Final Report

BSc (Hons) Interactive Multimedia Design

2014

Gareth Donaldson – B00564645

Peer support Group number

Dr Giuseppe Trombino

Acknowledgements

I would like to express my deepest appreciation for my mentor, Dr Giuseppe Trombino. He convincingly pushed me to learn a new framework was always on hand to help me. Without his guidance and support this project would not have been possible.

I would also like to extend my gratitude to my fiancée who helped me throughout this whole project.

Table of Contents

Chapter 1 - Introduction 5
1.1 Introduction 5
1.2 Project Aim 5
1.3 Project Purpose 5
1.4 Project Objectives 6
1.5 Overview of Report 6
1.5.1 Concept definition and Requirements 6
1.5.2 Design 6
1.5.3 Implementation 6
1.5.4 Testing 7
1.5.5 Evaluation 7
1.5.6 Conclusion 7
Chapter 2 - Concept Definition and User Requirements 8
2.1 Introduction 8
2.2 Concept definition 8
2.3 Anticipated Solution 9
2.4 Quality Objective 9
2.5 Technical Objective 10
2.6 Project Feasibility 10
2.7 Requirement Gathering 11
2.7.1 Current Market 11
2.7.2 Unified Modelling Language 12
2.8 Summary of User Requirements 13
2.9 Specific Requirement Risks 14
Chapter 3 – Design 15
3.1 Introduction 15
3.2 Logo Concept 15
3.3 Paper Prototype 17
3.4 Prototype Phase 20
3.4.1 Homepage 20
3.4.2 Job Search Page 23
3.4.3 Job Page 24
3.5 Flow Chart 26
3.6 System design 27
3.6.1 Laravel 27
3.6.2 Platform Architecture 28
3.6.3 'Model View Controller' (MVC) Architecture 29
3.7 Logic design 30
3.7.1 Data Structure 30
3.7.2 'Entity Relationship Diagram' 31
3.7.3 Communication between Laravel and the database 32
3.7.4 Displaying Data 33
Chapter 4 -Implementation 34
4.1 Introduction 34
4.2 Methodology Selection 34

4.3 The Prototyping Phase 35
4.4 User Authentication 35
4.4.1 Registration 35
4.4.2 Sign In 38
4.4.5 Sign Out 39
4.5 Edit User Profile 41
4.6 View all Jobs 42
4.7 Create a Job 43
4.8 Show Specific Job 43
4.9 Problems encountered 44
Chapter 5 - Testing 46
5.1 Introduction 46
5.2 Testing Approach Selection 46
5.3 Testing Process 46
5.4 Test Results 47
5.4.1 Sign In 47
5.4.2 Registration 48
5.4.3 Edit User Profile 48
5.4.4 Other Tests 49
5.5 System Test 49
5.6 Security Test 50
5.7 Summary of Test Results 50
Chapter 6 - Evaluation 51
6.1 Introduction 51
6.2 Evaluate Test/Survey Results 51
6.3 Evaluate Project Outcomes 51
6.4 User Observations 53
6.5 Evaluate the Methodology 54
6.6 Evaluate the Requirements 54
6.7 Future Development 55
6.7.1 Secure payment and private discussion 55
6.7.2 User Rating System 56
6.7.3 AJAX discussion thread 56
Chapter 7 - Conclusion 57
Chapter 8 – References 59

Chapter 1 - Introduction

1.1 Introduction

This chapter will introduce the purpose, aims, and objectives of this project. There will also be an overview of work undertaken and a brief summary of each chapter in the report.

1.2 Project Aim

The aim of this project is to create a website for any person who requires specific, professional and unique images, and for talented photographers who wish to offer their service to others. The user will have to register and sign up before they can have full use of the website. The user interface needs to be plain and simple so that any user can quickly understand the layout and the process involved in adding or applying for a job. Those signed up to the website will also have a user profile which other users can see, and which will include some personal information and useful details, such as a link to their portfolio.

1.3 Project Purpose

The purpose of this project is to create a marketplace where designers or others working in similar professions can create a job to receive high-quality, bespoke images. The current situation for designers when looking for such images is that they can either browse stock image websites or take the image themselves. There are several pitfalls to both options; for instance, in the case of using stock images, the picture or photograph wanted may not exist or other businesses might use the same image, and when using a self-made image, there may be a lack of professional skills and resources available to achieve the desired result.

This project will allow users to simply add a job to the website, and for other users to send a proposal to complete this job. This will potentially save the user

many countless and monotonous hours looking for the ‘right’ image, or trying to create it for themselves.

1.4 Project Objectives

1. To research and evaluate current systems that provide a service where users can get high-quality images.
2. Create a website in which the user interface is intuitive so that anybody can use this service.
3. Allow users to create an account, and update and delete their accounts
4. Enable users to add a job to get ‘personalised’ images and photographs.
5. Provide users with work opportunities and a platform to promote their skills and talents.

1.5 Overview of Report

1.5.1 Concept definition and Requirements

This chapter illustrates the research that was carried out for this project, such as the current market and the anticipated solution. The chapter will also outline different objectives such as the quality aspect of the website. Furthermore, it will detail how the user requirements were gathered by using a number of different techniques, before summarising said user requirements.

1.5.2 Design

This chapter will describe how the user experience was improved, beginning with the paper prototypes through to the final user interface. The system design will also be explained in detail, from the file directories to the Model View Controller architecture. Finally, the data structure will be outlined as well as how the database will communicate with the website.

1.5.3 Implementation

The implementation chapter will describe the development process involved in creating this project. It will describe how key aspects of the website were developed and will highlight problems that occurred during this phase of the project, along with how these issues were resolved.

1.5.4 Testing

This chapter will explain the testing strategy used throughout the project and summarise the testing results.

1.5.5 Evaluation

This chapter will evaluate the project as a whole and determine whether or not the user requirements were met. The methodology will also be examined and assessed as to whether it was the correct way to go when completing this project.

1.5.6 Conclusion

The final chapter will reflect on the entire project and summarise the written documentation for the project. A description of the achievements and experience gained after concluding the project will be noted. Lastly, future developments will be discussed for future enhancement.

Chapter 2 - Concept Definition and User Requirements

2.1 Introduction

This chapter will illustrate the research that has gone into evaluating the feasibility of Huddlepix. A set of defined requirements will determine whether or not the completed project is a success, and these will be closely examined within this chapter; this will allow for a more efficient workflow for the entire project.

2.2 Concept definition

To develop a marketplace where web designers, graphic designers and others working in similar professions can request and access high-quality, bespoke images. The user can place a request by filling out a form; this will include their requirements for the images such as the main subject, landmarks and locations, as well as preferred orientation, composition and overall style of the photograph. It will also be necessary for the user to specify their budget, clarifying what they are willing to pay. This information will then be advertised as a 'job' on a feed on the website, listed by newest to oldest.

All users will create a profile containing relevant personal information such as skills and location, in addition to a link to their portfolio of work.

Each job will have a web page where any registered user can see all information concerning that job. Users will have the option of posting a public question about the job if the job brief is not detailed enough. The next stage, after a user has found a job that they wish to undertake, is to fill out a proposal. The proposal will consist of a proposal brief and a proposed price; the proposal brief will be made up of information stating why they are best suited to the job at hand. The user that posted the job will be able to view all proposals from their

dashboard and view the profile of any other user who has either sent a public message or a proposal for their job.

Finally, the user that posted the job will be able to accept a proposal which will automatically close the job and send an email to the accepted user confirming they have been accepted to undertake this job. From this point onwards, the users will communicate via email or telephone.

2.3 Anticipated Solution

The anticipated solution is that this website will save people from wasting countless hours trawling through pictures on stock image websites and allow them to get the images they actually want and need, which in turn will help improve their work. It is also creating a community and helping people get paid work based on their photography skills; particularly beneficial for all those amateur photographers looking to get recognised and acknowledged.

2.4 Quality Objective

The website will adhere to industry standards and follow the best practices which have been set by W3C. To ensure that the website will meet these standards; the following tools will be used:

- http://validator.w3.org/
- http://jigsaw.w3.org/css-validator/
- http://validator.w3.org/checklink
- http://bradfrostweb.com/demo/ish/

The first two links will ensure that the HTML and CSS code are compliant with the industry standards. One reason for these validators being so invaluable to any developer is if when using valid code, another developer will face no problems maintaining the website (Bradley, 2011). The third link will identify if

there are any dead links within the website. The last link will display the website in many different viewpoints and sizes, helping to ensure that the website looks and functions well across all screen sizes and devices.

2.5 Technical Objective

The website will have to work across all the current browsers including mobile browsers and legacy browsers such as Internet Explorer 9. In achieving this however, it can take a considerable amount of resources to ensure that it is pixel-perfect in every browser. On considering this, it was felt that this might be a waste of valuable time and resources; therefore it will be designed so that each user will get an efficient, albeit possibly different experience.

The websites performance will be also taken into consideration, such as load times. The following tools will be used to maximise the websites performance:

- http://cssminifier.com/
- http://jscompress.com/
- https://tinypng.com/

Loading time is very important, as speed is critical to providing a good user experience in the digital world today. Users may abandon a page if it takes too long to load (Benjamin, 2013).

2.6 Project Feasibility

Taking into consideration all of the points above, it has been determined that this project is entirely feasible in relation to the criteria outlined/specified, and within the allocated time-frame. A gannt chart has been created to ensure that the project uses all available time wisely (see Appendix A).

2.7 Requirement Gathering

The requirements were gathered by a number of different methods, such as analysing the current markets and creating a unified modelling language diagram. These requirements are not, however, set in stone; a number of prototypes will be created and tested by users, and during these sessions the requirements may be adapted or changed to suit the users.

2.7.1 Current Market

Hiring a photographer will often guarantee the best image possible as they are regarded as being the professionals in their field. To get that image however, a person may have to spend a fair amount of time researching and contacting photographers and quite a considerable amount of money. An alternative to this would be to use your own images, yet if a person knows little or nothing about photography there is a strong likelihood that these may not turn out as well as they could.

One other way to attain good quality images is to use stock image websites. There are a wide range of these websites online, the majority of which are based on user libraries and searching. The problem with stock images is that they are not original, they are generic; this will not showcase a brand in the best way. The other main issue is that anybody can buy the rights to the stock image being used for a specific project, and this could prove problematic if a competitor chooses to use the image (Belkis Cardona-Rivera, 2012). One further disadvantage is that using stock images can sometimes lead to legal issues, as the buyer does not always understand what license they have purchased; ‘Royalty-Free’ or ‘Rights-Managed’.

2.7.2 Unified Modelling Language

UML visualises the high-level user requirements through case diagrams, ensuring no requirements are left out. The users that were identified are as follows:

- Registered Users: this user will be able to sign into the website, view other users profiles, view and post a job, send and accept proposals and ask a message on a specific job.
- Un-Registered Users: this type of user will be able to view the homepage, and FAQ page; they will be able to create an account and contact the administrator.

The UML diagram for Registered Users is shown in Fig 2.1, which has identified a number of requirements.

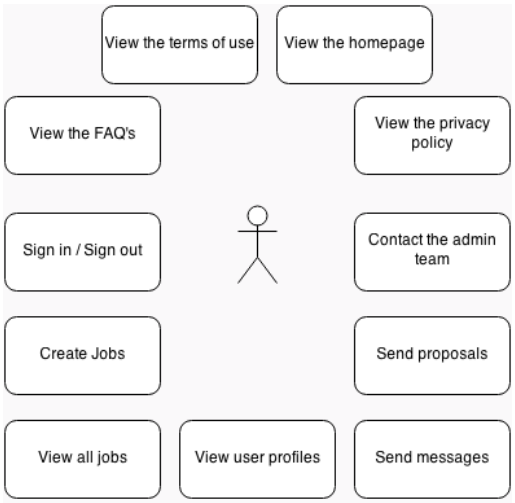


Figure 2.1 - UML diagram for registered users

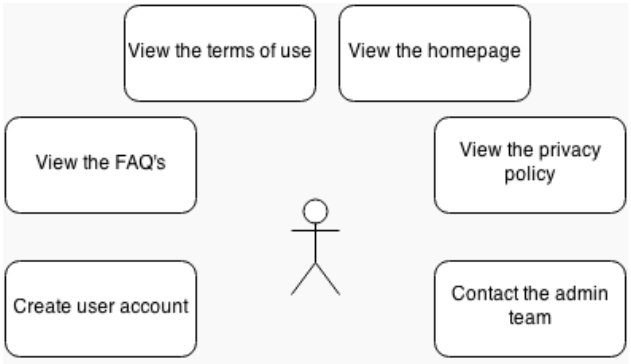


Figure 2.2 – UML diagram for un-registered users

Explanations of the registered users requirements are as follows:

- Sign In / Sign Out of user account: A registered user will be allowed to sign in and sign out of Huddlepix website.
- Post/delete a job: A registered user will be able to post a job which is visible to all other registered users and delete a job if they no longer require that service.

- Send a message: A registered user will be able to send a public message on any specific job.

The UML diagram for Un-Registered Users is shown in Fig 2.2, which has identified a number of requirements.

Explanations of the un-registered users requirements are as follows:

- View the homepage: an un-registered user will be able to view the homepage, where they will be able to get information on the service provided by Huddlepix. They will be able to see details about the three current most popular jobs that are available to registered users.
- FAQs: an un-registered user will be able to view the FAQ webpage; this will hopefully answer any questions they may have about Huddlepix.

2.8 Summary of User Requirements

The summary of the user requirements has been placed into three columns; the first column is the 'Requirement ID', the second the 'Priority Level', which is a number based column with the numbers ranging from 1-5 with 5 being the most important, and finally a brief 'Description' of the requirement. Appendix B provides a full list of the system requirements, which has been developed using the Volere framework.

Requirement ID	Priority Level	Description
001	3	Product Website
002	5	User Registration
003	5	User Sign In / Sign Out
004	4	User Profile Page Page
005	3	Edit User Profile
006	5	Create Jobs
007	4	Close Jobs
008	4	View Jobs
009	4	Send Messages

010	5	Send Proposal
011	5	Accept Proposal
012	3	View all Proposals sent for a job
013	4	Cancel Account
014	1	Social Media
015	2	Contact Admin Team
016	3	Search for a job using keywords
017	3	Report Jobs that breach the terms and conditions
018	5	Customer details are securely stored in the database
019	5	Website meets industry standards

Table 2.1 – User Requirements

2.9 Specific Requirement Risks

While compiling the requirement list, a number of risks where identified that could lead to the project failing. The first high risk is giving the user the freedom and ability to set up an account with the website. The reason why this is deemed to be a high risk requirement is due to the fact that a database will have to be linked up to the website, and this will need to be constantly updated when a new user sets up an account. Another risk that is closely linked with this requirement is the need to ensure all personal data remains confidential and secure. The project will use MySQL and Laravel built-in encryption functions to secure all sensitive data and make sure the database is protected from an SQL injection attack. SQL Injection is a type of web application security vulnerability in which an attacker is able to submit a database SQL command which is executed by a web application, exposing the back-end database (Glynn, 2013).

Chapter 3 – Design

3.1 Introduction

This chapter will discuss how the user interface was developed from the paper prototype to final design. The system design will be examined, from researching the framework to understanding the Model View Controller approach. Finally, the chapter will end by looking at how the database was created, ensuring that all the tables have been linked.

3.2 Logo Concept

The first step in this process was to go straight into Illustrator and come up with a number of designs for the logo, the majority of these focusing on the word 'huddle' (see Fig 3.1).

Figure 3.1 – Different concepts for Huddlepix

From this point, it was decided that a simple typography logo would be the best option; inspiration was taken from the new Instagram logo. The website 'wordmark.it' displays and offers a range of different fonts, and after some

browsing, it was a nice, clean font called Jubilet that stood out as a potential option.

The next stage was to experiment with this typeface, along with some others, to get a feel for what it could look like. It was here that point four of Logo Design Love's 'seven ingredients for your signature dish' came in to play: 'Aim for distinction' (Airey, 2009). After some more time and thought, a camera icon was incorporated into the logo. Fig 3.2 illustrates the process used to find the right font for the Huddlepix logo.

Figure 3.2 – Various fonts for Huddlepix

It was through experimentation that a better option than Jubliet was found; the font, named 'Bitter', is very similar to Jubilet, however it had more character and a nicer letter-style; more specifically the 'x'.

Figure 3. 3 – Huddlepix refinement

The next task was to alter the size of the camera, and after a little deliberation the camera was changed to be the same size as the letter 'h'. This looked much better, yet it then appeared that the camera icon and the text where too close together, so a small refinement was made to correct the spacing issue (see Fig 3.4).

Figure 3.4 – Final logo concept

3.3 Paper Prototype

This was the first stage when it came to designing the user interface. As stated by Medero (2007), 'the prototyping stage is the right time to catch design flaws and change directions, and the flexibility and disposability of paper encourages experimentation and speedy iteration'. It was determined that the most important web page of the website would be the job page; the reason for this is due the fact that the majority of the content will be made up of jobs posted by users. All the paper prototypes can be viewed in Appendix C.

Before the ideas were sketched, the main consideration was what information the user would want from this webpage. From here, it was decided that this information would be best split into five sections.

The first section is 'Job Information', which consists of the title of the job, a countdown to when the listing of the job will end, the users budget and the number of users that have sent a proposal for the job.

The second section is the 'Job Brief', where the user will go into more specific detail about the job required from the photographer; information such as the timeframe, location and the main subject and/or theme of the image will be outlined.

The ‘Proposal’ section- the third section- will allow the users to send a proposal to the user; this will contain information on how they would go about the job and a price it would cost to do this job.

The fourth section provides the photographer with information about the user that has posted the job, including an image and their ‘performance’ rating.

The discussion makes up the fifth and final section. The board itself will allow the photographers to ask the user questions directly and the discussion will be made public, as it is quite possible that the user may have left out some relevant information about the job which others may benefit from knowing.

For the first sketch, the layout was created to be very simple using a one and two column layout. The logo, main navigation and the job information would use the one-column layout, while the job brief, discussion board and the user profile used a two-column layout. The proposal section was left out, and instead two links to the proposal page were included. I found two problems with the first sketch. The first of these problems was that the proposal section needs to be on the job page, as without it the user has to load another page unnecessarily. The second problem is the user profile; it has the same amount of space as the job brief and discussion section, however it is not so important as to take up 50% of the web page.

The second sketch is where the problems in the first concept were fixed, predominantly by adding the proposal directly underneath the job brief. This was felt as being more inviting and encouraging to the photographer to send a proposal, as they can immediately see they are only required to fill out three small sections. The user profile section was also made smaller, and as a result it gave more ‘essential’ information extra room on the page. One further improvement was the inclusion of an off canvas navigation, as it would allow further space for any additional, relevant information.

The third sketch involved some experimenting with a 3-column layout, where the middle column is bigger than the first and last column. The first column is taken up with the user profile and their information, and the last column has the job information and discussion board. The middle column would comprise of a generic image with the job brief and proposal underneath it. On paper, however, this idea just didn't work as all the content seemed much too 'squashed' together and it did not have any real flow to it.

In the fourth sketch, the design reverted back to the 2-column layout; the ratio 1.618 was used, which is known as the 'golden ratio'. The first column incorporates the logo, search bar and the navigation. The second column takes the 'User Profile' section and splits it into a further 2 columns; the first column is the users image and the second displays their name and rating. Underneath the 'User Profile' section, the 'Job Information', 'Job Brief', 'Proposal' and the 'Discussion Board' follows. This was thought to be an improved design of the second sketch, however it was still judged as not being quite right as the final outcome.

The fifth sketch is a variation of sketch four. The header was adapted to have the logo and 'Sign-In' feature. The first column contained the user image and information followed by the navigation. The second column splits the job information into two columns, which then reverts back to one column for the job brief, discussion board and the proposal. It was considered that it might be better for the discussion board to be above the proposal, however on looking further at the fifth sketch, this did not fully work as there may not be any comments and furthermore, other users might not want to see the discussion.

The final sketch is basically the same as the fifth sketch, however under the header it has a generic image. This was seen to be a nice way to split the content up. The discussion board and proposal were also swapped around.

The next step was to come up with the final idea (see Appendix D). The best elements of each design were pulled together in order to create a web page that has the most appealing and effective layout. The header consists of the logo and the main navigation. The body of the website consists of a 2-column layout; the first column displays the user profile picture and their rating, which is then followed by the secondary navigation. The secondary navigation consists of the job information, job brief, proposal and the discussion board. This column, along with the header, is fixed; this means the user will also have access to the both of the navigation menus. The main content section has been split into three sections. The first section contains the job information and job brief, and is followed on by the proposal and then finally, the discussion board.

3.4 Prototype Phase

3.4.1 Homepage

All of the original designs for Huddlepix are located in Appendix E, and can also be seen throughout this chapter. All of the final designs for Huddlepix are located in Appendix F, and can also be seen throughout this chapter.

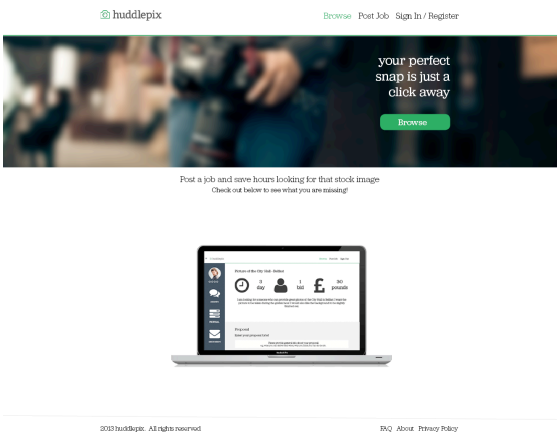


Figure 3.5 – Original Huddlepix design

The original design for Huddlepix's homepage is shown in Fig 3.5. The users that tested the first prototype discovered a number of issues with this design such as that it did not fully explain the service that will be provided within the

website. There were also no testimonials that could potentially encourage viewers to become registered users.

As the main purpose of the homepage is to convince visitors to become registered users, it was thought best to keep the content relevant and to the point, with a simple design. The final design clearly promotes the service which is provided within the website, and makes registering easy-to-do; there are numerous links for the user to register, testimonials that appear on a slider and popular jobs that are available to registered users are highlighted. Each main section is split up by either images or 'register now' buttons, allowing the content to flow better.

Figure 3.6 – Final Huddlepix design

The footer has also been improved by adding more useful links which give visitors more information, and added all the social media links. The final design can be seen in Fig 3.6.

When the user clicks on the 'Sign In' button, a model appears allowing the user to sign in (see Fig 3.7); there are two text inputs with username and password as the placeholders, a 'remember me' check-box and a Sign In button. It was important to again provide non-registered users with another link to the registration page.

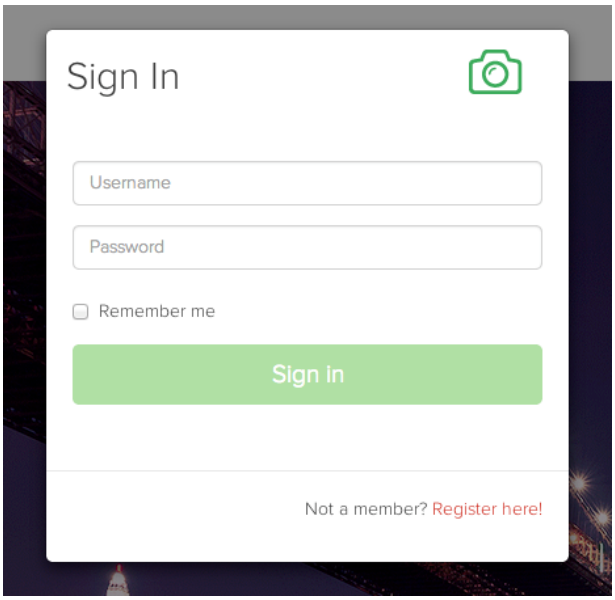


Figure 3.7 – Sign In model

It was decided that the user interface would be improved with the addition of more colour so as to create more of a contrast between each section, and so that the user is more aware that they are looking at the next section. A pastel colour palette was chosen to ensure that the design is kept soft, almost neutral, and that it complimented the other colours within the design. The typography was changed from Jubliet to Proxima Nova Light; this helped improve the readability of the text.

3.4.2 Job Search Page

The main concern with the original design was that the sidebar had no real functionality (see Fig 3.8). The table, on the other hand, has many different functions; the first function is that the user can select how many job entries they would like to view at one time. The next function is the ability to search for anything within the table which allows the users to quickly look for specific jobs. There is also pagination, which will allow the user to click through all the rest of the table

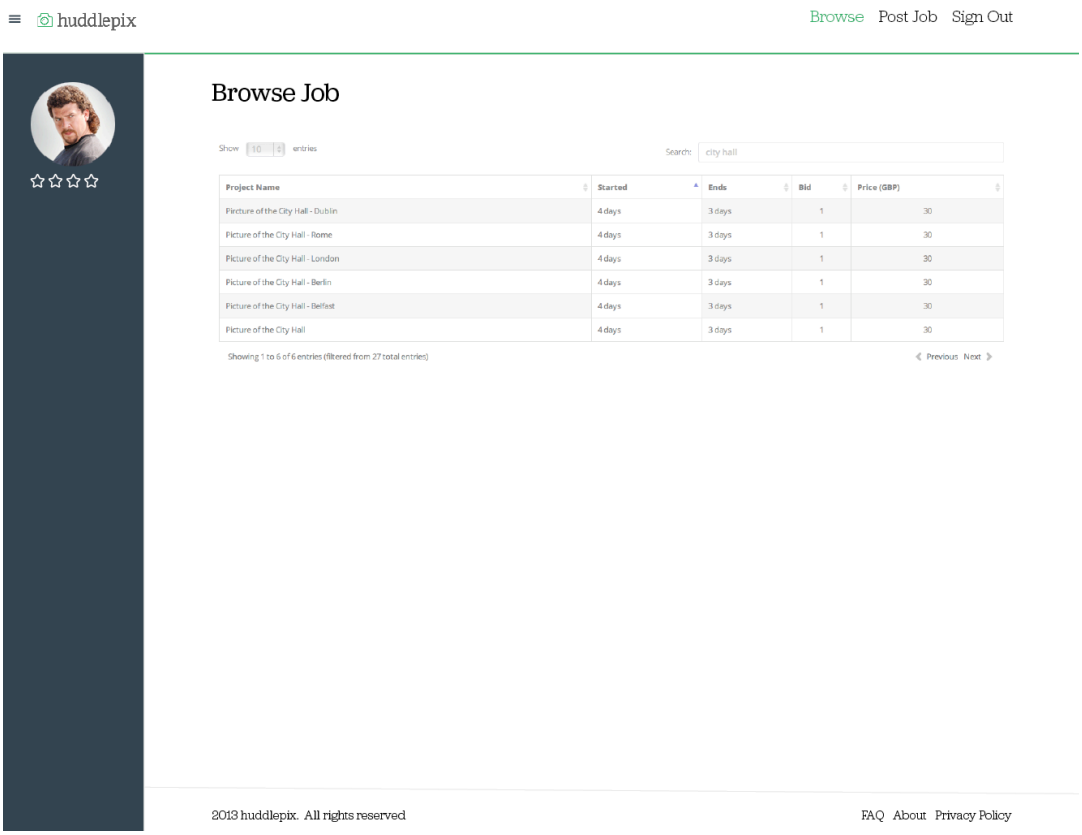


Figure 3.8 – Original Huddlepix design

The final design was altered slightly so the sidebar is now the navigation, through which users can view and post a job, view their profile and sign out which is shown in Fig 3.9. The header section now contains the users avatar, their name and tagline. I have changed the columns in the table to the following: Project Name, Location, Ends, Number of Proposals, Price, and added a link to the job page.

A further change involves the colour scheme for the dashboard section; the navigation is now a purple and the contrasting colour is green, which is also used in the homepage. The purple is still within the pastel palette and this ties in with the homepage colours. The background colour is a grey / purple colour with the page content on a white background, creating more contrast. As was observed by Boudreaux (2012), 'many people are colour blind or visually impaired, and ensuring that you website colour contrast is adjusted accordingly provides added responsiveness and accessibility'.

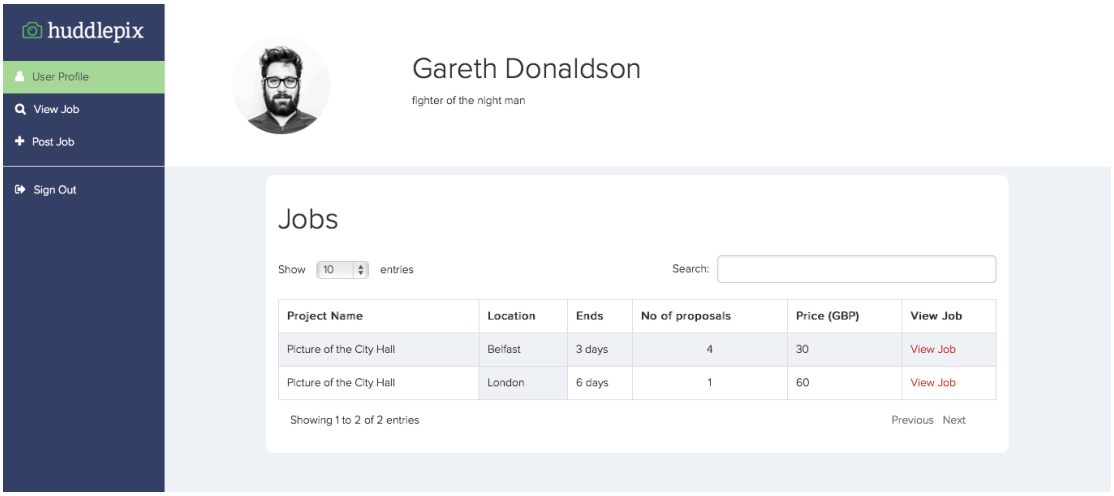


Figure 3.9 – Huddlepix final design

3.4.3 Job Page

The original design of the Job Page (see Fig 3.10) had a number of minor problems such as the 'X' icon beside the description being too far away as to understand its purpose. One major problem identified, however, involved one of the icons at the top of the page- the clock. Users initially thought that the job had to be completed in 3 days rather than the intended message being that the job would end in 3 days.

The final design which is shown in Fig 3.11, carries on using the same format as the job search page, but some of the aforementioned problems have now been addressed; firstly, by changing the clock icon text from '3 days' to 'closes 6d

23h'. This concept was implemented for all three icons. This small change managed to resolve the issue and thus will improve the user experience.

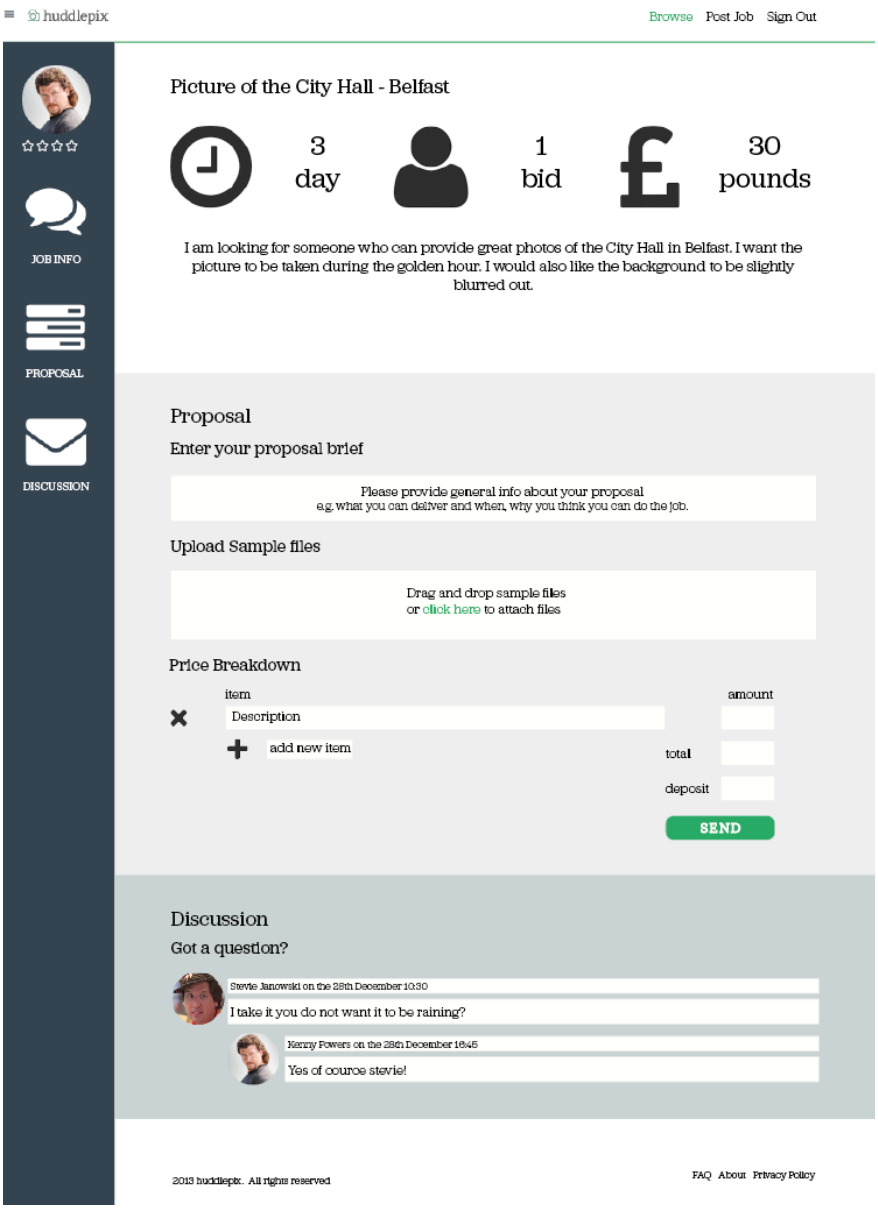


Figure 3.10 – Original Huddlepix design

The 'Proposal' section was then made to only include a proposal brief and the price the user will charge for the job; more information will be provided on the users profile page. There are no dramatic changes to the 'Discussion' section, except that it now allows the users to input text.

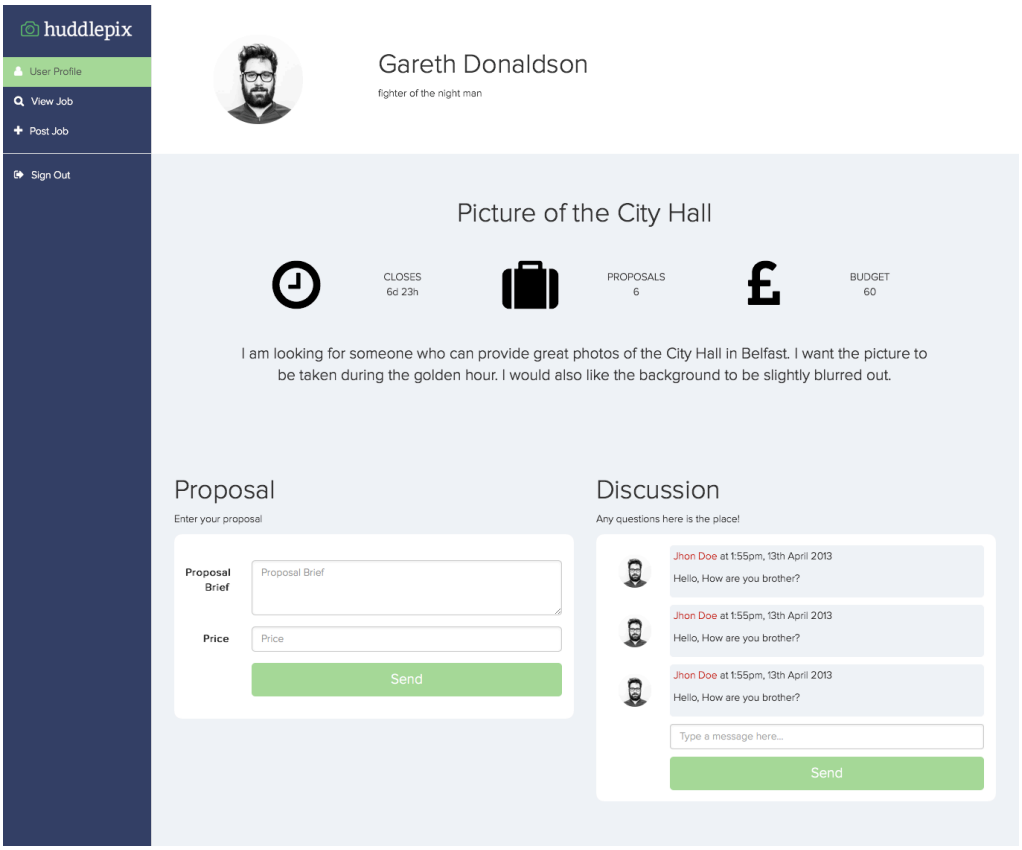


Figure 3.11 – Final Huddlepix design

3.5 Flow Chart

The chart shown in Fig 3.12 shows the process in which way the users can move about the website. From the homepage, the visitor can either go to the registration page or click the sign in modal, which will allow the user to sign in to the dashboard. Upon signing in, the user is redirected to their user profile; from here they can post a job or view all jobs. On the view all jobs page, they can view a specific job for which they will have the opportunity to simply ask a question or send a proposal. If the user wishes to send a proposal then they will fill out the form and once sent, the user that has posted that specific job will receive this in the form of an email.

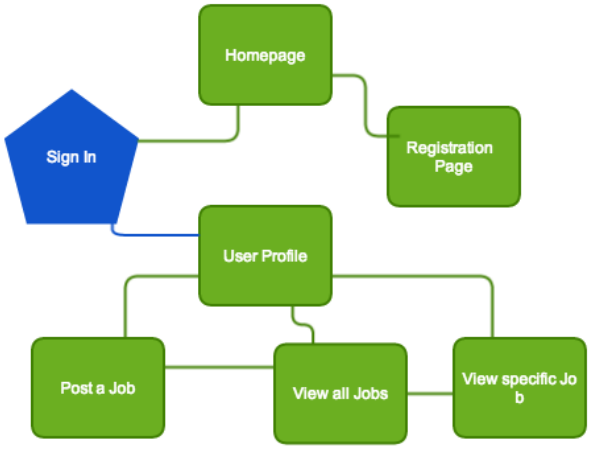


Figure 3.12 – Huddlepix Flow Chart

3.6 System design

3.6.1 Laravel

Any web framework should, in theory, make developing an application easier and act as an alternative to procedural PHP or plain JavaScript. Most applications are built on the same foundation; form validation, database abstraction or session and cookie handlers. By using a framework, these foundations will already be built in your system, therefore meaning that it is unnecessary to create them or to spend countless hours searching for 3rd party code. The web developer will then avoid having to search for and write less code, which will in turn lead to faster development; provided they have learned specific framework.

The reason why Laravel was chosen over numerous other frameworks is primarily because it has a fantastic community behind it for being such a young framework. It uses a 'Model View Controller' architecture, which will ensure that it is easier to maintain the website; the code will be easier to understand as it will be broken down into each different sections.

3.6.2 Platform Architecture

Laravel has a strict directory structure, which consists of four main directories and quite a few subdirectories. The four main directories are as follows: app, bootstrap, public and vendor.

The app folder contains all of the system models, views and controller, as well as many other configuration folders such as the database and storage folders.

The bootstrap folder contains scripts that relate to the start-up procedure for the framework itself.

The public folder will pointer to the web server; as this folder contains files that are used to build websites, the files that are found within this folder are CSS, JavaScript, images and the htaccess file.

The last directory is the vendor folder; this houses all the composer packages that will be used to create Huddlepix.

The image in Fig 3.13 shows all of the mentioned directories and the subfolders.

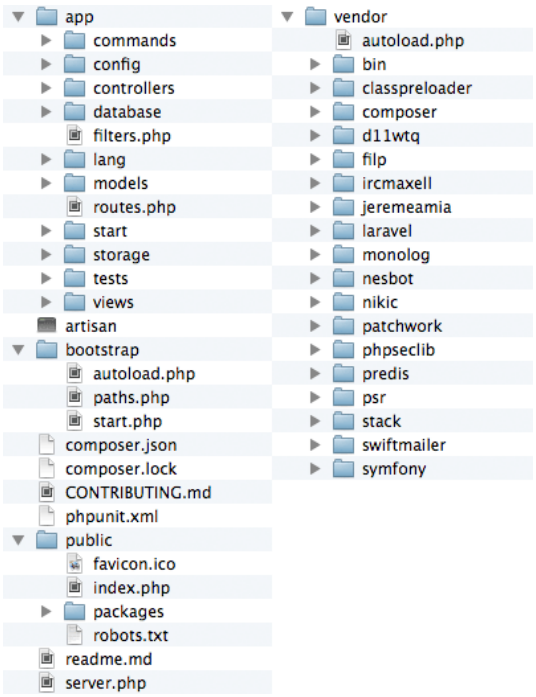


Figure 3.13 – Huddlepix Platform Architecture

3.6.3 'Model View Controller' (MVC) Architecture

The MVC approach will divide the code into specific responsibilities, which can be divided into three separate sections or folders; 'Model', 'View' and 'Controller'. These folders are located within the app directory of Laravel.

A 'model' can be summed up in one word- 'knowledge'. As mentioned by Way (2012), ‘Everything that is related to the knowledge of the domain should be located in the model layer’. The models should be split up into different layers, with each of the layers having a different responsibility; this will in turn result in a unique representation of each of the layers.

A 'view' is a visual representation of the application, containing static HTML; a view can only receive data and display the data which has been passed to a controller.

A 'controller' provides application logic, as it is forms the link between the 'model' and 'views'. The controllers will receive requests and return those requests with a response.

The image shown in Fig 3.14 illustrates how Laravel takes advantage of the MVC approach.

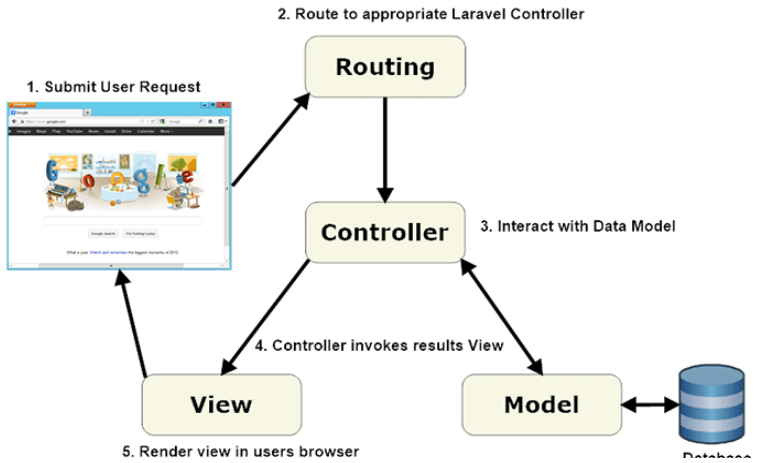


Figure 3.14 – How Laravel works

As can be seen in the Fig 3.14, Laravel uses the MVC approach with the assistance of routing; this is known as 'Laravel Routing Engine'. This engine can be accessed within the file called routes.php, which is located in the app directory. Within this routes file, the developer can define/register all the routes that the application is going to need to fulfil the system requirements. A short, simple explanation of the function of this file is that it will match what the user wants with the correct response.

3.7 Logic design

3.7.1 Data Structure

Huddlepix requires a database to hold all valuable data; the database design is essential to getting the application off the ground. The model within the app directory of Laravel will be constantly requesting data from the database, and as a result it is important to ensure that the data within the database is not duplicated. Relationships will also have to be created between the separate tables within the database; this will help in not duplicating the data. The reason why the database design is invaluable is down to the idea of performance; the quicker the model can search and gather data, the quicker it will be displayed on the users web browser.

The first decision made with regards to the database was which tables were needed and what data would be available for those tables. Four tables were felt as being enough for the first draft of Huddlepix, more specifically: 'users', 'job', 'proposal' and 'discussion'.

The 'users' table will contain all data about the user, from their username to their password, and whether or not their account is active. The user_id and username column will be unique for this table; this will mean that there will be only one specific username in the table.

The 'job' table contains all the data about the jobs that have been created by users. The data will range from the job title to the budget of the job.

The 'proposal' table contains all data about the proposals that have been sent for a specific job. Data will include the proposal brief and proposal price.

The 'discussion' table contains everything relating to the discussions that take place within each job. This data will include a message and a data.

3.7.2 'Entity Relationship Diagram'

The next step after creating all the tables that Huddlepix requires, was to move onto the relationships between the tables. An entity-relationship diagram was created this is a data modelling technique that creates a graphical representation of the entities, and the relationships between entities, within an information system (Rouse, 2007).

To better understand the relationships, it was necessary to break down exactly what was happening with the data within the database. It was in stating the different relationship types that a named association could be gained between the entities. This is how these relationships were established:

- A user can have many jobs
- A job can have many comments
- A user can only create one proposal

This means that relationships are as follows:

- USERS -> JOBS (1:N)
- JOBS -> COMMENTS (1:N)
- USERS -> PROPOSAL (1:1)

These relationships are visible in the 'Entity Relationship Diagram' below:

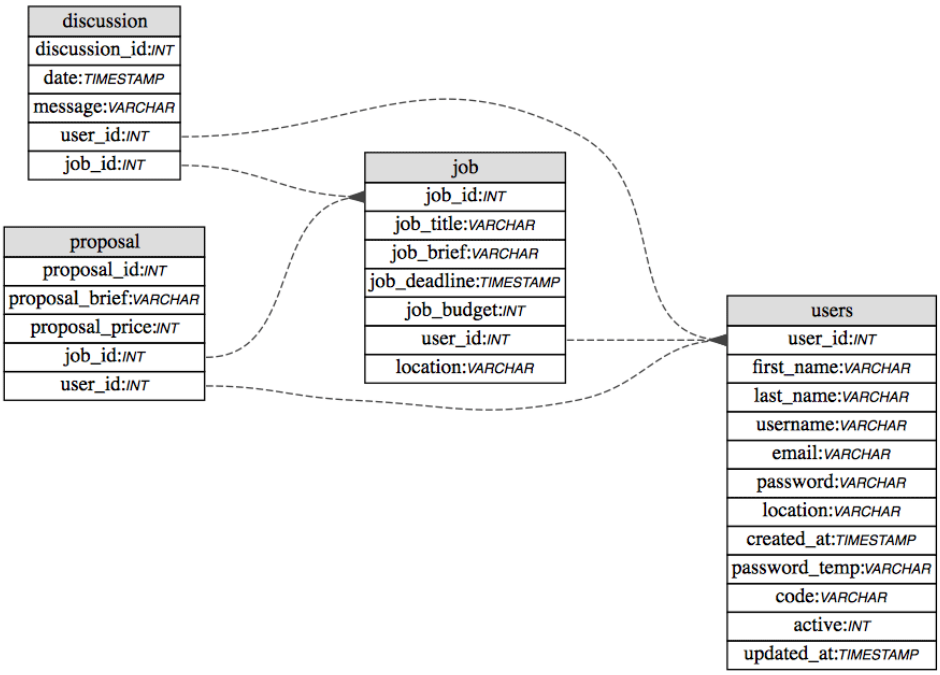


Figure 3.15 – Huddlepix Entity Relationship Diagram

3.7.3 Communication between Laravel and the database

The Laravel framework comes with an 'Object Relational Mapper' (ORM), namely, 'Eloquent'. An ORM is a clean way to map a database row to an object, and 'Eloquent' is a tool that allows the developer to query and manipulate the data within the database.

Every table within the database for Huddlepix will have a separate model; the model will start off by defining itself and extends ‘Eloquent’. The next step is shown in Fig 3.15; by default all of the primary keys are set to ‘id’, so to enable the database and Laravel to work correctly with each other, the primary key has to be updated. The last step is to define what fields can insert with data; by default Laravel protects all the fields against mass-assignment. Making an array with all the fields that can have data entered in by the user is one way to get around this issue.

```
protected $primaryKey = 'proposal_id';
protected $fillable = array('proposal_id','proposal_brief', 'proposal_price', 'job_id', 'user_id');
```

Figure 3.12 – Example of the Proposal model

3.7.4 Displaying Data

Laravel uses the 'Blade Templating Engine', which allows the developer to define sections of code, extend master layouts and include sections, loop and if statements. Blade is driven by template inheritance and sections (Taylor Otwell, 2013). Using the Blade Templating Engine improves the readability of the code for both developers and non-developers.

Chapter 4 -Implementation

4.1 Introduction

This chapter will describe the work undertaken to develop this project; this will range from the methodology to describing the code for the frontend and backend of the website. The chapter then ends with some of the implementation problems encountered whilst developing the website.

4.2 Methodology Selection

The methodology strategy that was chosen for this project was the Prototyping Model. The reason why this methodology worked for this project is down to the fact that most of the information was gathered at the start, and from this point, a prototype was designed based on those requirements. Once a prototype has been created there will be a greater understanding of the system, and thus the identified requirements can be refined ensuring the best possible website being developed.

A further reason for choosing this methodology is that the desired website would promote a lot of interaction with the end user, and so it was important to include these users within the initial design process in order to receive feedback on the website. Using this feedback, the previous prototype could then be modified and adapted as much as was needed (ISTQB Exam Certification, 2012).

Although the inclusion and contribution of the user was very useful and influential throughout the design and development, the prototype needed to be continually improved and consistently tested which, at times, increased the complexity of the project.

The way this project was managed was by finalising how the frontend of the website would look and feel; this was accomplished by developing a design

based on the one-up which was created in the paper prototype phase, and allowing a number of users to test how that design worked within the website. From this point, the design was changed, adapted and improved based on the feedback given during these meetings. Whilst developing the backend of the website, the same concept was used as that which worked to develop the frontend of website.

4.3 The Prototyping Phase

The design of the frontend can be seen in chapter 3; this outlines the evolution of the design, whilst constantly reviewing the users feedback. Once the frontend was completed, the next stage was to develop the website; all the information would have to be stored and retrieved from the database, users would need to be able to create an account and log in to the website and finally, jobs would need to become created and available to view and/or be applied for.

4.4 User Authentication

The first stage was to create a user authentication system. This system will allow users to register an account with the website, sign in to the website and equally importantly, sign out of the website. The first prototype concept allowed the users to do all of the above; when the user registered, a pop-up appeared confirming that the account had been created which then enabled the user to sign in and out as they pleased.

Upon trialling this first prototype, the received feedback flagged up a number of issues with the user authentication system. The ways in which these issues were resolved are discussed below.

4.4.1 Registration

When users were creating an account with the website, some of the users left out fields which were required to complete the registration process. To solve this issue, it was decided that all forms should be validated. As the system is being

built using the Laravel framework, this was achieved through the use of the inbuilt ‘validation class’.

```
$validator = Validator::make(Input::all(),
    array(
        'first_name' = > 'required|max:400',
        'last_name' = > 'required|max:400',
        'location' = > 'required|max:400',
        'email' = > 'required|max:400|email|unique:users',
        'username' = > 'required|max:50|min:3|unique:users',
        'password' = > 'required|min:6',
        'password_again' = > 'required|same:password',
        'tag' = > 'required|max:400',
        'occupation' = > 'required|max:400',
        'portfolio' = > 'required|max:400',
        'avatar' = > 'required',
        'birthday' = > 'required')
);
```

Figure 4.1 – Validation of form

As can be seen from Fig 4.1, each of the fields from the registration form have been identified and from here, the system can introduce specific rules for each field. For example: the e-mail field must contain a value, the maximum characters for that field is 400, the field must be formatted as an e-mail address and as a final rule, the value must be unique in the users table within the database.

The validation class was used when the end user has the ability to input anything into the database.

To run the validation class, an ‘if’ statement was made; if the validation passed, it continued on with the rest of the registration process. If the validation failed however, the user would be redirected back to the registration page, with all the validation errors shown beside the particular fields. To improve the user experience during this process, the fields that passed the validation test would automatically be filled in when the user was redirected back to the registration page, helping to avoid any re-entering of information which can be both time-consuming and frustrating.

The next stage was to insert all the data from the registration form into the user table. In a case were a hacker has got access to the database and can view all the records, the password field has been protected by using Laravel’s inbuilt

'Hash class'; this will secure the password by using the Bcrypt package, which will generate a random text input that will be stored in the database so that no one can view the actual password.

```
// Activation Code
$code = str_random(60);

$user = User::create(array(
    'first_name' => $first_name,
    'last_name' => $last_name,
    'location' => $location,
    'email' => $email,
    'username' => $username,
    'password' => Hash::make($password),
    'code' => $code,
    'active' => 0,
    'tag' => $tag,
    'occupation' => $occupation,
    'birthday' => $birthday,
    'portfolio' => $portfolio,
    'avatar' => $filename
));
```

Figure 4.2 – Inserting data into the database

The next problem was discovered whilst reviewing the security aspect of the registration process; any user could create an account with anyone's email address. It was decided that the user should have to verify the inputted e-mail address so as to confirm that it does, in fact, belong to that user. This will also aid in reducing the number of spam accounts that may be created, consequently resulting in improved search times in the database.

As is again displayed in Fig 4.2, the code field has a random collection of 60 characters and the active field is set at '0'. This details how the email is generated. The 'Mail::send method' is used to send the e-mail; the argument that is passed through this method is the name of the view, which will generate the e-mail body. The second is an array, which includes all the data that is needed to pass to the view. This includes the link, which links to the account-active model and the code field. The third and final argument specifies where the email will be sent, includes the username variable and defines the subject of the e-mail. Once this process has been completed, the user is directed to the homepage with a message confirming that the account has been created and an e-mail has been sent.

```
if($user) {

    Mail::send('emails.auth.activate',
        array('link' => URL::route('account-activate', $code), 'username' => $username),
        function($message) use ($user) {
        $message->to($user->email, $user->username)->subject('Activate your huddlepix Account');
    });

    return Redirect::route('home')
        ->with('global', 'Your account has been create please check your email to activate your account.');
}
```

Figure 4.3 – email the verification link

Upon the user receiving the email and clicking the link, the function 'getActiviated' will begin- this function is also visible in Fig 4.4. The function will require the code variable, which was passed though when the e-mail was sent. The code variable will be able to identify the user from the database and for added security, the query will ensure that the active field has the value '0' which has been set already (see Fig 4.2). The next step in this process is to update the users active state; this entails the active field to be set to '1' and the code field has no value. The final step is to save the updated fields and return the user to the homepage with a message stating that the account has been activated. If the code variable cannot identify the user, the user is redirected to the homepage with a message stating, 'we could not activate your account, try again later'.

```
public function getActivate($code){

    $user = User::where('code', '=', $code)->where('active', '=', 0);

    if($user->count()) {
        $user = $user->first();

        // Update user to active state
        $user->active    = 1;
        $user->code      = '';

        if($user->save()) {
            return Redirect::route('home')->with('global', 'We have activated your account, now sign in');
        }
    }
}
```

Figure 4.4 – activate user account

4.4.2 Sign In

The sign in facility took advantage of the Laravel's inbuilt authentication system; before the code got to this point, the form was validated similar to that in Fig 4.1. After the first prototype was built, the users requested that a 'remember me' function would improve the overall user experience. The Laravel authentication system can deal with this request very simply (see Fig 4.5). A

variable was created and has checked to see if the 'remember me' checkbox has been selected. After this process, the remember variable will either have a value of true or false.

```
$remember = (Input::has('remember')) ? true : false;


$auth = Auth::attempt(array(
    'username'   => Input::get('username'),
    'password'   => Input::get('password'),
    'active'     => 1
), $remember);

if($auth){
    return Redirect::action('ProfileController@user', array($username));
} else {
    return Redirect::route('account-sign-in')
        ->with('global', 'Email / Password incorrect or account not activated');
}
```

Figure 4.5 – code to allow users to sign in

To sign the user in, the 'Auth::attempt' method was utilised; this checks to see if the inputted username will match any of the usernames held within the database. From here, the password is matched and it will then confirm that the active field has the value 1; this is to ensure that the user has been verified. Laravel will automatically add the cookie to the selected browser, meaning that the user will not have to sign in for the foreseeable future.

If this has all been successful, the code is then redirected to the Profile Controller and the user function. If not, the user will be redirected back to the sign in page.

4.4.5 Sign Out

When the user clicks the sign out button, it will redirect them to the 'getSignOut' function (see Fig 4.6). This again uses the Laravel authentication system and destroys the session, before redirecting the user to the homepage.

```
public function getSignOut() {
    Auth::logout();
    return Redirect::route('home');
}
```

Figure 4.6 – sign out function

4.5 User Profile

Once the user signs into the website they are directed to the user profile page.

```
$user = User::where('username', '=', $username);
```

Figure 4.7 – Identify the user

A user variable has been created and it uses the username variable, which was passed to this function upon the user signing in (see Fig 4.7). It then searches the user table in the database and can now access all of the users data. Laravel's Blade Templating Engine has been used to display the data in the user profile view; an example of this is illustrated by Fig 4.8.

```
<p><span class="col-md-4">First Name </span>{{ $user->first_name }}</p>
<p><span class="col-md-4">Last Name </span>{{ $user->last_name }}</p>
<p><span class="col-md-4">Username </span>{{ $user->username }}</p>
<p><span class="col-md-4">Email </span>{{ $user->email }}</p>
```

Figure 4.8 – Blade Templating engine, which is display the user information

The second section of the user profile is the job overview; this will display all of the users current jobs that they have posted in a table. Within this table, the user will be able to identify each job with the job title. Additionally, they will be able to view any messages that may have been posted and will be able to view all of the job proposals that they have received for that specific job. To get all of the data required to populate this table, the job and user table needs to be joined; this was done by a simple query (see Fig 4.9). Firstly, the job table needed to be selected and then joined to the user table, making sure that the user_id in both the user and job tables are the same field. Following this, a 'where' statement was added to access all of the jobs relating to that user_id. A second where statement was also added to ensure that the job was still active, and all rows then retrieved which met the criteria.

```
$jobs = DB::table('job')
    ->join('users', 'users.user_id', '=', 'job.user_id')
    ->where('job.user_id', '=', $user_id)
    ->where('job.temp_del', '=', '0')
    ->get();
```

Figure 4.9 – Query, which joins the job and users table together

The third and final section of the user profile consists of a summary of all the proposals that the user has sent for other jobs. Within this table, the user will be able to view the Job Title, Proposal Brief, the Proposed Price and a link is provided to view the job. This was similar to the query which is shown in Fig 4.9, however another join was required as the proposal, users and job tables were needed to acquire the data to populate the table.

To display both of these tables, a 'for each loop' was required in the user profile view (see Fig 4.10). Where there is data available, it will keep adding a row.

```
@foreach($proposals as $proposal)
    <tr>
        <td>{{ $proposal->job_title }}</td>
        <td>{{ $proposal->proposal_brief }}</td>
        <td>{{ $proposal->proposal_price }}</td>
        <td>{{ link_to("/jobs/$proposal->job_id", 'view job') }}</td>
    </tr>
@endforeach
```

Figure 4.10 – for each loop to generate all proposals into a table

The last two sections are only viewable by that user; other users will not able to view these sections. This was accomplished by an 'if' statement (see Fig 4.11).

```
@if(Auth::user()->user_id === $user->user_id)
```

Figure 4.11 – Identifies if the authenticated user is the user that created the job

4.5 Edit User Profile

The user has the ability to edit some of their personal information, and as with all the other previous forms, this form has been validated; an example of this code is noticeable in Fig 4.1. When all the data has successfully passed the validation, it will then create variables and add the values from the form. The next stage is to update the database; the users table is selected where the user_id is equal to the user_id variable, then the update is stated and all the fields that can be

updated are matched up to the variables that were previously created (see Fig 4.12).

```
DB::table('users')
    ->where('user_id', $user_id)
    ->update(array(
        'email' => $email,
        'password' => Hash::make($password),
        'occupation' => $occupation,
        'location' => $location,
        'tag' => $tag,
        'portfolio' => $portfolio

));
```

Figure 4.12 – Update user information in the database

4.6 View all Jobs

The view all jobs web page was created by using data tables, however all the data was gathered by one query (see Fig 4.13). This query selected all the jobs from the database that the temp_del field equalled 0. When a job is no longer active, the temp_del value is changed to 1.

```
$jobs = Job::where('temp_del', '=', 0)->get();

return View::make('jobs.index', compact('jobs'));
```

Figure 4.13 – query to select all active jobs

All the jobs data is passed onto the index file in the jobs view folder; the 'foreach' loop shown in Fig 4.14 is located within a div with a class called 'adv-table'. This is initialised within the jQuery and generates the functionality of the table such as pagination, search and the number of entries shown. The code to take the data that has been passed to the view by the controller is generated the same as before. The link to view job will pass the job_id variable which is required for the show function, this will be explained further in the next paragraph.

```
@foreach($jobs as $job)
    <tr>
        <td>{{ $job->job_title }}</td>
        <td>{{ $job->job_location }}</td>
        <td>{{ $job->job_deadline }}</td>
        <td>{{ $job->job_budget }}</td>
        <td>{{ link_to("/jobs/{$job->job_id}", 'view job') }}</td>
    </tr>
@endforeach
```

Figure 4.14 – For each loop to display information about the job

4.7 Create a Job

Users must fill out a form to create a job, and this form will be validated similar to Fig 4.1; an 'if' statement is created and once the data has passed the validation, it will be able to move to the 'else' statement- the code within the 'else' statement is displayed in Fig 4.15.

The first two lines of this code define the users username and user id, and the next stage is to create variables for all the fields in the form and assign them to the relevant variable. All of this data will then be inserted into the database, so all the fields are defined and will be assigned to that variable. The last stage is to redirect the user to the user function located in the ProfileController- the username variable is also passed to this controller.

```
$username = Auth::user()->username;

$user_id = Auth::user()->user_id;

$job_title = Input::get('job_title');
$job_brief = Input::get('job_brief');
$job_budget = Input::get('job_budget');
$job_location = Input::get('job_location');

Job::create(array(
    'user_id' => $user_id,
    'job_title' => $job_title,
    'job_brief' => $job_brief,
    'job_budget' => $job_budget,
    'job_location' => $job_location
));

return Redirect::action('ProfileController@user', array($username));
```

Figure 4.15 – Create a job in the database

4.8 Show Specific Job

The show specific job is made up with a number of different functions; the first function will gather all the job information, the number of proposals which have been sent for that specific job and finally, all of the messages which have been sent for that job (see Fig 4.16).

The job variable contains a query which identifies what that specific job is; this query is made possible when the user clicks to view the job, and the job_id

variable is passed through to this function. The query will get the first row where the job_id variable equals the job_id in the database.

```
$job = Job::whereJob_id($job_id)->first();

$proposalCount = Proposal::where('job_id', '=', $job_id)->count();

$discussions = DB::table('discussion')
    ->join('users', 'users.user_id', '=', 'discussion.user_id')
    ->where('job_id', '=', $job_id)
    ->orderBy('date')
    ->get();
```

Figure 4.16 – first function, which is used to generate the specific job page

The proposal count is generated by another query; it targets the proposal table and counts the number of times the job_id appears on that table.

The last query will retrieve all the messages that have been posted by various users for that specific job. This requires data from two tables, so a join is required between both the discussion and the users tables. As with all the other previous joins, the foreign key is identified and the query informed that the user_id in both tables is the same field. The job_id variable is used to identify the specific job and it is ordered by the date that the message was created.

The second and third function will allow the users to send a proposal and a message for that specific job. It uses similar code to the ‘create a job’ function (see Fig 4.15). Both forms will go through validation, and once they pass this validation they can be successfully stored in the database.

4.9 Problems encountered

One of the challenges faced during the implementation stage occurred when allowing the user to upload an avatar during the registration process. The user will be able to select an image, which will be displayed on their user profile page. The first thing that needed to be considered was whether or not the image should be stored in the database or the file directory location. It was decided that the file location should be stored; the main reason was to keep the

database as small as possible, which in turn will also work to improve the whole performance of the website.

The first step was to create a folder in the img folder within the public directory; a destination variable will contain the link to this folder. The second variable is the filename; this starts off by adding a random 6 characters to the file name that was uploaded. The final step is to upload the file to the location, which was defined in the destination variable. The next step, which is not shown in Fig 4.17, is to add the filename variable to the avatar field in the database.

```
$destination = public_path().'/img/uploads';
$filename = str_random(6) . '_' . $avatar->getClientOriginalName();

$upload_success = Input::file('avatar')->move($destination, $filename);
```

Figure 4.17 – Upload a file

Chapter 5 - Testing

5.1 Introduction

This chapter will examine the strategy that was carried out while testing Huddlepix. The results of the testing will also be discussed.

5.2 Testing Approach Selection

Testing has been on-going throughout the development of the Huddlepix website. Now that the website meets the user requirements however, and the design has been refined to ensure a quality user experience, the last step in guaranteeing a quality user experience is to ensure that all the functions work as expected; final testing must be carried out. The result of a good testing model will ensure that the website, 'doesn't break, works efficiently and delights users', whilst helping to build 'a foundation of trust between you and your customers' (Ghoshal, 2012).

It was decided that the Integration testing method would best suit Huddlepix; Integration testing is that in which software components, hardware components, or both are combined and tested to evaluate the interaction between them (Williams, 2006). This form of testing includes white and black box testing, and the developer carries out the test. The testing phase has been made simpler due to the fact that users have already used several prototypes and the errors have been already dealt with.

5.3 Testing Process

The test results will be recorded in numerous tables, each table representing a section of the website. Each test case will have the following headers:

- Test ID: A unique identifier for each test case.

- Description: A clear description of the action that will be performed; this will be specific so that no further instructions are required.
- Expected Results: The anticipated correct result, which should happen upon this action being performed.
- Actual Results: The actual result of the action that was carried out; if the action produces the correct result, a simple 'pass' will be sufficient.

5.4 Test Results

The purpose of the first set of testing was to examine the functionality of the website. This was done by noting the expected results of an action and then recording the actual results; the full test plan can be found in Appendix G.

5.4.1 Sign In

The first feature tested was the ability for a user to sign in to the website, mainly so as to ensure that users could not sign into the website with invalid details, or with a non-verified account. It was also equally important that users with the correct details and a verified account could sign into the website with no hassle or complications. Table 5.1 shows an example of the test that was carried out to confirm that the sign in facility met the user's expectations.

Test ID	Description	Expected Results	Pass / Fail
H1	Click on the sign in button.	A model should appear with the sign in form.	PASS
H2	Fill in the correct username and password and sign in without selecting the remember me button.	The website will sign the user in, and be directed to the user profile page and no cookies where created.	PASS
H3	Fill in the correct username and password and sign in and select the remember me button.	The website will sign the user in, and be directed to the user profile page and cookies will be stored on the computer.	PASS

Table 5.1 – Summary of the Sign In Test table

5.4.2 Registration

The registration process was the next feature to be tested in order to make certain that all the data that processed by the user is valid, and can be successfully added to the database. Testing for the registration would also certify that each individual email and username is unique within the database. Table 5.2 shows an example of the test that was used to check that users can successfully add an account to the website.

Test ID	Description	Expected Results	Pass / Fail
H9	Fill the form in correctly.	A successful message will appear and an email verification link will be sent.	PASS
H10	Click the verification link.	The link will direct the user to the homepage and a message will appear stating that the account has been verified.	PASS
H11	Click register without filling in any data.	Error messages will be displayed beside each field stating that, that field is required.	PASS

Table 5.2 – Summary of Registration Test table

An example of the expected results for H10 is shown in Fig 5.1; it clearly states that each field is required to be filled in order to continue the registration process.

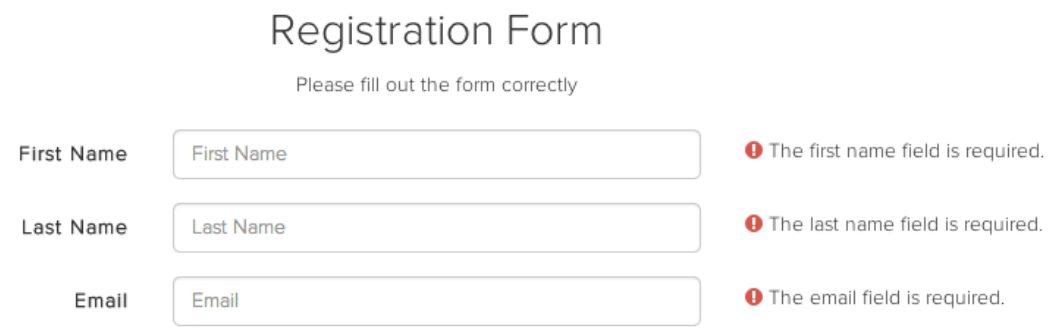


Figure 5.1 – Validation for Registration Form

5.4.3 Edit User Profile

In determining whether the user can successfully update their user profile, the next feature that needed testing was the edit profile feature. It was necessary to

confirm that not all fields can be changed, such as the 'date of birth' field; the reason for this is that the users birthday will never change. In contrast to this, one detail which users will have to change every few months is their password; the user will have to change their current password and verify the new password again. Table 5.3 shows an example of the test that was carried out to ensure that users can successfully edit their profiles.

Test ID	Description	Expected Results	Pass / Fail
H17	Fill Form in correctly.	The user profile will be updated.	PASS
H18	Click update user without filling in any data.	Error messages will be displayed beside each field stating that, that field is required.	PASS
H19	Enter an email address that has already been registered.	Error message states - The email has already been taken.	PASS

Table 5.3 – Summary of Edit User Profile Test table

5.4.4 Other Tests

Several other features of the website were tested so as to make sure that it was fully functional before the main release. This included the search facility located on the view all jobs page; users must be able to post a job, send a proposal and exchange messages.

5.5 System Test

The next part of the testing phase involved checking that the system worked as expected on various devices and legacy browsers. Table 5.4 shows an example of the system test that was carried out.

Test ID	Description	Expected Results	Pass / Fail
ST1	Website should work on all browsers.	Website should offer a similar experience across various browsers.	PASS
ST2	Website should adapt to various screen sizes.	Website layout should change to suit the screen size.	PASS
ST3	Website should be viewable on portable devices.	Website should be work on mobile devices	PASS

Table 5.4 – Summary of System Test Table

It was decided during the planning stage that the website would cater for Internet Explorer 9 and above. The bootstrap framework helped to ensure that the user would get a similar experience on all the browsers. The website was also tested on a various range of phones; the website worked as expected on all the phones. Finally, the website was validated using the W3C validator, which identified a number of mistakes; these have since been corrected and the website is now validated.

5.6 Security Test

Huddlepix stores users information and therefore security issues must be carefully taken into consideration, for instance, only authenticated users will have access to certain data such as the portfolio link. If it was the case that a hacker managed to gain access to the database, they would be unable to use any passwords that are stored there as they have been encrypted.

To test to see if the encryption had worked, a user was created and the password was set to ‘password’. The next step was to view the database and check to see if the password field did not have the value ‘password’. Upon examining the password field, it did not contain the value ‘password’, as was anticipated. The value can be seen in Fig 5.2, and this clarified that the password encryption had worked successfully.

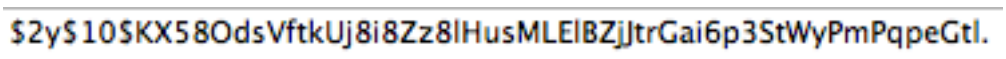


Figure 5.2 – Password value which is stored in the database

5.7 Summary of Test Results

As testing had been regularly carried out throughout the implementation stage, there were fortunately no failed tests at the functional testing stage, as with the security test. The dominant issue that arose in the testing phase was during the system testing; specifically, the legacy browsers.

Chapter 6 - Evaluation

6.1 Introduction

This chapter aims to evaluate the following; users experience of the website, the methodology used and the user requirements being successfully met by the final website.

6.2 Evaluate Test/Survey Results

A questionnaire was designed and used to retrieve the users opinions on and evaluations of the completed website. Boulton (2012) points out that, 'questionnaires can create an overview of the project, how users use the site and their opinions of it, with an idea of which parts of this picture are more important than others'. The questions were short and simple, with the intention that it would take less than a minute to complete. Those asked to complete the questionnaire ranged from web developers and designers to people with less knowledge of and experience in using computers and the Internet. A sample questionnaire can be viewed in Appendix H.

While they completed the questionnaire, those involved were also under observation; this was similar to the people chosen to trial the prototypes in the implementation phase. Any difficulties encountered by those surveyed were noted and will be taken under consideration for the future development of Huddlepix.

6.3 Evaluate Project Outcomes

The questionnaire was broken down into three sections; design, functionality of the website and overall satisfaction, with some additional space provided for further comments. There was a total of ten questions, each of which would rate a different aspect of the website. The answers would range from 1 to 5; with 5 being 'excellent'.

The design section asked people to rate the user interface and layout, the visual consistency of the website, the relevancy and readability of the content and lastly, how easy the navigation was to understand. The functionality section questioned whether or not the website was easy to use, if the website worked as expected and if any security flaws existed. The last main section was to gauge the users overall satisfaction of the website and whether or not they would recommend Huddlepix to a friend. Additional space was provided on the questionnaire to allow those who wished to comment further on Huddlepix the chance to do so.

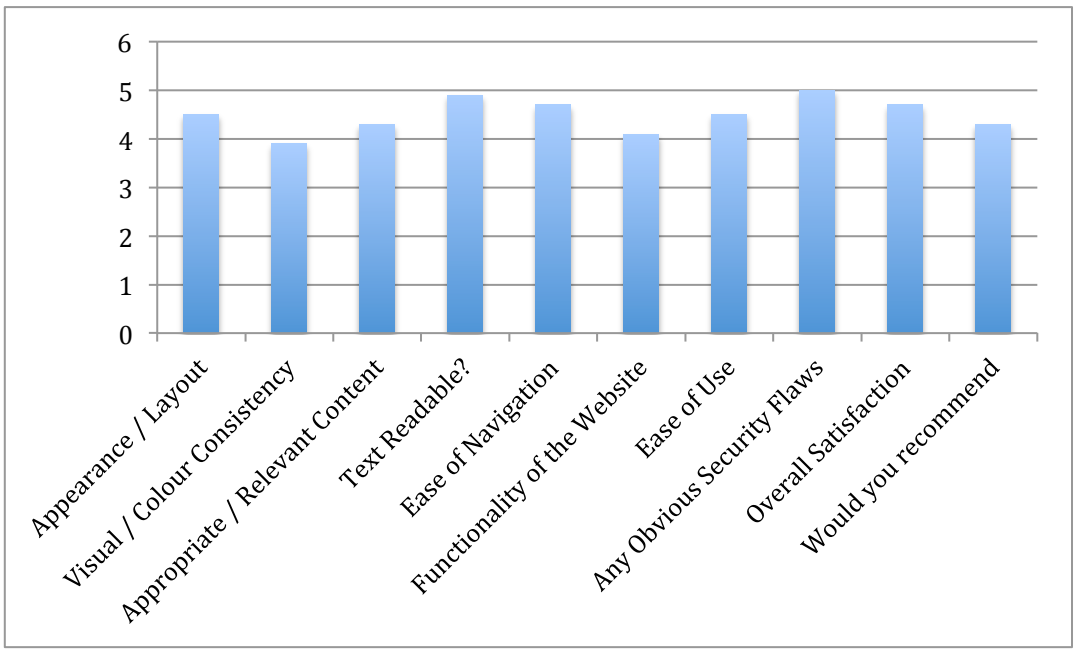


Figure 6.1 – Average results of the questionnaire

The results of the questionnaire were documented, recorded and then illustrated by two different graphs. The first graph displayed the results in the form of a summary of all of the questions and the average rating received. This graph can be seen in Fig 6.1. The second graph was a pie chart which showed the number of people that would recommend Huddlepix to other people (see Fig 6.2). It was felt that if they answered highly in this question, it could be assumed that they too, would use the service provided by Huddlepix.

Figure 6.2 – Pie Chart to illustrate if users would recommend Huddlepix to a friend.

6.4 User Observations

As mentioned previously, users were observed whilst completing the questionnaire, and this proved just as valuable as the questionnaire itself. The users did not have to ask how to use the website or any feature of the website; it was determined that the design of the website made this possible. This was confirmed verbally by users when, after filling out the questionnaire, they were asked on why they had not asked any questions on using Huddlepix. The overall response was that it was clearly laid out and this made it easy for them to do what they needed and wanted. Some users suggested certain updates and changes that might improve the service in the future; this was taken from the additional comment section. These are discussed towards the end of this chapter.

A quick summary of the positive points made during the observations is as follows:

- How the colours and images worked well together
- The minimalistic style of the website design
- The usefulness of the feature which allows users to ask additional questions about a specific job
- The simplicity of the search facility when looking at all the jobs
- The user tag and avatar; this made users feel that they could show their personality which may help them get their proposals accepted

A quick summary of the negative points made during the observations is as follows:

- Introduce a rating system to allow the community to grow and make it easier to decide which proposal to accept.
- Allow users to complete the whole process on the website, for example: make payments on the website, instead of having to do this by themselves via email or the telephone.

6.5 Evaluate the Methodology

The methodology was chosen upon determining that the project was feasible, and the chosen methodology was the prototype model. The prototype model gave the whole project a sense of flexibility; this was down to the fact that many of the requirements could be refined during the implementation phase. Another factor that allowed this project to thrive was the user testing that took place throughout the implementation and design phase. This allowed the system to grow and constantly improved the user experience. One of the drawbacks of this methodology was deciding when to stop developing and move on to testing the full website. The gannt chart helped to define when it was time to move past the implementation phase.

By using the prototype model, the project could be planned very carefully yet simply, and an appropriate amount of time allocated to each phase in order to complete the different tasks.

6.6 Evaluate the Requirements

The requirements for the website were gathered and documented in the requirement analysis, which can be found in Chapter 2. The requirements were modified throughout the design and implementation stages so as to ensure that

the final website would meet the users expectations. There was also a priority level attached to each requirement to display its importance within the website.

All of the user requirements listed were met, for example; requirement 012 was to allow the user to view all the proposals for their job on one page. This can be accessed through the user profile web page. This requirement evolved whilst various designs and layouts of this section were being tried and tested, and it was eventually decided that the information would be more easily viewed using a table. The final design of this table also allowed the user to accept any of the proposals to view easily; this also meets another requirement 011.

The system requirements were also met, however some of the code was required to be modified in order to meet some of those requirements. This was mainly a result of using a framework which had never been used before. Nevertheless, all system requirements were met successfully, proven by the fact that the website is secure and works across multiple platforms.

6.7 Future Development

Upon evaluating the whole project and taking the user observations into consideration, a number of features have came to light which could greatly improve the whole website. These features are described below:

6.7.1 Secure payment and private discussion

User feedback highlighted that they would like the whole job application/acceptance process and transaction to be completed on the website instead of it ending once the user had accepted the proposal. The users would like to see a new screen when they have accepted a proposal, which would allow them to further discuss the job, to upload the images and receive payment for those images.

6.7.2 User Rating System

The user rating system would allow users to review each other after a job has been completed; this system can be seen on various websites across the Internet. This feature would enable the best photographers to get the recognition they deserve, and improve their chances of getting more work from using the Huddlepix website.

6.7.3 AJAX discussion thread

This would improve the discussion section of the job page, more specifically that when a user sends a message, the website is refreshed with their message now on screen. With the use of AJAX, this whole process would be vastly improved; once the user has sent the message, the message would automatically be shown without the web page needing to be refreshed.

Chapter 7 - Conclusion

This report has been compiled to document each stage and process which took place in the development of the Huddlepix website. Each and all of the main elements of this project were examined and discussed; Concept Definition and User Requirements, Design, Implementation and Testing. A significant amount of research and planning was carried out for each of these phases, and proved to be crucial in ensuring the project went as smoothly and successfully as possible. This is not to say, however, that there were not several problems to resolve and obstacles to overcome throughout the project; the very task of imagining, creating and developing the Huddlepix concept was perhaps the biggest and most testing challenge of them all.

Consistent planning, assessment, evaluation and reflection allowed for Huddlepix to grow not only as an idea, but as a brand. Regular user feedback helped to flag up any actual or potential flaws with the site, as well as acknowledging the more positive and well-liked aspects. Frequent and intermittent testing worked to address any minor or major technical issues and confirm that all features were fully functional and accessible. These are only a small number of the requirements and responsibilities recognised and undertaken whilst completing the project, yet they were some of the most essential in making Huddlepix a success.

This project not only lead to the growth and development of Huddlepix however; there were many benefits and advantages at both a personal, and professional level. Learning was enhanced on a grand scale, as well as often tested and challenged whilst trying to grasp new ideas and understand unfamiliar frameworks. Organisation skills were put into practice in the form of time-management; with deadlines to be met, prioritisation became key not only in relation to the project, but also in maintaining a balance between work and lifestyle. Communication skills were greatly enriched through liaising with mentors, tutors and peers to gain feedback on the website, and at a general

level. On a more personal level though, this project built strength of character, in that it called for a sense of independence and self-belief from beginning to end. Personal goals needed to be achieved, which took confidence, commitment and determination; all of which proved to be just as important in the final outcome of this project as the more technical, work-based requirements. Overall, this project has been one in which many skills and qualities have been both improved and acquired, and though sometimes stressful, has instilled a great sense of achievement and pride in the Huddlepix brand.

Chapter 8 – References

Do You Know Why Validating Your Code Is Important? - Vanseo Design. Steven Bradley, 2011 [ONLINE] Available at: http://www.vanseodesign.com/web-design/validating-code/. [Accessed 18 April 2014].

Page Speed Matters! Why You Need to Improve Yours Today. Benjamin Spiegel, 2013. [ONLINE] Available at: http://marketingland.com/page-speed-matters-why-you-need-to-improve-yours-today-56774. [Accessed 18 April 2014].

6 Reasons Why a Generic Stock Photo Doesn't Work for Your Website | Belkis Marketing. Belkis Cardona-Rivera , 2012. [ONLINE] Available at: http://belkismarketing.com/6-reasons-why-a-generic-stock-photo-doesnt-work-for-your-website/. [Accessed 18 April 2014].

Prevent SQL Injection: Tutorial, Cheat Sheet to Avoid Attacks - PHP Example | Veracode. Fergal Glynn, 2013. [ONLINE] Available at: http://www.veracode.com/security/sql-injection. [Accessed 18 April 2014].

Paper Prototyping · An A List Apart Article. 2014. Shawn Medero, 2007. [ONLINE] Available at: http://alistapart.com/article/paperprototyping. [Accessed 18 April 2014].

Effective design principles for web designers: Contrast - TechRepublic. Ryan Boudreaux, 2012. [ONLINE] Available at: http://www.techrepublic.com/blog/web-designer/effective-design-principles-for-web-designers-contrast/#.. [Accessed 18 April 2014].

Laracasts | M-V-Huh? Jeffrey Way, 2013. [ONLINE] Available at: https://laracasts.com/series/laravel-from-scratch/episodes/2. [Accessed 18 April 2014].

What is entity-relationship diagram (ERD or ER diagram)? - Definition from WhatIs.com. Margaret Rouse, 2007. [ONLINE] Available at: http://searchcrm.techtarget.com/definition/entity-relationship-diagram. [Accessed 18 April 2014].

Laravel - The PHP Framework For Web Artisans. Taylor Otwell, 2013. [ONLINE] Available at: http://laravel.com/docs/templates. [Accessed 18 April 2014].

What is Prototype model- advantages, disadvantages and when to use it? ISTQB Exam Certification, 2012. [ONLINE] Available at: http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/. [Accessed 18 April 2014].

A Comprehensive Guide to Testing Web Apps. Ghoshal, 2012. [ONLINE] Available at: http://thenextweb.com/apps/2013/11/28/guide-testing-web-app-steps-approach-testing-get-sessions/. [Accessed 18 April 2014].

Testing Overview and Black-Box Testing Techniques. Laurie Williams, 2006. [ONLINE] Available at: http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf. [Accessed 18 April 2014].

Using Questionnaires for Design Research ◆ 24 ways. Emma Boulton, 2012. [ONLINE] Available at: http://24ways.org/2012/using-questionnaires-for-design-research/. [Accessed 18 April 2014].

Bibliography: Airey, D., 2010. Logo design love. 1st ed. Berkeley, CA: New Riders.

Appendix A - Gannt Chart

Appendix B - Requirements specification

Requirement ID: 001.
Requirement Type: User Requirement.
Description: Product Website.
Rationale: To attract clients to the product and provide information on the service provided.
Source: The frontend developer deemed this feature to be a requirement.
Fit Criteria: The product will have a large customer base and the website will have high visitor numbers.
Priority Level: 3.
Dependencies: There are no dependencies for this requirement.

Requirement ID: 002.
Requirement Type: Functional Requirement.
Description: User Registration.
Rationale: Allow the users to create an account on the website, to enable them to use the service.
Source: The backend developer deemed this feature to be a requirement.
Fit Criteria: This can be determined as being successful when a client can create an account on the website.
Priority Level: 5.
Dependencies: This requirement entails you to have a product website where the sign-up form is located. A database will have to be created to store all the information gathered from this form.

Requirement ID: 003.
Requirement Type: Functional Requirement.
Description: User Sign In/ Sign Out.
Rationale: Allow the user to sign in and out of the website, and give them access to post a job and send proposals.
Source: The frontend developer.

Fit Criteria: When the user can either sign in or out of the website.
Priority Level: 5.
Dependencies: This requirement needs the user to have set up an account on the website using the sign-up form. There will also need to be a database where all relevant information will be logged.

Requirement ID: 004.
Requirement Type: User Requirement.
Description: User Profile Page.
Rationale: This will allow a user to view their own, or other users information, access all the jobs and proposals they may have sent.
Source: The frontend developer.
Fit Criteria: This can be seen to be successful when a can view all their information and view their activity in the website.
Priority Level: 4.
Dependencies: The user will have to have an account set up with the website to enable them to view their profile. There must be a database for all users.

Requirement ID: 005.
Requirement Type: User Requirement.
Description: Edit User Profile Page.
Rationale: This will allow a user to maintain their account and modify when/if necessary.
Source: The frontend developer.
Fit Criteria: This can be seen to be successful when a user can modify their account, and any new information will replace the existing content on the database.
Priority Level: 4.
Dependencies: The user will have to have an account set up with the website to enable them to change their profile. There must be a database for all users.

Requirement ID: 006.
Requirement Type: Operational Requirement.
Description: Create Job.
Rationale: This requirement is important to the overall scope of the website. The reason for this is due to the fact that the jobs will form the main content of the website and will be the primary reason why any user will visit the website; to post a job or to search for a job.
Source: The developer.
Fit Criteria: This can be assessed by the user's ability to create and post a job onto the website. All the jobs posted will have to be stored on the job database and additionally, should be linked to the user database.
Priority Level: 5.
Dependencies: A job database should be set up and automatically updated with every job posted by a user. The customer database will also have to be previously created, as specific jobs will have to be linked to particular users.

Requirement ID: 007.
Requirement Type: Operational Requirement.
Description: Create Job.
Rationale: A user will need to have to the ability to close a job if they no longer require this job to be completed.
Source: The developer.
Fit Criteria: This can be assessed when the user has the ability to close a job, and it will be no longer viewable to any users
Priority Level: 4.
Dependencies: A user must be able to create a job and view the job before they can close it.

Requirement ID: 008.
Requirement Type: System Requirement.
Description: View Jobs.
Rationale: The users will be able to view every specific job they wish.

Source: Developer.
Fit Criteria: This can be deemed successful when a user can view any job that has been posted by users from the database.
Priority Level: 4.
Dependencies: Users will need to be able to post jobs in order to view any jobs.

Requirement ID: 009.
Requirement Type: User Requirement.
Description: Send Messages.
Rationale: The users will be able to post a public message on every specific job page.
Source: Developer.
Fit Criteria: This can be deemed successful when a user can post a message on a specific job, and it is viewable by anyone.
Priority Level: 4.
Dependencies: Users will need to be able to view jobs in order to post a message.

Requirement ID: 010.
Requirement Type: User Requirement.
Description: Send Proposal.
Rationale: The users will be able to send a proposal for every specific job page.
Source: Developer.
Fit Criteria: This can be deemed successful when a user can send a proposal for a specific job, and then can view it on their user profile page.
Priority Level: 5.
Dependencies: Users will need to be able to view jobs in order to send a proposal.

Requirement ID: 011.
Requirement Type: Functional requirement.
Description: Accept Proposal.

Rationale: The user that posted the job may receive many proposals, however when they come across a user to whom they would like to give the job, they will have the option of accepting their proposal. This will automatically remove the job from the job feed.
Source: Developer.
Fit Criteria: The criteria will be met when the job has been agreed upon and accepted by both users, and consequently removed from the job feed and the database.
Priority Level: 5.
Dependencies: This requirement involves the job database removing a specific job from the database, and moving it to a new database containing old jobs that are no longer 'in effect'.

Requirement ID: 012.
Requirement Type: Functional requirement.
Description: View all proposals sent for a job.
Rationale: The user that posted a job will be able to view all the proposals that have been sent for that job in order to decide which one they may pick.
Source: Developer.
Fit Criteria: The criteria will be met when the user can view all the proposals that have been sent for their job.
Priority Level: 3.
Dependencies: Other users must have the ability to send a proposal for specific jobs.

Requirement ID: 013.
Requirement Type: User Requirement.
Description: Cancel account.
Rationale: If you give a user the opportunity to sign-up to a website, they should also have the ability to cancel their account at any time; this requirement will allow them to do so.
Source: Developer.

Fit Criteria: The user will no longer have an account on the website, and all information will no longer be stored on the user database.
Priority Level: 4.
Dependencies: The user will have to be able to remove their information from the user database.

Requirement ID: 014.
Requirement Type: Non-Functional Requirement.
Description: Social media.
Rationale: Incorporating social media will help to increase the number of visits to the website. Furthermore, it will give more coverage to the jobs posted, dependent on if the user chooses to share information from their personal account to their social network profile.
Source: Developer.
Fit Criteria: When there is access to a 'share' button on the job page.
Priority Level: 1.
Dependencies: The user will have to have connected their social media details with the website.

Requirement ID: 015.
Requirement Type: Functional Requirement.
Description: Contact Admin Team.
Rationale: If a user has a query they have the option to contact the admin team.
Source: Developer.
Fit Criteria: When users can contact the admin team.
Priority Level: 2.
Dependencies: The users will have to access the homepage.

Requirement ID: 016.

Requirement Type: Functional Requirement.
Description: Search for a job using keywords.
Rationale: As Huddlepix grows the amount of jobs that will be viewable to users will also increase, a search facility will allow users to refine the job feed to their needs.
Source: Developer.
Fit Criteria: When the user can search the job feed.
Priority Level: 3.
Dependencies: Users must have the ability to post a job.

Requirement ID: 017.
Requirement Type: Functional Requirement.
Description: Report Jobs that breach the terms and conditions.
Rationale: If a user posts a job that may offend any other users, then they will have the ability to report that specific job to the admin team.
Source: Developer.
Fit Criteria: When a user can report a job.
Priority Level: 3.
Dependencies: Users must have the ability to post a job.

Requirement ID: 018.
Requirement Type: System Requirement.
Description: Customer details are securely stored in the database.
Rationale: In the unlikely event of someone gaining access to the database, the data will be encrypted.
Source: Developer.
Fit Criteria: This is to ensure all personal information is protected.
Priority Level: 5.
Dependencies: There must be data within the database.
Requirement ID: 019.

Requirement Type: System Requirement.

Description: Website meets industry standards.

Rationale: This is to ensure that users will receive a high quality service from Huddlepix.

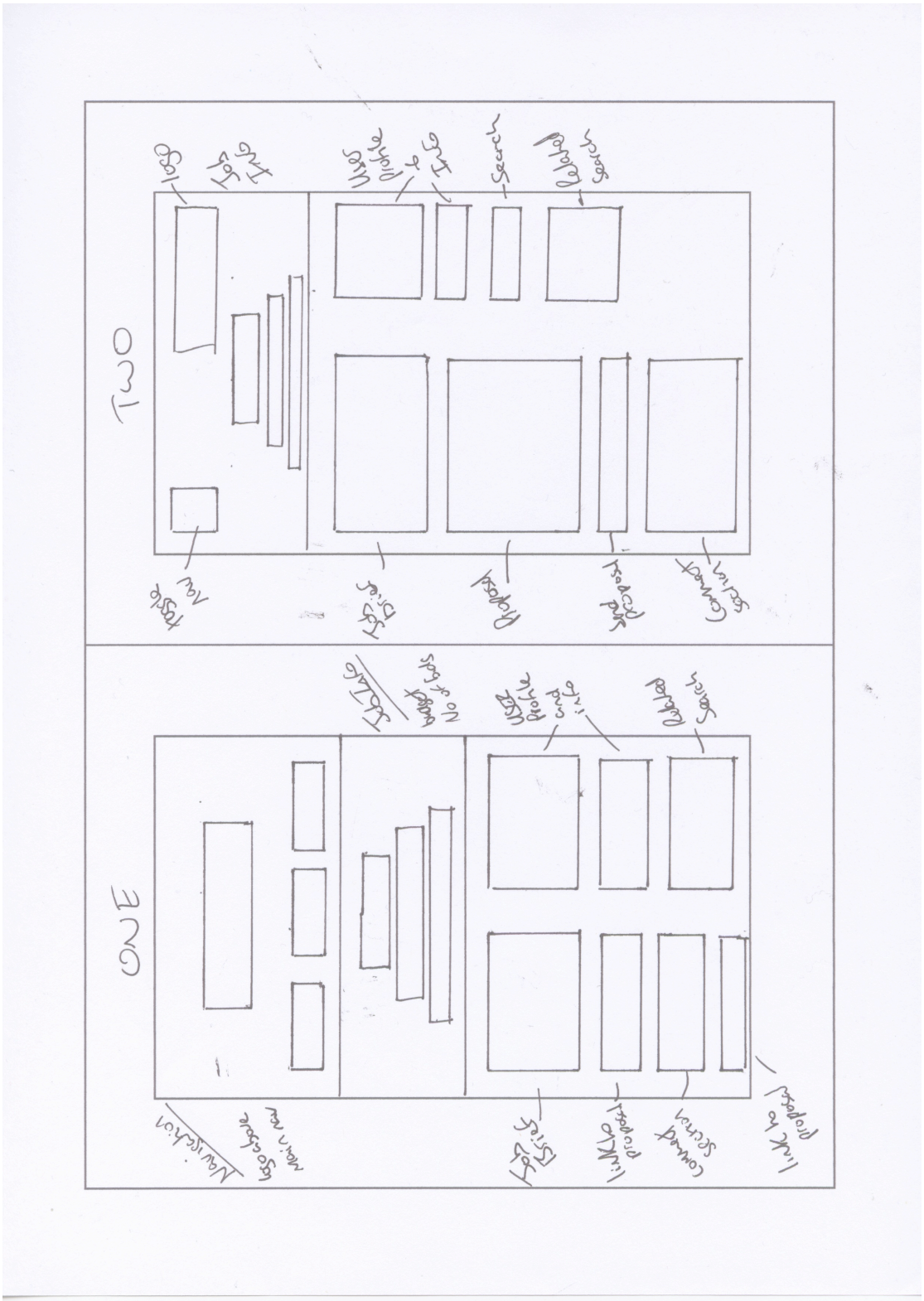
Source: Developer.

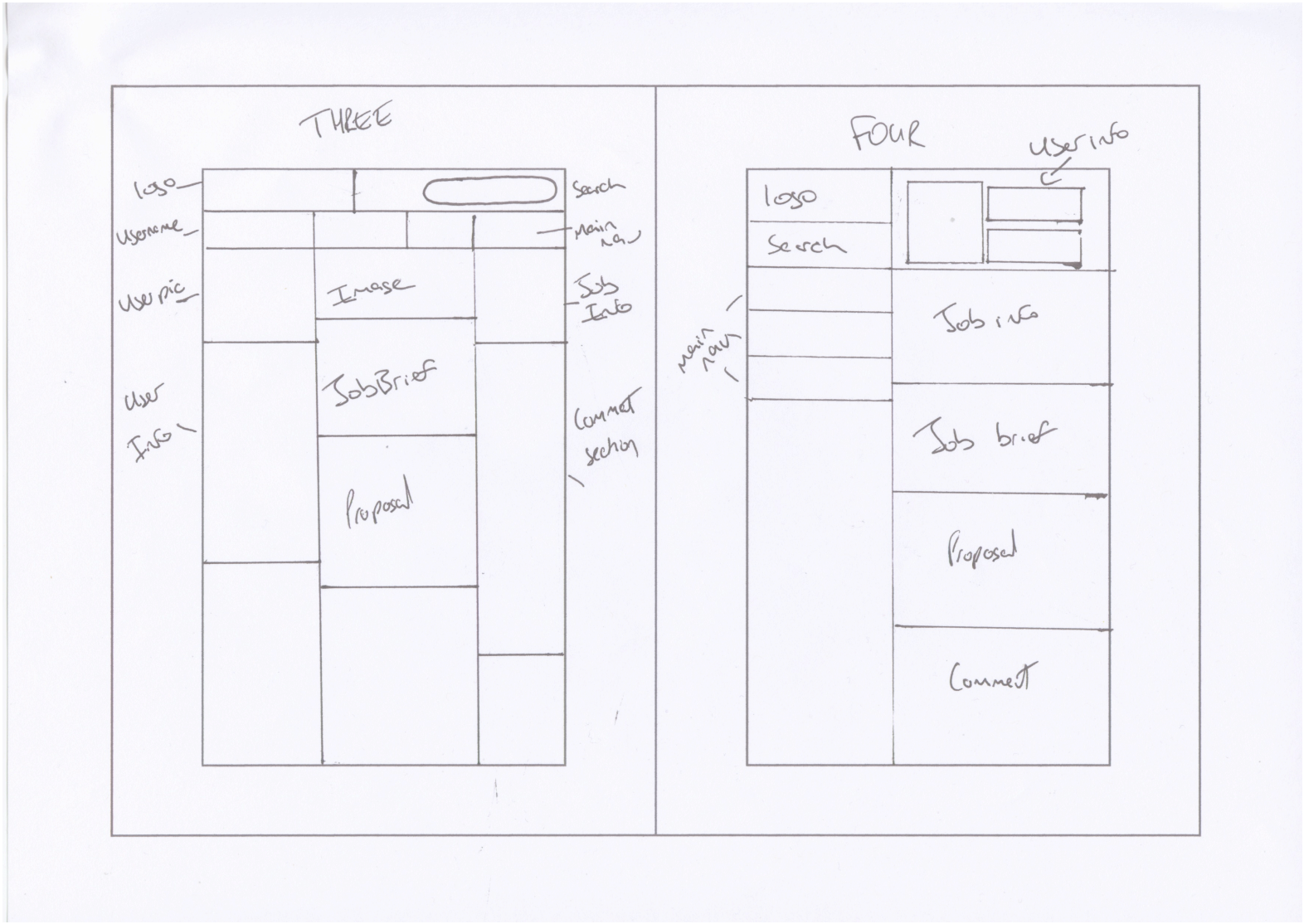
Fit Criteria: When the website has been tested by the W3C validator.

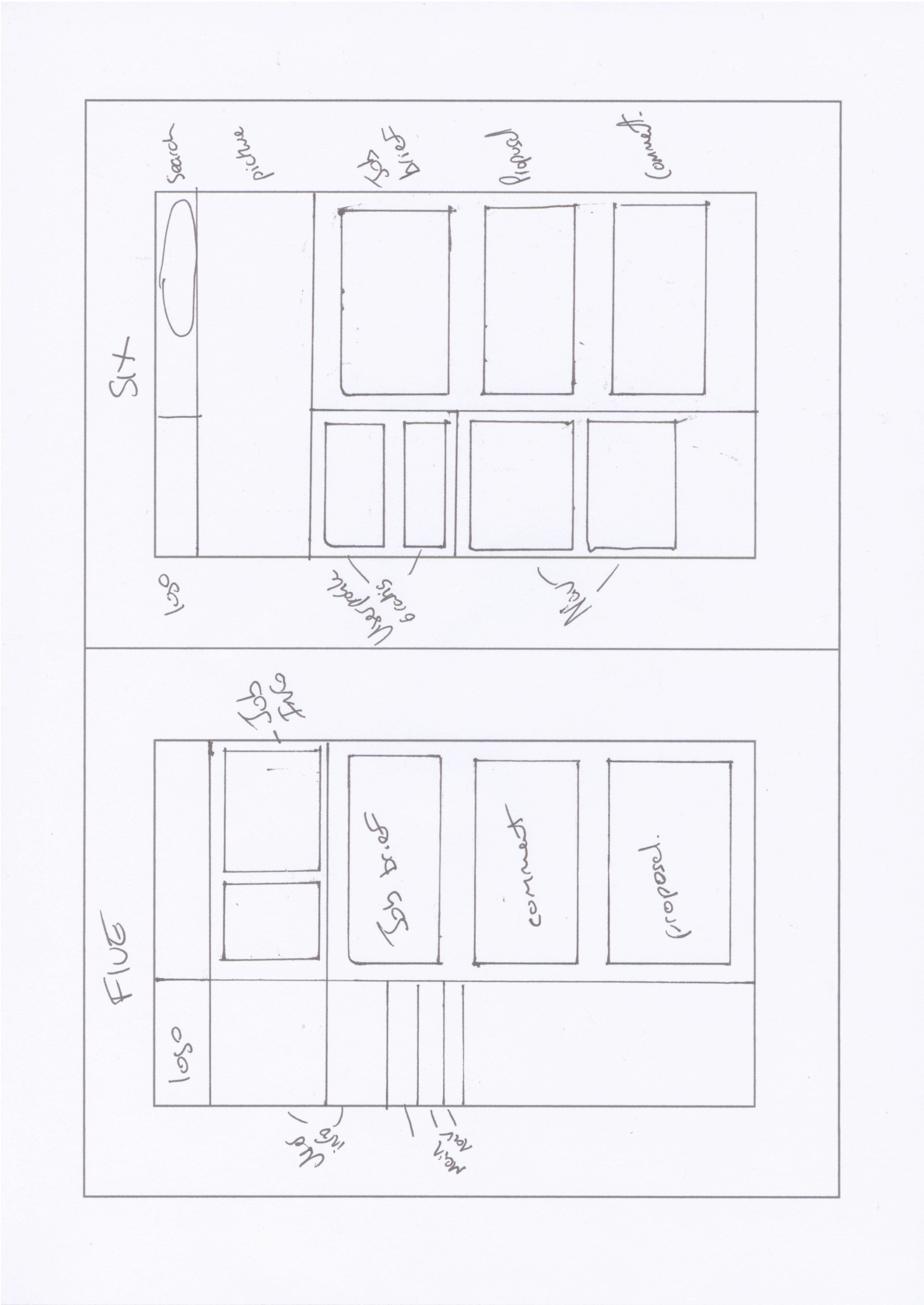
Priority Level: 5.

Dependencies: The website must be completed.

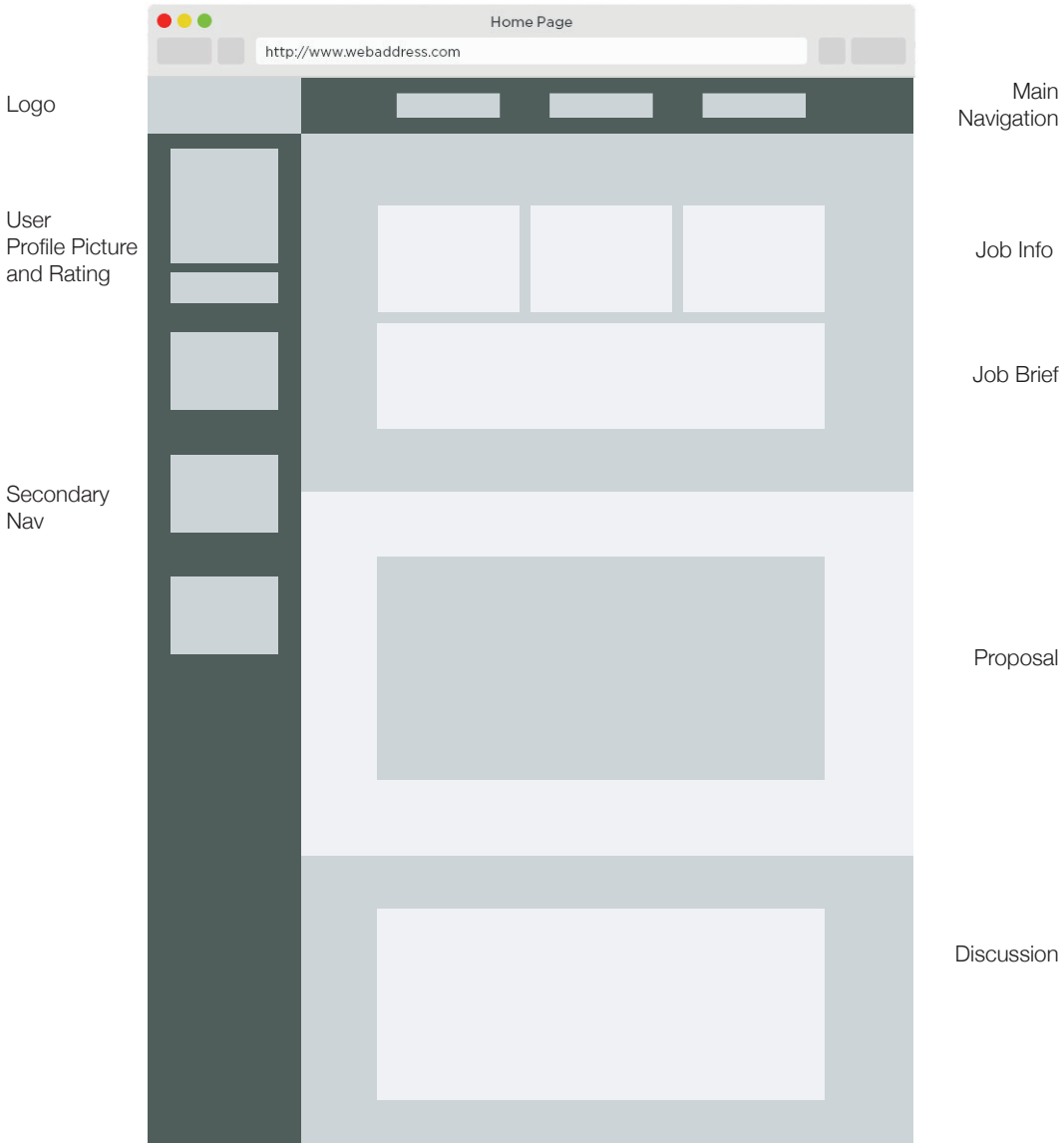
Appendix C - 6 ups




THREE
logo
Search
Username
main nav
User pic
Image
Job Info
Job Brief
User Info
Comment section
Proposal
FOUR
User info
logo
Search
main nav
Job info
Job brief
Proposed
Comment

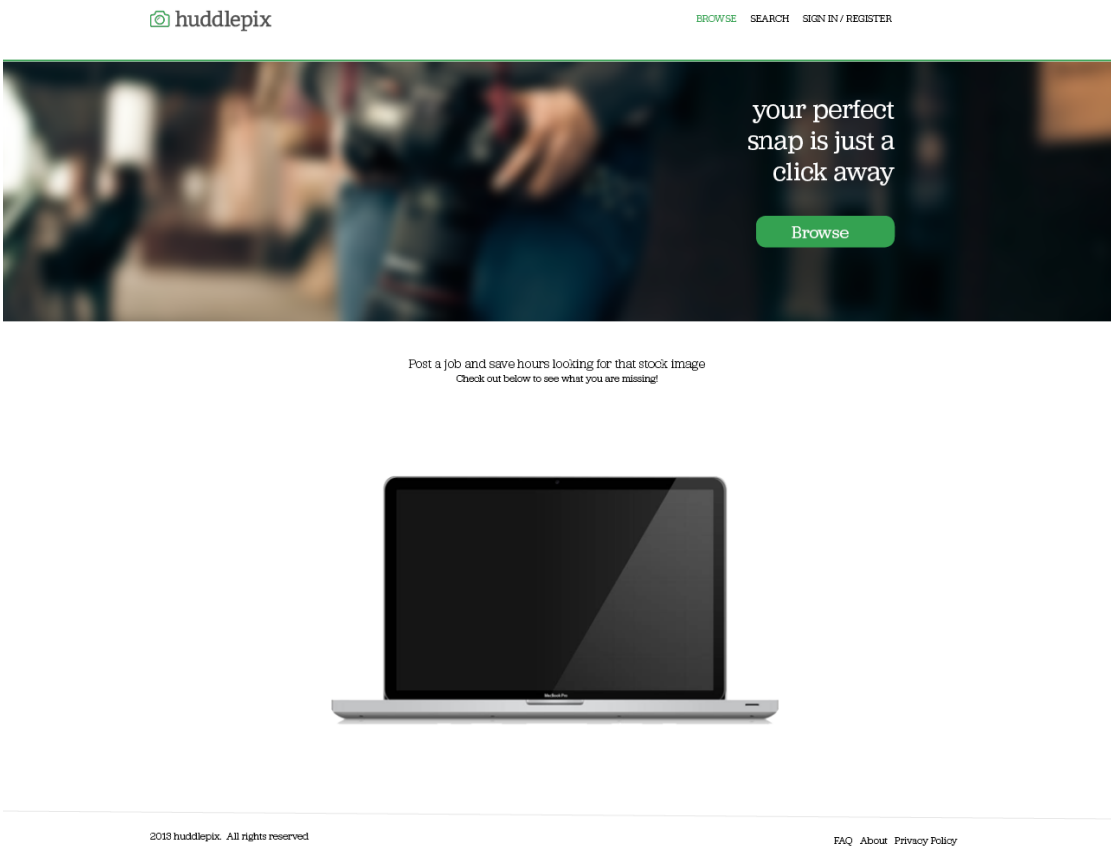

SIX
Search
picture
logo
Nav
FIVE
logo
comment

Appendix D – 1 Up

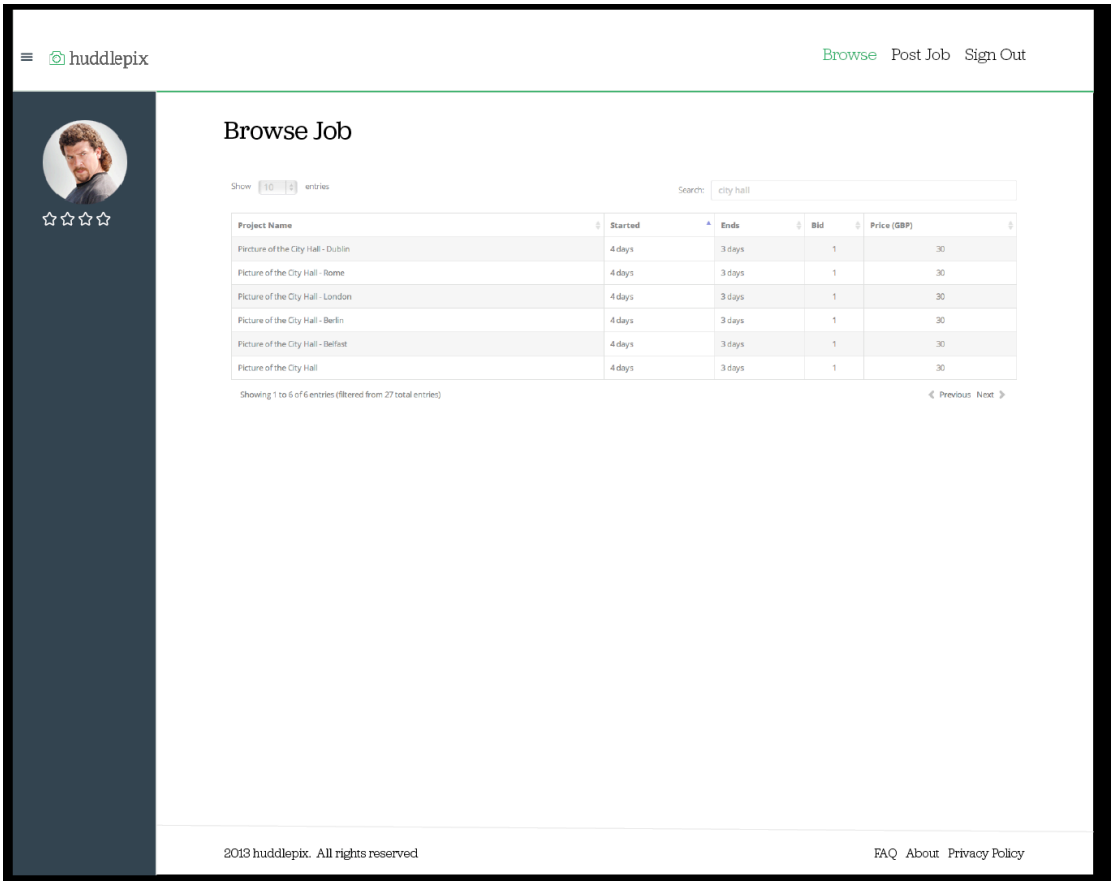


Appendix E – Original Designs

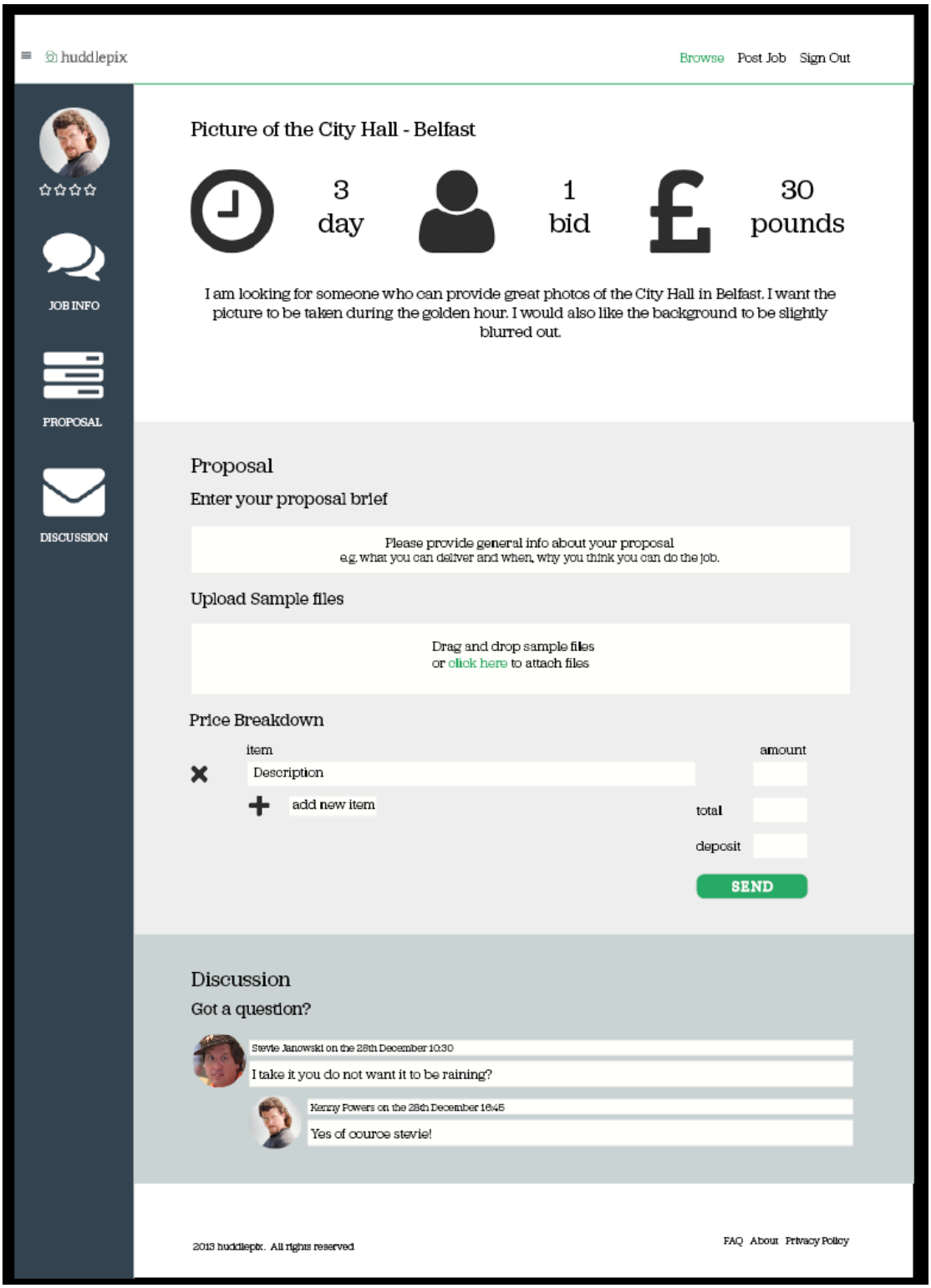
Homepage



Job List

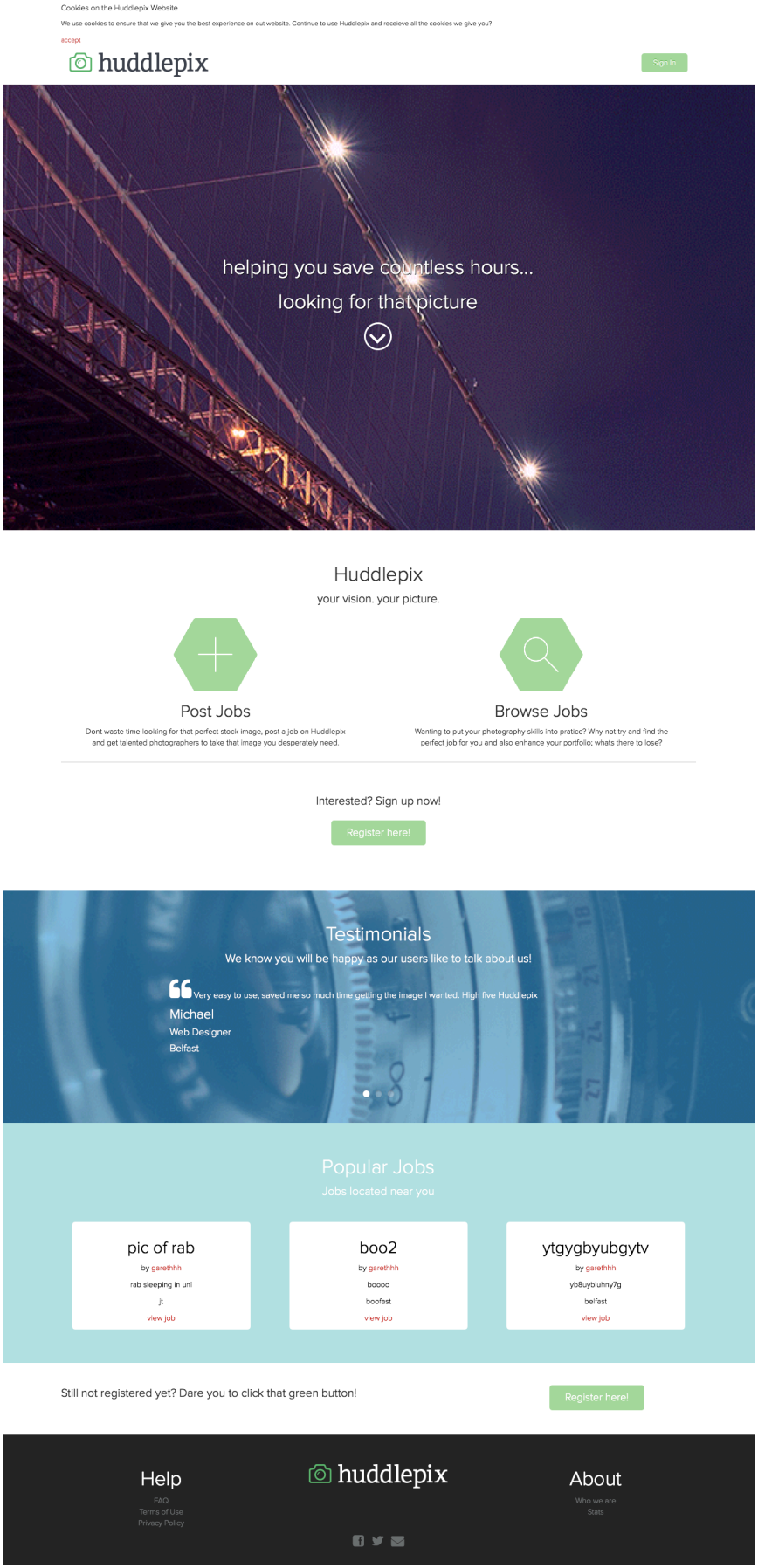

huddlepix
Browse
Post Job
Sign Out
Browse Job
Show 10 entries
Search: city hall
Project Name
Started
Ends
Bid
Price (GBP)
Pircture of the City Hall - Dublin
Picture of the City Hall - Rome
Picture of the City Hall - London
Picture of the City Hall - Berlin
Picture of the City Hall - Belfast
Picture of the City Hall
4 days
3 days
1
30
Showing 1 to 6 of 6 entries (filtered from 27 total entries)
Previous
Next
2013 huddlepix. All rights reserved
FAQ
About
Privacy Policy

Job Page


huddlepix
Browse
Post Job
Sign Out
JOB INFO
PROPOSAL
DISCUSSION
Picture of the City Hall - Belfast
3
day
1
bid
30
pounds
I am looking for someone who can provide great photos of the City Hall in Belfast. I want the picture to be taken during the golden hour. I would also like the background to be slightly blurred out.
Proposal
Enter your proposal brief
Please provide general info about your proposal
e.g. what you can deliver and when, why you think you can do the job.
Upload Sample files
Drag and drop sample files
or click here to attach files
Price Breakdown
item
amount
Description
add new item
total
deposit
SEND
Discussion
Got a question?
Stevie Janowski on the 28th December 10:30
I take it you do not want it to be raining?
Kenny Powers on the 28th December 16:45
Yes of cource stevie!
2013 huddlepix. All rights reserved
FAQ
About
Privacy Policy

Appendix F – Final Designs

Homepage



Registration

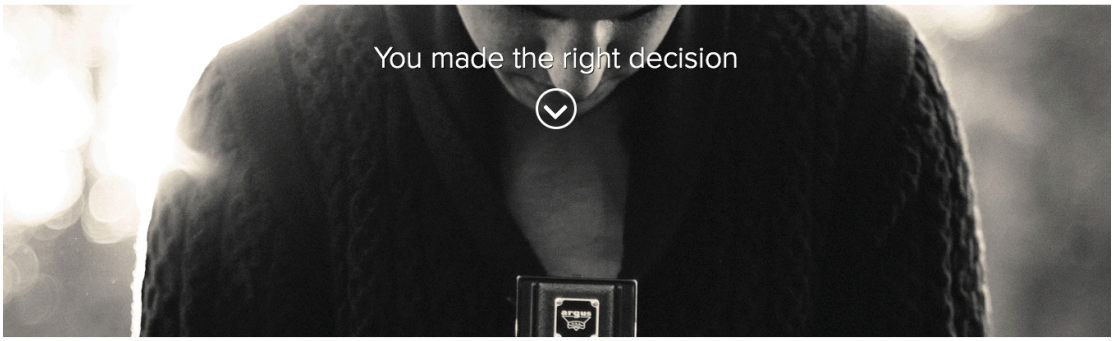
Cookies on the Huddlepix Website

We use cookies to ensure that we give you the best experience on out website. Continue to use Huddlepix and receieve all the cookies we give you?

accept

huddlepix

Sign In



Registration Form

Please fill out the form correctly

First Name	First Name
Last Name	Last Name
Email	Email
Location	Location
Username	Username
Password	Password
Password again	Password Again
Tag	tag
Occupation	Occupation
Portfolio	http://
Birthday	Birthday

for example: 1st January

Avatar: Choose File no file selected

Please upload an image for your account

Register

Help

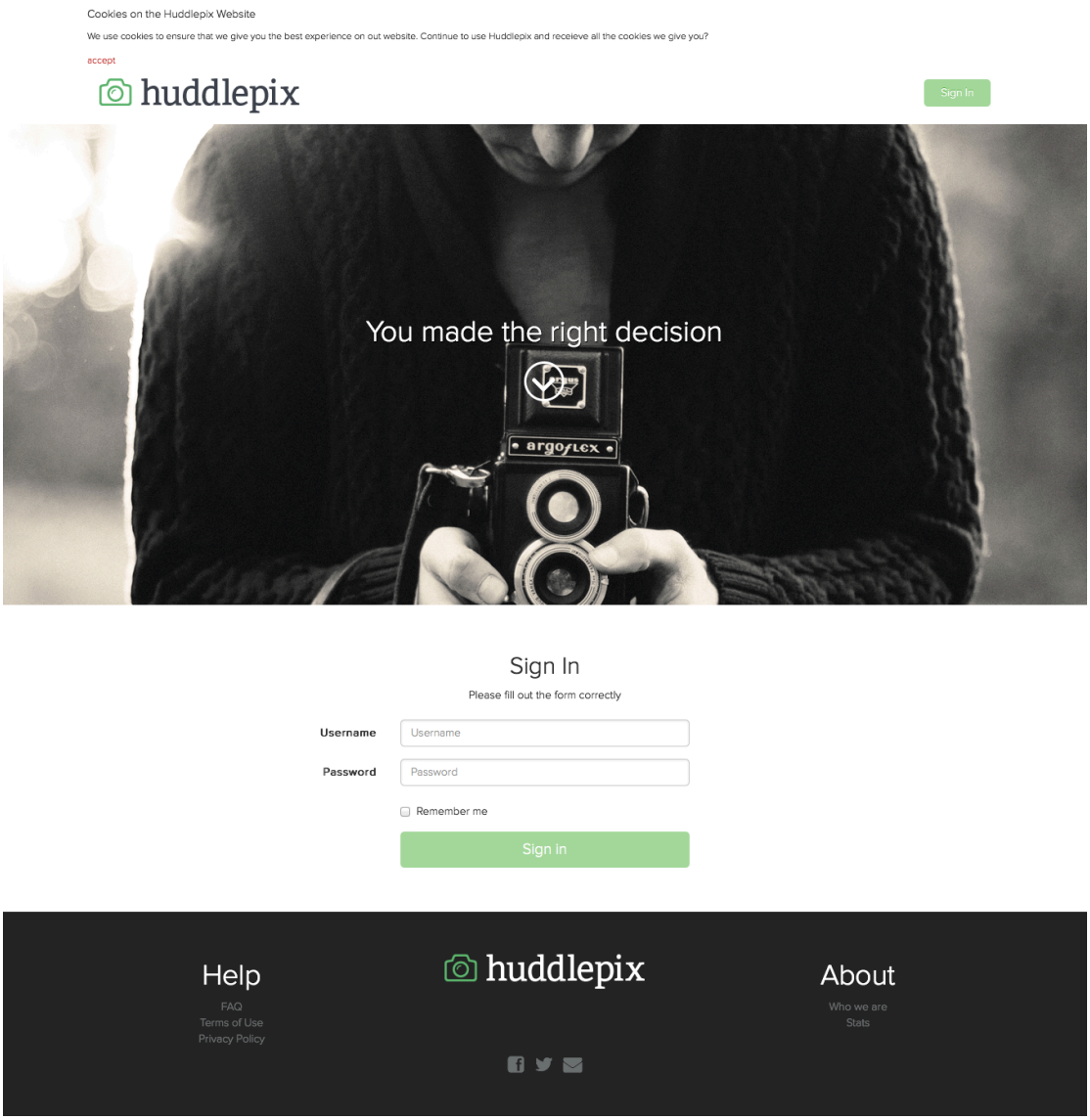
FAQ
Terms of Use
Privacy Policy

huddlepix

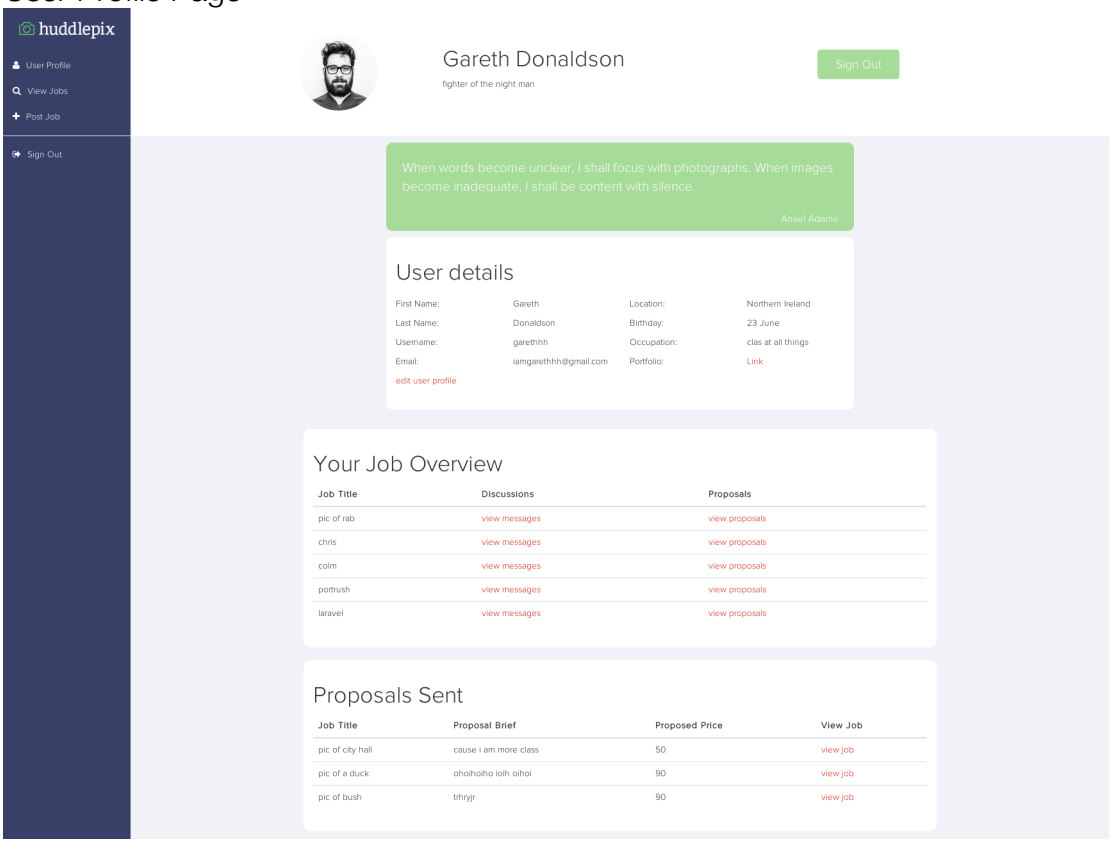
About

Who we are
Stats

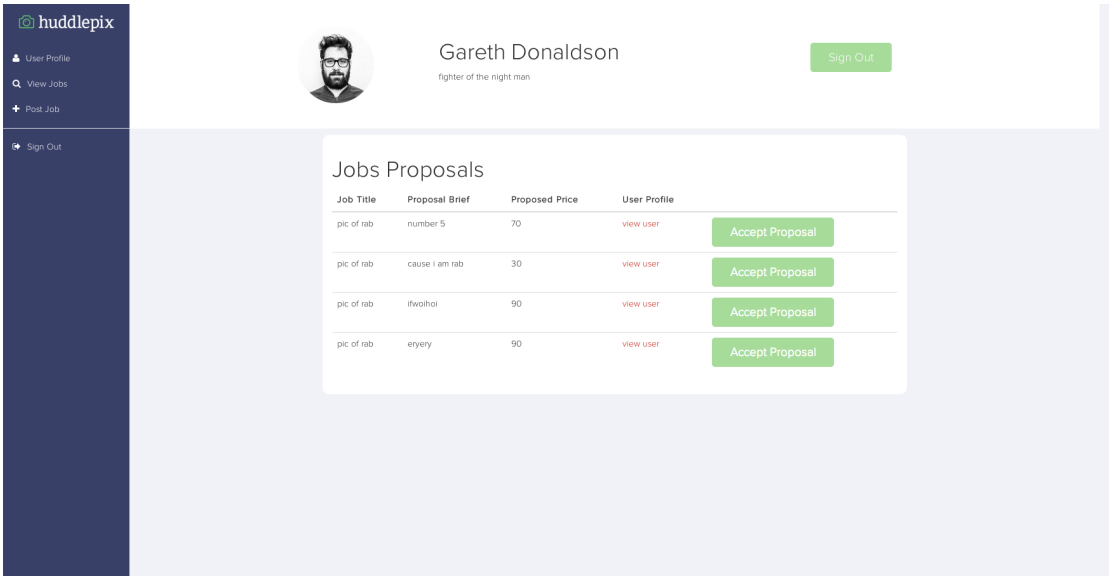
Sign In


Cookies on the Huddlepix Website
We use cookies to ensure that we give you the best experience on out website. Continue to use Huddlepix and receieve all the cookies we give you?
accept
huddlepix
Sign In
You made the right decision
Sign In
Please fill out the form correctly
Username
Username
Password
Password
Remember me
Sign in
Help
FAQ
Terms of Use
Privacy Policy
huddlepix
About
Who we are
Stats

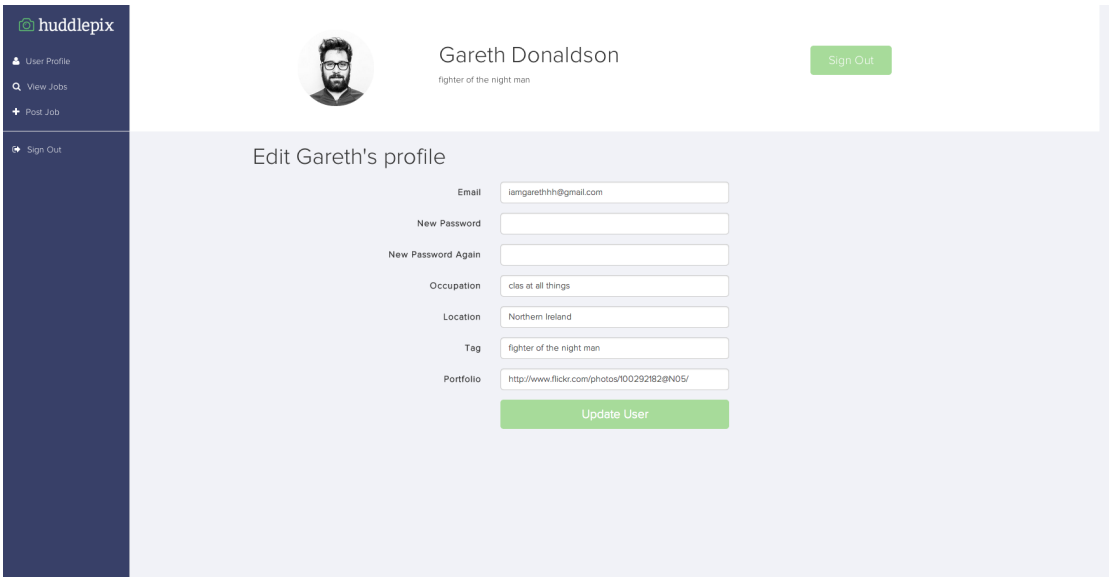
User Profile Page



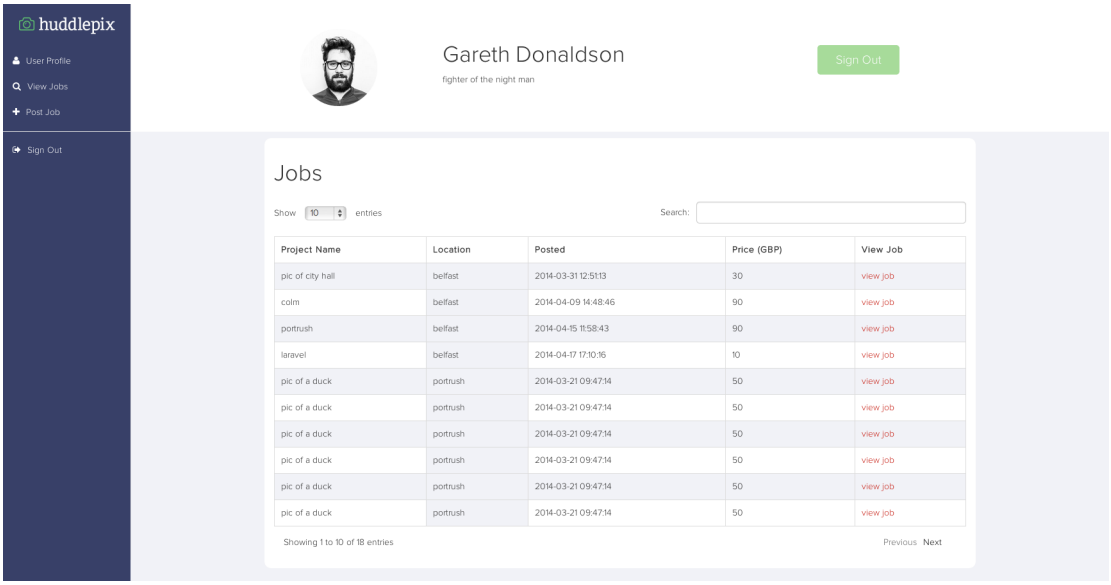
View all Proposals for a specific job



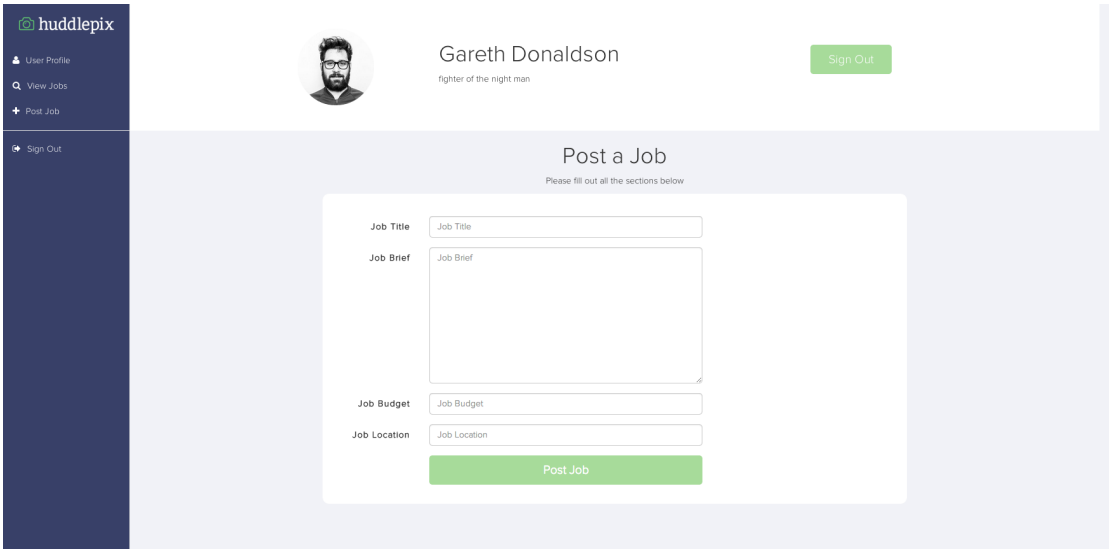
Edit User Profile



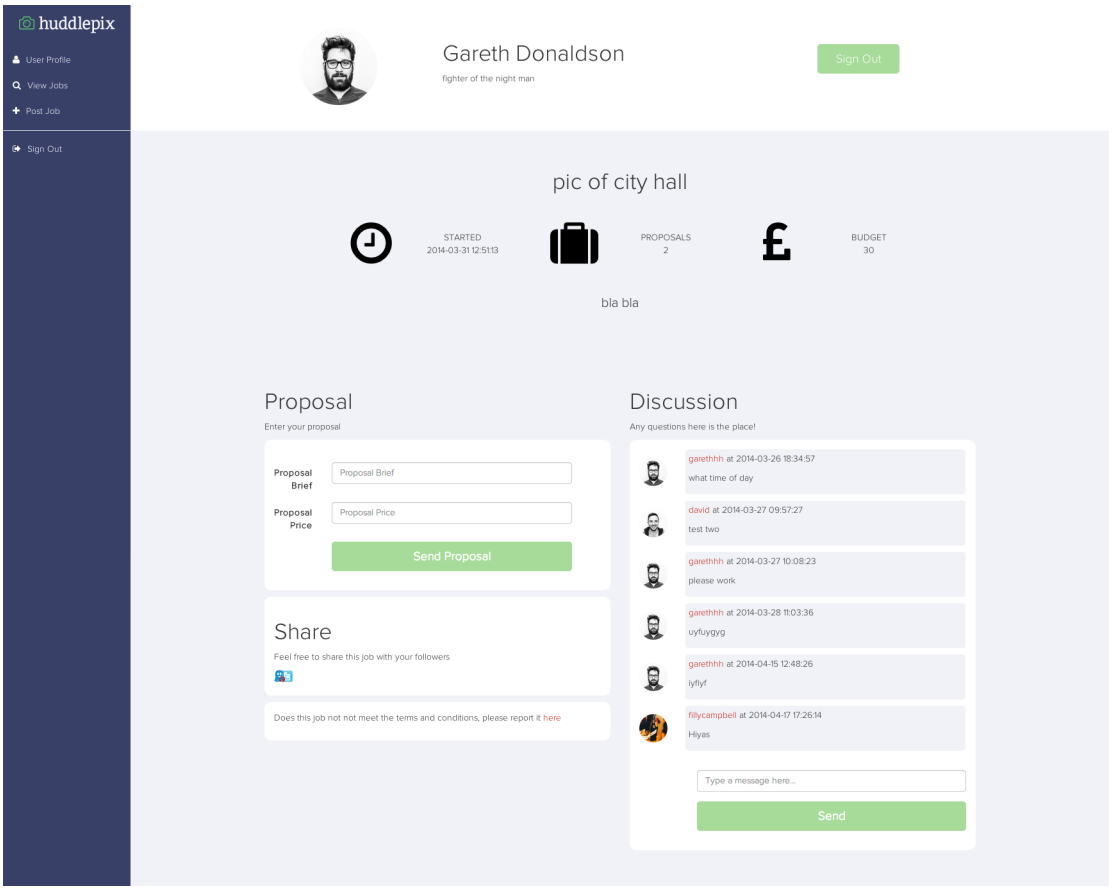
View All jobs



Post Job



Job Page



Appendix G – Testing

Sign In

Test ID	Description	Expected Results	Pass / Fail
H1	Click on the sign in button.	A model should appear with the sign in form.	PASS
H2	Fill in the correct username and password and sign in without selecting the remember me button.	The website will sign the user in, and be directed to the user profile page and no cookies where created.	PASS
H3	Fill in the correct username and password and sign in and select the remember me button.	The website will sign the user in, and be directed to the user profile page and cookies will be stored on the computer.	PASS
H4	Click on the sign in button with no username or password.	Error message beside both fields stating that field is required.	PASS
H5	Enter username and not the password field and sign in.	The username field should have the value that was entered previously and an error message should state that the password field is required.	PASS
H6	Enter password and not the and not the username field and sign in.	The username field will have an error message stating that that field is required. The password field will be empty.	PASS
H7	Enter an incorrect username and password.	Error message states Email / Password incorrect or account not activated.	PASS
H8	Enter a users details that have not verified there account.	Error message states Email / Password incorrect or account not activated.	PASS

Register an account

Test ID	Description	Expected Results	Pass / Fail
H9	Fill the form in correctly.	A successful message will appear and an email verification link will be sent.	PASS
H10	Click the verification link.	The link will direct the user to the homepage and a message will appear stating that the account has been verified.	PASS
H11	Click register without filling in any data.	Error messages will be displayed beside each field stating that, that field is required.	PASS
H12	Enter a username that has already been registered.	Error message states - The username has already been taken.	PASS
H13	Enter an email address that has already been registered.	Error message states - The email has already been taken.	PASS
H14	Enter a password under 6 characters.	Error message states - The password must be at least 6 characters.	PASS
H15	Enter two passwords that do not match.	Error message - The password again and password must match.	PASS
H16	Enter an invalid email address.	An error message will appear stating that a @ sign is required.	PASS

Edit User Profile

Test ID	Description	Expected Results	Pass / Fail
H17	Fill Form in correctly.	The user profile will be updated.	PASS
H18	Click update user without filling in any data.	Error messages will be displayed beside each field stating that, that field is required.	PASS
H19	Enter an email address that has already been registered.	Error message states - The email has already been taken.	PASS
H20	Enter an invalid email address.	An error message will appear stating that a @ sign is required.	PASS
H21	Enter two passwords that do not match.	Error message states - The password must be at least 6 characters.	PASS
H22	Enter a password under 6 characters.	Error message states - The password must be at least 6 characters.	PASS

Create a Job

Test ID	Description	Expected Result	Pass / Fail
H23	Fill the form in correctly.	The job will be created, and can be seen by the user.	PASS
H24	Click post job without filling in any data.	Error messages will be displayed beside each field stating that, that field is required.	PASS
H25	Leave a field out and click post job.	The entered data will still be present with an error message beside the field that was not entered	PASS
H26	Enter a letter into the Job Budget Field.	Error message states - The job budget must be an integer.	PASS

View all jobs

Test ID	Description	Expected Results	Pass / Fail
H27	Click on view jobs.	All jobs are present.	PASS
H28	Show different number of entries.	The number of jobs shown will match the number of entries selected.	PASS
H29	Search a job that exists.	The search results will display what the user has searched for.	PASS
H30	Search for something that dose not exist	Message will appear in the table stating – No matching records found.	PASS
H31	Sort any field (high to low)	The results are ordered from high to low.	PASS
H32	Sort any field (low to high)	The results are ordered low to high.	PASS
H33	The pagination works as expected.	Users will be able to view the next set you records.	PASS
H34	Click on view job	Users will be redirected to that specific job.	PASS

View Specific Job

Test ID	Description	Expected Results	Pass / Fail
H34	Click on view job.	All data should be correctly pulled from the database	PASS
H35	If the job belongs to the user that is viewing, a close job button should be shown.	Close Job Button is shown.	PASS
H36	Press close job.	The job is no longer shown on the job feed.	PASS
H37	The proposal form is filled in correctly.	A proposal will be sent for that job.	PASS
H38	Click send proposal without filling in any details.	Error messages will be displayed beside each field stating that, that field is required.	PASS
H39	Leave a field out and click post job.	The entered data will still be present with an error message beside the field that was not entered	PASS
H40	Enter a letter into the Proposal Price Field.	Error message states - The proposal price must be an integer.	PASS
H41	The discussion form is filled in correctly.	A message will be appear in the discussion section of the job page.	PASS
H42	Click send without filling in any details.	Error messages will be displayed beside each field stating that, that field is required.	PASS
H43	Leave a field out and click post job.	The entered data will still be present with an error message beside the field that was not entered	PASS
H44	A successful message will correctly display the users avatar, link to user profile and the message	User avatar, user name, and user message all present.	PASS

System Tests

Test ID	Description	Expected Results	Pass / Fail
ST1	Website should work on all browsers.	Website should offer a similar experience across various browsers.	PASS
ST2	Website should adapt to various screen sizes.	Website layout should change to suit the screen size.	PASS
ST3	Website should be viewable on portable devices.	Website should be work on mobile devices	PASS
ST4	All links should work.	Redirect the users to the correct web page.	PASS
ST5	User should be able to sign in / sign out on every webpage.	Sign in / Sign Out button located on every page.	PASS
ST6	All protected web pages should not be accessible for non-registered users.	Redirect the user to the sign in form.	PASS
ST7	Ensure the website is validated	Pass the validation Test	PASSED, upon minor changes being made.

Appendix H – Questionnaire

Website Evaluation Questionniare

Please circle the number that you consider most appropriate
and which reflects your opinions of this website:

1. Poor 3. Average 5. Excellent

Appearance / Layout	1	2	3	4	5
Visual / Colour Consistency	1	2	3	4	5
Appropriate / Relevant Content	1	2	3	4	5
Readablitiy of Text	1	2	3	4	5
Ease of Navigation	1	2	3	4	5
Functionality of the Website	1	2	3	4	5
Ease of Use	1	2	3	4	5
Any Obvious Security Flaws	1	2	3	4	5
Overall Satisfaction	1	2	3	4	5
Would you recommend Huddlepix to a friend?	1	2	3	4	5

Additional Comments