Progress Monitoring Web Application

Daniel James Wilson

April 2, 2014

Mobile and Web Application Development

Supervisor:
Dr Moi Hoon Yap

Manchester Metropolitan University

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Signed _____

Acknowledgments

I'd like to express my gratitude to my parents, for their support throughout my three years of studying at university, my other half, Grace Day, for proof reading this dissertation, and my supervisor Dr Moi Hoon Yap for her invaluable support and advice with this dissertation.

Abstract

This dissertation will cover the process of the building of the progress monitoring web application Recur. The application will be built using a number of tools, some of which are; Ruby, Ruby on Rails, Git, and jQuery. It will address the challenges faced when developing a web application in today's world. The three main areas of focus will be on designing and developing the application, and testing and evaluation the finished product.

The design of the application will look into designing with the visually impaired in mind. It will also talk about why responsive web design is important for mobile users. The development will talk through the development process, using modern technologies to create and efficient application that is easily extendable and maintainable. The testing chapter will talk about why testing is important, what testing options are available with focus on test driven development, and why test driven development is efficient. Part of the testing chapter will be an evaluation of the product which will look at how the application has changed throughout development and it will also look at some user feedback from the first release of the application.

Contents

List of Figures

Abbreviations

AJAX   Asynchronous JavaScript and XML

API   Application Programming Interface

CRUD   Create, Read, Update, Destroy

CSS   Cascading Stylesheets

HTML   HyperText Markup Language

HTTP   HyperText Transfer Protocol

MVC   Model View Controller

TDD   Test Driven Development

Chapter 1

Introduction

1.1 Fitness applications

Fitness applications are important because they provide people with a way to monitor how they are getting on with their health. Keeping fit and getting healthy is an everlasting venture, and improvements are only noticeable over a long period of time. Although, whilst the participant may not see a physical change, their health and fitness is improving with every visit to the gym. A fitness monitoring application allows the user to track what activities they completed at the gym, and how well they performed. For example, if you went on the treadmill, you could track how long it took you to run a mile. Next time you go on the treadmill, you will log your mile time again and see if you have improved. It also gives that edge of motivation. By showing you how well you are doing it gives you the motivation to keep at it. If you are not doing so great, it will allow you to see where you need to improve.

1.2 Benefits of being fit and healthy

Scientific studies and statistics provide evidence that obesity is a growing problem in modern society. It has been on the rise for a long time. Obesity in adult men was up to 35.5% and women up to 35.8% in 2009-2010 (KM et al. 2012). This is obviously not good at all. This is a problem that must be tackled. There are many benefits of leading a fit and healthy lifestyle. A number of which are (NHS 2013):

1. Up to a 35% lower risk of coronary heart disease and stroke.

2. Up to a 50% lower risk of type 2 diabetes.

3. Up to a 30% lower risk of depression.

4. Up to a 30% lower risk of dementia.

1.3 Applications

Mobile and web applications are on the rise. People are constantly on their phones interacting with their applications, be it a native mobile application or a web based application. Applications are useful because they usually perform one or two tasks really well. Be that recording notes or tracking your blood sugar levels. They allow the user to track information and keep it all organised. With this information, the application will be able to display back to the user in an orderly fashion;

either in the form of a table listing all their previous expenses, or a graph showing how much they have spent in certain categories.

1.4 Recur

Recur is the fitness application that is being documented in this dissertation. It focuses on three main aspects of fitness. These are:

1. Current and goal weight

2. Fitness specific goals

3. Fitness log

By keeping the application down to these three features, it will allow the users to maintain focus and not become distracted by other, unnecessary aspects. It will be displayed in a modern, minimalistic design and built using up to date technologies (this will be explored further in the development Chapter 5 on Page 17).

1.5 Learnings

Throughout the production of this application, there will a number of opporunities where knowledge can be gained. The design stage will have to take into account user accessibility. This covers catering the design for the visually impaired and making the website as user friendly as possible. The development stage will cover the learnings of using a new MVC framework, building and authentication system, and securing it from attacks. Testing will teach the why testing is important and how to be efficient when writing tests.

1.6 Dissertation structure

This dissertation will cover the entire process behind the building of the progress monitoring web application. The application is for monitoring the user's fitness progress. It shall allow a user to track their current and goal weight, any goals they may want to achieve, and a diary of fitness entries. The application will focus on these three main points so as not to not distract the user away from their progress. Other features like social media sharing and calorie tracking will not be implemented. More information on these choices can be found in the design Chapter 3 on Page 7 of this paper. The literature review Chapter 2 on Page 4 will focus on the current state of fitness progress monitor applications that are already available for use. It will discuss both the good and bad aspects of each application and highlight the areas which will be improved upon or performed differently by the new application. The design Chapter 3 on Page 7 will cover the reasoning behind most of the design choices that were made during the planning of this application. For example, which font was chosen and why. The design chapter will also cover some development choices, such as the data structure design. Following this, the resources Chapter 4 on page 14 will talk about what tools, languages, and frameworks were used to develop the applicatoin. After that, the development Chapter 5 on Page 17 will talk through the development process. It will go into detail regarding the problems and challenges that were faced during development and how they were overcome, and also the deployment of the web application to the internet. Following the development Chapter will be the testing Chapter 6 on Page 23. This will go over the testing methodologies

and tools used in the building of the application and why they were used. It will also cover how bugs and errors picked up by the testing were overcome. Also covered in the testing Chapter will be the evaluation. The evaluation section will cover what the outcome of the product was in comparison to the original idea. It will also cover feedback from users about their experience and how their feedback will be acted upon. The conclusion (Chapter 7 on Page 26) will conclude the paper by demonstrating what knowledge has been gained from this build, what would have been done differently, and also possible future developments from the product.

Chapter 2

Literature Review

2.1 Introduction

It is known that being overweight can be a risk to your health (Theodore B. Van Itallie 1985). This, along with other factors, is driving a percentage of the overweight population to try and lose weight. Most people give up on weight loss due to lack of motivation, this is what the new application aims to fix. By allowing users to track progress through an easy to use web application, that is accessible anywhere via the internet, it will help the user maintain their motivation to keep losing the weight, and also keep the weight off once they have reached their goal weight.

There are already a number of weight tracking, fitness, and calorie counting applications out there on the market. Some of them offer all three services, and often a lot of other additional features, whilst others are more stripped down and only offer one or two. This application will focus on three areas to keep the user on track and not confuse them with features they do not need to worry about. The areas that the application will be focusing on are weight, goals and a fitness log.

2.2 Related work

This section will analyse a number of related websites offering a similar service to the new application. It will give a brief explanation about the service and a breakdown of the good points and the bad.

2.2.1 Fitocracy

The first application is Fitocracy (Fitocracy 2007), a web and mobile application to monitor fitness. It is heavily featured, which is a plus for some people, but others can find it distracting. One of its main features is its social network site. It has a social network backend that will let the user follow your friends and give them kudos for completing certain exercises. This feature is excellent as it gives what is called a feedback loop. This is commonly used in mobile application today by having the ability to login with a social network and share progress to that social network. A social network backend is a great motivational tool but due the aim of keeping the application as simple as possible, the application will not have this. Fitocracy is very motivation based with features like challenges, quests, duels, and leaderboards all creating feedback loops which provide the user with constant rewards and recognition of your achievements. Again, the application will be staying very minimal so the new application won't be implementing such massive features. Instead, the new application will just have goals, which will create a small feedback loop, giving

the user motivation and recognition of their goals. The main downside to Fitocracy would be the large number of features. It has so many features and options all packed into the one screen it is very overwhelming and not a nice experience to use. The basis of the application is to focus on a number of key points and to strip down any of the unnecessary paraphernalia that surrounds current fitness applications, such as Fitocracy.

2.2.2 Fitbit

Fitbit is one of the biggest fitness apps out to date. What makes it so special is that you can purchase a wristband that monitors your activity and sleep, and then wirelessly syncs your statistics with your Fitbit account. This is one of many items, known as wearable technology, which enables users to track their activity. Others include the JawboneUp (Jawbone 2011) and the Nike+ Fuel Band (Nike 2012). This allows for less manual data entry. It counts how many steps you have taken, how many calories burnt, and how many hours sleep. Its downside is that you can only specify certain exercises. For example, working out in the gym, you cannot tell it how much you are lifting, so it cannot give you an accurate reading of how many calories you have burnt.

Fitbit focuses on balancing your caloric intake and expenditure to help you hit your goal weight. After inputting your details and your goal weight, it will give you a calorie intake goal. It then balances your caloric intake, by tracking your diet, and your caloric expenditure, by tracking your activity. It will then match the two and show you your calories in versus calories out. Focusing on calories in versus calories out is another popular way to lose weight. By decreasing your caloric intake and increase your caloric expenditure, you will lose weight.

Fitbit is very a well-designed web application. It takes the popular minimalist and flat design approach. It has a really clean dashboard that consists of six cards with information right there on them. If you click on one of the cards, you will see a modal pop up with more information on that card. In the bottom right hand corner of the pop up is an arrow, clicking the arrow will take you to even more information about that specific element of the application and this page will also allow you to enter data. This is a really efficient way of laying out an application as it give the user all the information on the first page, and more information and the ability to change that information is only two clicks away.

2.2.3 Nike+ FuelBand

The Nike+ FuelBand app is a little bit different from all of the others. It uses its own measurement (Nike+ fuel) to monitor your activity. You set yourself a goal and then as you move throughout the day, your fuel will go up. It connects to your mobile device via Bluetooth so you can see your stats on the mobile application, and that syncs the data with their servers so you can see even more data on the web application. Their main goal is similar to that of the new fitness application, to keep the user motivated to get out and get exercising. They use the NikeFuel as a measurement as it is a universal way to measure all kinds of activity (inc. 2013). You can also add your friends and see how they are doing with their goals. This adds yet more motivation by being able to compete with your friends. The ability to connect with your friends is a great motivational tool but it takes away from the simplicity that the application is aiming for.

2.3 Conclusion

To conclude, the website will follow a similar simple easy to use, design to Fitbit, but with a more focused aim on motivating the user to get back into the gym. Out of the two main types of weight

loss, it will focus on the fitness side. The decision was made to not implement both sides because to split the focus over two aspects would break the motivation from one focus to another. Also, the implementation of having calorie tracking would either be a big thing with a database of food types and their calorific value, or having to have the user enter both the food type and then the calorific value of that food type. This can lead to an incorrect calorific reading.

After seeing some of the over-complicated fitness applications with very poor user experience, the applications design was to be kept as simple as possible. Cutting down on user options and features while maintaining core, desirable features is how this will be achieved. Being a content driven application, the information needs to be readily available on the first page that the user sees, so that if they just wanted a quick view on how they are doing all the information would be visible as soon as they open the page.

Chapter 3

Design

3.1   Introduction

This chapter will discuss all of the design choices that were made during the production of the fitness monitoring web application Recur. It will go into more detail about the grid behind the design, why responsive web design is important, and will talk about the design of the database used to store all the application's information.

3.2   Grid design

Franco (2002) states that the concept of minimalist architecture is to strip everything down to its essential quality to achieve simplicity. The design of the application was to follow this simple rule. The bare essentials were the only aspects that needed to be displayed to the user. The set-up of the website is to have three blocks displayed in the centre of the screen, laid out on a six column grid. Grid design is important in web design as it adds continuity to each page. Each page will have the relevant information in a similar position to the last, allowing the user to be able to extract the information more quickly. A grid layout also makes it easier to design for other devices. Walkers & Digital (2013) found that their mobile traffic was up to 29% in the second quarter of 2013. From this, we can see that more and more people are browsing the internet from mobile devices. 43% of this traffic was from iPhones. One of the key challenges facing web designers is to make sure that the web application looks as good on a 320 pixel width devices as it does on a laptop monitor. This leads to the design of the application being based on a six column grid. This was for two main reasons: the first being to keep consistency through the sizing of different areas of the website and the second being it makes it much easier to scale the design down for mobile traffic. Due to the fact that mobile traffic is increasing, and also that the application will mainly be used on mobiles, responsiveness was very important when designing the application. It need to have the same feel on a 1920px screen as it would on a 320px mobile device screen.

3.3   Wireframe and mock-up

Simple designs were sketched up in a notebook to get a feel for how each of the pages would look on both a normal monitor and a mobile devices. After sketches were complete, they were then drawn up in Sketch (Coding 2013) where colour, text, and images can be added to get an even better feel for how the application is going to look and work. Once colour could be applied to the

design, user accessibility could be thought about. Around 8% of men in the United Kingdom are affected by some form of colour blindness and 0.5% of women (NHS 2012). If the design was to ignore their needs then there would be a percentage of the population that would be unable to use the application. For this reason, colour should never be the primary cue for information. During the design process, changes were made so that any occurrence where colour was the primary cue would also have a visual cue. For example: when the user checks off a goal, the text will have a strike through it as well as turning red. Also, a large percentage of the population have a visual impairment. A few steps were taken to keep the site safe for the visually impaired. The font size and weight is kept at a easy to read level so that it is not taking up to much space but is still easily readable. Also, the colour of the font has to contrast with that of its background. From a previous design, the background of the cards and headers was very light and the font colour was too light which would be quite difficult to read for a visually impaired person. Since then, the background colour has completely changed to a darker blue and the colour of the text has been made a lighter shade of grey. As for the typeface, Helvetica was chosen, as san-serif fonts are easier to read for the visually impaired. Where there is any large amount of text that the user will need to read, a line height of 1.5 will be used to make it easier to read for visually impaired people. Also, using ARIA HTML5 landmarks, such as header, footer and nav tags, will increase the usability of my website on screen readers. It is important when writing any web application to use semantic HTML that can be used easily without loading the style sheet file.
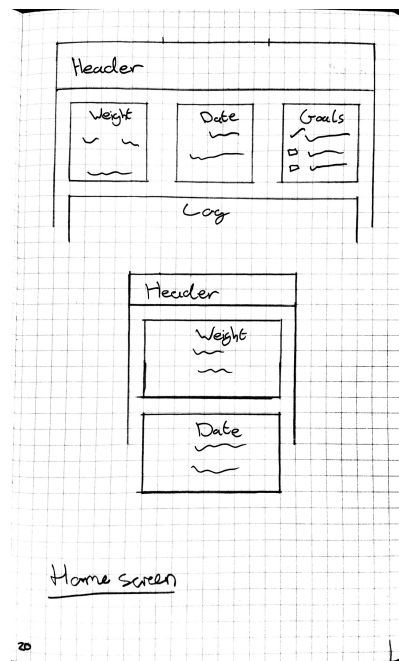
Figure 3.1: First wireframe for the web application's homepage.

3.4   Design in the browser

Most of the designs for the application were drawn up, but some of the easier designs were made directly in the browser. This was to speed up the process by cutting out unnecessary tasks. As opposed to the original plan, development began during the design stage. This was because there was enough of a design to begin developing some of the application. Once development was un-

Figure 3.2: First wireframe for the web application's input pages.

derway some elements had not been designed yet, for example; the forms for signing in and out. There was a clear vision of what the forms should look like, so instead of wasting time making the wireframe and mock-up of them, they were just designed as the web application was being constru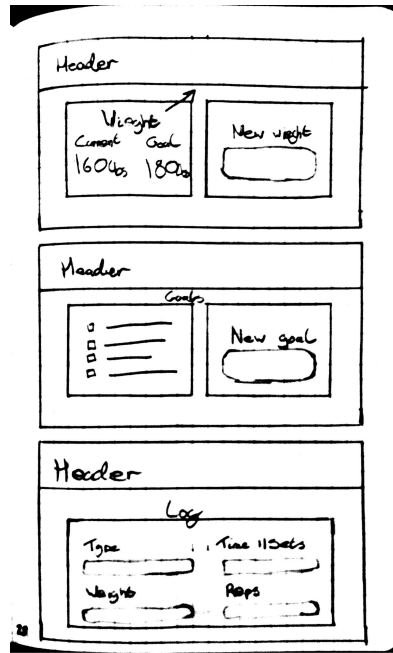cted, in the browser. This allows for rapid prototyping of the application's appearance and lessens the work load on the design front, and even if the design does not work there is no reason why another design cannot be drawn up and implemented.

3.5 Database design

As well as the design of the application, the design of the backend will need to be done before development takes place. For the backend, Ruby on Rails (Hansson 2013) will be used. The reasoning behind this will be discussed in the resource Chapter 4 Page 14. A number of attributes will be stored about each user and will be split up over a number of tables. Each table will have a one to many relationship with the user table, apart from the options table with will be one to one. This is because a user will only need to have one set of options in the database at any one time. The other tables are:

1. Current weight table (which will store the current weight of the user).

2. Goal weight table (which will store the goal weight of the user).

3. Fitness log table (which will store information about what exercises they did and how they did).

4. Goals table (which will store the users goals).

All of these will link back to the user's ID. Originally, there was going to be a user's weight table, which would store both the current and goal weight, but it was moved to two different tables because it is unlikely that the user will be updating the goal weight as much as they will be updating
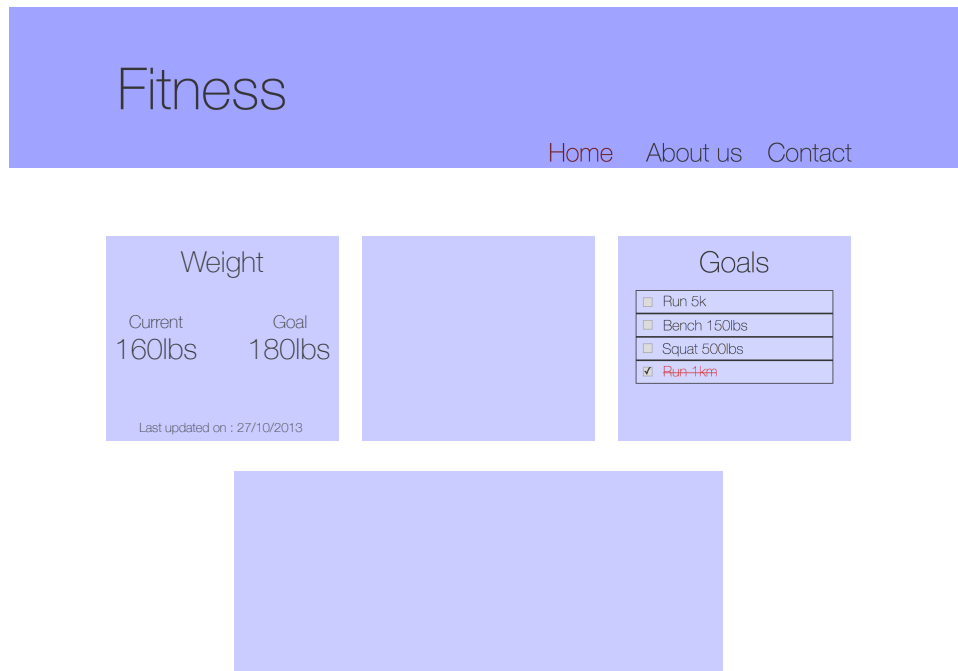
Figure 3.3: First initial mock-up of the website's home page done in Sketch.

their current weight. All of these tables will depend upon their user, so if the user is removed from the database, their data from the other tables will be removed as well. Figure 3.6 shows the Entity Relationship Diagram for the application.

There will also be two users of the application. A normal user and an admin user. The normal user will be able to interact with their data, and view their own page. The admin user will be able to view the admin section of the website. This will display a table with all of the users of the application, their current and goal weights, their number of goals and goals completed, and their last fitness log. They will also be able to view other user's pages but not interact with their data. Figure 3.7 shows the Use Case Diagram.

3.6 Conclusion

To conclude, designing with the user in mind is priority one. Making sure the application is accessible by almost everyone will give the widest scope of people that can potentially use your application. Making sure that the application has legible type and a strong contrast between background and font colour is important. Also, with the increase in traffic from mobile devices, designing for the mobile is another major priority. As well as having an efficient design, having an efficient backend is important as well. Making sure the data is stored in a way that will not result
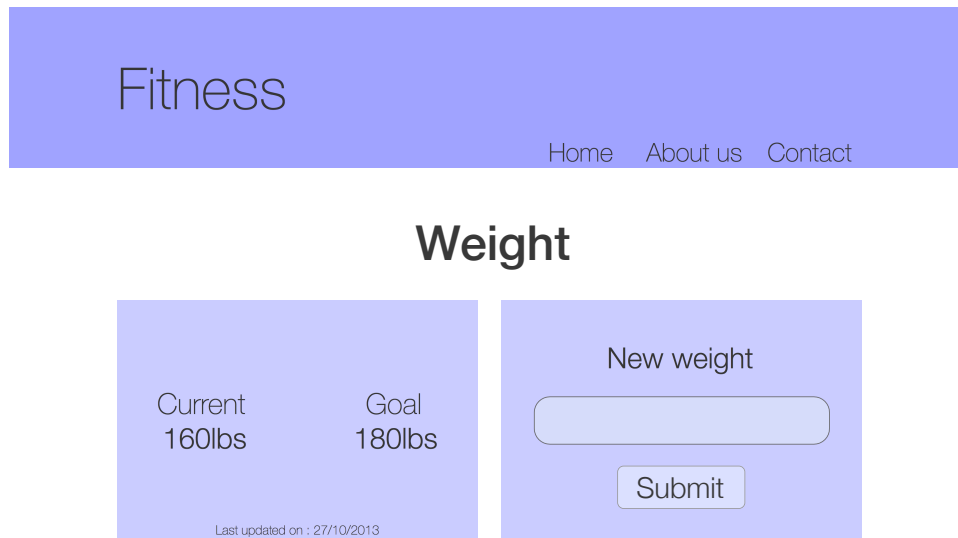
Figure 3.4: First initial mock-up of the website's input page done in Sketch.

in duplicated data and having all of the data easily accessible when needed will result in a smooth running application. All of the information covered at this stage can be handed over and used during development.
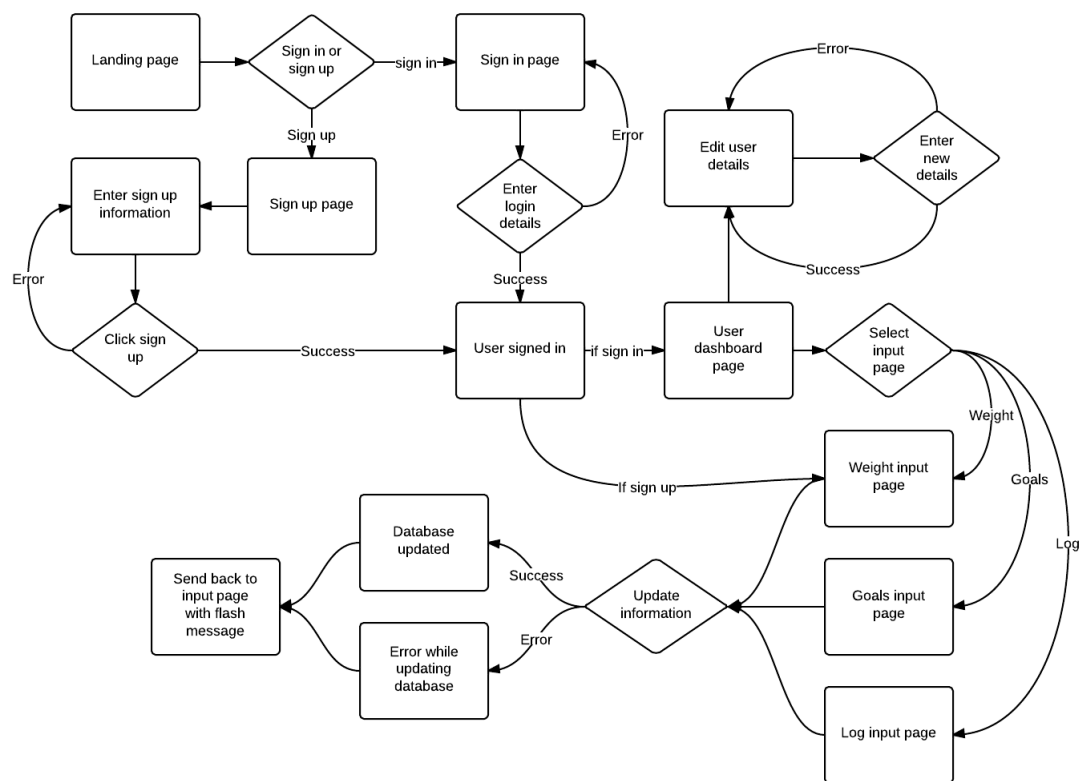
Figure 3.5: Flowchart for the processes and actions of the web application Recur.
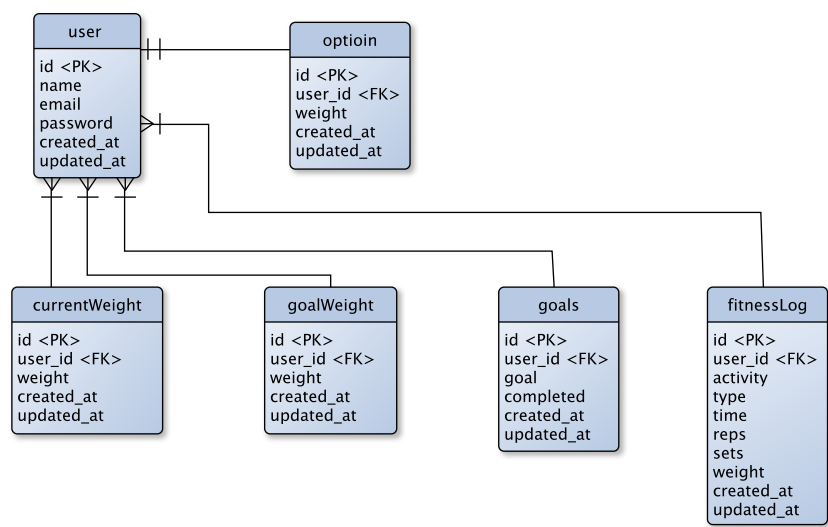
Figure 3.6: Entity Relationship Diagram for the website application Recur.
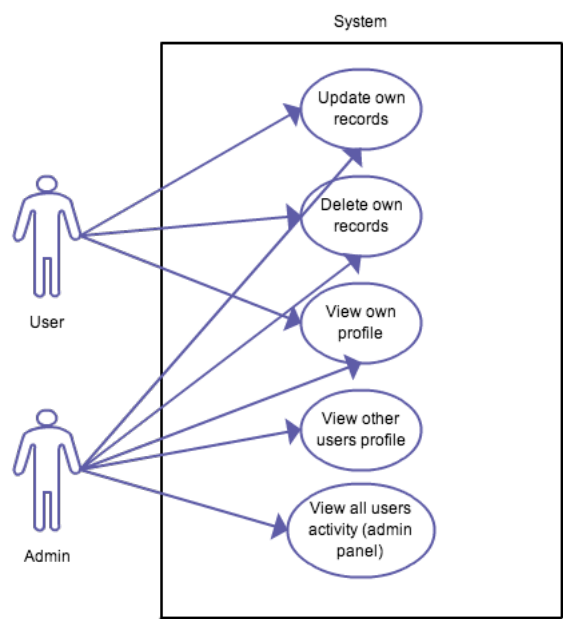
Figure 3.7: Use Case Diagram for the database backend of the website application Recur.

Chapter 4

Resources

4.1 Introduction

This chapter will talk about all of the resources that were used during the development of Recur. It will go into detail about the languages used, which applications were used, and also what RubyGems were used. It will talk about why each was used against its competitors.

4.2 Languages and Frameworks

4.2.1 Ruby on Rails

Ruby on Rails was the main backbone to this application. It was chosen for its rapid development characteristics, its extendibility with the mass amount of Gems that can be easily added into a project, and its scalability when it comes under heavy load with lots of traffic. It was also chosen on personal preference as compared with other languages that have similar frameworks. There is not much difference between a framework written in Ruby and one that is written in PHP. They are both going to do the same job but someone will find it easier to write in Ruby and others will find it easier to write in PHP. The main point is the use of a framework at all. MVC frameworks provide a really good backbone to any application. The MVC framework is comprised of three components; the model, the view, and the controller. Each component has a specific task to take care of. In Ruby on Rails's case, the model is what talks to the database. It can enforce rules on the data coming into the database. For example; it must contain an email that is valid against a certain regular expression. The controller is the logic behind the application. It contains functions that link to the views of the application which will display the information. Framworks also allow for rapid development of applications by using a system called CRUD . CRUD stands for create, read, update, and destroy. Each controller in Rails will use these four methods to create, read, update, and destroy a record of the model in the database. Instead of having to make your own CRUD system, Ruby on Rails gives you a number of calls that does it all for you. This is much easier and quicker than writing it all yourself.

4.2.2 JavaScript and jQuery

jQuery is a JavaScript library that allows developers to do very complex JavaScript functions with very little code. jQuery makes it really easy to interact and manipulate the web page. This means we can show and hide elements based on what the user has input in a box, or checked off in a check box. The main use for jQuery was the hiding of elements in forms and showing the menu on small

screen devices. It was also used for the chart switching on the fitness log. This will be explored in more detail later in this chapter. jQuery is also used for its AJAX method. AJAX is really useful when you need to update a database without refreshing the page. This was used when checking off goals. When it comes to alternatives to JavaScript and jQuery there are none. JavaScript is quite unique in that it is the only language that allows developers to manipulate the webpage while it is in use. With the progression to CSS3, developers can now do some page manipulation with that using element selectors and change style attributes on click and hover, but not to the extent that jQuery can.

4.3   Tools

A number of tools were used during development, some were used more than others but all of them played a key role in development. The main tool that was used is Sublime Text 2 (Skinner 2011). This was the text editor that was used to build both the static mock up of the site and the fully functioning web application. Sublime Text was chosen because of its vast amount of plug-ins. This makes it very versatile and efficient when it comes to building productions applications. There were a number of other editors considered for the job, for example; Vim (Moolenaar 2013) was going to be chosen due to its rapid development capabilities from all of the navigation being done on the keyboard. But it was not chosen due to the steep learning curve (as this would eat into development time). The time invested into learning Vim would probably not be beneficial to such a small project, but could be considered for future projects.

During the development of the static site, a JavaScript task runner called Grunt was used to compile Sass (Catlin et al. 2006), minify javaScript, and optimise images. When development moved onto Rails Grunt was dropped, as Rails compiles Sass and minifies JavaScript in the assets pipeline.

The second main tool that was used was iTerm 2 (Nachman 2011) using the oh-my-zsh (Russell 2009). Using the command when building a Ruby on Rails application is absolutely necessary. It is used to generate controllers and models, start the Rails server, interact with the database, and a number of other key features. The terminal emulator is not really that important but the use of the Zsh shell was very useful to the development stage. It has a number of excellent features like displaying the current Git (Torvalds 2014) branch that I was working in and if there was any changes that have not been committed and also highlighting in the terminal. Seeing as a lot of the time was spent in the terminal, it was important to make that time as productive and as efficient as possible.

Git was another tool that really helped with development. Git is a version control system. It allows the developer to commit files, and revert back any changes if something goes wrong. A slightly adapted branching methodology was used called Gitflow (Driessen 2010) which allowed easier version control during development and having a live and stable master branch that could be used to push to Heroku.

Heroku (Lindenbaum et al. 2007) was used to host the site, more information on this service can be found in the deployment section 5.5 of this chapter on page 21.

An application called Dash (Popescu 2014) was also used during development. Dash allows you to download documentation for certain languages and frameworks and stores it offline. It

allowed the Rails documentation to always be readily available whenever.

4.4 Gems

RubyGems (Quaranto 2009) is a package manager for the distribution of Gems. A Gem is a small package of Ruby code designed to either work on its own, for example a to-do list keeper, or work inside another application, for example the LazyHighCharts gem that was used to add charts to the log section of the website. They can be very useful by cutting down the amount of code needed to be written or by already having complex function built in and therefore not having to write them yourself. A number of other Gems were used during development, along side the default Rails Gems. The bcrypt-ruby gem was used for user creation and authentication. It allows you to create a password digest for storing in the user's record using one way encryption. This makes the system a lot more secure than storing the password in plain text, or even using a salt and hash to store the password. Sass-rails was used so that the style sheets could be written in Sass and compile to CSS at build time. Sass has a number of benefits over vanilla CSS, like being able to use variables for colours to allow for easier mass colour change. The Better Errors gem was also used during development to give a more in-depth error message when something went wrong. The default Rails error message is as descriptive as the better errors Gem so it becomes a lot easier to debug the application.

A number of testing gems were used during development to ensure the application was secure and working correctly. Rspec is the main gem which is used to write and run all of the tests. Using Rspec in conjunction with Capybara and FactoryGirl, it is easy to write extremely powerful tests. These tests not only interact with the Rails backend, but interact with pages and makes sure the redirects are working as intended.

4.5 Conclusion

To conclude this chapter, it is necessary to state that the outcome of the product will not be defined by the tools you use. Most tools are very similar and it almost always down to personal preference. In some circumstances, some tools will be better for a specific task than others, however, most tools are very similar and it is almost always down to personal preference.

Chapter 5

Development

5.1 Introduction

This chapter will discuss the development stage of the production of the progress monitoring web application. It will cover a number of topics, from development tools used during this stage to development methods uses whilst building this application. It will also provide a step-by-step walk-through of the building process highlighting problems and how they were overcome.

5.2 Development Log

Development began ahead of schedule because designs were already done. Even though not all the designs had been done, the essentials were complete, it was only separate pages that were missing. The colour scheme, font choice, and general feel were all there so development began with the building of the static HTML. Using Grunt (*Grunt.js* 2013) and sublime made getting out a static site very quick and easy. This stage did not take as long as I had originally planned. It was done within three days, where as originally it was to take two weeks. However, as design has not being fully completed, what is commonly known as designing in the browser took place. This is where a rough idea of what the page is to look like is in the developer's head and they just design it in HTML and CSS straight away, without going through the step by step process of wire framing and mock-ups. While this is not a good idea on a large scale project, the minimalist design that the application was adopting made it easier to design in the browser than to draw wireframes and create mock-ups.

Once the static site had been generated and the designs had been finalised it was straight into Rails development. The development was split up into features and each feature was implemented separately. This allowed deadlines to be set so it was easy to track the progress of development. The features that were split up were;

1. A user system with the ability to create a new user and login and out

2. Weight tracking, for both current and goal weights

3. Fitness log

4. Goal tracking

Once these deadlines were confirmed, development could properly begin.

Not much knowledge was known about Ruby on Rails before development, other than some basic syntax and that it is an MVC framework. So at first, the approach was to learn and build at the same time. This led to two outcomes and the first problem; it was very slow and the code was getting sloppy. This type of development style did not last very long and development came to a standstill when trying to implement the weight tracking. The decision was made to halt development and take the time to learn some Rails.

Two main resources were used for learning rails. The first one was a tutorial by Michael Hartl (Hartl 2013). This tutorial took you through making a Twitter (Dorsey et al. 2006) clone. It was useful because it went through how to create a authentication/authorisation system, create relationships between the user and other models, and also some test driven development. The second was the Rails guides (*RailsGuids* n.d.). These covered pretty much all parts of Rails so they were referred to as and whilst there was a problem that could not be solved by referring to the documentation. The Rails Documentation was also referred to a lot during development.

After a firm understanding of Rails was gained, development could continue. The downside to this break meant that development was now far behind schedule. As development went on though, it caught itself up. The learning break was worth it because development began to speed up and the code that was being produced was to a higher standard.

So far, only the user system had been implemented and it was time to move on to the weight tracking. This was where the problem of not knowing enough Rails to continue arose, which was solved by taking the time to learn it instead of learning as development went on. So when it was time to come back to it, all the weight tracking code that was used prior to learning Rails was removed. This was made easy with Git. Git allows you to reset changes to a certain commit. The code was rolled back to where the user system was just finished and so was the Rails database. For the weight tracking, two separate models were used. Originally, the goal weight was going to be inside the user model but the decision was made to split it off so that it could possibly be used for future analysis. For example; plotting a graph for both the user's current weight and their goal weights against time so they will be able to see when they hit their goals. This also led to splitting up the goal weight and the current weight into tables. Another factor into this was the thought that the user will be updating their current weight much more often than their goal weight. So to have them in the same table would lead to multiple duplicate goal weights. These weights were to be shown on the dashboard and updated through a different page. In the original design, there was going to be a little graph under where the weights are displayed on the dashboard. This idea was removed because of how small the graph would be. It was moved to the weight input page so that it could be a lot bigger and the user would be able to gain more information from it. The weight input screen was kept really simple. It has two input boxes and two submit buttons, one for the goal weight and one for the current weight. Rails allows for very simple AJAX integration with forms. AJAX allows the user to update the database without having to leave the current page. This will allow the user to update their weights and then consult the graph without having to load the page again. It will be a great help on mobiles where the data connection may not be great.

Once the weight tracking was implemented it was on to the next feature which is the fitness log. The fitness log has its own table which consists of six fields:

1. Activity - The name of the exercise

2. Type - What type of exercise (strength, cardio)

3. Time - How long the user spent doing the exercise

4. Reps - How many repetitions the user did

5. Sets - How many sets the user did

6. Weight - How heavy the weight was

There are two main types of exercises; strength and cardio. There were two options when it came to storing this information. Option one was to have three tables; fitness log, cardio, and strength. The fitness log would just store the user's ID and the ID of either a cardio entry or a strength entry. Or option two, which is to have them all in one table and use some logic in the controller and view to determine what to do with the data dependent upon whether it is a strength exercise or a cardio exercise. The reason for needing to either split them up or use some logic is because a cardio exercise does not have any sets, reps or weight. They are usually a time based exercise. Whereas, it is the other way around for strength training. The user will want to track their sets, reps, and weight, and not how long it took them to perform the exercise. The decision was made to keep them all in the same table because it was easier to implement. Both ways would need some logic at either the inputting into the database end or the extracting end. The logic that was used in the view controller for showing the fitness log was very simple. There is a foreach loop that prints out the information and if the type was equal to cardio then it would print out the format "Activity - time", for example "Cycling - 120 mins", and if it was strength it would print out "Activity - sets*reps @ weight", for example "Bench press - 3*6 @ 160 lbs". There is also some logic in the controller that groups the entries by date, so that all the entries for that day will be grouped together.

The application would have suggestions for exercises activities. This was for the same reason as having the drop down menu on type, to add consistency to the inputs and throughout the records in the database. Originally, the suggestions were hardcoded into the HTML using the datalist attribute in HTML5 but then a better solution occurred. Having the list hardcoded in to the HTML meant that it was hard to update and keep on top of. Also, what happens if the user did something that was not on the list? If they did that again, what if they input it without capital letters this time? Whilst changing from capitalised to non-capitalised will not break the application, it is better to have everything in the same format. The solution to this problem was to build the datalist from previous exercises the user has completed. This way, if the user has done a certain exercise repeatedly that was not on our list, they will still get the autocomplete function.

As well as showing a list of all the exercises that the user had completed that day, the application will need to show the users progressing through their time using Recur as this was one of the main goals to keep the user motivated by showing them their progression. Three charts were added to the log view of the website to show their progression on the three main compound weight lifting exercises; the bench press, the deadlift, and the squat. The charts were added using highchart.js (HighsoftAS 2009). A helper gem called LazyHighCharts (Martinez 2010) was used so that the chart could be built in the controller of the application instead of being built in JavaScript in the view of the application. Using the gem made the code a lot easier to manage and change.

Once the fitness log was completed, the date card could be worked upon. At first, the date card was just going to display the current date but the box looked very empty and out of place. To add motivation to the card, a "time since last gym visit" was added to this card. This would work in two ways.

1. Motivate them to keep this time down to as little as possible by getting into the gym as often as they could.

2. Motivate them to get back into the gym after a long break by reminding them that they have not been in a long time.

Using Rails `Date.current()` function to get the current date and the `Created_at()` date from the last fitness log entry from the current user, the application was able to figure out how long it had been since the users last gym visit. Rails also has another great function for this which is `distance_of_time_in_words()` which takes in two date functions and prints out the distance is real words. For example: "less than a minute since your last visit to the gym. or 5 days since your last visit to the gym.".

After the date card, there was only one feature left to implement. This was the goals feature. Goals was a table that stored two values; the goal and complete. Goal was a string and was what the goal was, and complete was boolean. Goals are shown on the dashboard and can be inputed from the `goals_user_path(current_user)` in the Rails application. Goals was not a very hard feature to implement. It was simply just one form that created the element and then a list the showed all the goals with a checkbox to check the goals off. Where it got difficult was with AJAX. The goals needed to be updated through AJAX so as to not have to refresh the page when a goal is checked off. When using AJAX to input the data for the goal, it was throwing a lot of errors. At first, it would input the goal multiple times, and then it would not update the list properly when the goal had been input. The decision was made to move this to a could have priority. As for checking the goals off, the AJAX worked really well. The user will check the box, the goal will then turn red and have strikethrough. After a second the goal will fade and update the record by setting complete to true. Currently, there is no way to uncheck the goal and get it back, the user will have to input. The completed goals are shown below the current goals and they also display their completed on date. With the way the grid works to make the website responsive, on the goals page it will put the first goals card above the input card. This will cause a big problem if the user has a lot of completed goals. To fix this problem, the list will be hidden when the display size is under 650 pixel and a jQuery function will be added to the display this and hide it again.

Now that all the main features are added and a working prototype is complete, all that needs to be done is tidying up. The development stage was still a learning process, so a lot of corners were cut. Mainly on aspects like styling and the feel of the website. So now development will continue by tidying up the loose ends of the application with tasks such as, fixing the responsiveness of the website so that it is completely mobile friendly, and making sure the flashes show when the user submits by AJAX.

After everything was done, it was apparent that there was no way to analysis the data as an admin. This was resolved by adding an access attribute to the user, that would default to zero. If the access of a user was changed to one, then they would have access to the admin panel. The admin is able to see a list of all the users, along with their current and goals weights, how many goals they have against how many they have completed, how many times they have updated their fitness log, and when they last updated their fitness log. They will also be able to view user's pages but not update their information.

5.3 Clean up

Once all of the main features were implemented, a clean up took place. This meant that actions such as catching bugs (that the testing did not pick up) and fixing them, making sure the correct

flash displayed with the correct styling and making sure that the website was accessible from all devices. Most of these had already been picked up during development so there was not a lot to do when it came to cleaning up. The main problem was the flashes. If the form was submitted through AJAX, then the flash would not show, but it would show on the next reload. This was fixed by moving the flash over to a partial and rendering the partial at the top of the page, and then when the AJAX javascript was called, it would update the flash and any other part of the webpage that needed to be dynamically updated. This also solved another problem of having a lot of repeated code. The way to display a flash is the same on every page, and it was repeated on almost every page. Moving it over to partial cut down the code per page.

5.4   Optimisation

Optimisation is a big part of today's web industry. With a lot of people browsing the web from their mobiles, their download speeds are slower than broadband speeds. They also have data caps which restrict the amount of data one user can download in a month. There are certain steps that can be taken to reduce the time it takes for a user to load up the webpage on their mobile devices. The first way we can do this is by manipulating the JavaScript and CSS files. HTTP request will slow down the page load time so we can concatenate the JavaScript and CSS files to reduce the number of HTTP requests we are calling per page. We can remove all of the unnecessary whitespace from this files. This process is called minification. Using both concatenation and minifcation of files, you stand to reduce a large chunk of requests from your website, improving your load time. Other options include optimising your images. Images are the biggest item to load on webpages, ranging from kilobytes to a few megabytes each. There are lots of tools to use for optimising your images. ImageOptim (Icons8 2014) is what was used for optimising the images. Because not many images are used it wasn't necessary to go through all of the image optimisation steps. ImageOptim gave enough results to justify not going further with image optimisation.

5.5   Deployment

Deployment of the web application was done through a service called Heroku. It offers free hosting services for small applications but will automatically scale up when demand for the website rises. This was perfect for the application as it was easy to get a prototype on the web quickly but if demand for the website rose then Heroku would scale up my application to match the demand. It was also chosen for its ease of use. It ties in well with Git and creates a new remote in your local repository to push to. It will then detect what type of application is being deployed, a Rails application in this case, and do all the setting up necessary to a Rails application. The only setup needed on our end was a few changes to the gem file. Adding a production and development group to switch between a MySQL (Oracle 1995) for local testing and PostgrepSQL (*PostgreSQL* 1995) database for Heroku, and adding the Git remote which was made simple with the Heroku tool belt. Heroku also has the option to allow to assign your own domain to it.

Other options were to deploy to a virtual Private server using the Capistrano gem (Buck & Hambley 2013). This was the original idea but after discovering Heroku it seemed unnecessary. Also, with Ruby applications, dependences have to match up, otherwise your application will not run. While you can force gem versions in the gem file, Heroku manages all that for you.

5.6   Conclusion

This chapter discussed the importance of having an understanding of the tools, languages, and frameworks used before undertaking the project. Not knowing enough about the framework that was used caused major delays in the development stage. It also discussed the importance of optimisation. Especially in the case of optimising assets for mobile use. It also talked about the options for deploying the application to the internet, and the pros and cons between the two main options. Deploying to a service, like Heroku, is much easy than setting up a virtual private server. Whilst being easy it can also incur more costs.

Chapter 6

Testing and Evaluation

6.1 Introduction

This chapter will cover the testing and evaluation of application. It will go into detail about the choices made and give reasons as to why certain tools or methodologies were used over others. It will also discuss the first release and reviewed release of the application, and talk about the user's feedback on each release.

6.2 Methodologies

For the main part of testing, test driven development (TDD) was used throughout the building of the application. TDD was used because of the constant deployments to Heroku. All of the features implemented had to be working perfectly before deployment. However, sometimes this was not the case. All of the tests are written so sometimes you miss a test case and this can lead to deploying broken code. Usually, most of the bugs were picked up either with the testing or quite quickly after deployment if any bugs got through. TDD fitted in well with the Agile development approach that was taken when building the application. TDD and Agile development were used to make testing easy. TDD tests the code as you are working so you are not left doing all the test last, or after every feature. This allowed for working prototypes to be deployed very quickly.

6.3 Tools

6.3.1 Rspec

Rspec is a RubyGem that is used to write test cases for test driven development. As test driven development was the chosen methodology, it was important to get the right testing tool for the job. Rspec was chosen for its easy to read and write test cases. Rspec's syntax is designed to human readable. With commands like `should have_content` or `should_not have_link` it makes writing and debugging the test much easier. Rspec can be coupled with other gems like FactoryGirl to make testing even easier. FactoryGirl (*Factory_Girl* 2013) was used to create fake users so testing could go further by interacting with a web page using another gem called Capybara (Nicklas 2013). Capybara allowed interaction with the webpage so we could navigate between pages and see if the correct flashes were showing. All three gems were chosen on their power of being able to test almost anything, their simple syntax which is almost like readable English, and they way they all work seamlessly together.

6.3.2 Chrome Development Tools

Chrome Developer Tools (Google 2014) was a key part to the testing of the layout of this web application. Using the developer tools you are able to target specific parts of the application and see where they are getting their CSS styles from. This was especially useful when reusing class selectors on the cards. The cards would share similar styles but also have some individual styles that would conflict if kept under one class. It was also very good for mobile testing. It also allows the user to edit the CSS selectors and their values, without reloading the page or changing any files. This proved useful for finding ways to fix certain layout issues as it does not save the changes. The Chrome Developer Tools has an emulation mode, which allows the user to see what the website would look like on a selection of devices. This proved more useful than just resizing the browser window as it took viewport into account.

6.4 Testing

Test were written before the majority of code. This is under the test driven development methodology. You write a test that you know will fail, and then implement the fix. A test case was written for any foreseeable possibility a user could enter. As original functionality of the application changed, more test cases were implemented.

While running the test, it actually brought up something that was not intended. It brought up that some of the methods of retrieving data from the database had been depreciated. Depreciation is a common downside to using frameworks that have multiple versions. Keeping on top of depreciation is very important when it comes to production application because calling a depreciated method can bring your application to a halt.

6.5 Evaluation

6.5.1 First release

March 11th 2014 marked the first release of Recur. It was posted to the fitness subreddit of the popular image board site Reddit(Ohanian 2005). The idea was to do a small release to pick up on any bugs that got through testing. It proved very useful because the developer will understand how the site works because they build it but users said they did not know where to click when first visiting the site. This subject had been raised by another user and fixed with an image of a chain link to denote a link. After this feedback had been submitted, changes were made to the title links. They were made more obvious when rolled over with a background colour change and also, if it is the user's first visit to the site then they will be shown a message telling them that the titles of the cards are links to more information. Other feedback included users receiving error flashes when really the database was successfully updated. This was because the error flash was getting called if the log entry was saved or not. This caused users to submit multiple entries to the database for the same activity. This then caused another problem of not being able to remove a logged activity after it was put into the database. An obvious feature was overlooked during development but was soon implemented in order to make sure users could delete unwanted logged activities. The way the log was displayed was also rewritten to make it easier to see which data went with which activity. Users also wanted to be able to see which goals they had completed on the dashboard. This was limited down to goals that had been completed in the last 2 weeks.

6.6   Design changes

Throughout the progression of development, although nothing major has changed, the design has changed in a number of ways for a number of different reason. Probably the biggest aspect that has changed during development was the colour of the website. Originally, it was a shade of blue with some red in it to create a dark lavender colour. It has now been changed to a shade of blue with more green in (as opposed to red). The change was purely aesthetic but I feel it was the better choice. The small chart on the front page was dropped quite early on during development. It was very had to implement such a stripped down chart that did not display very much information upon the quick glances it would receive. It was swapped out with a much larger, more descriptive chart on the dedicated weights page. Now the chart is one more click away but it allows the chart more space to display more information and therefore became more useful to the user.

6.7   Conclusion

Using Rspec and its accompanying Gems was a great help when it came to testing. Tests were very simple to read and write, and when they picked up any bugs it was easy to identify which test it failed on and what should have happened. Also, using test driven development allowed the application to be deployed very early on because each feature was tested as it was being developed, when it was finished it was ready to be deployed. It also saved a lot of time at the end of development, and saved time in the overall development because each test case was thought of before writing any code, and because each test case was thought of beforehand, it could be taken into account when coding the feature.

Chapter 7

Conclusion and Future Work

7.1 Learnings

The outcome of building this design has been a great help for me personally. It gave me the opportunity to learn a lot about what goes into an application that has the potential to gain a lot of users. The design process taught me a lot about designing for user accessibility. In the past, when designing my application, it was all about appearance. I now understand that there is more to web design than just aesthetics. It is important that the user is able to use the application easily without much hand holding. The design should ease users towards where they need to go, not just expect them to know exactly how to use it straight away. Also, some users may be visually impaired. This has the potential to cause a lot of problems when they're using the website. If you have not catered for them, it could be impossible for them to interact with the website. This is a major problem as you are alienating a large percentage of the population. On a similar note, being able to support a multitude of browsers and devices is important. As mentioned in the report, with mobile traffic increasing it is very important that the application has the same functionality on mobile as it does in the browser, and across all browsers, not just the main ones. The last thing I learnt was that mock-ups are important. I never used to think that they were and I would just jump straight into the browser, but with this project I found that they are a great help to have something to refer to instead of having to think up where you are going to put element X.

Development taught me that having an understanding of the tools you are going to be using is very important. I started this development with very little knowledge of Ruby on Rails and my project was set back many weeks because of it. I will, from now on, make sure that I have a good understanding of everything that will be involved with my project before I undergo any serious development. The pause in the middle of development (in order to learn more about the language and framework) was a massive set back. If I could go back and do this again I would spend the first few weeks gaining knowledge of the tools and technologies I would be using. Also, I had to learn about security and keeping my users safe from attackers. I was lucky enough to have a lot of the functionality from other tools (bcrypt, PostgreSQL hosted on Heroku) that I did not need to worry about it so much, but it was still something I had to take into consideration.

During testing I found that it is very important to test from the start. I was not using test driven development at the start and it was taking a very long time to make progression because I was finding bugs all the time in previously written code. Once I implemented test driven development, progression sped up and I was able to rapidly progress through and get a releasable product. I also learnt that you can only test so much. Even though I had been testing thoughout development, upon first release almost every user had found a bug. Whilst this was not ideal, it was certainly

helpful that they reported it. At first, I was using the built in Rails testing framework which I found to be restrictive and not as simple as I had seen from Rspec. As soon as I switched over to Rspec I found that I was writing test much quicker and it was easier to debug when an error was found. If I was to start again, I would definitely use Rspec from the beginning.

7.2   Future work

I plan to maintain Recur for the foreseeable future. I feel it has potential to become something useful for a broad spectrum of users. The application could benefit from a new design. Whilst I am satisfied with the design and I think that it fulfills its purpose, it definitely needs reworking in some parts. Also, some of the code could use refactoring. While this is not a major issue, if I was to ever sell this on to someone else, having a clean, commented, and well documented code base is very important.

Another project that could happen with this application is the development of an iOS or Android application. One of the great aspects Ruby on Rails is that it is very easy to extend the functionality of the web application to an Application Programming Interface (API) . This will then allow me to develop a mobile application that will make calls to Recur's API so the users will be able to retrieve and update their information from a native mobile application. Seeing as one of the main goals is to make sure that the application works in mobile browsers and the main use of this application is to update your fitness log on the go, it would make sense for a native mobile application to exist.

References

Buck, J. & Hambley, L. (2013), 'Capistrano'. <Accessed on : 17/02/2014>.
    URL: *http://www.capistranorb.com*

Catlin, H., Weizenbaum, N. & Eppstein, C. (2006), 'Sass'. <Accessed on : 17/02/2014>.
    URL: *http://sass-lang.com*

Coding, B. (2013), 'Sketchapp'. <Accessed on : 10/12/2013>.
    URL: *www.bohemiancoding.com/sketch/*

Dorsey, J., Glass, N., Williams, E. & Stone, B. (2006), 'Twitter'. <Accessed on : 17/02/2014>.
    URL: *http://twitter.com*

Driessen, V. (2010), 'A successful git branching model'. <Accessed on : 17/02/2014>.
    URL: *http://nvie.com/posts/a-successful-git-branching-model/*

Factory_Girl (2013). <Accessed on : 16/02/2014.
    URL: *https://github.com/thoughtbot/factory_girl*

Fitocracy (2007), 'Fitocracy'. <Accessed on : 5/11/2013>.
    URL: *http://www.fitocracy.com*

Franco, B. (2002), *Minimalist Architecture*, Birkhäuser, pp. 96–106.

Google (2014), 'Chrome developer tools'. <Accessed on : 24/03/2014>.
    URL: *https://developers.google.com/chrome-developer-tools/*

Grunt.js (2013). <Accessed on : 17/02/2014>.
    URL: *http://gruntjs.com*

Hansson, D. H. (2013), 'Ruby on rails'. <Accessed on : 21/01/2014>.
    URL: *www.rubyonrails.org*

Hartl, M. (2013), 'Ruby on rails tutorial. learn web development with rails'.

HighsoftAS (2009), 'Highcharts.js'. <Accessed on : 24/03/2014>.
    URL: *http://www.highcharts.com*

Icons8 (2014), 'Imageoptim'. <Accessed on : 24/03/2014>.
    URL: *http://imageoptim.com*

inc., N. (2013), 'Nike unviels nike+ fuelband se and nike+ fuelband app'. <Accessed on : 10/11/2013>.
    URL: *http://nikeinc.com/nike-fuelband/news/nike-unveils-nike-fuelband-se-and-nike-fuelband-app*

Jawbone (2011), 'Jawbone up'. <Accessed on : 8/11/2013>.
URL: *https://jawbone.com/up*

KM, F., MD, C., BK, K. & CL, O. (2012), 'Prevalence of obesity and trends in the distribution of body mass index among us adults, 1999-2010', *JAMA* **307**(5), 491–497.
URL: *+ http://dx.doi.org/10.1001/jama.2012.39*

Lindenbaum, J., Wiggins, A. & Henry, O. (2007), 'Heroku'. <Accessed on : 17/02/2014>.
URL: *http://heroku.com*

Martinez, M. M. (2010), 'Lazyhighcharts gem'. <Accessed on : 24/03/2014>.
URL: *https://github.com/michelson/lazy_high_charts*

Moolenaar, B. (2013), 'Vim'. <Accessed on : 16/02/2014>.
URL: *http://www.vim.org*

Nachman, G. (2011), 'iterm 2'. <Accessed on : 16/02/2014>.
URL: *http://www.iterm2.com*

NHS (2012), 'Colour vision deficiency'. <Accessed on : 10/12/2013>.
URL: *http://www.nhs.uk/conditions/Colour-vision-deficiency/Pages/Introduction.aspx*

NHS (2013), 'Bbenefits of exercise'. <Accessed on : 26/03/2014>.
URL: *http://www.nhs.uk/Livewell/fitness/Pages/Whybeactive.aspx*

Nicklas, J. (2013), 'Capybara'. <Accessed on : 16/02/2014.
URL: *https://github.com/jnicklas/capybara*

Nike (2012), 'Nike+ fuel band'. <Accessed on : 5/11/2013>.
URL: *http://www.nike.com/us*

Ohanian, A. (2005), 'Reddit'. <Accessed on : 14/03/2014>.
URL: *http://reddit.com*

Oracle (1995), 'Mysql'. <Accessed on : 17/02/2014>.
URL: *http://www.mysql.com*

Popescu, B. (2014), 'Dash'. <Accessed on : 17/02/2014>.
URL: *http://kapeli.com/dash*

PostgreSQL (1995). <Accessed on : 17/02/2014>.
URL: *http://www.postgresql.org*

Quaranto, N. (2009), 'Rubygems'. <Accessed on : 24/03/2014>.
URL: *http://rubygems.org*

RailsGuids (n.d.). <Accessed on : 16/02/2014>.
URL: *http://guides.rubyonrails.org*

Russell, R. (2009), 'Oh my zsh'. <Accessed on : 16/02/2014>.
URL: *https://github.com/robbyrussell/oh-my-zsh*

Skinner, J. (2011), 'Sublime text 2'. <Accessed on : 16/02/2014>.
URL: *http://www.sublimetext.com*

Theodore B. Van Itallie, M. (1985), 'Health implications of overweight and obesity in the united states', *American College of PhysiciansAmerican College of Physicians* .

Torvalds, L. (2014), 'Git'. <Accessed on : 16/02/2014>.
　　URL: *http://git-scm.com*

Walkers & Digital (2013), 'Walker sands mobile traffic report q2 2013'. <Accessed on : 10/12/2013>.
　　URL: *http://www.walkersandsdigital.com/Walker-Sands-Mobile-Traffic-Report-Q2-2013*

Appendices

Appendix A

User evaluation feedback

[-] **mferrito** 1 point 21 days ago (**1**|**0**)

I may be missing something, but I am unable to enter information for a specific date. Most of the time when I work out, I write it down on paper and then log in online the next day at work. The fitness log seems to only add things to the current date.

[-] **Penguinoo1** 2 points 21 days ago (**2**|**0**)

Yeah, Excellent, that's it fixed. Tried to submit a cardio session for 45 mins with activity name "Bike" and I get an error which states:

"We're sorry, but something went wrong.

If you are the application owner check the logs for more information."

Same things happens when entering strength information.

Sorry if I sound a bit nitpicking, just thought I would try and help you debug as much as possible.

[-] **Penguinoo1** 2 points 21 days ago (**2**|**0**)

I get a warning from chrome: SSL Error

This is probably not the site that you are looking for! You attempted to reach www.recurapp.co, but instead you actually reached a server identifying itself as *.herokuapp.com. This may be caused by a misconfiguration on the server or by something more serious. An attacker on your network could be trying to get you to visit a fake (and potentially harmful) version of www.recurapp.co. You should not proceed, especially if you have never seen this warning before for this site.

[-] **JUDGE_DREAD6** [+1]Personal Training 1 point 21 days ago (**1**|**0**)

Having problems submitting weight, rep set information. It keeps teling me that something went wrong. I closed out and went back to the sight and everything is there.

edit: I also have multiple instances of my squat and bench exercises now. Unfortunately I can't do 5x5 Squats @ 300lbs three times today.. :S

[-] **JUDGE_DREAD6** [+1]Personal Training 1 point 20 days ago (**1**|**0**)

Awesome thanks, next thing I see is that i can't find a tool to add workouts that I have previously done. is there a way you could have an adjustable date option that just defaults to today's date but you can just type in a date and then add the exercises if you forgot to log the day of the workout?

[−] **PinkfloydPurplefloyd** ⬜ 2 points 21 days ago (**2**|**0**)

One thing I can think of is the interactiveness on the home page, the page with the blue boxes showing weight, goals etc. I was lost for quite a bit before I realised that you had to click the heading of the blue boxes to navigate there.

So here my suggestion is that you should make it 'stick out' more. Maybe make tabs to each of the goals, log etc. at the top of the window so people know instinctively to click as well as the headings of the blue boxes. This is just an anecdote but I love it when websites create multiple pathways to get to where I want to go.

permalink   source   save   report   give gold   save-RES   reply   hide child comments   buffer