MAJOR PROJECT FINAL REPORT

Interactive Multimedia Design

COM553 & COM559

Peer Support Group 10
Mentor - Giuseppe Trombino

Joshua Whann
B00583382

ACKNOWLEDGEMENTS

Thank you to my Mentor Giuseppe Trombino. Giuseppe provided support and advice throughout the whole project, helping me out several times with solutions to product features. Without his technical knowledge and generosity with his time, this product would have taken much longer to create.

Thank you also to Jeffrey Way, without his screencasts on laracasts.com, my learning process of Laravel would have taken much longer and the back-end of my product wouldn't have worked as smoothly.

CONTENTS

1. INTRODUCTION

1.1 Background

The concept for this major project is a design focused tool to efficiently compare two pieces of code or text e.g. CSS or JavaScript. The tool will allow users to upload two code files via drag and drop or copy and paste into two areas on each side of the website. The users original file should be uploaded to the left area and a version of the same file which has been edited/changed by a lecturer or another developer for example can be uploaded to the right side of the website. The website will then compare the two pieces of code, highlighting lines of code which have differences or have been added. This will allow the user to learn efficiently from others, without having to search through their code for differences.

For frequent users of this tool, an account can be set up. This will not be required to use the tool but can provide extra features for account holders. The user will have the ability to save comparisons, store changes and previously compared files. They will also be able to open up their old comparisons, make more changes to them and share them with others.

The thought process that produced the idea began after working on a project with Eyekiller. When some work was passed on to another developer, the idea was sparked when changes to the project were made by another developer. In order to learn from the more experienced developer, looking at the code for his/her changes to the project is the best way to learn but also very time consuming if working with many lines of code. With a tool that could find the changes in a large css file, a lot of time searching for changes would be saved.

1.2 Contextualise

This concept will be aimed towards lecturers/teachers and students learning to code. It is the perfect way for students to learn from their lecturers e.g. a student learning basic CSS can compare their work on a coding assignment with a correct, working code file, provided by the lecturer.

It will also be targeted towards agencies to teach new developers how they structure code. This could be very valuable to a new employee, allowing them to see the differences in how

they code compared to developers at their new workplace. It can also be useful to revisit changes in different versions of a project.

1.2.1 Similar Solutions

This tool will differ from similar existing products by being much more design focused with and emphasise on user experience and efficiency. Similar products with the same concept are either unstyled or look very out dated with a poor user experience. Many of them are also plugins for text editors. By creating this tool as a website, it eliminates the need for a specific text editor as all developers have their own preferences and use a wide range of different text editors.

Unlike most current examples which target programmers and PC users, this project will focus on design and be aimed towards designers/developers wishing to learn from others code, either as a new student or in a new agency. Rather than using PHP for the comparison tool like some existing examples, JavaScript is used with the help of jsdifflib, a JavaScript library created by Chas Emerick. [*Chas Emerick, 2007*]

One of the most similar examples to compare code line by line is obviously Git. The difference between this product is that is it not used for version control and does not merge the files. It simply compares files to find differences. It is also aimed more towards beginner developers who are starting to learn, so it is very simple and not as intimidating for beginners as Git. It is also for one of/quick comparisons, so there are a lot less steps. The user can simply look up the website, drag and drop their files and their comparison is displayed. There is no need to create projects, upload them or sign up.

1.3 Product Definition

1.3.1 Aim

To provide a tool allowing developers to learn by comparing their code with other developers or other versions of their project. To also become a website used be everyone learning to code.

1.3.2 Objectives

The objectives behind this project are listed as follows:

- Learn more about the technologies and features that will be used. For example an MVC framework and the JavaScript that will be needed to create the comparison tool.
- Design a simple, easy to use interface with good user experience to set the product apart from all other competition. This will be the largest difference from existing products.
- Build the main comparison feature which allows the user to compare two code files and find the differences.
- Implement accounts for more frequent users to save comparisons. Giving the user an option to go back and view, change and share code that they had previously compared.
- Add a drag and drop feature which will load the code from the files into their specific area to be compared.

1.3.3 Scope

Time - Along with a report the product was made in 6 and a half months, completed by the end of April.

Cost - As a university project this project was free to create. All hardware and software was already purchased so there was no extra cost. Hosting is $3.95/month with Bluehost and the difline.com domain name was $14.99/year.

Resources - In terms of hardware and software, a Desktop, Notebook, Tablet and Phone were required for device testing. Photoshop, Illustrator, a Text Editor, GUI Complier, MAMP, FTP client and an MVC framework was also used. For knowledge on the specific features, resources used included JavaScript libraries, tutorials, blogs, articles and threads.

Quality - This project is responsive website with using HTML5 and CSS3 with an MVC framework, meeting the relevant BCS guidelines.

1.3 Overview of Work

1.3.1 Tasks

Planning

- Set out the requirements needed to create a successful project.
- Plan all of the features including the main comparison tool, drag and drop, accounts etc.
- Research technologies and techniques used to create the features e.g. JavaScript Libraries

Design

- Create a website structure with a sitemap
- Build wireframes in Omnigraffle to begin designing the layout of the website
- Design the website, adding typography, colours and images, focusing on user experience

Build

- Main features
    - Build the main comparison tool
    - Add a Drag and drop upload to load files into the tool
    - Create accounts which can save, edit and share comparisons.
- Front end (HTML5, CSS3, JavaScript)
- Back end (PHP, MySQL)
- Create a database to store information for the users with accounts.

Integrate

- Bring all of the features together
- Integrate with an MVC framework (Laravel)

Testing

1.3.2 Duration

Over the 6 and a half months, after some more research into the features, a large focus was put on the design of the product, as this is the biggest difference from similar concepts. However, most of the time was spent on the development of each feature as a lot of time was spent learning and becoming familiar with the new technologies. The specific times set for each stage was displayed in the schedule below.

1.3.3 Schedule

The duration for each task was spread out and displayed in a gantt chart (See Table 1.1). Most of the time spent on this project was in the development phase. There was a lot of learning to do about how to build the features. The feature bar on the gantt chart runs throughout the whole development phase, this counts for all features including the main comparison tool, drag and drop uploads and the accounts. The testing was focused around usability testing along with browser and devices testing, which was done continuously throughout the development.

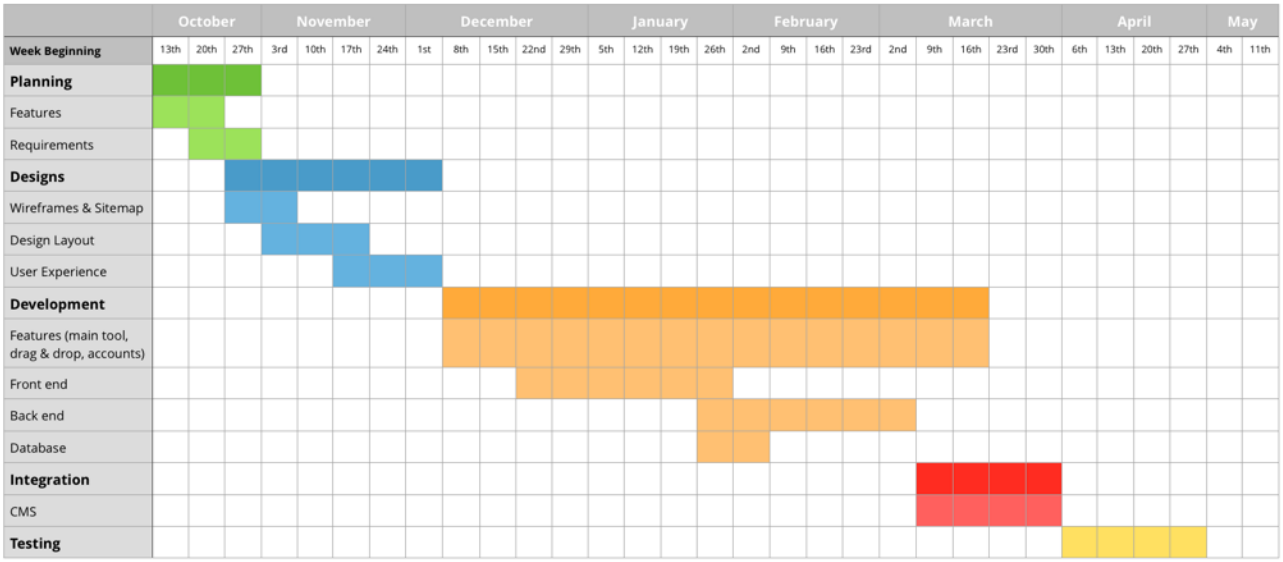

Table 1.1: Schedule

1.4 Rest of the Report

The rest of the report will roughy follow the structure of the waterfall methodology. Beginning with the concept definition. This will include requirements, paper prototypes and the feasibility of the project. Following that will be the designs. The design section will have all of the user experience design, system design and the data designs. The implementation of the designs will follow that with the technologies and risks. The report will then go in to the testing of the product, followed by an evaluation of the product and the process and a conclusion.

2. CONCEPT DEFINITION

2.1 Requirements specification

2.1.1 Purpose

The purpose of the project was to create a comparison tool for developers to compare two pieces of code and be able to see the highlighted differences in the files. Triggered from a situation in an agency when a project was passed on to another developer, seeing what code he had changed quickly and efficiently without scrolling through hundreds of lines of code, was the best way to learn. A lack of quality in comparison tools showed a space for a new product. With design and usability being a main factor in separating this product from competitors, an option for an account to save and share comparisons will also set it apart.

2.1.2 Product Scope

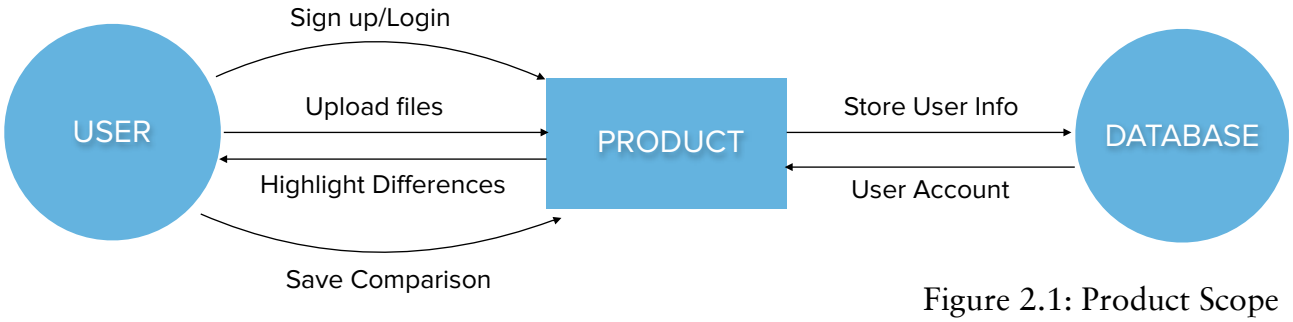

Figure 2.1: Product Scope

2.1.3 Example Use Cases

Some changes were made to the example use cases, both for a general user (See Figure 2.2) and an account holder (See Figure 2.3). The changes have been highlighted in orange boxes. The general user use case changed as the user can edit the files and download them to their device after viewing the highlighted differences in the comparison tool.

For an account user, reset password was added along with download files. The flow of events after opening a comparison or uploading new files was also changed, as it makes more sense with the user firstly viewing the highlighted differences in the code and then having four options - edit, save, share and download. View the initial use case diagrams in [Appendix 1]

General User - Refined

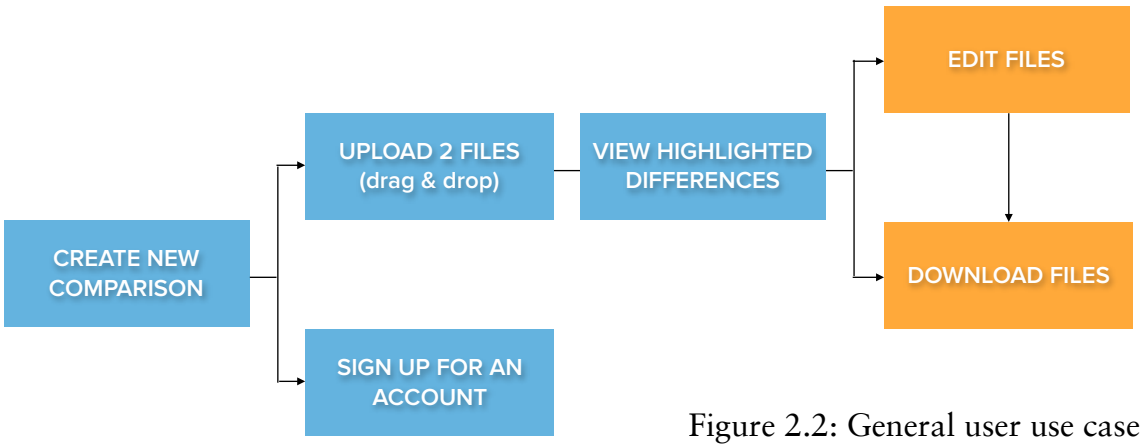

Figure 2.2: General user use case

Account User - Refined

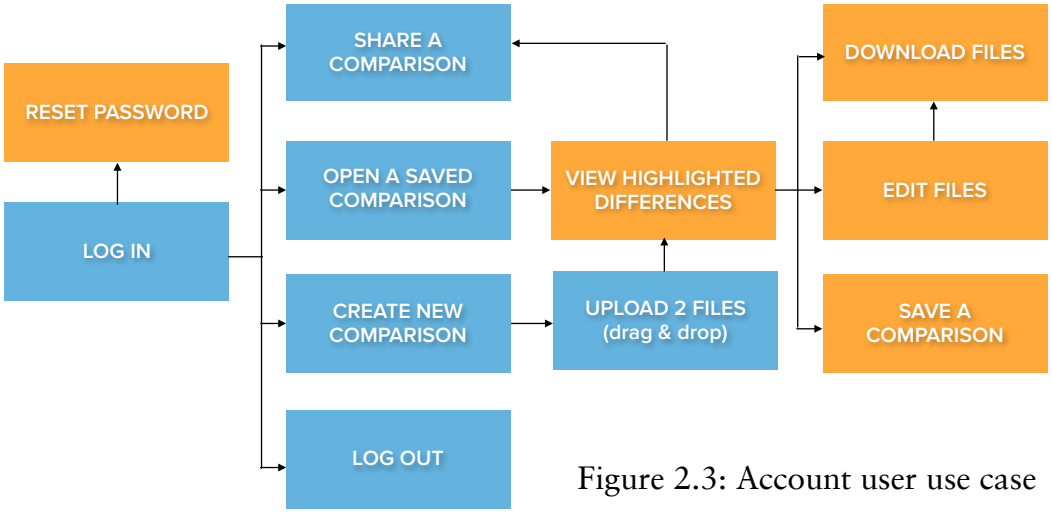

Figure 2.3: Account user use case

2.1.4 Functional Requirements

The functional requirements were mapped out using Volere Snow Cards [*James Robertson, 1995*]. Some of the main functional requirements are displayed below with the rest in [Appendix 2].

Requirement #: 1

Description: The product shall have a login form

Rationale: To allow the user to access their account

Dependancies: Requires the user to have previously signed up

Fit Criterion: The username and password that the user created in the sign up process will give them access to their account dashboard

Requirement #: 2

Description: The product shall have a sign up form

Rationale: To allow the user to create an account

Dependancies: Requires the user to enter details and create a username and password

Fit Criterion: The information entered in the sign up form will be stored in the database and an account will be created for that user

Requirement #: 3

Description: The product shall allow text or code files to be uploaded via Drag & Drop

Rationale: To allow the user to upload their files quickly and efficiently

Dependancies: Requires the user to drag & drop only 2 files. They must be code or text file type and must be dragged into a specific area

Fit Criterion: The contents of the uploaded files will be displayed in the 2 text areas when dropped

Requirement #: 4

Description: The product shall highlight differences in the code files by line

Rationale: To quickly show the user the differences in the files

Dependancies: This requires some differences in the code for anything to be highlighted

Fit Criterion: When different words and characters are on different lines in the two code files, the lines with differences will be highlighted

Requirement #: 5

Description: The product shall save a comparison to a specific user account

Rationale: To allow the user to go back and view previous comparisons

Dependancies: Requires the user to have an account

Fit Criterion: When the user clicks the save button, a comparison block is added to the comparison page of that users account dashboard.

2.1.5 System Requirements

There are some system requirements which are needed in order for the product to operate:

- The system will allow users to drag and drop two files into the comparison tool
- The system will allow users to upload files from a dialogue box
- The system will highlight differences between line by line from the two code files
- The system will allow users to sign up for an account
- The system will allow users to log into their dashboard
- The system will allow account users to save comparisons
- The system will allow account users to share comparisons
- The system will allow account users to delete comparisons

2.1.6 Non-Functional Requirements

The non-functional requirements cover everything including look and feel, usability, performance, operational and environmental, security, cultural and legal requirements. These requirements have not changed throughout the process and can be viewed in [Appendix 3].

2.2 Paper prototyping

2.2.1 Initial Sketches

Sitemap

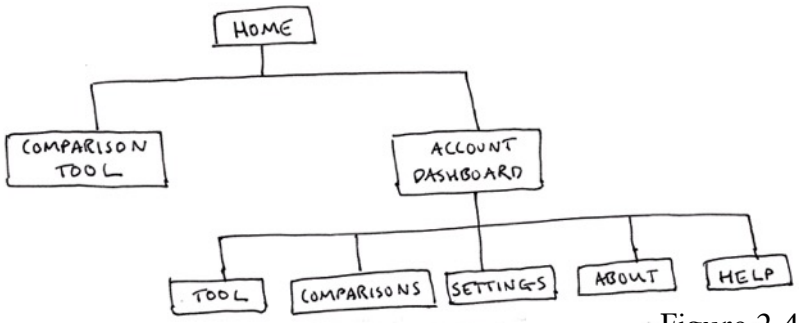

Figure 2.4: Initial Sitemap

Before sketching ideas and layouts for the major project, the structure of the website needed to be created (See Figure 2.4). Rather than having the comparison tool on the index page, a page was needed to explain how the product and the accounts work.

Ideas/Features

The first rough sketch was an idea to display the code comparison tool working on the index page with a CSS animation or an animated gif [Appendix 4]. This wasn't used in the final product.

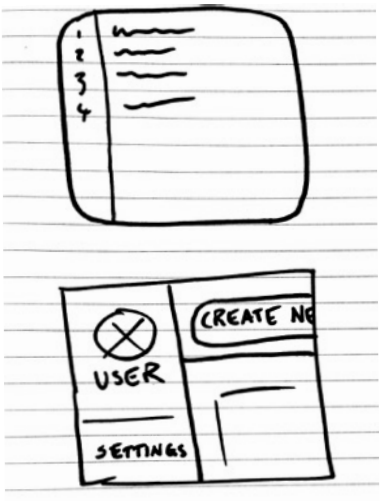

Figure 2.5: Tool sketch

The sketch on the left was an idea for the text area that the code will be displayed in (See Figure 2.5). The lines have been numbered like a text editor. This will make it easier to find differences in large code files.

A rough sketch of a section for the account page displays a dashboard with the users name and image with navigation option below. A create new button in the header with the saved comparisons displayed below. This was refined in later sketches.

Layout

The two sketches below display a rough layout for the comparison tool (See Figure 2.6) followed by the accounts dashboard (See Figure 2.7). This is to get a rough idea of where each section may go on the two main pages. At this stage, the code files being compared will be side by side and the saved comparisons on the dashboard will be displayed in blocks.

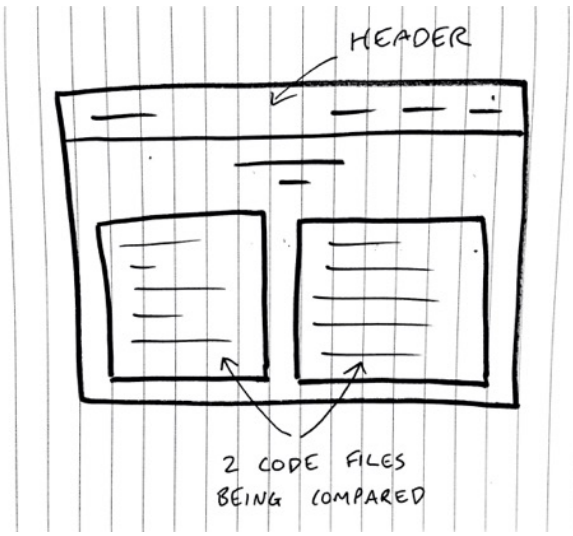

Figure 2.6: Comparison tool sketch

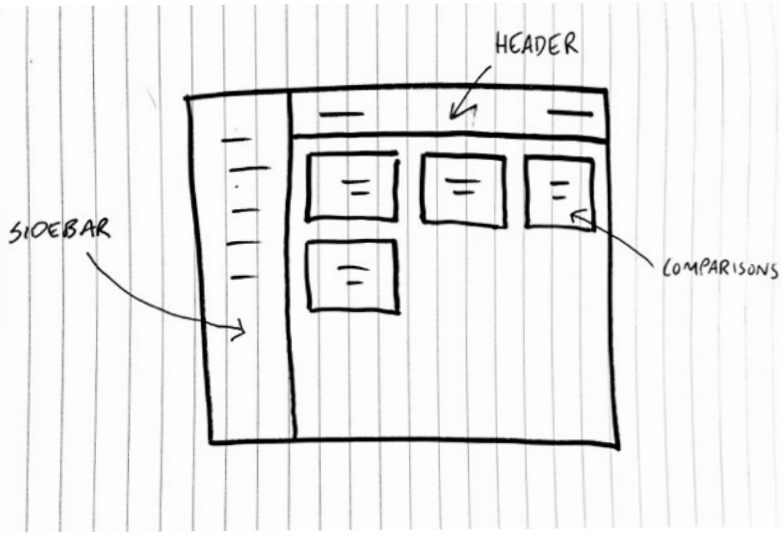

Figure 2.7: Account dashboard sketch

Help

The first iteration of the Help page in the account dashboard was just a simple text area for the user to enter an enquiry and a submit button for it to be emailed.

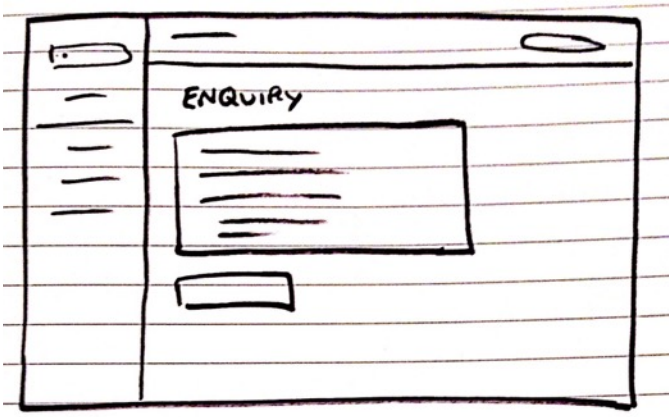

Figure 2.8

Settings

The Settings page will also be on the account dashboard. It will be a form allowing the user to change account details. The rough sketch (See Figure 2.9) displays some of the fields from the Settings page. These fields changed slightly later in the design process.

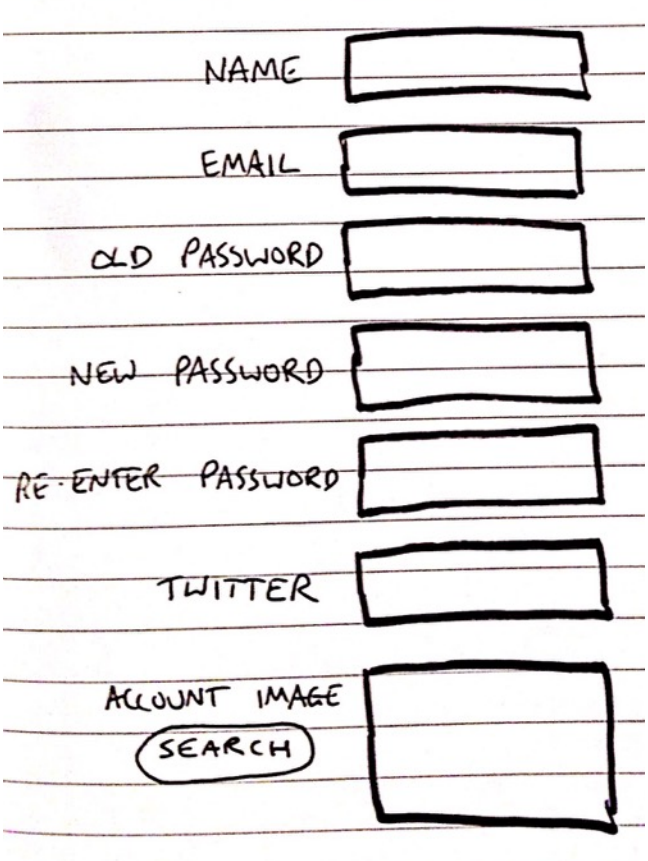

Figure 2.9: Settings page sketch

Login Form

The login form is simply a pop up overlay which asks for a username and password. A 'Forget password?' link and 'Sign Up' button was also created. (See Figure 2.10)

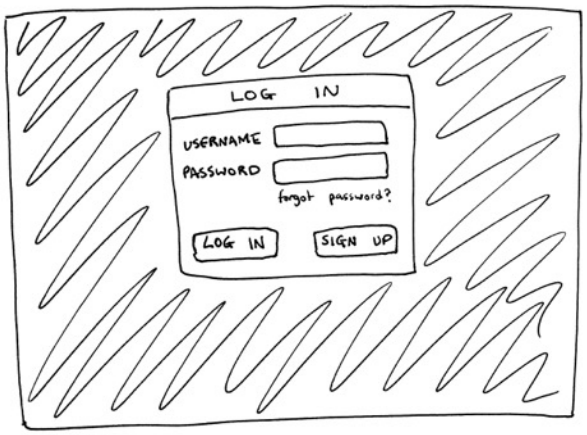

Figure 2.10: Login sketch

Homepage

The index page was needed in order to explain how the tool works, why the user should use it and why they should sign up for an account. The page is very design focused. With a beautiful, quality design, will come the expectation of a quality product. Initial sketches of the homepage display a rough layout (See Figure 2.11/12).

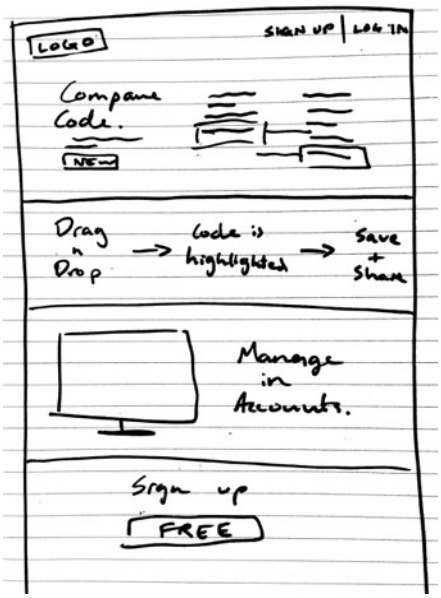

Figure 2.11: Homepage sketch

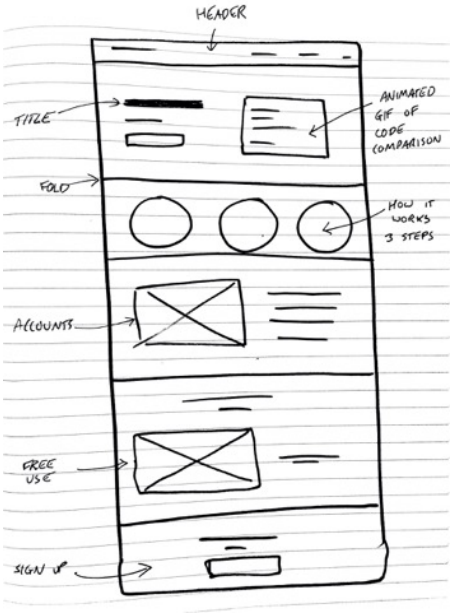

Figure 2.12: Homepage sketch

Responsive

Following some sketches for a responsive design, it became clear that this is not a tool that will ever be used on mobile as no developers code on their mobile device. Responsive design is still a must and will be necessary as a lot of the people who will see this tool, may see it on twitter, which will often be on their mobile. Even though this rough sketch did not go any further, it triggered the need to find a way to get users to sign up for an account or bookmark the website for later use from a mobile device (See Figure 2.12).

Figure 2.12: Responsive sketch

Below is a rough sketch (See Figure 2.13) to figure out what buttons and options the user might have when comparing their code. As well as drag and drop, the user should be able to upload a file. They can also download it again after making changes and share it. A lot of aspects from this initial sketch where kept through the whole design process including the side by side comparison, a share button, a compare/create new button and login/sign up options.

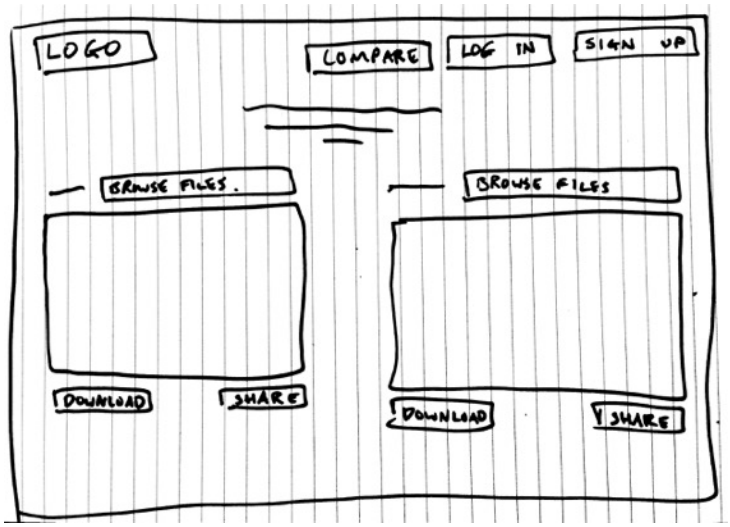

Figure 2.13: Comparison tool sketch

2.2.2 6 Ups

The first set of 6 ups (See Figure 2.14) are for the account dashboard. The pink section is the header, orange is the sidebar and the blocks are the saved comparisons. After sketching the 6 ups, sketch number 1 was chosen to be the the layout taken further for some higher fidelity sketches in slightly more detail.

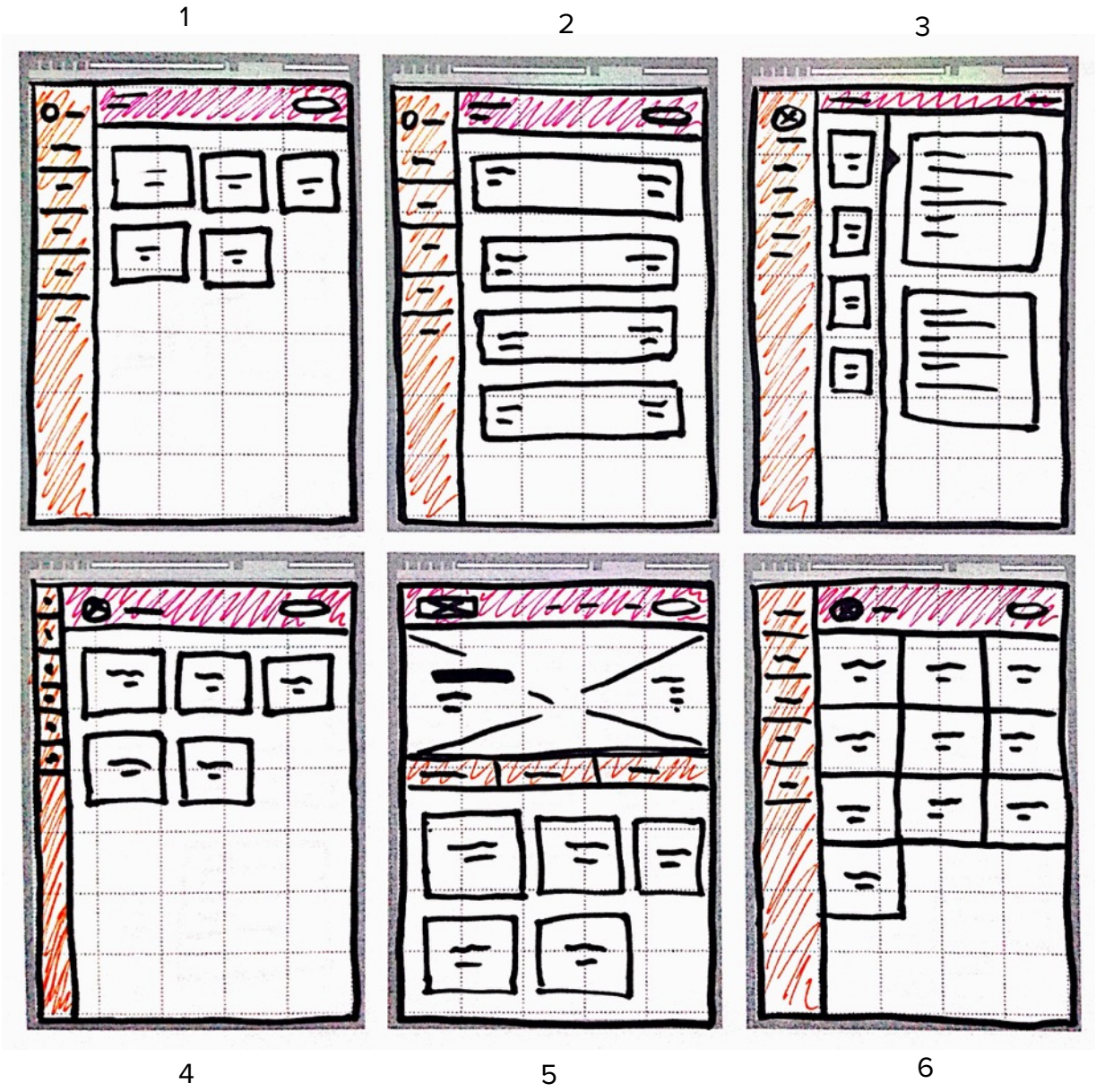

Figure 2.14: Comparison page 6 ups

The second set of 6 ups [Appendix 5] are for the comparison tool. Upon reflection of this account page, the code files displayed vertically may cause some usability issues and the code will be much easier to compare when side by side. In terms of the side by side options, the original vision of the product saw the first sketch being used but after sketching the 6 ups, option number 3 was chosen in order stay consistent with the dashboard layout for the accounts.

2.2.3 Refined Sketches

For the second iteration of the account dashboard, more detail has been added (See Figure 2.15). This has helped in figuring out which options and features the user should have when using their account. In the side bar, an image with the users name has been added. The account opens on the comparisons page which displays previously saved comparisons, a 'Settings' link which allows the user to change account details e.g. email, password. An 'About' link gives more information the the product and the 'Help' link displays a form for user enquiries. The header is very simple with two buttons, 'Create New' and 'Log Out'. Finally on the comparison blocks the user has two options, open the comparison or share it. Following the 6 ups, this slightly more refined sketch became much closer to the final design.

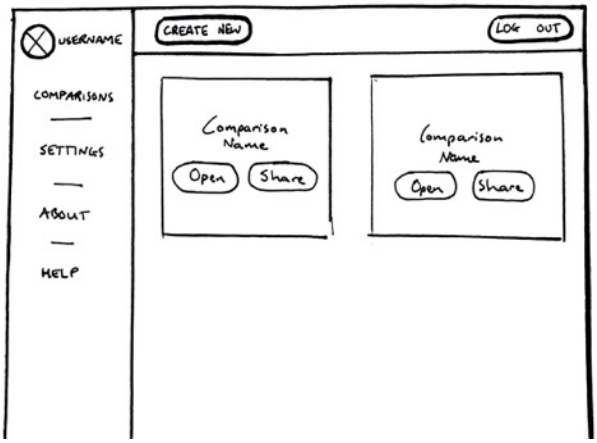

Figure 2.15: Comparison page refined sketch

When thinking about the features and options available to the user on the comparison tool, it became clear there would be different dashboards depending on the user having an account or not (See Figure 2.16). With an account the user is able to save that comparison. They also have a 'comparisons' option which will take them to the account dashboard, displaying all previously saved comparisons.

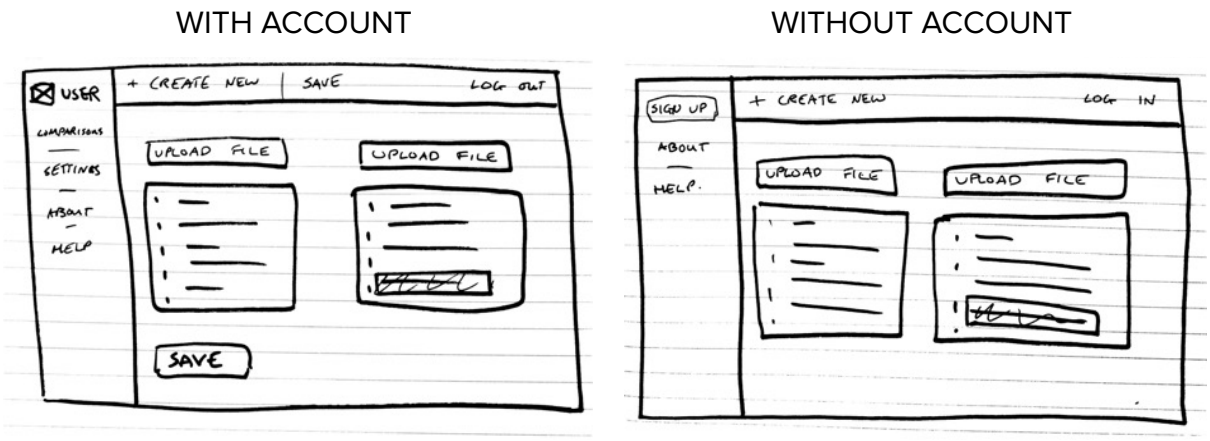

Figure 2.16: Comparison page with and without account

2.2.4 Wireframes

The final iteration of the paper prototype is the detailed wireframes. This is an even more refined version. With the account dashboard (See Figure 2.17), the sidebar has been tidied up and a logo added to the bottom. An extra column of blocks has also been added as two columns in the previous version leaves a very large block, which only needs to be large enough for its name and two buttons. As dashboards can often be very complicated, simplicity is very important for the user experience. This is hopefully reflected in the wireframes.

Account Dashboard

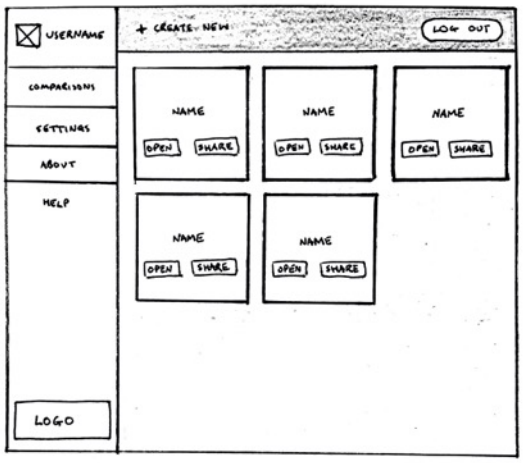

Figure 2.17: Comparisons page wireframe

The comparison tool itself (See Figure 2.18) has developed the most throughout this paper prototype phase. Changing to the dashboard used for the accounts, the wireframe below is for a user without and account. A user with an account will have the sidebar and header of the above wireframe. The user without an account is able to do everything apart from save the comparison, so the wireframe below displays a ‘sign up’ button in the place of the users profile and comparisons link. This was refined further in the design stage, mainly with the repositioning of buttons.

Comparison Tool

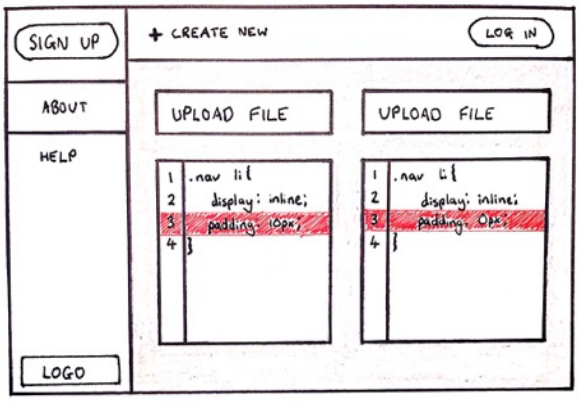

Figure 2.18: Comparison tool wireframe

2.2.5 Style Tiles

The following two style tiles are similar styles in terms of the flat images, outlined icons and ghost buttons. The main differences between the two are the colours and typography which portray completely different styles. The style tile used for this project was style tile v1.0 (See Fig 2.19).

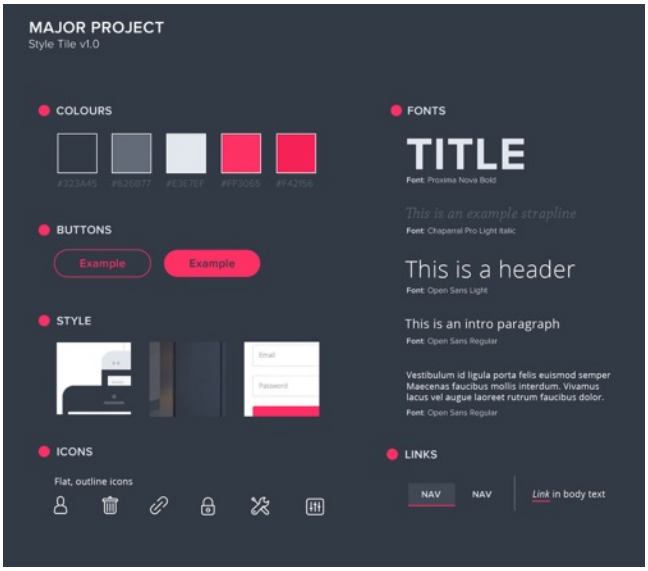

Figure 2.19: Style tile v1.0

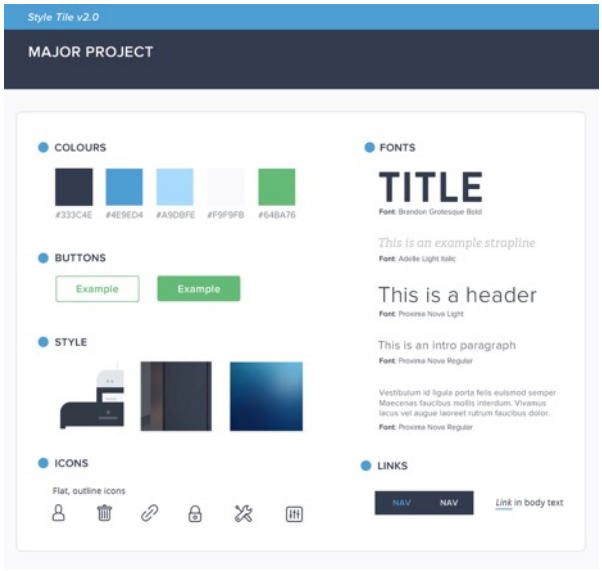

Figure 2.20: Style tile v2.0

2.3 Feasibility

2.3.1 Questionnaire

A short questionnaire was created in order to do some research on the product. This questionnaire is displayed in [Appendix 6]. From the questionnaire it was found that people who use Github which were often back-end/more experienced developers and would use Github's version control which highlights updates. Other people had a comparison tool in their text editor e.g. Atom, but it was found that no one used any website or web application for this which would save the user from having to download and install a plugin. It was also found that some people would use a tool like this for comparing large plain text files or documents and not only code.

At Break Conference a lot more was learnt after the questionnaire. When speaking in person to other designers and developers after the conference, about work and current projects, a lot of great feedback was given about the idea. One developer working for the Government Digital Service had actually been looking for a comparison tool for text files the day before and couldn't find anything. A lecturer who uses a comparison plugin with his first year

students when teaching code, also thought there was a need for a well designed comparison tool in the browser and was keen to try it out with his students when it is built.

2.3.2 Off-The-Shelf Solutions

In the researching of JavaScript libraries, Mergely was discovered [*Jamie Peabody, 2012*]. Mergely is a web application for viewing and merging changes to documents. As a similar product which highlights text in documents, it uses the JavaScript libraries which could be used for this project, jsdifflib and Codemirror, this will be a great reference to view some source code and get a better understanding of how a comparison tool like this will work. It is also distributed under GPL, LGPL and MPL open source licenses.

Mergely was left as a good backup option if the comparison tool is not developed within the time frame. Using Mergely will reduce the technical challenge for this project but if the comparison tool is not developed in time, Mergely can still be used with some improvements. Mergely can be improved by developing the ability to recognise keywords in different code languages and displaying those words in different colours. Currently when a text/code file is uploaded to the comparison tool, the text is displayed in a black font.

An example of some HTML code in Sublime Text 2 text editor (See Figure 2.21), displaying how code looks with coloured syntax. By combining Meagrely with syntax highlighting JavaScript libraries, this was a way of improving Mergely to be used in the final product.

Figure 2.21: Sublime Text syntax highlight

2.3.3 Initial Risks

The initial risk with project is the experience and knowledge with MVC frameworks and the technologies that will be needed in order for the project to work. Following some research into the different frameworks which will be explained in the system design, it was found that many of these frameworks have a steep learning curve. The only issue will be learning a framework and building the product within the time frame for this project.

2.3.4 Costs

There will be no costs for this project as it is a university project so there is no money involved. All of the technologies that are being used are open source and free to use. By using a free, open source PHP web application framework like Laravel over ExpressionEngine with CodeIgniter it will save paying $299 for ExpressionEngine. This may be worth paying for other projects but this project doesn't need a content a CMS and is more suited towards Laravel, which is free. The only other cost is $3.52/month on web hosting and $13.99/year for a domain name.

2.3.5 Waiting Room

There is one feature of the product which was considered but following some advice from developers at Break Conference, it was initially decided not to implement. The feature is the ability to recognise specific code languages when comparing the code files. Before Mergely was being used, it was thought that this may be too complex for the time frame in which the product needed to be created. A syntax highlighting JavaScript library could achieve this but this caused problems with Mergely when tested [*highlight.js, 2012*]. This may be something to add in the future but it will not be a requirement at this stage. By comparing line by line it also facilitates the users wanting to compare plain text files.

2.3.6 Considerations

One consideration for this product will have to be a pricing strategy. Advertisements could be one way of monetising the product but this might effect the clean design and style of the website which is a large factor in the difference from other competition. The other alternative consideration is a freemium strategy, allowing anyone to use the product for free, but for an account which will allow the user to save, edit and share their comparisons, a small fee will be charged.

2.3.7 Holding Page

While Difline was being developed, a simple holding page (See Figure 2.22) was created in order to build some interest and raise awareness about the product release. It also helped with researching into the interest for a product like this. The site uses Mailchimp and has already got some subscribers.

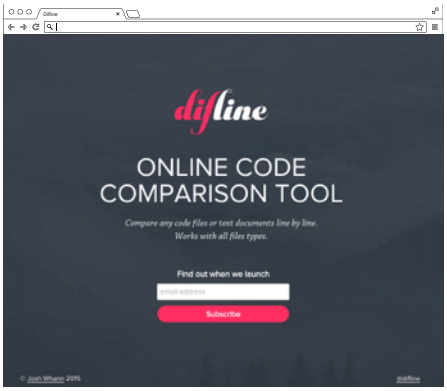

Figure 2.22: Holding page

2.3.8 Subscriptions

When a user subscribes, they will receive a confirmation email and a launch notification. There will be no spam emails and the subscribers will only be notified when Difline is going to launch. Using Mailchimp allows me to keep track of the subscribers and publish campaigns.

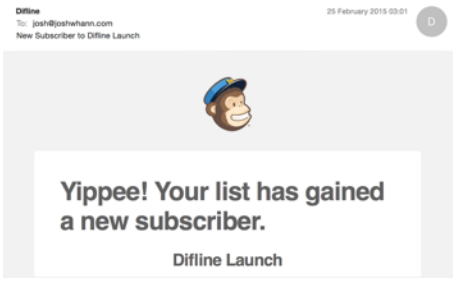

Figure 2.23: Subscribers email

2.3.9 Promotion

Following the development of the holding page, a few Dribbble shots were created. One of the Difline logo (See Figure 2.24) and one of the homepage design (See Figure 2.25). As the target audience will include designer/developers, Dribbble users are a perfect audience to display some designs and a link the holding page too. This will help to find out if there is any interest and want for a product like this. The Dribbble shot of the homepage design was viewed by several hundred people and generated some subscriptions by people wanting to be informed about the product launch.

Figure 2.24: Dribbble shot

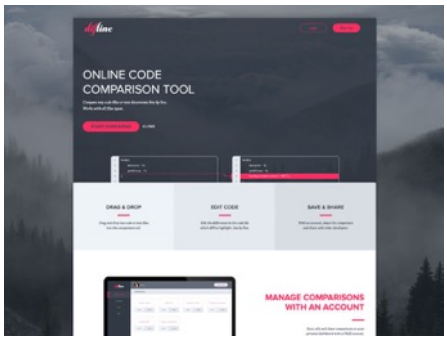

Figure 2.25: Dribbble shot

2.4 Methodology Selection

The methodology that will be used to manage this project will be a waterfall model. As a small project with one large main feature (the comparison tool), the waterfall model works well as a straightforward, easy to manage model. With the addition of accounts for more frequent users, there may be some small features on the users profile added along the way to improve user experience, potentially adapting to a slightly modified waterfall model. Even with this, the requirements are well enough defined for a waterfall model, rather than a prototyping or rapid application development model. As an individual project it is more appropriate than the agile methodology which may be better for a group project.

Figure 2.26: Waterfall methodology

3. DESIGN

3.1 UX design evolution

3.1.1 Refined Wireframes

Following the requirements and system design, the detailed wireframes in the paper prototypes stage have been refined further and digital wireframes have been created in OmniGraffle (See Figure 3.1/2).

Figure 3.1: Comparison tool wireframe

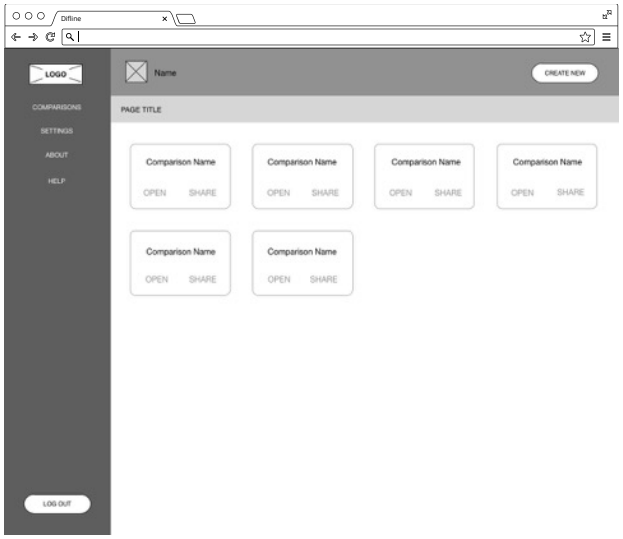

Figure 3.2: Comparisons page wireframe

The position of links on the dashboard have moved e.g. the logo, profile image, create new and log out buttons. A save file button has been added below each text area allowing the user to download the files to their computer after editing.

3.1.2 Branding

As the tool will use Mergely to compare the **differences** between two code files, **line** by **line**, difline seemed fitting as a name for the product. Especially as the domain and twitter handle were available (See Figure 3.2).

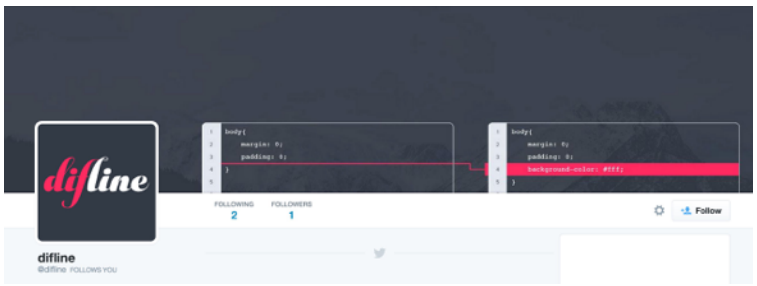

Figure 3.2: Twitter page

After experimenting with many different fonts and icons for a logo. It was decided the logo would just be a name (diffline), with no icon. The font 'Aparo' was chosen, the original can be seen below (See Figure 3.3). This font offers a range of Glyphs which following some experimenting, ended with the dropping of an 'F' from diffline. For the difline logo development.

Figure 3.3: Logo progression

The final logo (See Figure 3.3) is two separate colours with a joint glyph, joining the letters 'F' and 'L'. The pink 'dif' joining to the 'L' of 'line' represents the pink highlighted line of code, joining to the other code file to show where the line of code goes. This is displayed in the product (See Figure 3.4).

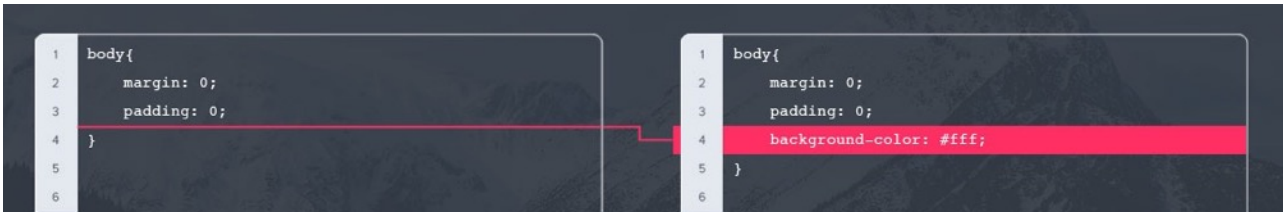

Figure 3.4: Homepage image

3.1.3 Users Flow

With only some minor changes and additions to the refined designs, a user flow was created in order to outline the process of getting and keeping customers. (See Figure 3.5)

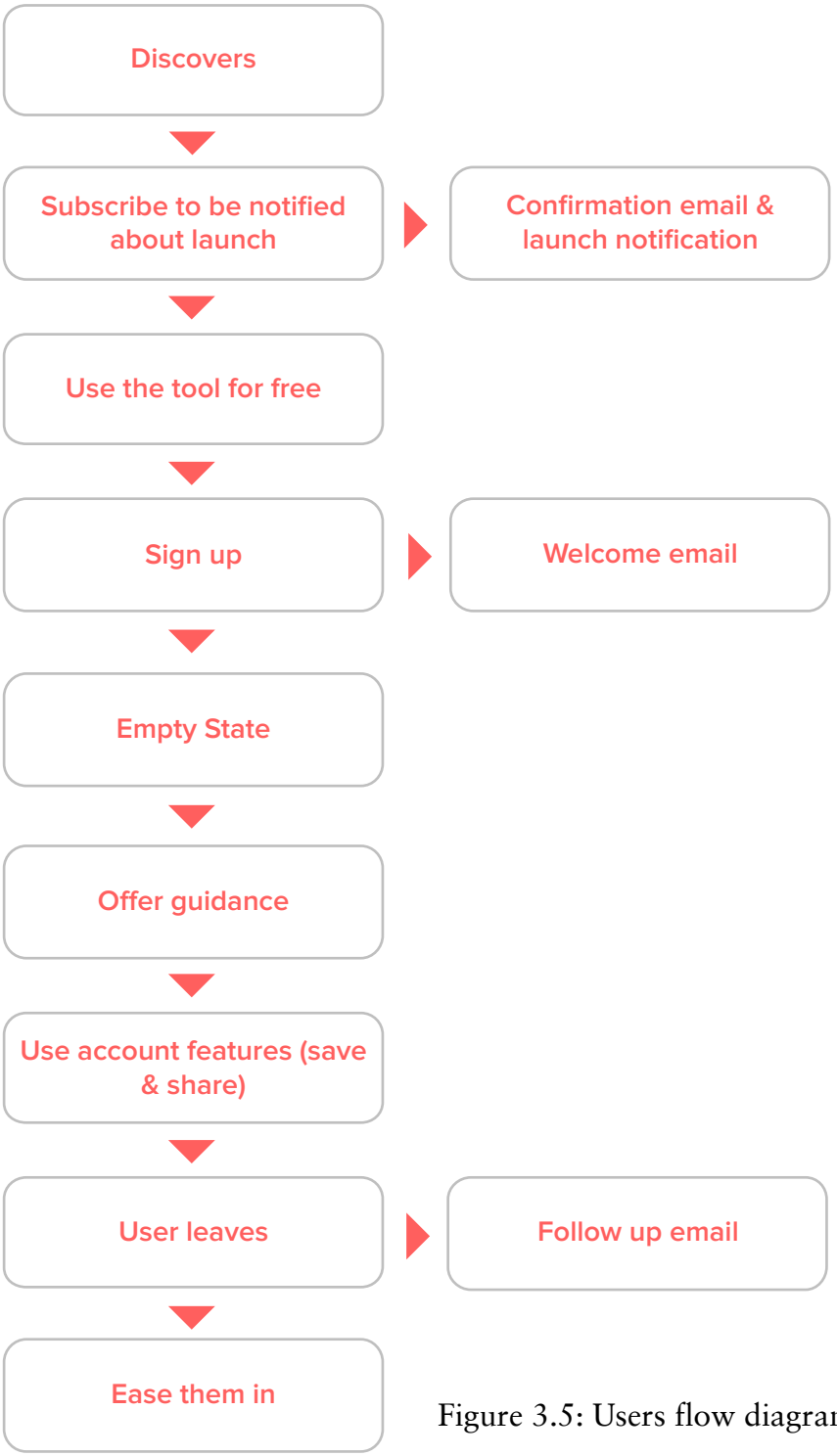

Figure 3.5: Users flow diagram

3.1.4 Users Journey

Below is a users journey through difline (See Figure 3.6), displaying the interactions as an account holder and non-account holder. Each page can be viewed in more detail in the following pages.

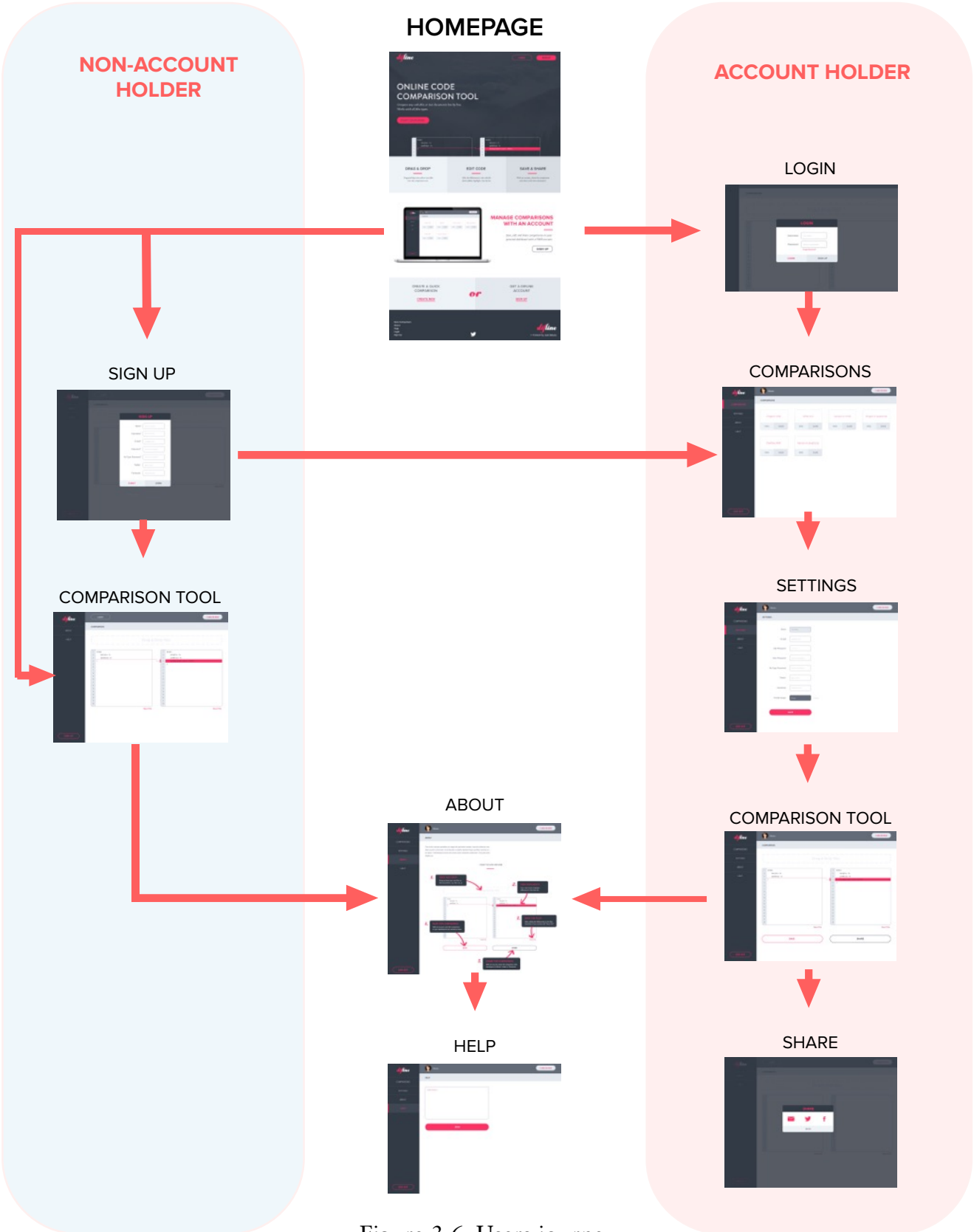

Figure 3.6: Users journey.

3.1.5 Homepage

Above the fold, the homepage displays an image of how the tool highlights code, there are also links to sign up, log in and start a comparison (See Figure 3.7).

Just below the fold, the product is explained in three simple steps. The user has another option to sign up with a visual of an account and a description of the perks. Having scrolled to the bottom of the page to find out more about the product, the user can start a comparison or sign up from here without having to scroll to the top again.

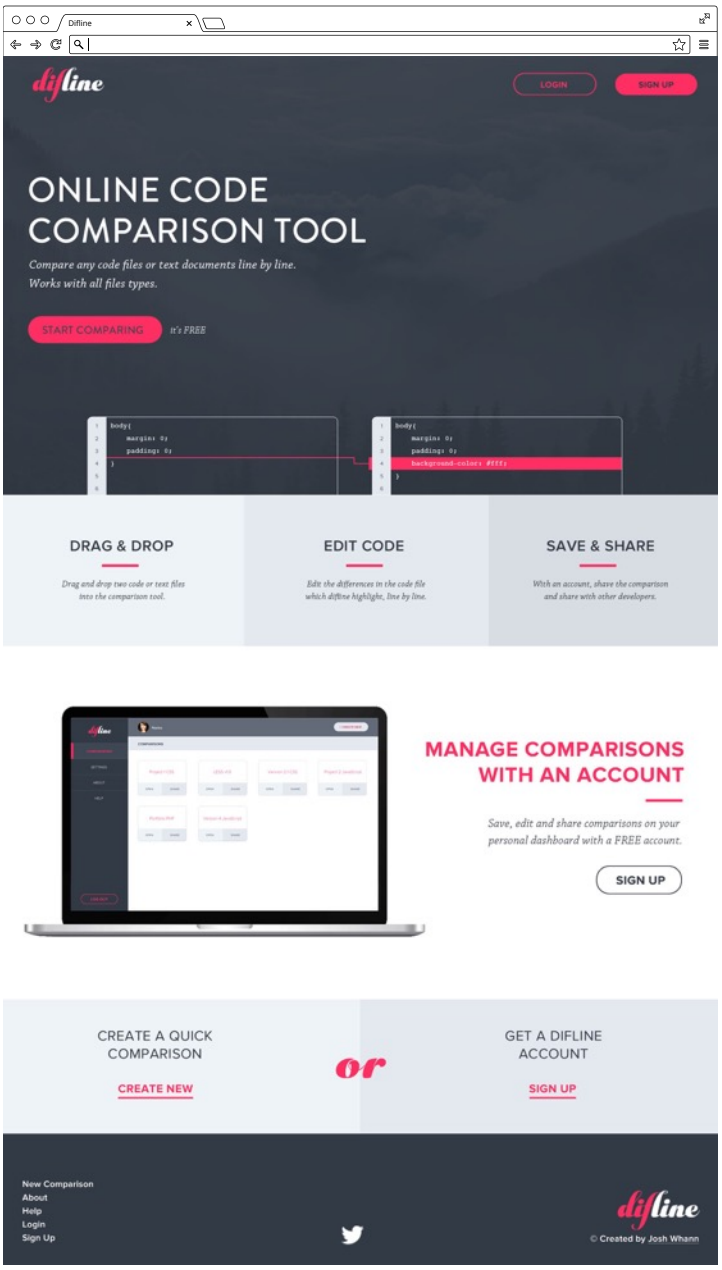

Figure 3.7: Homepage design

Navigation

Following the prototype, there are several user experience design changes and some additions to the navigation. Starting with the homepage, there is a small change, displaying different options depending on the user being logged in to their account or not. Below is the navigation options with Login and Sign Up options (See Figure 3.8) and Log Out and Account options (See Figure 3.9). A check is done to see if the user is logged into their account or not and the appropriate navigation is then displayed.

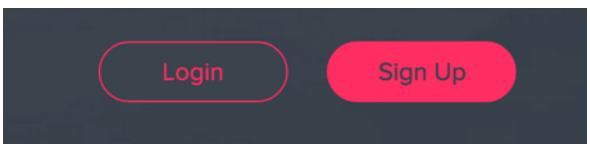

Figure 3.8: Login and Sign up

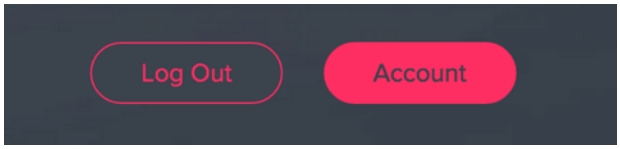

Figure 3.9: Log Out and Account

3.1.6 Responsive

For the responsive design, all links to the comparison tool and logins have been hidden. This is because the tool will not be available on mobile or tablet. The tool would not work on a device as small as a mobile and users will not be coding on a mobile or tablet. The responsive homepage (See Figure 3.13) will explain the tool, how it works and direct the user to return on a desktop as that is the device they will using when writing code.

The two images below will be at the top and bottom of the mobile and tablet design, ensuring it is the first and last thing that the user will see. (See Figure 3.10/11)

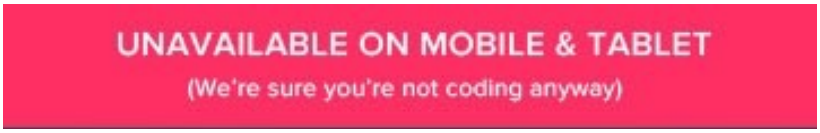

Figure 3.10: Mobile alert

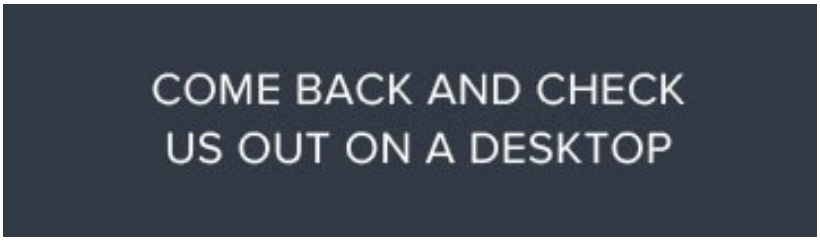

Figure 3.11: Mobile alert

Everything between these alerts will explain how the tool works, with all of the information from the desktop homepage. The only differences are the links and the size of some images which display the tool in more detail (See Figure 3.12).

Figure 3.12: Mobile image

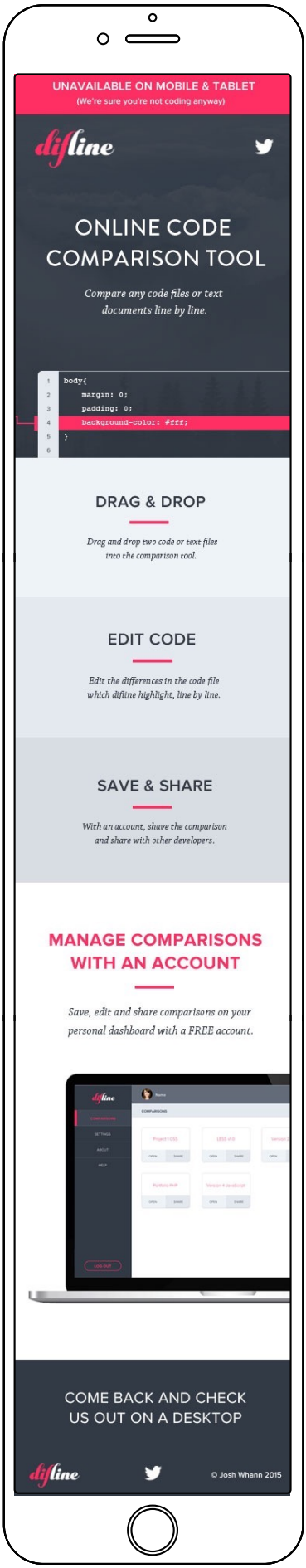

Figure 3.13: Mobile design

3.1.7 Account User

Log in

When a user with an account clicks the login button, this simple login form appears. If the user hasn't already signed up, they can from this screen (See Figure 3.14). They can also reset their password from this page.

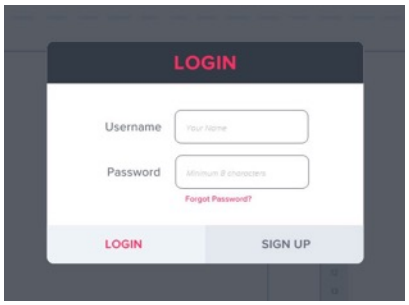

Figure 3.14: Login

Share

The user interacts with a similar pop up if they choose to share a comparison (See Figure 3.15). This is only available with an account. The original design [Appendix 7] allowed the user to share via Email, Twitter or Facebook. The final design generates a URL which the user can copy and paste to each of these services. Twitter and Facebooks API's will be introduced in the future.

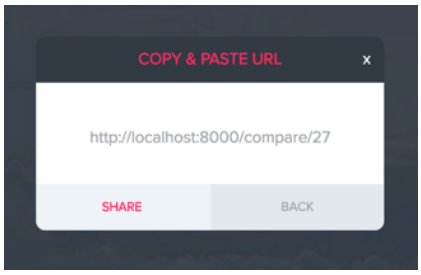

Figure 3.15: Share

Comparison Tool

Below is the dashboard an account user will see when using the comparison tool (See Figure 3.16). This account allows the user to save and share their comparisons, the comparisons link in the side bar is where saved comparisons can be found. This design has evolved from the original design [Appendix 8] as the download file button has been removed and a comparison name input has been added. This is due to the save feature needing a comparison name. This feature also didn't work with the download feature which was less important so it was removed. When the user drags and drops their files, the code is loaded into the text areas and the different lines in the files are highlighted. The user is also shown where the line goes.

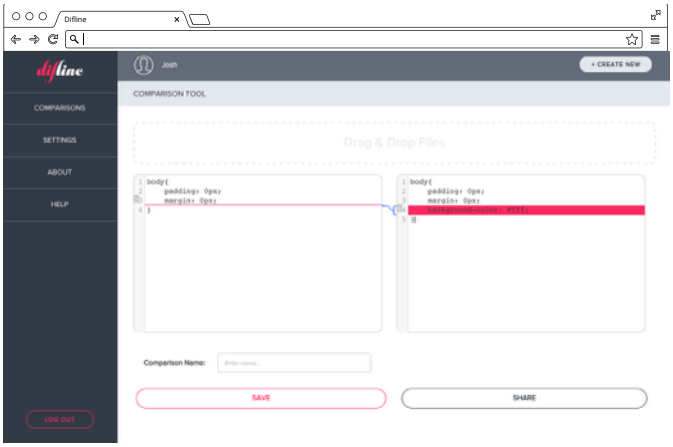

Figure 3.16: Comparison tool design

The design of the comparison tool was been altered to make the code uploaded to the tool highlight keywords in code files like a text editor, rather than all black text (See Figure 3.17). This clashed with Mergely and was not implemented.

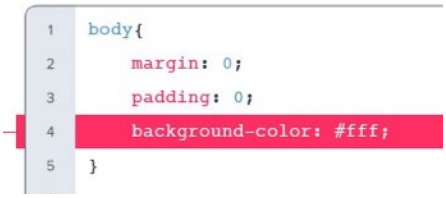

Figure 3.17: Syntax highlight

As well as changing the font colour of keywords in the code files. The name of the uploaded files will be displayed below the text area. This will only display when text files have been dragged and dropped into the text area. The image below shows the file names file_1.css and file_2.css in the comparison tool (See Figure 3.18). This also affected the save feature and was removed from the final version. This will be reimplemented in the future.

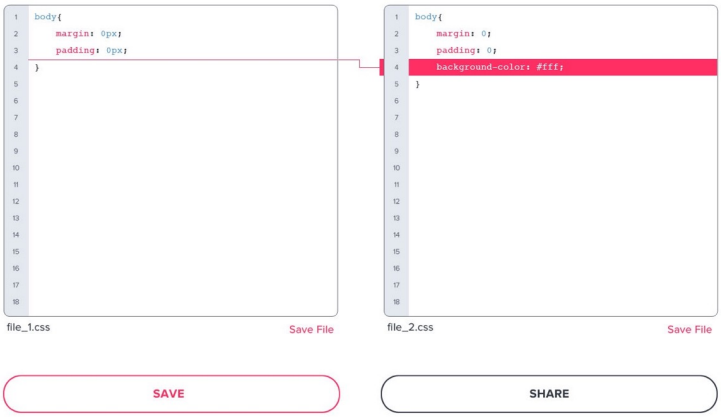

Figure 3.18: Comparison tool with highlighted syntax

Comparisons Page

Any saved comparisons are displayed on the comparisons page (See Figure 3.19). Each comparison block will be given a personal name and the option to open or share the comparison. The ‘Open’ link will open the tool with the code files loaded and highlighted and the ‘Share’ link will open the share pop up.

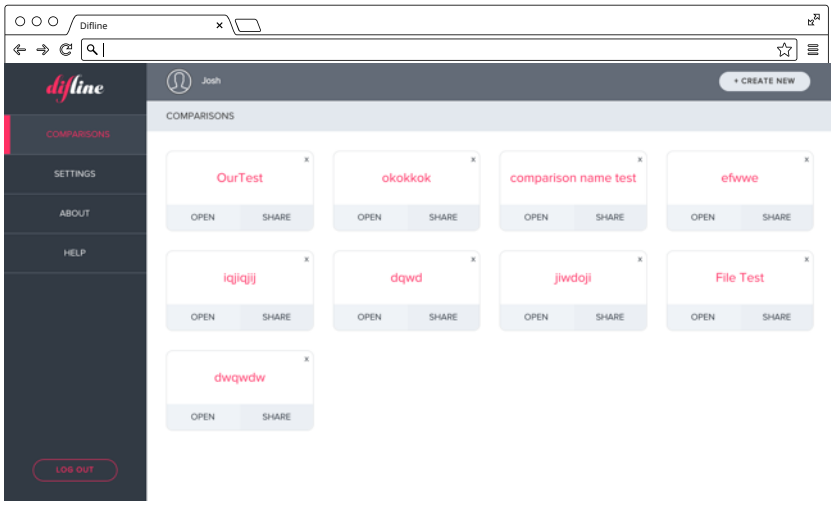

Figure 3.19: Comparisons page design

Settings Page

On the original designs [Appendix 9] the settings page would allow the user to personalise their account by adding a profile image and their Twitter and Facebook account's for the share feature. This will all be added in the future but the final version simply includes the details entered when the user signs up. This allows users to change their name, email and password. The extra fields are not necessary but can be added in the future. [Appendix 9]

About Page

The About page can be viewed by both account holders and non-account holders. This page gives a short paragraph about difline and what it is. There is also a 'How To' section, explaining all of the tools features in 5 steps (See Figure 3.20).

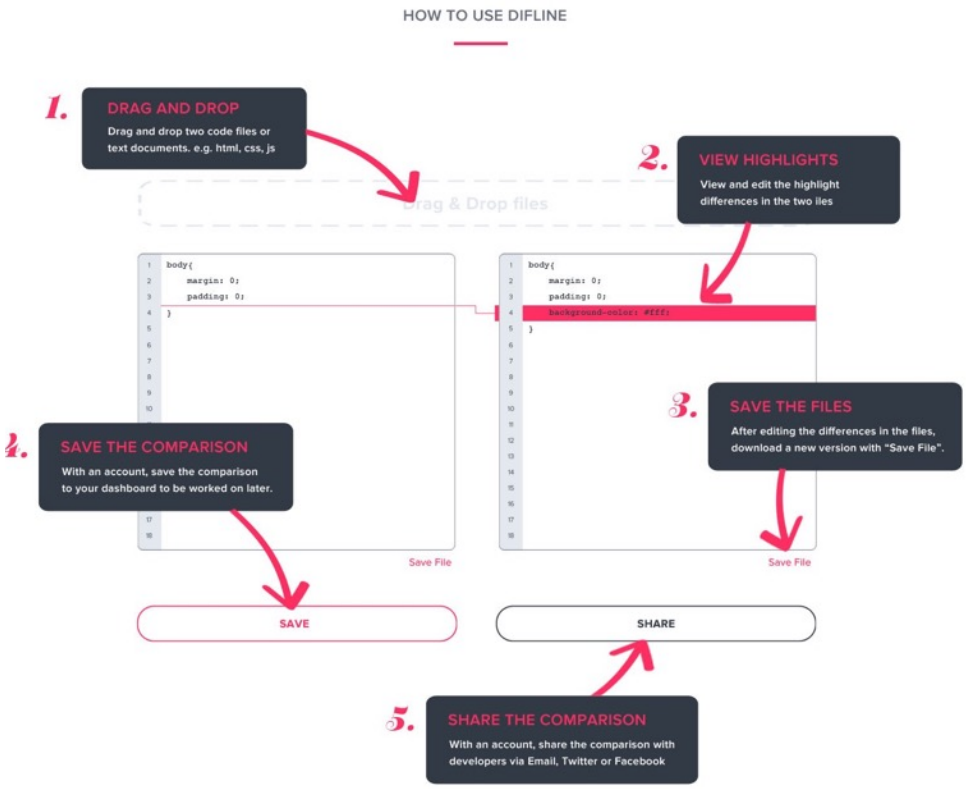

Figure 3.20: About page design

Help Page

The help page is also available to everyone and is a very simple page allowing the user to interact with difline by sending any enquiries they have about the product. Using the same dashboard, the content includes a simple form with a text area for enquiries. [Appendix 10]

Forget Password

For users with an account, a pop up for the forget password link was still needed (See Figure 3.21). This will be a simple form which only asks the user for their email address. A reset link will then be sent to that email, making the whole process very simple for the user. A back button was also added to the original design [Appendix 11] as the X may not be as clear to all users.

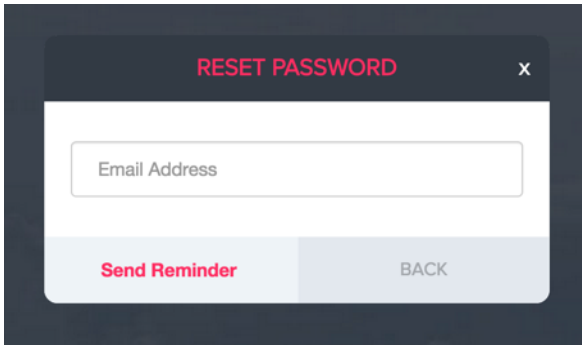

Figure 3.21: Reset password

Comparison Page (Delete Option)

Following the prototype, it was realised that there was no delete option for a saved comparison. Below are two design options for the delete button (See Figure 3.22/23). Both options were be tested but for this iteration, the left option (See Figure 3.22) was be used as it has a more subtle delete button. The delete option doesn't need to be as prominent as the open and share options, as it will probably be used less often.

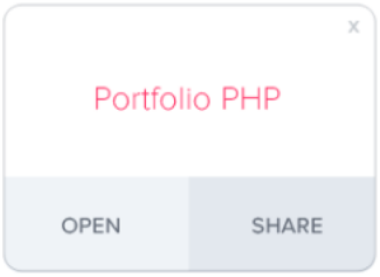

Figure 3.22: Delete block

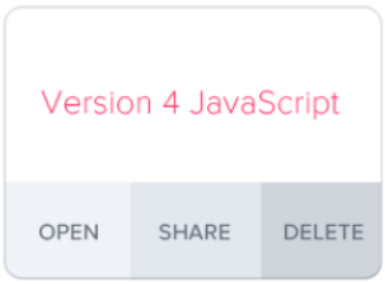

Figure 3.23: Delete block

When the delete button has been pressed the pop up below will be displayed (See Figure 2.24). All of the pop ups have been designed to make the experience very simple for the user.

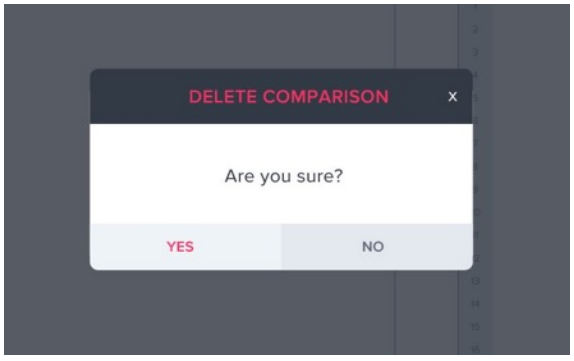

Figure 3.24: Delete option

The full comparisons page, (See Figure 3.19) with the delete option added. This is designed in a way that is very subtle but if the user wanted to delete a comparison, it is very easy for the user to do. With the button in the top corner of each comparison, it is where a close/ delete button would be expected, similar to a browser. When a user deletes a comparison, they are returned to this dashboard with an alert message which will be displayed in the next section.

Alert messages

Alert messages at the top of certain pages have been added to notify the user when they have successfully completed events (See Figure 3.25). Below are designs for alert messages when the user successfully signs up and for when they log out. The alert after signing up uses the same design and can be viewed in [Appendix 12].

Figure 3.25: Alert message

Alert messages have also been added for errors. Below are error messages if the username or password is incorrect when a user is attempting to log on (See Figure 3.26). The alert message is a simple pink bar along the top of the page.

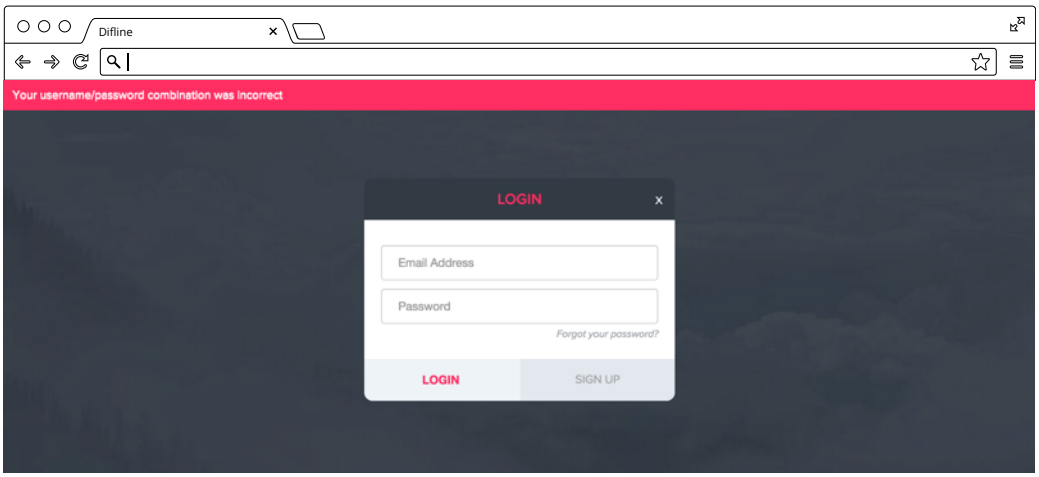

Figure 3.26: Error message

On the sign up error, notifications are displayed for each requirement that is not met (See Figure 3.27). This includes the first name and last name being more that 2 letters, a valid email address that hasn't already registered and the password matching the confirm password field.

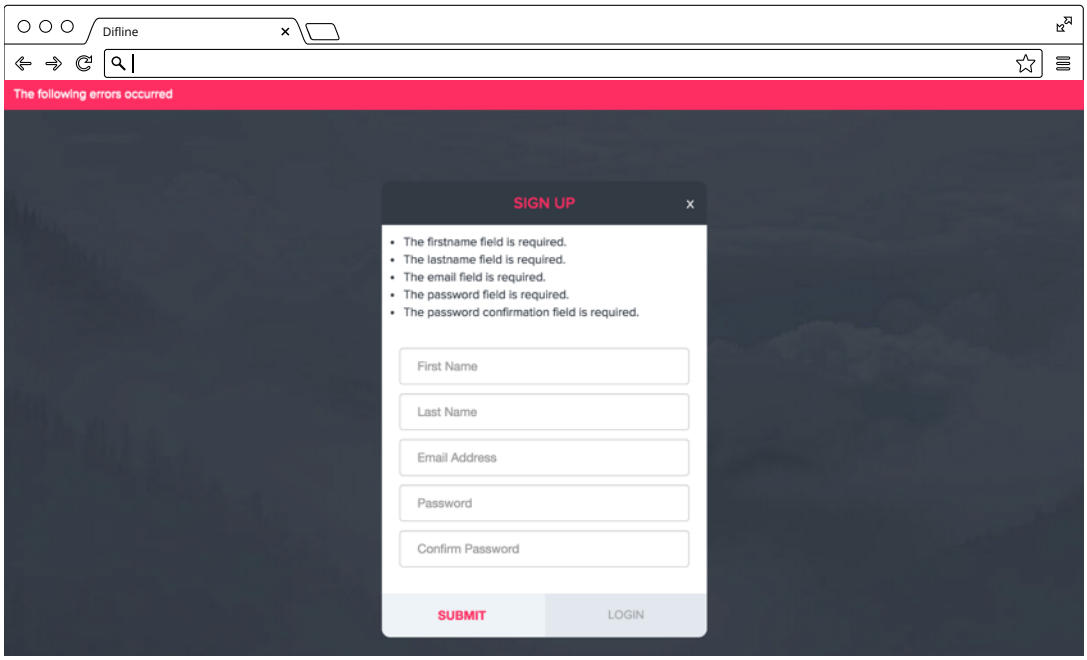

Figure 3.27: Sign up requirements error

Alert messages are also used within the dashboard (See Figure 3.28). The same pink design is used but not along the top of the page. This is used in the dashboard when a user deletes a comparison or updates their account settings. The pink bar alerts the user if it has been done successfully or if their is a problem.

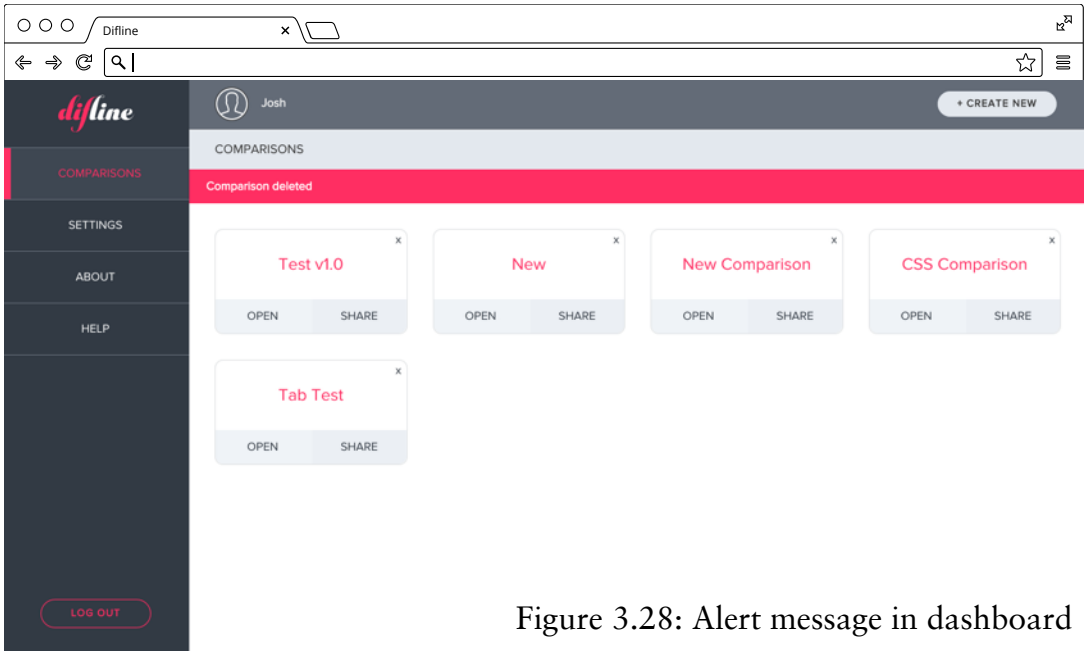

Figure 3.28: Alert message in dashboard

3.1.8 Non-account User

Sign up

Along with the validation and alerts, the fields in the sign up form have changed slightly. The username field has been removed and a surname field added as the profile dashboard will use the users real name, rather than a username. The Twitter and Facebook fields have also been removed to simplify the sign up process, making the experience shorter for the user. The longer the sign up form looks, the less likely the user is going to take the time for fill it out. The social network fields which have been removed from initial sign up can be added later. An "X" has also been added to give the user the options of closing the form and going back to the previous page.

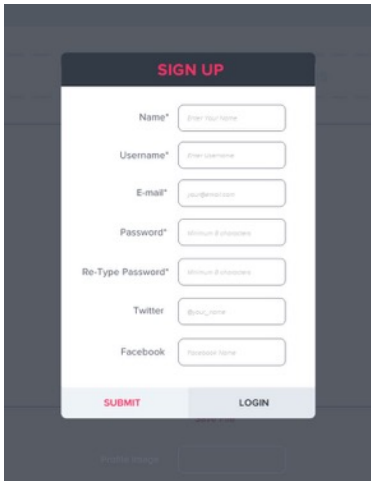

Figure 3.29: Initial Sign Up

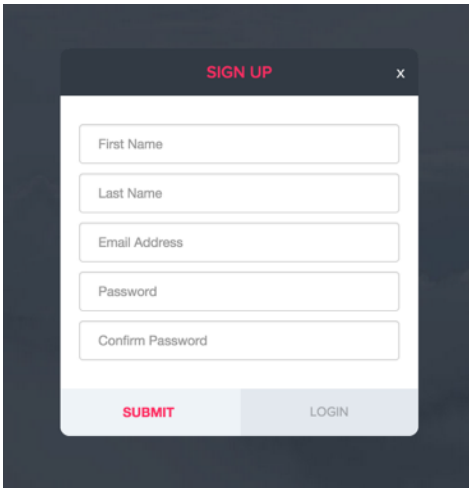

Figure 3.30: Refined Sign Up

Comparison Tool (no account)

The final page is the tool which a user can access for a one-of comparison with no account. The user can compare files in the same way, without the ability to save or share comparisons. This means there is also no comparisons section for the user to go back to old versions. This page is similar to the tool with an account and can be seen in [Appendix 13].

Buttons

Throughout the site there are a range of ghost buttons which when a user hovers over a link, some links change colour and others fill with colour. There are also a couple of links on the homepage which increase the side of the underline when hovered over. Each hover has a 0.3 second transition which makes it a slow and smooth interaction for the user.

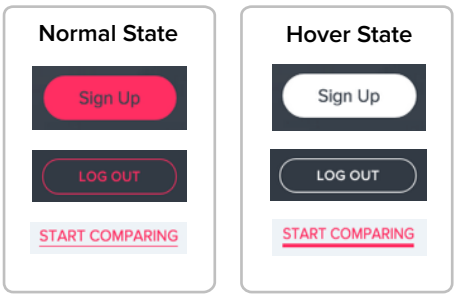

Figure 3.31

3.2 System design

3.2.1 Client-Server Model

Refined changes to the client-server model (See Table 3.1) include the removal of a AngularJS as a client-side framework as it is not needed. After deciding to change to the latest release, Laravel 5. That change has been changed again as the development will stick with Laravel 4.2 as a server-side framework which was used in the initial prototype. Changing to Laravel 5 did not benefit the product in any way and would have used time which would be better spent on other features. LESS and Bootstrap is still used for styling as it is more specific that just CSS3 and Mergely with Codemirror as JS libraries. Changes from the initial model have been highlighted in blue.

CLIENT		SERVER	
Markup		**Server Side Scripting**	
HTML5		PHP	
Styling		**Framework**	
LESS/CSS3 & Bootstrap		Laravel 4.2	
Scripting		**Database**	
jQuery		MySQL	
JS Libraries			
Mergely & Codemirror			
All Displays/Devices			
All Browsers			

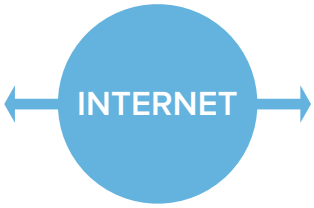

Table 3.1: Client-server model

3.3 Data design

3.3.1 Database Design

The actual structure of the entity relationship model did not change following the prototype and refined design. However some small details changed including the replacement of the username field with first name and surname fields. The account dashboard will display the users real name rather than a username, the user_id field will differentiate accounts in the database if several people have the same name, meaning a username is not needed.

Since the refined design, more changes have been made to the database (See Table 3.2). This is simply the removal of several fields in the User table. The original database design can be viewed in [Appendix 14]. The social network details have been removed as the share feature will now simply produce a URL to copy and paste. These API's can be added to the product in the future. The profile picture was also removed for now as it isn't really needed. The product is a tool for developers to be more efficient and productive. Uploading pictures defies this purpose. This may be added in the future along with capabilities to follow others but for now, there is no need for an image as other users cannot view your profile anyway. The phone field was also removed as it was not needed. It's not a requirement when signing up and it's not the kind of information a user will add if it's not required.

Entity Relationship Model

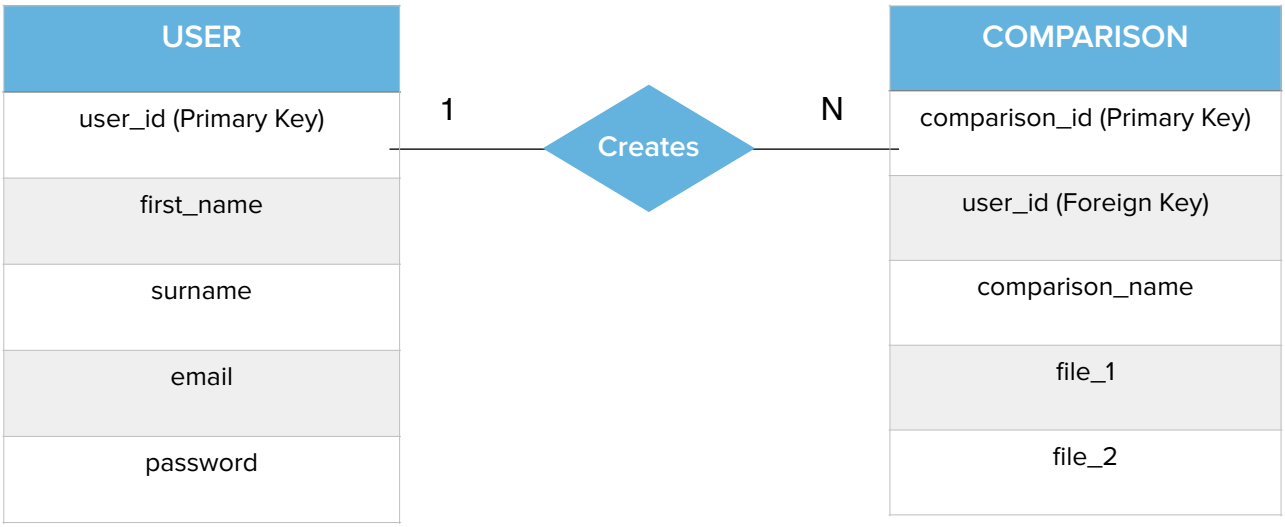

Table 3.2: Entity relationship model

3.3.2 Filesystem

For the save feature available to an account user, it was planned that a filesystem would be used. This would save space in the database, speeding up loading times when opening the saved comparison.

When saving a comparison to a users account, the back end would write the contents of the tool to two new files in the filesystem. The name/url for this file would then be saved to the database for both the left and right side of the tool. Rather than saving all of the text/code straight to the database, only a url would be saved.

Following some tests, it proved to cause some problems when trying to write to the filesystem while using Mergely for the front end of the comparison tool. In order for the save feature to be developed within the time limit for this project, the comparisons were built to save straight to the database. When a user saves a comparison, the entered name and contents of each side of the tool are sent to the back end with an AJAX post, the contents are then saved straight to the database. When the comparison is opened, the contents from the database are pulled and through AJAX, they are loaded into each side again. This may cause speed problems if the product started to grow and generate a lot of users. In the future saving to the filesystem will be implemented in order to prevent this problem and allow for the product to scale, but for now, the contents save straight to the database.

4. IMPLEMENTATION

4.1 Technology selection

4.1.1 Server-side

Server-side Scripting

As a front-end developer and designer, PHP was chosen as the server-side scripting language over other languages like Python or Ruby as it is the only back-end language which wouldn't have to be learnt from scratch. The is a desire, to learn more backend language's e.g. Ruby and Python but with this product, a PHP based MVC framework is used. Using PHP as the server side language provides all the capabilities for everything required on the server side in this project.

Server-side Framework

For this project an MVC framework is being used. This is something which had to be learnt form scratch. Originally it was thought that it would take a lot of time to learn how to use but would help take my project to another level by doing much of the heavy lifting in the back-end. This was true, however for basic functionality, it was picked up much quicker than expected. As a new framework, it is what most of the time researching was spent on. Other MVC frameworks were also researched. With a little bit of experience with ExpressionEngine, the CodeIgniter framework which was built by EllisLab, the same company that built ExpressionEngine, was compared. According to many articles and forums, the impression was given that CodeIgniter didn't have a lot of the features that frameworks like Lavarel or Ruby on Rails have. For example, Lavarel has built-in template languages including Blade, Smarty and Twig. It was also found that many developers who used CodeIgniter in the past have jumped ship to Laravel, feeling that CodeIgniter is now outdated. With this in mind Laravel was looked into further, showing very well-written documentation and some great tutorials. After watching several tutorials it was decided that Laravel would be a great framework to learn and would be perfect for this project.

4.1.2 Client-side

Markup

As a markup language, HTML5 is the language of choice. As the web standard, HTML5 is the best language for an application like this. However, other languages were compared to validate this choice. First of all, HTML5 is the latest version of HTML, which is a HyperText Markup Language. There have been some useful new elements added to HTML5 including semantic elements like <header>, <footer> and <section>, form controls have also been added e.g. date and time, graphic elements like <svg> and <canvas> and finally multimedia elements including <audio> and <video>. Other languages included XML and XHTML. XML is an Extensible Markup Language which is often used for storing data, allowing communication between applications. XHTML is like a combination of XML and HTML. As an XML-based HTML, it works like HTML but uses the same rules as XML for structuring documents. As most HTML5 features are now supported in all browsers, this is the best markup language for this project.

Styling

For the styling of this project LESS is used as a preprocessor to compile the CSS3. LESS is like CSS3 but allows for variables and nesting. The LESS files are be compiled using Prepros. LESS was chosen over Sass, simply because it is the preprocessor that Eyekiller used when a few months were spent there, becoming familiar with it. Sass does some things better than LESS but it is mainly just a different syntax. The advantages for Sass aren't big enough to learn a new syntax from scratch and time would be better spent learning Laravel. LESS does everything that is needed for this project.

Bootstrap will also be used as a CSS Framework. After comparing Bootstrap with Gumby, it was decided that Bootstrap would be used as it uses LESS or Sass and competitors like Gumby only use Sass. Bootstrap also comes built into Laravel which will be used.

Client-side Scripting

The client-side scripting language which was used is JavaScript. JavaScript was used over languages like Python because Python would need to be learnt from scratch, whereas JavaScript didn't. JavaScript was also chosen because of the Mergely JavaScript library which was used to create the comparison tool. Some related languages include jQuery and Ajax which will also also be used with several JavaScript libraries for the comparison tool and save feature.

Since its initial release in 2006, jQuery has become the most popular free, open source JavaScript library. It was designed for client-side scripting to simplify selecting DOM elements and handling events. With jQuery, Ajax is used for the comparison tool and the save feature so send the contents of the files to the back-end. Ajax is great for exchanging data between browser and server without needing to fully reload the page.

Libraries

In terms of the libraries that are being used in this project. jsdifflib.js was the original library of choice. It is an open source library on Github, created by Chas Emerick in 2007[*Chas Emerick, 2007*]. Created to help developers build in-browser visual diff tools, this library looks like exactly what is needed. It provides a visual diff view generator. This offers the formatting of file data, both side by side and inline. There are also other libraries including

Google's diff, match and patch libraries which uses Myer's diff algorithm which could be used. In the end Mergely was used as became clear that creating a comparison tool from scratch was not possible within the time frame.

Mergely uses CodeMirror which is another open-source project to create an in-browser text editor with JavaScript. [*CodeMirror, 2009*] CodeMirror supports over 60 languages and has some useful features including code folding and bracket and tag matching. It is the editor used for the dev tools in Chrome and Firefox works perfect with Mergely for this project.

Client-side Framework

On this topic, there are countless articles and blog posts comparing the three most popular client-side MVC frameworks, AngularJS, Ember.js and Backbone.js. Similar to the back end frameworks, a lot of time was spent researching these frameworks. They all provide excellent documentation and have great community support.

AngularJS was the first framework that was research. It was designed and built by Google in 2009. As the oldest of the three frameworks, AngularJS has built up a large community of users and is the most popular of the three with Ember.js right behind it, rapidly gaining popularity. It is used by some large companies including Google, YouTube and Nike. It is very easy to start with and a lot can be done by just knowing the basics. From there, the learning curve gets quite steep. However, it is very easy to test and debug.

With Backbone, if the developer is inexperienced, they can often be found writing repetitive code. Backbone was created in 2010 at just 6.4K it is extremely lightweight compared to angular at 36K and ember at 69K. As a much lighter framework, it requires plugins to become as feature complete as other MVC frameworks but it is very flexible. With a smaller community than Angular and Ember, it is a very active community with many Backbone developers on Github and some free tutorials. Backbone has been used by Twitter, Foursquare and several music apps including Soundcloud.

Ember is the newest and largest framework of the three. Created in 2011 and at 69K, Ember is used by Groupon, Zendesk and Yahoo. Ember has the steepest learning curve but once it is learnt it will reduce the amount of code and time taken to create an application. It is very fast for developing and prototyping with all the additions but hard to integrate third party

libraries as it is opinionated with how objects are named and files are organised. It also lacks testing tools.

In terms of popularity, as of August 2014, Ember.js has has the most commits and pull requests on Github, however AngularJS has the most contributors on Github and is the clear winner with the number to tagged questions on Stack Overflow and the number of Tweets per day. (See Figure 4.1)

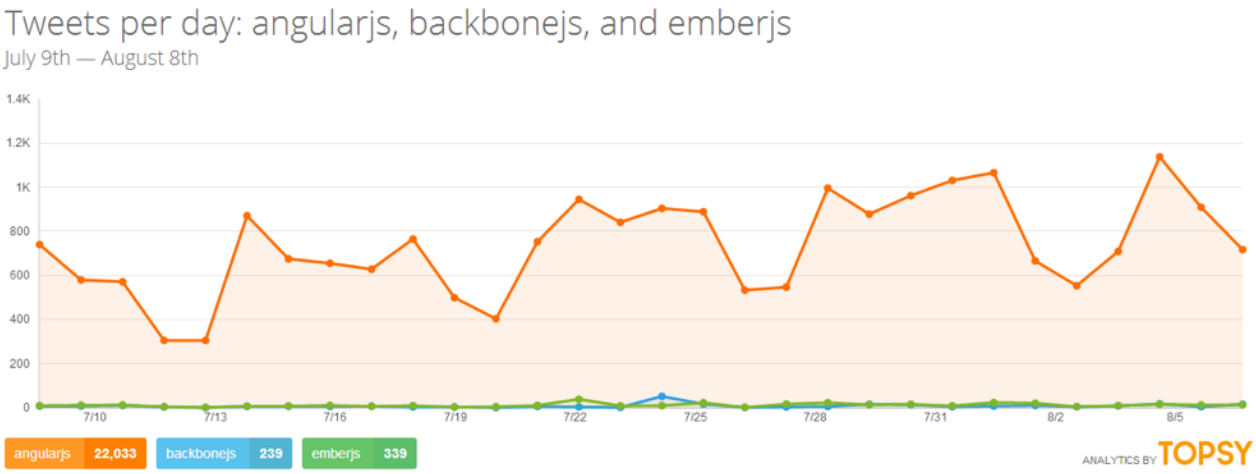

Figure 4.1: MVC framework comparison [*Shilpi, 2014*]

Following the research AngularJS was the choice of client-side framework, firstly because because of the popularity and number of resources for learning from a large community of users. It also has dependency injection and has incorporated some Bootstrap components. However, following the functional prototype with a Laravel as a server-side framework, it became clear that combining a client-side framework with Laravel was overkill and was not needed for a project of this scale. In the end, a lot was learnt about Angular, Ember and Backbone but a client-side framework was not used.

Database

Having only used MySQL for a database, some other options where researched to see if there were any relational database management systems which suited my project better than MySQL. Along with MySQL, PostgreSQL and SQLite were researched. With SQLite, it is more suited towards single-user applications or games and was good for testing. With PostgreSQL, it was known for having many more features that MySQL but it was also much slower. MySQL has since added more advanced functionality and PostgreSQL has improved its speed somewhat. However, PostgreSQL seems to be a good option for complex

procedures and designs, its also extensible and has strong third party support with open-source tools. The problems with PostgreSQL are the performance, speed and simple set ups. It is much better for more complex designs and is also harder to come by hosts or service providers that offer managed PostgreSQL instances. This leaves MySQL which is the most popular and the best choice for this project for the following reasons; it is perfect for websites and web applications, it is secure, fast, scalable, full of features and unlike PostgreSQL, it is easy to install and work with.

4.2 Technology Use

4.2.1 Client-side

Before the prototype and risk analysis, it was decided that AngularJS would be used as a client-side framework. Following the prototype, it was clear this AngularJS was not needed. With jsdiflib and Codemirror libraries available and Mergely as a backup for the comparison tool, Laravel did everything else that was needed. For a larger application, AngularJS may be useful and can pair well with Laravel, but for this project, there is no benefit of using AngularJS. There will be no replacement client-side framework for AngularJS because with Laravel as a server-side framework, it provides everything as an MVC needed for this app.

4.2.2 Server-side

The server-side framework was a good choice as Laravel proved to be much easier to learn than expected and provided all the functionality needed, reducing much of the risk on the back-end. Since the prototype was built with Laravel 4.2, a new version was released, Laravel 5. Some changes had been made to the latest version and it was initial thought that this project would change over to Laravel 5. However it did not benefit the project in any way. Although it wouldn't have been too complicated to change, it was time better spent on other features of the project.

4.3 Risks

The original risks for this project, before the prototype are displayed below and explained in order, starting with the highest level of risk. The risks were then re-evaluated following the prototype stage, and explained with the use of a risk matrix.

Risk - Combining Angular and Laravel

Evaluation - Picking up two new frameworks and learning them is a large enough risk before trying to combine them. These frameworks may pair well to create a web application but the risk is too large to combine two frameworks with no experience in either.

Plan - After some testing on the prototype and some more research, it was decided that angular was not needed and Laravel was more important for the heavy lifting on the back end.

Risk - Using Laravel as a server-side framework with no prior experience.

Evaluation - This is the largest risk as there is a deadline for the project to be complete. This means not only building the product for that deadline, but also learning a new framework within the same timeframe. With Laravel having a steep learning curve, it is a big risk, but it can also reduce the number of other risks which is why it is the main experiment with the prototype.

Plan - Following the prototyping, this risk was severely reduced as the learning curve for Laravel was not as steep as expected. This was only with a very basic understanding but with some basic knowledge of the framework, a lot was achieved in the prototype.

Risk - Back-end skills (Creating user accounts)

Evaluation - This includes everything to do with the users having an account, from the sign up and login, to the database and user privileges. Having less knowledge in back end development and PHP, with experience building only very basic
systems, it is a risk to build a secure product which can be used by both, users with and without an account but provide extra features for a user with an account.

Plan - After testing Laravel with the prototype, this is no longer a risk as Laravel makes all of this very easy, especially with some of the screencasts and tutorials available. Setting up secure accounts is not as much of a risk as expected.

Risk - Building the Comparison Tool

Evaluation - Creating a comparison tool using CodeMirror and jsdiflib. This is a risk as it is very complicated, identifying lines of code which differ between two files and then highlighting them.

Plan - With the risk on using laravel and the back-end risks reduced, this is what a lot of time will be spent working on, but as a back up, Mergely, which is used in the prototype can be used which reduces the risk if the tool is not built within the time frame.

After the prototype and risk analysis, the risks were severely reduced. Back-end risks were been removed by using Laravel. The steep learning curve of Laravel was reduced to around 2-3 on the risk matrix below (See Table 4.1), as basic functionality has been created in the prototype, so the risk of adding advanced functionality has a lower consequence. Having Mergely reduced the risk of building the comparison tool to around 4-6. This is due to the possibility of the comparison tool not being built, but because of the backup, the consequence is low. Colouring keywords in the code is around 6-8 on the matrix as there isn't much knowledge for this feature so the likelihood on the matrix is higher, but it doesn't effect the functionality of the tool making the consequence is low. This makes all risks low to moderate.

1–3 Low risk 4–6 Moderate risk 8–12 High risk 15–25 Extreme risk

Likelihood

Consequence	1 - Rare	2 - Unlikely	3 - Possible	4 - Likely	5 - Certain
5 - Catastrophic	5	10	15	20	25
4 - Major	4	8	12	16	20
3 - Moderate	3	6	9	12	15
2 - Minor	2	4	6	8	10
1 - Negligible	1	2	3	4	5

Table 4.1: Risk matrix

4.4 Challenges

Download file feature

The download file feature was created to allow the user to save a file after editing once it had been compared. This would prevent the user from having to copy and paste the content back into a text editor, they would just have to save the file and open it. The code for this feature was available through Mergely (See Figure 4.2).

```
function download_content(a, side) {
    //a.innerHTML = "preparing content..";
    var txt = $('#compare').mergely('get', side);
    var datauri = "data:plain/text;charset=UTF-8," + encodeURIComponent(txt);
    a.setAttribute('download', side+".txt");
    a.setAttribute('href', datauri);
    //a.innerHTML = "content ready.";
}
document.getElementById('save-lhs').addEventListener('mouseover', function() { download_content(this, "lhs"); }, false);
document.getElementById('save-rhs').addEventListener('mouseover', function() { download_content(this, "rhs"); }, false);
document.getElementById('ignorews').addEventListener('change', function() {
    $('#compare').mergely('options', { ignorews: this.checked });
}, false);
```

Figure 4.2: Download file code

There was a download button for each side. When the download button was clicked for a either left hand side or right hand side, the content from that specific side is then downloaded into a .txt file.

The problem with this feature was that it conflicted with the save feature, when it was developed. This feature also gets the content from either side, but sends it to the back end to be saved to a database, rather than download it in a .txt file.

The solution to include both features within the time frame wasn't found. It was decided that the save feature, allowing an account user to save a comparison to their account proved more important. This resulted in the removal of the download file feature for now. However, this is a feature will be developed in the future.

Highlighting syntax with Mergely

With the integration of Mergely, a technical challenge was needed as the comparison tool was not being made from scratch. This is achieved with features including the save and share feature. However, a syntax highlighting feature was also created by combing Mergely with another syntax highlighting JavaScript library. Several libraries where tested including highlight.js and prism.js. Highlight.js proved to be the best for this project but while it highlighted the syntax, it also effected Mergely's functionality.

As this problem was discovered late in the project, there wasn't much time left to fix the problem. While it initially caused some problems loading the content into each side of the comparison tool, the content eventually loaded into each side and did highlight the syntax (See Figure 53). However, it also effected the highlighting of differences in each side. With not much time to fix, the feature was removed for the final deadline but is likely the CSS for

Mergely was clashing with the Highlight CSS. This feature is less of a problem and was removed as it was a case of design over function. This will also be developed in the future.

Another solution to this problem would to replace Mergely and use only CodeMirror. Mergely does use CodeMirror mainly as a text area and split screen but CodeMirror also have syntax highlighting and many other features, including a merge addon, with does the same thing as Mergely. It also has a rich programming API and a CSS theming system in order to customise it and add new functionality. This may actually be a good solution which will allow Difline to grow and add more functionality in the future.

Even though CodeMirror is used with Mergely, it didn't completely replace Mergely at this stage as it would have required all other features around it to be redeveloped. The drag and drop feature and also the save feature, allowing the user to save comparisons to their account would both have to be developed from scratch. This wasn't achievable in the time frame for this project but can be implemented in the future if highlight.js doesn't combine with Mergely to solve the syntax highlighting problem.

Opening the saved data into tool

One of the biggest problems that arose towards the end of the project was opening a saved comparison back into the comparison tool.

When a comparison is saved, the content of each side and the entered comparison name is sent to the back end with an ajax call for each side. This data is then saved to as a database entry to a comparisons table. It also saves the users id, in order to assign that saved comparison to a specific user. These saved comparisons then displayed on that users comparison page.

When a comparison with a single line is saved, it works perfectly, saving to the database and opening back into the comparison tool when opened from the comparisons page. The saved content loads back into its correct side through an AJAX call for each side.

The problem is caused when when opening a comparison with multiple lines. When a comparison has multiple lines in each side of the tool, it saves to the database correctly, but when opened from the comparisons page, it doesn't load properly into its correct side of the

tool. It does load everything on one line when simply placed into the HTML page. Using nsbp(), it was then be displayed on the correct lines by recognising line break characters. However, this solution did not work inside the comparison tool.

An attempted solution was to write the contents to a file and save to the filesystem rather than directly to a database. This works using fwrite() but it is unable to load back into the comparison tool. This problem was discovered very late in the process, with not much time for a solution. Writing the contents to a file and loading it back into the tool was attempted and will be worked on to implement this feature in the future as this is the likely solution to the problem.

4.5 Achievements

Login and Sign up

The login and sign up was made very simple with Laravel. In fact with Lavavel 5, it is pre build into the framework. As explained in the client-server model, Laravel 5 wasn't used. However, this didn't effect the login and sign up forms as they were created before Laravel 5 was released.

Both forms send the input data from the view to the UsersController. In the controller there are several functions, dealing with the login, signup and logout. A validator in the User model sets requirements for each field in the form. When signing up, the function checks if it passes the validator, if it does, the user is added to the database with a unique ID and is redirected to the login page to log in. If it does not pass the validator, the user is redirected to the sign up page with an error message and a list of the errors that didn't pass the validator. When the user logs in, their username and password is checked in the database and they are directed to their account dashboard if the username and password is valid. If the username and password isn't valid, an error message is displayed, informing the user that their username and password is incorrect.

Updating user details

To update the users details in the setting page, a similar function used in the sign up is used. The only difference is that a new user isn't created, the signed in user is checked and the entered data in the form is updated to the database. (Appendix 15)

Comparing files with Mergely

The code comparison tool itself uses Mergely, which uses a combination of jQuery and CodeMirror. Mergely works by initially loading two text files through AJAX GET requests. The data from each file is then assigned to a side of the comparison tool and the contents are displayed in the text areas. The code behind the highlighting differences in the files is very complicated JavaScript which was unrealistic to create from scratch for this project. Mergely was edited in order to fit this project by stripping it down, changing the CSS, editing and removing features e.g. the download file button. Features where added to the tool allowing the user to save a comparison and share it, re-open a saved comparison and edit it. The syntax highlighting will be added in the future to improve it.

Drag and Drop feature

The drag and drop feature is implemented through CodeMirror. There are several drag and drop JavaScript libraries and APIs but it was a feature included with Mergely. Only supposed in browsers that support HTML5 FileReader, a check is done before reading the files. The files are then read and split into the two side of the comparison tool with Mergely.

Saving content from the tool to a database

The save feature save allows a user with an account to save a comparison to their account. This can then be re-opened and edited again or shared with a URL.

Saving the a comparison to a users account begins with a click function, triggered by the save button. This take the entered comparison name and the data from each side of the comparison tool and is sent to the PagesController with an AJAX POST. (See Figure 4.3)

```
$("#save").click(function(){
    var values = {
        'lhs': $('#compare').mergely('get', 'lhs'),
        'rhs': $('#compare').mergely('get', 'rhs'),
        'comparison-name': $('#comparison-name').val()
    };

    $.ajax({
        type: 'POST',
        url: 'save',
        data: values,
        success: function (result) {
            console.log(result);
            alert('Comparison saved');
        }
    });
});
```

Figure 4.3: Save click function and AJAX POST

A route for the save feature is created and an eloquent model (Appendix 16). A save function in the controller checks if there was an AJAX request. It then gets the current users ID, creates a new comparison entry in the comparisons table in the database, adds the data sent from the AJAX POST along with the users ID and saves it to the database. (See Figure 4.4)

```
public function save() {
    if (Request::ajax()){
        $id = Auth::id();
        $comparison = new Compare;
        $comparison->lhs = Input::get('lhs');
        $comparison->rhs = Input::get('rhs');
        $comparison->user_id = $id;
        $comparison->comparison_name = Input::get('comparison-name');
        $comparison->save();
        return Input::all();
    }
}
```

Figure 4.4: Save function

Displaying the saved comparison for a specific user

When a user saves a comparison, a block is displayed on the users comparison page with the comparisons name, a delete, a share and an open button. Using blade templating, a foreach loop displays each comparison which has been saved by a specific user (See Figure 4.5). In the User and Compare models, functions define the one to many relationship between with belongsTo() and hasMany() functions. The view is made in the controller and the users ID is found for the foreach loop. The comparison name is pulled through along with the comparison ID for the URLs.

```
<!-- Small boxes (Stat box) -->
<div class="row">
    @foreach ($user->compare as $compare)
    <div id="comparison-block" class="col-lg-3 col-xs-6">
        <!-- small box -->
        <div class="small-box bg-white">
            <a href="/delete/{{ $compare->id }}"><button id="delete">x</button></a>
            <div class="inner">
                <h3 class="name">
                   {{ $compare->comparison_name }}
                </h3>
            </div>
            <div class="icon">
                <i class="ion ion-bag"></i>
            </div>
            <div class="small-box-footer">
                <a href="/compare/{{ $compare->id }}"> <button  class="footer-left">
                    OPEN
                </button></a>
                <a href="/share/{{ $compare->id }}"><button class="footer-right">
                    SHARE
                </button></a>
            </div>
        </div>
    </div><!-- ./col -->
    @endforeach
</div><!-- /.row -->
```

Figure 4.5: Blade templating foreach loop

Deleting a saved comparison

When deleting a comparison, the 'X' in the comparison block is clicked (See Figure 3.22). This links to the delete route for that specific comparison, a conformation page is then displayed, asking the user to confirm if they wish to delete, preventing comparisons being deleted accidentally. On conformation, the ID of that comparison is retrieved and that entry in the database is deleted. The user is then returned to their comparisons page with an alert message, informing the user that the comparison has been deleted successfully (See Figure 4.6).

```
public function delete($id) {
    $compare = Compare::whereId($id)->first();
    return View::make('delete', ['compare' => $compare]);
}

public function confirmDelete($id){
    $compare = Compare::whereId($id)->first();
    $compare->delete();
    return Redirect::to('users/dashboard')->with('message', 'Comparison deleted');;
}
```

Figure 4.6: Delete functions

Generating a URL for the share feature

As previously explained, the share feature simply generates a URL to copy and paste, rather than implement social networks at this stage. Generating a URL also makes it easy to share on social networks. A comparison can be shared from the comparisons page and when the comparison is open.

A route and view is created, displaying the URL in a simple page, similar to the delete conformation and login/sign up forms. In the controller, the URL and comparisons ID are set in variables. (See Figure 4.7)

```
public function share($id){
    $url = URL::to('/compare');
    $compare = Compare::whereId($id)->first();
    return View::make('share', ['url' => $url], ['compare' => $compare]);
}
```

Figure 4.7: Share function

The URL is then generated using blade templating syntax {{ $url }}/{{ $compare->id }} to display the URL with the specific comparison ID for the comparison being shared.

Forget password form

For the forget password feature, Laravel provides methods for sending password reminders and password resets. The uses the RemindableInterface in the model. The forms simply need

built and to be linked together and connected to a database. The forget password feature begins with a simple form, asking for the users email address. This form then emails a link to that email address which expires after 60 minutes (See Figure 4.8). The link then provides another form which will reset the users password.

```
<h2>Password Reset</h2>
<div>
    To reset your password, complete this form: {{ URL::to('password/reset', array($token)) }}.<br/>
    This link will expire in {{ Config::get('auth.reminder.expire', 60) }} minutes.
</div>
```

Figure 4.8: Password reset

Enquiry email feature

This email feature made was very simple in Laravel with the Mail method. For this project the method was set to Mail::pretend (See Figure 4.9). In this mode, messages will be written to the application's log files instead of being sent to the recipient. When the product is ready to launch, an API Driver can be set up and the can be method changed to Mail::send.

```
public function submitFeedback(){

    $msg = Input::get('msg');

    Mail::pretend('feedback', $msg, function($message){
        $message -> to('josh@joshwhann.com', 'Difline')
                 -> subject('Difline Enquiry');
    });

    return Redirect::to('help')->with('message', 'Message was sent successfully');
}
```

Figure 4.9: Enquiry email

Blade templating

Blade templating is a very powerful templating engine which uses a .blade.php extension. It is very useful for a project like this which has different features, depending on whether the user has an account or not. With simple if statements (See Figure 4.10) sections can be changed depending on the user being signed in or not.

```
<div class="side-button">
    @if(!Auth::check())
        {{ HTML::link('users/login', 'Login', array('class'=>'button-login')) }}
    @else
        {{ HTML::link('users/logout', 'Log Out', array('class'=>'button-login')) }}
    @endif
</div>
```

Figure 4.10: Blade templating @if statement

It is also great for echoing data by using {{ }}. This is used in this project to echo variables like a username. Blade templating is used for the layouts in this project. For example the dashboard layout has @yield directives to pull @sections into the layout in which that file @extends.

5. TESTING

5.1 Testing approach selection

In section [2.4.1 Questionnaire] a user survey was created in order to gain information around the interest for the product. With the product developed, it then had to be tested. The testing method used for this project was a guerrilla method [GOV.UK, 2015]. This method was chosen as it is an effective, quick and easy to set up method for testing. It allows for tasks to be defined which gains valuable information on how effective a design is on the target audience. By viewing users interact with the product first hand, it is clear to see if certain features and functionality work in the way they are intended to work. It doesn't require many people and helps to identify areas for improvement if the users struggle to complete the set tasks.

5.2 Testing process

Rather that set a date and location for testing, several people were approached individually and asked to complete the testing process. This worked as it was quick a efficient. The product also wasn't on a live server so this allowed for users to test on a local server as they all tested individually.

5.2.1 Test tasks

The tasks given to the user to complete are as follows:

(1) Sign up
(2) Log in
(3) Create a new comparison
(4) Identify the differences in two provided files
(5) Make the files the same
(6) Give the comparison a name
(7) Save a comparison
(8) Open a saved comparison
(9) Edit and re-save a comparison
(10) Delete a comparison
(11) Go to the homepage and back to your account after signing in
(12) Change your password
(13) Update email address
(14) ~~Send and enquiry about the product~~
(15) Share a comparison
(16) Log out
(17) ~~Reset password~~

5.3 Test results

The test was taken by 8 people, all of which would be considered a target audience for this product. The following table (See Table 5.1) displays each user and the tasks which they were and were not able to complete. The tasks are numbered in the previous section for this table. The table is filled green if the task was completed and red if it was not. The length of time taken to complete each task was roughly judged with short and long, based on the amount of time taken. The grey blocks are for tasks which were removed from the test. The tasks that were removed were features that included email; reset password and enquiry form, which weren't working as the testing was being done locally. The failed tasks where all by users that tried to open a comparison which had more than one line in it. This is the biggest priority for future development of this product.

5.3.1 Results table

	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8
1	Short	Short	Short	Short	Short	Short	Short	Short
2	Short	Short	Short	Short	Short	Short	Short	Short
3	Short	Short	Short	Short	Short	Short	Short	Short
4	Short	Long	Short	Long	Short	Short	Short	Long
5	Short	Short	Short	Short	Short	Short	Short	Short
6	Short	Short	Short	Short	Short	Short	Short	Short
7	Short	Short	Short	Short	Short	Short	Short	Short
8	Short		Short	Short		Short		
9	Short	Short	Short	Short	Short	Short	Short	Short
10	Short	Short	Short	Short	Short	Short	Short	Short
11	Short	Short	Short	Short	Short	Short	Short	Short
12	Short	Short	Short	Short	Short	Short	Short	Short
13	Short	Short	Short	Short	Short	Short	Short	Short
14								
15	Short	Short	Short	Short	Short	Short	Short	Short
16	Short	Short	Short	Short	Short	Short	Short	Short
17								

Table 5.1: Test results

5.4 Browser/Device Testing

Difline has been tested throughout the whole development stage and works in all browsers including Google Chrome, Firefox and Safari. It has also been tested in several devices to test how responsive it is. On a mobile design, links and images change as the user cannot use the tool on their mobile (See Figure 5.2). The only change made to the original designs was the size and position of the test and button above the fold on the homepage on a device wider that 1600px (See Figure 5.1). For other devices which were not available for physically testing e.g. tablet, BrowserStack was used. All of the previous designs are for a Laptop device, below are screenshots on mobile (iPhone) and a large desktop (iMac).

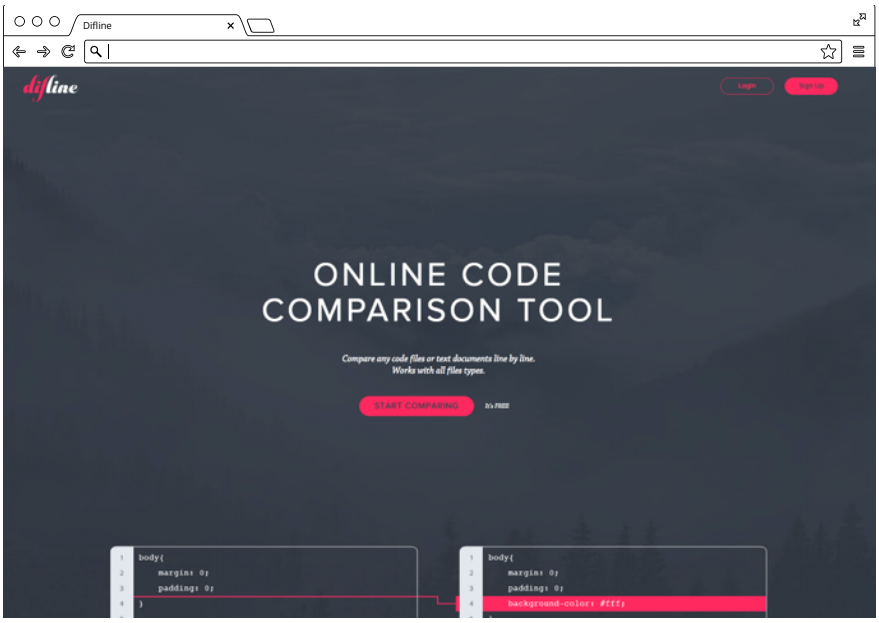

Figure 5.1: Desktop testing

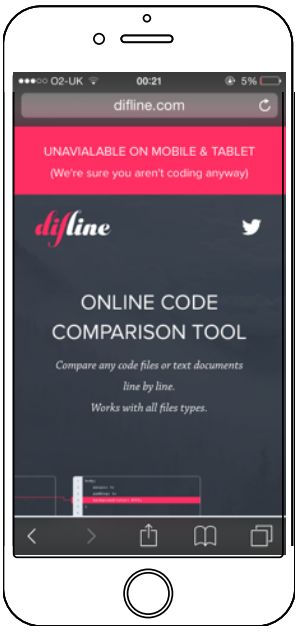

Figure 5.2: Mobile testing

6. EVALUATION

6.1 Test Result Evaluation

The guerrilla method used for testing was used as it was a quick and easy method to gain valuable information. This testing method was good as it allowed for the tester to be watched as they interacted with the product. The results were positive and showed the the product is easy to use. The results showed that all tasks were completed apart from 8, which wasn't completed by users that tried to open a saved comparison with multiple lines.

The only negative about this testing process is that it didn't test very many people. The more users that test the product, the better and more accurate the results are. With more testing in the future, there will be more time taken to get more testers.

6.2 Project outcomes

The result of this project is a complete product which fits a gap in the industry. There are very few areas or features that were not achieved. The methodology worked well for this project, taking a modified waterfall model. The design and branding has been well received on social networks, driving users to subscribe to be notified about the launch. During testing the users experience using the product was positive. All core functionality works well, allowing users to compare two files by highlighting the differences line by line. A user with an account can save a comparison, open, edit, share and delete it. They can change the details they signed up with and can email an enquiry if they have any problems with the product. The only big problem is opening saved comparisons with multiple lines.

On the front-end, the comparison tool was not built from scratch as this revealed itself to be an unrealistic task, given the time frame. Mergely was used for the tool and a lot was learnt by combining it with Laravel to save comparisons to an account. There were also many efforts to add syntax highlighting to the code within the tool but combining Mergely with other JavaScript libraries caused some small problems in the functionality of the comparison tool. It was decided to remove the syntax highlighting as it was a case of design over function. This was also the case with the download file feature which did not work after implementing the save feature. The save feature proved more important to the product and so the download file feature was removed until they are developed in a way in which they can work together.

The product could be improved by implementing the features that didn't get developed within the time frame. The branding was also well received but some feedback revealed that the logo did not represent code, or imply that the product was for code. Even though the logo and name represents highlighting differences line by line, it may be an improvement my representing code in some way through the brand.

6.3 Methodology Evaluation

The modified waterfall model has proved to be an effective methodology for this project. Unlike an Agile methodology which would get better results in a small team, the modified waterfall methodology has provided a solid structure and plan to follow when developing this product. While being slightly more flexible than Royce's original waterfall model, it has allowed for stages to be revisited and changed/added to.

The methodology went smoothly with no problems. The only negative would the ability to detect missing functionality of fix errors much earlier in the process like a prototyping model would allow for. This would help with problems and bugs being dealt with earlier in the process. Other than this aspect, the modified waterfall model has worked well.

6.4 Plan Evaluation

The overall plan and end result for the project has not changed, but there were a few deviations from it. With Mergely always there as a backup, the original plan was to build the comparison tool itself from scratch. The further into the project, the more unrealistic this became with the time frame. After some advice from other developers and my mentor, it became clear the product would benefit by using Mergely with the addition of other features, rather than trying to build a tool with all of the capabilities of Mergely from scratch.

The other deviation from the plan was to not include Twitter and Facebook API's. This was something that was initially planned for the share feature but could also be added to the sign up process in the future. With Eloquent in Laravel, the share feature was created by generating a URL which could simply be copy and pasted to any service. This allowed for the share feature to not be restricted to two services. Twitter and Facebook can be added in the future but the URL allowed the comparison to be shared anywhere.

Even with theses deviations and adaptations, the end product is what was initially planned.

7. CONCLUSION

7.1 Report Summary

This report has followed the structure of the chosen methodology, modified waterfall. Beginning with the concept definition, which includes requirements, some initial paper prototypes and the feasibility of the product. After this, the user experience and system was designed. With final designs, the technologies were explored for the development of the product. As the product was developed, it was then tested. The report has followed the same structure and documented each step, describing the work undertaken at each stage. The report helped the project in many ways by researching and planning, documenting risks and deciding what technologies are best for the development of the product.

7.2 Project Reflection

On reflection of the project. The process went well. Having a methodology to follow helped with planning and scheduling. A lot has been learnt on the back-end with the implementation of Laravel as an MVC framework. This has been the largest learning cure during the development and the biggest aspect taken away from the project. It also proved that having an MVC framework to do a much of the heavy lifting in the back-end, made the process much easier than it would have been without a framework. Without this, the project would have been much more of a challenge, especially without a team of people. I also learnt that using and combining existing technologies and libraries can produce a much better result that if it was to be built from scratch.

7.3 Role Reflection

The role for this project required everything from the planning, to UX and UI design, to front-end development, back-end development and testing. This role covered several jobs and included areas which would not be required buy one individual in the industry. A product like this would take several members of a team, each working of different areas e.g. design, front-end and back-end. As a full stack designer, a lot was learnt in this role, especially in back-end development. Following a methodology helped to define the roles for this project. On reflection of the role, it is very possible to tackle all aspects of a product.

Any weak areas become much stronger and learning Laravel as an MVC framework has helped with this back-end development.

7.4 Future

The future for difline is to launch it. Eventually there will be more features added. The syntax highlighting for the code will be worked on, to make it work along with Mergely. The ability to download a file after editing rather than copy and paste it back into their text editor will also be worked on. This feature worked before the save feature was developed. Combining these features will be implemented in the future. The highest priority for the future is to fix the saved comparisons to open successfully when there are large files with multiple lines. This is a crucial feature which needs to be fixed before a launch.

A main area of focus in the future will be the accounts. Developing the accounts into a network with the capabilities of connecting/following other account users. This will benefit users that work in an agency environment as they can follow their work colleagues. This would improve the share feature and allow users to see other accounts with the permissions to view all other saved comparisons. The future may also implement possible payment for accounts, adopting a freemium pricing strategy.

8. REFERENCES

Chas Emerick, 01/02/2007, jsdifflib [Github], [online]. Available: https://github.com/cemerick/jsdifflib [29/04/2015].

Jamie Peabody, n.d., Mergely [Mergely], [online]. Available: http://www.mergely.com/ [30/04/2015].

Shilpi, 08/08/2014, AngularJS vs Backbone.js vs Ember.js—Choosing a JavaScript Framework [Part 2] [FusionBrew], [online]. Available: http://blog.fusioncharts.com/2014/08/angularjs-vs-backbone-js-vs-ember-js%E2%80%95choosing-a-javascript-framework-part-2/ [20/11/2014].

James Robertson, 1995, Volere Requirements Specification Template [Volere Requirements Resources], [online]. Available: http://www.volere.co.uk/index.htm [01/05/2015].

Pixeden, 09/05/2014, Vector Browser Outline Presentation [Pixeden], [online]. Available: http://www.pixeden.com/psd-web-elements/vector-browser-outline-presentation [01/05/2015].

GOV.UK, n.d., Guerrilla testing [Gov.uk], [online]. Available: https://www.gov.uk/service-manual/user-centred-design/user-research/guerrilla-testing.html [01/05/2015].

CodeMirror, 30/05/2009, This is CodeMirror [CodeMirror], [online]. Available: https://codemirror.net/ [01/05/2015].

Jeffery Way, n.d., The best Laravel screencasts on the web. [Laracasts], [online]. Available: https://laracasts.com/ [01/05/2015].

highlight.js, 13/10/2012, Syntax highlighting for the Web [highlight.js], [online]. Available: https://highlightjs.org/ [01/05/2015].

9. APPENDICES

APPENDIX 1

Initial use case diagram for a general user.

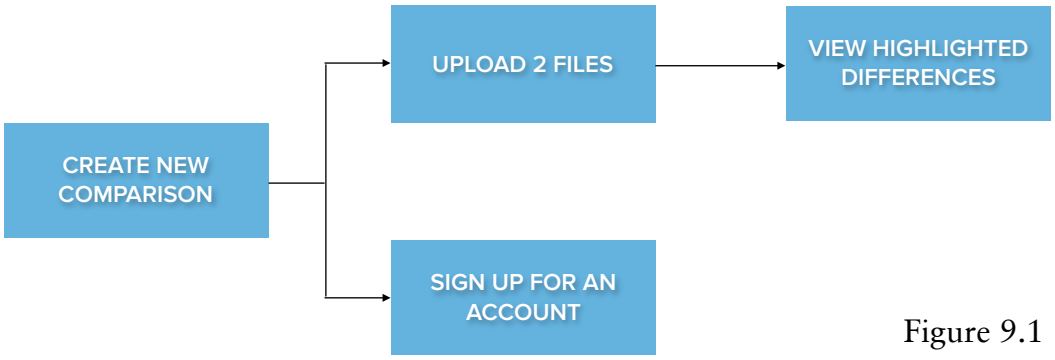

Figure 9.1

Initial use case diagram for an account user.

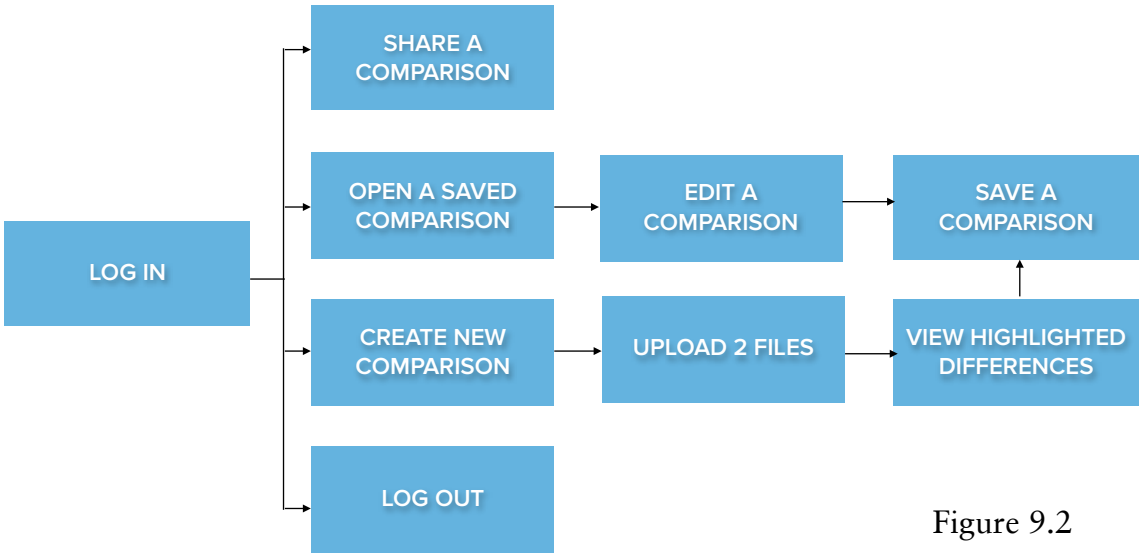

Figure 9.2

APPENDIX 2

Requirement #: 6

Description: The product can share a specific URL to their comparison via email, Facebook or Twitter

Rationale: To allow the user to share the comparison with other developers, possibly the author or the code that is being compared.

Dependancies: Requires the user to have an account, have a saved comparison and have email, Facebook and Twitter details enters in their profile settings.

Fit Criterion: A URL to the comparison will be created and can be accessed from the users Facebook Timeline, Twitter Feed or in the sent email. The URL will open the comparison

Requirement #: 7

Description: The product shall have an 'About' link in the dashboard

Rationale: To explain how the product works to the user

Dependancies: Requires the user to be in a dashboard, either in their account or in a new comparison

Fit Criterion: The the about link is clicked, a description of the product will be displayed in the dashboard.

Requirement #: 8

Description: The product shall allow the user to create a new comparison from a 'Create New' button

Rationale: To allow the user to start a new comparison

Dependancies: The user must click on the 'Create New' button for the page to load

Fit Criterion: When the button is pressed a new comparison will open, a standard comparison dashboard will open if the user does not have an account and an account dashboard will open if the user does have an account.

Requirement #: 9

Description: The product shall have a help form linked from the dashboard

Rationale: Allows the user to contact a person if the 'About' page does not help them

Dependancies: Requires the user be logged on to their dashboard or have an open comparison

Fit Criterion: When the user enters a message in the text area and clicks submit, an email with a message from the user will be sent.

Requirement #: 10

Description: The product will have a file upload button to upload the files into the text area

Rationale: To allow the user to browse the files on their compute from a dialog box if they to not have them ready to drag & drop

Dependancies: Requires the user to click a 'Browse' button and select two saved files

Fit Criterion: When the browse button is pressed, a file dialog opens, two files are selected and uploaded to the text areas.

Requirement #: 11

Description: The product will allow the user to change their settings

Rationale: To allow the user to update their profile information e.g. profile image, change email, change password or link their social network accounts etc.

Dependancies: Requires the user to have an account

Fit Criterion: The new data entered into the form updates the database and stores the new settings.

Requirement #: 12

Description: The product will allow the user to upload a profile image to their account dashboard.

Rationale: To make the product more personal and visual

Dependancies: Requires the user to have an account

Fit Criterion: The new image is added to the database and immediately displayed as the users profile image on their account dashboard.

Requirement #: 13

Description: The product will allow the user to log out at any time

Rationale: To stop another person accessing the users account if it has been left logged in

Dependancies: Requires the user to have an account

Fit Criterion: When the user logs out, without a username and password, they are unable to access their account dashboard and view or edit saved comparisons.

Requirement #: 14

Description: The product will allow the user to add a comparison blocks to their dashboard

Rationale: To keep the account dashboard clean and easy to use

Dependancies: Requires the user to have an account and save a comparison

Fit Criterion: When a user with an account saves a comparison, a comparison block will be added to their dashboard giving the user an option of opening the comparison to edit or to share it

Requirement #: 15

Description: The product will require two in browser text editors

Rationale: The product requires two areas for the two pieces of code that will be compared

Dependancies: Requires the user to create a new comparison

Fit Criterion: Two text areas display in the comparison tool, each displaying the content of two different files that have been uploaded

Requirement #: 16

Description: The product shall highlight single words if the rest of the line in the two files being compared are the same.

Rationale: There may be a spelling mistake on a long line of code which could be quickly highlighted

Dependancies: Requires the rest of the line to be same so that it isn't all highlighted

Fit Criterion: Two lines of code in each file are exactly the same with the exception of one word, the full line stays normal and that single word is highlighted.

Requirement #: 17

Description: The system will allow users to reset their password from the login page

Rationale: To allow the user to login after forgetting their password

Dependancies: Requires the user to be an account holder with info in the database

Fit Criterion: The forget password link is clicked, an email is sent to the user with a unique link which will allow the user to update the database with a new password

Requirement #: 18

Description: The system will colour keywords in different code languages like a text editor

Rationale: To make the code in the text areas more readable

Dependancies: Requires the user to upload code into the text areas.

Fit Criterion: Code is uploaded to the comparison tool, keywords in the code are displayed in different colours

Requirement #: 19

Description: The system will allow users with an account to delete saved comparisons

Rationale: To prevent a build up of unwanted saved comparisons

Dependancies: Requires the user to have an account and have saved a comparison

Fit Criterion: In the comparisons section of an accounts holders dashboard, the remove comparison button is clicked and the stored comparison is removed from the database and users account.

Requirement #: 20

Description: The system will allow users to download files after being compared and edited

Rationale: To allow the user to use the code after editing

Dependancies: Requires the user upload files to the comparison tool

Fit Criterion: The download button below each text area is clicked, the contents of the text area is downloaded in a txt file.

APPENDIX 3

Look and Feel

Appearance

- The product shall appeal to designers and developers
- The product shall appear flat and minimalistic

Style

- The product shall appear modern
- The product shall appear professional
- The product shall appear clear and simple to use

Usability

Ease of Use

- The product shall be easy for developers to use
- The product shall display clearly differences in the code files
- The product shall make users want to use it and sign up for an account
- The product shall help users compare their code quickly and efficiently

Personalisation

- The product shall display the users name and image on their dashboard
- The product shall allow the users to upload a profile image

Learning

- The product shall be easy for inexperienced developers to use
- The product shall help students learn by comparing their code with others

Accessibility

- The product shall be usable by partially sighted users.
- The product shall have alternative text for images

Performance Requirements

Speed

- The differences in code files will be highlighted immediately

Accuracy

- The code files will be highlighted line by line and specific words on a line

Reliability

- The product shall be available for use 24 hours per day, 365 days per year.

Operational and Environmental

- The product shall be used by a developer in a studio/agency environment.
- The product shall be used by students/lecturers in a University/College/School
- The product shall work on all web browsers.
- The product shall work on all web devices e.g. desktop, laptop, tablet, mobile
- The product shall be accessible online

Security

Access

- Only users with an account will be able to save comparisons
- Only users with an account will be able to view previously saved comparisons
- Only users with an account will be able to share comparisons.

Integrity

- The product shall prevent non-account holders from signing in to the dashboard
- The product shall prevent incorrect data being entered into the database

Privacy

- The product shall make its users aware of its information practices before collecting data from them.

Cultural

- The product shall not be offensive to religious or ethnic groups

Legal

- Personal information shall be implemented so as to comply with the Data Protection Act.

APPENDIX 4

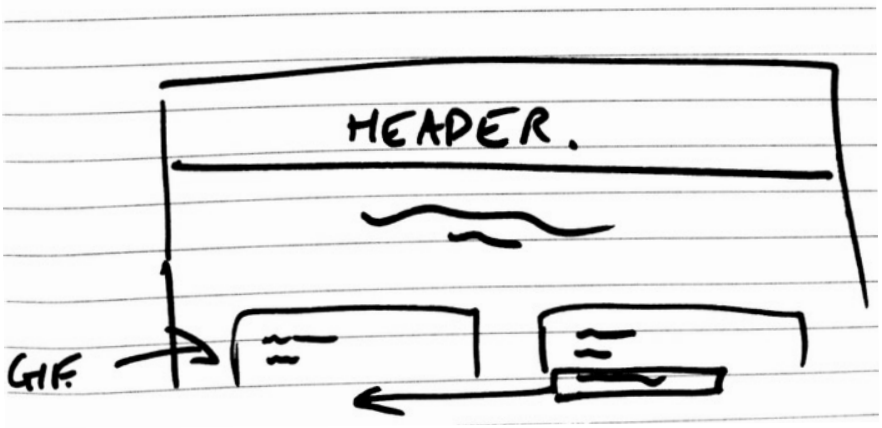

Figure 9.3: Initial homepage gif sketch

APPENDIX 5

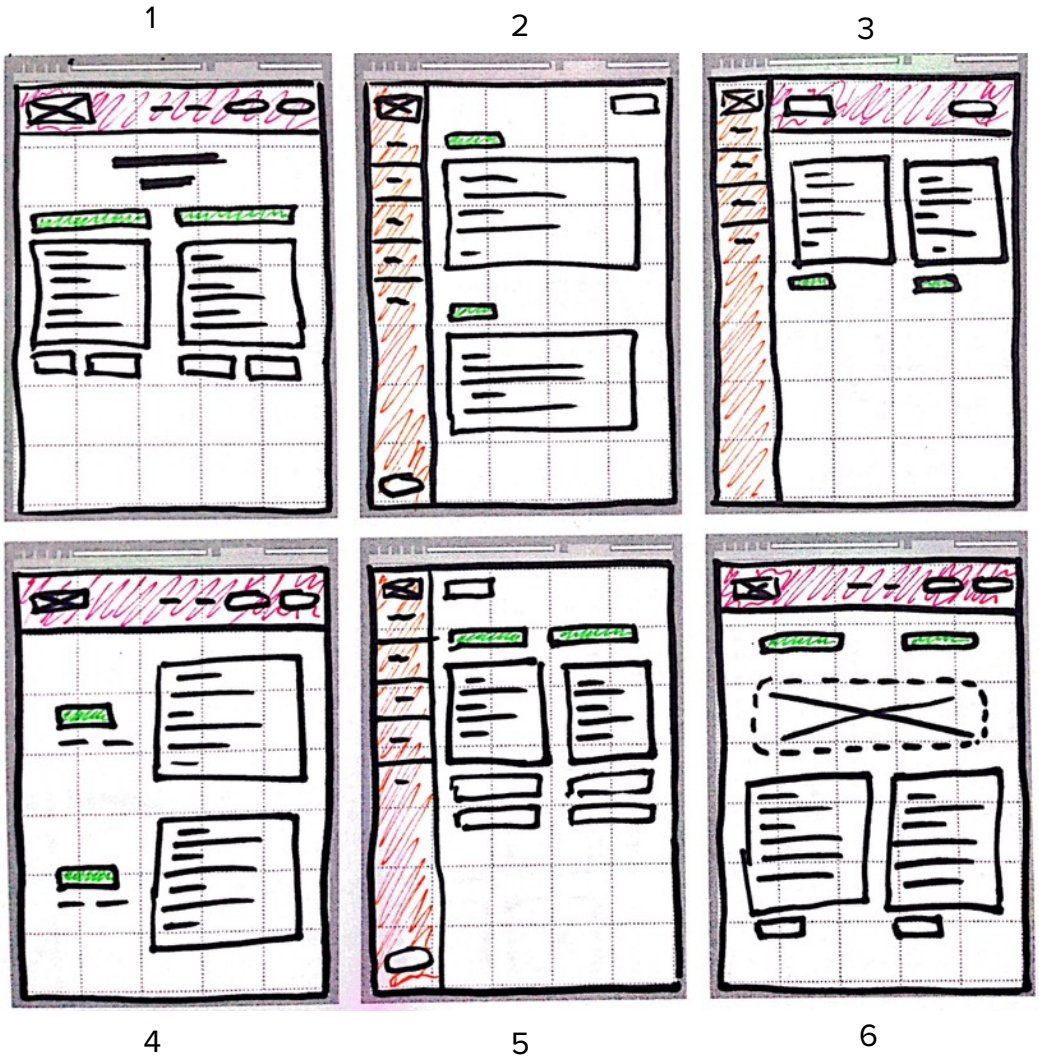

Figure 9.4

APPENDIX 6

Questionnaire

1. What is your job title?
2. Have you ever wanted to compare two code files?
3. Do you struggle to find lines of code other developers have added to a project? 4. Have you used comparison tools before?
5. Were you able to compare your code and what did you use to do this?
6. What code language did you want to compare?

7. Do you ever need to find differences in different versions of documents?

8. Do you use Github?

APPENDIX 7

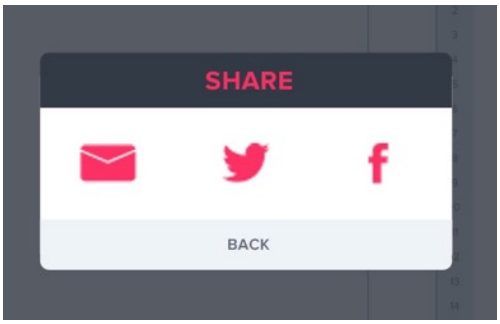

Figure 9.5

APPENDIX 8

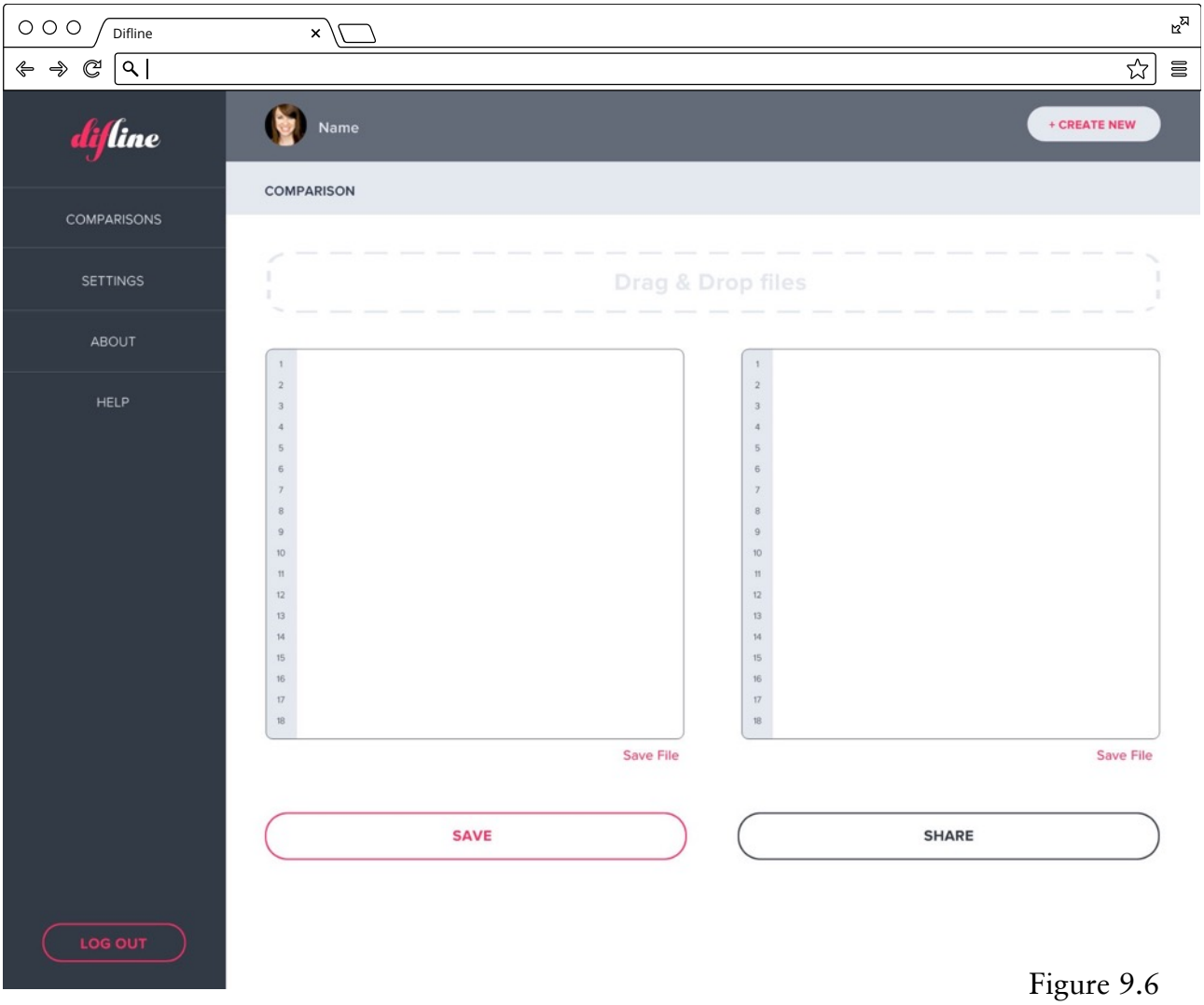

Figure 9.6

APPENDIX 9

Initial Settings form

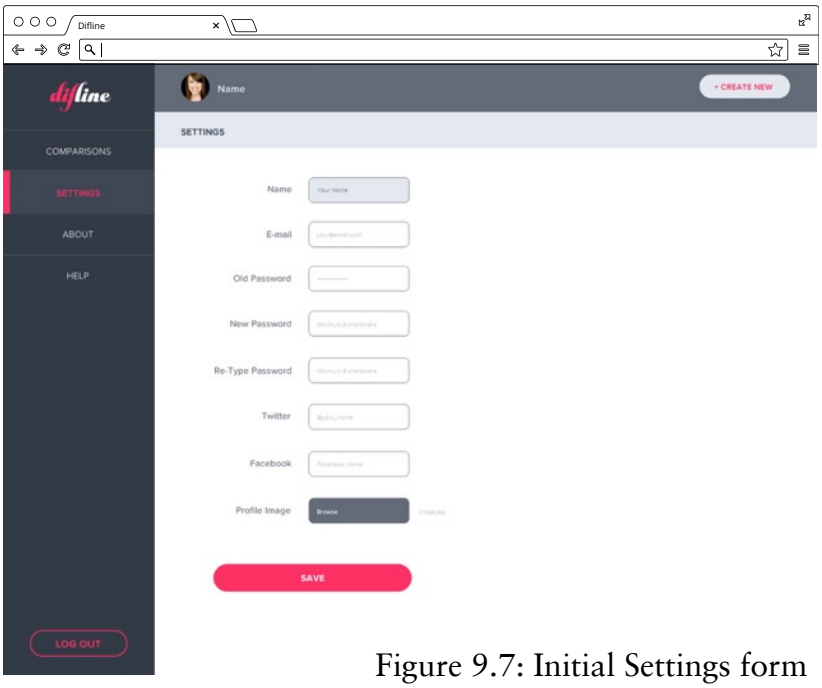

Figure 9.7: Initial Settings form

Final Settings form

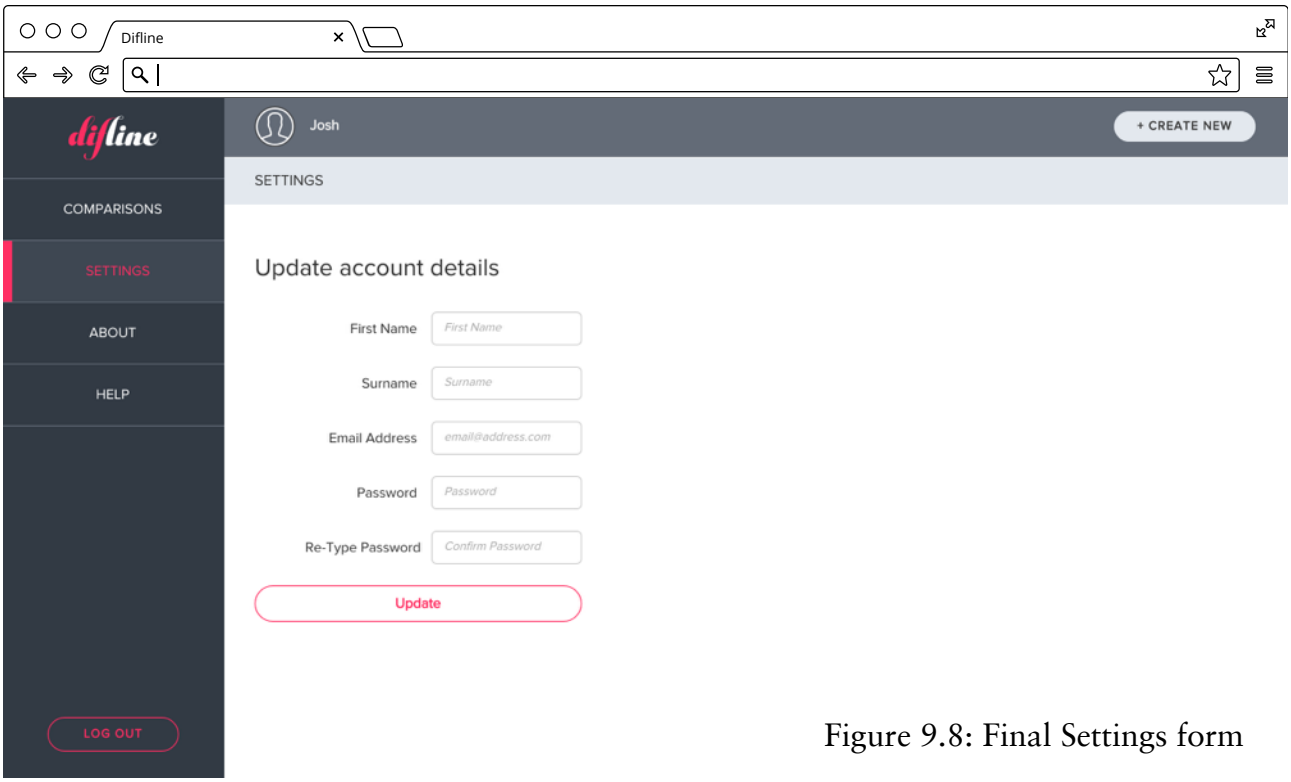

Figure 9.8: Final Settings form

APPENDIX 10

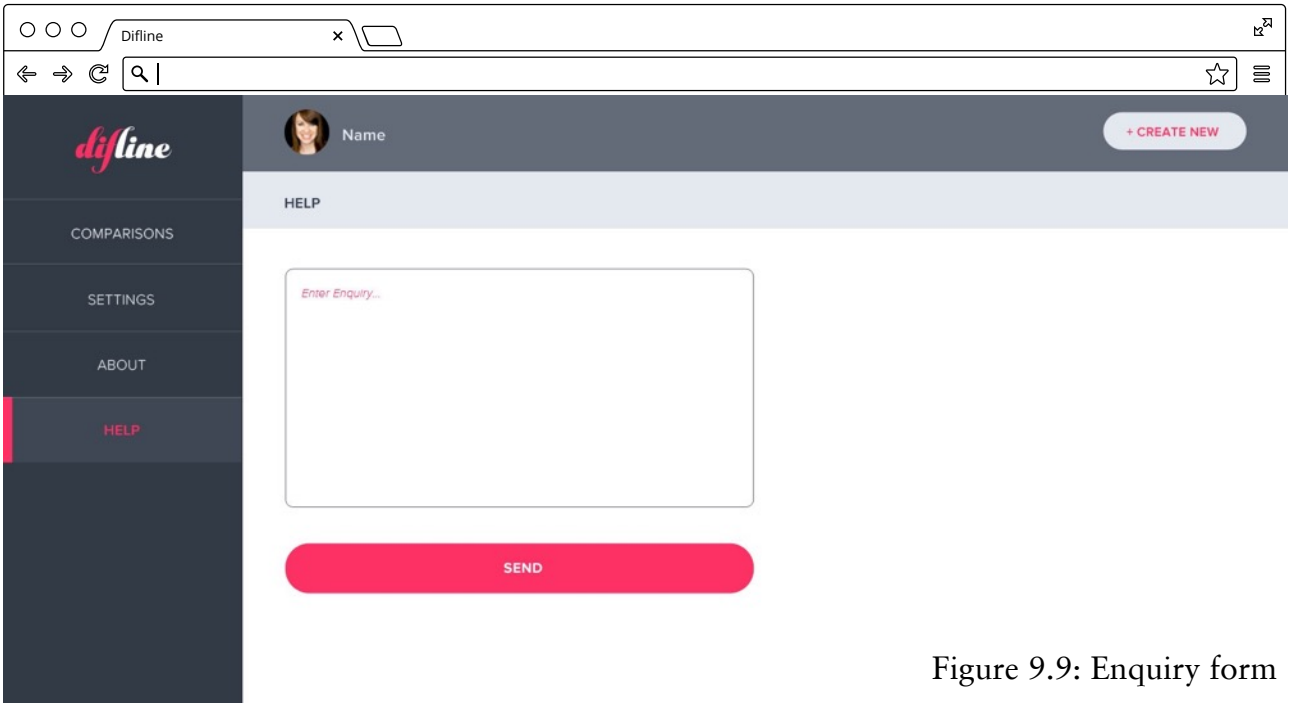

Figure 9.9: Enquiry form

APPENDIX 11

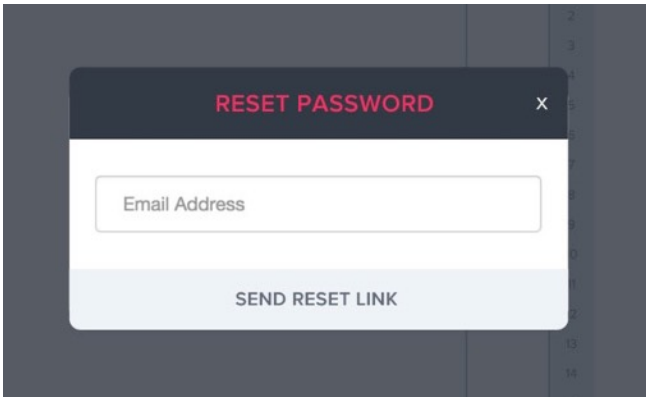

Figure 9.10: Initial reset password

APPENDIX 12

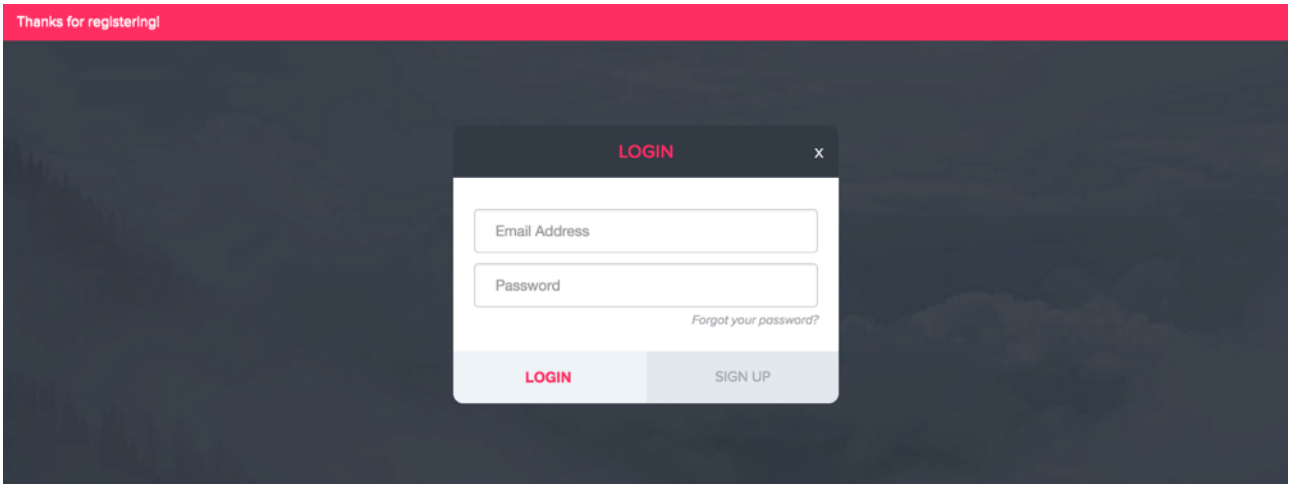

Figure 9.11: Sign up alert message

APPENDIX 13

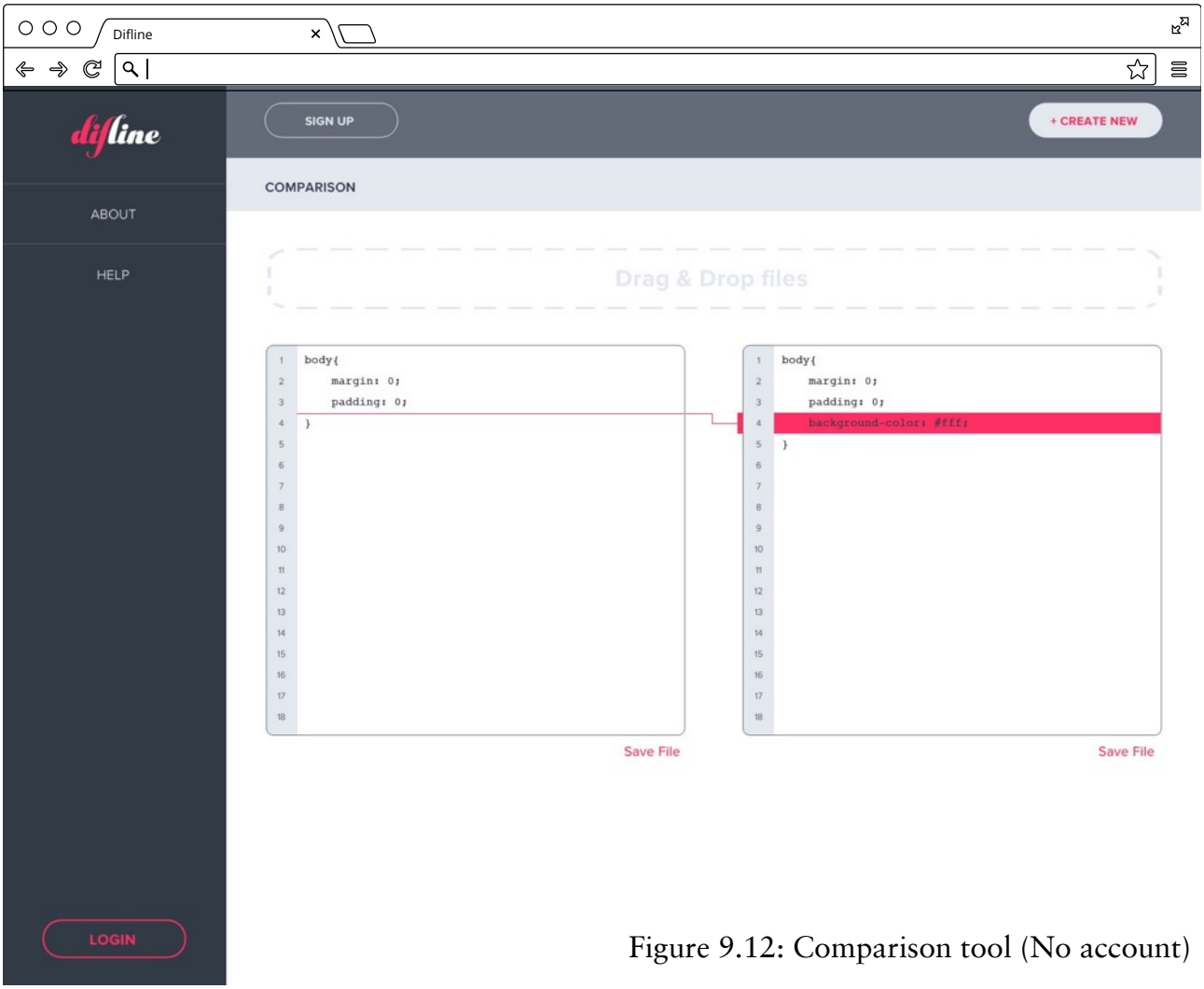

Figure 9.12: Comparison tool (No account)

APPENDIX 14

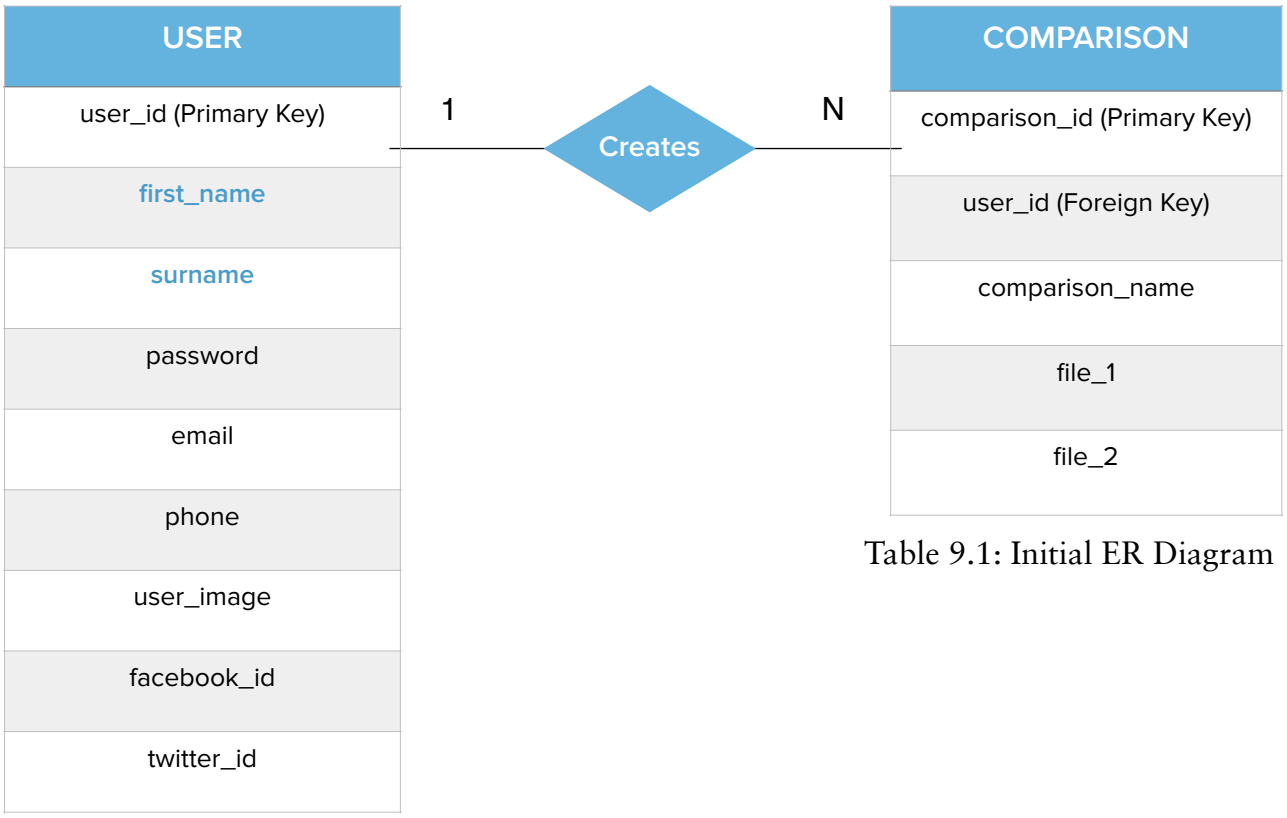

Table 9.1: Initial ER Diagram

APPENDIX 15

Update function

```
public function update() {
    $validator = Validator::make(Input::all(), User::$rules);

    if ($validator->passes()) {
        $user = User::find(1);
        $user->firstname = Input::get('firstname');
        $user->lastname = Input::get('lastname');
        $user->email = Input::get('email');
        $user->password = Hash::make(Input::get('password'));
        $user->save();
        return Redirect::to('/settings')->with('message', 'Settings updated successfully');
    }else {
        return Redirect::to('/settings')->with('message', 'The following errors occurred')->withErrors($validator)->withInput();
    }
}
```

Figure 9.13: Update function

APPENDIX 16

Eloquent model

```php
<?php

use Illuminate\Auth\UserTrait;
use Illuminate\Auth\UserInterface;
use Illuminate\Auth\Reminders\RemindableTrait;
use Illuminate\Auth\Reminders\RemindableInterface;

class Compare extends Eloquent implements UserInterface, RemindableInterface {

    public static $rules = array(
        'lhs'=>'required|alpha|min:2',
        'rhs'=>'required|alpha|min:2',
        'comparison_name'=>'required|alpha|min:2'
    );

    use UserTrait, RemindableTrait;

    /**
     * The database table used by the model.
     *
     * @var string
     */
    protected $table = 'comparison';

    /**
     * The attributes excluded from the model's JSON form.
     *
     * @var array
     */
    protected $hidden = array('password', 'remember_token');

    protected $fillable = array('comparison_name', 'lhs', 'rhs', 'id');

    public function user()
    {
        return $this->belongsTo('User');
    }

}
```

Figure 9.14: Eloquent model