Site Commander

Major Project Report

Student Name | **Paul Smith**

Student Number | **B00581846**

Mentor | **Dr George Moore**

Peer Support Group | **8**

Course | **BSc Hons Interactive Multimedia Design**

Year | **2014**

Acknowledgements

My faithful and ever-present help: God

My faith in God influences every aspect of my life including my studies, and without God's help throughout my time at University I would not have made it this far. In Psalm 121 verses 2 – 3 the writer recognizes *"2. My help cometh from the LORD, which made heaven and earth. 3. He will not suffer thy foot to be moved: he that keepeth thee will not slumber."* These words resonate with my own personal experiences throughout my life, and they have been no less true throughout my time at University of Ulster.

Mentor and Module Co-ordinator for the Major Project: Dr George Moore

Special thanks to George who has been of great source of help and encouragement throughout the course of my Major Project. Without his advice, especially at the important planning stages, I would not have been able to move the project forward. I would also like to thank him for all of his efforts as Module Co-ordinator for the Major Project in planning and delivering a module which has given me an understanding of how industry projects are managed and delivered.

Course Co-ordinator: Dr Peter Nicholl

Special thanks to Peter for all of his efforts and Course Co-ordinator. He has worked tirelessly to ensure that all of our modules, assignments and timetable run smoothly, and has always offered and supplied help where problems have arisen; but most importantly, by his attitude he has given the course a great reputation.

All of my Module Lecturers

Thanks is also due to each staff member who has taught me over the duration of my time at University of Ulster, and although not mentioned by name, your efforts to impart your knowledge and skills to me are very highly appreciated.

University of Ulster

Thanks finally to the University of Ulster, and all of the staff who have made it possible for me to attend the university. Although not known by name, you have provided me with an excellent place of study.

Contents

1. Introduction

1.1. What is the Project?

Site Commander is a content management system that allows clients to view and edit the content on their website as if they were simply making the changes directly on their live site. The product is aimed at professional web developers and designers, to give their clients an easy way to update their website remotely.

A combination of HTML, CSS, jQuery, PHP and MySQL has been applied to allow the client's website to be integrated with the CMS.

After the CMS has been installed on the server, the developer can create an HTML template for their client's website, and can then upload the template's header, footer, CSS and JavaScript files to the CMS. They can also upload a second custom header in the event that subpages look slightly different to the homepage header.

The developer can then upload a layout file; the section of the website which can be edited. The layout file is created based on the template the developer has made for the clients website, so it is the developers responsibility to ensure that the layout looks right on the website, meaning that they have to provide CSS and JavaScript files as required for it.

The layout file is different to the rest of the initial uploads in that some editing is required in order for the CMS to make its content editable. The developer must add the class 'scdr-editable' (short for 'site commander editable') to every element within the layout file that they want to allow content to be added to.

Once a website has been added to the CMS, the Client can log in and click on the Edit button to open the Edit Panel and see their website in the same format as it would be seen live. The two main differences with viewing the page in this mode (as opposed to viewing the website live) are the toolbar along the bottom of the browser window containing all of the tools required to make changes, and the handle on each of the content elements (provided content has been added) so that they can be dragged around on the page.

The CMS uses a neutral colour scheme so that it does not overpower the colour scheme of the website that is being viewed through it.

1.2. Aims and Objectives

1.2.1. Aims

- Offer web developers an adaptable Content Management System for their clients.
- Give the client control over their website without compromising its design.
- Create an end product that is commercially viable.
- Challenge and develop design, PHP and jQuery skills.

1.2.2. Objectives

- Establish the system features and specifications before design commences.
- Design and refine how the user interface will look before creating the program.
- Build and optimise the user interface and components to work as the design intended.
- Test run the system to identify pitfalls and bugs to rectify them before completion.

1.3. Overview of the Work Undertaken

The majority of the work undertaken is made up of three areas of the system design:

- CMS Framework
- Editing Software Integration
- Database Communication

1.3.1. CMS Framework

As the CMS is designed for developers specifically, the ability to add new features to it was essential from the beginning. This necessitated that a common layout be introduced for each task that could be carried out on system.

Each feature-set had to be contained inside its own frame, and be allowed to have its own supporting CSS, JavaScript and PHP files. Each of these feature-sets are now called panels, and every time a new panel is added, a button to access it is placed on the main menu in the CMS

1.3.2. Editing Software Integration

A huge part of a CMS is its ability to manipulate content on the website it has been linked with. The editing software that has been created for this CMS is slightly different because of it's ability to take the entire website and display it in an editable form.

Overcoming this challenge has required a blend of HTML elements, jQuery interaction, PHP processing and a JSON communication line to exchange data between these three technologies.

1.3.3. Database Communication

The dynamic nature of the system has required the use of JSON, an extension of AJAX which allows HTML pages to send data to PHP files using jQuery. Once the data reaches the PHP file, it can then be processed to Site Commander's MySQL database.

1.4. Report Overview

The next six chapters document the project's journey from an idea through to a functional product; they will look at the following:

- How the initial concept was developed into a manageable project.
- The system's transformation from paper sketches to a detailed user interface design.
- Taking the design into reality, looking at the challenges presented by building a CMS.
- Putting the final product through its paces to demonstrate it's good points, as well as areas which will need to be improved.
- Summing up the outcomes of each stage of the product's development.
- Rounding up and reflecting on what this report has looked at and the future plans for this project.

2. Concept Definition Statement

2.1. Idea generation

The original idea was to create an environment that would make it easier for a client to make changes to their website.

The main functionality would allow the client to drag and drop, a new text field, image gallery, single image, video, etc, onto their website. They would also be able to come back and make changes, such as reordering elements on the page, modifying text, adding new images, changing a video etc.

The user would be able to control all this while logged into their website backend. Once logged in, they would be met by their website, with a toolbar overlay containing all of the tools required to drop new elements onto the page, edit text, add new pages etc.

As well as the client being able to make their changes, the product would work for web developers too. A web developer would be able to add an HTML template directly into the product. The template would obviously include a header and footer, as well as a defined content area where all of the drag and drop elements could be placed into. They would also be able to upload templates for different scenarios such as a homepage, subpage or sub-subpage.

The end product could either have been made available as a service where all of the files would be controlled from a central domain name and server, or as a downloadable package that the end user could install on their FTP - this would mean the user setting up their own hosting and database.

2.2. Requirements Specification

2.2.1. Project Drivers

Purpose of Project

Site Commander is designed to tackle a problem that faces many website designers and their clients. A content management system traditionally allows a client to make changes to their website, but is limited to a single layout for pages or, if layout changes can be made, it isn't fast, or convenient for the designer to do, and often compromises the original website design.

Site Commander saves both the designers time and the client's money by not compromising the website which they have both worked hard for. New page layouts can be added quickly by the designer at the client's request, without damaging the website design. The client is also free to make content changes in a highly user-friendly environment; this should eliminate the need for them to request assistance from the website designer.

A simple example of how Site Commander will help adopters provide a better service to their clients would be this: If the client currently had a webpage with two columns for content, and wished to have another page with a large image across the top, with a two column layout underneath, they could explain this to the designer, who would be able to decide if this layout would be fitting with the current website design. Once deemed okay, the designer would be able to add this layout as an option within Site Commander so that the client could then access and use it.

Stakeholders

There are two stakeholders; the first is the project mentor, George Moore. His direction and guidance are essential to ensuring that the intended project aims are realized, and also that any new aims that are identified complement the original project aims, and do not compromise them.

The second stakeholder is a group of people, Web Designers. Their requirements will have heavy bearing on the final user experience. The requirements of web designers are of special importance because they are the purchasers, the users and also the resellers of the product.

Figure 1.1: Purpose Advantage Measurement (PAM) card

Purpose: To create a faster way of making changes to the client's website.

Advantage: Saves money for the client and time for the designer, as well as allowing the clients goals to be realized quickly

Measurement: After logging into the system 10 times, developers will be given the opportunity to feedback their thoughts on the system

For the purchaser to actually adopt Site Commander they need to see its benefits, these benefits can be established if ***Figure 1.1*** above is fulfilled.

Site Commander must be well designed as it needs to be easy to use. For this reason, I have identified the features that need to be included in Site Commander in order for it to meet the needs of web designers:

- Ability to customise CSS files
- Ability to add new content layouts
- Lock/unlock client functionality to meet their needs
- Limit number of pages that appear in the navigation
- Limit number of subpages that can be added under a main page

Without benefitting the web designer's clients, the product has no purpose, so to ensure that Site Commander will help their clients, I have identified the features that need to be included in order to help the client:

- Choose a content layout
- Edit text
- Upload/remove images
- Add/remove YouTube video
- Add/remove Google Maps
- Add/remove pages
- Add/remove subpages

2.2.2. *Project Constraints*

Mandated Constraints

The product must adhere to constraints so that the scope of the project does not become too large. I have set out three constraints that will have an effect on the entire project, **Figure 1.2**, **Figure 1.3** and **Figure 1.4** below.

Description: The product shall operate within the web browser of a computer.

Rationale: The product must be available to the customer anywhere as long as there is an Internet connection.

Fit criterion: The product must operate seamlessly in all Internet browsers.

Figure 1.2: Description Rationale 1

Description: Logins for the product must be secured with SHA1 security encryption.

Rationale: Unauthorized access to the product must be prevented.

Fit criterion: The product will keep the clients website and content safe from attack.

Figure 1.3: Description Rationale 2

Description: The product must inform both the designer and the client when the other makes a change.

Rationale: The product must create close collaboration between the designer and the client.

Fit criterion: The product will assist the client in reaching their desired goals in relation to their website.

Figure 1.4: Description Rationale 3

It is very important that the project runs to the schedule that has been laid out below. This schedule shows all of the deadlines for key points in the project; in order for the project to be delivered on time, these deliverables must be met.

Figure 1.5: Project Schedule

Task	Time Scale	Deadline
1 - Create a list of all of the features to be included in the program	3 Weeks	08/11/2013
2.1 - Prototype the design on paper - Sketch the user interface layout - Sketch each screen within the program - Storyboard how operations will be carried out - Design the file and folder structure 2.2 - Design and build the database that will store data for the CMS	1 Weeks	15/11/2013
3 - Code prototypes of individual system features - Combine HTML, PHP & AJAX to see how the individual parts of the system will function	1 Week	22/11/2013
4 - Create an initial system design in Photoshop - Brand the system - Design the user interface layout based on sketches - Design each screen within the program based on sketches - Design how operations will look based on storyboard sketches	2 Weeks	06/12/2013
5 - Code a working prototype of the system - Build HTML & CSS interface based on Photoshop designs - Add jQuery elements from individual system features - Add HTML & PHP code from individual system features - Link to the database	2 Weeks	20/12/2013
6 - Refine the original design	4 Weeks	21/02/2014

6 - Debug and refine the system - Add or modify functions - Fix code bugs - Get rid of bloat code - Ensure that code meets professional standards	7 Weeks	14/03/2014
7 - Final Touches - Design and build a website to provide the system from	5 Weeks	18/04/2014

Naming Conventions and Terminology

The Designer: This is the person who will actually purchase and use the product.

The Client: This is *The Designer's* client; the designer will implement the product to suit their needs.

Site Commander: The name of what is sometimes referred to as *the product* throughout documentation for the project.

CMS (Content Management System): The technical name for the main function of *Site Commander*.

Website: Dependent on context, if mentioned in relation to *The Client*, it refers to the website which is being supported by *Site Commander*, if mentioned in relation to *Site Commander*; it refers to the website through which *Site Commander* is supported.

2.2.3. *Functional Requirements*

Scope of the Work

There are four parts of the project that are essential to getting the system to work seamlessly:

- The Designer's interaction to the application
- The Client's interaction to the application
- How the browser deals with the application
- What the application does when problems arise with the internet connection

Scope of the Product

Below are the functional requirements that deal with what the product will actually be able to do. With the help of a dual login (one area for the designer, one for the client), both the web designer and their client will be able to care for the website in the ways that they know best, a designer being able to care for the look, feel and layout of the website, and a client being able to care for their brand through their websites content including text, images, videos, and contact details.

- The product must have a fully functional database where a user's interaction with the CMS can be stored and retrieved from.
- The designer must be able to make changes to the CSS Style Sheet for the website layout.
- The designer must be able to build a content layout quickly and simply.
- The designer must be able to specify limitations in regard to what the client can and cannot do within the product.
- The client must have the ability to select a content layout when they choose to add a new page
- The product must meet HTML5 and CSS3 compliant standards.
- The product must allow the website to respond to varying screen sizes.

2.2.4. *Non-functional Requirements*

Look and Feel

- The product interface must have a clean and smart appearance.
- The product must be enjoyable to use.
- Transitioning from task to task must be smooth, not abrupt

Usability and Humanity Requirements

- Users with basic computer skills will understand how to operate the product.
- The user will be able to easily remember how to use the product when they come back to it at a later date.
- The product will be structured to avoid the user making errors.
- The user be satisfied with the overall layout and operation of the product
- The product will give the user visual feedback of when they have completed a task or made a

mistake.

Performance Requirements

- The system should complete the users request within 5 seconds.
- They system will inform the user when a request is taking longer to complete than the specified default time.

Operational and Environmental Requirements

- The product shall be compatible with any web browser supporting HTML5 and CSS3.
- The product must be accessible by both the client and the designer at all times.
- The product must be used on a desktop, laptop or tablet; a Smartphone is too small.

Maintainability and Support Requirements

- The product must make the user aware when a change has been made to the software
- New features must be deployed automatically once the user agrees to accept them.

Security Requirements

- The product must include SHA1 password encryption
- A designer and a client account will be the only two access points into the system.
- The product will only store information which is to be output to the visible website

2.2.5. Project Issues

Risks

The major risk with this project is that time runs out, this is especially a risk during the first two months of the project where a lot of the groundwork is being completed.

Costs

There are no monetary costs with this project, however this is made up for in the amount of time required to complete the project to a high standard. If the schedule shown in **Figure 1.5** is followed closely, the total cost in time will be 25 weeks of work. One advantage with the schedule is that a lot of the groundwork is required within the first 2 months; if this work can be completed in this timeframe, the rest of the projects bulk can be completed ahead of the schedule.

Waiting Room

The whole project has changed slightly in regard to the output. Originally a drag and drop interface where the client could literally build their page was envisaged. With the advice and instruction given by the design lecturers, the end product has been refined to be an editing tool, rather than an creation tool.

2.3. Paper prototyping

Figure 1.6: 6-Up sketches

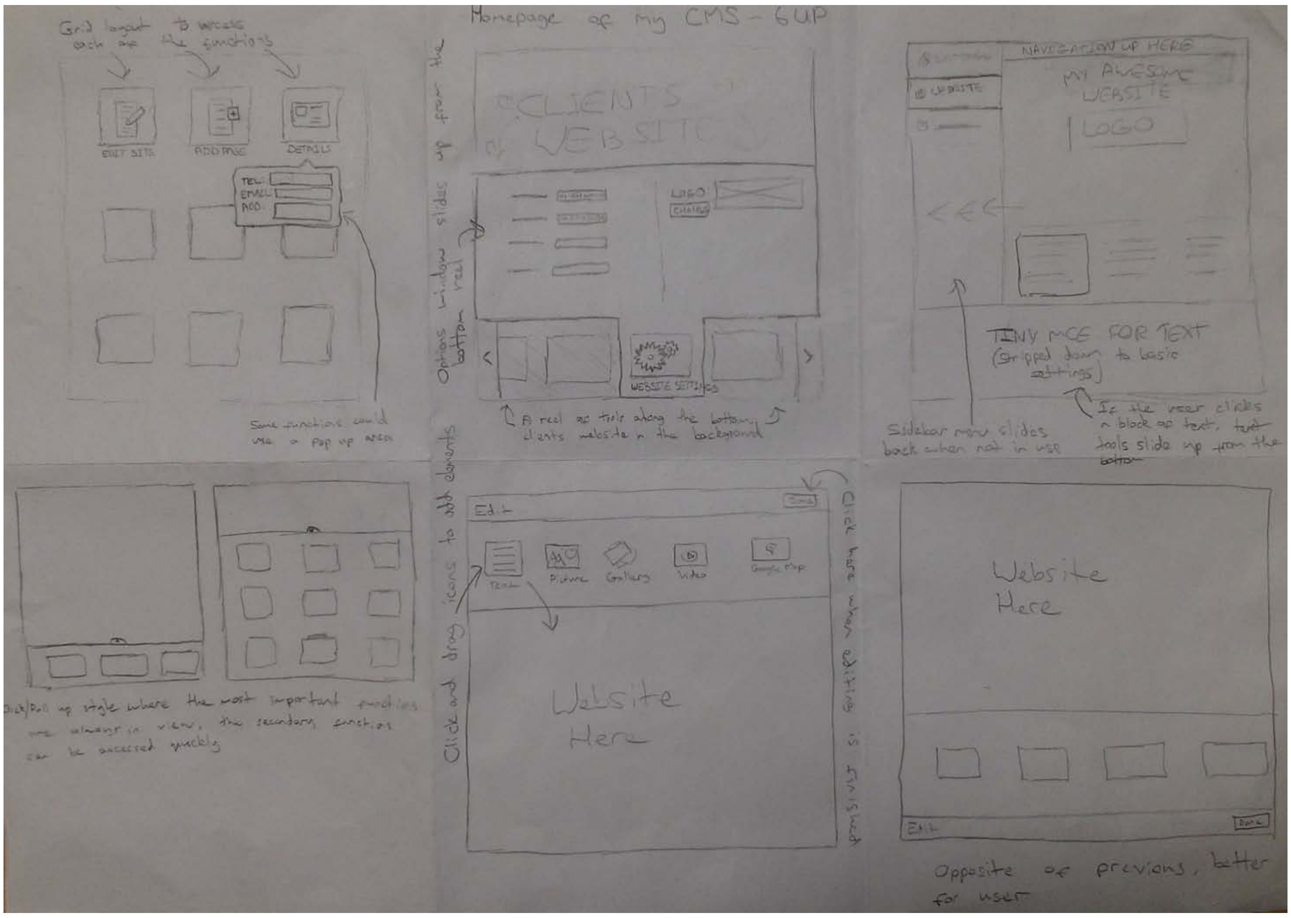

Grid layout to access each of the functions
EDIT SITE
ADD PAGE
DETAILS
TEL:
EMAIL:
ADD:
Some functions could use a pop up area
Homepage of my CMS - 6UP
CLIENTS WEBSITE
Options window slides up from the bottom reel
LOGO
CHANGE
WEBSITE SETTINGS
A reel of tools along the bottom, clients website in the background
NAVIGATION UP HERE
WEBSITE
MY AWESOME WEBSITE
LOGO
TINY MCE FOR TEXT (Stripped down to basic settings)
Sidebar menu slides back when not in use
If the user clicks a block of text, text tools slide up from the bottom
Stick/Pull up style where the most important functions are always in view, the secondary functions can be accessed quickly
Click and drag icons to add elements
Edit
Done
Text
Picture
Gallery
Video
Google Map
Website Here
Click here when editing is finished
Website Here
Edit
Done
Opposite of previous, better for user

Figure 1.6 above shows the first six ideas that came to my mind during the paper prototyping process.

Top Left

The initial idea for the UI was to have a grid menu when the user could click on one of the items to perform a function.

Top Middle

Secondly, the idea of keeping the website always in view was considered; here a carousel is envisaged from which each tool slides up out of.

Top Right

As an alternative, a side bar style interface was looked into, though it was deemed that this would affect the layout of the website, as websites are created to fit within the width of the screen, this would push the whole website over to one side, detracting from the user experience.

Bottom Left

A combination of the top left and top middle ideas was included as a way of addressing how the website is viewed in relation to the CMS interface, here a toggle button allows all but the most important functions to be hid, and the revealed again.

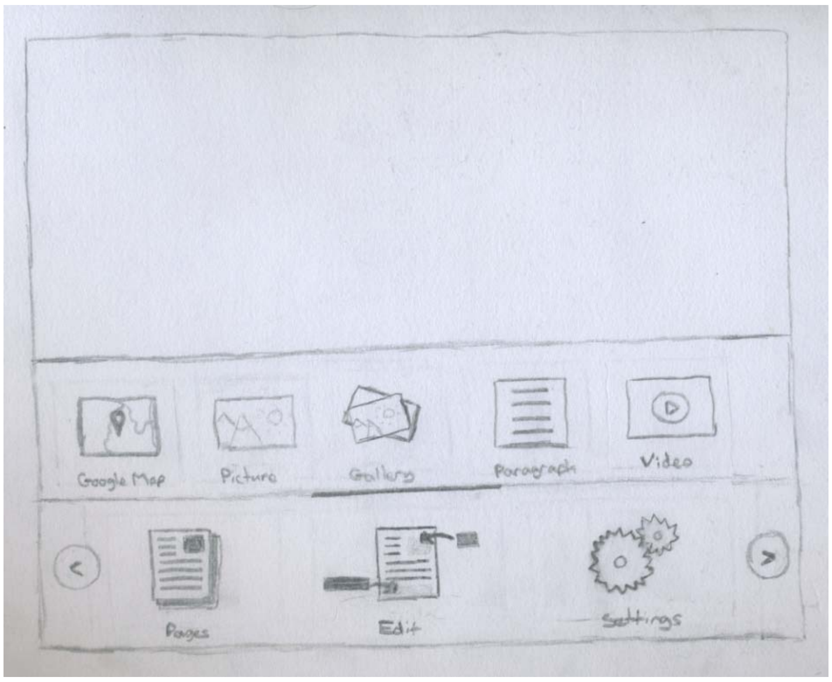
Bottom Middle

Adapting the top middle idea, having the interface above the website viewer was an attempt to see if this reversed interface felt more natural.

Bottom Right

Adapting the top middle idea, this sketch considered just the editing functionality of the CMS as a feature of a larger system.

Figure 1.7: Sketch



After completing the 6-ups, **Figure 1.7** was considered in more detail. A reel with all of the available tools runs along the bottom, when the client clicks on one of them, the available options for that tool slide up above it. It also shows the drag and drop tools that would be available to edit a web page.

Figure 1.8: Sketch

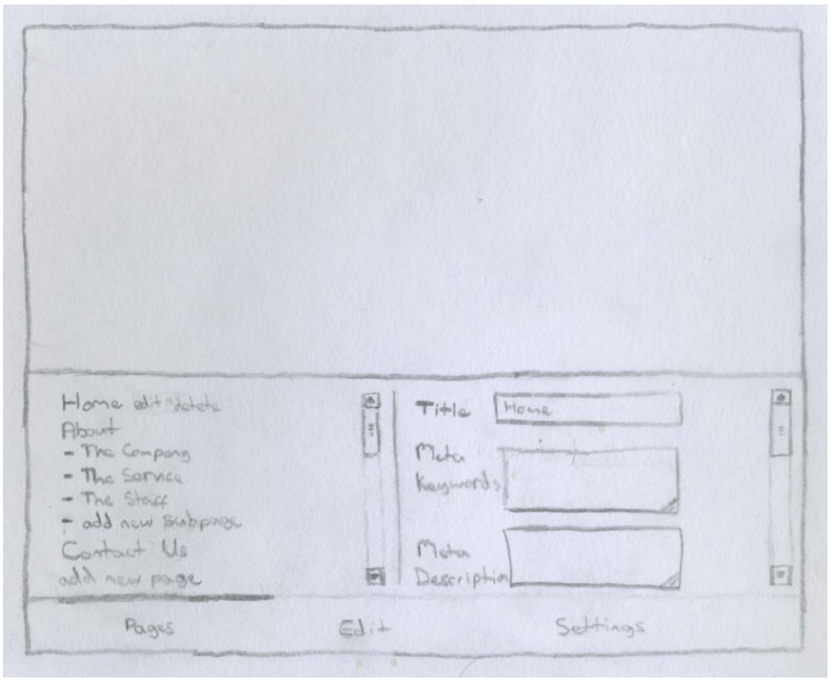


Figure 1.8 demonstrates how the main reel would hide itself to allow more room for the tool options available, it also shows some of the options that would be available for changing page settings such as the page title and META data.

Figure 1.9: Sketch

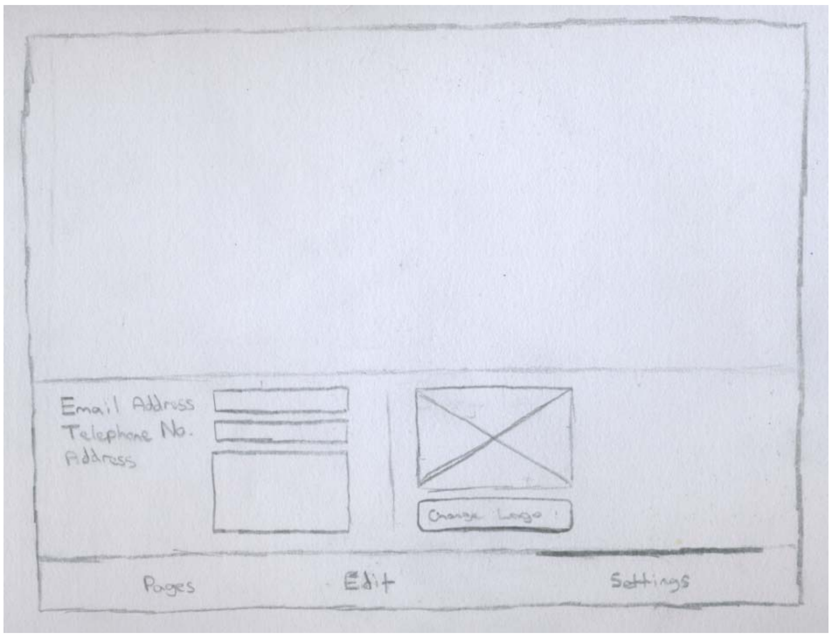


Figure 1.9 shows some of the settings that would be available to the client for making global changes; the email, telephone and address fields could link to areas on the website where this information is displayed.

Figure 2.0: Sketch

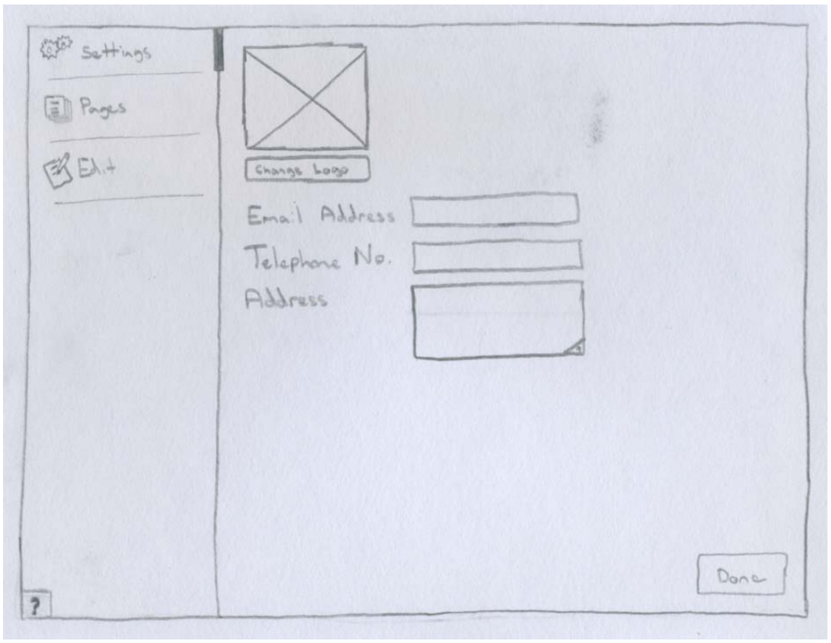


Figure 2.0 shows the second idea considered on paper, where the main menu of the CMS would be a sidebar, with information sliding out from the left. Again, tool options are shown in this new layout.

Figure 2.1: Sketch

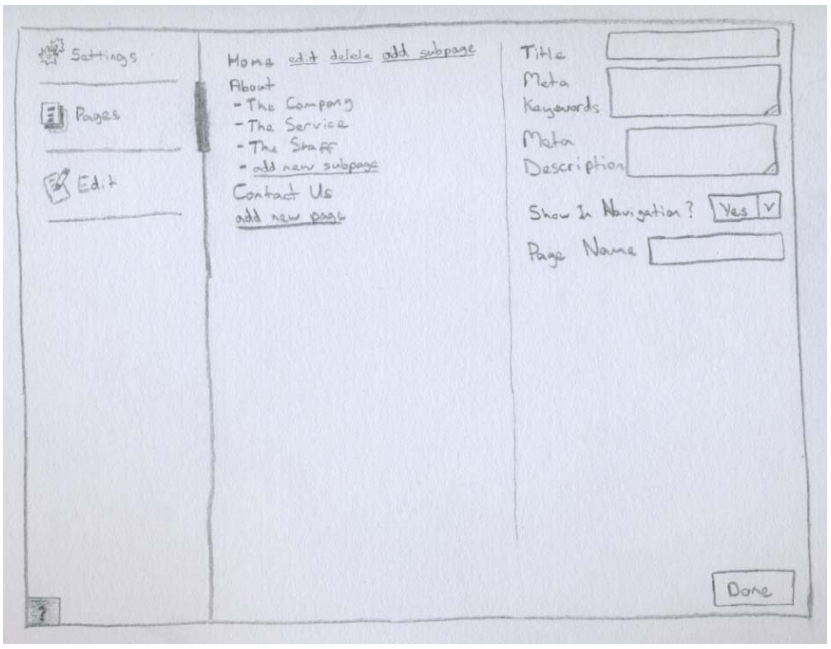


Figure 2.1 shows the sidebar menu with options demonstrated.

Figure 2.2: Sketch

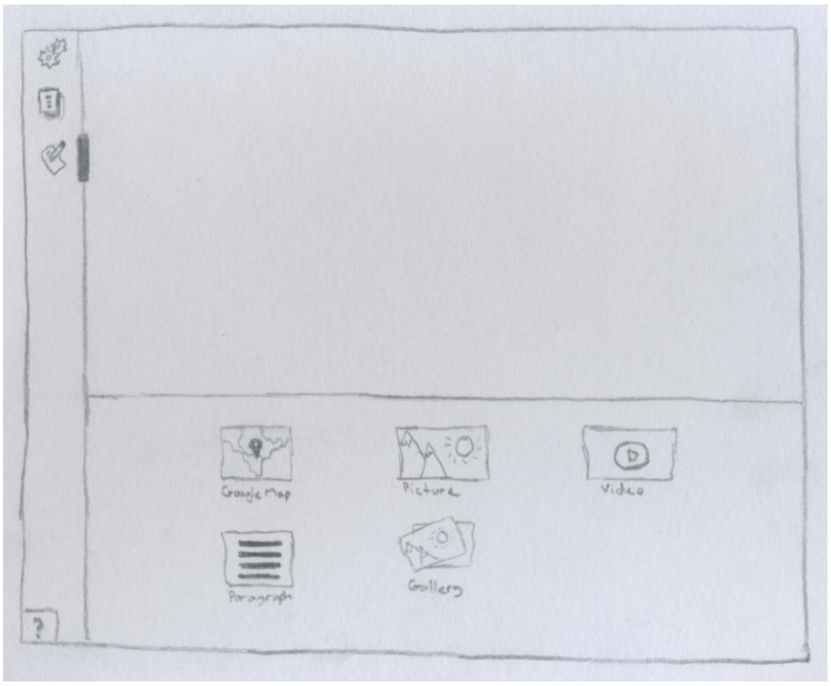
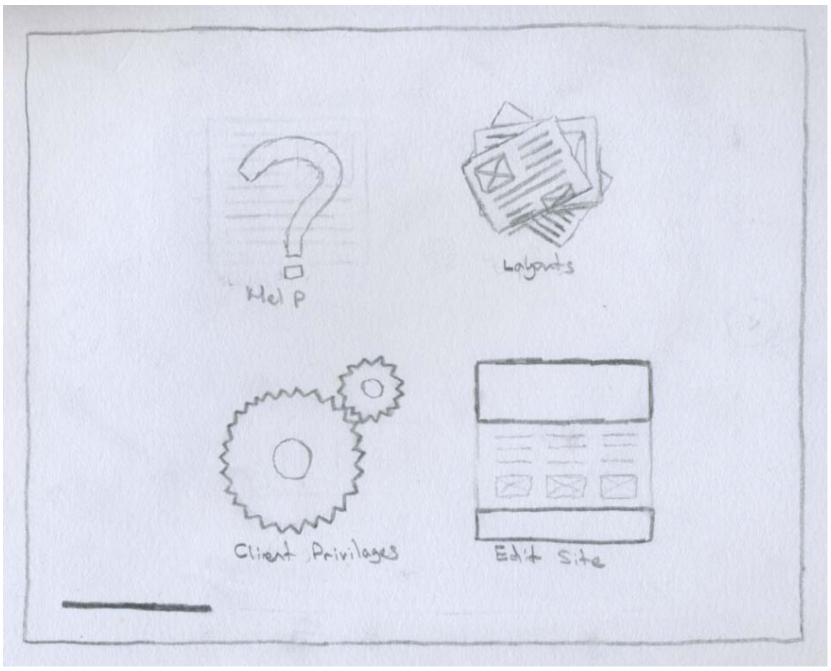


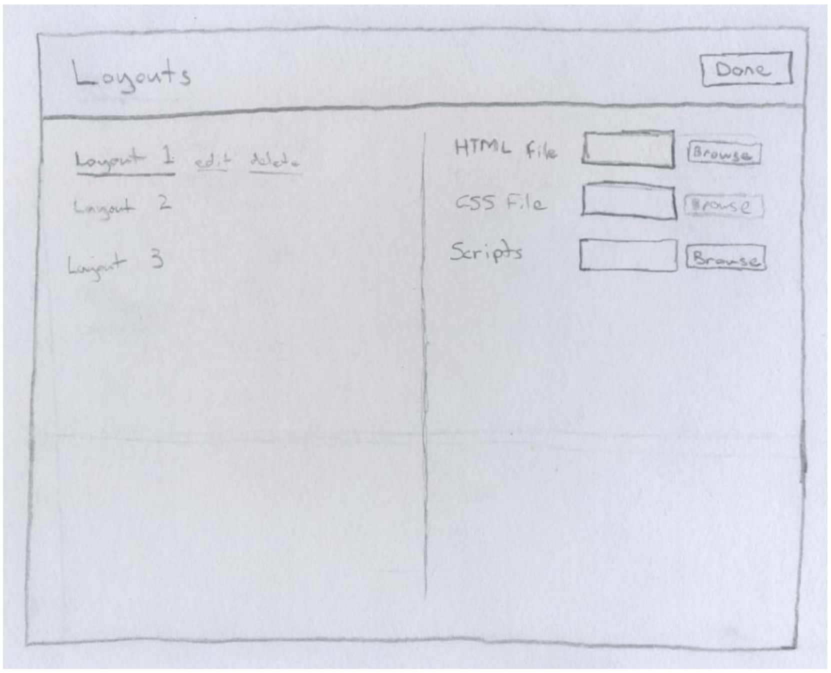
Figure 2.2 shows the slight difference in operation that would take place when editing page content; the sidebar would slim down to just icons, and the available editing tools would slide up from the bottom.

Figure 2.3: Sketch



The designer side of the system then had to be considered as to how it would look, **Figure 2.3** incorporates features from the last two ideas, and the original 6-ups.

Figure 2.4: Sketch



Options would be displayed full screen, and a done button would allow the designer to get back to the menu. **Figure 2.3** and **Figure 2.4** above show some of the tools available to the designer.

Figure 2.5: Sketch

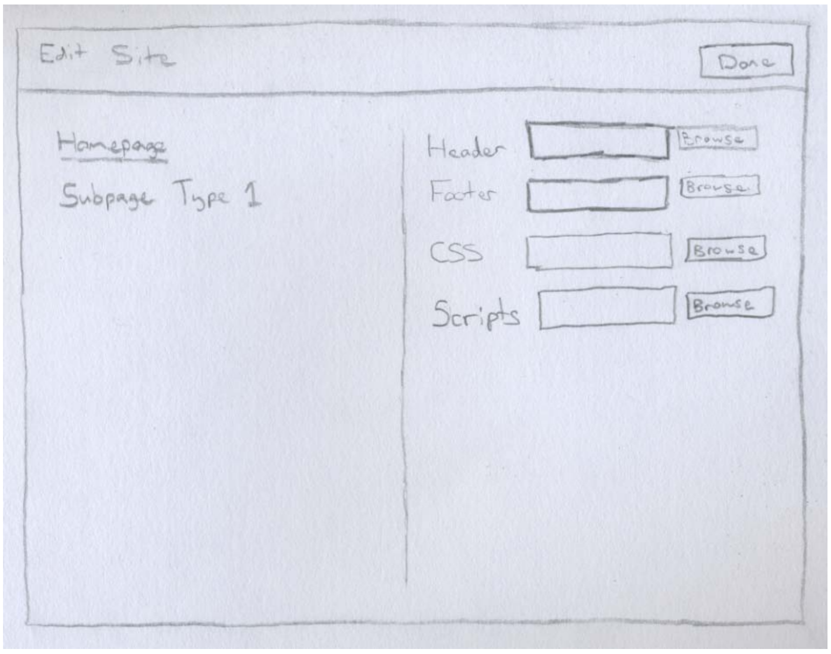


Figure 2.5 shows some more of the tools available to the designer.

Figure 2.6: Sketch

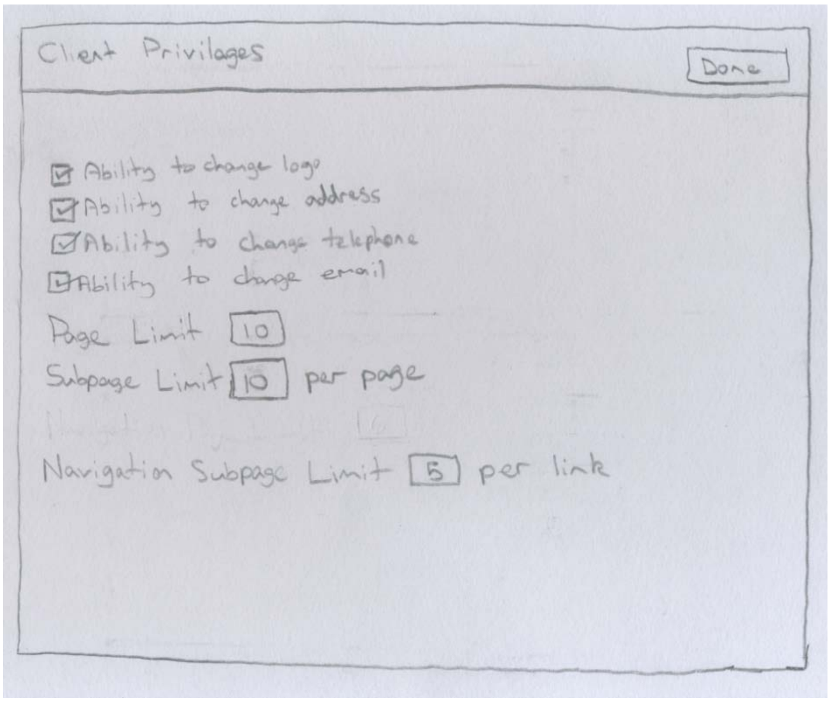
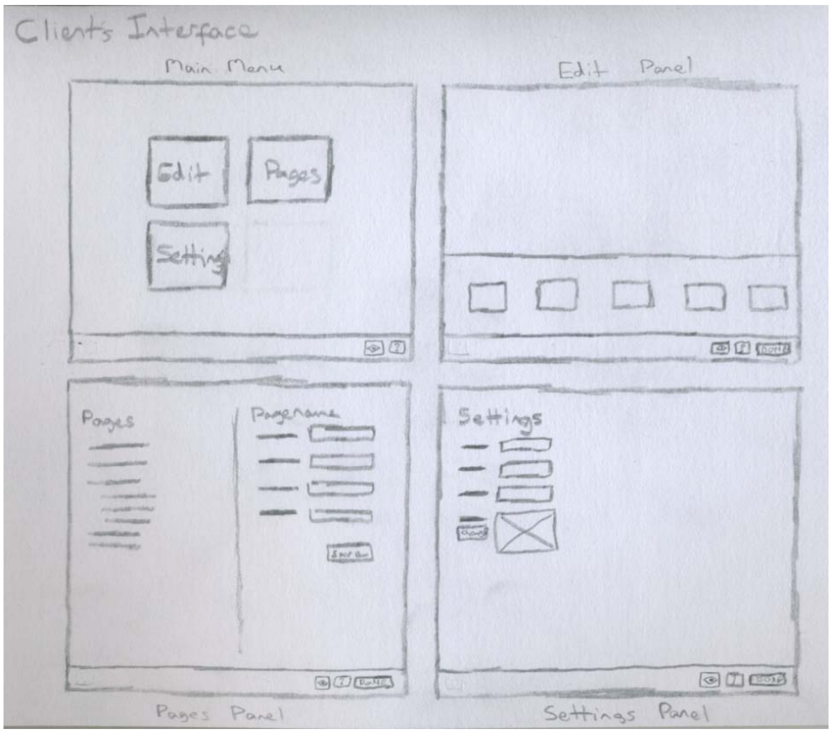

Client Privilages
Done
Ability to change logo
Ability to change address
Ability to change telephone
Ability to change email
Page Limit 10
Subpage Limit 10 per page
Navigation Subpage Limit 5 per link

Figure 2.6 again, shows some more of the tools available to the designer.

Figure 2.7: Sketch



Based on the idea for the designer side of the system, its interface was considered for the entire system, which includes the Setup, Designer and Client Interfaces. The toolbar that was at the top of the page is now at the bottom as its position proved to be a problem for the page editing panel, as it's controls are at the bottom; in other words it wouldn't be wise to limit the space where elements can be dropped onto. See **Figure 2.7**

Figure 2.8: Sketch

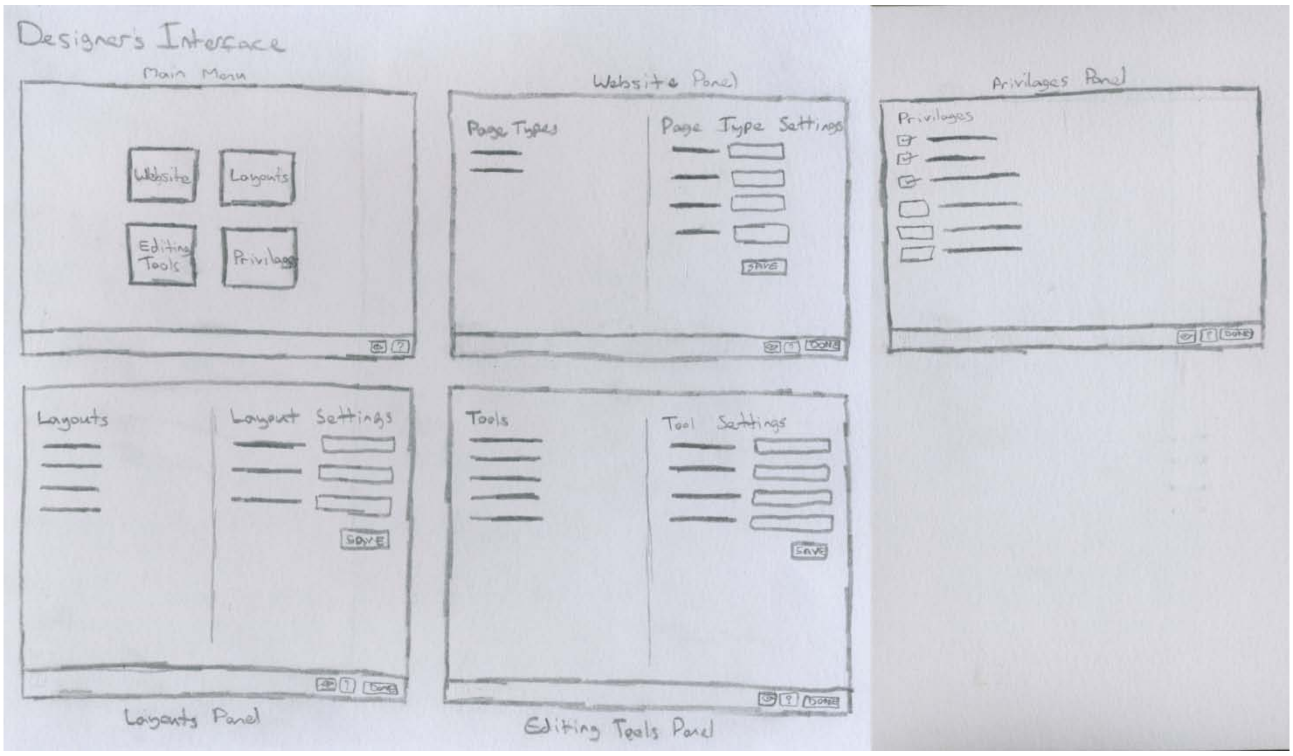


Figure 2.8 is the basic page layout for the designer's interface.

Figure 2.9: Sketch

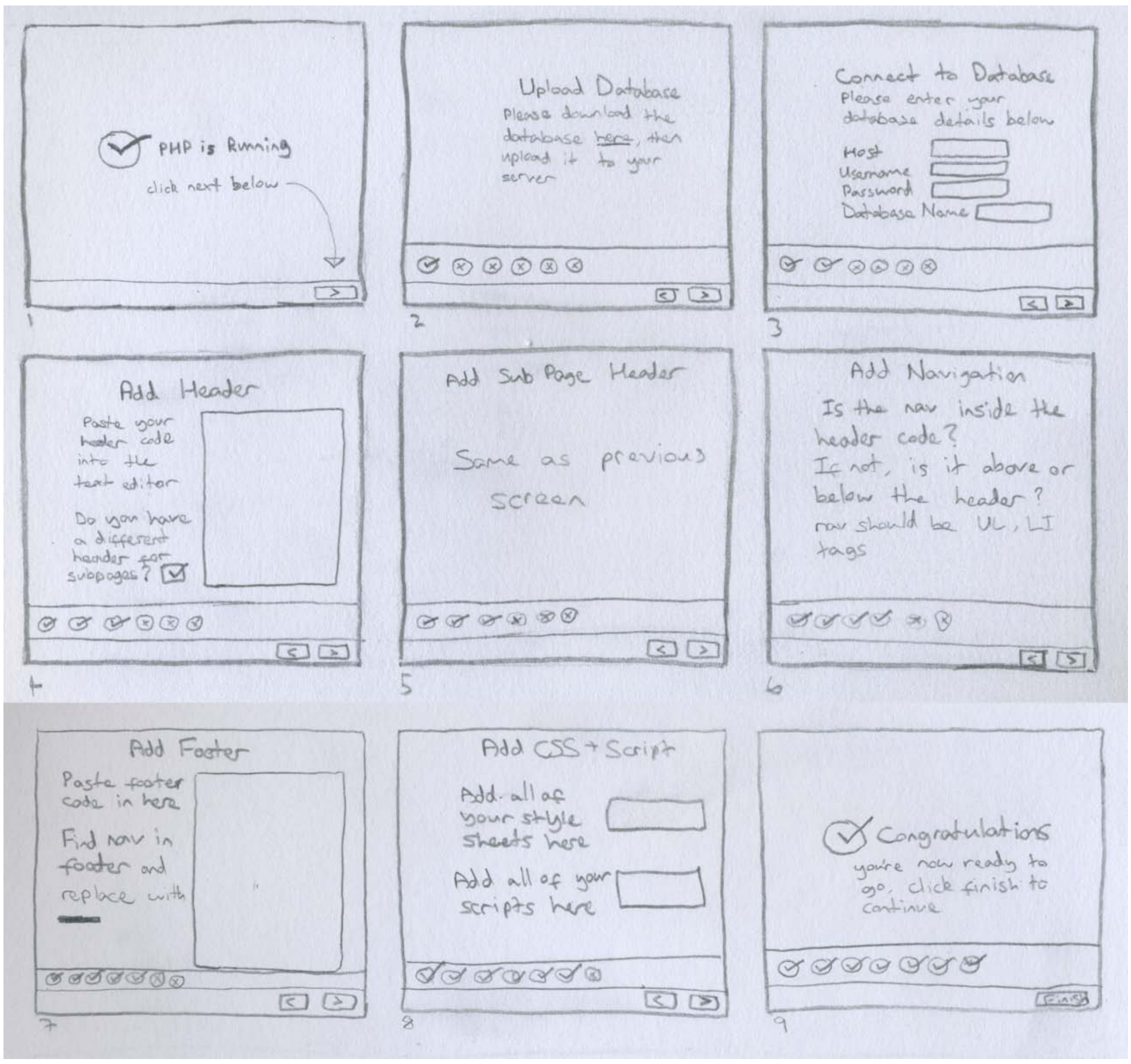

PHP is Running
click next below
Upload Database
Please download the database here, then upload it to your server
Connect to Database
Please enter your database details below
Host
Username
Password
Database Name
Add Header
Paste your header code into the text editor
Do you have a different header for subpages?
Add Sub Page Header
Same as previous screen
Add Navigation
Is the nav inside the header code?
If not, is it above or below the header?
nav should be UL, LI tags
Add Footer
Paste footer code in here
Find nav in footer and replace with
Add CSS + Script
Add all of your style sheets here
Add all of your scripts here
Congratulations
you're now ready to go, click finish to continue
Finish
1
2
3
4
5
6
7
8
9

Figure 2.9 is the layout for the software setup when the product is deployed on a server.

__Figure 3.0: Wireframe__

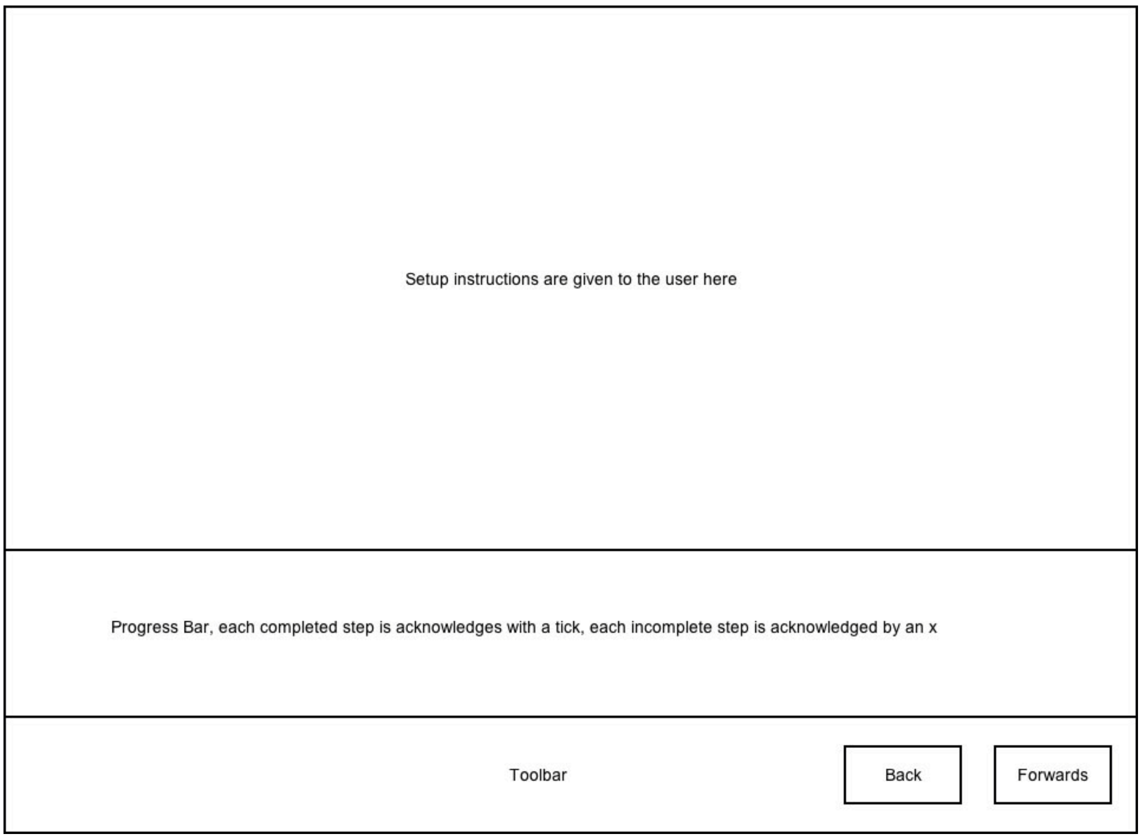

Setup instructions are given to the user here
Progress Bar, each completed step is acknowledges with a tick, each incomplete step is acknowledged by an x
Toolbar
Back
Forwards

Figure 3.0 is a wireframe which simply clarifies the previous sketches. Each stage of the product setup process will follow this layout.

Figure 3.1: Wireframe

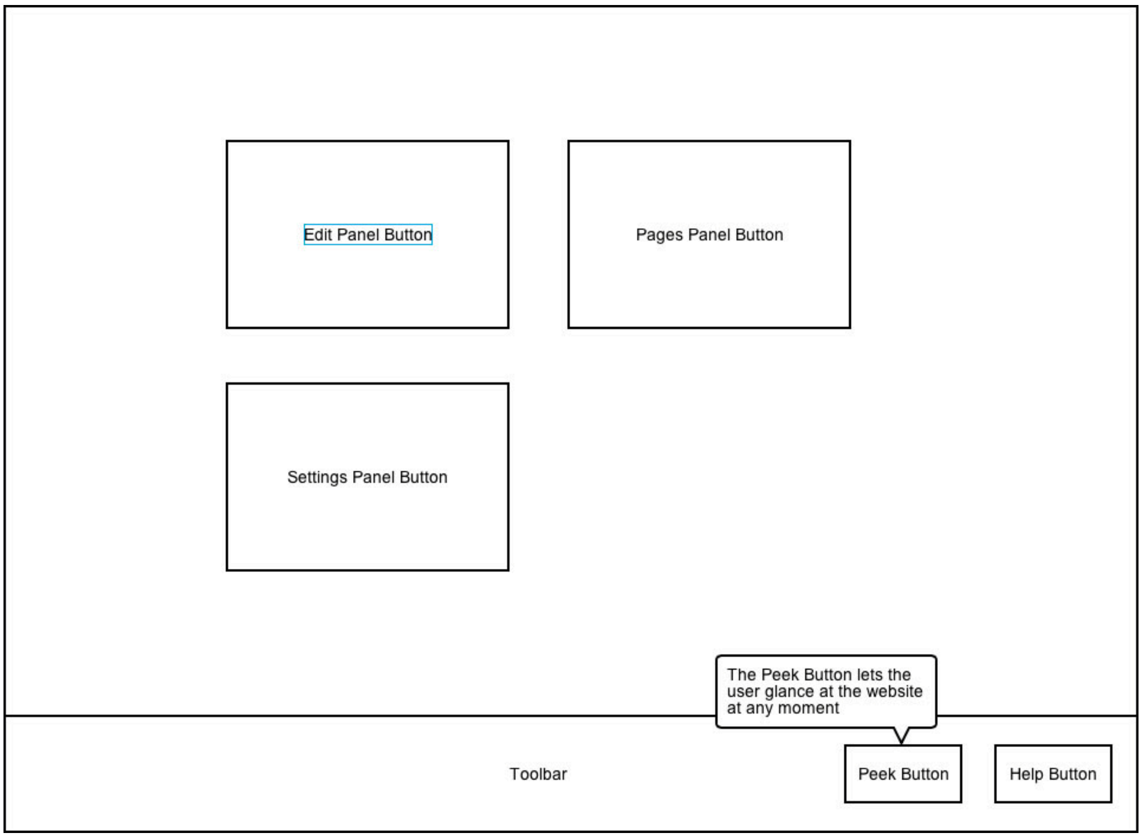


Figure 3.2: Wireframe

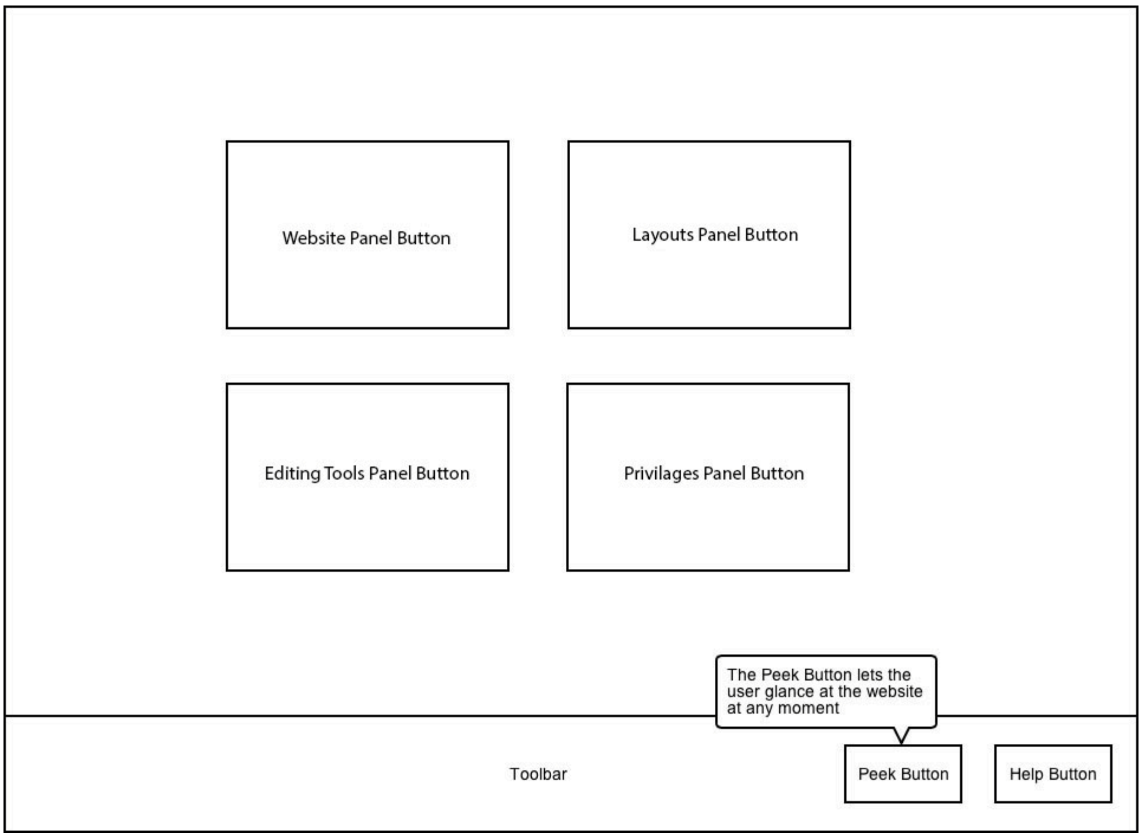


Figure 3.1 and **Figure 3.2** clarify the layouts for the designer and client interfaces. Details about a new button have been included; the peek button will allow the user to take a quick glance at the actual website anywhere within the CMS. I think this will be useful so that when a change is made, it can be seen visually.

Figure 3.3: Wireframe

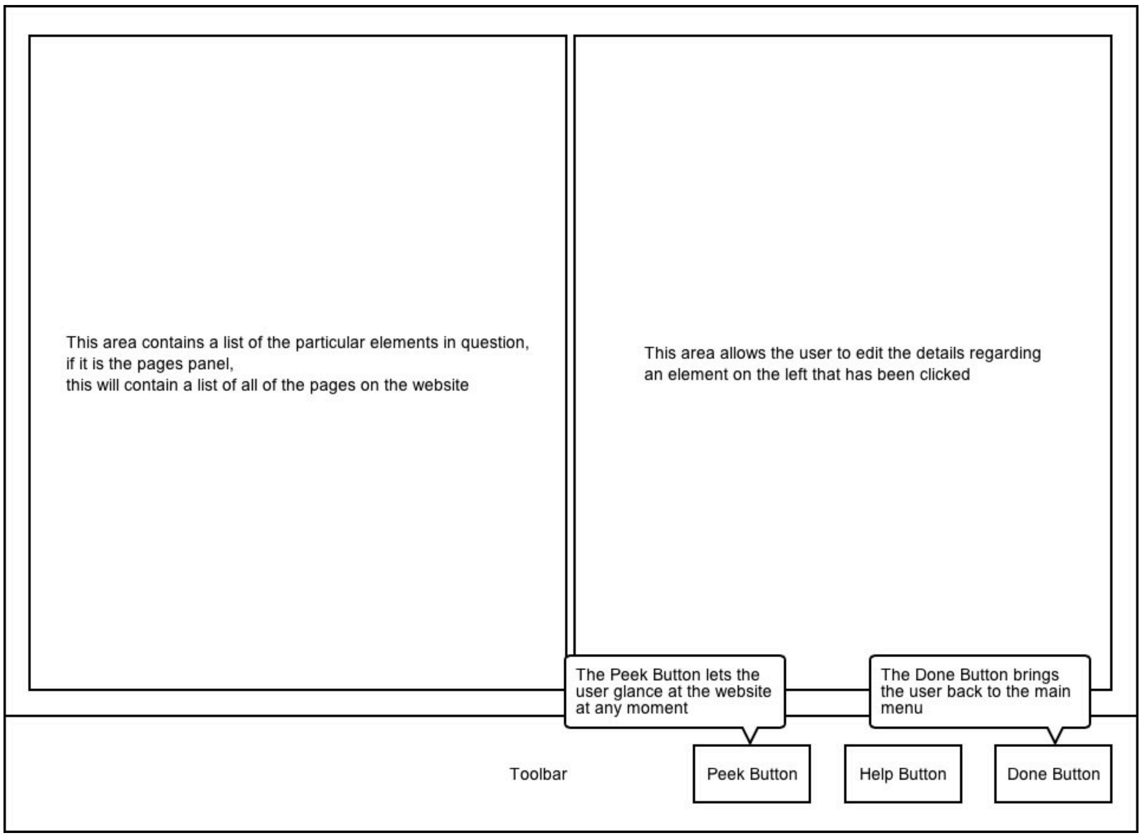
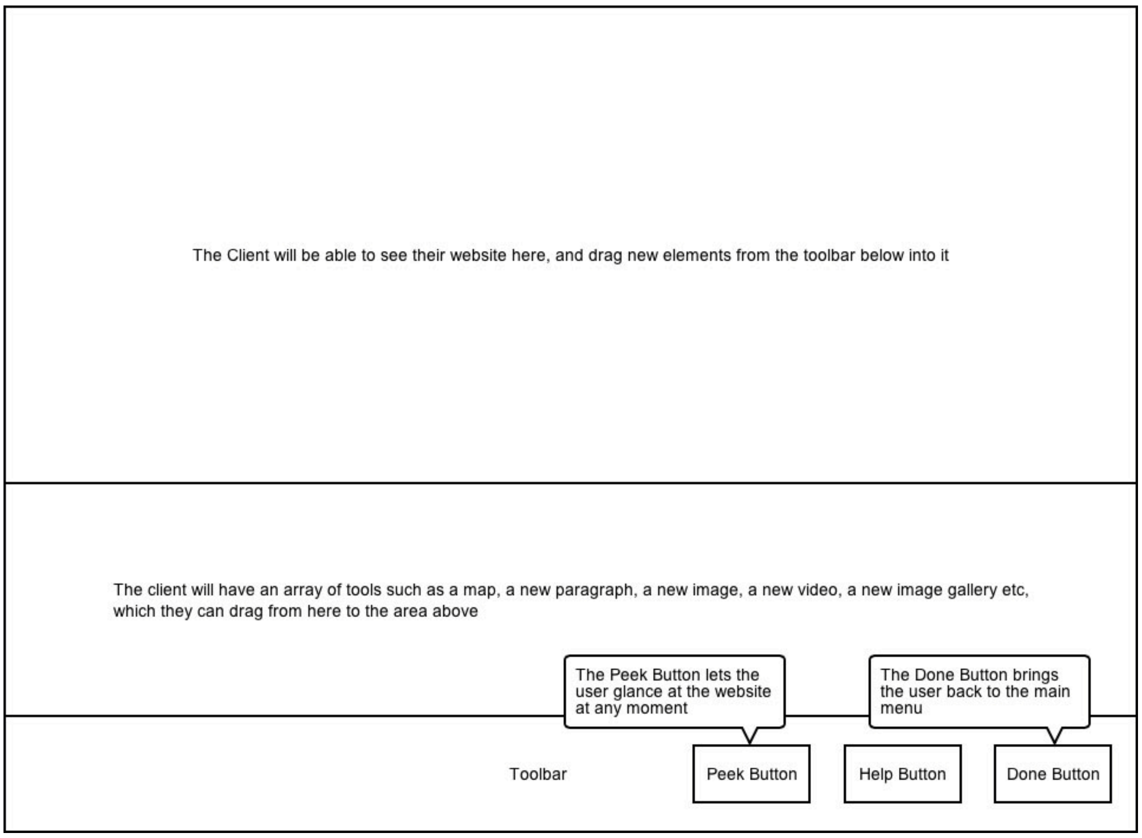

This area contains a list of the particular elements in question, if it is the pages panel, this will contain a list of all of the pages on the website
This area allows the user to edit the details regarding an element on the left that has been clicked
The Peek Button lets the user glance at the website at any moment
The Done Button brings the user back to the main menu
Toolbar
Peek Button
Help Button
Done Button

Figure 3.3 shows the layout format that each of the panels will follow; at the moment the panels that use this design include Pages and Settings on the Client's side, and Website, Layouts, Editing Tools and Privileges on the Designer's side.

Figure 3.4: Wireframe


The Client will be able to see their website here, and drag new elements from the toolbar below into it
The client will have an array of tools such as a map, a new paragraph, a new image, a new video, a new image gallery etc, which they can drag from here to the area above
The Peek Button lets the user glance at the website at any moment
The Done Button brings the user back to the main menu
Toolbar
Peek Button
Help Button
Done Button

The Edit Panel on the client side of the system is customised somewhat from the rest of the current panels as it has one part which is the website, and one part which is the array of tools which can be used to edit the content on the website. See **Figure 3.4**

2.4. Feasibility testing

2.4.1. Review of Aims

Aim 1 - *Offer web developers an adaptable Content Management System for their clients.*

This is the overall aim of the system, and so far is going to plan. With the ideas and considerations that were realised through section **2.2. Requirements Specification** and **2.3. Paper prototyping**, it has been possible to plan out what the system is going to look like, and understand more clearly in how the system will operate once it has been built.

Aim 2 - *Give the client control over their website without compromising its design.*

Through section **2.3. Paper** prototyping, this part of the system has been visualised, and an idea of how this will translate into a working part of the overall system can be understood.

Aim 3 - *Create an end product that is commercially viable.*

This aim is long term and may be outside the grasp of the time constraints of this project, through the use of a promotional website this could be pursued; but at this stage it's simply too early to tell.

Aim 4 - *Challenge and develop design, PHP and jQuery skills.*

This aim is continuous throughout the whole project, and only the end result will tell if it has truly been realized, but from the stages of the project that have been completed currently, it is clear that these skills will be tested and advanced.

2.4.2. Review of Objectives

Objective 1 - *Establish the system features and specifications before design commences.*

This objective has been well and truly completed. The main aspects of the system have been figured out on paper through section **2.1. Idea Generation**, **2.2. Requirements Specification** and **2.3. Paper prototyping**.

Objective 2 - *Design and refine how the user interface will look before creating the program.*

At this stage only the paper prototyping part of the design phase have been completed; the Photoshop comp will be the next part of this process and will detail what the UI elements will actually look like.

Objective 3 - *Build and optimise the user interface and components to work as the design intended.*

Objective 4 - *Test run the system to identify pitfalls and bugs to rectify them before completion.*

These two objectives have not yet been realised, so no comment can be made on their progress.

2.4.3. Review my project Scope

So far the project scope is coming together nicely, bar a few aspects relating to Task 2 in **Figure 1.5**.

"2.2 - Design and build the database that will store data for the CMS"

It was envisaged that Task 2.2 (shown above) especially could produce problems if it was not done on time, however so far the project has been able to progress without it. It's possible that the project could continue without it until the UI design has been completed, as this will clarify what tables and relationships will be needed in the database.

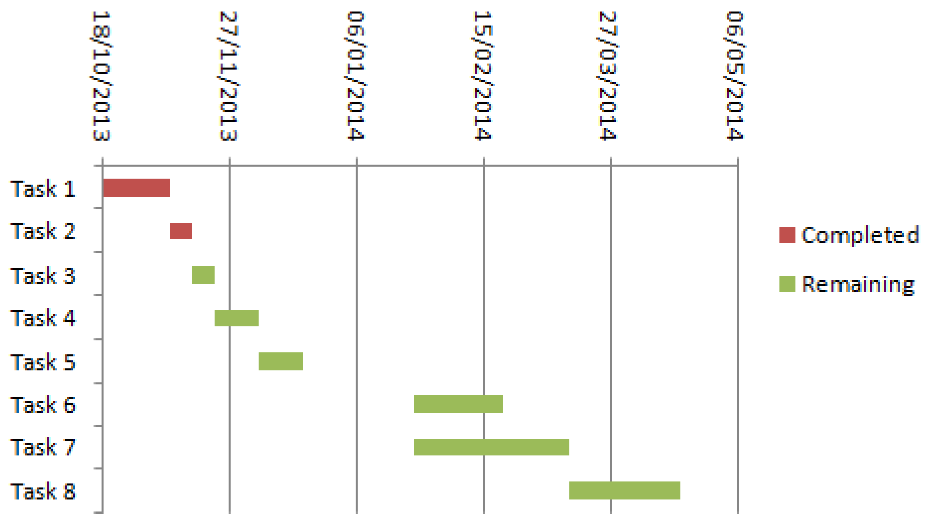
"Storyboard how operations will be carried out"
"Design the file and folder structure"

This is the same for two parts of Task 2.1 (shown above), in which case it won't be until the build phase that they will be properly clarified.

2.4.4. Gantt Chart

At present **Figure 3.5** shows how the project schedule is shaping up.

Figure 3.5: Gantt Chart

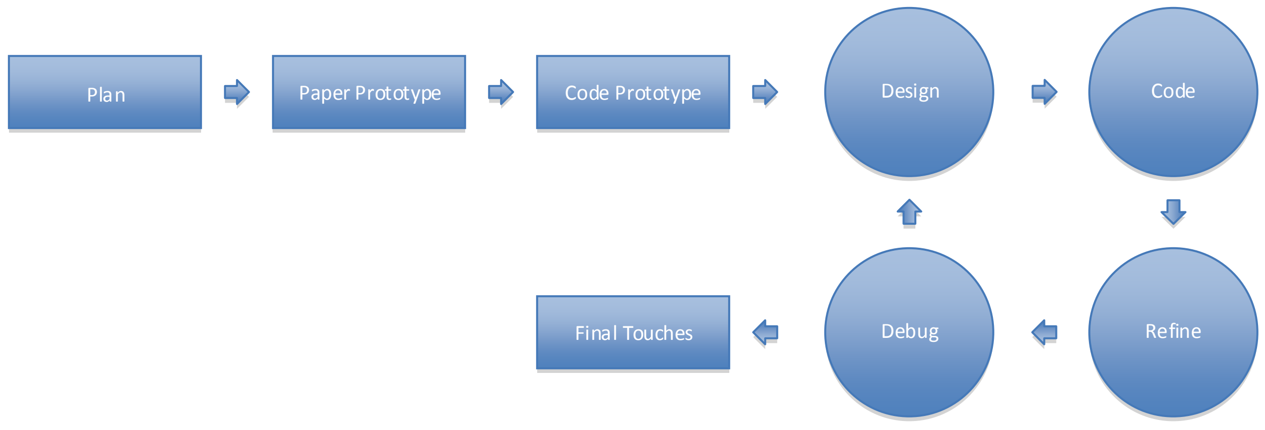


2.5. Methodology selection

A mixture of two methodologies will been used for this project to ensure it runs smoothly at each stage. Up until the prototype has been developed the waterfall methodology will be used. The waterfall methodology is useful for when the direction of development is fully understood, in the case of this project, the first three tasks must be completed in their respective order before the project can go further.

After this point an adapted rapid application development methodology will be used so that the systems design and its functionality can be debugged, refined and tested before final implementation. Below is how this hybrid model looks and works within the project.

Figure 3.6: Methodology



3. Design

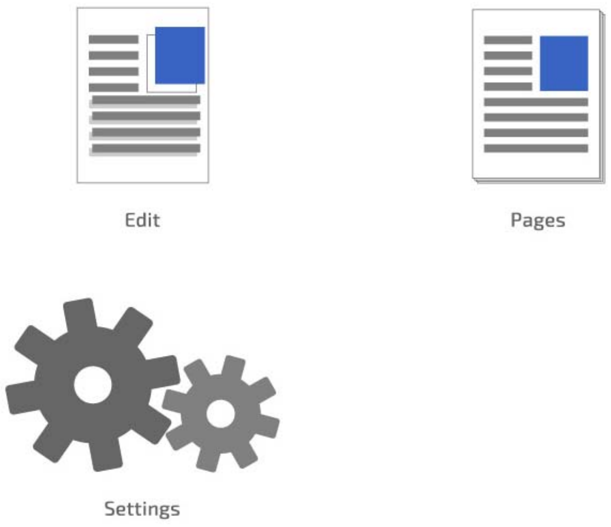
3.1. UX design evolution

3.1.1. The Visual Design

Over the course of the project, its final outcome has changed from what would have been Dreamweaver design view in the browser, to what is now a fun, tasteful website content editor.

After seeing six-ups and one-ups demonstrated, it was possible to come up with some ideas of how the project would look and feel. It was through this process that the decision was taken that the home menu would be an icon-style grid of options, and after playing around with several menu styles, this would be the simplest and fastest option for getting the user to the information they need. (see **Figure 3.7**)

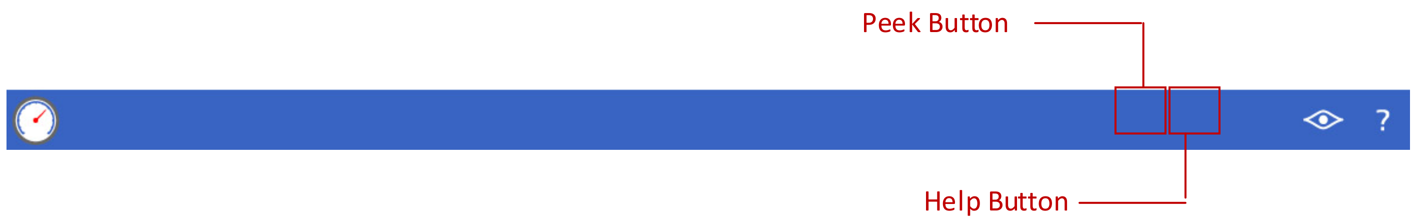
Figure 3.7: Site Commander Main Menu



The long term goal for the menu system would be to allow the designer to add their own custom functions to it. Should this goal be realised, the menu would take on a more ‘app grid’ feel where the user can click left or right to view more of the CMS functions.

It was also the paper prototyping stage which produced the idea to include both a ‘help’ and a ‘peek’ button. (see **Figure 3.8**)

Figure 3.8: Site Commander Taskbar



The help button will give the user prompts based on the task they're currently engaged in; this should help decrease the clients reliance on their designer/developer by giving them a way to help themselves.

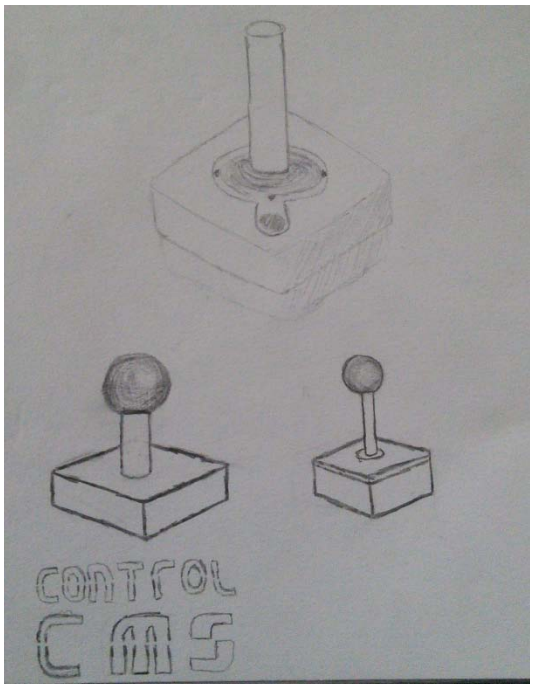
The 'peek' button will allow the user to hide the current panel so that they can glance at their website no matter where they are or what task they're doing, clicking the 'peek' button again will show the current panel again. The hope is that the client can use this to glance at information on their website without having to leave their current browser tab.

From the initial conception, It was the intention that the client would have their website in view at all times, but as the concept developed, it became apparent that this was neither practical, nor was it necessary. For the current design, the idea of having 'panels' was implemented; each panel is used to perform a different task, some of these tasks require the website to be in view, while other tasks simply don't – the 'peek' button has also eliminated the need to share the browser window with the website and the CMS for all but one task.

3.1.2. Branding

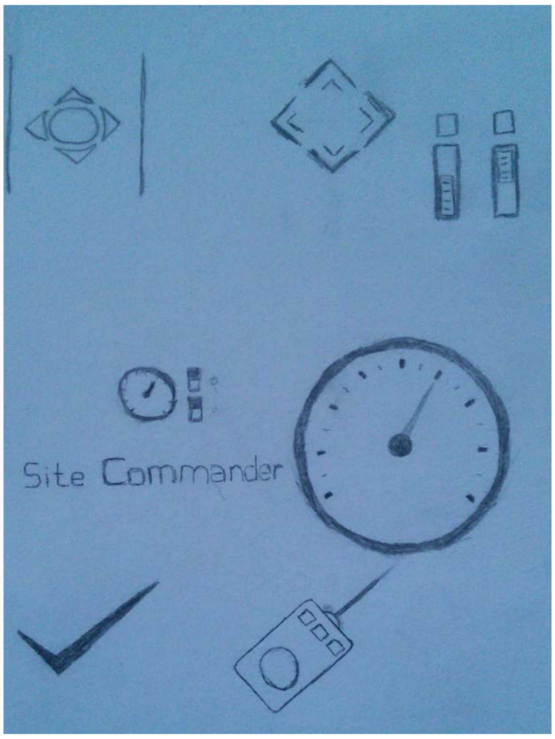
When the project idea was pitched at the Idea Generation Workshop, it was assigned the word **control**, this word helped to consider the nature of both branding the product and designing the system. The brand needed to be corporate to show trustworthiness, but also fun. The initial logo design was an arcade joystick and an arcade-style font for the name of the product. When this was tested both on paper and in Adobe Illustrator, it emphasised fun more so than corporate. (see **Figure 3.9**)

Figure 3.9: First Brand Sketch



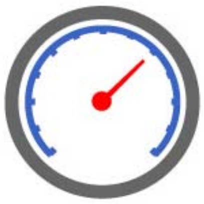
In order to create a more corporate-heavy brand, the word control had to be looked at in a more abstract way, the different forms of control including buttons, switches, remote controls, car dashboards and even symbols which represent control were considered. (see **Figure 4.0**)

Figure 4.0: Abstract Control Sketches



The car dashboard idea stood out as the project is designed as kind of dashboard for websites. After sketching the idea on paper, I drew it in Adobe Illustrator. (see **Figure 4.1**)

Figure 4.1: Site Commander Logo



The decided font for the branding is called Exo 2 (see **Figure 4.2**); it describes the product very well visually as it incorporates the fun, game like attributes from the first branding idea, but it also looks confident and smart enough for a professional product.

Figure 4.2: Exo 2 Font

Site Commander

It was then paired with Open Sans (see **Figure 4.3**), which will be used for the minimal amount of paragraph text throughout the system; Exo 2 will be used for headings.

Figure 4.3: Exo 2 Font paired with Open Sans Font

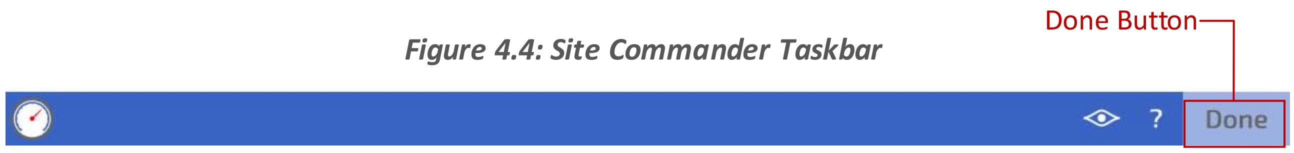
Heading Text

Paragraph Text

3.1.3. Navigation

The navigation for the system hinges on two parts, the main menu and the taskbar at the bottom.

Figure 4.4: Site Commander Taskbar



When the user clicks an item from the main menu, the menu will disappear, and the panel for the item they have clicked will appear in its place; a new button will now be visible on the taskbar called 'Done'. When the user has finished the task they are engaged in, they simply click this button, their changes are saved, the current panel disappears along with the 'Done' button, and the main menu reappears. (see **Figure 4.4**)

In my original design (shown in **Figure 1.7** in section 2.3. **Paper prototyping**), the user interface didn't have a designated homepage; this meant that there was a lot of information in view that wouldn't necessarily have helped the user. This would have made it very easy for the user to get lost when navigating the interface.

By adding a taskbar, the user now has a central point to refer back to; in other words, if the 'Done' button is visible, it will bring them back to the main menu, if it's not visible; they're at the main menu panel.

3.1.4. Interactions

In order to make the experience as seamless for the user as possible, all of the processes take place without a page refresh; this has necessitated the use of transitions to help changes look graceful, rather than abrupt. All of these transitions will use jQuery UI, as this works across multiple browsers.

I have planned to use the 'Explode' and 'Implode' effects to hide and show the main menu, and the 'Fade In' and 'Fade Out' effects to show and hide panels.

In order to do away with page refreshes, all of the information has to be loaded onto one page through JSON, which has been coupled with PHP so that the system can communicate with its database.

To ensure that the user can access information quickly when they login, all of the information for each of the panels will be loaded at the first page load; as the user makes changes to their website, these changes will be passed to the database through JSON.

This could cause problems if the user accidently changes something, so it is intended to include an undo feature so that the user can go back if they have made a mistake.

3.1.5. Information

The taskbar is where the user can see key information about the task they're currently engaged in. Notifications will appear here as their changes are applied, they can also seek help in relation to the current task by clicking the '?' button.

3.1.6. Personas

Working off Mail Chip's idea of having characters to define what different people are using the tool for; a set of Characters has been created for Site Commander. Site Commander has two login sides, Developer View which contains all of the tools for adding new features for the non-technical users; and Client View which contains all these tools to make changes to the website using a GUI (no code).

Scenario – Online Shop

The scenario has several users with different mindsets working together. With a variety of needs, each of these individuals requires different things from the application; below are a set of user focused needs regarding the system.

The Development-Side Team

Developer: Motivated, Smart, Reliable, Caffeine

Developers don't want to be concerned with the user interface so much as they do how the program executes; this means that consistency in the coding of the application is paramount. A consistent coding style throughout will also be a good basis for creating a framework for the application which a developer can understand.

Designer: Collaborative, User Focused, Creative, Aesthetic

A designer will hate building for an application that is not carefully thought out, therefore it is vital that the layout is thoughtfully considered with non-technical users considered. They will want to know the best ways to integrate new sections into the system, so pattern libraries will be essential.

The Client-Side Team

Managing Director: Controlling, Self Motivated, Profit, Professional

Managing Directors will want to see their most important assets catered for to the highest possible standards. An MD who runs an online shop will need to have access to their website to make changes where they deem appropriate, but if they have hired a technical team (like this example), it is likely that they are not technically minded, and will not want hassle every time something needs added, updated or changed. This means that the default toolset needs to be carefully designed for non-technical users.

Marketing: Resourceful, Knowledgeable, Researcher, Diligent

If marketing staff are employed, they will need to be able to report to the MD not only with ways to improve website traffic, but also with figures for the latest traffic. This will require some way of plugging in analytical data. This will require a new panel which has not yet been considered.

3.1.7. Design Principles

Modular

Site Commander's system architecture is designed so that the developer can easily attach new tools to the current installation as well as removing old ones.

Efficient

It is important that the program runs quickly so that a user can complete their tasks in a minimal amount of time.

Flexible

The system will be able to change based on the device they are using, and will give them ways to make the application more suited to their working style or the task they are performing.

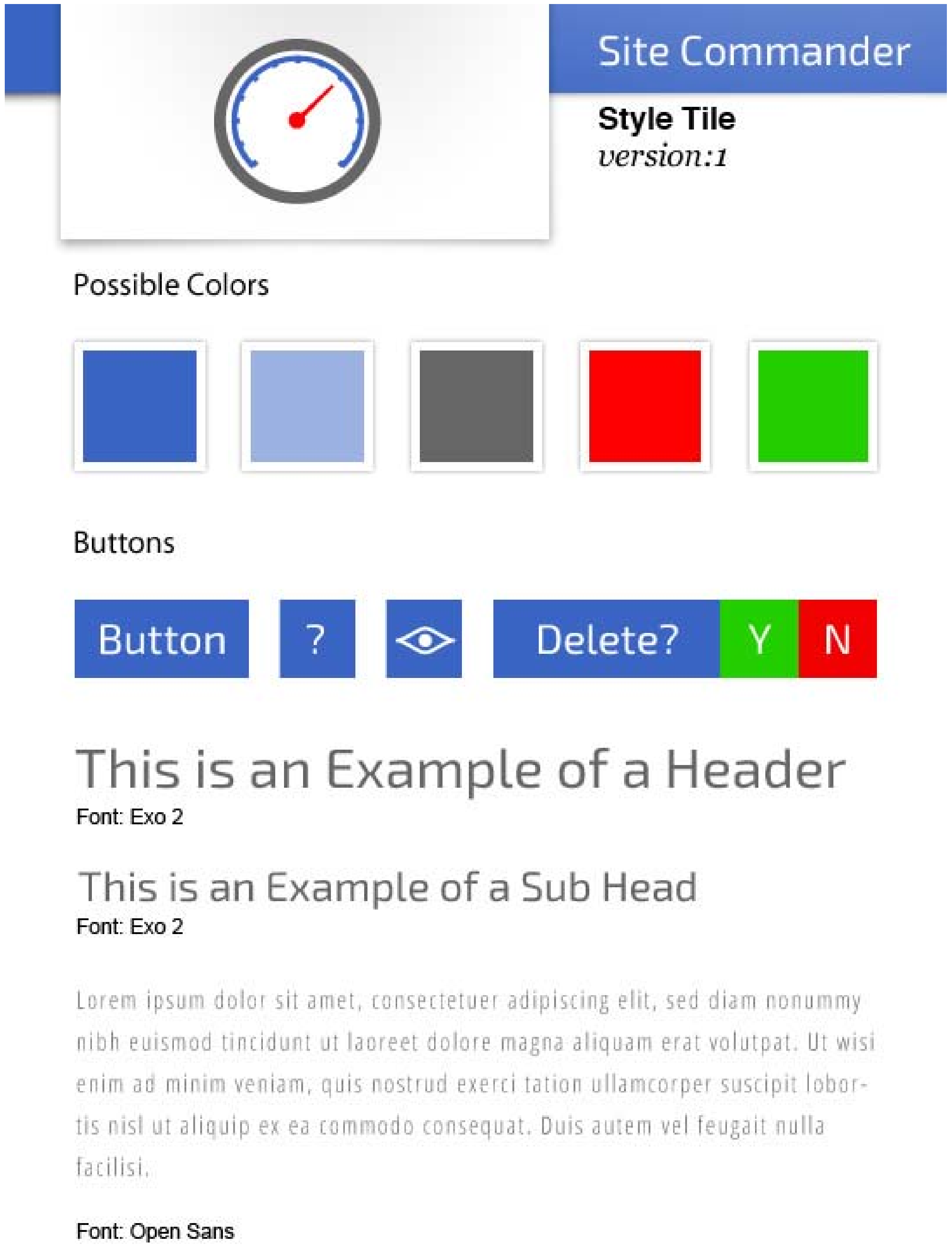
Sustainable

Even after many years of implementation, the system should still be capable of receiving updates and modifications.

3.1.8. Voice and Tone

The application does not use a lot of text based dialog because it is driven mainly by icons; where there is text, the user will be given very short, snappy instructions. Where more detail is required regarding a particular label, the help button can be clicked to give detailed instruction regarding the current screen.

3.1.9. Design Patterns



Site Commander

Style Tile

version:1

Possible Colors

🟦 🟦 ⬛ 🟥 🟩

Buttons

Button ? Delete? Y N

This is an Example of a Header

Font: Exo 2

This is an Example of a Sub Head

Font: Exo 2

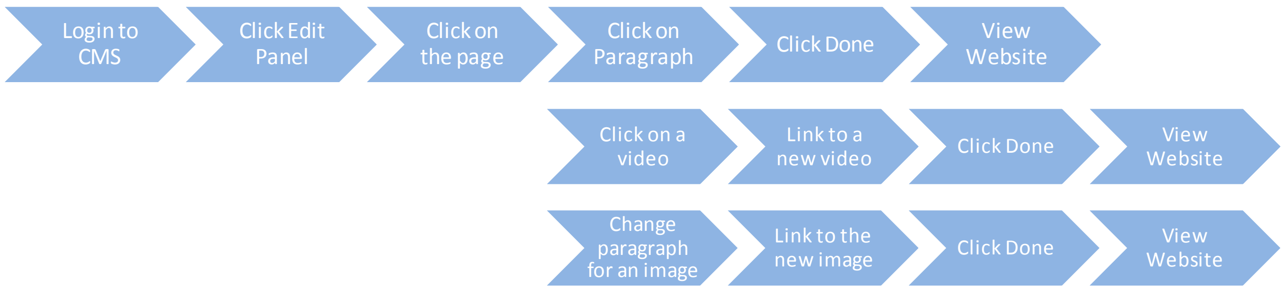
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel feugait nulla facilisi.

Font: Open Sans

3.1.9. User Journeys

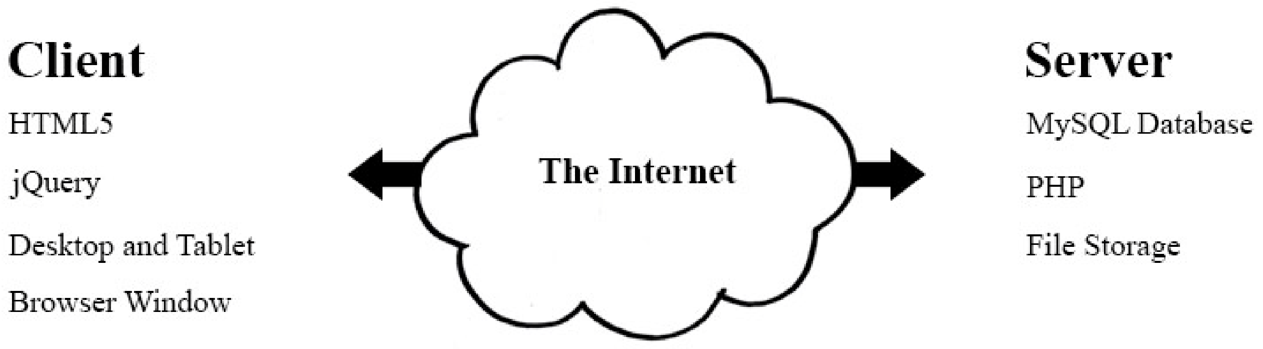
When a user goes to make a change, they can quickly engage in another task as shown in **Figure 4.5**.

Figure 4.5: User Journeys



3.2. System design

3.2.1. Client-server model



3.2.2. Design Patterns

To make the system run as efficiently as possible, a generic process will be developed to handle each component of the system. Every page tool that the client can use will communicate with the UI and the database in the exact same way; this will mean that each of the page elements created by the user will be stored in the same table, with the same column headings.

This will be difficult as page elements can be so diverse (eg: text from a paragraph is completely different to 6 image file names from a photo gallery). If the data is stored as a string that can be broken up, a generic process will be achievable.

Other sections of the system will be generic because the data they are storing does not vary in the way that page elements do; for example, the pages will be stored in the database with their Meta details and their names, this data will not vary in form.

As shown at the Paper Prototyping stage, the layout will be consistent for both the designer and the client, this will allow for the reusing of HTML, CSS and jQuery elements for the UI.

One of the challenges that will need to be faced as the project advances is how to make the system work proficiently, as well as seamlessly for any website that is plugged in. To achieve this, a framework for the page content layouts will need to be created.

3.2.3. Editable Content

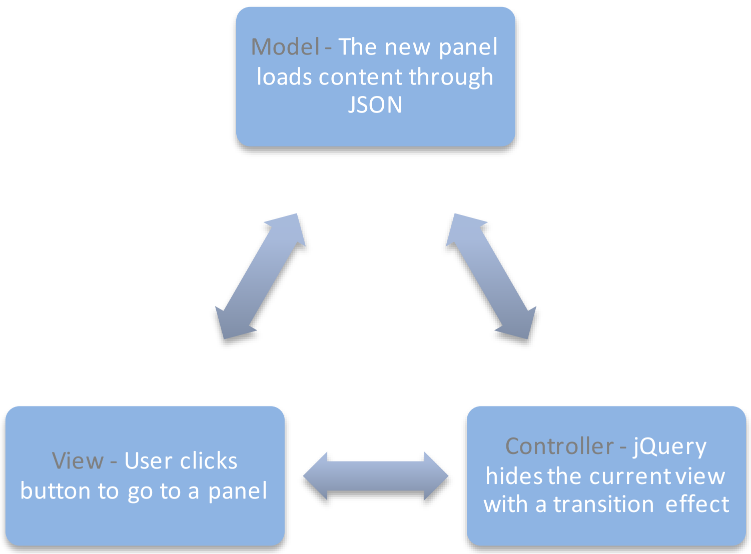
In order to allow content to be edited live, I used jQuery to convert text items into text boxes, I have since learned of an HTML5 attribute called ‘contentEditable’ which makes it possible to directly edit the data in almost any tag and so far with use appears to have good browser support. This has saved on jQuery code in the script files and has enabled the editable webpage to look identical to its live version, which was the original idea.

In order for the system to know what can be edited and what cannot be edited, the developer is required to add the class ‘scdr-editable’ to every element of the HTML that content can be added to.

3.3. Logic design

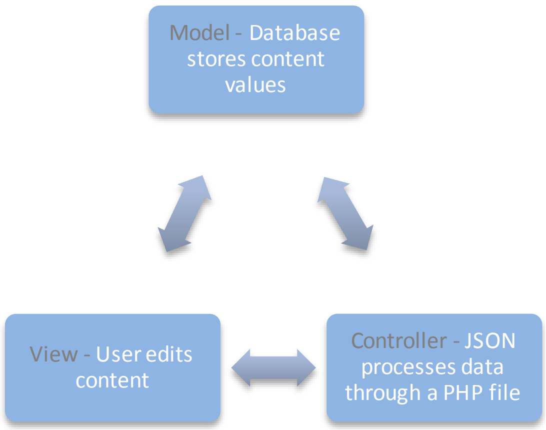
3.3.1. Model-View-Controller

Figure 4.6: Travelling through the User Interface


Model - The new panel loads content through JSON
View - User clicks button to go to a panel
Controller - jQuery hides the current view with a transition effect

1. The **view** here is the home screen of the CMS, the client will be able to click on a panel icon to navigate to the task they wish to complete on their website.
2. The **controller** here is jQuery, which will allow a transition effect to take place between the area that the client is coming from and the area that the client is going to. Some JSON may be executed at this point so that a lot of data does not have to be loaded when the user first logs in.
3. The **model** here is JSON, only when the transition takes place will data be loaded into the new panel for viewing.

Figure 4.7: How content is stored in the database



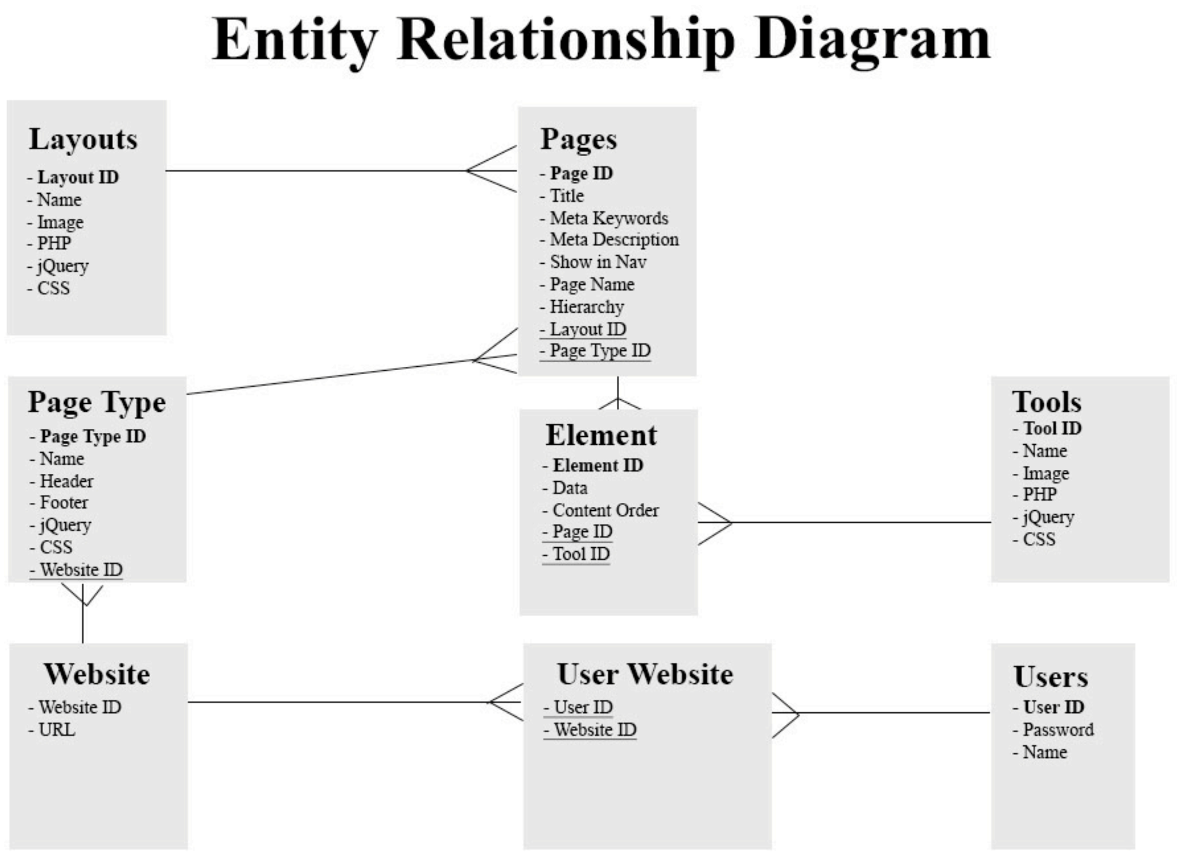
Model - Database stores content values
View - User edits content
Controller - JSON processes data through a PHP file

1. The **view** here is the UI that the client is provided with to make changes to content; it will be possible to simply click on editable content to start making changes.
2. The **controller** here is JSON, a part of jQuery that allows data to be sent to, or retrieved from a PHP file without leaving the current page that the user is on – this will allow for producing seamless updates to the clients website.
3. The **model** in this case is the database itself, which is where all of the data input by the client from the **view** is processed to through the **controller**.

3.4. Data design

3.4.1. Entity Relationship Diagram

Figure 4.7: Entity Relationship Model



4. Implementation

4.1. Technology/tool selection

I have chosen the following technologies to be used in the creation of the product because they are commonplace on the web, and are supported by a wide variety of environments such as servers and web browsers.

- HTML5
- jQuery
- MySQL Database
- PHP

4.2. Technology/tool use

HTML5

With the CMS running inside the web browser, HTML is the mark-up I have chosen for displaying information the user. To ensure that the system is as future proof as possible, I will be coding the systems layout in HTML5, the latest version of HTML.

jQuery

The nature of my concept necessitates the need for the CMS to feel as seamless as possible so the user experience cannot be interrupted by page loads. jQuery's AJAX JSON extension will allow a lot of important communications to take place between the MySQL database and the user interface.

MySQL Database

The MySQL Database will store all of the updates which are made to the client's website, as well as connections to all of the tools that will assist the client making those changes.

PHP

To provide communication between the database and the user interface, I have chosen PHP as my server script. PHP will enable content that the user changes to be updated to the database through SQL, files such as images or videos to be uploaded onto the server, and the websites content to be sent to the user interface.

File Storage

Although file storage is provided by the server (which makes it entirely up to the developer how much file space they provide); I felt it was important to mention this as the ability to upload file formats such as images, videos, audio, PDF’s, etc. needs to be provided through PHP. This ability will be made available through the use of third party source code such as PHP File Uploader (www.phpfileuploader.com).

4.3. Notable challenges

Creating the website editing tool combines an HTML interface with a jQuery script file to handle the client side user interactions, and a PHP processor to save the users changes to the database.

The HTML File

The interface for editing content is installed on the system as a panel that can be accessed from the menu screen. This panel is a set of list items with icons for each of the tools that can be dragged onto the content area of the website.

The jQuery File

This is the most vital link in the chain for the editing tool to work. In fewer than 400 lines of code this file handles every interaction that the user can make with the content. These interactions are:

- Drag and drop new elements onto the content area
- Change where each of the elements appear on the content area
- Edit text, change an image, change a video, change a maps location, add more images to a gallery
- Send all of the layout data to the server side for processing to the database.

The most challenging part of this tool was getting each of the elements to move around on the content area. To do this I used the jQuery *.wrap()* function to put each content element into a list item, the wrap function was then used again to group the content elements in each section into an unordered list.

These unordered lists were then given jQuery *.sortable()* functionality which allowed each of the list items within them to be clicked, dragged and reordered. In order to save the arrangement of each of the elements when the user exits the content editor, a while loop finds each of these sortable list items and adds a class containing the numerical section it is currently in, and the order in which it appears within that section. This class is then captured and, along with all of the other data for that element, is stored in a string variable which is then passed through a JSON request to a PHP file where the arrangement data is listed out and stored on that content's record in the database.

The PHP File

Without PHP the editor would not be able to store the website's content in the database.

4.4. Notable achievements

Storing data with no page refreshes

The beauty of Site Commander is its ability to transfer and store data into a MySQL database without a single page refresh; this allows for an uninterrupted user experience.

Using a vital mix of jQuery scripting and PHP processing, Site Commander is able to save the users changes almost without them noticing as a result of the JSON link between them.

JSON allows server files such as PHP to run without needing to leave the current page, once a server file a finished running, JSON is able to retrieve its processed data and output it on the current page.

Detecting a layout file

Site Commander is capable of detecting if a page layout can receive content from the user. When a page is loaded up in the CMS, it's layout file is scanned for anywhere that it contains the class 'scdr-editable'.

Each occurrence is rewritten with an id called 'scdr-editable-' followed by a number; the numbers start at 1 and for each occurrence that is replaced, this number is incremented by 1.

The database can then be checked to see if any of the content in the database corresponds to the page and section ID's.

5. Testing

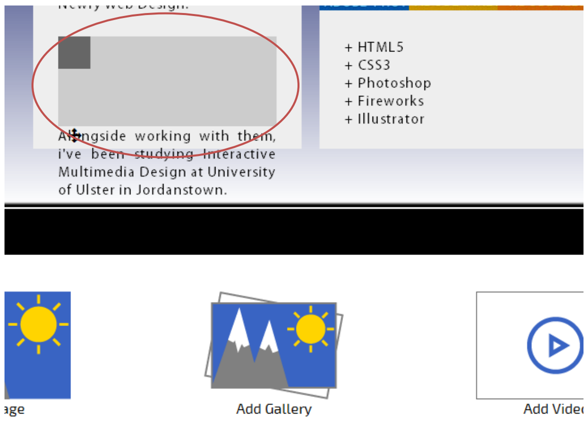
5.1. Testing approach selection

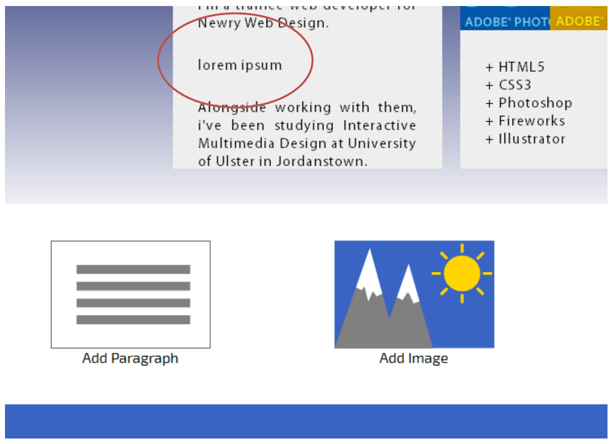
The main function of the system is to edit content, so this is the area of the system which is going to be tested out.

- Test each tool to see how well they work
- Test rearranging content on a webpage
- Saving Changes to the database

5.2. Testing process

5.1.1. Test each tool to see how well they work

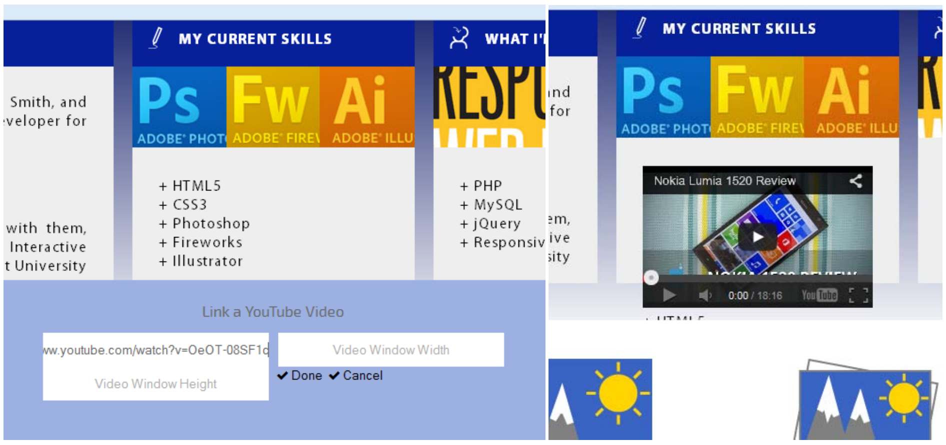

Alongside working with them, i've been studying Interactive Multimedia Design at University of Ulster in Jordanstown.


Newry Web Design.

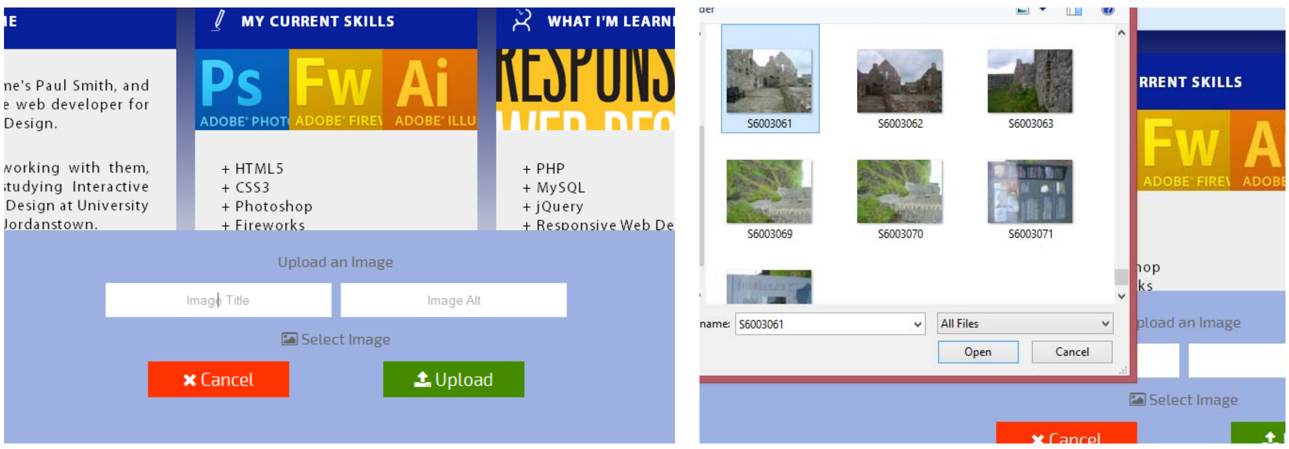
lorem ipsum

Alongside working with them, i've been studying Interactive Multimedia Design at University of Ulster in Jordanstown.

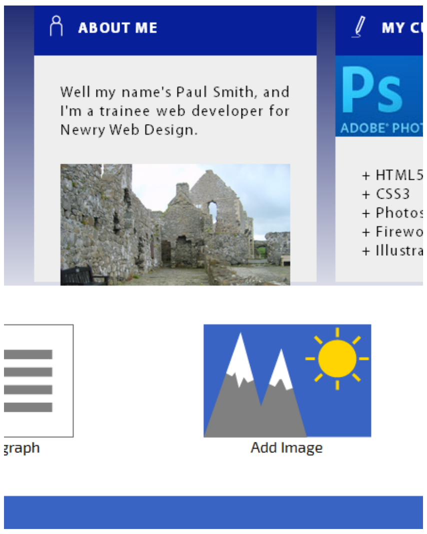
Above you can see a test of how text is dropped onto the website layout; when the text icon is dragged over, a placeholder appears to let the user know that they can drop it there. The second image above shows the end result with a few words to the user started; all that's required is a click to edit this text.



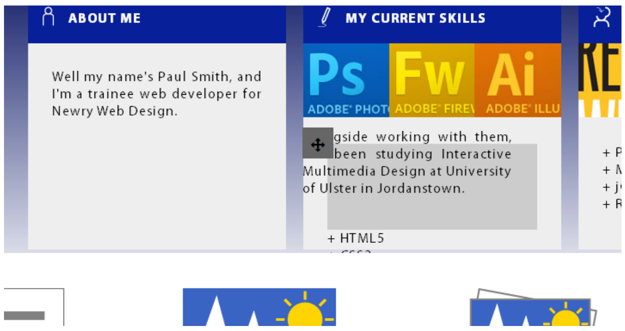
Adding a YouTube video is also an easy task, a link to a video can be directly pasted into the video tool and the YouTube video will be added directly into the content.



Pictures are also a simple upload process; with the simple drag of the icon, it is possible to add an image to the content quickly and conveniently straight from the computer, or if on a mobile device, from its camera. Below is the final result

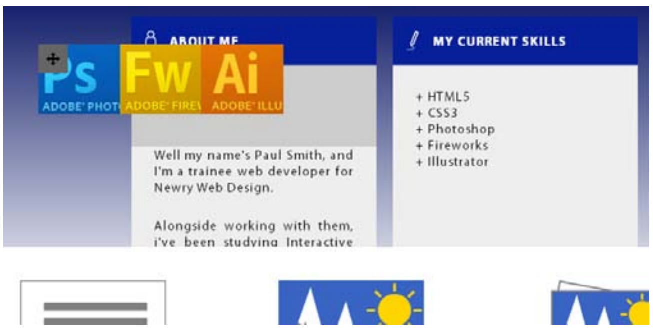

ABOUT ME
Well my name's Paul Smith, and I'm a trainee web developer for Newry Web Design.
Add Image

5.1.2. *Test rearranging content on a webpage*



Well my name's Paul Smith, and I'm a trainee web developer for Newry Web Design.

gside working with them, been studying Interactive Multimedia Design at University of Ulster in Jordanstown.

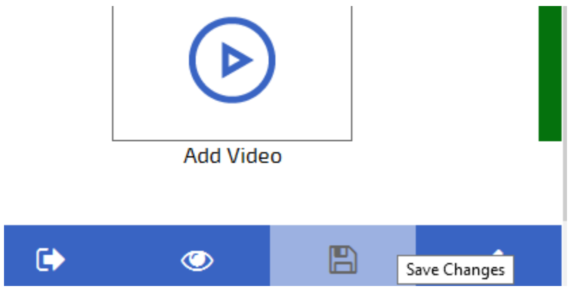


Well my name's Paul Smith, and I'm a trainee web developer for Newry Web Design.

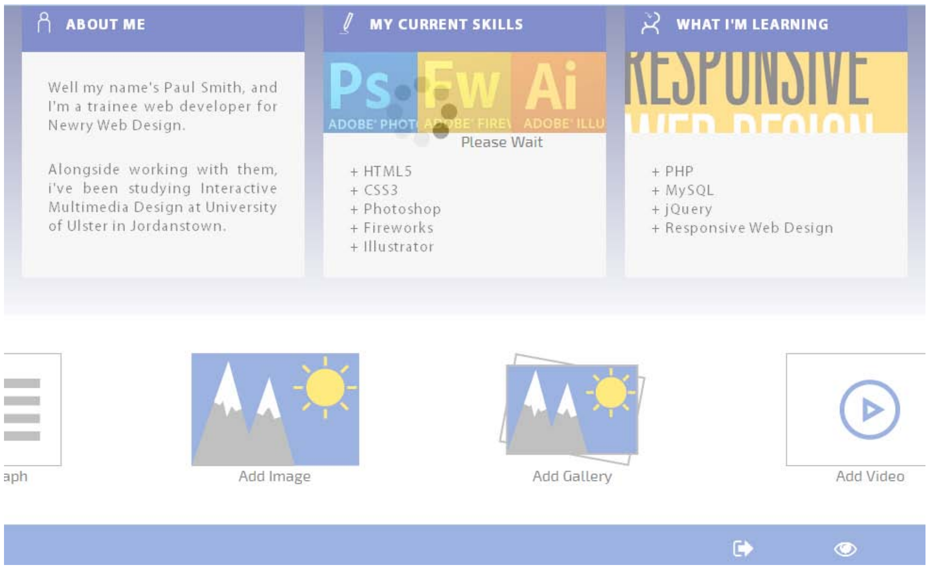
Alongside working with them, i've been studying Interactive

Here are some tests showing content being moved from one location to another, note the handles which give you a marker to click or touch.

5.1.3. *Saving Changes to the database*



Above you can see the save button which allows all of the changes that a user makes to be stored in the database. When this is clicked, a loading overlay fades in to show the user that a process is taking place; this is shown below.



Well my name's Paul Smith, and I'm a trainee web developer for Newry Web Design.

Alongside working with them, i've been studying Interactive Multimedia Design at University of Ulster in Jordanstown.

5.3. Test results

5.3.1 Test each tool to see how well they work

The tests have shown that creating a highly user friendly experience on a CMS is possible, and offers clients a very rapid update tool for their online content.

5.3.2. Test rearranging content on a webpage

Rearranging content also proved easy, however more tightly packed components do show the limitations, of this interface, but should inspire new ideas for overcoming such difficulties.

5.3.3. Saving Changes to the database

Saving to the database works like a charm, but at the moment all of the data on the page is collected into one string and is processed through PHP to update each content record, so it means that potentially 20 to 30 records are currently being updated unnecessarily. An update to this functionality would be to detect which content records have been modified, and only upload them instead of everything.

6. Evaluation

6.1. Evaluate test results

6.1.1. Test each tool to see how well they work

From the tests, it easy to see the potential for expansion of this toolkit, but there are some issues; one such being when dragging a tool from off of the toolbar, it often doesn't sit with the cursor as it moves around the screen as it tends to jump away outside of the visible browser window, this should serve as a reminder that the project still needs much tweaking and refining. It should be noted however that this does not affect the operation of the drag and drop mode as content will still land where the cursor lands.

6.1.2. Test rearranging content on a webpage

One major difficulty with arranging content in a space like this is the difficulty to pick up smaller elements, especially like the small icons next to the headers, which are almost completely inaccessible without moving the text next to them.

6.1.3 Saving Changes to the database

Saving to the database works like a charm, but at the moment all of the data on the page is collected into one string and is processed through PHP to update each content record, so it means that potentially 20 to 30 records are currently being updated unnecessarily. An update to this functionality would be to detect which content records have been modified, and only upload them.

6.2. Evaluate project outcomes

The project demonstrates the potential of a dynamic and seamless content management system, but requires a lot more polishing to become a commercially viable tool. Overall the outcomes of the project could have been larger, possibly the scope was too large and incorporated too many features to be achievable in the time frame.

6.3. Evaluate the methodology

Using a mixture of waterfall and rapid application development proved useful for this project as during the later stages refinement was required on a very regular basis. The methodology complemented the nature of the project because almost everything that was planned for the project was going to require new skills.

6.4. Evaluate the plan

The plan for the project would have been perfect if more time were available. It's likely that the aims and objectives were just a little too far reaching for the time frame available.

7. Conclusion

Unfortunately the project did not realise its full potential as each part took more time than was originally allocated to it, this coupled with being too focused on the small problems and not taking the time to try and address the larger ones. The size and scope of the system was also underestimated; I would take a lot more development time to get the project to a refined stage.

It is intended to continue with this project for the immediate future with the goal of either marketing it as a commercial venture, or opening it up for input from others who would be interested.

8. References

David Furfero. (2011). *jQuery UI Touch Punch.* Available: http://touchpunch.furf.com/. Last accessed 18th Apr 2014.

Fort Awesome. (N/A). *N/A.* Available: http://fortawesome.github.io/Font-Awesome/get-started/. Last accessed 18th Apr 2014.

jQuery. (N/A). *N/A.* Available: http://jquery.com/. Last accessed 18th Apr 2014.

jQuery. (N/A). *N/A.* Available: http://jqueryui.com/. Last accessed 18th Apr 2014.

Justin Cook. (N/A). *PHP – parse a string between two strings.* Available: http://www.justin-cook.com/wp/2006/03/31/php-parse-a-string-between-two-strings/. Last accessed 18th Apr 2014.

Matt West. (2013). *Native Rich-Text Editing with the contenteditable Attribute.* Available: http://blog.teamtreehouse.com/native-rich-text-editing-with-the-contenteditable-attribute. Last accessed 18th Apr 2014.

Ravishanker Kusuma. (N/A). *jQuery Upload File Plugin Demo.* Available: http://hayageek.com/docs/jquery-upload-file.php. Last accessed 18th Apr 2014.

Sten Hougaard. (2012). *Custom jQuery selector ":blank".* Available: http://jsfiddle.net/netsi1964/VwPab/. Last accessed 18th Apr 2014.