Progress Monitoring Web Application

Daniel James Wilson

March 4, 2014

Contents

1 Introduction

This dissertation will cover the whole process behind the building of the progress monitoring web application. The application is for monitoring the users 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 enteries. The application will focus on these three main points only 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 section 3 on page 5 of the paper. The literature review chapter will focus on current state of fitness progress monitor applications that are already available for use. It will discuss both the good and bad points about each application and what this application will do differently. The design chapter will also 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. After that, the development chapter will talk through the development process. It will go into detail about which tools were used and why, the problems and challenges that were faced during development and how they were over come, and also deployment of the web application to the internet. Following the development chapter will be the testing chapter. 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. The evaluation chapter will cover what the outcome of the product was like compared with the original idea. It will also cover feedback from users about their and how that feedback will be acted upon. The conclusion will wrap up the paper with what knowledge has been gained from this build, what would have been done differently, and also possible future outcomes from the product.

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 and that is what my application is going to try and fix. By allowing users to track progress through an easy to use web application, that accessible anywhere via the internet, it will help keep the user motivated 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 of these services and a lot of other features, while other are more stripped down and only offer one or two. My application will focus on two or three areas too keep the user on track and not confuse them with things they do not need to worry about. The areas that my 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 my one. It will give a brief explanation about the service and a break down 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 you 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 my application as simple as possible, I will not be implementing this. Fitocracy is very motivational based with things like challenges, quests, duels, and leaderboards all creating feedback loops you are getting constant rewards and recognition of your achievements. Again, my application will be staying very minimal so I wonâĂŹt be implementing such massive features. Instead, my application shall 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 overwhelming number of features. It has so many features and options all packed into the one screen it is very over whelming and not a nice experience to use. This is the basis of my application. To be stripped down bare of the nonsense that surrounds such applications like Fitocracy and focus on a few key points.

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 wireless syncs with your statistics with your Fitbit account. This is one of many items, known as wearable technology, that can track your 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've 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 can't tell it how much you are lifting, so it canĂ't give you an accurate reading on how many calories you have burnt.

Fitbit focuses on balancing your caloric intake and expenditure to help you hit your goal weight. After inputing 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 out. Focusing on calories in versus calories out is another big 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'ĂŹs taking 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 was similar to mine, to keep the user motivated to get out and get exercising. They use the NikeFuel as a measurement as it's a universal way to measure all kinds of activity (inc. 2013). You can also add your friends and see how they'ĂŹre 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 my application is aiming for.

2.3 Conclusion

To conclude, my 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 wait loss, it will focusing on the fitness side. The decision was made to not implement both sides because to split the focus over two things 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 good types and their caloric value or having to have the user enter both the food type and then the caloric value of that good type. This can lead to an incorrect caloric 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 it this will be achieved. Being a content driven application, the information needed to be readily available on the first page that the user sees, so that if they just wanted a quick view on how they're doing it would all be there as soon as they open the page.

3 Design

3.1 Grid design

Franco (2002) states that the concept of minimalist architecture is to strip everything down to its essential quality and 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 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 add 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 Q2 2013. From this, we can see that more and more people are browsing the internet from the mobile devices. 43% of this traffic was from iPhones. As web designers, we need to make sure that our web applications looks as good on a 320 pixel width devices as it does on a laptop monitor. This lead to the design of the application being based on a six column grid. This was for two main reasons: the first reason 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.

3.2 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 accessability 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 not to cater for their needs then that would be a percentage of the population that cannot use the application. 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 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, I went with Helvetica, 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 stylesheet file.

3.3 Design in the browser

Most of the design for the application was 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 begin during the design stage. This was because there was enough of a design to begin developing some of the application. Once development was underway some things had not been designed yet, for examples; 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 on the fly, in the browser. This allows for rapid prototyping of the applications looks 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.

3.4 Database design

As well as design 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. Reasons for this will be discussed in the development chapter ?? page ??. A number of attributes will be store about the user and will be split up over a number of tables all with a one to many relationship to the user table. The other tables will be current weight, which will store the current weight of the user, goal weight, which will store the goal weight of the user, fitness log, which will store information about what exercises they did and how they did, and the goals table, which will store the users goals. All of these will link back to the users ID. Originally, there was going to be a user?s weight table, which store both the current and goal weight but it was moved to two different tables because itś unlikely that the user will be updating the goal weight as much as they were going to update their current weight. All of these tables will have depend upon their user, so if the user is removed from the database so will all their data.

4 Development

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 walk through, step by step, of the building process highlighting problems and how they were overcome.

4.1 Tools

A number of tools were used during development, some were used more and 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 full functioning web application. Sublime Text was chosen because of it's vast amount of plugins. 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 on due to its rapid development capabilities from all of the navigation being done on the keyboard. But it was not chased due to the steep learning curve as this would eat into development time. The time investment into learning Vim would probably not be beneficial to such a small project but will be a consideration for future projects. During the development of the static site, a JavaScript task runner called Grunt. It was used to compile Sass (Catlin et al. 2006), minify javaScript, and optimise images. When development moved onto Rails it was dropped as Rails compiles Sass and minifies JavaScript in the assists 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's 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 haven't 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 application which allows the user to commit versions of code so that it would be easy to revert back to a previous version of code if something went wrong with development. 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 4.4 of this chapter on page 12. 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.

4.2 Gems

!!Description of what a Gem is and how it's useful!! 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 writing in Sass and compile to CSS at build time. Sass is has a number of benefits over vanilla CSS, like being able to use variables for colours to allow for easier mass colour change. 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.

4.3 Development Log

Development began ahead of schedule because designs were already done. Even though not all the designs had been done, the essentials were there, it was only separate pages that were missing. The colour scheme, font choice, and general feel were all there so development got started with the static HTML being built. 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, 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 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 large scale project, the minimalist design that the application was adopted made it almost easier to design in the browser than 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 develop was split up into features and each feature was implemented separate. This allowed deadlines to be made so it was easy to track the progress and see if development was on track. 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 was known before development, other than some basic syntax and that it's an MVC framework. So at first, the approach was to learn and build at the same time. This lead to two things and the first problem; it was very slow and the code was getting sloppy. This type of development style didn't last very long when development came to a stand still when trying to implement the weight tracking. The decision was made to halt development and take the time to learn some Rails.

Two main sources for learning rails were used. The first one was a tutorial by Marc Someone. 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 tutorials. These covered pretty much all parts of Rails so they were referred to as and when there was a problem that couldn't 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 were the problem of not knowing enough Rails to continue arose, which was solved with 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 changed 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 inside the user model but the decision was to split it off was so that it could possibly used for future analysis. For example; plotting a graph for both the users weight and their goal weights against time so they will be able to see when they hit their goals. This also lead to splitting up the goal weight and the current weight into tables. Another factor into this was the thought that user will be updating their current weight much more than their goal weight. So to have them in the same table would lead to multiple duplicate goal weights. These weight 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. 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 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 is it's 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 one table and use some logic in the controller and view to determine what to do with the data dependant upon whether it's 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 doesn't have any sets, reps or weight. They're usually a time based exercise. Whereas, it's 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 inputing 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's 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's 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 has 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 off. Also, what happens if the user did something that wasn't on the list? If they did that again, what if they inputed it without capitals this time? While changing from capitalised to no 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 does a certain exercise repeatedly that wasn't not on out list, they will still get the autocomplete function.

Once the fitness long was done, the date card could be worked on. 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 way.

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 haven't been in a long time.

Using Rails today "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 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. I wanted the goals to be updated using AJAX, both inputing the goal and checking the goal off. When using AJAX to input the data, I was running into all sort of trouble. At first, it would input the goal multiple times, and then it wouldn't update the last properly when the goal at been input. The decision was to move this to a could have priority. As for checking the goals off, the AJAX works 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 doing now 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 lose ends of the application with things like, fixing the responsiveness of the website so that it completely mobile friendly, and making sure the flashes show when the user submits by AJAX.

4.4 Deployment

Deployment of the web application was done through a service called Heroku. It offers a free hosting services for small application 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 just creates a new remote in your local repository to push to. It will thens 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* PostgreSQL) 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 slightly overkill. Also, with Ruby applications, dependences have to match up, other wise your application will not run. While you can force gem versions in the gem file, Heroku manages all that for you.

5 Testing

6 Evaluation

7 Conclusion

8 Bibliography

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

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

Coding, B. (2013), ‘Sketchapp’. < Dec, 2013 >.
URL: *www.bohemiancoding.com/sketch/*

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

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

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

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

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

Hansson, D. H. (2013), ‘Ruby on rails’. < Jan 21st, 2014 >.
URL: *www.rubyonrails.org*

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*

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

Moolenaar, B. (2013), ‘Vim’. < Feb 16th 2014 >.
URL: *http://www.vim.org*

Nachman, G. (2011), ‘iterm 2’. <Feb 16th 2014 >.
URL: *http://www.iterm2.com*

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

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

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

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

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

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

Skinner, J. (2011), 'Sublime text 2'. < Feb 16th 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'. <Feb 16th 2014>.
URL: *http://git-scm.com*

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