

Junior Honours Project



University of
St Andrews

Timothy Austen [ta22]
Elliot Davies [ed37]
Andrew McCallum [am266]
Maxim Raykhrud [mr469]

Supervisor: Alexander Voss

Date of submission: 2014-05-06

Abstract

The aim of the project was to create a sports training and analysis system that would help improve the fitness and technique of the user. We, Group G, took on the responsibility of creating a consistent graphic user interface across both the website and the Android app for the system. We were also responsible for creating the website and integrating it with other parts of the system. Finally, we were the group designated to create visualisations of the data captured and processed by the system.

Declaration

We declare that the material submitted for assessment is our own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 12,073 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, we give permission for it to be made available for use in accordance with the regulations of the University Library. We also give permission for the report to be made available on the World Wide Web.

We give permission for this work to be used in research within the University of St Andrews, and for any software to be released on an open source basis.

We retain the copyright in this work, and ownership of any resulting intellectual property.

Contents

0. Introduction

0.1. Background	6
0.2. STACS	6
0.3. Group G	6

1. Project details

1.1. Design process	7	
<u>1.1.1. Collaboration with other groups</u>	7	
<u>1.1.2. Design decisions</u>	7	
1.2. Development		8
<u>1.2.1. Technologies used</u>	8	
<u>1.2.2. Process and communication</u>	11	
<u>1.2.3. UI/UX design decisions</u>	12	
1.3. System overview	13	
<u>1.3.1. Data flow</u>	13	
<u>1.3.2. Android application</u>	15	
<u>1.3.3. Web application</u>	16	

2. Evaluation and critical appraisal

2.1. Comparison to original specifications		30
<u>2.1.1. Website</u>	30	
<u>2.1.2. Android app</u>	32	
<u>2.1.3. Data visualisations</u>	34	
<u>2.1.4. Overall UI design</u>	35	
2.2. User interface and user experience	35	
2.3. Integration	36	

<u>2.3.1. Group A (Rankings)</u>	37
<u>2.3.2. Group B (SmartCoach)</u>	37
<u>2.3.3. Group C (Social)</u>	37
<u>2.3.4. Group D (Data acquisition)</u>	38
<u>2.3.5. Group E (Database)</u>	38
<u>2.3.6. Group F (Wireframe)</u>	39
2.4. Comparison with similar systems	39
<u>2.4.1. Endomondo</u>	39
<u>2.4.2. MapMyFitness and its satellite applications</u>	40
<u>2.4.3. Strava</u>	41
<u>2.4.4. Fitbit</u>	41
<u>2.4.5. Nike+</u>	42
<u>2.4.6. Moves</u>	43
3. Conclusions	
3.1. Learning outcomes	44
<u>3.1.1. Tools and technologies</u>	44
<u>3.1.2. Communication and management</u>	44
3.2. Future work and potential improvements	45
<u>3.2.1. User interface and user experience design</u>	45
<u>3.2.2. Data visualisation</u>	46
<u>3.2.3. Extensibility</u>	46
3.3. Concluding statements	46
4. Testing summary	47
Appendices	48

0. Introduction

0.1. Background

With smartphones and mobile applications now all but ubiquitous, sports training apps such as Nike+ and Endomondo are becoming ever more popular. Mobile devices are small and easy to carry around, making them ideal for recording data on the go, while their increasingly powerful sensor arrays can record any and all information that a user might want to analyse.

The aim of this Junior Honours Project was to create such a sports training system - ideally one that offered features no other application did.

0.2. STACS

After some deliberation and discussion, the class decided to produce the Sports Training and Comparison System (STACS). The goal was to create a system that could be used to improve fitness, through a 'SmartCoach' program, and sporting technique, through video wireframe analysis of a user's posture.

STACS would consist of an Android application, which the user could take with them when training; a Windows desktop application for the Kinect, which could be used to record and process footage; and a website, which could display historical information and incorporate social aspects.

0.3. Group G

Our group was responsible for creating the system's website, ensuring a consistent design and user experience across the Android app and website, and producing visualisations of statistical data to make it accessible.

We were largely successful in these endeavours. We have created a website, available at <http://jhproj05.cs.st-andrews.ac.uk/web/app/>, which displays a user's historical data as collected and processed by the SmartCoach in the Android app. The user experience is generally consistent throughout the system, other than some aesthetic differences. And we have created and included various types of visualisation.

This report will now detail our group's work, including the tools used and an overview of the finished product, before critically evaluating the system against its original requirements as well as its competitors. We will also discuss our findings and conclusions.

1. Project details

We will first discuss the design and development process by which the system's requirements and features came about, and give an overview of the resulting product.

1.1. Design process

1.1.1. Collaboration with other groups

To begin with we held a class-wide meeting where we discussed the project as a whole. Each team put forward several requirements that they thought would suit the specification, and then the class decided which to try to implement. Each group then bid for the area of the project they most wanted to work on, after which we were designated as the user interface / user experience group.

This brought with it its own set of issues; while as a group we had experience developing web applications and user interfaces, being in charge of integrating each group's work into a web application also meant that we would be relying heavily on other groups finishing their work on time. If everyone worked up until the deadline, it was unlikely that we would be able to successfully include everything.

In an effort to negate the impact of this, we tried to schedule regular meetings with most groups. Although schedules often clashed, the basic requirement specification from each team let us see where our work may take us and what was likely to be involved. Most groups also gave us design documents outlining what they needed but, as might be expected, these were in a state of constant flux.

1.1.2. Design decisions

As we were the 'front-end' team, we wanted to make sure to keep the client and server as distinct as possible. Initially we had planned to host our own web server, which would then have made calls to the database when the client requested it. After consultation with the database team (Group E), however, we opted to host our website on their custom server, which simplified matters. As that group was handling the server-side scripting, we began looking at ways to make the client-side more interesting without needing to be tightly bound to the server.

As we all had previously experience with developing web applications, we wanted to try something we hadn't done before. After some deliberation we decided that we would create a single page application (SPA) - that is, one which uses AJAX to make sure all new pages within the site are accessible without further calls to the server.

SPAs offer advantages in terms of responsiveness, as all the HTML pages and JavaScript files are served on the initial page load and then simply displayed as needed. Additionally,

because held variables are usually erased on a page refresh, SPAs allow variables to remain intact throughout a user's visit to the site.

The main draw of an SPA website was the complete decoupling of the client-side code from the host server. This meant that any changes we wanted to make did not require access to server-side files, and we could play with and explore potential UI designs without worrying about needing permissions for Group E's virtual machine.

Client-side computation also reduces load on the server should any intense calculation (e.g. for graph visualisations) need to be done. As we were planning a collection of visualisations for various areas on the site, we wanted to ensure we wouldn't be straining the server too much in the event that several users tried to load multiple graphs at once - especially as the database runs on the same virtual machine as the web host.

1.2. Development

1.2.1. Technologies used

1.2.1.1. AngularJS

As we were developing a single page application we needed to research the various technologies available to implement one. As a group we had most frequently used regular Asynchronous JavaScript and XML (AJAX), but based on our previous experience developing with it we decided that writing an application of this size, without some form of abstraction away from XML HTTP requests, would not be a pleasant affair.

Our aim for the project was to learn about things we might not otherwise have had the chance to, so we opted to use a very different tool. We initially considered backbone.js, but chose not to use it (despite good support for REST APIs, which the database team were planning to offer) because other frameworks offered more consistent, inclusive support for various features. Eventually, we settled on AngularJS.

AngularJS is an open-source JavaScript framework maintained by Google. [AngularJS, 2014] It allows developers to relatively easily implement systems based on the Model-View-Controller (MVC) pattern, which helps separate DOM manipulation from business logic. Although we were sceptical of the torrential influx of new JavaScript libraries and frameworks (not all of which are sustained or maintained to a level that makes them suitable for use in anything other than hobby projects), AngularJS is reasonably well documented in most cases and has a large repository of support material available online.

One of the framework's goals is to increase the testability of code used in client-side web applications. It offers integration with the Karma JavaScript testing framework (also developed by the AngularJS team), which in turn can be run on Travis CI, the continuous

integration system we as a class had chosen to use. Karma tests are usually written in Jasmine, a declarative testing language for JavaScript.

The main aim of AngularJS, however, is to allow HTML to be more dynamic than it naturally is but without the mess of code that comes with programming using AJAX. This cleanliness is implemented through several abstractions:

- **Directives** are custom HTML tags that offer powerful functionality. Instead of manually altering the DOM to change the length of a dynamic list (e.g. for workouts a user has completed), for example, the 'ng-repeat' directive iterates through a collection of objects taken from the model, outputting a new list item for each.
- DOM elements can be assigned a **controller** - this controller has its own scope (i.e. variables and functions), and Angular's two-way data binding means that changes in the controller are automatically reflected on the user's webpage.
- Much of the application logic is hidden away in **services** - these can be accessed from controllers using Angular's dependency injection, and can offer any service required such as login, logout, get user data, load workouts, etc.

1.2.1.2. HTML + CSS

Use of HTML and CSS for the website's front end was the only real modern choice we had. While we could have tried to use Flash or Silverlight to create the UI for the service, it would have been a very limited and power-consuming setup.

Our web application made extensive use of the features offered by CSS3, including CSS transitions for animating UI elements, rounded corners, gradient backgrounds, background opacity, and others. In addition, some parts of the system, including those created by other groups, utilise HTML5 functionality, such as HTML Canvas.

1.2.1.3. D3 and SVG

Part of our group's remit was to produce visualisations of data provided by the other groups. For this we chose to use D3 (short for 'Data-Driven Documents'), a modern JavaScript library for "manipulating documents based on data". [D3, n/a]

D3 makes it simple to produce both dynamic and static graphs, charts and other visualisations based on (potentially changing) data sources. The concept is similar to that of AngularJS - elements on a web page (in this case visualisations) are bound to data, and these elements update automatically when the data does.

We chose D3 over other technologies because it is one of the most popular visualisation libraries in use today, with excellent documentation and plenty of examples. In addition

we were familiar with it, which meant we could spend more time thinking about designing good visualisations and less time learning how to get started. Finally, the library works well with JSON and is able to parse it from strings; this was useful because almost all the data in the database was stored in JSON format.

Although the visualisations themselves could be implemented several ways, the natural choice was through the use of SVG shape, path and text elements. D3 has inbuilt support for creating and manipulating SVG and the technology is well-supported by all browsers.

1.2.1.4. Github

For a project of this magnitude a version control system was a necessity. Even with one person, being able to rollback and save older versions during a project is helpful; in this case, the version control was needed for the ability to distribute and merge code across not only our team but the whole class.

We decided that it would be easiest if the entire class used the same system. To decide what to use, we looked at the two version control systems we were familiar with: Mercurial (hosted on the School servers) or Git (hosted on Github). For each system we looked at its ability to connect with a continuous integration system, its support for issue tracking, and how easy it was to set up and use on a day to day basis.

Mercurial has good support for Jenkins, but Jenkins is mainly designed for Java development environments and as a class we would not be using Java very much. Github meanwhile is easy to integrate with Travis CI [Github, 2014 (2)], which supports many languages including those we were going to use.

For issue tracking, Mercurial requires an additional plugin to be installed on top of the repository server and this would have been tricky to set up within the School. Github has an integrated issue tracking system that allows users to leave comments, which sends notifications to the relevant team both on the Github system and via email.

To set up Mercurial as the class wanted (to allow for issue tracking and easy access) would have been challenging. To do the same with Github was relatively easy: Github comes with a simple GUI for most platforms [Github, 2014 (1)] and code can be edited from the website without having to clone the repository, allowing for easy remote updates.

We also considered the security and practical implications of using an external system - that is, one hosted outwith the School - and how familiar members of the class were already with each option. After comparing the two, the class decided upon Github for simplicity and ease of use, as well as the availability of free private repositories for academic purposes.

1.2.1.5. Node.js

For our SPA website we required a server on which to host the pages and perform any back-end processing that we may need. We considered using the Apache system already set up within the School, but it would not have been able to perform any background processing or routing without modification to the School's set-up. We also looked at using a Django server, but eventually settled on Node.js after our supervisor suggested that we ought to keep the number of languages in the project to a minimum.

Node.js is a scalable, network application system created on top of Chrome's JavaScript runtime environment [node.js, 2014]. A Node server can easily be created using a non-blocking event-driven Javascript model, giving us the ability to create multiple servers so as to perform different processing tasks.

We therefore began to use Node.js as our server, but we ran into issues with the same-origin policy that Javascript enforces. For this and other reasons (as mentioned previously) we eventually decided to move our website onto the same server that hosted the database. This was an Nginx server that we could update by accessing a particular URL provided by the database team, which pulled from the master branch of our Github repository.

1.2.1.6. Java

We used Java for the work we did on the Android application, since the application was built using native Android libraries (as opposed to using a web stack in a wrapper or platform-agnostic development tools).

1.2.2. Process and communication

As decided upon during on the design phase, each of us had a team to liaise with (see 'System overview', below). This contact was maintained throughout the year. We would keep updated with what our respective groups were doing so that we could work out how to integrate their systems into the website once they were finished. The communication was mainly handled through email or instant messaging systems, though we did hold meetings in person where we could.

Holding whole-class coding sessions was beneficial as it made system integration and communication much easier, as well as speeding up development. Finding a time for the whole class to meet was challenging, however, with each team and indeed each person having their own commitments and priorities. We also held group coding sessions, which had the same benefits but were more focused towards specific group goals.

1.2.3. UI/UX design decisions

To provide a nice, coherent user experience, we tried to make the look and feel of the website and the mobile application as consistent as possible, organise the data in various pages and screens in a clear and concise manner, and to make all of the above as visually pleasant as possible.

1.2.3.1. Website

As the front end for various components, the website had to provide easy access to numerous modules of the system, organised in a way that would be easy to understand and work with.

1.2.3.1.1. Organising the content

Initially we decided to split the content into different pages based on the groups and functional modules it came from, giving us separate sections for “Technique analysis”, “Fitness” and “Statistics”. This proved to be a bad decision from the user’s viewpoint, however, as it made the system awkward to use.

We therefore focused instead on splitting the data according to whether it was used for general fitness (the “Fitness” section) or one of the sports in the system (the various sports pages, represented by “Boxing”, “Cricket” and “Running” in the final system).

1.2.3.1.2. Graphical design

In terms of graphical design, we wanted the website to be minimal and content-first, but with a very coherent aesthetic. This motivated us to use white backgrounds, simple grids and a semi-transparent navigation bar.

The colour of the navigation bar menu and active elements was chosen based on the following criteria:

- It had to be visually distinct from all the other colours that could/would be used by the system.
- It had to have distinct sports connotations.

This resulted in the current neon green being used.

Font choice was an important matter too, with legibility and legal availability being the two main factors to consider. Our initial choice was Proxima Nova, a hybrid humanist / geometric sans-serif font. It was carefully selected from a pool including Google’s Roboto font (used as the system font on Android, a “frankenfont” [Typographica.org, 2011] combining numerous stylistically distinct elements), Akzidenz-Grotesk (a traditional

grotesque font designed in 1896, considered by many to be a precursor of Helvetica [MyFonts, 2014 (2)] and Acrom (a modern geometric sans-serif [MyFonts, 2014 (1)]).

The original choice of Proxima Nova was replaced by Lato (a free Google Fonts font [Google Fonts, 2014]), however, owing to the former's steep licensing costs. The two fonts express a great level of similitude, but the latter is free.

1.2.3.2. Android application

The mobile application was a pivotal part of the system, being the user's only tool for data acquisition, live training and live sensor data monitoring. It needed to be easy to use without any practice.

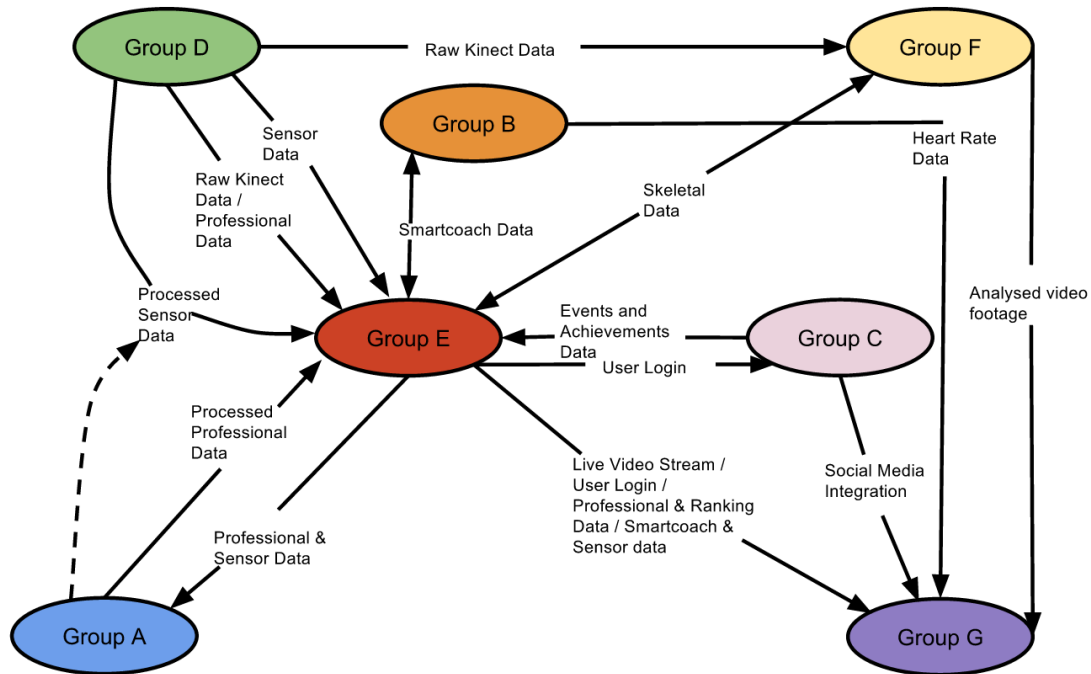
Originally, all the UI and UX were meant to be planned in full by our group, but this proved to make the development process difficult. Although we did some pre-development planning with the SmartCoach group, creating several detailed screen flows and mockups for the workout screens, the workouts were then redone, essentially making all of our work obsolete. Instead of designing the application from the start, then, we had to focus on bringing all of its cosmetic elements into line after they were built.

We created a colour scheme that was a continuation of the website's, worked on the application icon, and modified most of the fragment layouts of the application based on simple rules such as Fitts' Law and complex requirement frameworks like Google's Android Design Guidelines [Android Developers, n/a].

1.3. System overview

1.3.1. Data flow

The flow of data through the system is illustrated by this diagram (previously shown during the project demonstration):



As shown here, the data collection and processing is handled by the other groups before the information is stored in the database (Group E). Our group then retrieves and displays it. With the exception of the heart rate data in the Android app, which is visualised directly, this means our only point of contact for data is with the database API, which keeps the system architecture simple from our viewpoint.

1.3.1.1. Liaising with other groups

While all the data to be displayed on the website is retrieved from the database, we did work directly with some of the groups to ensure their work could be included in the website or app. To manage this, each of us took on responsibility for liaising with a particular group (though, inevitably, we each contributed somewhat to every part of the system).

Maxim worked with the SmartCoach group (Group B), who were also responsible for building the Android app, to monitor the design and user experience of both the app and the website.

Andrew worked with Group C on the social media integration, achievements, and other social features.

Elliot worked with the wireframe group (Group F) to embed the wireframe visualisations on the website and create graphs of relevant data.

Tim worked with Group A on the system's rankings and fantasy team features, and also had a hand in the wireframe visualisation graphs.

1.3.2. Android application

At early meetings one of our responsibilities was the creation of the whole Android application but, as the project progressed and more concrete requirements were defined for the system and integration of the components, it was agreed that this responsibility ought to shift to Group B. We then focused strictly on the application's UX and UI.

1.3.2.1. Application capabilities

The application is designed for versions of Android starting from 4.0 and is based on standard Android development libraries, with some external ones used where necessary. The application consists of the following modules:

1.3.2.1.1. Profile page

A screen that is meant to display user's profile picture, their information and achievements.

1.3.2.1.2. Heart rate monitor

A screen that displays real-time heart rate data coming from the connected Bluetooth device and visualises it as the screen background, depending on what 'heart rate zone' the user's heart rate is in currently.

1.3.2.1.3. Workout plan and past workouts

Workouts are an essential part of the application and are one of its main selling points.

They are sets of exercises designed to test the user's sporting abilities in multiple disciplines. After going through a test workout, the user gets assigned exercises that closely match his or her ability.

Exercise screens are specifically designed for the particular sports or fitness activities they are supposed to control. For example, the cycling screen shows real-time heart rate data, while the pressups screen is just one oversized button that is meant to be pushed by the user's nose.

1.3.2.2. Our input

Probably our single biggest contribution to the Android app was the screen for the heart rate monitor functionality, which was both designed and had its most important

visualisation functionality (background colour changing depending on the heart rate zone) implemented by our group members.

Apart from that, our contribution involved maintaining (or trying to maintain) uniform styling across the entirety of the app. This was difficult as it was put together by various other groups, each responsible for small parts and all with different style ideas.

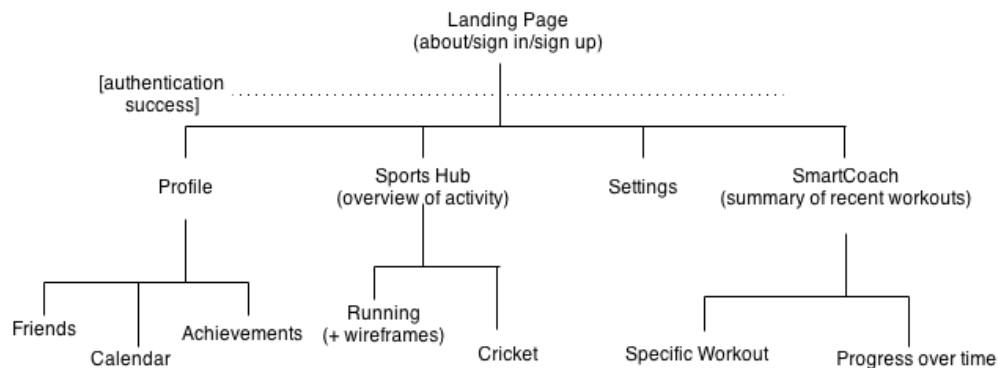
We also provided the application icon and the colour scheme.

1.3.3. Web application

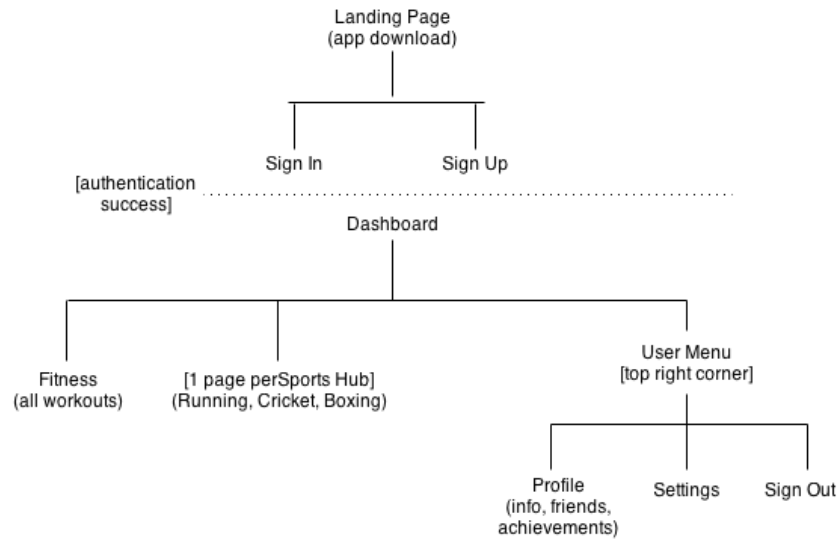
1.3.3.1. Overview

Our web application aims to provide a hub from which the user could view and analyse their progress from SmartCoach, while offering social features to allow sharing of achievements and results. The overall page flow went through several iterations, each differing due to the changing decisions made by and requirements set by other groups.

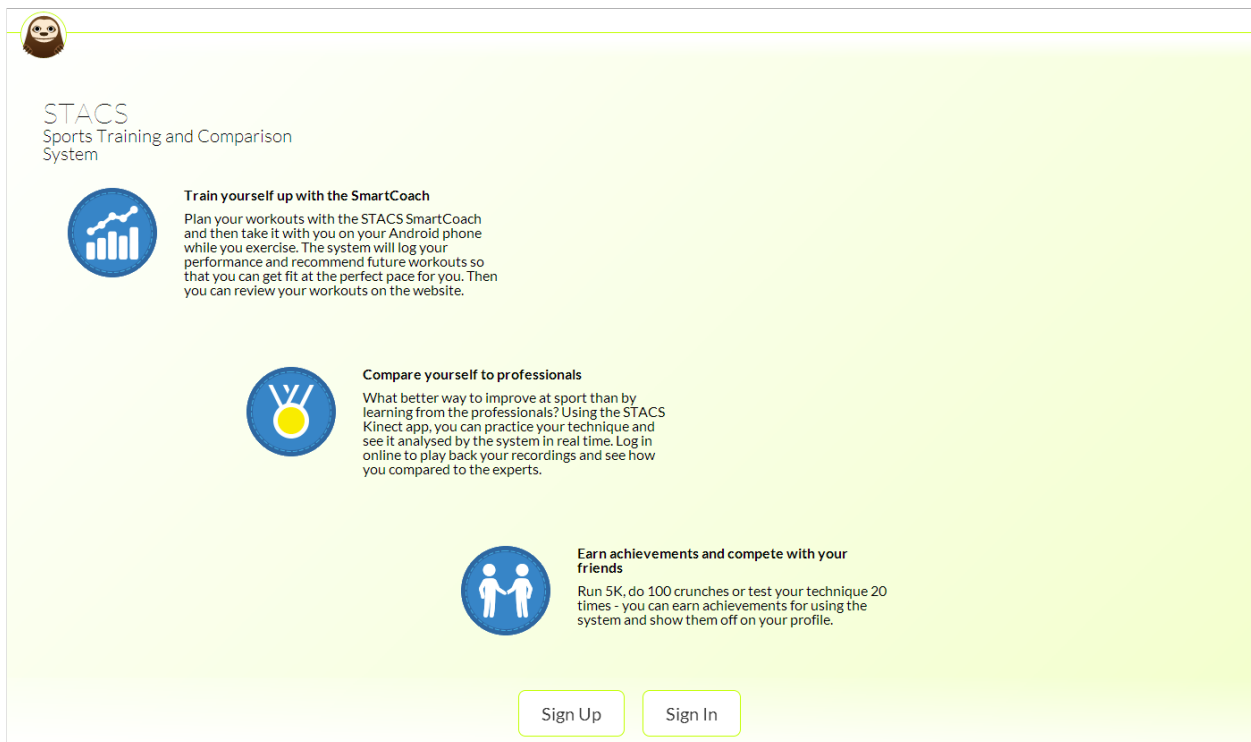
Originally, users would have a single ‘SmartCoach’ page, along with a single hub for all sports. From here, users would be able to move into other areas, e.g. pages for individual workouts or for specific sports.



Over time, this layout changed, and we reorganised the site so that each sport had its own hub accessible from the top menu (a feature missing in initial sketches), while the SmartCoach section was renamed to ‘Fitness’ in an effort to make its purpose more clear. In the final website, all workouts relevant to a section (e.g. ‘Running’) are displayed on one page, and can be expanded or contracted by the user. Some sections have been consolidated to bring together relevant information into single pages, and also to reduce the number of partial views required for the system. The final simplified page flow is as follows:

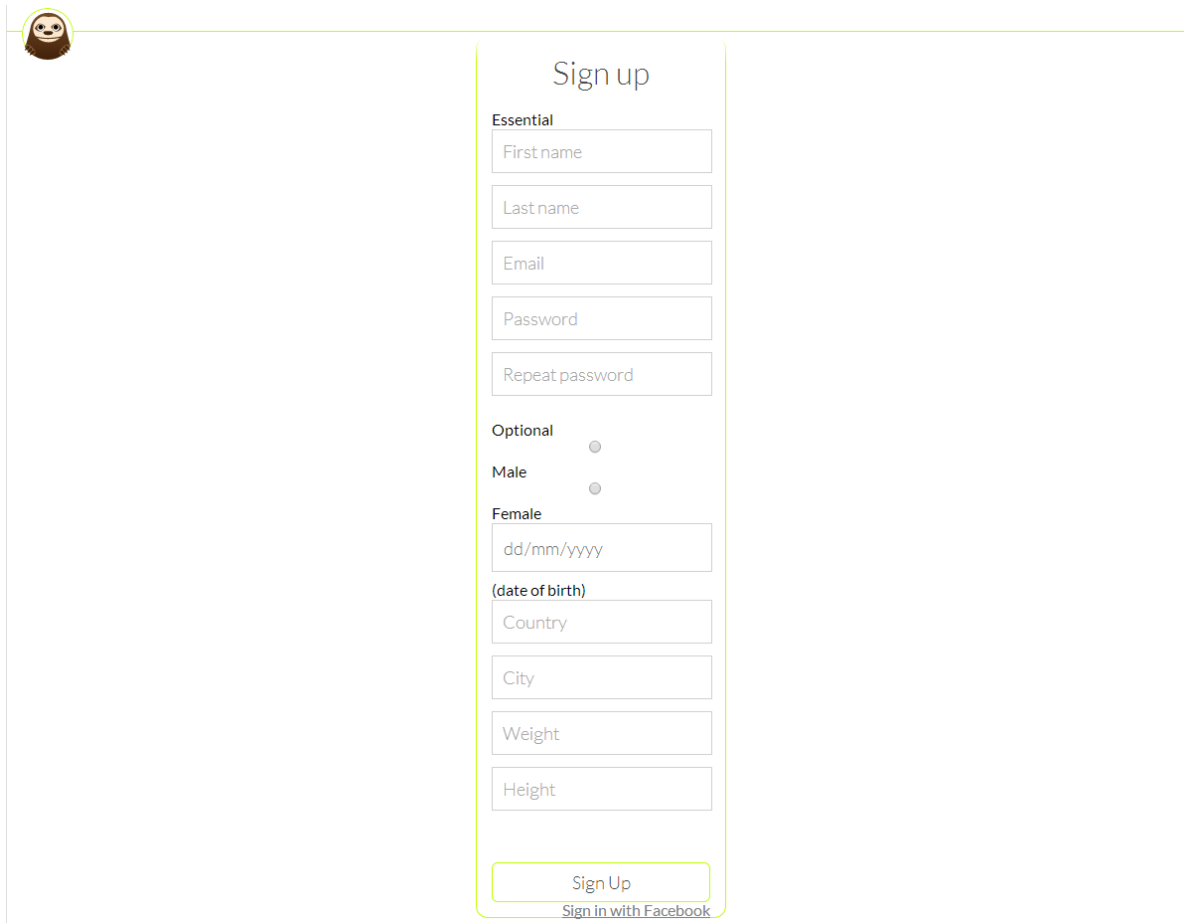


1.3.3.2. Landing page



The landing page is what users see when they visit the website for the first time. It provides information about STACS' features and generally acts as a way to 'sell' the system. It also provides the sign-up and sign-in functionality that a user would naturally expect to find. The landing page was entirely designed and produced by our group.

1.3.3.3. Sign-up / sign-in



The screenshot shows a web page with a sign-up form. In the top left corner, there is a small circular icon of a monkey's face. The form is titled "Sign up" and is enclosed in a light green border. It is divided into two sections: "Essential" and "Optional".

Essential

- First name
- Last name
- Email
- Password
- Repeat password

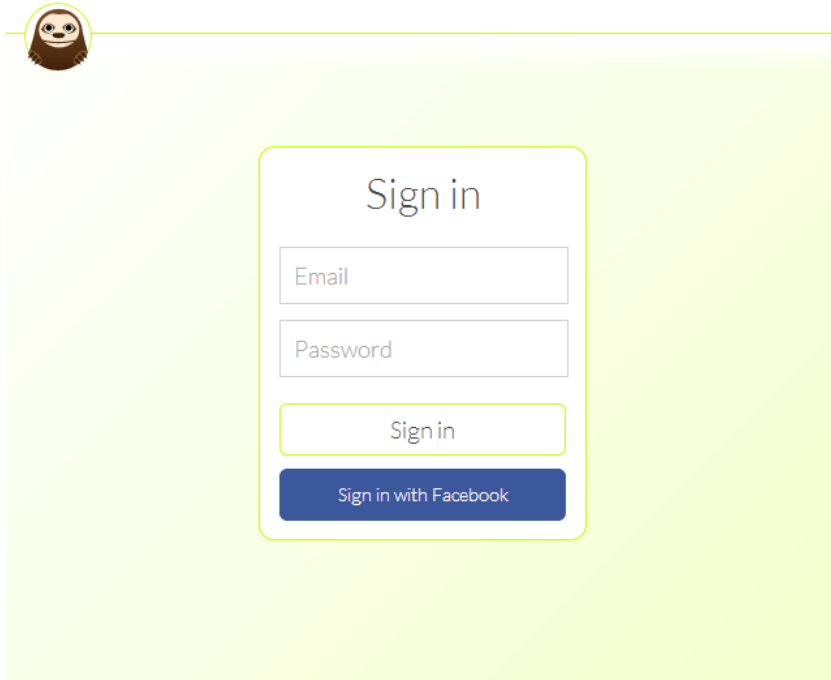
Optional

- Gender selection: "Male" and "Female" with radio buttons. The "Female" radio button is selected.
- Date of birth: A text input field with the placeholder "dd/mm/yyyy" and the label "(date of birth)".
- Country
- City
- Weight
- Height

At the bottom of the form, there is a "Sign Up" button and a link that says "Sign in with Facebook".

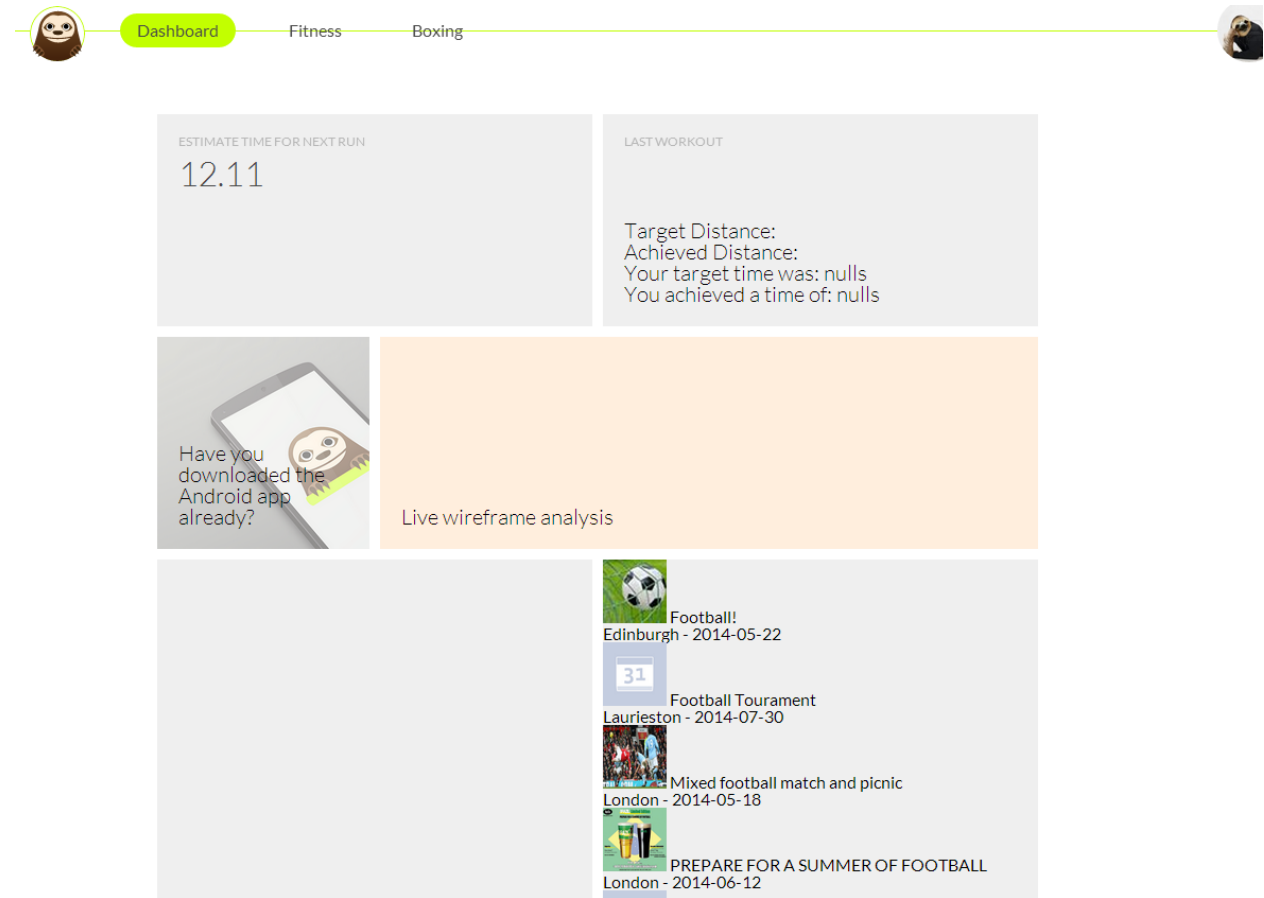
The sign-up page in the final system is non-functional. While it would not take long to properly implement, and while a necessity for a consumer-ready system, it slipped through the cracks and was forgotten about until just before the end of development. As all groups had been adding test users directly to the database as needed, allowing a real user to actually sign up on the website had not been at the forefront of our minds. It is possible to sign up through the Android app, but the functionality ought to be available on the website as well.

Behind the scenes, the sign-up page was supposed to be handled by the 'SignUpCtrl' controller, which disappeared in the last minute flurry of commits and merges. The 'SignUpService' that the controller made use of is still within the code, but the call to the database returned an error and was thus commented out during debugging - the state in which it remains.



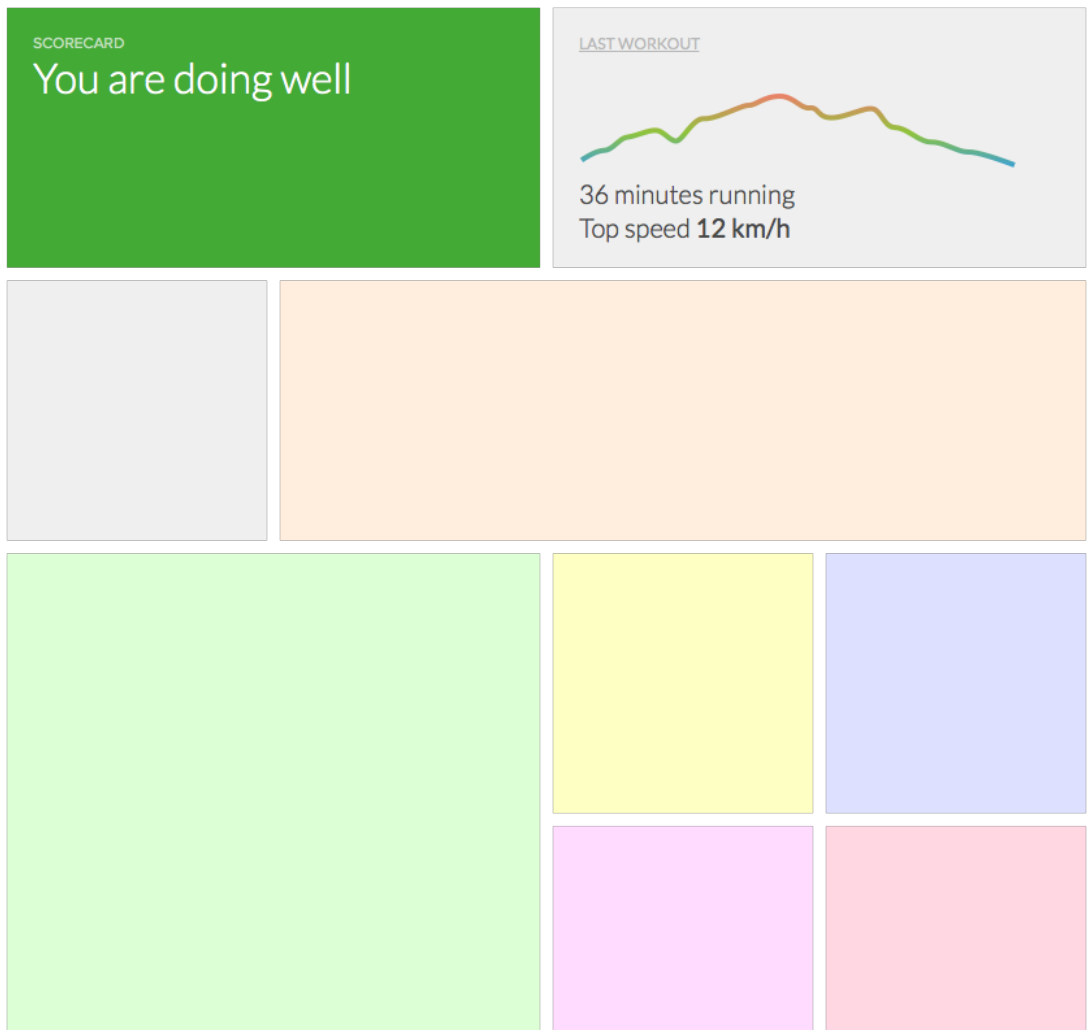
The sign-in page allows users to authenticate and gain access to the main section of the website. Users may sign in with their email and password (if that is how they initially signed up), or by using Facebook. We originally intended to also allow Twitter login, but integrating that code from Group C proved problematic. Once users have signed in they are redirected to their dashboard.

1.3.3.4. Dashboard



The dashboard was envisioned as the home screen for the system, serving as the main control hub for STACS' numerous functional modules. It was meant to display the user's current progression in the various activities he or she may have taken part in, show the latest updates and allow for quick access to non-core functionality of the system.

In its current state the dashboard is fairly limited, especially when compared to its planned appearance, which is shown below.



The original vision included the following elements on this page:

- An overall progress report, combining data from all the analysis done using the Kinect, mobile device sensors, the SmartCoach, etc.
- Details of the last workout with visualisations, coming from the SmartCoach system.
- Elements displaying the status of all the other system components (which were not properly defined at the time of the original design).

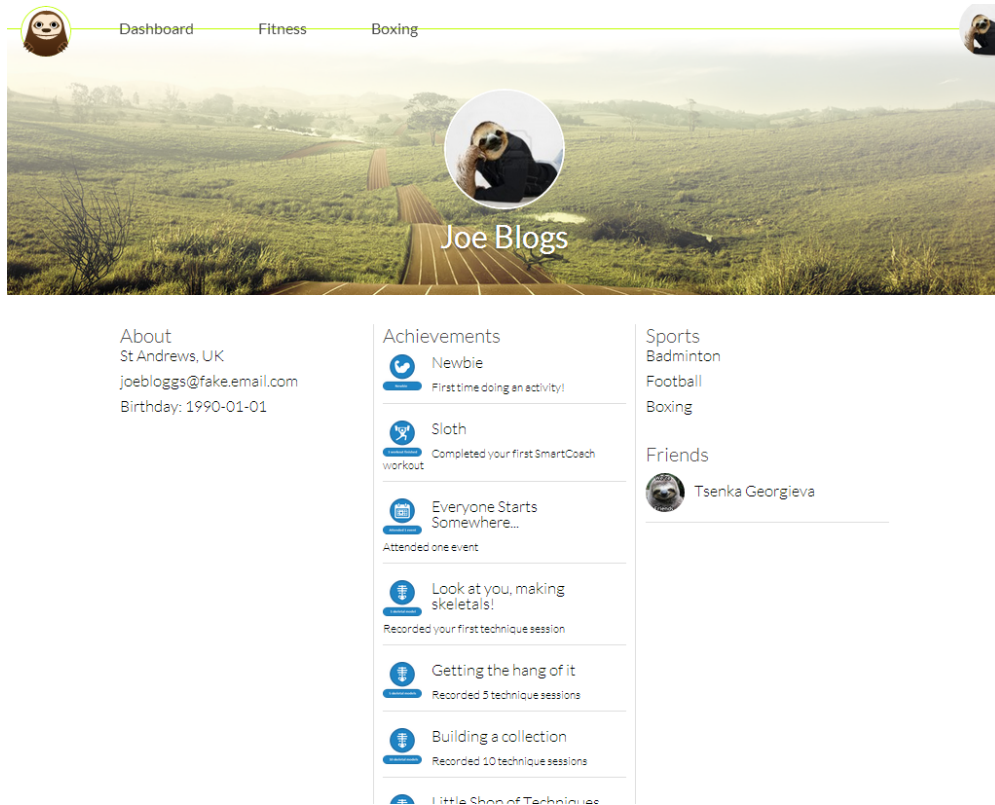
The current version includes the following elements (which, owing to integration problems, sometimes do not work as intended):

- The estimated time for the next run.

- Details of the last workout.
- A banner for the Android app.
- A banner for the live wireframe analysis functionality.
- A list of recommended events.

In its current state the dashboard still functions well as the system’s main hub.

1.3.3.5. Profile



A user’s profile page is comprised of all social-related information pertaining to that user, displaying their profile picture, name, general information, achievements earned, sports in which they are interested, and friends they have added. Note that although Joe Blogs is interested in three sports only one is visible in the top nav-bar; this is because only boxing is supported by STACS (i.e. has a sports hub) - were Joe to register interest in running or cricket, links to them would also appear.

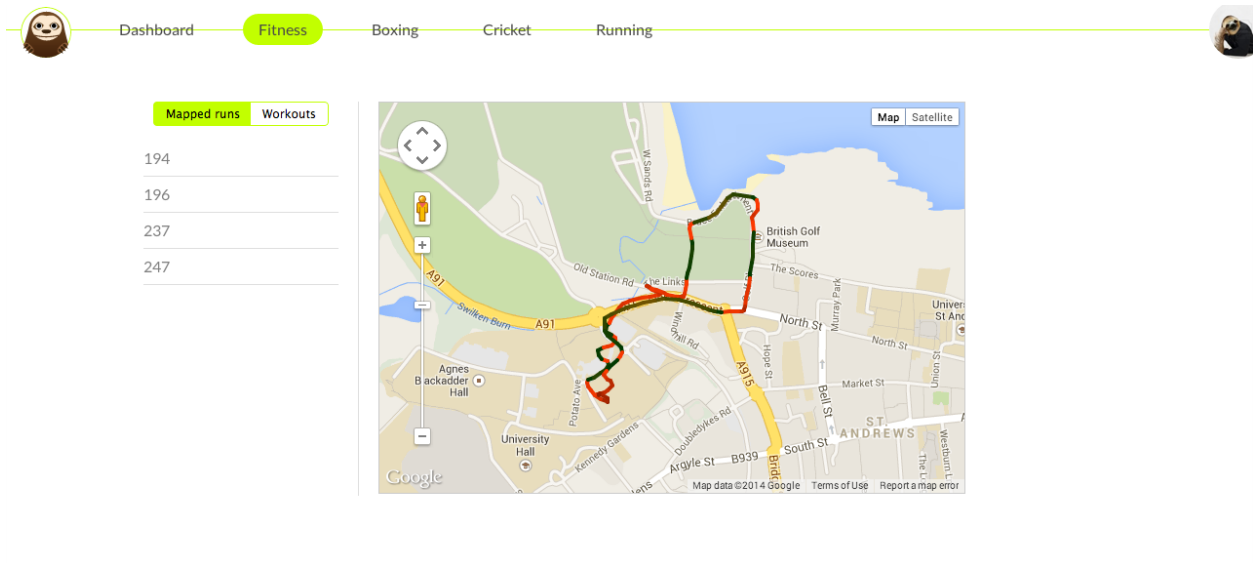
The page displays information generated or gathered by Group C and stored in the database, although it lacks certain functionality (e.g. editing info after sign-up, although information like the user’s location will be taken from a user’s Facebook profile on sign-in). Users may edit the sports they are ‘subscribed to’ in the Settings page, accessible through the ‘user menu’ in the top right.

1.3.3.6. Fitness



The fitness page brings together all of the user's SmartCoach information on the website. It provides details of recent workouts and the exercises they comprise, displaying both the user's target and actual statistics.

So as not to complicate the page, this information is initially hidden: the user can click on a workout to expand it, and then expand individual exercises. This is a simple and effective way to keep the page uncluttered, and using AngularJS made this easy to implement.



The page also displays maps of the user's most recent runs or cycle rides, plotted using the Android app's sensor data in the database. As well as displaying the route the user took (via the stored GPS coordinates) the map is also colour-coded to show the user's speed at any given point. Maps are only shown for those runs that have 15 GPS points or more, in order to filter out those with insufficient data.

The user can toggle between workouts and maps using the buttons on the left side of the page. This feature, which helps keep the page clean, was implemented with AngularJS similarly to the expanding workout data.

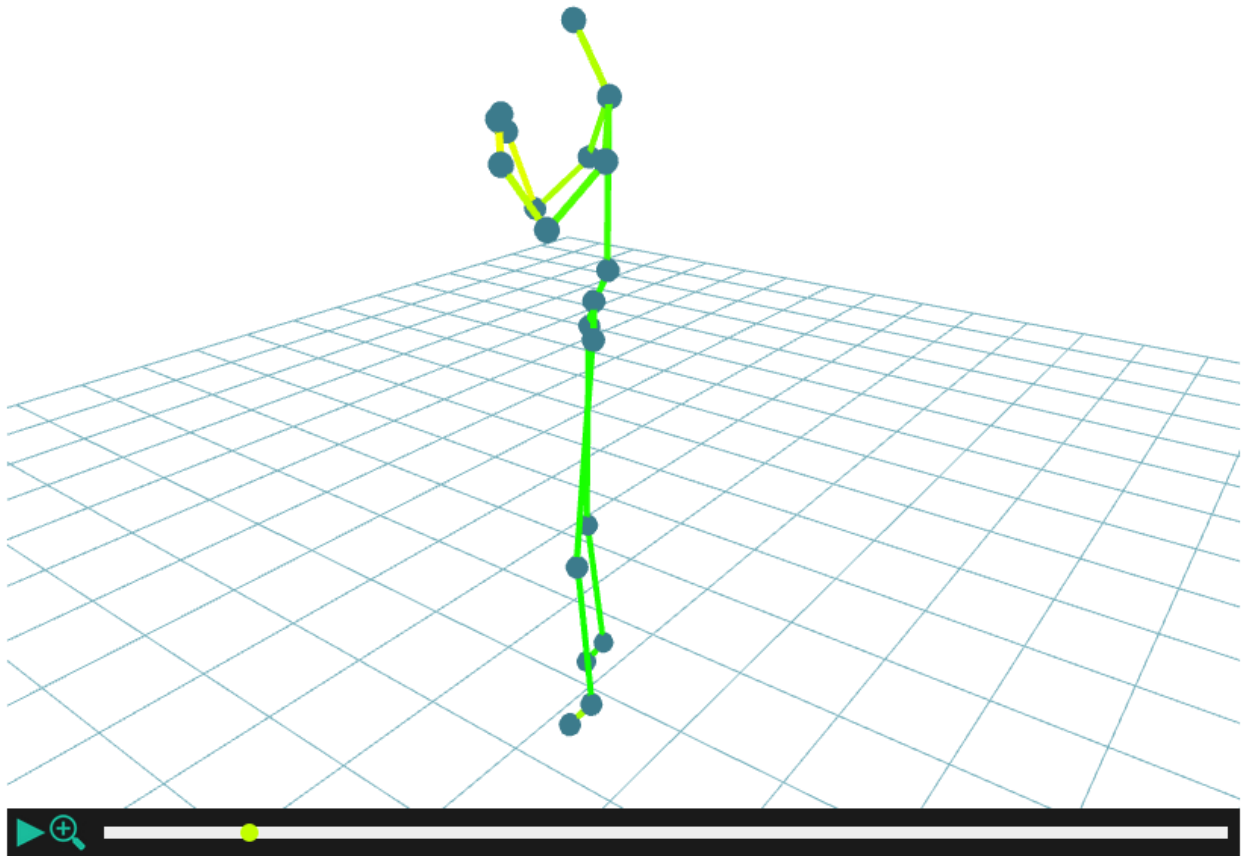
1.3.3.7. Sports hubs

In the later stages of the website's creation, when other groups were starting to complete the data gathering and analysis, it became apparent that the data being gathered would be specific to a particular sport. To handle this we decided it would be easiest to have pages dedicated to the individual sports that STACS would support. The user of the system would not see these pages initially, but would have to 'subscribe' to a particular sport before it would appear in the menu system. We as the GUI team did not specify the sports supported; instead we tried to make the system extensible so that sports could easily be added or removed in future.

The three sports current supported by the system are boxing, cricket and running. These were chosen by the other groups to best demonstrate their individual work: boxing and cricket were chosen by the wireframe team to show off their technique analysis, while running was chosen by the rankings and SmartCoach groups as it worked best with their data.

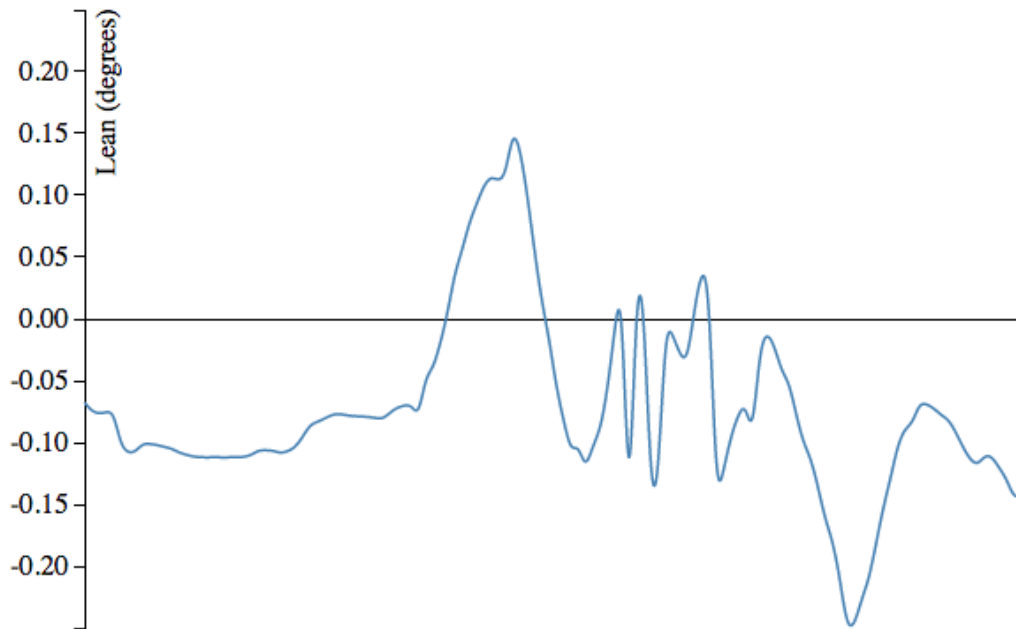
The boxing page contains a list of wireframe IDs that relate to boxing. When the user clicks on one of the IDs it expands to reveal that wireframe, which will play back using an embedded video player provided by Group F. The user can write and save notes about the wireframe. Below this, the page displays graphs (developed by our group) that visualise data from that wireframe analysis.

The wireframe player looks like so:



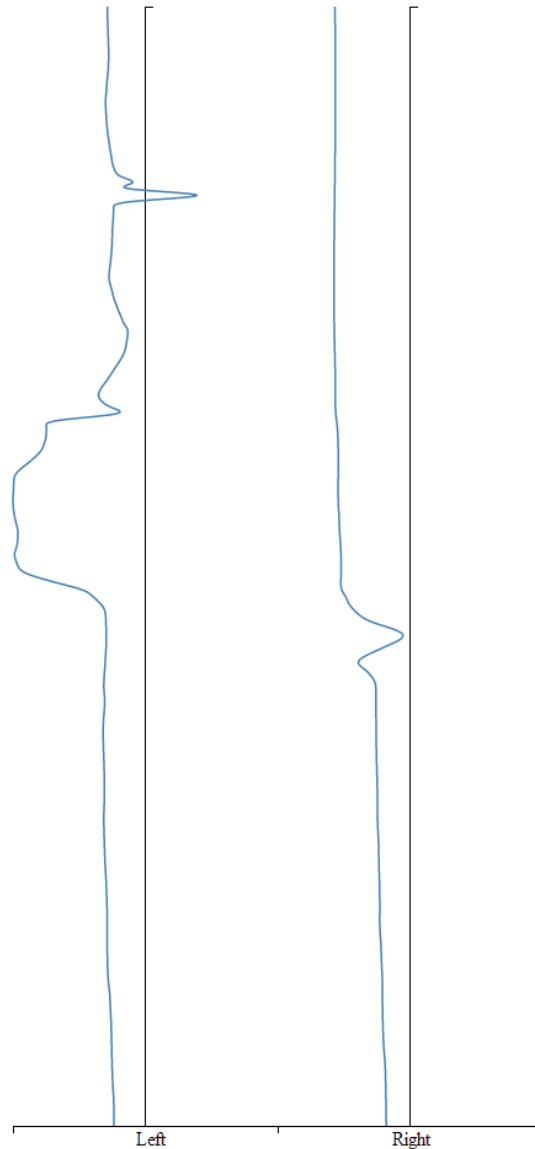
The first graph represents the user's forward and backward lean during the wireframe recording. Time is across the x-axis and progresses left to right. The y-axis represents the lean angle compared with the 'professional' wireframe that the user is being marked against i.e. the axis is a normalised value. The scales change dynamically between graphs depending the maximum lean value for that wireframe and the length of the recording. A positive value indicates a forward lean while a negative value is a backward lean.

The graph looks like so:



Next is a graph that represents the user's side-to-side movement their feet during the recording of the wireframe. In this graph, time is on the y-axis and progress from bottom to top. The x-axis shows the user's difference from the professional. The graph is set up this way to be very visually obvious: if the line of the graph veers left, it means the user's foot moved too far left.

It looks like so:

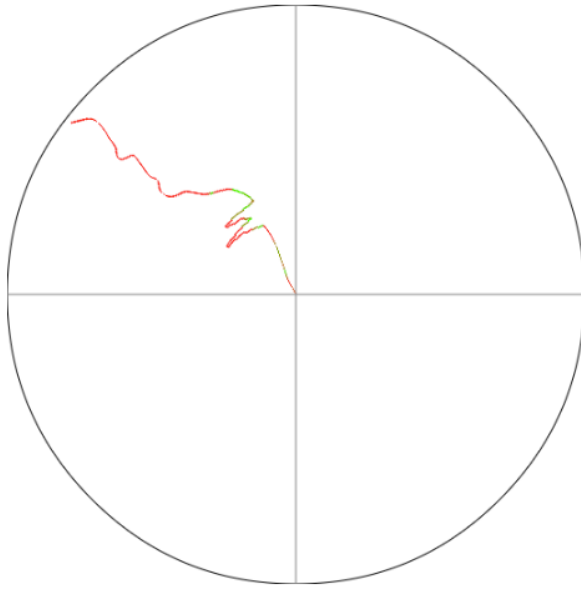


Third is a graph showing the user's left and right foot rotation during the recording. This is a circle graph where time begins in the centre of the circle and progresses outward toward the circumference. The location of a point on the graph represents the angle of the user's foot at that time. The user can thus easily follow how their feet rotated during the course of the recording.

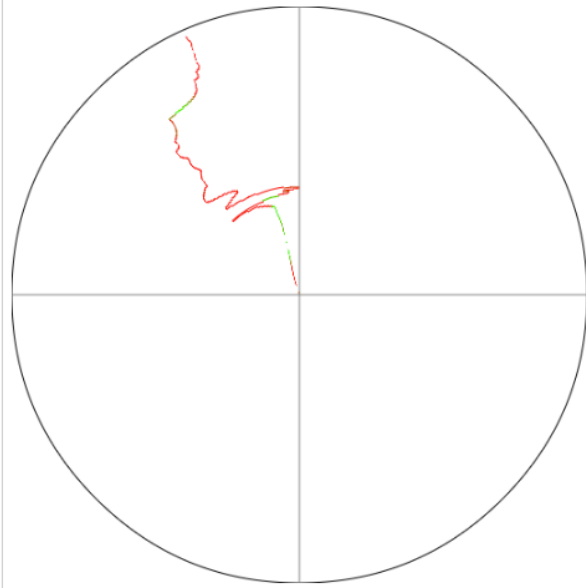
Unlike the two graphs above, this and the following graphs plot absolute values rather than comparisons with a professional. They are therefore coloured to represent the similarity to the professional: green indicates that the user was close to the professional while red suggests they were far away.

The graph of foot rotation looks like so:

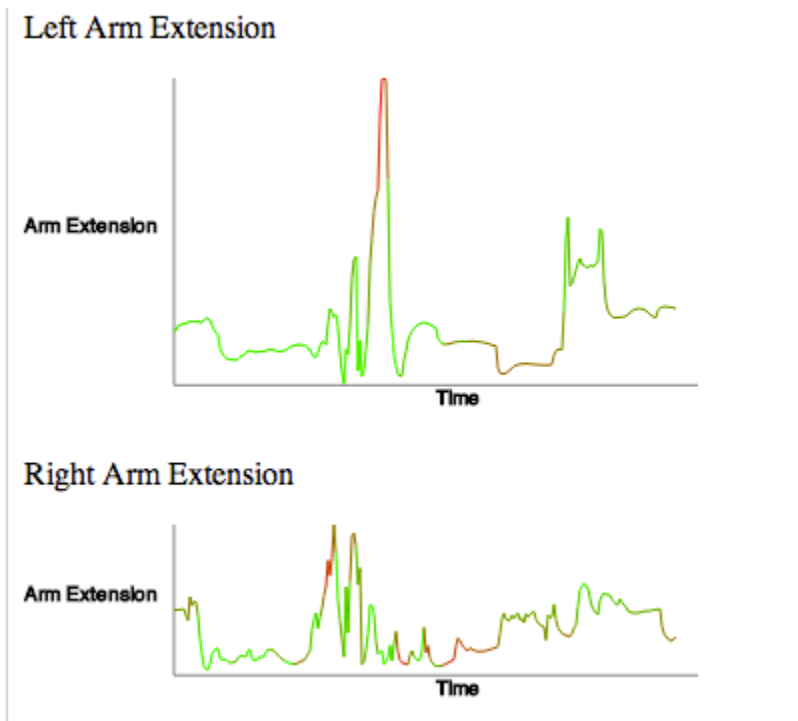
Left Foot Rotation overtime



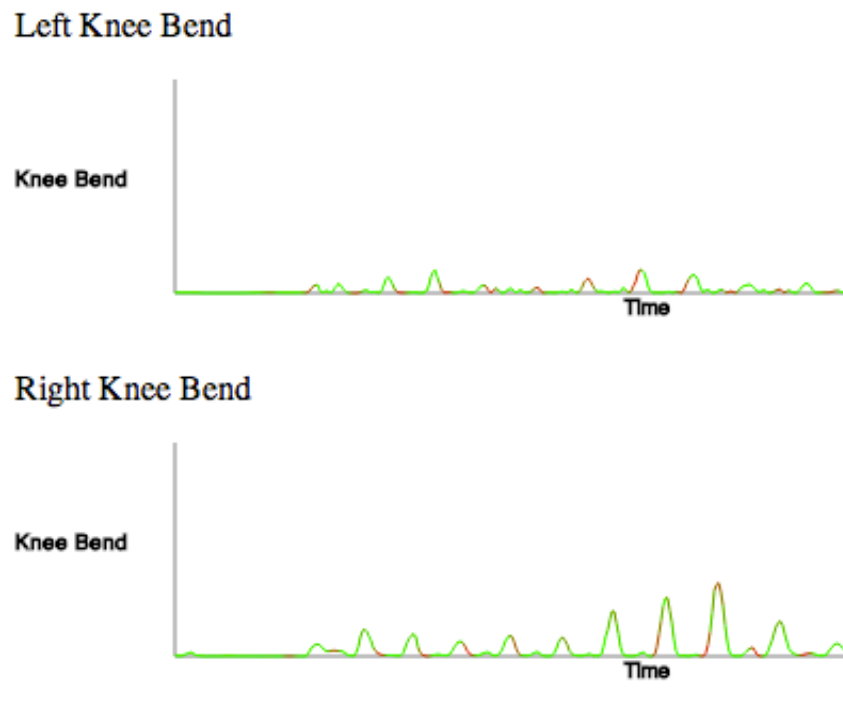
Right Foot Rotation overtime



Next is a graph representing the relative arm extension of the user during the recording. This is a relative graph that works on percentages as the users are unlikely to have the precise same arm length as the professional:



The knee bend graph is the final wireframe graph. This represents the user's relative (percentage) knee bend throughout the course of the recording:



The cricket page contains the same wireframe expanding section as the boxing page. This page also contains ranking information such as ELO scores for the current cricket league and information about the fantasy team implemented by Group A.

The running page contains the same wireframe analysis as the other sports but only contains the knee bend graph, as it was the only graph that applied to the sport. It also shows information about the user's last run times as compared with their friends and other users of the system.

2. Evaluation and critical appraisal

2.1. Comparison to original specifications

The original requirements specification is attached in the appendices.

2.1.1. Website

Functional requirements

- 1.1. - “There shall be a website (‘web app’) that integrates with the system.”

With assistance from most groups we partially achieved this goal. We created a single-page application that has areas for each of the other groups’ sections of the system. We have a fitness section which integrates with the SmartCoach system and displays information gathered from it, as well as the maps displaying the route and speed of the user’s runs.

The sports hubs integrate the ranking and wireframe parts of the system. It displays the rankings data gathered about each sport and the wireframe sessions recorded previously. The ranking system is not fully integrated as the rankings team did not get their system linked to the database correctly (the website currently displays sample data).

There is the live wireframe recording page, which uses the wireframe player and ties into the Kinect app. We also have the profile page, which includes the social media integration system created by Group C. This displays the user information.

All data transfer between sections of the system was performed through the database.

- 1.2. - “The website shall provide functionality for users to log in and hence have a personalised experience.”

We achieved this by creating a sign-in system to access the full website. When logged in, a session key is created and stored. This key allows for the information linked to a user in the database to be accessed by the website for a personalised experience. The menu bar is created according to the information provided by the user: only sports they have subscribed to are shown. While users can always log in, there were minor issues with signing out - sometimes cookies are not properly deleted, and thus users will remain signed in after clicking ‘Sign Out’.

- 1.2.1. - “The website should include user profiles, as proposed by other teams.”

We did create a profile page within the system which shows the user’s information. This includes the user’s location, birthday, email address and the sports they are interested in. This page also contains the achievement system created alongside Group C, showing the progress of the user within the system. The user gets achievements for using the system, such as recording wireframes or running a certain distance.

- 1.2.2. - “The website should include social media integration, as proposed by other teams.”

This was achieved by working with Group C. The user can sign-up or sign-in using Facebook, at which point we scrape some of their Facebook information to populate the database (e.g. their birthday) using JavaScript from Group C. We are also able to use their Facebook profile picture as the profile picture on our system, and populate our system’s friends list. On the dashboard we can display events that are happening near the user’s location.

- 1.3. - “The website shall display visualisations of data pertinent to the logged-in user.”

While there could always be more visualisations, we have fulfilled this requirement. We both created and integrated visualisations into the website, all of which are specific to the user logged in. The first is for a user’s run session in a workout for the SmartCoach system. This visualisation is an embedded Google map which shows the user’s path and speeds during the run. We also create visualisations based on the logged-in user’s technique/wireframe sessions.

- 1.4. - “The website shall display wireframe visualisations pertinent to the logged-in user.”

Our group achieved this requirement in full. Before the user can see any of the wireframe visualisations they are required to log in and generate wireframe data using the Kinect app. We then incorporate the wireframe visualisations created by Group F. This allows the user to play back their previous wireframe sessions. Alongside this player we created graphs to visualise data from the wireframe being shown.

Non-functional requirements

- 1.5. - “The website shall run on an Apache server.”

We did not achieve this requirement as we decided that there was a better solution than using the school’s Apache server for hosting our website (as mentioned above). Instead we moved our website to the same hosting system as the database, which is a server using Nginx. This allowed us to avoid cross-domain issues.

- 1.6. - “The framework to be used for the website will depend on support for other likely features of the system, and is to be decided once these are finalised.”

We achieved this requirement after deciding what the website’s features should be. They are: a functioning SPA website; visualisations; display of SmartCoach sessions; and social integration.

2.1.2. Android app

Functional requirements

- 2.1. “There shall be an app for Android devices that integrates with the system.”

Responsibility for the creation of the the Android app was eventually transferred to Group B because they were primarily working on it, however an integrated app was still created. Group B integrated the data gathering system from Group D to allow for the data analysis in the SmartCoach system, and the data is sent to the database to be stored and used elsewhere.

We provided sections of the Android app, such as the live heart rate monitor. Additionally we worked on the app to give it a design consistent with the rest of the system.

- 2.2. “Similar to the website, the app shall provide functionality for users to log in and hence have a personalised experience.”

A log-in system for the Android app was achieved completely. The user can log into the Android app to create a personalised session. To be able to log in they must have created an account on the system using the sign-up system. Logging in gives access to the achievements that the user has collected as well as their personalised workout sessions, which are based on previous workouts and the user’s information.

- 2.3. - “Similar to the website, the app shall display visualisations of data pertinent to the logged-in user”

We met this requirement minimally: the app includes a screen to visualise the user’s heart rate if they have connected a heart rate monitor.

- 2.4. - “The app shall contain functionality for capturing sensor data and processing data for the SmartCoach system, as proposed by other teams.”

This requirement was met in full, though in the end this work was mostly handled by the SmartCoach team. The app successfully captures sensor data using the libraries provided by Group D and processes it before it is uploaded to the database to be stored.

- 2.4.1. - “The app shall provide real-time feedback when data capture or processing is occurring.”

We met this requirement. The app provides real-time feedback when the SmartCoach is in use, displaying information about the current workout. During a run, for example, the app will display the current distance covered and time taken. During a cycle ride, real-time heart rate data is displayed if the user has connected a heart rate monitor.

There is also a dedicated heart rate screen, which visualises the user's heart rate by changing the colour of the screen background according to the heart rate zone the user is currently in. This is a simple way to provide easy-to-understand feedback to the user, and the use of colour rather than numbers makes the information easier to register at a glance during a workout.

Non-functional requirements

- 2.5 - “There should be a consistent design and user experience across all parts of the app.”

This requirement was partly met. Most sections of the app, such as the home screen and workout screens, have a consistent design and user experience. The heart rate monitor screen and the sensor data collection screen differ in their design, however. This was because they were built by other groups than built the app (our group and Group D, respectively) and we did not have time to make all the designs consistent after development of the app was completed.

2.1.3. Data visualisations

Functional requirements

- 3.1 - “Data visualisations will appear both on the website and within the Android app.”

We met this requirement successfully. The website includes many data visualisations, such as maps of the user's runs on the fitness page and various graphs and charts on the sports pages (for example, the visualisation on the cricket page of body-lean during the technique session). The app includes the heart rate visualisation screen.

- 3.2 - “There shall be data visualisations tailored to the specific sport being analysed.”

Each sport the system is currently capable of analysing - running, cricket and boxing - comes with tailored visualisations on the website. For example, the wireframe captures for boxing are accompanied by visualisations of how much the user's feet rotated during the recording, but this is not available for running.

These visualisations are not available on the app, however. This is because the wireframe analysis is not implemented on the app (for a variety of reasons, mainly that it would make for a poor user experience and require a lot of processing power) and so it would make little sense to display the graphs on their own. We therefore partly meet this requirement, but for good reason.

- 3.2.1 - “These visualisations will be decided upon by investigating the particular features of the sport and deciding which information will be most useful or relevant to display.”

The visualisations were decided in conjunction with Group F by looking at the particular sport and choosing relevant data to visualise - for example, we found that body lean was an important factor in cricket, and so chose to visualise that.

- 3.3 - “There shall be data visualisations for the user's general fitness and health data.”

This requirement is met on the app, with the heart rate visualisation screen, but not on the website. This was due partly to time constraints - we chose to focus on

the visualisations of the wireframe data instead - and partly to a lack of data with which to build or test any visualisations.

Non-functional requirements

- 3.4 - “The website visualisations should be rendered using a JavaScript framework that is yet to be decided.”

This requirement is met in full; the website visualisations are rendered using D3 and SVG, as mentioned in the ‘Development’ section above.

- 3.5 - “The Android app visualisations should be rendered natively e.g. using Java.”

This requirement is met in full insofar as there are visualisations on the Android app i.e. the heart rate screen.

2.1.4. Overall UI design

- 4.1 - “There should be a consistent design and user experience across all parts of the system.”

We partially met this requirement. The design and user experience is consistent within the website, and is largely consistent within the Android app (see requirement 2.5 above), but is different between the two. Again, the reason for this is that we ran out of time; the app came together too late for our group to be able to make it wholly internally consistent. If we had been able to standardise the design and user experience of the app we would have made it consistent with the website at the same time.

2.2. User interface and user experience

In the first stages of the project we had plenty of ideas about creating a well-tested, user-centred system that looked good and was functional and intuitive to use. Once we had written up some user stories and personas and drawn a variety of sketches for each page, however, some of these ideas got left behind. With more reference back to our resources, we would have had a better grasp on where we were in the project and what needed doing.

Our initial goal had been to follow some sort of iterative development cycle, where we had a working prototype of the web and Android applications from early on and could then carry out user tests to see which aspects worked best and which frustrated users. These

prototypes, while non-functional, would have allowed each other group to more easily see what our vision of the system was, and would have allowed us to more easily adapt when other teams finalised their aims.

One of our misconceptions was the idea that “we are integrating everyone else’s components, and thus cannot do anything until they are done” - in hindsight it is clear that we could have done more in the meantime. Having proper prototypes (e.g. a website that looked like the finished system but was filled with placeholders) may even have given other groups more structure to work within, meaning their final components would have been easier to integrate with the website even if they weren’t completely finished until the last few days.

While we still deliberated over various UI/UX decisions, carefully considering the options available, we ran out of time to implement some of the more in depth ones. The difficulty of time constraints was compounded with the sudden influx of new functionality and ideas for visualisations which had not been mentioned before the final week. While they are present on the final system, none of the pages are laid out exactly as we would like - for example, on each sports hub there is a list of technique sessions, but they are listed by their ID rather than by some more useful measure such as a session name and date.

Overall, the theming of the web app is consistent and works well, but it did not transfer completely to the Android app. The web app’s accessibility leaves something to be desired, but pages are well named and after a quick introduction to the system’s features it would be straightforward to use.

2.3. Integration

The biggest hurdle we faced during the project was probably with inter-team communication. Each group was willing to communicate, but other commitments often came before the project for all members of the class, especially during the first semester. We would often need to meet a group to discuss their current designs and plans, which would mean several days of emailing back and forth to find a time that suited most of the team members. After this meeting we would do the things we had agreed to, and then try to schedule another meeting, which would usually be another week later. This made for a very slow development/communication cycle.

While communication was strong when we were physically with other groups, the turnaround time between meetings had a huge impact on how effectively we could discern what each team wanted us to build, as requirements had often changed by the next time we met. In hindsight, we can see how it would have been much more effective to ‘put our foot down’ and give the other teams firm deadlines or frameworks to work within.

We will now briefly outline our relations with the other groups.

2.3.1. Group A (Rankings)

At the start of the project we planned to have a data flow between us and Group A directly, potentially using Java RMI or XML RPC to call processing methods on their server, but it soon became clear that it would be simpler to run everything through the database.

Group A provided us with a set of test files in the format expected to be created by their ranking system, and kept us up to date with the information they wanted to display on the website.

There were integration issues between Group A and Group E, however, meaning that the transfer of Group A's data through the database was not achieved. All the data that we could obtain from Group A therefore comes from test files, though this still allowed us to showcase their system within ours.

2.3.2. Group B (SmartCoach)

We were originally meant to be working closely with Group B while we built a framework for their SmartCoach app and the data collection app to be accessed via one application on the Android phone. In the end, however, they took over development for themselves as Group D had provided an accessible library of methods to gather data and there was thus little for us to do. Our task then became to put together the UI theme, but the app only came together properly towards the very end of the project and so we could make only rudimentary changes.

In terms of communication Group B were often in the lab and thus often available for discussion on their current plans. As all their data from the workouts went straight into the database it was fairly straightforward for us to pull it into the website, and thus to show users' data from workouts completed on the mobile app.

2.3.3. Group C (Social)

Group C developed methods to gather data from a user's Facebook and Twitter accounts, and then put them into the database. As their functionality was written in plain JavaScript (rather than in AngularJS) it was tricky to integrate. While we were able to just include the script files and call methods within them, the difficulty was in accessing data from it (and for them to access data from the website).

At the end of the project, users wishing to log in with Facebook were redirected to the Facebook website, which called back to the database, which in turn redirected the user back to us with the session key and Facebook token in the URL. This caused issues; as we only tried to properly integrate this procedure within the last few days of development, we did not have time to quite work out how to use cookies to store Facebook information

in between redirects, or how to clear the session key and token from the URL without having to refresh the page.

This page refresh (to clear the URL and thus not re-log in with FB at each page load) stopped the Facebook script from running, meaning data was not always scraped and sent to the database. The Twitter login remained unimplemented on the main site, although would have followed a similar pattern to Facebook login.

Overall, Group C were easy to communicate with, although often busy (as were we) with other coursework. They provided requirements documents and lists of achievements early in the process, although these were subject to change. Some elements of the social system (e.g. profile pictures or non-placeholder achievements) were only added to the database near the end of development, but techniques such as pair programming helped a lot with last-minute integration - we simply ran out of time.

2.3.4. Group D (Data acquisition)

As Group D were able to work independently for the most part, and as their goal was gathering data which other groups would process before we displayed it, we had little need for direct contact.

2.3.5. Group E (Database)

As with all groups, we had to work closely with the database team. While functionality was often implemented soon after requesting it, communication could be difficult; frequently an API for a certain function would exist in code but not be documented. Much time was thus wasted examining the API code to see which functions were available to us.

A lot of the time, our work was held up by lacking communication between other groups and Group E - for example, even though it was an obvious and necessary requirement from the start, wireframe captures did not have a field representing the sport they were for until the final few days of the project, invalidating all existing test wireframe captures (as a new compulsory field had to be added to the wireframe objects).

Our other main area of integration with Group E was owing to the website being hosted on their server. While there was a URL available which, when accessed, would pull down the most recent version of our repository to their server, this was not really sufficient. As the function only pulled down the master branch, in order to test changes that would only work on their server (e.g. Facebook integration) we had to commit and push code to the master branch that potentially did not work.

While Group E were invaluable, and put together a robust database system and set of APIs, they also inadvertently acted as something of a bottleneck for other groups - often

functionality could not be implemented until a new field or API was available, and while this was often very quick it could occasionally take a period of a few days. In hindsight, having a more flexible test database for development purposes would have been very useful, as would the ability to transfer test branches of the repository to their server, rather than only the master branch. On our part, using the issue-tracking facilities offered by GitHub may have provided an easier way to keep track of APIs which needed implementing, rather than chains of emails and instant messages.

2.3.6. Group F (Wireframe)

Group F were responsible for processing the wireframe data from the Kinect and visualising it as an interactive ‘video’. We agreed early on with Group F that they would provide an embeddable video player for the website, which we would place (using an iframe) and style, so that they did not need to know about the website design and we did not need to know about the mechanics of the video player.

This mostly worked well. We received such a player quite quickly, and were able to embed it easily enough. We did have to modify Group F’s code to enable streaming from the database, however, which we did not expect; it would have been better for them to have handled this. Additionally, they mentioned very late during the development process that they also wanted a player for live-streaming Kinect footage, which had not been communicated to us previously. Although we were able to include this on the website in time, it was rushed and not properly integrated with the rest of the design.

We also worked with Group F to produce graphs and other data visualisations related to the wireframes. They were able to tell us in good time what sort of visualisations they wanted and which data to use, and there were no problems with this part of the relationship.

2.4. Comparison with similar systems

We will now compare STACS to similar systems that are already publicly available.

2.4.1. Endomondo

Endomondo is a website and a suite of mobile applications for iOS, Android, Windows Phone and other mobile platforms that allow the user to track and map their runs and cycling activities [Endomondo, n/a].

2.4.1.1. Comparison to STACS

Endomondo could be compared to the running and cycling tracking exercises that constitute parts of workouts in our system’s Android app. It allows the user to easily start or stop recording workout data, supports external sensors (including the heart rate

sensors) and provides the ability to browse recorded workouts (analogous to individual exercises in our system) online, including mapped runs / cycle activities. It also has a SmartCoach-like system that is available only to paid users.

2.4.1.2. What it lacks

Endomondo is a very focused system that is based on tracking built-in smartphone sensor data. This means it doesn't have any tools for technique analysis. There is no way to properly record exercises that cannot be tracked well using built-in smartphone sensors, such as pushups, whereas our system has a simple way of doing that.

It also lacks an achievements system, but that may be compensated by the presence of challenges. Finally, 'smart coaching', while free in our system, is a premium feature of Endomondo.

2.4.1.3. Advantages

Being a fully developed system, Endomondo has more polished user interfaces and a more functional website. It also has mobile apps for a much greater range of devices, providing support for iOS, Android, Windows Phone, BlackBerry, Nokia Series 60 and Nokia Series 40 devices.

There are numerous different movement-based workout types, and Endomondo provides more ways to visualise workout results, including an approximate number of calories spent on it. Additionally, the system of challenges allows users to compete globally or locally by achieving some particular set goals.

2.4.2. MapMyFitness and its satellite applications

MapMyFitness and the associated applications (which include MapMyRun, MapMyRide, MapMyWalk, MapMyHike and even MapMyDogWalk) specialise in actually mapping the aforementioned activities, but also provide some fitness data to the user [MapMyFitness, n/a], which makes them systems of the same general purpose as STACS.

2.4.2.1. Comparison to STACS

MapMyFitness, like Endomondo, is based on tracking the user's location using the smartphone's built-in sensors. This makes it similar to the exercises in our Android app. It also allows the user to view the saved past runs online.

2.4.2.2. What it lacks

Like Endomondo, it doesn't have a way to do technique analysis, doesn't allow for any non-running/cycling/walking/dog walking exercise to be tracked and doesn't have a proper

achievement system. Just like Endomondo it has a system of challenges in place of achievements.

2.4.2.3. Advantages

The fully-fledged challenges system allows the users to participate in sponsored challenges no matter where in the world they are. There is a complex system that allows the user to input the food they eat and track their calorie budget, which would have been a great addition to the SmartCoach functionality of our system. Social features are very prominent in MapMy... apps, including friends the user could compete with. STACS has a friends system too, but it is less developed and integrated with the mobile application.

MapMy... apps are available for both iOS and Android, as opposed to our Android-only app.

2.4.3. Strava

Strava is another tracking-oriented application that provides fitness data to the user. It has dedicated applications for running and cycling. [Strava, n/a]

2.4.3.1. Comparison to STACS

Like MapMyFitness and Endomondo, this is a system focused on terrain surface movement: running and cycling being the two sports it focuses on.

2.4.3.2. What it lacks

Just like the previously compared systems, Strava only offers location tracked based workouts. It also does not have a SmartCoach-like system.

2.4.3.3. Advantages

Strava has a system that allows the user to explore “segments”: small parts of running or cycling routes. It is possible to find out the average speed for a segment, as well as more technical details, such as the incline and precise length.

2.4.4. Fitbit

A proprietary sensor-based system that focuses on constant tracking and using a dedicated external device. As well as fitness activity, it also tracks user’s sleep patterns. [Fitbit, n/a]

2.4.4.1. Comparison to STACS

Fitbit has some things in common with STACS, but it focuses on improving the user's health through constantly tracking their activities, including sleep. It could be said that it tries to accomplish the same job as STACS, but in a different way, stimulating more generic activity instead of particular exercise plans.

2.4.4.2. What it lacks

Fitbit, like the previous examples, does not have a technique analysis system, but it has achievements, although they are not as concrete as the ones in STACS.

2.4.4.3. Advantages

Fitbit tracks the user's activity at all times, which makes for more precise and finely tuned health data. It also has its own easy to use hardware, as well as iOS and Android apps with a very high degree of polish.

2.4.5. Nike+

Probably one of the most popular tracking systems in the market, Nike+ has the benefits of being integrated with a world leading sportswear manufacturer's gear [Nike, n/a (1)]. It uses a uniform points system for comparing results between various disciplines called 'NikeFuel' [Nike, n/a (3)].

2.4.5.1. Comparison to STACS

Nike+ is a massive suite of applications and services designed to work with the company proprietary periphery, as well as the smartphone sensors. Some parts of it, like Nike+ Running, are similar to workouts of STACS Android app, while some others, like Nike+ Kinect Training, offer technique analysis and personal advice very much like the technique analysis for Kinect that is part of STACS.

2.4.5.2. What it lacks

Even recommendations are not a part of Nike+ suite (or at least not an easily accessible part). Nike+ doesn't have any cricket technique analysis features.

2.4.5.3. Advantages

Nike+ is probably the most well-developed system of this kind. It has better sensor support, more apps for more platforms, support for numerous sports, achievement system ('badges'), a comprehensive Xbox application that does training using Kinect [Nikea, n/a (2)] and numerous other features.

One of the most interesting features is the ‘NikeFuel’ point system which allows users to compare their results to those of other users even if none of their sport disciplines overlap. It is, in its essence, a sports-agnostic ranking system.

2.4.6. Moves

Moves is a simple tracking system that tracks the user’s walking, running, cycling and other activities at all times (as opposed to the more session-oriented approach of the other applications), but also allows them to add any other activities from a curated list manually. Its main advantage over similar systems is that it only uses the smartphone: no external sensor devices are necessary. [Moves, n/a]

2.4.6.1. Comparison to STACS

Moves compares to STACS just like all the run/cycle trackers and Fitbit do: it does some of the things STACS does better, but doesn’t have the full scope.

2.4.6.2. What it lacks

There is no technique analysis, no achievements, no additional exercises (although the user is free to add any activity they wish).

2.4.6.3. Advantages

It is an incredibly easy to use application with an innovative user experience that focuses on simply displaying the basics. There is automatic calorie calculation.

3. Conclusions

3.1. Learning outcomes

In working on this project, there are two main areas in which we are now much more experienced - firstly, in the specific tools we used (notably JavaScript / AngularJS) and secondly in the general development of a large software project between a large team of people, especially in the realm of project management and its necessity. While the final product may have suffered a little as a result of our inexperience in these areas, we now have a much firmer grasp on how a project should operate and how to avoid the issues we encountered.

3.1.1. Tools and technologies

In terms of tools, probably the biggest challenge was developing an in-depth understanding of the Model-View-Controller (MVC) design pattern. Although this is something we had briefly covered in class, this is the first time any of us had actually tried to implement something using it. That said, AngularJS did provide many helpful abstractions to keep JavaScript organised into a well defined MVC structure, as opposed to the usual shambles that large-scale JavaScript projects tend to become.

While we chose AngularJS in part because of its clean abstractions away from the underlying AJAX code (i.e. through `$routeProvider` and `$location`), we failed to realise at the beginning just how much effort it would take to get our heads around everything the framework offers. Having spent many hours on support forums and developer blogs, it is clear that AngularJS is not the kind of tool you can pick up and fully understand in a weekend; even after the full project there are many things left for us to learn.

3.1.2. Communication and management

3.1.2.1. Class-wide

The general consensus is that encouraging groups to self-organise worked better than many might have expected. The class successfully held several meetings at the start of the year to make decisions about the features and structure of the system, which were well-attended, organised and minuted. Some of these meetings were class-wide and some just included representatives from each group. During the year there were also class-wide coding sessions in the lab.

However, we feel the project would have benefited from clearer oversight and leadership. The tendency was for each group to become focused on getting their part of the system working, occasionally without regard for how their work would integrate with that of other groups. This sometimes led to mismatching standards, out-of-date documentation or other easily avoidable problems.

There was also a lack of clarity as to exactly what the system should do and the features it should include. This was partly because of the choice of project - few in the class knew much about sports or related software - and partly because there was no real guiding force for the system. As a result, we feel, the system has many exciting features (wireframe analysis, the SmartCoach, data visualisations, social media integration, and so on) but does not have a clear and unified purpose.

In industry, any project of this size would be assigned at least one project manager to ensure inter-group cooperation, enforce deadlines, and generally keep the project on track and focused. Although it would be quite a lot of work, we feel assigning either a student or a member of staff to perform this role really would have helped the development of the system. This figure would have been able to set regular deadlines for the whole class throughout the year, assess the merits and relevance of proposed features, and make sure that groups were thinking about the system as a whole rather than just their own requirements.

3.1.2.2. As a group

Within our group, we feel we worked together very well. We kept up constant communication during the year through online group chats, made use of team management software such as TeamBox, and met regularly in person for our supervisor meetings and otherwise. We also became quite adept at breaking down tasks and splitting up the work equally among the four of us. The project did however highlight the importance of assigning ourselves internal deadlines and meeting them, which we did not always do, so this is something we can work on in future.

3.2. Future work and potential improvements

3.2.1. User interface and user experience design

As the UI team, the thing we would most like to improve is the final system's user interface and experience. As previously mentioned, an iterative development cycle with well-documented user testing would have made an enormous difference in the final design, and aided us in our communication with other groups. As the project stands, it is well themed but not fully complete - various aspects on each page lack the polish we envisaged, and while it is functional the user experience leaves something to be desired.

Now that each other group's features are actually confirmed, finalised and (for the most part) working, it would be much more straightforward for us to design a site which incorporated everything into a suitable structure, along with well thought-out onboarding and guiding processes for new users.

3.2.2. Data visualisation

While many useful graphs are presented in the website and Android app, it would have been preferable for them to be both more interactive and for their purposes to be more obvious on first viewing. While each graph makes sense once you understand it, there is no in-app description or guide to them. Additionally, graphs tend to be scaled based on how long a session was - while this works for shorter sessions (like those we show), allowing users to zoom in on particular aspects of a session or to scroll through graphs automatically as the wireframe was played would make it a lot easier to analyse their technique.

An interesting (although complex) extension would have been to offer natural-language analyses of each technique session; for example, telling users that they “need to keep their elbows closer together in their pre-bowling stance” is more useful than showing them a complex graph of how far from their side their elbows were at certain times and expecting them to work it out themselves. However, this would most likely have involved some intense computation being performed on the client-side, which may not work so well if a user’s computer was not up to standard (or if they were on a mobile device).

3.2.3. Extensibility

The system currently contains hubs for three sports (running, boxing and cricket). As each hub follows a very similar pattern it would not be difficult to add extra sports to the system as demand for them appeared. While each hub would take a reasonable amount of work (especially if aiming to provide written feedback) to reinterpret each of the graphs in a relevant way, the layout and process would remain constant.

This is also made easier by the MVC layout used, and by the structure laid out by AngularJS. For example, each hub would have its own controller (e.g. ‘CyclingCtrl’) which dealt with user interaction on the webpage, and then a service for each hub which offered various functions (e.g. ‘analyseElbowSeparationGraph’) to help with the gathering and analysis of each graph.

3.3. Concluding statements

During this Junior Honours Project we have designed and implemented part of a sports training system, working as a sub-group within a larger project team. This report has detailed our plans, decisions and experiences. We have learned a considerable amount about the need for project management and communication; about new technologies such as AngularJS and single-page architectures; and about working on large-scale developments.

Although STACS is not as fully finished as we would have liked, we are nonetheless proud of what we have achieved.

4. Testing summary

Since we were responsible for presenting the work of all the other teams, our products – the website and the Android application, in which we were tasked with providing coherent user experience – had to be thoroughly tested to ensure they worked perfectly in all the regular and corner cases.

Initially, we considered fully automating the testing process by using Selenium WebDriver and AndroidDriver [Selenium, n/a (1)][Selenium, n/a (2)] (now discontinued). However, at a later stage we realised that the way our collaboration was organised meant we would only be able to test the whole assembled system quite late in the cycle, at which point writing automated tests would take too much of the limited development time. Writing automated tests for the web app using Karma may have saved some time spent debugging, but often the reason something wasn't working was due to a simple syntax error which was easily corrected thanks to error messages from the JavaScript console.

Instead, we focused on doing as much real user testing as possible, encouraging other group members to spend time using our software and noting which aspects they found particularly confusing or odd.

For the Android application, testing it on various hardware and supported OS versions was crucial in the process of determining whether the application was fully functional. We tested it on a Motorola Moto G running Android 4.4.2, Nexus 4 and Nexus 5 running 4.3 and 4.4, and others.

Overall, the testing could have been done in a number of ways different from the one we employed. While the end service has not suffered immensely from lack of proper automated testing, it would have been nice to have allocated some time solely to testing and debugging the complete system.

Appendices

- 1 - Requirements specification (see MMS)
- 2 - Interim report (see MMS)

Bibliography

- “Acrom™ - Webfont & Desktop Font”, MyFonts. Accessed May 05, 2014.
<http://www.myfonts.com/fonts/northernblock/acrom/>.
- “Akzidenz-Grotesk® BQ - Desktop Font”, MyFonts. Accessed May 05, 2014.
<http://www.myfonts.com/fonts/berthold/akzidenz-grotesk-bq/>.
- “AndroidDriver” Selenium. Accessed May 05, 2014.
<https://code.google.com/p/selenium/wiki/AndroidDriver>.
- “Build Software Better, Together.” GitHub (1). Accessed May 05, 2014.
<https://github.com/features>.
- “D3.js - Data-Driven Documents.” D3. Accessed May 05, 2014
<http://d3js.org/>.
- “Design | Android Developers.” Android Developers. Accessed May 05, 2014.
<https://developer.android.com/design/index.html>.
- “Endomondo | Community Based on Free GPS Tracking of Sports.” Endomondo. Accessed May 05, 2014.
<http://www.endomondo.com/>.
- “Fitbit® Official Site: Force, Flex, One & Zip Wireless Activity & Sleep Trackers.” Fitbit. Accessed May 05, 2014.
<http://www.fitbit.com/uk>.
- “HTML Enhanced for Web Apps!” AngularJS. Accessed May 05, 2014.
<https://angularjs.org/>.
- “Lato.” Google Fonts. Accessed May 05, 2014.
<https://www.google.com/fonts/specimen/Lato>.
- “Map Fitness Training and Track Fitness Workouts.” MapMyFitness. Accessed May 05, 2014. <http://www.mapmyfitness.com/>.
- “Moves - Activity Diary for iPhone and Android.” Moves. Accessed May 05, 2014.
<http://www.moves-app.com/>.
- “Nike+” Nike. Accessed May 05, 2014.
<https://secure-nikeplus.nike.com/plus/>.

- "Nike Kinect Training." . Nike.com. Accessed May 05, 2014.
http://www.nike.com/us/en_us/c/training/nike-plus-kinect-training.
- "Node.js." Node.js. Accessed May 05, 2014.
<http://nodejs.org/>.
- "Roboto Is a Four-headed Frankenfont." Typographica.org. Accessed May 05, 2014.
[http://typographica.org/on-typography/roboto-typeface-is-a-four-headed-frankenst
ein/](http://typographica.org/on-typography/roboto-typeface-is-a-four-headed-frankenst
ein/).
- "Running and Cycling GPS Tracker, Performance Analytics, Maps, Clubs and Competition | Strava." Strava. Accessed May 05, 2014.
<http://www.strava.com/>.
- "Travis CI Integration." GitHub. Accessed May 05, 2014.
<https://github.com/theintern/intern/wiki/Travis-CI-integration>.
- "What is NikeFuel?" Nike. Accessed May 05, 2014.
https://secure-nikeplus.nike.com/plus/what_is_fuel/.
- "What is Selenium." Selenium. Accessed May 05, 2014.
<http://docs.seleniumhq.org/>.