there will be many parts to this main menu, such as a progress button, a timetable button, an exercise for the day button and a diet for the day button. Therefore, this will require decomposition in order to program each “button” and everything inside it individually, and then put it all together.
2. The app needs to be able to provide a user interface that can provide an at-a-glance summary of things such as what your timetable will be and progress. Therefore, visualisation will be required, with text being used for the timetable and types of notifications that are received being in different colours so as to differentiate. Also, graphs and tables will be made to keep track of data stored for progress of users.
3. This app will need to be able to use inputs based on a quiz and using this data, patterns will need to be spotted. Therefore, selection and data mining will be required so as to store input and to find ways to reduce bad patterns in the input, such as eating too much or too little lunch or dinner. Using this approach, it will be easier for output to come that will improve lifestyle.
4. The app needs to provide the user with a timetable that can provide both an at glance summary and an in-depth look at what is happening

## 1.3 - Stakeholders:

* I have two stakeholders

1. Dhanush Ammineni
2. Nick Marshall

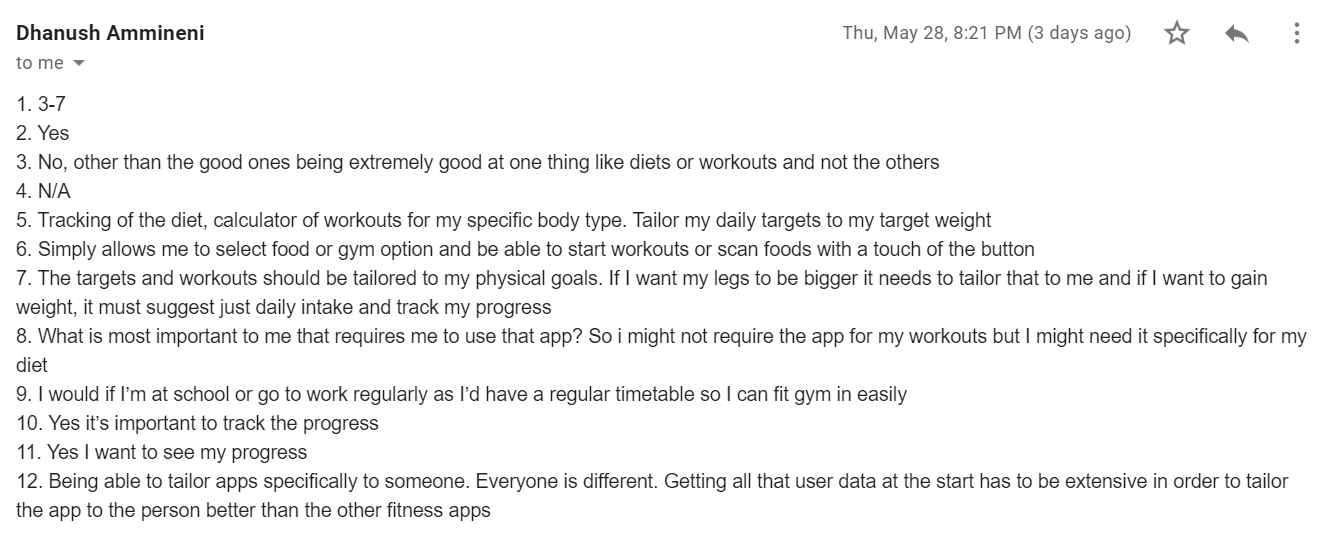
* These are two people in my year who regularly work out. Dhanush works out for himself, but Nick as a runner has a necessity to keep up with his health However, they have two things in common:

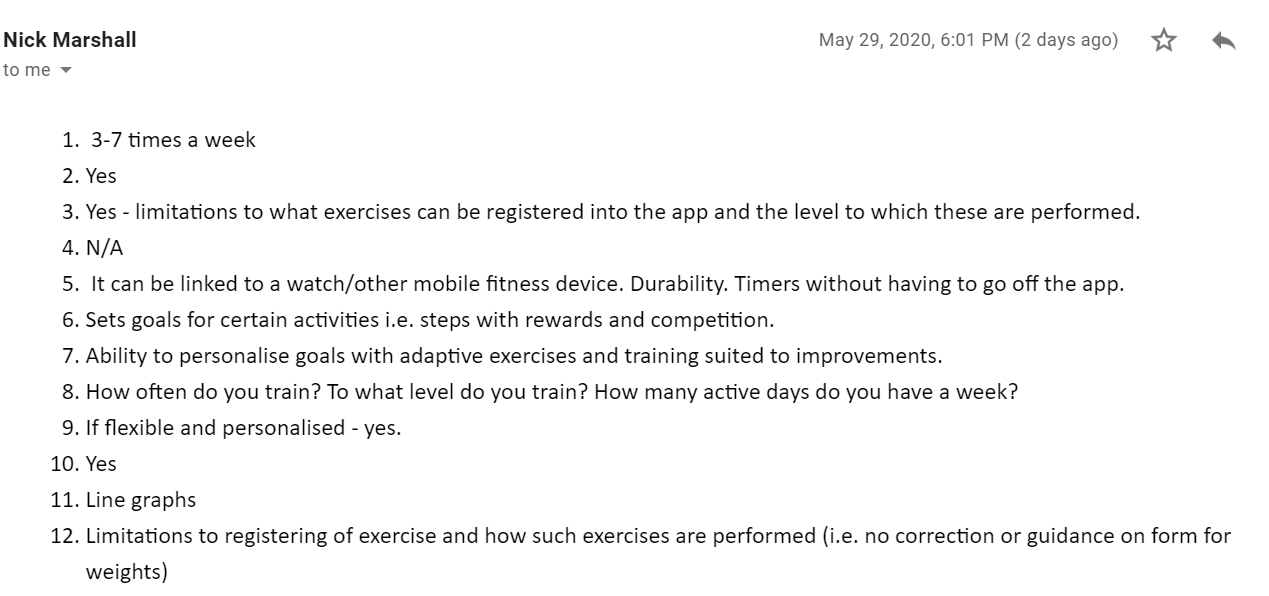
1. They are struggling to work out
2. They use, or have used, fitness apps but are not entirely satisfied with them

* For these reasons, I have decided that they’d be perfect for the project
* I will be contacting them via email throughout this project
* I sent them a questionnaire to begin with:

|  |
| --- |
| 1. How often do you work out?   * 3–7 times a week * 1–2 times a week * More than once a day   2. Do you use, or have you ever used, any fitness apps?   * Yes * No   3. If yes, do you see any limitation in the system you use / used?   * Yes (Explain further) * No     4. If not, what has been stopping you from using fitness apps?  5. If you were to use a fitness app, what would be your major requirements?  6. What would be your requirements of the user interface?  7.  How would you want a fitness app to be personalised to yourself?  8. If you were to be given a quiz to personalise the fitness app, what kind of questions would you like to be asked?  9. Do you think you would follow a timetable of exercise if given to you?  10. Would you manually give your weight in order to see progress?  11. How would you like your progress to be presented?  12. What limitations are there for fitness apps in your opinion? |

Responses I received were as such:





These answers told me that, as they have both used fitness apps, the apps need to be well rounded. It needs to be balanced between both fitness and diet. However, the emphasis needs to be on fitness. It will also help to be able to see your progress. There can be limitations to what exercises can be registered, and the quiz questions need to be aimed at the training done by users

## 1.4 - Research of Existing Solutions

* My system is trying to combine fitness apps and nutrition apps with a personalised timetable
* A fitness app is an application that can be downloaded on any mobile device and used anywhere to get fit
* A nutrition app is an app that helps you eat healthy, however I will also like the nutritional side of the app to also be able to be

System 1: Nike+ Run Club:

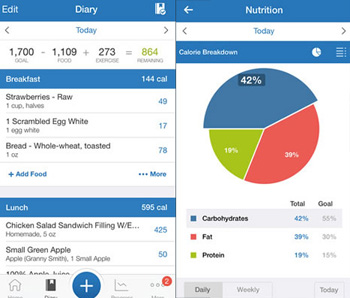


The purpose of the application is to make people run more. Nike Run Club has the tools you need to run better, including GPS run tracking; Audio-Guided Runs; weekly, monthly and custom distance Challenges; customised coaching plans for your goals.

|  |
| --- |
| Features to Adapt  GPS Run Tracking:  The app tracks how far you have run, and how long for  However, I want to expand it to be able to work for all types of exercise, so if you’re working out it should play music and be able to show overall progress, for example how much your BMI or body weight has changed or how much the number of weights you lift has increased. |

System 2: FitnessPal

The purpose of the application is to help you keep track of your daily food and beverage intake, calculating all nutrients, calories and vitamins that you consume. It is a mobile app that connects you to a large database of food in order to look up and record what you eat. It gives you insightful feedback than can be used to help reach your weight goal.



|  |
| --- |
| Features to Adapt:   * 1. Figure out what your diet is missing or what you need to eat more of. * I want it to be able to connect to the amount of exercise you do and what your goal is for your body by the end of the course. * For example, if you do really intense exercise and want to cut down fat, it may make you eat more than someone who also wants to cut down but doesn't work out as much. Someone who goes to gym five times a week will be told to eat a little more than someone who goes gym four times a week.   1. Tracking how much calories in food * I think that it is essential to scan food so as to be able to calculate how many calories you are eating, and it can then be increased or decreased. * I would additionally like to allow you to record this information on the app, automatically by pressing a button about whether you want to record it or not. This is because it would be very simple to forget calorie information, especially with food you like. Therefore, by keeping it recorded, it would help you remember whether or not you should eat it. * I believe that it is useful to be able to use a scanner as this will help you to see the calories of food at home much quicker, and I think it will positively impact users to decide not to eat unhealthy food |

## 1.5 - Essential Features

* My system will require a new account feature. This should involve users clicking on a “create new account” button and creating a new username and password to log in. If a username is already in place, it should start again with the sign that “This username has already been taken, please choose another” until they choose a new username. This feature is required as there will be many people using the app who need to be able to see their own data and timetables, and so choosing their own username and password will allow them to see their data whatever device they use.
* My system will also require a “quizzing” feature. Once a user has registered, it should involve them being asked various questions so as to find out what kind of person they are and how busy they would be every day. For example, one of the questions will be “Do you follow a vegetarian diet?” If they say “Yes”, it should store this information, otherwise it should move to the next question. More examples of the type of questions would be “What is your current weight and height?”, the follow up question should be “What weight would you like to be at?” as this will tell the app whether they are losing fat or gaining muscle, and will also put them on a calorie deficit diet for weight loss or calorie surplus diet for weight gain, and so will also be base exercise on their goals. Using the information based on this quiz, a timetable should be set up so as to be able to measure up to the requirements of the user.
* My system will require a “Music-Guided Workout” feature. This should involve the quiz asking about their favourite music, and it being able to make a playlist connected to the music service they use using the information they have. This feature is needed because there is only so much time that people can spend working out, and this will speed it up. While I could just make an automated playlist that plays any workout songs, it is best to stick to what the user likes, as skipping music could waste time
* My system will require a feature that will be able to find out what current diet is and what it should be changed to. This will involve the quiz, and keeping track of what is being eaten. This will be done by users scanning food so as to see the number of calories eaten per day, and then gradually decreasing it week by week until the target number of calories per week is reached, and then staying there. I feel this is important as eating right will make it easier to reach the goal you want.
* My system will require a feature that will show progress over time. This will be done by, using the timetable and diet calculator, to be able to take all data and represent it. For example, if the timetable changes and becomes harder, or the number of calories change, it will be stored to be represented. I feel this is important as people will be more likely to be motivated if they can visually see their progress.

## 1.6 - Limitations:

* 1. Motivation of users. My system assumes that users already have the motivation within them to want to become healthier and that the motivation will be sustained. However, people can become bored if they do the same thing over and over again and therefore the quality of exercise may decrease. This is why it is essential that the timetable changes on a weekly basis, becoming harder and harder and increasing the number of exercises done.
  2. Drained battery. It takes a lot of energy to track the activity for users, which translates to more data used and therefore more battery will be taken up. So, in order to reduce this limitation, it is essential to focus on narrowing down information in order for the app to be more effective, therefore less data will be used and so battery life will be less drained
  3. Counting calories may remove a level of mindfulness because you are trying to hit a target number instead of listening to your body. This will mean that if you are hungrier one day but you ate your target number of calories that day, then you will not be able to eat anymore. Or you may have calories left to eat, but if you aren’t hungry you will still have to eat. Therefore, what could be done is taking calories out of one day and putting it in another, or taking calories out of one day to put it in another. This could be done too many times, therefore there is a limit wherein you can only move calories around twice a week.
  4. I believe another one of the main limitations I have is the lack of time. It’s possible there will not be enough time to implement some additional feature such as:
* My app will not be able to link itself to other devices such as watches. The main reason for this limitation is that I do not have the time to be able to create a mobile app and then recreate the app for devices such as Fitbit or Apple Watch, as this will require a completely new interface in order for it to be able to look good on these devices
* My app will not be able to give rewards for goals. While I personally do not think it is required to give rewards, I also do not think I have the time to do this on top of all the other features I want to add
* While I would like to do these, I need to prioritise and for that reason I would not have the time to implement these

## 1.7 - Hardware and Software Requirements

* + Hardware:

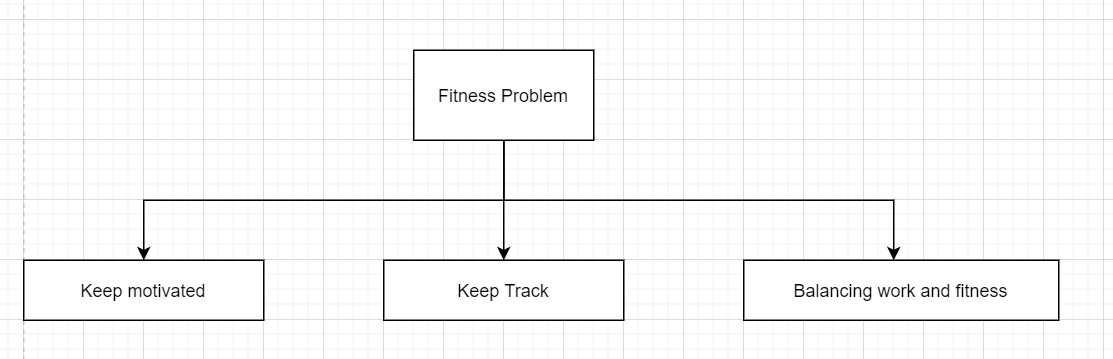
1. A two button USB mouse. USB ports will be available in the laptop, wherein the left button will trigger the events of on-screen objects and the right button accesses the help feature for each object. While the touchpad would suffice, i prefer to use the mouse as it is easier to progress while using it
2. 8 GB RAM. RAM provides computer memory, which is important in order to run several different programs at the same time and when you have enough memory, the programs will run smoothly.
3. Processor clock speed of 3 GHz. A processor provides the instructions that multiple applications and processes need to perform their jobs. Therefore, the faster the processor the faster a computer operates, and would translate to increased efficiency.
4. 3 GB free hard disk space
5. Display of 1080 x 768. If we use the example of Visual Studio, it has a lot of panels which all need to fit somewhere so if you have a small screen, it will all be very cramped. Therefore, having a bigger display is better to fit everything on the screen comfortably.
   * Software:
6. Windows 10 Operating System
7. .NET framework (minimum 4.5)
8. C# using Visual Studio 2019 Community Edition
9. SQL Server Express 2019

## 1.8 - Success Criteria

|  |  |
| --- | --- |
| **Criteria #1** | **A secure register system** |
|  | * 1. - Quick to fill out |
|  | * 1. - Requirement of lowercase and uppercase letters, and numbers |
|  | 1.3 - If the username or password are already in use, the user needs to type in a new username and password until they have a username and password nobody else uses |
| **Criteria #2** | **A fast login system** |
|  | 2.1 - Highly secure |
|  | 2.2 - Quick to get through |
|  | 2.3 - If the user cannot remember their password, they should be able to |
| **Criteria #3** | **Quiz that will help the app to be able to set exercise and diet for each person** |
|  | 3.1 - It should be able to get enough information to make a detailed timetable. |
|  | 3.2 - Personalise the app for separate people |
|  | 3.3 - Set diets in terms of the person using. |
|  | 3.4 - Set exercise in terms of the person, how often they exercise and what their goal is |
| **Criteria #4** | **Progress Checker** |
|  | 4.1 - Show progress in terms of a line graph |
|  | * 1. - Send notifications to the user if they are closer to their goal |
|  | 4.3 - Send notifications if they are dangerously moving away from their goal |
| **Criteria #5** | **Music-Guided Workout:** |
|  | 5.1 - The users’ favourite genres should be used to make a playlist |
|  | 5.2 - The playlist should be played as soon as the user chooses and from there not be able to be paused until they finish |
|  | 5.3 - Music should be able to stop when the user finishes exercising with the use of a timer that the user can set themselves |
| **Criteria #6** | **Timetable** |
|  | 6.1 - The timetable should be able to accessed at any time |
|  | 6.2 - Show the types of exercise to do and at what time |
|  | 6.3 - Show what to eat and at what time |
|  | 6.4 - It should be weekly and stay the same |
| **Criteria #7** | **Exercises:** |
|  | 7.1 - A wide range of exercise that will be chosen by the app |
|  | 7.2 - Exercise intensity should increase on a weekly scale |
|  | 7.3 - Exercise should be in line with the goal of the user |
|  | 7.4 - The same exercise should not be used more than twice in the same week |
| **Criteria #8** | **Diet** |
|  | 8.1 - Diet should be in line with the goal of the user |
|  | 8.2 - Diet should take in factors such as religion and personal choices |
|  | 8.3 - Diet should consist of rest days twice a week |
| **Criteria #9** | **Food Scanning** |
|  | 9.1 - Adapt a barcode scanner in order to scan a barcode and see the nutritional requirements |
|  | 9.2 - Be able to look up food manually by searching on the app and then see nutritional requirements |
|  | 9.3 - It should be able to show food like the one you looked for |

# **Section Two - Documented Design**

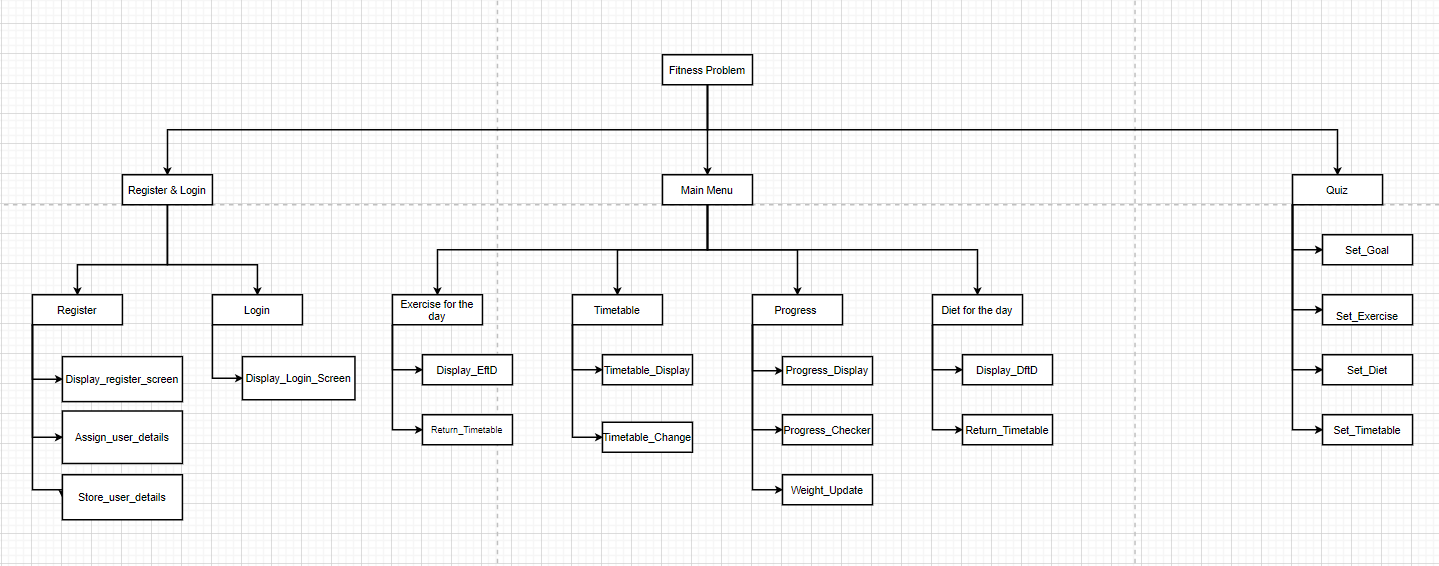
## 2.1 - Problem Decomposition



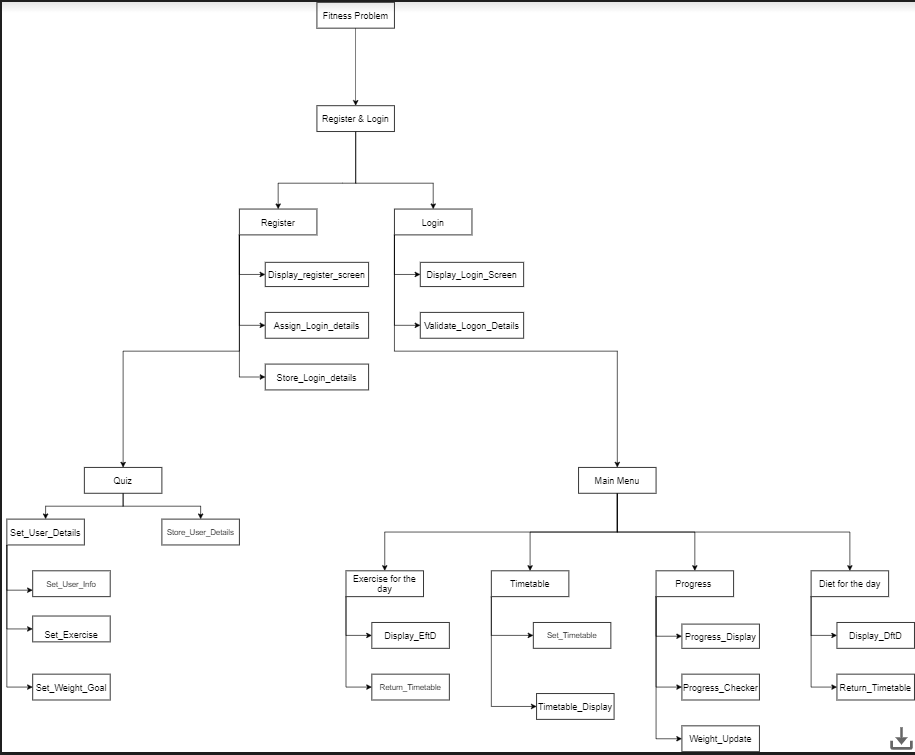
1. Keep motivated - People lose their motivation too quickly when they think they aren't reaching their goals. They get frustrated and decide to give up, but need to realize that they are reaching their goal but it will take some time. One of the biggest problems with fitness apps such as this is that users do not stay satisfied. The reason why people set goals and try to stay healthy is so that, in the end, they are happy and satisfied. However, users are never happy as once they reach their goal, they don't know what to do. Users need a way to see how far they have come and all the work they put in, and they need to be satisfied with their effort. In order for this to happen, they need to stay motivated
2. Keep track - Users need to be able to keep track of the exercise they do and how much time they give towards their health to have the energy to continue to put in the effort that they put in
3. Balancing work and fitness - Users have other priorities other than their fitness, and it is important that fitness works around people’s own schedule

* There could have been many other ways the problem could've been decomposed. For example, other important sub-problems could've been that tracking calories is difficult, or that it is hard to show progress. These are all very valid. However, I have chosen these sub problems because I believe solving these problems will give way to the highest satisfaction of users. The most important thing to look out for in a fitness application is the satisfaction of the user. If users are satisfied with the application, they will continue to use it, and the number of people who use it will get higher and higher.

## 2.2 - Structure of Solution



My original structure was the one above. We had the fitness problem, and all the separate iterations below it, which included the register & login, main menu and the quiz. However, I had many problems with it. For example, there was no way of storing data, or validating details. It is also far more separate and linear than how it would really be. Therefore, I decided to change it to the one below, which better represent the structure:

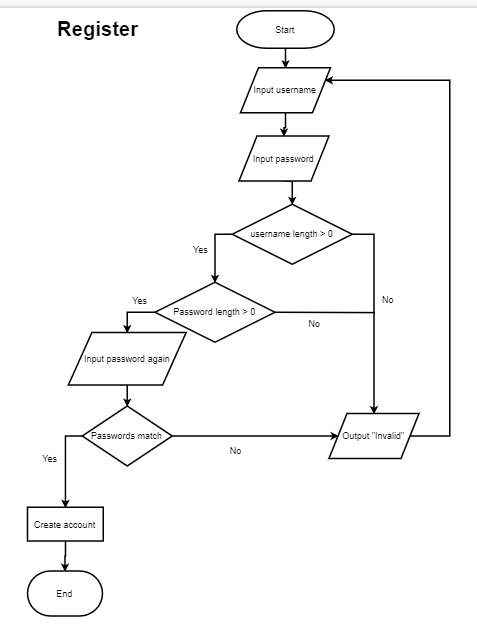


The first iteration, and the first are we go to, is the register and login. Here, we register ourselves and store the data. From the registration, we go to the next iteration, which is the quiz. In this quiz, there are two smaller iterations, set\_user\_details and store\_user\_details. From the login, we go to the final iteration, which is the Main menu. In the main menu, there are 4 smaller iterations, called Exercise for the day, timetable, progress and diet for the day. I decided to do the structure like this as it models how the app will look, starting from the register and login area and from there going to the quiz and main menu section

## 2.3 – Algorithm Design

**Register and Login:**

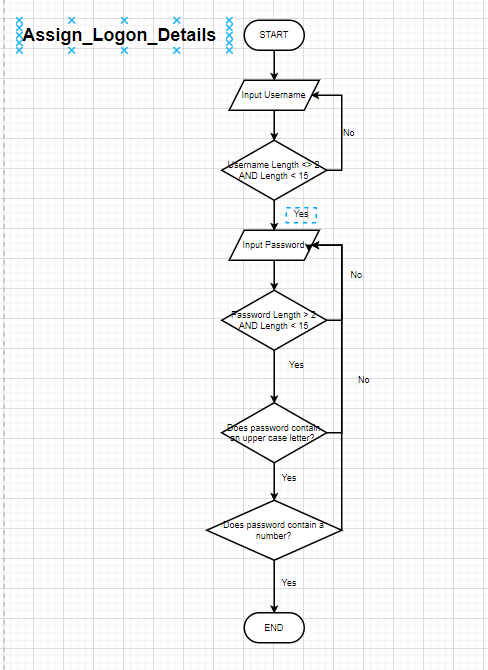
1. Assign\_Login\_Details



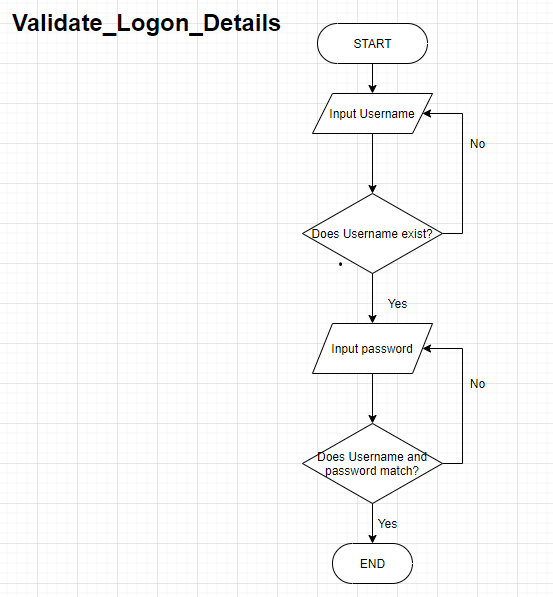
I decided to create a separate login subroutine since this aid’s modularity and maintainability. If a new login system were to be developed, or the existing system were to be updated, only this subroutine would need to be rewritten. Had I incorporated this into a larger “admin” subroutine, subsequent maintenance would have been made more difficult, as any code needed to be first would need to be located within a larger body of code. In this original algorithm, the user would put their details, userName and password, into the algorithm, and if password length is greater than 0, it would be allowed else sent back.

However, I discovered that this algorithm was ineffective. While it checks for passwords, there was nowhere in there to check for usernames. Also, the checking of passwords was not very effective. While it checks for length of password, there was nowhere in the algorithm to check for special characters or numbers.

This led me to create the second version of this algorithm:



1. Validate\_Logon\_Details



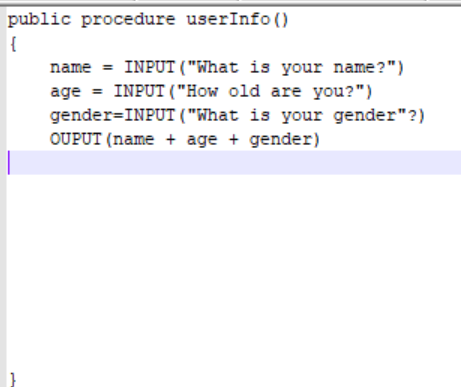
* In this Validate\_Logon\_Details algorithm, we make sure the username exists first. If it does exist, we input the password. If it doesn’t, we go back and input the username
* If the username does exist, we check that the password input matches with the corresponding username, if they do not we go back to input password otherwise we end

I separated this from my Assign\_Login\_Details in order to keep the modularity, and it is also advantageous as it is much simpler to code

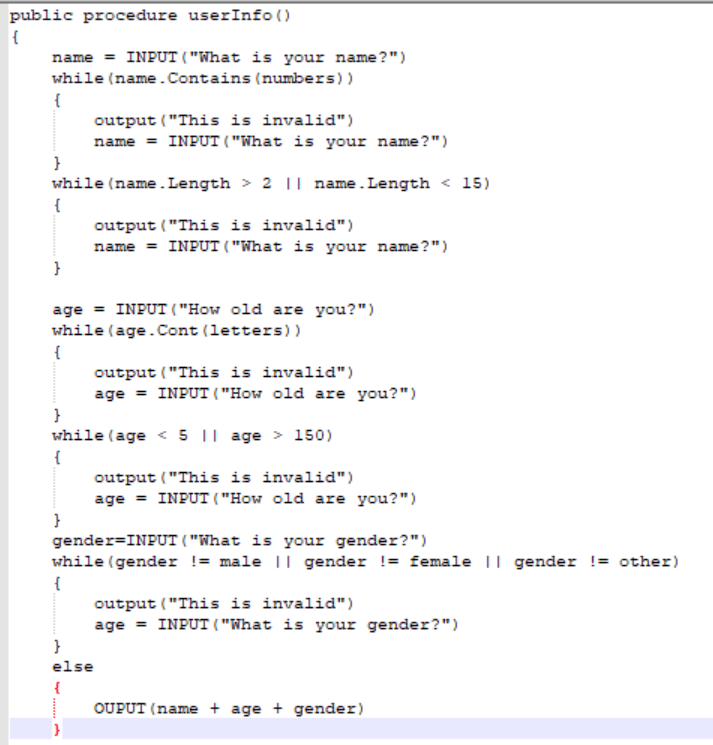
**Quiz**

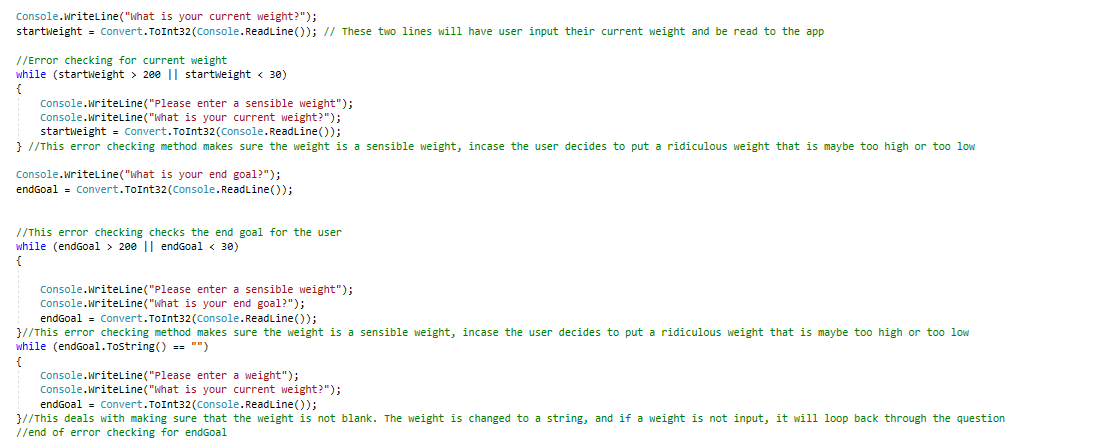
As these are question and answers, I decided to do these on NotePad++;

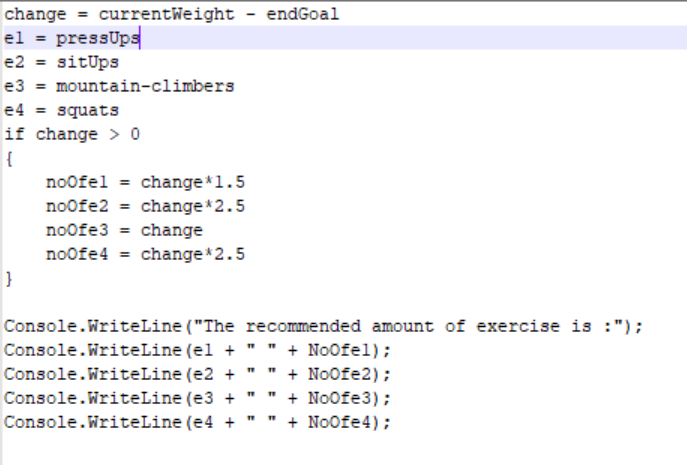
Set\_User\_Info:



* These were the three basic details that will be taken into consideration, and then stored. However, there was no way to error check, therefore I redid it, adding error checking methods, such as needing to ensure that name only contains letters:

- -

* Set\_Weight\_Goal
* Set\_Exercise



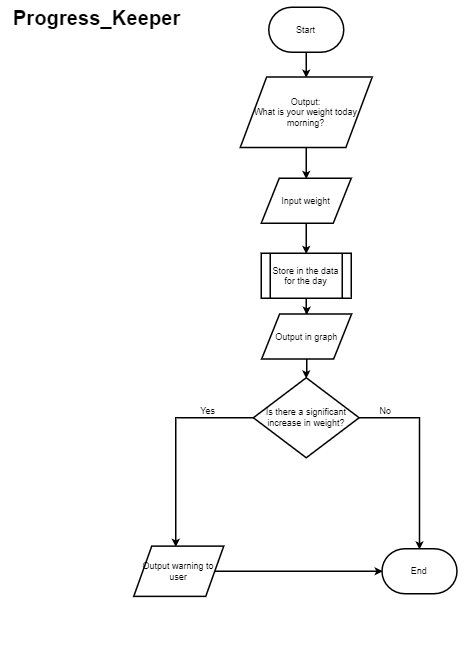
**Main Menu**

1. Set\_Timetable



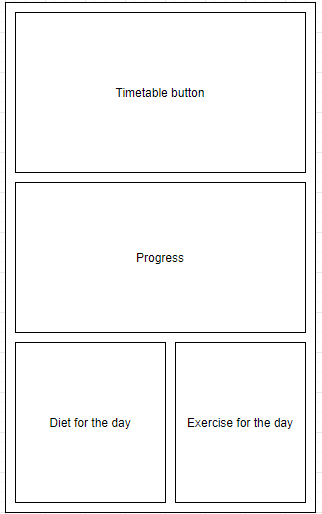
I decided to create a separate subroutine to set the timetable to make it simpler and to implement modular programming. While it will be made via the main menu and the inputs of the user, putting all the subroutines together into a big “main menu” subroutine would make it confusing and difficult to change or upgrade.

1. Progress\_Checker / Weight\_Checker

I decided to create a separate subroutine for the progress checker as it is separate from most of the other features of the app. This is because, while most other features will be decided by the quiz, this will not. It will need manual input from the user to be able to be used.  


## 2.4 – Usability Features

User Interface Design:



|  |
| --- |
| This is the main design of the program. This shows the user all the features that they can use in front of them   1. Timetable button:   This button provides access to the full timetable, including the current day and all other days of the week. This information will be taken from the quiz, and will stay mostly constant apart from a change in intensity every few weeks, whenever the user is ready   * Links to success criteria:   + 3.2 - Personalise the app for separate people   + 6.1 - The timetable should be able to accessed at any time   + 7.2 - Exercise intensity should increase on a weekly scale  1. Progress   This will show the progress of the user from when they started and where they are now. It will show it in terms of a graph. It will also show it in depth, with statistics and converted to BMI, telling you if you’re healthy, and how far you are away from your goal. However, if you are dangerously moving away from your goal, or if you’ve been getting closer, you will receive a message telling you about this.   * Links to success criteria:   + 4.1 - Show progress in terms of a line graph   + 4.2 - Send notifications to the user if they are closer to their goal   + 4.3 - Send notifications if they are dangerously moving away from their goal  1. Diet for the day   This button provides a way to break the timetable into only the diet. If you press this button, you can see your diet in more detail. You can also see nutritional values of food you want to eat and of the food in your diet. You can compare food you have to eat to similar foods of different brands, or similar types of food. For example, you can compare brown rice to white rice or you can see why carbohydrates may not be best for you   * Links to success criteria:   + 6.3 - Show what to eat and at what time   + 9.1 - Adapt a barcode scanner in order to scan a barcode and see nutritional requirements   + 9.2 - Be able to look up food manually by searching on the app   + 9.3 - Be able to show food like the one you are searching for  1. Exercise for the day   This button provides a way to break down your timetable into only your exercise. If you press this button, you can access your exercise into more detail, seeing what part of the body it affects. You can also change the intensity of the exercise, so if you think it’s too much, then you can reduce the intensity. Otherwise, after a certain period of time, using your progress, the intensity will increase or reduce automatically   * Links to success criteria:   + 6.2 - Show the types of exercise to do and at what time   + 7.1 - A wide range of exercises that will be chosen by the app   + 7.2 - Exercise intensity should increase on a weekly scale   In order to maintain a good visual hierarchy, the elements on the user interface will be positioned in a way that highlights the most important actions to the user. On the main menu, “timetable” will be on the top followed by all the other options in order of maintenance. In terms of colour, I would put high contrast colours so that it’s easier for colourblind people to read. I would choose the colours dark blue for the background and white for the font. |

- Waiting on user feedback

## 2.5 – Key Variables and Validation

|  |  |  |
| --- | --- | --- |
| Name | Data Type | How it is used (Explain and Justify) |
| firstName | string | It will be a way to address the user |
| lastName | string |  |
| userName | string | A way to login into the app |
| password | string | A secure string that can be hidden and, if it matches with the username, can login |
| confirmPassword | string | Used in the registration section as the password field masks the user's input. If users mistype their password, they won't recognize it. The confirm password catches typos by prompting users to type their password twice. |
|  |  |  |
| goToRegister | button | Allows us to go to the register page |
| FirstNameTextBox | textBox | We can input first name here |
| LastNameTextBox | textBox | We can input last name here |
| userNameRTextBox | textBox | We can input the user name here |
| passwordRTextBox | textBox | We can input the password here |
| confirmPasswordTextBox | textBox | Input the password in the previous textbox in order to make sure they match |
|  |  |  |
| Dbo.users | SQL database | A database that holds a users login credentials, and stores it for later and allows us to login to the app |
| conn | SQLconnectionObject | establish a connection with the SQL server data source and allow us to access the database: |

(I will update as I go along)

# **Section 3 – Iterative Development**

## Iteration 1 – Login and Registration

* In this iteration, I have made a login and registration form. It will allow a user to input data about themselves, which will be stored in a database and allow a user to login to this app. I have divided this big iteration into 2 sub iterations:

1. Register Iteration

* In the register sub iteration, we focus on the parts that occur in the registration screen, such as making the register screen, assigning the login values, which include first name, last name, user name and password, which we confirm, and then storing this data in a database

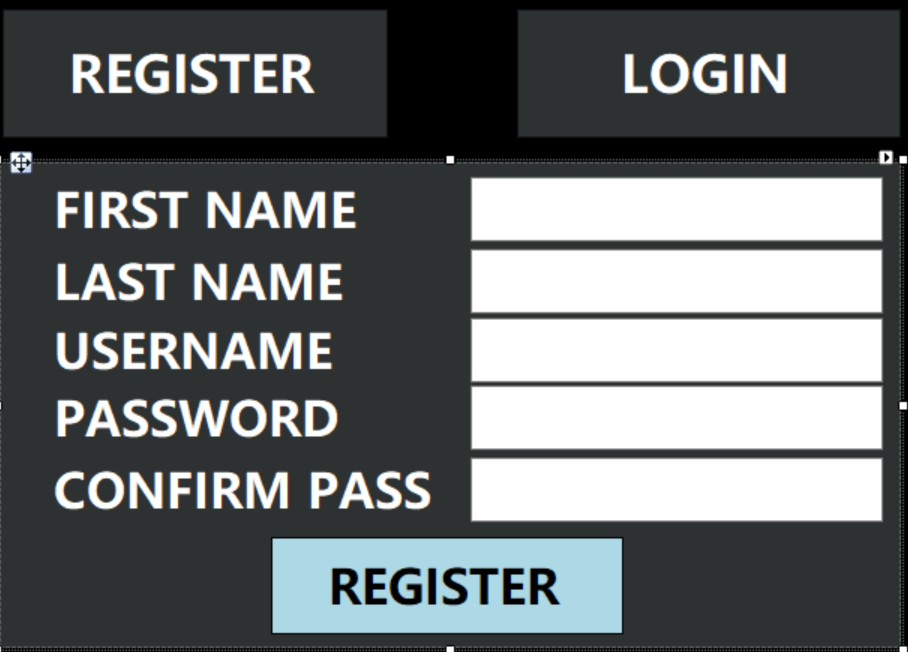
1. Login Iteration

* In the login iteration, we are creating the login screen, and validating the user.
* To validate the user, we do two things:
  + Ensure the user exists
  + Ensure that the username and password match one another

### Iteration 1.1 – Register

#### Iteration 1.1.1 – Display\_Register\_Screen:

To create a register screen, I decided to use Windows Forms, and create both a register and login screen together



For the register screen, the first thing I needed to do is create a GoToRegisterButton:



I then needed to be able to take all the details of the user, such as firstName, lastName, userName and password, therefore there would need to be the corresponding labels and textboxes to be able to do that:



* I would then need a final Register button to be able to submit all these details:



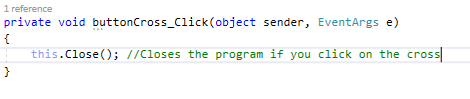
* In order to keep the password characters, and the confirm password characters hidden, I changed it to output \*:



* In order to be able to close the program, I put a cross button:



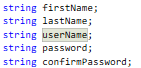
* And made it so that it closes the program:



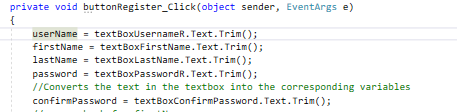
#### Iteration 1.1.2 – Assign\_Logon\_Details

##### Cycle 1:

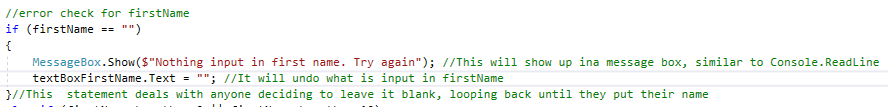
* While I had all the buttons and panels, I needed a way to assign a user with their first name, last name, username and password
* I therefore created the string variables firstName, lastName, userName and password:

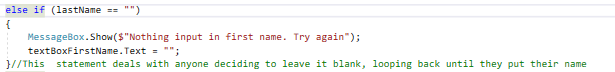


* Going into the register button, I made these variables correspond to the value that would be input by the user:



* I then error checked both the first name and last name, making sure that they aren't blank:

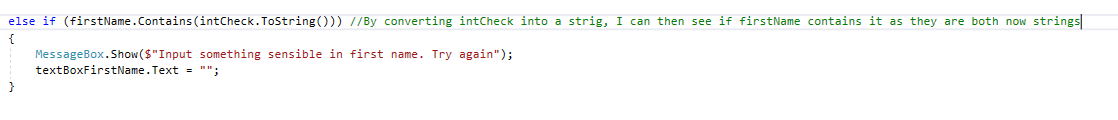
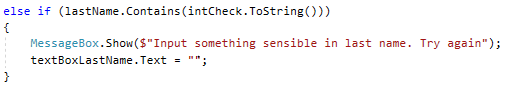




* And they aren’t too long or too short
* In order to check that they don’t put a number in their first or last name, I did the error checking inside a for loop called intCheck:

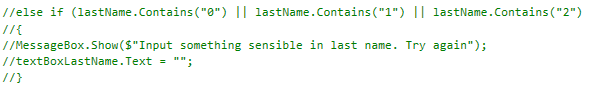


I then used converted the int check into a string, and then checked if firstName or lastName contain it:

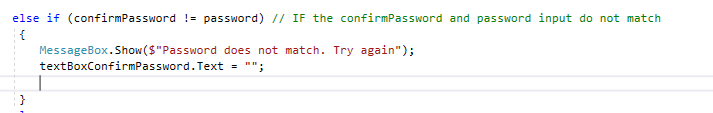


* Originally, what I did was I took each little number from 0 to 9 as a string, and checked if firstName and lastName contains any of them:

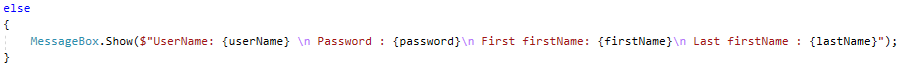




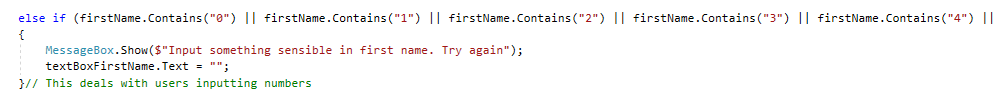
* However I found this method was extremely tedious and not very effective, therefore I changed it
* I then had to make sure the password and confirmPassword matched, otherwise it will not go through. Therefore I made an if statement and I then wrote a message, incase password and confirmPassword does not match, that they do not match and the user has to try again:

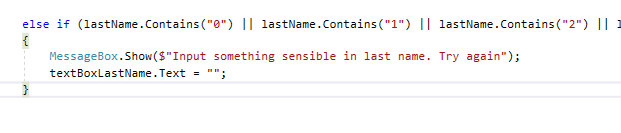


* If everything is correct then, and only then, it will move on and output all the details:



* In this cycle, I found that , while it work, it would actually loop through the messagebox 9 times, therefore I cannot use a for loop and have to just go through every number as a string
* I removed the for loop and put the original statement back in:





###### Cycle 1 Test Data

firstName and surName

|  |  |  |
| --- | --- | --- |
|  | firstName | lastName |
| Normal 1 | Gopala: | Watkins: |
| Normal 2 | Prabhav: | Nightingale: |
| Boundary 1 | Jo: | Leonardsen: |
| Boundary 2 | Alastair: | No: |
| Erroneous 1 | G1pala: | Bhamidipat1: |
| Erroneous 2 | G$pala: | Bhamidipati£: |
| Blank |  |  |

userName and password:

|  |  |  |
| --- | --- | --- |
|  | Username | Password |
| Normal 1 | myUsername: | myPassword: |
| Normal 2 | myUser123: | myPass123: |
| Blank 1 |  |  |
| Blank 2 |  |  |
| Boundary 1 | J | J |

* Password and confirmPassword test data:

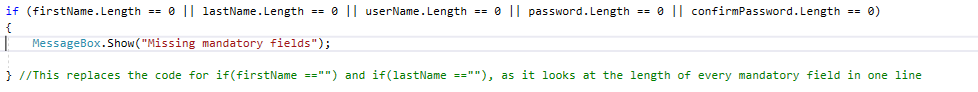
|  |  |
| --- | --- |
| Password and confirmPassword: | Output: |
|  |  |
|  |  |

* There are many issues with this cycle:
  + It read special characters, such as $ or £ as letters and accepted them for firstName and lastName, when it shouldn’t
  + It accepts blank usernames
  + If password and Confirm Password are both blank it accepts that
  + There’s no length limit or minimum length for both first name and last name
  + It allows usernames and passwords to be one letter long
  + The if statement to not accept numbers I have is very inefficient

However, making sure the password and confirm password match works, therefore there is no reason to test it anymore

##### Cycle 2:

* The first thing I decided to do was make sure that nothing could be left blank, where it would show “Missing mandatory fields”:

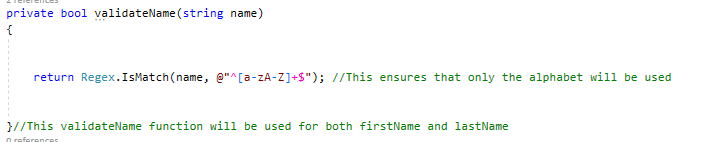


* Doing this in one line ensured it was more efficient as well

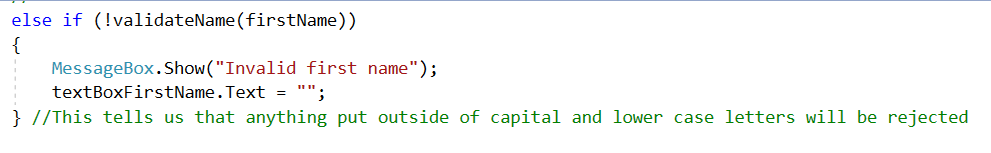
The next thing was to make sure it doesn’t accept special characters for first name or last name, therefore I used Regex classes. To be able to use this, I needed the regular expressions directive:

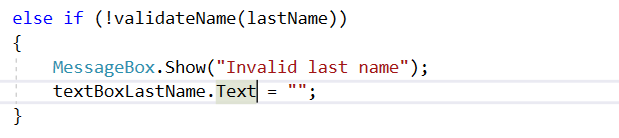
#

To ensure that only letters will be used for first name and last name, I added a validateName function that passes the string name, and only accepts lower letters and capital letters



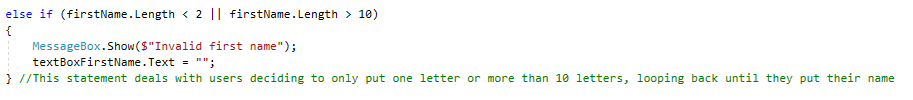
I passed the first name and the last name through this function:

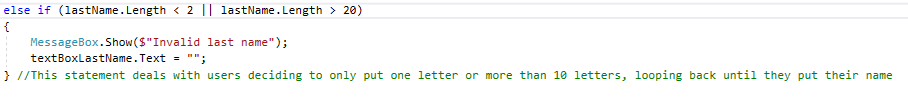


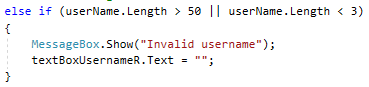


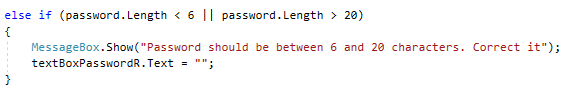
* This now ensured that there could be no special characters or numbers in first name or last name

I then added if statements to ensure firstName, lastName, userName and password were at suitable lengths. I couldn’t jabe done this in one line as all the textboxes are different therefore they all require separate messages:









##### Cycle 2 Test Data

firstName and lastName

|  |  |  |
| --- | --- | --- |
|  | firstName | lastName |
| Normal 1 | Gopala: | Watkins: |
| Normal 2 | Prabhav: | Nightingale: |
| Boundary 1 | J: | i: |
| Boundary 2 | fdaaswerfefwes | fdaaswerfefwes: |
| Erroneous 1 | G1pala: | Bhamidipat1: |
| Erroneous 2 | G$pala: | Bhamidipati£: |
| Blank |  |  |

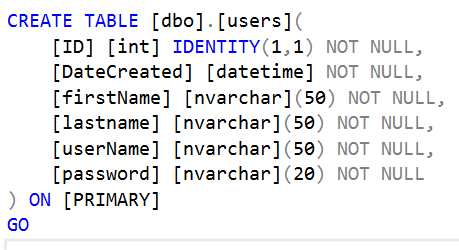
userName and password:

|  |  |  |
| --- | --- | --- |
|  | Username | Password |
| Normal 1 | myUsername: | myPassword: |
| Normal 2 | myUser123: | myPass123: |
| Blank 1 |  |  |
| Blank 2 |  |  |
| Boundary | J | J |

* I am now happy with how Assign\_User\_Details works

#### Iteration 1.1.3 – Store\_Logon\_Details

* Now that I could assignuser details, I needed a way to store the details in a database
* The very first thing I did was create a new database named NEAAppdb. In this database, I created a table called users that holds user’s login credentials. This table has columns firstName, lastName, userName and password, the definitions given below



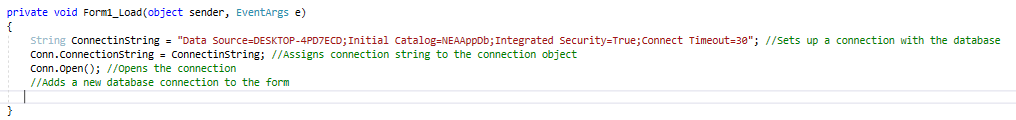
* Then in my login and registration forms, I added the namespace System.Data.Sqlclient:



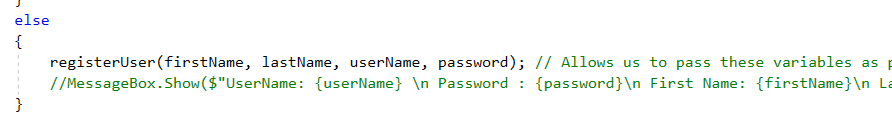
* I then created the SQL connection object called “Conn” which will establish a connection with the SQL server data source and allow us to access the database:

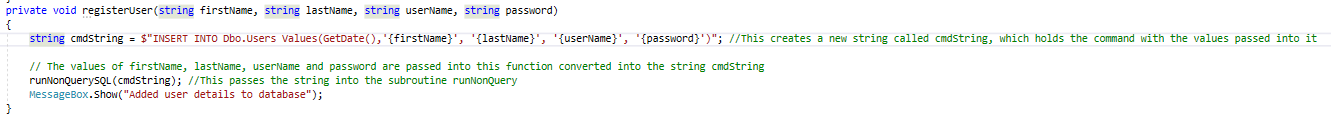


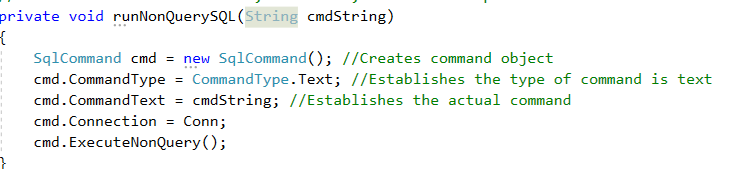
* Then, in the private void for the form, I established a connection between the data source and the form itself:



* I needed a way to ensure that the data given in the register form would be used as the variables firstName, lastName, userName and password in the NEAAppdb.
* I therefore created a subroutine called register that takes in the variables firstName, lastName, userName and password as parameters, and uses the values put in the original register form:

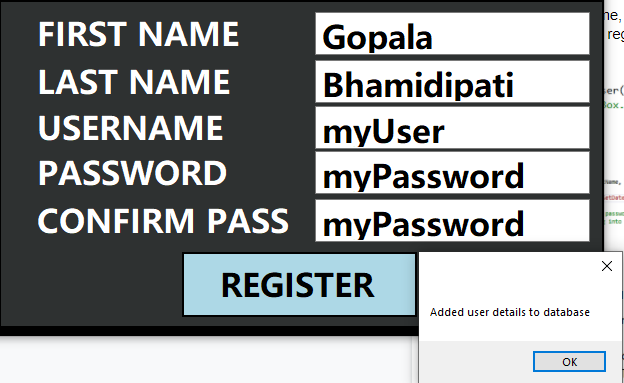


* The string “cmdString” creates a SQL command that can be used in the subroutine runNonQuerySQL:



* In this, we create a command object, assign the type of command, which is text, establish cmdString as the command and assign connection to the command object. We can then execute the command
* ExecuteNonQuery means that we can execute commands if the text is created, altered, dropped, inserted, updated or deleted. This ensures that we add all these details to the database

Demo run:



Using the SQL line select \* from dbo.users I output:



### Iteration 1.2 – Login

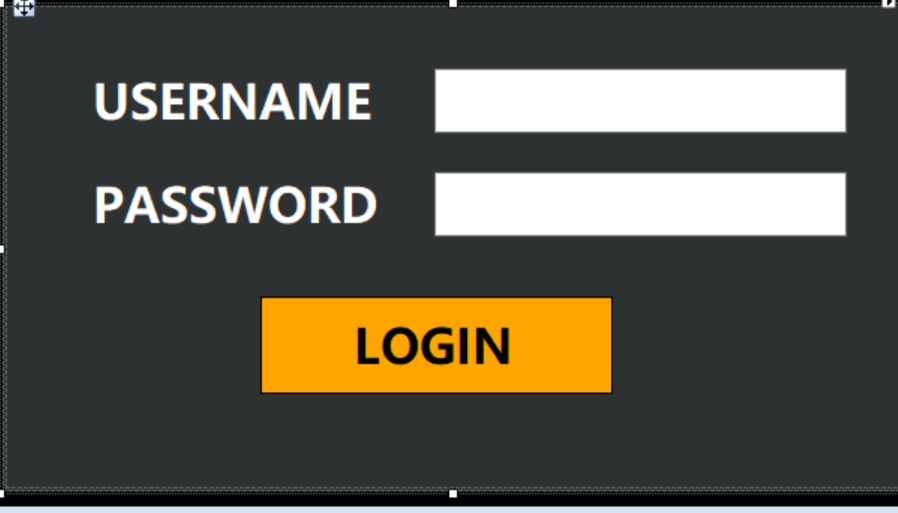
#### Iteration 1.2.1 – Display\_Login\_Screen

* Like my register form, I used Windows Forms to create this

For this, I created a goToLogin button:



And then a Login panel:



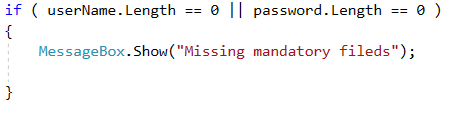
In the login panel, there are two labels a username label and a password label, and corresponding textboxes

#### Iteration1.2.2- Validate\_Logon\_Details

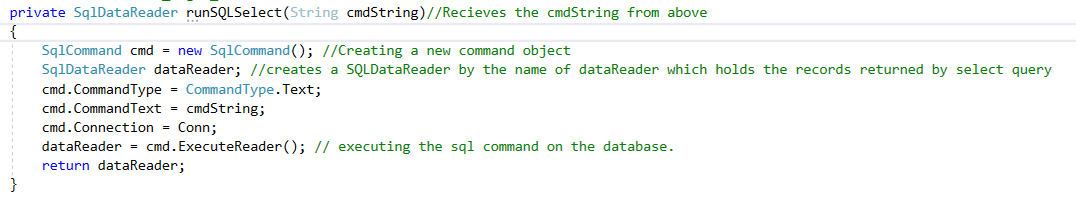
* I defined the username and password as data put into the textboxes



* The first thing to make sure was nothing could be left blank:



* In order to validate the code, we needed to connect to the database and execute the SQL query. When the form is loaded, connection is established to the database using an SQL connection object which is named “Conn”



* We pass an undefined cmdString through it and create a new object code called cmd:



* We then establish the properties of the command: the type of command, which is text, and the actual command as cmdString, which as stated is undefined:



* The command object uses connection established by Conn (SQL connection object):

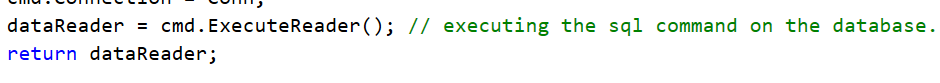




* Like when storing login details, we then create a SqlDataReader by the name of dataReader which holds the records returned by select query:



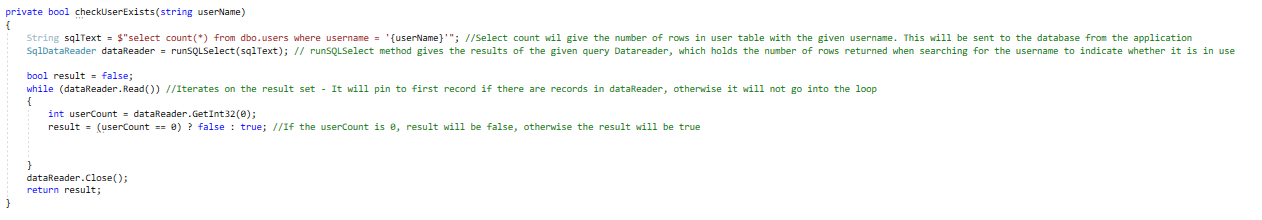
* The cmd returns the results to the dataReader object, which is returned



In order to ensure that the login works properly, there were two main things to do:

1. Ensure the account exists

* To ensure the username exists in the database, I created a new function called checkUserExists:

This function passes the userName defined in the registration through it

* We then create a sql command wherein we:

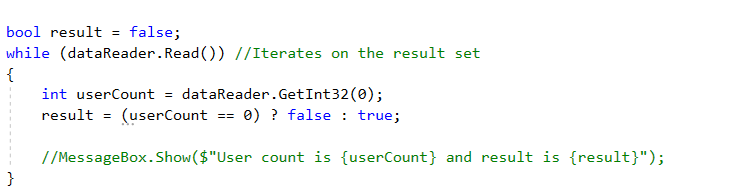


We then pass the string sqlText through runSQLSelect as the cmdString, which we establish as the sql command and the commandType as text:

* dataReader object holds the results. You process the dataReader object to extract the results sent by database

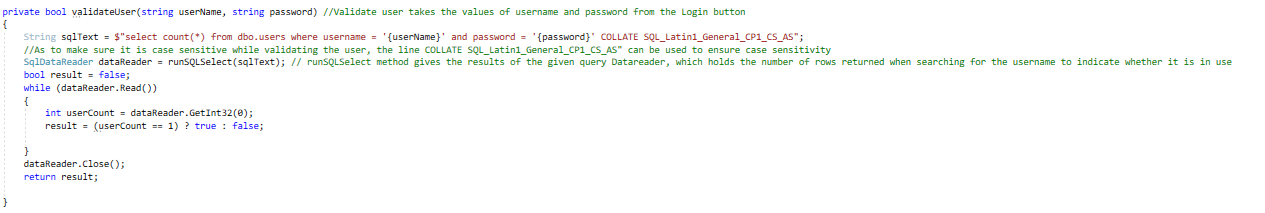


* A while loop is used to process the data in the dataReader object. We use the Read method for this purpose. When the data is exhausted, the while loop exists



* We then use this in the login button, passing username through it:



1. Ensure that the username and password matches with the details in the database

This method takes both the username and password and works out whether there is a matching combination for both. The user should exists and have the password specified.

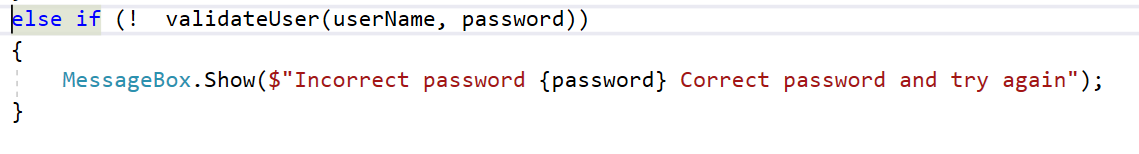
It uses the below sql statement and passes it to runSQLSelect which executes the SQL command against the database and returns results through dataReader object

We process the dataReader in a while loop

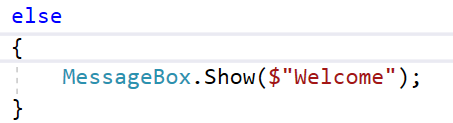
In the while loop, we read the data from the dataReader using read method, get the userCount using:



* If the user count is 1, this means that the user is valid
* We then use this method in the login button, passing both userName and password:

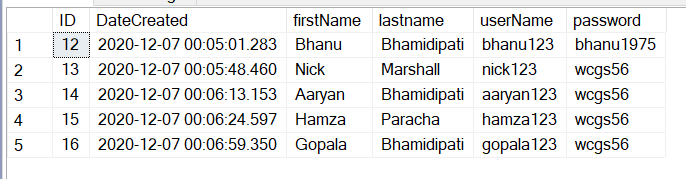


However, if everything was ok, it would message welcome:



Test data:

I first made 5 users using select \* from dbo.users:



|  |  |  |
| --- | --- | --- |
| userName | password | output |
| nick123 | wcgs56 |  |
| nick12 | wcgs56 |  |
| nick123 | wcgs5 |  |
| Blank | Blank: |  |

## Iteration 2 – Quiz

In the quiz iteration, we create a quiz that will enable the user to enter data about themselves. When the user registers and logs in for the first time, it takes them to the quiz page. This data includes current weight, target weight, date of birth, gender and diet type. This iteration has been broken down into two sub iteration:

* Setting user details
  + Setting user details involves making the quiz that will be answered by the user
  + In this quiz, it will question the user in order to extract that data from them
* Storing user details
  + Taking the data from the users, it will now be stored in the SQL database

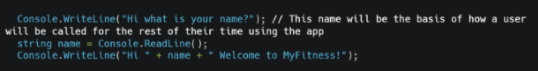
### Iteration 2.1 – Set\_User\_Details

#### Iteration 2.1.1 – Set\_Weight\_Goals

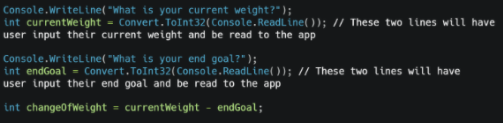
##### Cycle 1:



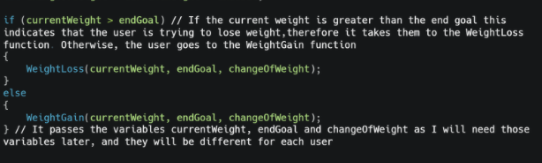
* For this section, I wanted a way to introduce the user to the app, after registration.
* The first thing to do was to ask the user for their name, so as to put them at ease:



* I then asked the user for the currentWeight and the endWeight, so as to calculate how much weight they want to gain or lose, which would be put into an integer called change:



* This would take them, based on whether change was positive or negative, to two separate functions, weightGain or weightLoss:



Cycle 1 Test Data:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Normal Data | Erroneous | Boundary |
| name | Gopala: | Blank: | G - I wanted to test how it works if you only use 1 letter: |
|  | Ivan: |  | Gggggggghggg - I wanted to test a long name that obviously isn’t real: |
| currentWeight | 70: | Blank: | 0: |
|  | 65: |  | 20000: |

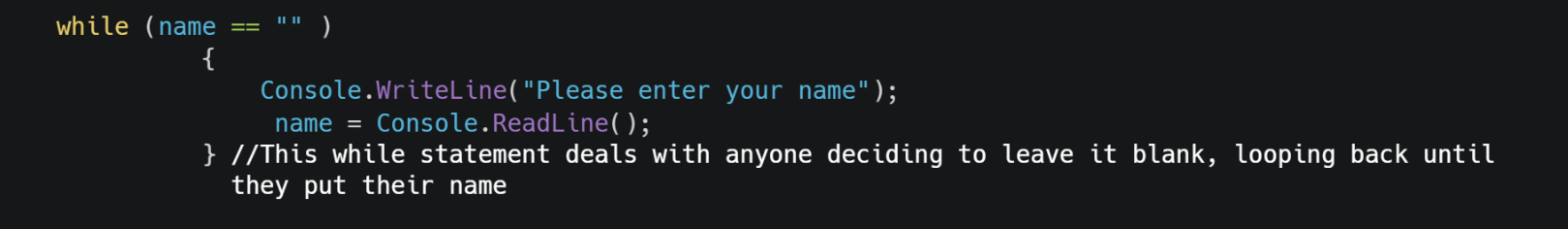
* Based on this Cycle, I found out that there was no way to check for errors, in case a user put a ridiculous name or weight that was far too high or a blank input
* I needed a way to be able to loop back into the original question
* I also needed a way to be able to use the variable name in other subroutines

Cycle 2

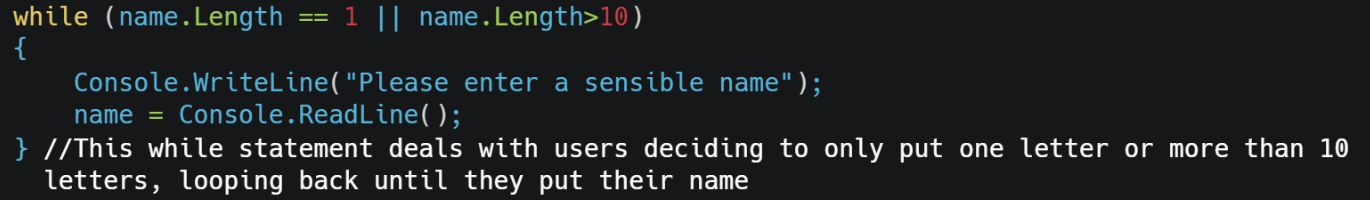
Based on what I learnt in the first Cycle, I decided that I would now add error-checking methods:

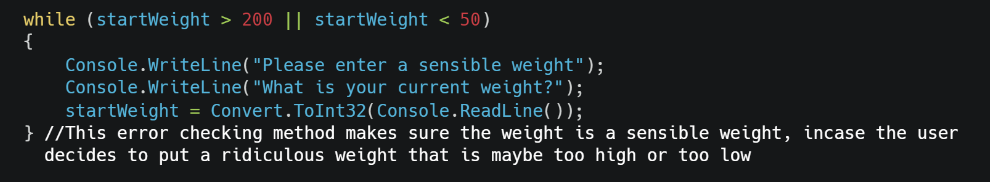
The first I added error checking to was “Name”:

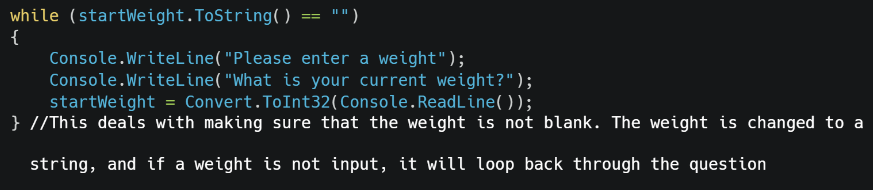
In case of a blank input, it needed to loop back to the original question:



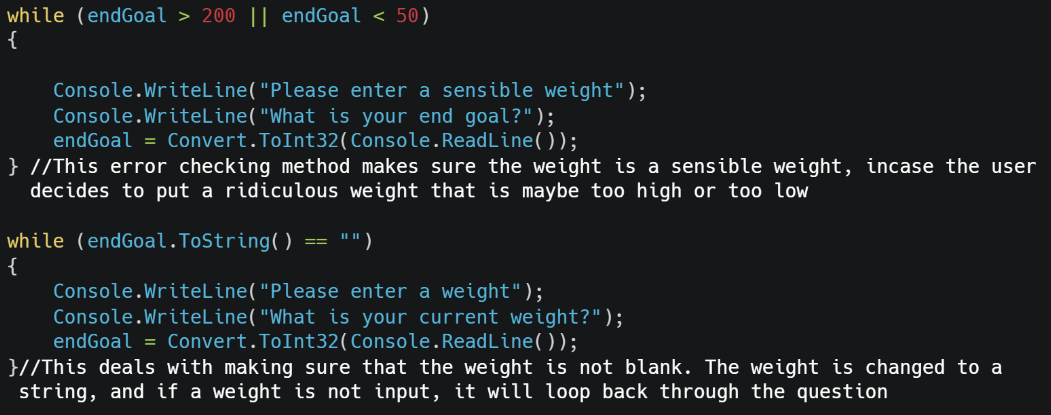
I then needed a way to make sure that, if a name is too big or too small, it will loop back as well:



For the startWeight variable, I added two error checking methods:

* I added a while loop in which it states that if the integer startWeight is above 200 or below 50, it will reject it and loop back around, asking the same question
* What this does is making sure the startWeight is in a certain range, and cannot go to ridiculously high or ridiculously low value
* This deals with a blank input
* The value is changed to a string, and checked to see if nothing is input
* If this is true, it will loop around and ask the same question

The same things are done for the endGoal variable:



Cycle 2 Test Data

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Normal Data | Erroneous | Boundary |
| name | Gopala: | Blank: | G: |
|  | Ivan: |  | Gggggggghggg: |
| startWeight | 70: | Blank: | 0: |
|  | 65: |  | 20000: |

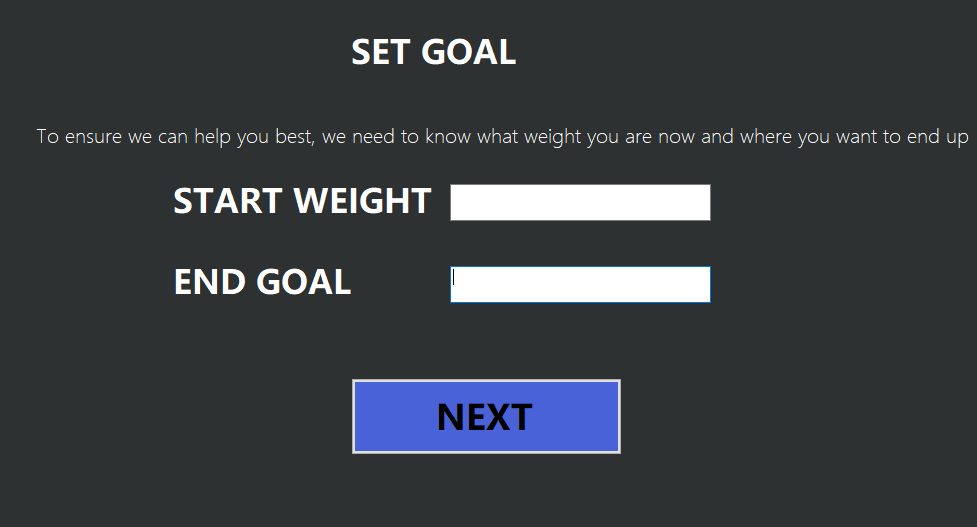
* Based on this Cycle, while some of it was working much better, some of the error checking methods were not working properly. For example:

1. Putting a blank in startWeight, and effectively endWeight, would not loop back to the question, instead declaring an error
2. There was no way in my code to be able to call an error for names that obviously aren’t real, such as Gggggggghg

##### Cycle 3 - Converting to Windows Forms

Now, I had to convert this all to Windows Forms. When doing this, I decided to take out the name and name checking the name as it was going to be done in the login and registration form

The first thing to do was to make a set Weight Goal panel:



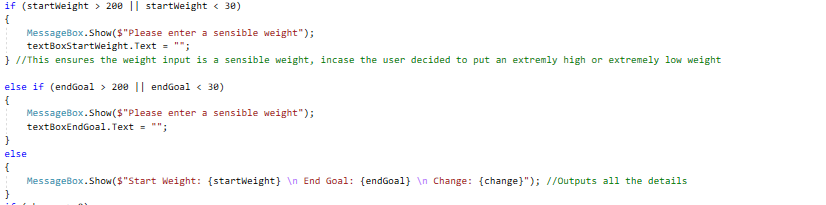
* In the c sharp code, there were two variables startWeight and endGoal. I have used the same algorithm and created two labels, startWeight and endGoal
* I then created error checking methods in the code:
* The first thing I did was create separate variables called startWeight, endGoal and change:



* As the textboxes are set by default as a string, I needed to convert it into an integer by using int.Parse



* I then needed to ensure that the weight input would be a sensible weight, therefore I used if statements:

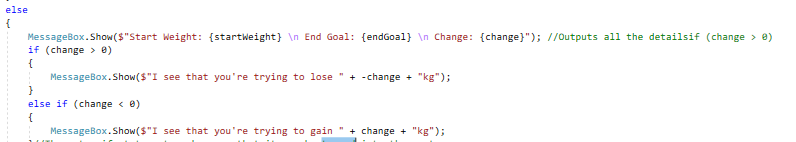


|  |  |  |
| --- | --- | --- |
|  | StartWeight | EndWeight |
| Normal 1 | 75: | 85: |
| Normal 2 | 55: | 45: |
| Boundary 1 | 0: | 0: |
| Boundary 2 | 1000: | 1000: |
| Erroneous 1 | -5: | -5: |
| Erroneous 2 |  |  |

* From this cycle, I found out that, while it does indicate that it isn’t a sensible startWeight or endGoal, it still continues to tell you how much weight to lose, which it shouldn’t.
* The change was the opposite way, which needed to change
* However, the biggest issue was that it had no way of handling a letter or special characters being added into it

##### Cycle 4:

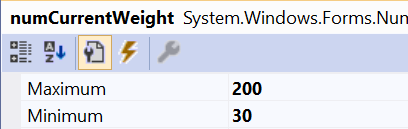
* To implement the changes, I decided to put the if change statements into the else statement itself:

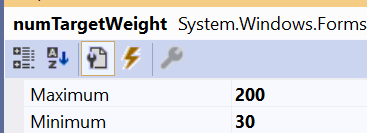


I then turned the change into endGoal - startWeight:

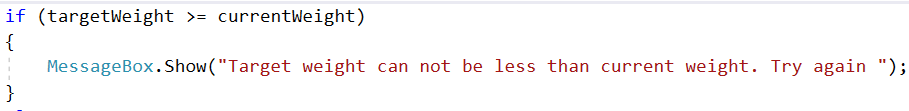


* I then added minimum and maximum weights to both, to ensure it can’t go over a certain value:





* However, I want to only work with weight loss for now, therefore I decided to make it impossible to make targetWeight > currentWeight:



Cycle 4 Test Data:

|  |  |  |
| --- | --- | --- |
|  | StartWeight | EndWeight |
| Normal 1 | 75: | 85: |
| Boundary 2 | 1000: | 0: |

Cycle 2 Test Data:

|  |  |  |
| --- | --- | --- |
|  | StartWeight | EndWeight |
| Normal 1 | 75: | 85: |
| Normal 2 | 55: | 45: |
| Boundary 1 | 0: | 0: |
| Boundary 2 | 1000: | 1000: |
| Erroneous 1 | -5: | -5: |

* From this cycle, I found out that, while it does indicate that it isn’t a sensible startWeight or endGoal, it still continues to tell you how much weight to lose, which it shouldn’t.
* The change was the opposite way, which needed to change
* However, the biggest issue was that it had no way of handling a letter or special characters being added into

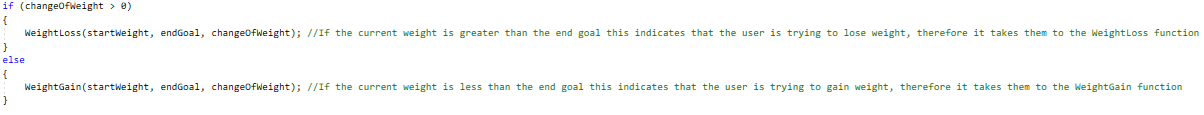
#### Iteration 2.1.2 – Set\_Exercise

##### Cycle 1:

I created two functions, called weightLoss and weightGain both with the integer parameters current, end and change:



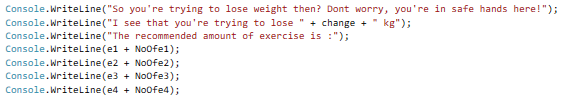


Using the variables startWeight, endGoal, changeOfWeight and from the Set\_Goal function, I used them as parameters to be able to pass them onto the weightLoss and weightGain functions:

* I decided to start on the WeightLoss function
* While these can be changed later, I decided to assign default exercises, which included press ups, sits ups, mountain climbers, squats:

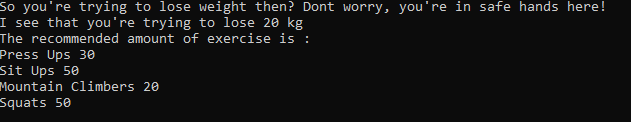


* To work out the number of each exercise which should be done, I took the value of change and multiplied it by a certain number. For example, for pressups I multiplied the change by 1.5:

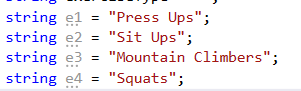


Cycle 1 Test Data:

For example, if startWeight was 65 and the endGoal was 45:

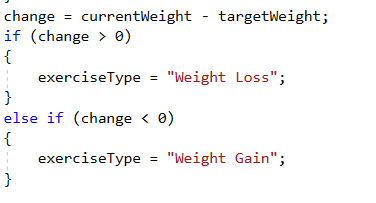


* While it didn’t need to be used yet, I implemented the same algorithm to my windows Forms:

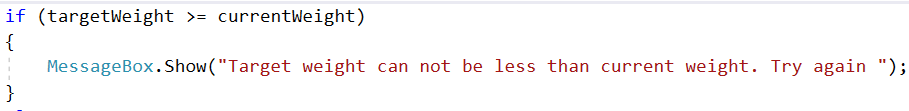




* I used currentWeight and targetWeight to calculate change, wherein we can define the exerciseType:



* However, I want to only work with weight loss for now, therefore I decided to make it impossible to make targetWeight > currentWeight:



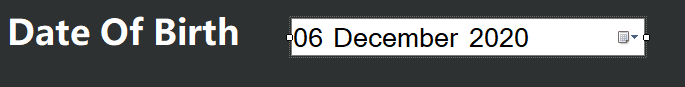
#### Iteration 2.1.3 – Set\_User\_Info.

##### Cycle 1:

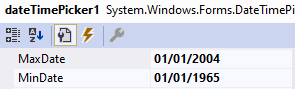
* I decided to do this straight away on Windows Forms, and in this section, I just asked basic questions:

1. What is your date of birth?

* For this, I used a dateTimePicker, and labeled it dateTimePickerDoB

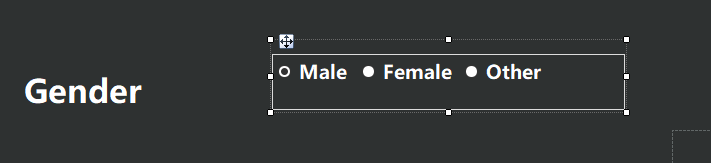


* I then set up a maxDate of 01/01/2004 and a minDate of 01/01/1965:



1. What is your gender?

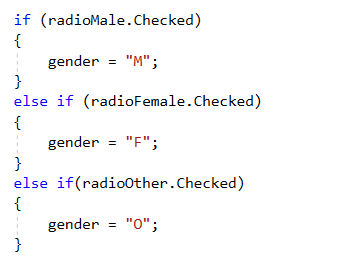
I used a groupBox and added three radios,  Male, Female and Other:



* I created a string called gender:



And defined each radio as the gender they are:



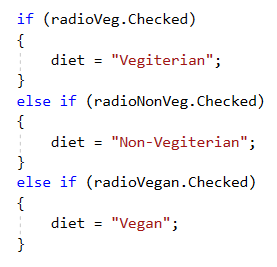
1. What is your diet type?

As above, I decided to add a group box and add three radios: Vegetarian, non-Vegetarian and vegan:



* I decided to use group boxes for both Diet Type and Gender as it reduces the scope for an error in this area, as the user cannot type anything
* Again, I defined a string as diet:





### Iteration 2.2 – Store\_User\_Details

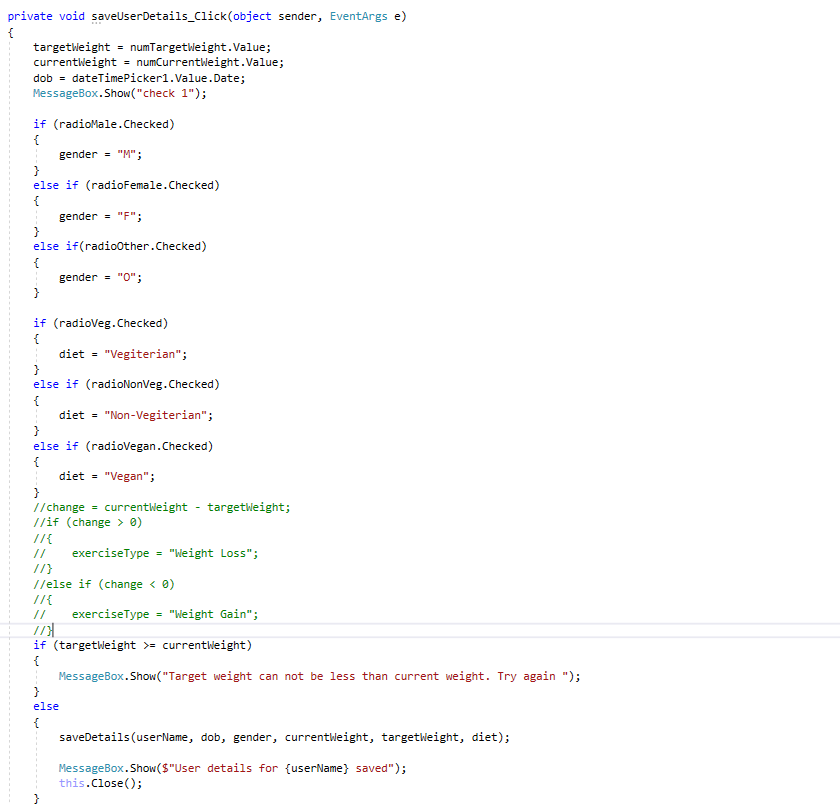
I first created a table called dbo.user\_details1:



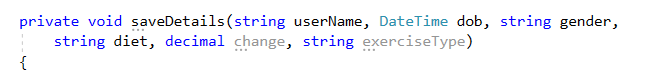
We use the button SAVE:

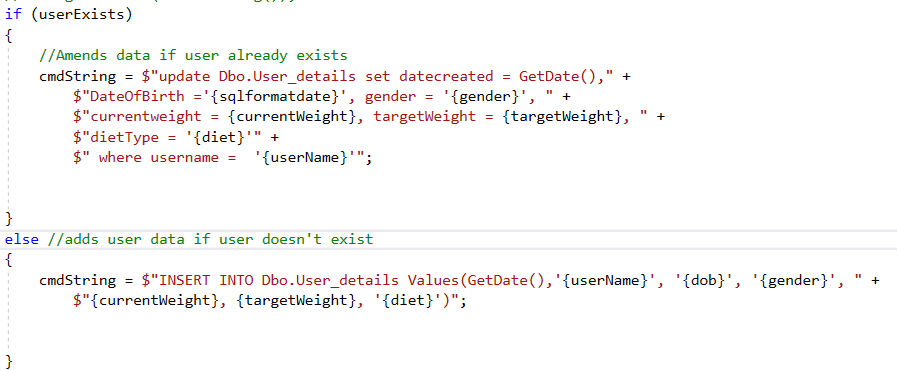


* To store the user details in database in a table called User\_details
* The click method of the save button gathers user details from the form and calls saveDetails method:



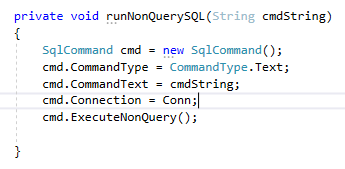
* The saveDetails method either adds or amends user data:





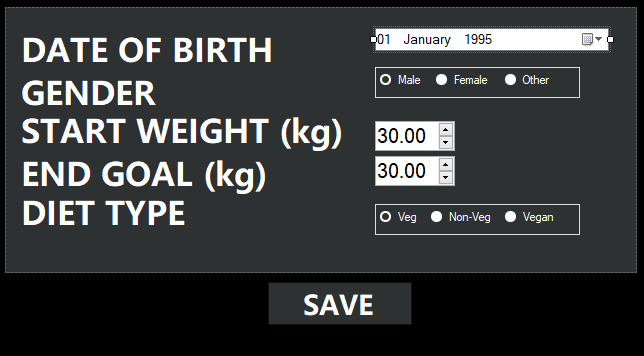
* The command string is then passed to the runNonQuerySQL as the command string and in the runNonQuerySQL connection is established to the database using an SQL connection object which is named “Conn”:





### Iteration 2.3 - Putting it all Together

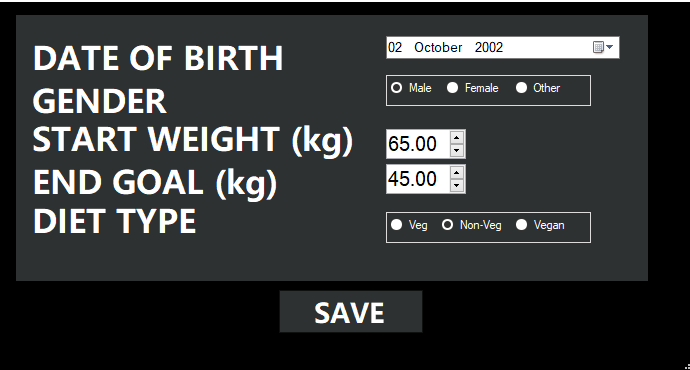
Having all the quiz separately, I put them all together:



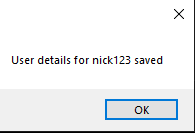
* Using a user I had previously by the name of Nick Marshall:



I inserted data into this:



In this data, I put date of birth as 2nd October 2002, gender as Male, startWeight as 65, endGoal as 45 and the diet type as Non-Veg. When I pressed save the user details were saved:



And using the line select \* from dbo.user\_details, I output in the database:



* This showed that the database user\_Details was connected to the quiz,