WJEC GCE Computing CG2 - Extended Task

Candidate Name: Daniel Roberts
Candidate Number: 4699
Centre Name: Shrewsbury Sixth Form College
Centre Number: 29285

Contents

Part I
Analysis and Design

This part of the documentation contains the analysis that was performed on Parkwood Vale Harriers, taking into account what the running club asked for in their brief, and exploring these requirements. It also covers the preliminary design that was created for the system, including the interface design for every page, the design of the data structures and process design, detailing the different algorithms that have been used, and how the system interacts with itself.

1 Problem Definition

1.1 Background

Parkwood Vale Harriers is a running club that serves the fitness needs of many different members, through regular training sessions, as well as races. The club gets involved in the local community, a position that consists, in part, of raising money for local charities.

Recently, the club has decided to raise money for one of the charities by putting on a relay event, wherein a team of runners will run, non-stop, from John O' Groats to Lands End, in the shortest time possible. The team will consist of eight members, and each runner will run for an hour at a time, whilst the others rest in the minibus. The entire trip is estimated to take three days and as a result of this, each member of the team will have to be very fit.

In order to increase their chances of completing the run, the club has decided to find out the most appropriate team, based on the results of a physically challenging training programme. This programme will consist of running, cycling and swimming, and will serve to ensure that only the top members of the club are included in the team.

1.2 Broad Aims

The running club has commissioned a computer based system that will allow the runners to keep an accurate record of their running, cycling and swimming sessions. This data will then be used to calculate an informed decision of the most appropriate team for the relay race.

The system must allow each runner to monitor their progress during the training programme, clearly showing them the extent to which they have improved. As such, the system must provide an interface to allow the runner to add each training session they perform, with spaces for the type of training, the time spent, how hard they pushed themselves, and other such parameters. Using this data, the system must then calculate the number of calories burned in the training session, providing a series of data points through which the performance of the runner can be monitored.

To further aid in this, the system must be able to output these training sessions in a clear format that the runner is able to clearly understand. This can be achieved through the use of tables to display each training session in a listed, tabular format, as well as through graphs and charts to display the data in a graphical form; this maes overall performance trends easy to visualise.

Due to the nature of the system, the ability to store certain personal information, such as the name, age and weight of the runner, must also be included. The runner should have the ability to input this information themselves, most likely upon first use of the system. There should be the ability to modify this data, in the result of an error being made or the circumstances of the runner changing.

An important aspect of the system, and one that is key to promoting the competitive values of the club, is the ability to compare results with other participants in the program. This area of the system should allow runners to compare key aspects of their performance, such as the results of their individual training sessions, as well as their overall performance over time in all three of the training activities.

As the main point of the system, the ability to select the final team must also be included. By analysing the data points provided by the runners, the system should be able to choose the most appropriate team.

1.3 Limitations

Though the brief provided by the running club contains several good ideas and acts as an effective base upon which to work, there are a number of areas which the running club has not thought about that could be factored into the solution, creating a more effective system.

One very important factor that the running club has left out is security. In a system like this, where intensely personal data is being stored, including data that the user may not which to become public, such as their weight, it is important that the data is stored in a secure manner that allows only those with the correct permissions to access it.

Another issue with the brief is that of an objective decision being made when selecting the team. Running a marathon is about far more than just physical fitness; more personal aspects, such as how well the runners get along and different roles within the team, should also be taken into account for maximum efficiency. The system would be unable to do this (without each runner giving their opinion on the others, which is unrealistic), and so the team it comes up with may not be the most appropriate choice.

Another limitation in the system is that data will have to be entered manually: there is no way of taking the data from some sort of personal tracking device. This could result in some issues with accuracy, or even with malpractice: people entering exaggerated data in order to manipulate the rankings and make themselves seem better. A mixture of validation and verification can be put in place to prevent this, such as ensuring users cannot go for a straight eight hour swim (something which is obviously unrealistic), but this will be unable to catch all cases of exaggeration; it is therefore necessary to rely on the goodwill and sportsmanship of the runners.

Furthermore, the system relies on the premise that the runners will add every training session they perform to the application. It is not unlikely that they will go on unsolicited training sessions that they do not bother adding, or they may simply forget. There is no foolproof manner to prevent these occurences, but a number of steps can be taken to reduce their likelihood, such as by making the process of adding a session as simple as possible - the easier the process is, the more likely the runner is to do it.

In addition, the brief asks for only the top eight members of the running team to be calculated. This does not take into account the possibilities of injuries or runners dropping out for other reasons; as such, the system should also calculate a number of reserve runners, in the event of an accident.

1.4 Assumptions

Throughout the system, a number of assumptions have been made in order to increase the ease of development.

One of these is that in each individual training session, only one method of exercise will be used, such as breaststroke for an entire swimming session or a leisurely speed for an entire cycling session. Though this is alleviated to some extent by the ability to add multiple sessions for each sport on a single day, the assumption still has to be made.

In addition to this, the assumption that each session lasts for at least an hour has been made: the time picker only uses stages of sixty minutes, as opposed to thirty or fifteen.

Naturally, the system also assumes that the user is relatively proficient with a computer based interface. Effort has been put in to make the system as user friendly and as easy to use as possible, but someone using a computer for the first time will undoubtedly find it more difficult than someone with at least a little experience.

1.5 Objectives

In order to create the system to an acceptable quality, a number of objectives will have to be fulfilled. The system must:

- Have a simple, clear interface that allows tasks to be performed easily.
- Allow the runner to add, view, update and, if they choose, delete their personal information, such as their name, email address, date of birth and phone number.
- Allow the runner to add, view and delete the training sessions they perform in over the course of the training period; this will include information like the date and time of the session, the speed they were training at, and how well it went.

- Persistently store this data in appropriately named tables in a database.
- Ensure the security of this data by giving each runner their own personal account, protected by a username and an encrypted password.
- Calculate the number of calories burned in each training session, by taking into account the runner's weight, the time spent on the session, the nature of the session, and how well the runner thought it went.
- Allow the user to view graphical, interactive graphs of their training sessions, allowing them to easily view trends in their performance.

1.6 Justification of Proposed Solution

When building a solution to a problem like the one faced by Parkwood Vale Harriers, there are generally two methods available: utilising the features of an existing software package, such as Microsoft Office Access, or programming an existing solution in a programming language, such as Visual Basic or Python. Both have their advantages and drawbacks: by utilising an existing package, much of the system will already be developed; it only remains to manipulate the system to meet the needs of the brief; but, on the other hand, one can be limited by the restrictions of the software package, perhaps preventing the final solution being as capable as it might otherwise have been.

An original solution created using a programming language would suffer from rather the opposite issues: as a result of the practically endless results that can be achieved through their use, there is a definite learning curve that is not present (or is less exacerbated) in software packages; as a result of this, development time will likely be considerably longer. Despite these drawbacks, it is clear that, if a programming language is used, the final solution is likely to be of a higher quality: not only can more advanced features be implemented, these features - as well as those of a more basic level - are likely to be of a higher quality. In addition, the developer will have a greater understanding of the system, as they will have built it entirely themselves (aside from any additional packages/libraries used); this will aid in areas like debugging, and will also make it easier to write up system documentation and the like.

The question then falls to exactly which programming language is the most appropriate. There are a large number of languages available, ranging from *compiled* languages like Java, C# and Visual Basic to *interpreted* languages like Ruby, Python and PHP. The differences between compiled and iterpreted languages are complex and varied, but, in essence, compiled languages are likely to perform algorithms more quickly (due to directly using the native code of the target machine), whereas code written in an interpreted language can be executed "on the fly", so to speak, increasing development speed.

2 Data Structures and Methods of Access

In order to persistently store the runner's data, a database is needed. As is the custom with applications of this sort, there will be one single database file, within which will be a number of tables. The system will also make use of a number of arrays and JSON structures, to temporarily store data.

2.1 Database Tables

The system will use the SQLite database system. SQLite is a very popular database system (in the same vein as MySQL). All of the database tables will be accessed sequentially - every item is ordered according to their primary key, which, as is custom for an SQLite database, is always an id number stored as an integer.

A note on validation: *SQLite does not perform any validation itself. All validation will be performed during the processing of the data, before it is added into the database. As such, details on the validation performed on the data saved to these tables can be found in their relevant section.*

2.1.1 Users Table

This table will store the personal information for each runner. Whenever a runner creates an account, the data they input into the registration form will end up in this table.

Field Name	**Primary Key**	**Typical Data**	**Data Type**
id	True	01	Integer
name	n/a	John Smith	String
email	n/a	john@smith.com	String
username	n/a	john5	String
password_hash	n/a	pbkdf2:sha1:1000$02	String
dob	n/a	1997-02-02	Date
phone	n/a	07722895880	String
weight	n/a	74	Integer
distance	n/a	less than 1	String
joined	n/a	2015-01-04	Date
charity_event	n/a	True	Boolean

Table 1: Users Table

Each user is given an id which serves as their primary key; it is automatically incremented whenever a new user is added, hence the data type of integer. The name is used as an identifier throughout the system; as a string of characters, it has been given the string data type. Likewise with the email field: it can contain a combination of letters, numbers and other characters, and so has been set as a string. The username field is a combination of the runner's first name and a random number; as such it is a string. The password hash field stores an encrypted version of the user's password; depending on the length of the password, it can contain a very large number of letters, numbers and symbols - it is therefore a string. The dob field stores the runner's date of birth, the most appropriate data type would therefore be date; likewise with the date the runner joined the application. No calculations are being performed on the runner's phone number, so it is more efficient to store it as a string - one character takes just 1 bit. Conversely, calculations are being performed with the runner's weight, so it is appropriate to store it as an integer. The charity event field stores either True or False depending on whether the runner wishes to be chosen to run in the charity event; the most appropriate data type is therefore Boolean.

2.1.2 Activities Table

Every activity that the runners add will be given its own record in this table. It is accessed sequentially, according to the id of each activity. In addition, each activity will be linked to a user through a foreign key, called user_id. It is a one-to-many relationship.

Field Name	**PK / FK**	**Typical Data**	**Data Type**
id	Primary	01	Integer
sport	n/a	running	String
effigy	n/a	5 mph	String
date	n/a	2015-01-04	Date
start	n/a	8:00AM	String
finish	n/a	10:00AM	String
hours	n/a	2	Integer
opinion	n/a	Brilliant	String
thoughts	n/a	It was great.	String
user_id	Foreign	02	Integer

Table 2: Activities Table

The id of each activity serves as its primary key; it is automatically incremented whenever a new activity is added, hence the data type of integer. The sport field will be a string; it will store the type of sport that the activity belongs to, and so string is the most appropriate data type. The effigy field will store the specific detail for each activity, such as the speed for running sessions, or the type of stroke for swimming sessions. Due to the wide range of options that can be stored in this, and the fact that no calculations will be performed, the string data type would be the most appropriate.

3 User Interface Design

The system will use a web based, graphical user interface. It will be simple and easy to use, making use of user interface paradigms well known to users, such as buttons, form inputs and drop-down boxes, through their use of other computer systems. In order to increase usability, the system will make use of a consistent colour palette - each sport will be associated with a particular colour:

Green - rgb(82, 170, 94) - associated with running
Yellow - rgb(240, 173, 78) - associated with cycling
Blue - rgb(91, 192, 222) - associated with swimming

In addition, the system will make use of a consistent font: Raleway, and its variants. Raleway is a distinctive yet readable sans-serif font, and is the only font used throughout the system. It can be seen in the User interface documentation.

3.1 Main Layout Template

To ensure visual consistency throughout the system, every page will derive itself from a master template, which will contain aspects like the navigation, footer and placement of elements.

4 Hardware and Software Requirements

5 Processing Stages

6 Evaluation Criteria

Part II
Program Documentation

7 User Interface

This section contains screen captures of the all the different areas of the completed system, along with additional notes stating how they are fit for purpose.

7.1 Main Layout

Every other page derives the constant elements, like the navigation bar and footer, from this template, to ensure visual consistency. Every other page derives the constant elements, like the navigation bar and footer, from this template, to ensure visual consistency.

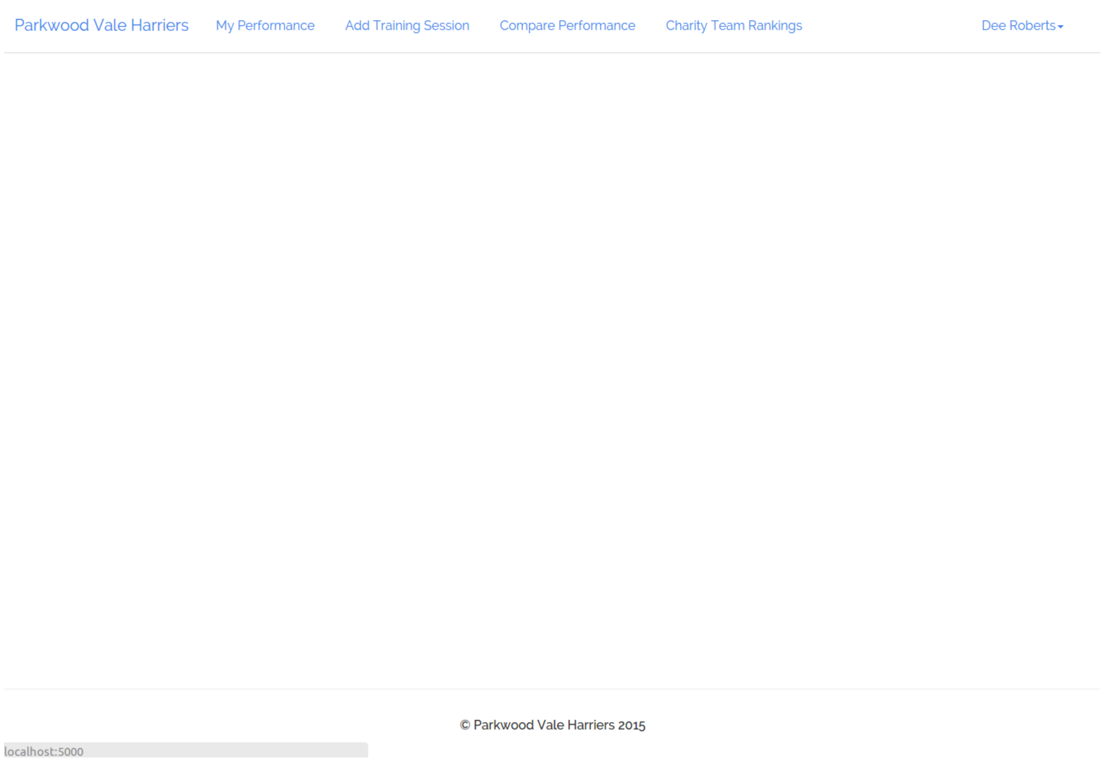
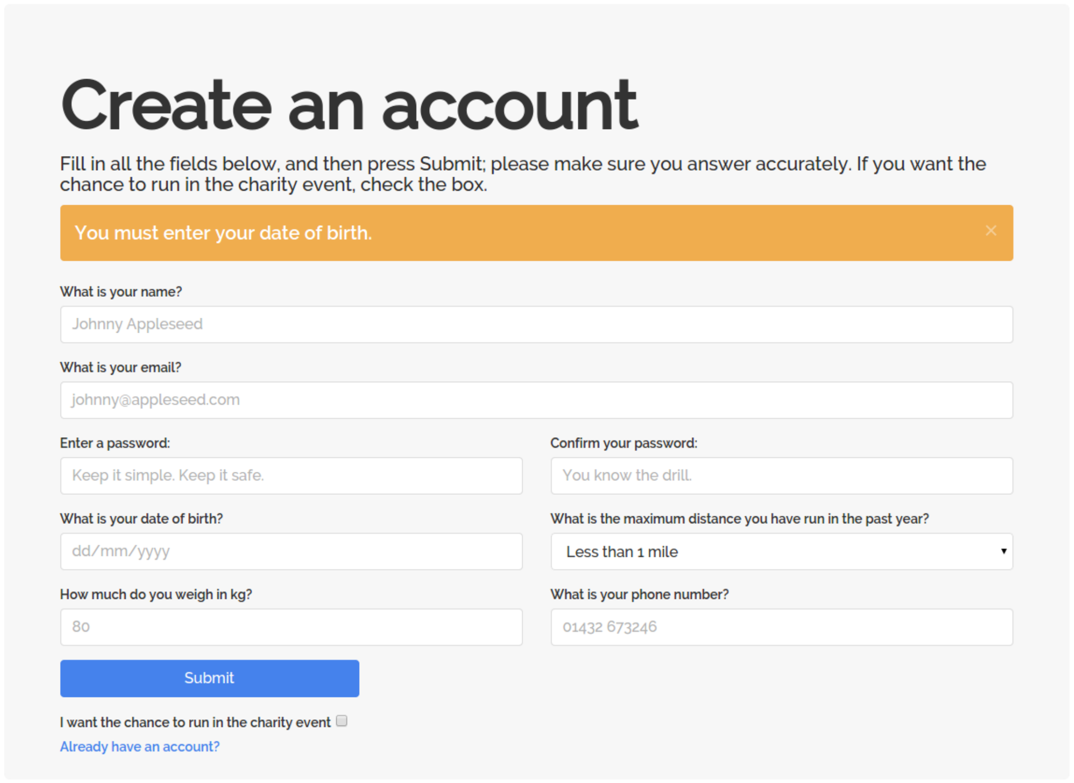


Figure 1: Master Layout

7.2 Register Page

The register page features a clear, simple design, with input boxes laid out in a consistent style. Each input box features placeholder text, to provide a visual guide to the user as to what sort of data they should be typing in. To simplify entry, the date of birth input brings up a datepicker widget upon click, making it simple for users to enter their date of birth. Additionally, validation errors are featured at the top of the page in a big yellow box, making them easy to see; they also have a close button to prevent them getting in the way. A link to the login page allows users who already have an account to quickly login.



Create an account

Fill in all the fields below, and then press Submit; please make sure you answer accurately. If you want the chance to run in the charity event, check the box.

You must enter your date of birth. ×

What is your name?
Johnny Appleseed

What is your email?
johnny@appleseed.com

Enter a password:
Keep it simple. Keep it safe.

Confirm your password:
You know the drill.

What is your date of birth?
dd/mm/yyyy

What is the maximum distance you have run in the past year?
Less than 1 mile

How much do you weigh in kg?
80

What is your phone number?
01432 673246

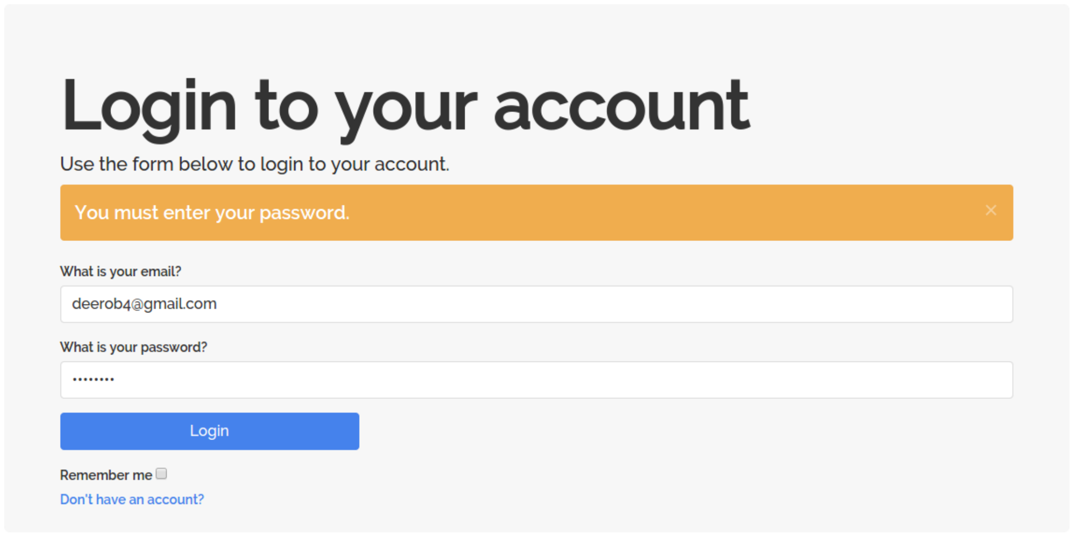
I want the chance to run in the charity event ☐

Already have an account?

Figure 2: Registration Page

7.3 Login Page

As the login page has only one function - to get the user logged in to the system - it features a very simple layout, with only two input forms and a button. Like the registration form, though not visible in this capture, placeholders are overlaid on the inputs to provide a visual guide as to should be typed in. Additionally, the password field blanks out the input, a helpful security measure that prevents onlookers viewing the user's password. For consistency, the same validation error system as with the registration page is used.


Login to your account

Use the form below to login to your account.

You must enter your password.

What is your email?

deerob4@gmail.com

What is your password?

••••••••

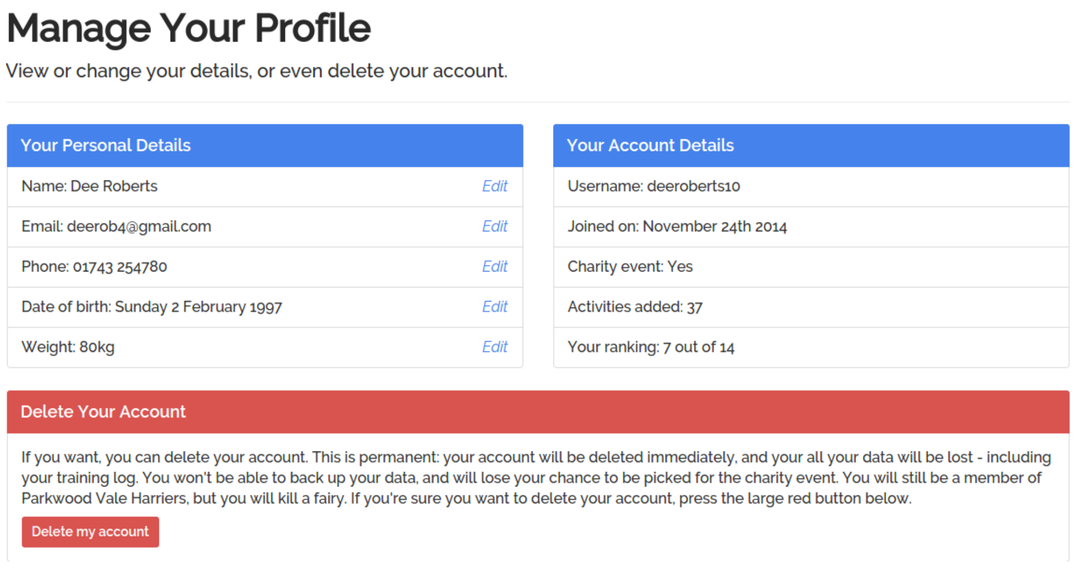
Figure 3: Login Page

7.4 Profile Page

The profile page allows the user to view and update their personal information. Certain sections appear on demand, so multiple captures have been taken.

7.4.1 Main View

Due to the large amount of data that is being presented on this page, a structured approach has been taken, with a three-panel view being implemented. This helps to seperate the different areas of the page in a logical manner, making it easier for the user to find what they are looking for. Each item of data is given its own row, making it clear which is which. The delete account section has been coloured in red, a colour traditionally associated with danger. This helps to convey to the user that bad things will happen if they delete their account. Furthermore, playful text has been used in the delete account panel, to bring a sense of amusement and, hopefully, dissuade the user from following through with their actions.


Manage Your Profile

View or change your details, or even delete your account.

Your Personal Details	
Name: Dee Roberts	Edit
Email: deerob4@gmail.com	Edit
Phone: 01743 254780	Edit
Date of birth: Sunday 2 February 1997	Edit
Weight: 80kg	Edit

Your Account Details
Username: deeroberts10
Joined on: November 24th 2014
Charity event: Yes
Activities added: 37
Your ranking: 7 out of 14

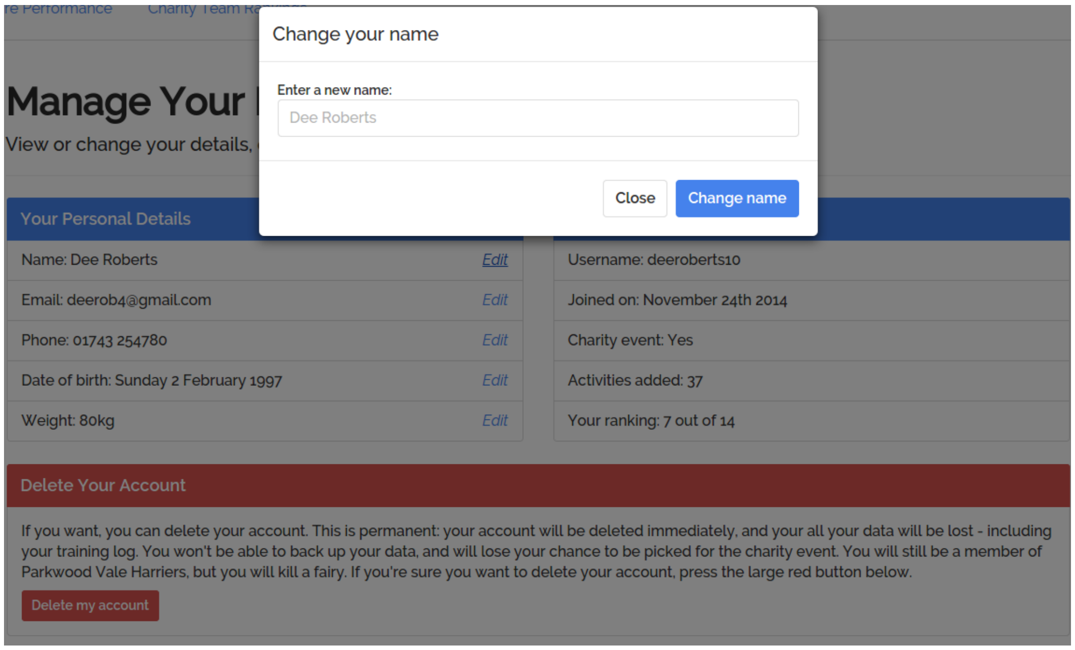
Delete Your Account

If you want, you can delete your account. This is permanent: your account will be deleted immediately, and your all your data will be lost - including your training log. You won't be able to back up your data, and will lose your chance to be picked for the charity event. You will still be a member of Parkwood Vale Harriers, but you will kill a fairy. If you're sure you want to delete your account, press the large red button below.

Figure 4: Profile Page - Main View

7.4.2 Change Details

The interface for changing details is very simple - it features just an input for changing the element, and a button to confirm. The placeholder text for the input is set to the current item, for visual consistency. Making this panel pop up as opposed to being on a seperate page improves the flow of the page, preventing the user becoming disorientated.



Name: Dee Roberts Edit
Email: deerob4@gmail.com Edit
Phone: 01743 254780 Edit
Date of birth: Sunday 2 February 1997 Edit
Weight: 80kg Edit

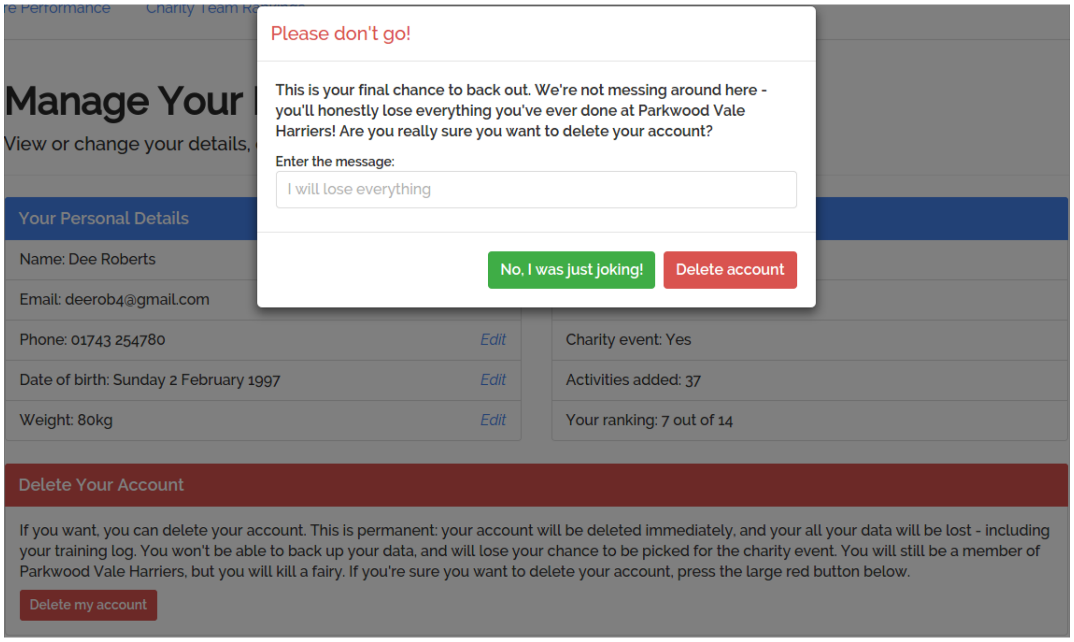
Username: deeroberts10
Joined on: November 24th 2014
Charity event: Yes
Activities added: 37
Your ranking: 7 out of 14

If you want, you can delete your account. This is permanent: your account will be deleted immediately, and your all your data will be lost - including your training log. You won't be able to back up your data, and will lose your chance to be picked for the charity event. You will still be a member of Parkwood Vale Harriers, but you will kill a fairy. If you're sure you want to delete your account, press the large red button below.

Figure 5: Profile Page - Change Details

7.4.3 Delete Account

To ensure that the user is fully aware of the severity of deleting their account, they must type in ”I will lose everything” into the box; this also makes it harder for them to accidentally delete their account. Positive reinforcement is used in this section through the use of colours - greeen is associated with positivity, and users have been shown to click on green coloured buttons more often than red; this further dissuades them from deleting their account.



Please don't go!

This is your final chance to back out. We're not messing around here - you'll honestly lose everything you've ever done at Parkwood Vale Harriers! Are you really sure you want to delete your account?

Enter the message: I will lose everything

Name: Dee Roberts

Email: deerob4@gmail.com

Phone: 01743 254780

Date of birth: Sunday 2 February 1997

Weight: 80kg

Charity event: Yes

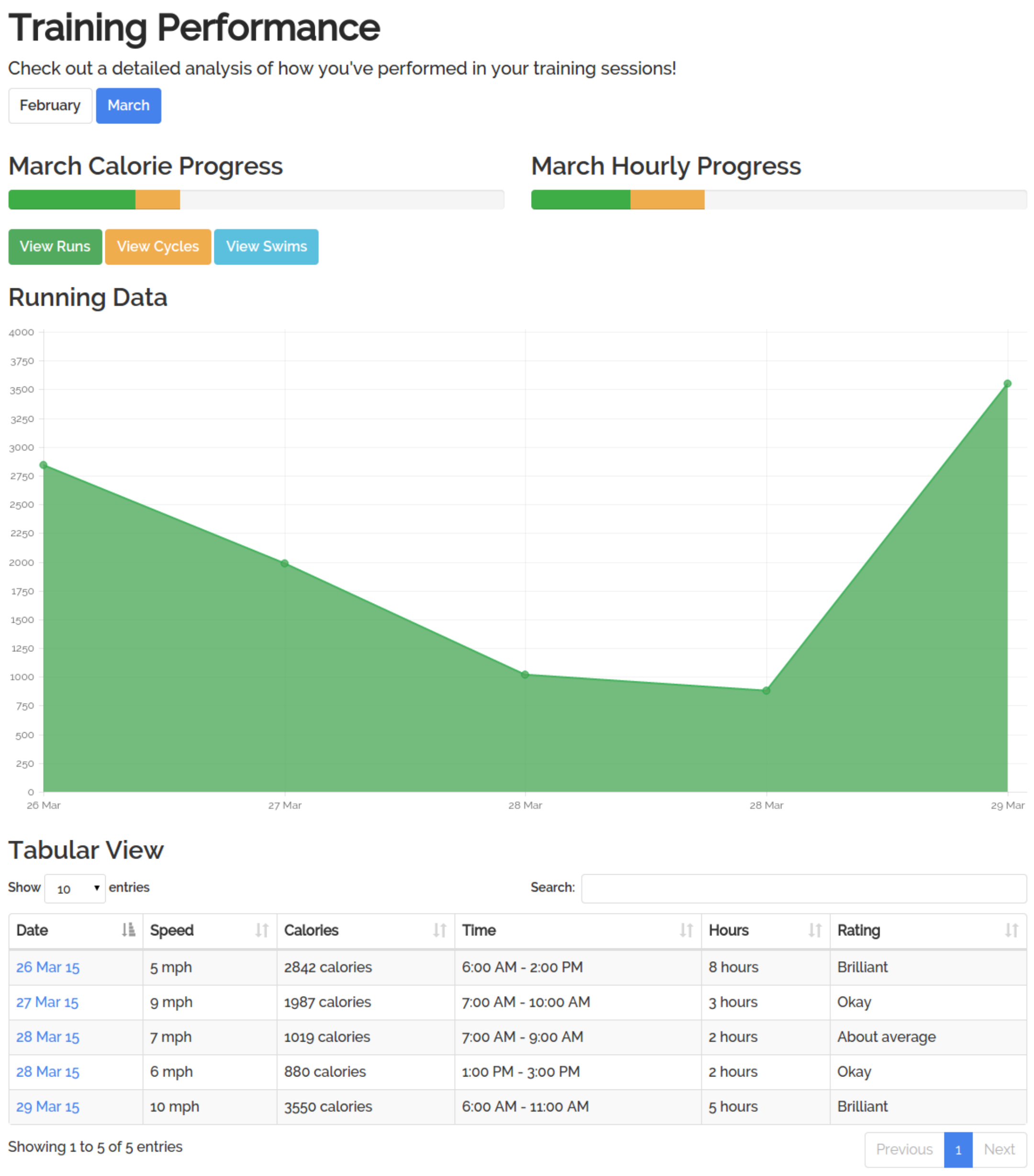
Activities added: 37

Your ranking: 7 out of 14

If you want, you can delete your account. This is permanent: your account will be deleted immediately, and your all your data will be lost - including your training log. You won't be able to back up your data, and will lose your chance to be picked for the charity event. You will still be a member of Parkwood Vale Harriers, but you will kill a fairy. If you're sure you want to delete your account, press the large red button below.

Figure 6: Profile Page - Delete Account

7.5 User Performance Page



Training Performance

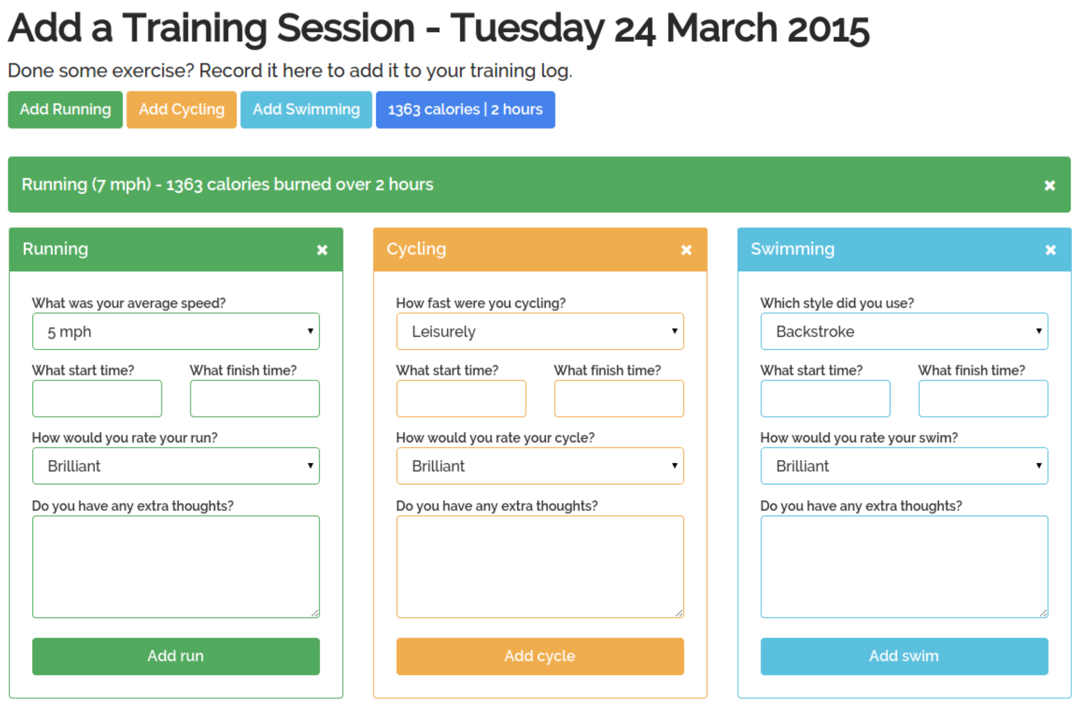
Check out a detailed analysis of how you've performed in your training sessions!

Tabular View

Date	Speed	Calories	Time	Hours	Rating
26 Mar 15	5 mph	2842 calories	6:00 AM - 2:00 PM	8 hours	Brilliant
27 Mar 15	9 mph	1987 calories	7:00 AM - 10:00 AM	3 hours	Okay
28 Mar 15	7 mph	1019 calories	7:00 AM - 9:00 AM	2 hours	About average
28 Mar 15	6 mph	880 calories	1:00 PM - 3:00 PM	2 hours	Okay
29 Mar 15	10 mph	3550 calories	6:00 AM - 11:00 AM	5 hours	Brilliant

Showing 1 to 5 of 5 entries

7.6 Add Activity Page



Add a Training Session - Tuesday 24 March 2015

Done some exercise? Record it here to add it to your training log.

Add Running | Add Cycling | Add Swimming | 1363 calories | 2 hours

Running (7 mph) - 1363 calories burned over 2 hours ✖

Running ✖

What was your average speed?
5 mph

What start time? What finish time?

How would you rate your run?
Brilliant

Do you have any extra thoughts?

Add run

Cycling ✖

How fast were you cycling?
Leisurely

What start time? What finish time?

How would you rate your cycle?
Brilliant

Do you have any extra thoughts?

Add cycle

Swimming ✖

Which style did you use?
Backstroke

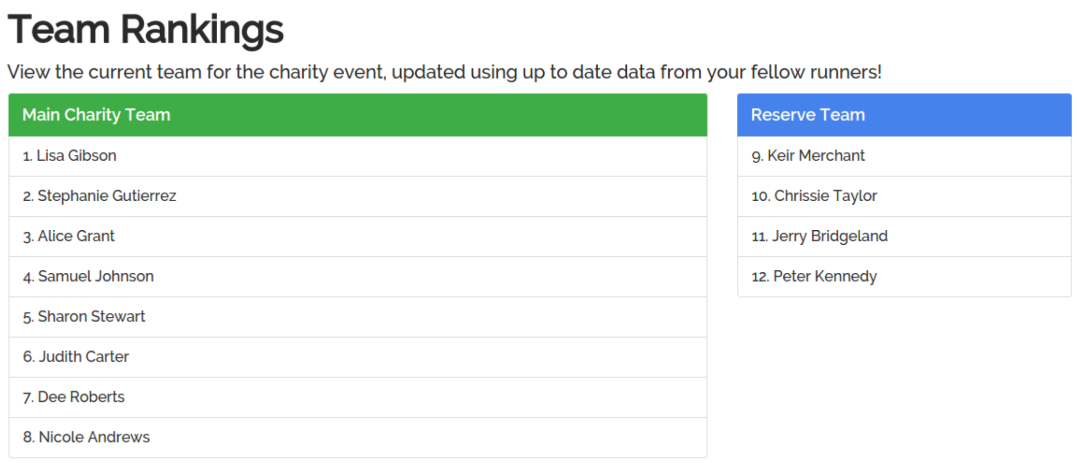
What start time? What finish time?

How would you rate your swim?
Brilliant

Do you have any extra thoughts?

Add swim

7.7 Rankings Page



Team Rankings

View the current team for the charity event, updated using up to date data from your fellow runners!

Main Charity Team
1. Lisa Gibson
2. Stephanie Gutierrez
3. Alice Grant
4. Samuel Johnson
5. Sharon Stewart
6. Judith Carter
7. Dee Roberts
8. Nicole Andrews

Reserve Team
9. Keir Merchant
10. Chrissie Taylor
11. Jerry Bridgeland
12. Peter Kennedy

8 Database Models

This section contains documentation on the finished database tables and models, including an ER diagram showing the relationship between the tables, schematics, and a visual view.

8.1 Table Relationships

The activities and users are linked through a foreign key This means that there is a one to many relationship between users and activities - one user can have many activities, but each activity can only have one user.

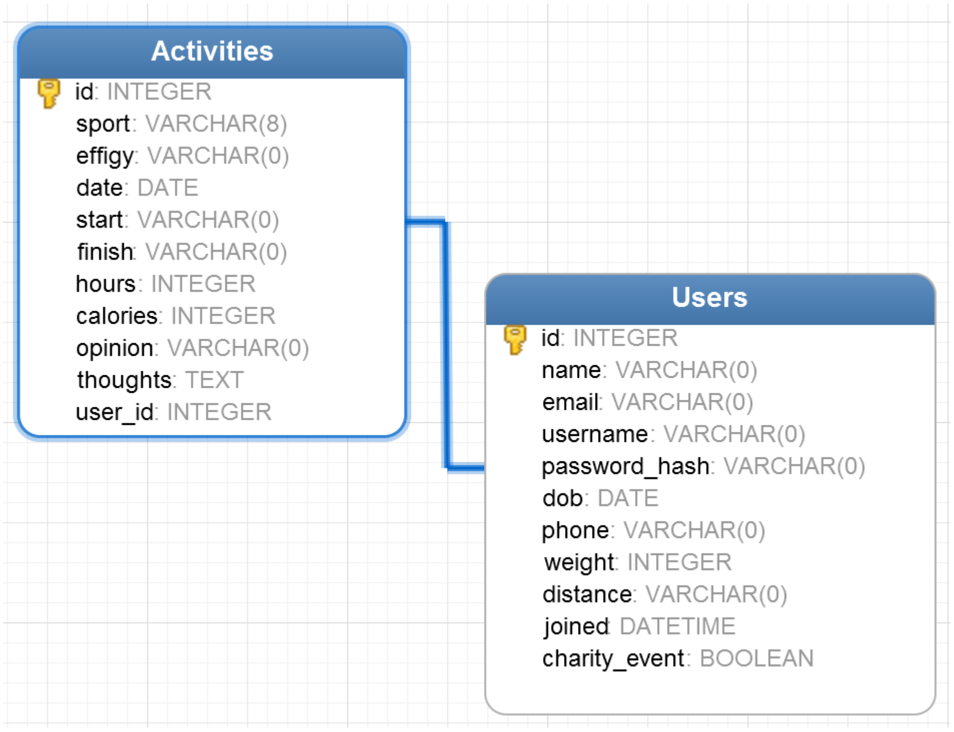


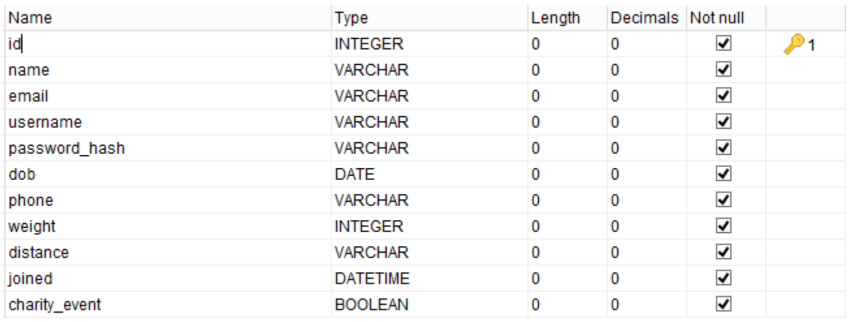
Figure 7: Table Relationships

8.2 Table Schemas

Each table in the database has its own schema, in which is described the name, data type and key type of each column. They can be found below.

8.2.1 Users Table

The id column is the primary key. Email is used to login. Username is used in certain routes; see processes. Weight is used to calculate calories.

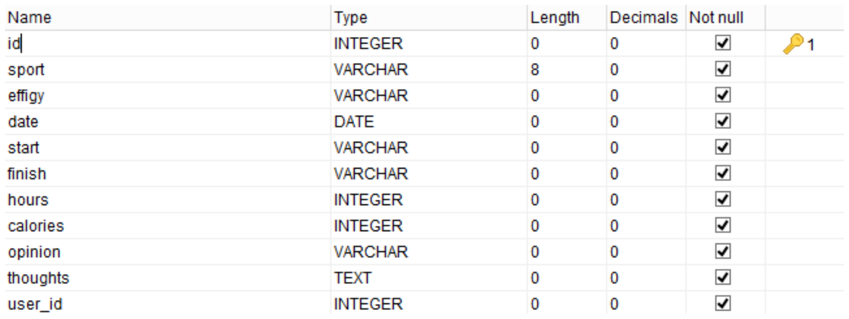


Name	Type	Length	Decimals	Not null	
id	INTEGER	0	0	✓	🔑 1
name	VARCHAR	0	0	✓	
email	VARCHAR	0	0	✓	
username	VARCHAR	0	0	✓	
password_hash	VARCHAR	0	0	✓	
dob	DATE	0	0	✓	
phone	VARCHAR	0	0	✓	
weight	INTEGER	0	0	✓	
distance	VARCHAR	0	0	✓	
joined	DATETIME	0	0	✓	
charity_event	BOOLEAN	0	0	✓	

Figure 8: Users Table Schema

8.2.2 Activities Table

The id column is the primary key. Effigy is the specific details of each activity, such as the swimming stroke or running speed. user_id is the foreign key linking the activity to a user.



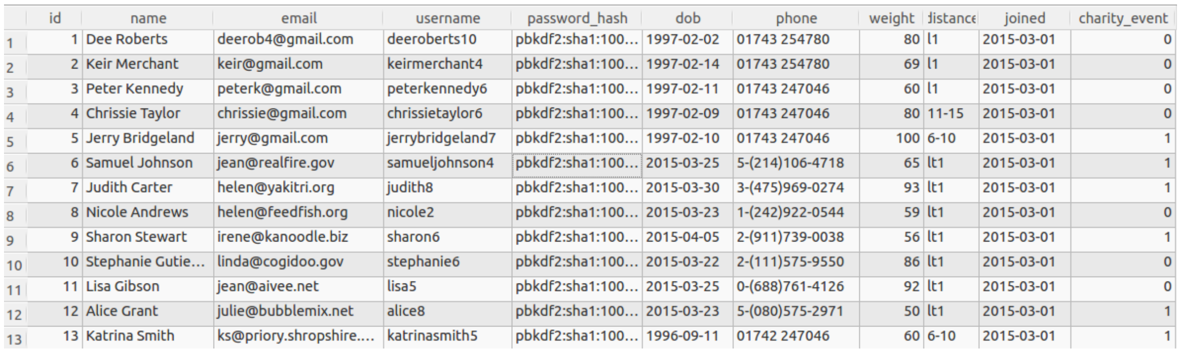
Name	Type	Length	Decimals	Not null	
id	INTEGER	0	0	✓	🔑 1
sport	VARCHAR	8	0	✓	
effigy	VARCHAR	0	0	✓	
date	DATE	0	0	✓	
start	VARCHAR	0	0	✓	
finish	VARCHAR	0	0	✓	
hours	INTEGER	0	0	✓	
calories	INTEGER	0	0	✓	
opinion	VARCHAR	0	0	✓	
thoughts	TEXT	0	0	✓	
user_id	INTEGER	0	0	✓	

Figure 9: Activities Table Schema

8.3 Data Views

The following is a snapshot of the data in the two tables at the time of writing, to illustrate how they will appear in production.

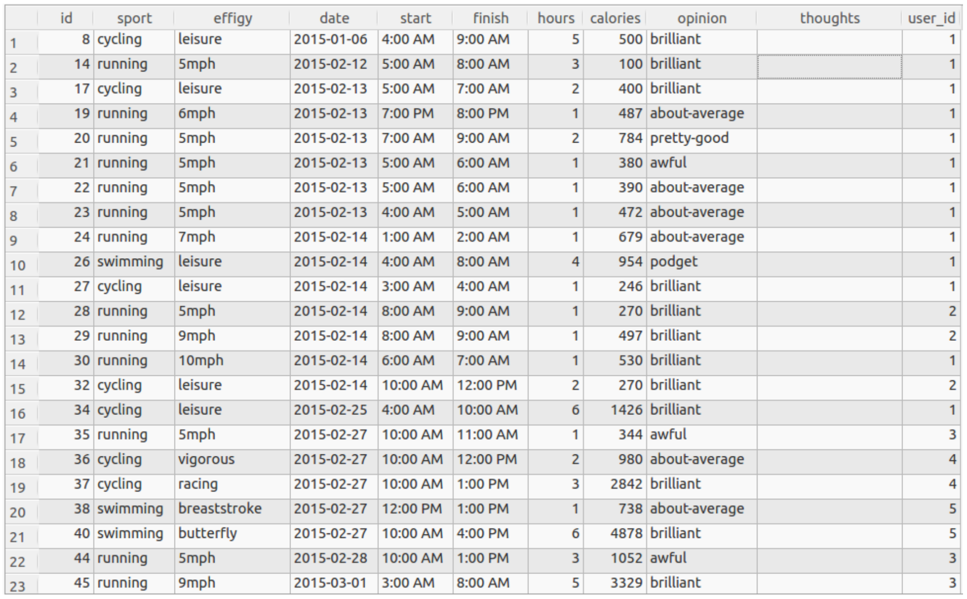
8.3.1 Users Table



	id	name	email	username	password_hash	dob	phone	weight	distance	joined	charity_event
1	1	Dee Roberts	deerob4@gmail.com	deeroberts10	pbkdf2:sha1:100…	1997-02-02	01743 254780	80	l1	2015-03-01	0
2	2	Keir Merchant	keir@gmail.com	keirmerchant4	pbkdf2:sha1:100…	1997-02-14	01743 254780	69	l1	2015-03-01	0
3	3	Peter Kennedy	peterk@gmail.com	peterkennedy6	pbkdf2:sha1:100…	1997-02-11	01743 247046	60	l1	2015-03-01	0
4	4	Chrissie Taylor	chrissie@gmail.com	chrissietaylor6	pbkdf2:sha1:100…	1997-02-09	01743 247046	80	11-15	2015-03-01	0
5	5	Jerry Bridgeland	jerry@gmail.com	jerrybridgeland7	pbkdf2:sha1:100…	1997-02-10	01743 247046	100	6-10	2015-03-01	1
6	6	Samuel Johnson	jean@realfire.gov	samueljohnson4	pbkdf2:sha1:100…	2015-03-25	5-(214)106-4718	65	lt1	2015-03-01	1
7	7	Judith Carter	helen@yakitri.org	judith8	pbkdf2:sha1:100…	2015-03-30	3-(475)969-0274	93	lt1	2015-03-01	1
8	8	Nicole Andrews	helen@feedfish.org	nicole2	pbkdf2:sha1:100…	2015-03-23	1-(242)922-0544	59	lt1	2015-03-01	0
9	9	Sharon Stewart	irene@kanoodle.biz	sharon6	pbkdf2:sha1:100…	2015-04-05	2-(911)739-0038	56	lt1	2015-03-01	1
10	10	Stephanie Gutie…	linda@cogidoo.gov	stephanie6	pbkdf2:sha1:100…	2015-03-22	2-(111)575-9550	86	lt1	2015-03-01	0
11	11	Lisa Gibson	jean@aivee.net	lisa5	pbkdf2:sha1:100…	2015-03-25	0-(688)761-4126	92	lt1	2015-03-01	0
12	12	Alice Grant	julie@bubblemix.net	alice8	pbkdf2:sha1:100…	2015-03-23	5-(080)575-2971	50	lt1	2015-03-01	1
13	13	Katrina Smith	ks@priory.shropshire.…	katrinasmith5	pbkdf2:sha1:100…	1996-09-11	01742 247046	60	6-10	2015-03-01	1

Figure 10: Users Table Data View

8.3.2 Activities Table



	id	sport	effigy	date	start	finish	hours	calories	opinion	thoughts	user_id
1	8	cycling	leisure	2015-01-06	4:00 AM	9:00 AM	5	500	brilliant		1
2	14	running	5mph	2015-02-12	5:00 AM	8:00 AM	3	100	brilliant		1
3	17	cycling	leisure	2015-02-13	5:00 AM	7:00 AM	2	400	brilliant		1
4	19	running	6mph	2015-02-13	7:00 PM	8:00 PM	1	487	about-average		1
5	20	running	5mph	2015-02-13	7:00 AM	9:00 AM	2	784	pretty-good		1
6	21	running	5mph	2015-02-13	5:00 AM	6:00 AM	1	380	awful		1
7	22	running	5mph	2015-02-13	5:00 AM	6:00 AM	1	390	about-average		1
8	23	running	5mph	2015-02-13	4:00 AM	5:00 AM	1	472	about-average		1
9	24	running	7mph	2015-02-14	1:00 AM	2:00 AM	1	679	about-average		1
10	26	swimming	leisure	2015-02-14	4:00 AM	8:00 AM	4	954	podget		1
11	27	cycling	leisure	2015-02-14	3:00 AM	4:00 AM	1	246	brilliant		1
12	28	running	5mph	2015-02-14	8:00 AM	9:00 AM	1	270	brilliant		2
13	29	running	9mph	2015-02-14	8:00 AM	9:00 AM	1	497	brilliant		2
14	30	running	10mph	2015-02-14	6:00 AM	7:00 AM	1	530	brilliant		1
15	32	cycling	leisure	2015-02-14	10:00 AM	12:00 PM	2	270	brilliant		2
16	34	cycling	leisure	2015-02-25	4:00 AM	10:00 AM	6	1426	brilliant		1
17	35	running	5mph	2015-02-27	10:00 AM	11:00 AM	1	344	awful		3
18	36	cycling	vigorous	2015-02-27	10:00 AM	12:00 PM	2	980	about-average		4
19	37	cycling	racing	2015-02-27	10:00 AM	1:00 PM	3	2842	brilliant		4
20	38	swimming	breaststroke	2015-02-27	12:00 PM	1:00 PM	1	738	about-average		5
21	40	swimming	butterfly	2015-02-27	10:00 AM	4:00 PM	6	4878	brilliant		5
22	44	running	5mph	2015-02-28	10:00 AM	1:00 PM	3	1052	awful		3
23	45	running	9mph	2015-03-01	3:00 AM	8:00 AM	5	3329	brilliant		3

Figure 11: Activities Table Data View

9 Variables

A very large number of variables have been used throughout the system in order to store data temporarily. It should be noted that variables in Python do not work in the same way as in other languages like Visual Basic: a variable acts as a pointer to something already in memory (created by Python automatically), as opposed to creating a space in memory to store the contents of the variable. Therefore, the code x = 10 merely sets the variable x to an address that points at 10, as opposed to writing 10 into memory.

9.1 Global Variables

Throughout the system, a small number of global variables are used in order to provide functionality. The following table lists their names, type and purpose.

Name	**Type**	**Purpose**
login_manager	Object	Creates the actual Flask application object.
db	Object	Returns a connection to the database.
User	Object	Creates a connection to the Users db table.
Activity	Object	Creates a connection to the Activities db table.
current_user	Object	Returns details about the logged in user.

Table 3: Global Variables

9.2 Local Variables

The following table contains a list of all the variables used locally throughout the functions / classes.

10 Annotated Listings

This section contains all of the code for the system, split into several logical categories. The system is made up of a very large number of Python functions, as well as some additional aspects, such as Jinja2 HTML templates to display the interface, and CSS to provide styling.

10.1 HTML Views

Every page of the system has its own corresponding HTML template. These are used to display the data passed by the Python back-end, and provide interface elements such as buttons and dropdown boxes. A comparison can be drawn between them and the XML built by the Design Mode in Visual Basic, but, as thesealso contain some logic of their own, such as for-loops to loop through arrays, it is appropriate to include them in the documentation.

10.1.1 layout.html

```html
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootswatch/3.3.1/readable/bootstrap.min.css"/>
    <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/animate.css/3.2.0/animate.min.css"/>
    <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.3.1/css/datepicker.min.css"/>
    <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/pickadate.js/3.5.3/compressed/themes/classic.css"/>
    <link rel="stylesheet"
          href="//cdnjs.cloudflare.com/ajax/libs/pickadate.js/3.5.3/compressed/themes/classic.time.css"/>
    <link rel="stylesheet" href="//cdn.datatables.net/plug-ins/f2c75b7247b/integration/bootstrap/3/dataTables.bootstrap.css"/>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/main.css') }}"/>
    <title>{% block title %} - Parkwood Vale Harriers{% endblock %}</title>
</head>
<body>
<nav class="navbar navbar-default" role="navigation">
    <div class="container-fluid">
        <div class="navbar-header">
            <button class="navbar-toggle" data-toggle="collapse" data-target="#main_nav" type="button">
                <span class="sr-only">Toggle Navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="{{ url_for('main.home') }}">Parkwood Vale Harriers</a>
        </div>
        <div class="collapse navbar-collapse" id="main_nav">
            {% if current_user.is_authenticated() %}
                <ul class="nav navbar-nav">
                    <li><a href="{{ url_for('main.performance', month='march') }}">My Performance</a></li>
                    <li><a href="{{ url_for('main.add_training') }}">Add Training Session</a></li>
                    <li><a href="{{ url_for('main.compare_performance') }}">Compare Performance</a></li>
                    <li><a href="{{ url_for('main.rankings') }}">Charity Team Rankings</a></li>
                </ul>
                <ul class="nav navbar-nav navbar-right">
                    <li class="dropdown">
```

```
                        <a class="dropdown-toggle" data-toggle="dropdown" href="#">{{ current_user.name }}<span
                                class="caret"></span></a>
                        <ul class="dropdown-menu" role="menu">
                            <li><a href="{{ url_for('main.profiles', username=current_user.username) }}">Your
                                Profile</a></li>
                            <li><a href="#">Change Password</a></li>
                            <li class="divider"></li>
                            <li><a href="{{ url_for('auth.logout') }}">Logout</a></li>
                        </ul>
                    </li>
                </ul>
            {% else %}
                <ul class="nav navbar-nav">
                    <li><a href="{{ url_for('auth.login') }}">Login</a></li>
                    <li><a href="{{ url_for('auth.register') }}">Register</a></li>
                </ul>
            {% endif %}
        </div>
    </div>
</nav>
<div class="container">
    {% block content %}{% endblock %}
</div>
<footer class="footer">
    &copy; Parkwood Vale Harriers 2015
</footer>
</body>
<script src="//ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.3.1/js/bootstrap-datepicker.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/pickadate.js/3.5.3/compressed/picker.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/pickadate.js/3.5.3/compressed/picker.time.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/Chart.js/1.0.1/Chart.min.js"></script>
<script src="//cdn.datatables.net/1.10.5/js/jquery.dataTables.min.js"></script>
<script src="//cdn.datatables.net/plug-ins/f2c75b7247b/integration/bootstrap/3/dataTables.bootstrap.js"></script>
{% block scripts %}{% endblock %}
<script src="{{ url_for('static', filename='js/main.js') }}"></script>
</html>
```

Listing 1: Main Layout

10.1.2 register.html

```
{% extends 'layout.html' %}

{% block title %}Register{{ super() }}{% endblock %}

{% block content %}
    <div class="jumbotron">
        <h1>Create an account</h1>

        <h4>Fill in all the fields below, and then press Submit; please make sure you answer accurately. If you want the
            chance to run in the charity event, check the box.</h4>

        <form method="POST" class="register-form">
            {{ form.csrf_token }}
            {% with messages=get_flashed_messages(with_categories=True) %}
                {% if messages %}
                    <div class="row">
                        <div class="col-md-12">
                            {% for category, message in messages %}
                                <div class="alert alert-{{ category }} alert-dismissable">
                                    <button type="button" class="close" data-dismiss="alert">
                                        <span aria-hidden="true">&times;</span>
                                        <span class="sr-only">Close</span>
                                    </button>
                                    <p>{{ message }}</p>
                                </div>
                            {% endfor %}
                        </div>
                    </div>
                {% endif %}
            {% endwith %}
            <div class="form-group">
                {{ form.name.label }}
                {{ form.name(class='form-control input_membership_name', placeholder='Johnny Appleseed') }}
            </div>
            <div class="form-group">
                {{ form.email.label }}
                {{ form.email(class='form-control input_membership_email', type='email', placeholder='johnny@appleseed.com') }}
            </div>
            <div class="row">
                <div class="col-md-6">
                    <div class="form-group">
                        {{ form.password.label }}
                        {{ form.password(class='form-control input_membership_password', placeholder
                            ='Keep it simple. Keep it safe.') }}
                    </div>
                </div>
                <div class="col-md-6">
                    <div class="form-group">
                        {{ form.confirm.label }}
                        {{ form.confirm(class='form-control input_membership_confirm', placeholder='You know the drill.') }}
                    </div>
                </div>
            </div>
            <div class="row">
                <div class="col-md-6">
                    <div class="form-group">
                        {{ form.dob.label }}
                        {{ form.dob(class='form-control input_membership_dob datepicker', placeholder='dd/mm/yyyy') }}
                    </div>
                </div>
                <div class="col-md-6">
                    <div class="form-group">
                        {{ form.distance.label }}
                        {{ form.distance(class='form-control') }}
                    </div>
                </div>
            </div>
            <div class="row">
                <div class="col-md-6">
                    <div class="form-group">
                        {{ form.weight.label }}
                        {{ form.weight(class='form-control', type='number', placeholder='80', min=0, max=100) }}
                    </div>
                </div>
                <div class="col-md-6">
                    <div class="form-group">
                        {{ form.phone.label }}
                        {{ form.phone(class='form-control', type='tel', placeholder='01432 673246') }}
                    </div>
                </div>
            </div>
            <div class="row">
                <div class="col-sm-12 col-md-4">
                    <div class="form-group">
                        {{ form.submit(class='btn btn-primary') }}
                    </div>
                </div>
            </div>
            <div class="row charity-row">
                <div class="col-md-7">
                    <div class="form-group">
                        {{ form.charity_event.label(class='charity-label') }}
                        {{ form.charity_event(class='
                            input_membership_charity') }}
                        <br/>
                        <label><a href="{{ url_for('auth.login') }}
                            ">Already have an account?</a></label>
                    </div>
                </div>
            </div>
        </form>
    </div>
{% endblock %}
```

Listing 2: Register Page

10.1.3 login.html

```
{% extends 'layout.html' %}

{% block title %}Login{{ super() }}{% endblock %}

{% block content %}
    <div class="jumbotron">
        <h1>Login to your account</h1>

        <h4>Use the form below to login to your account.</h4>

        <form method="POST" class="register-form">
            {{ form.csrf_token }}
            {% with messages=get_flashed_messages(with_categories=
                True) %}
                {% if messages %}
                    <div class="row">
                        <div class="col-md-12">
                            {% for category, message in messages %}
                                <div class="alert alert-{{ category
                                     }} alert-dismissable">
                                    <button type="button" class="
                                        close" data-dismiss="alert"
                                        >
                                        <span aria-hidden="true">&
                                            times;</span>
                                        <span class="sr-only">Close
                                            </span>
                                    </button>
                                    <p>{{ message }}</p>
                                </div>
                            {% endfor %}
                        </div>
                    </div>
                {% endif %}
            {% endwith %}
            <div class="form-group">
                {{ form.email.label }}
                {{ form.email(class='form-control', type='email',
                    placeholder='johnny@appleseed.com') }}
            </div>
            <div class="form-group">
                {{ form.password.label }}
                {{ form.password(class='form-control', placeholder
                    ='Something secret!') }}
            </div>
            <div class="row">
                <div class="col-sm-12 col-md-4">
                    <div class="form-group">
                        {{ form.login(class='btn btn-primary') }}
                    </div>
                </div>
            </div>
            <div class="row charity-row">
                <div class="col-md-7">
                    <div class="form-group">
                        {{ form.remember.label(class='remember-
                            label') }}
                        {{ form.remember(class='
                            input_membership_charity') }}
                        <br/>
                        <label><a href="{{ url_for('auth.register')
                             }}">Don't have an account?</a></label>
                    </div>
                </div>
            </div>
        </form>
    </div>
{% endblock %}
```

Listing 3: Login Page

10.1.4 user_performance.html

```
{% extends 'layout.html' %}

{% block title %}Training Performance{{ super() }}{% endblock %}

{% block content %}
    <h1 class="trainingHeading">Training Performance</h1>
    <h4>Check out a detailed analysis of how you've performed in
        your training sessions!</h4>

    <ul class="month_buttons">
        {% for month in months %}
            <a href="{{ url_for('main.performance', month=month) }}
                "><li class="btn btn-{% if current_month.lower() ==
                 month %}primary{% else %}default{% endif %}">{{
                month|title }}</li></a>
        {% endfor %}
    </ul>

    {% with messages=get_flashed_messages(with_categories=True) %}
        {% if messages %}
            <div class="row">
                <div class="col-md-12">
                    {% for category, message in messages %}
                        <div class="alert alert-{{ category }}
                            alert-dismissable">
                            <button type="button" class="close"
                                data-dismiss="alert">
                                <span aria-hidden="true">&times;</
                                    span>
                                <span class="sr-only">Close</span>
                            </button>
                            <p>{{ message }}</p>
                        </div>
                    {% endfor %}
                </div>
            </div>
        {% endif %}
    {% endwith %}

    <div class="row">
        <div class="col-md-6">
            <h3 class="performance-subtitle calorie-subtitle">{{
                current_month|title }} Calorie Progress</h3>

            <div class="progress">
                <div class="progress-bar progress-bar-success
                    running-calories-bar"
                     style="width: {{ user_data.progress_data.
                         running.calories.percentage }}%;"
                     role="progressbar" data-toggle="tooltip"
                     title="{{ user_data.progress_data.running.
                         calories.value }} calories"></div>

                <div class="progress-bar progress-bar-warning
                    cycling-calories-bar"
                     style="width: {{ user_data.progress_data.
                         cycling.calories.percentage }}%;"
                     role="progressbar" data-toggle="tooltip"
                     title="{{ user_data.progress_data.cycling.
                         calories.value }} calories"></div>

                <div class="progress-bar progress-bar-info swimming
                    -calories-bar"
                     style="width: {{ user_data.progress_data.
                         swimming.calories.percentage }}%;"
                     role="progressbar" data-toggle="tooltip"
                     title="{{ user_data.progress_data.swimming.
                         calories.value }} calories"></div>

            </div>
        </div>

        <div class="col-md-6">
            <h3 class="performance-subtitle hour-subtitle"><span
                class="month-text">{{ current_month|title }}</span>
                 Hourly
                Progress
            </h3>

            <div class="progress">
                <div class="progress-bar progress-bar-success
                    running-hours-bar"
                     style="width: {{ user_data.progress_data.
                         running.hours.percentage }}%;"
                     role="progressbar" data-toggle="tooltip"
                     title="{{ user_data.progress_data.running.
                         hours.value }} hours"></div>

                <div class="progress-bar progress-bar-warning
                    cycling-hours-bar"
                     style="width: {{ user_data.progress_data.
                         cycling.hours.percentage }}%;"
                     role="progressbar" data-toggle="tooltip"
                     title="{{ user_data.progress_data.cycling.
                         hours.value }} hours"></div>

                <div class="progress-bar progress-bar-info swimming
                    -hours-bar"
                     style="width: {{ user_data.progress_data.
                         swimming.hours.percentage }}%;"
                     role="progressbar" data-toggle="tooltip"
                     title="{{ user_data.progress_data.swimming.
                         hours.value }} hours"></div>
            </div>
        </div>
    </div>

    <button class="btn btn-running activity-change" id="running">
        View Runs</button>
    <button class="btn btn-warning activity-change" id="cycling">
        View Cycles</button>
    <button class="btn btn-info activity-change" id="swimming">View
         Swims</button>

    <div class="running-data active">
        <h3>Running Data</h3>
        <canvas id="runningChart" width="1140" height="550"></
            canvas>
        <h3>Tabular View</h3>
        <table id="running-activities" class="table table-bordered
            table-striped table-hover" style="border-radius: 4px;">
            <thead>
                <tr>
                    <th>Date</th>
                    <th>Speed</th>
                    <th>Calories</th>
                    <th>Time</th>
                    <th>Hours</th>
                    <th>Rating</th>
                </tr>
            </thead>
            <tbody>
                {% for run in user_data.sport_data.running %}
                    <tr>
                        <td><a href="{{ url_for('main.
                            individual_activity', activity_id=run.
                            id) }}">{{ run.date }}</a></td>
                        <td>{{ run.effigy }}</td>
                        <td>{{ run.calories }} calories</td>
                        <td>{{ run.start }} - {{ run.finish }}</td>
                        <td>{{ run.hours }} hours</td>
                        <td>{{ run.opinion }}</td>
                    </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>

    <div class="cycling-data">
        <h3>Cycling Data</h3>
        <canvas id="cyclingChart" width="1140" height="550"></
            canvas>
        <h3>Tabular View</h3>
        <table id="cycling-activities" class="table table-bordered
            table-striped table-hover"  style="border-radius: 4px;"
            >
            <thead>
                <tr>
                    <th>Date</th>
                    <th>Speed</th>
                    <th>Calories</th>
                    <th>Time</th>
                    <th>Hours</th>
                    <th>Rating</th>
                </tr>
            </thead>
            <tbody>
                {% for cycle in user_data.sport_data.cycling %}
                    <tr>
                        <td><a href="{{ url_for('main.
                            individual_activity', activity_id=cycle
                            .id) }}">{{ cycle.date }}</a></td>
                        <td>{{ cycle.effigy }}</td>
                        <td>{{ cycle.calories }} calories</td>
                        <td>{{ cycle.start }} - {{ cycle.finish }}<
                            /td>
                        <td>{{ cycle.hours }} hours</td>
                        <td>{{ cycle.opinion }}</td>
                    </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>

    <div class="swimming-data">
        <h3>Swimming Data</h3>
        <canvas id="swimmingChart" width="1140" height="550"></
            canvas>
        <h3>Tabular View</h3>
        <table id="swimming-activities" class="table table-bordered
             table-striped table-hover" style="border-radius: 4px;"
            >
            <thead>
                <tr>
                    <th>Date</th>
                    <th>Speed</th>
                    <th>Calories</th>
                    <th>Time</th>
                    <th>Hours</th>
                    <th>Rating</th>
                </tr>
            </thead>
            <tbody>
                {% for swim in user_data.sport_data.swimming %}
                    <tr>
                        <td><a href="{{ url_for('main.individual_activity', activity_id=swim.id) }}">{{ swim.date }}</a></td>
                        <td>{{ swim.effigy }}</td>
                        <td>{{ swim.calories }} calories</td>
                        <td>{{ swim.start }} - {{ swim.finish }}</td>
                        <td>{{ swim.hours }} hours</td>
                        <td>{{ swim.opinion }}</td>
                    </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>

{% endblock %}

{% block scripts %}
    <script src="{{ url_for('static', filename='js/individual_charts.js') }}"></script>
{% endblock %}
```

Listing 4: User Performance Page

10.1.5 own_profile.html

```
{% extends 'layout.html' %}

{% block title %}Your Profile{{ super() }}{% endblock %}

{% block content %}
    <h1>Manage Your Profile</h1>
    <h4>View or change your details, or even delete your account.</h4>
    <hr/>
    {% with messages=get_flashed_messages(with_categories=True) %}
        {% if messages %}
            <div class="row">
                <div class="col-md-12">
                    {% for category, message in messages %}
                        <div class="alert alert-{{ category }} alert-dismissable">
                            <button type="button" class="close" data-dismiss="alert">
                                <span aria-hidden="true">&times;</span>
                                <span class="sr-only">Close</span>
                            </button>
                            <p>{{ message }}</p>
                        </div>
                    {% endfor %}
                </div>
            </div>
        {% endif %}
    {% endwith %}

    <div class="row">
        <div class="col-md-6">
            <div class="panel panel-primary">
                <div class="panel-heading">
                    <div class="panel-title">Your Personal Details<
                        /div>
                </div>
                <ul class="user-details list-group panel-list">
                    <li class="name list-group-item">Name: {{
                        current_user.name }} <span class="right"
                                                                                         data
                                                                                             -
                                                                                             t
                                                                                             =
                                                                                             "
                                                                                             m
                                                                                             "
                                                                                         data
                                                                                             -
                                                                                             t
                                                                                             =
                                                                                             "
                                                                                             #
                                                                                             c
                                                                                             "
                                                                                             >
                                                                                             <
                                                                                             a
                            href="#">Edit</a></span></li>
                    <li class="email list-group-item">Email: {{
                        current_user.email }} <span class="right"
                                                                                            da
                                                                                            da
```

```
                                href="#">Edit</a></span>
                        </li>
                        <li class="phone list-group-item">Phone: {{
                            current_user.phone }} <span class="right"
                                                                                        da
                                                                                        da
                                href="#">Edit</a></span>
                        </li>
                        <li class="dob list-group-item">Date of birth:
                            {{ current_user.dob.strftime('%A %e %B %G')
                             }} <span
                                class="right" data-toggle="modal"
                                data-target="#changeDobModal"><a href="
                                    #">Edit</a></span></li>
                        <li class="weight list-group-item">Weight: {{
                            current_user.weight }}kg <span class="right
                            "


                                href="#">Edit</a></span></li>
```

```
                </ul>
            </div>
        </div>
        <div class="col-md-6">
            <div class="panel panel-primary">
                <div class="panel-heading">
                    <div class="panel-title">Your Account Details</div>
                </div>
                <ul class="account-details list-group panel-list">
                    <li class="username list-group-item">Username: {{ current_user.username }}</li>
                    <li class="joined list-group-item">Joined on: {{ current_user.joined }}</li>
                    <li class="charity-event list-group-item">Charity event: {% if current_user.charity_event %}
                        Yes{% else %}No{% endif %}</li>
                    <li class="activities-added list-group-item">Activities added: {{ activity_number }}</li>
                    <li class="joined list-group-item">Your ranking: 0 out of {{ total_users }}</li>
                </ul>
            </div>
        </div>
    </div>

    <div class="panel panel-danger">
        <div class="panel-heading">
            <div class="panel-title">Delete Your Account</div>
        </div>
        <div class="panel-body">
            If you want, you can delete your account. This is permanent: your account will be deleted
            immediately, and your all your data will be lost - including your training log. You won't be able to back up
            your data, and will lose your chance to be picked for the charity event. You will still be a member of
            Parkwood Vale Harriers, but you will kill a fairy. If you're sure you want to delete your account, press the
            large red button below.
            <br/>

            <div class="btn btn-danger btn-sm delete-account" data-toggle="modal"
                 data-target="#deleteAccountModal">Delete my account
            </div>
        </div>
    </div>

    <div class="modal fade" id="changeNameModal">
        <div class="modal-dialog">
            <div class="modal-content">
```

```html
                <div class="modal-header">
                    <h4 class="modal-title">Change your name</h4>
                </div>
                <form method="POST" id="changeNameForm">
                    <div class="modal-body">
                        <label>
                            Enter a new name:
                            <input type="text" name="name" placeholder="{{ current_user.name }}" class="form-control" />
                        </label>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                        <button type="submit" class="btn btn-primary btn-modal">Change name</button>
                    </div>
                </form>
            </div>
        </div>
    </div>

    <div class="modal fade" id="changeEmailModal">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title">Change your email</h4>
                </div>
                <form method="POST" id="changeEmailForm">
                    <div class="modal-body">
                        <label>
                            Enter a new email:
                            <input type="text" name="email" placeholder="{{ current_user.email }}" class="form-control"/>
                        </label>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                        <button type="submit" class="btn btn-primary btn-modal">Change email</button>
                    </div>
                </form>
            </div>
        </div>
    </div>

    <div class="modal fade" id="changePhoneModal">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
```

```html
                    <h4 class="modal-title">Change your phone number</h4>
                </div>
                <form method="POST" id="changePhoneForm">
                    <div class="modal-body">
                        <label>
                            Enter a new phone number:
                            <input type="text" name="phone" placeholder="{{ current_user.phone }}" class="form-control"/>
                        </label>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                        <button type="submit" class="btn btn-primary">Change phone number</button>
                    </div>
                </form>
            </div>
        </div>
    </div>

    <div class="modal fade" id="changeDobModal">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title">Change your date of birth</h4>
                </div>
                <form method="POST" id="changeDobForm">
                    <div class="modal-body">
                        <label>
                            Enter a new date of birth:
                            <input type="text" name="dob" placeholder="{{ current_user.dob }}" class="form-control datepicker"/>
                        </label>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                        <button type="submit" class="btn btn-primary">Change date of birth</button>
                    </div>
                </form>
            </div>
        </div>
    </div>

    <div class="modal fade" id="changeWeightModal">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
```

```html
                    <h4 class="modal-title">Change your weight:</h4>
                </div>
                <form method="POST" id="changeWeightForm">
                    <div class="modal-body">
                        <label>
                            Enter a new weight:
                            <input type="number" name="weight" min="10" max="100" placeholder="{{ current_user.weight }}"
                                   class="form-control"/>
                        </label>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-default" data-dismiss="modal">Close</button>
                        <button type="submit" class="btn btn-primary">Change weight</button>
                    </div>
                </form>
            </div>
        </div>
    </div>

    <div class="modal fade" id="deleteAccountModal">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h4 class="modal-title text-danger">Please don't go!</h4>
                </div>
                <form method="POST">
                    <div class="modal-body">
                        <p>This is your final chance to back out. We're
                            not messing around here - you'll honestly lose
                            everything you've ever done at Parkwood Vale Harriers! Are you really sure you want to delete
                            your account?</p>
                        <label> Enter the message:
                            <input type="text" name="delete" placeholder="I will lose everything" class="form-control delete-input" />
                        </label>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-success" data-dismiss="modal">No, I was just joking!</button>
                        <button type="submit" class="btn btn-danger">Delete account</button>
                    </div>
                </form>
            </div>
        </div>
    </div>

{% endblock %}
```

Listing 5: User Profile Page

10.1.6 add_training.html

```
{% extends 'layout.html' %}

{% block title %}Add Training Session{{ super() }}{% endblock %}

{% block content %}
    <h1>Add a Training Session - {{ date.strftime('%A %e %B %G') }}
        </h1>
    <h4>Done some exercise? Record it here to add it to your
        training log.</h4>

    <button class="btn btn-running sport-button" id="running">Add
        Running</button>
    <button class="btn btn-warning sport-button" id="cycling">Add
        Cycling</button>
    <button class="btn btn-info sport-button" id="swimming">Add
        Swimming</button>

    <button class="btn btn-primary"><span class="total-calories">{{
         total_calories }}</span> calories | <span
            class="total-hours">{{ total_hours }}</span> hours
    </button>
    <br/>

    <div class="row">
        <ul class="activity-list">
            <li class="row">
                {% for activity in activities %}
                    <li class="saved-activity activity-block-{{
                        activity.sport|lower }} added col-md-12"
                        id="{{ activity.id }}">
                        <span class="sport">{{ activity.sport|title
                             }} ({{ activity.effigy|lower }})</span
                            >
                        <span class="calories"> - {{ activity.
                            calories }} calories</span>
                        <span class="hours">burned over {{ activity
                            .hours }} {% if activity.hours == 1 %}
                            hour{% else %}
                            hours{% endif %}</span>
                        <span class="glyphicon glyphicon-remove"></
                            span>
                    </li>
                {% endfor %}
            </li>
        </ul>
    </div>

    {% if activities|length < 1 %}
```

```
        <h4 class="no-activities">You haven't added any activities
            today! Use the buttons above to add one.</h4>
    {% endif %}
{% endblock %}
```

Listing 6: Add Training Session Page

10.1.7 compare_performance.html

```
{% extends 'layout.html' %}

{% block title %}Compare Performance{{ super() }}{% endblock %}

{% block content %}

    <h1>Compare Performance</h1>
    <p>Want to see how you re doing compared to others? Use this
        page!</p>

    <label for="user_list">Select user to compare aginst:</label>
    <select name="user_list" id="user_list" class="form-control">
        {% for user in user_list %}
            <option value="{{ user[0] }}">{{ user[1] }}</option>
        {% endfor %}
    </select>

    <h3>Graphical Comparison</h3>

    <div class="graph_buttons">
        <div class="btn-group">
            <div class="btn btn-success" id="running_calories">
                Running Calories</div>
            <div class="btn btn-success" id="running_hours">Running
                 Hours</div>
        </div>
        <div class="btn-group">
            <div class="btn btn-warning" id="cycling_calories">
                Cycling Calories</div>
            <div class="btn btn-warning" id="cycling_hours">Cycling
                 Hours</div>
        </div>
        <div class="btn-group">
            <div class="btn btn-info" id="swimming_calories">
                Swimming Calories</div>
            <div class="btn btn-info" id="swimming_hours">Swimming
                Hours</div>
        </div>
    </div>

    <br>

    <div class="row">
        <div class="col-md-12"><canvas id="running_comparison"
            width="1140" height="600"></canvas></div>
    </div>

    <h3>Statistical Comparison</h3>

    <div class="row">
        <div class="col-md-6">
            <div class="panel panel-success">
                <div class="panel-heading">
                    <div class="panel-title">Your Performance</div>
                </div>
                <ul class="list-group panel-list">
                    <li class="list-group-item">gosh</li>
                    <li class="list-group-item">gosh</li>
                    <li class="list-group-item">gosh</li>
                    <li class="list-group-item">gosh</li>
                    <li class="list-group-item">gosh</li>
                </ul>
            </div>
        </div>
        <div class="col-md-6">
            <div class="panel panel-info">
                <div class="panel-heading">
                    <div class="panel-title">Their Performance</div
                        >
                </div>
                <ul class="list-group panel-list">
                    <li class="list-group-item">gosh</li>
                    <li class="list-group-item">gosh</li>
                    <li class="list-group-item">gosh</li>
                    <li class="list-group-item">gosh</li>
                    <li class="list-group-item">gosh</li>
                </ul>
            </div>
        </div>
    </div>

{% endblock %}
```

Listing 7: Compare Performance

10.1.8 rankings.html

```
{% extends 'layout.html' %}

{% block title %}Team Rankings{{ super() }}{% endblock %}

{% block content %}

<h1>Team Rankings</h1>
<h4>View the current team for the charity event, updated using up
    to date data from your fellow runners!</h4>

<div class="rankings">
    <div class="row">
        <div class="col-md-8">
            <div class="panel panel-success">
                <div class="panel-heading">
                    <div class="panel-title">Main Charity Team</div
                        >
                 </div>
                 <ul class="user-details list-group panel-list">
                     {% for runner in running_team %}
```

```
                        {% if loop.index <= 8 %}
                            <li class="list-group-item">{{ loop.index }}. {{ runner }}</li>
                        {% endif %}
                     {% endfor %}
                 </ul>
            </div>
        </div>
        <div class="col-md-4">
            <div class="panel panel-primary">
                <div class="panel-heading">
                    <div class="panel-title">Reserve Team</div>
                 </div>
                 <ul class="user-details list-group panel-list">
                     {% for runner in running_team %}
                        {% if 9 <= loop.index <= 12 %}
                             <li class="list-group-item">{{ loop.index }}. {{ runner }}</li>
                        {% endif %}
                     {% endfor %}
                 </ul>
            </div>
        </div>
    </div>
</div>

{% endblock %}
```

Listing 8: Rankings Page

10.1.9 running_block.html

```
<li class="activity">
    <div class="col-lg-4 col-md-6 col-sm-12 inner-activity">
        <div class="panel panel-running activity-block" id="Running">
            <div class="panel-heading">
                <div class="panel-title">
                    <span class="sport">Running</span>
                    <span class="glyphicon glyphicon-remove"></span>
                </div>
            </div>
            <div class="panel-body">
                <form>
                    <div class="form-group">
                        <label>What was your average speed?
                            <select name="effigy" id="effigy" class="form-control activity-input running-input">
                                <option value="5 mph">5 mph</option>
                                <option value="6 mph">6 mph</option>
                                <option value="7 mph">7 mph</option>
                                <option value="8 mph">8 mph</option>
                                <option value="9 mph">9 mph</option
                                    >
                                <option value="10 mph">10 mph</
                                    option>
                            </select>
                        </label>
                    </div>
                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                                <label>What start time?
                                    <input class='form-control
                                        activity-input time running
                                        -input' id="start">
                                </label>
                            </div>
                        </div>
                        <div class="col-md-6">
                            <div class="form-group">
                                <label>What finish time?
                                    <input class='form-control
                                        activity-input time running
                                        -input' id="finish">
                                </label>
                            </div>
                        </div>
                    </div>
                    <div class="form-group">
                        <label>How would you rate your run?
                            <select name="rating" id="rating" class
                                ="form-control activity-input
                                running-input">
                                <option value="Brilliant">Brilliant
                                    </option>
                                <option value="Pretty good">Pretty
                                    good</option>
                                <option value="About average">About
                                     average</option>
                                <option value="Okay">Okay</option>
                                <option value="Awful">Awful</option
                                    >
                            </select>
                        </label>
                    </div>
                    <div class="form-group">
                        <label>Do you have any extra thoughts?
                            <textarea name="thoughts" id="thoughts"
                                      class="activity-input form-
                                          control running-input"></
                                          textarea>
                        </label>
                    </div>
                    <div class="row">
                        <div class="col-sm-12 col-md-12">
                            <input type="button" class="btn btn-
                                running activity-input add-activity
                                 running-input"
```

```
                                   value="Add run"/>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</li>
```

Listing 9: Running Block

10.1.10 cycling_block.html

```
<li class="activity">
    <div class="col-lg-4 col-md-6 col-sm-12 inner-activity">
        <div class="panel panel-warning activity-block" id="Cycling">
            <div class="panel-heading">
                <div class="panel-title">
                    <span class="sport">Cycling</span>
                    <span class="glyphicon glyphicon-remove"></span>
                </div>
            </div>
            <div class="panel-body">
                <form>
                    <div class="row">
                        <div class="col-md-12">
                            <div class="form-group">
                                <label>How fast were you cycling?
                                    <select name="effigy" id="effigy" class="form-control activity-input cycling-input">
                                        <option value="Leisurely">Leisurely</option>
                                        <option value="Gently">Gently</option>
                                        <option value="Moderately">Moderately</option>
                                        <option value="Vigorously">Vigorously</option>
                                        <option value="Very fast">Very Fast</option>
                                        <option value="Racing">Racing</option>
                                    </select>
                                </label>
                            </div>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                                <label>What start time?
                                    <input class='form-control activity-input time cycling-input' id="start">
                                </label>
                            </div>
                        </div>
                        <div class="col-md-6">
                            <div class="form-group">
                                <label>What finish time?
                                    <input class='form-control activity-input time cycling-input' id="finish">
                                </label>
                            </div>
                        </div>
                    </div>
                    <div class="form-group">
                        <label>How would you rate your cycle?
                            <select name="rating" id="rating" class="form-control activity-input cycling-input">
                                <option value="Brilliant">Brilliant</option>
                                <option value="Pretty good">Pretty good</option>
                                <option value="About average">About average</option>
                                <option value="Okay">Okay</option>
                                <option value="Awful">Awful</option>
                            </select>
                        </label>
                    </div>
                    <div class="form-group">
                        <label>Do you have any extra thoughts?
                            <textarea name="thoughts" id="thoughts" class="activity-input form-control cycling-input"></textarea>
                        </label>
                    </div>
                    <div class="row">
                        <div class="col-sm-12 col-md-12">
                            <input type="button" class="btn btn-warning activity-input add-activity"
                                   value="Add cycle"/>
                        </div>
                    </div>

                </form>
            </div>
        </div>
    </div>
</li>
```

Listing 10: Cycling Block

10.1.11 swimming_block.html

```
<li class="activity">
    <div class="col-lg-4 col-md-6 col-sm-12 inner-activity">
```

```html
        <div class="panel panel-info activity-block" id="Swimming">
            <div class="panel-heading">
                <div class="panel-title">
                    <span class="sport">Swimming</span>
                    <span class="glyphicon glyphicon-remove"></span>
                </div>
            </div>
            <div class="panel-body">
                <form>
                    <div class="form-group">
                        <label>Which style did you use?
                            <select name="effigy" id="effigy" class="form-control activity-input swimming-input">
                                <option value="Backstroke">Backstroke</option>
                                <option value="Breaststroke">Breaststroke</option>
                                <option value="Butterfly">Butterfly</option>
                                <option value="Freestyle (slow)">Freestyle (slow)</option>
                                <option value="Freestyle (fast)">Freestyle (fast)</option>
                            </select>
                        </label>
                    </div>
                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                                <label>What start time?
                                    <input class='form-control activity-input time swimming-input' id="start">
                                </label>
                            </div>
                        </div>
                        <div class="col-md-6">
                            <div class="form-group">
                                <label>What finish time?
                                    <input class='form-control activity-input time swimming-input' id="finish">
                                </label>
                            </div>
                        </div>
                    </div>
                    <div class="form-group">
                        <label>How would you rate your swim?
                            <select name="rating" id="rating" class="form-control activity-input swimming-input">
                                <option value="Brilliant">Brilliant</option>
                                <option value="Pretty good">Pretty good</option>
                                <option value="About average">About average</option>
                                <option value="Okay">Okay</option>
                                <option value="Awful">Awful</option>
                            </select>
                        </label>
                    </div>
                    <div class="form-group">
                        <label>Do you have any extra thoughts?
                            <textarea name="thoughts" id="thoughts"
                                      class="activity-input form-control swimming-input"></textarea>
                        </label>
                    </div>
                    <div class="row">
                        <div class="col-sm-12 col-md-12">
                            <input type="button" class="btn btn-info activity-input add-activity" value="Add swim"/>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</li>
```

Listing 11: Swimming Block

10.2 JavaScript Functions

The system makes use of some JavaScript in order to create links between the front-end (the HTML files above) and the Python functions. Very little processing is done here; mainly data is transmitted back and forth between the client and the server.

10.2.1 main.js

```
$(document).ready(function () {

    // Initialises the datepicker plugin for all inputs with a class of "datepicker"
    $('.datepicker').datepicker({endDate: '-18y', startDate: '-75y', format: 'yyyy-mm-dd'});

    $('#running-activities, #cycling-activities, #swimming-activities').DataTable({
        //"filter": false
    });

    function genericAnimation($element, animation, timeout) {
        $element.addClass('animated ' + animation);
        if (timeout === true) {
            setTimeout(function () {
                $element.removeClass('animated ' + animation);
            }, 1400);
        }
    }

    // Animates the removal of the block
    function animateRemove($activity) {
        genericAnimation($activity, 'zoomOut', false);
        setTimeout(function () {
            $activity.remove();
        }, 175);
    }

    // Called when the delete button on an activity block is
        pressed
    $('.saved-activity .glyphicon').click(function () {
        var $activity = $(this).closest('li'),
            toRemove = {"activityId": $activity.attr('id')};
        // If the activity block has been returned from the
            database
        if ($activity.hasClass('added')) {
            animateRemove($activity);
            ajaxCall('/ajax/remove-activity', 'POST', 'json', '
                application/json', JSON.stringify(toRemove), null);
        } else {
            animateRemove($activity);
        }
    });

    // Sends a request to the server for the correct
    $('.sport-button').click(function () {
        var activity = $(this).attr('id');
        ajaxCall('/ajax/sport-block', 'POST', 'text', 'text/plain',
             activity, updateActivities);
    });

    // Validates that times have been entered in the activity block
    function validateActivity($activity) {
        var $start = $activity.find('#start'),
            $finish = $activity.find('#finish');
        ($start, $finish).removeClass('animated zoomIn');
        if ($start.val() === '') {
            genericAnimation($start, 'shake', true);
        }
        if ($finish.val() === '') {
            genericAnimation($finish, 'shake', true);
        }
        if ($start.val() !== '' && $finish.val() !== '') {
            animateActivity($activity);
        }
    }

    function updateActivities($activity) {
        $activity = $($activity);
        genericAnimation($('.no-activities'), 'fadeOutDown', 300);
        $('.activity-list').append($activity);
        genericAnimation($activity, 'zoomIn', false);
        $('.time').pickatime({interval: 60, formatLabel: 'HH:i A', formatSubmit: 'HH:i A'});
        // If the delete button is pressed, call the remove function
        $('.activity-block .glyphicon').click(function () {
            animateRemove($(this).closest('li'));
        });
        // If the add button is clicked, call the validate function
        $('.add-activity').click(function () {
            validateActivity($(this).closest('.panel'));
        });
    }

    function animateActivity($activity) {
        var sport = $activity.attr('id'),
            containerWidth = $('.container').width();
        $activity.find('label, input, select, textarea, .panel-body').addClass('animated zoomOut');
        setTimeout(function () {
            $activity.find('.panel-heading').animate({
                width: containerWidth, height: 60, borderBottomLeftRadius: 4,
                borderBottomRightRadius: 4, paddingTop: 17
            }, 500);
            $activity.find('.activity-block').css('margin-bottom', '15px');
            $activity.parent().removeClass('col-lg-4 col-md-6 col-sm-12').addClass('col-lg-12 col-md-12 col-sm-12');
            $activity.find('label, input, select, textarea, .form-group, .panel-body').hide();
        }, 200);
        calculateCalories(sport, $activity);
    }

    // Calculates the number of hours between the start and finish times
    function calculateHours($activity) {
        var start = new Date('01/01/2000 ' + $activity.find('#start').val()).getHours(),
            stop = new Date('01/01/2000 ' + $activity.find('#finish').val()).getHours();
        return stop - start;
    }

    function calculateCalories(sport, $activity) {
        // Activity information needed for calculations are displayed here
        var effigy = $activity.find('#effigy').val(),
            rating = $activity.find('#rating').val(),
            start = $activity.find('#start').val(),
            finish = $activity.find('#finish').val(),
            thoughts = $activity.find('#thoughts').val(),
            hours = calculateHours($activity);
        ajaxCall('/ajax/calculate-calories', 'POST', 'json', 'application/json', JSON.stringify({
```

```javascript
            "sport": sport,
            "effigy": effigy,
            "hours": hours,
            "thoughts": thoughts,
            "start": start,
            "finish": finish,
            "rating": rating
        }), addActivity, $activity);
    }

    function addActivity(data, $activity) {
        var caloriesBurned = data.calories,
            currentCalories = parseInt($('.total-calories').text()),
            currentHours = parseInt($('.total-hours').text()),
        // Builds a string to display in the animated activity block
        // activityString = data.sport + ' (' + effigy.toLowerCase() + ') - ' + caloriesBurned + ' calories burned over ' + data.hours + ' hours',
            activityString = data.sport,
        // Constructs the final activity object in JSON, to send to the server and save to the database
            activityObject = {
                "sport": data.sport.toLowerCase(),
                "effigy": data.effigy,
                "calories": caloriesBurned,
                "start": data.start,
                "finish": data.finish,
                "hours": data.hours,
                "rating": data.rating,
                "thoughts": data.thoughts
            };

        $('.total-hours').text(currentHours + data.hours);
        $('.total-calories').text(currentCalories + caloriesBurned);

        $activity.find('.sport').text(activityString);

        ajaxCall('/ajax/send-activity', 'POST', 'json', 'application/json', JSON.stringify(activityObject), null)
    }

    // A generic function that sends a request to the server and calls a function with the returned data
    function ajaxCall(url, requestType, dataType, contentType, data, callbackFunction, activity) {
        $.ajax({
            url: url,
            type: requestType,
            dataType: dataType,
            contentType: contentType,
            data: data,
            success: function (data) {
                if (typeof activity != 'undefined') {
                    callbackFunction(data, activity);
                } else {
                    callbackFunction(data);
                }
            }
        })
    }

    $('[data-toggle="tooltip"]').tooltip();
    Chart.defaults.global.scaleFontFamily = "'Raleway', 'Helvetica
        ', 'Arial', sans-serif";

});
```

Listing 12: Main JavaScript Functions

10.2.2 individual_charts.js

```
$(document).ready(function () {

    $.ajax({
        url: '/ajax/user-charts',
        type: 'POST',
        dataType: 'json',
        contentType: 'application/json',
        data: JSON.stringify({"month": $('.calorie-subtitle').text
            ().replace(' Calorie Progress', '')}),
        success: function (data) {
            constructUserChart(data)
        }
    });

    function constructUserChart(chartData) {
        var runningCtx = document.getElementById("runningChart").
            getContext("2d");
        var runningData = {
            labels: chartData.activities.running.dates,
            datasets: [{
                label: 'Running',
                strokeColor: "rgba(16,170,59, 0.8)",
                fillColor: "rgba(82,170,94, 0.8)",
                data: chartData.activities.running.calories
            }]
        };
        var cyclingCtx = document.getElementById("cyclingChart").
            getContext("2d");
        var cyclingData = {
            labels: chartData.activities.cycling.dates,
            datasets: [{
                label: 'Cycling',
                strokeColor: "rgba(236,151,31,0.8)",
                fillColor: "rgba(240,173,78,0.8)",
                data: chartData.activities.cycling.calories
            }]
        };
        var swimmingCtx = document.getElementById("swimmingChart").
            getContext("2d");
        var swimmingData = {
            labels: chartData.activities.swimming.dates,
            datasets: [{
                label: 'Swimming',
                strokeColor: "rgba(49,176,213,0.8)",
                fillColor: "rgba(91,192,222,0.8)",
                data: chartData.activities.swimming.calories
            }]
        };

        var runningChart = new Chart(runningCtx).Line(runningData,
            {bezierCurve: false});
        var cyclingChart = new Chart(cyclingCtx).Line(cyclingData,
            {bezierCurve: false, animation: false});
        var swimmingChart = new Chart(swimmingCtx).Line(
            swimmingData, {bezierCurve: false, animation: false});
    }

    $('.activity-change').click(function () {
        var sport = $(this).attr('id');
        if ($('.' + sport + '-data').hasClass('active') == false) {
            $('.active').addClass('animated bounceOutRight');
            setTimeout(function () {
                $('.active').css('display', 'none').removeClass('
                    animated bounceOutRight active');
                $('.' + sport + '-data').css('display', 'block').
                    addClass('animated bounceInLeft active');
            }, 600)
        }
    })

    $('.trainingHeading').click(function() {
        $('.runningChart').update();
    })

});
```

Listing 13: User Charts

10.3 CSS Styling

A master CSS file is used to provide styling for the system, setting out things like the typography, layout and a little animation in places.

```
@font-face {
    font-family: 'ralewayitalic';
    src: url('../fonts/raleway-regular-italic-webfont.eot');
    src: url('../fonts/raleway-regular-italic-webfont.eot?#iefix')
        format('embedded-opentype'),
    url('../fonts/raleway-regular-italic-webfont.woff2') format('
        woff2'),
    url('../fonts/raleway-regular-italic-webfont.woff') format('
        woff'),
    url('../fonts/raleway-regular-italic-webfont.ttf') format('
        truetype'),
    url('../fonts/raleway-regular-italic-webfont.svg#ralewayitalic'
        ) format('svg');
    font-weight: normal;
    font-style: normal;
```

```css
}
@font-face {
    font-family: 'ralewaymedium';
    src: url('../fonts/raleway-medium-webfont.eot');
    src: url('../fonts/raleway-medium-webfont.eot?#iefix') format('embedded-opentype'),
    url('../fonts/raleway-medium-webfont.woff2') format('woff2'),
    url('../fonts/raleway-medium-webfont.woff') format('woff'),
    url('../fonts/raleway-medium-webfont.ttf') format('truetype'),
    url('../fonts/raleway-medium-webfont.svg#ralewaymedium') format('svg');
    font-weight: normal;
    font-style: normal;

}
@font-face {
    font-family: 'ralewaysemibold';
    src: url('../fonts/raleway-semibold-webfont.eot');
    src: url('../fonts/raleway-semibold-webfont.eot?#iefix') format('embedded-opentype'),
    url('../fonts/raleway-semibold-webfont.woff2') format('woff2'),
    url('../fonts/raleway-semibold-webfont.woff') format('woff'),
    url('../fonts/raleway-semibold-webfont.ttf') format('truetype'),
    url('../fonts/raleway-semibold-webfont.svg#ralewaysemibold') format('svg');
    font-weight: normal;
    font-style: normal;

}
/*------------------------------------------------------
|                      Begin footer styles              |
-------------------------------------------------------*/
.footer {
    width: 100%;
    border-top: 1px solid #eeeeee;
    text-align: center;
    font-family: ralewaymedium, "Helvetica Neue", Helvetica, Arial, sans-serif !important;
    padding-top: 35px;
    vertical-align: middle;
    line-height: normal;
    margin: 0;
    position: fixed;
    bottom: 35px;
}
/*------------------------------------------------------
|                      Begin misc hacks                |
-------------------------------------------------------*/
.input_membership_charity {
    margin-left: 5px;
}
.remember-label {
    width: 17%;
}
.charity-label {
    width: 50%;
```

```css
}
/*-----------------------------------------------------
|         Begin general typography styles              |
------------------------------------------------------*/
h1 {
    color: #292929;
    font-family: ralewaymedium, sans-serif;
}
h4 {
    color: #2d2d2d;
    font-weight: 400;
    font-size: 20px;
    font-family: ralewaymedium, sans-serif;
}
label, p, .btn, ul.add-sport-buttons, .datepicker {
    font-family: ralewaysemibold, sans-serif, "Helvetica Neue",
        Helvetica, Arial, sans-serif;
    font-weight: 100;
}
label {
    font-size: 14px;
    width: 100%;
}
.activity-block label {
    width: 100%;
}
.timepicker {
    background-color: #ffffff !important;
    cursor: auto !important;
}
.details p {
    margin-bottom: 3px;
    font-size: 20px;
    font-weight: 800;
}
.jumbotron .alert p {
    font-size: 20px;
}
/*-----------------------------------------------------
|              Begin general input styles              |
------------------------------------------------------*/
input:not(.input_membership_charity):not(.add-activity):not(.btn-
    modal), select, textarea {
    width: 100%;
    border-radius: 4px;
    box-shadow: none !important;
    -webkit-box-shadow: none !important;
    font-family: ralewaymedium, "Helvetica Neue", Helvetica, Arial,
         sans-serif;
}
.datepicker {
    padding-left: 12px !important;
}
/*-----------------------------------------------------
|             Begin general button styles              |
------------------------------------------------------*/

```

```
.btn {
    font-family: ralewaysemibold, "Helvetica Neue", Helvetica,
        Arial, sans-serif;
}
.btn-running {
    background-color: #52aa5e;
    color: #ffffff;
}
.btn-running:hover {
    background-color: #10aa3b;
    color: #ffffff;
}
.btn-running:focus {
    color: #ffffff;
}
/*-------------------------------------------------------
|              Begin register form styles               |
--------------------------------------------------------*/
.charity-row {
    height: 25px;
}
/*-------------------------------------------------------
|                 Begin add training styles               |
--------------------------------------------------------*/
/*The ul container in which the activity li's are placed.*/
.activity-list {
    margin-top: 30px;
    list-style-type: none;
    padding: 0;
}
.activity-list .glyphicon {
    float: right;
    font-size: 14px;
    top: 7px;
    color: #ffffff;
}
.activity-list .glyphicon:hover {
    color: rgba(255, 255, 255, 0.5);
    transition: all 0.3s ease;
    cursor: pointer;
}
.activity-list textarea {
    height: 110px;
}
.activity-list .form-group {
    margin-bottom: 7px;
}
.activity-list .btn {
    margin-top: 9px;
}
/*The actual activity li.*/
.activity {
    -webkit-animation-duration: 0.375s;
}
.add-activity {
    width: 100%;
}
```

```
#Cycling .panel-body, .cycling-input {
    border: 1px solid #f0ad4e;
}
#Running .panel-body, .running-input {
    border: 1px solid #52aa5e;
}
#Swimming .panel-body, .swimming-input {
    border: 1px solid #5bc0de;
}
.activity-block-cycling {
    background-color: #f0ad4e;
}
.activity-block-running {
    background-color: #52aa5e;
}
.activity-block-swimming {
    background-color: #5bc0de;
}
.saved-activity {
    height: 60px;
    border-radius: 4px;
    margin-bottom: 15px;
    margin-left: 15px;
    font-family: ralewaysemibold, sans-serif;
    color: #ffffff;
    font-size: 18px;
    padding-top: 17px;
}
.activity-block .panel-body {
    padding: 24px;
    border-bottom-left-radius: 4px;
    border-bottom-right-radius: 4px;
}
.panel-running > .panel-heading {
    background-color: #52aa5e;
    color: #ffffff
}
/*Misc activity adder styles*/
.time {
    background-color: #ffffff !important;
    cursor: default !important;
}
/*-----------------------------------------------------
|                   Begin account page                  |
------------------------------------------------------*/
.delete-account {
    margin-top: 8px;
}
.panel-heading {
    font-weight: 600;
    font-family: ralewaysemibold, sans-serif;
}
.panel {
    font-family: ralewaymedium, sans-serif;
}
.panel-list {
    border-left: 1px solid #dddddd;
    border-bottom: 1px solid #dddddd;
    border-right: 1px solid #dddddd;
    border-bottom-left-radius: 4px;
    border-bottom-right-radius: 4px;
}
.right {
    float: right;
    font-family: ralewayitalic, sans-serif;
}
/*-------------------------------------------------------
|                  Begin table styles                   |
--------------------------------------------------------*/
.calorie-progress-bars {
    list-style-type: none;
    margin-bottom: 35px;
    padding: 0;
}
h3 {
    font-family: ralewaysemibold, sans-serif !important;
}
.tooltip {
    font-family: ralewaysemibold, sans-serif;
}
.performance-subtitle, .calorie-subtitle, .hour-subtitle {
    -webkit-animation-duration: 0.575s;
}
.month-buttons {
    list-style-type: none;
    display: inline;
}
.month-buttons li {
    display: inline;
}
.nav-pills, .no-footer {
    font-family: 'ralewaymedium', sans-serif;
}
input[type=search] {
    width: 90% !important;
}
.activity-view {
    margin-top: 25px;
}
.running-data, .cycling-data, .swimming-data {
    margin-bottom: 100px;
}
.cycling-data, .swimming-data {
    display: none;
}
table {
    border-right: 4px;
}
/*-------------------------------------------------------
|                  Begin comparison styles              |
--------------------------------------------------------*/
.graph_buttons {
    padding: 0;
}
```

```
ul.month_buttons {
    padding: 0 !important;
}
```

Listing 14: main.css

10.4 Python Processes

The vast majority of the system is written in Python. These function handle everything from connecting and writing to the database, to calculating the number of calories burned in a training session, and everything in between. For a full rundown of what each function does, view the processes section.

10.4.1 __init__.py

This file handles very low level functions of the system, like creating and initialising the actual Flask application.

```python
from flask import Flask
from flask.ext.login import LoginManager

from app.models import db, User


def create_app():
    """Generates an instance of the app.

    This function contains all the config values
    for the different parts of the app; it returns
    a variable 'app' that contains all these values
    for use throughout the rest of the application.
    """
    app = Flask(__name__)

    # Sets the application into debug mode
    app.debug = True

    # Sets configuration variables used application-wise
    app.config['SECRET_KEY'] = 'vYqTMY88zsuXSG7R4xYdPxYk'
    app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///../database.
        db'

    # Configures SQLAlchemy
    db.init_app(app)

    # Configures the login manager
    login_manager = LoginManager()
    login_manager.init_app(app)
    login_manager.login_view = 'auth.login'  # Sets the login view.
    login_manager.login_message_category = 'warning'

    # Loads the current user by running a query on the id
    @login_manager.user_loader
    def load_user(id):
        return User.query.get(int(id))

    # Configures application blueprints
    from app.controllers.main import main
    app.register_blueprint(main)

    from app.controllers.auth import auth
    app.register_blueprint(auth)

    from app.controllers.ajax import ajax
    app.register_blueprint(ajax)

    return app

if __name__ == '__main__':
    app = create_app()
    app.run(debug=True)
```

Listing 15: __init__.py

10.4.2 forms.py

This file defines the input forms used in the login and register pages. It sets the validation for each input, and defines the appropriate HTML element.

```python
from flask.ext.wtf import Form
from wtforms import StringField, PasswordField, DateField,
    BooleanField, SubmitField, SelectField, IntegerField
from wtforms.validators import DataRequired, Email, Length, EqualTo
    , Regexp, ValidationError, NumberRange

from app.models import User
from app.helpers import calculate_age


class MemberForm(Form):
    """Contains the fields and validators for the new member form.
        """

    name = StringField("What is your name?", validators=[
        DataRequired('You must enter your name.'),
                                                         Regexp(r'
                                                             ^[A-Za
                                                             -z\-"
                                                             "]*$',
                                                                message
                                                                    =
                                                                    '
                                                                    Your

                                                                    name

                                                                    may

                                                                    only

                                                                    contain
```

```
letters.')])
    dob = DateField("What is your date of birth?", validators=[
        DataRequired('You must enter your date of birth.')])
    email = StringField("What is your email?",
                        validators=[DataRequired('You must enter
                            your email.'), Email('You must enter a
                            valid email.')])
    password = PasswordField("Enter a password:", validators=[
        DataRequired('You must enter a password.'),
                                                        Length(8, 20,
                                                            'Your password must be 8 - 20 characters.')])
    confirm = PasswordField("Confirm your password:", validators=[
        DataRequired('You must confirm your password.'),
                                                            EqualTo('password', 'Your passwords must match
                                                                .
                                                                ;
                                                                )
                                                                ])
    charity_event = BooleanField("I want the chance to run in the
        charity event")
    distance = SelectField('What is the maximum distance you have
        run in the past year?',
                           choices=[('l1', 'Less than 1 mile'), ('
                               1-5', '1 - 5 miles'), ('6-10', '6 -
                               10 miles'),
                                     ('11-15', '11 - 15 miles'), ('
                                         16-20', '16 - 20 miles'),
                                     ('g20', 'More than 20 miles')])
    weight = IntegerField('How much do you weigh in kg?',
        validators=[DataRequired('You must enter your weight.'),
                                                                NumberRange
                                                                    (10,

                                                                    100,

                                                                            '
                                                                                Your

                                                                                weight

                                                                                must

                                                                                be

                                                                                between

                                                                                10
                                                                                kg

                                                                                -

                                                                                100
                                                                                kg
                                                                                .
                                                                                '
                                                                                )
                                                                                ])
    phone = StringField('What is your phone number?', validators=[
        DataRequired('You must enter your phone number.'),
                                                            Regexp
                                                                (

                                                                r
                                                                    '
                                                                    ^\
                                                                    s
                                                                    *\(?(020[78]\)
                                                                    ?
```

```
?[1-9][0-9]{2}
?[0-9]{4})
|(0[1-8][0-9]{3}\)
?
?[1-9][0-9]{2}
?[0-9]{3})
\s*$'
,
message
=
'You must enter a valid UK phone number.'
)
])
    submit = SubmitField('Submit')

    def validate_distance(self, field):
        """Ensures the user has not ticked the charity event and is
            a poor runner."""
        charity_event = self.charity_event
        if field.data == 'l1' and charity_event.data is True:
            raise ValidationError('You must be physically fit to
                run in the charity event.')

    def validate_dob(self, field):
        """Ensures the user is between 18 - 75 years old."""
        age = calculate_age(field.data)
        if not 18 <= age <= 75:
            raise ValidationError('You must be 18 - 75 years old to
                join.')

    def validate_email(self, field):
        """Ensures the email address is unique"""
        if User.query.filter_by(email=field.data).first():
            raise ValidationError('That email address has already
                been registered.')


class LoginForm(Form):
    """Contains the fields and validators for the login form."""
    email = StringField('What is your email?',
                        validators=[DataRequired('You must enter
                            your email.'), Email('You must enter a
                            valid email.')])
    password = PasswordField('What is your password?', validators=[
        DataRequired('You must enter your password.')])
    remember = BooleanField('Remember me')
    login = SubmitField('Login')
```

Listing 16: forms.py

10.4.3 models.py

This file defines the database models used by the database. It sets up aspects like foreign/primary keys, and the data type of each column.

```
from flask.ext.sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash,
    check_password_hash
from flask.ext.login import UserMixin

db = SQLAlchemy()


class User(UserMixin, db.Model):
    """Defines the user table and the fields.

    Each variable represents an individual field
    for the database, pertaining to the data collected
    in app.forms.MemberForm. The data type is also declared.
    All fields are of variable length. There is a one-to-many
    relationship between users and activities.
    """
    __tablename__ = 'Users'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String)
    email = db.Column(db.String)
    username = db.Column(db.String)
    password_hash = db.Column(db.String)
    dob = db.Column(db.Date)
    phone = db.Column(db.String)
    weight = db.Column(db.Integer)
    distance = db.Column(db.String)
    joined = db.Column(db.DateTime)
    charity_event = db.Column(db.Boolean)

    activities = db.RelationshipProperty('Activity', backref='user'
        , lazy='dynamic')

    # Initialises the class to allow it to be referenced in helper
        functions.
    def __init__(self, name, username, email, dob, password,
        distance, charity_event, weight, phone, joined):
        self.name = name
        self.username = username
        self.email = email
        self.password = password
        self.dob = dob
        self.distance = distance
        self.charity_event = charity_event
        self.phone = phone
        self.weight = weight

    # Ensures the password is accessible.
    @property
    def password(self):
        raise AttributeError('Password is not a readable attribute.
            ')

    # Encrypts the password and assigns it to the class variable.
    @password.setter
    def password(self, password):
        self.password_hash = generate_password_hash(password)

    # Checks the entered password against the decrypted password
        hash.
    def check_password(self, value):
        return check_password_hash(self.password_hash, value)

    # Returns the id of the current user.
    def get_id(self):
        return self.id

    # Obligatory identification function.
    def __repr__(self):
        return '<User: %r>' % self.id


class Activity(db.Model):
    """Defines the activities table and the fields.

    Each variable represents an individual field
    for the database, pertaining to the data collected
    in app.static.js.main. The data type is also declared.
    A foreign key is established between the user table,
    with users.id acting as the key; this creates a
    one-to-one link between the two tables (one user can
    have multiple activities.
    """
    __tablename__ = 'Activities'
    id = db.Column(db.Integer, primary_key=True)
    sport = db.Column(db.String(8))
    effigy = db.Column(db.String)
    date = db.Column(db.Date)
    start = db.Column(db.String)
    finish = db.Column(db.String)
    hours = db.Column(db.Integer)
    calories = db.Column(db.Integer)
    opinion = db.Column(db.String)
    thoughts = db.Column(db.Text)

    user_id = db.Column(db.Integer, db.ForeignKey('Users.id'))

    # Initialises the class to allow it to be referenced in helper
        functions.
    def __init__(self, sport, effigy, date, start, finish, calories
        , opinion, thoughts, hours, user_id):
        self.sport = sport
        self.effigy = effigy
        self.date = date
        self.start = start
        self.finish = finish
        self.hours = hours
        self.calories = calories
        self.opinion = opinion
        self.thoughts = thoughts
        self.user_id = user_id

    # Obligatory identification function.
    def __repr__(self):
        return '<Activity: %r (%r)>' % (self.id, self.sport)
```

Listing 17: models.py

10.4.4 helpers.py

This file defines several smaller helper functions used multiple times throughout the system.

```
from flask import flash, redirect, url_for
from flask.ext.login import current_user

from app.models import db, Activity

from datetime import date


def update_user(user, element, redirect_user=True):
    """Adds the updated user to the db and reloads the page."""
    db.session.add(user)
    db.session.commit()
    flash('Your %s has been successfully changed!' % element, '
        success')
    if redirect_user:
        return redirect(url_for('main.profiles', username=user.
            username))


def validation_error(message):
    """Displays an appropriate error message and reloads the page.
        """
    flash(message, 'warning')
```

```python
    return redirect(url_for('main.profiles', username=current_user.
        username))


def calculate_age(born):
    """Calculates the age of the user"""
    today = date.today()
    return today.year - born.year - ((today.month, today.day) < (
        born.month, born.day))


def remove_sport(activity_id):
    """Removes the activity from the database"""
    Activity.query.filter_by(id=activity_id).delete()
    db.session.commit()
    print('Activity %s deleted' % id)
    return 'Activity %s deleted' % activity_id
```

Listing 18: helpers.py

10.4.5 performance_data.py

This file returns a JSON object containing all the training sessions for a user in a particular month. It is used throughout the system to return training data for use in tables and graphs.

```python
from calendar import month_name
from flask.ext.login import current_user
from app.models import db, Activity, User


def performance_data(month):
    """Creates a dictionary object with training data

    This function is used throughout the system to create
    a collection of a particular user's training activities.
    It performs several queries to the db and uses a number
    of loops and list comprehensions in order to
    """

    # Creates a list of months - January, February, etc.
    months = [month_name[x].lower() for x in range(1, 13)]

    # Queries the db for all of the user's activities.
    all_activities = Activity.query.filter_by(user_id=current_user.
        get_id()).all()

    # Queries the db for all of the user's different activities.
    all_runs = Activity.query.filter_by(user_id=current_user.get_id
        (), sport='running').all()
    all_cycles = Activity.query.filter_by(user_id=current_user.
        get_id(), sport='cycling').all()
    all_swims = Activity.query.filter_by(user_id=current_user.
        get_id(), sport='swimming').all()

    # Creates a dict with month names and values - Jan: 1 etc.
    month_map = dict(zip(months, range(1, 13)))

    # Sets the total monthly calorie and hourly goal.
    calorie_goal = 40000
    hour_goal = 100

    # [0] contains the calories burned; [1] contains the hours.
    total_run_data = [0, 0]
    total_cycle_data = [0, 0]
    total_swim_data = [0, 0]

    # Generates a list containing the data for every running activity using the above queries.
    run_list = [{'id': run.id, 'date': run.date.strftime('%d %b %y'), 'effigy': run.effigy, 'calories': run.calories,
                 'start': run.start, 'finish': run.finish, 'hours': run.hours, 'opinion': run.opinion} for run in
                all_runs if run.date.month == month_map[month]]

    cycle_list = [
        {'id': cycle.id, 'date': cycle.date.strftime('%d %b %y'), 'effigy': cycle.effigy, 'calories': cycle.calories,
         'start': cycle.start, 'finish': cycle.finish, 'hours': cycle.hours, 'opinion': cycle.opinion} for cycle in
        all_cycles if cycle.date.month == month_map[month]]

    swim_list = [
        {'id': swim.id, 'date': swim.date.strftime('%d %b %y'), 'effigy': swim.effigy, 'calories': swim.calories,
         'start': swim.start, 'finish': swim.finish, 'hours': swim.hours, 'opinion': swim.opinion} for swim in all_swims
        if swim.date.month == month_map[month]]

    # Updates the total_sport_data variables with the total calories and hours of each sport.
    for run in all_runs:
        if run.date.month == month_map[month]:
            total_run_data[0] += run.calories
            total_run_data[1] += run.hours

    for cycle in all_cycles:
        if cycle.date.month == month_map[month]:
            total_cycle_data[0] += cycle.calories
            total_cycle_data[1] += cycle.hours

    for swim in all_swims:
        if swim.date.month == month_map[month]:
            total_swim_data[0] += swim.calories
            total_swim_data[1] += swim.hours

    # Takes all the above data and creates a large dict structure by which it can be accessed.
    user_data = {
        'progress_data': {
            'running': {
                'calories': {
                    'value': total_run_data[0],
                    'percentage': total_run_data[0] / calorie_goal * 100
                },
                'hours': {
                    'value': total_run_data[1],
                    'percentage': total_run_data[1] / hour_goal * 100
                }
            },
            'cycling': {
                'calories': {
                    'value': total_cycle_data[0],
                    'percentage': total_cycle_data[0] / calorie_goal * 100
                },
                'hours': {
                    'value': total_cycle_data[1],
                    'percentage': total_cycle_data[1] / hour_goal * 100
                }
            },
            'swimming': {
                'calories': {
                    'value': total_swim_data[0],
                    'percentage': total_swim_data[0] / calorie_goal * 100
                },
                'hours': {
                    'value': total_swim_data[1],
                    'percentage': total_swim_data[1] / hour_goal * 100
                }
            }
        },
        'sport_data': {
            'running': run_list,
            'swimming': swim_list,
            'cycling': cycle_list
        },
        'month': month.title()
    }

    return user_data
```
Listing 19: performance_data.py

10.4.6 auth.py

This file defines the routes and processes used in the login / register process. They were placed in their own file for efficiency, and because they play a different part to others.

```
from datetime import datetime

from flask import Blueprint, render_template, flash, redirect, url_for
```

```python
from flask.ext.login import current_user, login_user, logout_user
from random import randint

from app.forms import MemberForm, LoginForm
from app.models import db, User


auth = Blueprint('auth', __name__)


@auth.route('/register', methods=['GET', 'POST'])
def register():
    """Renders the register page and saves new users to the database"""
    # Makes sure logged in users cannot access the register page
    if not current_user.is_authenticated():
        form = MemberForm()
        # If the submit button is pressed
        if form.validate_on_submit():
            # Generates a username for the user composed of their real name and a random number
            username = form.name.data.lower().replace(' ', '') + str(randint(1, 10))
            # Creates a User object with the data they typed in
            user = User(name=form.name.data, username=username, email=form.email.data, password=form.password.data,
                        dob=form.dob.data, distance=form.distance.data, charity_event=form.charity_event.data,
                        phone=form.phone.data, weight=form.weight.data, joined=datetime.now())
            # Saves the user to the database
            db.session.add(user)
            db.session.commit()
            print('%s has been registered.' % user.name)
            # Returns the user to the login page with a message
            flash('You can now login!', 'success')
            return redirect(url_for('auth.login'))
        # If there were validation errors, re-render the view and show them
        for error in form.errors.items():
            flash(error[1][0], 'warning')
        return render_template('auth/register.html', form=form)
    return redirect(url_for('main.home'))


@auth.route('/login', methods=['GET', 'POST'])
def login():
    """Renders the login page and logs in the user"""
    if not current_user.is_authenticated():
        form = LoginForm()
        if form.validate_on_submit():
            # Query that returns the first user with the entered email address.
            user = User.query.filter_by(email=form.email.data).first()
            # Checks that a user was returned and that the password is correct.
            if user is not None and user.check_password(form.password.data):
                # If so, log them in and redirect them to the home page
                login_user(user, form.remember.data)
                return redirect(url_for('main.home'))
            flash('Invalid email address or password.', 'warning')
        # If there were validation errors, re-render the view and show them
        for error in form.errors.items():
            flash(error[1][0], 'warning')
        return render_template('auth/login.html', form=form)
    return redirect(url_for('main.home'))


@auth.route('/logout')
def logout():
    """Logs the user out of the system"""
    logout_user()
    return redirect(url_for('main.home'))
```

Listing 20: auth.py

10.4.7 ajax.py

This file defines the routes used by the AJAX calls in the JavaScript files. All of these return a value, usually a JSON object, that is then used to dynamically update the page.

```python
from datetime import datetime
from math import ceil
from calendar import month_name

from flask import Blueprint, render_template, request, jsonify
from flask.ext.login import current_user

from app.models import Activity, db
from app.performance_data import performance_data
from app.helpers import remove_sport


ajax = Blueprint('ajax', __name__)


# Defines the route for displaying the activity blocks
@ajax.route('/ajax/sport-block', methods=['POST'])
def sport_block():
    sport = request.get_data().decode("utf-8")
    if sport == 'running':
        return render_template('training/running_block.html')
    elif sport == 'cycling':
        return render_template('training/cycling_block.html')
    elif sport == 'swimming':
        return render_template('training/swimming_block.html')
    else:
        return '%s was passed as a sport - no template is available
            for this.' % sport, 400


# Defines the route for uploading activity block data
@ajax.route('/ajax/send-activity', methods=['POST'])
def send_activity():
    sport = request.json['sport']
    effigy = request.json['effigy']
    calories = request.json['calories']
    hours = request.json['hours']
    start = request.json['start']
    finish = request.json['finish']
    opinion = request.json['rating']
    thoughts = request.json['thoughts']

    activity = Activity(sport=sport, effigy=effigy, calories=
        calories, hours=hours, start=start,
                        finish=finish, opinion=opinion, thoughts=
                            thoughts,
                        user_id=current_user.get_id(), date=
                            datetime.now().date())

    db.session.add(activity)
    db.session.commit()
    print('Successfully saved Activity %s (%s) to the database.' %
        (activity.id, activity.sport))
    return 'success', 200


@ajax.route('/ajax/remove-activity', methods=['POST'])
def remove_activity():
    activity_id = request.json['activityId']
    return remove_sport(activity_id)


@ajax.route('/ajax/calculate-calories', methods=['POST'])
def calculate_calories():
    """Calculates the number of calories burned in a session

    The base values were arrived at by dividing each value provided
         by the
    board by 80. The formula takes the correct base value, and
        multiplies it
    by the weight of the user. This is then multiplied by
    the number of hours. This value is modified based on how well
        the activity went -
    each of the five options is assigned a value from -10 to 10;
        this is then
    added to the total value to arrive at the final number of
        calories.
    """
    base_calories = {
        'swimming': {'Backstroke': 5.1625, 'Breaststroke': 7.375, '
            Butterfly': 8.1125, 'Freestyle (slow)': 5.1625,
                     'Freestyle (fast)': 7.375},
        'running': {'5 mph': 5.9, '6 mph': 7.375, '7 mph': 8.4875,
            '8 mph': 9.9625, '9 mph': 11.0625, '10 mph': 11.8},
        'cycling': {'Leisurely': 2.95, 'Gently': 4.425, 'Moderately
            ': 5.9, 'Vigorously': 6.125, 'Very fast': 8.85,
                    'Racing': 11.8},
        'modifiers': {'Brilliant': 10, 'Pretty good': 5, 'About
            average': 0, 'Okay': -5, 'Awful': -10}
    }
    sport = request.json['sport'].lower()
    effigy = request.json['effigy']
    hours = request.json['hours']
    start = request.json['start']
    finish = request.json['finish']
    thoughts = request.json['thoughts']
    rating = request.json['rating']

    base_value = base_calories[sport][effigy]
    calories = (base_value * current_user.weight) * hours
    modifier = base_calories['modifiers'][rating]
    calories += modifier

    activity_data = {'calories': str(ceil(calories)), 'sport':
        sport, 'hours': hours, 'effigy': effigy,
                     'start': start, 'finish': finish, 'rating':
                         rating, 'thoughts': thoughts}

    return jsonify(activity_data)


@ajax.route('/ajax/user-charts', methods=['POST'])
def user_charts():
    month_map = dict(zip([month_name[x].lower() for x in range(1,
        13)], range(1, 13)))
    user_month = month_map[request.json['month'].lower()]

    runs = Activity.query.filter_by(user_id=current_user.get_id(),
        sport='running').all()

    cycles = Activity.query.filter_by(user_id=current_user.get_id()
        , sport='cycling').all()

    swims = Activity.query.filter_by(user_id=current_user.get_id(),
         sport='swimming').all()

    activity_data = {
        'running': {'calories': [run.calories for run in runs if
            run.date.month == user_month],
                    'dates': [run.date.strftime('%d %b') for run in
                         runs if run.date.month == user_month]},
        'cycling': {'calories': [cycle.calories for cycle in cycles
             if cycle.date.month == user_month],
                    'dates': [cycle.date.strftime('%d %b') for
                        cycle in cycles if cycle.date.month ==
                        user_month]},
        'swimming': {'calories': [swim.calories for swim in swims
            if swim.date.month == user_month],
                    'dates': [swim.date.strftime('%d %b') for swim in swims if swim.date.month == user_month]}
    }
    return jsonify(activities=activity_data)


@ajax.route('/ajax/performance', methods=['POST'])
def ajax_performance():
    month = request.get_data().decode("utf-8").lower()
    user_data = performance_data(month)
    return jsonify(user_data=user_data)


@ajax.route('/ajax/comparison-graph', methods=['POST'])
def comparison_graphs():
    graph_type = request.json['graphType']
    comparison_user = int(request.json['comparisonUser'])

    user_runs = Activity.query.filter_by(user_id=current_user.get_id(), sport='running').all()
    comparison_runs = Activity.query.filter_by(user_id=comparison_user, sport='running').all()
    run_months = []
    for run in user_runs:
        if run.date.strftime('%B') not in run_months:
            run_months.append(run.date.strftime('%B'))

    user_cycles = Activity.query.filter_by(user_id=current_user.get_id(), sport='cycling').all()
    comparison_cycles = Activity.query.filter_by(user_id=comparison_user, sport='cycling').all()
    cycle_months = []
    for cycle in user_cycles:
        if cycle.date.strftime('%B') not in cycle_months:
            cycle_months.append(run.date.strftime('%B'))

    user_swims = Activity.query.filter_by(user_id=current_user.get_id(), sport='swimming').all()
    comparison_swims = Activity.query.filter_by(user_id=comparison_user, sport='swimming').all()
    swim_months = []
    for swim in user_swims:
        if swim.date.strftime('%B') not in swim_months:
            swim_months.append(swim.date.strftime('%B'))

    if graph_type == 'running_calories':
        graph_data = {'current_user': [run.calories for run in user_runs],
                      'comparison_user': [run.calories for run in comparison_runs], 'months': run_months}

    elif graph_type == 'running_hours':
        graph_data = {'current_user': [run.hours for run in user_runs],
                      'comparison_user': [run.hours for run in comparison_runs], 'months': run_months}

    elif graph_type == 'cycling_calories':
        graph_data = {'current_user': [cycle.calories for cycle in
            user_cycles],
                      'comparison_user': [cycle.calories for cycle
                          in comparison_cycles], 'months':
                          cycle_months}

    elif graph_type == 'cycling_hours':
        graph_data = {'current_user': [cycle.hours for cycle in
            user_cycles],
                      'comparison_user': [cycle.hours for cycle in
                          comparison_cycles], 'months':
                          cycle_months}

    elif graph_type == 'swimming_calories':
        graph_data = {'current_user': [swim.calories for swim in
            user_swims],
                      'comparison_user': [swim.calories for swim in
                           comparison_swims], 'months': swim_months
                          }

    elif graph_type == 'swimming_hours':
        graph_data = {'current_user': [swim.hours for swim in
            user_swims],
                      'comparison_user': [swim.hours for swim in
                          comparison_swims], 'months': swim_months}

    print(graph_data)
    return jsonify(graphData=graph_data)
```

Listing 21: ajax.py

10.4.8 main.py

This file defines the majority of routes used by the system.

```
from datetime import datetime
from math import floor
from calendar import month_name

from flask import Blueprint, render_template, flash, redirect,
    url_for, abort, request
from flask.ext.login import current_user, login_required,
    logout_user
from flask.ext.sqlalchemy import *
from random import randint
import re

from app.models import User, Activity, db
from app.helpers import validation_error, update_user, remove_sport
from app.performance_data import performance_data

main = Blueprint('main', __name__)

current_date = datetime.now().date()

```

```python

@main.route('/')
@login_required
def home():
    return redirect(url_for('main.performance', month='march'))


@main.route('/profiles/<username>', methods=['GET', 'POST'])
@login_required
def profiles(username):
    # If the user has attempted to change their profile
    if request.method == 'POST':
        user = User.query.filter_by(id=current_user.get_id()).first()

        # If the user tries to change their name
        if request.form.get('name'):
            only_letters = re.compile(r'^[A-Za-z\-" "]*$')
            if only_letters.match(request.form.get('name')):
                user.name = request.form.get('name').title()
                user.username = request.form.get('name').lower().replace(' ', '').replace('-', '') + str(randint(1, 10))
                update_user(user, 'name', False)
                return redirect(url_for('main.profiles', username=user.username))
            else:
                validation_error('Your name may only contain letters and dashes.')

        # If the user tries to change their email
        elif request.form.get('email'):
            valid_email = re.compile(r'^.+@[^.].*\.[a-z]{2,10}$')
            if valid_email.match(request.form.get('email')):
                user.email = request.form.get('email')
                update_user(user, 'email')
            else:
                validation_error('You must enter a valid email.')

        # If the user tries to change their phone number
        elif request.form.get('phone'):
            valid_phone = re.compile(
                r'^\s*\(?(020[78]\)? ?[1-9][0-9]{2} ?[0-9]{4})|(0[1-8][0-9]{3}\)? ?[1-9][0-9]{2} ?[0-9]{3})\s*$')
            if valid_phone.match(request.form.get('phone')):
                user.phone = request.form.get('phone')
                update_user(user, 'phone number')
            else:
                validation_error('You must enter a valid UK phone number.')

        # If the user tries to change their dob
        elif request.form.get('dob'):
            user.dob = request.form.get('dob')
            update_user(user, 'date of birth')

        # If the user tries to change their weight
        elif request.form.get('weight'):
            check_integer = re.compile(r'^-?[0-9]+$')
            if not check_integer.match(request.form.get('weight')):
                validation_error('You must enter a number.')
            elif not 10 <= int(request.form.get('weight')) <= 100:
                validation_error('Your weight must be between 10kg - 100kg.')
            else:
                user.weight = request.form.get('weight')
                update_user(user, 'weight')

        elif request.form.get('delete'):
            if request.form.get('delete') != 'I will lose everything':
                validation_error('You must type in the message exactly!')
            else:
                user_id = current_user.get_id()
                logout_user()
                User.query.filter_by(id=user_id).delete()
                Activity.query.filter_by(user_id=user_id).delete()
                db.session.commit()
                flash('Your account was successfully deleted - sorry to see you go!', 'success')
                return redirect(url_for('auth.login'))

    possible_user = User.query.filter_by(username=username).first_or_404()
    if current_user.username == possible_user.username:
        activity_number = len(Activity.query.filter_by(user_id=current_user.get_id()).all())
        total_users = len(User.query.all())

        return render_template('profiles/own_profile.html', current_user=current_user, activity_number=activity_number,
                               total_users=total_users)
    abort(403)

    return redirect(url_for('main.profiles', username=current_user.username))


@main.route('/add-training', methods=['GET', 'POST'])
@login_required
def add_training():
    activities = Activity.query.filter_by(user_id=current_user.get_id(), date=current_date).all()
    total_calories = 0
    total_hours = 0
    for activity in activities:
        total_calories += activity.calories
        total_hours += activity.hours
    return render_template('training/add_training.html', date=current_date,
                           current_user=current_user, activities=
                               activities, total_calories=
                               total_calories,
                           total_hours=total_hours)


@main.route('/performance/<month>', methods=['GET', 'POST'])
@login_required
def performance(month):
    months = [month_name[x].lower() for x in range(1, 13)]
    all_activities = Activity.query.filter_by(user_id=current_user.
        get_id()).all()
    available_months = []

    for activity in all_activities:
        for x in range(1, 13):
            if activity.date.month == x and months[x - 1] not in
                available_months:
                available_months.append(months[x - 1])
    print(available_months)

    if month.lower() in available_months:
        user_data = performance_data(month.lower())
        return render_template('performance/user_performance.html',
             user_data=user_data,
                                  current_month=month.title(), months=
                                      available_months)
    abort(404)


@main.route('/performance/compare', methods=['GET', 'POST'])
@login_required
def compare_performance():
    users = User.query.filter_by(charity_event=0).filter(User.id !=
         current_user.id).all()
    user_list = sorted([[user.id, user.name] for user in users])
    return render_template('/performance/compare_performance.html',
         users=users, user_list=user_list)


@main.route('/rankings')
@login_required
def rankings():
    user_ranking = {}
    runners = User.query.filter_by(charity_event=False).all()
    for runner in runners:
        total_calories = 0
        training_sessions = Activity.query.filter_by(user_id=runner
            .id).all()
        for session in training_sessions:
            total_calories += session.calories
        user_ranking[runner.name] = total_calories

    user_ranking = sorted(user_ranking, key=user_ranking.get,
        reverse=True)

    return render_template('/training/rankings.html', running_team=
        user_ranking)


@main.route('/delete/<int:activity_id>')
def delete_activity(activity_id):
    remove_sport(activity_id)
    flash('Your training session was deleted!', 'success')
    return redirect(url_for('main.home'))


@main.errorhandler(404)
def page_not_found(error):
    return render_template('errors/404.html'), 404
```

Listing 22: main.py

Name	**Type**	**File Found**	**Function / Class**	**Purpose**
login_manager	Object	_init_.py	create_app	Creates the login object.
name	Object	forms.py	MemberForm	Creates the name input.
dob	Object	forms.py	MemberForm	Creates the dob input.
password	Object	forms.py	MemberForm	Creates the password input.
confirm	Object	forms.py	MemberForm	Creates the confirm input.
charity_event	Object	forms.py	MemberForm	Creates the charity input.
distance	Object	forms.py	MemberForm	Creates the distance input.
weight	Object	forms.py	MemberForm	Creates the weight input.
phone	Object	forms.py	MemberForm	Creates the phone input,
submit	Object	forms.py	MemberForm	Creates the submit button.
age	int	forms.py	validate_dob	Stores the user's age.
email	Object	forms.py	LoginForm	Creates the email input.
password	Object	forms.py	LoginForm	Creates the password input.
remember	Bool	forms.py	LoginForm	Creates the remember input.
login	Object	forms.py	LoginForm	Creates the login button.
id	Object	models.py	User	Creates the id column.
name	Object	models.py	User	Creates the name column.
email	Object	models.py	User	Creates the email column.
username	Object	models.py	User	Creates the username column.
password_hash	Object	models.py	User	Creates the hash column.
dob	Object	models.py	User	Creates the dob column.
phone	Object	models.py	User	Creates the phone column.
weight	Object	models.py	User	Creates the weight column.
distance	Object	models.py	User	Creates the distance column.
joined	Object	models.py	User	Creates the joined column.
charity_event	Object	models.py	User	Creates the charity column.
activities	Object	models.py	User	Creates relationship property.
id	Object	models.py	Activity	Creates the id column.
sport	Object	models.py	Activity	Creates the effigy column.
date	Object	models.py	Activity	Creates the date column.
start	Object	models.py	Activity	Creates the start column.
finish	Object	models.py	Activity	Creates the finish column.
hours	Object	models.py	Activity	Creates the hours column.
calories	Object	models.py	Activity	Creates the calories column.
opinion	Object	models.py	Activity	Creates the opinion column.
thoughts	Object	models.py	Activity	Creates the thoughts column.
user_id	Object	models.py	Activity	Creates the user_id column.
today	Date	helpers.py	calculate_age	Stores the current date.
months	List	perf_data.py	perf_data	Stores a list of months.
all_activities	Object	per_data.py	perf_data	Stores all the user's sessions.
all_runs	Object	per_data.py	perf_data	Stores all the user's runs.
all_cycles	Object	per_data.py	perf_data	Stores all the user's cycles.
all_swims	Object	per_data.py	perf_data	Stores all the user's swims.
month_map	Dict	per_data.py	perf_data	Maps months to integers.
calorie_goal	int	per_data.py	perf_data	Stores the calorie goal.
hour_goal	int	per_data.py	perf_data	Stores the hourly goal.
sport	String	ajax.py	sport_block	Stores the type of sport
sport	String	ajax.py	send_activity	Retrieves the sport
effiy	String	ajax.py	send_activity	Retrieves the effigy
hours	int	ajax.py	send_activity	Retrieves the hours
start	int	ajax.py	send_activity	Retrieves the start
finish	int	ajax.py	send_activity	Retrieves the finish
opinion	String	ajax.py	send_activity	Retrieves the opinion

Part III

Testing and Evaluation