Title: Major Project

Course: Interactive Multimedia Design

Year: 4

Student name: Christopher Marks

Peer support Group number: 04

Mentor: Richard Davies

Table of Contents

Acknowledgements

Over the course of the last four years, there have been so many people who have helped guide me on my journey. I will cover and credit those while being short and concise.

Firstly, I want to personally thank Dr. Peter Nichol for his guidance and inspiring words over the years. Without his knowledge and expertise, none of the work I have undertaken in this project would have been possible. The COM601 module, which Dr Nichol delivered, proved to be the most vital module I have completed to date and provided me with the beginnings of a learning curve that has taken me up to this point. Being exposed to Ajax and JSON within the course of the module, allowed me to visualise how they could be integrated within my project. Thank you.

I would also like to thank the design lectures Paul McCormack and Tim Potter, for their support over the last number of years. Their lectures always inspired a new way to solve design issues and these practical applications will stay with me on my onward journey.

As a mentor, Richard Davies has helped me maintain a level head throughout this major project. His expertise and guidance, combined with an approachable personality were vital when discussing issues and ideas and proved valuable in helping steer me through some of the challenges of the project. He provided inspiring words during this time and these words and teachings will stay with me in my career.

I would also like to thank all the teaching staff in UUJ, specifically the Interactive Multimedia Design staff who have created an excellent course. I have found the last 4 years of learning very enjoyable and I will miss this greatly when I leave.

A personal thank you goes to Symfony2 developer Philippe Lagnas, who has provided a knowledgeable ear when questioning the Silex PHP framework. He gave valuable feedback about his own programming experiences, and shared valuable resources that have helped to further develop my working knowledge of PHP. Without his kind words of wisdom and encouragement, the journey of Projectified would have been a very different experience- no more spaghetti code!

1. Introduction

Prior to retraining as a web designer, I had spent a few years working as a Methods Engineer, planning the workflow in a local engineering company. It was during this early period in my career, that it became clear the extreme importance that time management and job prioritising were in business- vital to ensure a streamlined workflow for employees in the company. My time in this role gave me an in-depth grounding in this area, and I was able to learn first hand the value of structure within daily work processes. When millions of pounds were at stake, accuracy, precision and working to deadline had to be planned effectively and this planning was the lynchpin of the business.

I witnessed occasions when staff failed to plan and document things correctly and how overall this led to inadvertent outcomes, having detrimental consequences to parts of the project. It was also during my time in Cronin Designs on my placement year that I witnessed how the necessary structure and organisational planning which was crucial for success in a small team could be a struggle for some, and how counterproductive a lack of scheduling was in SME terms. I thought about how I could create a framework for people to work within in a company and manage projects and deadlines effectively and this provided me with the initial brainstorming that became the basis of Projectified.

While analysing the market of project management tools, it became apparent that there was a gap for a simple and more intuitive user experience product. Some project management tools on the market lacked clarity for the user and appeared confusing, demonstrating a lack of understanding of some basic user experience techniques. This therefore demonstrated that there was room to: devise, measure, develop and produce a structured interface that would be of use for people when planning projects and allow them to input data in an organised manner and provide them with a framework to work within. In creating this programme I could tap into my current skill set adopting an array of techniques learnt over these last four years to develop a user-friendly option for the market, and solve some of the issues that I had witnessed first hand in both large and small businesses.

1.1 Project Aim

The primary aim of Projectified was to provide a project management tool for a group collaborative project to effectively manage deliverables. In this project it was decided that the main focus of the project should be from a single user perspective, to effectively communicate how the programme works from an in-depth and individual analysis, which could in time expand and be further rolled out to cover more users and groups.

When designing and developing this tangible web application, the main aim was that it would be intuitive, dynamic and harmonious: the main focus would be on allowing the user to create, read, update, delete and manage their chosen projects by organising goals and objectives into manageable boards that are easy to engage with, navigate and maintain. It had to be a seamless interaction platform that would save individuals and especially small businesses their limited time and resources

1.2 Project Objectives

Projectified's objective is clear and unambiguous: to deliver a crafted and robust solution that encompasses an attractive and intuitive UX experience front end with a database back end that can pull data to structure the project effectively. This will make it easy to create, read, update and delete information freely, triggered by the user. Projectified will be live on the 18th April 2014. To assist with this objective "Projectified" will be hosted online on its own domain:

1. Projectified web server, http://www.projectified.com

1.2.1 Measurable Objectives

To assist with effective reporting, proven project management methodologies were adopted in order to meet requirements and deliverables effectively. Using a time-based application called OfficeTime, meant that tasks and their duration could be tracked accurately. In constructing a PERT analysis table, this enabled the project to be dissected into smaller manageable pieces, allowing for accurate reporting and updating. By clearly highlighting the maximum and the minimum time needed to complete different parts of the project, it meant that performance was analysed on an ongoing basis, detecting any issues

at early stage and therefore maintaining a tight schedule. This meant that all man-hours were tracked accordingly so that meant it was clear what to prioritise and therefore adjust the focus to different elements of the design process. From the market research carried out, there was a concise list of requirements to follow to ensure that the project delivered on desirable technical and visual aspects. In todays image conscious world, it was also necessary to ensure that the project maintained a consistent level of branding and style layout through cross platform devices.

1.2.2 Constraints

The key constraint in the development of this project was time. To be given a 16 week period in which all design, development and testing must be completed and accompanied by a comprehensive report, was an in-depth task. To be accountable for detailing the progression of the project and the methods used to develop Projectified and to balance the project deliverables so that the scope lay within realistic boundaries was my main focus.

The second constraint was that the level of skill set currently possessed meant that it was key to ensure that all work was to an optimum performance level personally to meet the requirements requested within the allotted time. This meant individually researching, trialling, testing and implementing all stages of the project effectively in the allotted timeframe, being aware of new tools and practices.

When adopting a new framework it takes an element of time to digest the requirements and to then carry these out, so the use of resources and tutorials were key in the development of this project.

1.2.3 Quality Objective

When undertaking a build of a system such as Projectified, it requires a certain level of quality assurance. With the right mechanisms in place, this will help safeguard well written code and ensure tasks are performed exactly as specified. To meet this objective, key testing tools and methodologies were factored in to meet quality assurance.

- HTML was validated using W3C validator http://validator.w3.org/

- CSS validation has been processed using http://jigsaw.w3.org/css-validator/
- Black box testing will test the functionality of the application by initiating unit testing. Results will be demonstrated by a pass or fail outcome.
- Debugging will also be ongoing throughout the application sprint phase using the frameworks built in debugging service.
- Testing cross browser for visual consistency.

1.2.4 Technical Objective

Projectified will cross platform and multi functional for browsers including, Safari, Chrome, Firefox, Opera and Internet Explorer 11. However Projectified will not be cross browser compatible, this means it will not operate efficiently in all instances of Internet Explorer or other legacy clients. To assist with any browsers that struggle to comprehend new HTML5 and CSS3 attributes and selectors, modernizr has been incorporated to help assist this need.

With the vast increase in mobile users, Projectified has been developed to cater for the mobile market as well as desktop environments. Therefore it is of paramount importance that design and content would be consistent and easily accessible through various devices. By providing an intuitive user experience, Projectified has harnessed this change and made it easy for people to access and navigate while using mobile devices. This also safeguards the future of the application as the explosion in the mobile market continues apace, it cements the position of Projectified across various platforms and multiple devices.

To further assist clientele with sporadic or weak internet connectivity, the back end of Projectified was developed to enable the minification of all its files, to provide faster loading times. An example file size that originally started off as 1mb may be reduced to 250kb saving a total of 75% on bandwidth giving an improved loading experience, this will also benefit clientele who wish to access Projectified through a mobile device. Projectified has also incorporated AJAX to asynchronously update and load content on the fly, this means content loads and updates in the background without having to refresh entire pages, all aiding for a smooth user experience

1.3 Report Outline

The objective of this report is to narrate in detail the process undertaken in developing the application and an overview of proceeding sections of the following report are outlined below:

Starting from **Section 2,** the focus is on requirement specifications, outlining what needs done and to what degree, through use of a Volere template and partaking a feasibility study to evaluate and analyse the current market.

Then **Section 3** narrates the design process, primarily the user interface including the project branding and also the system design, covering data design and logic design.

Moving into **Section 4** the scope of the project is expanded on in detail and shows how project was managed from a high level and the methodology used to manage its development cycle,-the technology and coding practices adopted , including the create/read/update/delete process.

As we progress into **Section 5** the focus turns to testing methods- user testing, to see if the features work as described and outlined and presented in table format in the appendix and also black box testing to check valid/invalid functionality.

By **Section 6** the task will be evaluation of the project as a whole, by looking at the methodology chosen and forming a conclusive analysis of how this worked**.**

The concluding **Section 7** contains the overview of the project from start to finish and where Projectified could progress forward continuing with its development.

2. Requirements

2.1 Introduction

This chapter will outline and document the requirements process carried out for this project. The need for a clear set of requirements at the start of the project can determine whether the system is a success or failure. Accurate and coherent requirements are crucial in going forward within a projects lifecycle as they ensure the project will meet a range of expectations such as personal and business objectives.

2.2 Requirements Gathering

Accumulating and analysing a set of user requirements is vital in any project. Having carried this out will ensure that during the development cycle of the Projectified project, there lies a higher likelihood of creating a system that will meet the users expectations. However, if approached in the wrong manner this could have harsh implications, leading to a dissatisfied user, resulting in a lower uptake and return value. Projectifed has used a modification of a Volere template to highlight the set of requirements that were specified at the beginning of the project.

Requirement ID: Requirement ID

Requirement Type: The type of requirement, system functional, look & feel

Description: A one sentence statement of the intention of the requirement

Rational: A justification of the requirement

Source: Who raised this requirement

Fit Criteria: A measurement of the requirement such that it is possible to test if the solution matches the original requirement

Priority Level: Importance of requirement on a scale 1 -5, 5 being most important

Dependencies: A list of other requirements that have some dependency on this one

Table 2.2 **Appendix A** contains a full set of requirements

2.3 UML

UML is used throughout the IT industry to help IT professionals to model computer applications. A use case illustrates a unit of functionality provided by the system. The main purpose of the use-case diagram is to help development teams visualise the functional requirements of a system. Applying this method meant it clarified all the system and functional requirements from a user's perspective when using Projectified.

Two users were identified when modeling:

- **Un-registered user** : A person who is not authorised to access the Projectified system. They cannot access the main Project area of the web application.
- **Registered user**: This is a person who has requested a beta access by emailing the Projecified admin. They will be invited to create an account. Having successfully completed registration, the user will be able to access the main system.

The diagram below shows a high level view of what a registered user can expect to do within the Projectified web application.

Figure 1 High Level view of a registered user of Projectified system

Full UML can be seen in **Appendix A.I** outlines the full UML with a detailed explanation.

2.4 Feasibility

Projectifed's target audience primarily consists of an age group bracket of 18-30. This is a wide demographic of people but this project was focused towards students and young professionals, employed in SME's.

The more that we delved into the user profiles of these personae, looking at their adoption of software and what tools they were currently using to manage their projects, meant that when designing Projectified we could empathise with a specific pool of people, so we could effectively design an agile and streamlined product, for their use.

Current offerings in project management include Basecamp, which holds the moniker for being the number 1 paid-for tool on the market, and over the last 10 years has helped 283,000 companies to manage their projects. Trello, another tool, boasts that it is the quickest way to organise projects, accommodates group collaborations and is used by large organisations from Adobe to The New York Times. Despite being a relative newcomer to the market, it holds greatest traction and unlike Basecamp it is free. Finally Atlassian, which was founded in 2002 to help enterprises collaborate better and its user benefits include collaboration, content sharing and project issue tracking but can be cumbersome to navigate and has a bloated user interface which is heavy and dated.

The market is broken up into categories within each tool:

- Project Planning & tracking
- Multi User interface
- Scheduling
- Document managing
- Budgeting time and expense tracking
- Billing and invoicing

- Resource allocation
- Risk management
- Customer management

In terms of the market going forward, the project management software future is a promising one, as Forrester currently estimates the market value to be 4.8billion USD a year. The main industries that use project tracking are aerospace, manufacturing and technology companies. Future trends show a movement towards cloud based solutions so that managers can access software on the go and maintain a mobile solution when on the road without needing to be tied to the office as much. Post recession this will also be more effective at streamlining projects to meet deadlines and keep costs under control. There is a keen demand for new products in the market and users are still searching for something innovative to fill this gap.

3. Design

3.1 Introduction

This chapter describes and conceptualises the design process for Projectified. Design for the front end will be expanded on within the User interface design section, showing the progression of this. The system design follows on from the requirements phase and within this section the system architecture is defined and outlined, demonstrating a clear understanding of the implementation of the robust data structure that unifies Projectified.

3.2 User Interface Design

Having identified the target audience, this greatly facilitated the styling of the application as research could focus on trends, colours and layout variations that are favoured by the target generation; this has therefore been used to the advantage of Projectified when selecting design features to create an intuitive user experience. Users want a clean and uncluttered workspace to navigate within, so they can easily find what they are looking for without distraction or procrastination.

When people look at a new interface, they scan, read and rapidly focus on the goal of getting to where they want, and the best user experience interfaces interact seamlessly with the user and allow them to achieve the end result unhindered. Projectified incorporates a term called “satisfying”, which typically means people do not like to think any more than they have to, it’s effort and people don’t have the time to comprehend the unusual. To overcome this the following has been considered in the design:

- Simple calls to action, allow the user to reach their goals with speed and efficiency
- Labelling is short and meaningful, plainly worded and quick to read.
- Clean interface to communicate meaning, via colour and form.
- Ease of navigation by uncluttered working environment within the interface

3.2.1 Initial Planning

Projectifed's initial design phase started with a method that has been reinforced over the years. Rapid paper prototyping allowed the project to test interactive concepts and interaction flows. When designing, 'sketch often' is often a motto heard and repeated regularly.

By careful planning and paper prototyping this meant that time can be dedicated freely to the generation of ideas and the only thing committed is time and paper, allowing you to refine ideas quickly and easily on paper and this in turn valuable time during the high fidelity design stage. This paper prototyping allowed the project to experiment without having to adjust key components, which within the development stage could have had detrimental consequences and jeopardise the project. Incorporating such a method aims to stop potential issues becoming major problems, as the only resources have been committed to an idea are paper, pencil and some time. It is much easier to test the feasibility of the design process and revise as needed.

Figure 3a Initial Planning – image https://flic.kr/p/bfzJAB

To entice users there were certain foundations incorporated. First and foremost, the interface had to be functional and usable; this would allow users to complete basic tasks quickly without having to learn a considerable amount. Secondly the interface had to be robust and reliable, no bugs that would possibly affect the users experience while interacting with the application, ensuring a simplicity of navigation and usability. This is how and why people want to use certain web applications as they deliver an entire experience from their entrance point, until well after they have left.

3.2.2 6UP

Prototyping was initiated, which would display concept layouts at a high level quickly and effectively provide six different concepts as opposed to one. A 6up (6 variations in design) was created to emulate various variations of visual hierarchy and information architecture, using multiple designs, while still focusing on key calls to action, remaining flat and not interactive. The six chosen concepts all used proven design patterns on a wide scale and the proposed designs emulated the following patterns:

- Two Panel selector
- One window drill down
- Row striping
- Nav based grid selection
- Modular Tab
- Common grid

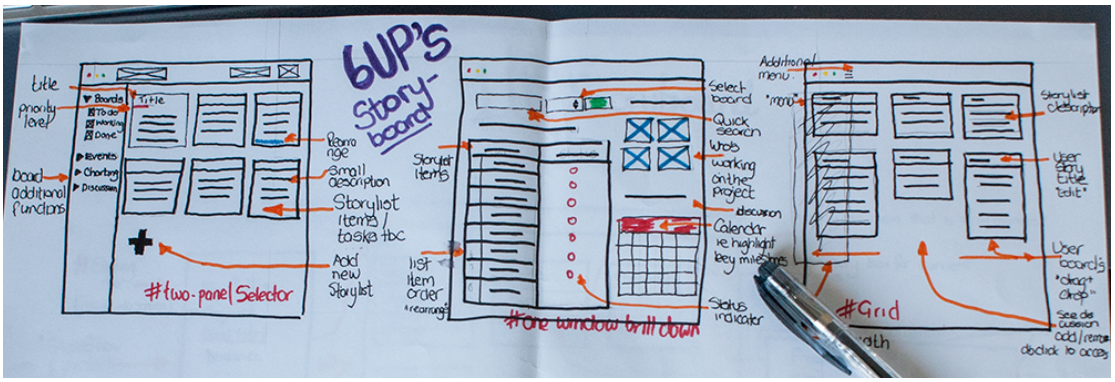


Figure 3b 6Up Sample

To see the full 6-up and initial sketches see **Appendix B – 6UP**

3.2.3 1UP

By analysing the initial variations of the 6up designs, it became clear that there were elements in all mockups of key attributes that would, when selected, aid towards a harmonious design and fulfill an end objective. Projectified incorporated a combination of

UI patterns, one known as the two-panel selector this was further refined and amalgamated with a grid structure to present the main data that now is Projectified.

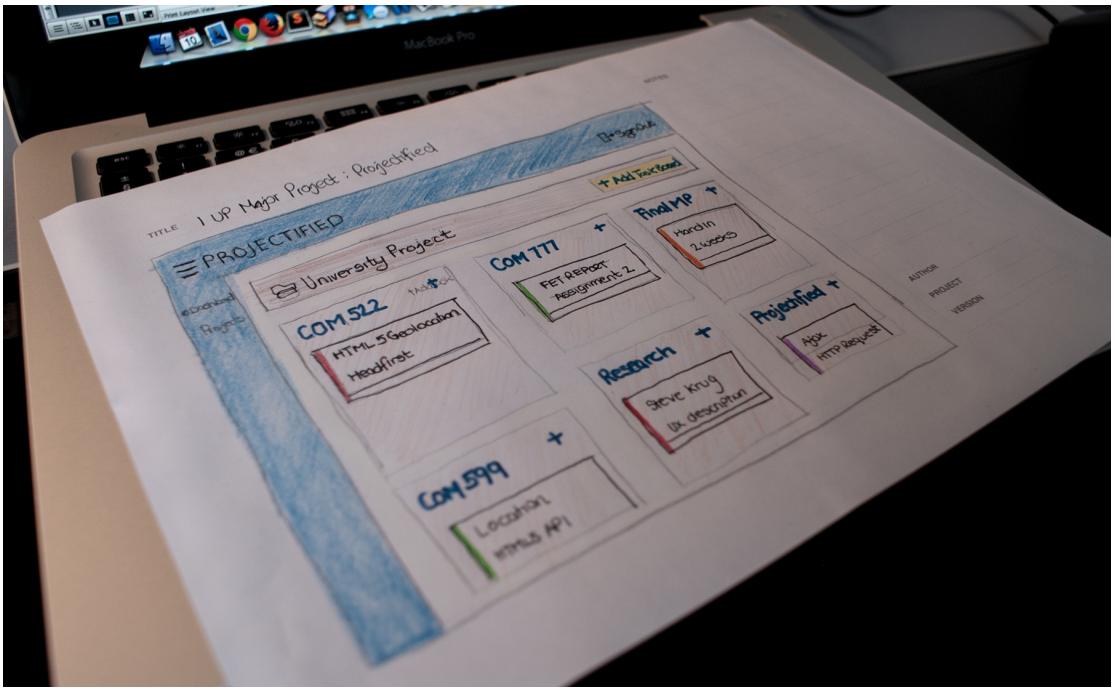


Figure 3c 1Up Refinement

3.2.3.1 Two Panel Selector

The two-panel selector was influenced by the Mac OS system preferences; it aided excellent usability. It clearly displays a list of items that the user can select at will on the left panel, while the right panel displays the relating content to that particular selected item.

This meant the user can see the overall structure of the list and keep their list items in view at all times, it would also be easy for the user to traverse through their boards by using the left panel. This therefore reduces visual cognitive load- the users' eyes don't have to travel far distances to access the relevant information, reducing the users memory for learning a new platform- one of the main aims. Another big benefit would see the same area being refreshed with new content as apposed to loading the page again.

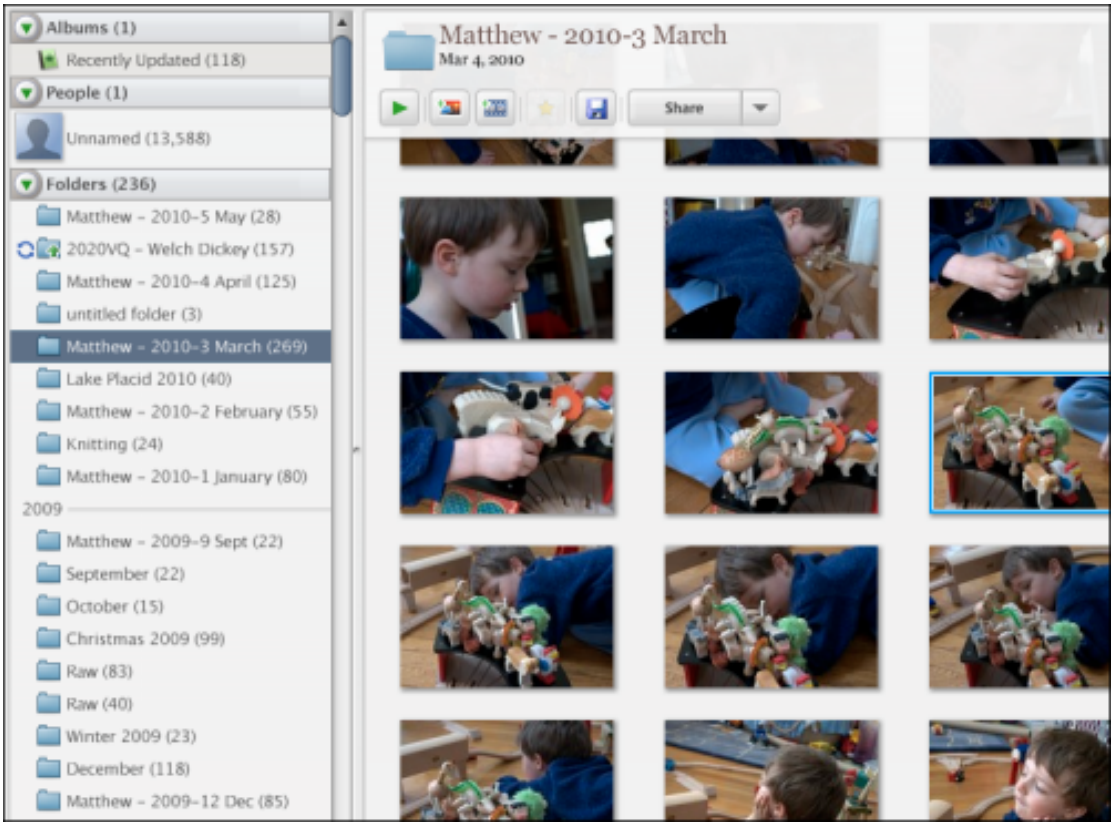


Figure 3d Two Panel Selector Application

3.2.3.2 Grid Structure

The final solution that complimented the 1UP, incorporated a sliding navigation panel and a main body harnessing a grid structure. The grid structure works well with what Projectified was aiming to achieve- to make the presentation of a large number of items attractive (Task Board's). It creates an immediate visual hierarchical structure that visualizes clear and concise meaning, helps users to see a complete overview of the project and aids them to scan quickly and efficiently to find the relevant "Task Board". With a grid like structure it also limits the learning curve for a user by communicating the information in a simple but intuitive way.

Figure 3e Nike grid structure

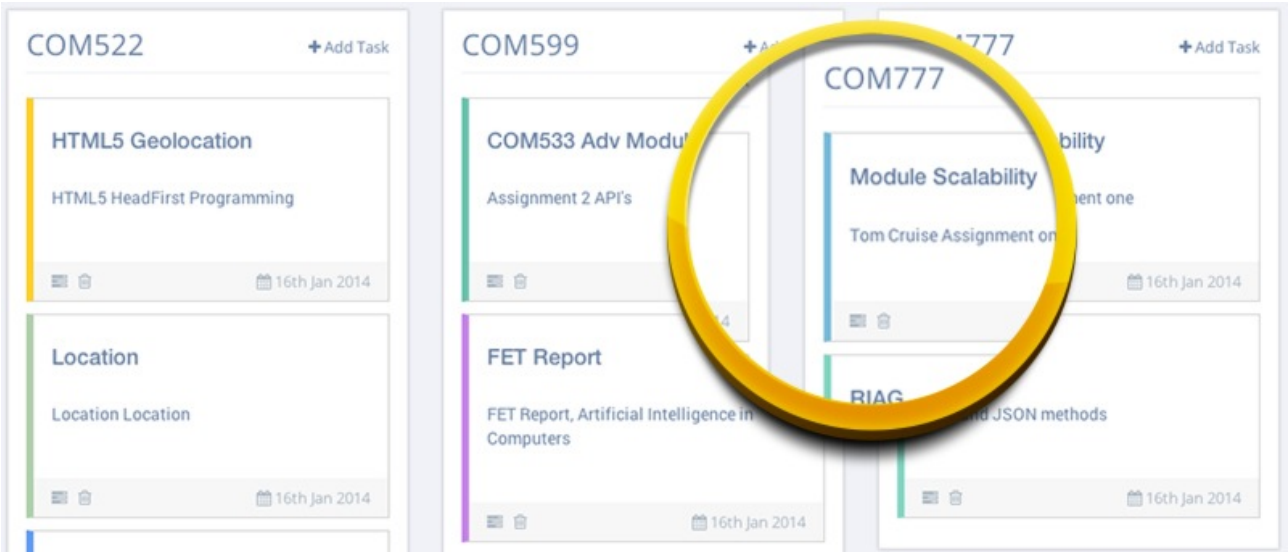


Figure 3f Projectified Grid Structure

3.2.4 Colour

In choosing a subdued palette for Projectified , the aim was to make the interface simple and user friendly : eye-catching but not glaring. Since the web application consists of similar elements, it made sense to go for a flat trend (a current trend) that would attract the target audience. Such companies like Apple and Microsoft have adopted this trend and therefore have made it widely popular with its users, people interact with these types of applications and devices daily, so it made logical sense to integrate a palette of this style that would translate across the spectrum of the application.

3.2.5 High Fidelity Design

Producing a high fidelity prototype demonstrates an almost finished final product which incorporated further detail and functionality. This was the final step after having completed various prototypes. From a user perspective, a high-fidelity prototype is close enough to a final product to be able to examine usability questions in detail and make strong conclusions about how behaviour will relate to use of the final product. The UML concludes the main logical steps a user must take to ensure the system would behave as predicted.

3.2.5.1 Homepage

The homepage is fundamentally the most important aspect of the application and needs to deliver a clear concise message within seconds. The homepage carries the bulk of the responsibility for attracting potential users to the application. With that in mind, it has been designed as a one page scroll website to reducing clicking action. It uses effective imagery and colours to entice the user, aided with descriptive context to grab the users attention.

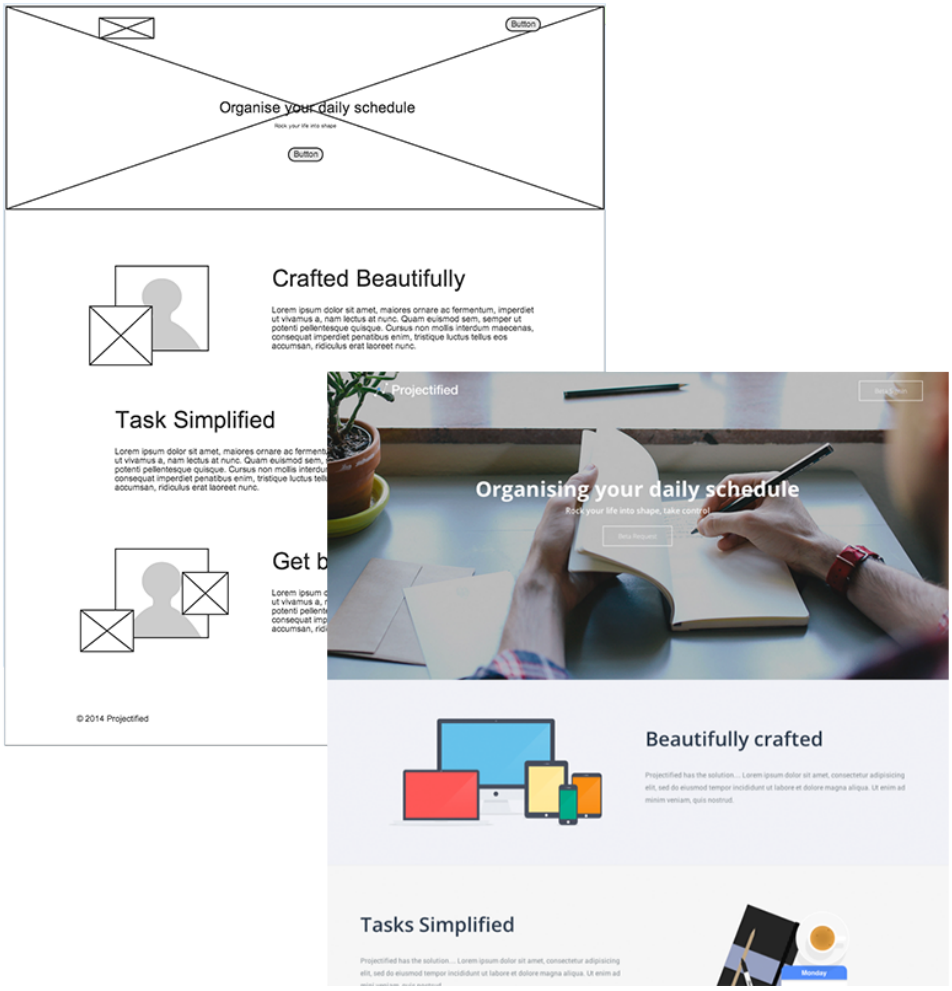


Figure 3g Transition from wireframe to High fidelity design

3.2.5.2 Dashboard View

The main dashboard view provides the user with a sense of overview for their entire account, in a brief summary the user can see what projects they are working on and see the activity that relates to their projects via the timeline. To see the design and key areas of interest please see **Appendix B – Main Dashboard View**. The key areas of interest have been elaborated below.

- Project Overview – Details all the current projects that you are currently working on. Inside the section, information can be found on the project, details such as project title, description, goals, created date, deadline date, delete project and colleagues can be found. The user can also update information regarding the project description and project goals and objectives if they suddenly change. Colleagues is another feature that will be in the next version of Projectified, this area demonstrates where the information is to be placed.
- Recent Activity – This area shows all recent activity within your account, the user can see when and piece of information has been altered, the user will be notified when something updates within a project, the information is then appended to this area showing the relevant information.

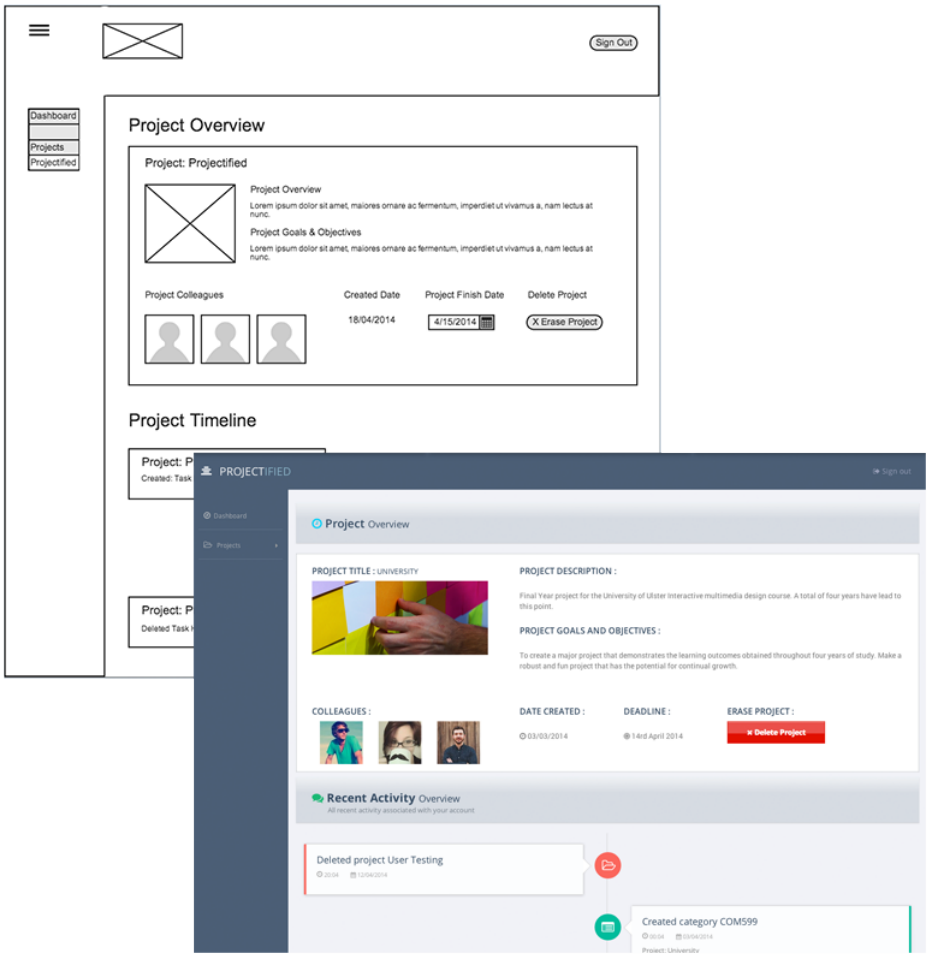


Figure 3h Transition for wireframe to high-fidelity Dashboard view

3.2.5.3 Main Project Page

The main project page will be the main point of interaction between the user and application, to see the “main project page” please see **Appendix B – Main Project Page**. Below describes the main features of this page.

1. Logo & Toggle Navigation – The toggle navigation is clear and unambiguous, this represents the sliding navigation panel that acts as a clear call to action. This is triggered by a single click action, after the action has been called the panel slides to the left and out of view, optimising the view stage, this therefore allows the content to take centre stage and helps highlight the clear focus, the content, the logo will always stay in view, this time it has been simplified again to just plain type.

2. Navigation Menu – The global navigation of the application; it has been identified by friendly identifiable icons aided with clear concise descriptions. The “projects” navigation

title, houses a sub navigational drop down menu which will use a subtle transition to display the users projects, here the user will also be able to add a new project that they wish to manage. This is how the user will traverse through the application.

3. Project Name - reflects the selected project, it is there to highlight to the user what project they are currently working in.

4. Sign out - The utility navigation label “sign out” has been placed as where many people expect to see such a label (top right), this will redirect the user back to the homepage.

5. Add Task Board - The feature of adding a new “Task Board” allows the user to boards effectively. The bright colour is there to draw attention to this action as it is a primary goal to get the project populated by categorising tasks. This will open a dialog box that will ask the user to insert a title for the “Task Board”, this method helps draw attention to the specific task that has been present to the user.

6. Task Board Title – This display the specific board category name, the user can also update the name if they feel the need to change the name at any point.

7. Add Task Button – This feature allows the user to add a task to the relevant category, this will open a modal window asking the user to insert a relevant title. Using a modal window will request the user to insert a title and description, the same reasoning has been employed with this specific action as “Add Task Board”

8. Task Title – This indicates the name of the specific task, again the user can inline edit this if they feel the need to.

9. Task Description – The task description is a supportive role, it is there to give a more descriptive breakdown of the task. Inline editing has also been built in to accommodate change.

10. Delete Task – Gives the ability to delete a task if the user so desires.

11. Task date – This is an area still to be implemented, it has been included for the next set of development cycles, where the user will be able to set a specific deadline for a task. This will be linked in the sub task section to help dynamically generate a target date.

12. Add Sub Task - this section is another future feature that will allow supporting documentation and separate sub tasks to be populated under a parent task. It is there to accommodate a task if it needs specific information that has to be elaborated on.

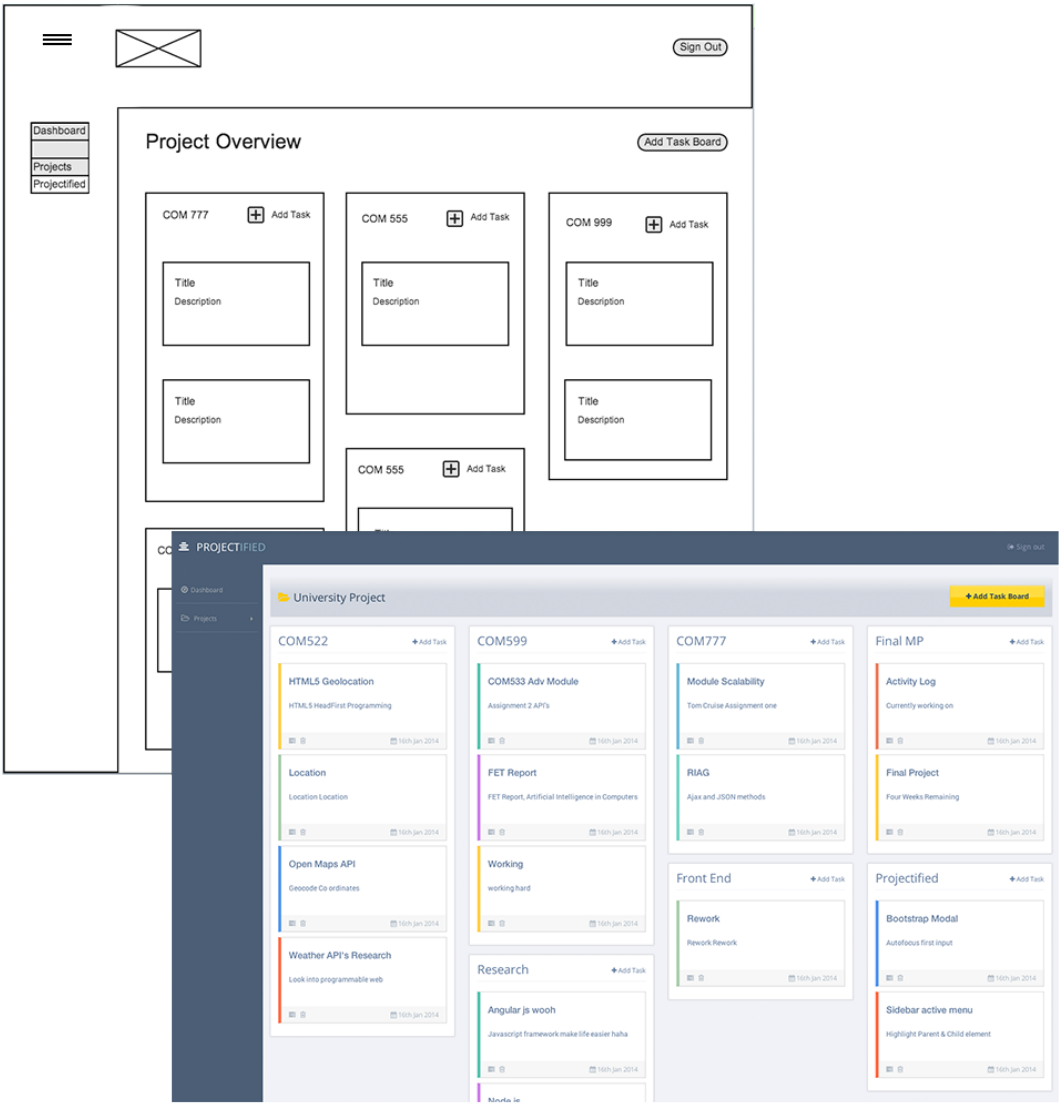


Figure 3I Project page wireframe to high-fidelity

3.2.6 Projectified Branding

Branding is really the pinnacle of any project; people often forget how important this can be, it defines what you do, the way you work, what your products should be and how you present the product to the world. Having honed the above, creating a brand helps you attract the customers that you want to serve, gain trust and keep them coming back. A

zealous brand may even be rewarded with admiration and devotion, but branding is more than surface and good looks.

To deliver a holistic brand that would create a recognisable identity, applying techniques already adopted in the user interface design section were used once again, as mentioned before little time is really wasted when prototyping with paper, it allows ideas to flow and flourish.

Following the method outlined enabled Projectified to arrive at its final solution:

- Idea generation, sketching
- Refinement incorporating different fonts, introducing colours
- Launch

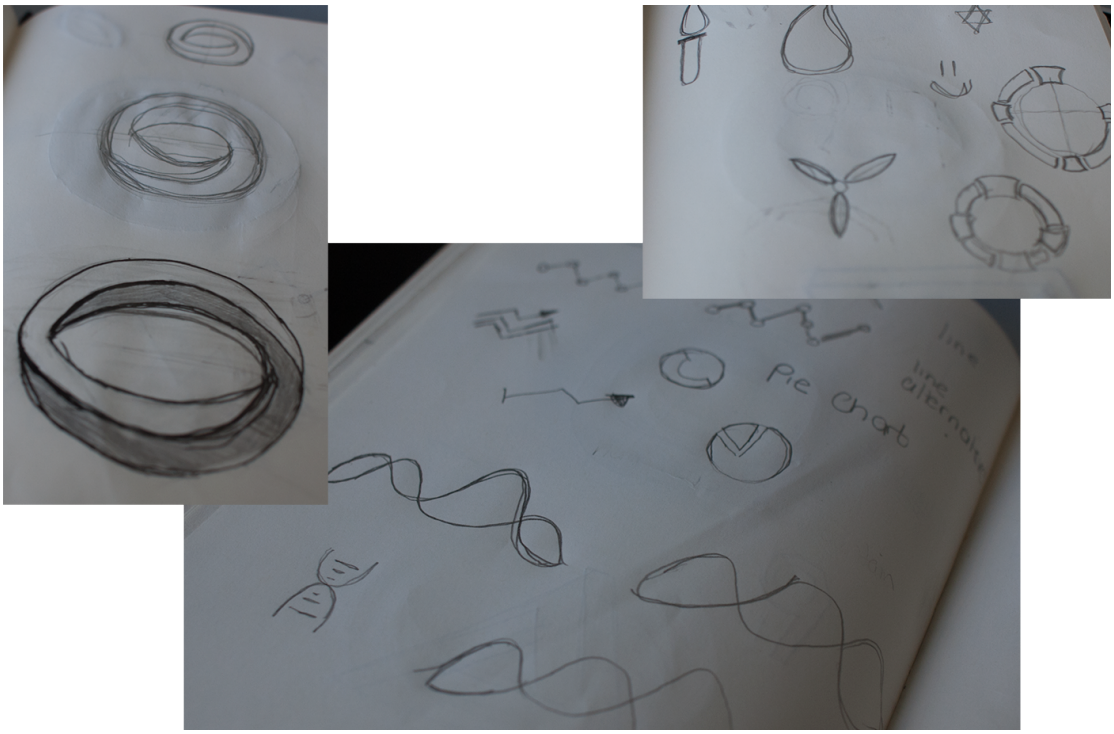


Figure 3J Initial Concept design sketching on paper

The final concept communicates a relevant message, the name Projecitified came from wanting to predict a time when a project and project segments would be completed. The inspiration word was “Projectify”, using this word with “fied” resulted is a strong and rounded complete word that unified what this project was trying to achieve. The pinnacle

graphic a “graph timeline” communicates a connected message through its staggered appearance, the key graphic represents key points on a graph which is interlinked to a typical project’s timeline. A powerful sans-serif font “OpenSans” has been selected for its optimised legibility through print, web and mobile devices which provide brand consistency. Using a font of this stature communicates a powerful, friendly, trusting relationship, which Projectified is striving to promote.

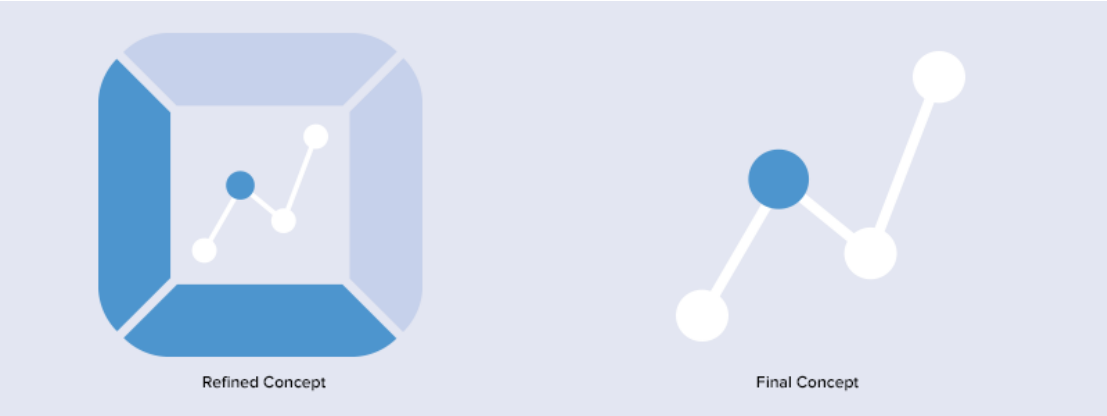


Figure 3k Logo Refinement

3.3 Data Design

The database design played a pinnacle role in the initial planning stage. Early analysis from the requirements document clearly identified Projectified to be transactional based, this meant the user was interested in a CRUD approach. A CRUD approach meant the user wanted to do the following: create, read, update and delete records. To create such a database that would facilitate the needs of the user, a sequence of steps needed to be fulfilled.

- From the requirements: create a unified database that would facilitate a create, read, update and delete approach
  - Determine the data needed, identify key entities that would facilitate key attributes and superimpose logical relationships on the data.

3.3.1 Entity relationships

An ERD demonstrates a high-level view of the data structure contained within side Projectifed. Initially key entities were outlined to contain specific attributes. Each entity holds attributes, which specially describe the type of data it holds, this then allowed the project to superimpose a logical structure upon the data on the basis of relationships. First off the entities and their attributes were outlined, this helped visualise the data that would be involved within the project.

- Users: Store information about the user
- Project User: This will allow a unique relationship between users and projects
- Project: Stores all information relating to Projects
- Category: The user can add task boards to a project
- Task: Contains all information relating to a task
- Action: Hold information about sub tasks.

Entity Type	Attributes
users	User_id, first_name, last_name, email, pass, reg_date
project_users	user_id, project_id
project	project_id, project_name, deleted, updated, created, description, objectives
category	category_id, category_name, project_id, deleted, created, updated
task	ID, name, description, category_id, deleted, created, updated
action	Id, title, taks_id

Table 3l Initial Entities Outline

Outlined are the entity relationships, this describes the relations between each specific entity.

- **Users – project_users (1: N)** - This implies a one to many relationship; One user can work on many projects through a join table called project_users.
- **Project** – **project_users (1: N) –** The relationship shows a project can have many users, through a join table called project_users.
- **Project - category (1 : N) –** Represents one project can have many categories.
- **Category – Task (1: N) –** This relationship transpires that one category can have many tasks.
- **Task – action (1 : N) –** This is another one to many relationship; it demonstrates a task can have many actions i.e. sub tasks.

Projectified has incorporated a many-to-many relationship due to future evolution and development of the application. Projectified wanted to use this pattern specifically so that: a user can work on more than one project, and a project can have more than one user assigned.

To get around the problem of having a many-to-many relationship Projectified broke apart the many-to-many relationship into two one-to-many relationships. Using a third table, called a “join table”, does this. Each record in the “join table” incorporates the foreign key fields of the two tables it is joining together (users, project). Nothing special needs to be done with the foreign key fields in the join table as they will get populated with data from the other two tables as records are created. It is not uncommon for a join table to have a lot of records in it – since records are created in the join table as records are created in the two tables it joins.

The full ERD gives a conceptual model of the world that is represented in the database, a full ERD can be found in **Appendix B – ERD**.

3.4 System design

Projectified employs the user of a 3 tier architecture commonly referred to as a multi tier architecture to send and receive data between client, server and database. This is commonly the most adopted software pattern for database driven applications and dynamic websites.

There are 3 tiers to the architecture:

- **Presentation Tier:** The presentation tier is responsible for the client, which contains the user interface this is responsible for displaying results a user can understand. A user can also interact with UI and prove events such as a click event. This specific layer doesn't contain any logic or sever code.
- **Logic Tier:** The logic tier controls the projects functionality by performing sets of instructions which is then processed and transferred between the two surrounding tiers. An example of this within the project would be the setup of the controller.
- **Data Tier:** The final tier, the data tier can store and retrieve information based of the request made. The information can be passed to the logic tier for processing and then eventually back to the user.

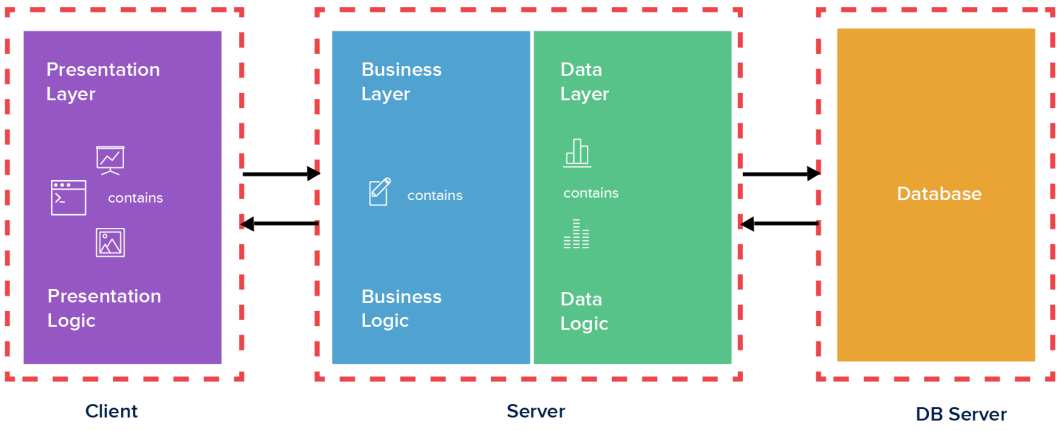


Figure 3m Diagram showing the multi tier architecture

3.4.1 System framework

Evaluating a number of fthe rameworks concluded that they were all very bloated in their features, this highlighted the need for a micro framework to develop the project within. Silex is a small PHP micro framework built on the shoulders of Symfony2, Symfony2 is a fully fledged framework featuring ample dependices, which the project did not need.

Silex has multiple features such as powerful routing systems, undefined architecture, simple configuration and template rendering, all which the project desperately needed to be successful. Silex also allowed the project to use previous experience gained with PHP and help build upon it by giving structure and organisation to make the code coherent when reading the contents of a controller. Secondly Silex allowed the project to use third party services such as utilities and libraries that helped the system expand when necessary.

Projectified has made use of the lightweight system to construct its controllers. Projectified uses a view controller method to deliver, update and structure its content to the user. This allowed the project to serve up a complete feature or also known as vertical slice - 'the sum of the work that has to be done in every layer that is involved in getting a specific feature working'.

3.4.2 Controller

The controller is the main epicentre this takes a HTTP request and constructs and returns an HTTP response, namely an HTML Page, a JSON array or even a redirect. Each request the user makes, is handled by the resulting controller matching the URL. The route reads the information and matches the information to the specific controller. The matched controller then executes the code inside the controller and creates a response object (contains HTTP header and body). The HTTP headers and content of the response object are then sent back the client. The controllers for this project have been broken into maintainable sections to be easily identified when traversing through the backend of the application.

The project broke the logic into different categories to make it easier to identify which controller did what, below is the general outline for how controllers are stored for the project.

```
└── src (Parent folder that contains all controllers)
    └── category.php
    └── dashboard.php
    └── Project.php
    └── task.php
    └── user.php
```

Figure 3n Folder structure for the controllers inside Projectified project

One category is “projects” which contains logic of controllers about projects.

```
└── Project.php (controller)
    └── project add
    └── display project
    └── project navigation building
    └── delete project
```

Figure 3o File structure for “project” controller, specific relating logic entailed inside

General steps that happen inside the controller:

- Get user input
- Validate the user input
- Persist (save to database)
- Serve back the right page via the render ability

3.4.2.1 Controller Routing

Commonly within the application we are using Route Parameters as Controller arguments. For example, to see and read a project when a user clicks on a specific project. That project has a specific id which is passed to the controller as a single argument $id, which corresponds to the {id} parameter from the matched route. In fact, when executing the controller Silex matches each argument of the controller with a parameter from the matched route. Below demonstrates this within the Project controller to display a specific project.

```
$app->match('/project/{id}', function (Request $request, $id) use ($app) {
```

Figure 3n Shows Routing Paramater { id } requested from URL. Resulting URL www.projectified.co.uk/project/1

The URL is matched and the $id is retrieved via the parameter which can then be passed into the query to display the specific project id.

```
WHERE project_users.user_id = ?
       AND project.project_id = ?


       ORDER BY project_name, category_name, task_name
   ", array(
       $app['session']->get('user_id'),
       $id //get id of current project
   ));
```

Figure 3O shows the $id being passed as a blind parameter to the query

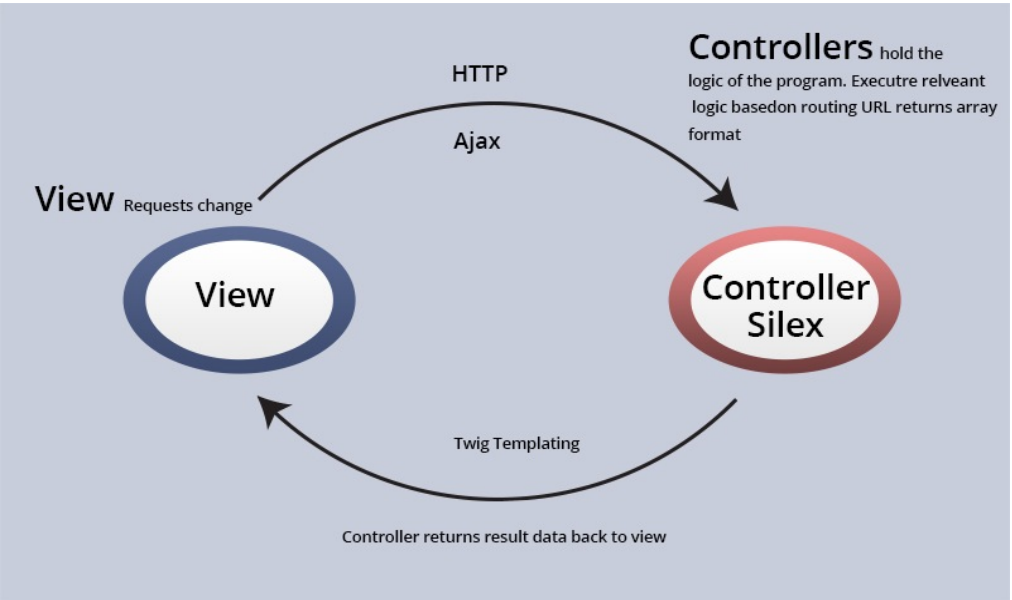


Figure 3P Application Flow

4. Implementation

4.1 Introduction

This chapter will outline the implementation phase for the project. Over the course of this chapter it looks deeper into the implementation strategy used to develop the project along with industry standard techonology and outlines of various technologies applied within the coding to achieve the results in which you will see inside the project.

4.2 Implementation Strategy

An agile methodology was chosen as the implementation strategy for this project due to its iterative and incremental approach. This approach emphasizes the rapid delivery of complete functional components. The requirements outlined are broken down into unique iterative increments called “sprints”. Each sprint has a defined duration usually in weeks, this project it opted to break the duration into hours derived from a total capacity. The total capacity was outlined with the use of a PERT analysis with a running list of deliverables. If planned work for a sprint could not be completed, work was re-prioritised and rescheduled due to an implication. Sprints outlined Appendix E – Projectified Gannt Chart

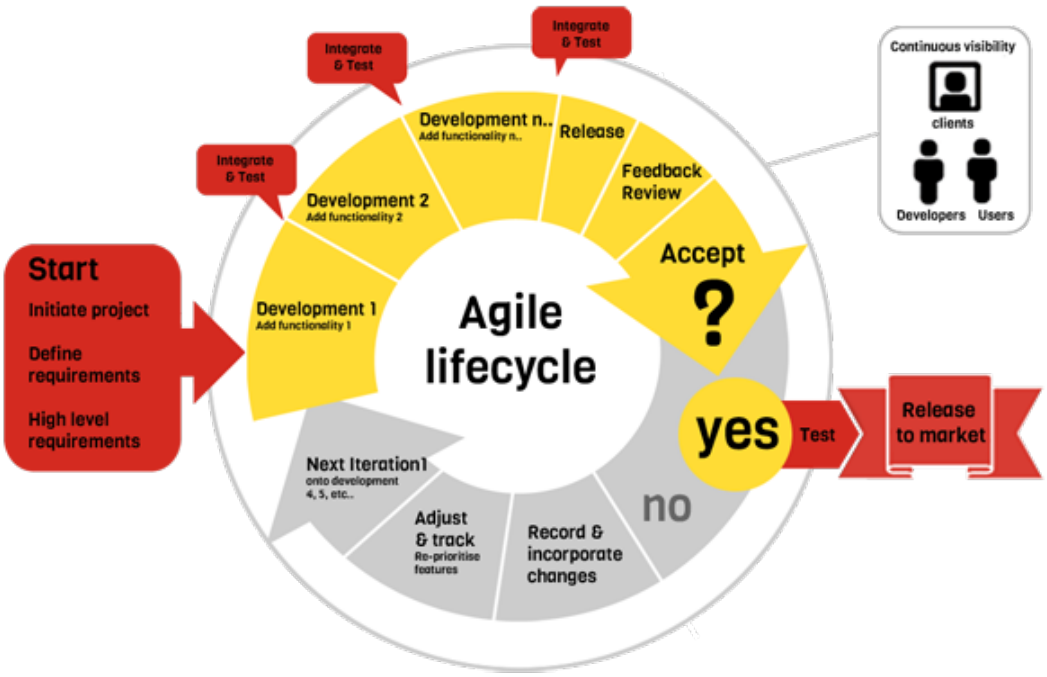


Figure 4 Agile lifecycle

4.2.1 PERT Analysis

To complement and gain further control within the project and to further safeguard the development, a PERT (Program Evaluation and Review Technique) was adopted to help manage and predict the project lifecycle for the development stage.

The beneficial factor of using PERT to facilitate with the planning process is to help develop and test the plan to ensure that it is robust. PERT formally identifies the tasks which must be completed on time for the whole project to be completed on time and to schedule. It also identifies which tasks can be delayed if resources need to be reallocated to accommodate tasks running over schedule. A further benefit of Critical Path Analysis is that it helps you to identify the minimum length of time needed to complete a project. This type of project management is more commonly found with in large corporations to help streamline high-risk projects that have could affect tight budgets and jeopardise a project or product.

Using the PERT formula PERT = (P+O+4M)/6 Where P is Pessimistic estimate O is Optimistic estimate M is Most likely estimate. The purpose of PERT analysis is to identify an estimate for a sprint. This method addresses the issue of uncertainty in estimating the sprint duration. The uncertainty in the duration estimate can be calculated by making a three-point estimate in which each point corresponds to one of the following estimate types:-

- Most likely scenario - The activity duration is calculated in most practical terms by factoring in; resources likely to be assigned, realistic expectations of the resources, dependencies, and interruptions.
- Optimistic scenario - This is the best-case version of the situation described in the most likely scenario.
- Pessimistic scenario - This is the worst-case version of the situation described in the most likely scenario.

The spread of these three estimates determines the uncertainty. The resultant duration is calculated by taking the average of the three estimates. This type of approach can be applied to everyday life and help plan for the unexpected, a very simple yet effective way to manage scope.

$$pert\ estimate = \frac{p + 4 \times m + o}{6}$$

Figure 4.1 PERT Analysis formula

Included below shows how PERT was used within the project. As this analysis was a recursive pattern a snippet has been included to show the use of the PERT to predict cycle times for certain aspects within the project.

Activit	Activeity Description	Technologies	Weig	Time Estimation			Expected Timef	Actual
A	Delete Project via Dashboard - Mark inside database with datestamp	Silex PHP, Jquery, I	5	6	8	12	8hr 33mins	8hrs 20
B	Add Modal for sub Tasks, involve Creating, Reading, Store and Delete	Jquery, Silex PHP, /	1	3	6	12	6hr 50min	N/A
C	Create Dynamic Activity log CRUD	Jquery, Silex PHP, /	1	6	8	12	8.33hrs	7hrs 52
D	Link up homepage with signin modal Login rework	Jquery, Ajax, Silex	1	4	6	8	6hrs	5hr 23
F	Create Project Overview		1	6	12	18	12hrs	3hrs 09
G	List All Projects within Overview		1	1	2	5	2.33hrs	4hrs 33
K	Logout rework		1	2	4	5	4.23hrs	3hrs 20
N	Inline edit task		1	4	6	10	6.33hrs	4hrs 05
P	Inline Edit task board title resave ajax		1	1	2	3	1.33hrs	
				Optimistic (o)	Normal(M	pessimistic(P)		

Figure 4.2 PERT analysis of completed work

4.3 Technologies Adopted

The scope of this section covers all the relevant technologies integrated into the system, the sections expand on the details as to why they have been selected and integrated to help unify the project. Topics include PHP, MySQL, AJAX, jQuery, HTML and CSS.

4.3.1 PHP – Hypertext Preprocessor

PHP (Hypertext Preprocessor) –. PHP is used by 77.9% of all the websites whose server-side programming language is known. WordPress is used by 16.6% of all the websites in the world. If you have a look at the top three CMS's, for the websites that use a monitored

content management system: Wordpress is first with 54.3%, Joomla is second with 9.2%, and Drupal is third with 6.8%. Three products are all written in PHP. Companies including Facebook created their system using PHP, and it still resides in the top programming language for the web today based on Google Trends as shown in figure 4B. Having prior knowledge of PHP and backed with statistics made it the most viable choice in choosing this language as the server side language of choice made for a strong stack when moving into the development phase.

Apr 2014	Apr 2013	Change	Programming Language	Ratings	Change
1	1		C	17.631%	-0.23%
2	2		Java	17.348%	-0.33%
3	4	^	Objective-C	12.875%	+3.28%
4	3	v	C++	6.137%	-3.58%
5	5		C#	4.820%	-1.33%
6	7	^	(Visual) Basic	3.441%	-1.26%
7	6	v	PHP	2.773%	-2.65%
8	8		Python	1.993%	-2.45%
9	11	^	JavaScript	1.750%	+0.24%
10	12	^	Visual Basic .NET	1.748%	+0.65%
11	10	v	Ruby	1.745%	-0.23%
12	17	^^	Transact-SQL	1.170%	+0.45%
13	9	vv	Perl	1.027%	-1.31%

Figure 4.3 Programming Trend April 2014, PHP remains the top web language

4.3.2 MySQL

The worlds most used open source database management system, provides capabilities such as: scalability and flexibility, high performance due to its unique storage structure allowing you to configure the MySQL database server to your needs and high availability meaning all hosts provide MySQL as standard. MySQL allowed Projectified to build and implement a robust data structure that would easy integrate with PHP, which would then further help to manage large amounts of data and store it securely via a transaction. The data structure and relationships are easily mapped as demonstrated previously in the ERD.

4.3.3 jQuery

The use of jQuery was vital as it provided an easy way of targeting elements, DOM traversal and modification, DOM manipulation based on CSS selectors that uses node elements name and node elements attributes (id and class) as criteria to build selectors, support for AJAX. Also extensive knowledge and support was found widely on the web, websites like http://stackoverflow.com provide a wealth of knowledge on every topic imaginable. This is also an industry standard library that uses common javascript functions and builds upon them, Projectified depends heavily on jQuery to deliver its features. jQuery had a turbulent time due to other libraries such as dojo and prototype trying to replace the library, but now the industry recognise jQuery to be as valuable as javascript.

4.3.4 Ajax

Ajax enabled Projectified to be dynamic, this made certain elements run faster within the application as smaller requests were made to the server. This allowed changes to take place on the web page without having to reload the entire page. All the server has to do is send a small piece of data to the caller when requested. This small piece of data travels faster over the network, which means that request latency is also smaller, meaning faster page speed and resulting in an highly interactive web application.

4.3.5 LAMP Stack

Using a local development environment required having a LAMP stack installed, using a custom environment and not a downloaded package. The LAMP stack allowed Projectified to ultimately build the application locally before deploying it to the web. Using this method allowed the project to trial and test the development of the application and ensure its reliability and stability. This helped save valuable time, and the well-defined development process ensures a highly efficient application as the end result.

4.3.6 HTML & CSS

Projectified uses twitter bootstrap to format the main bulk of the HTML, while CSS has been incorporated to give visual style and hierarchy to elements.

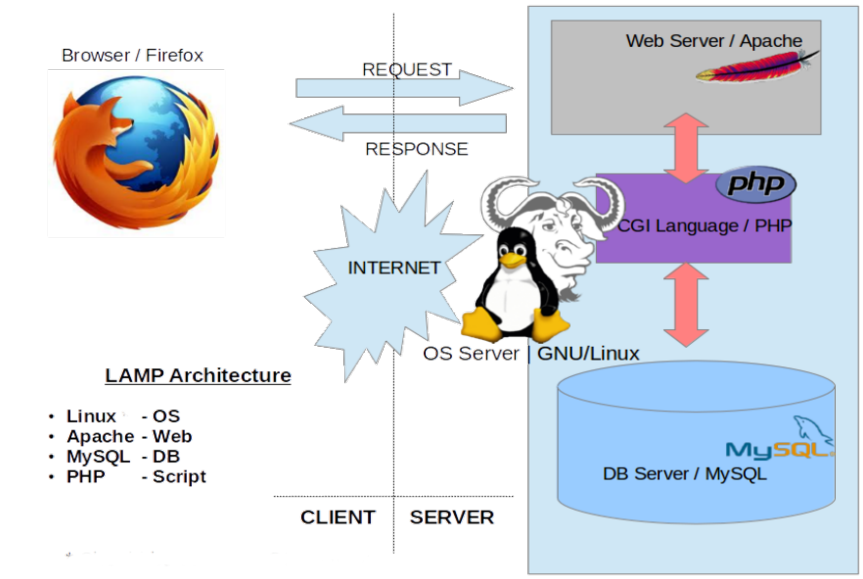


Figure 4.4 Demonstrates A LAMP stack on a local environment identical to the technology stack employed to develop Projecitifed

4.4 Software Implementation

Due to the restricted space the key steps in creating Projectified are outlined within this section. The areas of development have been identified to help understand how Projectifed was created. Over the course of this sub section topic areas have been elaborated on:

- Implementing the database from the ERD,
- How the framework was installed onto the hardware
- How Projectified used templates to drive the front-end
- How information was read from the database and made available to the templates
- Explore how information was created
- Explore how information was updated
- Finally how information was deleted within the Project

4.4.1 Database Integration

Using the previous ERD diagram outlined all the tables and relationships needed to create a solid data foundation for the project. Projectified identified it would need to use a set of commands in order to create the database itself. This set of commands were used to perform the initalisation, as per below, outlining the process used in initailising the database for the project, Sequel Pro was used as a GUI to help create the more tedious aspects such as the relationships.

- Create the database

```
CREATE DATABASE projectified;
```

Figure 4.5 Create database command

- Create a User table

```
CREATE TABLE user (
    user_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    email TEXT NOT NULL,
    pass VARCHAR(60) NOT NULL,
    reg_date DATETIME NOT NULL
    ) DEFAULT CHARACTER SET utf8;
```

Figure 4.6 Table CREATE query

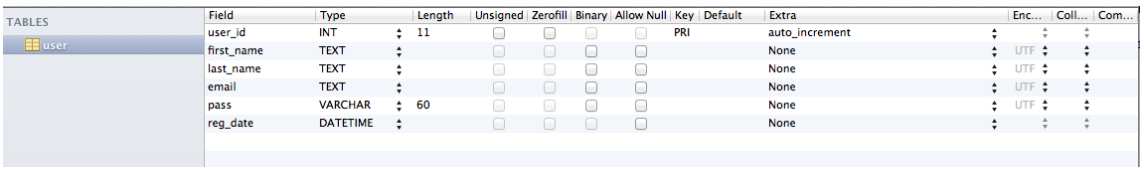


Figure 4.7 Database Implementation User table, not null forced the column not to be null i.e. not left blank

- Load sample data

```
INSERT INTO user (
        user_id, first_name, last_name, email, pass, reg_date)
    VALUES (
    "1", "Chris", "Marks", "marks-c1@email.ulster.ac.uk", md5("12345"), "2014-01-13"
);
```

Figure 4.8 sample data being inserted

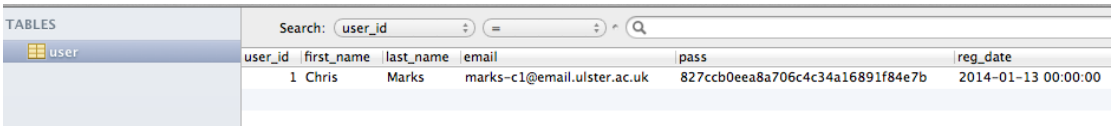


Figure 4.9 Result in the GUI

Implementing the database was the first stage in getting Projectified off the ground. Projectifed used a database abstraction layer in which to manage its queries using Redbean, removing unnecessary clutter and allowing for a more intuitive approach to SELECT, INSERT, UPDATE and DELETE information. Below shows the Redbean data abstraction when connecting to the database within the application.

```
// RedBean Database credentials
R::setup('mysql:host=localhost; dbname=projectified','root','root');
```

Figure 4.10 shows abstraction to quickly connect to the database

4.4.2 Framework Installation

Having a solid database meant Projectified could acquire its framework; there were multiple steps involved in getting the Silex framework up and running. Using dependency packages with composer, allowed projectified to acquire third party services that helped with pretty URL's, sessions, templating and http request and response methods.

Installing Silex was processed through the command line. Using the composer documentation the following steps were undertaken to install the framework.

- Get package manager: composer
- Define the dependencies using a: composer.json

- Get packages: composer install

Next Projectified development folder was targeted to composer:

```
$ curl -sS https://getcomposer.org/installer | php
```

Figure 4.11 Installing composer within projectified development folder

After the install of composer the next phase was to define and install the package of Silex. Composer required the project to create a composer.json file, this file describes the dependencies for project.

```
{
    "require": {
        "gabordemooij/redbean": "dev-master",
        "silex/silex": "dev-master",
        "symfony/form": "dev-master",
        "twig/twig": "dev-master",
        "symfony/twig-bridge": "dev-master",
        "symfony/translation": "dev-master",
        "ircmaxell/password-compat": "dev-master"
    },
    "minimum-stability": "dev"
}
```

Figure 4.12 Projectifieds composer.json file for dependencies

We request to use the dev-master branch of silex to keep on top of any updates to the system that could further help enhance features, one feature was built in over the course of the Project for REST when it allowed to use the PATCH method to update segments of code.

Finally the last command was instructed to ‘run’, installing the list of dependices including the Silex framework.

```
php composer.phar install
```

Figure 4.13 Installing the list of dependencies

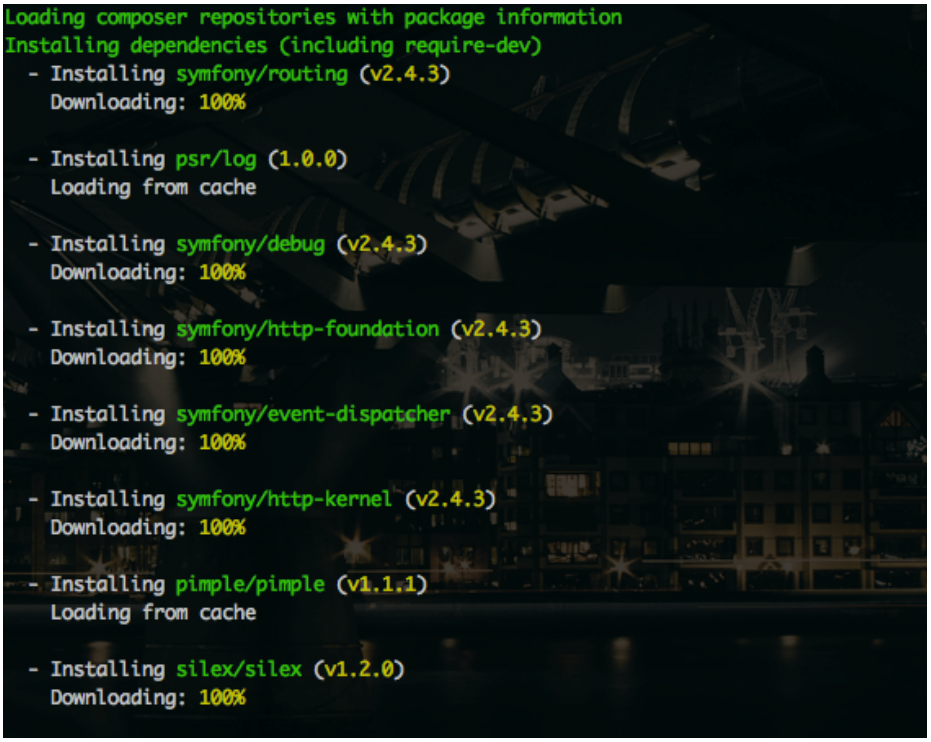


Figure 4.14 Successful installation via terminal

4.4.3 Reading and Displaying Information

This process of reading information from the database for a specific project proved to be one of the largest and most complicated parts of the project. The query designed to solve this issue was the largest by nature ,as it needed to JOIN multiple tables accurately to acquire the relevant information.

To show a specific project the user requests a web page, the URL contains a project id which is a parameter and is stored into a variable, the variable is then passed to the query in order to SELECT the right project. This is done by the Request object.

```
$app->match('/project/{id}', function (Request $request, $id) use ($app) {
```

Figure 4.15 requesting ID from the URL via request object

The query is used to SELECT the relevant information based on having the specific ID, we also use the session management system to provide the query to get the correct user_id which has been already set from the login.

```
$rows = R::getAll("
        SELECT project_name, category.category_id, category_name, task.id as
 task_id, task.name as task_name, task.description as task_description
        FROM project

        LEFT JOIN category
        ON project.project_id = category.project_id
        AND category.deleted is NULL

        LEFT JOIN task
        ON task.category_id = category.category_id
        AND task.deleted is NULL

        LEFT JOIN project_users
        ON project.project_id = project_users.project_id

        WHERE project_users.user_id = ?
        AND project.project_id = ?


        ORDER BY project_name, category_name, task_name
    ", array(
        $app['session']-
>get('user_id'), //using the current user id from login
        $id //get id of current project
    ));
```

Figure 4.16 shows the select query to retrieve a project and all the relevant information. We are using an abstraction layer to get the information using PHP Redbean

At this stage the code dumped an associative array, which contained repeating elements such as the Project name and category name. As this was not ideal we needed a way in which to remove redundant information from inside the list, which would result in the format which was needed to pass to the front-end. Please see **Appendix C – Initial Row Dum**

To get the data we needed to loop through the results to structure the data so it could be easily handled throughout the Project, which stopped duplicates like the Project name.

```php
/*
 * Structures data to be easily handled throughout the project it removes
 * duplicate data which we do not need ie like Project name
*/

$project = array(
    'id' => $id,
    'name' => $rows[0]['project_name'],
    'categories' => array()
);

//var_dump($rows);

foreach ($rows as $k => $row) {
    $category_found = false;

    foreach ($project['categories'] as $key => $category) {
        if ($category['name'] === $row['category_name']) {
            $category_found = true;
            // $category is the same as $project['categories'][$key]
            // But $category is a temporary value
            // And $project['categories'][$key] will modify the orignal array
            $project['categories'][$key]['tasks'][] = array(
                'id' => $row['task_id'],
                'name' => $row['task_name'],
                'description' => $row['task_description']
            );
        }
    }

    if ($category_found === false) {
        if (isset($row['task_name'])) {//Assuming category is present, when name is present
            $tasks = array(array(
                'id' => $row['task_id'],
                'name' => $row['task_name'],
                'description' => $row['task_description']
            ));
        } else {
            $tasks = array();
        }

        $project['categories'][] = array(
            'id' => $row['category_id'],
            'name' => $row['category_name'],
            'tasks' => $tasks
        );
    }
```

Figure 4.17 Formats the arrays and removes duplicate values

Now we can easily use the information and access it via dot notation; meaning we could format the output with HTML and CSS to deliver the correct structure reflected upon within the design section. See **Appendix C – Structured Row Dump.**

4.4.4 Create Task Board

Another unique feature built into Projectified is the use of AJAX to create a highly interactive application. This makes the application respond more quickly. Using this technology demonstrates the power and knowledge required from lessons during the previous semester in how to implement such a feature.

When the user clicks on the Task Board button, it performs a number of functions of which the visitor is totally oblivious to. Firstly the user fires a request, which the AJAX object matches via its POST method, it then posts the values to the category controller. Below shows how the URL pattern is matched.

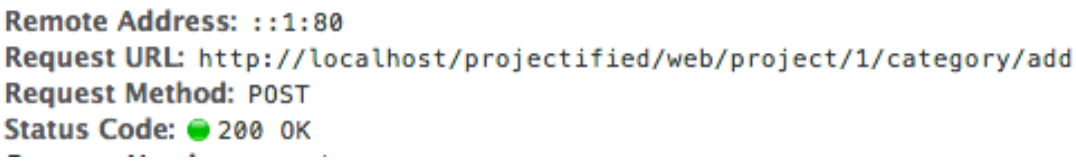


Figure 4.18 Request URL sent via client

```
$.ajax({
        type: "POST",
        // ie dynamic linking pulling in a global variable 'basepath'
        url: basepath + "/project/" + project_id + "/category/add",
        data: form.serialize(), // $('input.project-title').val(),
        cache: false
```

Figure 4.19 AJAX then passes the data asynchronously to the controller

The next phase is to pass the input value and the current project id into the resulting query so we can store the information inside the database. The query has been constructed to receive blind parameters.

```
//Inital inserting the project title and getting ID
R::exec("INSERT INTO category (category_name, project_id, created) VALUES (?, ?, NOW())",
array(
        //getting modal input val
        $request->get('board-title'),
        //id of current project
        $id
    ));
```

Figure 4.20 Inserting values into database

Having successfully completed, we then build the template server side and we encode the data with JSON ready to pass back using a response object. The AJAX is then able to continue processing the data and output the relevant information to the client.

```
}).done(function(data) {
      $("#add-board").modal('hide');
      // Template was build server-side (big complicated template)
      data = $.parseJSON(data);
      id = data.id;
      //grab all cateogry HTML
      category_html = data.category_html;
      //place 1 category into the DOM
      $('#categories').prepend(category_html);
      //for single category change css to opactiy 1
      $('#category-' + id).css('opacity', 1);
```

Figure 4.21 Finalisation and continuing to insert data into the front-end

4.4.5 Update Task

When updating a task, Projectified applied a new way of doing so. Using a jQuery plugin that was compatible with the twitter bootstrap model, allowed Projectified to update task names and descriptions with relative ease (only after known how it was done of course). jQuery X-editable can be used in conjunction with twitter bootstrap, allowing the project to create editable items on the front-end. When a new set of values were submitted, it also sent an AJAX request- these values contained the field name in the database and the specific value to be updated.

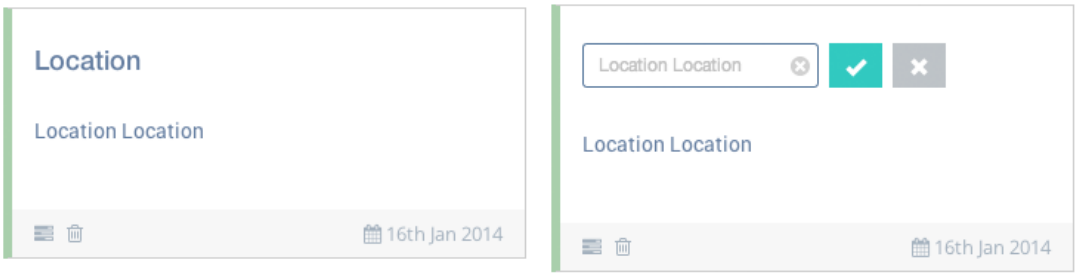


Figure 4.22 jQuery X editable front end view

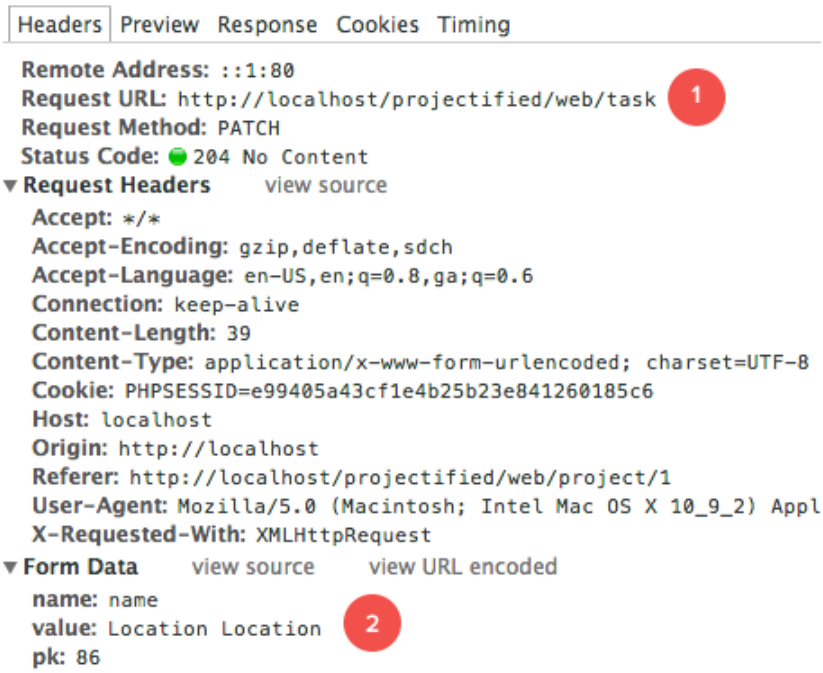


Figure 4.23 shows the request URL and the data to be passed

Above shows

1. The user wanting to pass the information to a specific URL while requesting to do a PATCH method.

2. Shows the data being passed from the form input via the front-end

```
  //Task name / titles
  $.fn.editable.defaults.ajaxOptions = {
    type: "PATCH"
  };
  $.fn.editable.defaults.mode = 'inline';
      $('.task-name').editable({
        name: 'name', //name of field (column in db)
        type: 'text', //type of input (text, textarea, select, etc)
        url: basepath + '/task', // url to server-
//side script to process submitted value (/post, post.php etc)
        title: 'Modify Task Title',
        status: 200, //ok response
        responseTime: 1000,
```

figure 4.24 shows the jQuery that handles the request, this will wait for the backend before issuing a response to the user

Using a HTTP patch method on the controller meant the project could do a partial update on the specific table.

```
/**
 * Update Task Name via jquery x-editable app-script.js 217
 */
$app->patch('task', function (Request $request) use ($app) {
```

figure 4.25 controller using patch method to partially update the task table

A variable,defined inside the controller, allowed the task title / description to be retrieved via the input. Two queries were made depending on the input using an if / else statement. This updated the relevant column within the backend, and once executed the controller sent a response back, which was picked up again by the plugin.

```
    //get the name from the jquery x ediatble script line 334
    $property = $request->get('name');

    if (strlen($request->get('value')) > 2000) {
        return new Response('Value is more then 2000 characters.', 422);
    }

    //if the property is task title use name as property as this will contain
    //text to update
    if ($property === 'name') {
        $sql = 'UPDATE task SET name = ?, updated = NOW() WHERE id  = ?';
    //if the property is task description use description as property as it w
    //contain the text to update from jquery line 369
    } elseif ($property === 'description') {
        $sql = 'UPDATE task SET description = ?, updated = NOW() WHERE id  =
    } else {
        return new Response('Unknown property to update.', 422);
    }

    //Execute SQL query with requested values name/description and taskID
    R::exec($sql, array(
        $request->get('value'),
        $request->get('pk')
    ));
```

figure 4.26 shows the values stored in variables being passed to the query for update

On the succession of a title or description being updated a success handler was engaged which invoked another jQuery plugin (toastr) to display a successful message with the relevant updated value to the front end.

```
    success: function(response, newValue) {
      //jQuery plugin to show update toastr, newValue is the value set, specified
      //by jQuery editable
      toastr.success(newValue, 'Successfully updated task title')
    }
```

figure 4.27 the success that is generated and returned to the user via response

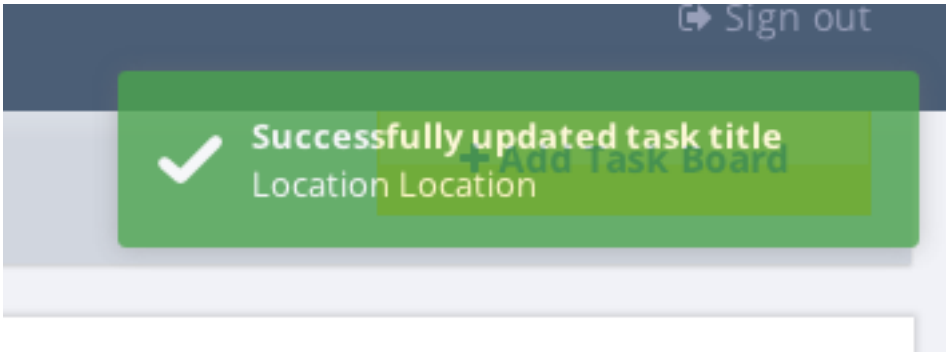
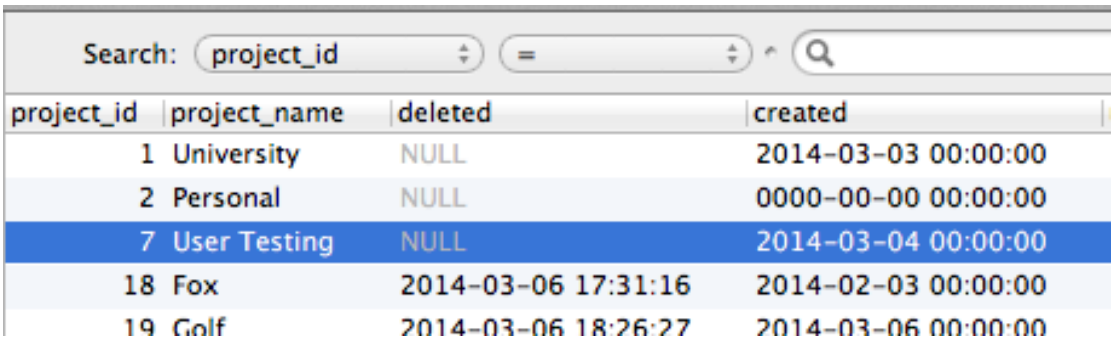


figure 4.28 The alert generated helped with the jQuery toastr plugin which include the data that was just updated.

4.4.6 Delete Project

To avoid potential data loss Projectified incorporated a soft delete method within its database. This would ensure the safety of information if the delete button was triggered accidentally, it would ensure the data could be restored if an event like this occurred. It also shows further steps were taken in securing users data, thus applying another analytical approach to the development cycle.

Firstly it was identified that the database needed some further attributes to enable such a feature to be accommodated for. The deleted column would hold a DATETIME format so when queried if such a value existed it would ignore the Project.



project_id	project_name	deleted	created
1	University	NULL	2014-03-03 00:00:00
2	Personal	NULL	0000-00-00 00:00:00
7	User Testing	NULL	2014-03-04 00:00:00
18	Fox	2014-03-06 17:31:16	2014-02-03 00:00:00
19	Golf	2014-03-06 18:26:27	2014-03-06 00:00:00

figure 4.29 Delete field added to the database

Secondly there were two main functions that would occur when a user initiates the delete button. This triggers an event within the front-end and within the back-end. This identifies

the need to remove all the project contents from a div within the front-end while update the back-end using the soft delete method.

To remove the effected elements jQuery was used to target specific elements within the DOM to remove the associated Project. From the button we traverse up through the elements to the main surrounding div to remove.

```
  /**
   * Delete Project
   */
  //selecting delete project button
  $('.project-review-delete').on('click', function(e) {
  //select specific button, store project id in var, with this selected get the
  // closest button and data attribute of id
    project_id = $(e.target).closest('button').data('id');
    var that = this; // grab this from current scope, copy to variable
    $.get(basepath + "/project/" + project_id + "/delete").done(function() {
        $(that).parents('.project-review-details').effect('blind');
    });
});
```

Figure 4.30 jQuery used to remove a project from the front end

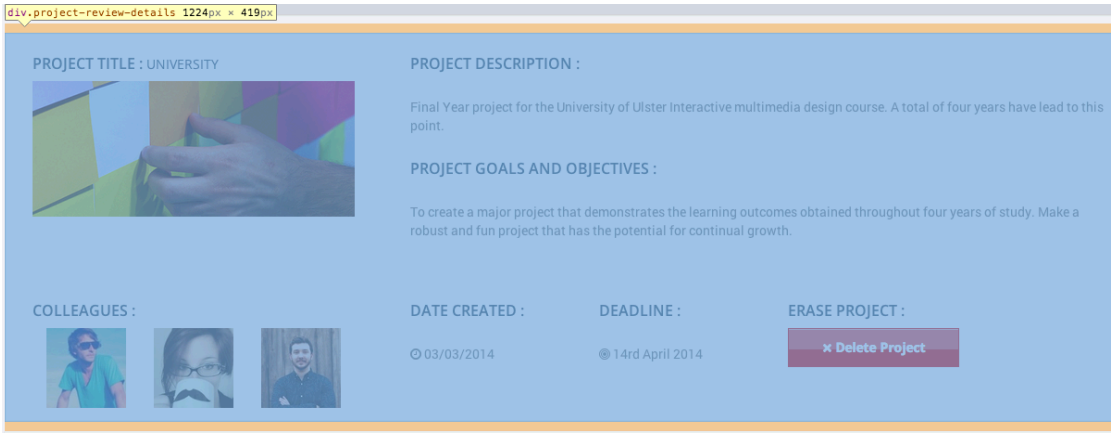


Figure 4.31 highlights the main parent div, .project-review-details

The next step was to amend the database and apply a soft delete action. Again we target the id from the URL by retrieving it using the request object. Once retrieved we can pass this into the query and you will see that this query uses Redbean for abstraction. Once executed we apply a datestamp to the deleted column, this means when the Project is queried it will not show via the front end. To show that it is a success we return a HTTP response consisting of a 204 meaning the server has fulfilled the request but does not need to return an entity body. The 204 response MUST NOT include a message-body, and thus is always terminated by the first empty line after the header fields.

```
/**
 * Delete Project
 */
$app-
>match('/project/{id}/delete', function (Request $request, $id) use ($app) {
    //using Update to set a soft delete against the matched project id,
    R::exec( 'UPDATE project SET deleted = NOW() WHERE project_id = ?', array(
        $id
    ));
    //return response no content, sucessful delete
    return new Response('', 204); // 204 No Content

//project-
delete assert make sure the id argument is numeric, since \d+ matches any amount
 of digits:
})->bind('project-delete')->assert('id', '\d+');
```

Figure 4.32 back end code to update project table and specific project

Once this has been processed the results can be seen within the database and specifically the project table, below shows the consequence of a soft delete.

project_id	project_name	deleted	created
1	University	NULL	2014-03-03 00:00:00
2	Personal	NULL	0000-00-00 00:00:00
7	User Testing	2014-04-12 20:09:53	2014-03-04 00:00:00
18	Fox	2014-03-06 17:31:16	2014-02-03 00:00:00

Figure 4.33 Implemented solution, deleted column shows two projects with a datetime stamp.

5. Testing

5.1 Introduction

This chapter outlines the testing strategy used for the Projectified web application. The tests clarified whether the system conformed to the requirements previous set. Concluding the chapter will show the results obtained from a variety of testing techniques. Testing is a integral part of software development, it has to locate defects and eradicate them before a feature had been launched within the project. This enables the user to benefit from an error free application, and makes for happy navigating.

5.2 Testing Strategy

The testing strategy applied to this project consisted of four different strategies all testing various features including functionality, device consistency via browser and mobile, and whether it conformed to W3C standards. Dynamic testing was also adopted throughout development. This involved actually running the program with a set of tests. This was commonly practiced as it involved checking the code implemented at each stage to ensure it was correctly performing e.g. (updating the database, retrieving data from the database and checking requests made via Headers). This was made possible as test data was inserted into the database during the implementation stage, this allowed to test the code before being implemented to make sure it was working as intended.

The 4 test strategies implemented into Projectified:

- Black box testing method
- Cross browser testing
- Mobile testing
- W3C validation

5.3 Black Box testing

Blackbox testing also commonly referred to as functional testing and behavioural testing. This testing method focuses on determining whether Projectified does what is supposed to do based on requirements. Adopting this method attempted to find errors in the code, determine incorrect / missing functionality, any interface errors, errors in data structure used by the interface and any behavior of performance errors.

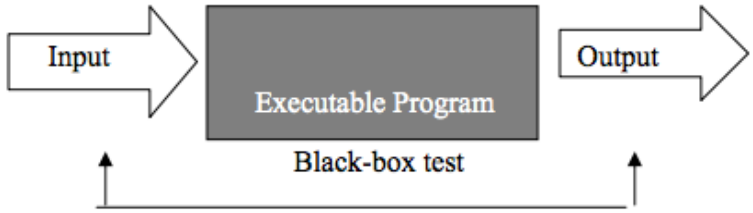


Figure 5 black box testing. Black box testing takes into account only the input and output of the software without regard to the internal code of the program

Tests were made small but relevant to the project. Projectified's test subjects varied from family, friends and peers from the IMD year group. It would be better to have test subjects who have no prior knowledge of the system, but with time constraints having an impact it was better having these test subjects than none at all.

5.3.1 Black box test case

A test template was created for main components of the web application. The tests were designed to test the main requirements of the web application outlined. Each test followed the format outlined below:

- **Test ID** – Unique identifier for each test case
- **Action** – Describes the step of the particular test to be carried out
- **Expect Results** – What is expected to come out from the black box based upon input
- **Pass / Fail** – Did the input meet the result described using pass / fail
- **Comments** – Any comments that support the test result

Test ID	Action	Expected Results	Pass / Fail	Comments

Figure 5.1 Testing template for Projectified Blackbox method

The results for the full application can be found inside **Append D – Black Box testing**. As outlined, there were other features that did not get implemented due to the time constraint outlined in the initial chapters, however the these features are seen visually but are not interactive as they are there for demonstration purposes which will aid the next development iteration.

5.4 Cross browser testing

With a wide range of web browsers available, end users are using different web browsers to access the web application, meaning it is crucial to test the web application on multiple browsers to ensure consistency in design and content delivery. Using browser stats provided by W3 Schools, indicated that Chrome currently secured half the market share closely followed by Firefox and this was taken into consideration when developing for browser consistency. To see the full list, see **Appendix D**.

Since Projectified incorporated a HTML and CSS framework (Twitter Boostrap) it specifically stated the browsers that it supported. Below outlines the browser versions which twitter bootstrap accommodates.

- Chrome (Mac, Windows, iOS, and Android)
- Safari (Mac and iOS only, as Windows has more or less been discontinued)
- Firefox (Mac, Windows)
- Internet Explorer
- Opera (Mac, Windows)

Unofficially, Bootstrap should look and behave well enough in Chromium for Linux and Internet Explorer 7, though they are not officially supported. Internet Explorer 8 and 9 are also supported, however CSS3 properties—e.g., rounded corners and shadows—are not supported by IE8. In addition, Internet Explorer 8 requires the use of respond.js to enable media query support.

To test the design and data structure through various browsers, browser stack was incorporated. This provided a virtual environment to test on to check the consistency, a virtual box of Windows 7 was incorporated, as it is the most widely used operating system.

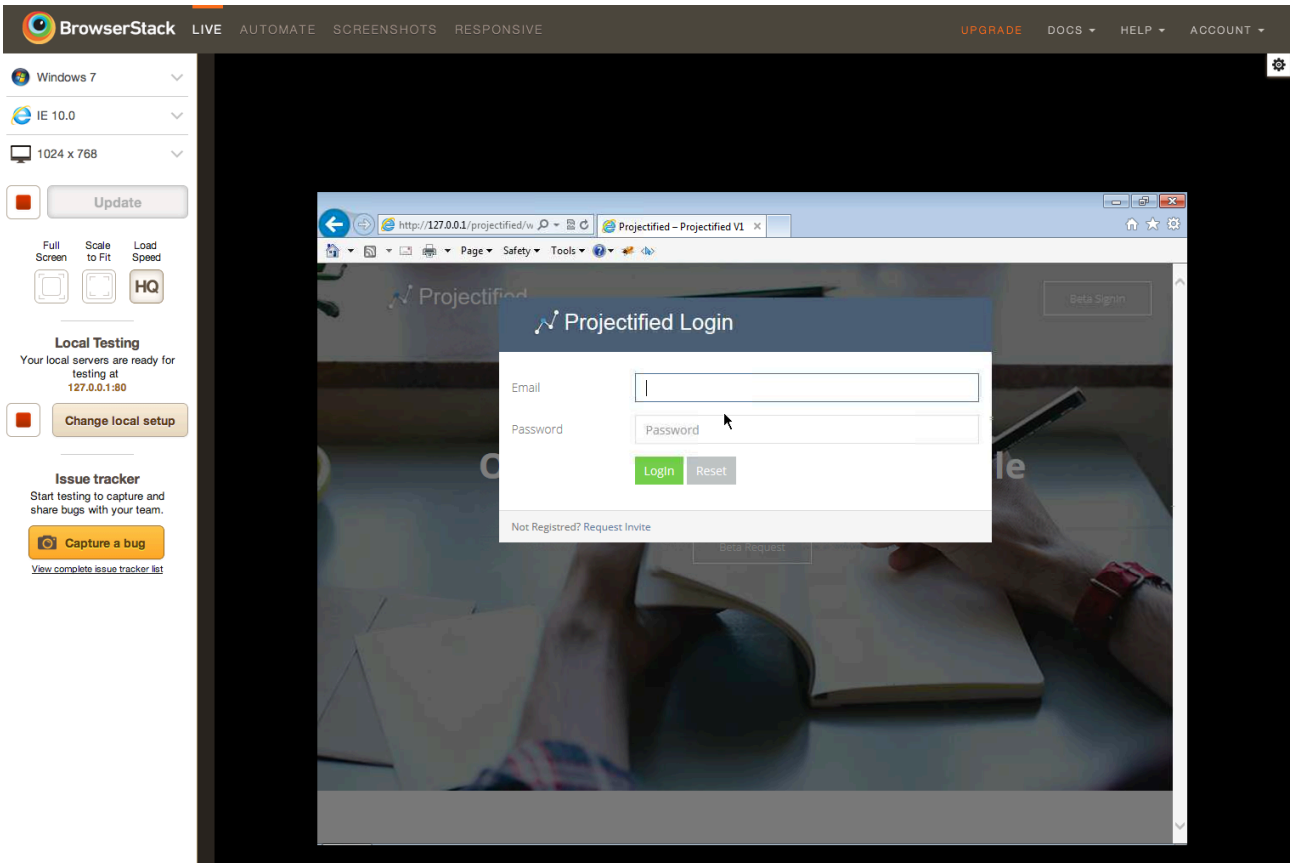


Figure 5.2 Browser testing with Browserstack.com

5.4.1 Cross browser results

The results of the cross browser tests have proved beneficial, this outlines the application is in a positive state to move forward with small adjustments to be made.

In Internet Explorer the application proved the least reliable, only catering for version 10 and greater. A small amount of work would be needed to make it ready for version 9 as elements were clearly visible within the page but not correctly positioned.

Firefox showed excellent support. It accommodated the design and interaction as far back as version 16 and supports all clients up to the latest version. Firefox is now in beta stage 28.0.

Safari supported the application from version 5.1 and greater. Main elements on version 4 would appear on the homepage but not in the correct place, some further refinement would be needed to accommodate version 4.

Chrome provided the best results, it supported all features right back to version 16.0 The application was primarily built on a chrome instance while in development and checked regularly on all latest instances of Safari, Firefox and Opera.

Lastly Opera catered for the application from version 16.0 onwards it is now moving onto version 20.0.

Projectified remained true to its earlier stance in accommodating all modern browser versions, and although there wasn't a viable instance to cater for legacy clients in decline, Projectified made sure it was providing an optimal experience for the technologically savvy.

5.5 Mobile Testing

The testing on mobile devices required the use of an online mobile emulator to replicate the design and interaction mapped through various devices. As the mobile market has exploded with exceptional growth, more and more people are seeking to access the web and applications while on the move. It was paramount that Projectified catered for the active user who endorsed the mobile.

Projectifeid used free a online service (http://www.responsinator.com/) to test on a wide variety of mobile platforms to help replicate a live environment. To test this while reading the report it is suggest you navigate to the emulator used by Projectified. To see the results please inject (http://www.projectified.co.uk) into the URL bar provided.

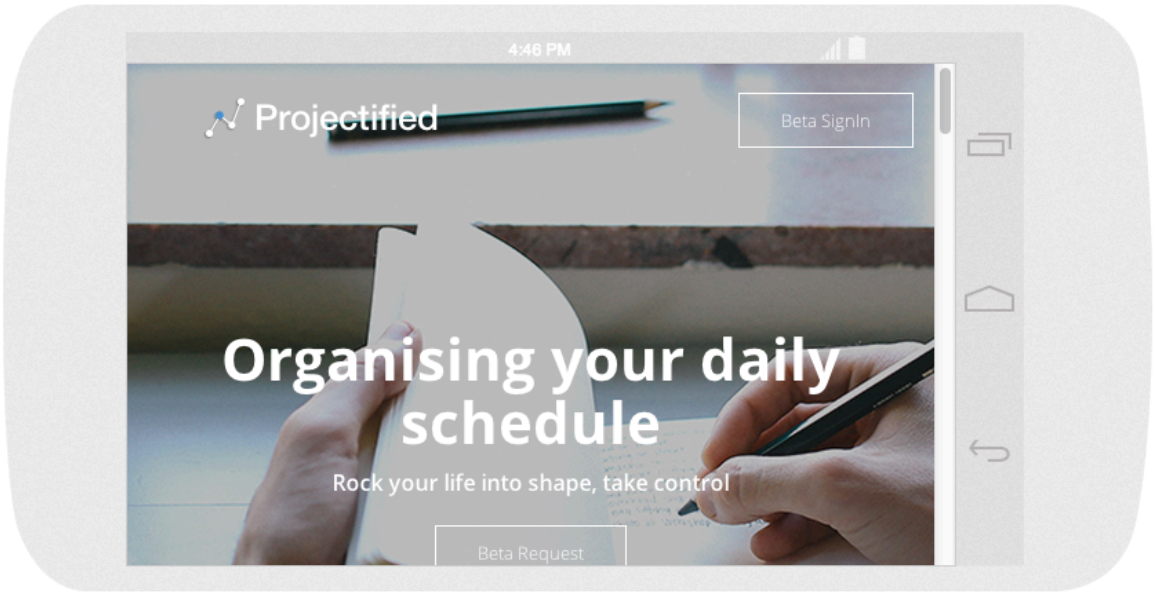


Figure 5.3 Responsinator Emulator of Android Nexus 4 Landscape Orientation

Once injected you will see the emulator of various devices load Projectified. The results are very pleasing, but there is room for small improvements to be made to ensure absolute consistency is met, however Projectified does deliver excellent usability cross the mobile landscape which further demonstrates the use of effective mobile design and delivery through the use of media queries.

5.6 W3C Validation

To further conform with official web standards, further testing and validation were carried out to make sure the application met the standards implied by the World Wide Web Consortium (W3C). This is still an ongoing testing exercise currently within the project as it is still seen as an ever-evolving prototype, however it is important to obey the testing a validation set by W3C. Below shows the Projecitifed Dashboard approved by the W3C.

More tests will be carried out to facilitate the consistency of the application to a high standard.

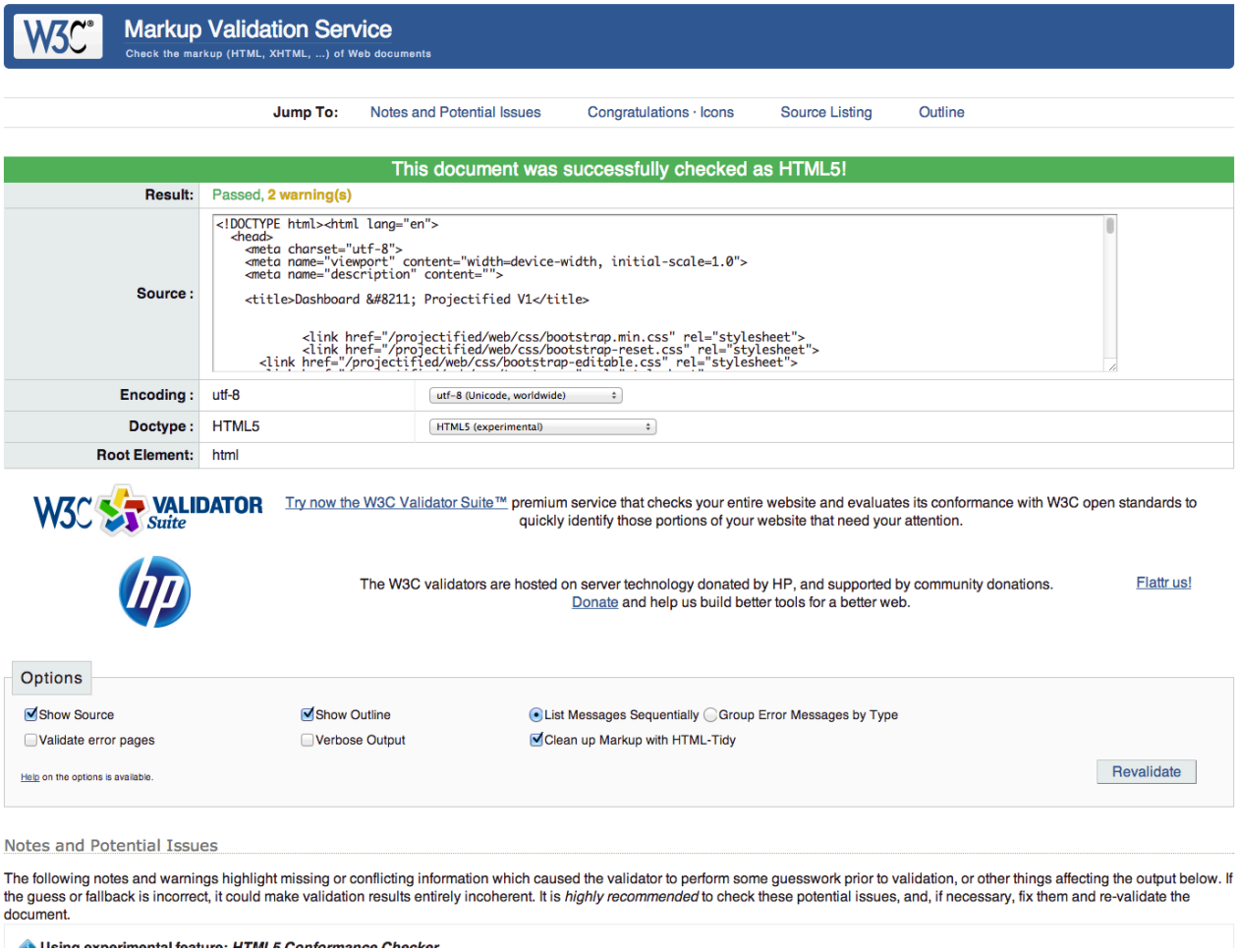


Figure 5.4 Projectified Dashboard Validation

6. Evaluation

6.1 Introduction

This chapter will be based on the evaluations and analysis of the detailed results of the outlined target audience survey. Then the evaluations will turn to the methodology and outline plan of the project and this will be commented upon to provide a conclusive overview as to how the project was managed.

6.2 Evaluation Survey results

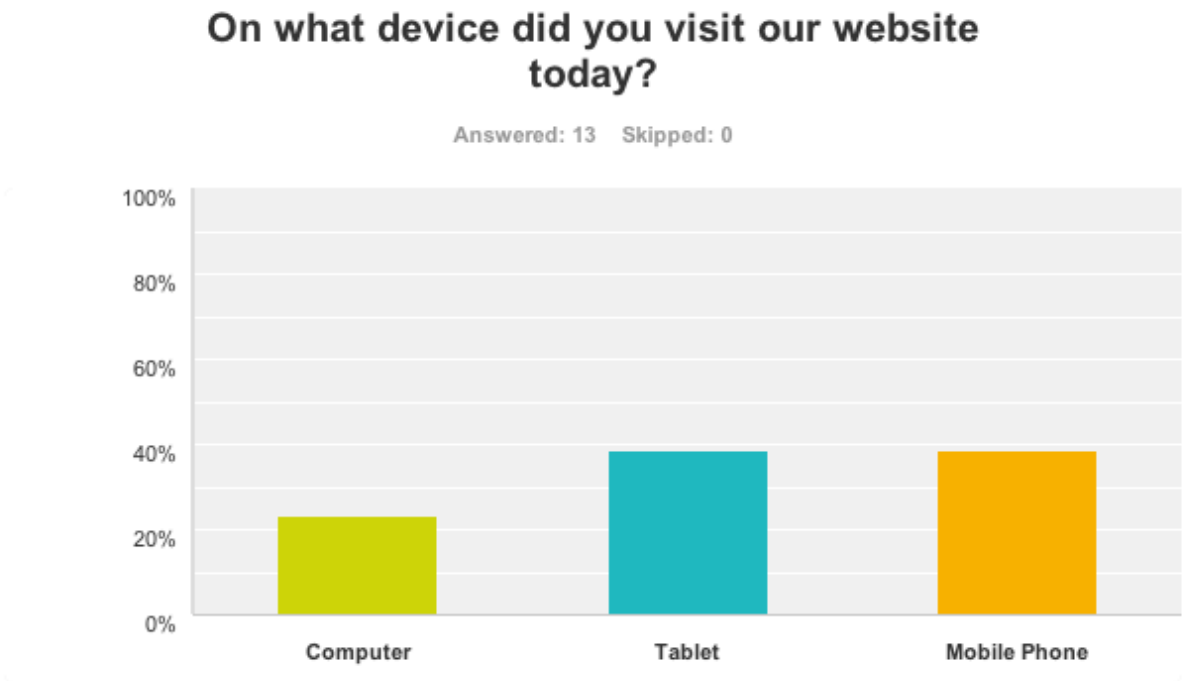
To evaluate that the system met the high standards and fulfilled the requirements, a questionnaire was devised. This collated 'user evaluation' of the system and provided some necessary feedback. The questionnaire was designed to accommodate the target user and their fast-paced lifestyle and so the questions were deliberately short and concise. A target pool of the 18-30 audience was screened which consisted of peer group, family relations, friends and associates in order to provide relevant and realistic results. To make the survey easy and accessible, survey monkey was the chosen platform adopted to manage the survey and a link was forwarded by email to the relevant people.

The survey results (Appendix E Survey Results) were positive from the all users who took the questionnaire. They all agreed the application was well designed and translated consistently across to mobile.

Some of the results provided very interesting feedback for post-survey analysis. From the demographic; 38% of users visited the application via a mobile device, while only 23% accessed the application via computer. Another interesting statistic showed that 38% of users who undertook the survey also visited from a tablet device. This shows testing for mobile devices was vital, but catering for the identified growth of mobile was vital insuring success. When analysing the design 53% of users agreed the application was aesthetically pleasing and found it to be extremely engaging while the remaining majority thought the application design was very engaging, highlighting that the time spent on researching styles and also then the paper prototyping exercise for the design stages was all vali. All the users agreed they could conduct the individual tasks highly efficiently in Projectified, but they identified tasks which they would feel would be of benefit when

working as a group project in a larger organization. This is something that could be developed in the future but with the limited time of the project meant that it wasn't able to be fulfillied within this brief. On a concluding note 69% of surveiers said they would highly recommend the survey to someone they knew, while 31% said they would moderately recommend the application.

The results show a positive forecast for Projectified. It was great to see the target audience highly appraise the aesthetics of the application, which helped deliver an intuitive UI experience. It was identified that two of the requirements were not fulfilled, which being aware of the imposed time restrictions, was a realistic outcome as it was clear in advance that some elements of further design would need to be conducted at a later stage, as the scope of the Protecified project far outweighs the current timescale. Moving forward this enables Projecitifed to further develop and continue to improve on and adopt additional features for larger organisations.



Answer Choices	Responses	
Computer	23.08%	3
Tablet	38.46%	5
Mobile Phone	38.46%	5
Total		13

figure 5 – Average visitor device ratings

6.3 Methodology Evaluation

Adopting an Agile methodology allowed the project to manage the sprint iterations effectively. Pre-planning the initial sprints meant they were pinpointed from the outset and were therefore able to be expanded. As always, awareness of timing was key and when the PERT analysis was adopted mid way through the project, this helped predict the cycle times more accurately of the remaining body of work and helped priortise the workload. Using the combination of Agile and PERT made for a robust development plan, PERT in particular highlighted risks associated with time, this allowed the project to reshuffle sprint iterations to ensure the project would finish on time and to a high standard. PERT proved to be a a valuable tool when gauging development time, by applying its unique formula it can hone precise cycles time while estimating. The Project life cycle can be found in Appendix E –Project Gantt Chart.

6.3 Plan Evaluation

In beginning the project the aim was to “create an intuitive, dynamic and harmonious application, that allows the user to create, read, update, delete information and manage their chosen projects by organising goals and objectives into manageable boards.” and at this point in the analysis, it has been achieved.

Key findings, through the use of surveys, portray clearly that users of the Projectified management tool agree with that the main project aim outlined above has been met. They felt the application communicated a strong visual appeal and dynamically responded to key calls to action when using. The seamless interface, colour palette and functional design meant that users found it easy to adopt and of great aid when organizing and planning their workload and projects.

The approach adopted was robust and successful, however the project could benefit from further refinement for group collaboration projects on a larger scale. The next iteration could deliver further beneficial features that the project needs in order for it to be a continual success. Balancing and juggling a mammoth project of this scale has been tough, but extremely beneficial to learning, developing and lastly time management. It has provided a foundation of detailed research and analysis and in doing this has provided a wealth of practiced and newly gained experience.

7. Conclusion

Before starting the Project it was clear that it would be a time consuming and detailed study, but to actually undertake it, has provided me with a clear outline of how much workactually goes into researching, developing and implementing an application of this stature. It has provided the project leader with a greater degree of accuracy as to how to effectively plan, research, refine, integrate and deliver a tailored solution for a large web application in the future, and has cemented the knowledge that has been gained over the last few years and driven a desire to learn more to enable the further development of the project..

Although this report covered a broad spectrum, it clearly identifies all the key aspects of the project: a coherent aim, objective and goal. Further investigation and experimentation would be of great benefit to identify further features that could be of use in the management of larger projects. This is an area that could greatly gain from more time and attention, while also analysing further the main competitors within the identified market. Implementing a robust methodology concluded that a project can meet a deadline imposed by an academic institution by analysing workload and adjust accordingly using PERT. Results from the user evaluation dictates the system has a prosperous future but needs some minor adjustments for mobile.

7.1 Future Work

Further work and assesments are needed to implement further changes. Moreover, not all requriements were met, meaning there is room for the project to fulfill the missing requierements at a later date. There are some features that the project lead would like to implement over a period of time to see it continually grow and flourish.

- Rearranging task boards – Many users thought this would be an advantageous feature to offer. It would further enhance the controls given to the user, who could then choose to organise and priortise the workload as they so desired.

- Deadline Date – A feature which is high on the agenda to implement. This could facilitate the option to create realistic deadline that would aid toward effective project management.

- Group Collaboration – The additional work within the database has already been initially structured to accommodate this feature. Further development could make this feature part of the tool and would improve the application greatly by allowing it to develop into a new era of online communication and project management.

Delivering these features would certainly enhance the experience for the user and help drive the traction within Projectified, as longer times would no doubt be spent within the application. Projectified is looking towards the future with a fresh and an adaptive approach, spurred on by the findings outlined within the report.

7.2 Role Reflection

As with everything, there is always an element of continual further refinement to enhance the offering of this tool, as with others, as due to the ongoing flux of the adaptive society with which we live in, nothing is constant. Notwithstanding this future refinement, there is a clear feeling that the project undertaken has been a success and with this also comes a sense of pride at the work dedicated to the project over the last academic year and this will be knowledge and awareness that will go forward in the ensuing career that lies ahead. As this project was mammoth in size, the project lead was required to be dynamic and intuitive in their approach. This meant areas of design, development, research and project management were all singularly managed to meet the end goal: delivering a completed solution. It has to be noted how the project lead has grown in confidence over the course of this project and over the course of the year to develop a skill set that would be advantageous to employment.. The project has had its trials and tribulations, but in reality this has stoked the fires within to ensure that on leaving university that I continue learning , adapting and growing in this field. The time, effort and dedication to this report, although all consuming at times, have certainly been worthwhile and to see it completed means that finally the fruits of the labour can be witnessed!

References

[1] Story mapping user stories & 'buy a feature' prioritisation. 2014. Story mapping user stories & 'buy a feature' prioritisation. [ONLINE] Available at: http://www.boost.co.nz/blog/2011/05/story-mapping-prioritisation/. [Accessed 14 April 2014].

[2] Design patterns. 2014. Design patterns. [ONLINE] Available at: http://ui-patterns.com/patterns. [Accessed 18 April 2014].

[3] Pttrns - Mobile User Interface Patterns. 2014. Pttrns - Mobile User Interface Patterns. [ONLINE] Available at: http://pttrns.com/. [Accessed 18 April 2014].

[4] 2014. Understanding the Z-Layout in Web Design - Tuts+ Web Design Article. [ONLINE] Available at: http://webdesign.tutsplus.com/articles/understanding-the-z-layout-in-web-design--webdesign-28. [Accessed 18 April 2014].

[5] Databases - Tuts+ Code Category. 2014. Databases - Tuts+ Code Category. [ONLINE] Available at: http://code.tutsplus.com/categories/databases. [Accessed 18 April 2014].

[6] 11 important database designing rules which I follow - CodeProject. [ONLINE] Available at: http://www.codeproject.com/Articles/359654/11-important-database-designing-rules-which-I-foll. [Accessed 18 April 2014].

[7] RightNow Fatal Error. 2014. RightNow Fatal Error. [ONLINE] Available at: http://help.filemaker.com/app/answers/detail/a_id/9922/~/understanding-and-creating-many-to-many-relationships-in-filemaker-pro. [Accessed 18 April 2014].

[8] Stack Overflow. 2014. mysql - SQL join multiple tables - Stack Overflow. [ONLINE] Available at: http://stackoverflow.com/questions/9853586/sql-join-multiple-tables. [Accessed 18 April 2014].

[9] 2014. IBM Information Management Software for z/OS Solutions Information Center. [ONLINE] Available at: http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.db2z10.doc.intro%2Fsrc%2Ftpc%2Fdb2z_relationshipentities.htm. [Accessed 18 April 2014].

[10] URL Rewriting for Beginners - Web Development in Brighton - Added Bytes. [ONLINE] Available at: http://www.addedbytes.com/articles/for-beginners/url-rewriting-for-beginners/. [Accessed 18 April 2014].

[11] 2014. PHP Best Practices: a short, practical guide for common and confusing PHP tasks. [ONLINE] Available at: https://phpbestpractices.org/. [Accessed 18 April 2014].

[12] What to know about building a website . 2014. What to know about building a website . [ONLINE] Available at: http://hashphp.org/. [Accessed 18 April 2014].

[13] YouTube. 2014. Symfony2 Tutorial 4 - Html5 Boilerplate & Twig - YouTube. [ONLINE] Available at: http://www.youtube.com/watch?v=elFXIglFauk. [Accessed 18 April 2014].

[14] Overflow. 2014. Twig runs in 'if variable is defined', even when variable is defined - Stack Overflow. [ONLINE] Available at: http://stackoverflow.com/questions/13941324/twig-runs-in-if-variable-is-defined-even-when-variable-is-defined. [Accessed 18 April 2014].

[15] . 2014. . [ONLINE] Available at: http://silex.sensiolabs.org/pdf/Silex.pdf. [Accessed 18 April 2014].

[16] 2014. Scratching the Surface of Silex | Iterate: Writing the Web. [ONLINE] Available at: http://iterate.blendinteractive.com/scratching-the-surface-of-silex/. [Accessed 18 April 2014].

[17] HTTP Status Codes. 2014. HTTP Status Codes. [ONLINE] Available at: http://www.restapitutorial.com/httpstatuscodes.html. [Accessed 18 April 2014].

[18] Open Sans | Typ.io: Be better at typography. 2014. Open Sans | Typ.io: Be better at typography. [ONLINE] Available at: http://www.typ.io/s/paf4. [Accessed 18 April 2014].

[19] Create, read, update and delete - Wikipedia, the free encyclopedia. [ONLINE] Available at: http://en.wikipedia.org/wiki/Create,_read,_update_and_delete. [Accessed 18 April 2014].

[20] Transaction processing - Wikipedia, the free encyclopedia. [ONLINE] Available at: http://en.wikipedia.org/wiki/Transaction_processing. [Accessed 18 April 2014].

[21] The Controller and the View (1_4) - Symfony. [ONLINE] Available at: http://symfony.com/legacy/doc/jobeet/1_4/en/04?orm=Doctrine. [Accessed 18 April 2014].

[22] Successful Web Development Methodologies Article. 2014. Successful Web Development Methodologies Article. [ONLINE] Available at: http://www.sitepoint.com/successful-development/. [Accessed 18 April 2014].

[23] Agile Vs. Waterfall: Evaluating The Pros And Cons. 2014. Agile Vs. Waterfall: Evaluating The Pros And Cons. [ONLINE] Available at: https://www.udemy.com/blog/agile-vs-waterfall/. [Accessed 18 April 2014].

[24] 2014. . [ONLINE] Available at: http://web2.concordia.ca/Quality/tools/20pertchart.pdf. [Accessed 18 April 2014].

[25] What are the steps involved in PERT analysis. [ONLINE] Available at: http://wiki.answers.com/Q/What_are_the_steps_involved_in_PERT_analysis#slide=20. [Accessed 18 April 2014].

[26] . 2014. [ONLINE] Available at: http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf. [Accessed 18 April 2014].

[27] . 2014. . [ONLINE] Available at: http://homepages.laas.fr/kader/Robertson.pdf. [Accessed 18 April 2014].

[28] UML basics: An introduction to the Unified Modeling Language. 2014. UML basics: An introduction to the Unified Modeling Language. [ONLINE] Available at: http://www.ibm.com/developerworks/rational/library/769.html. [Accessed 18 April 2014].

[29] Controller (current) - Symfony. 2014. Controller (current) - Symfony. [ONLINE] Available at: http://symfony.com/doc/current/book/controller.html. [Accessed 18 April 2014].

[30] Testing the app - App Center | MDN. 2014. Testing the app - App Center | MDN. [ONLINE] Available at: https://developer.mozilla.org/en-US/Apps/Tutorials/General/Testing_the_app. [Accessed 18 April 2014].

[31] Ólafur Arason - Google+ - If You Are Designing Your Own REST backend You're Doing It…. 2014. Ólafur Arason - Google+ - If You Are Designing Your Own REST backend You're Doing It…. [ONLINE] Available at: https://plus.google.com/+%C3%93lafurArason/posts/fCmaosPgmwN. [Accessed 18 April 2014].

[32] Usability Testing for Mobile. 2014. Usability Testing for Mobile. [ONLINE] Available at: http://www.nngroup.com/articles/mobile-usability-testing/. [Accessed 18 April 2014].

[33] Flexible Usability Testing Tips. 2014. Flexible Usability Testing Tips. [ONLINE] Available at: http://www.nngroup.com/articles/flexible-usability-testing/. [Accessed 18 April 2014].

[34] Getting started · Bootstrap . 2014. Getting started · Bootstrap . [ONLINE] Available at: http://getbootstrap.com/getting-started/. [Accessed 18 April 2014].

Bibliography

[1] Ryan Benedetti, 2011. Head First jQuery. 1 Edition. O'Reilly Media.

[2] Bill Scott, 2009. Designing Web Interfaces: Principles and Patterns for Rich Interactions. 1 Edition. O'Reilly Media.

[3] Jenifer Tidwell, 2011. Designing Interfaces. Second Edition Edition. O'Reilly Media.

[4] Jason Lengstorf, 2013. Realtime Web Apps: With HTML5 WebSocket, PHP, and jQuery. 1 Edition. Apress.

[5] 2009. jQuery Cookbook: Solutions & Examples for jQuery Developers (Animal Guide). 1 Edition. O'Reilly Media.

Appendix

Appendix A

Appendix A contains all the requirements for the Projectified Project as outlined in Section 2 of the report.

Requirement ID: RID 1

Requirement Type: System Functional

Description: System will allow users to register their credentials

Rational: People will need to access the system, to do this they must have valid credentials stored within a database.

Source: Feature that is necessary to access the system

Fit Criteria: This can be determined by the number of records stored within the database, or by use of Google analytics program to determine the success rate

Priority Level: 5

Dependencies: This relies on an accessible domain, the site must be live in order to access this feature.

Requirement ID: RID2

Requirement Type: System Functional

Description: System will allow users to login to Projectified

Rational: This will allow people to access the Projectified system and use the web application.

Source: A feature necessary to grow the user base of Projectified.

Fit Criteria: Database population increase

Priority Level: 5

Dependencies: RID1

Requirement ID: RID3

Requirement Type: System Functional

Description: System will allow the user to create project

Rational: This allows the user to create a project. Unlocking this feature opens up the rest of the application.

Source: A feature necessary by the user, stated by the developer

Fit Criteria: Database population increase specifically within the project entitiy

Priority Level: 5

Dependencies: RID 2

Requirement ID: RID4

Requirement Type: System Functional

Description: The system will allow the user to add a Task Board to their project

Rational: This helps to categorise tasks into relevant manageable boards that aids towards organisation.

Source: A feature necessary for the user outlined by the developer.

Fit Criteria: Expect to see an increase in population within the category entity

Priority Level: 5

Dependencies: RID3

Requirement ID: RID5

Requirement Type: System Functional

Description: The system will allow a user to add a task to a specific Task Board.

Rational: The user can add their requirements to their project. This allows the user to take control.

Source: A feature necessary to the user stated by the developer

Fit Criteria: Database population increase within the task entity

Priority Level: 5

Dependencies: RID 4

Requirement ID: RID6

Requirement Type: System Functional

Description: The system will accommodate the user when needing to delete a task.

Rational: If a task becomes obsolete to the Task Board the user will want to remove it from the front-end. This will also remove the task from the backend system.

Source: A feature necessary stated by the developer that would benefit the user.

Fit Criteria: Database will fluxiate within this particular field, the deleted attribute of the task entity will be populate with deleted dates.

Priority Level: 4

Dependencies: RID 5

Requirement ID: RID7

Requirement Type: System Functional

Description: The system will allow the user to delete a task board

Rational: To de-clutter the user interface with unused boards

Source: A feature necessary stated by the developer

Fit Criteria: The database category entity will have an increase in deleted dates within the deleted attribute.

Priority Level: 4

Dependencies: RID 4

Requirement ID: RID8

Requirement Type: System Functional

Description: The system will allow the user to delete a project.

Rational: To declutter the user interface with unused projects

Source: A feature necessary stated by the developer that would better the user experience

Fit Criteria: Database Project entity will have an increase number of deleted values applied to the attribute deleted.

Priority Level: 4

Dependencies: RID 3

Requirement ID: RID9

Requirement Type: Look and feel

Description: Dashboard view to allow the user to get a quick overview of current projects.

Rational: Allows the user to quickly see all current projects and activities that are going on within their own private space

Source: A feature necessary stated by the designer

Fit Criteria: When data and design merge to create an engaging UX**.**

Priority Level: 4

Dependencies: RID 1,2,3

Requirement ID: RID10

Requirement Type: Look and feel, System

Description: The system will allow the user to add unique colours to their tasks.

Rational: Dynamic highlighting will benefit the user in a number of ways. This allows them to highlight important segments, just like a highlighter.

Source: A feature necessary stated by the designer

Fit Criteria: When data and design merge to create an engaging UX.

Priority Level: 3

Dependencies: RID 5

Requirement ID: RID11

Requirement Type: System Functional

Description: The user can recover a deleted project

Rational: If the user accidently deletes a project, they will be able to get it back

Source: A feature necessary stated by the developer

Fit Criteria: When a user emails Projectified asking for a project restore**.**

Priority Level: 4

Dependencies: RID3

Requirement ID: RID12

Requirement Type: Look and Feel

Description: A coherent booklet for brand guidelines

Rational: Ensures brand is communicated consistently across market channels

Source: A requirement set by the designer for the brand

Fit Criteria: Consistency within the booklet and a final draft.

Priority Level: 3

Dependencies: Brand name must be already agreed upon

Requirement ID: RID13

Requirement Type: System Functional

Description: User will be able to update Task Board title

Rational: It will be useful if the Task Category needs to change the user can amend and update the change to reflect the title change

Source: A requirement set by the user

Fit Criteria: Once a user can inline edit and update the change, this will show within the database within the category entity along with the updated date attribute changing to reflect the update.

Priority Level: 4

Dependencies: RID 4

Requirement ID: RID14

Requirement Type: System Functional / Look Feel

Description: System to display updates within a timeline feature

Rational: Provides the user with useful feedback, this shows what is currently going on within the system, ie updates, created and deleted attributes.

Source: A requirement set by the developer

Fit Criteria: Once the feature has been integrated into the system and is fully tested and working

Priority Level: 3

Dependencies: RID 3, 4, 5

Requirement ID: RID15

Requirement Type: System Functional / Look Feel

Description: System automatically notifies the user of an update, delete and created situation

Rational: Will alert the user if an update occurs, will be more effective for group collaboration, but also important to know the actions taken within your profile.

Source: A requirement set by the developer

Fit Criteria: Once the feature has been integrated into the system and is fully tested and working it can be noted as a success

Priority Level: 3

Dependencies: RID 3, 4, 5

Requirement ID: RID16

Requirement Type: System Functional

Description: The user can invite friends or colleagues to their project

Rational: This will allow group collaboration within Projectified, having this feature enables people to communicate and therefore make group organization and interaction better between teams.

Source: A requirement set by the developer and user

Fit Criteria: Once the feature has been integrated into the system and is fully tested and working it can be noted as a success

Priority Level: 2

Dependencies: RID 2, 3, 4, 5

Requirement ID: RID17

Requirement Type: System Functional

Description: A user can rearrange their tasks in order of precedence

Rational: Makes priortising task in order of importance easy, this would aid the user when using the system

Source: A requirement set by the user

Fit Criteria: Once the feature has been integrated into the system and is fully tested and working it can be noted as a success

Priority Level: 2

Dependencies: RID 2, 3 , 4, 5

Requirement ID: RID18

Requirement Type: System Functional

Description: A user can sign out of the system easily

Rational: This will allow the user to leave the system securely

Source: A requirement set by the user

Fit Criteria: Once the user can sign out of the system it will be a success

Priority Level: 3

Dependencies: RID 1

Requirement ID: RID19

Requirement Type: System Functional

Desription: The user can inline edit and update task description

Rational: If the user makes a mistake with their spelling or types the wrong phrase, it will allow the user to rectify the issue.

Source: A requirement set by the user

Fit Criteria: The database especially the task entity was show an updated time on the update attribute

Priority Level: 2

Dependencies: RID 5

Requirement ID: RID20

Requirement Type: System Functional

Description: The user can inline edit and update task title

Rational: If the user makes a mistake with their spelling or types the wrong title it will allow the user to rectify the issue.

Source: A requirement set by the user

Fit Criteria: The database especially the task entity was show an updated time on the update attribute. By showing the change on the front end will also be deemed as a success

Priority Level: 2

Dependencies: RID 5

Requirement ID: RID21

Requirement Type: Look and feel

Description: The system will respond and adapt to mobile devices

Rational: For optimal viewing experience Projectified will need to adapt and present its data in the most accessible way possible.

Source: A requirement set by the designer

Fit Criteria: Testing on mobiles devices will be completed through an emulator. This will show the consistency through multiple viewing platforms

Priority Level: 4

Dependencies: RID 2, 3, 4, 5

Requirement ID: RID22

Requirement Type: System Functional

Description: System will support file upload capability

Rational: File uploads would help support any additional information that may be needed while working on a Project

Source: A requirement from the user

Fit Criteria: Testing the software requirement to support an array of file types would be key.

Priority Level: 2

Dependencies: RID 2, 3, 4, 5

Requirement ID: RID23

Requirement Type: System Functional

Description: The user can promote a task into its own category

Rational: If a task supports too many sub tasks the user can decide to change the task into its own separate Task Category, if it is deemed necessary.

Source: A requirement from the user and developer

Fit Criteria: When the product is fully functional and is tested it will determine the success.

Priority Level: 2

Dependencies: RID 2, 3, 4, 5

Requirement ID: RID24

Requirement Type: System Functional

Description: A task will support multiple sub tasks

Rational: If a task can support additional information, instead of adding more tasks a user can add information that will create a cleaner UX

Source: A requirement from the user

Fit Criteria: When the product is fully functional and is tested it will determine the success.

Priority Level: 3

Dependencies: RID 2, 3, 4, 5

Requirement ID: RID25

Requirement Type: System Functional / Look and feel

Description: System will allow a user to view / read a project

Rational: The user will need to see the project created via the front-end

Source: A requirement from the user and developer

Fit Criteria: When the user adds a project, the user will see this in the navigation and dedicated area this will be a success

Priority Level: 5

Dependencies: RID 1, 2, 3,

Requirement ID: RID26

Requirement Type: System Functional

Description: System will validate a users login credentials

Rational: The user's credentials will need to be valid in order to gain access to Projectified.

Source: A requirement from the developer

Fit Criteria: When the user logs in successfully there will be no error messages, but if the user tries to login with invalid credentials this will be flagged and displayed via the front-end.

Priority Level: 3

Dependencies: RID 1

Requirement ID: RID27

Requirement Type: System Functional

Description: The system will validate the users registration details to establish if they are valid

Rational: This will stop any malicious strings or content being injected into the database.

Source: A requirement from the developer

Fit Criteria: If the user tried to submit data in the incorrect format, red errors messages will appear indicating a mistake has been made. Successful completed will see an increase of registered users

Priority Level: 3

Dependencies: RID 1

Requirement ID: RID28

Requirement Type: System Functional

Description: When updating a task title or description the system will check to see if it exceeds a sting length of 300 for the title and 2000 for a description

Rational: This will stop any spam that might try and enter the database

Source: A requirement from the developer

Fit Criteria: Will be successful when the user inserts data that exceeds 300 characters and a return statement is made.

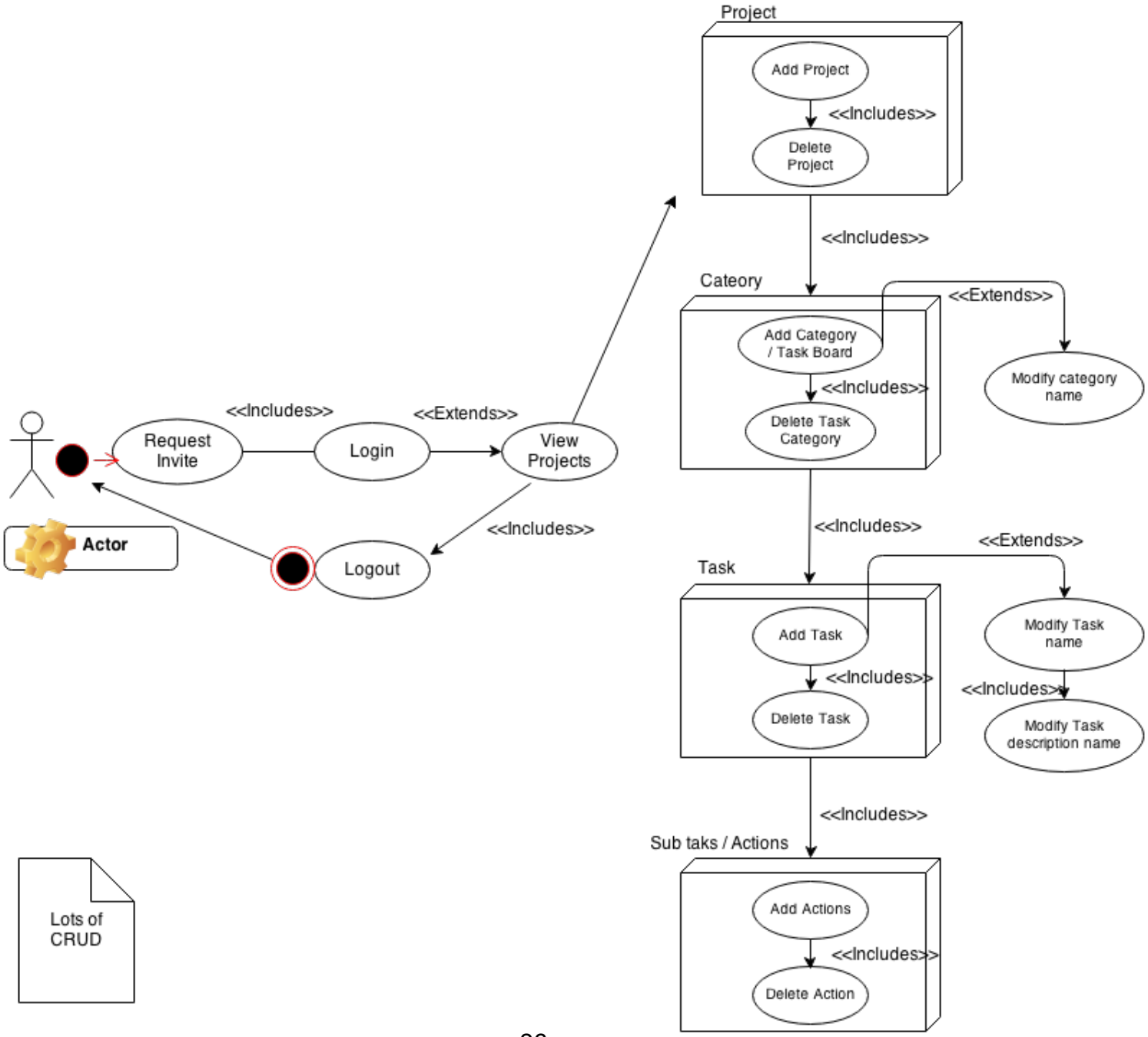
Priority Level: 3

Dependencies: RID 5

Requirement ID	Priority	Description
1	5	System will user to register
2	5	System will allow user to login
3	5	System will allow user to create a project
4	5	System will allow user to create a task board
5	5	System will allow user to add a task
25	5	System will allow a user to view / read a project
21	4	The application will respond and adapt to multiple devices for optimised viewing
6	4	The system will accommodate the user when needing to delete a task
7	4	The system will allow the user to delete a task board
8	4	System will allow user to delete a project
9	4	Provide a dashboard view to allow the user to get a quick overview of current projects
11	4	System will be able to recover a deleted project if the user requests a restore
13	4	The user will be able to amend a task board title
20	4	System will respond to mobile devices
10	3	The system will allow the user to add a colour to the task
14	3	System will display a timeline of all activity within the system
18	3	A user can sign out of the system securely
15	3	System will notify user of any changes, created, updated and deleted
12	3	A coherent booklet that outlines brand guidelines
18	3	A user can sign out of the system securely
23	3	System will facilitate additional sub tasks off a main task
24	3	A task will support multiple sub tasks
26	3	System will validate user login credentials
27	3	The system will validate the users registration details to establish if they are valid

16	2	The user can invite colleagues to their project
17	2	The user can rearrange their task in order of precedence
19	2	System will allow user to inline edit and update task descrption
20	2	The user can inline edit and update task title
22	2	The system will support file upload capability
23	2	The user can promote a task into its own category

Appendix A.I


Project
Add Project
<<Includes>>
Delete Project
<<Includes>>
Cateory
<<Extends>>
Add Category / Task Board
<<Includes>>
Delete Task Category
Modify category name
<<Includes>>
Task
<<Extends>>
Add Task
<<Includes>>
Delete Task
Modify Task name
<<Includes>>
Modify Task description name
<<Includes>>
Sub taks / Actions
Add Actions
<<Includes>>
Delete Action
<<Includes>>
<<Extends>>
Request Invite
Login
View Projects
<<Includes>>
Logout
Actor
Lots of CRUD

Flow of Main Events

1. **Request Invite**

This use case starts when the user requests an invite to the projectified system. They will be emailed with a generated password access the system.

2. **LogIn**

The user access's the Projectified system. At this point the user will enter his or hers login credentials, the system then validates the user.

3. **View Projects**

The system displays a list of Projects that the user has created. This allows the user to traverse to a specific project. The user will click on a project that is listed to access it.

4. **Create Project / Add Project**

The system prompts the user to create a project by inviting the user to click 'Add Project'. The user then enters the title of the Project in the provided modal window.

5. **Select Project**

The user must then select the project he or she wants to work on via the sidebar menu system. The system then retrieves the information related to that specific project including category boards, tasks and sub tasks.

6. **Create Project specific Categories / Add Category**

The user can then add a category board to the project. This will hold the relevant tasks related to that category. When the user clicks “Add Category”, he or she will be prompted with a modal to insert the title of that category. On confirmation the category board will appear on the front end.

7. Create Category Related Tasks / Add Task

At this point once a category board has been initiated the user can then create tasks. The user clicks on “Add Task”, this will activate a modal window which the user inputs a title and description. On confirming this it will appear on the front end.

8. **Create Actions / Sub tasks**

Once the system has compiled tasks within categories, the user may wish to expand on their task and further support it with further details. The user can do this by clicking on “Add Action”. A new modal will appear this is where the user can add more descriptive tasks or actions to be completed.

Alternate Flow

Since the system has a core flow, there is the chance a user may want to do something else within the system or deviate from the main interaction. These actions would not be seen to occur often, only initiated if the user has made an error.

1. **Modify Category Title**

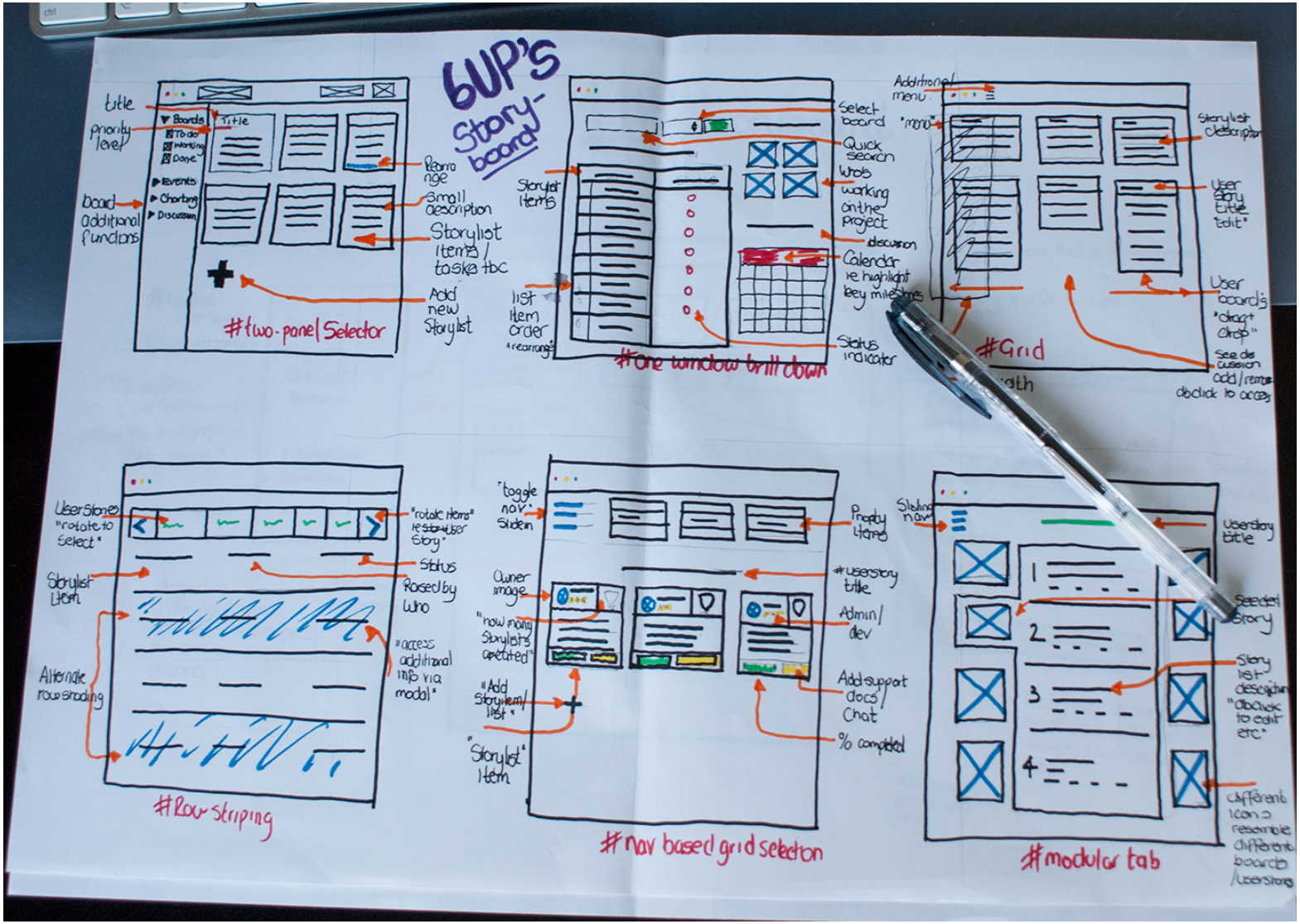
The user has already created a category that has been saved to the database. As the system is currently displaying the listed categories, the user wishes to change the title to reflect something more meaningful. The user clicks on the title, and by method of inline the user can edit the title.

2. **Delete Project**

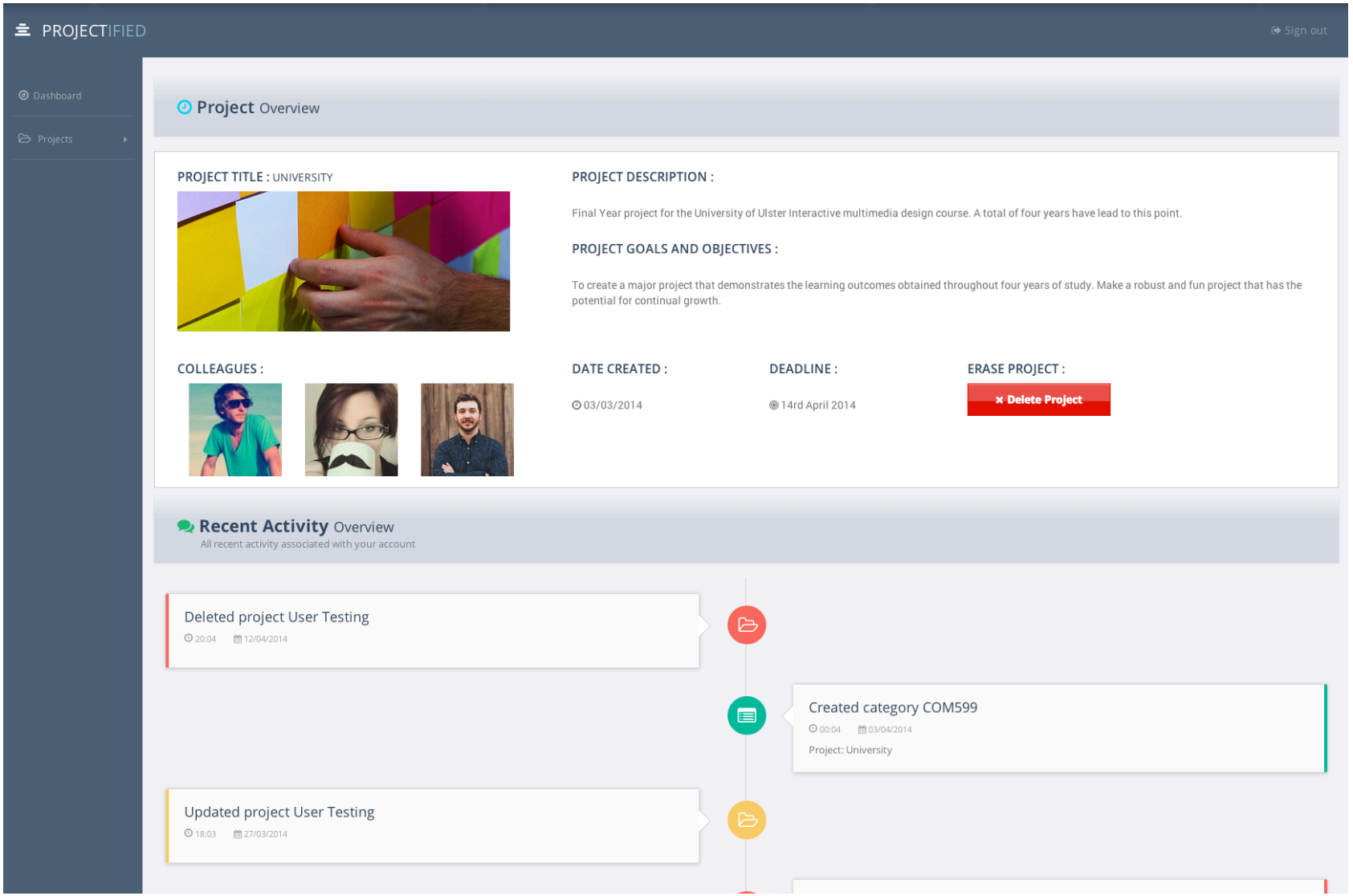
This option depends on the user having created a project. Simply the user visits the dashboard page that lists all current projects. The user finds by method of scanning the project they wish to delete. The user clicks on the trash icon. The system prompts the user to verify the deletion. The user verifies the deletion, the system timestamps the delete that in turn preforms a soft delete.

Appendix B

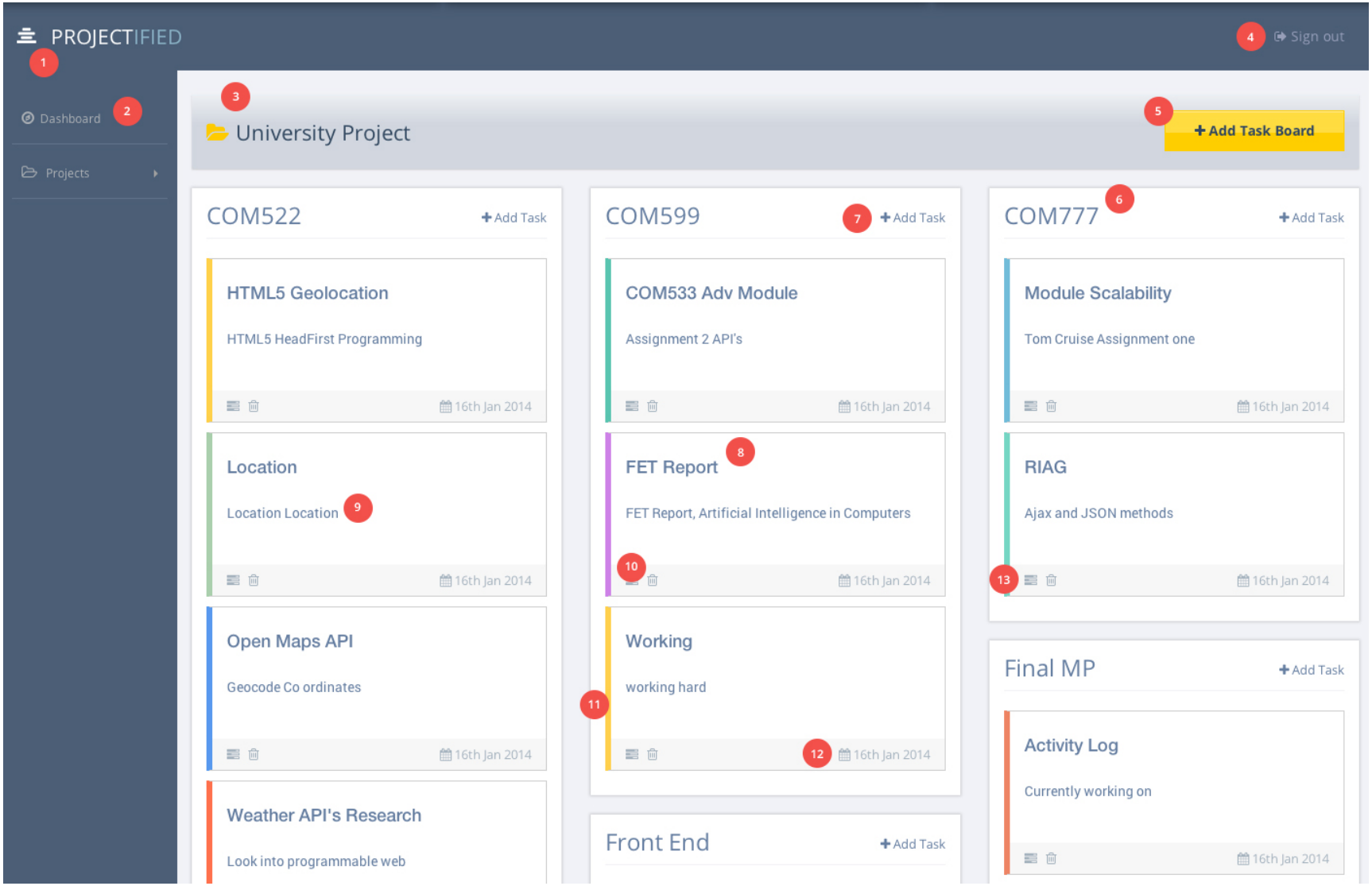
Appendix B – 6UP



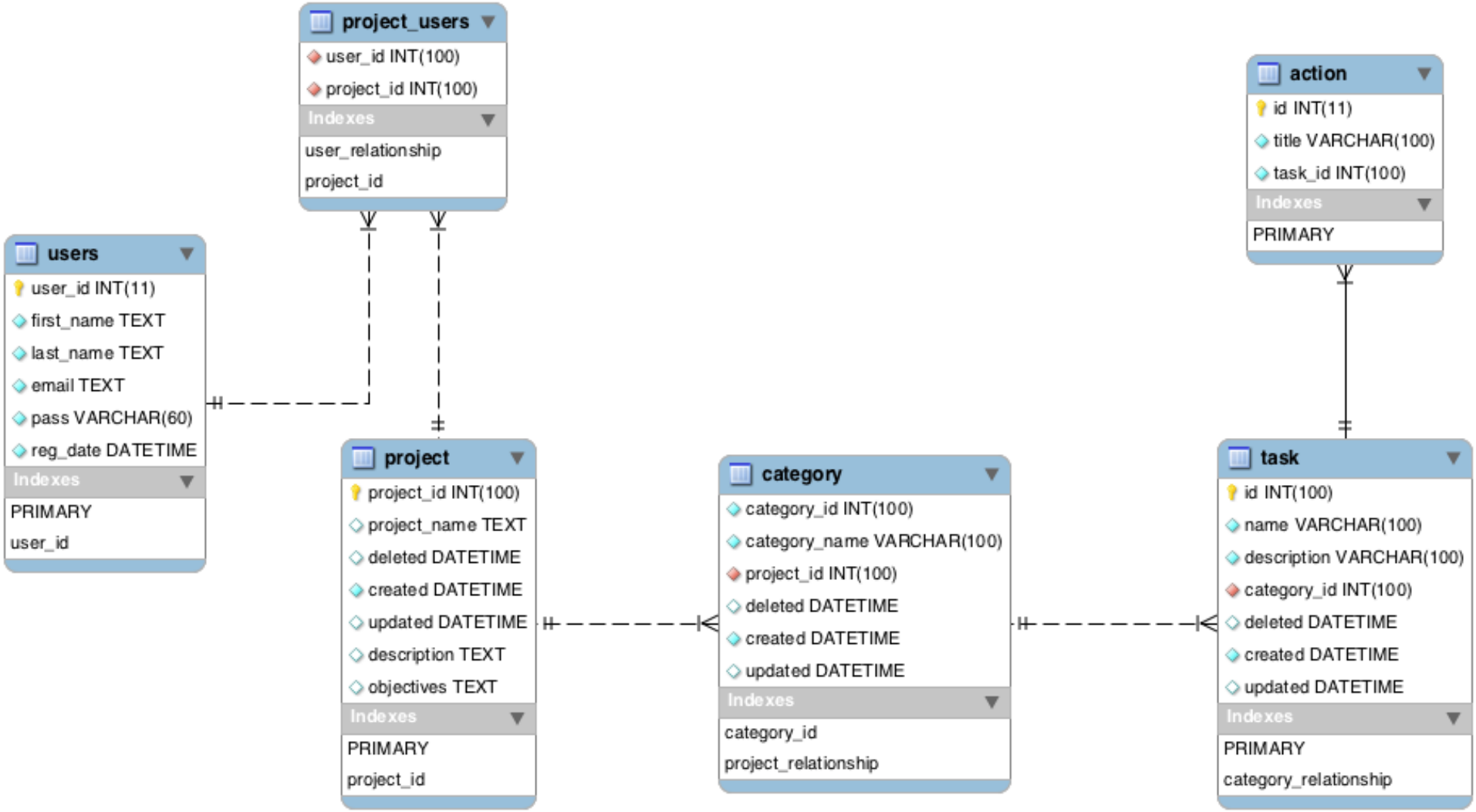
Appendix B – Dashboard View



Appendix B – Main Project Page



Appendix B – ERD



Appendix C

Appendix C – Initial Row Dump

Shows initial data dump using var_drump($rows). Shows data structure, includes a lot of repetitive information.

```
array(16) {
    [0]=>
    array(6) {
        ["project_name"]=>
        string(10) "University"
        ["category_id"]=>
        string(1) "2"
        ["category_name"]=>
        string(6) "COM522"
        ["task_id"]=>
        string(1) "7"
        ["task_name"]=>
        string(17) "HTML5 Geolocation"
        ["task_description"]=>
        string(28) "HTML5 HeadFirst Programming "
    }
    [1]=>
    array(6) {
        ["project_name"]=>
        string(10) "University"
        ["category_id"]=>
        string(1) "2"
        ["category_name"]=>
        string(6) "COM522"
        ["task_id"]=>
        string(2) "86"
        ["task_name"]=>
        string(8) "Location"
        ["task_description"]=>
        string(17) "Location Location"
    }
```

Appendix C – Structured Data Dump

```
array(3) {
  ["id"]=>
  string(1) "1"
  ["name"]=>
  string(10) "University"
  ["categories"]=>
  array(7) {
    [0]=>
    array(3) {
      ["id"]=>
      string(1) "2"
      ["name"]=>
      string(6) "COM522"
      ["tasks"]=>
      array(4) {
        [0]=>
        array(3) {
          ["id"]=>
          string(1) "7"
          ["name"]=>
          string(17) "HTML5 Geolocation"
          ["description"]=>
          string(28) "HTML5 HeadFirst Programming "
        }
        [1]=>
        array(3) {
          ["id"]=>
          string(2) "86"
          ["name"]=>
          string(8) "Location"
          ["description"]=>
          string(17) "Location Location"
        }
        [2]=>
        array(3) {
          ["id"]=>
          string(2) "80"
          ["name"]=>
          string(13) "Open Maps API"
          ["description"]=>
          string(20) "Geocode Co ordinates"
        }
        [3]=>
        array(3) {
          ["id"]=>
          string(2) "81"
          ["name"]=>
          string(22) "Weather API's Research"
          ["description"]=>
          string(26) "Look into programmable web"
        }
      }
    }
    [1]=>
    array(3) {
      ["id"]=>
      string(1) "1"
      ["name"]=>
      string(6) "COM599"
      ["tasks"]=>
      array(3) {
        [0]=>
        array(3) {
          ["id"]=>
          string(2) "75"
          ["name"]=>
          string(17) "COM533 Adv Module"
          ["description"]=>
          string(18) "Assignment 2 API's"
```

Appendix D

Append D – Black Box Testing

Test ID	Action	Expected Results	Pass / Fail	Comments
TID1	Bring the window down in size	The app will adapt to the change	Pass	
TID2	Load the app in different browsers	The look will be consistent through the application	Pass	
TID3	Load the homepage	All elements fadeIn when page scroll initiated	Pass	
TID4	On the homepage click Beta Sign In	A modal window appears	Pass	
TID5	In the login modal enter credentials, click submit	The login will fadeout and redirect into app	Pass	
TID6	Add a project by clicking on the button located in the sidebar	A modal window will appear asking for a project title	Pass	
TID7	Can you view project via the sidebar	This will take you to the specific project	Pass	
TID8	Click add Task Board	Loads a modal window asking for credentials	Pass	
TID9	Insert credentials task board modal	Modal window has area to insert credentials	Pass	
TID10	Click on Task Board modal submit	The modal window closes and values added to the new board	Pass	
TID11	Click Add New Task	Displays a new modal window	Pass	
TID12	Insert title & description	Input fields will take new data	Pass	
TID13	Click the modal submit button	Modal disappears new Task is added to the front end	Pass	
TID14	Click the title of the board	This will show an option to change the	Pass	

		board title		
TID14	Enter new title and click the tick option	This will show an update on the top right corner and new value is added to front end	Pass	
TID15	Click a Task title	This will show an option to change the task title	Pass	
TID16	Insert new title and click the tick option	This will show an update notification on the top right corner and update with new title	Pass	
TID16	Click on the task description	This will show an option to invite to change the description	Pass	
TID17	Insert a new description, click the tick option	This will show an update status top right corner with the change and add the new description to the front end	Pass	
TID18	Hover over the task icon	This should show a tool tip indicating delete task	Pass	
TID20	Click the trash icon	This will delete a task from view	Pass	
TID21	Click the dashboard via the sidebar	This will load the contents for the overview of your projects	Pass	
TID22	Click on Project Description	Shows a new textarea containing the text to edit	Pass	
TID23	Insert new text, click the tick icon to update	New text will appear in place of the previous	Pass	
TID24	Click inside Project Goals and Objectives	A textarea will appear in the area of the text	Pass	
TID25	Update the relevant text inside the new textarea, click on the tick button	The new text will append into the area provided, the old text will be removed	Pass	
TID26	Click the nav icon	The navigation panel	Pass	

	beside Projectified logo	will slide to accommodate small screen sizies		
TID27	Navigate to the sign out option this will allow to sign out of the navigation successfully	This will redirect to the homepage meaning a successful sign out	Pass	

Appedix D – Browser Stats

http://www.w3schools.com/browsers/browsers_stats.asp

Browser Statistics

2014	Internet Explorer	Firefox	Chrome	Safari	Opera
February	9.8 %	26.4 %	56.4 %	4.0 %	1.9 %
January	10.2 %	26.9 %	55.7 %	3.9 %	1.8 %
2013	**Internet Explorer**	**Firefox**	**Chrome**	**Safari**	**Opera**
December	9.0 %	26.8 %	55.8 %	3.8 %	1.9 %
November	10.5 %	26.8 %	54.8 %	4.0 %	1.8 %
October	11.7 %	27.2 %	54.1 %	3.8 %	1.7 %
September	12.1 %	27.8 %	53.2 %	3.9 %	1.7 %
August	11.8 %	28.2 %	52.9 %	3.9 %	1.8 %
July	11.8 %	28.9 %	52.8 %	3.6 %	1.6 %
June	12.0 %	28.9 %	52.1 %	3.9 %	1.7 %
May	12.6 %	27.7 %	52.9 %	4.0 %	1.6 %
April	12.7 %	27.9 %	52.7 %	4.0 %	1.7 %
March	13.0 %	28.5 %	51.7 %	4.1 %	1.8 %
February	13.5 %	29.6 %	50.0 %	4.1 %	1.8 %
January	14.3 %	30.2 %	48.4 %	4.2 %	1.9 %

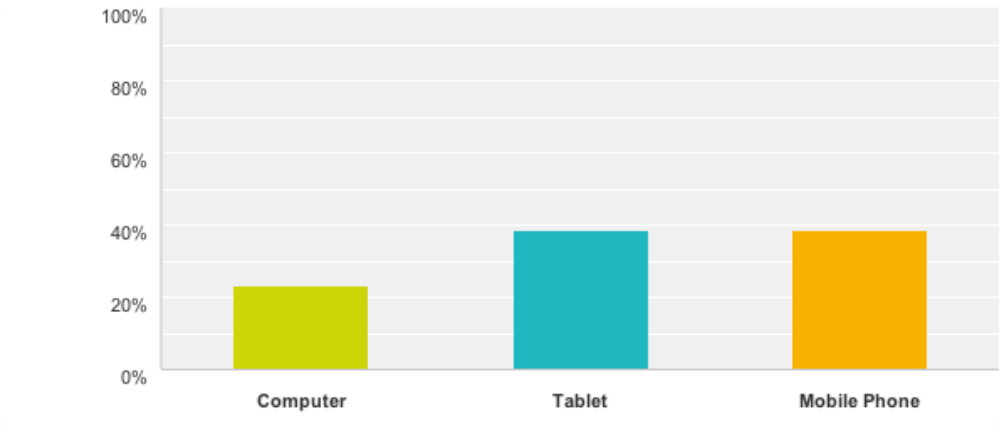
Appendix E

Appendix E - Survey Results

Q1 Customize Export

On what device did you visit our website today?

Answered: 13 Skipped: 0

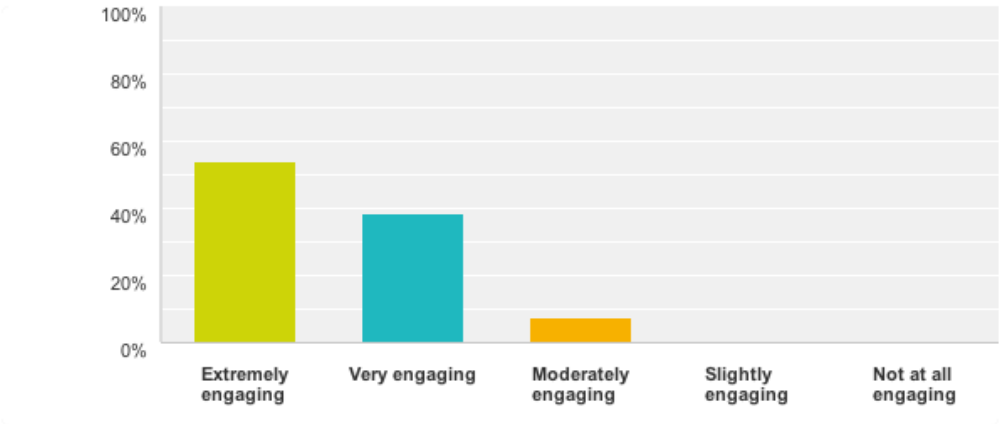


Answer Choices	Responses	
Computer	23.08%	3
Tablet	38.46%	5
Mobile Phone	38.46%	5
Total		13

Q2 Customize Export

How engaging is the design of the website?

Answered: 13 Skipped: 0



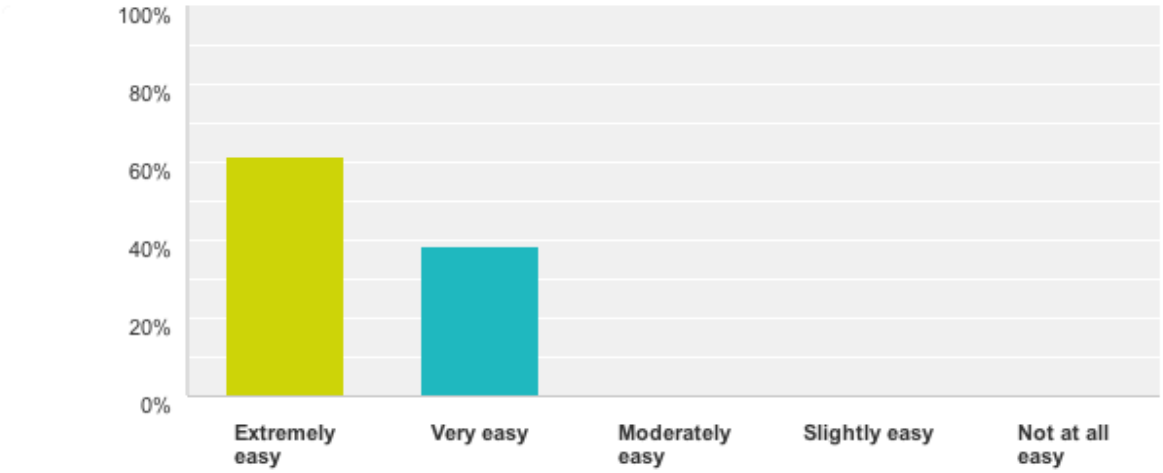
Answer Choices	Responses	
Extremely engaging	53.85%	7
Very engaging	38.46%	5
Moderately engaging	7.69%	1
Slightly engaging	0.00%	0
Not at all engaging	0.00%	0
Total		13

Q3

Customize Export

Does the website appear easy to navigate?

Answered: 13 Skipped: 0

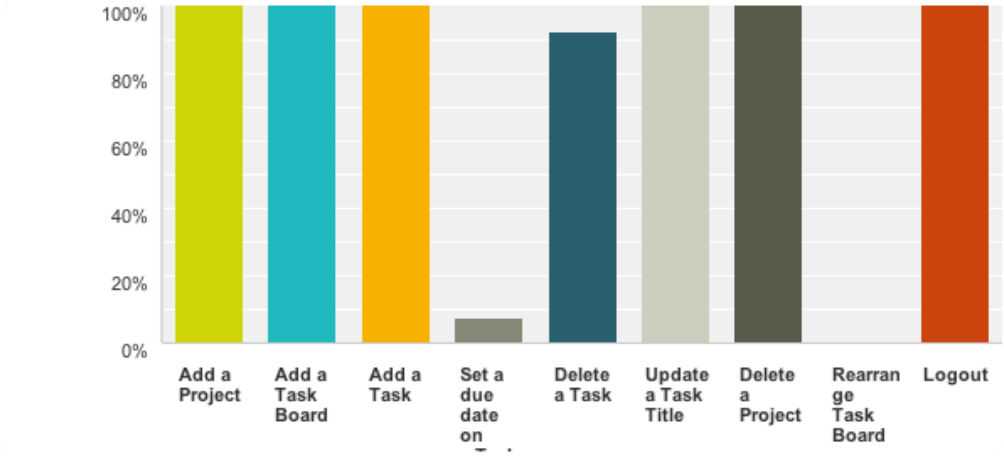


Answer Choices	Responses	
Extremely easy	61.54%	8
Very easy	38.46%	5
Moderately easy	0.00%	0
Slightly easy	0.00%	0
Not at all easy	0.00%	0
Total		13

Q4 Customize Export

Could you do the following inside the application?

Answered: 13 Skipped: 0

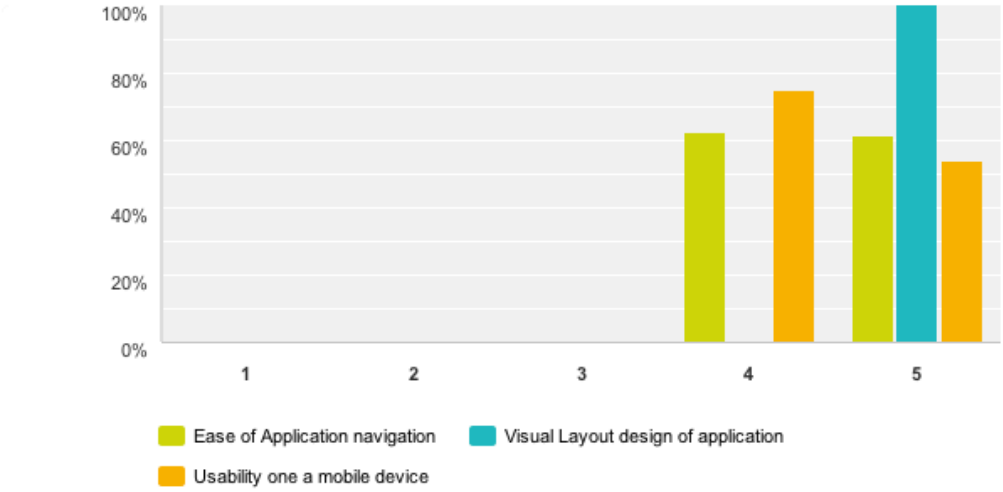


Answer Choices	Responses	
Add a Project	100.00%	13
Add a Task Board	100.00%	13
Add a Task	100.00%	13
Set a due date on a Task	7.69%	1
Delete a Task	92.31%	12
Update a Task Title	100.00%	13
Delete a Project	100.00%	13
Rearrange Task Board	0.00%	0
Logout	100.00%	13
Total Respondents: 13		

Q5 Customize Export

Please rate the following 5 being exceptional and 1 the lowest

Answered: 13 Skipped: 0

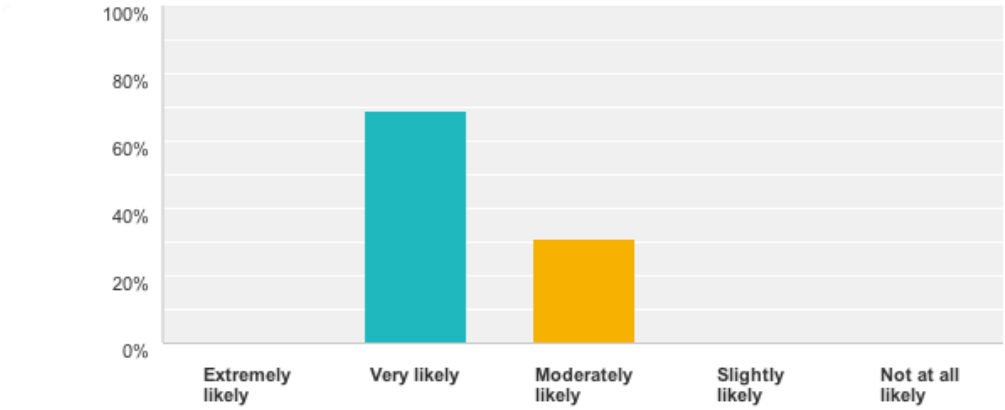


	Ease of Application navigation	Visual Layout design of application	Usability one a mobile device	Total Respondents
1	0.00% 0	0.00% 0	0.00% 0	0
2	0.00% 0	0.00% 0	0.00% 0	0
3	0.00% 0	0.00% 0	0.00% 0	0
4	62.50% 5	0.00% 0	75.00% 6	8
5	61.54% 8	100.00% 13	53.85% 7	13

Q6 Customize Export

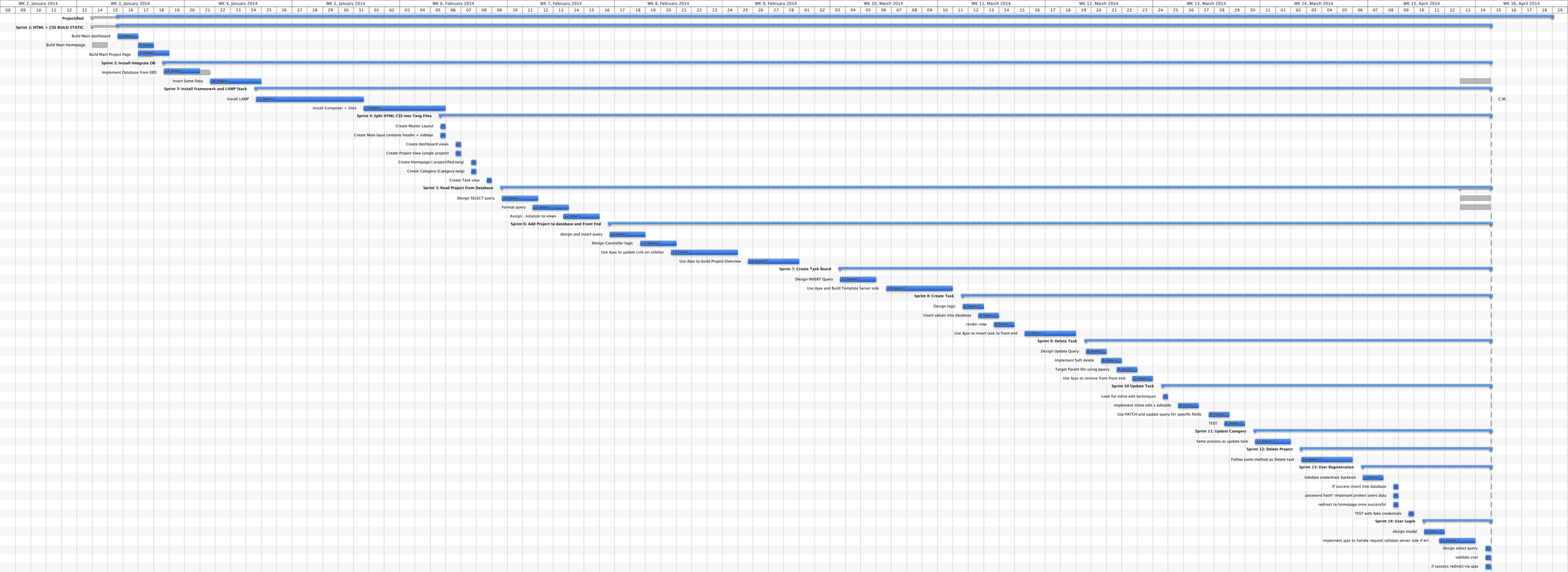
How likely are you to recommend this application to someone you know?

Answered: 13 Skipped: 0



Answer Choices	Responses	
Extremely likely	0.00%	0
Very likely	69.23%	9
Moderately likely	30.77%	4
Slightly likely	0.00%	0
Not at all likely	0.00%	0
Total		13

Appendix E –Project Gantt Chart


WK 2, January 2014
WK 3, January 2014
WK 4, January 2014
WK 5, January 2014
WK 6, February 2014
WK 7, February 2014
WK 8, February 2014
WK 9, February 2014
WK 10, March 2014
WK 11, March 2014
WK 12, March 2014
WK 13, March 2014
WK 14, March 2014
WK 15, April 2014
WK 16, April 2014
Projectified
Sprint 1: HTML + CSS BUILD STATIC
Build Main dashboard
8 hours,7,
Build Main Homepage
8 hours,
Build Main Project Page
12 hours,
Sprint 2: Install Integrate DB
Implement Database from ERD
11 hours,
Insert Same Data
16 hours,
Sprint 3: Install framwowrk and LAMP Stack
Install LAMP
22 hours,7
Install Composer + Silex
23 hours,
Sprint 4: Split HTML CSS into Twig Files
Create Master Layout
4h
Create Main layut contains header + sidebar
4h
Create dashboard views
4h
Create Project View (single project)
4h
Create Homepage (projectified.twig)
4h
Create Category (Category.twig)
4h
Create Task view
4h
Sprint 5: Read Project from Database
Design SELECT query
11 hours,
Format query
12 hours,
Assign . notation to views
12 hours,
Sprint 6: Add Project to database and Front End
design and insert query
11 hours,7
Design Controller logic
12 hours,7
Use Ajax to update Link on sidebar
19 hours,7
Use Ajax to build Project Overview
16 hours,7
Sprint 7: Create Task Board
Design INSERT Query
12 hours,7
Use Ajax and Build Template Server side
19 hours,7
Sprint 8: Create Task
Design logic
8 hours,7
Insert values into database
8 hours,7
render view
4 hours,7
Use Ajax to insert task to front end
15 hours,7
Sprint 9: Delete Task
Design Update Query
8 hours,7
Implement Soft delete
8 hours,7
Target Parent Div using Jquery
8 hours,7
Use Ajax to remove from front end
7 hours,7
Sprint 10 Update Task
Look for inline edit techniques
4h
Implement inline edit.x editable
8 hours,7
Use PATCH and update query for specific fields
8 hours,7
TEST
8 hours,7
Sprint 11: Update Category
Same process as update task
11 hours,7
Sprint 12: Delete Project
Follow same method as Delete task
16 hours,7
Sprint 13: User Registeration
Validate credentials backend
7 hours,7
If success insert into database
4h
password hash* important protect users data
4h
redirect to homepage once successful
4h
TEST with fake credentials
4h
Sprint 14: User Login
design modal
8 hours,7
implement ajax to handle request validate server side if err...
11 hours,7
design select query
4h
validate user
4h
if success redirect via ajax
4h
C.M.

Projectified
Start: 15/01/2... Work: 359 hours ?
End: 18/04/2... Com: 1
Res: C.M.

Sprint 1: HTML + CSS BUILD STATIC
Start: 15/01/2... Work: 347 hours ?
End: 14/04/2... Com: 1

Build Main Homepage
Start: 16/01/2... Work: 8 hours
End: 14/01/2... Com: 1

Build Main dashboard
Start: 15/01/2... Work: 8 hours ?
End: 16/01/2... Com: 1

Build Main Project Page
Start: 16/01/2... Work: 12 hours
End: 17/01/2... Com: 1

Sprint 2: Install Integrate DB
Start: 18/01/2... Work: 335 hours
End: 14/04/2... Com: 1

Implement Database from ERD
Start: 18/01/2... Work: 11 hours
End: 21/01/2... Com: 1

Insert Same Data
Start: 21/01/2... Work: 16 hours
End: 14/04/2... Com: 1

Sprint 3: Install framework and LA...
Start: 24/01/2... Work: 312 hours ?
End: 14/04/2... Com: 1
Res: C.M.

Install LAMP
Start: 24/01/2... Work: 27 hours ?
End: 14/04/2... Com: 1
Res: C.M.

Install Composer + Silex
Start: 31/01/2... Work: 23 hours
End: 14/04/2... Com: 1

Sprint 4: Split HTML CSS into Twig Files
Start: 05/02/2... Work: 266 hours
End: 14/04/2... Com: 1

Create Master Layout
Start: 05/02/2... Work: 4 hours
End: 14/04/2... Com: 1

Create Main layut contains header...
Start: 05/02/2... Work: 4 hours
End: 14/04/2... Com: 1

Create dashboard views
Start: 06/02/2... Work: 4 hours
End: 14/04/2... Com: 1

Create Project View (single project)
Start: 06/02/2... Work: 4 hours
End: 14/04/2... Com: 1

Create Homepage (projectified.twig)
Start: 07/02/2... Work: 4 hours
End: 14/04/2... Com: 1

Create Category (Category.twig)
Start: 07/02/2... Work: 4 hours
End: 14/04/2... Com: 1

Create Task view
Start: 08/02/2... Work: 4 hours
End: 14/04/2... Com: 1

Sprint 5: Read Project from Database
Start: 09/02/2... Work: 250 hours
End: 14/04/2... Com: 1

Design SELECT query
Start: 09/02/2... Work: 11 hours
End: 14/04/2... Com: 1

Format query
Start: 11/02/2... Work: 12 hours
End: 14/04/2... Com: 1

Assign . notation to views
Start: 13/02/2... Work: 12 hours
End: 14/04/2... Com: 1

Sprint 6: Add Project to database an...
Start: 16/02/2... Work: 223 hours ?
End: 14/04/2... Com: 1

design and insert query
Start: 16/02/2... Work: 11 hours ?
End: 14/04/2... Com: 1

Design Controller logic
Start: 18/02/2... Work: 12 hours ?
End: 14/04/2... Com: 1

Use Ajax to update Link on sidebar
Start: 20/02/2... Work: 19 hours ?
End: 14/04/2... Com: 1

Use Ajax to build Project Overview
Start: 25/02/2... Work: 16 hours ?
End: 14/04/2... Com: 1

Sprint 7: Create Task Board
Start: 03/03/2... Work: 166 hours ?
End: 14/04/2... Com: 1

Design INSERT Query
Start: 03/03/2... Work: 12 hours ?
End: 14/04/2... Com: 1

Use Ajax and Build Template Server...
Start: 06/03/2... Work: 19 hours ?
End: 14/04/2... Com: 1

Sprint 8: Create Task
Start: 11/03/2... Work: 135 hours ?
End: 14/04/2... Com: 1

Design logic
Start: 11/03/2... Work: 8 hours ?
End: 14/04/2... Com: 1

Insert values into database
Start: 12/03/2... Work: 8 hours ?
End: 14/04/2... Com: 1

render view
Start: 13/03/2... Work: 8 hours ?
End: 14/04/2... Com: 1

Use Ajax to insert task to front end
Start: 15/03/2... Work: 15 hours ?
End: 14/04/2... Com: 1

Sprint 9: Delete Task
Start: 19/03/2... Work: 104 hours ?
End: 14/04/2... Com: 1

Design Update Query
Start: 19/03/2... Work: 8 hours ?
End: 14/04/2... Com: 1

Implement Soft delete
Start: 20/03/2... Work: 8 hours ?
End: 14/04/2... Com: 1

Target Parent Div using Jquery
Start: 21/03/2... Work: 8 hours ?
End: 14/04/2... Com: 1

Use Ajax to remove from front end
Start: 22/03/2... Work: 7 hours ?
End: 14/04/2... Com: 1

Sprint 10 Update Task
Start: 24/03/2... Work: 85 hours ?
End: 14/04/2... Com: 1

Look for inline edit techniques
Start: 24/03/2... Work: 4 hours ?
End: 14/04/2... Com: 1

Implement inline edit x editable
Start: 25/03/2... Work: 8 hours ?
End: 14/04/2... Com: 1

Use PATCH and update query for sp...
Start: 27/03/2... Work: 8 hours ?
End: 14/04/2... Com: 1

TEST
Start: 28/03/2... Work: 8 hours ?
End: 14/04/2... Com: 1

Sprint 11: Update Category
Start: 30/03/2... Work: 61 hours ?
End: 14/04/2... Com: 1

Same process as update task
Start: 30/03/2... Work: 11 hours ?
End: 14/04/2... Com: 1

Sprint 12: Delete Project
Start: 02/04/2... Work: 50 hours ?
End: 14/04/2... Com: 1

Follow same method as Delete task
Start: 02/04/2... Work: 16 hours ?
End: 14/04/2... Com: 1

Sprint 13: User Registeration
Start: 06/04/2... Work: 34 hours ?
End: 14/04/2... Com: 1

Validate credentials backend
Start: 06/04/2... Work: 7 hours ?
End: 14/04/2... Com: 1

If success insert into database
Start: 08/04/2... Work: 4 hours ?
End: 14/04/2... Com: 1

password hash' important protect...
Start: 08/04/2... Work: 4 hours ?
End: 14/04/2... Com: 1

redirect to homepage once successful
Start: 08/04/2... Work: 4 hours ?
End: 14/04/2... Com: 1

TEST with fake credentials
Start: 09/04/2... Work: 4 hours ?
End: 14/04/2... Com: 1

Sprint 14: User Login
Start: 10/04/2... Work: 19 hours ?
End: 14/04/2... Com: 1

design modal
Start: 10/04/2... Work: 8 hours ?
End: 14/04/2... Com: 1

implement ajax to handle request v...
Start: 11/04/2... Work: 11 hours ?
End: 14/04/2... Com: 1

design select query
Start: 14/04/2... Work: 4 hours ?
End: 14/04/2... Com: 1

validate user
Start: 14/04/2... Work: 4 hours ?
End: 14/04/2... Com: 1

if success redirect via ajax
Start: 14/04/2... Work: 4 hours ?
End: 14/04/2... Com: 1