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

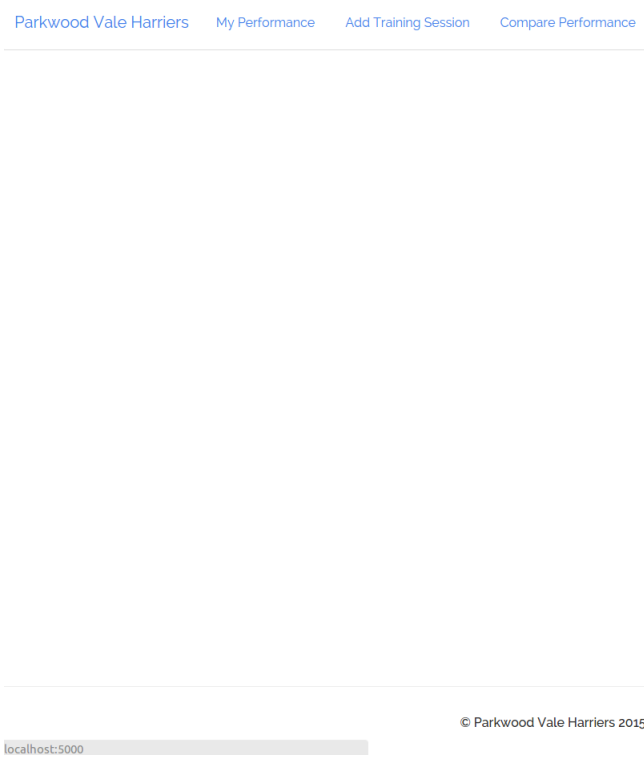
Part II
Program Documentation

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.

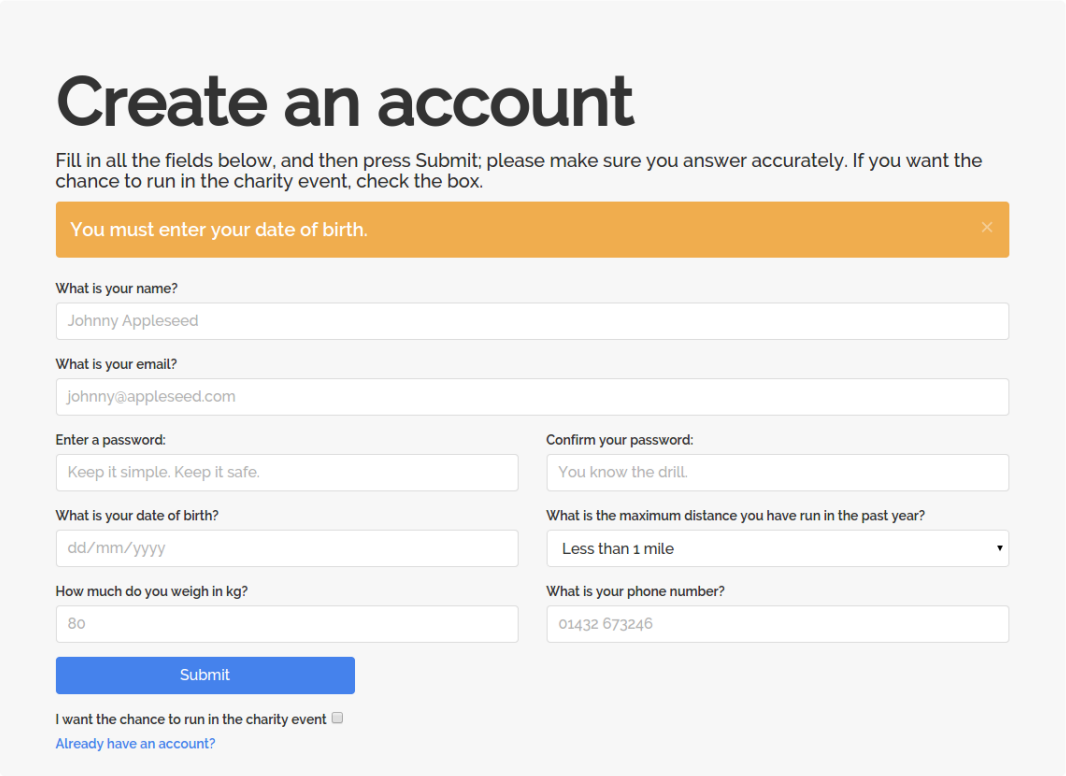
7.1 Main Layout

Every other page derives the constant elements, like the navigation bar and

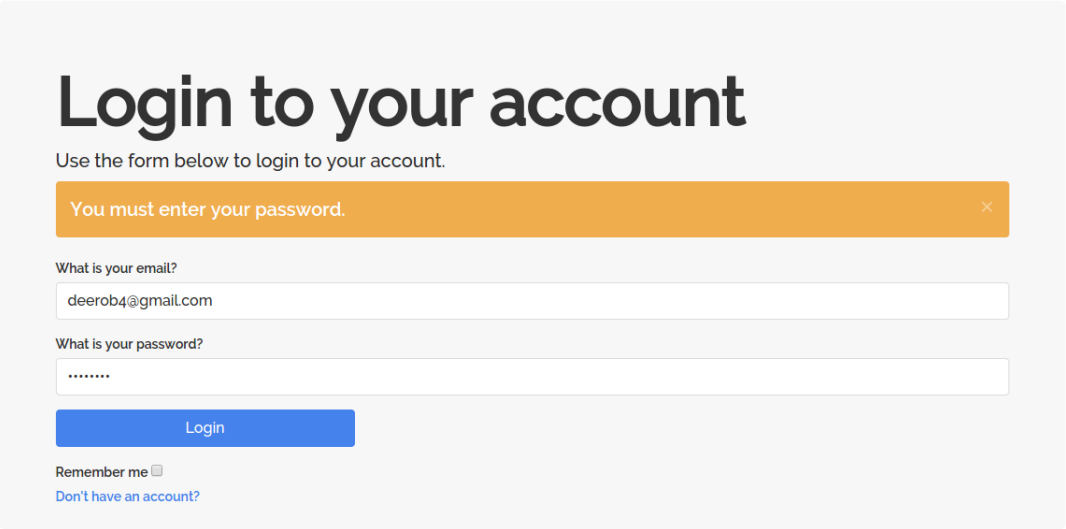


footer, from this template, to ensure visual consistency.

7.2 Register Page

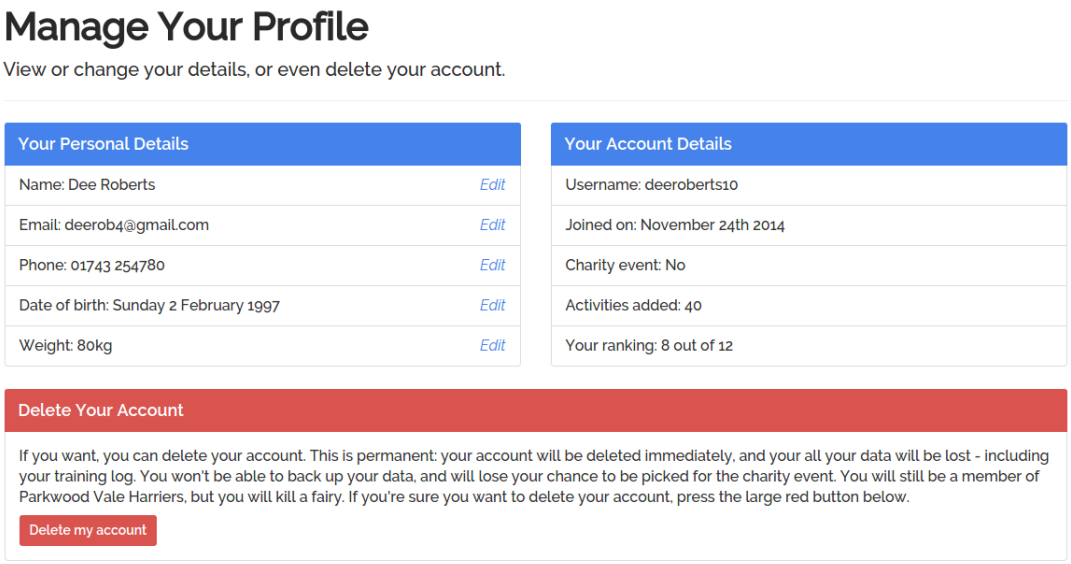


7.3 Login Page

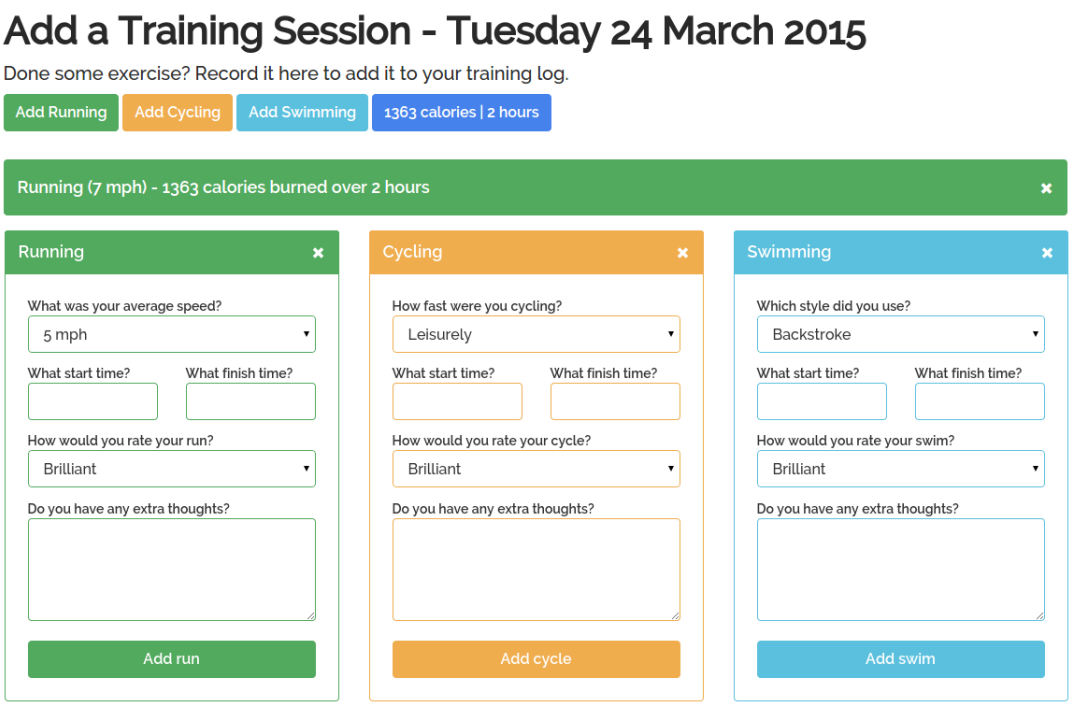


7.4 Profile Page

7.4.1 Main Profile Page



7.4.2 Delete Profile Section



7.5 User Performance Page

Training Performance

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

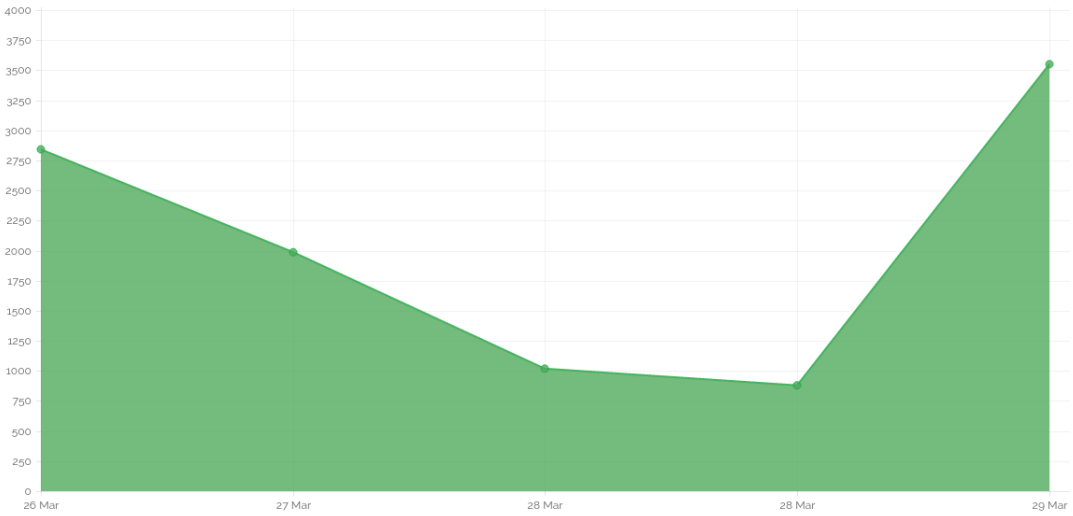
February | March

March Calorie Progress

March Hourly Progress

View Runs | View Cycles | View Swims

Running Data



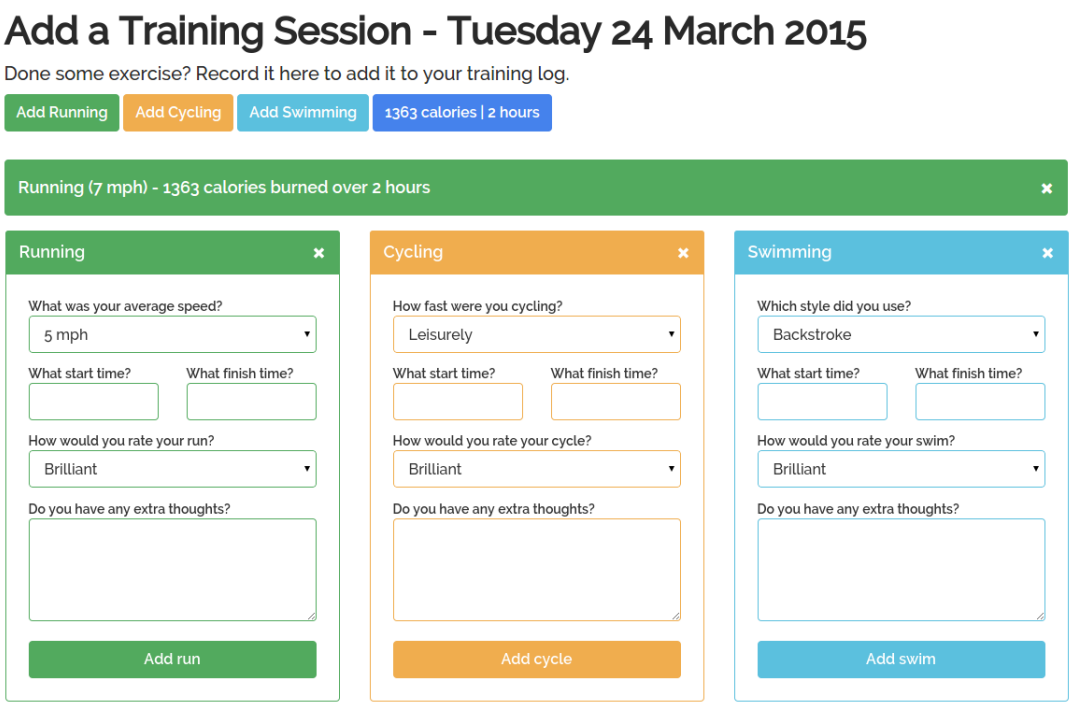
Tabular View

Show 10 entries Search:

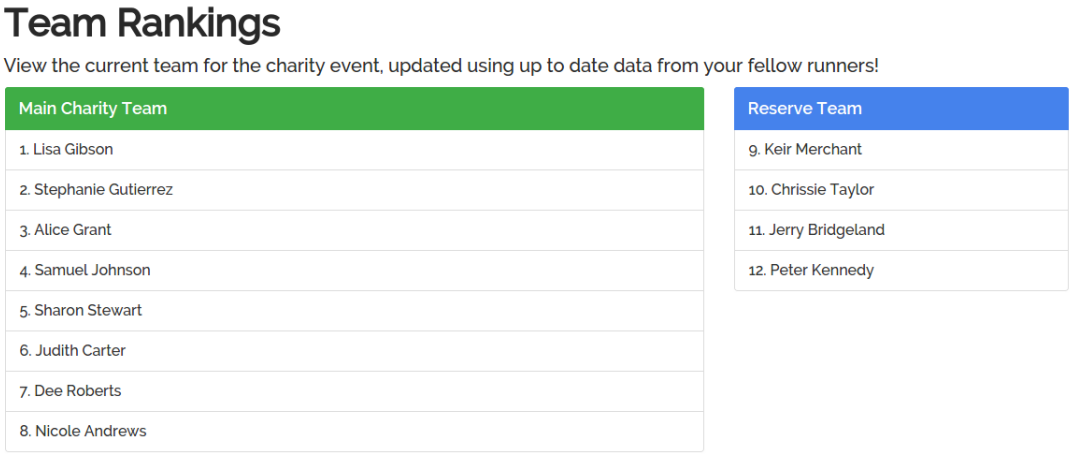
Date	Speed	Calories	Time	Hours	Rating
26 Mar 15	5 mph	2842 calories	6:00 AM - 2:00 PM	8 hours	Brilliant
27 Mar 15	9 mph	1987 calories	7:00 AM - 10:00 AM	3 hours	Okay
28 Mar 15	7 mph	1019 calories	7:00 AM - 9:00 AM	2 hours	About average
28 Mar 15	6 mph	880 calories	1:00 PM - 3:00 PM	2 hours	Okay
29 Mar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

Showing 1 to 5 of 5 entries Previous 1 Next

7.6 Add Activity Page



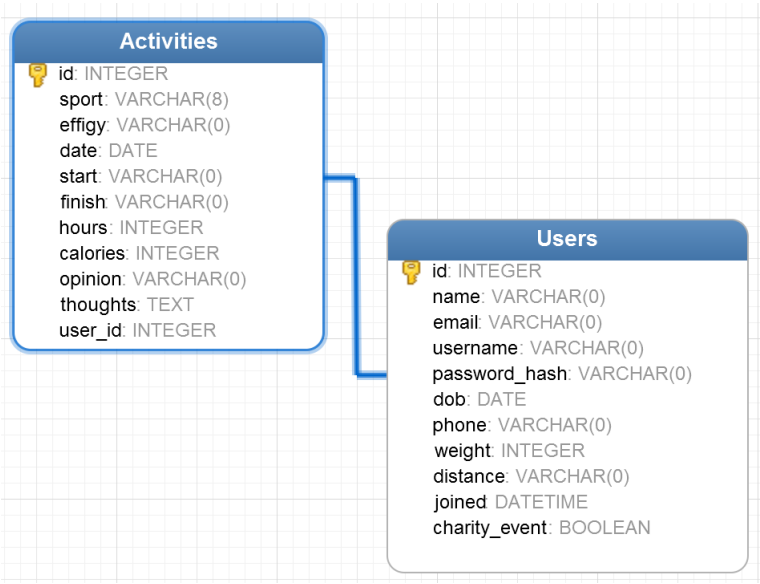
7.7 Rankings Page



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



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

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	☑	

8.2.2 Activities Table

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	☑	

9 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.

9.1 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.

9.1.1 layout.html

```
1  <!DOCTYPE html>
2  <html>
3  <head lang="en">
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-
           scale=1">
6      <link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/
           bootswatch/3.3.1/readable/bootstrap.min.css"/>
7      <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/
           animate.css/3.2.0/animate.min.css"/>
8      <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/
           bootstrap-datepicker/1.3.1/css/datepicker.min.css"/>
9      <link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/
           pickadate.js/3.5.3/compressed/themes/classic.css"/>
10     <link rel="stylesheet"
11           href="//cdnjs.cloudflare.com/ajax/libs/pickadate.js
                /3.5.3/compressed/themes/classic.time.css"/>
```

```
12      <link rel="stylesheet" href="//cdn.datatables.net/plug-ins/
            f2c75b7247b/integration/bootstrap/3/dataTables.bootstrap.
            css"/>
13      <link rel="stylesheet" href="{{ url_for('static', filename='css
            /main.css') }}"/>
14      <title>{% block title %} - Parkwood Vale Harriers{% endblock %}
            </title>
15  </head>
16  <body>
17  <nav class="navbar navbar-default" role="navigation">
18      <div class="container-fluid">
19          <div class="navbar-header">
20              <button class="navbar-toggle" data-toggle="collapse"
                    data-target="#main_nav" type="button">
21                  <span class="sr-only">Toggle Navigation</span>
22                  <span class="icon-bar"></span>
23                  <span class="icon-bar"></span>
24                  <span class="icon-bar"></span>
25              </button>
26              <a class="navbar-brand" href="{{ url_for('main.home')
                    }}">Parkwood Vale Harriers</a>
27          </div>
28          <div class="collapse navbar-collapse" id="main_nav">
29              {% if current_user.is_authenticated() %}
30                  <ul class="nav navbar-nav">
31                      <li><a href="{{ url_for('main.performance',
                            month='march') }}">My Performance</a></li>
32                      <li><a href="{{ url_for('main.add_training') }}
                            ">Add Training Session</a></li>
33                      <li><a href="{{ url_for('main.
                            compare_performance') }}">Compare
                            Performance</a></li>
34                      <li><a href="{{ url_for('main.rankings') }}">
                            Charity Team Rankings</a></li>
35                  </ul>
36                  <ul class="nav navbar-nav navbar-right">
37                      <li class="dropdown">
38                          <a class="dropdown-toggle" data-toggle="
                                dropdown" href="#">{{ current_user.name
                                 }}<span
39                                    class="caret"></span></a>
40                          <ul class="dropdown-menu" role="menu">
41                              <li><a href="{{ url_for('main.profiles
                                    ', username=current_user.username)
                                    }}">Your
42                                  Profile</a></li>
43                              <li><a href="#">Change Password</a></li
                                    >
44                              <li class="divider"></li>
45                              <li><a href="{{ url_for('auth.logout')
                                    }}">Logout</a></li>
46                          </ul>
47                      </li>
48                  </ul>
49              {% else %}
50                  <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

9.1.2 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 %}
```

```
16                      <div class="row">
17                          <div class="col-md-12">
18                              {% for category, message in messages %}
19                                  <div class="alert alert-{{ category
                                        }} alert-dismissable">
20                                      <button type="button" class="
                                            close" data-dismiss="alert"
                                            >
21                                          <span aria-hidden="true">&
                                                times;</span>
22                                          <span class="sr-only">Close
                                                </span>
23                                      </button>
24                                      <p>{{ message }}</p>
25                                  </div>
26                              {% endfor %}
27                          </div>
28                      </div>
29                  {% endif %}
30              {% endwith %}
31              <div class="form-group">
32                  {{ form.name.label }}
33                  {{ form.name(class='form-control
                        input_membership_name', placeholder='Johnny
                        Appleseed') }}
34              </div>
35              <div class="form-group">
36                  {{ form.email.label }}
37                  {{ form.email(class='form-control
                        input_membership_email', type='email',
                        placeholder='johnny@appleseed.com') }}
38              </div>
39              <div class="row">
40                  <div class="col-md-6">
41                      <div class="form-group">
42                          {{ form.password.label }}
43                          {{ form.password(class='form-control
                                input_membership_password', placeholder
                                ='Keep it simple. Keep it safe.') }}
44                      </div>
45                  </div>
46                  <div class="col-md-6">
47                      <div class="form-group">
48                          {{ form.confirm.label }}
49                          {{ form.confirm(class='form-control
                                input_membership_confirm', placeholder
                                ='You know the drill.') }}
50                      </div>
51                  </div>
52              </div>
53              <div class="row">
54                  <div class="col-md-6">
55                      <div class="form-group">
56                          {{ form.dob.label }}
57                          {{ 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

9.1.3 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

9.1.4 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">
```

```
38                  <div class="progress-bar progress-bar-success
                        running-calories-bar"
39                       style="width: {{ user_data.progress_data.
                             running.calories.percentage }}%;"
40                       role="progressbar" data-toggle="tooltip"
41                       title="{{ user_data.progress_data.running.
                             calories.value }} calories"></div>
42
43                  <div class="progress-bar progress-bar-warning
                        cycling-calories-bar"
44                       style="width: {{ user_data.progress_data.
                             cycling.calories.percentage }}%;"
45                       role="progressbar" data-toggle="tooltip"
46                       title="{{ user_data.progress_data.cycling.
                             calories.value }} calories"></div>
47
48                  <div class="progress-bar progress-bar-info swimming
                        -calories-bar"
49                       style="width: {{ user_data.progress_data.
                             swimming.calories.percentage }}%;"
50                       role="progressbar" data-toggle="tooltip"
51                       title="{{ user_data.progress_data.swimming.
                             calories.value }} calories"></div>
52
53              </div>
54          </div>
55
56          <div class="col-md-6">
57              <h3 class="performance-subtitle hour-subtitle"><span
                    class="month-text">{{ current_month|title }}</span>
                     Hourly
58                  Progress
59              </h3>
60
61              <div class="progress">
62                  <div class="progress-bar progress-bar-success
                        running-hours-bar"
63                       style="width: {{ user_data.progress_data.
                             running.hours.percentage }}%;"
64                       role="progressbar" data-toggle="tooltip"
65                       title="{{ user_data.progress_data.running.
                             hours.value }} hours"></div>
66
67                  <div class="progress-bar progress-bar-warning
                        cycling-hours-bar"
68                       style="width: {{ user_data.progress_data.
                             cycling.hours.percentage }}%;"
69                       role="progressbar" data-toggle="tooltip"
70                       title="{{ user_data.progress_data.cycling.
                             hours.value }} hours"></div>
71
72                  <div class="progress-bar progress-bar-info swimming
                        -hours-bar"
73                       style="width: {{ user_data.progress_data.
                             swimming.hours.percentage }}%;"
74                       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

9.1.5 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
                                                                    -
```

```
                href="#">Edit</a></span></li>
            <li class="email list-group-item">Email: {{
                current_user.email }} <span class="right"
                href="#">Edit</a></span>
            </li>
            <li class="phone list-group-item">Phone: {{
                current_user.phone }} <span class="right"
```

```
                        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 %}
```

```
67                          Yes{% else %}No{% endif %}</li>
68                      <li class="activities-added list-group-item">
                            Activities added: {{ activity_number }}</li
                            >
69                      <li class="joined list-group-item">Your ranking
                            : 0 out of {{ total_users }}</li>
70                  </ul>
71              </div>
72          </div>
73      </div>
74
75      <div class="panel panel-danger">
76          <div class="panel-heading">
77              <div class="panel-title">Delete Your Account</div>
78          </div>
79          <div class="panel-body">
80              If you want, you can delete your account. This is
                    permanent: your account will be deleted
81              immediately, and your all your data will be lost -
                    including your training log. You won't be able to
                    back up
82              your data, and will lose your chance to be picked for
                    the charity event. You will still be a member of
83              Parkwood Vale Harriers, but you will kill a fairy. If
                    you're sure you want to delete your account, press
                    the
84              large red button below.
85              <br/>
86
87              <div class="btn btn-danger btn-sm delete-account" data-
                    toggle="modal"
88                   data-target="#deleteAccountModal">Delete my
                         account
89              </div>
90          </div>
91      </div>
92
93      <div class="modal fade" id="changeNameModal">
94          <div class="modal-dialog">
95              <div class="modal-content">
96                  <div class="modal-header">
97                      <h4 class="modal-title">Change your name</h4>
98                  </div>
99                  <form method="POST" id="changeNameForm">
100                     <div class="modal-body">
101                         <label>
102                             Enter a new name:
103                             <input type="text" name="name"
                                    placeholder="{{ current_user.name
                                    }}" class="form-control" />
104                         </label>
105                     </div>
106                     <div class="modal-footer">
107                         <button type="button" class="btn btn-
                                default" data-dismiss="modal">Close</
                                button>
```

```
108                         <button type="submit" class="btn btn-
                                primary btn-modal">Change name</button>
109                     </div>
110                 </form>
111             </div>
112         </div>
113     </div>
114
115     <div class="modal fade" id="changeEmailModal">
116         <div class="modal-dialog">
117             <div class="modal-content">
118                 <div class="modal-header">
119                     <h4 class="modal-title">Change your email</h4>
120                 </div>
121                 <form method="POST" id="changeEmailForm">
122                     <div class="modal-body">
123                         <label>
124                             Enter a new email:
125                             <input type="text" name="email"
                                    placeholder="{{ current_user.email
                                    }}" class="form-control"/>
126                         </label>
127                     </div>
128                     <div class="modal-footer">
129                         <button type="button" class="btn btn-
                                default" data-dismiss="modal">Close</
                                button>
130                         <button type="submit" class="btn btn-
                                primary btn-modal">Change email</button
                                >
131                     </div>
132                 </form>
133             </div>
134         </div>
135     </div>
136
137     <div class="modal fade" id="changePhoneModal">
138         <div class="modal-dialog">
139             <div class="modal-content">
140                 <div class="modal-header">
141                     <h4 class="modal-title">Change your phone
                            number</h4>
142                 </div>
143                 <form method="POST" id="changePhoneForm">
144                     <div class="modal-body">
145                         <label>
146                             Enter a new phone number:
147                             <input type="text" name="phone"
                                    placeholder="{{ current_user.phone
                                    }}" class="form-control"/>
148                         </label>
149                     </div>
150                     <div class="modal-footer">
151                         <button type="button" class="btn btn-
                                default" data-dismiss="modal">Close</
                                button>
```

```
152                         <button type="submit" class="btn btn-
                                primary">Change phone number</button>
153                 </div>
154             </form>
155         </div>
156     </div>
157 </div>
158
159 <div class="modal fade" id="changeDobModal">
160     <div class="modal-dialog">
161         <div class="modal-content">
162             <div class="modal-header">
163                 <h4 class="modal-title">Change your date of
                        birth</h4>
164             </div>
165             <form method="POST" id="changeDobForm">
166                 <div class="modal-body">
167                     <label>
168                         Enter a new date of birth:
169                         <input type="text" name="dob"
                                placeholder="{{ current_user.dob }}
                                " class="form-control datepicker"/>
170                     </label>
171                 </div>
172                 <div class="modal-footer">
173                     <button type="button" class="btn btn-
                            default" data-dismiss="modal">Close</
                            button>
174                     <button type="submit" class="btn btn-
                            primary">Change date of birth</button>
175                 </div>
176             </form>
177         </div>
178     </div>
179 </div>
180
181 <div class="modal fade" id="changeWeightModal">
182     <div class="modal-dialog">
183         <div class="modal-content">
184             <div class="modal-header">
185                 <h4 class="modal-title">Change your weight:</h4
                        >
186             </div>
187             <form method="POST" id="changeWeightForm">
188                 <div class="modal-body">
189                     <label>
190                         Enter a new weight:
191                         <input type="number" name="weight" min=
                                "10" max="100" placeholder="{{
                                current_user.weight }}"
192                                    class="form-control"/>
193                     </label>
194                 </div>
195                 <div class="modal-footer">
196                     <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

9.1.6 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>
```

```
7       <h4>Done some exercise? Record it here to add it to your
            training log.</h4>
8
9       <button class="btn btn-running sport-button" id="running">Add
            Running</button>
10      <button class="btn btn-warning sport-button" id="cycling">Add
            Cycling</button>
11      <button class="btn btn-info sport-button" id="swimming">Add
            Swimming</button>
12
13      <button class="btn btn-primary"><span class="total-calories">{{
             total_calories }}</span> calories | <span
14              class="total-hours">{{ total_hours }}</span> hours
15      </button>
16      <br/>
17
18      <div class="row">
19          <ul class="activity-list">
20              <li class="row">
21                  {% for activity in activities %}
22                      <li class="saved-activity activity-block-{{
                            activity.sport|lower }} added col-md-12"
23                          id="{{ activity.id }}">
24                          <span class="sport">{{ activity.sport|title
                                }} ({{ activity.effigy|lower }})</span
                                >
25                          <span class="calories"> - {{ activity.
                                calories }} calories</span>
26                          <span class="hours">burned over {{ activity
                                .hours }} {% if activity.hours == 1 %}
                                hour{% else %}
27                              hours{% endif %}</span>
28                          <span class="glyphicon glyphicon-remove"></
                                span>
29                      </li>
30                  {% endfor %}
31              </li>
32          </ul>
33      </div>
34
35      {% if activities|length < 1 %}
36          <h4 class="no-activities">You haven't added any activities
                today! Use the buttons above to add one.</h4>
37      {% endif %}
38  {% endblock %}
```

Listing 6: Add Training Session Page

9.1.7 compare_performance.html

```
1   {% extends 'layout.html' %}
2
3   {% block title %}Compare Performance{{ super() }}{% endblock %}
4
5   {% block content %}
6
7       <h1>Compare Performance</h1>
```

```
8       <p>Want to see how you re doing compared to others? Use this
            page!</p>
9
10      <label for="user_list">Select user to compare aginst:</label>
11      <select name="user_list" id="user_list" class="form-control">
12          {% for user in user_list %}
13              <option value="{{ user[0] }}">{{ user[1] }}</option>
14          {% endfor %}
15      </select>
16
17      <h3>Graphical Comparison</h3>
18
19      <div class="graph_buttons">
20          <div class="btn-group">
21              <div class="btn btn-success" id="running_calories">
                    Running Calories</div>
22              <div class="btn btn-success" id="running_hours">Running
                    Hours</div>
23          </div>
24          <div class="btn-group">
25              <div class="btn btn-warning" id="cycling_calories">
                    Cycling Calories</div>
26              <div class="btn btn-warning" id="cycling_hours">Cycling
                    Hours</div>
27          </div>
28          <div class="btn-group">
29              <div class="btn btn-info" id="swimming_calories">
                    Swimming Calories</div>
30              <div class="btn btn-info" id="swimming_hours">Swimming
                    Hours</div>
31          </div>
32      </div>
33
34      <br>
35
36      <div class="row">
37          <div class="col-md-12"><canvas id="running_comparison"
                width="1140" height="600"></canvas></div>
38      </div>
39
40      <h3>Statistical Comparison</h3>
41
42      <div class="row">
43          <div class="col-md-6">
44              <div class="panel panel-success">
45                  <div class="panel-heading">
46                      <div class="panel-title">Your Performance</div>
47                  </div>
48                  <ul class="list-group panel-list">
49                      <li class="list-group-item">gosh</li>
50                      <li class="list-group-item">gosh</li>
51                      <li class="list-group-item">gosh</li>
52                      <li class="list-group-item">gosh</li>
53                      <li class="list-group-item">gosh</li>
54                  </ul>
55              </div>
56          </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

9.1.8 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

9.1.9 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

9.1.10 cycling_block.html

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

```
3           <div class="panel panel-warning activity-block" id="Cycling
                ">
4               <div class="panel-heading">
5                   <div class="panel-title">
6                       <span class="sport">Cycling</span>
7                       <span class="glyphicon glyphicon-remove"></span
                            >
8                   </div>
9               </div>
10              <div class="panel-body">
11                  <form>
12                      <div class="row">
13                          <div class="col-md-12">
14                              <div class="form-group">
15                                  <label>How fast were you cycling?
16                                      <select name="effigy" id="
                                            effigy" class="form-control
                                             activity-input cycling-
                                            input">
17                                          <option value="Leisurely">
                                                Leisurely</option>
18                                          <option value="Gently">
                                                Gently</option>
19                                          <option value="Moderately">
                                                Moderately</option>
20                                          <option value="Vigorously">
                                                Vigorously</option>
21                                          <option value="Very fast">
                                                Very Fast</option>
22                                          <option value="Racing">
                                                Racing</option>
23                                      </select>
24                                  </label>
25                              </div>
26                          </div>
27                      </div>
28                      <div class="row">
29                          <div class="col-md-6">
30                              <div class="form-group">
31                                  <label>What start time?
32                                      <input class='form-control
                                            activity-input time cycling
                                            -input' id="start">
33                                  </label>
34                              </div>
35                          </div>
36                          <div class="col-md-6">
37                              <div class="form-group">
38                                  <label>What finish time?
39                                      <input class='form-control
                                            activity-input time cycling
                                            -input' id="finish">
40                                  </label>
41                              </div>
42                          </div>
43                      </div>
44                      <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

9.1.11 swimming_block.html

```
<li class="activity">
    <div class="col-lg-4 col-md-6 col-sm-12 inner-activity">
        <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">
15                              <option value="Backstroke">
                                    Backstroke</option>
16                              <option value="Breaststroke">
                                    Breaststroke</option>
17                              <option value="Butterfly">Butterfly
                                    </option>
18                              <option value="Freestyle (slow)">
                                    Freestyle (slow)</option>
19                              <option value="Freestyle (fast)">
                                    Freestyle (fast)</option>
20                          </select>
21                      </label>
22                  </div>
23                  <div class="row">
24                      <div class="col-md-6">
25                          <div class="form-group">
26                              <label>What start time?
27                                  <input class='form-control
                                        activity-input time
                                        swimming-input' id="start">
28                              </label>
29                          </div>
30                      </div>
31                      <div class="col-md-6">
32                          <div class="form-group">
33                              <label>What finish time?
34                                  <input class='form-control
                                        activity-input time
                                        swimming-input' id="finish"
                                        >
35                              </label>
36                          </div>
37                      </div>
38                  </div>
39                  <div class="form-group">
40                      <label>How would you rate your swim?
41                          <select name="rating" id="rating" class
                                ="form-control activity-input
                                swimming-input">
42                              <option value="Brilliant">Brilliant
                                    </option>
43                              <option value="Pretty good">Pretty
                                    good</option>
44                              <option value="About average">About
                                     average</option>
45                              <option value="Okay">Okay</option>
46                              <option value="Awful">Awful</option
                                    >
47                          </select>
48                      </label>
49                  </div>
50                  <div class="form-group">
51                      <label>Do you have any extra thoughts?
52                          <textarea name="thoughts" id="thoughts"
53                                  class="activity-input form-
                                        control swimming-input"><
```

```
                                                /textarea>
54                          </label>
55                      </div>
56                      <div class="row">
57                          <div class="col-sm-12 col-md-12">
58                              <input type="button" class="btn btn-
                                    info activity-input add-activity"
                                    value="Add swim"/>
59                          </div>
60                      </div>
61                  </form>
62              </div>
63          </div>
64      </div>
65  </li>
```

Listing 11: Swimming Block

9.2 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.

9.2.1 main.js

```
1   $(document).ready(function () {
2
3       // Initialises the datepicker plugin for all inputs with a
            class of "datepicker"
4       $('.datepicker').datepicker({endDate: '-18y', startDate: '-75y'
            , format: 'yyyy-mm-dd'});
5
6       $('#running-activities, #cycling-activities, #swimming-
            activities').DataTable({
7           //"filter": false
8       });
9
10      function genericAnimation($element, animation, timeout) {
11          $element.addClass('animated ' + animation);
12          if (timeout === true) {
13              setTimeout(function () {
14                  $element.removeClass('animated ' + animation);
15              }, 1400);
16          }
17      }
18
19      // Animates the removal of the block
20      function animateRemove($activity) {
21          genericAnimation($activity, 'zoomOut', false);
22          setTimeout(function () {
23              $activity.remove();
24          }, 175);
25      }
26
```

```
27      // Called when the delete button on an activity block is
            pressed
28      $('.saved-activity .glyphicon').click(function () {
29          var $activity = $(this).closest('li'),
30              toRemove = {"activityId": $activity.attr('id')};
31          // If the activity block has been returned from the
                database
32          if ($activity.hasClass('added')) {
33              animateRemove($activity);
34              ajaxCall('/ajax/remove-activity', 'POST', 'json', '
                    application/json', JSON.stringify(toRemove), null);
35          } else {
36              animateRemove($activity);
37          }
38      });
39
40      // Sends a request to the server for the correct
41      $('.sport-button').click(function () {
42          var activity = $(this).attr('id');
43          ajaxCall('/ajax/sport-block', 'POST', 'text', 'text/plain',
                activity, updateActivities);
44      });
45
46      // Validates that times have been entered in the activity block
47      function validateActivity($activity) {
48          var $start = $activity.find('#start'),
49              $finish = $activity.find('#finish');
50          ($start, $finish).removeClass('animated zoomIn');
51          if ($start.val() === '') {
52              genericAnimation($start, 'shake', true);
53          }
54          if ($finish.val() === '') {
55              genericAnimation($finish, 'shake', true);
56          }
57          if ($start.val() !== '' && $finish.val() !== '') {
58              animateActivity($activity);
59          }
60      }
61
62      function updateActivities($activity) {
63          $activity = $($activity);
64          genericAnimation($('.no-activities'), 'fadeOutDown', 300);
65          $('.activity-list').append($activity);
66          genericAnimation($activity, 'zoomIn', false);
67          $('.time').pickatime({interval: 60, formatLabel: 'HH:i A',
                formatSubmit: 'HH:i A'});
68          // If the delete button is pressed, call the remove
                function
69          $('.activity-block .glyphicon').click(function () {
70              animateRemove($(this).closest('li'));
71          });
72          // If the add button is clicked, call the validate function
73          $('.add-activity').click(function () {
74              validateActivity($(this).closest('.panel'));
75          });
76      }
77
```

```
78      function animateActivity($activity) {
79          var sport = $activity.attr('id'),
80              containerWidth = $('.container').width();
81          $activity.find('label, input, select, textarea, .panel-body
                ').addClass('animated zoomOut');
82          setTimeout(function () {
83              $activity.find('.panel-heading').animate({
84                  width: containerWidth, height: 60,
                        borderBottomLeftRadius: 4,
85                  borderBottomRightRadius: 4, paddingTop: 17
86              }, 500);
87              $activity.find('.activity-block').css('margin-bottom',
                    '15px');
88              $activity.parent().removeClass('col-lg-4 col-md-6 col-
                    sm-12').addClass('col-lg-12 col-md-12 col-sm-12');
89              $activity.find('label, input, select, textarea, .form-
                    group, .panel-body').hide();
90          }, 200);
91          calculateCalories(sport, $activity);
92      }
93
94      // Calculates the number of hours between the start and finish
            times
95      function calculateHours($activity) {
96          var start = new Date('01/01/2000 ' + $activity.find('#start
                ').val()).getHours(),
97              stop = new Date('01/01/2000 ' + $activity.find('#finish
                    ').val()).getHours();
98          return stop - start;
99      }
100
101     function calculateCalories(sport, $activity) {
102         // Activity information needed for calculations are
                displayed here
103         var effigy = $activity.find('#effigy').val(),
104             rating = $activity.find('#rating').val(),
105             start = $activity.find('#start').val(),
106             finish = $activity.find('#finish').val(),
107             thoughts = $activity.find('#thoughts').val(),
108             hours = calculateHours($activity);
109         ajaxCall('/ajax/calculate-calories', 'POST', 'json', '
                application/json', JSON.stringify({
110             "sport": sport,
111             "effigy": effigy,
112             "hours": hours,
113             "thoughts": thoughts,
114             "start": start,
115             "finish": finish,
116             "rating": rating
117         }), addActivity, $activity);
118     }
119
120     function addActivity(data, $activity) {
121         var caloriesBurned = data.calories,
122             currentCalories = parseInt($('.total-calories').text())
                    ,
123             currentHours = parseInt($('.total-hours').text()),
```

```
124         // Builds a string to display in the animated activity
                block
125         // activityString = data.sport + ' (' + effigy.toLowerCase
                () + ') - ' + caloriesBurned + ' calories burned over '
                 + data.hours + ' hours',
126             activityString = data.sport,
127         // Constructs the final activity object in JSON, to send to
                 the server and save to the database
128             activityObject = {
129                 "sport": data.sport.toLowerCase(),
130                 "effigy": data.effigy,
131                 "calories": caloriesBurned,
132                 "start": data.start,
133                 "finish": data.finish,
134                 "hours": data.hours,
135                 "rating": data.rating,
136                 "thoughts": data.thoughts
137             };
138
139         $('.total-hours').text(currentHours + data.hours);
140         $('.total-calories').text(currentCalories + caloriesBurned)
                ;
141
142         $activity.find('.sport').text(activityString);
143
144         ajaxCall('/ajax/send-activity', 'POST', 'json', '
                application/json', JSON.stringify(activityObject), null
                )
145     }
146
147     // A generic function that sends a request to the server and
            calls a function with the returned data
148     function ajaxCall(url, requestType, dataType, contentType, data
            , callbackFunction, activity) {
149         $.ajax({
150             url: url,
151             type: requestType,
152             dataType: dataType,
153             contentType: contentType,
154             data: data,
155             success: function (data) {
156                 if (typeof activity != 'undefined') {
157                     callbackFunction(data, activity);
158                 } else {
159                     callbackFunction(data);
160                 }
161             }
162         })
163     }
164
165     $('[data-toggle="tooltip"]').tooltip();
166     Chart.defaults.global.scaleFontFamily = "'Raleway', 'Helvetica
            ', 'Arial', sans-serif";
167
168 });
```

Listing 12: Main JavaScript Functions

9.2.2 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});
```

```
49      }
50
51      $('.activity-change').click(function () {
52          var sport = $(this).attr('id');
53          if ($('.' + sport + '-data').hasClass('active') == false) {
54              $('.active').addClass('animated bounceOutRight');
55              setTimeout(function () {
56                  $('.active').css('display', 'none').removeClass('
                        animated bounceOutRight active');
57                  $('.' + sport + '-data').css('display', 'block').
                        addClass('animated bounceInLeft active');
58              }, 600)
59          }
60      })
61
62      $('.trainingHeading').click(function() {
63          $('.runningChart').update();
64      })
65
66  });
```

Listing 13: User Charts

9.3 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.

```
1   @font-face {
2       font-family: 'ralewayitalic';
3       src: url('../fonts/raleway-regular-italic-webfont.eot');
4       src: url('../fonts/raleway-regular-italic-webfont.eot?#iefix')
            format('embedded-opentype'),
5       url('../fonts/raleway-regular-italic-webfont.woff2') format('
            woff2'),
6       url('../fonts/raleway-regular-italic-webfont.woff') format('
            woff'),
7       url('../fonts/raleway-regular-italic-webfont.ttf') format('
            truetype'),
8       url('../fonts/raleway-regular-italic-webfont.svg#ralewayitalic'
            ) format('svg');
9       font-weight: normal;
10      font-style: normal;
11  }
12  @font-face {
13      font-family: 'ralewaymedium';
14      src: url('../fonts/raleway-medium-webfont.eot');
15      src: url('../fonts/raleway-medium-webfont.eot?#iefix') format('
            embedded-opentype'),
16      url('../fonts/raleway-medium-webfont.woff2') format('woff2'),
17      url('../fonts/raleway-medium-webfont.woff') format('woff'),
18      url('../fonts/raleway-medium-webfont.ttf') format('truetype'),
19      url('../fonts/raleway-medium-webfont.svg#ralewaymedium') format
            ('svg');
20      font-weight: normal;
21      font-style: normal;
22
23  }
```

```
@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%;
}
/*-------------------------------------------------------
|        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 {
```

```
77      font-family: ralewaysemibold, sans-serif, "Helvetica Neue",
            Helvetica, Arial, sans-serif;
78      font-weight: 100;
79  }
80  label {
81      font-size: 14px;
82      width: 100%;
83  }
84  .activity-block label {
85      width: 100%;
86  }
87  .timepicker {
88      background-color: #ffffff !important;
89      cursor: auto !important;
90  }
91  .details p {
92      margin-bottom: 3px;
93      font-size: 20px;
94      font-weight: 800;
95  }
96  .jumbotron .alert p {
97      font-size: 20px;
98  }
99  /*-----------------------------------------------------
100 |             Begin general input styles              |
101 -----------------------------------------------------*/
102 input:not(.input_membership_charity):not(.add-activity):not(.btn-
        modal), select, textarea {
103     width: 100%;
104     border-radius: 4px;
105     box-shadow: none !important;
106     -webkit-box-shadow: none !important;
107     font-family: ralewaymedium, "Helvetica Neue", Helvetica, Arial,
            sans-serif;
108 }
109 .datepicker {
110     padding-left: 12px !important;
111 }
112 /*-----------------------------------------------------
113 |             Begin general button styles             |
114 -----------------------------------------------------*/
115
116 .btn {
117     font-family: ralewaysemibold, "Helvetica Neue", Helvetica,
            Arial, sans-serif;
118 }
119 .btn-running {
120     background-color: #52aa5e;
121     color: #ffffff;
122 }
123 .btn-running:hover {
124     background-color: #10aa3b;
125     color: #ffffff;
126 }
127 .btn-running:focus {
128     color: #ffffff;
129 }
```

```
/*-------------------------------------------------------
|             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;
```

```
244     padding: 0;
245 }
246 h3 {
247     font-family: ralewaysemibold, sans-serif !important;
248 }
249 .tooltip {
250     font-family: ralewaysemibold, sans-serif;
251 }
252 .performance-subtitle, .calorie-subtitle, .hour-subtitle {
253     -webkit-animation-duration: 0.575s;
254 }
255 .month-buttons {
256     list-style-type: none;
257     display: inline;
258 }
259 .month-buttons li {
260     display: inline;
261 }
262 .nav-pills, .no-footer {
263     font-family: 'ralewaymedium', sans-serif;
264 }
265 input[type=search] {
266     width: 90% !important;
267 }
268 .activity-view {
269     margin-top: 25px;
270 }
271 .running-data, .cycling-data, .swimming-data {
272     margin-bottom: 100px;
273 }
274 .cycling-data, .swimming-data {
275     display: none;
276 }
277 table {
278     border-right: 4px;
279 }
280 /*-------------------------------------------------------
281 |                  Begin comparison styles              |
282 -------------------------------------------------------*/
283 .graph_buttons {
284     padding: 0;
285 }
286 ul.month_buttons {
287     padding: 0 !important;
288 }
```

Listing 14: main.css

9.4 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.

9.4.1 __init__.py

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

```
1  from flask import Flask
2  from flask.ext.login import LoginManager
3
4  from app.models import db, User
5
6
7  def create_app():
8      """Generates an instance of the app.
9
10     This function contains all the config values
11     for the different parts of the app; it returns
12     a variable 'app' that contains all these values
13     for use throughout the rest of the application.
14     """
15     app = Flask(__name__)
16
17     # Sets the application into debug mode
18     app.debug = True
19
20     # Sets configuration variables used application-wise
21     app.config['SECRET_KEY'] = 'vYqTMY88zsuXSG7R4xYdPxYk'
22     app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///../database.
           db'
23
24     # Configures SQLAlchemy
25     db.init_app(app)
26
27     # Configures the login manager
28     login_manager = LoginManager()
29     login_manager.init_app(app)
30     login_manager.login_view = 'auth.login'  # Sets the login view.
31     login_manager.login_message_category = 'warning'
32
33     # Loads the current user by running a query on the id
34     @login_manager.user_loader
35     def load_user(id):
36         return User.query.get(int(id))
37
38     # Configures application blueprints
39     from app.controllers.main import main
40     app.register_blueprint(main)
41
42     from app.controllers.auth import auth
43     app.register_blueprint(auth)
44
45     from app.controllers.ajax import ajax
46     app.register_blueprint(ajax)
47
48     return app
49
50 if __name__ == '__main__':
51     app = create_app()
```

```
52      app.run(debug=True)
```

Listing 15: __init__.py

9.4.2 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.

```
1  from flask.ext.wtf import Form
2  from wtforms import StringField, PasswordField, DateField,
       BooleanField, SubmitField, SelectField, IntegerField
3  from wtforms.validators import DataRequired, Email, Length, EqualTo
       , Regexp, ValidationError, NumberRange
4
5  from app.models import User
6  from app.helpers import calculate_age
7
8
9  class MemberForm(Form):
10     """Contains the fields and validators for the new member form.
           """
11
12     name = StringField("What is your name?", validators=[
           DataRequired('You must enter your name.'),
13                                                     Regexp(r'
                                                           ^[A-Za
                                                           -z\-"
                                                           "]*$',
14                                                            message
                                                                  =
                                                                  '
                                                                  Your

                                                                  name

                                                                  may

                                                                  only

                                                                  contain

                                                                  letters
                                                                  .
                                                                  '
                                                                  )
                                                                  ])
15     dob = DateField("What is your date of birth?", validators=[
           DataRequired('You must enter your date of birth.')])
16     email = StringField("What is your email?",
17                         validators=[DataRequired('You must enter
                               your email.'), Email('You must enter a
                               valid email.')])
18     password = PasswordField("Enter a password:", validators=[
           DataRequired('You must enter a password.'),
```

```
19                                                          Length
                                                                (8,

                                                                20,
20                                                                  '
                                                                    Your

                                                                    password

                                                                    must

                                                                    be

                                                                    8

                                                                    -

                                                                    20

                                                                    characters
                                                                    .
                                                                    '
                                                                    )
                                                                    ])
21      confirm = PasswordField("Confirm your password:", validators=[
            DataRequired('You must confirm your password.'),
22                                                          EqualTo
                                                                (
                                                                '
                                                                password
                                                                '
                                                                ,

                                                                '
                                                                Your

                                                                passwords

                                                                must

                                                                match
                                                                .
                                                                '
                                                                )
                                                                ])
23      charity_event = BooleanField("I want the chance to run in the
            charity event")
24      distance = SelectField('What is the maximum distance you have
            run in the past year?',
25                             choices=[('l1', 'Less than 1 mile'), ('
                                   1-5', '1 - 5 miles'), ('6-10', '6 -
                                   10 miles'),
26                                       ('11-15', '11 - 15 miles'), ('
                                             16-20', '16 - 20 miles'),
```

```
27                                      ('g20', 'More than 20 miles')])
28      weight = IntegerField('How much do you weigh in kg?',
            validators=[DataRequired('You must enter your weight.'),
29                                                                  NumberRange
                                                                        (10,

                                                                        100,

30                                                                              '
                                                                                    Your

                                                                                    weight

                                                                                    must

                                                                                    be

                                                                                    between

                                                                                    10
                                                                                    kg

                                                                                    -

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

33                                                                      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
                                                                        *
                                                                        $
                                                                        '
```

```
                                                                    ,
34                                                                  message
                                                                        =
                                                                        '
                                                                        You
                                                                        must
                                                                        enter
                                                                        a
                                                                        valid
                                                                        UK
                                                                        phone
                                                                        number
                                                                        .
                                                                        '
                                                                        )
                                                                        ])
35      submit = SubmitField('Submit')
36
37      def validate_distance(self, field):
38          """Ensures the user has not ticked the charity event and is
                a poor runner."""
39          charity_event = self.charity_event
40          if field.data == 'l1' and charity_event.data is True:
41              raise ValidationError('You must be physically fit to
                    run in the charity event.')
42
43      def validate_dob(self, field):
44          """Ensures the user is between 18 - 75 years old."""
45          age = calculate_age(field.data)
46          if not 18 <= age <= 75:
47              raise ValidationError('You must be 18 - 75 years old to
                    join.')
48
49      def validate_email(self, field):
50          """Ensures the email address is unique"""
51          if User.query.filter_by(email=field.data).first():
52              raise ValidationError('That email address has already
                    been registered.')
53
54
55  class LoginForm(Form):
56      """Contains the fields and validators for the login form."""
57      email = StringField('What is your email?',
58                          validators=[DataRequired('You must enter
                                your email.'), Email('You must enter a
                                valid email.')])
59      password = PasswordField('What is your password?', validators=[
            DataRequired('You must enter your password.')])
```

```
60      remember = BooleanField('Remember me')
61      login = SubmitField('Login')
```

Listing 16: forms.py

9.4.3 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.

```
1  from flask.ext.sqlalchemy import SQLAlchemy
2  from werkzeug.security import generate_password_hash,
       check_password_hash
3  from flask.ext.login import UserMixin
4
5  db = SQLAlchemy()
6
7
8  class User(UserMixin, db.Model):
9      """Defines the user table and the fields.
10
11     Each variable represents an individual field
12     for the database, pertaining to the data collected
13     in app.forms.MemberForm. The data type is also declared.
14     All fields are of variable length. There is a one-to-many
15     relationship between users and activities.
16     """
17     __tablename__ = 'Users'
18     id = db.Column(db.Integer, primary_key=True)
19     name = db.Column(db.String)
20     email = db.Column(db.String)
21     username = db.Column(db.String)
22     password_hash = db.Column(db.String)
23     dob = db.Column(db.Date)
24     phone = db.Column(db.String)
25     weight = db.Column(db.Integer)
26     distance = db.Column(db.String)
27     joined = db.Column(db.DateTime)
28     charity_event = db.Column(db.Boolean)
29
30     activities = db.RelationshipProperty('Activity', backref='user'
           , lazy='dynamic')
31
32     # Initialises the class to allow it to be referenced in helper
           functions.
33     def __init__(self, name, username, email, dob, password,
           distance, charity_event, weight, phone, joined):
34         self.name = name
35         self.username = username
36         self.email = email
37         self.password = password
38         self.dob = dob
39         self.distance = distance
40         self.charity_event = charity_event
41         self.phone = phone
42         self.weight = weight
43
```

```
44      # Ensures the password is accessible.
45      @property
46      def password(self):
47          raise AttributeError('Password is not a readable attribute.
                ')
48
49      # Encrypts the password and assigns it to the class variable.
50      @password.setter
51      def password(self, password):
52          self.password_hash = generate_password_hash(password)
53
54      # Checks the entered password against the decrypted password
            hash.
55      def check_password(self, value):
56          return check_password_hash(self.password_hash, value)
57
58      # Returns the id of the current user.
59      def get_id(self):
60          return self.id
61
62      # Obligatory identification function.
63      def __repr__(self):
64          return '<User: %r>' % self.id
65
66
67  class Activity(db.Model):
68      """Defines the activities table and the fields.
69
70      Each variable represents an individual field
71      for the database, pertaining to the data collected
72      in app.static.js.main. The data type is also declared.
73      A foreign key is established between the user table,
74      with users.id acting as the key; this creates a
75      one-to-one link between the two tables (one user can
76      have multiple activities.
77      """
78      __tablename__ = 'Activities'
79      id = db.Column(db.Integer, primary_key=True)
80      sport = db.Column(db.String(8))
81      effigy = db.Column(db.String)
82      date = db.Column(db.Date)
83      start = db.Column(db.String)
84      finish = db.Column(db.String)
85      hours = db.Column(db.Integer)
86      calories = db.Column(db.Integer)
87      opinion = db.Column(db.String)
88      thoughts = db.Column(db.Text)
89
90      user_id = db.Column(db.Integer, db.ForeignKey('Users.id'))
91
92      # Initialises the class to allow it to be referenced in helper
            functions.
93      def __init__(self, sport, effigy, date, start, finish, calories
            , opinion, thoughts, hours, user_id):
94          self.sport = sport
95          self.effigy = effigy
96          self.date = date
```

```
97          self.start = start
98          self.finish = finish
99          self.hours = hours
100         self.calories = calories
101         self.opinion = opinion
102         self.thoughts = thoughts
103         self.user_id = user_id
104
105     # Obligatory identification function.
106     def __repr__(self):
107         return '<Activity: %r (%r)>' % (self.id, self.sport)
```

Listing 17: models.py

9.4.4 helpers.py

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

```
1  from flask import flash, redirect, url_for
2  from flask.ext.login import current_user
3
4  from app.models import db, Activity
5
6  from datetime import date
7
8
9  def update_user(user, element, redirect_user=True):
10     """Adds the updated user to the db and reloads the page."""
11     db.session.add(user)
12     db.session.commit()
13     flash('Your %s has been successfully changed!' % element, '
           success')
14     if redirect_user:
15         return redirect(url_for('main.profiles', username=user.
               username))
16
17
18 def validation_error(message):
19     """Displays an appropriate error message and reloads the page.
           """
20     flash(message, 'warning')
21     return redirect(url_for('main.profiles', username=current_user.
           username))
22
23
24 def calculate_age(born):
25     """Calculates the age of the user"""
26     today = date.today()
27     return today.year - born.year - ((today.month, today.day) < (
           born.month, born.day))
28
29
30 def remove_sport(activity_id):
31     """Removes the activity from the database"""
32     Activity.query.filter_by(id=activity_id).delete()
33     db.session.commit()
```

```
34      print('Activity %s deleted' % id)
35      return 'Activity %s deleted' % activity_id
```

Listing 18: helpers.py

9.4.5 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.

```
1  from calendar import month_name
2  from flask.ext.login import current_user
3  from app.models import db, Activity, User
4
5
6  def performance_data(month):
7      """Creates a dictionary object with training data
8
9      This function is used throughout the system to create
10     a collection of a particular user's training activities.
11     It performs several queries to the db and uses a number
12     of loops and list comprehensions in order to
13     """
14
15     # Creates a list of months - January, February, etc.
16     months = [month_name[x].lower() for x in range(1, 13)]
17
18     # Queries the db for all of the user's activities.
19     all_activities = Activity.query.filter_by(user_id=current_user.
           get_id()).all()
20
21     # Queries the db for all of the user's different activities.
22     all_runs = Activity.query.filter_by(user_id=current_user.get_id
           (), sport='running').all()
23     all_cycles = Activity.query.filter_by(user_id=current_user.
           get_id(), sport='cycling').all()
24     all_swims = Activity.query.filter_by(user_id=current_user.
           get_id(), sport='swimming').all()
25
26     # Creates a dict with month names and values - Jan: 1 etc.
27     month_map = dict(zip(months, range(1, 13)))
28
29     # Sets the total monthly calorie and hourly goal.
30     calorie_goal = 40000
31     hour_goal = 100
32
33     # [0] contains the calories burned; [1] contains the hours.
34     total_run_data = [0, 0]
35     total_cycle_data = [0, 0]
36     total_swim_data = [0, 0]
37
38     # Generates a list containing the data for every running
           activity using the above queries.
39     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
                },
```

```
87                  'hours': {
88                      'value': total_cycle_data[1],
89                      'percentage': total_cycle_data[1] / hour_goal *
                            100
90                  }
91              },
92              'swimming': {
93                  'calories': {
94                      'value': total_swim_data[0],
95                      'percentage': total_swim_data[0] / calorie_goal
                            * 100
96                  },
97                  'hours': {
98                      'value': total_swim_data[1],
99                      'percentage': total_swim_data[1] / hour_goal *
                            100
100                 }
101             }
102         },
103         'sport_data': {
104             'running': run_list,
105             'swimming': swim_list,
106             'cycling': cycle_list
107         },
108         'month': month.title()
109     }
110
111     return user_data
```

Listing 19: performance_data.py

9.4.6 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.

```
1  from datetime import datetime
2
3  from flask import Blueprint, render_template, flash, redirect,
       url_for
4  from flask.ext.login import current_user, login_user, logout_user
5  from random import randint
6
7  from app.forms import MemberForm, LoginForm
8  from app.models import db, User
9
10
11 auth = Blueprint('auth', __name__)
12
13
14 @auth.route('/register', methods=['GET', 'POST'])
15 def register():
16     """Renders the register page and saves new users to the
           database"""
17     # Makes sure logged in users cannot access the register page
```

```
18      if not current_user.is_authenticated():
19          form = MemberForm()
20          # If the submit button is pressed
21          if form.validate_on_submit():
22              # Generates a username for the user composed of their
                    real name and a random number
23              username = form.name.data.lower().replace(' ', '') +
                    str(randint(1, 10))
24              # Creates a User object with the data they typed in
25              user = User(name=form.name.data, username=username,
                    email=form.email.data, password=form.password.data,
26                          dob=form.dob.data, distance=form.distance.
                                data, charity_event=form.charity_event.
                                data,
27                          phone=form.phone.data, weight=form.weight.
                                data, joined=datetime.now())
28              # Saves the user to the database
29              db.session.add(user)
30              db.session.commit()
31              print('%s has been registered.' % user.name)
32              # Returns the user to the login page with a message
33              flash('You can now login!', 'success')
34              return redirect(url_for('auth.login'))
35          # If there were validation errors, re-render the view and
                show them
36          for error in form.errors.items():
37              flash(error[1][0], 'warning')
38          return render_template('auth/register.html', form=form)
39      return redirect(url_for('main.home'))
40
41
42  @auth.route('/login', methods=['GET', 'POST'])
43  def login():
44      """Renders the login page and logs in the user"""
45      if not current_user.is_authenticated():
46          form = LoginForm()
47          if form.validate_on_submit():
48              # Query that returns the first user with the entered
                    email address.
49              user = User.query.filter_by(email=form.email.data).
                    first()
50              # Checks that a user was returned and that the password
                     is correct.
51              if user is not None and user.check_password(form.
                    password.data):
52                  # If so, log them in and redirect them to the home
                        page
53                  login_user(user, form.remember.data)
54                  return redirect(url_for('main.home'))
55              flash('Invalid email address or password.', 'warning')
56          # If there were validation errors, re-render the view and
                show them
57          for error in form.errors.items():
58              flash(error[1][0], 'warning')
59          return render_template('auth/login.html', form=form)
60      return redirect(url_for('main.home'))
61
```

```
62
63 @auth.route('/logout')
64 def logout():
65     """Logs the user out of the system"""
66     logout_user()
67     return redirect(url_for('main.home'))
```

Listing 20: auth.py

9.4.7 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.

```
1  from datetime import datetime
2  from math import ceil
3  from calendar import month_name
4
5  from flask import Blueprint, render_template, request, jsonify
6  from flask.ext.login import current_user
7
8  from app.models import Activity, db
9  from app.performance_data import performance_data
10 from app.helpers import remove_sport
11
12
13 ajax = Blueprint('ajax', __name__)
14
15
16 # Defines the route for displaying the activity blocks
17 @ajax.route('/ajax/sport-block', methods=['POST'])
18 def sport_block():
19     sport = request.get_data().decode("utf-8")
20     if sport == 'running':
21         return render_template('training/running_block.html')
22     elif sport == 'cycling':
23         return render_template('training/cycling_block.html')
24     elif sport == 'swimming':
25         return render_template('training/swimming_block.html')
26     else:
27         return '%s was passed as a sport - no template is available
               for this.' % sport, 400
28
29
30 # Defines the route for uploading activity block data
31 @ajax.route('/ajax/send-activity', methods=['POST'])
32 def send_activity():
33     sport = request.json['sport']
34     effigy = request.json['effigy']
35     calories = request.json['calories']
36     hours = request.json['hours']
37     start = request.json['start']
38     finish = request.json['finish']
39     opinion = request.json['rating']
40     thoughts = request.json['thoughts']
```

```
41
42      activity = Activity(sport=sport, effigy=effigy, calories=
            calories, hours=hours, start=start,
43                          finish=finish, opinion=opinion, thoughts=
                                thoughts,
44                          user_id=current_user.get_id(), date=
                                datetime.now().date())
45
46      db.session.add(activity)
47      db.session.commit()
48      print('Successfully saved Activity %s (%s) to the database.' %
            (activity.id, activity.sport))
49      return 'success', 200
50
51
52  @ajax.route('/ajax/remove-activity', methods=['POST'])
53  def remove_activity():
54      activity_id = request.json['activityId']
55      return remove_sport(activity_id)
56
57
58  @ajax.route('/ajax/calculate-calories', methods=['POST'])
59  def calculate_calories():
60      """Calculates the number of calories burned in a session
61
62      The base values were arrived at by dividing each value provided
                by the
63      board by 80. The formula takes the correct base value, and
            multiplies it
64      by the weight of the user. This is then multiplied by
65      the number of hours. This value is modified based on how well
            the activity went -
66      each of the five options is assigned a value from -10 to 10;
            this is then
67      added to the total value to arrive at the final number of
            calories.
68      """
69      base_calories = {
70          'swimming': {'Backstroke': 5.1625, 'Breaststroke': 7.375, '
                Butterfly': 8.1125, 'Freestyle (slow)': 5.1625,
71                       'Freestyle (fast)': 7.375},
72          '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},
73          'cycling': {'Leisurely': 2.95, 'Gently': 4.425, 'Moderately
                ': 5.9, 'Vigorously': 6.125, 'Very fast': 8.85,
74                      'Racing': 11.8},
75          'modifiers': {'Brilliant': 10, 'Pretty good': 5, 'About
                average': 0, 'Okay': -5, 'Awful': -10}
76      }
77      sport = request.json['sport'].lower()
78      effigy = request.json['effigy']
79      hours = request.json['hours']
80      start = request.json['start']
81      finish = request.json['finish']
82      thoughts = request.json['thoughts']
83      rating = request.json['rating']
84
```

```
85      base_value = base_calories[sport][effigy]
86      calories = (base_value * current_user.weight) * hours
87      modifier = base_calories['modifiers'][rating]
88      calories += modifier
89
90      activity_data = {'calories': str(ceil(calories)), 'sport':
            sport, 'hours': hours, 'effigy': effigy,
91                       'start': start, 'finish': finish, 'rating':
                             rating, 'thoughts': thoughts}
92
93      return jsonify(activity_data)
94
95
96  @ajax.route('/ajax/user-charts', methods=['POST'])
97  def user_charts():
98      print(request.get_data().decode("utf-8").lower())
99      month_map = dict(zip([month_name[x].lower() for x in range(1,
            13)], range(1, 13)))
100     user_month = month_map[request.json['month'].lower()]
101
102     runs = Activity.query.filter_by(user_id=current_user.get_id(),
            sport='running').all()
103
104     cycles = Activity.query.filter_by(user_id=current_user.get_id()
            , sport='cycling').all()
105
106     swims = Activity.query.filter_by(user_id=current_user.get_id(),
             sport='swimming').all()
107
108     activity_data = {
109         'running': {'calories': [run.calories for run in runs if
                run.date.month == user_month],
110                     'dates': [run.date.strftime('%d %b') for run in
                          runs if run.date.month == user_month]},
111         'cycling': {'calories': [cycle.calories for cycle in cycles
                 if cycle.date.month == user_month],
112                     'dates': [cycle.date.strftime('%d %b') for
                         cycle in cycles if cycle.date.month ==
                         user_month]},
113         'swimming': {'calories': [swim.calories for swim in swims
                if swim.date.month == user_month],
114                      'dates': [swim.date.strftime('%d %b') for swim
                           in swims if swim.date.month == user_month
                          ]}
115     }
116     return jsonify(activities=activity_data)
117
118
119 @ajax.route('/ajax/performance', methods=['POST'])
120 def ajax_performance():
121     month = request.get_data().decode("utf-8").lower()
122     user_data = performance_data(month)
123     return jsonify(user_data=user_data)
124
125
126 @ajax.route('/ajax/comparison-graph', methods=['POST'])
127 def comparison_graphs():
```

```
128     graph_type = request.json['graphType']
129     comparison_user = int(request.json['comparisonUser'])
130 
131     user_runs = Activity.query.filter_by(user_id=current_user.
            get_id(), sport='running').all()
132     comparison_runs = Activity.query.filter_by(user_id=
            comparison_user, sport='running').all()
133     run_months = []
134     for run in user_runs:
135         if run.date.strftime('%B') not in run_months:
136             run_months.append(run.date.strftime('%B'))
137 
138     user_cycles = Activity.query.filter_by(user_id=current_user.
            get_id(), sport='cycling').all()
139     comparison_cycles = Activity.query.filter_by(user_id=
            comparison_user, sport='cycling').all()
140     cycle_months = []
141     for cycle in user_cycles:
142         if cycle.date.strftime('%B') not in cycle_months:
143             cycle_months.append(run.date.strftime('%B'))
144 
145     user_swims = Activity.query.filter_by(user_id=current_user.
            get_id(), sport='swimming').all()
146     comparison_swims = Activity.query.filter_by(user_id=
            comparison_user, sport='swimming').all()
147     swim_months = []
148     for swim in user_swims:
149         if swim.date.strftime('%B') not in swim_months:
150             swim_months.append(swim.date.strftime('%B'))
151 
152     if graph_type == 'running_calories':
153         graph_data = {'current_user': [run.calories for run in
                user_runs],
154                       'comparison_user': [run.calories for run in
                           comparison_runs], 'months': run_months}
155 
156     elif graph_type == 'running_hours':
157         graph_data = {'current_user': [run.hours for run in
                user_runs],
158                       'comparison_user': [run.hours for run in
                           comparison_runs], 'months': run_months}
159 
160     elif graph_type == 'cycling_calories':
161         graph_data = {'current_user': [cycle.calories for cycle in
                user_cycles],
162                       'comparison_user': [cycle.calories for cycle
                           in comparison_cycles], 'months':
                           cycle_months}
163 
164     elif graph_type == 'cycling_hours':
165         graph_data = {'current_user': [cycle.hours for cycle in
                user_cycles],
166                       'comparison_user': [cycle.hours for cycle in
                           comparison_cycles], 'months':
                           cycle_months}
167 
168     elif graph_type == 'swimming_calories':
```

```
169         graph_data = {'current_user': [swim.calories for swim in
                user_swims],
170                       'comparison_user': [swim.calories for swim in
                              comparison_swims], 'months': swim_months
                             }
171
172     elif graph_type == 'swimming_hours':
173         graph_data = {'current_user': [swim.hours for swim in
                user_swims],
174                       'comparison_user': [swim.hours for swim in
                          comparison_swims], 'months': swim_months}
175
176     print(graph_data)
177     return jsonify(graphData=graph_data)
```

Listing 21: ajax.py

9.4.8 main.py

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

```
1  from datetime import datetime
2  from math import floor
3  from calendar import month_name
4
5  from flask import Blueprint, render_template, flash, redirect,
       url_for, abort, request
6  from flask.ext.login import current_user, login_required,
       logout_user
7  from flask.ext.sqlalchemy import *
8  from random import randint
9  import re
10
11 from app.models import User, Activity, db
12 from app.helpers import validation_error, update_user, remove_sport
13 from app.performance_data import performance_data
14
15 main = Blueprint('main', __name__)
16
17 current_date = datetime.now().date()
18
19
20 @main.route('/')
21 @login_required
22 def home():
23     return redirect(url_for('main.performance', month='march'))
24
25
26 @main.route('/profiles/<username>', methods=['GET', 'POST'])
27 @login_required
28 def profiles(username):
29     # If the user has attempted to change their profile
30     if request.method == 'POST':
31         user = User.query.filter_by(id=current_user.get_id()).first
               ()
32
33         # If the user tries to change their name
```

```
34          if request.form.get('name'):
35              only_letters = re.compile(r'^[A-Za-z\-" "]*$')
36              if only_letters.match(request.form.get('name')):
37                  user.name = request.form.get('name').title()
38                  user.username = request.form.get('name').lower().
                        replace(' ', '').replace('-', '') + str(randint
                        (1, 10))
39                  update_user(user, 'name', False)
40                  return redirect(url_for('main.profiles', username=
                        user.username))
41              else:
42                  validation_error('Your name may only contain
                        letters and dashes.')
43
44          # If the user tries to change their email
45          elif request.form.get('email'):
46              valid_email = re.compile(r'^.+@[^.].*\.[a-z]{2,10}$')
47              if valid_email.match(request.form.get('email')):
48                  user.email = request.form.get('email')
49                  update_user(user, 'email')
50              else:
51                  validation_error('You must enter a valid email.')
52
53          # If the user tries to change their phone number
54          elif request.form.get('phone'):
55              valid_phone = re.compile(
56                  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
                        *$')
57              if valid_phone.match(request.form.get('phone')):
58                  user.phone = request.form.get('phone')
59                  update_user(user, 'phone number')
60              else:
61                  validation_error('You must enter a valid UK phone
                        number.')
62
63          # If the user tries to change their dob
64          elif request.form.get('dob'):
65              user.dob = request.form.get('dob')
66              update_user(user, 'date of birth')
67
68          # If the user tries to change their weight
69          elif request.form.get('weight'):
70              check_integer = re.compile(r'^-?[0-9]+$')
71              if not check_integer.match(request.form.get('weight')):
72                  validation_error('You must enter a number.')
73              elif not 10 <= int(request.form.get('weight')) <= 100:
74                  validation_error('Your weight must be between 10kg
                        - 100kg.')
75              else:
76                  user.weight = request.form.get('weight')
77                  update_user(user, 'weight')
78
79          elif request.form.get('delete'):
80              if request.form.get('delete') != 'I will lose
                    everything':
```

```
81                  validation_error('You must type in the message
                        exactly!')
82              else:
83                  user_id = current_user.get_id()
84                  logout_user()
85                  User.query.filter_by(id=user_id).delete()
86                  Activity.query.filter_by(user_id=user_id).delete()
87                  db.session.commit()
88                  flash('Your account was successfully deleted -
                        sorry to see you go!', 'success')
89                  return redirect(url_for('auth.login'))
90
91      possible_user = User.query.filter_by(username=username).
            first_or_404()
92      if current_user.username == possible_user.username:
93          activity_number = len(Activity.query.filter_by(user_id=
                current_user.get_id()).all())
94          total_users = len(User.query.all())
95
96          return render_template('profiles/own_profile.html',
                current_user=current_user, activity_number=
                activity_number,
97                                 total_users=total_users)
98      abort(403)
99
100     return redirect(url_for('main.profiles', username=current_user.
            username))
101
102
103 @main.route('/add-training', methods=['GET', 'POST'])
104 @login_required
105 def add_training():
106     activities = Activity.query.filter_by(user_id=current_user.
            get_id(), date=current_date).all()
107     total_calories = 0
108     total_hours = 0
109     for activity in activities:
110         total_calories += activity.calories
111         total_hours += activity.hours
112     return render_template('training/add_training.html', date=
            current_date,
113                            current_user=current_user, activities=
                                activities, total_calories=
                                total_calories,
114                            total_hours=total_hours)
115
116
117 @main.route('/performance/<month>', methods=['GET', 'POST'])
118 @login_required
119 def performance(month):
120     months = [month_name[x].lower() for x in range(1, 13)]
121     all_activities = Activity.query.filter_by(user_id=current_user.
            get_id()).all()
122     available_months = []
123
124     for activity in all_activities:
125         for x in range(1, 13):
```

```
126             if activity.date.month == x and months[x - 1] not in
                    available_months:
127                 available_months.append(months[x - 1])
128     print(available_months)
129
130     if month.lower() in available_months:
131         user_data = performance_data(month.lower())
132         return render_template('performance/user_performance.html',
                user_data=user_data,
133                                current_month=month.title(), months=
                                    available_months)
134     abort(404)
135
136
137 @main.route('/performance/activity/<int:activity_id>')
138 @login_required
139 def individual_activity(activity_id):
140     activity = Activity.query.filter_by(id=activity_id).
            first_or_404()
141     if activity.user_id == current_user.get_id():
142         return render_template('performance/individual_activity.
                html', activity=activity)
143     return abort(404)
144
145
146 @main.route('/performance/compare', methods=['GET', 'POST'])
147 @login_required
148 def compare_performance():
149     users = User.query.filter_by(charity_event=0).filter(User.id !=
            current_user.id).all()
150     user_list = sorted([[user.id, user.name] for user in users])
151     return render_template('/performance/compare_performance.html',
            users=users, user_list=user_list)
152
153
154 @main.route('/rankings')
155 @login_required
156 def rankings():
157     user_ranking = {}
158     runners = User.query.filter_by(charity_event=False).all()
159     for runner in runners:
160         total_calories = 0
161         training_sessions = Activity.query.filter_by(user_id=runner
                .id).all()
162         for session in training_sessions:
163             total_calories += session.calories
164         user_ranking[runner.name] = total_calories
165
166     user_ranking = sorted(user_ranking, key=user_ranking.get,
            reverse=True)
167
168     return render_template('/training/rankings.html', running_team=
            user_ranking)
169
170
171 @main.route('/delete/<int:activity_id>')
172 def delete_activity(activity_id):
```

```
173     remove_sport(activity_id)
174     flash('Your training session was deleted!', 'success')
175     return redirect(url_for('main.home'))
176
177
178 @main.errorhandler(404)
179 def page_not_found(error):
180     return render_template('errors/404.html'), 404
```

Listing 22: main.py

Part III
Testing and Evaluation