OCR A-Level Coursework Documentation

BODYBUILDING AND FITNESS PROGRAM

Kiran Rajappan

Royal Grammar School
Candidate number:2443
Centre number: 52423

Table of Contents

Analysis

Background information

Fitness is very important to stay in shape and life a healthy life. More recently, being healthy has become a really important topic. Now, by law all food items must show all ingredients, nutritional information and how it was made on all items. In addition, recent campaigns to increase daily activity and reduce the amount of stuff we eat have popped up everywhere. It has also reached a point to where a sugar tax has been introduced to promote a healthy lifestyle. Even with all the added emphasis to make people healthier, people still struggle to live a healthy life whilst trying to look as good as they want. Health and fitness doesn't directly affect a small population, it affects everyone, thus making it a very big problem. However, the amount of people that know the basic and key facts on how to stay healthy and keep fit are surprisingly little. According to Public Health England for 2014 showed that 61.7% of adults (over 16 years old) were overweight or obese and by 2050 obesity will affect 60% of men, 50% of women and 25% of children. These figures are quite alarming, showing that the majority of people will be affected by health and fitness related issues.

In the future, I hope to transfer my program into a mobile device, using programming languages like swift, for iphone and ipad. This is the original intent, so that clients can take this into their workouts with them portably rather than on a computer. As this is a prototype, this will be completed on a computer. For this to be able to be transferred to a mobile device, the window sizes will have to be variable, in order to fit all phone screens and portrait and landscape views.

Current system

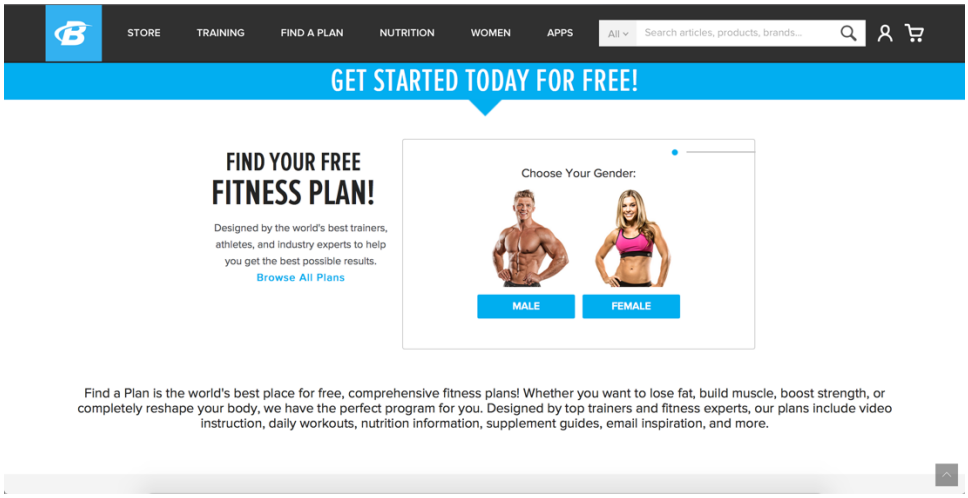
For most people, when they realize they need to change their ways in order to stay fit and healthy, they will usually find a personal trainer at a gym, app on their phone like SHealth or go onto google and take a couple hours to find a good website like BodyBuilding.com.

If they choose to use a personal trainer, usually they have to go to a gym, sign up to the gym and do inductions before they even start to talk about fitness. Usually, a personal trainer will take an entire week to fill out a form about the person and general habits. Once this is complete, the trainers usually fill this into a computerized form by hand over and over for each person. They then usually take down the person's email and send personalized emails for their clients every week. The personalized programs are usually planned for an entire month and can take time. Even the simplest aspect of working out daily calories using simple maths can be time consuming due to the repeated data for each person. Then the trainer must plan for workouts based on the client's special requirements, this can take a long time too.

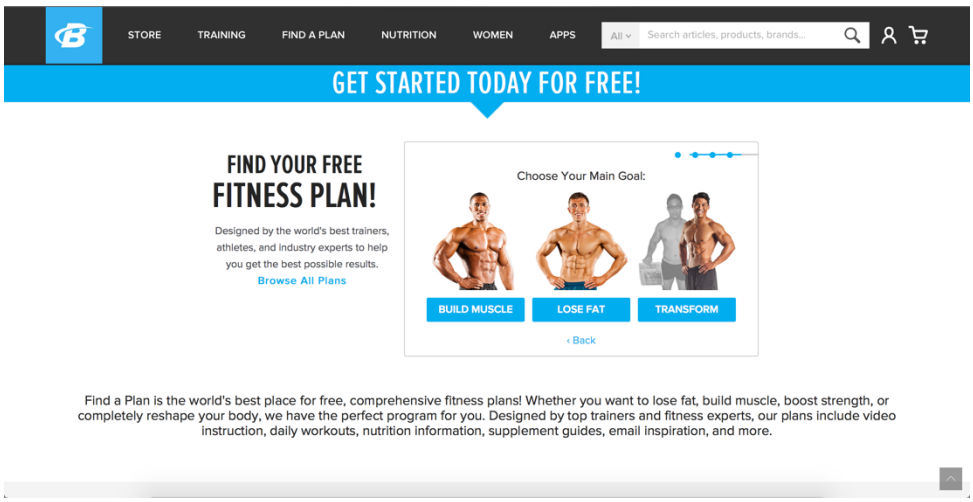
If they chose to use an online fitness aid, they usually go through creating an account. All personal data inputted are restricted only to the user and thus will not be seen by anyone else unless the user permits this. The account is secured by a username and password, the username being unique to prevent redundant data. The user must fill out an online form (this consists of basic information such as age, weight, height and gender as well as a certain physique the user wishes to achieve) and receiving an array of programs which are 'best suited' for them. This usually varies from between 20-30 different options for a user to pick out of. Once they have read through every program to see which is the best one for them, they will save it to their profile. Each time they wish to see their workouts, they will have to login in and find their way to their profile to view current programs. The programs listed are created by both

professionals and also anyone's own personalized workouts. These are pre-made and pre-set workouts that are specific for certain days and are recommended to be followed exactly.

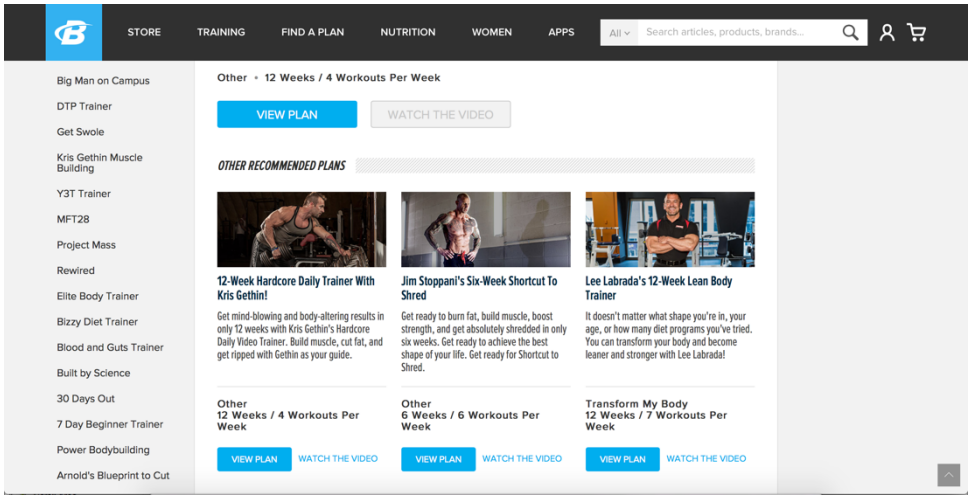
These are examples of online forms. These are usually the most popular forms which are used. These do require an internet connection to use and navigate. The similarities show how these programs, need an input from the user, just some basic information like height, weight, lifestyle etc. This is for the preliminary checks that must be used to work out stats like BMI. This is probably needed for all programs alike, and will also be present in my program.



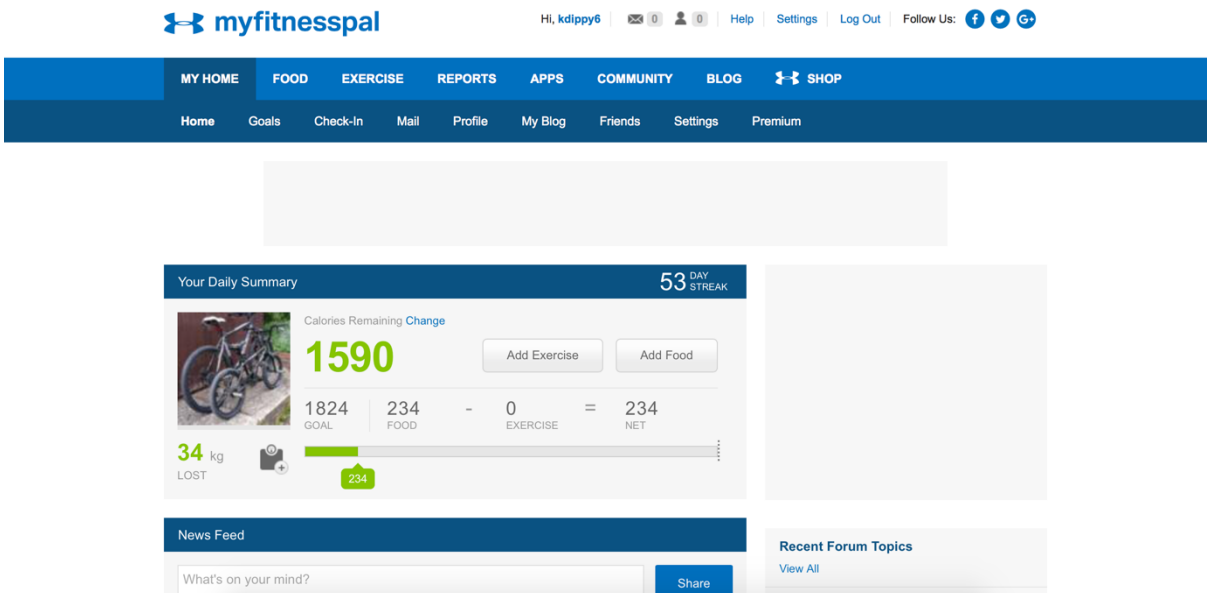
This is an example of a popular fitness program, which is bodybuilding.com. It asks you to fill out some basic information including age, height and weight.



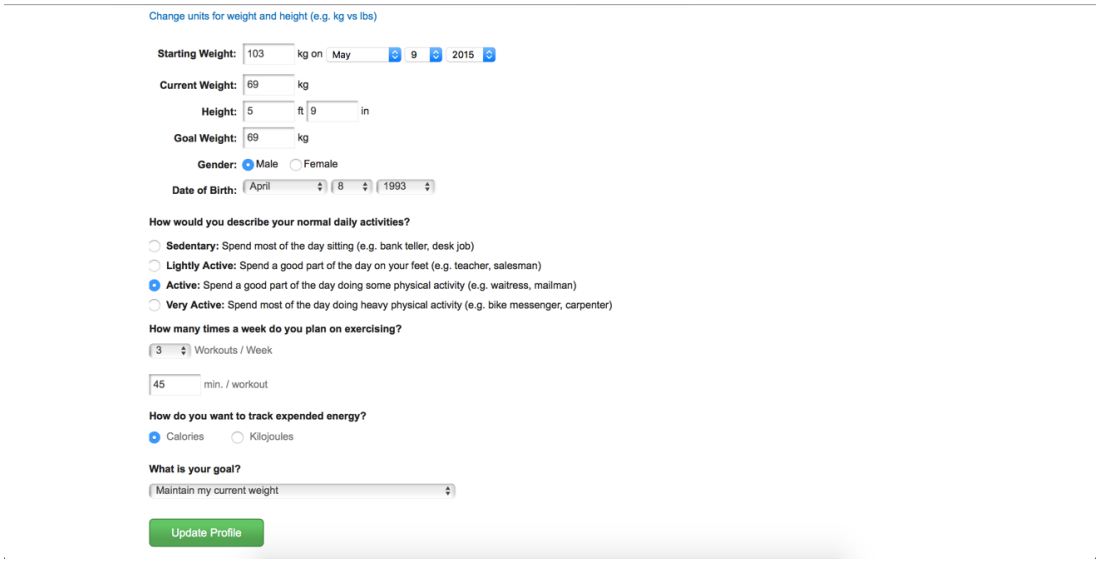
You then have to choose a type of physique you would like to achieve. Once you have decided, it will show you a list of workouts suitable for you.



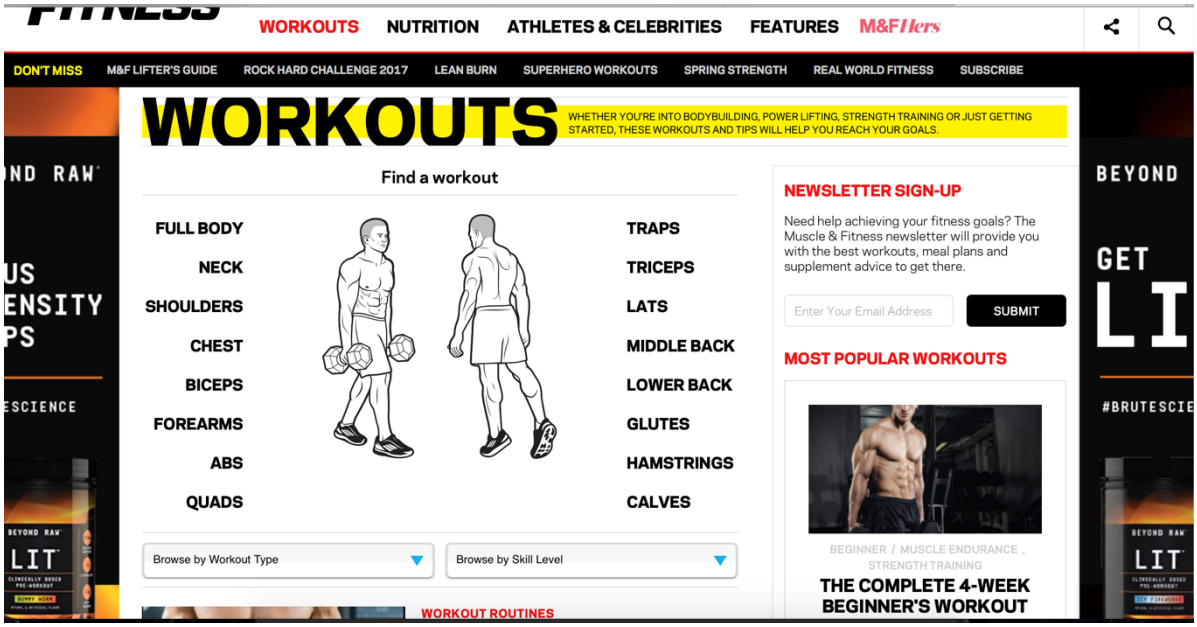
Another popular example is the app MyFitnessPal. This is a simple app which allows users to track the amount of activity they are doing in a day. The program can also allow users to scan the barcodes of any foods and track the amount of calories they are eating in a day. This can then be used to calculate if they will lose weight on that day. Although this is not directly linked to what I want to achieve with my program, it uses certain aspects that will be of very good use to me.



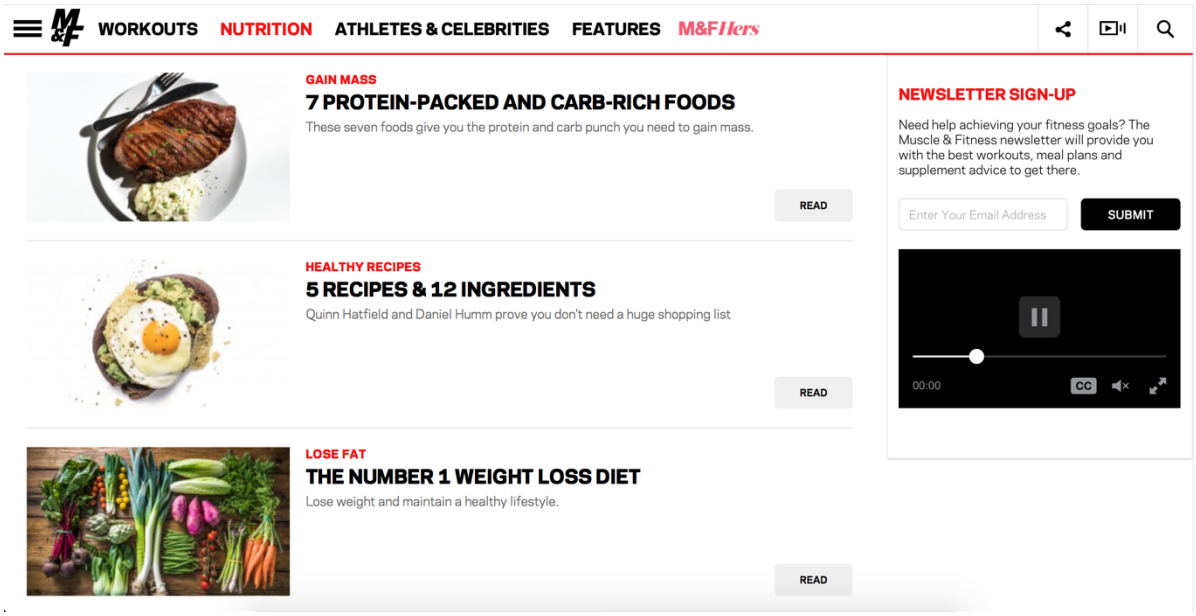
The main screen shows the amount of calories you should be having a day in order to achieve the goals for the user. This is a target set based on the information you have given when you signed up to the program. This includes height, weight, age, activity level and everything you see below. Although I am not planning or able to include a calorie counter feature in my program, I will be able to set the basic calorie target based on information filled out by the user. This will be possible by using basic algorithms based on the information and made easier using technology as it is quick and easy. Without technology, this would be a lengthy and laborious task. A personal trainer would have to take in hundreds of applications and work out each person's targets. With programming, this is made easier due to the fact that a computer can easily work on repeated enquiries, will not create errors and can work for long periods of time.



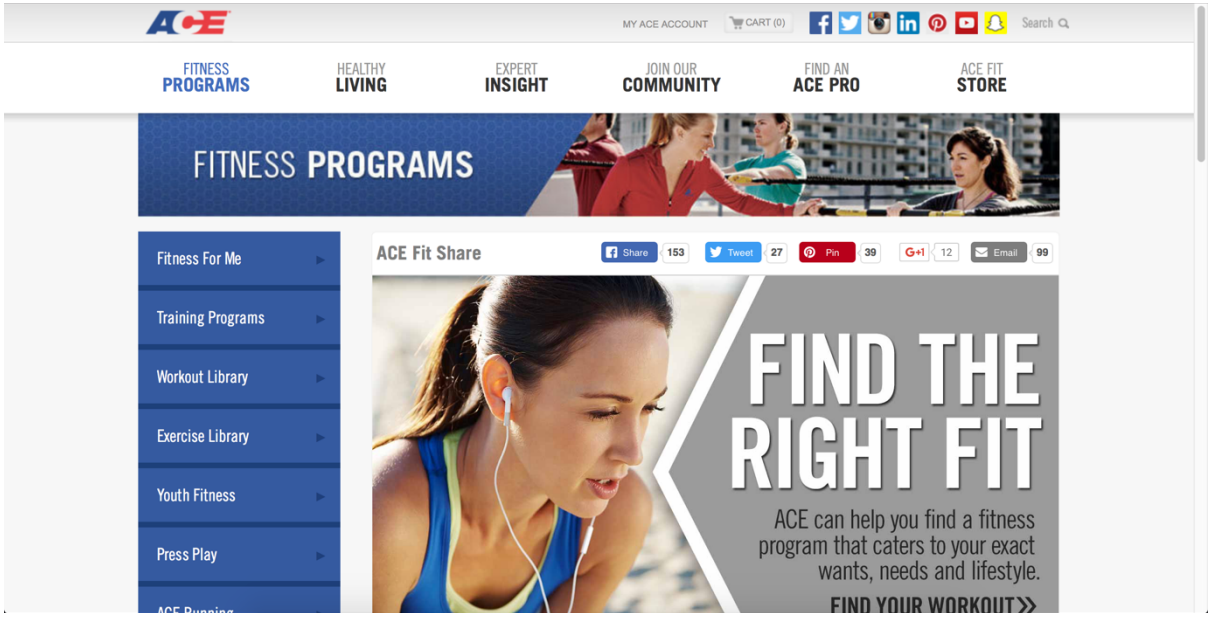
Another example is muscle and fitness website. This has both similarities with the two other sites, but decides to have a little twist that makes it different. Muscle and Fitness really just focusses on the aspect of separating workouts and nutrition. It allows the user to be free with their choices and allow them to make the decision themselves to what they want. They have very large database of different workouts for people to add and choose at their own will. It can be separated into the different muscle groups that are on offer to have specific choices. A recurring theme shows allows a choice of workouts to choose from, so that the user can change and or pick to what they would like better.



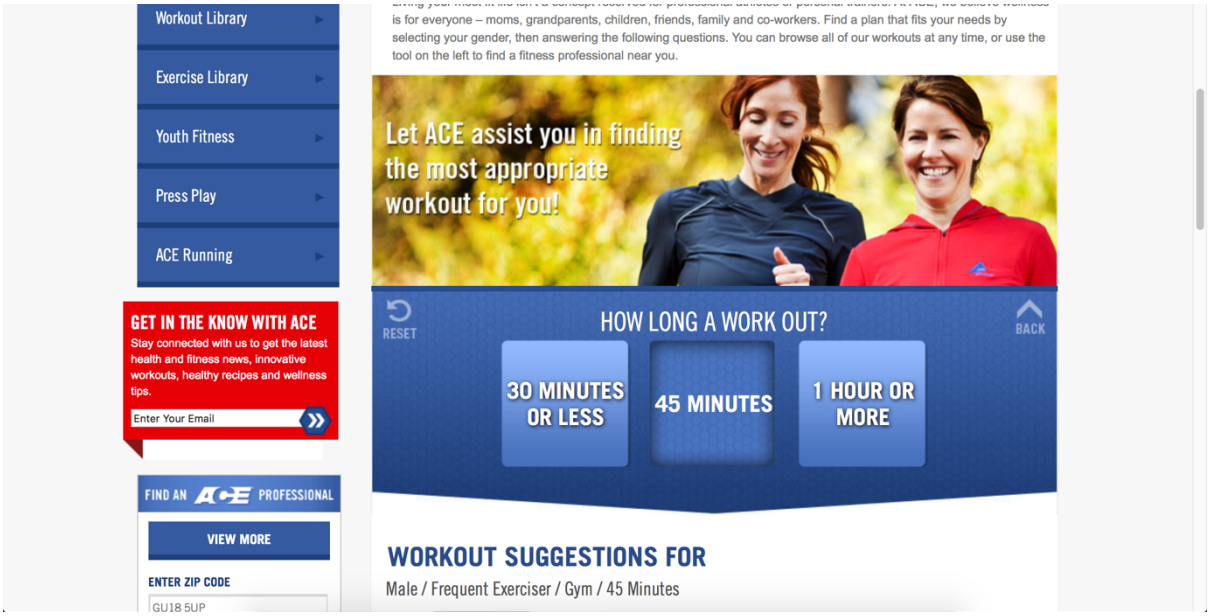
The nutrition screen also follows this concept, as just a library of different options for the user to choose themselves. Muscle and fitness does not offer a profile based system, which allows the user to input their details so that the choices can be personalized, unlike both myfitnesspal and bodybuilding.com. They all offer the same services, Muscle and Fitness just being less personalized focused.



The final program that I will be looking into is ACE fitness. This is not a website that I am not familiar with, but researching it further, it seems to be a mix between Bodybuilding.com and Muscle and Fitness. Offering a library of different training programs and exercises just to browse and then a personalized plan following some basic information.



The basic information consists of gender, daily activity levels, where you frequently workout and how long for. Following this, it will give some options of different workouts that is available. Unlike bodybuilding.com and muscle and fitness, this is not solely tied to just the gym. It offers cycling, running, track and at home workouts to adhere to everyone.



After reviewing my competition, its necessary to have some aspects that separate my program from there's. Even though all of these programs offer different aspects, they don't offer a built in function to workout calories and what exactly the user wants. With my program, I am planning

Possible by computational methods

Even though this is possible to do in a number of ways, by collating all of the information onto one platform, it is easy to compare and contrast with other users to see where you are placed out of all users. This is also much quicker as the algorithms to calculations are already stored and can be used to work out the person's details depending on multiple conditions. Instead of a trainer or user to work out multiple people's conditions and work out algorithms for each, the program will be able to compare and understand the conditions needed for each user and work this out for them much quicker. Most of these a preliminary calculation that all fitness users need and want. So even if this is not for personal use, personal trainers could set up accounts for users, just to have the calculations for their users digitally.

Stakeholders

The main target that my program will be attacking are those interested into fitness or ready for a lifestyle change. In recent years, health and fitness has become the center point of attention, with rising levels of obesity and new knowledge of foods and general diets. The main target market would be people aged 16-35. This is the ideal age for people to be interested into fitness and to follow the main dietary and exercise needs. However, anyone who is over 16 years of age can still use this. I believe that the perfect group of people that would use this are college/university students who are heavily engaged with socializing and are keen on keeping fit. To help with the testing and development of my program, I have decided to ask the help of Olek. Olek is a 17 year old, student at the Royal Grammar School. My overall client criteria is PureGym, this is a nationally acclaimed gym, having 180+ gyms

located around the UK. The company is growing at a rapid rate and also have good backing and advertising from Sir Chris Hoy. The company currently supports approximately 500,000 customers, which my program should be accommodating for. The company usually employees 2 staff and 12 self employed trainers, all of which could sign up users individually, or out of the 500,000 customers signing themselves up individually. To research the needs of pure gym, I will observe the current programs used by PureGym at my local site. From here, I will interview some of the trainers and staff on their views and how they would improve and like to see what it could offer. Furthermore, as this is not strictly for use of pureGym and gyms in general, I will ask some of the gym goers on their input.

I plan to implement, from the research of other products, a detailed list of workouts. These will be split into the different muscle groups that it targets, and will also have some detailed explanations and instructions of how to do these. From my initial plans, I was not planning on doing this, however after looking at competitors, this may be a good idea. When talking with Olek, he said that this could be a good idea, as not all of the users will have a personal trainer, and some of these people will also be beginners and so will not know how to do these for the first time. This would allow the customers to make better use of their time in the gym.

Meeting and interview

Real life potential user and candidate for my program – Olek Dzialak. Olek is a 17-year-old full time computing student. He is an average male, moderately active and a busy individual. This is a perfect candidate to help me with the evaluation and planning of my program.

With Olek included, I asked 3 other people including 1 that attend my school (17 year old male) as well as 2 other friends (23 year old male and 17 year old female) to help answer some questions to help me with my program.

I asked them the following questions:

1. Have you used a personal trainer to help you with your fitness goals before?

Yes	2
No	2

2. If so did you prefer the online form that I asked you to fill out and try? (BodyBuilding.com)

Yes	4
No	0

3. What were some of the positives that you liked in the online form?

I liked that I received all the help I needed straight away and didn't need to wait for a reply.
Filling in the forms were very quick and easy.
All the information I would ever need are all in close proximity and easy to access.
I can take the program wherever I go and use at all times.

4. Would you say this sped up the process in comparison to using a personal trainer?

Yes	2
No	0

5. In general do you prefer online forms or pen and paper?

Yes	3
No	1

6. Do you use and have access to a computer at all times?

Yes	4
No	0

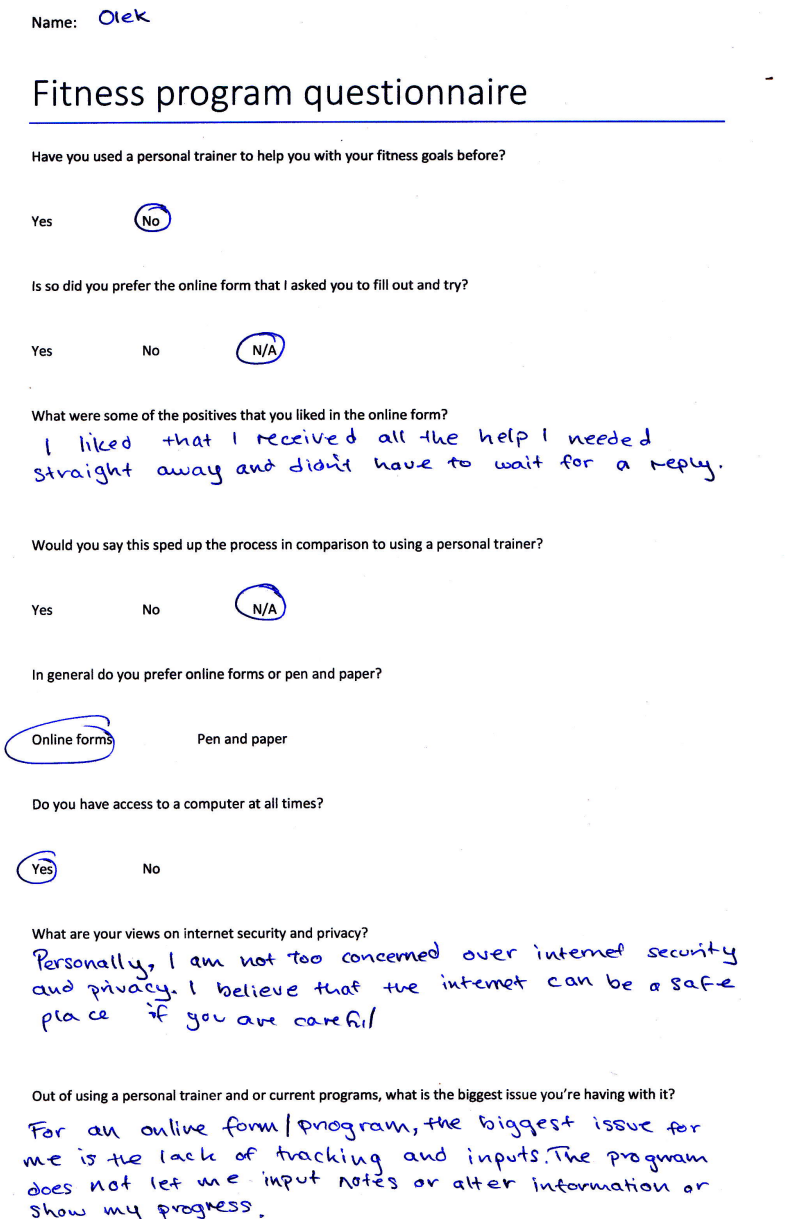
7. What are your views on internet security and privacy?

Personally, I am not too concerned over internet security and privacy. I believe that the internet can be a safe place if you are careful.
I am very cautious of this and rarely trust sites that I use. I rarely fill information unless it I am positive it is safe.
I am not bothered about internet security and safety, I believe that it doesn't really matter. I usually go on every site and fill in information where needed. I think this heightens the experience when a program/site knows all information needed.
Apart from sites that need my bank details, location and personal information like passport details, I am open to releasing information. Internet security and privacy are little relevance to me.

8. Out of using a personal trainer and or current programs, what is the biggest issue you're having with it?

For an online form/program, the biggest issue for me is the lack of tracking and inputs. The program does not let me input notes or alter information or show my progress.
I use a personal trainer and the biggest issue for me is the lengthy time to get my information from them. Although they are quick, it usually takes a couple days for them to email me my personalized plans.
With a personal trainer, I can not be independent and do my own work. I would like the professional help and guidance of a PT without needing them everyday. I feel like this will slow me down.
The online programs that currently stand don't offer flexibility. It gives you plans and programs, however, it doesn't allow you to personalize them for a specific day.

Below is a photocopied image of one of the questionnaires I handed out. This one is the paper of Olek, showing his answers.



Name: Olek

Fitness program questionnaire

Have you used a personal trainer to help you with your fitness goals before?

Yes (No)

Is so did you prefer the online form that I asked you to fill out and try?

Yes No (N/A)

What were some of the positives that you liked in the online form?

I liked that I received all the help I needed straight away and didn't have to wait for a reply.

Would you say this sped up the process in comparison to using a personal trainer?

Yes No (N/A)

In general do you prefer online forms or pen and paper?

(Online forms) Pen and paper

Do you have access to a computer at all times?

(Yes) No

What are your views on internet security and privacy?

Personally, I am not too concerned over internet security and privacy. I believe that the internet can be a safe place if you are careful

Out of using a personal trainer and or current programs, what is the biggest issue you're having with it?

For an online form/program, the biggest issue for me is the lack of tracking and inputs. The program does not let me input notes or alter information or show my progress.

Research analysis

1. This was a very good result for me, showing that some people use traditional personal trainers whilst others have adapted and used more modern techniques. This can help me to design the program to be better than already existing programs similar to mine and to become a better alternative to personal trainers.

2. I used bodybuilding.com as a good referencing point as it is similar to what I want to create. This is a very good example as an online form. The results were not surprising to me due to how more advanced and easy online forms are to filling forms in by hand. This already indicates to me that users prefer and like the idea of a virtual personal trainer and programs.

3. All of the responses highlighted how an online form made the process a lot more convenient. This being quicker, easier to access and very easy to use. This fixes most of the problems with personal trainers, making this a very good solution.

4. This is quite self explanatory and shows that the program hugely speeds up the process of using a personal trainer.

5. The results of this reveal how technology has been implemented successfully into everyone's daily life. Most people are used to having technology in their lives making it easier, which makes it quite clear that a personal trainer process should also become virtualized.

6. This shows that virtually everyone has access to a computer and shows how much the world has modernised. Which proves that everyday processes should also be made easier using technology.

7. The answers were quite varied from my group. Two said that they were not too concerned over internet security whereas the other two candidates were not overly confident with sharing information. To ensure that I adhere to the majority of people's views, I will have to make sure that I do not ask for overly personal questions and ensure that appropriate security measures are taken. This will ensure that the safety of information is an important factor to us to gain the trust of the consumers.

8. The main consensus of the results show how current methods are slow and quite inconvenient. Each of these comments must be taken into account to ensure that my program is the best solution possible.

The problem

To fix the problem with using a personal trainer can be fixed quite easily. By having an admin login for my program, I can computerize a form for personal trainers to bulk import information and sort clients to do the work for them. By granting extra access for personal trainers, they can fill out forms for clients easily.

The computerizing of this process will make data more secure, and easier to view when the trainers must overview the data. In addition to this, the program will use algorithms to work out calories and plans based on the information inputted. All the clients will have their own account that they can view once the trainers have set up their accounts for them, so that the clients can see the workouts and dietary needs needed for them.

The other issue with the current online services are the difficulty and time consuming aspect of navigating and using it. BodyBuilding.com supports multiple utilities such as forums, videos and an online store. This makes it quite complicated to use. Furthermore, once the basic information has been entered, an array of 20+ options will be listed ranging from professional programs to amateur personalized programs. These will vary in usefulness and may not even be beneficial. For an average person looking for a way to stay fit, this has far too many options and will confuse them.

Sites like BodyBuilding.com lack basic needs to plan and view their schedules. This can be inconvenient for the user as they will have to plan around the program, which sometimes is unavoidable. In addition to this, this program lacks basic diary capabilities. It has no options to add notes or make any changes along the way. Alongside this, there is no way of tracking data as you embark on your fitness journey. There are no dedicated programs just for this, which means that people will always have to be connected to the internet. This is not always possible, especially if this needs to be accessed at the gym with no connection to the internet.

Proposed solution

I plan to design and produce a dedicated health and fitness program to help people reach their fitness goals easily and simply. By allowing users to sign up to the program with a unique username and password, I will allow users to have their own profile with their own personal details. These details will be entered when users sign up and will include age, height, weight, gender and the physique that the user wants to look like.

Once the user has their profile and has chosen a physique they would like to look like (picked out of 3 options to keep it simple), the program will dedicate one simple program to them. Once this has been picked, the user can go to their profile to view the workout and schedule what days they would like to exercise on. This can be planned ahead or on that week. They can then add notes and reminders if they need to, too. Once they have figured out their exercise plans, they will have a dedicated area to check their dietary needs too. This will provide an accurate and personalized calorie plan, which will be calculated using the user's height, weight and body fat %. This can all be altered with the users needs. This will all be linked to a single account name and password, and thus can be accessed where ever they are. There are many different objectives that I will need to meet in order to make the program successful. Below are some of these objectives, which are also solvable by computational methods.

Calorie calculator – This is achieved by determining the base maintenance level of calories first. By taking in an input from the user of their weight (in lbs) converted from kg (multiple kg by 2.2). This will then be multiplied by a set multiplier. This will depend on the person's current activity. The more active their current lifestyle is, the more calories the person will need. If a persons lifestyle is sedentary, the multiplier must be 14. If the lifestyle is moderately active, the multiplier is 16. Finally, if the person has a very active lifestyle, then the multiplier is 18. This is just a preliminary calculation to work with the majority of people. This has been conducted through research and offers an accurate answer based on its simplicity. This will be a simple calculation easily solved by computational thinking just by multiplying. This will then have to be altered depending on the goals of the user. This will be split into another 3 options. Cut (lose weight), Bulk (gain muscle) and finally Maintain (increase athleticism). For both cut and bulk, a change in calories must be needed, however for maintaining, a change of calories is not needed but simply a change of the way the person will need to train. For a cut, the person's calories must be reduced, clearly. The size of the reduction must be dependent of the speed of which the person would like to reach their goal. The options are Aggressive (quickest possible), Balanced and finally Slow. For both cut and bulk would be + (for bulk) and – (for cut) of 200 calories from the original set calories. For a slow

rate would be +- 50 calories and for aggressive would be +- 100 calories. A projected weight for each would be +- 5lbs/week for an aggressive rate and +- 1lb/week for a slow rate. Finally, for a balanced rate, it will be +-3lbs/week. For Maintenance, this will always be 0lbs/week. This is easily possible by computation thinking and is much easier than doing it in person. The various options will need lots of variables needed to be remembered and depending on the person can always change. By doing this computationally, it reduces the redundancy of a person working this out by themselves and makes it simple to use.

The workouts are preset by fitness trainers and can be altered depending on the user's goal (i.e Warrior, Superhero and Greek God). This will change the sets and reps of each program, to offer different ways of activating the muscle groups to help them to their specific goal. This will be achieved through computational methods by checking the persons age and goal to give them the specific reps and sets. This will consist of SS- straight sets which will be in the following format – 4x10. This will mean 10 repetitions, take a break and repeat 4 times. RP- Rest pause in the format – 12 reps 3 x 4-5 . This means a first set of 12 reps, then take a 15 sec break followed by 1 set of 4-5 reps, rest and then another set of 4-5. DS – Drop set in the format 4x10. This will mean that the person must start with a heavier weight within 10 reps, drop the weight and add an extra 2 reps.

BMI – Even though this is a simple calculation, it is a good basis in order for the user to ensure that they are progressing and to show a general level of their health and where they are in comparison to average. To work out the BMI of a user, it will take the inputs of the user's height and weight, taken from when the user signs up. The calculation will be the person's weight (in kg) / height (cm). This will then show on a table, highlighting the category that the person is in, depending on their BMI. This is underweight, overweight, obese.

Limitations with solution

Some ideas that I would like to implement would take too long to complete, or is not within my skill range to do. One idea that I would like to have is a personal tracker. This would allow the program, like in myfitnesspal, to track the amount of steps the person takes in a a day, which will be converted into calories burned. This will then be coupled with a tracker in order to calculate the amount consumed in a day, to have a full diary. This could be possible, given some extra time, however with time limitations I have this is not possible. Furthermore, I would like to allow the user to time their workouts, to ensure that they are taking correct breaks and not wasting time. However, as my prototype is being constructed using a PC, this will not be worth the time taken as this will not be useful for computer use. It will be fairly simple to implement, however.

Product requirements

Requirements	Justification
Display: 480x360px	This is the largest set size window on pyqt
RAM:2GB	This is the minimum requirement for python to run pyqt4

PROCESSOR: 1GHZ or higher	This is the minimum requirement for python to run pyqt4
Graphics: On-board processor graphics	Minimum specs required for pyqt4 windowed tabs to work
50Mb of free disk space	Minimum space needed to store modules and programs
OS: Windows vista or higher	Minimum OS required for python to run pyqt4 and extra modules.
VGA or higher-resolution monitor	Needed to display outputs of program
Microsoft mouse or other compatible	To navigate the menus and press the buttons on screen
HP printer or other compatible	To print out workouts and any notes made on program
Computer keyboard	To input data into text boxes and line edits
Working internet connection	To ensure an email can be sent to the user

Usability

- User must register to an account
- User must log on account with provided username and password
- Allow user to pick the days that they wish to workout on
- Provide admin screen to maintain
- When user requests to change password, make sure password hash is the same as db hash
- Ensure all windows have back buttons for user to navigate
- Ensure admin can access admin screen
- Have access to reclaim password if forgotten

Reliability

- Ensure username is no longer than 12 characters
- Ensure username is longer than 4 characters
- Ensure username does not contain prohibited characters
- Ensure forename is no longer than 15 characters
- Ensure forename is longer than 2 characters
- Ensure forename does not contain prohibited characters
- Ensure surname is no longer than 20 characters
- Ensure surname is longer than 4 characters
- Ensure surname does not contain prohibited characters
- Ensure password is no longer than 20 characters
- Ensure password is longer than 4 characters
- Ensure password does not contain prohibited characters
- Ensure password and re-enter password is the same
- Ensure that email address follows a valid format (must contain '@', followed by '.')
- Ensure age is within limits that is allowed to use the program (16 -80 years old)

- Ensure height entered is within appropriate level (130cm-230cm)
- Ensure weight entered is within appropriate level (40kg-250kg)
- Ensure a gender is picked
- Ensure a physique type is picked
- Ensure a security question is picked
- Ensure security answer is hashed in database
- Ensure the user inputs lifestyle
- Ensure the user inputs goal
- When storing password in db, this must be hashed
- Prevent user inputting more or less days than needed
- Prevent user entering the same day more than once
- Ensure user inputs rate of goal achieved
- Ensure pie chart calculations are correct
- Ensure a security answer is entered
- Ensure amount of days picked is not below 3
- Ensure amount of days picked is not above 3
- Ensure the same day is not picked twice

Performance

- When user requests forgotten password
    - Check security answer entered by user matches stored security answer
    - If match, send user an email with random code
    - Change password in database with hashed random code
- Show all relevant information of all the users
- Allow admin to alter specific information
- Allow admin to delete users
- Show on calendar, the specific workout they have on a set day or have rest day
- Show the different workouts available
- Calculate BMI with weight and height inputted
- Calculate Calories needed for person
- User can alter and update details including passwords
- User must be able to retrieve account if forgotten password
- Must be able to send retrieval password to linked email
- Show a pie chart of amount of users picking different physiques
- Must show the user specific workouts based on details
- Calculate each part of the pie chart
- Ensure days picked, correlate to each of the workouts, for correct order
- Send email to user for forgotten password

Maintainability

- Comments for coding throughout
- Have modular code for functions
- Meaningful variable names
- Make sure database is in 3rd normal form

I will prove and ensure that these objectives are met by creating a test plan for each of these points. These points will have a list of objectives to test for to ensure that they are successful and are all met.

Design

Decomposing the problem

Justifying the problem

A checklist of requirements needed for my program:

- Profile system – Username and password
- Basic profile details – Name, Age, Gender, Weight, Height, Physique type
- Workout planner
- Workout editor
- Calorie planner
- Calorie editor
- Profile viewer
- Profile editor
- Quick, easy and simple to use GUI.
- Clean, attractive and professional interface.
- Be successful and helpful to any users.

Decomposing the program into ordered lists

The main objective that must first be in my schedule should be the database. This is the basis of my program and the program must be designed to fit a normalized database. This is to ensure it runs efficiently and safely.

To set up all of the navigation windows, it is necessary to create the GUI for the users and admin to navigate the program. This has to be set up before any coding development is completed.

The next part that should be addressed is the profile system, of logging in and out. This will provide the start to the GUI structures. From here, it will be split into customer navigation and admin utilities.

Selection, searching and deletion must be next to link with the database. While navigating through the GUI windows, it is necessary to show, update and delete any information required by the user at any stage.

Before any significant progress will be made to the program, it is important to validate as the program is being completed. This will prevent any errors that Olek, or any users could have while testing my program throughout its progress.

Pseudocode and explaining the problem

Login

```
Username= INPUT("Enter Username")
Password= INPUT("Enter Password")
ListsOfUsers=[]
C=OPEN "USERS.TABLE"
ListUsers=C.READ_TABLE()
FOR i=0 TO LENGTH(ListUsers)
        APPEND (ListsofUsers) with ListUsers[0][i]
        If Username = ListUsers THEN
                IndexUser=i
                If Password = ListUsers[1][IndexUser] THEN
                        Print ("succesful login")
        Else
                Print("incorrect username or password")
        ENDIF
NEXT i
```

Sign up

```
Username=INPUT("Enter username")
Password=INPUT("Enter password")
ListsOfUsers=[]
REPassword=INPUT("Re-enter Password")
IF Password=REPassword THEN
        Pass
Else
        Print("Passwords must match")
ENDIF

If LENGTH username > 15 THEN
        print ("error with length of username")

If LENGTH password > 15 THEN
        print ("error with length of password")
END IF
FOR i=0 TO LENGTH(ListUsers)
        If username = usernameslist[i] THEN
                print error username exists.
        END IF
```

Change Password

```
Var=INPUT("Enter Username")
ListsOfUsers=[]
C=OPEN "USERS.TABLE"
ListUsers=C.READ_TABLE()
FOR i=0 TO LENGTH(ListUsers)
        IF Var=ListUsers[i] THEN
                Index=i
                If length password > 15 THEN
                        Then print error with length of password
                        If length password > 15 THEN
                                Then print error with length of password
                                Users.passwordlist[index]=password
                                Print("password changed")
                        ENDIF
                ENDIF
        ELSE
        Print("user does not exist")
        END IF
NEXT i
```

Request Password

```
When button pressed
        Enter security answer
        If security answer = profileanswer
                Update profilepassword to 1234
        END IF
Send email to profileuser
Print 'password has been reset, please update'
```

Calculating calories

```
Weight=Input("Input your weight")
Height=Input("Input your height")
BMITemp=height/weight
BMI=round(BMITemp)
Lifestyle=Input("What's your type of lifestyle")
Speed=Input("How fast would you like to reach your goal")
Goal=Input("What is your goal")
If lifestyle = ("Sedentary") THEN
        Multiplier=14
If lifestyle = ("Active") THEN
        Multiplier=16
If lifestyle = ("Very Active") THEN
        Multiplier=18
ENDIF
Weightlbs=Weight*2.2
CalorieIntake=Weightlbs*Multiplier
If goal=("Cut") THEN
```

```
        Calorieintake=calorieintake-200
        If speed=(“Aggressive”) THEN
                Calorieintake=calorieintake-100
                Projectedweight=(“-5lbs/week”)
        If speed=(“Balanced”) THEN
                Calorieintake=calorieintake+50
                Projectedweight=(“-1lbs/week”)
        Else
                Projectedweight=(“-3lbs/week”)
        ENDIF
ENDIF
If goal=(“Bulk”) THEN
        Calorieintake=calorieintake+200
        If speed=(“Aggressive”) THEN
                Calorieintake=calorieintake+100
                Projectedweight=(“+5lbs/week”)
        If speed=(“Balanced”) THEN
                Calorieintake=calorieintake+50
                Projectedweight=(“+1lbs/week”)
        Else
                Projectedweight=(“+3lbs/week”)
        ENDIF
ENDIF
If goal=(“Maintain”) THEN
        Projectedweight=(“+0lbs/weel”)
ENDIF
```

Registering Account Validation

```
Forename=Input(“Enter Forename”)
Surname= Input(“Enter Surname”)
Gender= Input(“Enter Gender”)
If Gender(“Male”) or (“Female”) THEN
        Pass
Else
        Print(“Enter valid response”)
ENDIF
DOBAge= Input(“Enter Date of Birth”)
TodaysDate= GET_SYSTEM_DATE()
this year=SPLIT_ALONG_SLASH(TodaysDate)
bornYear=SPLIT_ALONG_SLASH(DOB)
Age=bornYear-this year
If Age<16 THEN
        Print(“User is too young”)
ENDIF
Physique= Input(“Enter Physique”)
IF Physique = (“GreekGod”) or (“Superhero”) or (“Warrior”) THEN
```

```
        Pass
ELSE
        (“Enter Valid Physique)
ENDIF
Weight= Input(“Enter Weight”)
If 40>weight>200 THEN
        Print(“weight is not within limits”)
Height= Input(“Enter Height”)
If 120>height>250 THEN
        Print(“height is not within limits”)
C=OPEN “PROFILE.TABLE”
Profiles=C.WRITE_READ_TABLE()
APPEND(Profiles) with
FirstName=Forename,SecondName=Surname, GenderInput=Gender,
DOB=DOBAge, ChosenPhysique=Physique, UserWeight=Weight, UserHeight=Height
```

Admin Delete Users

```
C=OPEN “PROFILE.TABLE”
Profiles:=C.READ_TABLE()
selectedUser=Input(“Enter account name to be deleted”)
FOR i=0 TO LENGTH(Profiles)
        IF Profiles[i].UserID=”selectedUser” THEN
                REMOVE Customers[i]
        ELSE
                Print(“User does not exist”)
        END IF
NEXT i
```

Describing the solution

The problems of the market that I am trying to tackle can be broken down, so that it can be computationally fixed. The problems can be broken down thusly:

- Give clients the knowledge they need about their basic nutritional, physical and mental stats.
- Offer clients various options on how to improve their future depending on their circumstances.
- Make clients aware and set goals towards their future.
- Allow clients to be responsible about their goals and view their progress.
- Set out client’s objectives and help them along the way.

By breaking down the bigger problem into smaller, more accessible and specific problems, we are able to give better detail into how to fix this.

Give clients the knowledge they need about their basic nutritional, physical and mental stats. – This can be solved quite simply. By asking the user quite detailed and relevant information, whilst complying with the 1998 Data Act laws, it is possible to deduce some basic data to the user that they may not previously have known about. Data such as height, weight, age, sex and daily activity. These can be used to produce, recommended daily calorie allowance.

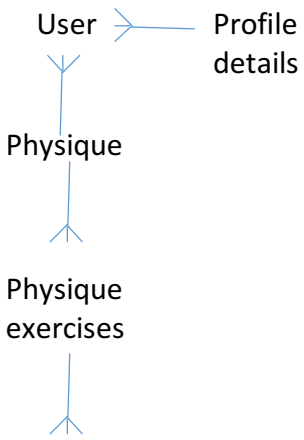
The processes used throughout my program are as follows

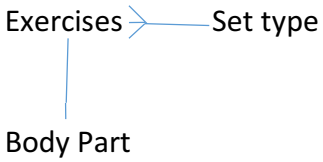
Pseudocode for validation

Length and prohibited characters checking

```
PROCEDURE LengthCheck(item)
        prohibChars=[£,#,%,^,&,*,(,),:,;,’,[,],{,},|,\,/,?,<,>,±,§,=.-._]
        LengthofItem=LENGTH(ITEM)
        IF LengthofItem<4 THEN
                Print(“Input is too short”)
        IF LengthofItem>12 THEN
                Print(“Input is too long”)
        FOR i=0 TO LENGTH(prohibChars)
                FOR j=0 TO LENGTH(LengthofItem)
                        IF prohibChars[i]=item[j] THEN
                                Print(“Item contains prohibited characters”)
                        ELSE
                                Return (True)
                        ENDIF
        ENDIF

END PROCEDURE
```

Identifying key variables and data structures





Database tables

Bodypart Table

Columns	Data type	Description	Example
BodyPartID	Integer	Primary key	01
BodyPartName	String	Name of body part	Chest

Exercises

Column	Data type	Description	Example
ExerciseID	Integer	Primary key	01
BodyPartID	Integer	Foreign key for Bodypart table	01
WorkoutDescription	String	Description of specific workout	Squat

GreekGodBuild Exercises

Column	Data type	Description	Example
GGBExerciseID	Integer	Primary key	01
ExerciseID	Integer	Foreign key for exercises table	01
Sets	Integer	Amount of sets to complete	2
Reps	Integer	Amount of reps to complete	4-6
SetTypeID	Integer	Foreign key for set type table	01

GreekGodCut Exercises

Column	Data type	Description	Example
GGCExerciseID	Integer	Primary key	01
ExerciseID	Integer	Foreign key for exercises table	01
Sets	Integer	Amount of sets to complete	2
Reps	Integer	Amount of reps to complete	4-6
SetTypeID	Integer	Foreign key for set type table	01

WarriorBuild Exercises

Column	Data type	Description	Example
WBExerciseID	Integer	Primary key	01
ExerciseID	Integer	Foreign key for exercises table	01
Sets	Integer	Amount of sets to complete	2
Reps	Integer	Amount of reps to complete	4-6
SetTypeID	Integer	Foreign key for set type table	01

WarriorCut Exercises

Column	Data type	Description	Example
WCExerciseID	Integer	Primary key	01
ExerciseID	Integer	Foreign key for exercises table	01
Sets	Integer	Amount of sets to complete	2
Reps	Integer	Amount of reps to complete	4-6
SetTypeID	Integer	Foreign key for set type table	01

SuperheroCut Exercises

Column	Data type	Description	Example
SCExerciseID	Integer	Primary key	01
ExerciseID	Integer	Foreign key for exercises table	01
Sets	Integer	Amount of sets to complete	2
Reps	Integer	Amount of reps to complete	4-6
SetTypeID	Integer	Foreign key for set type table	01

SuperheroBuild Exercises

Column	Data type	Description	Example
SBExerciseID	Integer	Primary key	01
ExerciseID	Integer	Foreign key for exercises table	01
Sets	Integer	Amount of sets to complete	2
Reps	Integer	Amount of reps to complete	4-6
SetTypeID	Integer	Foreign key for set type table	01

Physique

Column	Data type	Description	Example
PhysiqueID	Integer	Primary key	01
PhysiqueType	String	Type of physique	Warrior

Set Type

Column	Data type	Description	Example
SetTypeID	Integer	Primary key	01
Type	String	Type of set	RPT
Description	String	Description of type of set	Reverse pyramid training – drop weight and do more reps each set

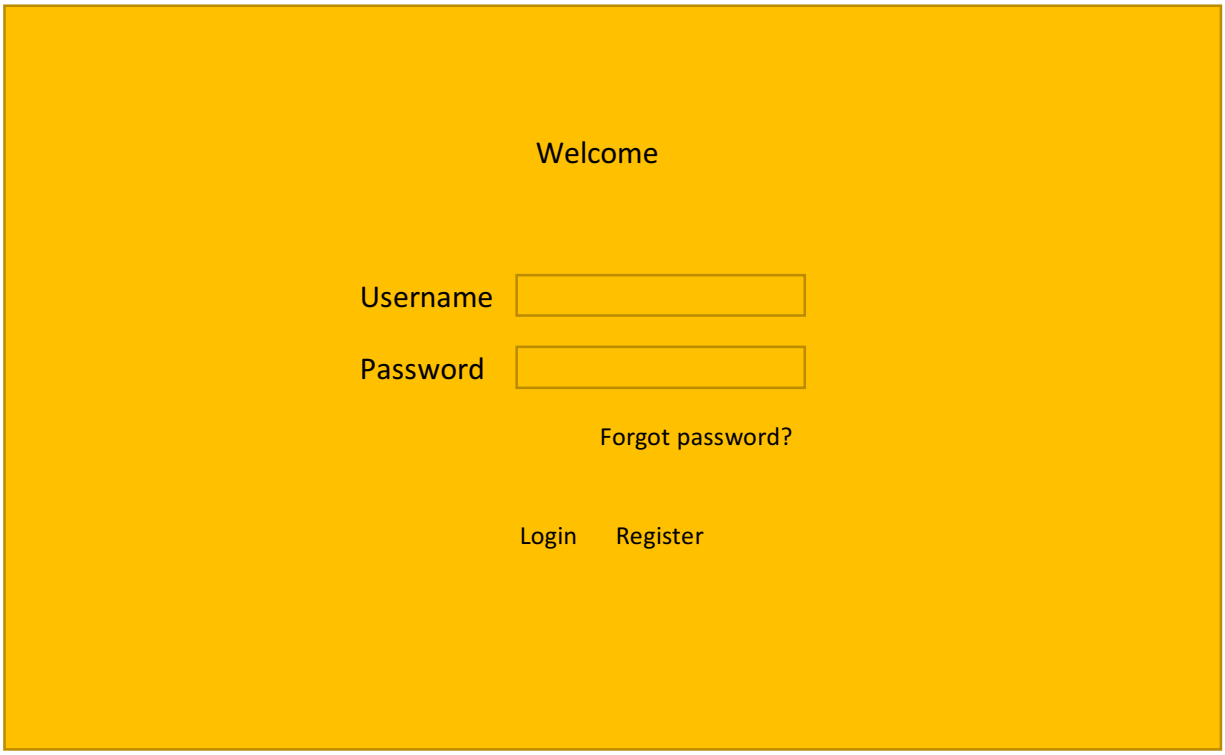
Profile Details

Column	Data type	Description	Example
UserID	Integer	Primary key	01
Forename	String	Users forename	John
Surname	String	Users surname	Doe
Gender	String	Gender of user	AlphaMale
Age	Integer	Date of birth of user	1999/04/08
Weight	Float	Weight of user	69.00 kg
Height	Float	Height of user	177.00 cm
PhysiqueID	Integer	Foreign key for physiqueID	01

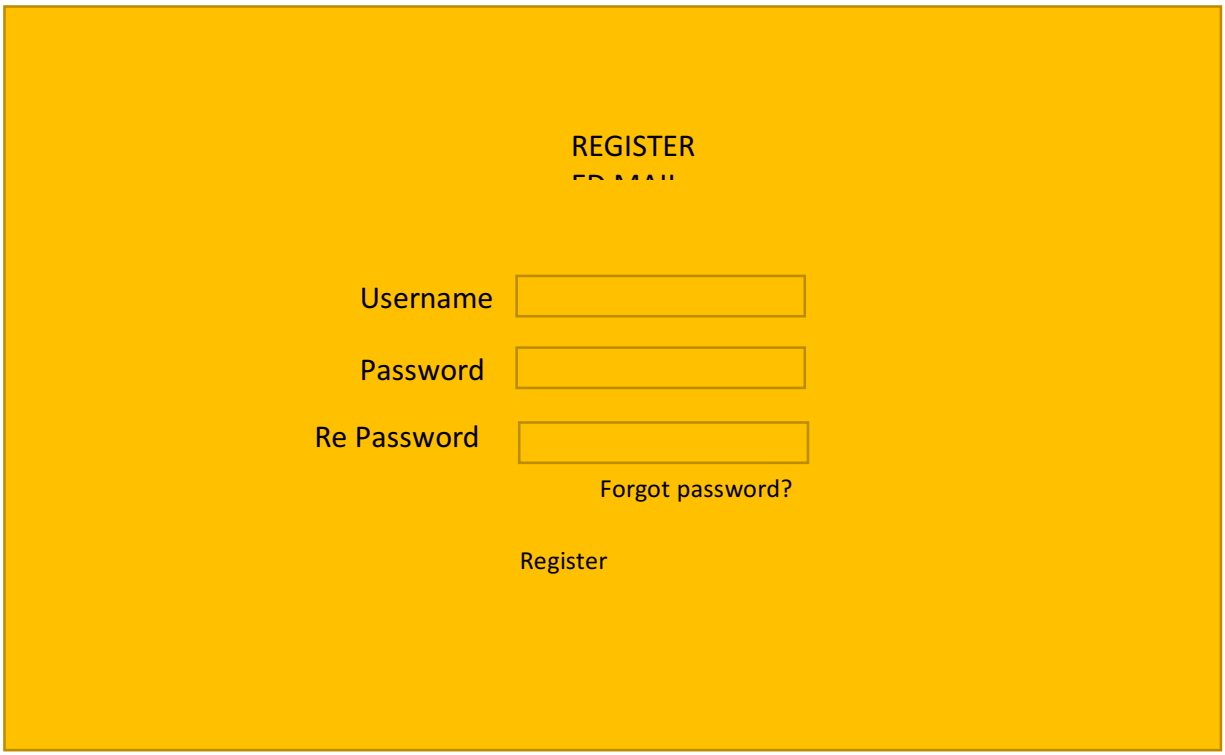
2nd NF due to not completed.

Input screens

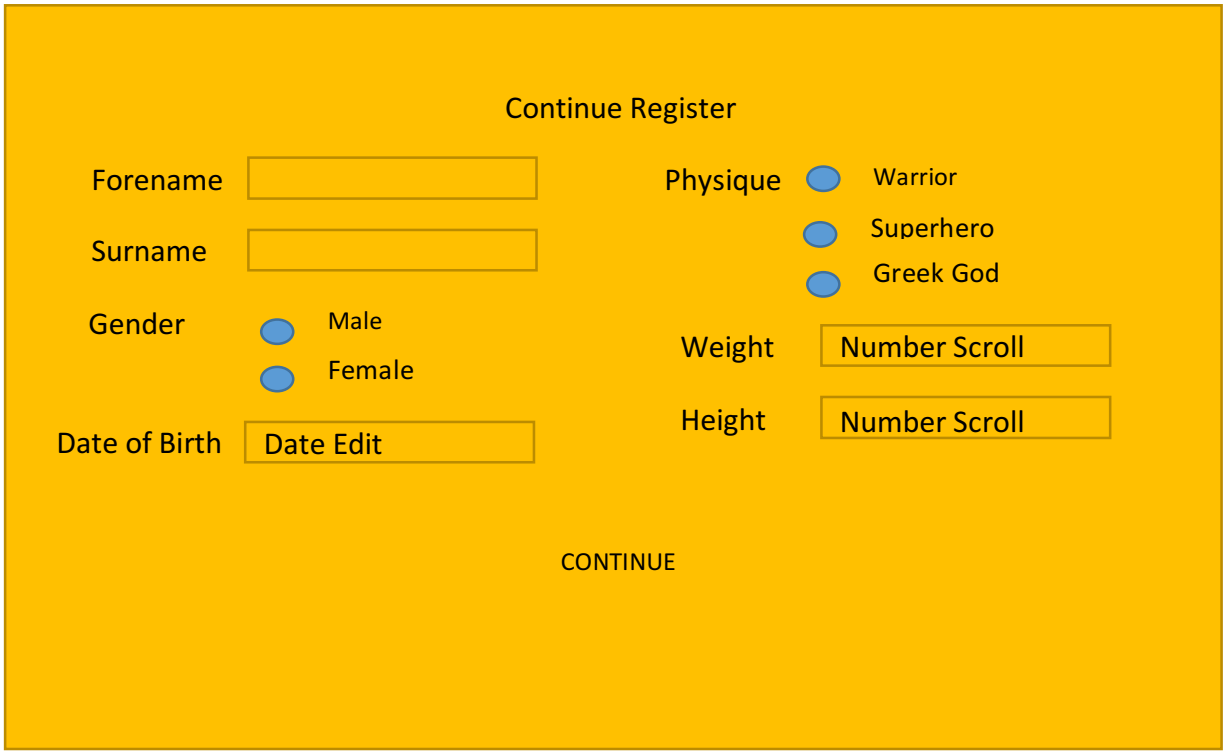
The following GUI screens are coloured different in order to be easily seen for documentation purposes. For the actual GUI windows, this will be a mixture of a white background and pictures to add to aesthetics. The current colour is just to make it more visible.



This is the welcome screen which the user will first see. This has three options, including logging in, register or forgotten password. This is a simple screen which only needs to forms of input. This is only possible with text entry boxes, as shown above.

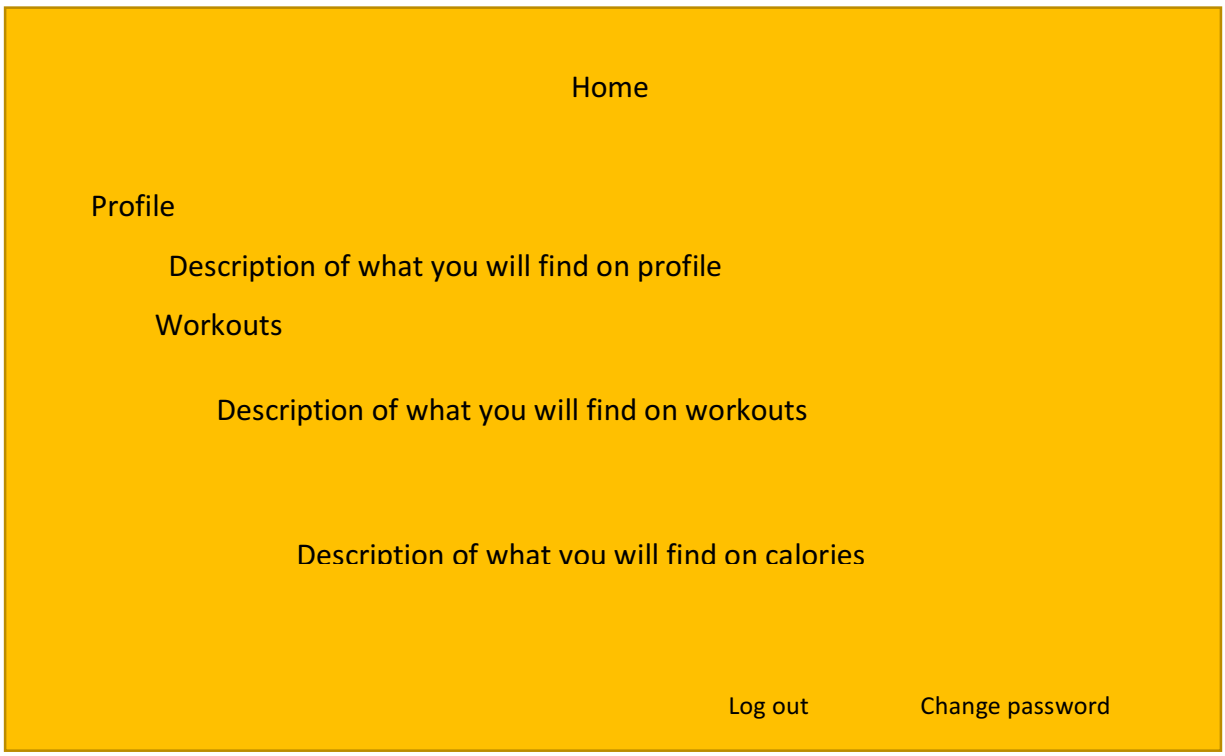


To register we must have the username and password login. This must be separate to the rest of the registration so I can check the username is taken or not. Again with this screen, only forms of entry can be buttons and text input boxes.

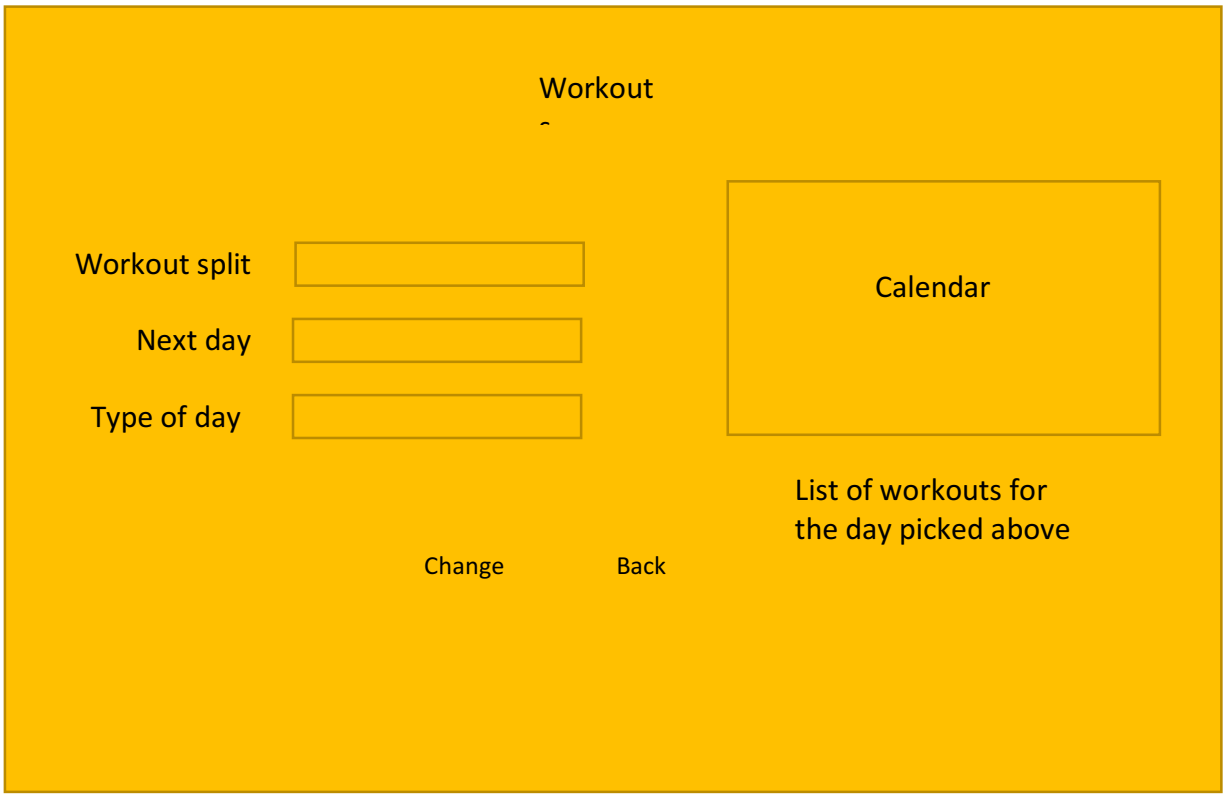


Once the username is allowed, it will go onto the basic details for the calculations needed later. This screen is a little more complex, and so, to reduce any unneeded information for further validation. Gender will be limited to 2 options by clicking on either one. Date will be limited to pick a date by scrolling in the correct date format. The physique will be limited to 3 options too. Finally weight and height will be a scroll wheel to determine an accurate number.

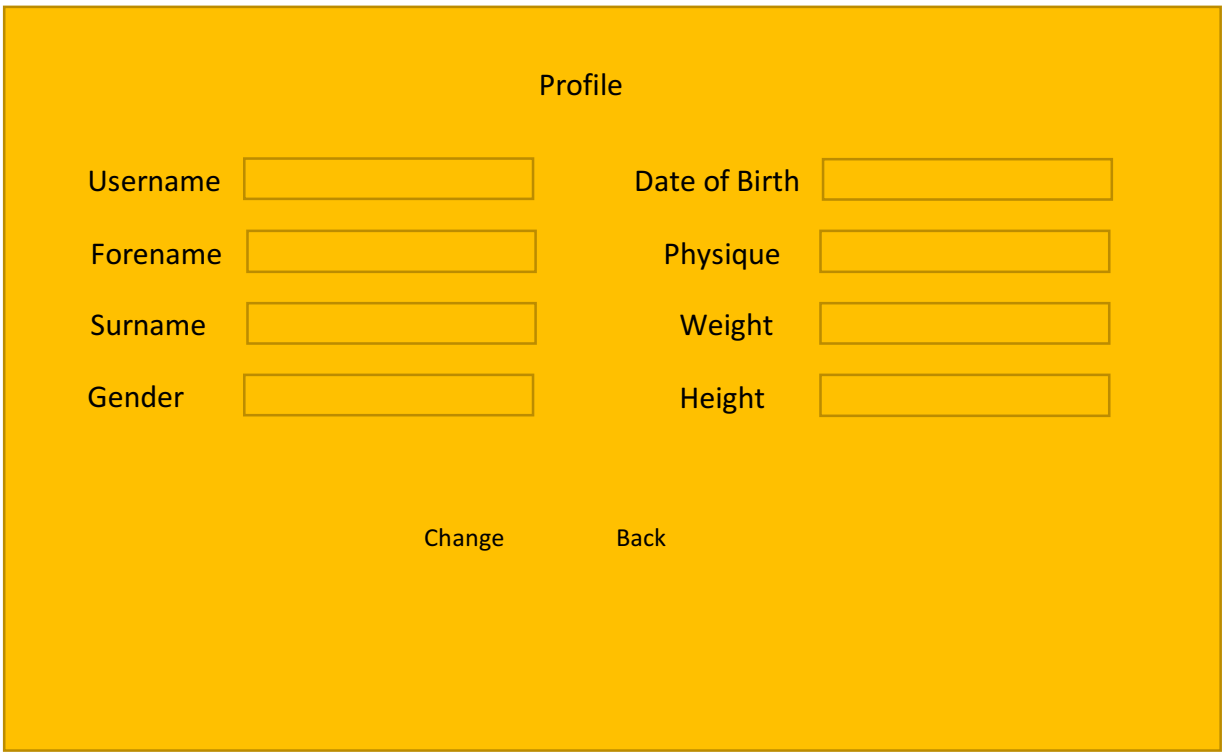
This shows all the calorie needs and nutritional information for the user. This is essential to show the user the basics of heir body and knowledge is key to show them. This screen mainly provides the information and does not need input boxes. This will just be view boxes to output information to the users.



This is the home screen that the user will see once they have logged in. This has buttons to navigate around the program, this includes logging out, changing password and viewing profile, workouts and calorie details. This only has options for the users to pick. This will only require buttons for the users to click.



This shows and updates details about the workouts planned for you in the future. This will have to include a calendar widget to output data on a picked day. The rest of the outputs will be output boxes showing text information.



This is the profile screen which allows users to check and update their profile details. Unlike the registration screen, not all of the sections have to have inputs and only need to show the information that the user inputted. However, a few of these must be allowed for re-entry. Such as physique, weight and height

Variables and inputs needed

Variables	Function	Data Type	Sample Data	Validation rule
AmountGreekGod_Int	This is to show to the Admin, all of the stats of the users who choose the physique, to see which is most popular.	Integer	4 (users)	Ensure variable is only an integer. Not minus figure. Not more than Amount of users.
AmountSuperhero_Int	This is to show to the Admin, all of the stats of the users who choose the physique, to see which is most popular.	Integer	4 (users)	Ensure variable is only an integer. Not minus figure. Not more than Amount of users.
AmountWarrior_Int	This is to show to the Admin, all of the stats of the users who choose the physique, to see which is most popular.	Integer	4 (users)	Ensure variable is only an integer. Not minus figure. Not more than Amount of users.
AmountUsers_Int	This will be used for statistics for the Admin to see	Integer	30 (users)	Ensure variable is only an integer. Not minus figure.
Counter_Int	This is a counter which will help to iterate through lists in my program	Integer	2	Ensure variable is only an integer. Not minus figure. Not longer than list being iterated through
Username_Str	This will temporarily store the username of the current user that is logged in. This is quicker and easier than importing from a database all the time, which is needed throughout the program.	String	Kiran667	Ensure variable is more that 2 characters and no more than 14 and no special characters. Not already picked.

UsernameInput_str	This is the variable set to the username that the user inputs text boxes when signing up	String	Kiran667	Ensure variable is more that 2 characters and no more than 14 and no special characters. Not already picked.
PasswordInput_str	This is the variable set to the password that the user inputs text boxes when signing up	String	***** humps	Ensure variable is more that 2 characters and no more than 14 and no special characters.
EmailAddressInput_str	This is the variable set to the email address that the user inputs text boxes when signing up	String	Kiran675@ gmail.com	Ensure variable is no longer than 24 characters and no less that 7. Ensure it contains '@', followed by one or two '.' No other characters after @ sign. No prohibited characters.
GenderInput_str	This is the variable set to the gender that the user inputs text boxes when signing up	String	Male	Ensure it is string, no numbers and only Male or Female.
WeightInput_int	This is the variable set to the weight that the user inputs text boxes when signing up	Integer	74.00(kg)	Ensure it is only an integer. Number is between 40 and 200.
HeightInput_int	This is the variable set to the height that the user inputs text boxes when signing up	Integer	177.00cm(kg)	Ensure it is only an integer. Number is between 120 and 250.
State_Bool	This is used for validation purposes. It's set to False, but when conditions are met it is set to True.	Boolean	True	Only True or false

UserDOB_int	This is the variable set to the DOB that the user inputs text boxes when signing up	Integer	1999/04/08	In the correct format YYYY/MM/DD. Is not higher than todays date subtracted 17 years. No further than date subtracted 80.
TodayDate_int	This variable is set to todays date	Integer	2017/03/27	Ensure it is correct to todays date and in correct format YYYY/MM/DD
projectedCals_int	This will set this variable to the projected cals after calculations are applied	Integer	2045(kcals)	Ensure that it is all integers (apart from kcals which is added after). Ensure that it is not like than 1500kcals.
Lifestyle_str	This will set to the option picked by the user	String	Active	Ensure that the only input is Active, Very Active or Sedentary. No integers. No more than one option.
Goal_str	This will set to the option picked by the user	String	Cut	Ensure that the only input is Bulk, Cut or Maintain. No integers. No more than one option.
GoalSpeed_str	This will set to the option picked by the user	Integer	Aggressive	Ensure that the only input is Slow, Aggressive or Balanced. No integers. No more than one option.
BMI_int	This will calculate BMI from entered information, set this variable and save to database.	Integer	23	Ensure it is only integers. Not more than 50, no less than 15.
Lists				
Usernames	List of usernames to cross-reference for validation to check if the usernames are already taken.	String inputs only	Kiran667	Each individual input is more than 2 characters and no more than 14 and no special characters. Not already picked.
Profile Details	A list of all the relevant information	Various data types	(Kiran667,1999/04/08, Male, Cut, Aggressive)	

	needed when importing from			
GUI Widgets				
Calendar	Needed to give input to allow the program to output			
LineEdit	To allow users to input data			
TableView	Allows a table to be inputted and edited			
TextEdit	To show the user any notes and allows them to change it and print it off			
Button	Allows users to navigate and input data			
ComboBox	Offer limited information of choice for users to pick			
RadioButton	Offer limited information of choice for users to pick			
Dropdown selection	Offer limited information of choice for users to pick			
DateEdit	Allows users to use calendar to pick their date of birth.			

Validation needed

Field	What it does?	What the user inputs	Validation
Login			
Username	This allows the user to input their chosen username to enter their profile in the game. Checks	The user inputs the username that they chose	This validates the length of entry entered (less than 4 and more than 20), and checks if there is a

	the database for a corresponding username.	when creating their account.	username in the database which corresponds to entry.
Password	This verifies that the user inputting the username is the actual user by having to input their password. Checks the database username for a corresponding password and checks this.	The user inputs the password that they chose when creating their account, corresponding to their username.	Validates length of entry entered (less than 4 and more than 20), and checks if the password entered corresponds the linked password to the username entered previously.
Register Screen			
Email Address	When signing up, it is necessary to have an email address to validate the user and link accounts. This is added to the database which links with account, only one email address allowed.	The user inputs their valid email address.	Must be in the following format (account@domain.(com/co.uk). If the following format is not followed, the program will not allow entry. One email address allowed in the program. Checks to see if the email address is already in use.
Username	Adds a username to a table on the database to be the primary key. This is the indicator for the profile.	The user will input a desired username that they would like to be known as across the program.	This will validate the length of the username (4<20) and without any prohibited characters. This will also check the database to check if it currently exists, therefore not unique.
Password	Adds a password to the database to correspond with the profile to keep it secure.	The user will input a desired password that they can use and remember to access and keep their profile safe.	This will check the entered password for its length (4<20) to ensure security and non spam account as well as prohibited characters.
Re-enter Password	This is to check and ensure the password entered previously was entered correctly and to the users preference (no mistype)	The user enters the password that they would like to use again.	This will check if it matches the password entered in the previous input textbox.
Continue Register screen			
Forename	This will add the forename information into the database of the user for future use.	The user will enter their first name.	Checks if any integers or special characters were inputted, as well as size limits (1<20)
Surname	This will add the surname information into the database of the user for future use.	The user will enter their second name.	Checks if any integers or special characters were inputted, as well as size limits (1<20)
Male	Enters male gender into database for further user	The user will check radio box if they are male	Checks if either Male or Female radio box is entered, else will bring up error.

Female	Enters female gender into database for further user	The user will check radio box if they are female	Checks if either Male or Female radio box is entered, else will bring up error.
Date of birth	Enters the date of birth of the user into the database.	Uses the date box to enter their date of birth in YYYY/M/DD format.	Checks if the user is above 16 years old, by taking current date and subtracting inputted date. Checks if user is above 90 (not recommended)
Weight	Retrieves data about weight from the user as a reference for later use to work out basic data.	Uses the up and down buttons to enter their weight (kg) into the program.	This will not allow the user to enter a weight below 30kg and more than 200kg. This will only allow integers to be inputted.
Height	Retrieves data about height from the user as a reference for later use to work out basic data.	Uses the up and down buttons to enter their height (cm) into the program.	This will not allow the user to enter a height below 120cm and more than 250cm. This will only allow integers to be inputted.
Greek God	Inputs that the user would like to look like a Greek god into the database. This will be used to show dedicated workouts to follow in order to achieve this.	The user will check the radio box if the user would like this choice out of the other two.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Superhero	Inputs that the user would like to look like a Superhero into the database. This will be used to show dedicated workouts to follow in order to achieve this.	The user will check the radio box if the user would like this choice out of the other two.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Warrior	Inputs that the user would like to look like a Warrior into the database. This will be used to show dedicated workouts to follow in order to achieve this.	The user will check the radio box if the user would like this choice out of the other two.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Nutrition Guide			
Sedentary	Enters a sedentary lifestyle into the database for this user. This will personalize their recommended intake.	Checks this radio box if they think they have a sedentary lifestyle.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Moderately Active	Enters a moderately active lifestyle into the database for this user. This will personalize their recommended intake.	Checks this radio box if they think they have a moderately active lifestyle.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Highly Active	Enters a highly active lifestyle into the database for this user. This will personalize their recommended intake.	Checks this radio box if they think they have a highly active lifestyle.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.

Cut	Enters into the database that this user would like to lose weight and needs a personalized workout for this.	Checks this radio box if they think they would like to lose weight.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Bulk	Enters into the database that this user would like to build muscle and weight and needs a personalized workout for this.	Checks this radio box if they think they would like to build muscle.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Maintain	Enters into the database that this user would like to just increase athleticism and needs a personalized workout for this.	Checks this radio box if they think they would like to maintain weight.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Aggressive	Enters into the database that this user would like to achieve their goals quickly.	Checks this radio box if it meets their preferences.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Balanced	Enters into the database that they would like to achieve their goals at a balanced rate.	Checks this radio box if it meets their preferences.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Slow	Enters into the database that they would like to achieve their goals at a slow and steady rate.	Checks this radio box if it meets their preferences.	This has been put into a group so that only one of the three radio boxes can be chosen. Also checks if none of the radio boxes have been checked and produces an error.
Change Password			
Username	This will check that the user is the intended the user by verifying their username and password before allowing them to change their password.	The user will enter the username that they logged into the program with.	This will validate the length of the username (4<20) and without any prohibited characters. This will also check if it is the same username as the account that is currently logged in with.
Password	This will check that the user is the intended the user by verifying their password that is linked to the username that they entered previously.	The user will enter the password that is linked to the username that they entered previously and are currently logged in with.	This will check the entered password for its length (4<20) to ensure security and non spam account as well as prohibited characters. Will check if it matches the password in the database linked with the account.

Test plan

Test no	Testing for	What is it?	How it will be tested?	Test data	Expected output	When will it be tested for?
1	Navigation of all windows	Checking if all the windows, as a part of my GUI, are all linked and coherent. Buttons navigate to the correct screen and allow users to go back etc.	This will be tested by allowing Olek and myself to press every button on each of the screens. I will have a plan of the windows showing where each should be connected to. If these buttons take the user to the correct window, it will be checked off the list showing that it works.		All the windows will be checked off list that ensures all the windows are connected. None of the buttons will be dead, i.e, don't not have a specific function. All buttons connect to the correct window and function.	This will be tested throughout the coding stage, while each of the sections are being completed. So that it is coherent throughout.
2	Check if navigating windows crashes	To check that when navigating through windows, there are no crashes, i.e clicking too fast	This will be tested by allowing Olek, and myself to spam all of the navigation buttons and try to break program.		The expected output, is that for the program to have no errors and navigate through all windows.	This will be tested throughout the coding stage, while each of the sections are being completed. So that it is coherent throughout.
3	User must register to an account	For the user to access their account, they first have to sign up, otherwise they can not enter the program	This will be tested by allowing a new user to the program, to try and gain access. They will try logging in with details.	The user will enter username Test Account and password Test. This is their preferred details.	The outcome of this, assuming the user has not already set up an account or an already existing account with the username should show an error with username/password not valid.	This will have to be tested during the beginning phases of production, before completion. This is the first line of entry to the program and so must be completed and tested first.
4	User must log on account with	Once the user can't gain access from above, the	This will be tested by inputting the same test data	The user will enter username Test Account	The outcome of this, should be access to the profile. With the	This will have to be tested once the program is completed

	provided username and password	user must set up an account with their data. This should then allow them access.	from previous test to ensure user now has access to the program after registering.	and password Test. This is their preferred details.	correct details entered.	through registration screens, and login screen and database, just to gain access
5	Allow user to pick the days that they wish to workout on	Once the user has set up their profile, they will have to navigate through, to fill in their personalized data.	This will be tested by inputting three days to work out on. This will then be checked after logging out, then back in again.	To test this, Monday, Wednesday and Friday will be the days that are picked.	This should then show the workouts that I have for these days in order when I click them on the calendar.	This will have to be the last section of tests to be completed. The calendar and workouts are the last things to be completed in my program, and needs a fully working GUI, database and profiles.
6	Provide admin screen to maintain	An admin screen is necessary to a program to allow control over users instead of needing a programmer to hard code it.	This will be checked by logging into regular accounts in the same login screen, then inputting an admin username and password	The username Admin and password Admin, will be the login credentials for Admin user.	When the user enters normal account usernames and passwords, it should enter to their own individual profiles. When Admin credentials are inputted	This can be tested during the initial stages of coding. This can be set as soon as the login screen has been completed and main users have a working platform to run on.
7	When user requests to change password, make sure password hash is the same as db hash	This is to ensure users have a safe and secure account throughout their profile. This is not known by the user but protects them	This will be tested by trying to print the password and read from the database which it is saved to, whenever a password is required.	This will be test by inputting password password. Every time this is needed, I will try to output the password to expect the return in hash. I will	The expected output, should be that every password input should return with a hashed version. The only person that should see the original password is the user. In any other case this should return the hash.	This should be tested in the beginning of the products life development. This is one of the first things that must be coded for, so that it is secure in later stages.

				check the database fort he hash.		
8	Ensure all windows have back buttons for user to navigate	This is to ensure that users do not have a dead end, if they click the wrong button	I, along with Olek, will navigate through every screen to try and find a dead end, to see if it is possible to not be stuck	Press all buttons	There should not be any dead ends. For every screen there is a returning button to either the previous screen or the home screen	This will have to be tested when the program is completed so that all the final GUI's are set up.
9	Ensure admin can access admin screen	An admin screen is necessary to a program to allow control over users instead of needing a programmer to hard code it.	This will be checked by logging into regular accounts in the same login screen, then inputting an admin username and password	The username Admin and password Admin, will be the login credentials for Admin user.	When the user enters normal account usernames and passwords, it should enter to their own individual profiles. When Admin credentials are inputted	This can be tested during the initial stages of coding. This can be set as soon as the login screen has been completed and main users have a working platform to run on.
10	Have access to reclaim password if forgotten	If the user has forgotten their password and has no entry, they should be able to reclaim their account without having	This will be tested by trying to reset the password and trying to regain access.	Try to enter an invalid password, then request a password change using security question and answer.	When the user requests password change, when entering the correct password to the security question, should reset their password with a code, which is sent to their email address which is set up with their email.	This will have to be tested once the program is complete, as this is one of the latter objectives to be completed. This will need the database, email checker and return email all generated before testing this.
11	Ensure username is no longer than 12 characters	This is to ensure that no spam accounts are created, this will be hard	When registering an account, I will try to input a username that is 13+ characters long.	I will enter the username Iamnotaspam Account which is 18 characters long	When I try to enter this username, the program should return an error and not accept the username.	This can be done in early stages of development as this is one of the first objectives that need to be completed.

		for the user to remember.				
12	Ensure username is longer than 4 characters	This is to ensure that no spam accounts are created.	When registering an account, I will try to input a username that is less than 4 characters long.	I will enter the username hi Account which is 2 characters long	When I try to enter this username, the program should return an error and not accept the username.	This can be done in early stages of development as this is one of the first objectives that need to be completed.
13	Ensure username does not contain prohibited characters	This is to ensure that no spam accounts are created.	To test this, I will enter a username with some prohibited characters	To test this, I will enter a username with the following characters ‘%,^	The program should return an error showing that the input contain prohibited characters	This can be done in early stages of development as this is one of the first objectives that need to be completed.
14	Ensure forename is no longer than 15 characters	This is to ensure that no spam accounts are created	When registering an account, I will try to input a forename that is 15+ characters long.	I will enter the name Iamnotaspam Account which is 18 characters long	When I try to enter this username, the program should return an error and not accept the username.	This can be done in early stages of development as this is one of the first objectives that need to be completed.
15	Ensure forename is longer than 2 characters	This is to ensure that no spam accounts are created.	When registering an account, I will try to input a username that is less than 4 characters long.	I will enter the name hi Account which is 2 characters long	When I try to enter this username, the program should return an error and not accept the username.	This can be done in early stages of development as this is one of the first objectives that need to be completed.
16	Ensure forename does not contain prohibited characters	This is to ensure that no spam accounts are created.	To test this, I will enter a username with some prohibited characters	To test this, I will enter a username with the following characters ‘%,^	The program should return an error showing that the input contain prohibited characters	This can be done in early stages of development as this is one of the first objectives that need to be completed.
17	Ensure surname is no longer than 20 characters	This is to ensure that no spam accounts are created	When registering an account, I will try to input a username that is 13+ characters long.	I will enter the username Iamnotaspam Account which is 18 characters long	When I try to enter this username, the program should return an error and not accept the name.	This can be done in early stages of development as this is one of the first objectives that need to be completed.

18	Ensure surname is longer than 4 characters	This is to ensure that no spam accounts are created.	When registering an account, I will try to input a username that is less than 4 characters long.	I will enter the name hi Account which is 2 characters long	When I try to enter this username, the program should return an error and not accept the username.	This can be done in early stages of development as this is one of the first objectives that need to be completed.
19	Ensure surname does not contain prohibited characters	This is to ensure that no spam accounts are created.	To test this, I will enter a surname with some prohibited characters	To test this, I will enter a surname with the following characters '%,^	The program should return an error showing that the input contain prohibited characters	This can be done in early stages of development as this is one of the first objectives that need to be completed.
20	Ensure password is no longer than 20 characters	This is to ensure that no spam accounts are created and the user will not forget the password	When registering an account, I will try to input a username that is 20+ characters long.	I will enter the username Iamnotaspam Accounthere which is 20 characters long	When I try to enter this username, the program should return an error and not accept the name.	This can be done in early stages of development as this is one of the first objectives that need to be completed.
21	Ensure password is longer than 4 characters	This is to ensure that no spam accounts are created and ensure account safety	When registering an account, I will try to input a username that is less than 4 characters long.	I will enter the name hi Account which is 2 characters long	When I try to enter this username, the program should return an error and not accept the username.	This can be done in early stages of development as this is one of the first objectives that need to be completed.
22	Ensure password does not contain prohibited characters	This is to ensure that no spam accounts are created.	To test this, I will enter a password with some prohibited characters	To test this, I will enter a password with the following characters '%,^	The program should return an error showing that the input contain prohibited characters	This can be done in early stages of development as this is one of the first objectives that need to be completed.
23	Ensure password and re-enter password is the same	This is to ensure that the user did not accidently spell their password wrong and is sure	I will type in two different passwords to see if the program accepts the input	I will enter password hello1 and then re-enter password as hello2	This should not accept this input and ask the user to re-enter their password until they match	This can be done in early stages of development as this is one of the first objectives that need to be completed.

24	Ensure that email address follows a valid format (must contain '@', followed by '.')	This is to ensure the email used is legitimate and not spam	The email address used is needed in the program and is vital, and so can not be a fake email. Used to gain forgotten password.	I will enter an email address in the wrong format 'kdippy@.gmail. Co.uk.'	This should not accept the following email address and return an error until the user inputs a valid email address	This can be done in early stages of development as this is one of the first objectives that need to be completed.
25	Ensure age is within limits that is allowed to use the program (16 -80 years old)	This is to keep the users safe, the program is not recommended for people under 16 and over 80.	This will be tested by entering both a date of birth of below 16 and above 80	I will enter date of birth 08/04/2007 and date 08/04/1930	This should not accept these ages and report an error	This can be done in early stages of development as this is one of the first objectives that need to be completed.
26	Ensure height entered is within appropriate level (130cm-230cm)	This is to ensure that no spam accounts are created and prevent people that are not recommended to use this.	When registering, I will enter values outside these limits of 130 and 230cm	I will enter heights of 260cm and 100cm	This should not accept these heights and report an error	This can be done in early stages of development as this is one of the first objectives that need to be completed.
27	Ensure weight entered is within appropriate level (40kg-250kg)	This is to ensure that no spam accounts are created and prevent people that are not recommended to use this.	When registering, I will enter values outside these limits of 40 and 250kg	I will enter heights of 260kg and 30kg	This should not accept these heights and report an error	This can be done in early stages of development as this is one of the first objectives that need to be completed.
28	Ensure a gender is picked	This is needed for my program as basic details	When completing the registration, I	I will not choose any of the gender options and	This should report and error and prompt the user to pick an option	This can be done in early stages of development as this is one of the

		and to personalize programs for them	will not pick an option for this	leave this blank		first objectives that need to be completed.
29	Ensure a physique type is picked	This is needed for my program as basic details and to personalize programs for them	When completing the registration, I will not pick an option for this	I will not choose any of the physique options and leave this blank	This should report and error and prompt the user to pick an option	This can be done in early stages of development as this is one of the first objectives that need to be completed.
30	Ensure a security question is picked	This is needed for my program to regain access if the password is lost or forgotten.	When completing the registration, I will leave this section empty	I will not include an input in this section	This should report and error and prompt the user to pick an option and write in a security answer.	This can be done in early stages of development as this is one of the first objectives that need to be completed.
31	Ensure security answer is hashed in database	When user registers to the program make sure answer is hashed	This is to ensure users have a safe and secure account throughout their profile. This is not known by the user but protects them	This will be tested by trying to print the security answer and read from the database which it is saved to, whenever a answer is required.	This will be test by inputting password password. Every time this is needed, I will try to output the password to expect the return in hash. I will check the database fort he hash.	The expected output, should be that every security answer input should return with a hashed version. The only person that should see the original answer is the user. In any other case this should return the hash.
32	Ensure the user inputs lifestyle	This is needed for my program as basic details and to personalize programs for them	When completing the registration, I will not pick an option for this	I will not choose any of the lifestyle options and leave this blank	This should report and error and prompt the user to pick an option	This can be done in early stages of development as this is one of the first objectives that need to be completed.
33	Ensure the user inputs goal	This is needed for my program as	When completing the registration, I	I will not choose any of the goal	This should report and error and	This can be done in early stages of development as

		basic details and to personalize programs for them	will not pick an option for this	options and leave this blank	prompt the user to pick an option	this is one of the first objectives that need to be completed.
34	When storing password in db, this must be hashed	When user registers to the program make sure password is hashed	This is to ensure users have a safe and secure account throughout their profile. This is not known by the user but protects them	This will be tested by trying to print the security answer and read from the database which it is saved to, whenever a passed is required.	This will be test by inputting password password. Every time this is needed, I will try to output the password to expect the return in hash. I will check the database fort he hash.	The expected output, should be that every password input should return with a hashed version. The only person that should see the original answer is the user. In any other case this should return the hash.
35	Prevent user inputting more or less days than needed	To view the workouts that users have on specific days, the user will decide their workout days	I will test this by trying to enter less than 3 days and more than 3 days.	I will first enter 2 days 'Monday and Wednesday', I will then enter 'Monday, Wednesday, Friday and Saturday'	This should report an error with amount of days chosen and ask the user to clear and start again	This will have to be done later in the programs life. This is a latter objective and needs a set up database, GUI and platform first.
36	Prevent user entering the same day more than once	This only choses the days the user wants to work out in a given week. It is not suggested to do two workouts in the same day	I will test this by trying to pick the same day twice.	I will try to enter Monday twice in one week.	This should return an error and the user should not be able to pick the same day. This should disappear when it is picked.	his will have to be done later in the programs life. This is a latter objective and needs a set up database, GUI and platform first.
37	Ensure user inputs rate of goal achieved	This is needed for my program as basic details and to	When completing the registration, I will not pick an option for this	I will not choose any of the goal options and	This should report and error and prompt the user to pick an option	This can be done in early stages of development as this is one of the first objectives

		personalize programs for them		leave this blank		that need to be completed.
38	Ensure pie chart calculations are correct	This is to allow the admin to monitor the progress of the app	I will go through the data of amount of users that have picked each of the programs available	The amount of users should be 7 warrior, 7 superhero and 9 for greek god when inputted.	The calculations should be correct and show on a pie chart the correct dimensions on the chart.	This can be done in early stages of development as this is one of the first objectives that need to be completed.
39	Ensure a security answer is entered	This is needed for my program to regain access if the password is lost or forgotten.	When completing the registration, I will leave this section empty	I will not include an input in this section	This should report and error and prompt the user to pick an option and write in a security answer.	This can be done in early stages of development as this is one of the first objectives that need to be completed.
40	Ensure amount of days picked is not below 3	To view the workouts that users have on specific days, the user will decide their workout days	I will test this by trying to enter less than 3 days and more than 3 days.	I will first enter 2 days 'Monday and Wednesday'	This should report an error with amount of days chosen and ask the user to clear and start again	This will have to be done later in the programs life. This is a latter objective and needs a set up database, GUI and platform first.
41	Ensure amount of days picked is not above 3	To view the workouts that users have on specific days, the user will decide their workout days	I will test this by trying to enter more than 3 days.	I will then enter 'Monday, Wednesday, Friday and Saturday'	This should report an error with amount of days chosen and ask the user to clear and start again	This will have to be done later in the programs life. This is a latter objective and needs a set up database, GUI and platform first.
42	Check security answer entered by user matches stored	To ensure the user requesting the reset password is the holder of the account,	I will check the recorded data of the answer and input, first an incorrect answer, then the correct answer	I will first enter Mr.Travi (wrong answer) then Mr.Avroutine (the correct	This should reject Mr.Travi as an answer and then accept Mr.Avroutine	This will have to be done later in the programs life. This is a latter objective and needs a set up database, GUI

	security answer	it is necessary to ask them for an answer		answer) to see if this will allow the answer.		and platform first.
43	If match, send user an email with random code	This is to ensure the correct user accesses their profile	I will check the email registered to the account to see if it is directed to the right email, and then check the code posted too.	I will request the password for account kdippy6 and the password should direct to email kdippy6@gmail.com	Once the correct details have been entered, an email should be sent to kdippy6@gmail.com with the reset code.	This will have to be done later in the programs life. This is a latter objective and needs a set up database, GUI and platform first.
44	Change password in database with hashed random code	When user registers to the program make sure password is hashed	This is to ensure users have a safe and secure account throughout their profile. This is not known by the user but protects them	This will be tested by trying to print the security answer and read from the database which it is saved to, whenever a passed is required.	This will be test by inputting password password. Every time this is needed, I will try to output the password to expect the return in hash. I will check the database fort he hash.	The expected output, should be that every password input should return with a hashed version. The only person that should see the original answer is the user. In any other case this should return the hash.
45	Show all relevant informatio n of all the users	This is for the admin to see all the users in the program. If they need to delete or update details themselves.	This will be tested by viewing the table views when logging on with an Admin account	I will check each of the users that the Admin will be able to see on their screen. This should show basic information like name, age and email address.	This should show that the Admin will be able to see on their screen. This should show basic information like name, age and email address.	This will have to be completed at the end of the program as this is the last objective to be completed
46	Allow admin to alter specific informatio n	This is for the admin to see all the users in the program. If they need to	This will be tested by viewing the table views when logging on	I will check each of the users that the Admin will be able to alter information	This should show that the Admin will be able to see on their screen. This should show basic information	This will have to be completed at the end of the program as this is the last objective to be completed

		delete or update details themselves.	with an Admin account	on their screen. This should be basic information like name, age and email address.	like name, age and email address.	
47	Allow admin to delete users	This is for the admin to delete the users in the program. If they need to delete or update details themselves.	This will be tested by deleting a user when logging on with an Admin account	I will check this by deleting user test from my account. I will then try to log into the test account after logging out.	When trying to re log into the deleted account, this should not be possible	This will have to be completed at the end of the program as this is the last objective to be completed
48	Show on calendar, the specific workout they have on a set day or have rest day	This is a personal planner for the user to have on the program. This is to ensure that they keep track of the workouts they have so they don't miss a day.	This will have to be tested by selecting a certain order in days to ensure the right workouts show under the correct day and show off days too.	I will pick days Mon, Wed and Thurs, in that order with workout A picked. This should show on Monday, Wednesday and Friday, I have workouts and the rest should show that they have a rest day.	When I click the days Mon, Wed and Fri, this should show the workouts for that section in order. When I click any other day, it should say I have a rest day.	This will have to be completed at the end of the program as this is the last objective to be completed
49	Show the different workouts available	This is to allow users to pick a workout suited to them best	This will be tested by accessing the workouts page and searching for different workouts to pick.	To test this I will have to sign up with a new account test32, and go to the workouts window to see if different	This should show 3 different workout options for the user to pick from depending on what they would like to. This should be personalized depending on the	This will have to be completed at the end of the program as this is the last objective to be completed

				workouts appear for each of these.	needs the user chose	
50	Calculate BMI with weight and height inputted	This is to allow the user to determine their body type and basic health	This will simply need to be tested with hard coding using pre determined data already calculated and view the response	This will be tested using the height 177cm and weight 74kg. The result should be 23.6 which should be within the normal weight range limit	This should result in the same BMI result of 23.6 and then show on a table that the user is normal and average body weight.	This can be completed in the earlier stages of development as this is a simple calculation without needing the rest of the program to be completed.
51	User can alter and update details including passwords	Once the user has set up their account, some details may have changed and needs updating.	This can be simply tested once the user has registered their account and going to their profile to see if it is possible to update details and to check if it saves.	I will use the account kdippy6 with the name Kiran. I will then try to change the name to Kieran, weight to 60kg, height to 189 and physique of the user to Warrior and then log out and re-login to see if the changes have been made.	The results should show that the details have been updated and even when the user logs out and then logs in again the details should be updated and saved	This will have to be completed at the end of the program as this is one of the last objectives to be completed
52	User must be able to retrieve account if forgotten password	This is to ensure the correct user will be able to regain access to their account	This will be tested by checking if the emailed code is the correct code that will be able to regain access to the linked account.	I will use the account kdippy6 to test this. When I enter the details to get the email, I will check the linked	The code will be random, but should be the same that the password is reset to and that is sent through email	This will have to be completed at the end of the program as this is the last objective to be completed

				email – being kdippy6@gmail.com. I will then enter that code when trying to login, to see if I can gain access to this		
53	Must be able to send retrieval password to linked email	This is to ensure the correct user will be able to regain access to their account	This will be tested by checking the email linked to the account when the email should be sent, to ensure that it is being received.	I will email the kdippy6@gmail.com account to check if the email is being sent to the correct email and to ensure that the interface is working. I will then check if the correct text is being sent, which should have a code and reason as to why the email is sent.	The email should be sent to the correct email (kdippy6@gmail.com) also revealing the correct message, including header and subject too.	This can be completed in the earlier stages of development
54	Show a pie chart of amount of users picking different physiques	This is so that the admin can monitor the people that are using the program and which are most popular	I will go through the data of amount of users that have picked each of the programs available	The amount of users should be 7 warrior, 7 superhero and 9 for greek god when inputted.	The calculations should be correct and show on a pie chart the correct dimensions on the chart.	This can be done in early stages of development as this is one of the first objectives that need to be completed.
55	Calculate each part of the pie chart	This is to draw the pie chart in tk. This will be	I will have predetermined data set up and use these	The calculation is made up of proportions.	The calculations should produce the following result and when	This can be completed in the earlier stages of development

		simple maths to set the proportions.	calculations to ensure that the pie chart is complete.	A total of 23 users, is split into 7,7 and 9. 360/23=13.65 . Thus the sizes of 7,7 is 109.55, then 140.85 is 359.95 rounded up to 360.	constructing the pie chart, should show a full pie chart and accurate results.	
56	Comments for coding throughout	This is for maintenance after the program is completed and for when the clients need to update certain factors.	Once the program is completed, the program should have comments, this should relate to all bits of code so it is understandable.	I, along with another coding developer, will go through the code to ensure that the comments are coherent and relate throughout.	The comments should provide enough information for other coders to understand the basics of it and allow them to change code in the future	This can only be tested once the entire program is complete
57	Have modular code for functions	This is for maintenance after the program is completed and for when the clients need to update certain factors.	Once the program is completed, the program should be split into module, so that it is easier to understand and follow in the future.	I, along with another coding developer, will go through the code to ensure that the modular functions are easy to follow and can easily be changed.	The modules should have enough information for other coders to understand the basics of it and allow them to change code in the future	This can only be tested once the entire program is complete
58	Meaningful variable names	This is for maintenance after the program is completed and for when the clients	Once the program is completed, the program should have comments, this should relate to all bits	I, along with another coding developer, will go through the code to	The variables should all be self explanatory and easy to understand each of their functions.	This can only be tested throughout the coding stage.

		need to update certain factors.	of code so it is understandable.	ensure that the variables are easy to follow and can easily be understood		

Development

Iterative development

One of the first pieces of code that I decided to tackle, which I also mentioned earlier, was the login screen. Naturally, this is the first thing that the user will see and open and so explains why I decided to complete this first. I decided that some basic code, just checking login details were needed just to set up and start.

Below is the initial code, which just checks the basic entry into the program.

```
profiledetails=[[["kdippy6"],["password"]],[["test1"],["testpass"]]]
counter=0
for i in range (len(profiledetails)):
    userinput=input("input username")
    passinput=input("input password")
    if userinput==profiledetails[0][i][0]:
        passwordindex=i
        if passinput==profiledetails[0][1][passwordindex]:
            print("entry authorised")
        else:
            print("incorrect")
```

I also completed tests on this so that it meets the test plan and success criteria. This allows the test no.4 to be complete which is to ensure a user has to login with correct details in order to gain access to the program.

```
input usernamekdippy6
input passwordpassword
entry authorised
```

```
input usernamekdippy6
input passwordincorrectpassword
incorrect
```

Above shows the code working, allowing the user to input kdippy6 username and password – password to gain access, which are the correct credentials. Profiledetails contains the usernames and passwords in a 3D list to gain access. The two accounts are kdippy6 and test 1, with their passwords in the list too. I also tested by trying to enter an incorrect username and password and was rejected appropriately.

I then continued to progress this code by importing in some of the basic code to set up a class and window for the login pages.

This simply sets up the basic windows for the login function.

```
import sys, os
from PyQt4 import QtCore, QtGui, uic
import sqlite3 as lite

loginwin = uic.loadUiType("LoginScreen.ui")[0]

class LoginWindow(QtGui.QMainWindow, loginwin):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.RegisterButton.clicked.connect(self.open_register)
        self.LoginButton.clicked.connect(self.open_home)

    def open_register(self):
        RegWind.show()
        self.hide()
    def open_home(self):
        HomeWind.show()
        self.hide()

app = QtGui.QApplication(sys.argv)
TheFirstWindow = LoginWindow(None)
TheFirstWindow.show()

app.exec_()
```

I then merged the two pieces of code that I currently set up, into the class login. This allowed the program to read from the sql table and return all the usernames and passwords that are stored in it. It will then be stored into a list in python, where the login checker that I previously coded would then check if the username and passwords matched and printed the result.

```
import sys, os
from PyQt4 import QtCore, QtGui, uic
import sqlite3 as lite

loginwin = uic.loadUiType("LoginScreen.ui")[0]

class LoginWindow(QtGui.QMainWindow, loginwin):
  def __init__(self, parent=None):
    QtGui.QMainWindow.__init__(self, parent)
    self.setupUi(self)
    self.RegisterButton.clicked.connect(self.open_register)
    self.LoginButton.clicked.connect(self.open_home)

  def open_register(self):
    RegWind.show()
    self.hide()
  def open_home(self):
    self.username=self.UsernameInput.text()
    self.password=self.PasswordInput.text()
    cur.execute("SELECT username, password FROM profile")
    lists = cur.fetchall()
    profdetails=[]
    profdetails.append(lists)
    for i in range (len(profiledetails)):
     if self.username==profiledetails[0][i][0]:
          passwordindex=i
          if self.password==profiledetails[0][1][passwordindex]:
               print("entry authorised")
          else:
               print("incorrect")


app = QtGui.QApplication(sys.argv)
TheFirstWindow = LoginWindow(None)
TheFirstWindow.show()

app.exec_()
```

Once I reached this stage, it was apparent that I needed to continue the development of the rest of the GUI windows to link on and continue the program. From this stage, I then implemented the rest of the windows to connect the buttons. The following code is the development of the previous code. The only update to this from the last piece is the inclusion of setup code from the pyqt for the other windows as stated. These are connected windows and so, are copied and pasted and altered to depend on the window being loaded.

```
import sys, os
from PyQt4 import QtCore, QtGui, uic
import sqlite3 as lite

con=lite.connect('testerdb.db') #connect sql file
cur=con.cursor() # as above

loginwin = uic.loadUiType("LoginScreen.ui")[0]
registerwin = uic.loadUiType("RegisterScreen.ui")[0]
RegisterContinued = uic.loadUiType("RegisterScreenContinued.ui")[0]
homewin = uic.loadUiType("HomeScreen.ui")[0]
profilescreen = uic.loadUiType("ProfileScreen.ui")[0]
workoutwindow = uic.loadUiType("WorkoutsScreen.ui")[0]
Profilechangingwin = uic.loadUiType("ProfileScreenChange.ui")[0]


class LoginWindow(QtGui.QMainWindow, loginwin):
  def __init__(self, parent=None):
    QtGui.QMainWindow.__init__(self, parent)
    self.setupUi(self)
    self.RegisterButton.clicked.connect(self.open_register)
    self.LoginButton.clicked.connect(self.open_home)

  def open_register(self):
    RegWind.show()
    self.hide()
  def open_home(self):
    self.username=self.UsernameInput.text()
    self.password=self.PasswordInput.text()
    cur.execute("SELECT username, password FROM profile")
    lists = cur.fetchall()
    profdetails=[]
    profdetails.append(lists)
    for i in range (len(profiledetails)):
     if self.username==profiledetails[0][i][0]:
         passwordindex=i
         if self.password==profiledetails[0][1][passwordindex]:
             print("entry authorised")
         else:
             print("incorrect")
    HomeWind.show()
    self.hide()

class RegisterWindow(QtGui.QMainWindow, registerwin):
  def __init__(self, parent=None):
    QtGui.QMainWindow.__init__(self, parent)
    self.setupUi(self)
    self.HomeButton.clicked.connect(self.ReturnLogin)
    self.RegisterButton_2.clicked.connect(self.ContinueRegister)
  def ContinueRegister(self):
    ConRegWin.show()
    self.hide()
  def ReturnLogin(self):
    TheFirstWindow.show()
    self.hide()

class ContinuedRegisterWindow(QtGui.QMainWindow, RegisterContinued):
  def __init__(self, parent=None):
    QtGui.QMainWindow.__init__(self, parent)
    self.setupUi(self)
    self.HomeButton.clicked.connect(self.ReturnLogin)
    self.RegisterButton.clicked.connect(self.RegToHome)
  def ReturnLogin(self):
    TheFirstWindow.show()
    self.hide()
  def RegToHome(self):
    HomeWind.show()
    self.hide()

class HomeWindow(QtGui.QMainWindow, homewin):
  def __init__(self, parent=None):
    QtGui.QMainWindow.__init__(self, parent)
    self.setupUi(self)
    self.ProfileButton.clicked.connect(self.ToProfileScr)
    self.WorkoutsButton.clicked.connect(self.ToWorkoutsScr)
  def ToProfileScr(self):
    ProfScreen.show()
    self.hide()
  def ToWorkoutsScr(self):
    WorksScreen.show()
    self.hide()

class ProfWindow(QtGui.QMainWindow, profilescreen):
  def __init__(self, parent=None):
    QtGui.QMainWindow.__init__(self, parent)
    self.setupUi(self)
    self.ReturnButton.clicked.connect(self.RegToHome)
```

This bit of code links the pyqt windows to the program so that can be called, shown and coded within python.

Login window, which has been explained previously

This is the class setup for an entire window. This contains all of the variables and windows associated with this window. This links buttons and allows functions to be executed when pressed.

This links the button and the function. This specifically, when button pressed, connects to the registration screen.

This links the button to return to home screen

Each of the classes follow the following format of linking buttons and loads up the next or previous window.

```
        self.ChangeButton.clicked.connect(self.ChangeProfile)
    def RegToHome(self):
        HomeWind.show()
        self.hide()
    def ChangeProfile(self):
        ProfChange.show()
        self.hide()

class WorkoutWindow(QtGui.QMainWindow, workoutwindow):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.HomeButton.clicked.connect(self.RegToHome)
    def RegToHome(self):
        HomeWind.show()
        self.hide()

class ProfChangeWin(QtGui.QMainWindow, Profilechangingwin):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.ChangeButton.clicked.connect(self.ToProfileScr)
        self.ReturnButton.clicked.connect(self.ToProfileScr)
    def ToProfileScr(self):
        ProfScreen.show()
        self.hide()

app = QtGui.QApplication(sys.argv)
RegWind = RegisterWindow(None) #THIS IS IMPORTANT
ConRegWin = ContinuedRegisterWindow(None)
HomeWind = HomeWindow(None)
ProfScreen = ProfWindow(None)
ProfChange = ProfChangeWin(None)
WorksScreen = WorkoutWindow(None)
TheFirstWindow = LoginWindow(None)
TheFirstWindow.show()

app.exec_()
```

This is the rest of the code, which links each of the classes, to the windows that have been imported from pyqt.

After this stage was completed and I was happy with the progress of the next few windows being connected, I focused on progressing the login screen. I added a few extra functions to meet the objectives I set earlier. It came clear to me that I needed a forgotten password window, and so I created a new button and function to accommodate this. Furthermore, I needed to import in further information, as I needed a security question and answer for the forgotten password screen. Finally, I included the functionality in order to set up an admin screen and area which in itself needed extra validation and code added to it. The following screenshots show these changes.

```
class LoginWindow(QtGui.QMainWindow, loginwin):
  def __init__(self, parent=None):
    QtGui.QMainWindow.__init__(self, parent)
    self.setupUi(self)
    self.RegisterButton.clicked.connect(self.open_register)
    counter=0
    self.LoginButton.clicked.connect(self.open_home)
    self.ForgottenPassword.clicked.connect(self.GetPassword)
    variablename=self.UsernameInput.text()
```

Sets up buttons and windows to link to each of the functions.

```
  def open_register(self):
    RegWind.show()
    self.hide()
    TheFirstWindow.UsernameInput.clear()
    TheFirstWindow.PasswordInput.clear()
```

Clears inputs from previous user inputs

```
  def open_home(self):
    profdet=[]
    usernames=[]
    passwords=[]
    securityqs=[]
    global usernome
    usernome=self.UsernameInput.text()
    self.password=self.PasswordInput.text()
    cur.execute("SELECT * FROM profile")
    rows = cur.fetchall()
    for row in rows:
      profdet.append(row)

    securityq = [row[15] for row in profdet]
    for z in securityq:
      securityqs.append(z)

    useresults = [row[0] for row in profdet]
    for x in useresults:
      usernames.append(x)

    passresults = [row[1] for row in profdet]
    for y in passresults:
      passwords.append(y)
```

Setting up lists for inputted information from sql

Putting imported sql information into lists

```
def profileuser():
    lenuser=len(usernome)
    lenpas=len(self.password)
    state=False
    counter=0
    print(counter)
    admin=False

    for j in range (len(usernames)):
        if usernames[j]==usernome:
            if usernome==("Admin"):
                admin=True
                print(usernome,admin)


            if passwords[j]==self.password:
                state=True
                cur.execute("SELECT Forename FROM profile where user='%s'"%usernome)
                rows = cur.fetchall()
                global name
                name=(rows[0][0])


    if state==True:
        if admin==True:
            cur.execute("SELECT * FROM profile")
            rows = cur.fetchall()

            AdminScreen.show()
            self.hide()
        else:
            welcome(self)
            HomeWind.label_2.setText("Welcome, %s"%name)
            HomeWind.show()
            self.hide()
    else:
        error(self)

profileuser()

TheFirstWindow.UsernameInput.clear()
TheFirstWindow.PasswordInput.clear()
```

Checks if this is an admin account

Checks if input is an admin account

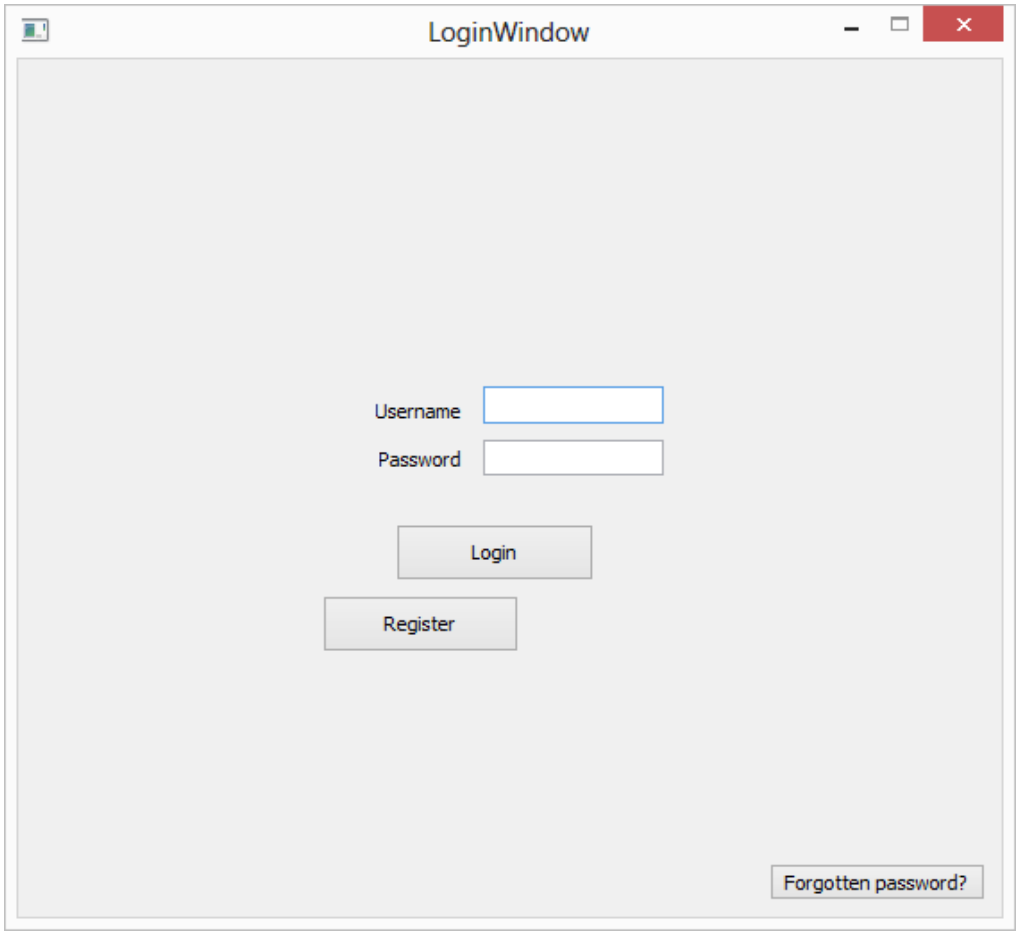
If it is an admin account then it is diverted to the admin screen class

If it isn't then it just continues to a regular account

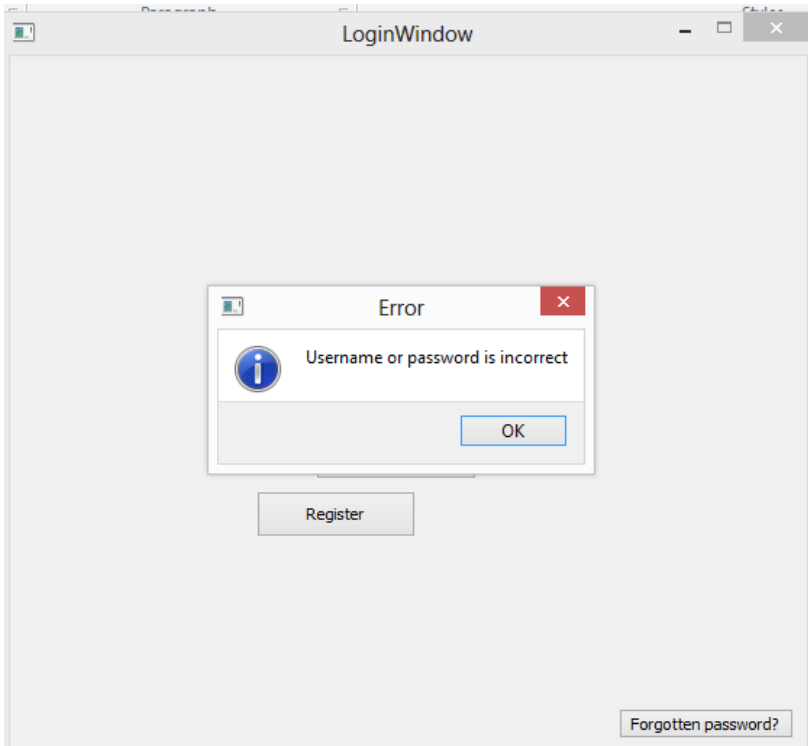
If none of the conditions are met, then the username and or password are incorrect and a self made module

Before I continued with developing more code around my program. I decided to a preliminary test on some of the code that I had completed during this time.

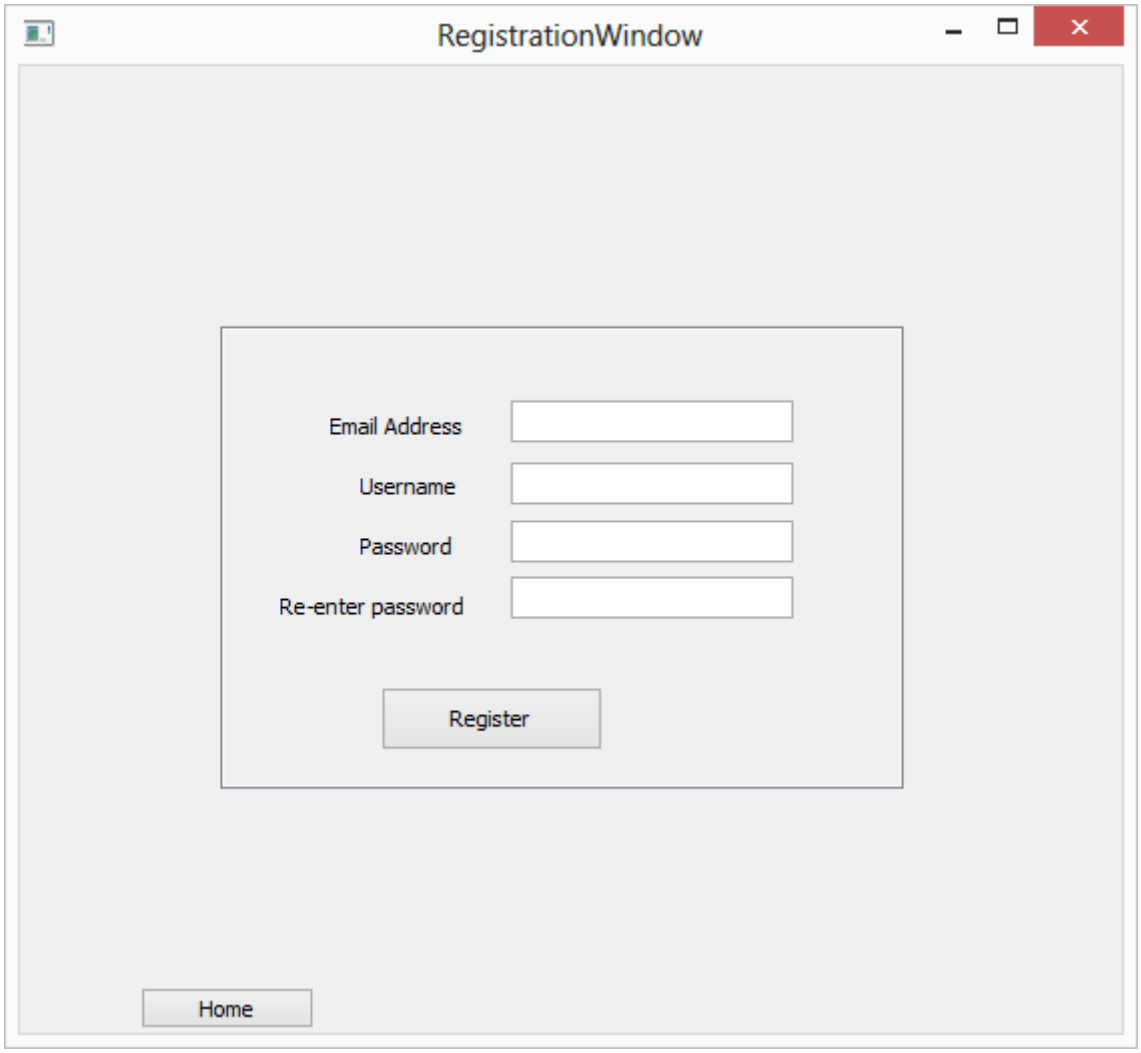
The first tests of which was to ensure that the buttons located on the screen are connected to the correct windows.



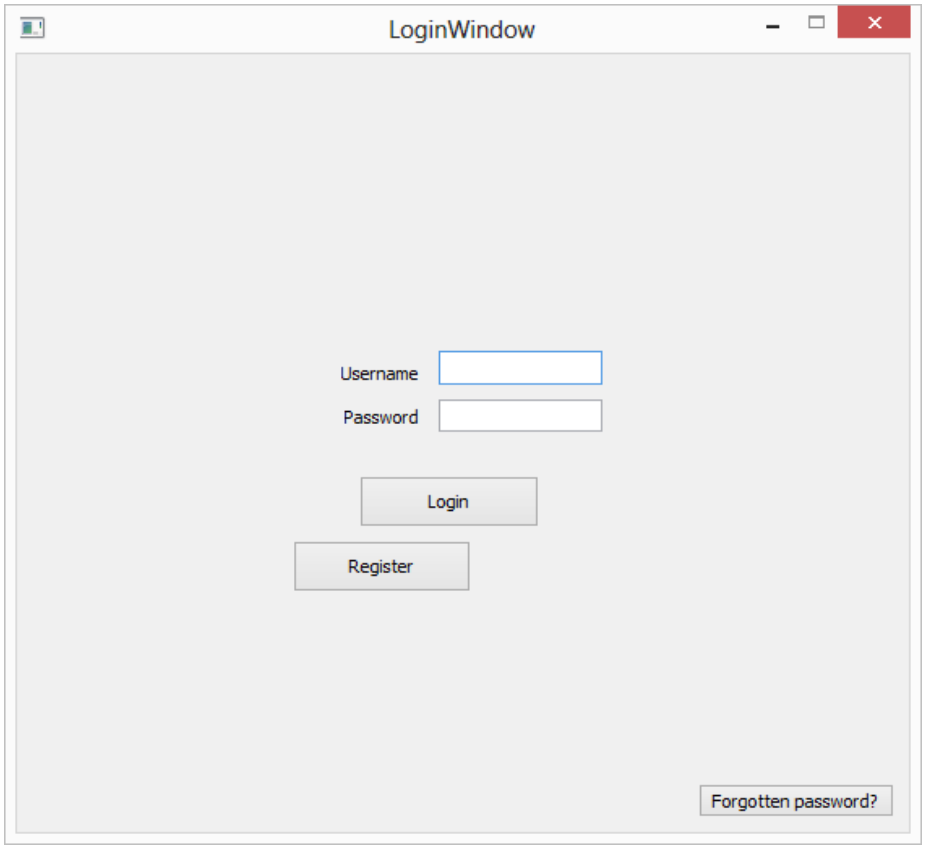
This is the login screen which users see when they first open the program. To start testing, I first pressed the login screen without inputting any data to see where this would navigate the user.



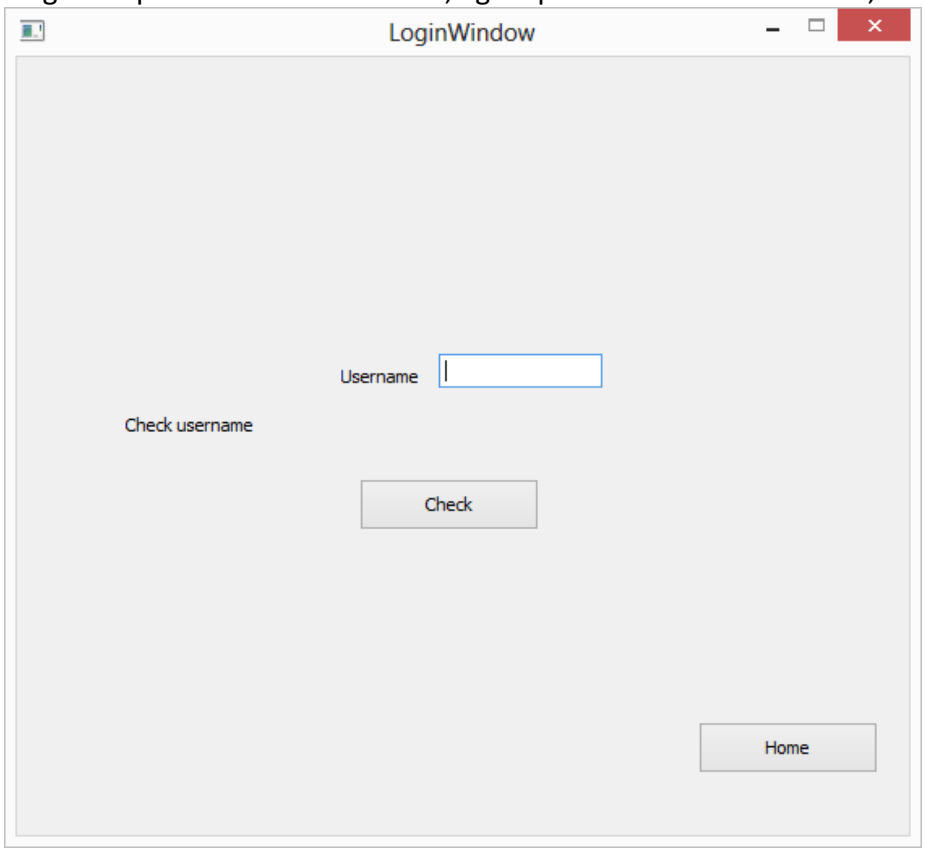
This returned a pop up window showing that they user has entered an invalid username and or password. Considering the user did not enter anything, this is the correct response. The next button to be tested was the register button.



This linked to the register window. This navigated the user to the register screen. This is the correct response to clicking this button. To test the return button, I pressed it, which returned me back to the login window, with all data wiped that was previously on the screen.

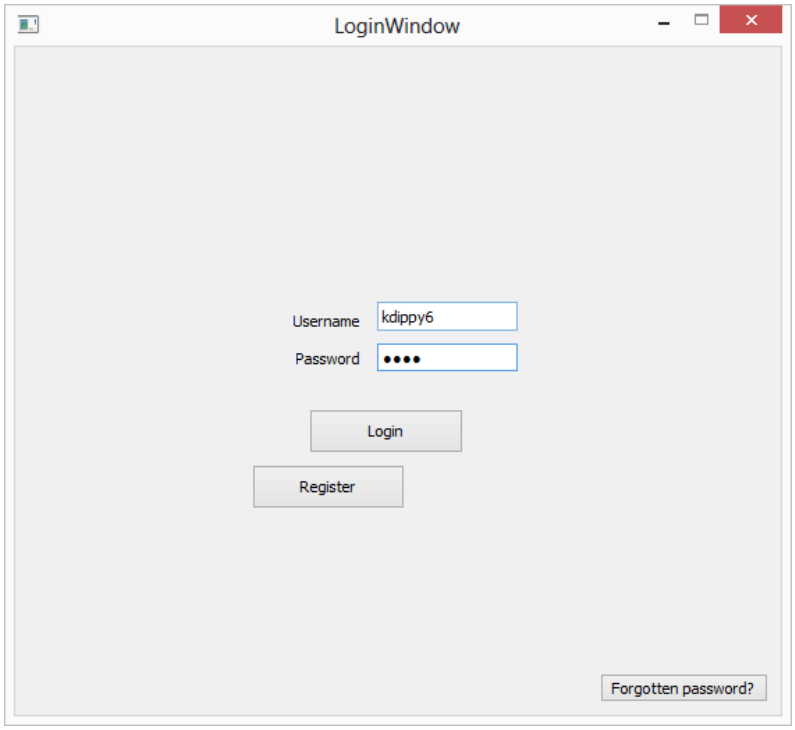


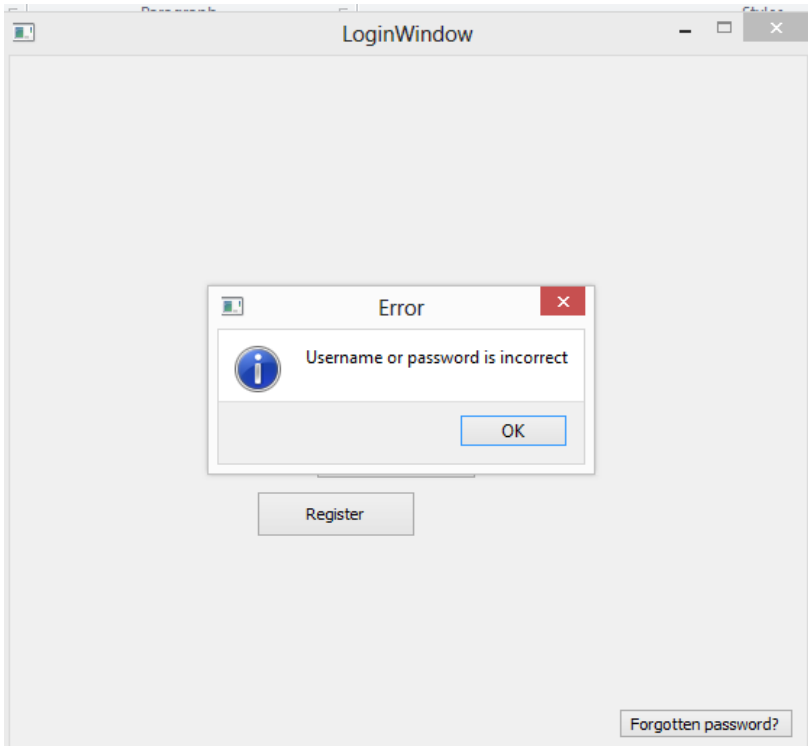
I then decided to press the forgotten password button. This, as expected led to the forgotten password screen. I then, again pressed the home button, which returned home.



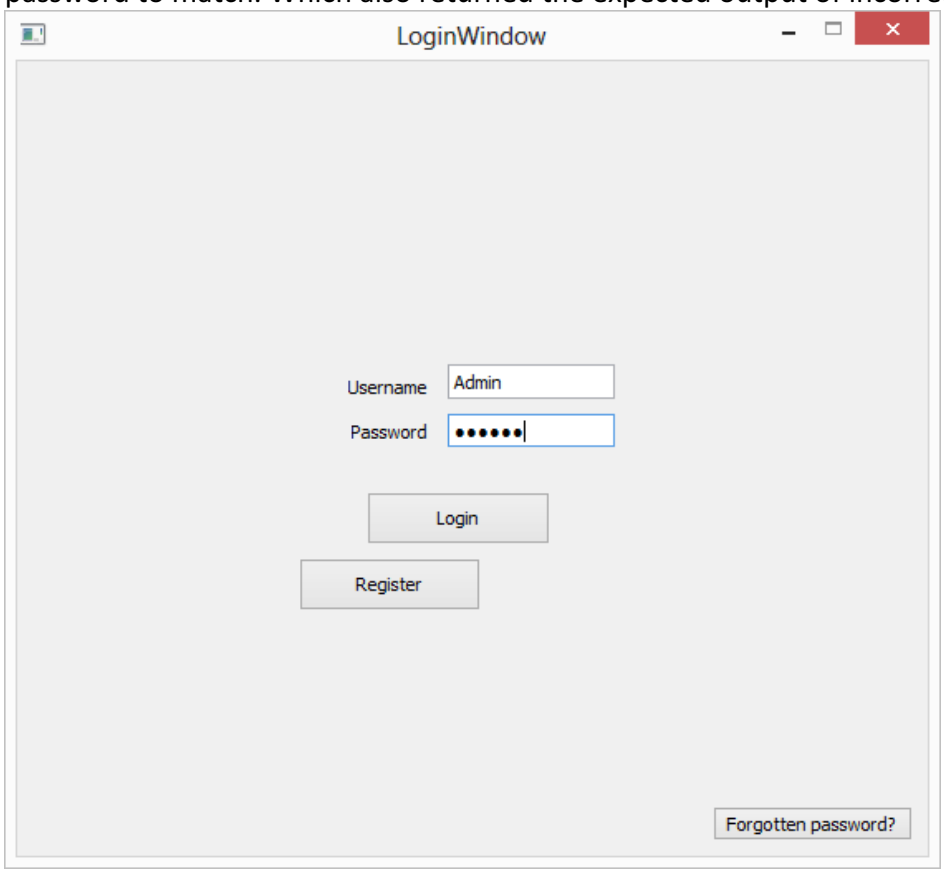
Next, I inputted an invalid username and password, to see the response to this. As expected, it returned to the user that they inputted incorrect details.

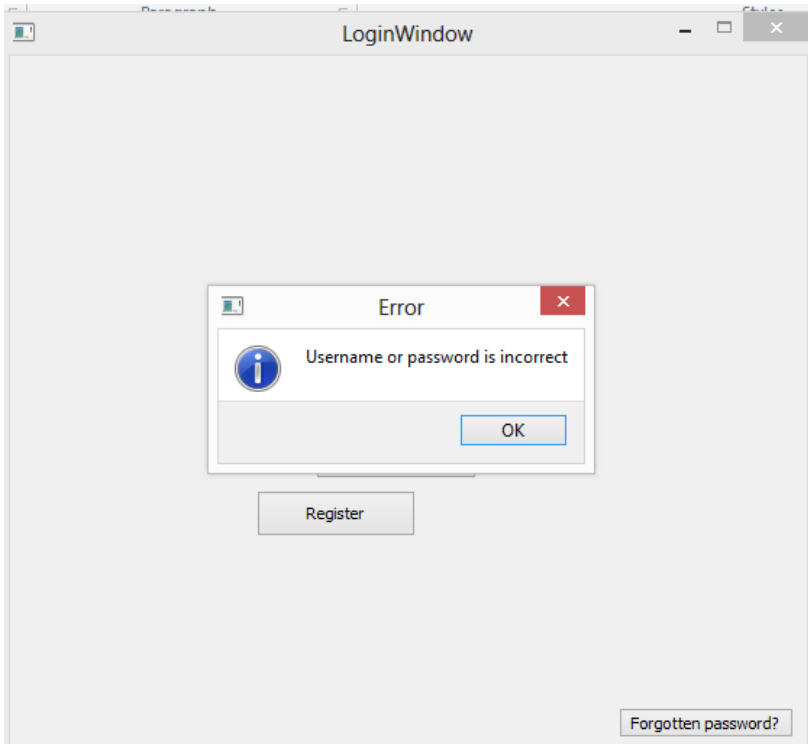
I then inputted a correct username with an incorrect password, which, as expected returned incorrect details.



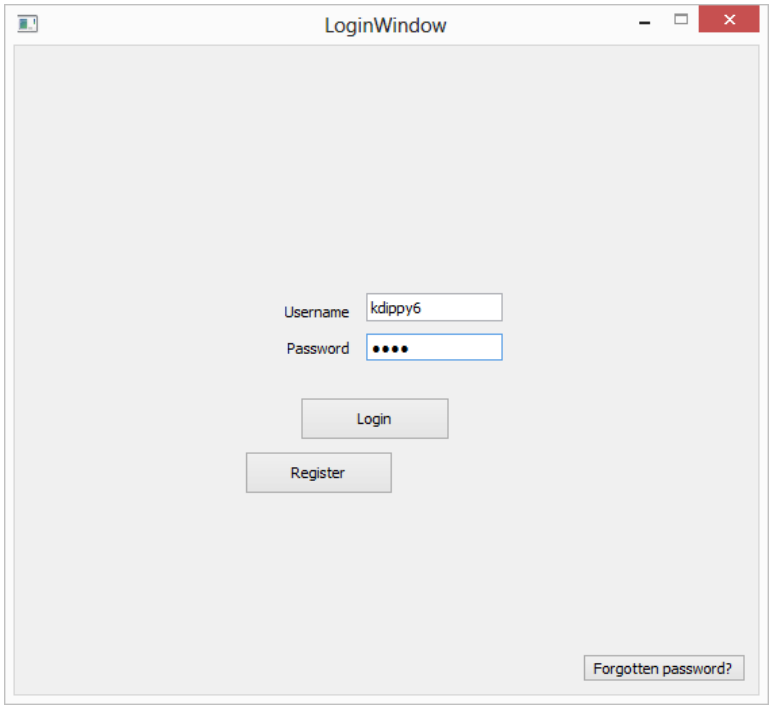


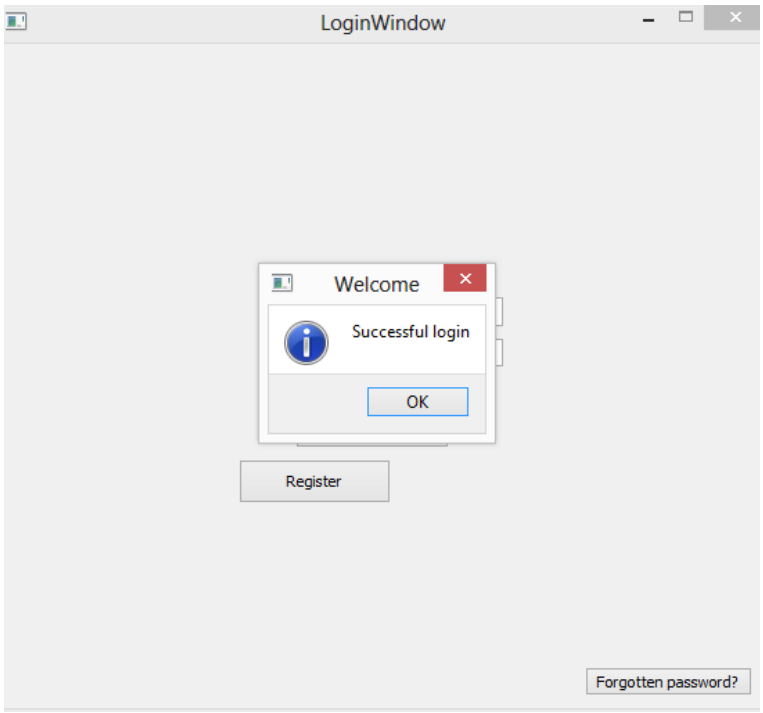
I then tested the same previous tested but substituted an Admin login name with incorrect password to match. Which also returned the expected output of incorrect details.

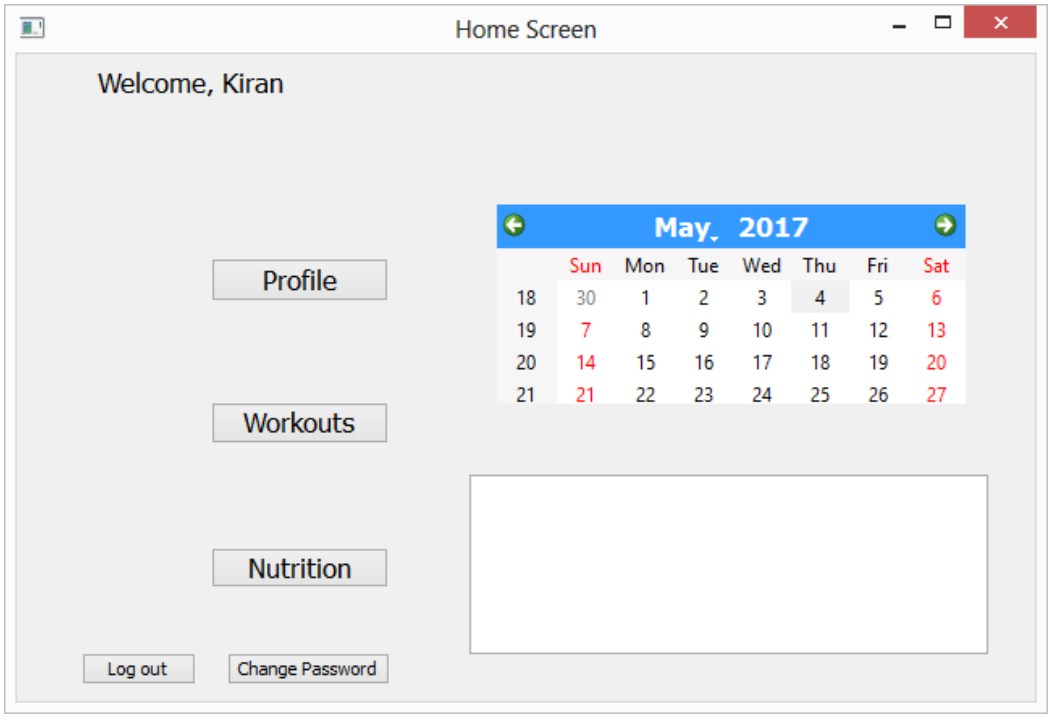


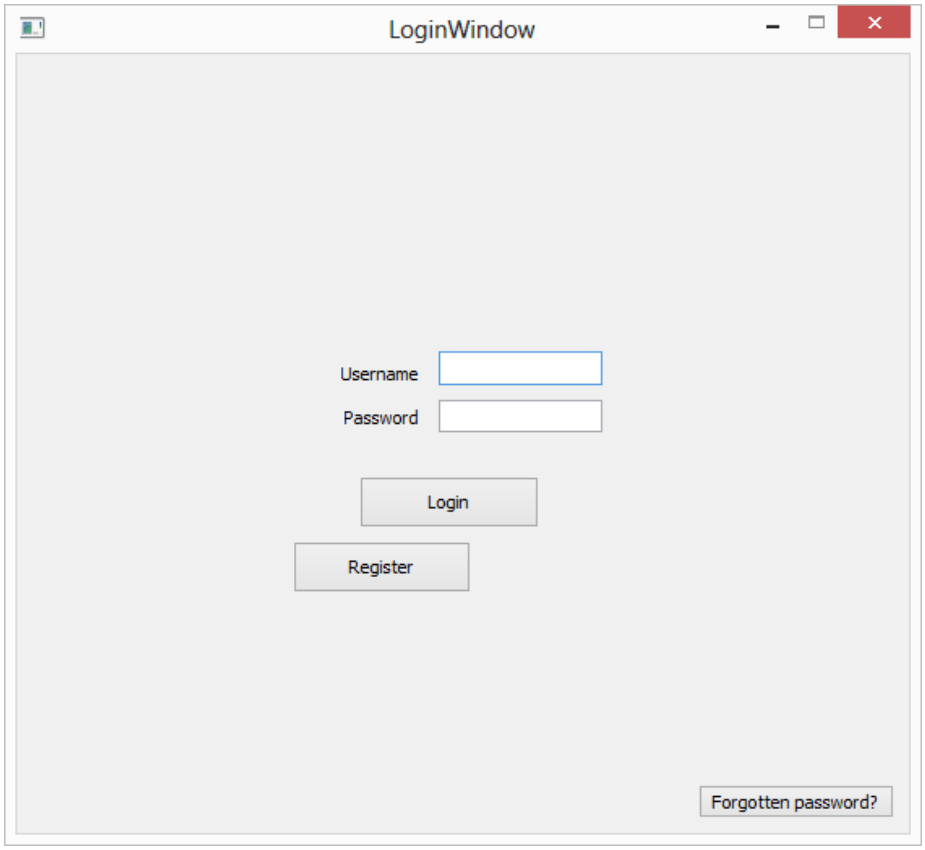


Finally, I tested if a normal user can login to their profile with the correct user details. As expected this allowed the user successful entry into their profile.









To follow I tested the log out button, which allowed the user to log out, and then return to the login screen. All data was wiped on the login screen to prevent the next user to see and potentially login with the last person's user details.

Finally, I tested the Admin login credentials and tested if inputting the correct Admin details would allow the Admin to gain access to their admin area rather than a regular profile.

This was a success as it opened the Admin screen for the user to navigate through. This also had a log out button which returned to the login screen without any information present. One of the next bits of code that I decided to complete was the nutrition. This was a simple process and still a key part of my program. This started with hard coding, then continued into a module and class in my program. It then proceeded to be implemented into a working GUI and then linked with a database.

```
lifestyle=("sedentary")
goal=("cut")
speed=("aggressive")
weightlbs=("165")

calorieintake=round(weightlbs*multiplier)

if lifestyle=("sedentary")
   multiplier=14
if lifestyle=("moderate")
   multiplier=16
if lifestyle=("Very Active")
   multiplier=18

if goal==("cut"):
      calorieintake=calorieintake-200
      if speed==("Aggressive"):
         calorieintake=calorieintake-100
         projectedweight=("-5lbs/week")
      elif speed==("Slow"):
         calorieintake=calorieintake+50
         projectedweight=("-1lb/week")
      else:
         projectedweight=("3lbs/week")
if goal==("bulk"):
      calorieintake=calorieintake+200
      if speed==("Aggressive"):
         calorieintake=calorieintake+100
         projectedweight=("+5lbs/week")
      elif speed==("Slow"):
         calorieintake=calorieintake-50
         projectedweight=("+1lb/week")
      else:
         projectedweight=("+3lbs/week")
if goal==("Maintain"):
   projectedweight=("+0lbs/week")

print(calorieintake, projected weight)
```

This was the first stage of coding. It only contains basic information, including lifestyle, goal, speed and weight as variables. These are hard coded, without any input from the user. This has basic if functions to ensure the right option is picked and appropriate functions are added to it.

This simply prints the calories that the user should be consuming and the projected weight that they will achieve over the progressing weeks.

The calorie intake variable first outputted

```
class NutrScr(QtGui.QMainWindow, NutritionScr):
  def __init__(self, parent=None):
    QtGui.QMainWindow.__init__(self, parent)
    self.setupUi(self)
    self.ChangeButton.clicked.connect(self.calorieint
    self.ReturnButton.clicked.connect(self.RegToHon

  def calorieintake(self):

    Sedentarys=self.Sedentary.isChecked()
    Moderately=self.Moderate.isChecked()
    VActive=self.Active.isChecked()
    cut=self.Cut.isChecked()
    bulk=self.Bulk.isChecked()
    maintain=self.Maintain.isChecked()
    aggressive=self.Aggressive.isChecked()
    slow=self.Slow.isChecked()
    balanced=self.Balanced.isChecked()

    if Sedentarys==True:
      life=1
      multiplier=14
    if Moderately==True:
      life=2
      multiplier=16
    if VActive==True:
      life=3
      multiplier=18
    if cut==True:
      goal=("Cut")
    if bulk==True:
      goal=("Bulk")
    if maintain==True:
      goal=("Maintain")
    if aggressive==True:
    if aggressive==True:
      speed=("Aggressive")
    if balanced==True:
      speed=("Balanced")
    if slow==True:
      speed=("Slow")
    weight=("165")


    calorieintake=round(weightlbs*multiplier)
    if goal==("Cut"):
        calorieintake=calorieintake-200
        if speed==("Aggressive"):
          calorieintake=calorieintake-100
          projectedweight=("-5lbs/week")
        elif speed==("Slow"):
          calorieintake=calorieintake+50
          projectedweight=("-1lb/week")
        else:
          projectedweight=("3lbs/weel")
    if goal==("Bulk"):
        calorieintake=calorieintake+200
        if speed==("Aggressive"):
          calorieintake=calorieintake+100
          projectedweight=("+5lbs/week")
        elif speed==("Slow"):
          calorieintake=calorieintake-50
          projectedweight=("+1lb/week")
        else:
          projectedweight=("+3lbs/week")
    if goal==("Maintain"):
      projectedweight=("+0lbs/week")

    message=(str(calorieintake))
    self.CalorieView.setText(message)
    self.ProjectedView.setText(projectedweight)
```

I then progressed this code as I decided to implement the GUI functions. This allowed inputs from the user through its user interface. This had to be put into a new class and function so it is callable when the window is opened. This reads in data from the GUI function to replace the hard coded lifestyle, goal and speed previously entered. This will then, instead of printing the outcome, output the result into the GUI for the user to see.

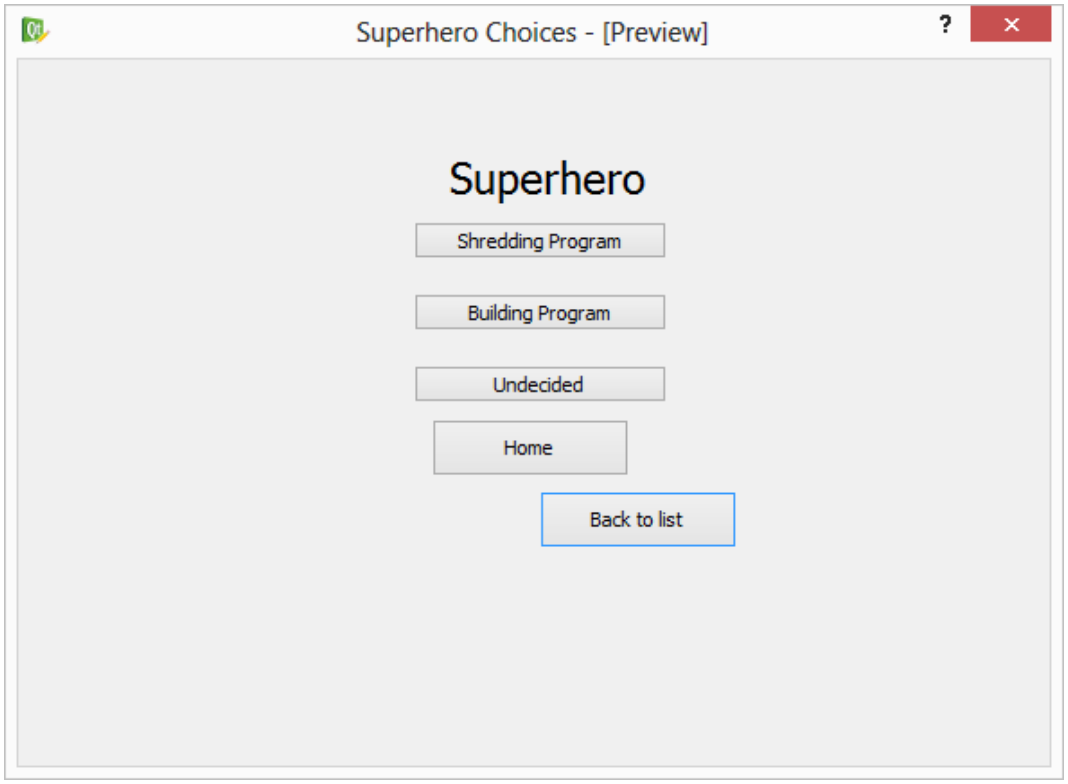
```
class NutrScr(QtGui.QMainWindow, NutritionScr):
  def __init__(self, parent=None):
    QtGui.QMainWindow.__init__(self, parent)
    self.setupUi(self)
    self.ChangeButton.clicked.connect(self.calorieintake)
    self.ReturnButton.clicked.connect(self.RegToHome)
  def calorieintake(self):
    Sedentarys=self.Sedentary.isChecked()
    Moderately=self.Moderate.isChecked()
    VActive=self.Active.isChecked()
    cut=self.Cut.isChecked()
    bulk=self.Bulk.isChecked()
    maintain=self.Maintain.isChecked()
    aggressive=self.Aggressive.isChecked()
    slow=self.Slow.isChecked()
    balanced=self.Balanced.isChecked()
    if Sedentarys==True:
      life=1
      multiplier=14
    if Moderately==True:
      life=2
      multiplier=16
    if VActive==True:
      life=3
      multiplier=18
    if cut==True:
      goal=("Cut")
    if bulk==True:
      goal=("Bulk")
    if maintain==True:
      goal=("Maintain")
    if aggressive==True:
      speed=("Aggressive")
    if balanced==True:
      speed=("Balanced")
    if slow==True:
      speed=("Slow")
    cur.execute("SELECT weight FROM profile where user='%s'"%usernome)
    s=cur.fetchall()
    weightlbs=((s[0][0])*2.2)
    calorieintake=round(weightlbs*multiplier)
    if goal==("Cut"):
        calorieintake=calorieintake-200
        if speed==("Aggressive"):
          calorieintake=calorieintake-100
          projectedweight=("-5lbs/week")
        elif speed==("Slow"):
          calorieintake=calorieintake+50
          projectedweight=("-1lb/week")
        else:
          projectedweight=("3lbs/weel")
    if goal==("Bulk"):
        calorieintake=calorieintake+200
        if speed==("Aggressive"):
          calorieintake=calorieintake+100
          projectedweight=("+5lbs/week")
        elif speed==("Slow"):
          calorieintake=calorieintake-50
          projectedweight=("+1lb/week")
        else:
          projectedweight=("+3lbs/week")
    if goal==("Maintain"):
      projectedweight=("+0lbs/week")

    message=(str(calorieintake))
    self.CalorieView.setText(message)
    self.ProjectedView.setText(projectedweight)
    cur.execute("UPDATE profile set calories='%s', projected='%s', lifestyle='%s', goal
    con.commit()
```

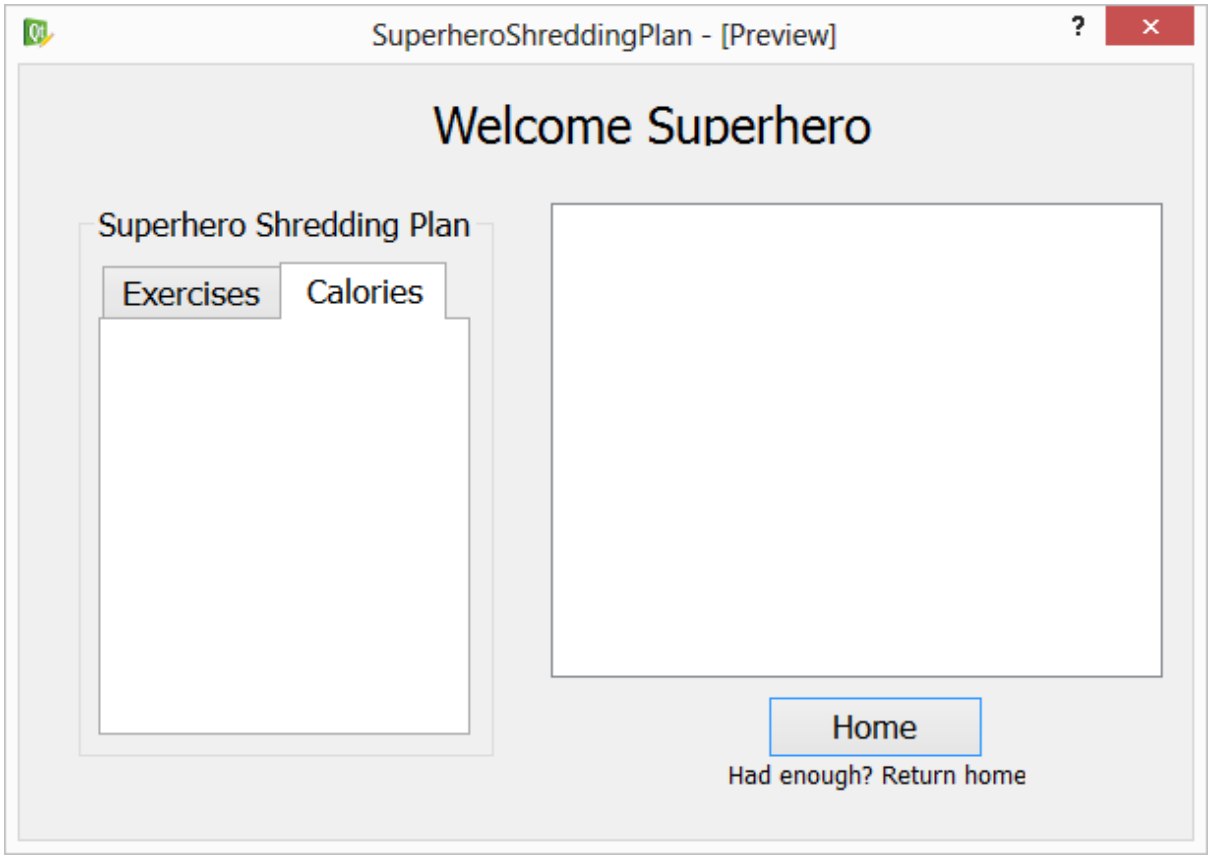
The final bit of progress that I made on this piece of code was linking it to SQL. This allows all the inputs that the user enters to be saved and outputted for later use. From the previous update, not much change has occurred except from two SQL queries reading in data and then another to update the SQL tables too.

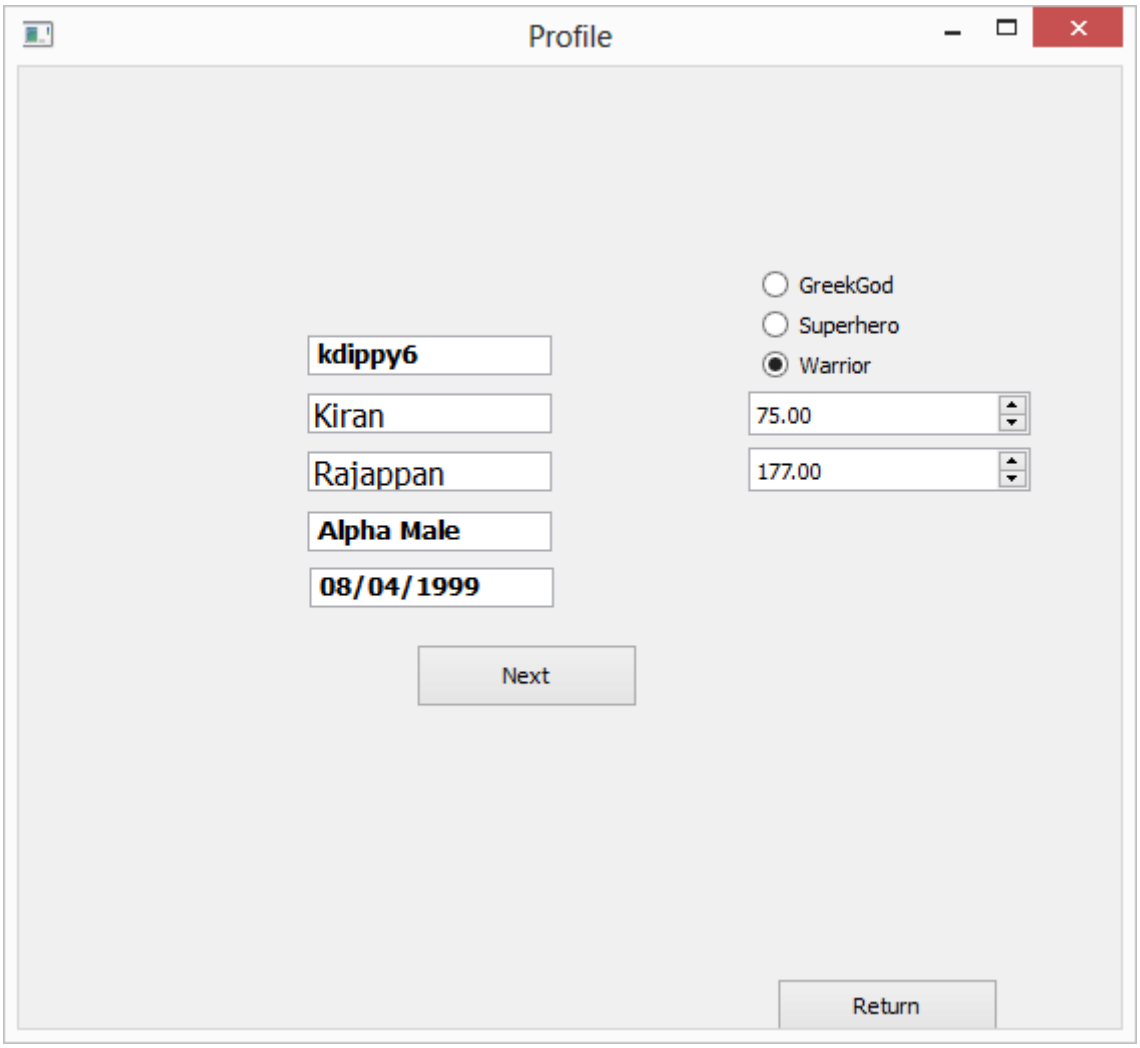
Changes and significant changes

A significant change that I have made in the development of my program was the GUI aspect of it. When beginning the GUI, I decided that a picture based interface would be the most appropriate and pleasing. However, after referring to Olek and the rest of my target market group, they all agreed that a clean form based interface was more appropriate below is a few screenshots of the previous interface and the new updated one. One approach I had to complete my program was to have complete separate windows for each of the physiques that were available. This included a GreekGod, Warrior and Superhero stages which had their own complete profiles. However, I started this, I soon realized that the work and time that would be needed to complete this, was far greater than that I had

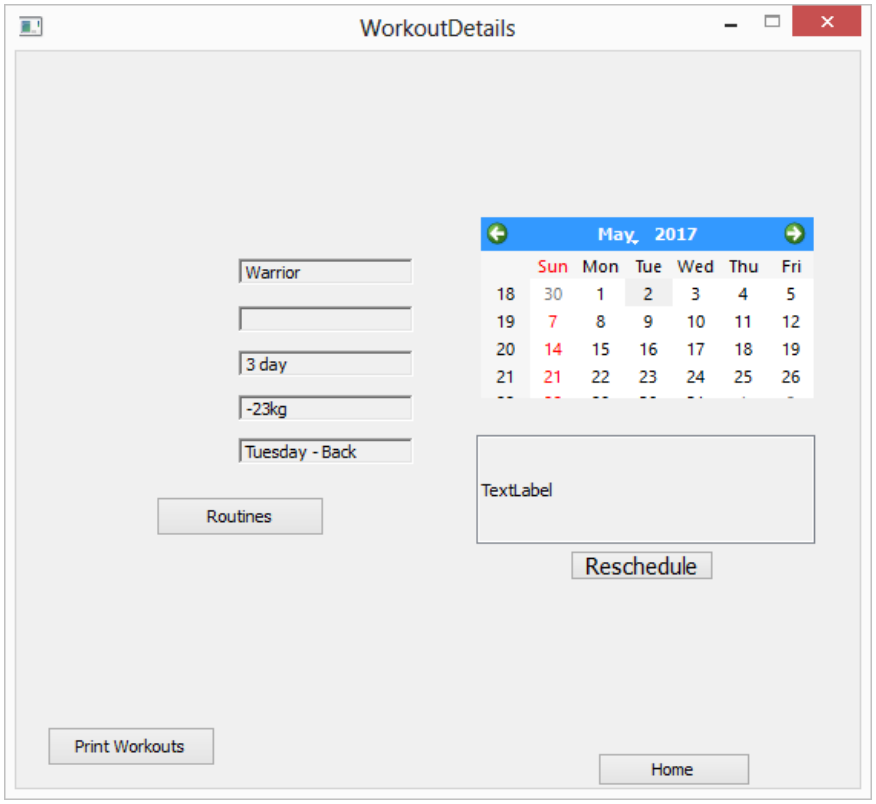
time to do and thus was not appropriate. Furthermore, I found that there was a lot of redundancy within this and was not efficient.

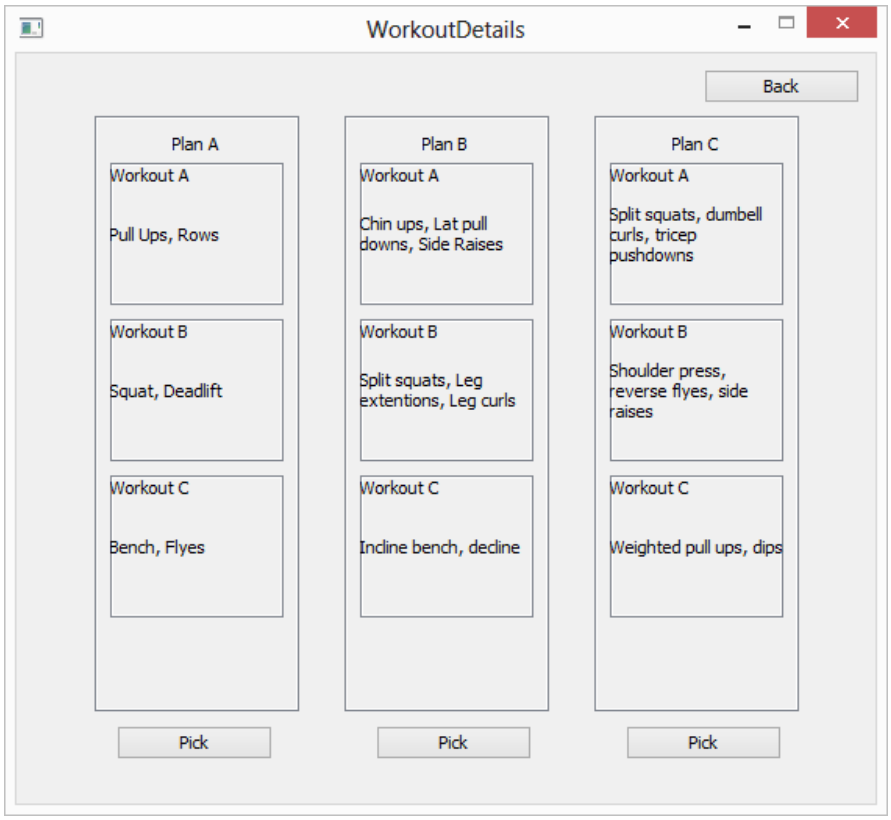


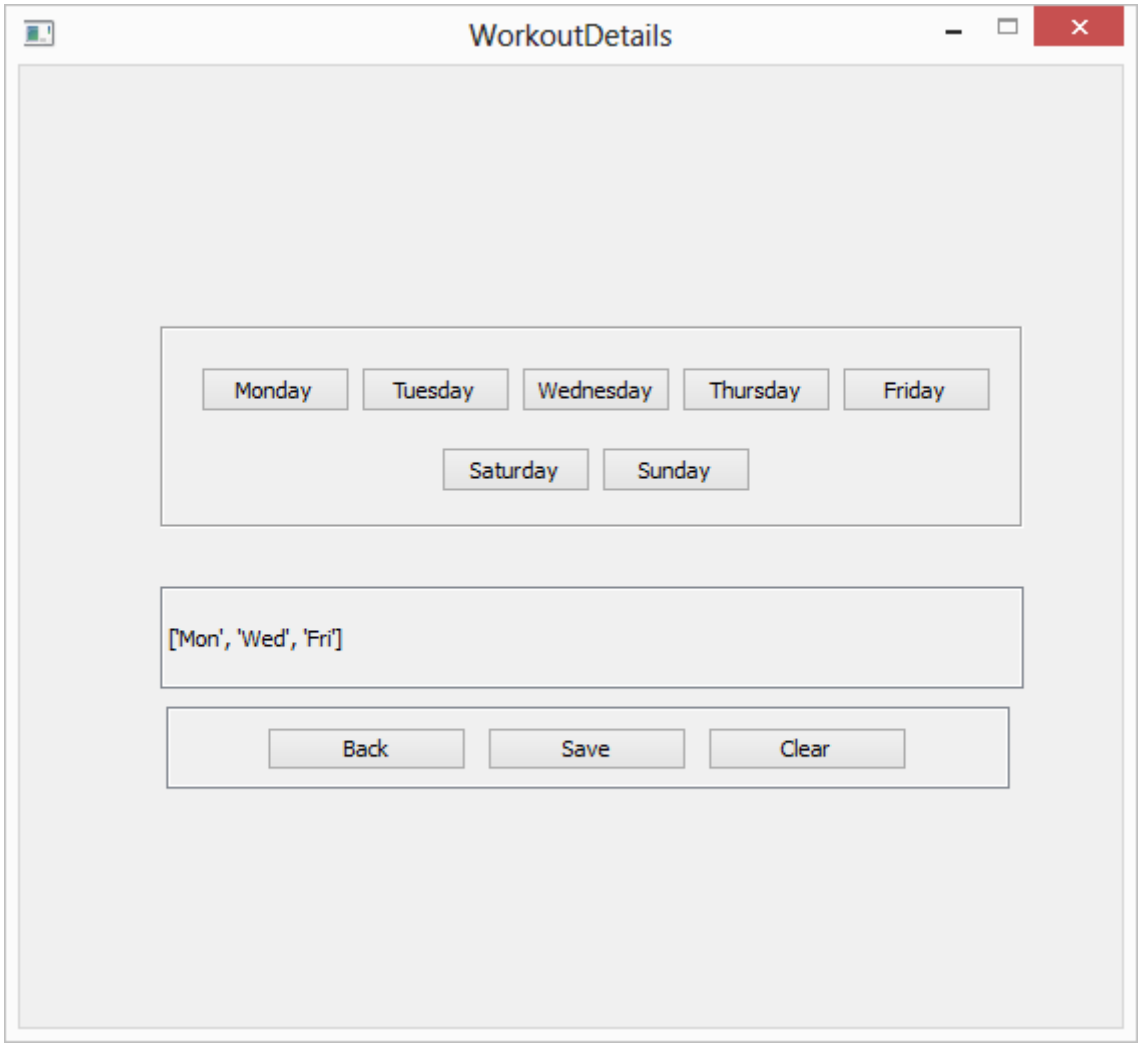




The first two windows show the Superhero equivalent of their workout screens. This would be completely separate to any of the other plans. This was far less efficient than what I currently have, which is a simple radio group box button to choose which physique that they would like, and then just personalize the main general workouts window for each place. The previous idea was not continued as this was a dead end approach. This also shows the difference between the start and the end progress of the workout screens.


WorkoutDetails
Warrior
3 day
-23kg
Tuesday - Back
Routines
May, 2017
Sun Mon Tue Wed Thu Fri
18 30 1 2 3 4 5
19 7 8 9 10 11 12
20 14 15 16 17 18 19
21 21 22 23 24 25 26
TextLabel
Reschedule
Print Workouts
Home


WorkoutDetails
Back
Plan A
Workout A
Pull Ups, Rows
Workout B
Squat, Deadlift
Workout C
Bench, Flyes
Pick
Plan B
Workout A
Chin ups, Lat pull downs, Side Raises
Workout B
Split squats, Leg extentions, Leg curls
Workout C
Incline bench, decline
Pick
Plan C
Workout A
Split squats, dumbell curls, tricep pushdowns
Workout B
Shoulder press, reverse flyes, side raises
Workout C
Weighted pull ups, dips
Pick



Proof of programming constructs

```
import sys, os
from tkinter import * ;import tkinter.ttk as ttk; import cls_pie as pie
from PyQt4 import QtCore, QtGui, uic
import sqlite3 as lite
import re
from errorcheck import*
from regcheck import *
import time
from sendingemail import*
from datetime import date
import calendar
import csv
on=lite.connect('testerdb.db') #connect sql file
cur=con.cursor() # as above
```

```
def checkingpass(regpass,repass,self):
    prohibchar=["¬","`","¦","%","^","&","*","(",")'
    profdet=[]
    checkuser=[]
    cur.execute("SELECT * FROM profile")
    rows = cur.fetchall()
    for row in rows:
        profdet.append(row)
    useresults = [row[0] for row in profdet]
    useremail = [row[5] for row in profdet]
    for x in useresults:
        checkuser.append(x)

    lenpas=len(regpass)
    check=0

    if lenpas<4:
        errorpaslenmin(self)
        check=check+1
    if lenpas>20:
        errorpaslenmax(self)
        check=check+1
    for m in prohibchar:
        if m in regpass:
            passerror(self)
            check=check+1
    if regpass!=repass:
        errorreg(self)
        check=check+1
    print(check)
    if check==0:
        return True
```

```
reguser=self.UsernameInput.text()
regpass=self.PasswordInput.text()
emailaddress=self.EmailAddressInput.text()
repass=self.ReEnterPasswordInput.text()
checking(reguser,emailaddress,self)
checkingpass(regpass,repass,self)
```

The following code has been imported as a part of a separate module that I have created. The code is an example of a function which passes through parameters by values (only reads in the value and so can not be changed inside the function). This then checks the length of the input (in the case the password) for these basic tests. This then returns a Boolean value of either True or False to determine if the password the expectations of the length. This function passes through a parameter by reference, in a separate module, to check the password for strength and correct format. This first checks the length, to ensure that it is strong enough, it then ensures that there are no spaces and prohibited characters, checks if the password is the entered the same too.

This class is set up for my registration window. This contains all the code needed to generate this part of my program. It sets up the the objects that I have (buttons) and then calls different functions to link to the buttons.

```
def __init__(self, parent=None):
    QtGui.QMainWindow.__init__(self, parent)
    self.setupUi(self)
    self.model = QtGui.QStandardItemModel(self)
    self.tableView.setModel(self.model)
    self.load_data()
    self.tableView.clicked.connect(self.show_selected) #links button to specific function to run code as button is pressed
    self.DeleteUser.clicked.connect(self.AdmDeleteUser)
    self.UpdateDetails.clicked.connect(self.AdmUpdate)
    self.LogOut.clicked.connect(self.ReturnLogin)
    self.ViewStats.clicked.connect(self.Stats)

def load_data(self):

        con = lite.connect('testerdb.db')#opens a file
        cur = con.cursor()#keeps track of the "tape" position
        #parameter query - get criteria from Python
        s='select user, forename, surname, gender, emailaddress, dob from profile' #not case sensitive inside the string
        cur.execute(s)
        self.data = cur.fetchall() #query data comes back
        if len(self.data)>0:
            for i in range(len(self.data)-1,-1): #iterates through the list to remove the selected person from the list
                print("removing row",i)
                self.data.pop(i)
                self.model.removeRow(index.row(self.data[i])) #removes row from the list where the user had chosen
        self.model=QtGui.QStandardItemModel(self)
        self.tableView.setModel(self.model)
        for row in self.data:
                items = [
                    QtGui.QStandardItem(str(field))
                    for field in row
                ]
                self.model.appendRow(items)
        #print(self.model.data(self.model.index(2,1)))

def show_selected(self):
        index = self.tableView.currentIndex() #returns currently selected cell
        r=index.row() #returns the row of the currently selected cell
        c=index.column()#returns the column of the currently selected cell
        cell_contents=index.data()#returns the contents of the currently selected cell
        print("index,r,c",r,c,cell_contents)
        rowname=""
        if c==0:
            rowname="Username"
```

This code chunks contains class setups which allows inheritance, multiple inheritance, polymorphism, procedures, SQL queries and evidence of Hungarian notation.

```
    def show_selected(self):
        index = self.tableView.currentIndex() #returns currently selected cell
        r=index.row() #returns the row of the currently selected cell
        c=index.column()#returns the column of the currently selected cell
        cell_contents=index.data()#returns the contents of the currently selected cell
        print("index,r,c",r,c,cell_contents)
        rowname=""
        if c==0:
            rowname="Username"
        if c==1:
            rowname="Forename"
        if c==2:
            rowname="Surname"
        if c==3:
            rowname="Gender"
        if c==4:
            rowname="Email Address"
        if c==5:
            rowname="DOB"
        message=("Current "+rowname+" is "+cell_contents+", please double click slots to update")
        self.changelabel.setText(message)
##          self.lbl_row.setText(str(r) )
##          self.lbl_col.setText(str(c) )
```

Comments

```
    def AdmUpdate(self):
        index = self.tableView.currentIndex()
        cell_contents=index.data()
        ids=self.model.data(self.model.index(index.row(), 0))
        print(ids,self.model.data(self.model.index(index.row(), 1)))
        name=self.model.data(self.model.index(index.row(), 1))
        surnameedit=self.model.data(self.model.index(index.row(), 2))
        genderedit=self.model.data(self.model.index(index.row(), 3))
        emaileditinput=self.model.data(self.model.index(index.row(), 4))
        self.model.data(self.model.index(4,1))
        con = lite.connect('testerdb.db')
        cur = con.cursor()
        cur.execute("UPDATE profile set forename='%s',surname='%s',gender='%s',emailaddress='%s'  WHERE user=('%s')"%(name,surnameedit,genderedit,emaileditinput,ids))
        con.commit()
        self.load_data()
```

Hungarian notation

I have used Hungarian notation in order to keep track of the variables that I use and so that I know what data type I must be using for each case. This, along with meaningful variable names, allow the program to be easily maintainable in the future too.

SQL parameters including %s to search through with user inputs

Function

```
def ContinueRegister(self):
    global reguser
    global regpass
    global emailaddress

    reguser=self.UsernameInput.text()
    regpass=self.PasswordInput.text()
    emailaddress=self.EmailAddressInput.text()
    repass=self.ReEnterPasswordInput.text()
    checking(reguser,emailaddress,self)
    checkingpass(regpass,repass,self)

    resultans=checking(reguser,emailaddress,self)
    resultans1=checkingpass(regpass,repass,self)

    if resultans and resultans1==True:
        RegWind.UsernameInput.clear()
        RegWind.PasswordInput.clear()
        RegWind.EmailAddressInput.clear()
        RegWind.ReEnterPasswordInput.clear()
        ConRegWin.show()
        self.hide()


    else:
        print("re-enter")


    ConRegWin.A_ForenameInput.clear()
    ConRegWin.B_SurnameInput.clear()
    ConRegWin.SecurityAnswer.clear()
    ConRegWin.comboBox.setCurrentIndex(0)
    ConRegWin.AlphaMale.setChecked(False)
    ConRegWin.Goddess.setChecked(False)
    ConRegWin.GreekGod.setChecked(False)
    ConRegWin.Superhero.setChecked(False)
    ConRegWin.Warrior.setChecked(False)
    ConRegWin.E_WeightInput.setValue(65.00)
    ConRegWin.F_HeightInput.setValue(165.00)
#     ConRegWin.dateb.setDate(20/Jan/2001)
```

Global variables

This code focuses on the using the imported modules while passing through some parameters, as well as setting the results of which with different uses of variables including private and public. This also shows links to different classes .

Local Variables

Checking the return values of funtions

This shows some of the local variables that I also use. This is usually assigned as a temporary storage for calculations.

Use of Hungarian notation and resetting inputs

This shows one of the global variables that I use in my program. This makes the name of the current user, available throughout the program, without the need of having to fetch it from the database every time it is needed. This is a suitable global variable, as this does not cause any security concerns as it not sensitive.

Throughout my program, I use multiple modules that have been both linked and loaded to help me. I have also created my own modules, that I coded and have linked to the code in the main program. An example of me using one of these modules is datetime, to retrieve the current date.

This chunk of code also includes lots of examples of programming constructs. This includes presence and data type check, local variables SQL parameters and finally, using imported files again.

```
tempdob=self.dateb.date()
try:
    dob=str(tempdob.toPyDate())
except:
    wronginputtype(self)
dobyear=dob[0:4]
todaysDate=time.strftime("%Y/%m/%d)")
year=todaysDate[0:4]
try:
    personsage=int(year)-int(dobyear)
except:
    notentered(self)
else:
    pass


if 16>personsage or personsage>65:
    nextwind=nextwind-1
    ageoutofrange(self)


male=self.AlphaMale.isChecked()
female=self.Goddess.isChecked()

if male==True:
    regender=("Male")
if female==True:
    regender=("Goddess")

if male ==False and female == False:
    notpickedgen(self)
    nextwind=nextwind-1

gg=self.GreekGod.isChecked()
sh=self.Superhero.isChecked()
Warrior=self.Warrior.isChecked()
if gg==True:
    physiq=("GreekGod")
if sh==True:
    physiq=("Superhero")
if Warrior==True:
    physiq=("Warrior")

if gg==False and sh==False and Warrior==False:
    notpickedphysique(self)
    nextwind=nextwind-1


weight=self.E_WeightInput.text()
height=self.F_HeightInput.text()
    printweight,height)
if ("40")>weight or weight>("200"):
    nextwind=nextwind-1
    weighterror(self)
if ("120")>height or height>("250"):
    nextwind=nextwind-1
    heighterror(self)

if SecurityQ1==("Select"):
    nextwind=nextwind-1
    secureque(self)
if self.SecurityAns==(""):
    secureans(self)
    nextwind=nextwind-1



if nextwind==0:
    successlogin(self)
    TheFirstWindow.show()
    self.hide()
    cur.execute("INSERT INTO profile (user, password, emailaddress, forename, surname, gender,
    con.commit()

                                    pass,emailaddress, self.regname,self.regsurname,regender,physiq, weight, height,dob,SecurityQ1,self.SecurityAns)) e
```

This will check the data type, to ensure that the only data type that is inputted. If an integer is inputted here, the result will run a function, to prompt the user to enter the correct data type.

This will check if there is an input by the user. If the user fails to input anything, a function will be run, which will prompt the user to input data.

```
import re

ADDRESSTOVERIFY ='kdippy6@gmail.co.uk'
match = re.match('^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,4})$',ADDRESSTOVERIFY)

if match == None:
  print('Bad Syntax')
```

I have used regex in order to validate the emails, that users enter. This is to make sure that no spam accounts are created and to ensure that users are entering a valid email, so that they can access their account later. This is an entire self made module which has been imported in. This also shows the use of a capitalized variable to signify a constant, which will not be altered in the program. This will remain the same throughout the duration of the program running, unless a developer needs to update a certain regex function.

```
    def profileuser():
      lenuser=len(usernome)
      lenpas=len(self.password)
      state=False
      counter=0
      print(counter)
      admin=False



##        def login():
####        counter=counter+1
      for j in range (len(usernames)):
        if usernames[j]==usernome:
          if usernome==("Admin"):
            admin=True
            print(usernome,admin)


            counter=counter+1
##              while counter<6:
            if passwords[j]==self.password:
              state=True
              cur.execute("SELECT Forename FROM profile where user='%s'"%usernome)
              rows = cur.fetchall()
              global name
              name=(rows[0][0])

##          login()

      if state==True:
        if admin==True:
          cur.execute("SELECT * FROM profile")
          rows = cur.fetchall()
      AdminScreen.show()
      self.hide()
    else:
      welcome(self)
      HomeWind.label_2.setText("Welcome, %s"%name)
      HomeWind.show()
      self.hide()
  else:
    error(self)

profileuser()

TheFirstWindow.UsernameInput.clear()
TheFirstWindow.PasswordInput.clear()

def GetPassword(self):
```

This is the code which splits the account into admin and regular users. It first checks if, when the username and password entered matches an admin account, it will divert the screen to the Admin screen otherwise it will continue on as normal. Admin, can then change a users name and basic details which will effect the users account. This also shows use of SQL queries, loops and counters.

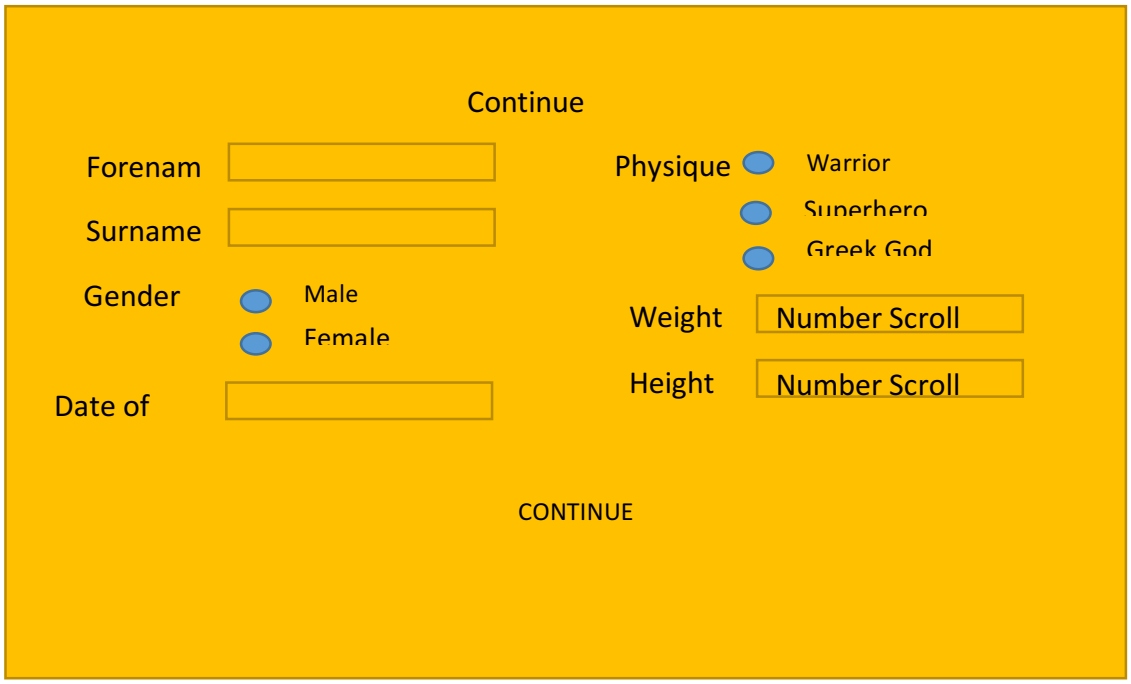
Finalising development

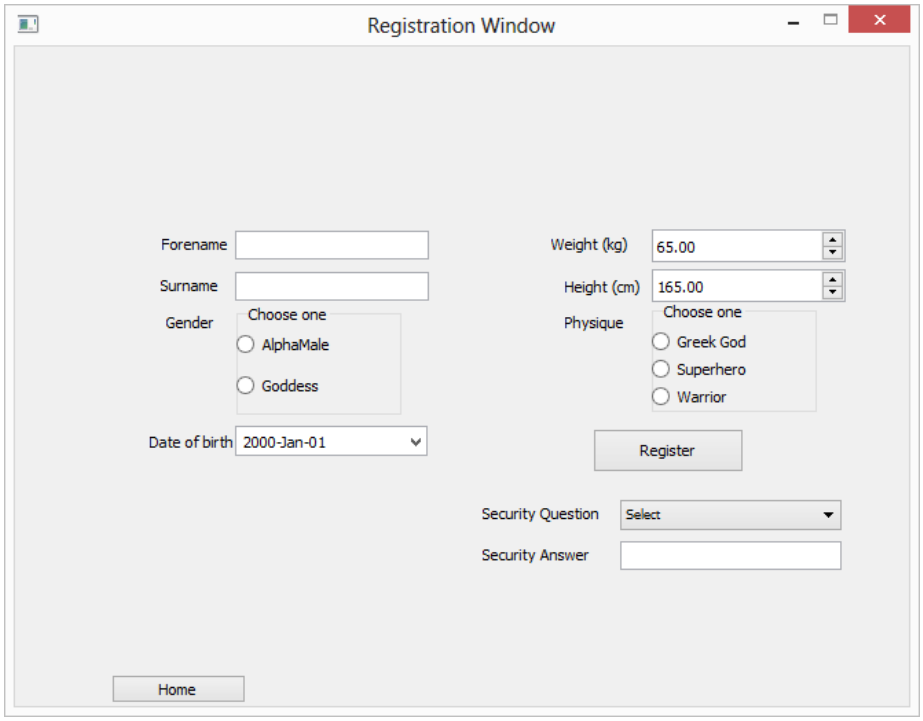
To sum up all the programming constructs I have used, I will list them below and briefly explain again what and how I have used them. Although I have already noted this on the previous sides, I decided to do this as extra means of explaining this. I have shown that I have used comments to ensure the maintainability of my program to any future developers. Not only is this contained in my program files, at the end of this document file, the entire pasted code of my program is present. Here, I have added further comments to adhere to the programmer of choice to refer to. I have shown multiple times examples of parameter passing, presence and data checks, functions and procedures (both returning values or non-returning), different variables throughout (public and private), use of constants and self made modules which are imported in and linked to the main program. I have shown different types of users (both admin and regular user), set up modules to check for strength and validity of passwords entered, as well as using regex to check the email addresses inputted, multiple SQL queries, both reading and writing (using ?? as well as %S to read in values to the queries), Inheritance, Hungarian notation where possible to allow easier use and understanding by future developers. Finally, I have shown multiple inheritance and polymorphism throughout the program. To explain inheritance, as this lacked previously in the captions, I will mention the classes I have used throughout the program. These classes are set up to contain all the variables, tables and code generated for a single window within PyQt 4. The main classes inherits these values from QMain.Window which is the initial set up of the pyqt structure and basic boilerplate code for this. This then allows python to import in functions which can be read and executed to exist inside of python and to also create values within pyqt itself, like for example a calendar function which I used. This can be seen by the clicked to connect values scattered around the program, which reads in the basic value of being clicked in order to run a function within the class to execute futher code.

Evaluation

Testing success criteria

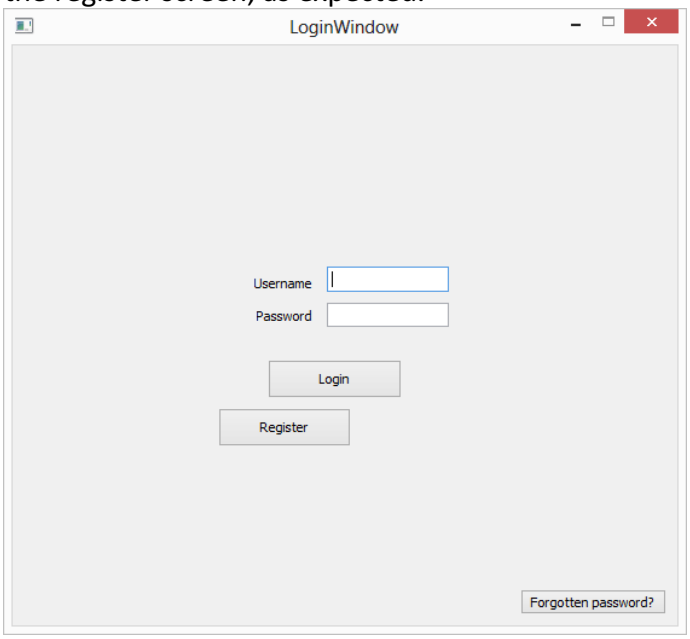
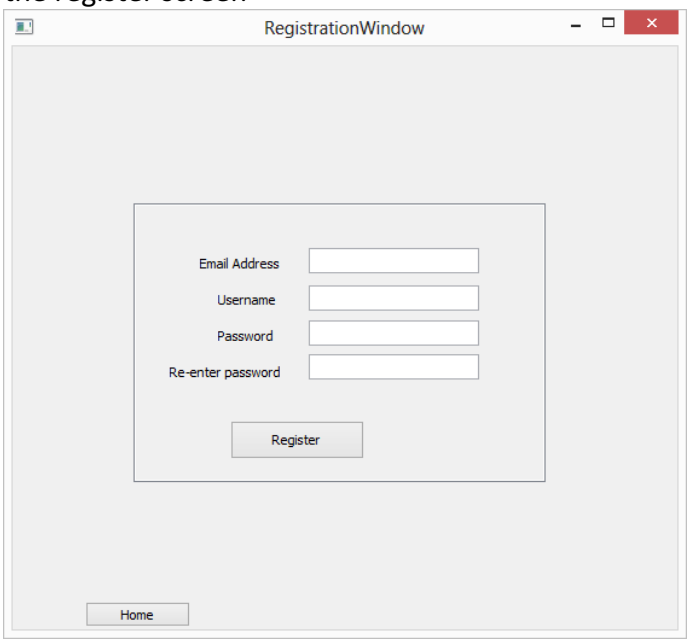
This is the planned design of the registration compared to the final design. The final outcome was very similar to what I planned to do, with the only difference being the colour and final groupings. The order was changed for easier functionality, and added security question and answer to ensure the safety of my users. If I had more time, to go back and change this, I would ensure that the program includes some more branding overall. This being professional background screening like on other parts of my program. Another thing that I would like to add is an icon on all the windows where possible. However, due to time constraints, this is not possible. Furthermore, as I am limited with what pyqt can offer, I would have liked to implement further animated inputs for a greater aesthetically pleasing use.

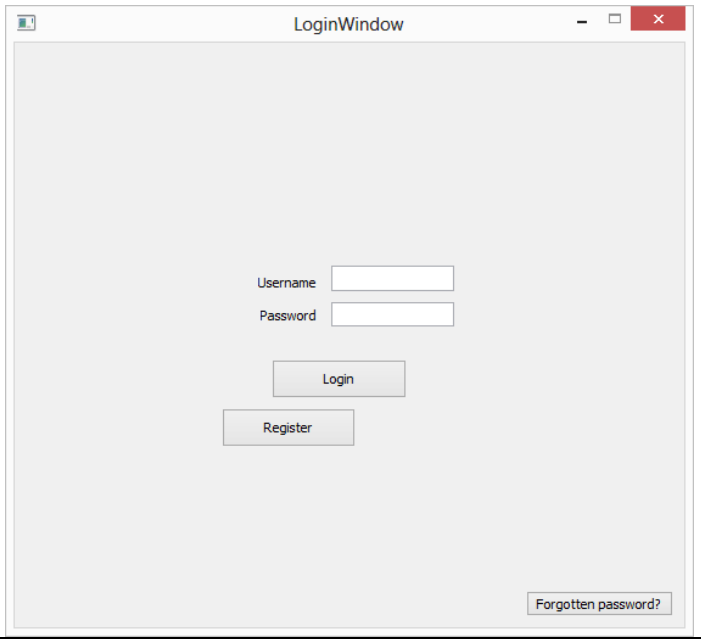
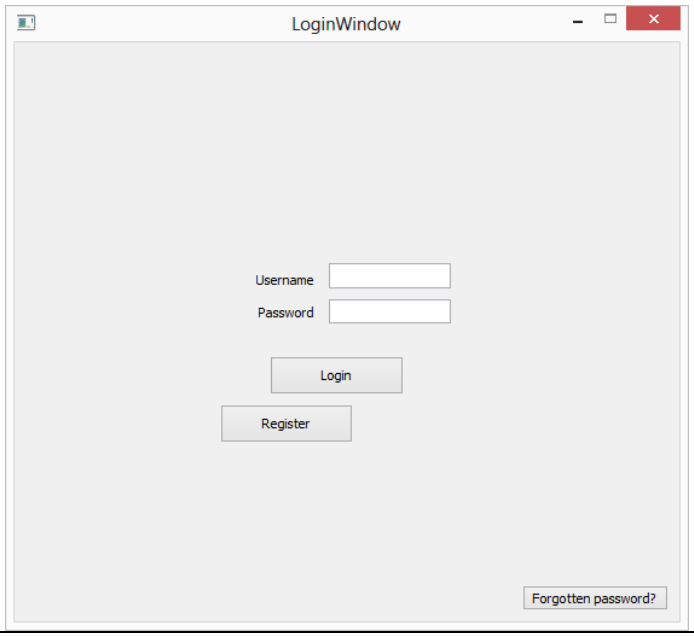


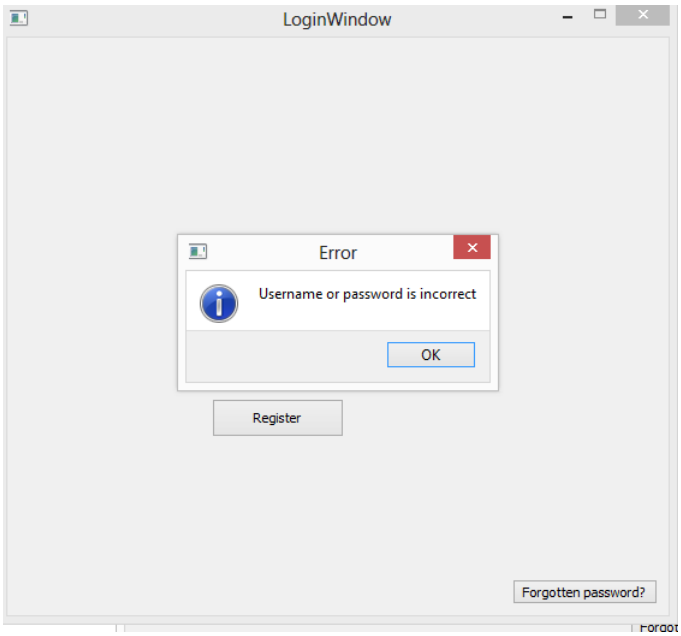
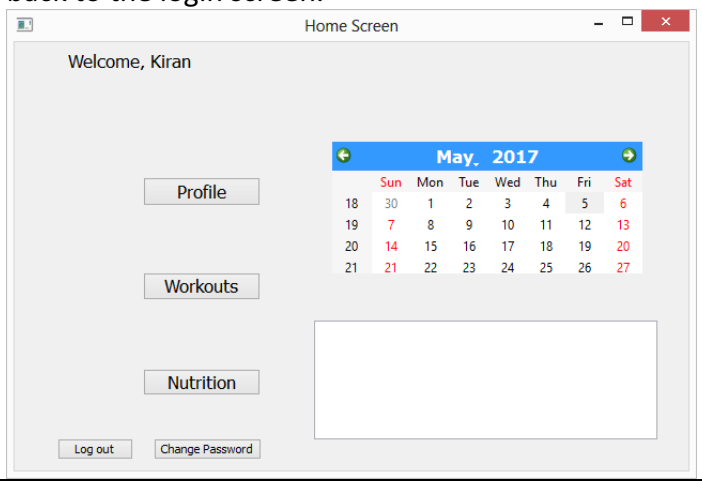


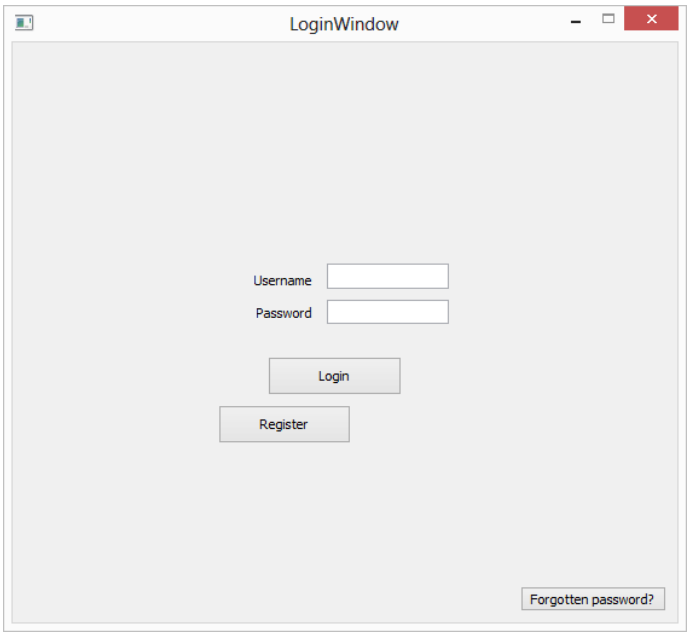
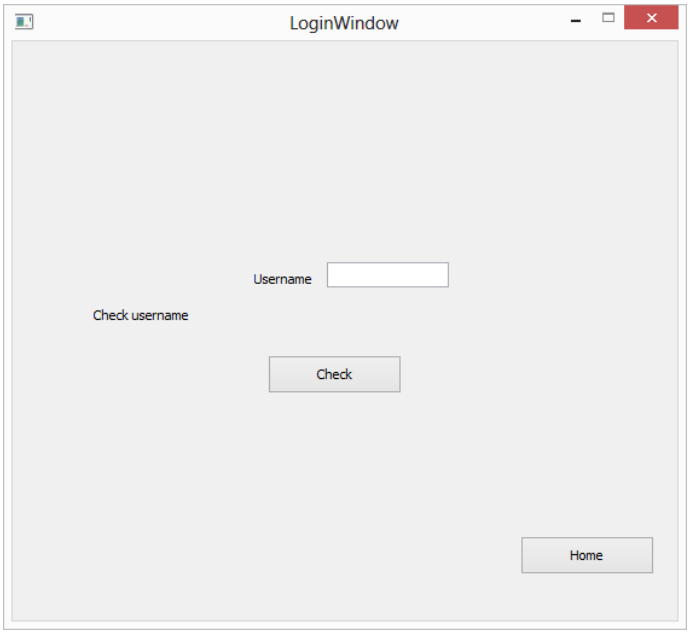
Timing the use of registration screen

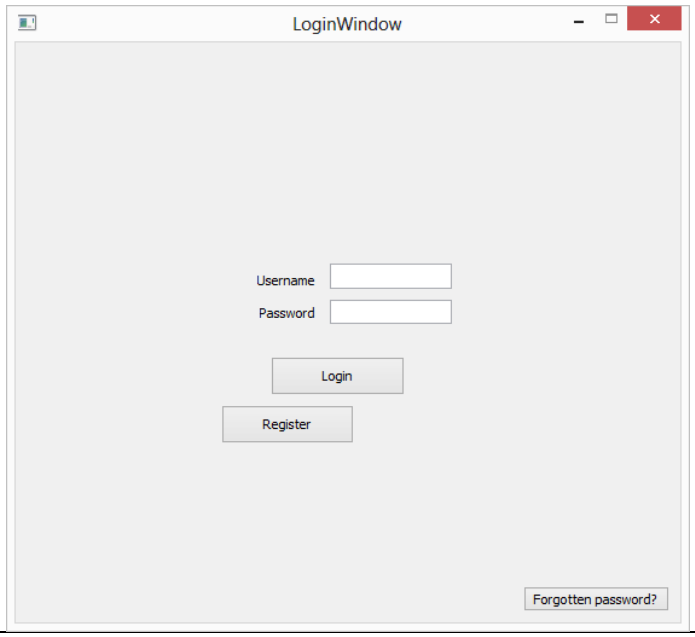
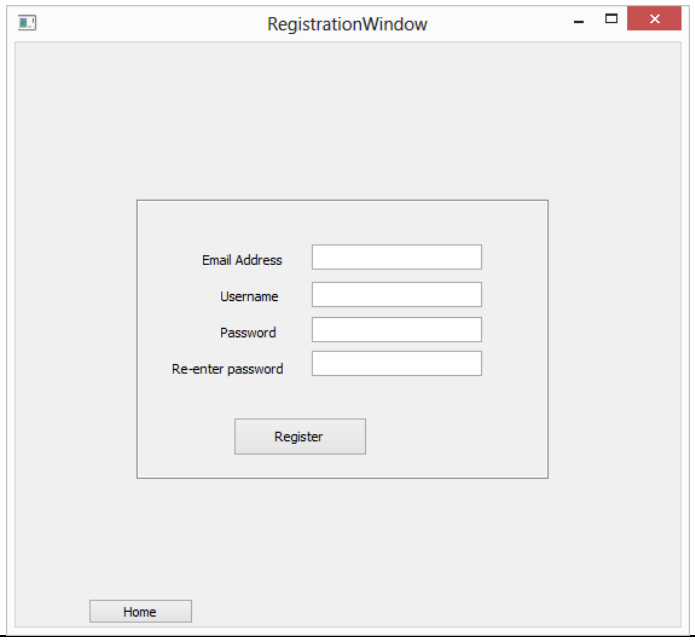
To further test this, I asked Olek to help me run through the program to see how he found it. It took him 2:45 minutes to complete the entire registration portion and to be able to login into his account. He noted that the format was easy to follow and included all the information needed. He noted that security question and answer could have been placed in a better position, but only was a minor issue.

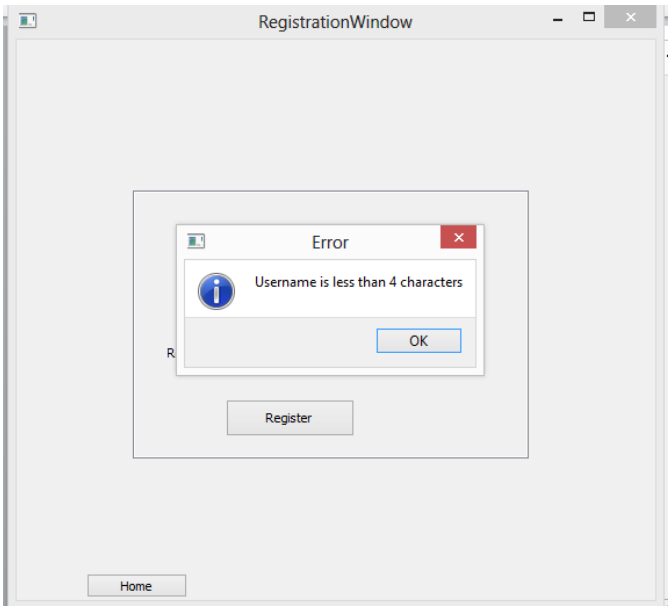
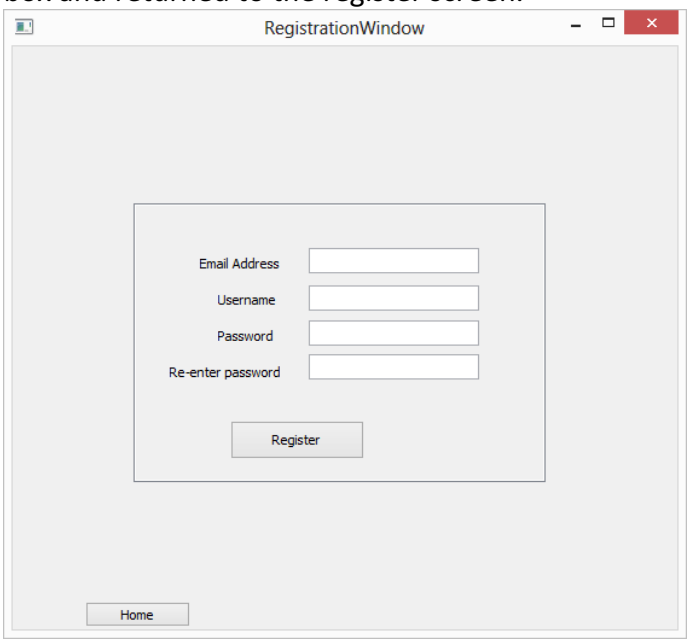
Test Number	Example of input	Relation to success criteria	Expected output	Actual Output
1a	Login screen – press register button and then return button	This will be tested by allowing Olek and myself to press every button on each of the screens. I will have a plan of the windows showing where each should be connected to. If these buttons take the user to the correct window, it will be checked off the list showing that it works.	All the windows will be checked off list that ensures all the windows are connected. None of the buttons will be dead, i.e, don't not have a specific function. All buttons connect to the correct window and function.	When pressing the register screen, this navigated to the register screen, as expected.  I clicked the register button and then navigated to the register screen  I then reached the register screen and the pressed home to return to the home screen

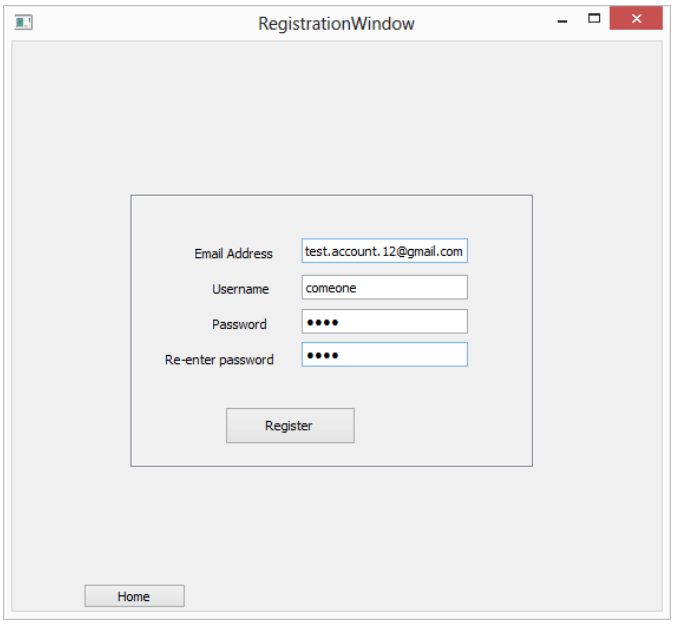
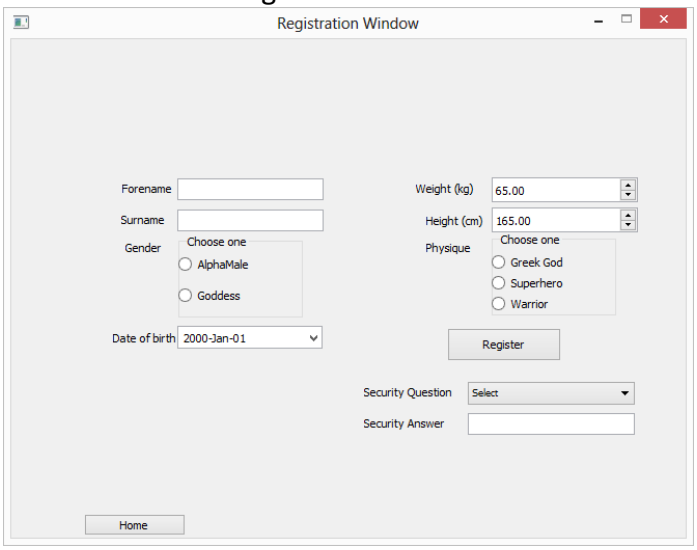
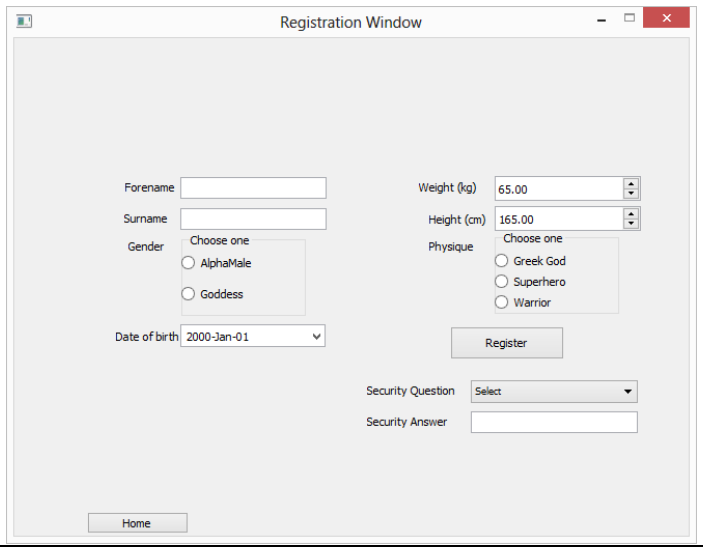
				
b	Login screen – press login button (no entry)	Same as above	Same as above	When I pressed this button with no input entries, it popped up an error as the username and password entered was incorrect. 

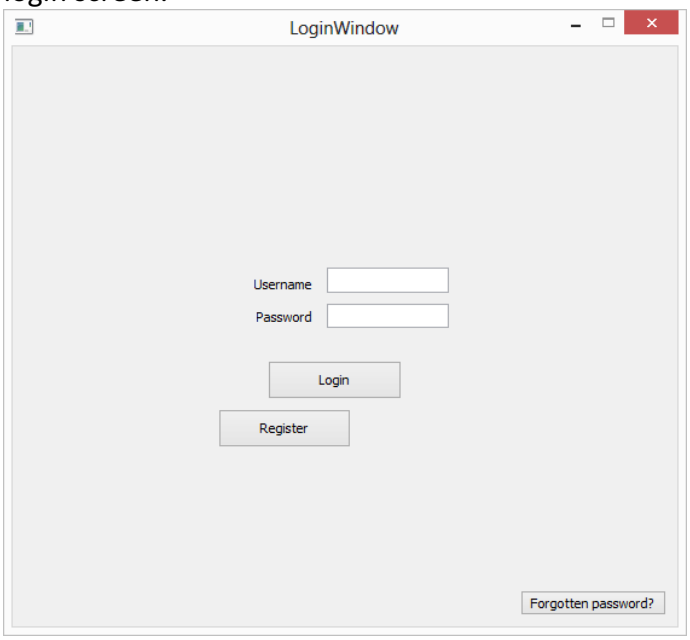
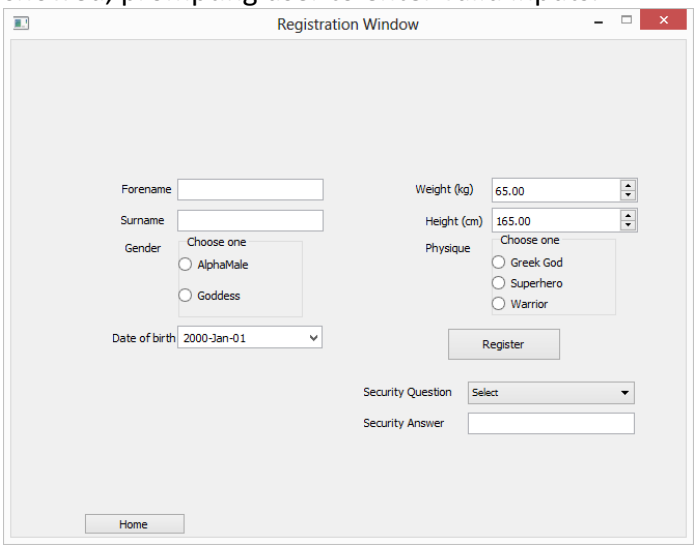
				 I then pressed ok on the popup. Which returned me back to the login screen. 
c	Login Screen – press forgotten password button and then home button	Same as above	Same as above	I pressed the forgotten password button and was navigated to the forgotten password screen.

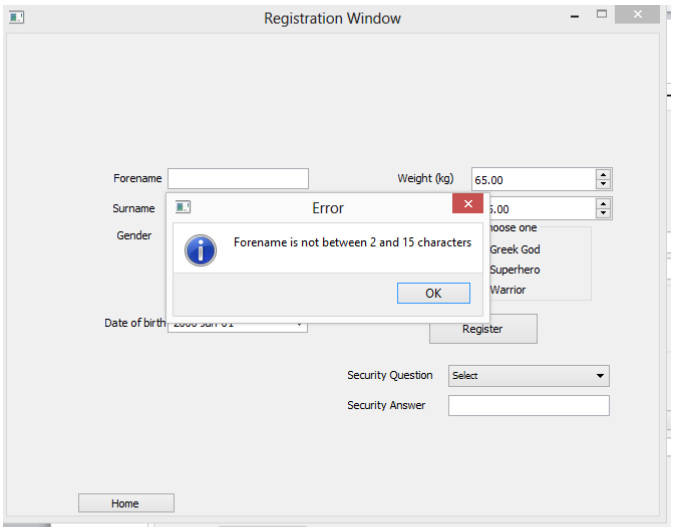
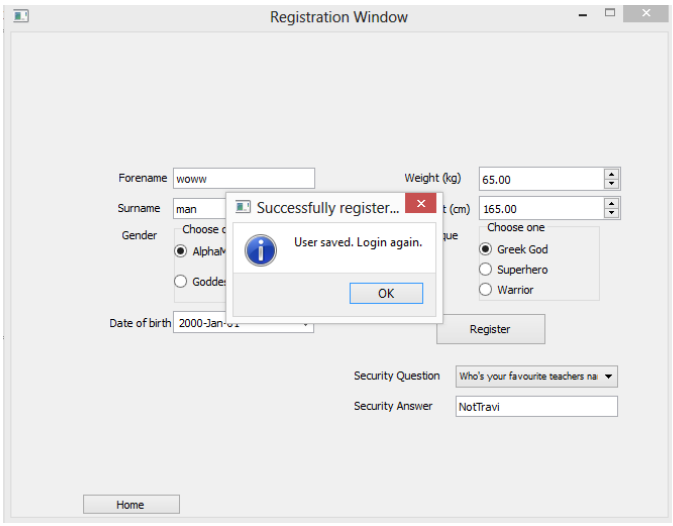
				 Once I reached this page, I then pressed the home button.  This then returned me to the home page.

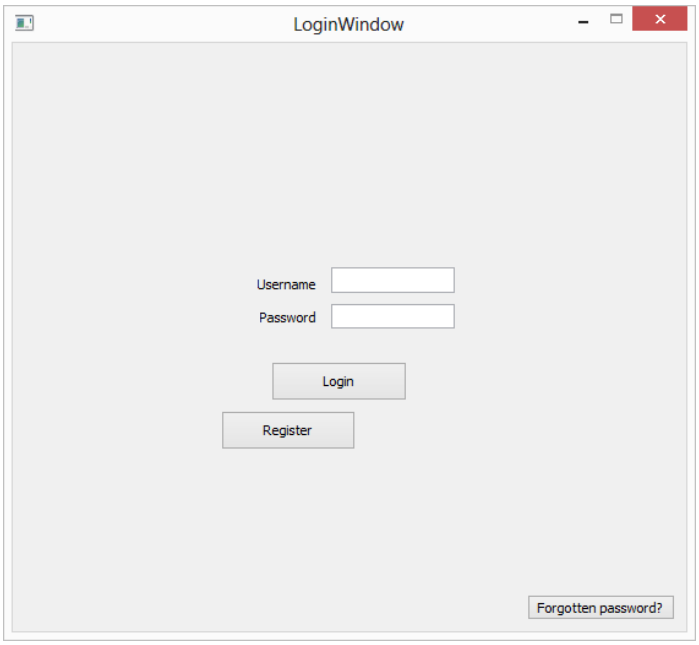
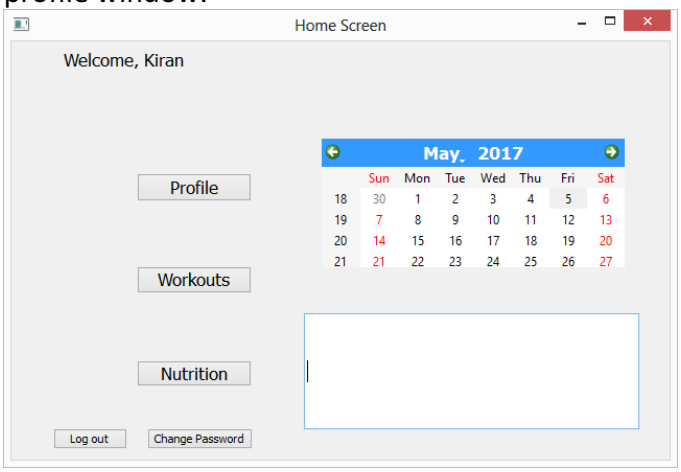
				
d	I will then click the register button and proceed on this path	Same as above	Same as above	When I clicked the register button, I navigated to the register screen, I then proceeded to press the register button. 

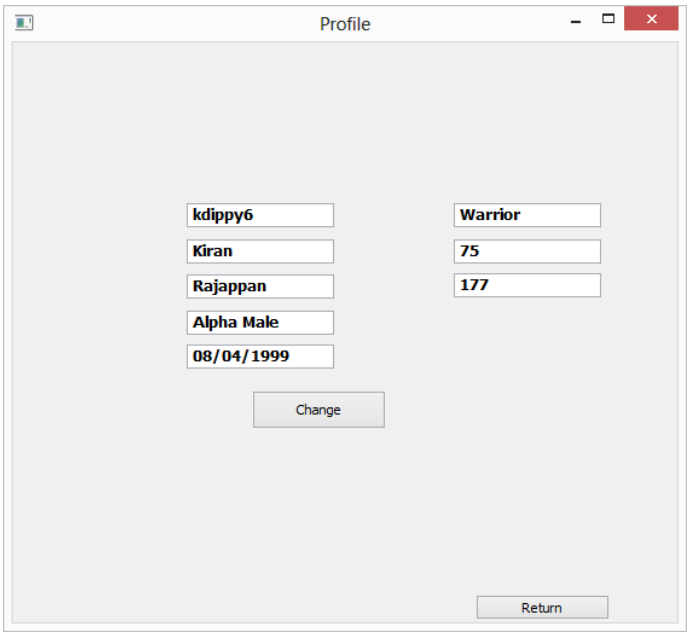
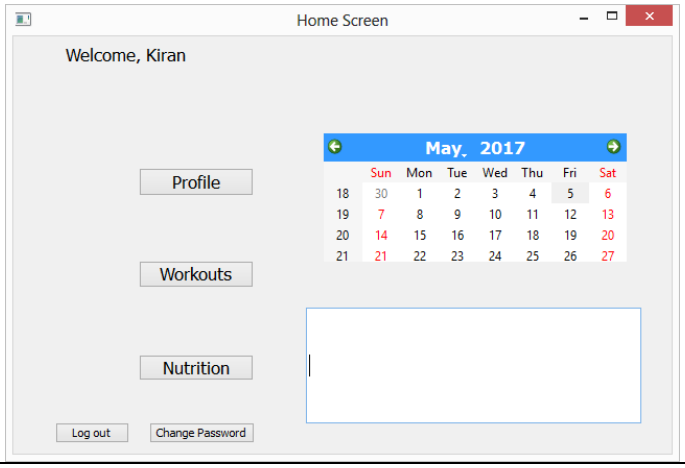
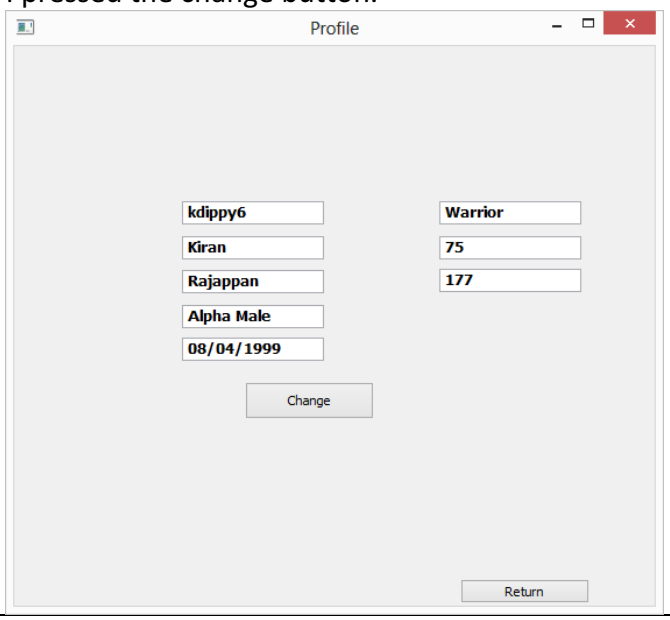
				 This popped up with multiple errors prompting the user to enter valid data. I pressed ok on the pop up box and returned to the register screen.  I then entered valid data to continue to the next screen.

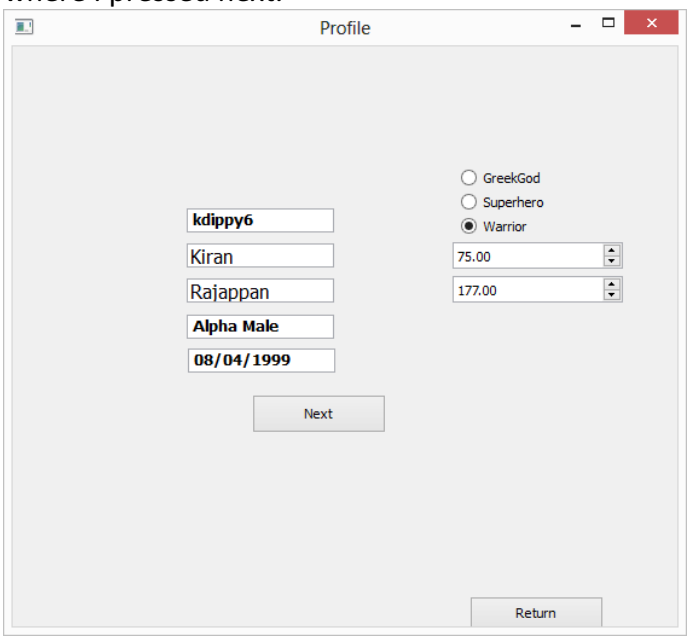
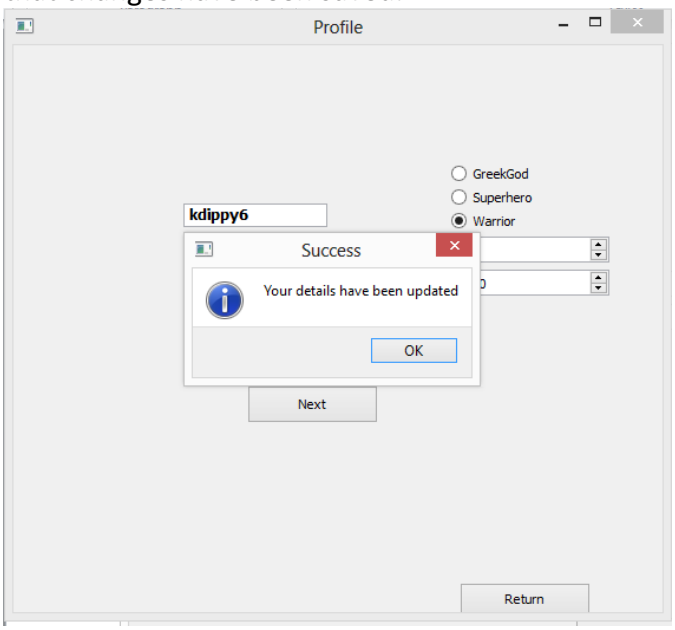
				 Once I pressed this, I reached the next window which was the continue register screen. 
e	Continued register screen – I will press the home button	Same as above	Same as above	

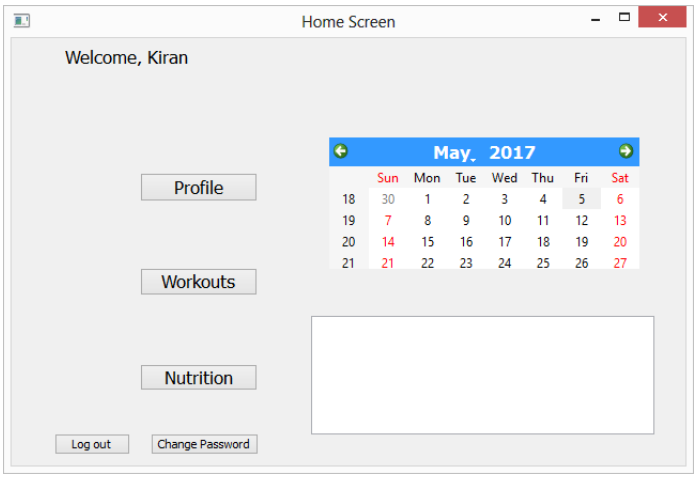
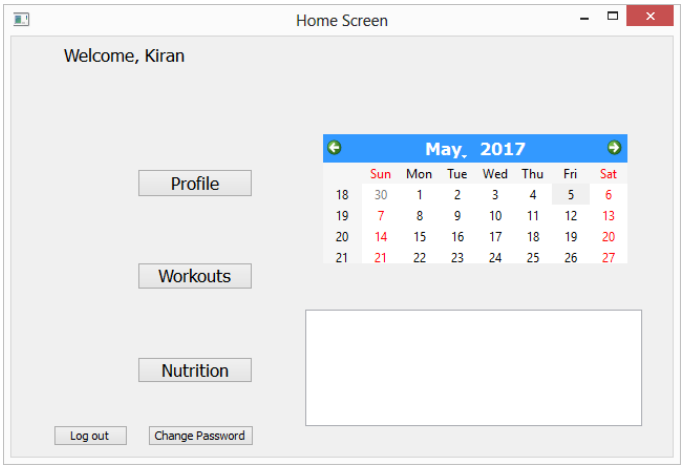
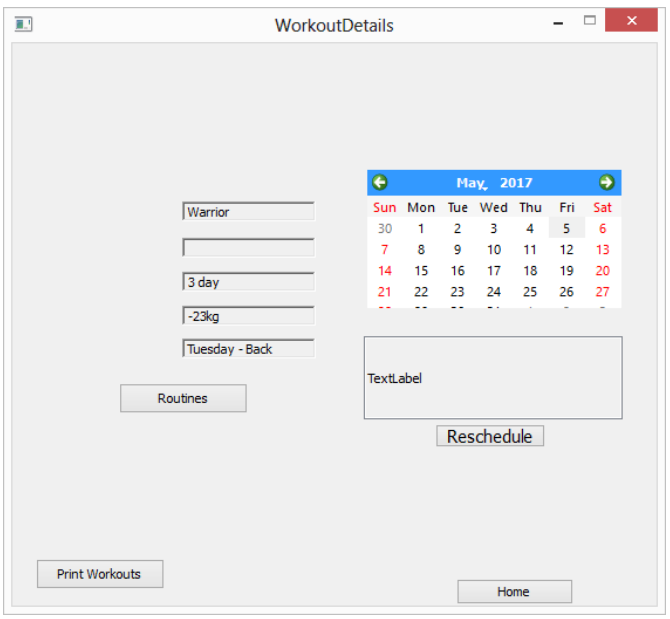
				When I pressed the home button, I returned to the login screen. 
f	Continued register screen – I will press the register button without inputs.	Same as above	Same as above	When I pressed this button, a pop up window showed, prompting user to enter valid inputs. 

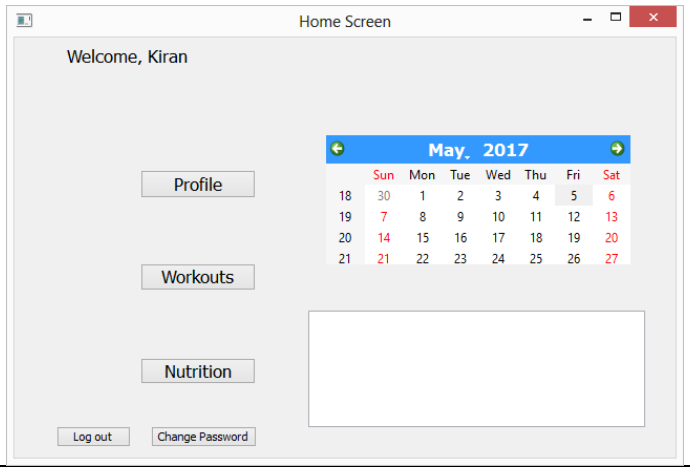
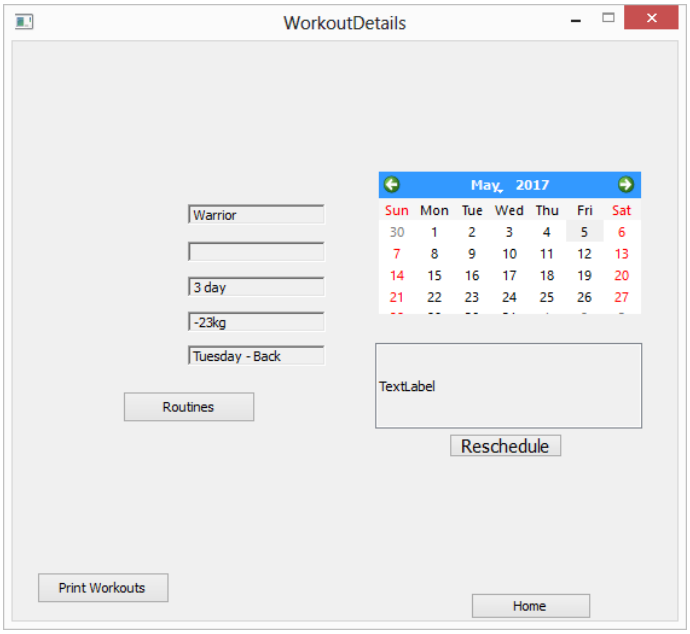
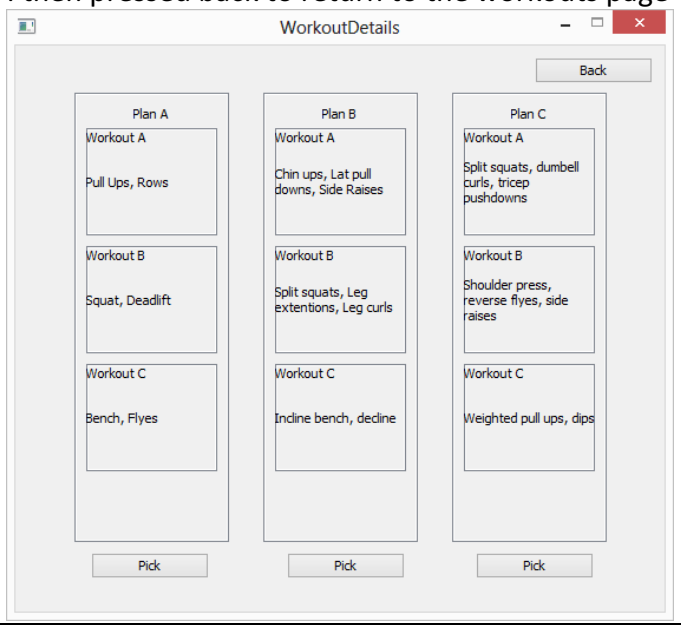
				 I then decided to fill out all of the valid data to continue to the next screen.  A pop up appeared prompting the user that they have successfully created an account. I pressed ok, and was then taken to the login screen to login again.

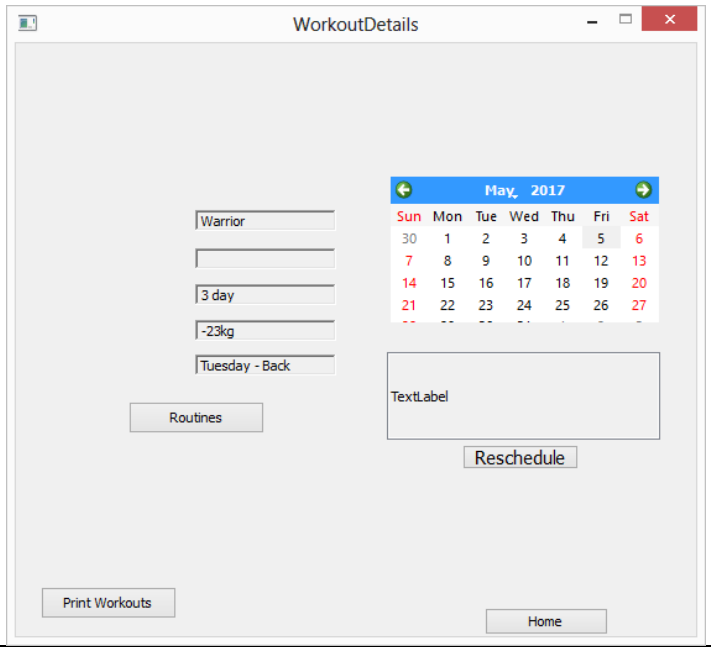
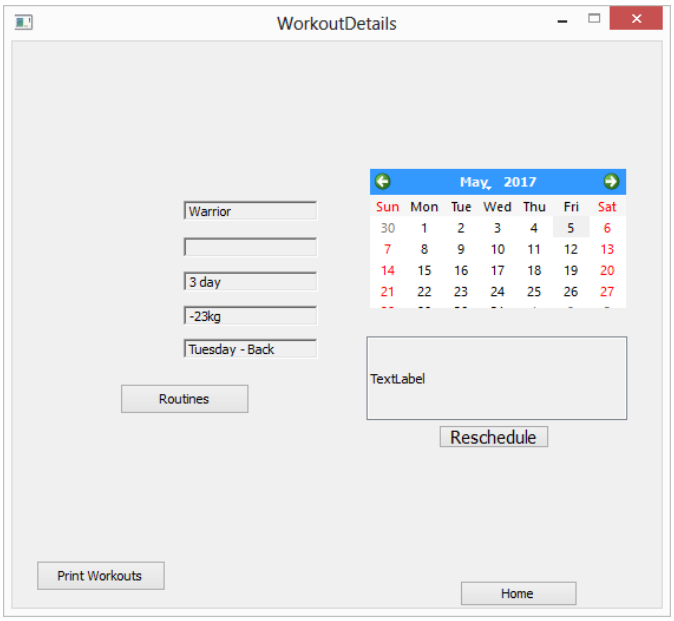
				
g	Home window – press profile button and return	Same as above	Same as above	When I pressed the profile button, I was taken to the profile window.  Once I got to this stage, I then pressed return, to get back to the home screen.

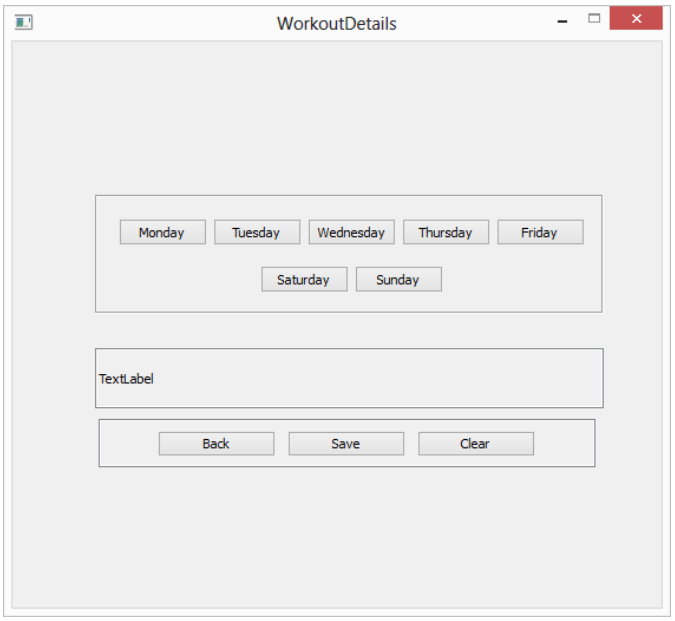
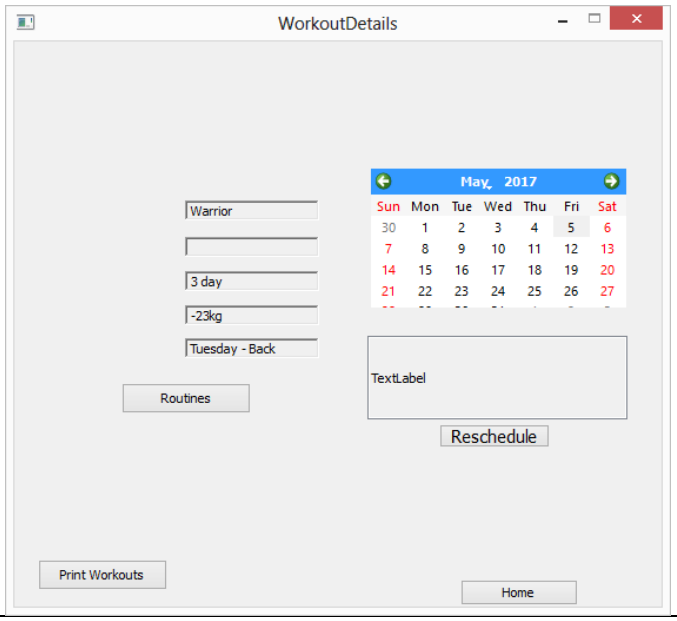
				 This then went back to the home screen. 
h	Profile screen- press change button	Same as above	Same as above	I pressed the change button. 

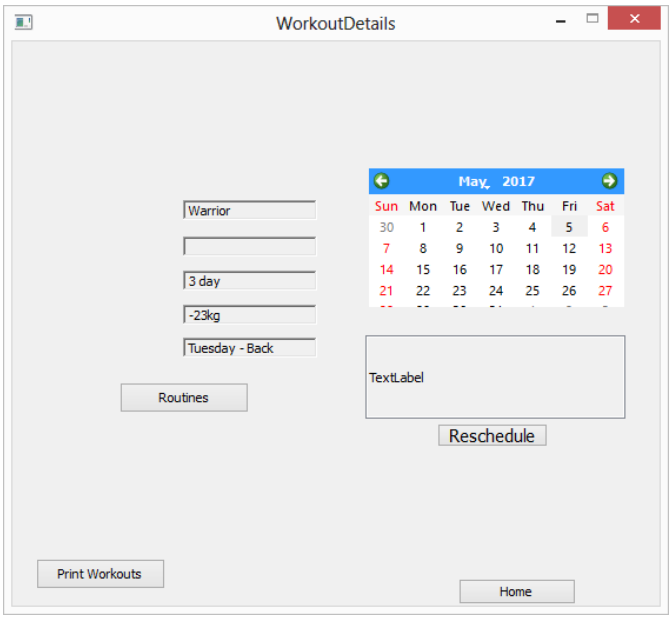
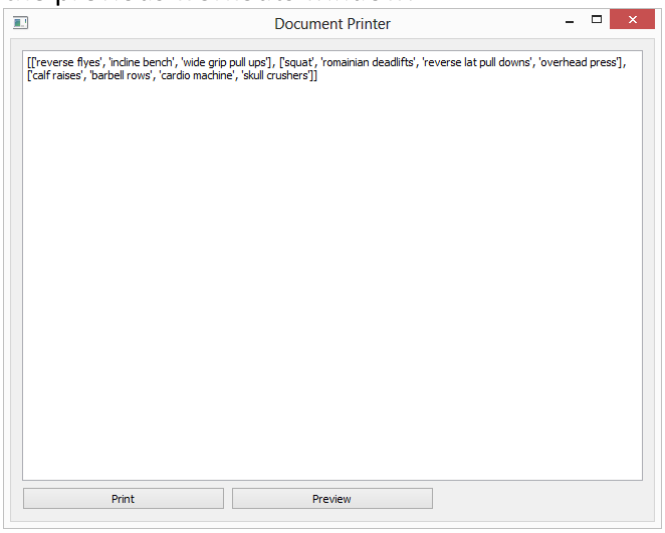
				This then took me to the profile change window, where I pressed next.  Here a pop up appeared which prompted the user that changes have been saved.  This then navigated the user back to the home screen.

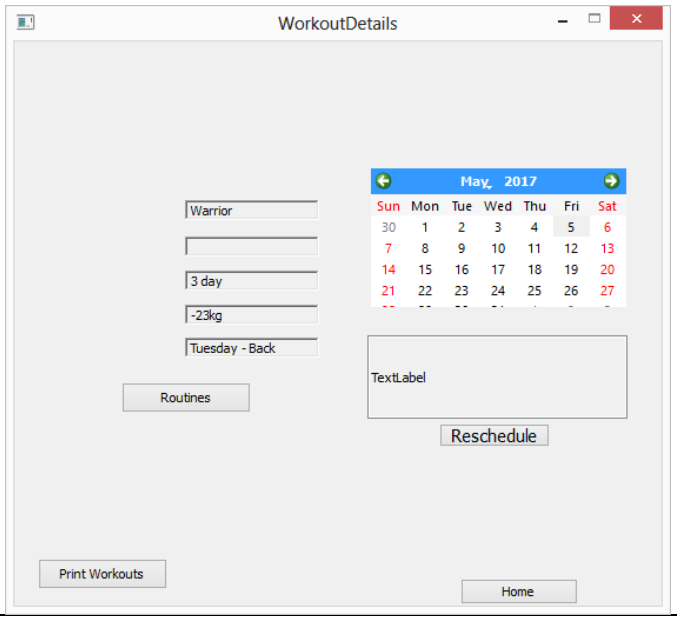
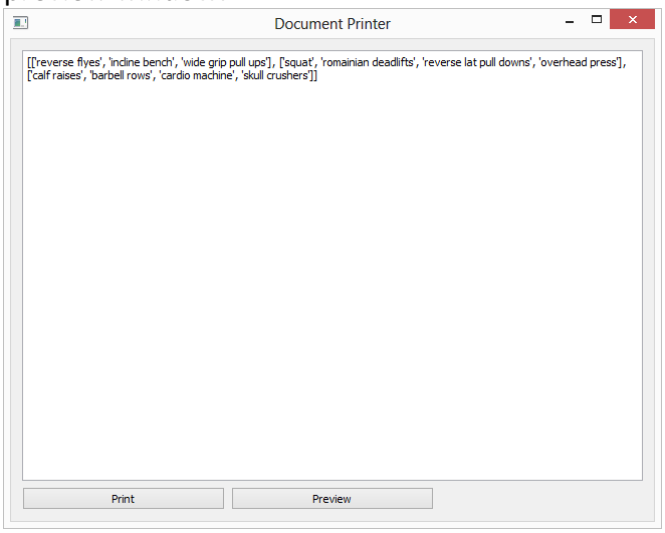
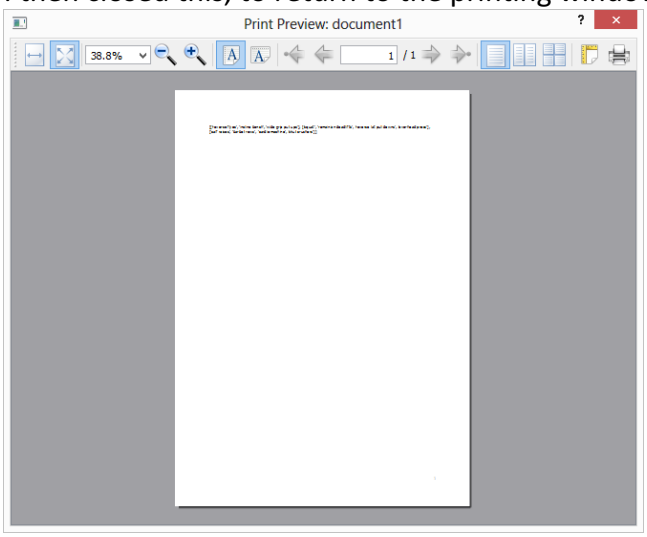
				
i	Home screen – press workouts button and return to home screen	Same as above	Same as above	I pressed the workouts button and I was navigated to the workouts screen.   I then navigated back to home by pressing the home button.

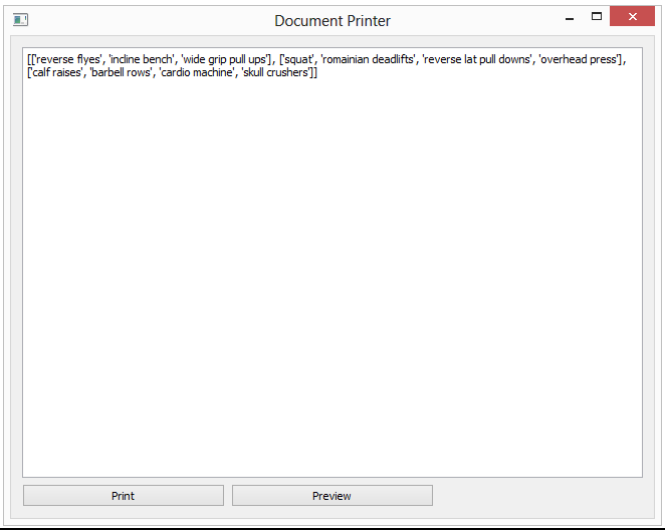
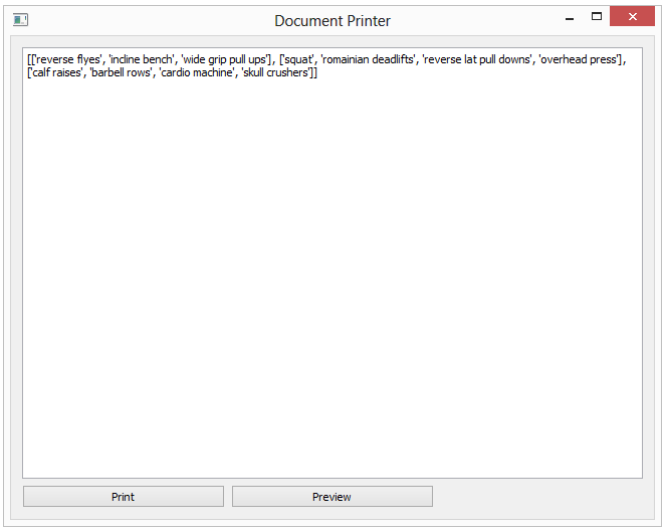
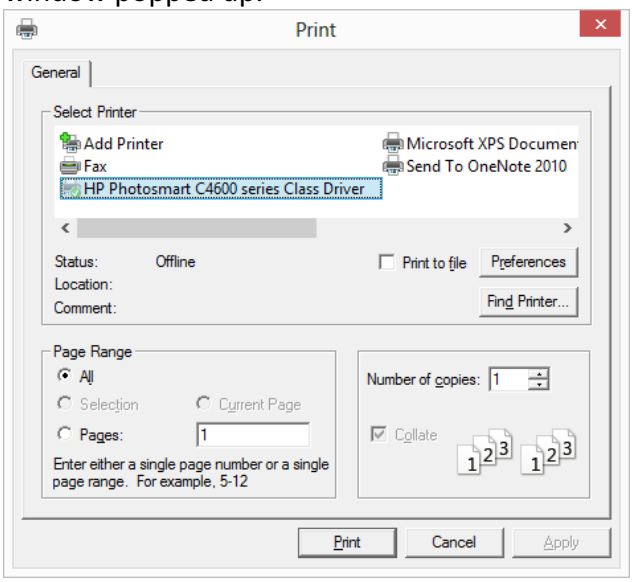
				
j	Workouts screen – press routines button and back	Same as above	Same as above	I pressed the routines button and navigated to the correct window.  I then pressed back to return to the workouts page 

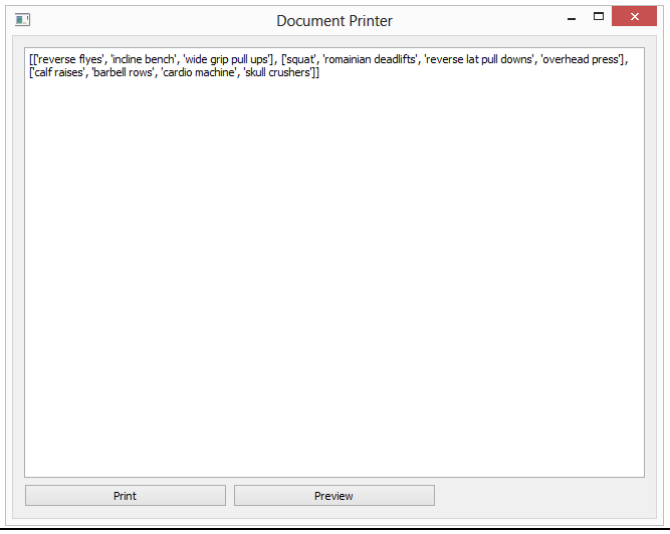
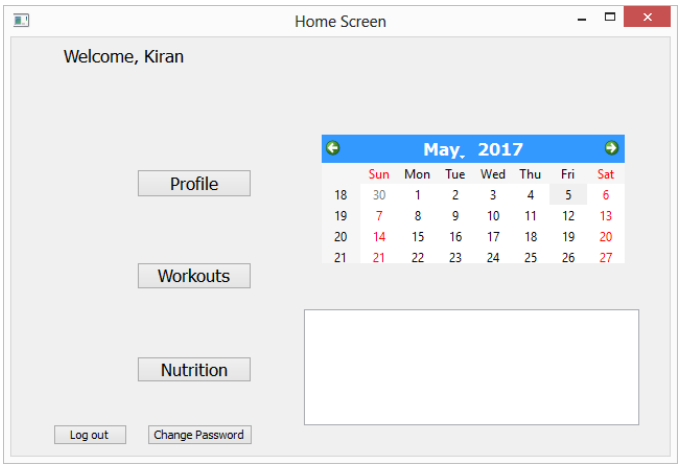
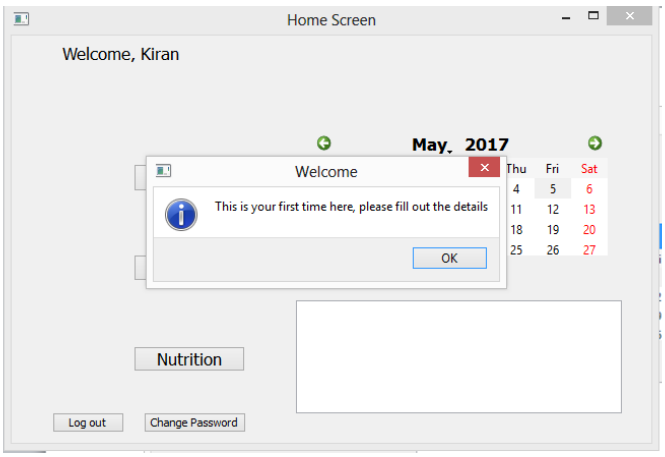
				
k	Workouts window – press the reschedule button and back	Same as above	Same as above	I pressed the reschedule button and navigated to the rescheduling screen.  I then pressed the back button, to return to the workouts screen.

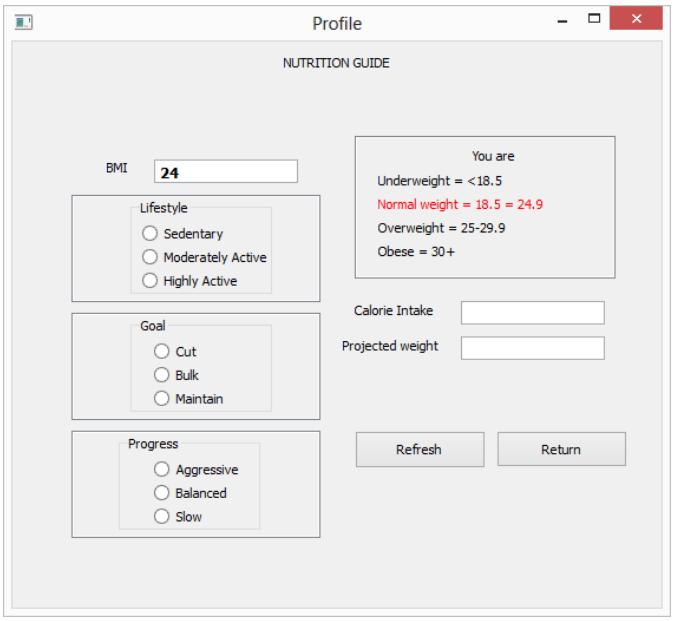
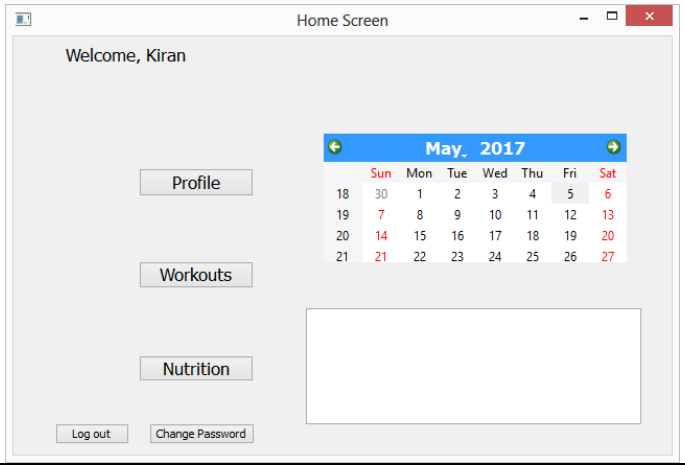
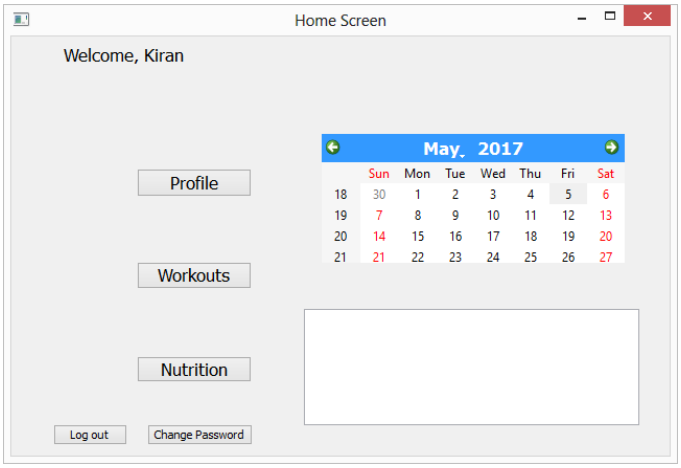
				 When I pressed the back button, I returned to the workouts screen. 
l	Workouts screen – press the print workouts button and exit.	Same as above	Same as above	When I pressed this, a print screen popped up.

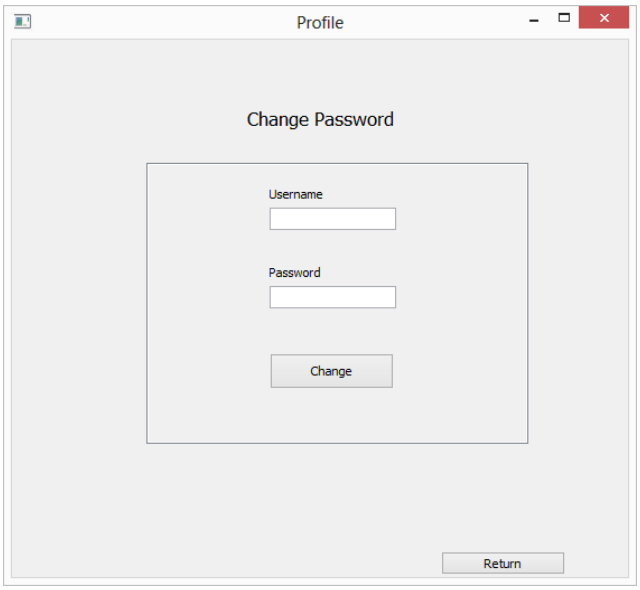
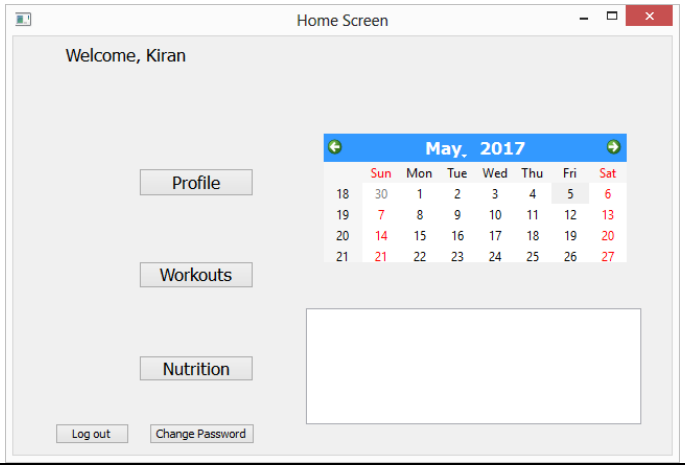
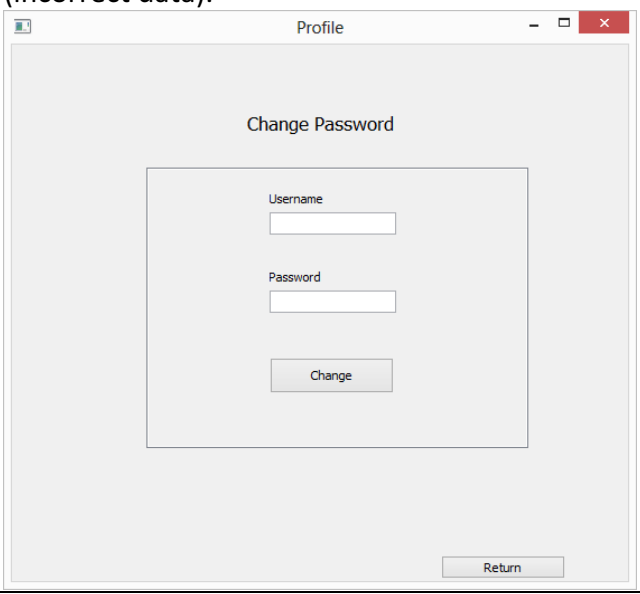
				 There was no visible back button to press, however, it is only a pop up window and can be closed to reveal the previous workouts window.  This returns to workout window.

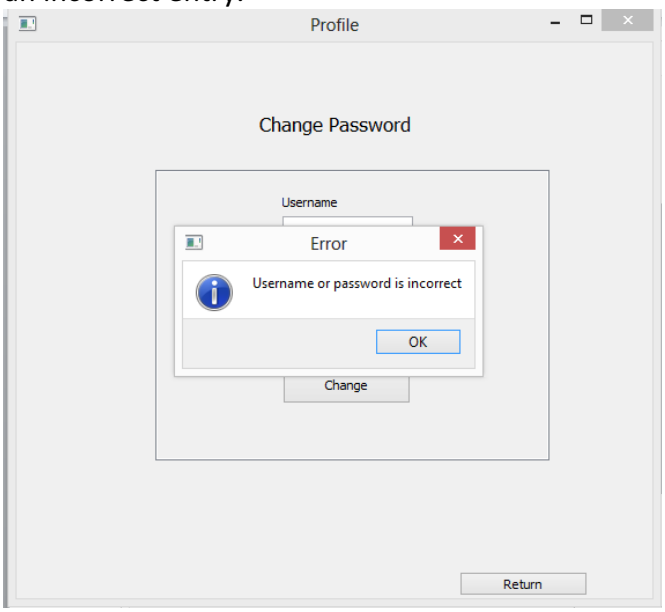
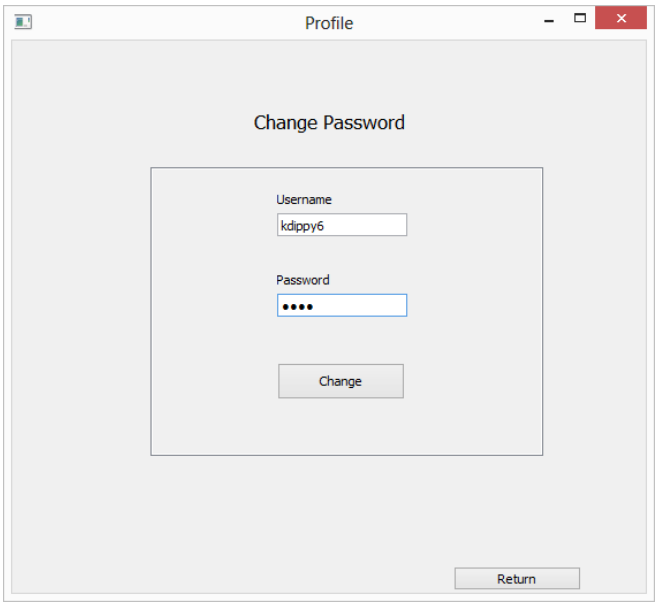
				
m	Printing screen press preview window	Same as above	Same as above	When I pressed this, this popped up yet another print preview window.  I then closed this, to return to the printing window.  This then returns to the printing window.

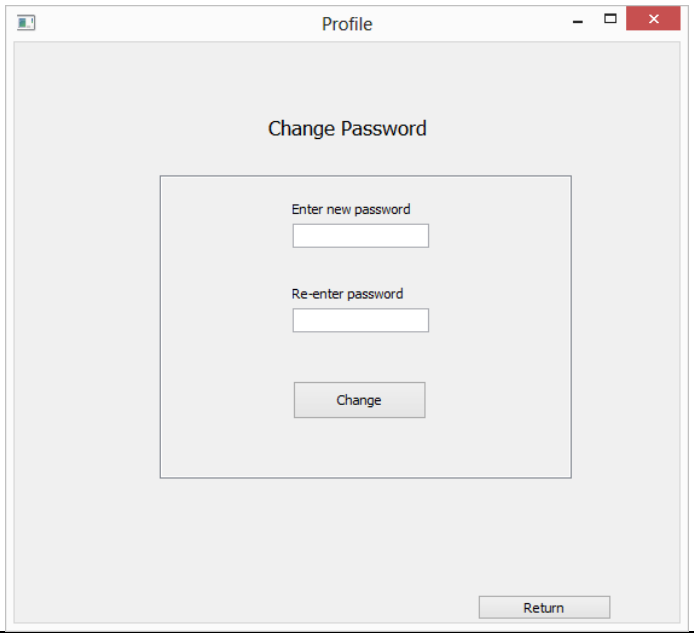
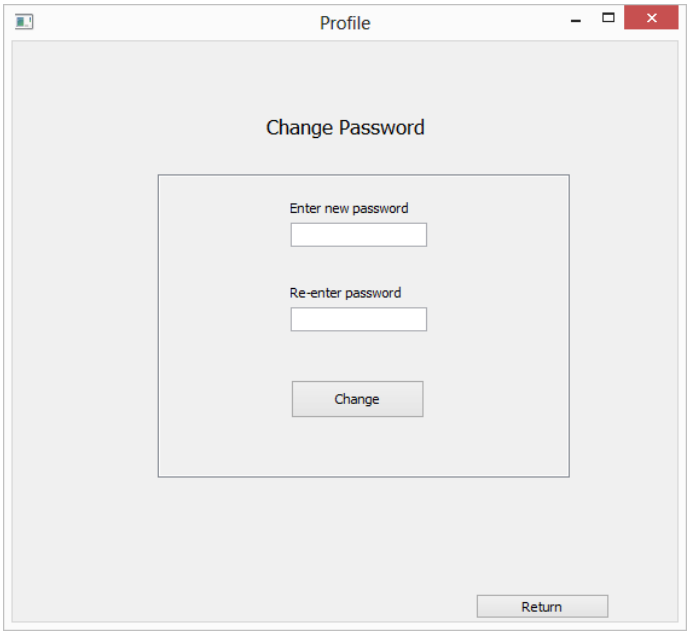
				
n	Printing window-press print button and back.	Same as above	Same as above	 When I pressed the print button, the generic print window popped up.  I then cancelled this, to return to the printing window, if I pressed the print button, this should actually print the page.

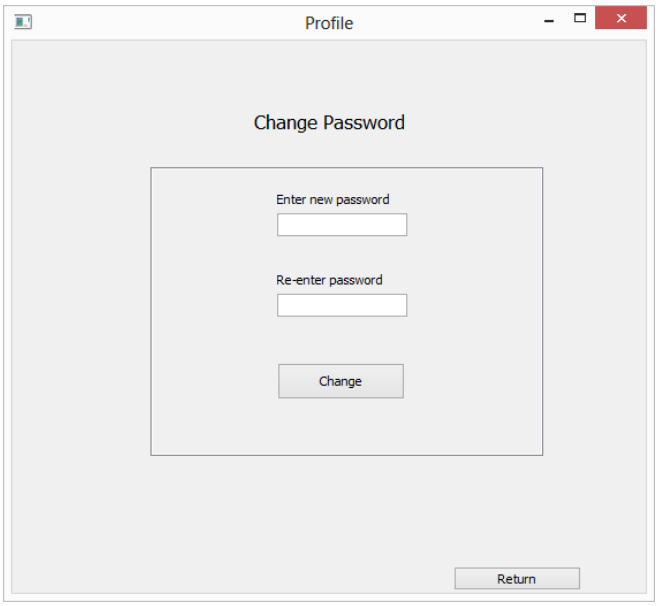
				
o	Workouts window- press nutrition button and return	Same as above	Same as above	 I then clicked the nutrition screen. This popped up with the nutrition screen window.  As this is the first time this user has reached this screen, a popup appeared. I then pressed ok which navigated to the nutrition screen.

				 From here, I pressed return to navigate back to the home window. 
p	Home window- press change password button and return	Same as above	Same as above	 From here, I pressed change password, which navigated to the change password window.

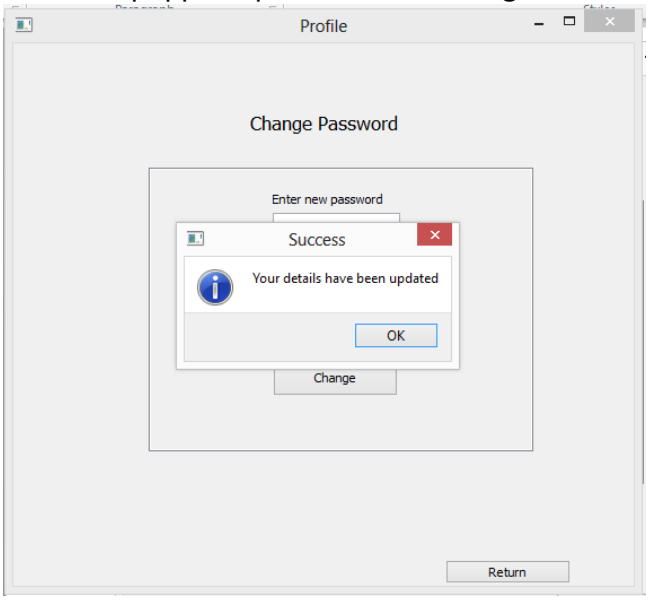
				 I then pressed return, to return to the home window, which was successful. 
q	Change password screen – press change	Same as above	Same as above	I first pressed this button without inputting any data (incorrect data). 

				This returned an error report for the user, prompting an incorrect entry.  From here, I then pressed ok and inputted correct data.  This then navigated to the actual change password screen.

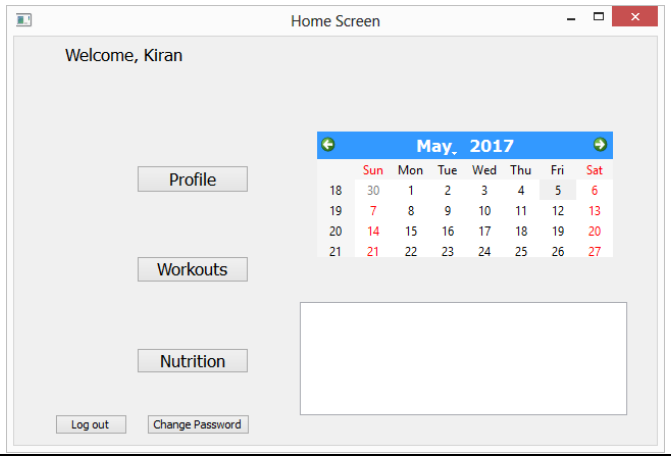
				
r	Acutal change password screen – press change, then return.	Same as above	Same as above	 I pressed change without any data and pop-up windows showed, prompting the user to input correct data. I then inputted correct data.

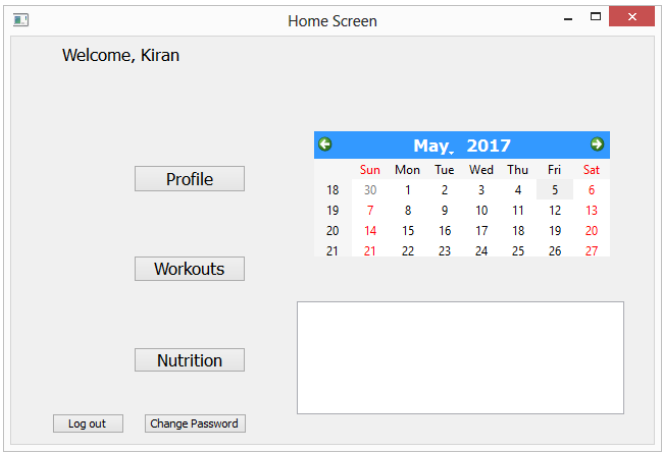
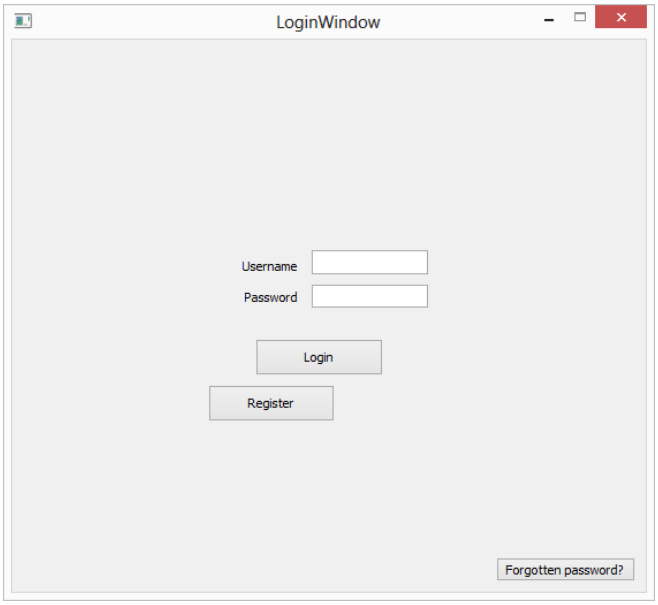
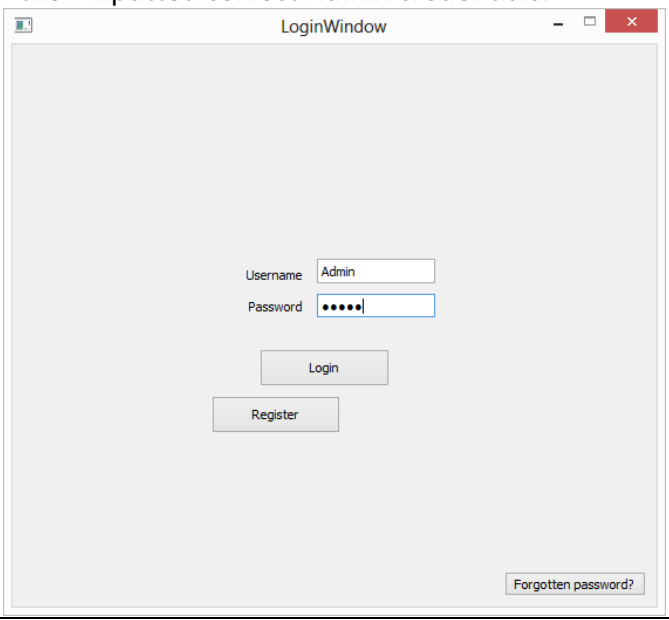


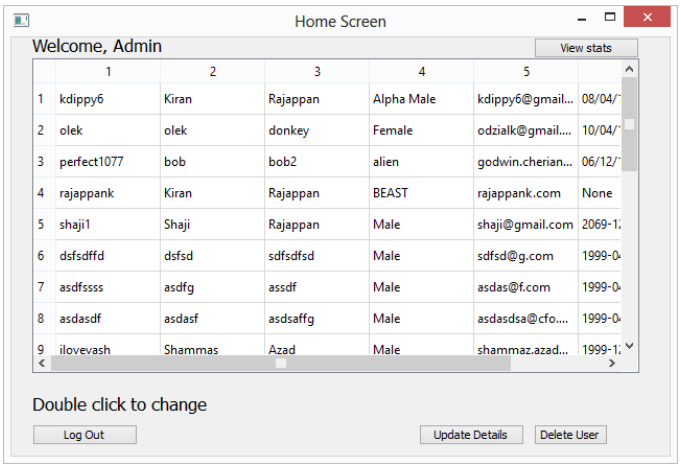
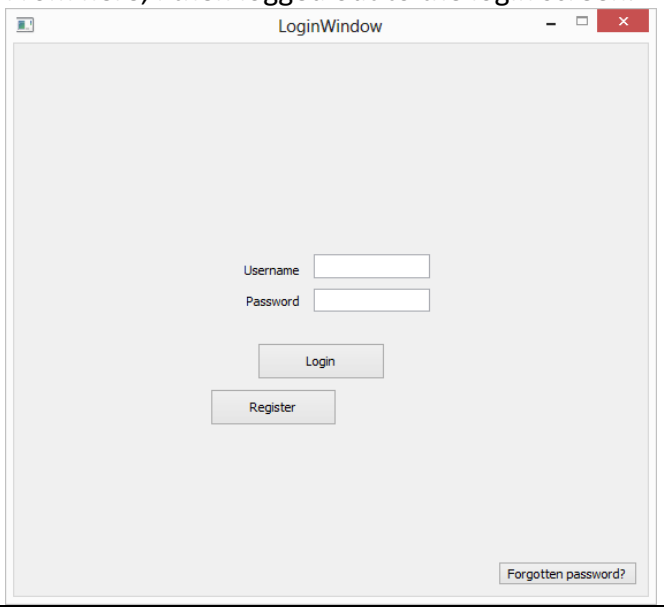
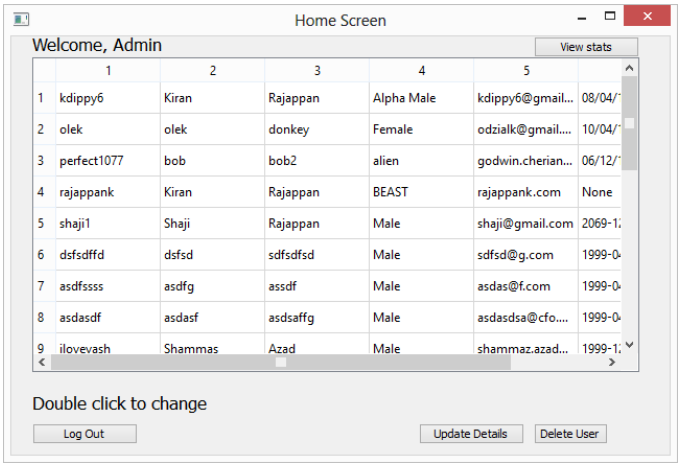
This then popped up a successful change window.

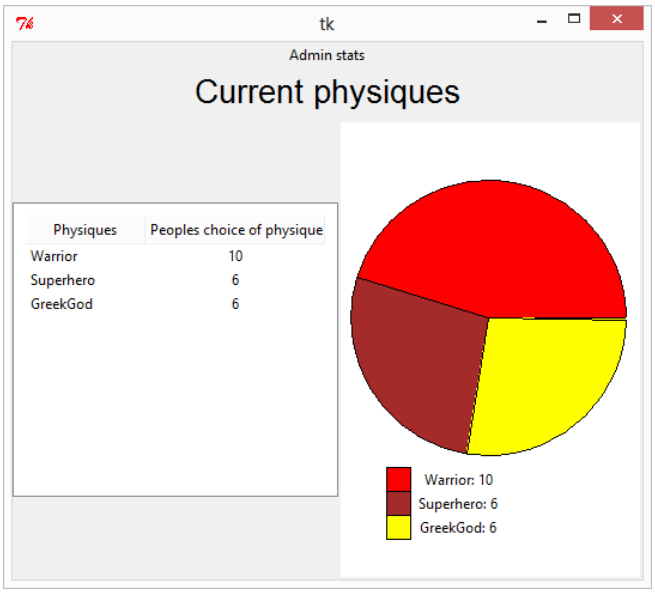
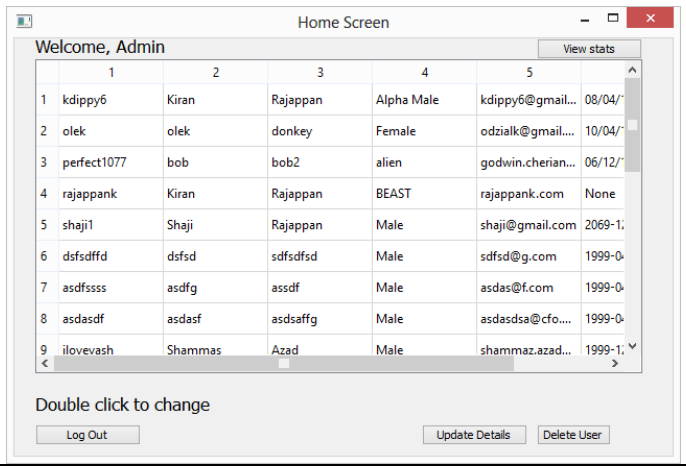


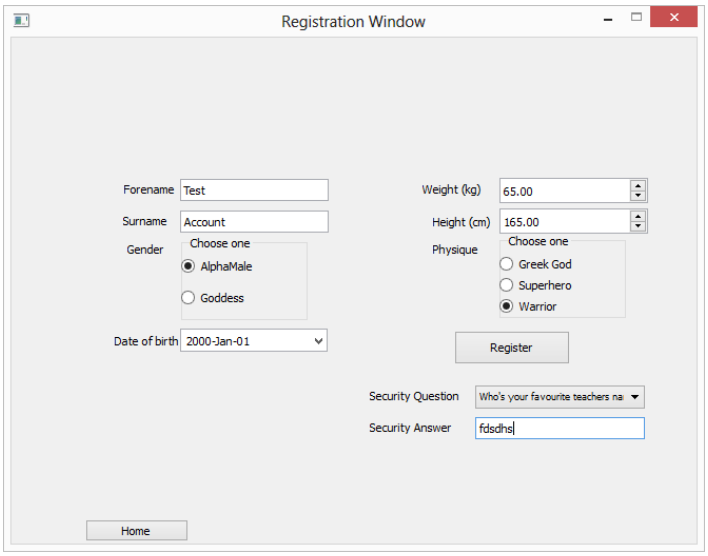
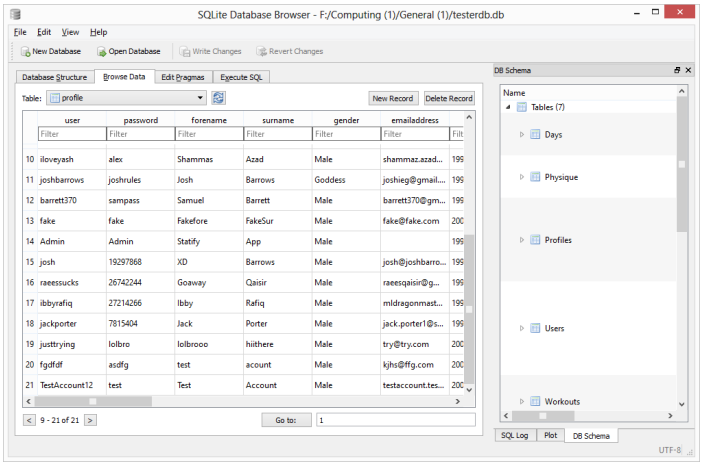
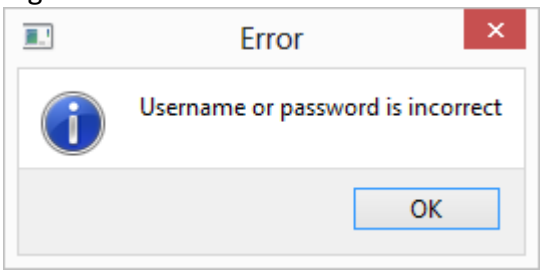
From here, I pressed ok, which returned me back to the home screen.

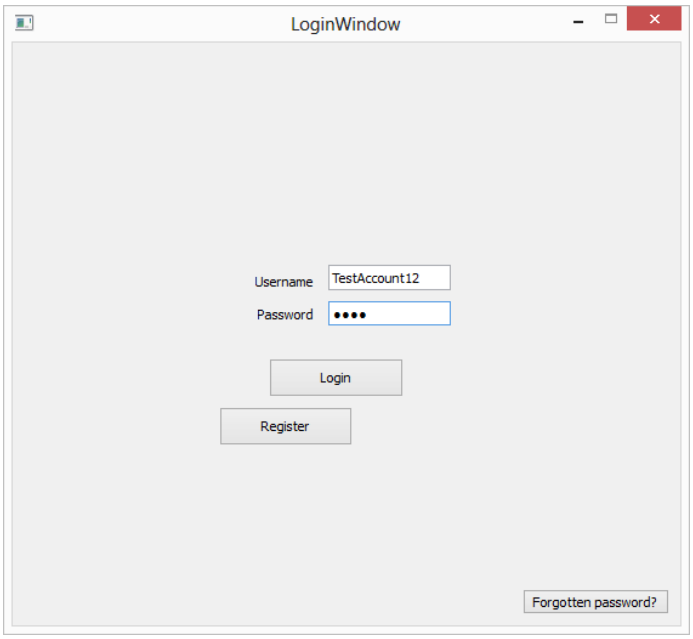
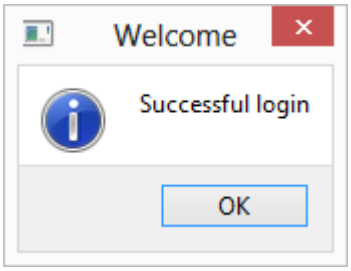
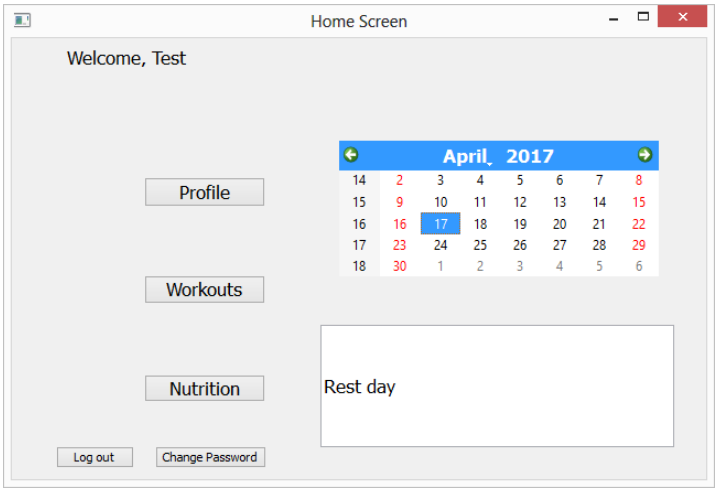


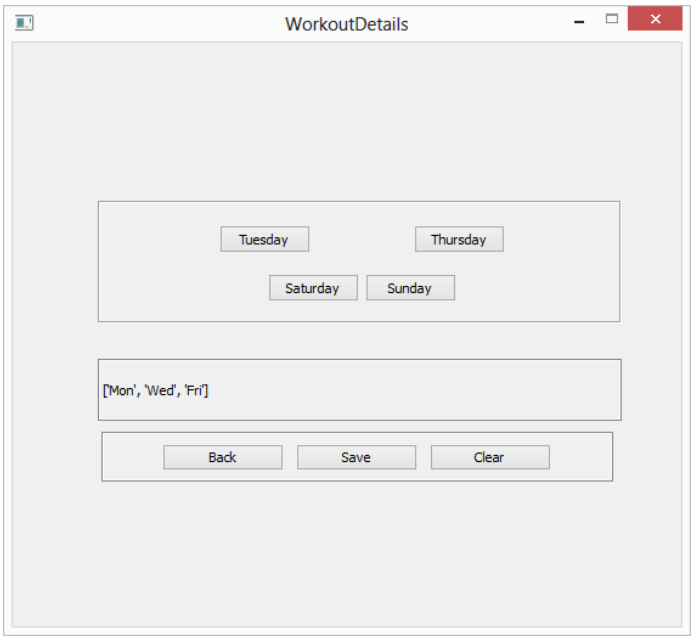
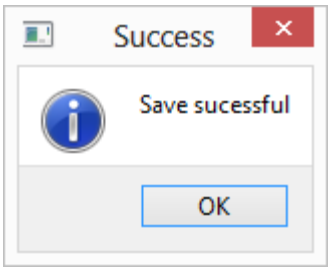
s	Home screen- log out and login as Admin			 When I logged out, I was navigated to the login screen with all cleared data.  I then inputted correct Admin credentials. 

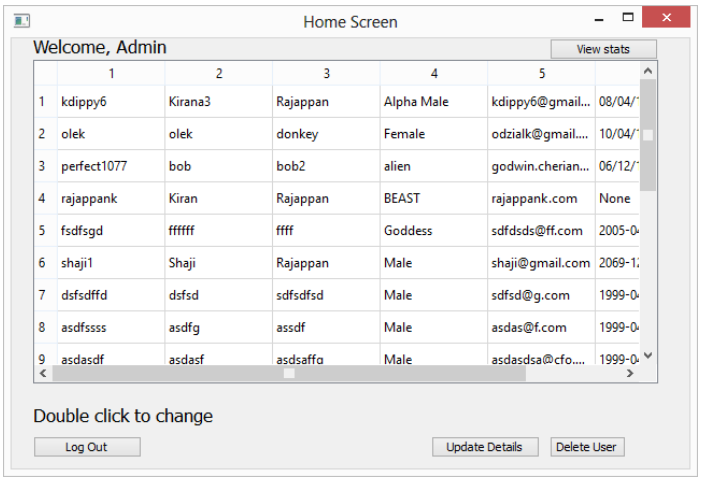
				From here I was navigated to the Admin home screen.  From here, I then logged out to the login screen. 
t	Admin screen – view stats button and return	Same as above	Same as above	 when I pressed this button, a separate stats window popped up.

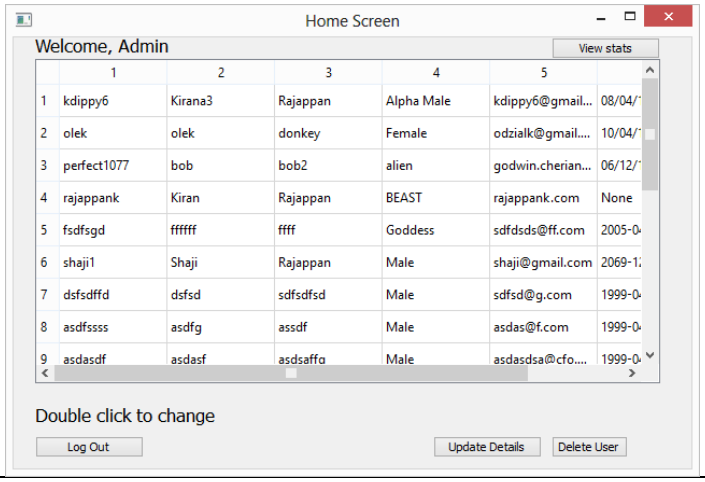
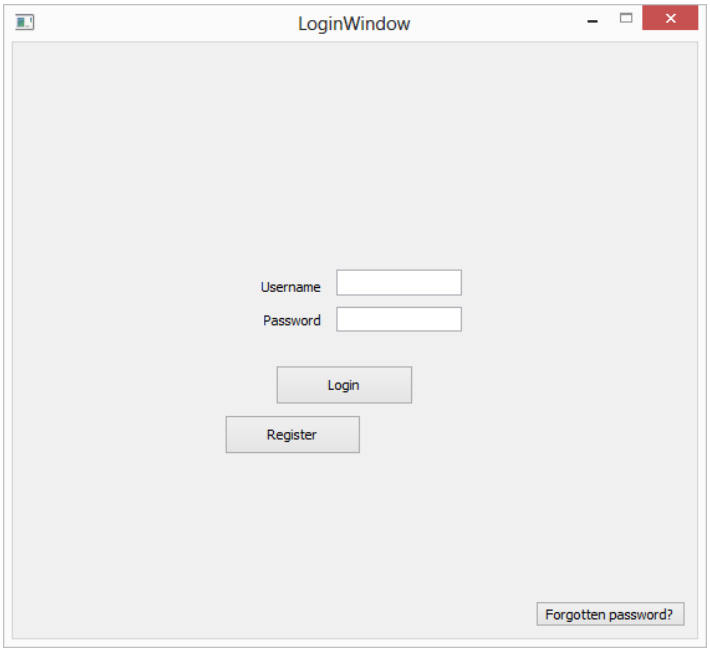
				 As this is a separate window, this included no back button but simply close the screen to return to the admin screen. 
2	Repeatedly press all buttons on page	This will be tested by allowing Olek, and myself to spam all of the navigation buttons and try to break program.	The expected output, is that for the program to have no errors and navigate through all windows.	None of the buttons caused a crash in the program. When testing criteria 1, I ensure I spammed the buttons while progressing through to see if any errors showed. This however, did not occur.
3	User registers an account on program	This will be tested by allowing a new user to the program, to try and gain access. They	The outcome of this, assuming the user has not already set up an account or an already existing account with the	I used a test account with name test account and random information. I then checked sql to check if it saved the account.

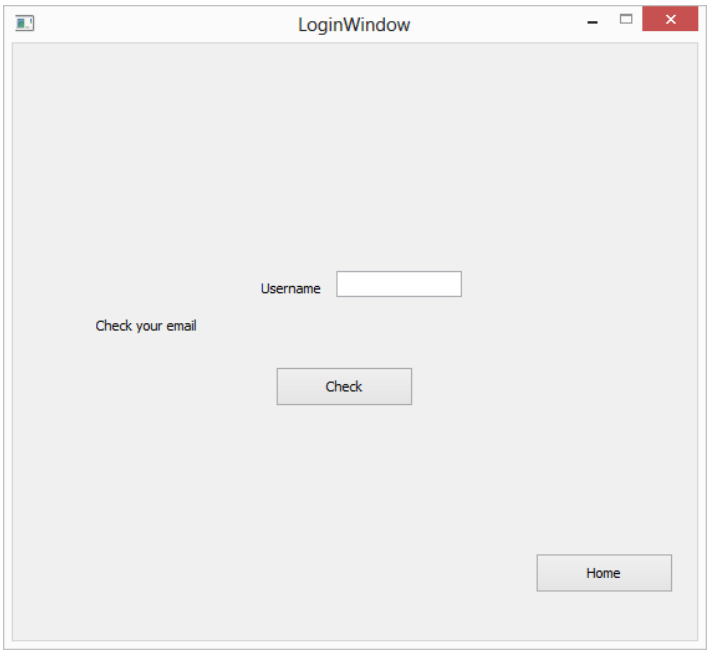
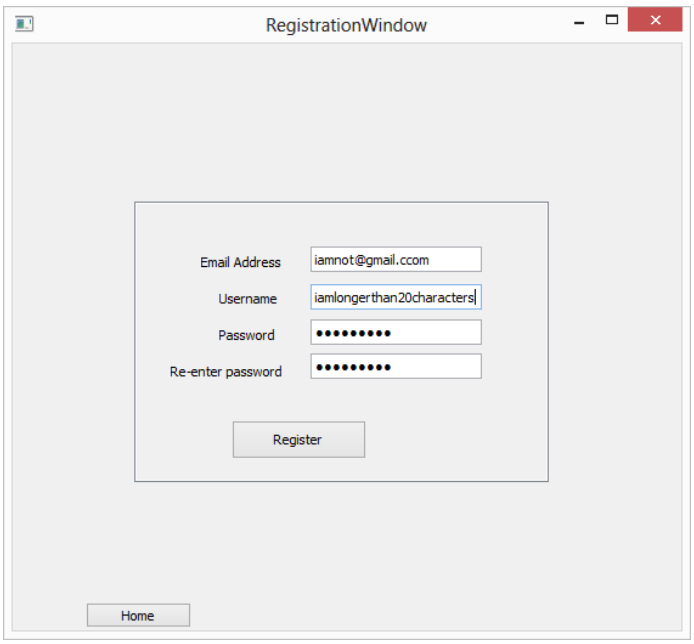
		will try logging in with details.	username should show an error with username/password not valid.	  This shows that I registered with the following account details. I then went into SQL and searched for an account and outputted all the data retrieved from the account.
4	User must log on account with provided username and password	This will be tested by inputting the same test data from previous test to ensure user now has access to the program after registering.	The outcome of this, should be access to the profile. With the correct details entered.	Using the same account as I registered with previously, I tried to log in with the same details. Before I made the account, I tried logging in with the same username and password and ran into the error login details wrong. However, once I registered and logged in again with the same details, I was able to login. 

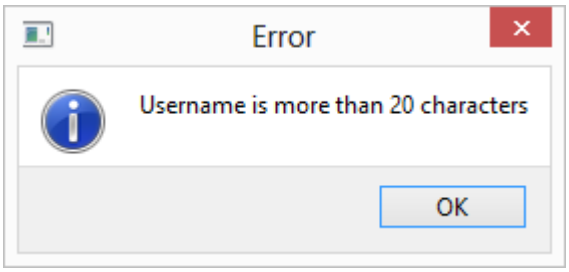
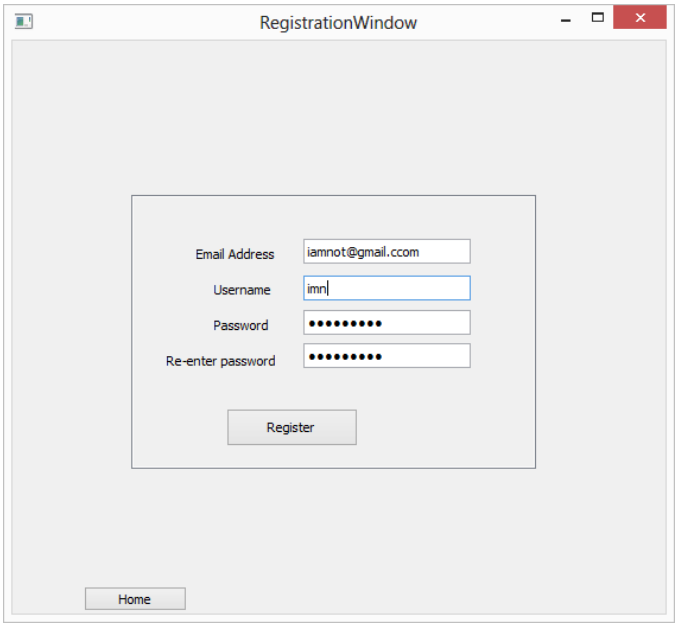
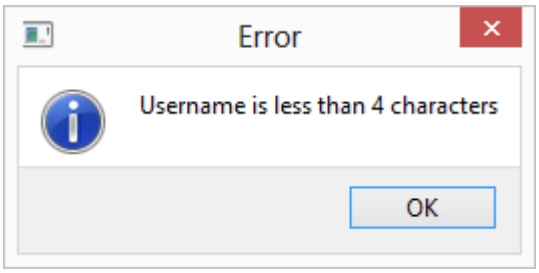
				 
5	Allow user to pick the days that they wish to workout on	This will be tested by inputting three days to work out on. This will then be checked after logging out, then back in again.	This should then show the workouts that I have for these days in order when I click them on the calendar.	With the account that I just created, I will pick the days Monday, Wednesday and Fri, to see if I can pick certain days. 

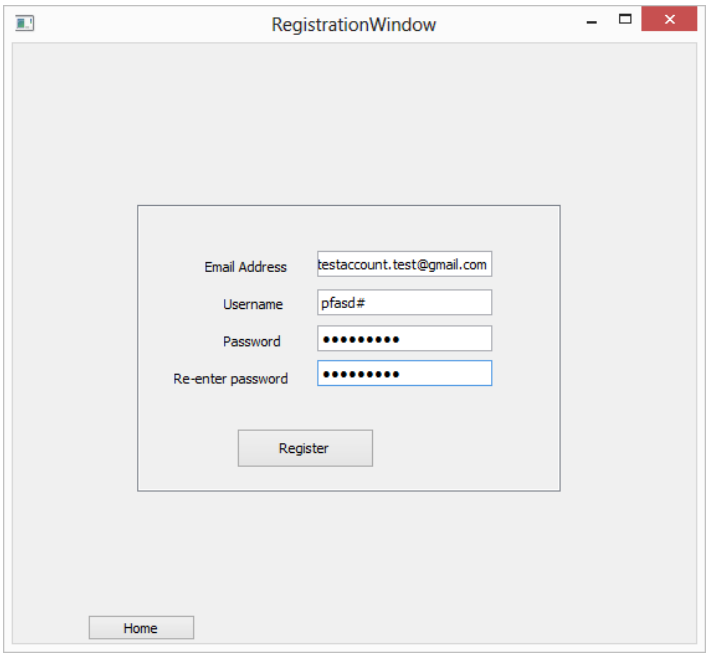
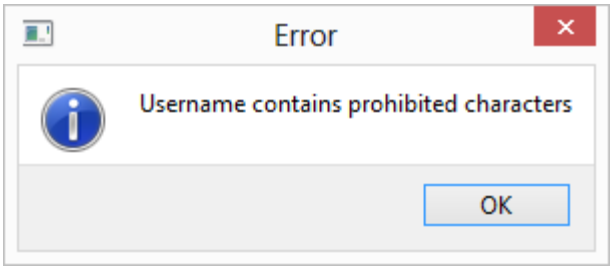
				  This shows that I have picked Monday, Wednesday and Friday and it was successfully changed.
6	Provide admin screen to maintain	This will be checked by logging into regular accounts in the same login screen, then inputting an admin username and password	When the user enters normal account usernames and passwords, it should enter to their own individual profiles. When Admin credentials are inputted	I will try to login to an Admin account, to try and access the Admin screen. This will be with the account Admin and password Admin, which should navigate to the Admin screen and not a regular user screen.

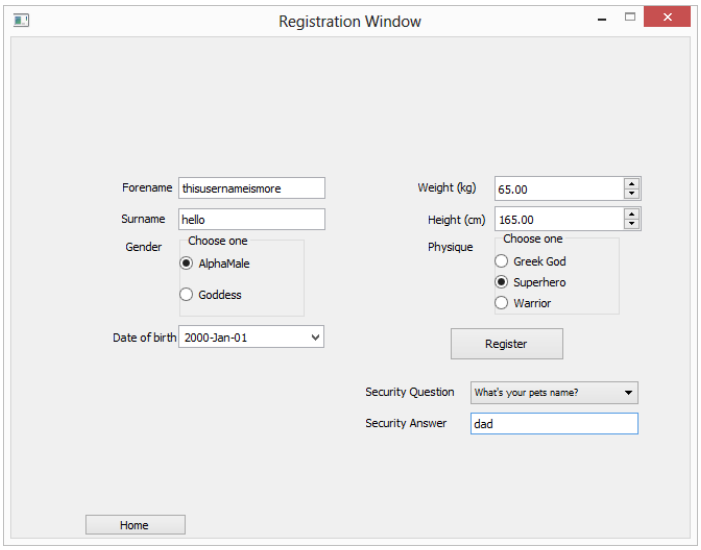
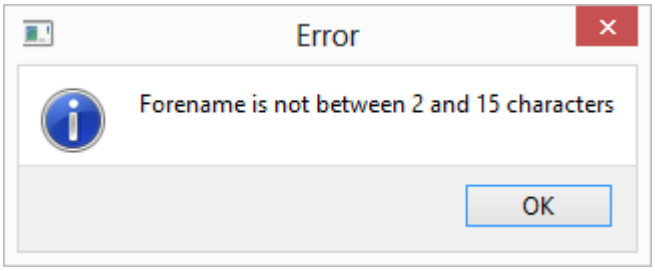
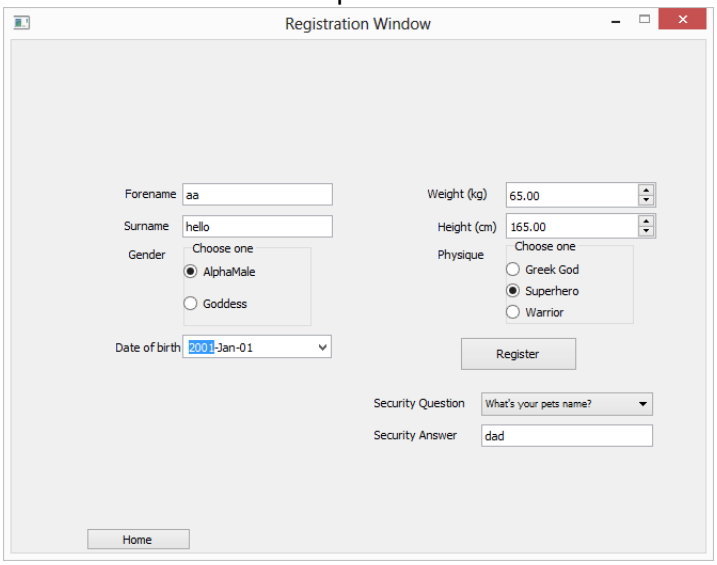
				
7	When user requests to change password, make sure password hash is the same as db hash	This will be tested by trying to print the password and read from the database which it is saved to, whenever a password is required.	The expected output, should be that every password input should return with a hashed version. The only person that should see the original password is the user. In any other case this should return the hash.	
8	Ensure all windows have back buttons for user to navigate	I, along with Olek, will navigate through every screen to try and find a dead end, to see if it is possible to not be stuck	There should not be any dead ends. For every screen there is a returning button to either the previous screen or the home screen	I went through each of the screens to check if they all include a back or home button to ensure the user did not have any dead ends, to which I did not find any.

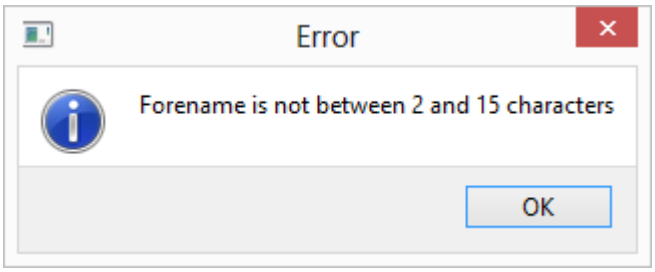
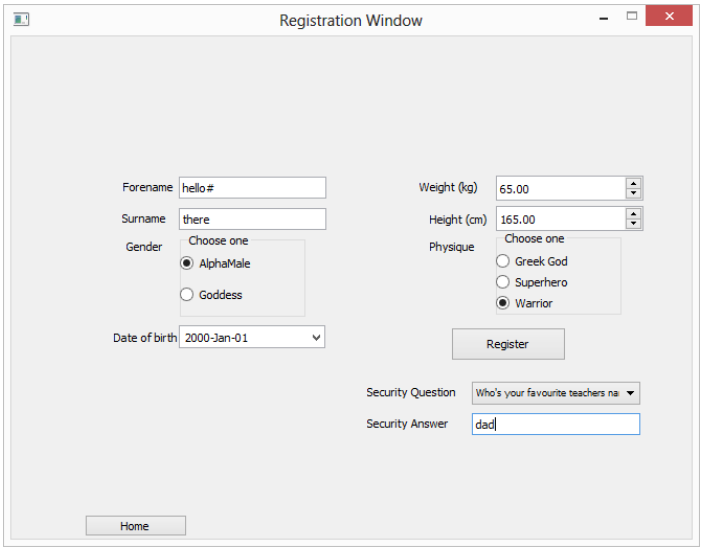
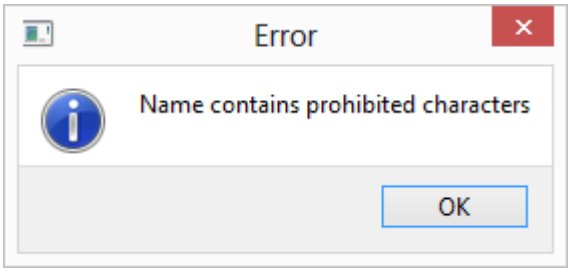
9	Ensure admin can access admin screen	This will be checked by logging into regular accounts in the same login screen, then inputting an admin username and password	When the user enters normal account usernames and passwords, it should enter to their own individual profiles. When Admin credentials are inputted	I previously stated that there was an Admin screen accessible. This shows that an Admin can access this screen. I gave Olek the correct credentials to try to login to the Admin screen and we successfully able to do so. 
10	Have access to reclaim password if forgotten	This will be tested by trying to reset the password and trying to regain access.	When the user requests password change, when entering the correct password to the security question, should reset their password with a code, which is sent to their email address which is set up with their email.	This shows that the user has access to the forgotten password function on the home screen. This also shows the window that pops up when the button is pressed. 

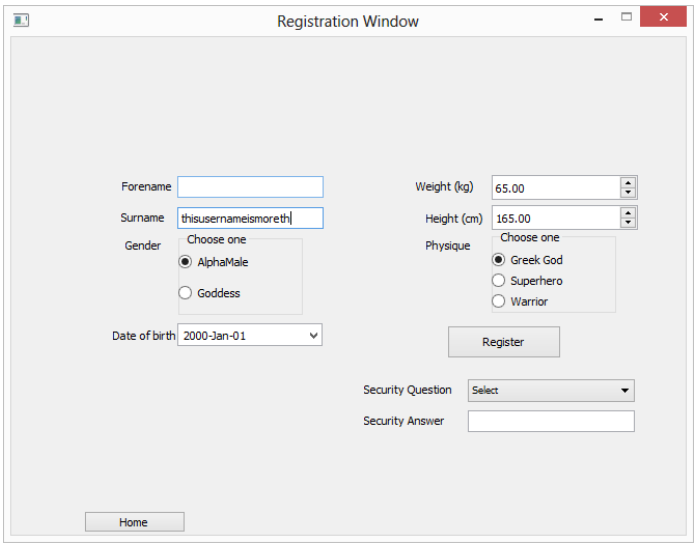
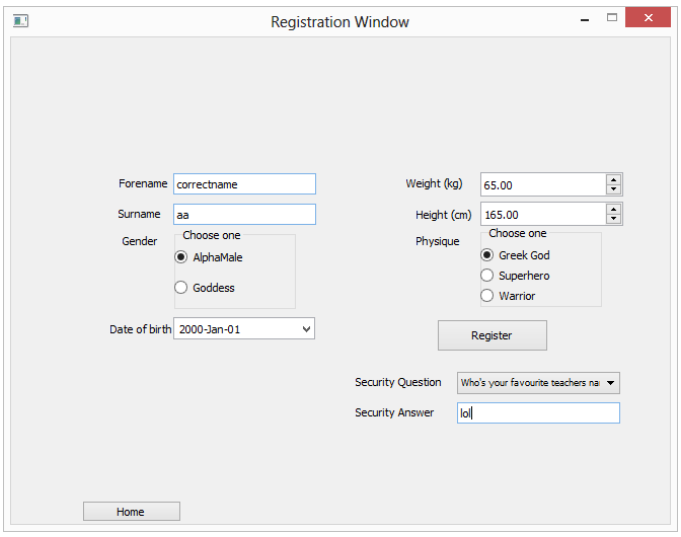
				
11	Enter username longer than 20 characters	When registering an account, I will try to input a username that is 21+ characters long.	When I try to enter this username, the program should return an error and not accept the username.	When I entered the username Iamlongerthan20characters the program reported an error showing that the username entered is not the correct length. 

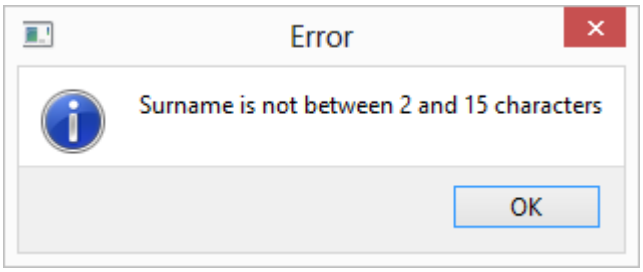
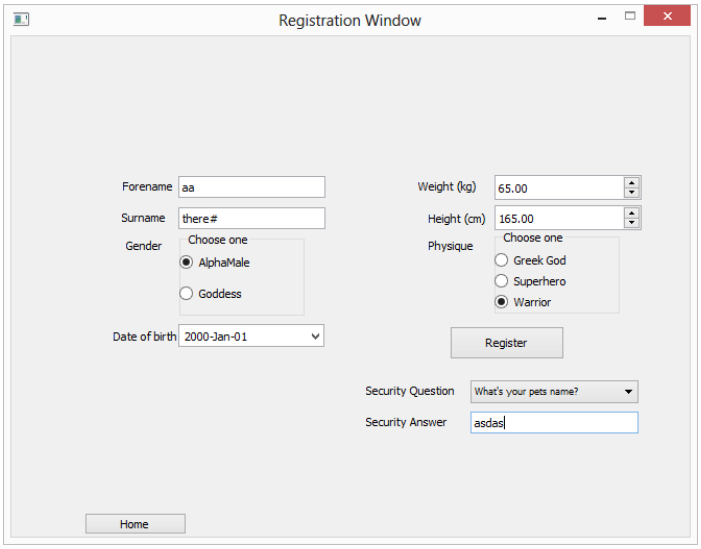
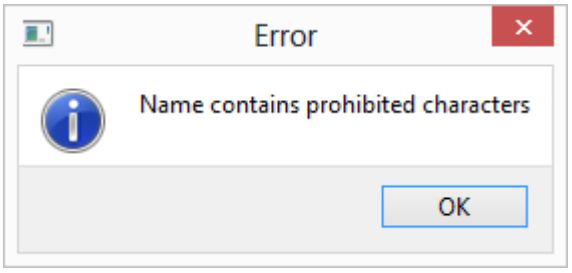
				
12	Ensure username is longer than 4 characters	When registering an account, I will try to input a username that is less than 4 characters long.	When I try to enter this username, the program should return an error and not accept the username.	When I entered the username imn the program reported an error showing that the username entered is not the correct length.  
13	Ensure username does not contain prohibi	To test this, I will enter a username with some prohibited characters	The program should return an error showing that the input contain prohibited characters	I will enter the username pfasd#, this contains the prohibited character # and so will report an error.

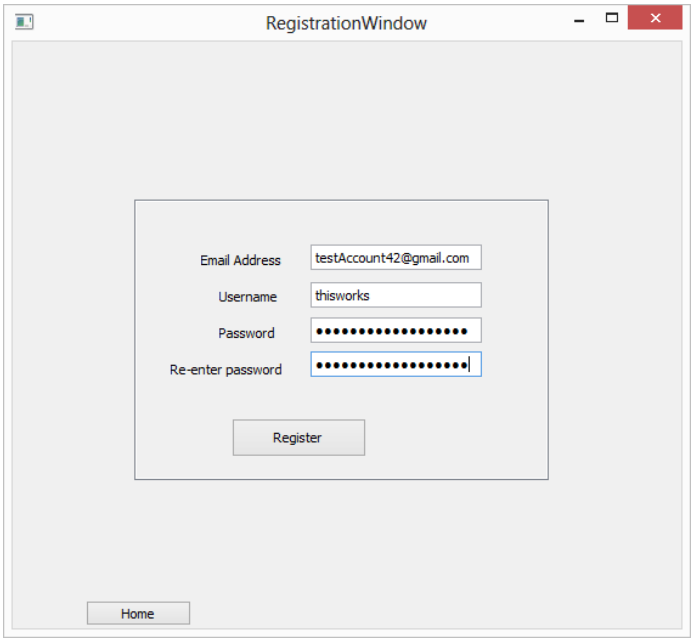
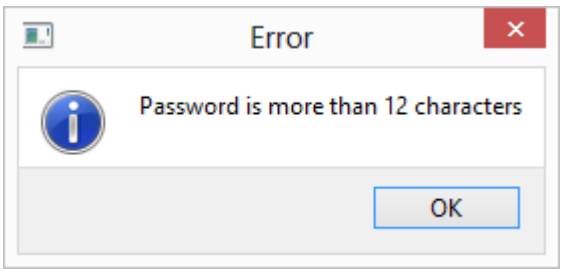
	ted characters			  This shows the pop up window producing an error to the username
14	Ensure forename is no longer than 15 characters	When registering an account, I will try to input a forename that is 15+ characters long.	When I try to enter this username, the program should return an error and not accept the username.	I will enter the forename thisusernameismore, which is more than 15 characters. The input text box maxes at 20 characters so it is not possible to spam. This should report an error

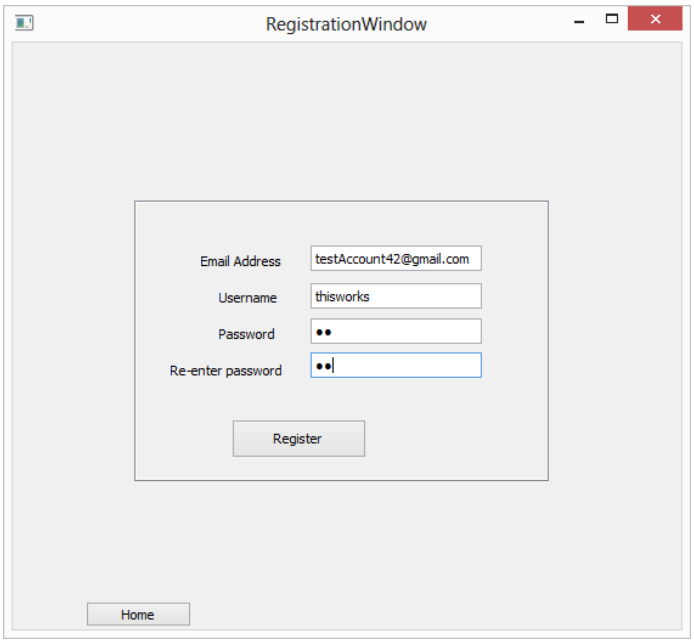
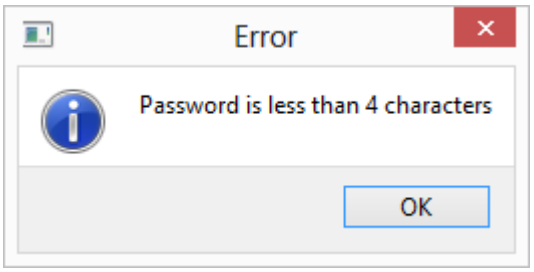
				  This shows that a pop up window is revealed and prevents the user from inputting an incorrect length of forename.
15	Ensure forename is longer than 2 characters	When registering an account, I will try to input a username that is less than 4 characters long.	When I try to enter this username, the program should return an error and not accept the username.	I will enter the forename aa, which is less than 3 characters. This should report an error 

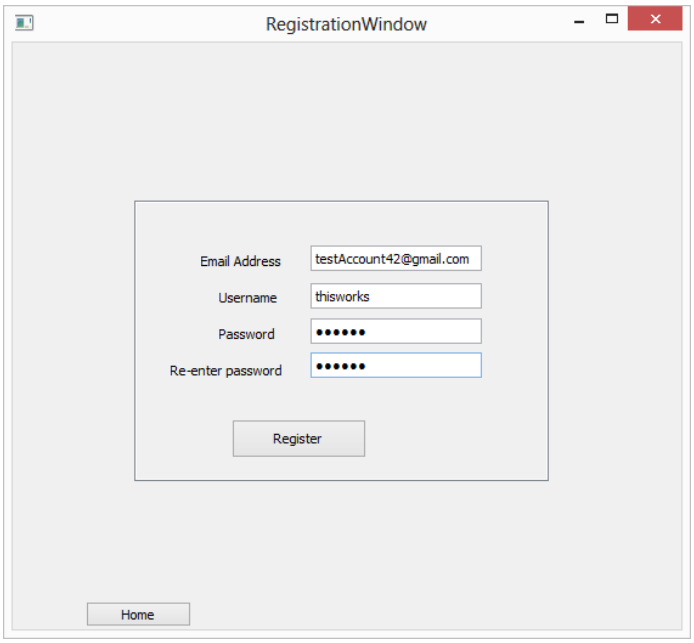
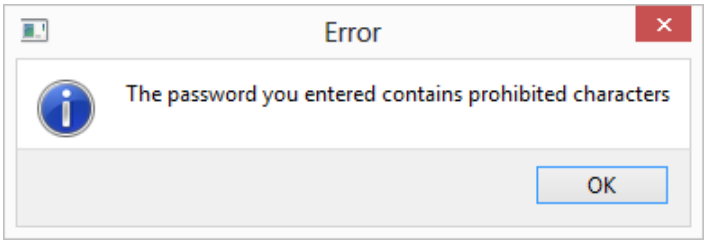
				 This shows that a pop up window is revealed and prevents the user from inputting an incorrect length of forename.
16	Ensure forename does not contain prohibited characters	To test this, I will enter a username with some prohibited characters	The program should return an error showing that the input contain prohibited characters	I will enter forename hello#. This contains prohibited character #. This reports an error and does not allow the user to accept this input and makes them re-enter.  

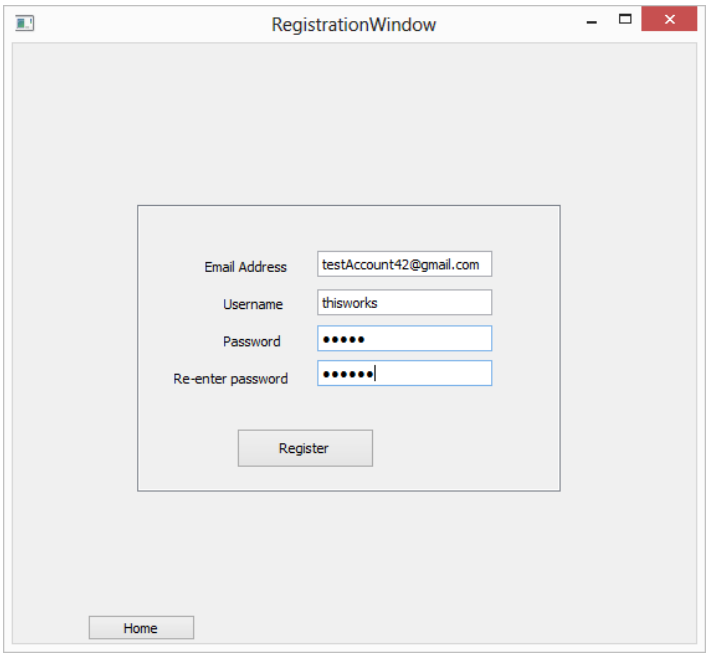
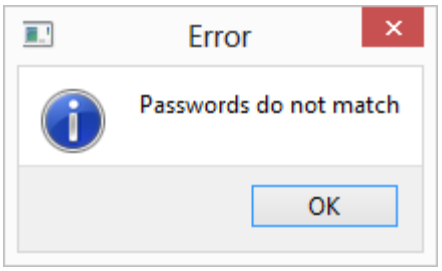
17	Ensure surname is no longer than 20 characters	When registering an account, I will try to input a username that is 13+ characters long.	When I try to enter this username, the program should return an error and not accept the name.	I will enter the surname thisusernameismorethan20, which is more than 20 characters. The input text box maxes at 20 characters so it is not possible to spam. This should report an error  This limits the text box to 20, so that the full name cannot be inputted in the first place. This can be seen as the full input I intended to input is cut off.
18	Ensure surname is longer than 4 characters	When registering an account, I will try to input a username that is less than 4 characters long.	When I try to enter this username, the program should return an error and not accept the username.	I will enter the forename aa, which is less than 3 characters. This should report an error 

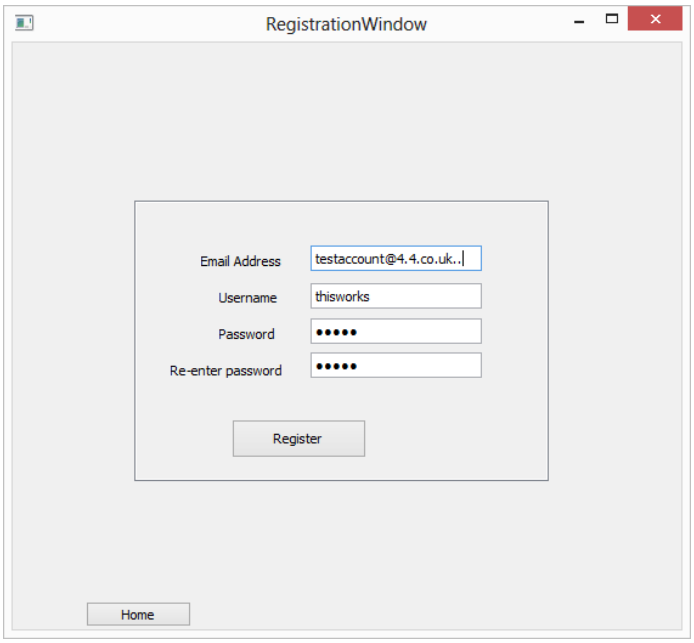
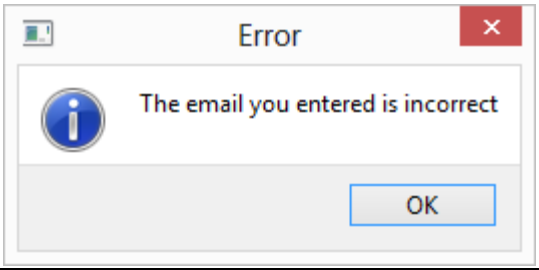
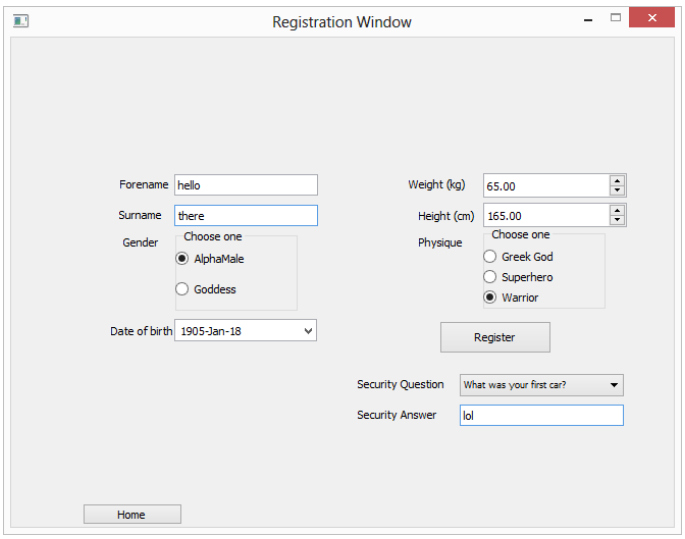
				 This shows that a pop up window is revealed and prevents the user from inputting an incorrect length of surname.
19	Ensure surname does not contain prohibited characters	To test this, I will enter a surname with some prohibited characters	The program should return an error showing that the input contain prohibited characters	I will enter forename there#. This contains prohibited character #. This reports an error and does not allow the user to accept this input and makes them re-enter.  

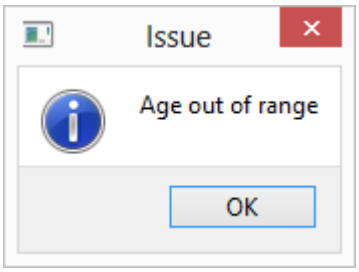
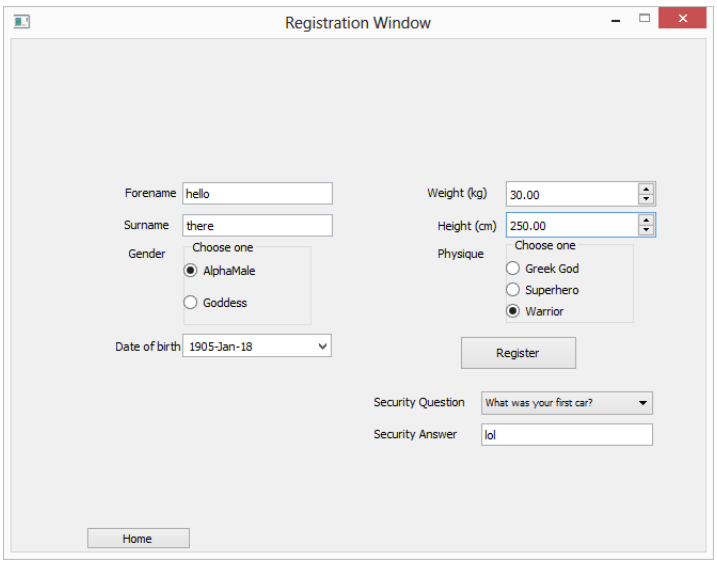
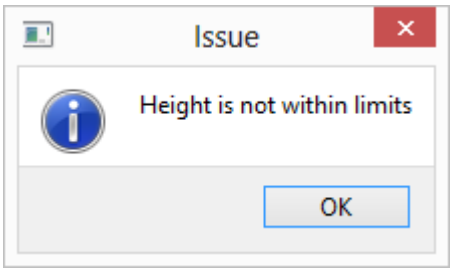
20	Ensure password is no longer than 20 characters	When registering an account, I will try to input a username that is 20+ characters long.	When I try to enter this username, the program should return an error and not accept the name.	I will enter the password thispasswordismorethan20, which is more than 20 characters. The input text box maxes at 20 characters so it is not possible to spam. This should report an error   This shows that a pop up window is revealed and prevents the user from inputting an incorrect length of password.
21	Ensure password is longer than 4	When registering an account, I will try to input a username that is less than 4	When I try to enter this username, the program should return an error and not accept the username.	I will enter the forename aa, which is less than 4 characters. This should report an error

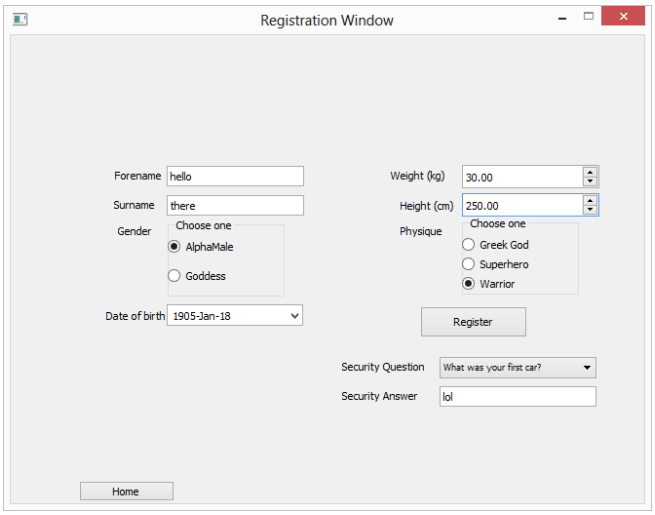
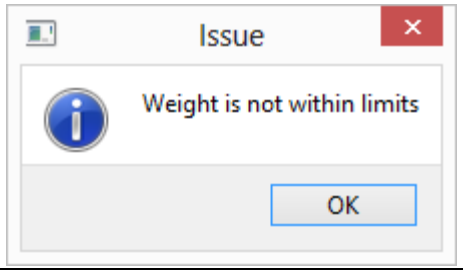
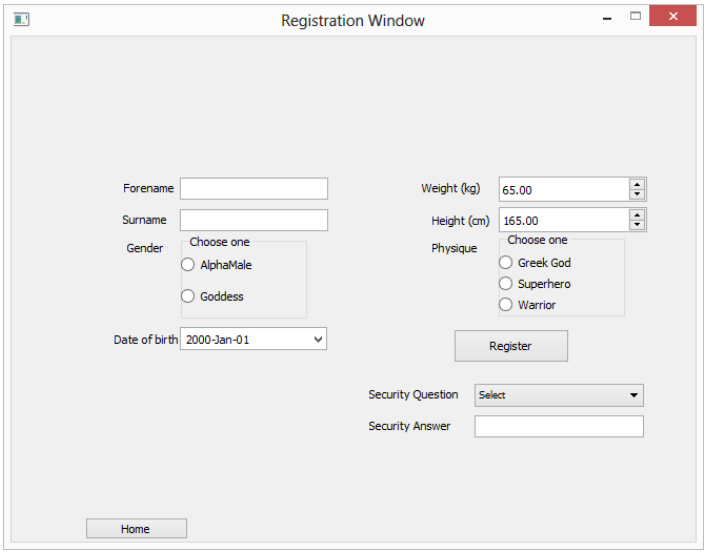
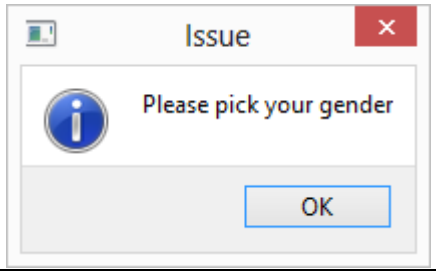
	characters	characters long.		  This shows that a pop up window is revealed and prevents the user from inputting an incorrect length of password.
22	Ensure password does not contain prohibited characters	To test this, I will enter a password with some prohibited characters	The program should return an error showing that the input contain prohibited characters	I will enter password there#. This contains prohibited character #. This reports an error and does not allow the user to accept this input and makes them re-enter.

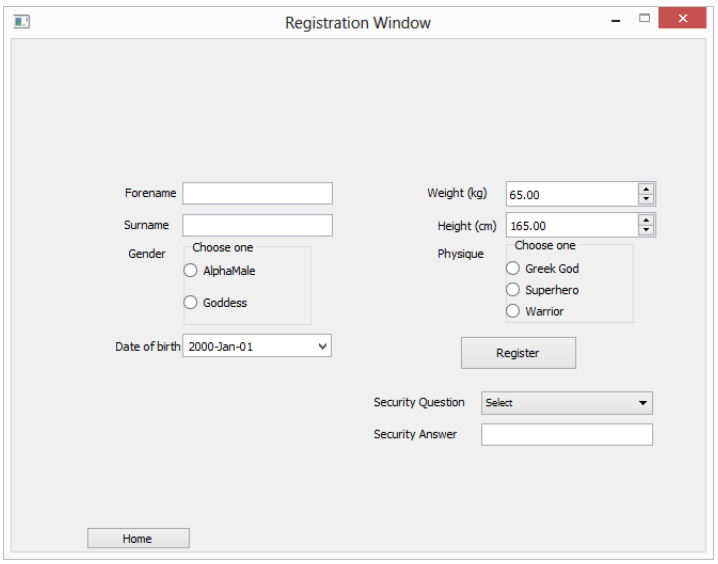
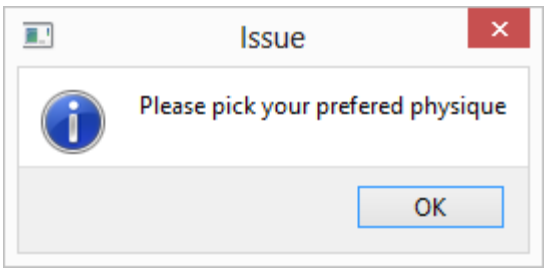
				 
23	Ensure password and re-enter password is the same	I will type in two different passwords to see if the program accepts the input	This should not accept this input and ask the user to re-enter their password until they match	I entered password there, then re-entered there1. This reports an error prompting the user that the passwords do not match and need to be corrected

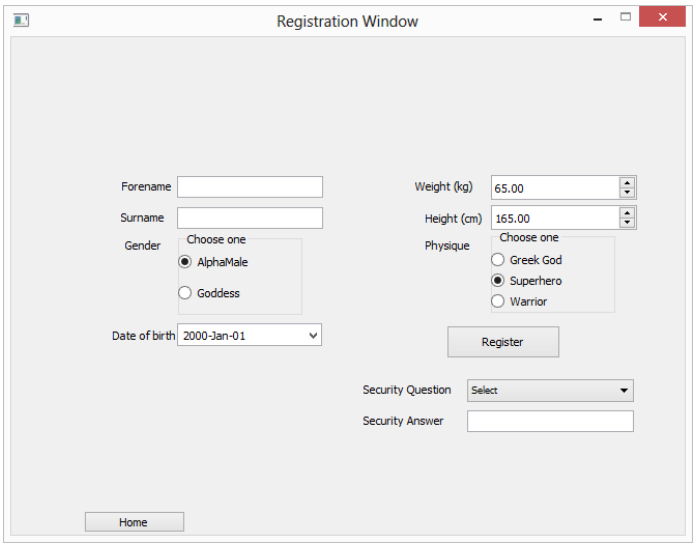
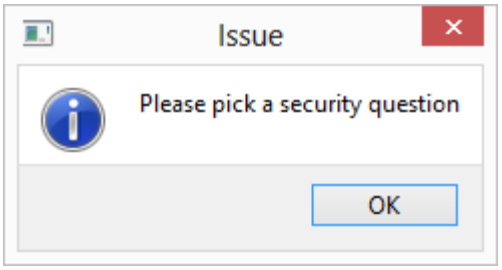
				 
24	Ensure that email address follows a valid format (must contain '@', followed by '.')	The email address used is needed in the program and is vital, and so can not be a fake email. Used to gain forgotten password.	This should not accept the following email address and return an error until the user inputs a valid email address	I have inputted the incorrect format for the email as testaccount@4.4.co.uk.. This is the incorrect format and should not be accepted. This produces a prompt for the user.

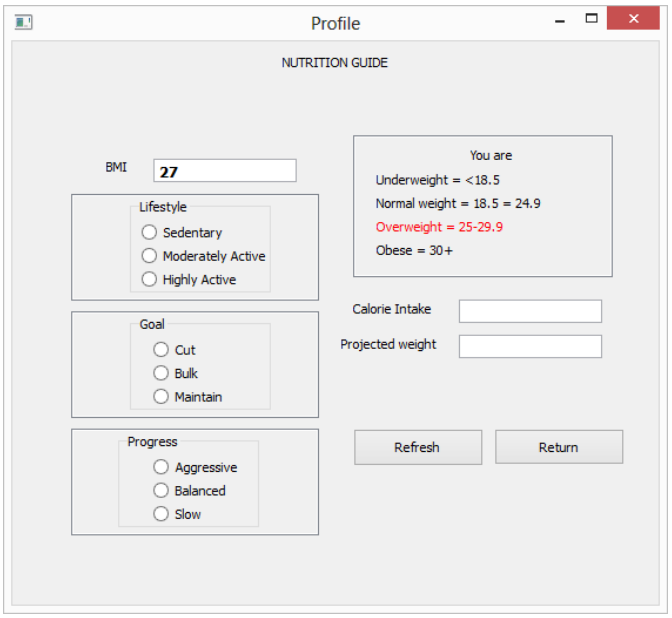
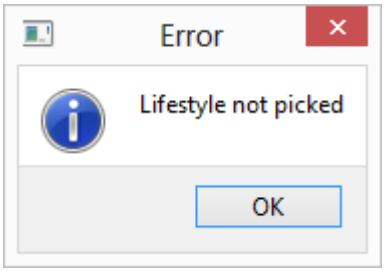
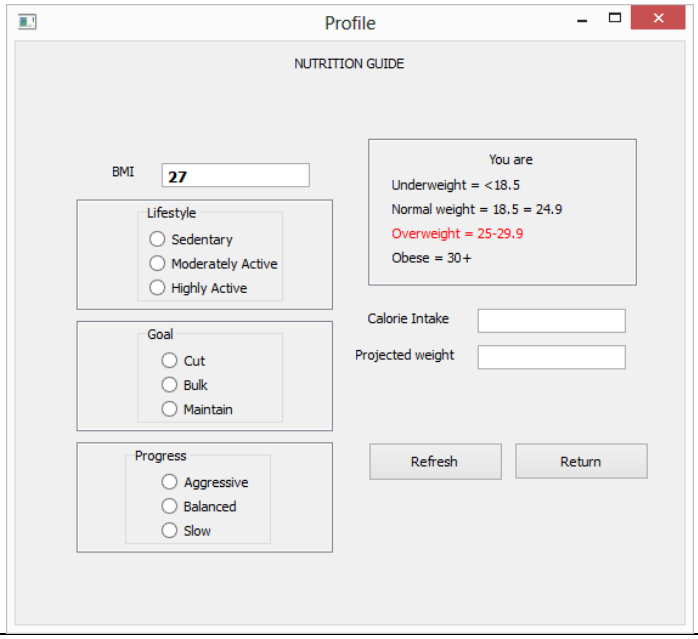
				 
25	Ensure age is within limits that is allowed to use the program (16 - 80 years old)	This will be tested by entering both a date of birth of below 16 and above 80	This should not accept these ages and report an error	Entering the date of birth as 1905-Jan-18, will report an error as it is outside the limits of what the program accepts. 

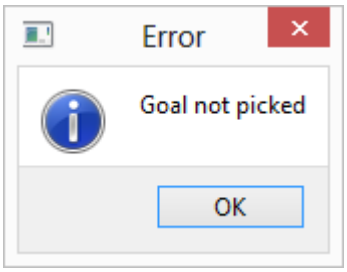
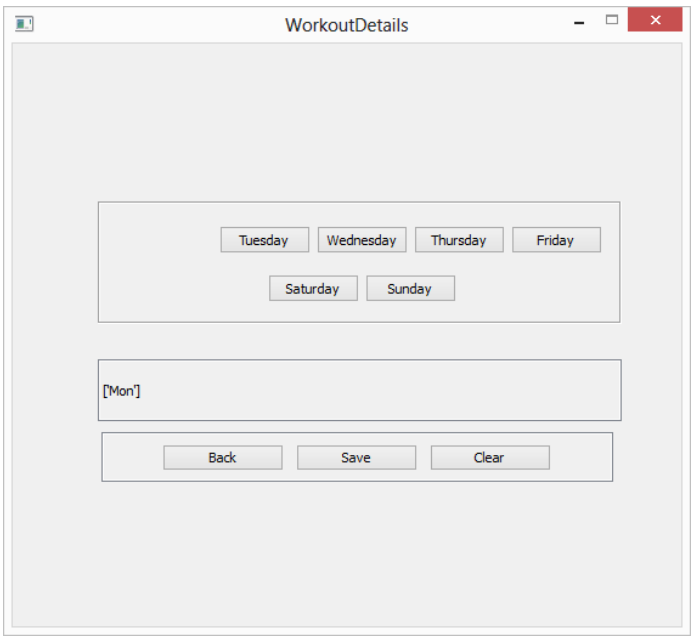
				 This will have a pop up window showing the error.
26	Ensure height entered is within appropriate level (130cm - 230cm)	When registering, I will enter values outside these limits of 130 and 230cm	This should not accept these heights and report an error	I will enter inputs that are outside the limits, in this case I will input the height 250. This reports the error below and prompts the user to enter a valid input.  
27	Ensure weight entered is within appropriate	When registering, I will enter values outside these limits of 40 and 250kg	This should not accept these heights and report an error	I will enter inputs that are outside the limits, in this case I will input the weight 30. This reports the error below and prompts the user to enter a valid input.

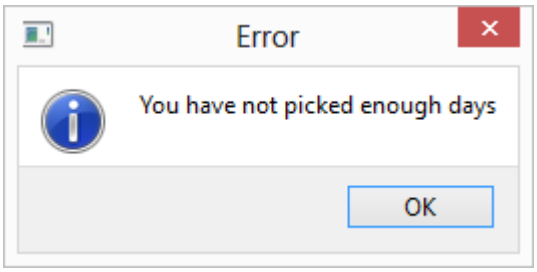
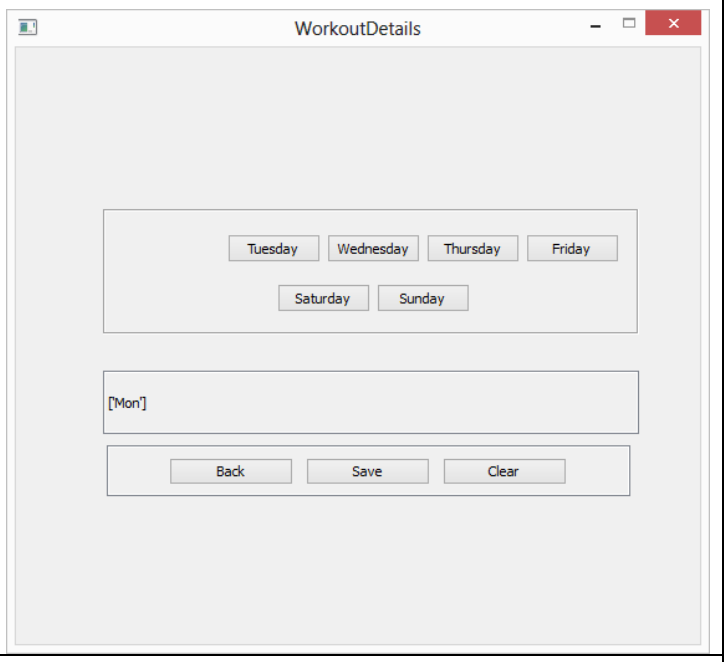
	level (40kg-250kg)			 
28	Ensure a gender is picked	When completing the registration, I will not pick an option for this	This should report and error and prompt the user to pick an option	If this is left blank, the program prompts the user to pick one of the options, and will not continue until chosen.  

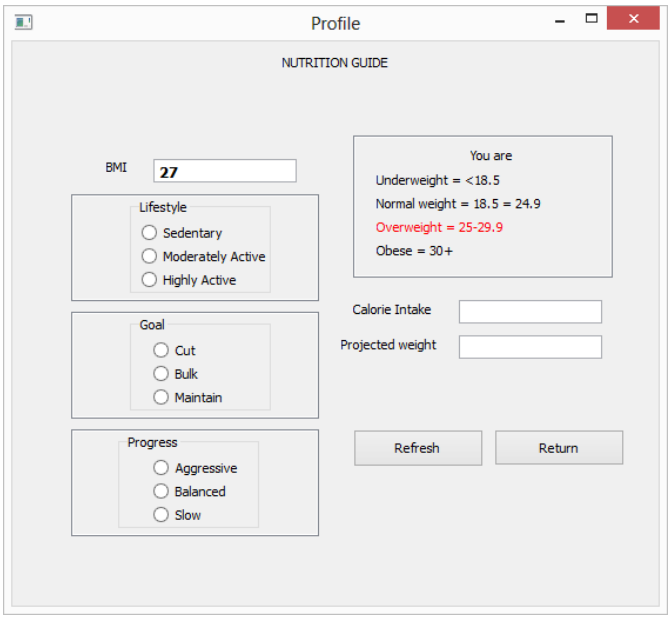
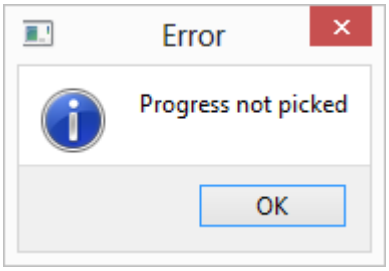
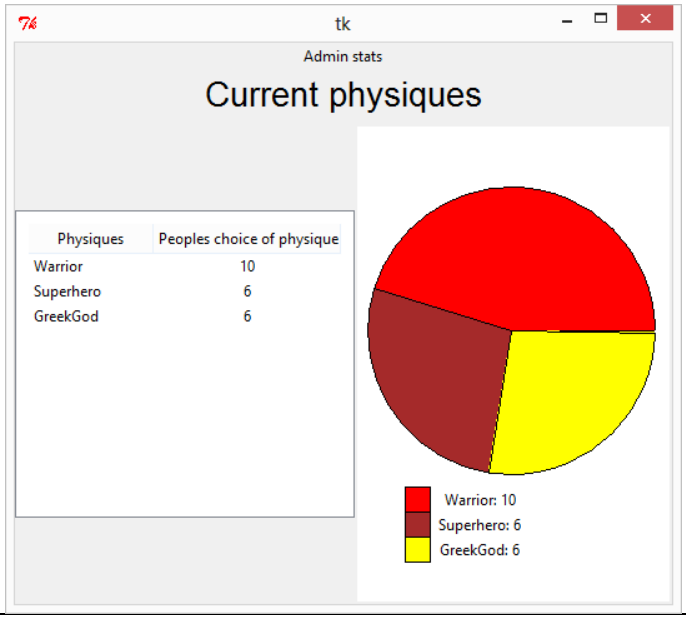
29	Ensure a physique type is picked	When completing the registration, I will not pick an option for this	This should report and error and prompt the user to pick an option	If this is left blank, the program prompts the user to pick one of the options, and will not continue until chosen.  
30	Ensure a security question is picked	When completing the registration, I will leave this section empty	This should report and error and prompt the user to pick an option and write in a security answer.	If this is left blank, the program prompts the user to pick one of the options, and will not continue until chosen.

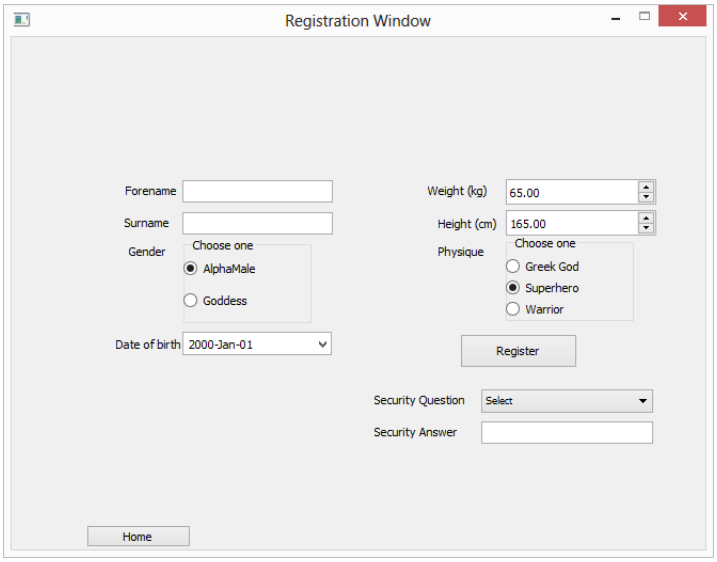
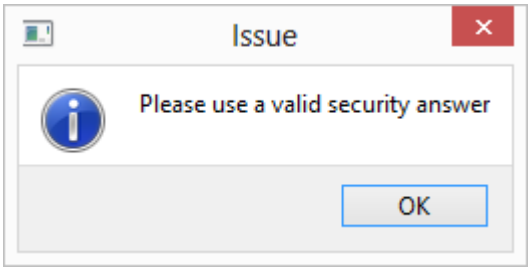
				 
31	Ensure security answer is hashed in database	This is to ensure users have a safe and secure account throughout their profile. This is not known by the user but protects them	This will be test by inputting password password. Every time this is needed, I will try to output the password to expect the return in hash. I will check the database fort he hash.	If this is left blank, the program prompts the user to pick one of the options, and will not continue until chosen.
32	Ensure the user inputs lifestyle	When completing the registration, I will not pick an option for this	This should report and error and prompt the user to pick an option	If this is left blank, the program prompts the user to pick one of the options, and will not continue until chosen.

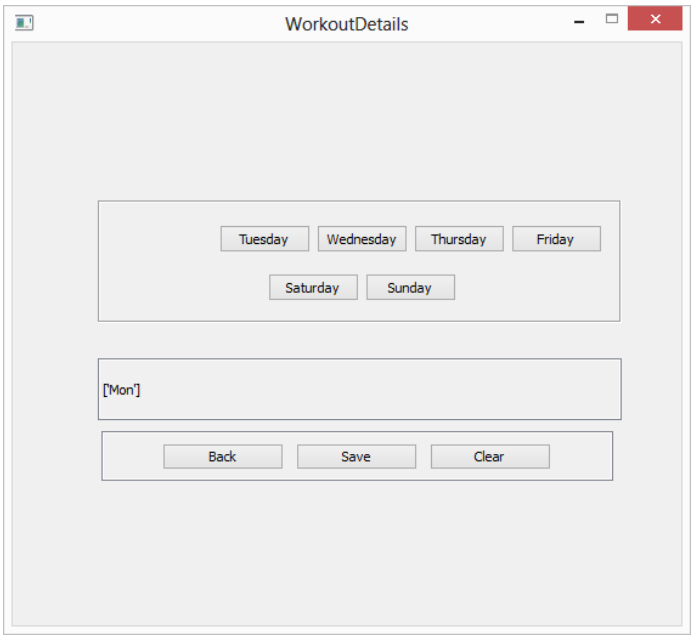
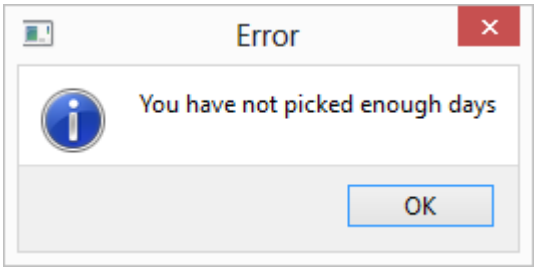
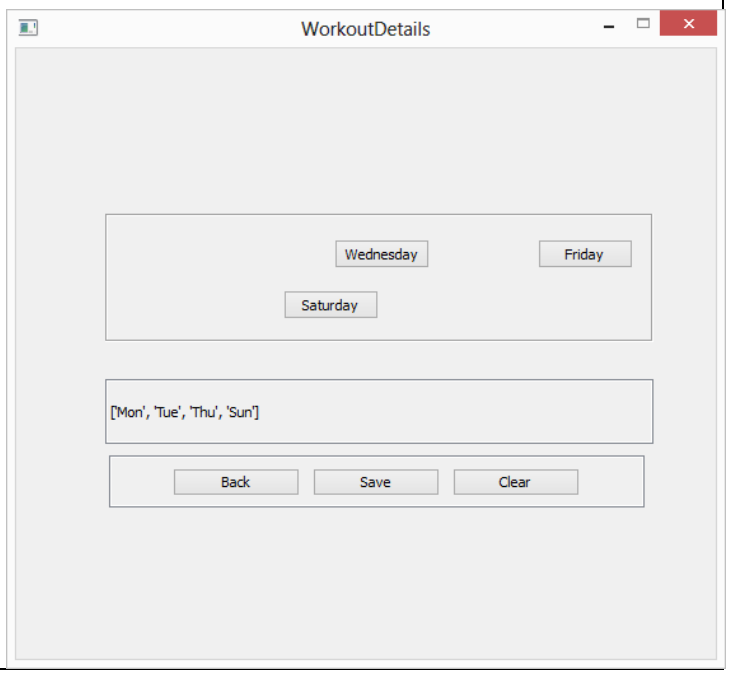
				 
33	Ensure the user inputs goal	When completing the registration, I will not pick an option for this	This should report and error and prompt the user to pick an option	If this is left blank, the program prompts the user to pick one of the options, and will not continue until chosen. 

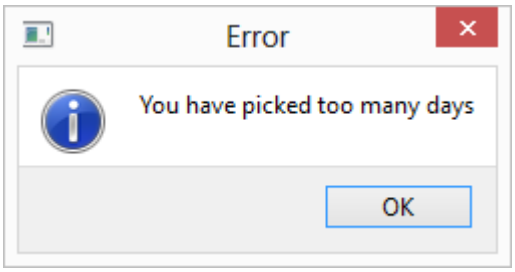
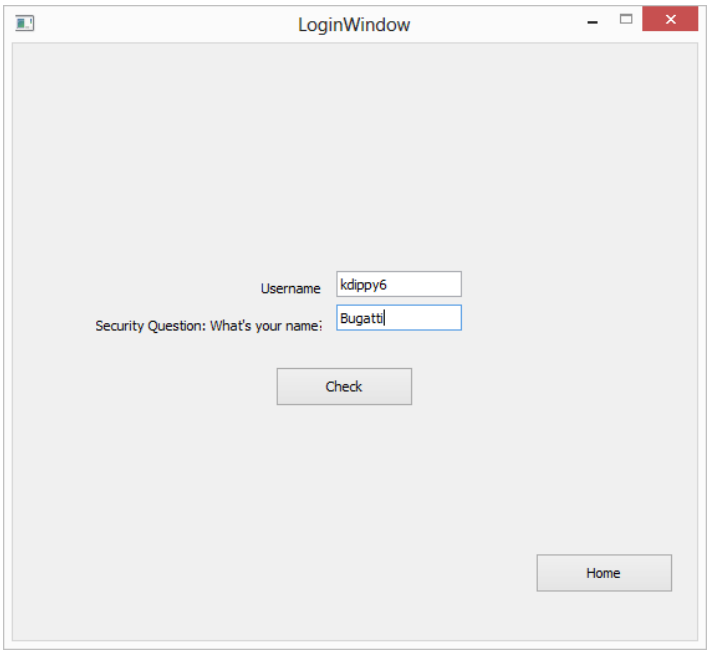
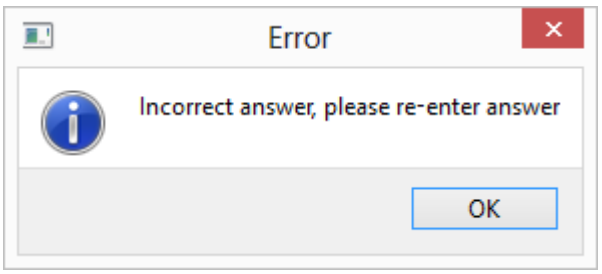
				
34	When storing password in db, this must be hashed	This is to ensure users have a safe and secure account throughout their profile. This is not known by the user but protects them	This will be test by inputting password password. Every time this is needed, I will try to output the password to expect the return in hash. I will check the database fort he hash.	
35	Prevent user inputting more or less days than needed	I will test this by trying to enter less than 3 days and more than 3 days.	This should report an error with amount of days chosen and ask the user to clear and start again	This will report an error if only one day is picked, where three days are needed. This will prompt the user to enter more days to reach the target needed. 

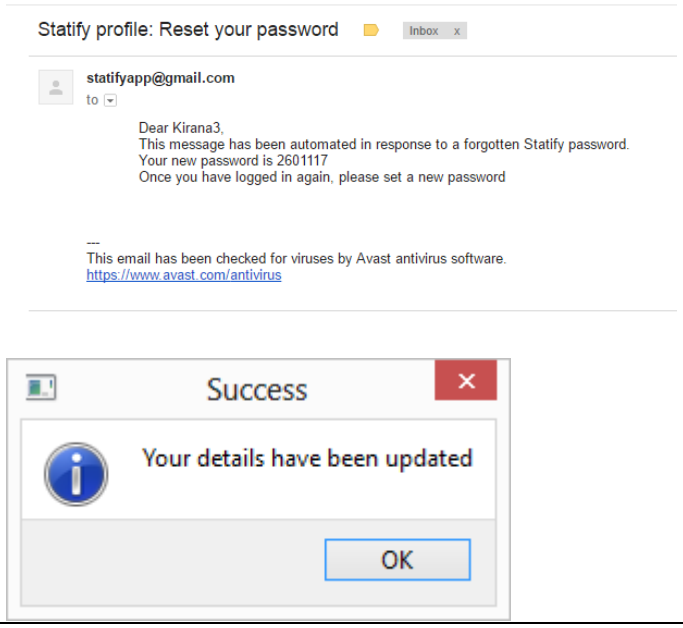
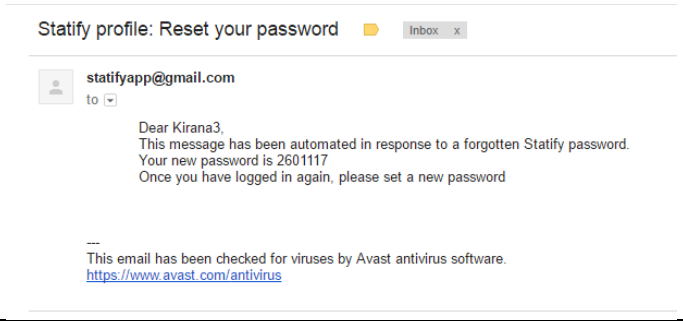
				
36	Prevent user entering the same day more than once	I will test this by trying to pick the same day twice.	This should return an error and the user should not be able to pick the same day. This should disappear when it is picked.	When the user clicks a day, the day disappears so that the user can not pick the same day again. 
37	Ensure user inputs rate of goal achieved	When completing the registration, I will not pick an option for this	This should report and error and prompt the user to pick an option	If this is left blank, the program prompts the user to pick one of the options, and will not continue until chosen.

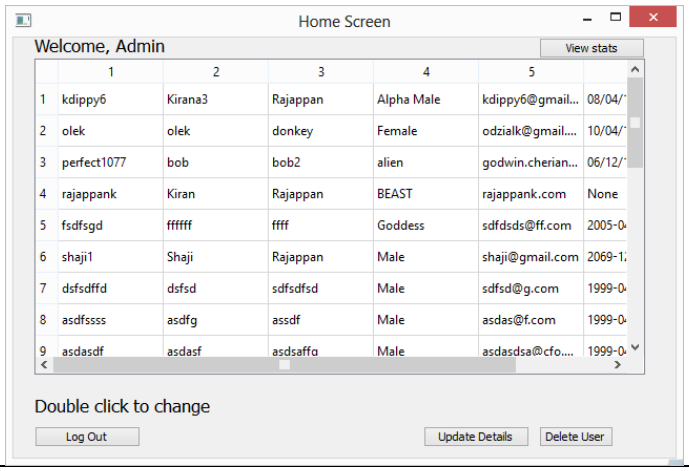
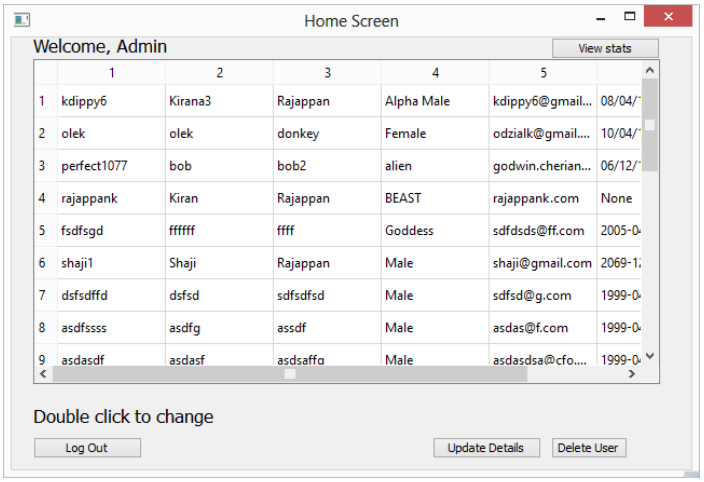
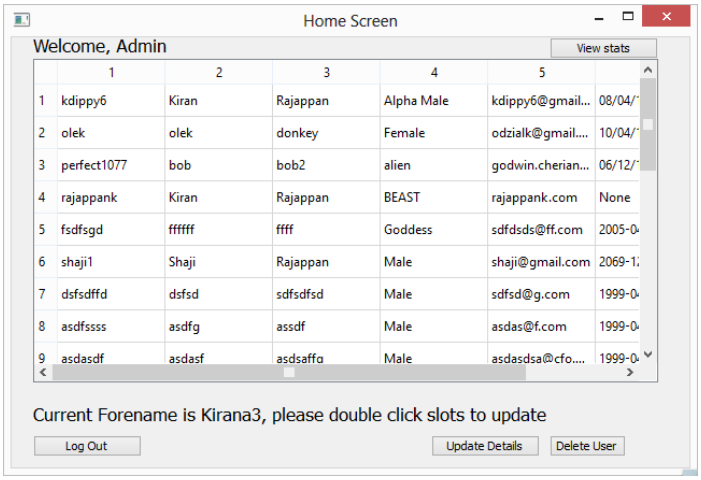
				 
38	Ensure pie chart calculations are correct	I will go through the data of amount of users that have picked each of the programs available	The calculations should be correct and show on a pie chart the correct dimensions on the chart.	This will be calculated to the amount of users who have picked the specific physiques and show on the pie chart. This should create a full circle with the correct proportions. 

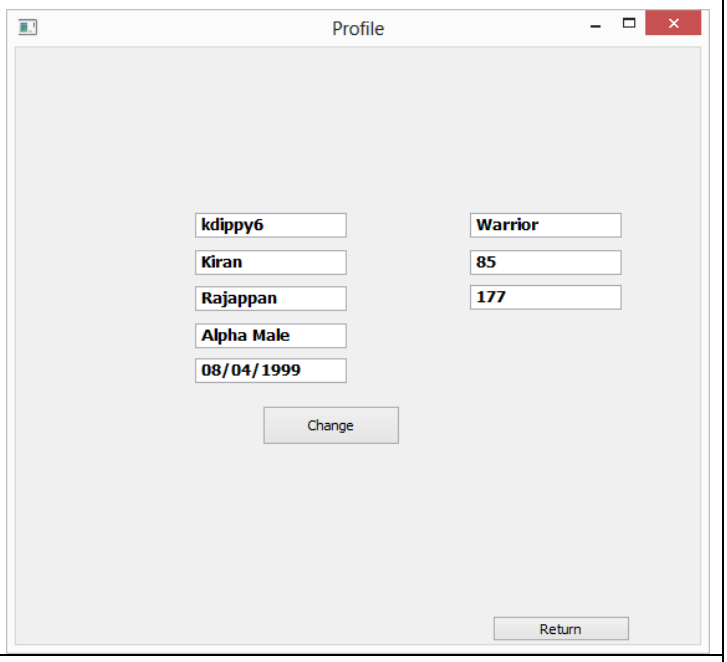
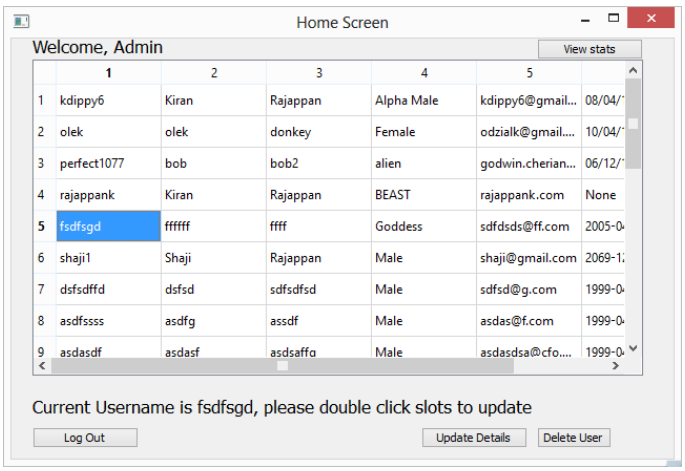
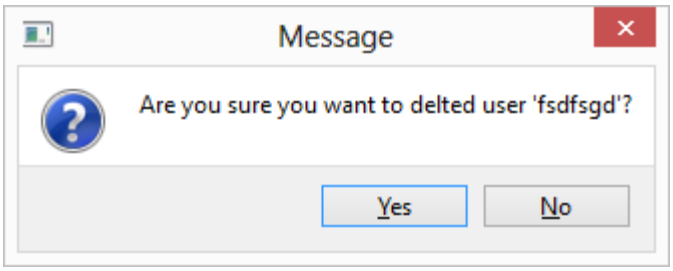
39	Ensure a security answer is entered	When completing the registration, I will leave this section empty	This should report and error and prompt the user to pick an option and write in a security answer.	If this is left blank, the program prompts the user to pick one of the options, and will not continue until chosen.  
40	Ensure amount of days picked is not below 3	I will test this by trying to enter less than 3 days and more than 3 days.	This should report an error with amount of days chosen and ask the user to clear and start again	This makes sure that the correct days are picked and a not below 3. Prompts user to input more days.

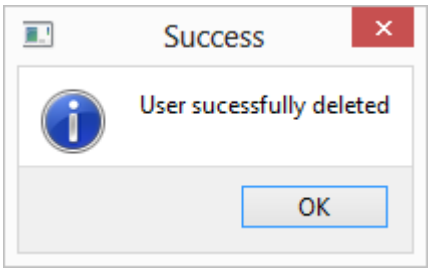
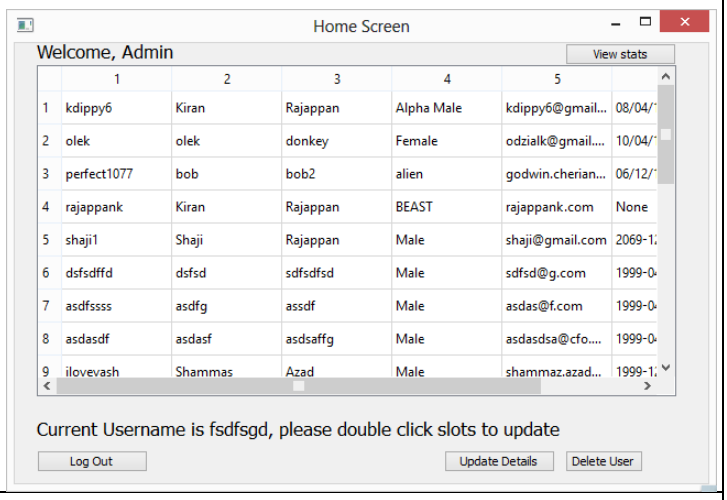
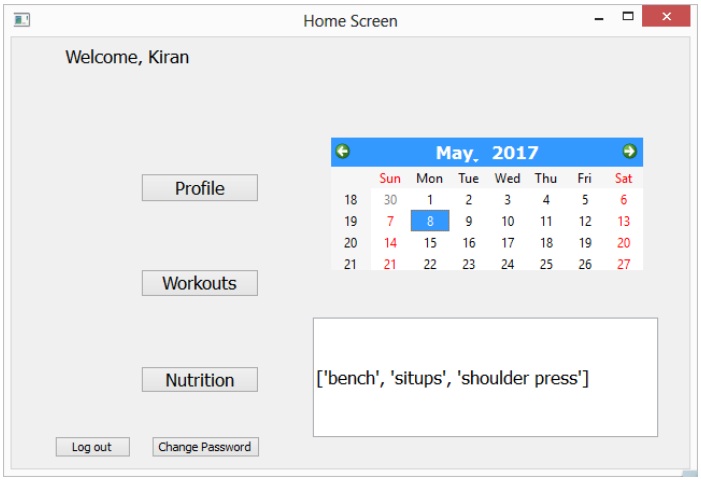
				 
41	Ensure amount of days picked is not above 3	I will test this by trying to enter more than 3 days.	This should report an error with amount of days chosen and ask the user to clear and start again	This makes sure that the correct days are picked and a not above 3. Prompts user to input less days. 

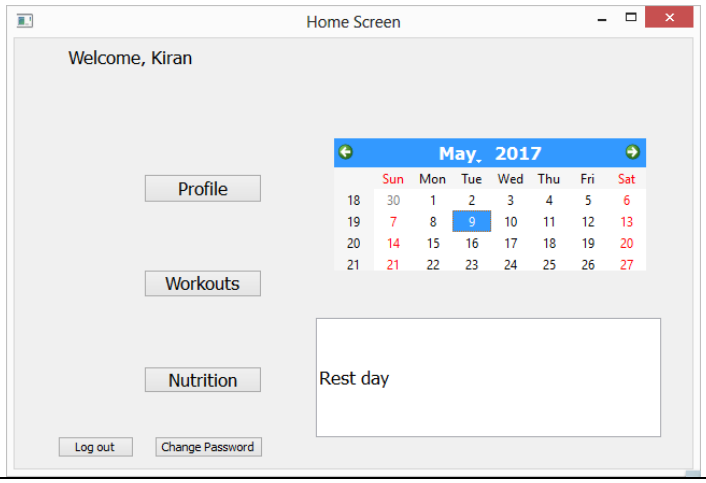
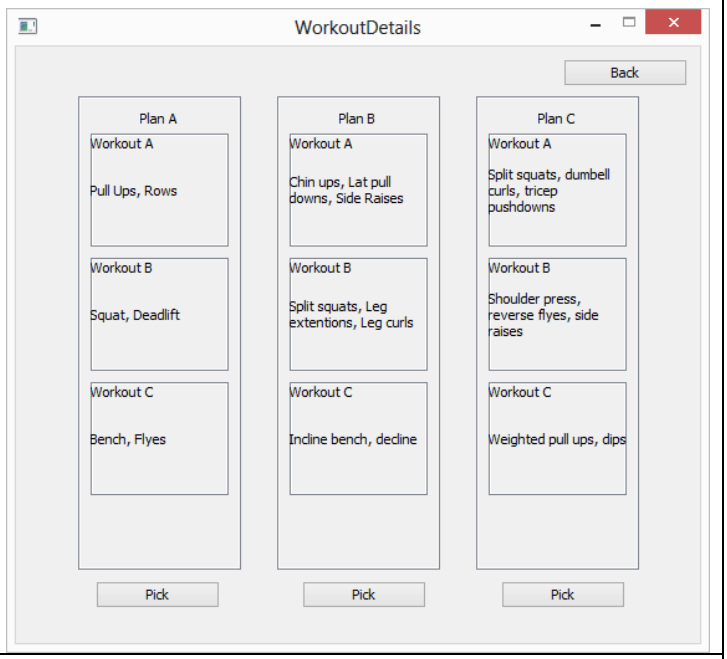
				
42	Check security answer entered by user matches stored security answer	I will check the recorded data of the answer and input, first an incorrect answer, then the correct answer	This should reject Buggati but accept the correct answer which is Kiran	When I entered Bugatti this rejected the input as the correct security answer is Kiran. When this was entered, the email was sent to the user.  

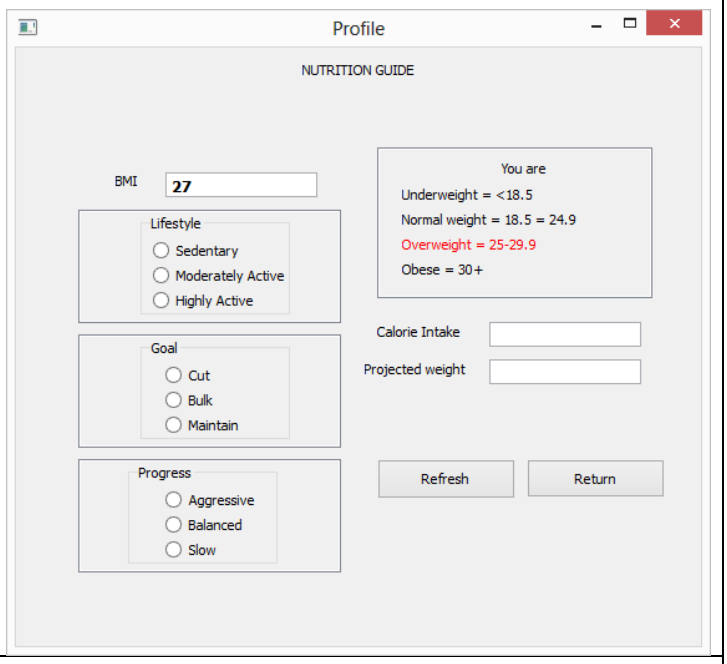
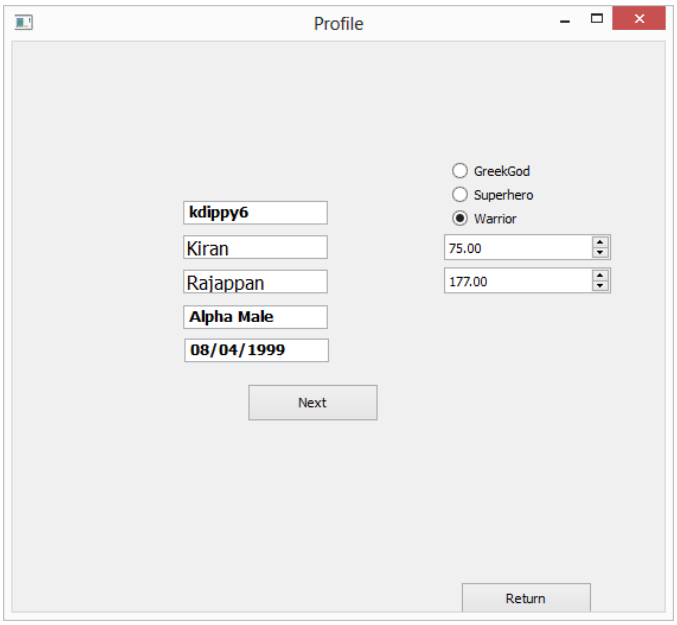
				
43	If match, send user an email with random code	I will check the email registered to the account to see if it is directed to the right email, and then check the code posted too.	Once the correct details have been entered, an email should be sent to kdippy6@gmail.com with the reset code.	Once the user correctly answers security question, the linked account receives the email with the random code. This shows this, below. 
44	Change password in database with hashed random code	This is to ensure users have a safe and secure account throughout their profile. This is not known by the user but protects them	This will be test by inputting password password. Every time this is needed, I will try to output the password to expect the return in hash. I will check the database fort he hash.	
45	Show all relevant information of	This will be tested by viewing the table views when logging	This should show that the Admin will be able to see on their screen. This should show basic information like	The admin has basic information of all users, that isn't private to adhere to basic DPA laws. This shows the Admin home screen with all the information.

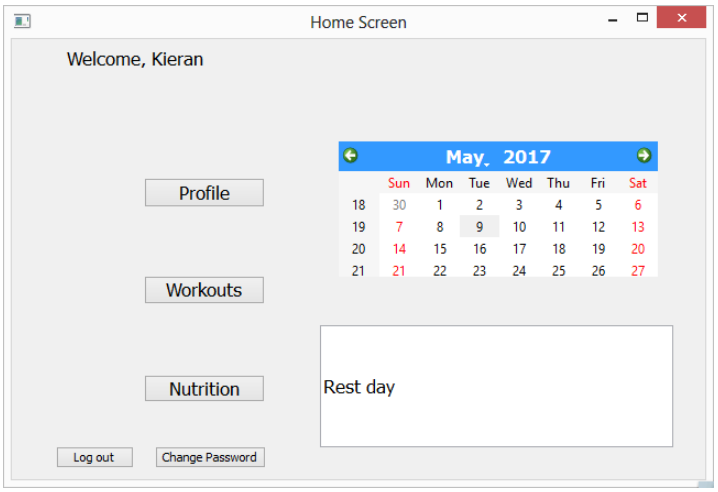
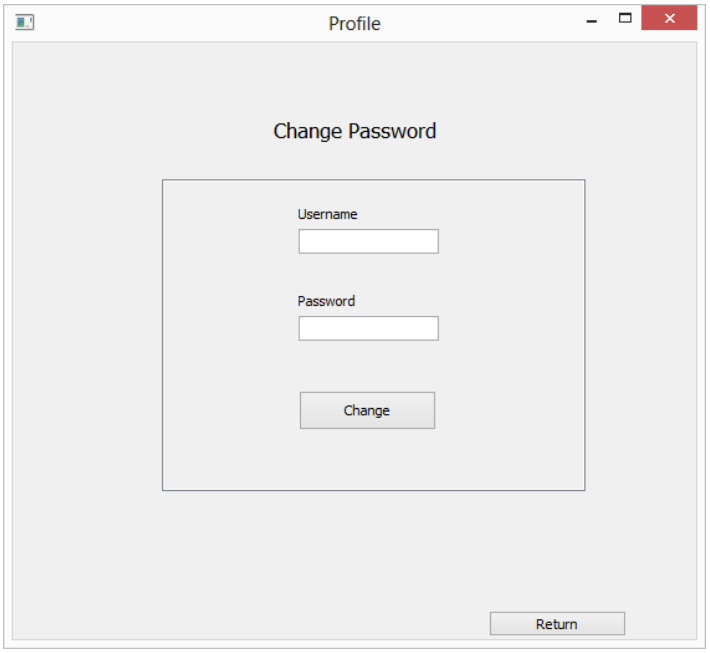
	all the users	on with an Admin account	name, age and email address.	
46	Allow admin to alter specific inform ation	This will be tested by viewing the table views when logging on with an Admin account	This should show that the Admin will be able to see on their screen. This should show basic information like name, age and email address.	I altered user kdippy6 name to Kiran, then login to that user to find that the name has changed.  

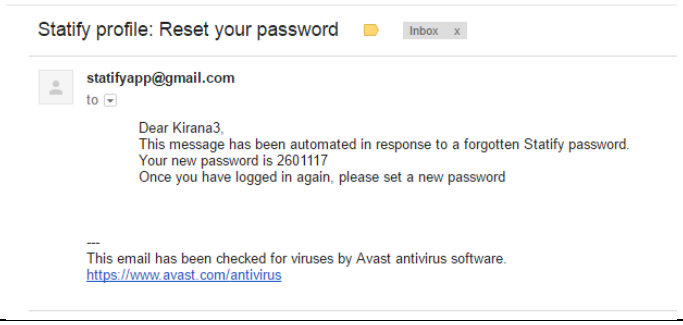
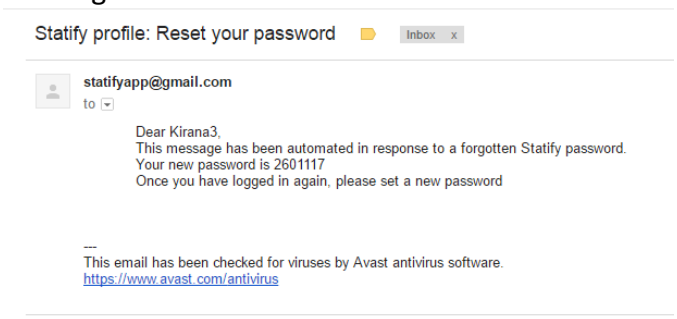
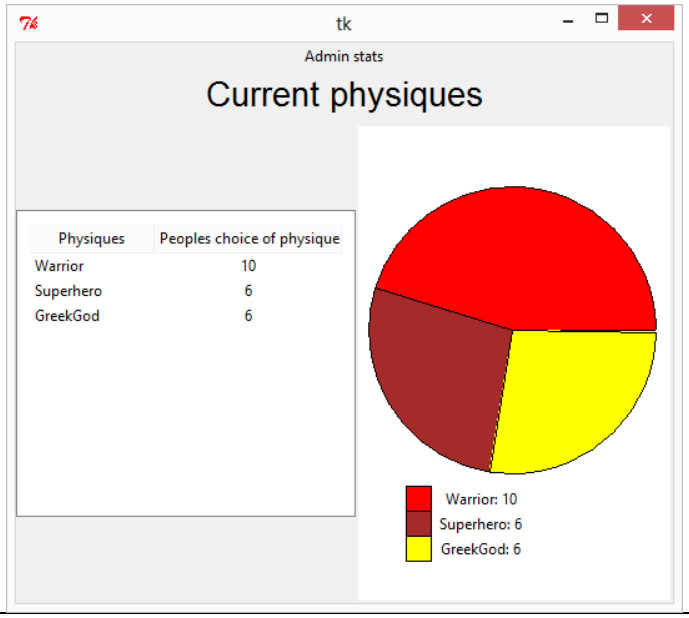
				
47	Allow admin to delete users	This will be tested by deleting a user when logging on with an Admin account	When trying to re log into the deleted account, this should not be possible	I deleted user 5 (fsdfsgd). This will delete this user off of the database and that user will not be able to login again.  

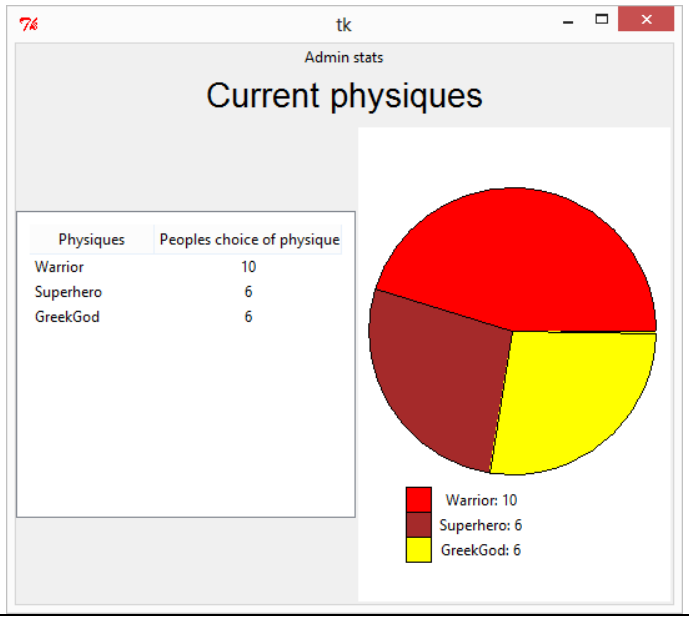
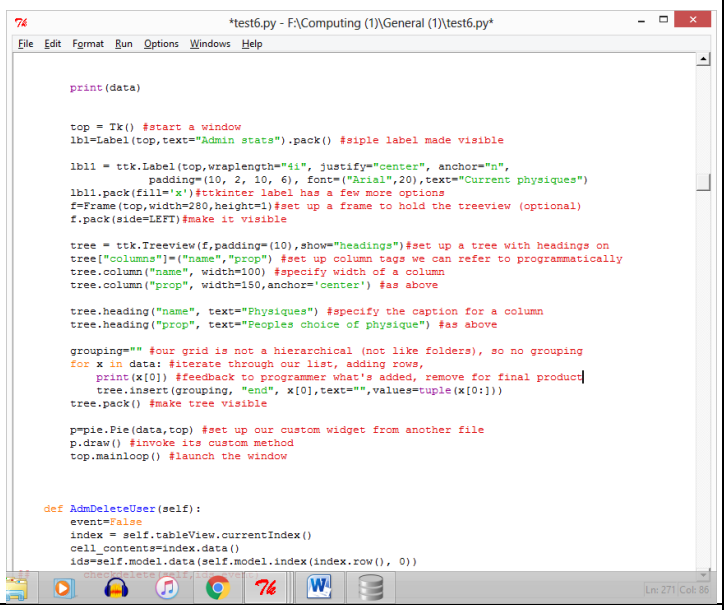
				 
48	Show on calendar, the specific workout they have on a set day or have rest day	This will have to be tested by selecting a certain order in days to ensure the right workouts show under the correct day and show off days too.	When I click the days Mon, Wed and Fri, this should show the workouts for that section in order. When I click any other day, it should say I have a rest day.	As previously shown, I picked days Monday, Wednesday and Friday. This shows that on Monday I have the first list of workouts and on Tuesday shows that I have a rest day. 

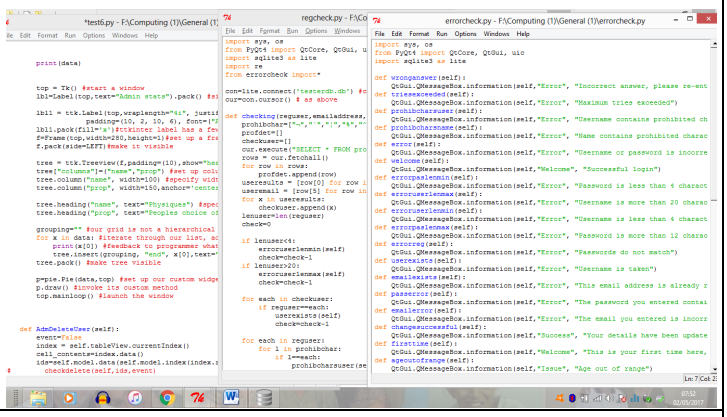
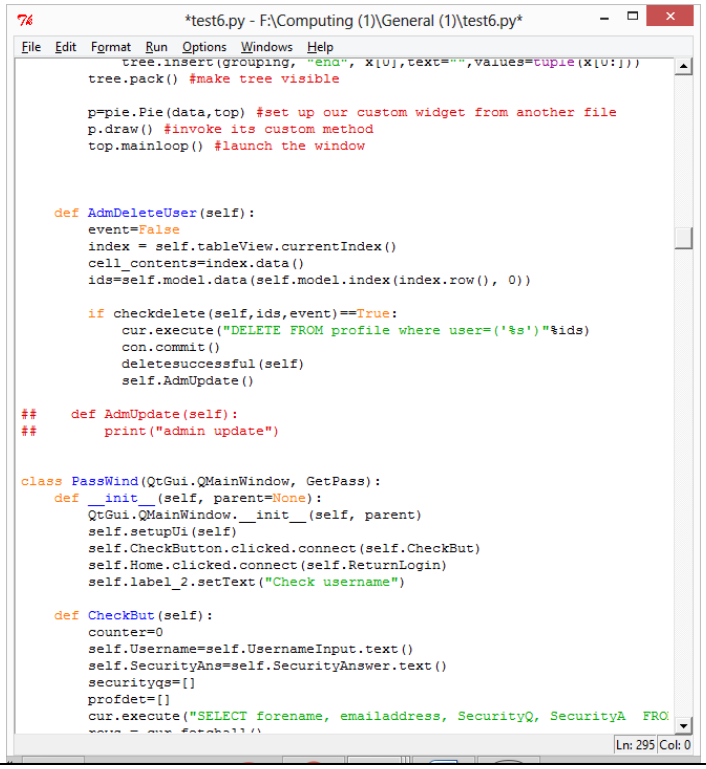
				
49	Show the different workouts available	This will be tested by accessing the workouts page and searching for different workouts to pick.	This should show 3 different workout options for the user to pick from depending on what they would like to. This should be personalized depending on the needs the user chose	This is the workout routine page which offers 3 different workouts for the user to chose from. 
50	Calculate BMI with weight and height inputted	This will simply need to be tested with hard coding using pre determined data already calculated and view the response	This should result in the same BMI result of 23.6 and then show on a table that the user is normal and average body weight.	With the current user details of 85kg and 177cm, the BMI is calculated at 27BMI and plotted on the table as Overweight.

				
51	User can alter and update details including passwords	This can be simply tested once the user has registered their account and going to their profile to see if it is possible to update details and to check if it saves.	The results should show that the details have been updated and even when the user logs out and then logs in again the details should be updated and saved	This shows that the user can update their own personal details and also have a password change option too. This shows that I changed the name from Kiran, to Kieran. 

				  This also shows the change password screen
52	User must be able to retrieve account if forgotten password	This will be tested by checking if the emailed code is the correct code that will be able to regain access to the linked account.	The code will be random, but should be the same that the password is reset to and that is sent through email	As shown the user can request the password and receive and an email with a reset password. When the user inputs the correct password, they can then go into their profile and change their password to what they like.

				
53	Must be able to send retrieval password to linked email	This will be tested by checking the email linked to the account when the email should be sent, to ensure that it is being received.	The email should be sent to the correct email (kdippy6@gmail.com) also revealing the correct message, including header and subject too.	As shown previously, the account email is linked to the email address and successfully sends an email to the target user. 
54	Show a pie chart of amount of users picking different physiques	I will go through the data of amount of users that have picked each of the programs available	The calculations should be correct and show on a pie chart the correct dimensions on the chart.	This is a success, this shows a pie chart of the users choices, which is plotted and shown to the admin 
55	Calculate each part of the pie chart	I will have predetermined data set up and use these calculations to ensure that	The calculations should produce the following result and when constructing the pie chart, should	Each part of the pie chart is calculated to provide the proportions that are shown on the popup window.

		the pie chart is complete.	show a full pie chart and accurate results.	
56	Comments for coding throughout	Once the program is completed, the program should have comments, this should relate to all bits of code so it is understandable.	The comments should provide enough information for other coders to understand the basics of it and allow them to change code in the future	This shows the pie chart code which both proves that the calculations are being met and appropriate comments are also included so that it can be maintained. 
57	Have modular code for functions	Once the program is completed, the program should be split into module, so that it is easier to understand	The modules should have enough information for other coders to understand the basics of it and allow them to change code in the future	This shows two modular self coded windows which are then imported into the main program to be read from. This includes errorcheck and regcheck, which are set up to show pop up windows and check the validity of registration inputs.

		and follow in the future.		
58	Meaningful variable names	Once the program is completed, the program should have comments, this should relate to all bits of code so it is understandable.	The variables should all be self explanatory and easy to understand each of their functions.	I have many examples of usage of meaningful variable names, this shows the Admin deletion code, showing each of the variables having clear meanings and easy to follow. 

```
*test6.py - F:\Computing (1)\General (1)\test6.py*
File  Edit  Format  Run  Options  Windows  Help
            tree.insert(grouping, "end", x[0],text="",values=tuple(x[0:]))
        tree.pack() #make tree visible

        p=pie.Pie(data,top) #set up our custom widget from another file
        p.draw() #invoke its custom method
        top.mainloop() #launch the window



    def AdmDeleteUser(self):
        event=False
        index = self.tableView.currentIndex()
        cell_contents=index.data()
        ids=self.model.data(self.model.index(index.row(), 0))

        if checkdelete(self,ids,event)==True:
            cur.execute("DELETE FROM profile where user=('%s')"%ids)
            con.commit()
            deletesuccessful(self)
            self.AdmUpdate()

##    def AdmUpdate(self):
##        print("admin update")


class PassWind(QtGui.QMainWindow, GetPass):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.CheckButton.clicked.connect(self.CheckBut)
        self.Home.clicked.connect(self.ReturnLogin)
        self.label_2.setText("Check username")

    def CheckBut(self):
        counter=0
        self.Username=self.UsernameInput.text()
        self.SecurityAns=self.SecurityAnswer.text()
        securityqs=[]
        profdet=[]
        cur.execute("SELECT forename, emailaddress, SecurityQ, SecurityA  FRO
```

Ln: 295 Col: 0

Success of solution

Test number	Testing for	Relation to success critera	Success?	Comments
1	Navigation of all windows	This will be tested by allowing Olek and myself to press every button on each of the screens. I will have a plan of the windows showing where each should be connected to. If these buttons take the user to the correct window, it will be checked off the list showing that it works.	Full success	All the windows were easy to navigate and the users found that this was a success.
2	Check if navigating windows crashes	This will be tested by allowing Olek, and myself to spam all of the navigation buttons and try to break program.	Full success	None of the windows crashed with heavy testing and were very smooth.
3	User must register to an account	This will be tested by allowing a new user to the program, to try and gain access. They will try logging in with details.	Full success	Users found that it was easy and simple to register an account with Statify App.
4	User must log on account with provided username and password	This will be tested by inputting the same test data from previous test to ensure user now has access to the	Full success	The only way to access a Statify profile is by logging into a registered account with a username and password.

		program after registering.		
5	Allow user to pick the days that they wish to workout on	This will be tested by inputting three days to work out on. This will then be checked after logging out, then back in again.	Partial Success	Users could pick the days that they would like to workout out on, however this was only limited to three days due to the workouts available. When talking to the target market, they agreed that they would like some flexibility with the days so that they can choose the amount of days to work out on too. This can be amended in the next prototype and If I had more time available.
6	Provide admin screen to maintain	This will be checked by logging into regular accounts in the same login screen, then inputting an admin username and password	Success	The admin has control of all accounts registered to Statify. They can change basic information, while not being able to access sensitive data.
7	When user requests to change password, make sure password hash is the same as db hash	This will be tested by trying to print the password and read from the database which it is saved to, whenever a password is required.	Not successful	If I had more time and resources to work on this, it would be a simple implementation. As currently stands, hashed passwords and sensitive data are not implemented into my program. As this is only a prototype and will not be available for commercial use, this will suffice. However, before mass scale production, this must be amended.
8	Ensure all windows have back buttons for user to navigate	I, along with Olek, will navigate through every screen to try and find a dead end, to see if it is possible to not be stuck	Success	All windows have return buttons to take them back to the previous screen or the home screen depending on the situation. Users commented, that it might be easier if the return button stayed in the same place for all screens, this would make it easier to find.
9	Ensure admin can	This will be checked by	Success	Admin could access Admin screen with correct credentials.

	access admin screen	logging into regular accounts in the same login screen, then inputting an admin username and password		
10	Have access to reclaim password if forgotten	This will be tested by trying to reset the password and trying to regain access.	Success	This allows the user to go to their email to access a unique random code to access their account. This is only accessible to them.
11	Ensure username is no longer than 12 characters	When registering an account, I will try to input a username that is 13+ characters long.	Success	Prompts user and does not allow invalid entries.
12	Ensure username is longer than 4 characters	When registering an account, I will try to input a username that is less than 4 characters long.	Success	Prompts user and does not allow invalid entries.
13	Ensure username does not contain prohibited characters	To test this, I will enter a username with some prohibited characters	Success	Prompts user and does not allow invalid entries.
14	Ensure forename is no longer than 15 characters	When registering an account, I will try to input a forename that is 15+ characters long.	Success	Prompts user and does not allow invalid entries.
15	Ensure forename is	When registering an	Success	Prompts user and does not allow invalid entries.

	longer than 2 characters	account, I will try to input a username that is less than 4 characters long.		
16	Ensure forename does not contain prohibited characters	To test this, I will enter a username with some prohibited characters	Success	Prompts user and does not allow invalid entries.
17	Ensure surname is no longer than 20 characters	When registering an account, I will try to input a username that is 13+ characters long.	Success	Prompts user and does not allow invalid entries.
18	Ensure surname is longer than 4 characters	When registering an account, I will try to input a username that is less than 4 characters long.	Success	Prompts user and does not allow invalid entries.
19	Ensure surname does not contain prohibited characters	To test this, I will enter a surname with some prohibited characters	Success	Prompts user and does not allow invalid entries.
20	Ensure password is no longer than 20 characters	When registering an account, I will try to input a username that is 20+ characters long.	Success	Prompts user and does not allow invalid entries.
21	Ensure password is longer than 4 characters	When registering an account, I will try to input a username that	Success	Prompts user and does not allow invalid entries.

		is less than 4 characters long.		
22	Ensure password does not contain prohibited characters	To test this, I will enter a password with some prohibited characters	Success	Prompts user and does not allow invalid entries.
23	Ensure password and re-enter password is the same	I will type in two different passwords to see if the program accepts the input	Success	Doesn't allow user to save password unless it is identical
24	Ensure that email address follows a valid format (must contain '@', followed by '.')	The email address used is needed in the program and is vital, and so can not be a fake email. Used to gain forgotten password.	Success	Prompts user and does not allow invalid entries.
25	Ensure age is within limits that is allowed to use the program (16 -80 years old)	This will be tested by entering both a date of birth of below 16 and above 80	Success	Prompts user and does not allow invalid entries.
26	Ensure height entered is within appropriate level (130cm-230cm)	When registering, I will enter values outside these limits of 130 and 230cm	Success	Prompts user and does not allow invalid entries.

27	Ensure weight entered is within appropriate level (40kg-250kg)	When registering, I will enter values outside these limits of 40 and 250kg	Success	Prompts user and does not allow invalid entries.
28	Ensure a gender is picked	When completing the registration, I will not pick an option for this	Success	Prompts user and does not allow invalid entries.
29	Ensure a physique type is picked	When completing the registration, I will not pick an option for this	Success	Prompts user and does not allow invalid entries.
30	Ensure a security question is picked	When completing the registration, I will leave this section empty	Success	Prompts user and does not allow invalid entries.
31	Ensure security answer is hashed in database	This is to ensure users have a safe and secure account throughout their profile. This is not known by the user but protects them	Not successful	If I had more time and resources to work on this, it would be a simple implementation. As currently stands, hashed passwords and sensitive data are not implemented into my program. As this is only a prototype and will not be available for commercial use, this will suffice. However, before mass scale production, this must be amended.
32	Ensure the user inputs lifestyle	When completing the registration, I will not pick an option for this	Success	Prompts user and does not allow invalid entries.
33	Ensure the user inputs goal	When completing the registration, I will not pick an option for this	Success	Prompts user and does not allow invalid entries.
34	When storing	This is to ensure users have a	Not successful	If I had more time and resources to work on this, it would be a simple implementation. As

	password in db, this must be hashed	safe and secure account throughout their profile. This is not known by the user but protects them		currently stands, hashed passwords and sensitive data are not implemented into my program. As this is only a prototype and will not be available for commercial use, this will suffice. However, before mass scale production, this must be amended.
35	Prevent user inputting more or less days than needed	I will test this by trying to enter less than 3 days and more than 3 days.	Success	This only accepts three days to be inputted
36	Prevent user entering the same day more than once	I will test this by trying to pick the same day twice.	Success	Hides days that are already picked.
37	Ensure user inputs rate of goal achieved	When completing the registration, I will not pick an option for this	Success	Prompts user and does not allow invalid entries.
38	Ensure pie chart calculations are correct	I will go through the data of amount of users that have picked each of the programs available	Partial success	This shows the correct values, however with calculations that are not perfectly divisible shows a gap between the charts. Users did not like the appearance of the pie chart used and would prefer a more professional looking chart.
39	Ensure a security answer is entered	When completing the registration, I will leave this section empty	Success	Prompts user and does not allow invalid entries.
40	Ensure amount of days picked is not below 3	I will test this by trying to enter less than 3 days and more than 3 days.	Success	This only accepts three days to be inputted

41	Ensure amount of days picked is not above 3	I will test this by trying to enter more than 3 days.	Success	This only accepts three days to be inputted
42	Check security answer entered by user matches stored security answer	I will check the recorded data of the answer and input, first an incorrect answer, then the correct answer	Success	Only allows email to be sent and password reset if user inputs the correct security answer.
43	If match, send user an email with random code	I will check the email registered to the account to see if it is directed to the right email, and then check the code posted too.	Success	Sends user an email prompting them that they requested a password reset and code.
44	Change password in database with hashed random code	This is to ensure users have a safe and secure account throughout their profile. This is not known by the user but protects them	Not successful	If I had more time and resources to work on this, it would be a simple implementation. As currently stands, hashed passwords and sensitive data are not implemented into my program. As this is only a prototype and will not be available for commercial use, this will suffice. However, before mass scale production, this must be amended.
45	Show all relevant information of all the users	This will be tested by viewing the table views when logging on with an Admin account	Success	Allows admin to see all related and vital information of all users so that it can be altered. Users requested a function to directly talk to or communicate with admin to request a change that they themselves are not authorized to do. This can be amended by having a separate screen for requests that the admin can see as requests.

46	Allow admin to alter specific information	This will be tested by viewing the table views when logging on with an Admin account	Success	Admin can alter information for users. Admin user requested a backup file, incase they accidently altered information and so they can redo entries. This can be created by having a backup database for up to 24 hours. This would have to be researched as this is an unknown skill to me.
47	Allow admin to delete users	This will be tested by deleting a user when logging on with an Admin account	Success	Admin can alter information for users. Admin user requested a backup file, incase they accidently altered information and so they can redo entries. This can be created by having a backup database for up to 24 hours. This would have to be researched as this is an unknown skill to me.
48	Show on calendar, the specific workout they have on a set day or have rest day	This will have to be tested by selecting a certain order in days to ensure the right workouts show under the correct day and show off days too.	Partial Success	Shows user the workouts that they have on a specific day. This does not however go into detail about how many sets and or reps. Users also found this hard to read.
49	Show the different workouts available	This will be tested by accessing the workouts page and searching for different workouts to pick.	Partial Success	Shows some workouts, however users are concerned about the amount available. They would like more options.
50	Calculate BMI with weight and height inputted	This will simply need to be tested with hard coding using pre determined data already calculated and view the response	Success	This calculated the persons BMI and plotted it to show the user where their health is.

51	User can alter and update details including passwords	This can be simply tested once the user has registered their account and going to their profile to see if it is possible to update details and to check if it saves.	Success	Users can update details and save changes.
52	User must be able to retrieve account if forgotten password	This will be tested by checking if the emailed code is the correct code that will be able to regain access to the linked account.	Success	Users can request password with correct security answers
53	Must be able to send retrieval password to linked email	This will be tested by checking the email linked to the account when the email should be sent, to ensure that it is being received.	Success	Users receive email linked to account
54	Show a pie chart of amount of users picking different physiques	I will go through the data of amount of users that have picked each of the programs available	Success	Admin can see the progress users are making with the choice of physiques. Appearance of pie chart could be improved.
55	Calculate each part of the pie chart	I will have predetermined data set up and use these	Partial Success	Works out data to not enough detail, shows gap with smaller data. This is still accurate however.

		calculations to ensure that the pie chart is complete.		
56	Comments for coding throughout	Once the program is completed, the program should have comments, this should relate to all bits of code so it is understandable.	Success	Other coding colleagues were able to read and understand code after reading through comments
57	Have modular code for functions	Once the program is completed, the program should be split into module, so that it is easier to understand and follow in the future.	Success	Have multiple modules which can be easily updated and changed in the future.
58	Meaningful variable names	Once the program is completed, the program should have comments, this should relate to all bits of code so it is understandable.	Success	Other coding colleagues were able to read and understand code after reading through variables.

What can be tested before the program is complete

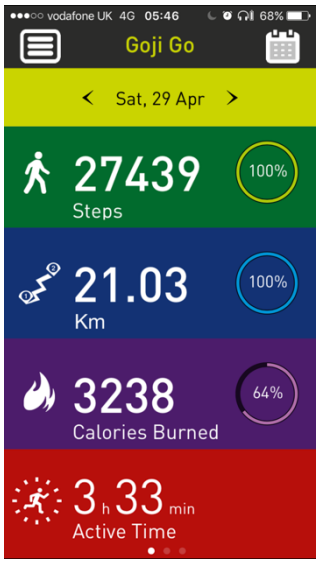
Describing the final product

To ensure my program runs as efficiently as possible I need to consider the big O complexities of the algorithms that I have used. Big O complexity looks at the worst possible outcome and shows how the different methods can be used to solve them. The most appropriate being, the most suitable to tackle the problem the easiest. The majority of my programs algorithm relies on linear sequencing, O(n) whereby time and data are proportional. If I double the amount of data, I also double the time taken. This can be seen with a simple while loop, as the time taken to iterate through until a certain condition is met, is entirely up to the amount of data that it will have to trawl over, to meet the objective. One module that could rely on a separate model could be the email sender, which could send multiple sets of the same data over to multiple emails. However, the way the I personally set this module, means that only one set of data is being sent to one email at one single time, and thus meets the O(n) linear complexity. Thus showing how the worst possible case complexity that my program will have to be dependent on is linear, which scales data with time, more time more data.

Maintenance and development

Improvements with more time

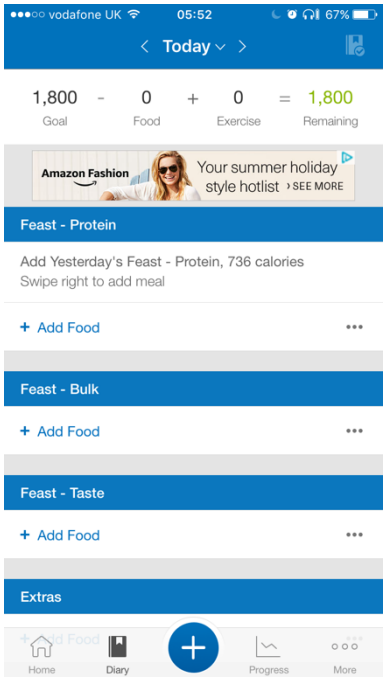
Possible ways to improve my program could be by increasing the amount of functionality around my program. One function that could greatly improve my program could be implementing a pedometer, when the program is developed into mobile. This would then allow the amount of calories burned to be tracked. This could be further integrated by allowing fitness trackers to be natively available with my program. This would show some basic facts about how many steps the user had, amount of elevation the user gained, heartbeat and vital readings and max speed too. An example of this is the Go Goji app with my fitness tracker of choice. Below is a picture of its app, although I would not plan to use a 3rd party app, this provides the basic premise of what I am trying to achieve.



To supplement this, and to add extra useablitity, I would like to connect this to social media, allowing users to compare with other users about their fitness and step goals. This would help to motivate each of the users to try and outdo each other. Not only this, but with the increase in users of social media, it would also provide a lot of attention though social media use and potentially

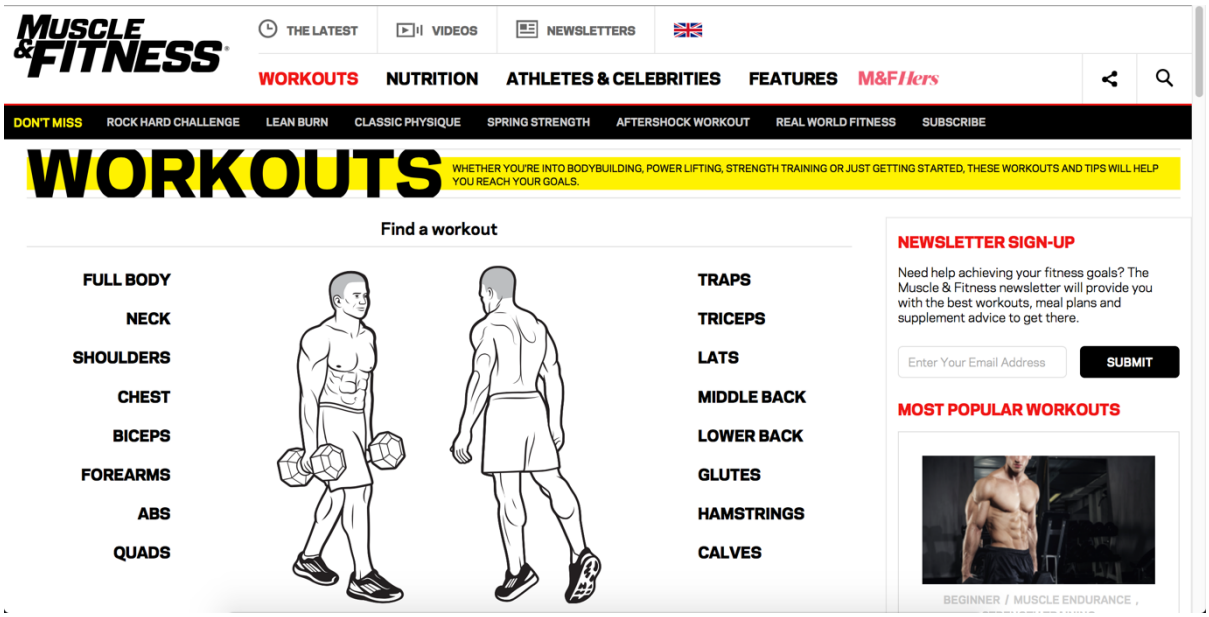
more clients.

Furthermore, having access to the camera to read data (barcodes) to track calories inputted too. This may present a problem as licensing may need to be bought in order to use these features. If I had more time to research and learn how to, I would like to connect my program online, to transfer data across platforms for ease of use for users. An example of this is myfitnesspal,, which tracks the calories inputted by the user after scanning the barcode to retrieve this data.



This brings me to the next point that I would like to address if I had more time. When I started the journey of creating my program, I decided to design and produce a fully prepared and professional GUI and branding. The first few windows allowed this, with the stratify branding, some boxes and a professional looking layout. However, due to licensing issues, I ran out of my permissions to use Adobe Photoshop, my preferred platform to create the GUI. This was not the only reason for this however, if I did continue to create a personalized branding for my entire product, I would have run out of time due to not creating a precise and solid plan about when I will create what, in what time. I decided to create the rest of the GUI with basic pyqt constructs.

Working with other companies could also bring in greater attention to the Statify app. Teaming up with a bigger brand company such as Muscle and Fitness could allow users to benefit from the information from here, already implemented and integrated into their Statify App. Combining these apps could provide an endless list of workouts routines, meal plans and basic advice which would far greater exceed those that personal trainers at gyms could provide. However, to satisfy these trainers, there could be a separate function completely to access Muscle and Fitness for those that prefer personal trainers. However, those more interested in Muscle and Fitness could import the workouts from the website to sync with the Statify App to use these workouts.



This would simply have include an import button which will open up the muscle and fitness page so that the user can input a certain workouts. Furthermore, Muscle and Fitness offers lots of advice on how to do workouts. This was an objective that the users wanted to see. I didn't manage to implement this into Statify, however, linking Muscle and Fitness would allow users to click a workout to hyperlink to Muscle and Fitness to view how to do a specific workout.

Lastly, I would like to implement a music feature to the program. This is to allow users to link and connect their songs to a specific workout, so depending on the day of their workout, a specific song playlist would play. This would be best suited to the workout that they are completing. With the basic information that I found online, it shows that EDM, Dance music with high beats/ minute are best suited for a cardio workout, to allow the user to keep their work ethic to a higher pace and keep their heart rate elevated.

Porting program to mobile

To port this program to a mobile platform should not cause too much concern. Firstly, the screen dimensions need to be altered to fit the average mobile screen
, and then to adapt to a portrait or landscape set up. This will then need to be adjusted for a touch screen set up, over a pointer. The fonts and GUI will also have to be altered to be able to be seen in a smaller screen. The code written does not use python specific code and thus can be easily transferred, however transferring the exact code over to a mobile app language, like Apples swift, may cause some concern. The code natively doesn't transfer and some extra attention will be needed to convert this over. Finally, using a different GUI creator may have to be an option, as of current state, python and pyqt do not natively support the use of mobile services and thus a separate program will be needed to replace all pyqt services.

Final code including modules.

MAIN CODE

```
import sys, os
from tkinter import * ;import tkinter.ttk as ttk; import cls_pie as pie
from PyQt4 import QtCore, QtGui, uic
import sqlite3 as lite
import re
from errorcheck import*
from regcheck import *
import time
from sendingemail import*
from datetime import date
import calendar
import csv
on=lite.connect('testerdb.db') #connect sql file
cur=con.cursor() # as above


loginwin = uic.loadUiType("LoginScreen.ui")[0]
registerwin = uic.loadUiType("RegisterScreen.ui")[0]
RegisterContinued = uic.loadUiType("RegisterScreenContinued.ui")[0]
homewin = uic.loadUiType("HomeScreen.ui")[0]
profilescreen = uic.loadUiType("ProfileScreen.ui")[0]
workoutwindow = uic.loadUiType("WorkoutsScreen.ui")[0]
Profilechangingwin = uic.loadUiType("ProfileScreenChange.ui")[0]
ChngPass = uic.loadUiType("ChangePassword.ui”)[0]a
ChngPassVerify = uic.loadUiType("ChangePasswordVerify.ui")[0]
NutritionScr= uic.loadUiType("NutritionScreen.ui")[0]
GetPass=uic.loadUiType("GetPassword.ui")[0]
WorkoutScrChange=uic.loadUiType("WorkoutsScreenChange.ui")[0]
Workoutsplit=uic.loadUiType("WorkoutDays.ui")[0]
AdminScreen=uic.loadUiType("AdminScreen.ui")[0]

class LoginWindow(QtGui.QMainWindow, loginwin):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.RegisterButton.clicked.connect(self.open_register)
        counter=0
        self.LoginButton.clicked.connect(self.open_home)
        self.ForgottenPassword.clicked.connect(self.GetPassword)
        variablename=self.UsernameInput.text()
```

```
    def open_register(self):
        RegWind.show()
        self.hide()
        TheFirstWindow.UsernameInput.clear()
        TheFirstWindow.PasswordInput.clear()


    def open_home(self):
        profdet=[]
        usernames=[]
        passwords=[]
        securityqs=[]
        global usernome
        usernome=self.UsernameInput.text()
        self.password=self.PasswordInput.text()
        cur.execute("SELECT * FROM profile")
        rows = cur.fetchall()
        for row in rows:
            profdet.append(row)

        securityq = [row[15] for row in profdet]
        for z in securityq:
            securityqs.append(z)

        useresults = [row[0] for row in profdet]
        for x in useresults:
            usernames.append(x)

        passresults = [row[1] for row in profdet]
        for y in passresults:
            passwords.append(y)


        def profileuser():
            lenuser=len(usernome)
            lenpas=len(self.password)
            state=False
            counter=0
            print(counter)
            admin=False



##          def login():
####            counter=counter+1
            for j in range (len(usernames)):
```

```
            if usernames[j]==usernome:
                if usernome==("Admin"):
                    admin=True
                    print(usernome,admin)


                counter=counter+1
##                  while counter<6:
                if passwords[j]==self.password:
                    state=True
                    cur.execute("SELECT Forename FROM profile where user='%s'"%usernome)
                    rows = cur.fetchall()
                    global name
                    name=(rows[0][0])

##              login()

            if state==True:
                if admin==True:
                    cur.execute("SELECT * FROM profile")
                    rows = cur.fetchall()
##                  for each in rows:
##                      setItem(, 1, new QTableWidgetItem("Hello"))
##                  AdminScreen.tableWidget.setRowCount(len(rows))
##                  AdminScreen.tableWidget.setColumnCount(len(each))

                    AdminScreen.show()
                    self.hide()
                else:
                    welcome(self)
                    HomeWind.label_2.setText("Welcome, %s"%name)
                    HomeWind.show()
                    self.hide()
            else:
                error(self)

        profileuser()

        TheFirstWindow.UsernameInput.clear()
        TheFirstWindow.PasswordInput.clear()

    def GetPassword(self):
        GetPass.SecurityAnswer.hide()
        GetPass.show()
        self.hide()
        TheFirstWindow.UsernameInput.clear()
        TheFirstWindow.PasswordInput.clear()
```

```
class AdminScr(QtGui.QMainWindow, AdminScreen):
   def __init__(self, parent=None):
      QtGui.QMainWindow.__init__(self, parent)
      self.setupUi(self)
      self.model = QtGui.QStandardItemModel(self)
      self.tableView.setModel(self.model)
      self.load_data()
      self.tableView.clicked.connect(self.show_selected)
      self.DeleteUser.clicked.connect(self.AdmDeleteUser)
      self.UpdateDetails.clicked.connect(self.AdmUpdate)
      self.LogOut.clicked.connect(self.ReturnLogin)
      self.ViewStats.clicked.connect(self.Stats)

   def load_data(self):

         con = lite.connect('testerdb.db')#opens a file
         cur = con.cursor()#keeps track of the "tape" position
         #parameter query - get criteria from Python
         s='select user, forename, surname, gender, emailaddress, dob from profile' #not case
sensitive inside the string
         cur.execute(s)
##          s='''SELECT Title,Artistid FROM '''+table_name+''' WHERE Artistid=? ;'''
##          cur.execute(s,(my_id,)) #run our query
         self.data = cur.fetchall() #query data comes back
         if len(self.data)>0:
            for i in range(len(self.data)-1,-1):
               print("removing row",i)
               self.data.pop(i)
               self.model.removeRow(index.row(self.data[i]))
         self.model=QtGui.QStandardItemModel(self)
         self.tableView.setModel(self.model)
         for row in self.data:
                  items = [
                     QtGui.QStandardItem(str(field))
                     for field in row
                  ]
                  self.model.appendRow(items)
         #print(self.model.data(self.model.index(2,1)))

   def show_selected(self):
         index = self.tableView.currentIndex() #returns currently selected cell
         r=index.row() #returns the row of the currently selected cell
         c=index.column()#returns the column of the currently selected cell
         cell_contents=index.data()#returns the contents of the currently selected cell
         print("index,r,c",r,c,cell_contents)
         rowname=""
```

```
        if c==0:
            rowname="Username"
        if c==1:
            rowname="Forename"
        if c==2:
            rowname="Surname"
        if c==3:
            rowname="Gender"
        if c==4:
            rowname="Email Address"
        if c==5:
            rowname="DOB"
        message=("Current "+rowname+" is "+cell_contents+", please double click slots to
update")
        self.changelabel.setText(message)
##          self.lbl_row.setText(str(r) )
##          self.lbl_col.setText(str(c) )

    def AdmUpdate(self):
        index = self.tableView.currentIndex()
        cell_contents=index.data()
        ids=self.model.data(self.model.index(index.row(), 0))
        print(ids,self.model.data(self.model.index(index.row(), 1)))
        name=self.model.data(self.model.index(index.row(), 1))
        surnameedit=self.model.data(self.model.index(index.row(), 2))
        genderedit=self.model.data(self.model.index(index.row(), 3))
        emaileditinput=self.model.data(self.model.index(index.row(), 4))
        self.model.data(self.model.index(4,1))
        con = lite.connect('testerdb.db')
        cur = con.cursor()
        cur.execute("UPDATE profile set
forename='%s',surname='%s',gender='%s',emailaddress='%s'  WHERE
user=('%s')"%(name,surnameedit,genderedit,emaileditinput,ids))
        con.commit()
        self.load_data()

    def ReturnLogin(self):
        TheFirstWindow.show()
        self.hide()

    def Stats(self):
        cur.execute("SELECT physique FROM profile")
        rows = cur.fetchall()
        print(len(rows),rows[0][0])
        data=[]
        tempW=[]
        tempS=[]
```

```
    tempGG=[]
    WarriorCount=0
    GGCount=0
    SuperheroCount=0
    for each in rows:
       if each[0]=="Warrior":
          WarriorCount=WarriorCount+1
       if each[0]=="Superhero":
          SuperheroCount=SuperheroCount+1
       if each[0]=="GreekGod":
          GGCount=GGCount+1
    tempW.append("Warrior")
    tempW.append(WarriorCount)
    tempS.append("Superhero")
    tempS.append(SuperheroCount)
    tempGG.append("GreekGod")
    tempGG.append(GGCount)
    data.append(tempW)
    data.append(tempS)
    data.append(tempGG)


    print(data)


    top = Tk() #start a window
    lbl=Label(top,text="Admin stats").pack() #siple label made visible

    lbl1 = ttk.Label(top,wraplength="4i", justify="center", anchor="n",
            padding=(10, 2, 10, 6), font=("Arial",20),text="Current physiques")
    lbl1.pack(fill='x')#ttkinter label has a few more options
    f=Frame(top,width=280,height=1)#set up a frame to hold the treeview (optional)
    f.pack(side=LEFT)#make it visible

    tree = ttk.Treeview(f,padding=(10),show="headings")#set up a tree with headings on
    tree["columns"]=("name","prop") #set up column tags we can refer to
programmatically
    tree.column("name", width=100) #specify width of a column
    tree.column("prop", width=150,anchor='center') #as above

    tree.heading("name", text="Physiques") #specify the caption for a column
    tree.heading("prop", text="Peoples choice of physique") #as above

    grouping="" #our grid is not a hierarchical (not like folders), so no grouping
    for x in data: #iterate through our list, adding rows,
       print(x[0]) #feedback to programmer what's added, remove for final product
       tree.insert(grouping, "end", x[0],text="",values=tuple(x[0:]))
```

```
        tree.pack() #make tree visible

        p=pie.Pie(data,top) #set up our custom widget from another file
        p.draw() #invoke its custom method
        top.mainloop() #launch the window



    def AdmDeleteUser(self):
        event=False
        index = self.tableView.currentIndex()
        cell_contents=index.data()
        ids=self.model.data(self.model.index(index.row(), 0))

        if checkdelete(self,ids,event)==True:
            cur.execute("DELETE FROM profile where user=('%s')"%ids)
            con.commit()
            deletesuccessful(self)
            self.AdmUpdate()

##    def AdmUpdate(self):
##        print("admin update")


class PassWind(QtGui.QMainWindow, GetPass):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.CheckButton.clicked.connect(self.CheckBut)
        self.Home.clicked.connect(self.ReturnLogin)
        self.label_2.setText("Check username")

    def CheckBut(self):
        counter=0
        self.Username=self.UsernameInput.text()
        self.SecurityAns=self.SecurityAnswer.text()
        securityqs=[]
        profdet=[]
        cur.execute("SELECT forename, emailaddress, SecurityQ, SecurityA  FROM profile
where user='%s'"%self.Username)
        rows = cur.fetchall()
        for row in rows:
            profdet.append(row)
        try:
            bq=(profdet[0][2])
            ba=(profdet[0][3])
            emailad=(profdet[0][1])
```

```
            name=(profdet[0][0])
        except:
            errorusername(self)
        else:
            pass
            counter=counter+1
        self.label_2.setText("Security Question: %s"%bq)
        self.SecurityAnswer.show()

        counter=counter+1
        if self.SecurityAns==ba:
            resetpass=send(emailad,name)
            cur.execute("UPDATE profile set password='%s' where
user=('%s')"%(resetpass,self.Username))
            con.commit()
            changesuccessful(self)
            self.label_2.setText("Check your username in our database")



        else:
            if counter>1:
                wronganswer(self)

    def ReturnLogin(self):
        GetPass.UsernameInput.clear()
        GetPass.SecurityAnswer.clear()
        TheFirstWindow.show()
        self.hide()




class RegisterWindow(QtGui.QMainWindow, registerwin):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.HomeButton.clicked.connect(self.ReturnLogin)
        self.RegisterButton_2.clicked.connect(self.ContinueRegister)
    def ContinueRegister(self):
        global reguser
        global regpass
        global emailaddress

        reguser=self.UsernameInput.text()
        regpass=self.PasswordInput.text()
```

```
        emailaddress=self.EmailAddressInput.text()
        repass=self.ReEnterPasswordInput.text()
        checking(reguser,emailaddress,self)
        checkingpass(regpass,repass,self)

        resultans=checking(reguser,emailaddress,self)
        resultans1=checkingpass(regpass,repass,self)

        if resultans and resultans1==True:
            RegWind.UsernameInput.clear()
            RegWind.PasswordInput.clear()
            RegWind.EmailAddressInput.clear()
            RegWind.ReEnterPasswordInput.clear()
            ConRegWin.show()
            self.hide()


        else:
            print("re-enter")


        ConRegWin.A_ForenameInput.clear()
        ConRegWin.B_SurnameInput.clear()
        ConRegWin.SecurityAnswer.clear()
        ConRegWin.comboBox.setCurrentIndex(0)
        ConRegWin.AlphaMale.setChecked(False)
        ConRegWin.Goddess.setChecked(False)
        ConRegWin.GreekGod.setChecked(False)
        ConRegWin.Superhero.setChecked(False)
        ConRegWin.Warrior.setChecked(False)
        ConRegWin.E_WeightInput.setValue(65.00)
        ConRegWin.F_HeightInput.setValue(165.00)
##      ConRegWin.dateb.setDate(20/Jan/2001)

    def ReturnLogin(self):
        RegWind.UsernameInput.clear()
        RegWind.PasswordInput.clear()
        RegWind.EmailAddressInput.clear()
        RegWind.ReEnterPasswordInput.clear()
        TheFirstWindow.show()
        self.hide()




class ContinuedRegisterWindow(QtGui.QMainWindow, RegisterContinued):
    def __init__(self, parent=None):
```

```
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.HomeButton.clicked.connect(self.ReturnLogin)
        self.RegisterButton.clicked.connect(self.RegToHome)
    def ReturnLogin(self):
        TheFirstWindow.show()
        self.hide()
    def RegToHome(self):
        nextwind=0
        self.regname=self.A_ForenameInput.text()
        self.regsurname=self.B_SurnameInput.text()
        self.SecurityAns=self.SecurityAnswer.text()
        SecurityQ1=self.comboBox.currentText()

        if 2>=len(self.regname) or len(self.regname)>10:
            nextwind=nextwind-1
            nameerror(self)
        if 2>len(self.regsurname) or len(self.regsurname)>15:
            nextwind=nextwind-1
            surnameerror(self)

        prohibchar=["¬","`","¦","%","^","&","*","(",")","_","-
","+","=","[","]","{","}",":",";","'","~","#","?","/","<",">",".",",","|"]
        for each in self.regname:
            for l in prohibchar:
                if l==each:
                    prohibcharsname(self)
                    nextwind=nextwind-1
        for each in self.regsurname:
            for l in prohibchar:
                if l==each:
                    prohibcharsname(self)
                    nextwind=nextwind-1
#####################################
        tempdob=self.dateb.date()
        dob=str(tempdob.toPyDate())
        dobyear=dob[0:4]
        todaysDate=time.strftime("%Y/%m/%d)")
        year=todaysDate[0:4]
        personsage=int(year)-int(dobyear)


        if 16>personsage or personsage>65:
            nextwind=nextwind-1
            ageoutofrange(self)
```

```
        male=self.AlphaMale.isChecked()
        female=self.Goddess.isChecked()

        if male==True:
           regender=("Male")
        if female==True:
           regender=("Goddess")

        if male ==False and female == False:
           notpickedgen(self)
           nextwind=nextwind-1

        gg=self.GreekGod.isChecked()
        sh=self.Superhero.isChecked()
        Warrior=self.Warrior.isChecked()

        if gg==True:
           physiq=("GreekGod")
        if sh==True:
           physiq=("Superhero")
        if Warrior==True:
           physiq=("Warrior")

        if gg==False and sh==False and Warrior==False:
           notpickedphysique(self)
           nextwind=nextwind-1


        weight=self.E_WeightInput.text()
        height=self.F_HeightInput.text()
##         printweight,height)
        if ("40")>weight or weight>("200"):
           nextwind=nextwind-1
           weighterror(self)
        if ("120")>height or height>("250"):
           nextwind=nextwind-1
           heighterror(self)

        if SecurityQ1==("Select"):
           nextwind=nextwind-1
           secureque(self)
        if self.SecurityAns==(""):
           secureans(self)
           nextwind=nextwind-1
```

```
        if nextwind==0:
            successlogin(self)
            TheFirstWindow.show()
            self.hide()
            cur.execute("INSERT INTO profile (user, password, emailaddress, forename, surname,
gender, physique,weight, height,dob,SecurityQ, SecurityA) VALUES (?,?,?,?,?,?,?,?,?,?,?,?)",
(reguser,regpass,emailaddress, self.regname,self.regsurname,regender,physiq, weight,
height,dob,SecurityQ1,self.SecurityAns))
            con.commit()




class HomeWindow(QtGui.QMainWindow, homewin):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.ProfileButton.clicked.connect(self.ToProfileScr)
        self.ChangePasswordBut.clicked.connect(self.ChangePassword)
        self.WorkoutsButton.clicked.connect(self.ToWorkoutsScr)
        self.Logout.clicked.connect(self.ReturnLogin)
        self.NutritionButton.clicked.connect(self.ToNutrition)
        self.calendarWidget.clicked.connect(self.showworkouts)
    def ToProfileScr(self):
        ProfScreen.show()
        profdet=[]
        cur.execute("SELECT * FROM profile where user='%s'"%usernome)
        rows = cur.fetchall()
        for row in rows:
            profdet.append(row)
        useresults = [row[0] for row in profdet]
        userfore = [row[2] for row in profdet]
        usersurname = [row[3] for row in profdet]
        gender = [row[4] for row in profdet]
        email = [row[5] for row in profdet]
        dob = [row[6] for row in profdet]
        weight = [row[7] for row in profdet]
        height = [row[8] for row in profdet]
        physique = [row[9] for row in profdet]
##      var_name=ProfScreen.dateb.date()
##      print (var_name)
        ProfScreen.UsernameView.setText(useresults[0])
        ProfScreen.ForenameView.setText(userfore[0])
        ProfScreen.SurnameView.setText(usersurname[0])
```

```
        ProfScreen.GenderView.setText(gender[0])
        ProfScreen.AgeView.setText(dob[0])
        ProfScreen.WeightView.setText(str(weight[0]))
        ProfScreen.HeightView.setText(str(height[0]))
        ProfScreen.PhysiqueView.setText(physique[0])
        self.hide()
    def showworkouts(self):
        profdet=[]
        cur.execute("SELECT workoutoption, workoutA, workoutB, workoutC FROM profile
where user='%s'"%usernome)
        rows = cur.fetchall()
        for row in rows:
            profdet.append(row)
##          print(row)
        workoutoption = [row[0] for row in profdet]
        workoutday=[row[1:] for row in profdet]

        global AWorkout
        global BWorkout
        global CWorkout

        AWorkout=[["bench","situps","shoulder press"],["squats","deadlift","RDL", "split
squats"],["pull ups","sit ups"]]
        BWorkout=[["reverse flyes", "incline bench", "wide grip pull ups"],["squat", "romainian
deadlifts","reverse lat pull downs","overhead press"],["calf raises","barbell rows","cardio
machine","skull crushers"]]
        CWorkout=[["tricep pushdowns","skullcrushers","barbell curls","dumbell
curls"],["dumbell incline chest press","decline bench press","dumbell pullover", "reverse
delt flyes"],["deadlift","squat","30 minutes cardio","tricep rope pushdowns"]]

        d=self.calendarWidget.selectedDate()
        lol=d.toString()
        lol2=lol[:3]
        self.WorkoutView.setText("Rest day")
        daysoftheweek=[["Mon"],["Tue"],["Wed"],["Thu"],["Fri"],["Sat"],["Sun"]]
        condition=False

        nextwork=0
        for type in range (len(workoutday[0])):
            nxtday=type
            tt=(workoutday[0][type])
            for i in range (len(daysoftheweek)):
                day=daysoftheweek[i][0]
                if lol2==(day) and day==tt:
                    condition=True
                    if workoutoption[0]==("A"):
                        self.WorkoutView.setText(str(AWorkout[type]))
```

```
                try:
                    nxtday=type+1
                    nextwork=str(AWorkout[type+1])
                except:
                    nxtday=0
                    nextwork=str(AWorkout[0])
                else:
                    pass
##                      print(nextwork)

            if workoutoption[0]==("B"):
                self.WorkoutView.setText(str(BWorkout[type]))
            if workoutoption[0]==("C"):
                self.WorkoutView.setText(str(CWorkout[type]))

            print(nxtday)
          elif condition==False:
            self.WorkoutView.setText("Rest day")
##
##      my_date = date.today()
##      weekday= calendar.day_name[my_date.weekday()]
##      myweekday=weekday[:3]
##      print(myweekday)




################################################################################
#######
##   def __init__(self, *args):
##      QCalendarWidget.__init__(self, *args)
##      self.color = QColor(self.palette().color(QPalette.Highlight))
##      self.color.setAlpha(64)
##      self.selectionChanged.connect(self.updateCells)
##
##   def paintCell(self, painter, rect, date):
##
##       QCalendarWidget.paintCell(self, painter, rect, date)
##
##       first_day = self.firstDayOfWeek()
##       last_day = first_day + 6
##       current_date = self.selectedDate()
##       current_day = current_date.dayOfWeek()
##
##       if first_day <= current_day:
```

```
##            first_date = current_date.addDays(first_day - current_day)
##        else:
##            first_date = current_date.addDays(first_day - 7 - current_day)
##        last_date = first_date.addDays(6)
##
##        if first_date <= date <= last_date:
##            painter.fillRect(rect, self.color)
#############################################################################
#
    def ToWorkoutsScr(self):
        cur.execute("SELECT physique, goal FROM profile where user='%s'"%usernome)
        profdet=[]
        rows = cur.fetchall()
        for row in rows:
            profdet.append(row)
        userphysique = [row[0] for row in profdet]
        usergoal = [row[1] for row in profdet]

        WorksScreen.Goal.setText(usergoal[0])
        WorksScreen.Physique.setText(userphysique[0])
##        WorksScreen.NextWorkout.setText(nxtworkoutday[0])
        WorksScreen.Split.setText("3 day")
        WorksScreen.show()
        self.hide()
    def ReturnLogin(self):
        TheFirstWindow.show()
        LogoutVar=True
        self.hide()
    def ChangePassword(self):
        CallPassVerify.EnterUser.clear()
        CallPassVerify.Password.clear()
        CallPassVerify.show()
        self.hide()
    def ToNutrition(self):
        Nutri.Under.setStyleSheet('color: black')
        Nutri.Normal.setStyleSheet('color: black')
        Nutri.Over.setStyleSheet('color: black')
        Nutri.Obese.setStyleSheet('color: black')
        profdet=[]
        cur.execute("SELECT weight,height,calories, projected, lifestyle, goal, speed FROM
profile where user='%s'"%usernome)
        rows = cur.fetchall()
        #This loops through each of the rows in the database and then splits them
        #each row is then split into its own variable so that it can be named
        #and sorted into the correct information
        for row in rows:
            profdet.append(row)
```

```
tempweight = [row[0] for row in profdet]
tempheight = [row[1] for row in profdet]
kcalstemp = [row[2] for row in profdet]
project= [row[3] for row in profdet]
lifestyle = [row[4] for row in profdet]
goal = [row[5] for row in profdet]
speed = [row[6] for row in profdet]
weight=tempweight[0]
kcals=(str(kcalstemp[0]))
height=(tempheight[0]/100)
BMI=round((weight/height)/height)
#This selection is to determine the users BMI and highlight this. This checks
#if it is within the limits and then sets the colour for this
if BMI<18.5:
    Nutri.Under.setStyleSheet('color: red')
if 18.5<= BMI <= 24.9:
    Nutri.Normal.setStyleSheet('color: red')
if 25<= BMI <=29.9:
    Nutri.Over.setStyleSheet('color: red')
if BMI>30:
    Nutri.Obese.setStyleSheet('color: red')
Nutri.BMIView.setText(str(BMI))
Nutri.CalorieView.setText(str(kcals))
Nutri.ProjectedView.setText(str(project[0]))
if lifestyle[0]==("1"):
    Nutri.Sedentary.setChecked(True)
if lifestyle[0]==("2"):
    Nutri.Moderate.setChecked(True)
if lifestyle[0]==("3"):
    Nutri.Active.setChecked(True)
if goal[0]==("Cut"):
    Nutri.Cut.setChecked(True)
if goal[0]==("Bulk"):
    Nutri.Bulk.setChecked(True)
if goal[0]==("Maintain"):
    Nutri.Maintain.setChecked(True)
if speed[0]==("Aggressive"):
    Nutri.Aggressive.setChecked(True)
if speed[0]==("Slow"):
    Nutri.Slow.setChecked(True)
if speed[0]==("Balanced"):
    Nutri.Balanced.setChecked(True)
else:
    firsttime(self)
```

```
        Nutri.show()
        self.hide()

class PassScrVerify(QtGui.QMainWindow, ChngPassVerify):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.ReturnButton.clicked.connect(self.RegToHome)
        self.ChangeButton.clicked.connect(self.RegToHome2)
    def RegToHome(self):
        HomeWind.label_2.setText("Welcome, %s"%name)
        HomeWind.show()
        self.hide()
    def RegToHome2(self):
        reguserr=self.EnterUser.text()
        repass=self.Password.text()
        s=cur.execute("SELECT password FROM profile where user='%s'"%usernome)
        rows = cur.fetchall()

        if reguserr==usernome:
            if repass==(rows[0][0]):
                CallPass.EnterPassInput.clear()
                CallPass.PasswordAgain.clear()
                CallPass.show()
                self.hide()
        else:
            error(self)

##      welcome(self)



class PassScr(QtGui.QMainWindow, ChngPass):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.ReturnButton.clicked.connect(self.RegToHome)
        self.ChangeButton.clicked.connect(self.RegToHome2)

    def RegToHome(self):
        HomeWind.label_2.setText("Welcome, %s"%name)
        HomeWind.show()
        self.hide()
    def RegToHome2(self):
        regpass=self.EnterPassInput.text()
```

```
        repass=self.PasswordAgain.text()
        if checkingpass(regpass,repass,self)==True:
            cur.execute("UPDATE profile set password='%s' where
user=('%s')"%(regpass,usernome))
            con.commit()
            changesuccessful(self)
            HomeWind.label_2.setText("Welcome, %s"%name)
            HomeWind.show()
            self.hide()



class ProfWindow(QtGui.QMainWindow, profilescreen):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.ReturnButton.clicked.connect(self.RegToHome)
        self.ChangeButton.clicked.connect(self.ChangeProfile)

#       print(LoginWindow.Username)
#       self.UsernameView.setText("CurrentUser")
    def RegToHome(self):
        HomeWind.label_2.setText("Welcome, %s"%name)
        HomeWind.show()
        print(usernome)
        self.hide()
    def ChangeProfile(self):
        ProfChange.show()
        ###############
        cur.execute("SELECT * FROM profile where user='%s'"%usernome)
        con.commit()
        profdet=[]
        rows = cur.fetchall()
        for row in rows:
            profdet.append(row)
        useresults = [row[2] for row in profdet]
        usersurname = [row[3] for row in profdet]
        gender = [row[4] for row in profdet]
        email = [row[5] for row in profdet]
        dob = [row[6] for row in profdet]
        weight = [row[7] for row in profdet]
        height = [row[8] for row in profdet]
        tempphysique = [row[9] for row in profdet]
        physique= tempphysique[0]
        ProfChange.GreekGod.setChecked(False)
        ProfChange.Superhero.setChecked(False)
```

```
        ProfChange.Warrior.setChecked(False)
        ProfChange.UsernameView.setText(usernome)
        ProfChange.ForenameInput.setText(useresults[0])
        ProfChange.SurnameInput.setText(usersurname[0])
        ProfChange.GenderView.setText(gender[0])
        ProfChange.AgeView.setText(dob[0])
        ProfChange.E_WeightInput.setValue(weight[0])
        ProfChange.F_HeightInput.setValue(height[0])
        if physique==("Superhero"):
            ProfChange.Superhero.setChecked(True)
        if physique==("Warrior"):
            ProfChange.Warrior.setChecked(True)
        if physique==("GreekGod"):
            ProfChange.GreekGod.setChecked(True)


##        ProfChange.PhysiqueView.setText(physique[0])
        ################


        self.hide()

class WorkoutWindow(QtGui.QMainWindow, workoutwindow):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.HomeButton.clicked.connect(self.RegToHome)
        self.RescheduleButton.clicked.connect(self.gotoworkouts)
        self.ChangeButton.clicked.connect(self.ToWorkoutsScr)
        self.PrintButton.clicked.connect(self.printing)
    def ToWorkoutsScr(self):
        WorkoutScrChange.show()
        self.hide()
    def gotoworkouts(self):
        global splitdays
        splitdays=[]
        Workoutsplit.show()
        self.hide()
    def RegToHome(self):
        HomeWind.label_2.setText("Welcome, %s"%name)
        HomeWind.show()
        self.hide()
    def printing(self):
        cur.execute("SELECT workoutoption FROM profile where user='%s'"%usernome)
        con.commit()
        profdet=[]
        rows = cur.fetchall()
```

```
        for row in rows:
            profdet.append(row)
        AWorkout=[["bench","situps","shoulder press"],["squats","deadlift","RDL", "split
squats"],["pull ups","sit ups"]]
        BWorkout=[["reverse flyes", "incline bench", "wide grip pull ups"],["squat", "romainian
deadlifts","reverse lat pull downs","overhead press"],["calf raises","barbell rows","cardio
machine","skull crushers"]]
        CWorkout=[["tricep pushdowns","skullcrushers","barbell curls","dumbell
curls"],["dumbell incline chest press","decline bench press","dumbell pullover", "reverse
delt flyes"],["deadlift","squat","30 minutes cardio","tricep rope pushdowns"]]
        option=[row[0] for row in profdet]

        if (option[0])==("A"):
            printwin.editor.setText(str(AWorkout))
        elif (option[0])==("B"):
            printwin.editor.setText(str(BWorkout))
        elif (option[0])==("C"):
            printwin.editor.setText(str(CWorkout))
        else:
          printwin.editor.setText("Select a workout routine")

        printwin.show()

class printwork(QtGui.QWidget):
    def __init__(self):
        QtGui.QWidget.__init__(self)
        self.setWindowTitle(self.tr('Document Printer'))
        self.editor = QtGui.QTextBrowser(self)
        self.buttonPrint = QtGui.QPushButton('Print', self)
        self.buttonPrint.clicked.connect(self.handlePrint)
        self.buttonPreview = QtGui.QPushButton('Preview', self)
        self.buttonPreview.clicked.connect(self.handlePreview)
        layout = QtGui.QGridLayout(self)
        layout.addWidget(self.editor, 0, 0, 1, 3)
        layout.addWidget(self.buttonPrint, 1, 0)
        layout.addWidget(self.buttonPreview, 1, 1)


    def handlePrint(self):
        dialog = QtGui.QPrintDialog()
        if dialog.exec_() == QtGui.QDialog.Accepted:
            self.editor.document().print_(dialog.printer())

    def handlePreview(self):
        dialog = QtGui.QPrintPreviewDialog()
        dialog.paintRequested.connect(self.editor.print_)
        dialog.exec_()
```

```
class ProfChangeWin(QtGui.QMainWindow, Profilechangingwin):
   def __init__(self, parent=None):
      QtGui.QMainWindow.__init__(self, parent)
      self.setupUi(self)
      self.ChangeButton.clicked.connect(self.ToProfileScr1)
      self.ReturnButton.clicked.connect(self.ToProfileScr)
   def ToProfileScr(self):
      ProfScreen.show()
      self.hide()
   def ToProfileScr1(self):
      global chngfore
      chngfore=ProfChange.ForenameInput.text()
      chngsur=ProfChange.SurnameInput.text()
      gg=self.GreekGod.isChecked()
      sh=self.Superhero.isChecked()
      Warrior=self.Warrior.isChecked()
      if gg==True:
         physiq=("GreekGod")
      if sh==True:
         physiq=("Superhero")
      if Warrior==True:
         physiq=("Warrior")
      weight=self.E_WeightInput.text()
      height=self.F_HeightInput.text()
      cur.execute("UPDATE profile set forename='%s', surname='%s', weight='%s',
height='%s', physique='%s' where
user=('%s')"%(chngfore,chngsur,weight,height,physiq,usernome))
      con.commit()
      changesuccessful(self)
      HomeWind.label_2.setText("Welcome, %s"%chngfore)
      HomeWind.show()
      self.hide()

class WorkSplit(QtGui.QMainWindow, Workoutsplit):
   def __init__(self, parent=None):
      QtGui.QMainWindow.__init__(self, parent)
      self.setupUi(self)
      self.Mon.clicked.connect(self.Mondayclick)
      self.Tue.clicked.connect(self.Tuesdayclick)
      self.Wednesday.clicked.connect(self.Wednesdayclick)
      self.Thursday.clicked.connect(self.Thursdayclick)
      self.Friday.clicked.connect(self.Fridayclick)
      self.Saturday.clicked.connect(self.Saturdayclick)
      self.Sunday.clicked.connect(self.Sundayclick)
      self.Back.clicked.connect(self.ToWorkoutsScr)
```

```
    self.Clear.clicked.connect(self.clearsel)
    self.Save.clicked.connect(self.save)

def Mondayclick(self):
    splitdays.append("Mon")
    self.Mon.hide()
    self.SplitView.setText(str(splitdays))
def Tuesdayclick(self):
    splitdays.append("Tue")
    self.Tue.hide()
    self.SplitView.setText(str(splitdays))
def Wednesdayclick(self):
    splitdays.append("Wed")
    self.Wednesday.hide()
    self.SplitView.setText(str(splitdays))
def Thursdayclick(self):
    splitdays.append("Thu")
    self.Thursday.hide()
    self.SplitView.setText(str(splitdays))
def Fridayclick(self):
    splitdays.append("Fri")
    self.Friday.hide()
    self.SplitView.setText(str(splitdays))
def Saturdayclick(self):
    splitdays.append("Sat")
    self.Saturday.hide()
    self.SplitView.setText(str(splitdays))
def Sundayclick(self):
    splitdays.append("Sun")
    self.Sunday.hide()
    self.SplitView.setText(str(splitdays))
def ToWorkoutsScr(self):
    HomeWind.ToWorkoutsScr()
    self.hide()
    self.Mon.show()
    self.Tue.show()
    self.Wednesday.show()
    self.Thursday.show()
    self.Friday.show()
    self.Saturday.show()
    self.Sunday.show()

def clearsel(self):
    WorksScreen.gotoworkouts()
    self.SplitView.setText(str(splitdays))
    self.Mon.show()
    self.Tue.show()
```

```
        self.Wednesday.show()
        self.Thursday.show()
        self.Friday.show()
        self.Saturday.show()
        self.Sunday.show()
    def save(self):
        if len(splitdays)>3:
            toomanydays(self)
        if len(splitdays)<3:
            lessmanydays(self)
        save(self)
        workA=splitdays[0]
        workB=splitdays[1]
        workC=splitdays[2]
        cur.execute("UPDATE profile set workoutA='%s', workoutB='%s', workoutC='%s' where
user=('%s')"%(workA,workB,workC,usernome))
        con.commit()
##        for each in splitdays:
##        string=string.join(each)
##        print(string)
##        for each in splitdays:
##            print(each)
##
##        cur.execute("UPDATE profile set workoutdays='%s' where
user=('%s')"%(splitdays,usernome))
##        con.commit()



class WorkoutScrChng(QtGui.QMainWindow, WorkoutScrChange):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.HomeButton.clicked.connect(self.RegToHome)
        self.PickC.clicked.connect(self.PickCChoice)
        self.PickB.clicked.connect(self.PickBChoice)
        self.PickA.clicked.connect(self.PickAChoice)
    def PickCChoice(self):
        option=("C")
        cur.execute("UPDATE profile set workoutoption='%s' where
user=('%s')"%(option,usernome))
        con.commit()
        changessaved(self)
        WorksScreen.show()
        self.hide()
    def PickBChoice(self):
        option=("B")
```

```
        cur.execute("UPDATE profile set workoutoption='%s' where
user=('%s')"%(option,usernome))
        con.commit()
        changessaved(self)
        WorksScreen.show()
        self.hide()
    def PickAChoice(self):
        option=("A")
        cur.execute("UPDATE profile set workoutoption='%s' where
user=('%s')"%(option,usernome))
        con.commit()
        changessaved(self)
        WorksScreen.show()
        self.hide()




    def gotoworkouts(self):
        Workoutsplit.show()
        self.hide()
    def RegToHome(self):
        WorksScreen.show()
        self.hide()









class NutrScr(QtGui.QMainWindow, NutritionScr):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.ChangeButton.clicked.connect(self.calorieintake)
        self.ReturnButton.clicked.connect(self.RegToHome)
    def calorieintake(self):
        Sedentarys=self.Sedentary.isChecked()
        Moderately=self.Moderate.isChecked()
        VActive=self.Active.isChecked()
        cut=self.Cut.isChecked()
        bulk=self.Bulk.isChecked()
        maintain=self.Maintain.isChecked()
        aggressive=self.Aggressive.isChecked()
        slow=self.Slow.isChecked()
```

```
balanced=self.Balanced.isChecked()
if Sedentarys==True:
   life=1
   multiplier=14
if Moderately==True:
   life=2
   multiplier=16
if VActive==True:
   life=3
   multiplier=18
if cut==True:
   goal=("Cut")
if bulk==True:
   goal=("Bulk")
if maintain==True:
   goal=("Maintain")
if aggressive==True:
   speed=("Aggressive")
if balanced==True:
   speed=("Balanced")
if slow==True:
   speed=("Slow")
cur.execute("SELECT weight FROM profile where user='%s'"%usernome)
s=cur.fetchall()
weightlbs=((s[0][0])*2.2)
calorieintake=round(weightlbs*multiplier)
if goal==("Cut"):
      calorieintake=calorieintake-200
      if speed==("Aggressive"):
         calorieintake=calorieintake-100
         projectedweight=("-5lbs/week")
      elif speed==("Slow"):
         calorieintake=calorieintake+50
         projectedweight=("-1lb/week")
      else:
         projectedweight=("3lbs/week")
if goal==("Bulk"):
      calorieintake=calorieintake+200
      if speed==("Aggressive"):
         calorieintake=calorieintake+100
         projectedweight=("+5lbs/week")
      elif speed==("Slow"):
         calorieintake=calorieintake-50
         projectedweight=("+1lb/week")
      else:
         projectedweight=("+3lbs/week")
if goal==("Maintain"):
```

```
            projectedweight=("+0lbs/week")

        messagetemp=(str(calorieintake))
        message=(messagetemp+ "kcals")
        self.CalorieView.setText(message)
        self.ProjectedView.setText(projectedweight)
        cur.execute("UPDATE profile set calories='%s', projected='%s', lifestyle='%s', goal='%s',
speed='%s' where user=('%s')"%(message,projectedweight,life,goal,speed, usernome))
        con.commit()

    def RegToHome(self):
        HomeWind.label_2.setText("Welcome, %s"%name)
        HomeWind.show()
        self.hide()


app = QtGui.QApplication(sys.argv)
TheFirstWindow = LoginWindow(None)
RegWind = RegisterWindow(None)  #THIS IS IMPORTANT
ConRegWin = ContinuedRegisterWindow(None)
HomeWind = HomeWindow(None)
ProfScreen = ProfWindow(None)
CallPass = PassScr(None)
CallPassVerify = PassScrVerify(None)
ProfChange = ProfChangeWin(None)
WorksScreen = WorkoutWindow(None)
Nutri= NutrScr(None)
GetPass=PassWind(None)
WorkoutScrChange=WorkoutScrChng(None)
Workoutsplit=WorkSplit(None)
AdminScreen=AdminScr(None)
printwin = printwork()
printwin.resize(640, 480)
##sys.exit(app.exec_()
TheFirstWindow.show()

app.exec_()
```

Error Check module

```
import sys, os
from PyQt4 import QtCore, QtGui, uic
import sqlite3 as lite

def wronganswer(self):
    QtGui.QMessageBox.information(self,"Error", "Incorrect answer, please re-enter answer")
def triesexceeded(self):
```

```
    QtGui.QMessageBox.information(self,"Error", "Maximum tries exceeded")
def prohibcharsuser(self):
    QtGui.QMessageBox.information(self,"Error", "Username contains prohibited
characters")
def prohibcharsname(self):
    QtGui.QMessageBox.information(self,"Error", "Name contains prohibited characters")
def error(self):
    QtGui.QMessageBox.information(self,"Error", "Username or password is incorrect")
def welcome(self):
    QtGui.QMessageBox.information(self,"Welcome", "Successful login")
def errorpaslenmin(self):
    QtGui.QMessageBox.information(self,"Error", "Password is less than 4 characters")
def erroruserlenmax(self):
    QtGui.QMessageBox.information(self,"Error", "Username is more than 20 characters")
def erroruserlenmin(self):
    QtGui.QMessageBox.information(self,"Error", "Username is less than 4 characters")
def errorpaslenmax(self):
    QtGui.QMessageBox.information(self,"Error", "Password is more than 12 characters")
def errorreg(self):
    QtGui.QMessageBox.information(self,"Error", "Passwords do not match")
def userexists(self):
    QtGui.QMessageBox.information(self,"Error", "Username is taken")
def emailexists(self):
    QtGui.QMessageBox.information(self,"Error", "This email address is already registered")
def passerror(self):
    QtGui.QMessageBox.information(self,"Error", "The password you entered contains
prohibited characters")
def emailerror(self):
    QtGui.QMessageBox.information(self,"Error", "The email you entered is incorrect")
def changesuccessful(self):
    QtGui.QMessageBox.information(self,"Success", "Your details have been updated")
def firsttime(self):
    QtGui.QMessageBox.information(self,"Welcome", "This is your first time here, please fill
out the details")
def ageoutofrange(self):
    QtGui.QMessageBox.information(self,"Issue", "Age out of range")
def weighterror(self):
    QtGui.QMessageBox.information(self,"Issue", "Weight is not within limits")
def heighterror(self):
    QtGui.QMessageBox.information(self,"Issue", "Height is not within limits")
def notpickedgen(self):
    QtGui.QMessageBox.information(self,"Issue", "Please pick your gender")
def surnameerror(self):
    QtGui.QMessageBox.information(self,"Error", "Surname is not between 2 and 15
characters")
def nameerror(self):
```

```
    QtGui.QMessageBox.information(self,"Error", "Forename is not between 2 and 15
characters")
def notpickedphysique(self):
    QtGui.QMessageBox.information(self,"Issue", "Please pick your prefered physique")
def errorusername(self):
    QtGui.QMessageBox.information(self,"Error", "Username is not valid")
def toomanydays(self):
    QtGui.QMessageBox.information(self,"Error", "You have picked too many days")
def lessmanydays(self):
    QtGui.QMessageBox.information(self,"Error", "You have not picked enough days")
def changessaved(self):
    QtGui.QMessageBox.information(self,"Success", "You have successfully picked your
workout")
def save(self):
    QtGui.QMessageBox.information(self,"Success", "Save sucessful")
def successlogin(self):
    QtGui.QMessageBox.information(self,"Successfully registered", "User saved. Login
again.")
def secureque(self):
    QtGui.QMessageBox.information(self,"Issue", "Please pick a security question")
def secureans(self):
    QtGui.QMessageBox.information(self,"Issue", "Please use a valid security answer")
def checkdelete(self,ids,event):
    deletemsg = "Are you sure you want to delted user '%s'?"%ids
    reply = QtGui.QMessageBox.question(self, 'Message',
                deletemsg, QtGui.QMessageBox.Yes, QtGui.QMessageBox.No)

    if reply == QtGui.QMessageBox.Yes:
        event=True
    else:
        event=False
    return event
def deletesuccessful(self):
    QtGui.QMessageBox.information(self,"Success", "User sucessfully deleted")
```

Email Checker

```
import re

addressToVerify ='kdippy6@gmail.co.uk'
match = re.match('^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,4})$',
addressToVerify)

if match == None:
    print('Bad Syntax')
```

Pie chart code

```
from tkinter import *
from math import sin, cos, pi
class Pie: #limited to 6 items of data, to use more - add more colours
   def __init__(self,data_in,top): #default constructor takes in the data and the parent
window
      self.top=top;  self.data=data_in
      print(self.data)  #just for debugging purposes
      self.colours=["red","brown","yellow","grey","white","purple"] #set up colours for 6
slices
      #the next line is interesting we have to map our data to "1 o'clock", "5 o'clock", etc.
      #in other words, to draw a slice, we need to know a starting angle and the ending angle
      #25 would correspond to a quarter of 360. We use nested list comprehension to extract
just
      #the 2nd column from a particular row of our nested list.
      self.refactor=[int(x*360/sum([y[1] for y in self.data])) for x in [y[1] for y in self.data]]
      print(self.refactor)
   def draw(self): #set up our method for actually drawing the data with slices/arcs
      self.C = Canvas(self.top, bg="white", height=380, width=250)#set up drawing area
      self.coord = 10, 50, 240, 280 #circle drawing takes 4 numbers - the "bounding box"
      self.w=self.coord[2]-self.coord[0]#working out the middle x
      self.h=self.coord[3]-self.coord[1]#working out the middle y - optional as equal to w
      self.s=0 #s is the cumulative angle of all the slices drawn so far - otherwise
      #they will be drawn on top of each other
      self.os=300 #under the slices we will draw our legend - rectangles+text starting at
y=300
      for j in range(len(self.refactor)):
         #drawing slices with filled arcs, beginning angle, ending angle, colour
         arc = self.C.create_arc(self.coord, start=self.s,
                        extent=self.refactor[j], fill=self.colours[j])
         self.s=self.s+self.refactor[j] #increment the cumulative angle
         #colour boxes for the legend
         self.C.create_rectangle(40, self.os-10, 60, self.os+10, fill=self.colours[j])
         #text to the right of the boxes
         self.txt = self.C.create_text(100,self.os,
                              text=(self.data[j][0]+": "+str(self.data[j][1])),
                                   justify=LEFT)
         self.os=self.os+20 #increment y so that boxes are not drawn over each other
      self.C.pack()#make the canvas visible
```

Registration Check

```
import sys, os
from PyQt4 import QtCore, QtGui, uic
import sqlite3 as lite
```

```
import re
from errorcheck import*

con=lite.connect('testerdb.db') #connect sql file
cur=con.cursor() # as above

def checking(reguser,emailaddress,self):
    prohibchar=["¬","`","¦","%","^","&","*","(",")","_","-
","+","=","[","]","{","}",":",";","'","~","#","?","/","<",">",".",",","|"]
    profdet=[]
    checkuser=[]
    cur.execute("SELECT * FROM profile")
    rows = cur.fetchall()
    for row in rows:
        profdet.append(row)
    useresults = [row[0] for row in profdet]
    useremail = [row[5] for row in profdet]
    for x in useresults:
        checkuser.append(x)
    lenuser=len(reguser)
    check=0

    if lenuser<4:
        erroruserlenmin(self)
        check=check-1
    if lenuser>20:
        erroruserlenmax(self)
        check=check-1

    for each in checkuser:
        if reguser==each:
            userexists(self)
            check=check-1

    for each in reguser:
        for l in prohibchar:
            if l==each:
                prohibcharsuser(self)
                check=check-1

    for n in useremail:
        if emailaddress==n:
            emailexists(self)
            check=check-1

    match = re.match('^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,4})$',
emailaddress)
```

```
    if match == None:
        emailerror(self)
        check=check-1
    print(check)
    if check==0:
        return True



def checkingpass(regpass,repass,self):
    prohibchar=["¬","`","¦","%","^","&","*","(",")","_","-
","+","=","[","]","{","}",":",";","'","~","#","?","/","<",">",".",",","|"]
    profdet=[]
    checkuser=[]
    cur.execute("SELECT * FROM profile")
    rows = cur.fetchall()
    for row in rows:
        profdet.append(row)
    useresults = [row[0] for row in profdet]
    useremail = [row[5] for row in profdet]
    for x in useresults:
        checkuser.append(x)

    lenpas=len(regpass)
    check=0

    if lenpas<4:
        errorpaslenmin(self)
        check=check+1
    if lenpas>20:
        errorpaslenmax(self)
        check=check+1
    for m in prohibchar:
        if m in regpass:
            passerror(self)
            check=check+1
    if regpass!=repass:
        errorreg(self)
        check=check+1
    print(check)
    if check==0:
        return True
```

Sending email

```
def send(recipient,name):
```

```
import smtplib
import random
user = 'StatifyApp@gmail.com'
password = 'Statify123'
reset_password = (random.randint(0, 31415926))
to = [recipient]
body = """
      Dear %s,
      This message has been automated in response to a forgotten Statify password.
      Your new password is %s
      Once you have logged in again, please set a new password

""" % (name, reset_password)
msg = 'Subject: %s\n\n%s' % ('Statify profile: Reset your password', body)
try:
   server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
   server.ehlo()
   server.login(user, password)
   server.sendmail(user, to, msg)
   print("email sent! with password "+str(reset_password))
   return (reset_password)
   server.close()

except:
   print('Something went wrong...')
```

References

https://www.noo.org.uk/NOO_about_obesity/adult_obesity/UK_prevalence_and_trends