Media Engineering and Technology Faculty
German University in Cairo

Real-time fitness monitor

Bachelor Thesis

Author: Mohannad Banayosi
Supervisor: Professor Georg Jung
Submission Date: 04 June, 2013

Media Engineering and Technology Faculty
German University in Cairo

Real-time fitness monitor

Bachelor Thesis

Author: Mohannad Banayosi
Supervisor: Professor Georg Jung
Submission Date: 04 June, 2013

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

---

Mohannad Banayosi
04 June, 2013

Acknowledgments

This project would not have been possible without my supervisor, professor Georg Jung's continuous support and encouragement. I am also very grateful to my family because of the support they gave to me from the beginning of the project till the very end.

I would like to give a special thank you to my grandmother, who has motivated me to work when I needed it the most.

Lastly, I would like to thank my friends Omar Mowafi, Fady Kamal, Nada Nasr, Mai Nasser, Amr Tharwat, Tarek Samy and many others for giving me feedback about my work when I needed it, as well as supporting me the whole time.

Abstract

IMPACT is a multidisciplinary project in which students of computer science (more precisely: software engineering), material sciences, mechatronics and embedded system design come together to create Thai-boxing pads with builtin impact sensors and wireless connection to a base station. In this project, we will implement a multi-sensor tracking and analysis tool for a physical workout routine specific to Thai/kick-boxing or similar contact sports to track and display performance and fitness level of a practitioner over a single and over multiple sessions. It will be used to record and correlate the inputs from impact sensors and technique recognition and track improvemets of practitioners over time.

Contents

Chapter 1

Introduction

Nowadays, technology is a fundamental part of our daily lives, as it facilitates solving problems that humans face as well as analyze data they can not easily interpret. In the world of martial arts, there is no easy accurate way to study and analyze the performance of a trainee statistically. This absence of analysis is due to the lack of tools presented to the trainers and trainees.

1.1 Aim of this project

The aim of this project is to deliver an easy-to-use interface that displays training sessions' info and statistics. This in turn will help the trainers to get an accurate analysis of the trainee's performance and examine different aspects of a training session. This interface will allow multiple plugins to be to be plugged in simultaneously.

1.2 Background

In this project a multi-sensor tracking and analysis tool will be implemented. This tool would be connected to multiple plugins and sensors to record and analyze a specific physical workout routine to track performance and fitness level of a practitioner over a single or multiple sessions.

This interface itself is very generic that any type of plugins could be connected to it (check 4.2.1 for configuration steps)[1].

[1]The plugin should have the ability to send the data to the main interface through sockets.

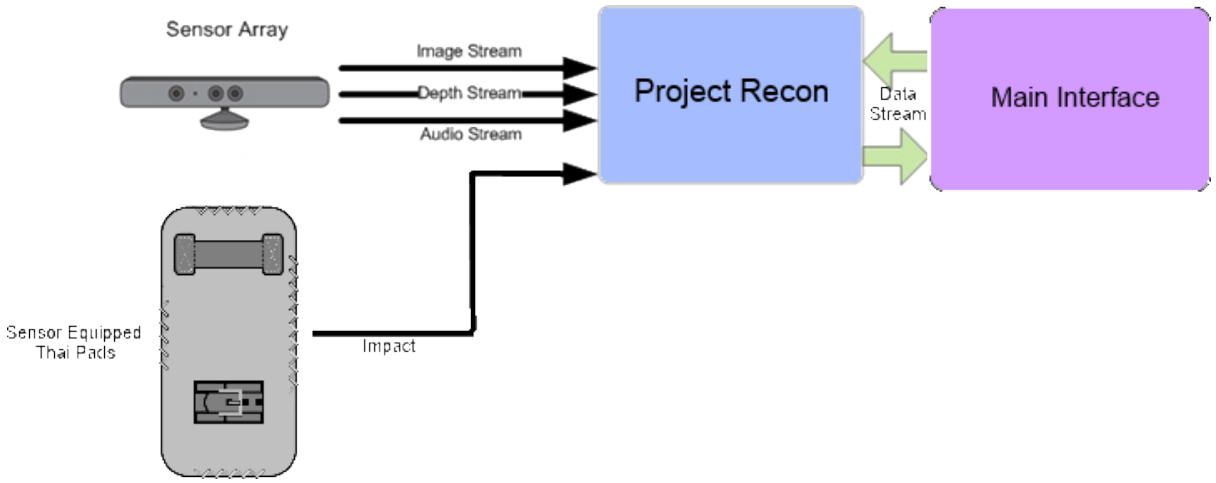

Figure 1.1: General Overview

Chapter 2

Implementation

The application is coded using Django (a Python based framework), SQLite, HTML, CSS3, Jquery and Javascript. Python and SQLite are used to implement the back-end layer, while HTML, CSS3, JavaScript, jQuery and Ajax are for the front-end layer.

2.1 Architecture

The architecture's design 2.1 represents the different components included in the interface.

2.1.1 Interpreter

Containing several interpreter modules (Check figure 2.2), the interpreter serves as the brain of the interface receiving the data (initially received through the sockets) and then begins digesting and translating it into actual info. It then sends it to the monitor (to display the data received) and the database manager (to save the data).

2.1.2 Database engine

The database engine saves the users and their data; this data, or history, is collected from previous sessions the trainee took.

2.1.3 Statistics engine

Containing several statistics modules, this engine generates the analysis (this analysis depends on the current data received + the history of the trainee) and sends it to the monitor (Check figure 2.3).

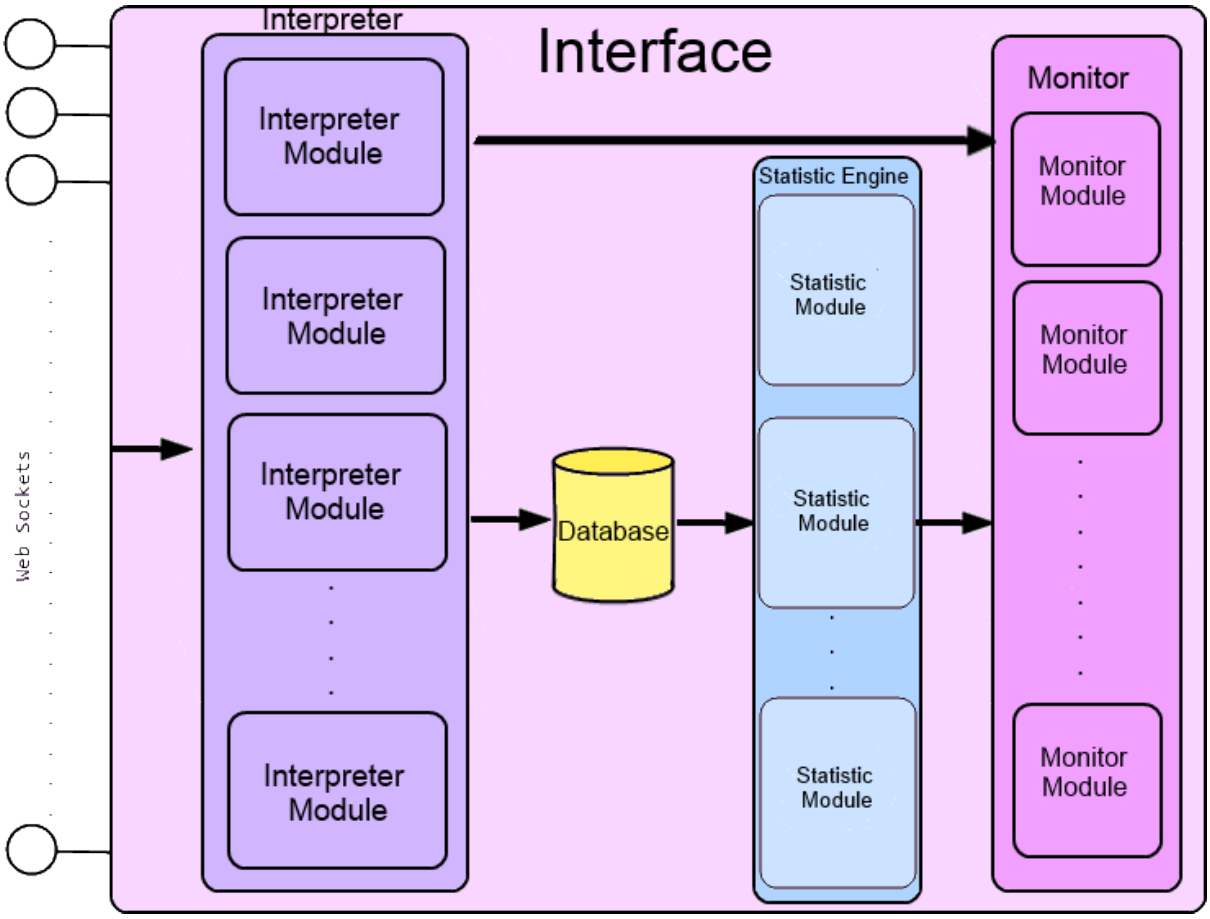

Figure 2.1: The Architecture

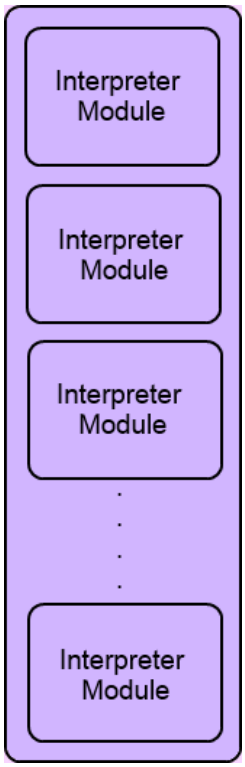

Figure 2.2: The Interpreter

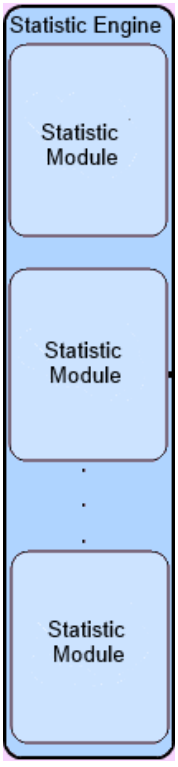

Figure 2.3: The Statistics Engine

2.1.4 Monitor

The monitor is the front end component of the interface that displays all the info (including live data and statistics-related data). It contains several monitoring modules together make up the monitor (Check figure 2.4).

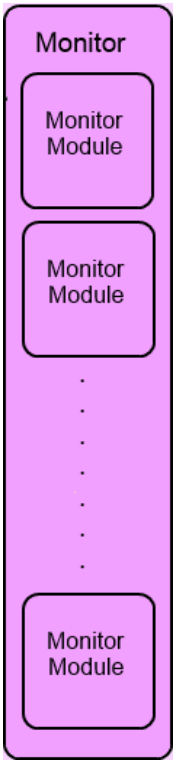

Figure 2.4: The Monitoring System

2.2 Technologies and frameworks researched

Various of technologies and frameworks have been researched, some of them were chosen to be used and others not. Following is a list of them. Most of the technologies researched where not used because of their violation of the core feature of the interface being generic.

Depthjs [1] Depthjs was developed by four MIT students, Aaron Zinman, Doug Fritz, Greg Elliott and Roy Shilkrot in 2010. Depthjs allows any web page to interact with the Microsoft Kinect using Javascript. It provides the low-level raw access to the Kinect as well as high-level hand gesture events to simplify development.
Depthjs is an open source project, but with no enough documentation, compatible with Google chrome only, and is more helpful in web navigation.

[1] Depthjs website http://depthjs.media.mit.edu/

Zigfu [2] Zigfu was developed by Ted Blackman, Shlomo Zippel, Roee Shenberg, Amir Hirsch, Bryce Tucker. They developed the ZDK (Zigfu Development Kit) to make cross-platform, motion-controlled apps with Kinect in HTML5/Javascript, Unity3D and Flash. Applications made with this development kit are portable across all operating systems, web browsers, computer vision middlewares, and 3D sensors.
Zigfu is not an open source project, still have problems in performance and accuracy in gesture recognition.

OpenDepth [3][4] OpenDepth - Extending the Web - is a client server application developed by Ecaterina Paun, Narcis Paun and Mircea Piturca. OpenDepth brings Kinect SDK methods and data to the web with the server side written on C# and built on top of Kinect SDK 1.5, while the client side is a written in JavaScript and uses the Three.js webGL rendering engine.
It is an open source project, with enough documentation, and is more helpful in web navigation.

Django [5] Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Developed by a fast-moving online-news operation, Django was designed to handle two challenges: the intensive deadlines of a newsroom and the stringent requirements of the experienced Web developers who wrote it. It lets you build high-performing, elegant Web applications quickly.

Web Sockets WebSockets technology provides a new World Wide Web Consortium (W3C) JavaScript API and protocol for two-way communication over the Internet. This new protocol makes it easier to work directly with fixed data formats, and it bypasses the slower document-based HTTP protocol.

Pusher [6] Pusher is a hosted API, based on HTML5 and web sockets programming, that is used for quickly, easily and securely sending data in realtime functionality to applications.

[2]Zigfu website http://zigfu.com/
[3]OpenDepth website http://opendepth.net/
[4]Kinect to Web Browser http://social.msdn.microsoft.com/Forums/en-US/kinectsdknuiapi/thread/5fadcf7c-f261-4580-9302-7500c10bff40/
[5]Django website https://www.djangoproject.com/
[6]Pusher website http://pusher.com/

2.3 Technologies and frameworks used

2.3.1 Back-end layer

This layer is the one responsible for the work done in the engine of the application. It does the part of the work invisible to the user and serves indirectly as a support for the front-end layer, usually by being closer to the required resource or database, or having the capability to communicate with the required resource or database

Python [7] Conceived in the late 1980s by Guido van Rossum, Python is a general-purpose, high-level programming language with a syntax that allows developers to express concepts in fewer lines of code than would be possible in languages such as C, C++, Java. It supports multiple programming paradigms such as imperative, object-oriented and functional programming paradigms. It also features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts with its interpreters available for many operating systems.

Python is used to implement the logic of the monitor. This logic serves as the engine of the monitor that handles all the data received, generating the statistics and communication between the database and the front-end layer.

SQLite [8] Designed in the spring of 2000 by D. Richard Hipp, SQLite is a relational database management system contained in a C programming library. Unlike clientserver database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, the SQLite library is linked in and thus becomes an integral part of the application program. It is a popular choice as embedded database for local/client storage in application software such as web browsers. It is the most widely deployed database engine, as it is used today by several widespread browsers, operating systems, and embedded systems, among others.

SQLite is used to store all the data that would be needed for future use. This data include the practitioners' basic info (usernames and passwords), sessions' info (start time, end time, duration and off-round measurements), rounds' info (start time, end time, duration and on-round measurements + data) and statistics.

[7] Python website http://www.python.org/
[8] SQLite website http://www.sqlite.org/

2.3.2 Front-end layer

This layer does the part of the work visible to the users. It is the one that the users interact with directly.

HTML HTML stands for HyperText Markup Language. It is the main markup language for creating web pages and other information that can be displayed in a web browser. It is written in the form of HTML elements consisting of tags enclosed in angle brackets (like ¡html¿), within the web page content. In between these tags, text, tags, comments and other types of text-based content can be added.

HTML is used to layout the monitor's user interface UI.

CSS3 CSS stands for Cascading Style Sheets. It is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document, including plain XML, SVG and XUL. It is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content.

CSS3 is used to design the layout created by HTML.

JavaScript JavaScript is an interpreted computer programming language originally implemented as part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that was displayed. Now though it has many uses involving popular game development and the creation of applications.

jQuery [9] Released in January 2006 at BarCamp NYC by John ResigjQuery, jQuery is a multi-browser JavaScript library designed to simplify the client-side scripting of HTML. It is currently developed by a team of developers led by Dave Methvin. Used by over 65% of the 10,000 most visited websites, jQuery is the most popular JavaScript library in use today.

JavaScript and jQuery are used to code the logic part needed in the front-end layer. This logic includes the starting and stopping of the rounds and sessions, the stopwatch/-timer and displaying the data received from the back-end layer (after the data is parsed).

[9] jQuery website http://jquery.com/

Ajax Ajax is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page. Data can be retrieved using the XMLHttpRequest object. The use of XML is not required (JSON is often used instead), and the requests do not need to be asynchronous.

Ajax is used to allow the data received from the sensors to be sent through the front-end layer to the back-end layer in the background (in a "live" style). It is also used to send data from the back-end layer to the front-end layer and displaying it without interfering with the ongoing processes.

2.4 Interface implementation

2.5 Connection with the plugins

2.5.1 Hand shaking

This is the process where each plugin reserves a web socket to start the data stream with the interface. This begins when the plugin sends a request to the interface via a socket, then the interface reserves the socket for the plugin and sends an OK signal. Once the OK signal is received, the plugin can start the data stream at any time.

2.5.2 Data stream

The data stream contains the type of the data sent (on-session or off-session), a timestamp and the actual data. Depending on the type of the sensor sending the data, the actual data will differ. These data stream elements will be received by the front-end of the monitor, which in turn sends it to the back-end layer to interpret. The back-end layer then parses the data, saves it in the database and sends it back to the front-end layer (this time only the actual data is sent). The front-end layer displays the data according to its type.

Chapter 3

Features

3.1 Generic plugins

One of the most important features of the monitor is the fact that it can be connected to any type of sensors. The monitor is built to accept any types of plugins, all what is needed is a couple of steps of installation 4.2.1. The aim of this feature is to open the door to future users and developers to expand the project for future plugins and sensors.

3.2 Live session monitoring

In this feature, the user creates a session (with a number of rounds, round duration and break duration) and then connects the sensor(s) needed and can do off-session measurements. Then the user starts the session and records the actions done.

One of the biggest challenges faced during the implementation of this feature was that the rounds should be saved to the database very quickly that the user would not notice that their is something happening in the background of the application. A proposed solution was that the round is saved at the end of it, this caused a small lag in the timer (this was very noticeable in sessions that had a short break between the rounds). The solution implemented was creating and saving the rounds objects to the database while creating the session and then when something is needed to be updated in the round's data, it is saved during the round, therefore minimizing the number of queries executed during the session.

3.3 Statistics

This feature serves to show the practitioners their progress over time. They can see their best-done moves, frequent-done moves, moves they need to practice on, rate of their moves with respect to the round's duration and other session-related info.

3.4 Social

In this feature, the practitioners can check other practitioners' level, progress and session data. The main reason behind this is to give the application a bit of gaming taste.

Chapter 4

Manual

4.1 User Experience

4.1.1 Registration

The user simply enters a username (which should be unique) and a password and clicks "Register". An account will be created for the user and will be able to log in at anytime (Check figure 4.1).

4.1.2 Logging in

The user simply enters the username and password and clicks "Log in". Now he/she is logged in and can use all the features of the monitor (Check figure 4.2).

4.1.3 Logging out

To log out, click the "LOG OUT" button in the left top corner (Check figure 4.3).

4.1.4 Recording a new session

1. Make sure the user is registered (Check figure 4.1).
2. Log in to the system (Check figure 4.2).
3. Click on "Start a new session" (Check figure 4.4).
4. Enter the number of rounds, duration of these rounds and the duration of the breaks between the rounds (Check figure 4.5).

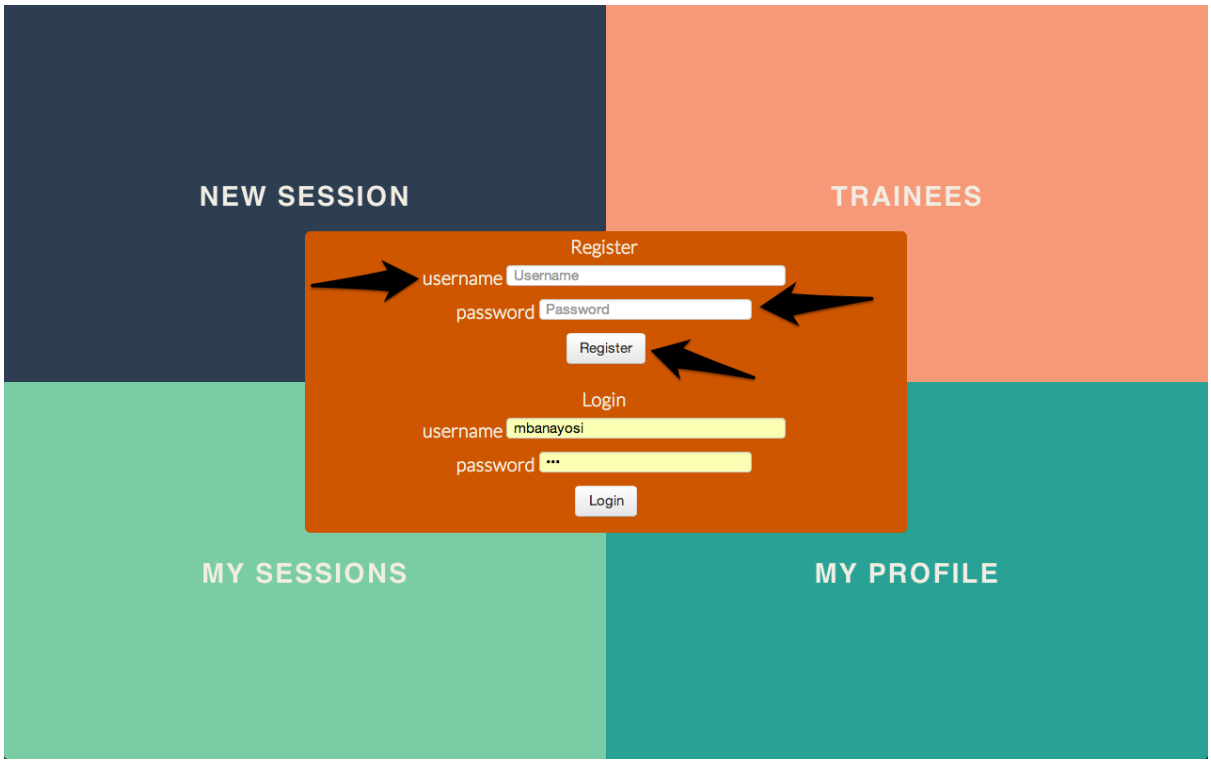

Figure 4.1: Registering

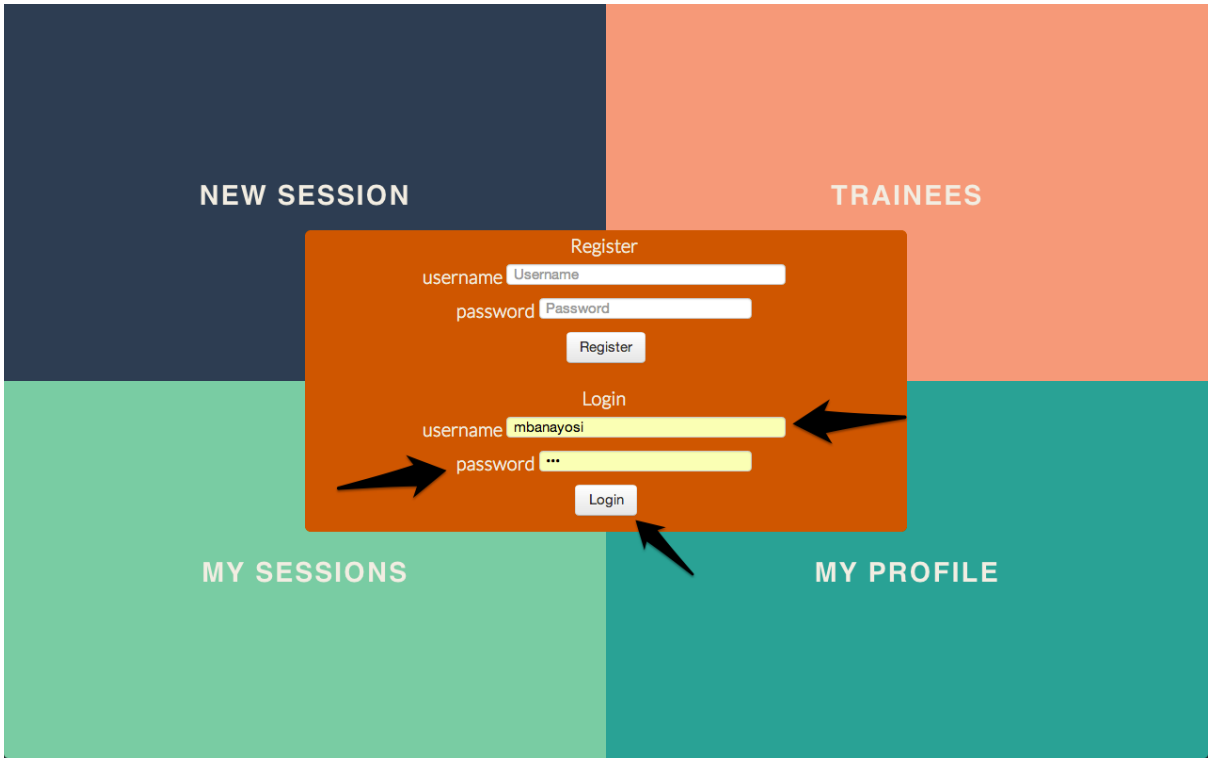

Figure 4.2: Logging in

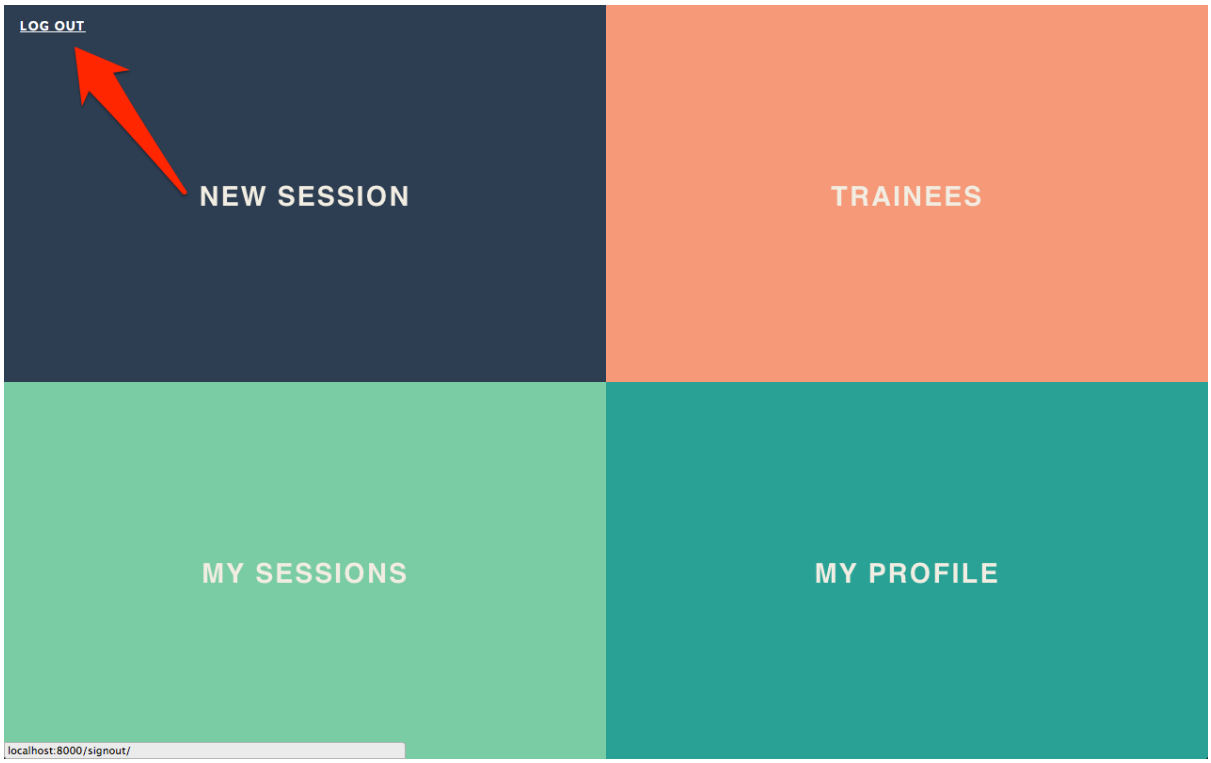

Figure 4.3: Logging out

5. Make sure that all plugins and sensors are connected and ready (Check 4.3 to see how to get a plugin ready).
6. Make off-rounds readings.
7. Click the "START" button to start the session (Check figure 4.6).

4.1.5 Viewing old sessions

To view your past sessions, simply click on the "My Sessions" button and a list of your sessions with additional general info will be listed. To view the detailed view of a certain session, click on the session and the rounds' info will be displayed (Check figures 4.7 and 4.8).

4.1.6 My profile

Viewing my profile

Click the "My Profile" button to view your profile with your info (Info like name, description photo and other stuff) listed. The info listed in this view will be publicly viewed to other users and practitioners (Check figures 4.9 and 4.10).

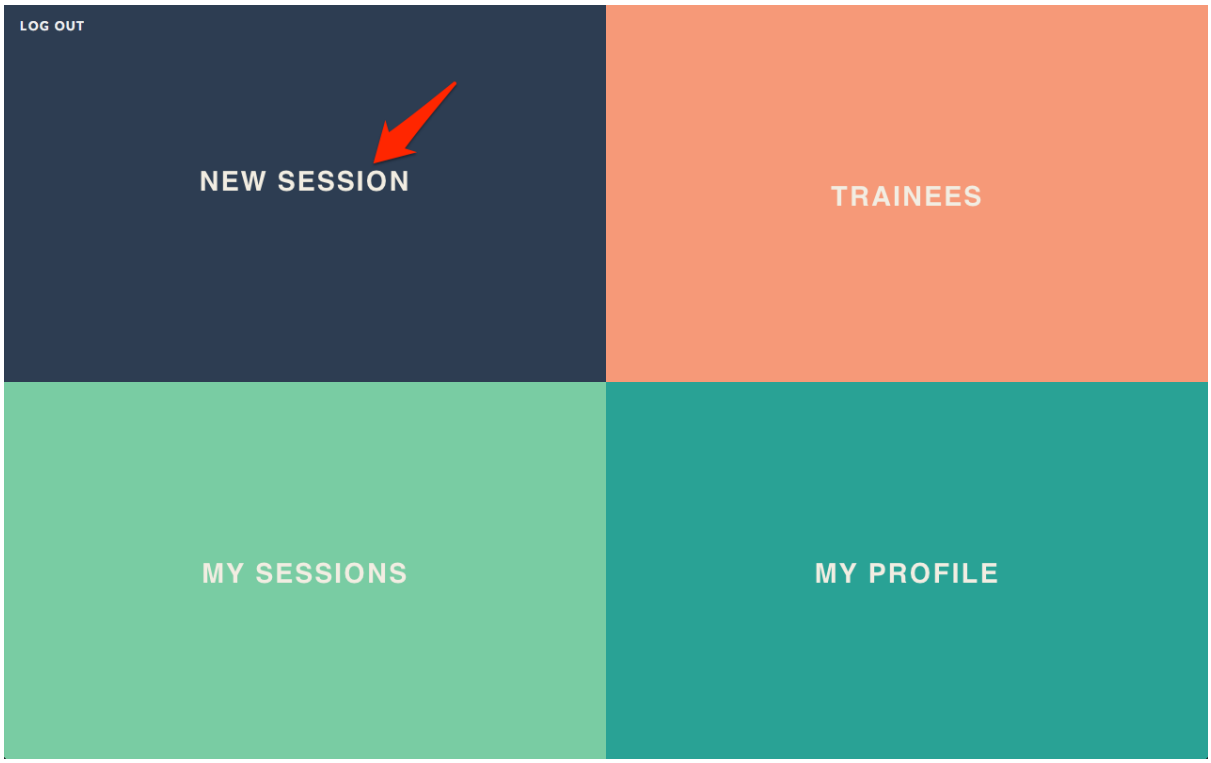

Figure 4.4: Starting a new session

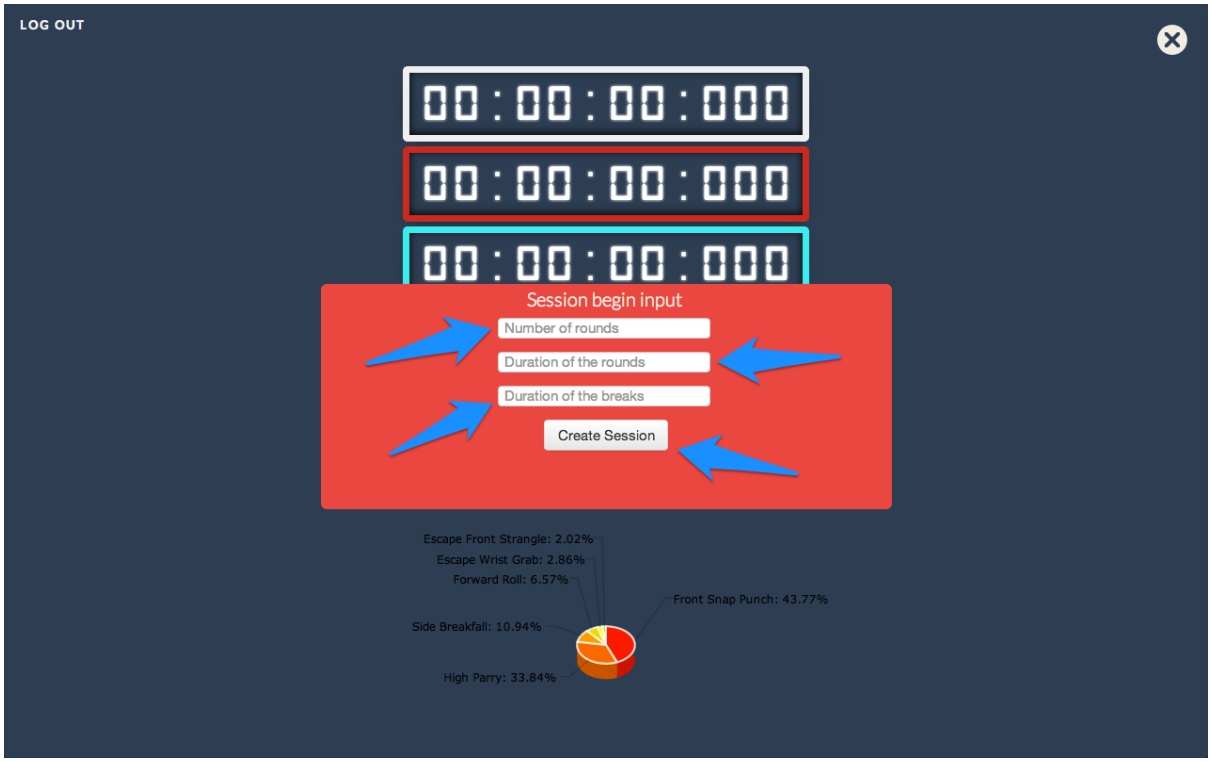

Figure 4.5: Starting a new session

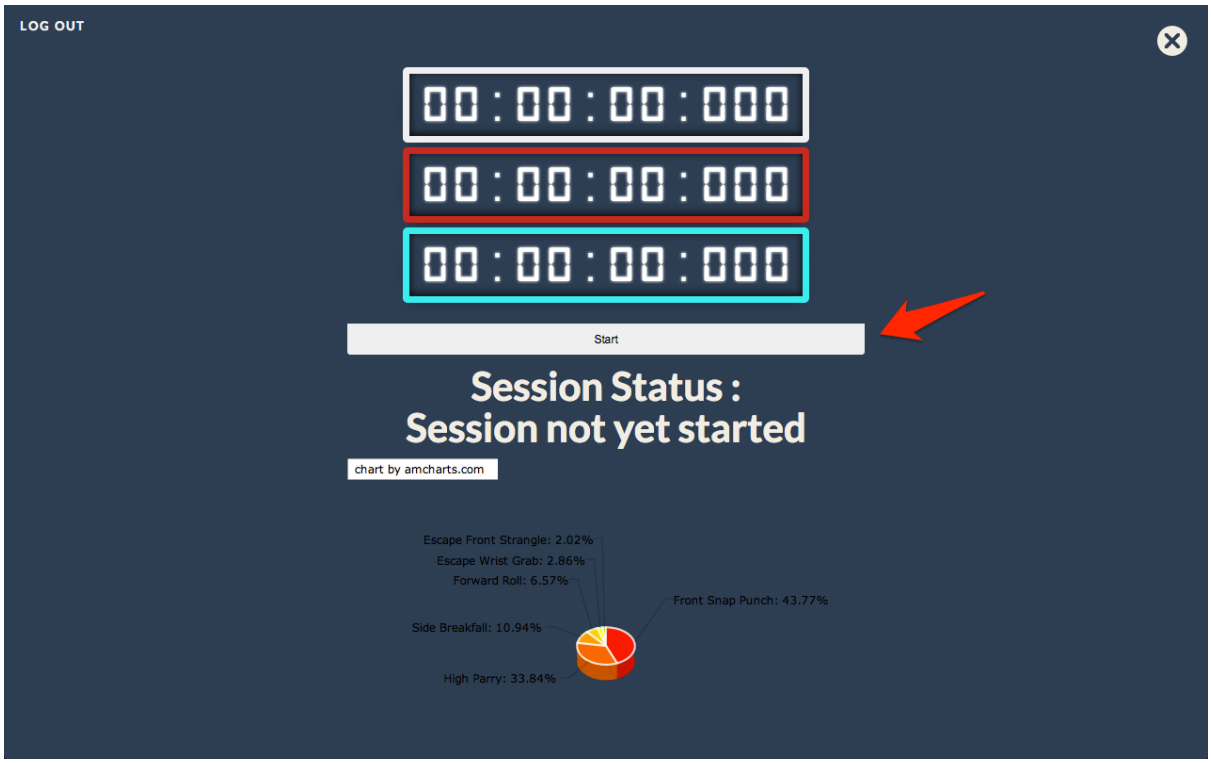

Figure 4.6: Starting a new session

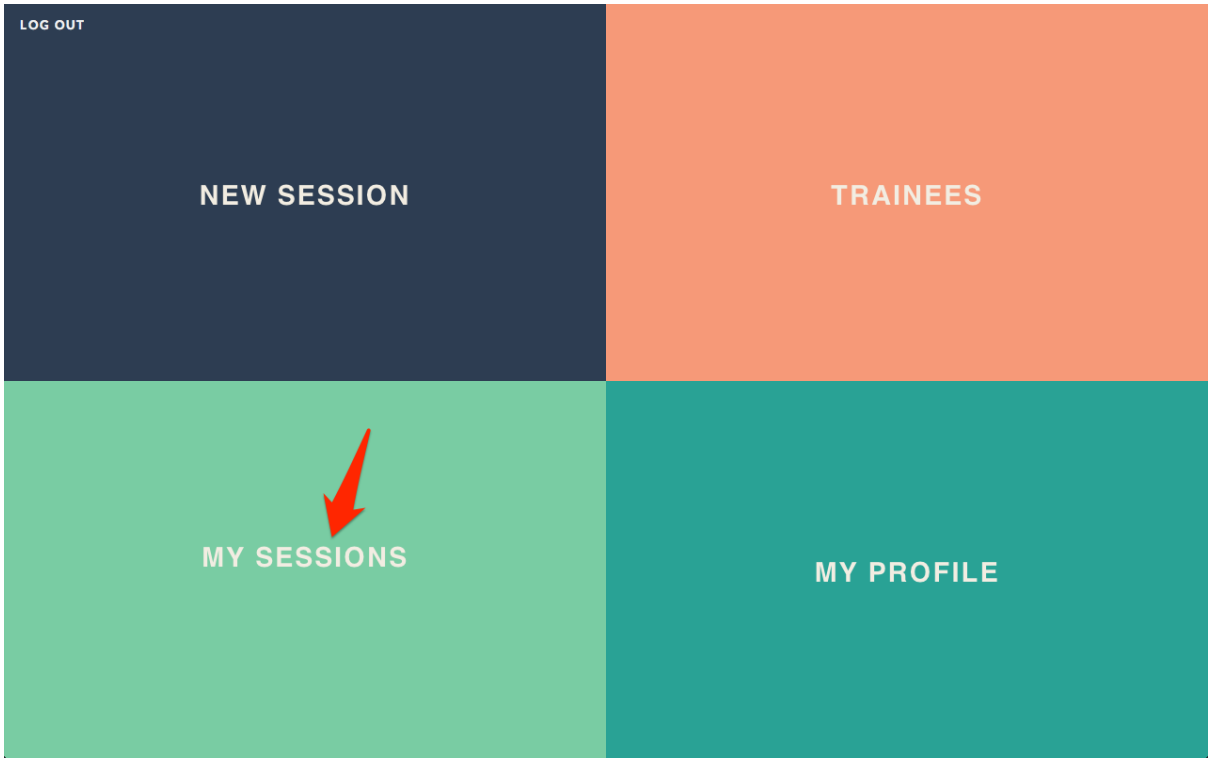

Figure 4.7: Viewing old sessions

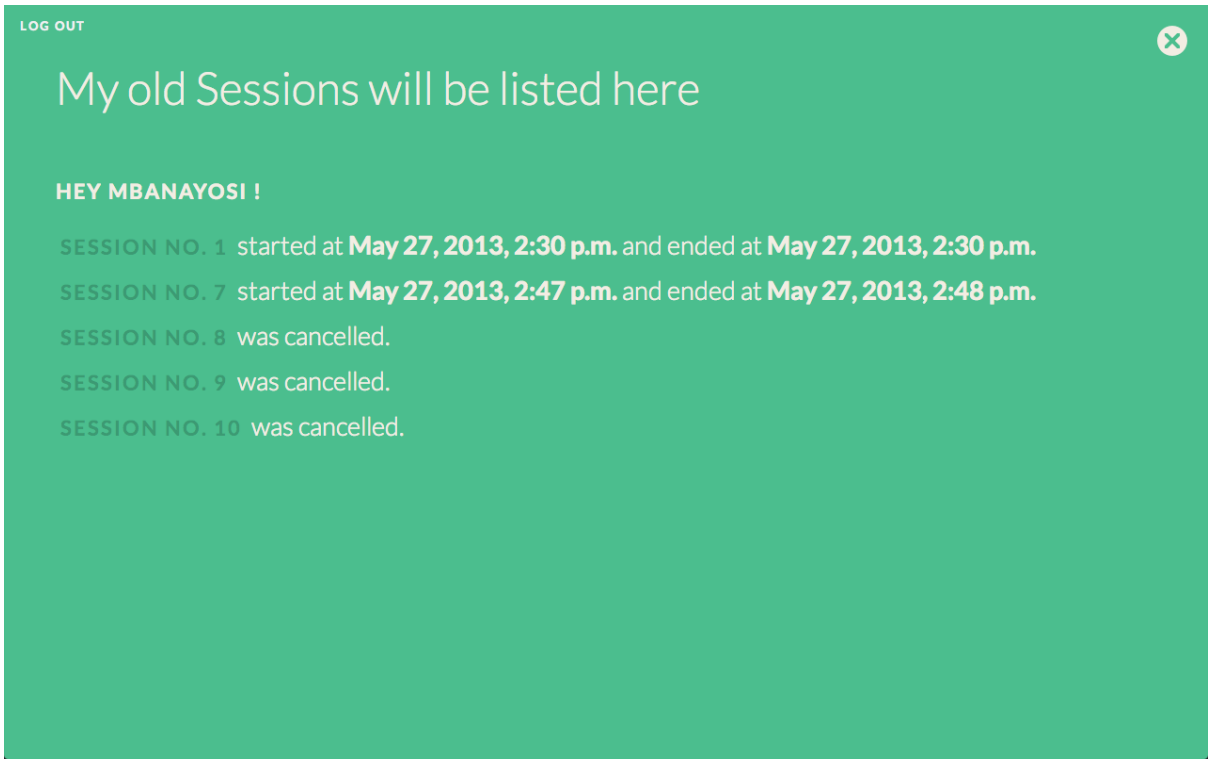

Figure 4.8: Viewing old sessions

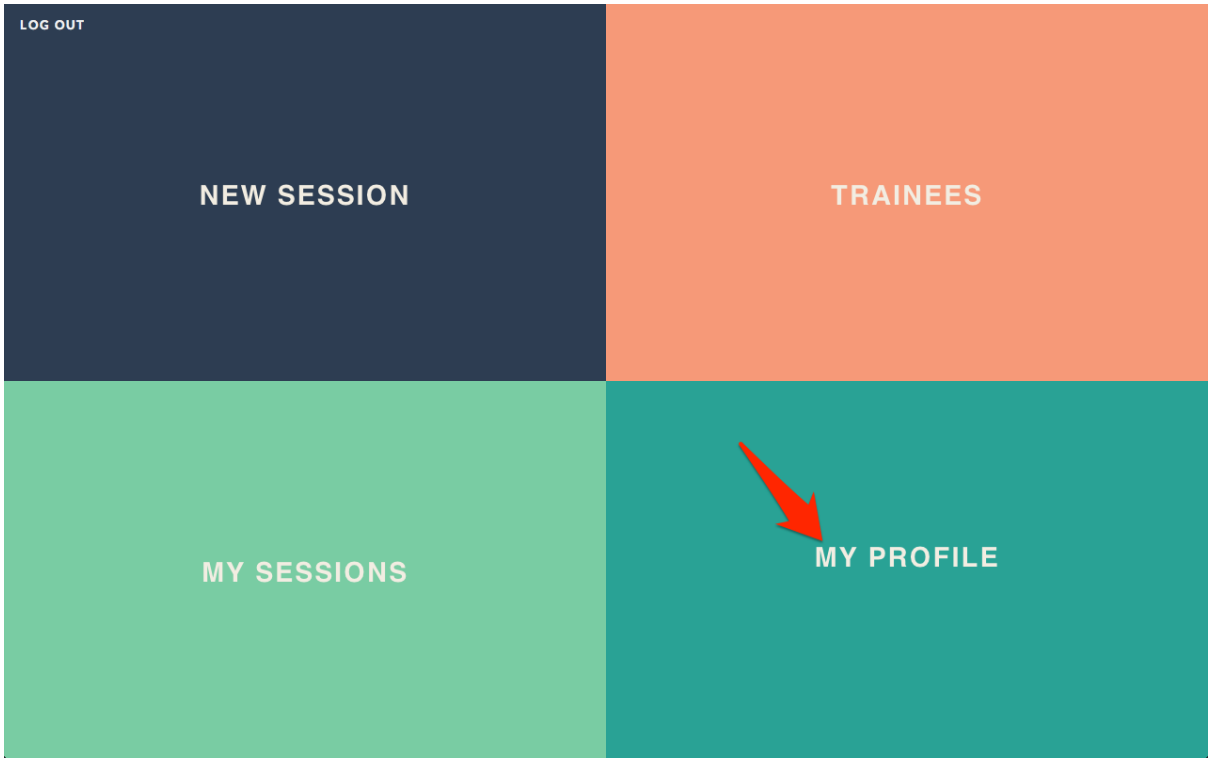

Figure 4.9: Viewing my profile

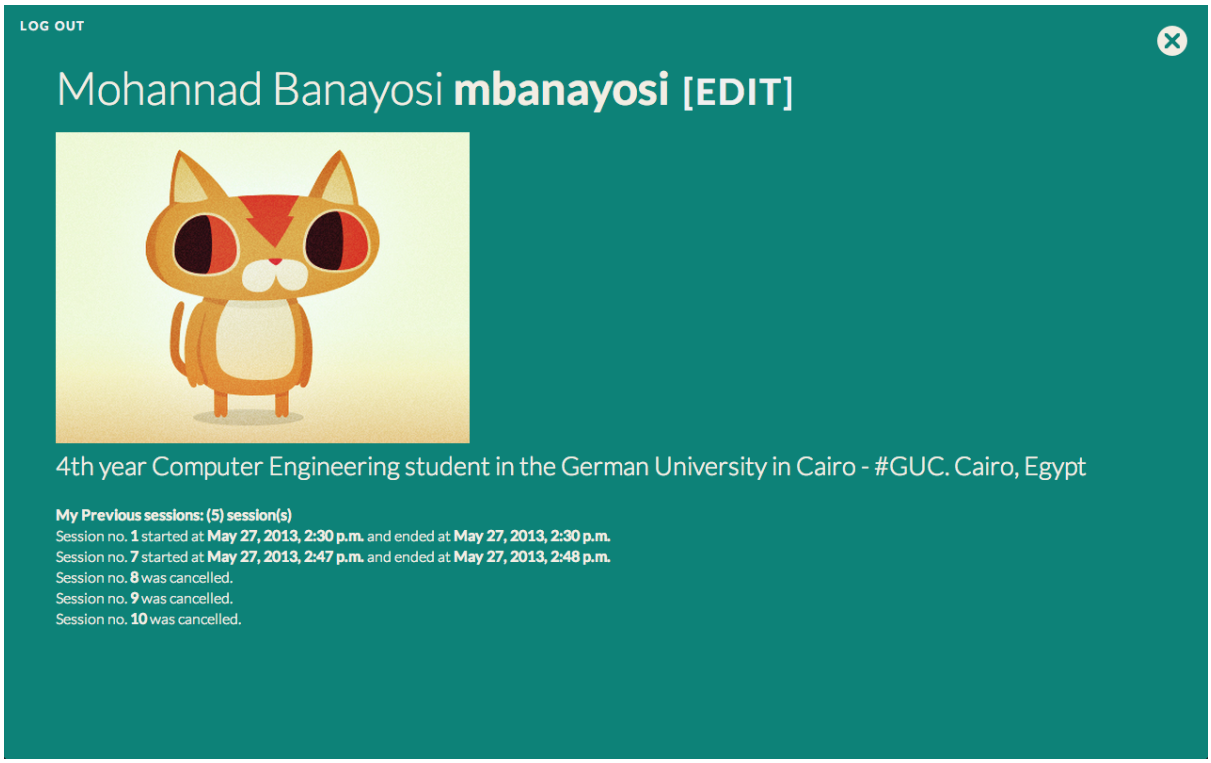

Figure 4.10: Viewing my profile

Edit my info

To edit your info, visit your profile (Check 4.1.6 to view your profile) and click the "EDIT" button. A box with with your info will appear, do the changes and then click "SAVE" (Check figures 4.11 and 4.12).

4.1.7 Viewing other practitioners

To view other registered users, their info and sessions, click on the "Trainees" button and a list of the users with their profile photos will be displayed. The profile photo can be clicked and a detailed view of the trainee's profile will be displayed; click the "NEXT" button to iterate between the users (Check figures 4.13, 4.14 and 4.15).

4.1.8 Navigating to the previos page

To go to the previous page, simply click on the X button in the top-right corner (Check figure 4.16).

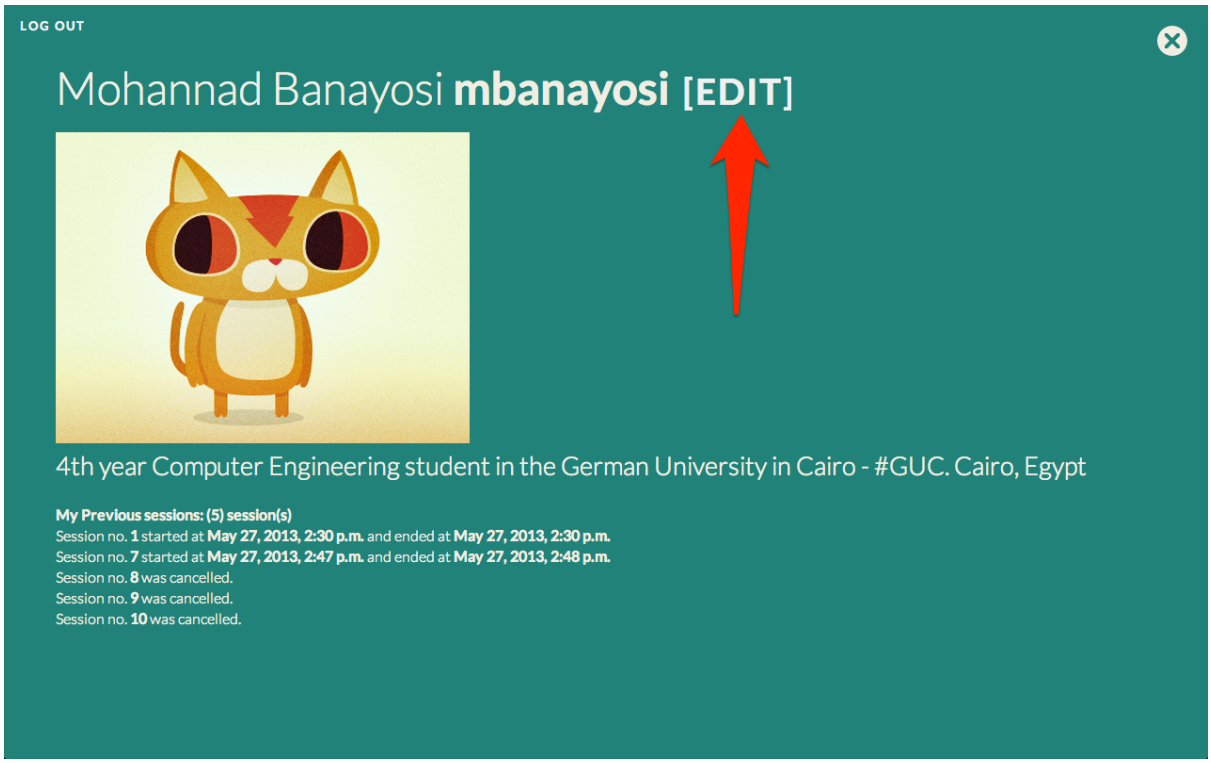

Figure 4.11: Editing my profile

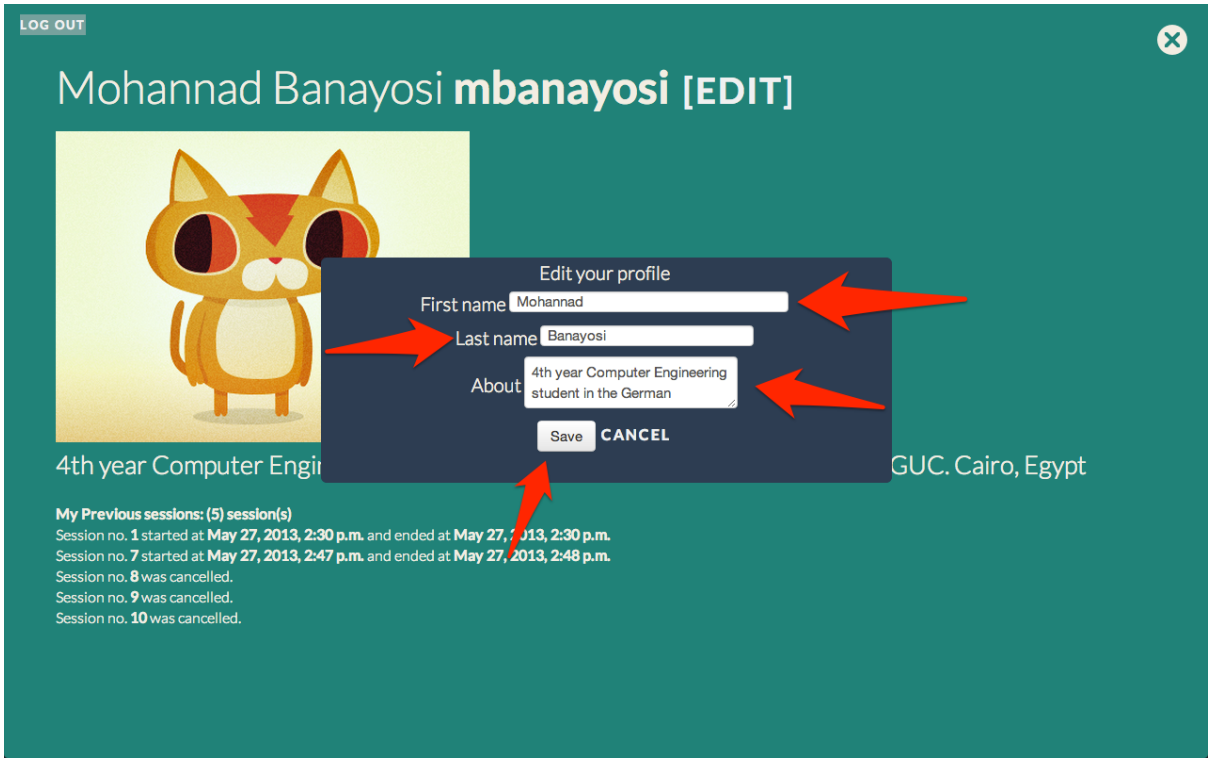

Figure 4.12: Editing my profile

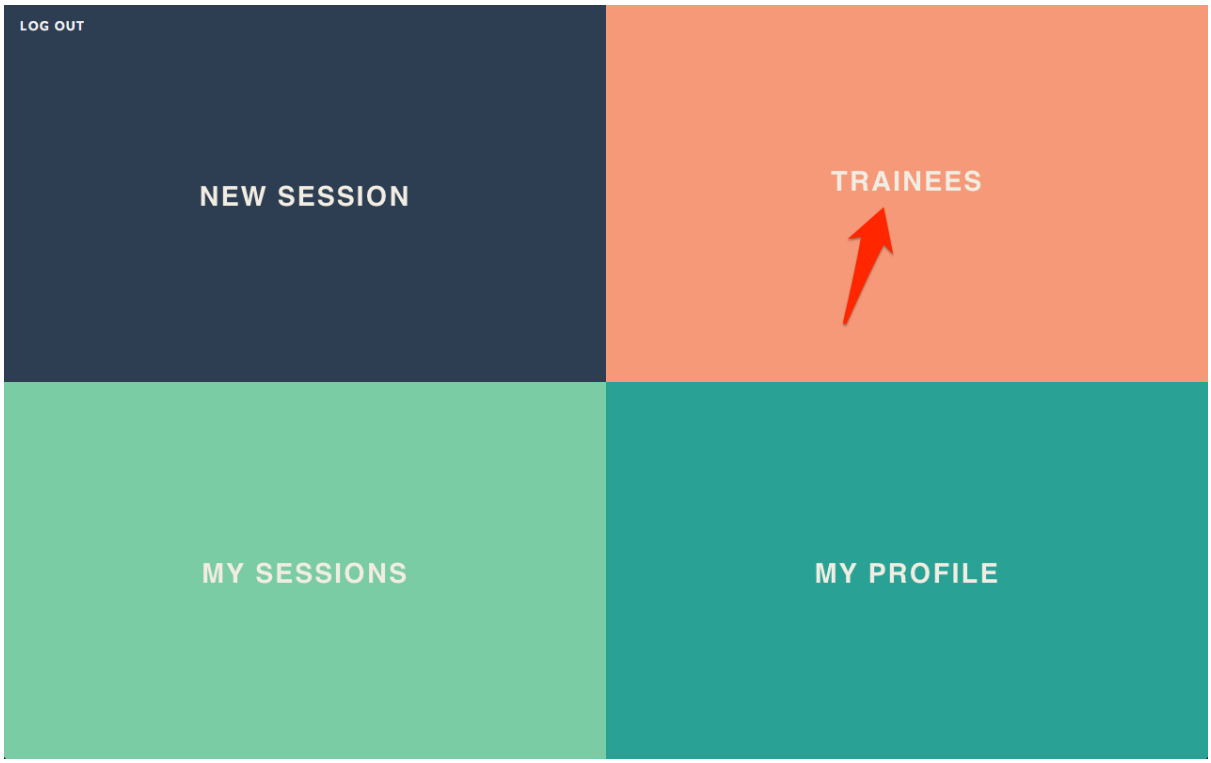

Figure 4.13: Viewing other practitioners

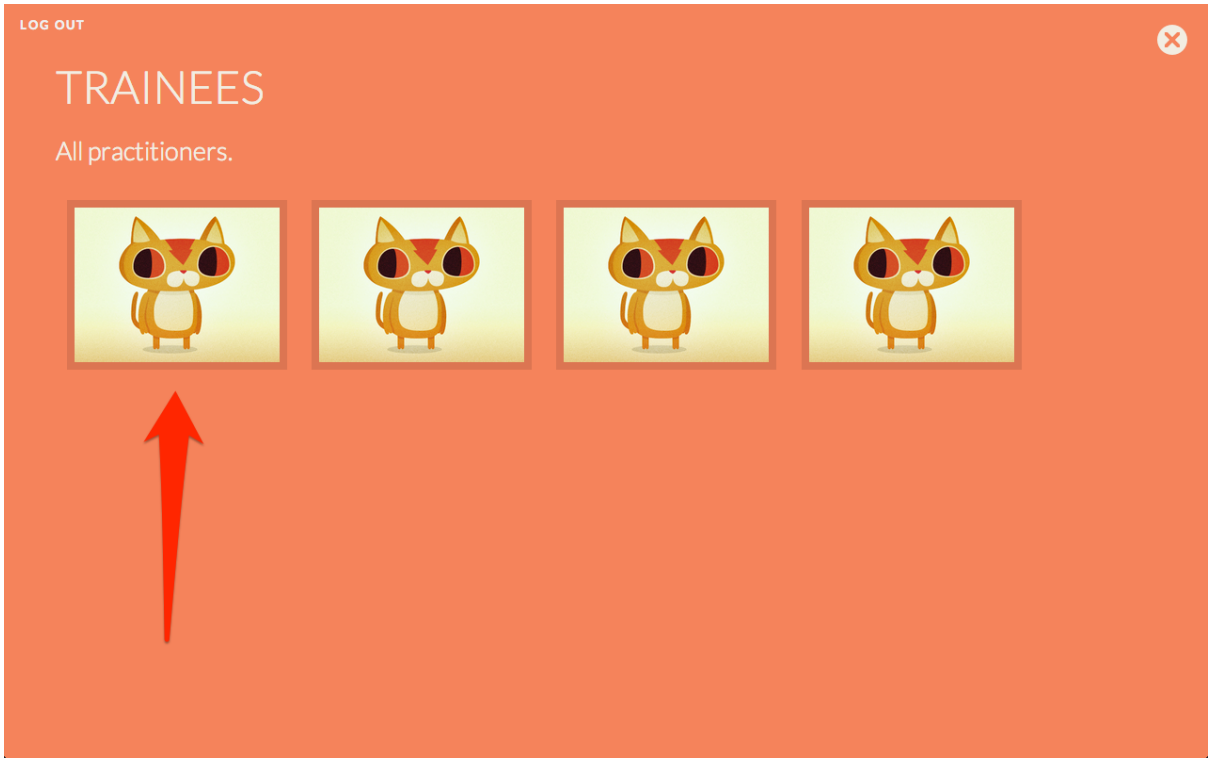

Figure 4.14: Viewing other practitioners

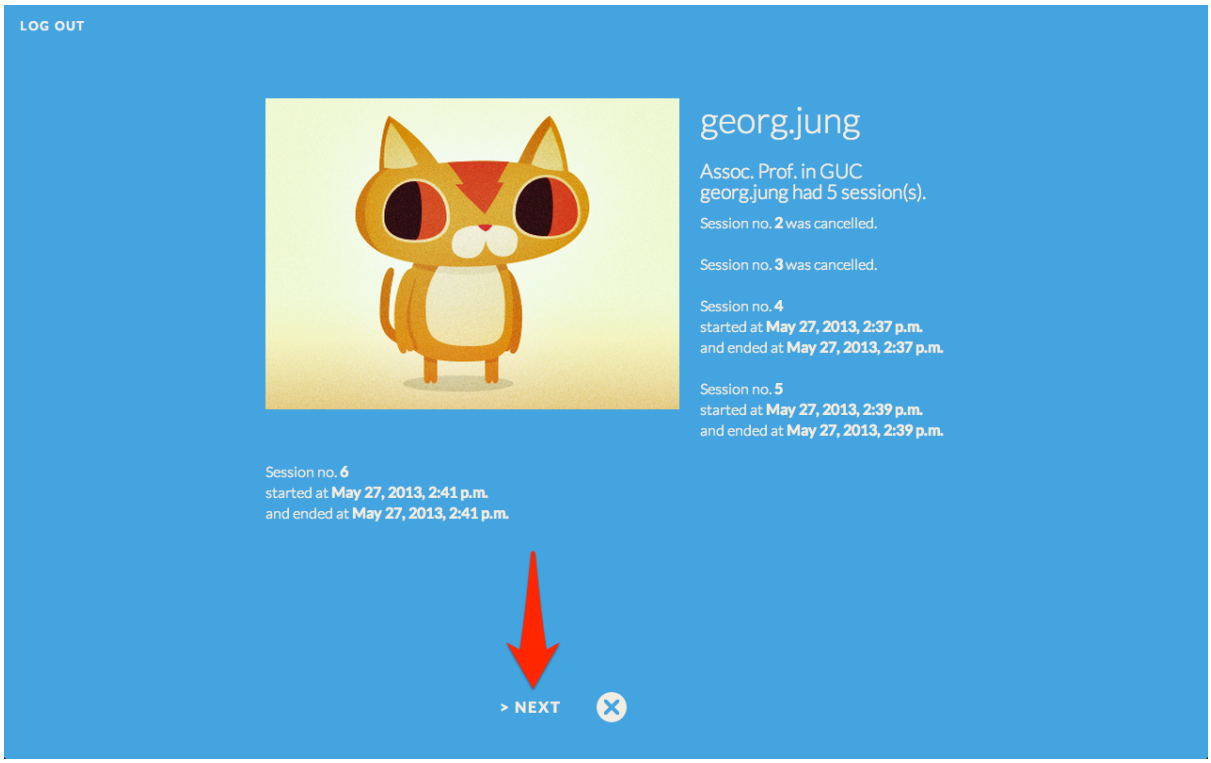

Figure 4.15: Viewing other practitioners

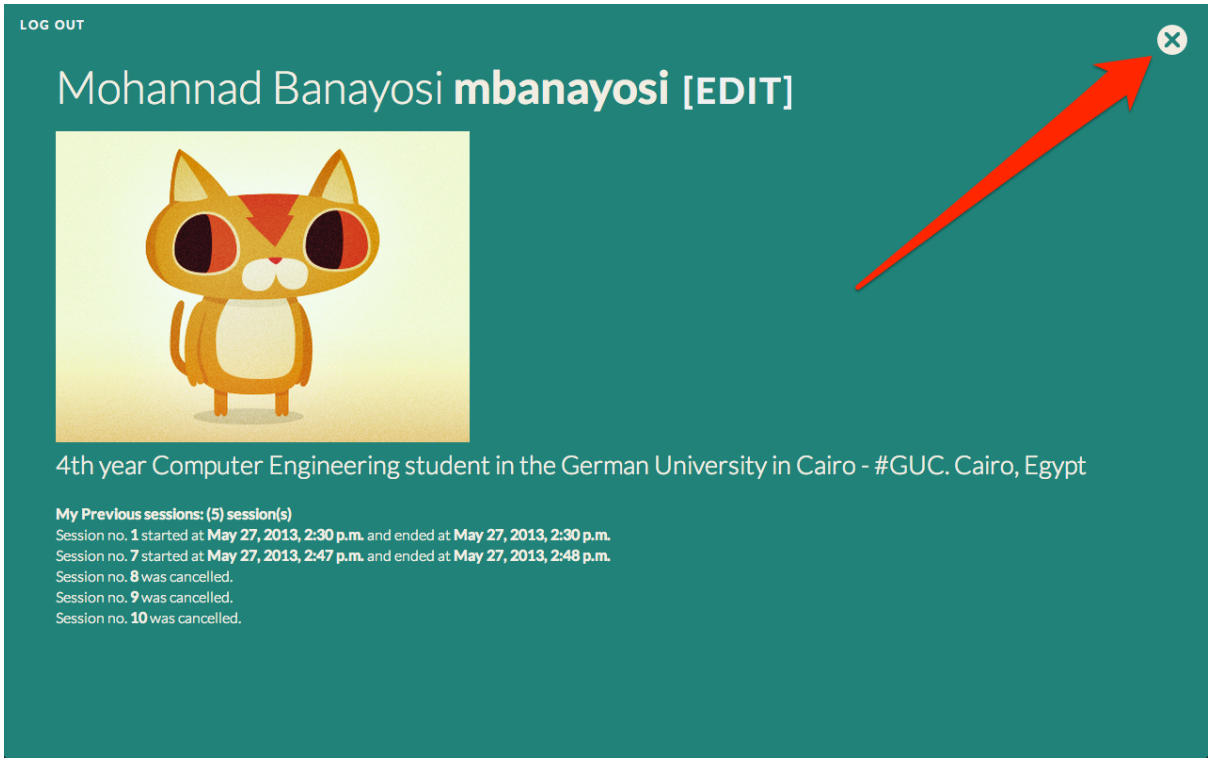

Figure 4.16: The back button

Listing 4.1: Detect Method

```
1 def heartrate_offsessionreading:
        # insert code for the interpreter here.
```

4.2 Advanced users interface configuration

4.2.1 New plugin installation steps

To add a new plugin to the system, the following steps should be done.

1. Add the interpreter module, responsible for the interpretation of the data received, to the interpreter (Check 4.2.2 for guidelines).
2. Add the necessary fields to the database (Check 4.2.3 for guidelines).
3. Add the statistic module, responsible for generating the data received from the new plugin (Check 4.2.4 for guidelines).
4. Add the view responsible for showing the live data received while a session is running (Check 4.2.5 for guidelines).

4.2.2 Adding an interpretation module

To add a new interpretation module, all you have to do is add a method to the file *interpretations.py* and name it *pluginname_eventname* [1].

For example, if the plugin name is *heartrate* and the event it is sending is called *offsessionreading*, then the code will be like in the code snippet 4.2.2

4.2.3 Adding fields to the database

The main database file that contains all tables and fields is called *models.py* and is located in the *fitnessmonitor* directory. For example, to add a table called *moves* with the fields *name*, *count* and *trainee* (a pointer to a practitioner), check the code snippet 4.2.3. After any change in the database, the command 4.2.3 must be executed from the shell.

```
python manage.py syncdb
```

[1] The plugin name is the name given to the plugin when it was first configured and the event name is the name of event sent (”start session”, ”end session”,etc...)

Listing 4.2: Detect Method

```
class Moves(models.Model):
    trainee = models.ForeignKey(Practitioner)
    name = models.CharField(max_length=128)
    count = models.IntegerField(null=True, blank=True,default=0)
```

Listing 4.3: Detect Method

```
def heartrate_offsessionreading:
        # insert code for the statistics generating here.
```

4.2.4 Modifying the statistics engine

To add a new plugin's data to the statistics engine, all you have to do is add a method to the file *statistics.py* and name it *pluginname_eventname*.

For example, if the plugin name is *heartrate* and the event it is sending is called *offsessionreading*, then the code will be like in the code snippet 4.2.4

4.2.5 Adding a view

To add a new view for the plugin, simply add the code to the file *home.html* that is found in *templates* directory (depending on the type of plugin, a chart, graph, image or text can be displayed).

4.3 Making sure a plugin is ready

To make sure a plugin is ready to be used, the following steps should be done.

1. The plugin should be registered to an empty port.
2. A hand shake should be done to initiate the connection.

Chapter 5

Conclusion

Analyzing the level and studying the performance of a trainee is a very hard task. In fact, getting accurate results without the use of an external tool is almost impossible. The tool created will help the trainers accomplish their tasks easily.

The objective of this project was to deliver an easy-to-use tool that could be easily expanded by adding different types of plugins.

Chapter 6

Future Work

In this section, I will list some ideas and expansions to the tools that could be implemented in future versions of the impact project.

6.1 Trainers integration

In this feature, trainers will have their own accounts where they can handle multiple trainees and assign some sessions for them. They can monitor the level of their trainees, as well as giving feedback regarding their performance.

6.2 Social training platform

The idea of this expansion is that the tool could be an online social platform where practitioners share their sessions and experience with other fellow practitioners.

6.3 Gamification

This will add a competition-like environment to the trainees by having levels (amateur, expert,etc..), competitions between them and other features.

6.4 Plugins

There are many great ideas for plugins that will help users.

Heartbeat rate sensor The heartbeat rate sensor could be used during the round or off-session to measure the heartbeat rate of the trainees to give an image of their fitness level.

Leap Motion sensor [1] This sensor is a sensor that looks like the xbox Kinect sensor. It will help get more accurate view of the position of the trainee during the session.

[1] Leap motion main site https://www.leapmotion.com/

Appendix

Appendix A

Lists

List of Figures