Bridging User and Developer Communities via Online Platform

Ruairí Kell
B.A.(Mod.) Business and Computing
Final Year Project  April 2014
Supervisor: Dave Lewis

School of Computer Science and Statistics
O’Reilly Institute, Trinity College, Dublin 2, Ireland
Acknowledgements

Firstly I wish to acknowledge and thank my supervisor for this project, Dave Lewis. Without his support, encouragement and unending helpfulness I would not have been able to complete this project.

I would like to thank all my friends and family who have supported me, prayed for me, and shown me tremendous support for this entire year, and throughout my time in college.

A special thanks must be given to Suzy Percy, who has loved and encouraged me endlessly through every high and low in this project. I would not have managed to do this project without her support.

Finally, and most importantly, I wish to thank God, ever present and always looking out for me over the course of the project, and the reason why I do everything to the standard that I have achieved.

DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university

______________________________  ________________________
Name                        Date
# Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Chapter 1 - Introduction</strong></td>
<td></td>
</tr>
<tr>
<td>1.1 Introduction</td>
<td>4</td>
</tr>
<tr>
<td>1.2 Reader's Guide</td>
<td>5</td>
</tr>
<tr>
<td><strong>Chapter 2 - Background &amp; Requirements</strong></td>
<td></td>
</tr>
<tr>
<td>2.1 Project Background</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Background Research</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Requirements</td>
<td>10</td>
</tr>
<tr>
<td><strong>Chapter 3 - Design</strong></td>
<td></td>
</tr>
<tr>
<td>3.1 Shaping the Website</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Shaping the Back-end</td>
<td>17</td>
</tr>
<tr>
<td><strong>Chapter 4 - Implementation: Part 1</strong></td>
<td></td>
</tr>
<tr>
<td>4.1 Wordpress</td>
<td>19</td>
</tr>
<tr>
<td>4.2 Languages</td>
<td>21</td>
</tr>
<tr>
<td>4.3 Plug-ins</td>
<td>22</td>
</tr>
<tr>
<td>4.4 Wordpress Features</td>
<td>23</td>
</tr>
<tr>
<td>4.5 Fulfilling Requirements</td>
<td>24</td>
</tr>
<tr>
<td><strong>Chapter 5 - Implementation: Part 2</strong></td>
<td></td>
</tr>
<tr>
<td>5.1 Front End Full Implementation</td>
<td>25</td>
</tr>
<tr>
<td>5.2 Back End Full Implementation</td>
<td>33</td>
</tr>
<tr>
<td><strong>Chapter 6 - Evaluation</strong></td>
<td></td>
</tr>
<tr>
<td>6.1 Agile Testing</td>
<td>36</td>
</tr>
<tr>
<td>6.2 Full Site Testing</td>
<td>37</td>
</tr>
<tr>
<td>6.3 Problems Encountered</td>
<td>38</td>
</tr>
<tr>
<td><strong>Chapter 7 - Conclusion: Part 1 - Critique</strong></td>
<td></td>
</tr>
<tr>
<td>7.1 Fulfilling Project Aims</td>
<td>40</td>
</tr>
<tr>
<td>7.2 Critique</td>
<td>41</td>
</tr>
</tbody>
</table>
• 7.3 Personal Development

Chapter 8 - Conclusion: Part 2 - Future Work

• 8.1 What Could Be Improved? 43
• 8.2 What Could Be Done Differently? 43
• 8.3 What Scope Is There For Future Developments? 44

Bibliography

• References 45
• List of Websites 45

Appendices

• 1: Original Idea 47
• 2: Original Draft Requirements Document 47
• 3: Complete Terms and Conditions from Site 50
• 4: Screenshots 53
• 5: List of Users and Passwords for Access 54
Chapter 1 - Introduction

1.1 Introduction

1.1.1 Distinctions

In order to begin the discussion of this project, *Bridging User and Developer Communities via Online Platform*, first a brief description must be offered of what this project is about. Therefore, a distinction should be made as to what is meant by *User* and *Developer*. For this project, a User is somebody who uses code and software etc. online or locally, and can have some or no experience with developing code (i.e. they are not a regular programmer, developer, or software engineer). A Developer is someone who has spent a lot of time using, creating, editing and updating code and software, and as such is very experienced in that area.

1.1.2 Brief Description

The idea of bridging the User and Developer communities is to allow them to have better communication than is currently in place. The concept is simple - to provide that platform for them. From this, a Wordpress online platform has been taken in order to build a website, allowing Developers to upload their code online, create discussion around this code and (unlike other sites online) allow easy access for Users, without needing to entirely integrate and immerse Users in a Developer-centric site. The gap in existing systems here is simply this: there are no sites that cater for User discussion, queries, and information that comes directly from Developers who not only use the site, but developed the content that exists upon it. For the Users, they are able to comment and discuss with little to no restrictions or obligations to the Developer community, allowing them to interact only as much as they want or need to. The original plan for the project was to focus specifically on a plug-in/s that would facilitate this *(see Appendix 1)* however, the focus then shifted to a self-contained model that would allow for creation, uploading, and discussion all in one place, that would still cater to both Users and Developers (however there is still quite a large use of plug-ins on this website, in order to aid the project's aims as a whole, as well as for numerous other reasons).
As such, the website (called ConnectIt) can be found at http://ruairikell.com/ and can be used for all that is mentioned above.

1.2 Reader's Guide

It is recommended when reading this report that the ConnectIt website is kept open for reference. In order to make use of any of the website’s features that require logging in, it is requested that one of the Usernames and Passwords found in Appendix 5 is used.

This report shall read as follows:

- **Chapter 2** will discuss the background and requirements for the project, including motivations, research and aims.
- **Chapter 3** will discuss the way in which the project has been designed, tackling both the front-end and the back-end.
- **Chapter 4** will discuss the primary implementation involving setting up the project, the specific aspects to be included and languages used.
- **Chapter 5** will discuss the full implementation including the coding completed, and steps taken to achieve the aims of the project.
- **Chapter 6** will discuss how the project was evaluated in terms of testing and the problems encountered.
- **Chapter 7** will discuss a critical conclusion to the project, what was achieved, and personal development.
- **Chapter 8** will discuss how the work could be improved, and what scope there is for future development.
Chapter 2 - Background & Requirements

2.1 Project Background

2.1.1 Primary Motivations

There were a lot of motivations behind this project and the ideas that it embraces. First and foremost was the fact that there lacked an entire, cohesive, all-encompassing scene online that allowed for communication between developers, while also allowing for easy and simple access for those outside of the Developer community (i.e. Users). To put this in clearer terms, Users would have had to create an account, for a site aimed specifically at Developers, and engage with a Developer only community, in order to get responses to their simple questions and queries (please see Background Research section 2.2 below for further explanation on these sites). Also, the Users may be dissuaded by a Developer centric forum, as it could appear quite intimidating should they not be technically minded. Not only this, but by going to a site that is not the place that they encountered whatever problem they have, they also move away from the community on that site who may have been using the same software etc. All of this has created a virtual rift between two communities, allowing Developers to thrive amongst their own - other developers - while restricting Users from gaining access to this hub, and as such limiting their discussions (however unintentional this may have been). From this, it was felt that there was a problem that needed to be addressed, and that the solution to this was to build some kind of bridge between the two distinct communities. As will be seen, the solution ended up being achieved through a simpler concept than it first appears was needed.

2.1.2 Secondary Motivations

A secondary motivation for the project had to do with the idea of doing a final year project itself, and personal motivations. As a student of Business and Computing, I have a unique perspective on approaching projects (with skills from both areas crossing over). However, this also offers some limits with the amount of programming and coding that have been experienced. Taking both of these aspects into account, as well as the primary motivations mentioned previously, it was felt that a web-based project would be perfect (even though it involved learning new languages and technology), and also that it would serve a business-
type purpose - in that it would be existing to fill a gap in the market and cater to those consumers’ needs.

2.2 Background Research

In order to prepare for this project, and as a result of the motivations, research had to be conducted into what options were already out there. Not only was this to find where the project would have its particular niche market, but also in order to see what similarities could be drawn from, changed and adapted in order to aid it.

2.2.1 Developer Research

Research began by looking at some sites that offered discussion for Developers. The three main sites that were looked at for this section were: GitHub, Stack Overflow, and Sourceforge. There were also some plug-ins that focused more specifically on discussions and were included in this research too: Get Satisfaction, UserEcho, and Uservoice

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GitHub</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Stack Overflow</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Sourceforge</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Get Satisfaction</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>UserEcho</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Uservoice</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
The table uses the term "acceptable" for each column. What is meant by this is the question - is this feature at a level that allows for ease of use of itself, and is the standard of that feature of an appropriate level to fulfil the relatable requirement that might be had for this project?

To explain the above table further, GitHub is perfect for Developers, as it allows for easy upload, storage and access of code repositories, as well as providing a commenting feature available for other Developers on the site. It does not however make it easy for a User to ask questions on code they may have found on this site (perhaps through a third party search engine), as they would need to break into that community by creating an account for a site they might never use again, as such pushing them away.

The same can problem can be said for Stack Overflow. On this site, there is slightly better access (e.g. login with Facebook accessibility), however, the front page clearly states that it is only for programmers, "Stack Overflow is a question and answer site for professional and enthusiast programmers.", pushing away those who are just using the code. Added to this, the site does not have a specific upload feature, but rather relies on the copying/pasting of certain lines of code. This can push aside those who wish to showcase the entirety of their project or code too.

Sourceforge is a slightly different type of site. Rather than code being available, it is software - entire projects. It is open source, and easy for Users to access, but in order to create discussion around the software, one must create an account, again for a site they may never use again. This appeared to be a running theme and one which was a problem that needed to be solved.

The three discussion sites/plug-ins turned up trumps in all three discussion related columns in the table. However, because they are not integrated with any particular site by default, they lack the other features. Added to this, is that they all require payment to unlock the majority of their potential (even up to $1200 per month for a professional package), and they are for organising a community that sits somewhere on top of where the code/software would be, but not integrated directly into it.
2.2.2 User Research

As previously mentioned in 2.2.1, discussion for Users was not happening with full integration on any site researched. However, it was still necessary to research some secondary aspects (i.e. not full requirements), and one which was prevalent in my mind was new Users wanting to learn how to code through being gradually exposed to more and more code. While ConnectIt would be a forum/space that allowed for discussion rather than teaching, there would be an aim that those who were exposed to this, would eventually use it as a resource. Therefore, it was important to look at other resources for learning coding.

The first site to be examined was Codecademy. This site offers free, interactive learning for those wanting to create code and learn how to program. For this site there is a strong emphasis on community ("Join the Community" being one of their three core principles), however it appears to be aimed more at connecting with friends and groups, rather than a generic discussion forum or thread. The unfortunate problem with this is that the more experienced Developers would not be in those groups, and as such the Users would not be getting the same level of help.

The second site to be examined was CoderDojo. This site does not offer online learning, but instead directs a User to their closest "dojo", where they can get help learning how to code. This is included here because despite being a physical presence - as opposed to a virtual presence - it does offer discussion with more experienced Developers, as well as with others learning to code. The only downside to this is that the pool of resources (i.e. Developers) does not stretch as far as an online/virtual pool potentially could.

The final site to look at was W3schools. They offer a lot of tutorials, for all kinds of web development languages. The site also offers forums for discussion, which are very well used, however they are on a completely separate part of the website to the tutorials, and this may be off-putting to some Users who are only learning to code. As with the other two sites above, registration is also needed to facilitate discussion.

While this is only a small sample of code-learning sites, it shows the common themes that runs in all similar sites. The lack of integration and ease of use, as well as other downsides mentioned above are issues that can hopefully be addressed by ConnectIt in the long run.
2.3 Requirements

Following from the research that was done, requirements needed to be drafted. This meant deciding what features the project should contain, what issues it should address, and how the project should function overall. Initially, a requirements document was formed (along with UML diagrams), that reflected what was felt would be the final direction of the project. However, due to the changing nature of this type of project, a complete revised draft of requirements document had to be created [Please see Appendix 2 for the original draft Requirements Document]. The first step was to build UML Use Case diagrams in order to begin to outline exactly who would require what.

2.3.1 UML Use Case

![ConnectIt System UML Diagram]

2.3.2 Introduction

To begin this project, a decision had to be made on what methodology would be best for use, and for this project the choice came down to either using a "waterfall" style approach, or using an "Agile" style approach. (Base36, 2012; Mikoluk, 2013) While it is true that a choice must be made in most cases, it is entirely possible to use aspects from both of these styles. A lot of Agile aspects will be in place as numerous iterations of the process will be used in order to further the development of the project. However, there will still be more of a focus on one particular aspect each time around, following with a waterfall approach. For
example, the project will have five phases, with phase one being Requirements, phase two being Design, phase three being Implementation/Development (coding and testing), phase four being Evaluation, and phase five being Launch (final testing and "okayings" the project). This would follow with a *waterfall* approach. However, during each of these stages each of the other aspects would also be incorporated, e.g. doing building and testing right through phase one, two, three, and four. This would follow with an *Agile* approach of testing as the project develops through iterations.

### 2.3.3 Functionality

This section is to address what exactly the end product should be able to do. The end product must by definition be accessible to both Developers and Users. As can be seen from the UML Use Case diagram (2.3.1), this means that it must contain the following:

1. Discussion functionality for both User and Developer
2. Ability for Developer to upload to their content
3. Ability for Developer to create a post around their content
4. Ability for Developer to edit said post
5. Login capabilities for Developers
6. Linking capabilities for Developers
7. Moderator capabilities to edit any/all posts or discussions
8. Moderator access to manage database
9. Moderator access to manage emails

An extra addition to these requirements taken from the UML would be what was discussed in 2.2.2 - regarding the entire site, over the course of time, that Users would gradually be exposed to more and more code, helping them to learn code too. This is not a primary requirement per se, but is a natural progression as a result of the nine points above.
2.3.4 Software/Hardware

In order that this should be able to develop this properly, it had to be decided what would be used to develop the project. In terms of hardware, there is nothing needed for this other than a laptop. For software, there is a little to be discussed:

- Using Notepad++/online editors to write any code.
  - This allows for simplicity in writing all code without any unnecessary features.
  - An IDE (Integrated Development Environment) such as Netbeans was considered, however the goal was to keep everything as simple as possible, and due to the tools provided within Wordpress and the web hosting control panel, the simplest option was seen to be the best (Occam's Razor).

- Plan to develop as a Wordpress site, minimizing the amount of coding necessary
  - Wordpress has a clear plug-in architecture that is of utmost importance to this project.
  - Wordpress is open source, and numerous web based languages can be used inside and in conjunction with it.
  - Wordpress is far more widely used than Drupal or Joomla (Website Set-Up, 2014).

- Most development should take place using PHP, HTML, and some JavaScript

- A liberal use of Wordpress plug-ins is necessary for:
  - Allowing PHP/JavaScript functionality.
  - Allowing Developers to post via plug-in.
  - Creating forums.

- Use of an online web-host and control panel/database at AgilityHoster.

Having never used Wordpress, online web-hosting/control panels/databases, PHP, or JavaScript before, it was of utmost necessity to put in time to learn all of these in advance of beginning to build the project. This ended up being a big time consumer for the project. To expand:

- Time was spent time downloading, installing and getting used to Wordpress, as described in 4.1.1
• Self-teaching of the languages was done by using W3schools tutorials. This covered completing the HTML tutorials, and doing basic PHP and JavaScript tutorials (and returning to them over the course of the project as needed).

• However, all of this took quite a while due to the steep learning curve, previously unknown syntax and unforeseen time constraints.

2.3.5 Performance

As a Wordpress site, it is relatively lightweight. There would have been performance issues with free-web hosting speeds and storage (as well as limits on access to certain things), so paid hosting was opted for. As such, the performance required from this project should pose no problems at all.

2.3.6 Considerations

For this project, there are a number of things that need to be taken into consideration, such as:

• Steep learning curve for software/applications being used
  o This is explained above in 2.3.4 in more detail

• Security for Users/Developers
  o While there are numerous third party applications being used in the development of this project, and they all have their own levels of security, it is important to make sure that all of them work to an acceptable standard, and cover all the security that would be expected of any high-level site.

• What testing should be done?
  o For a site like this, a vast amount of testing is done as each separate part is being developed. This ties in with the Agile approach to the project (2.3.2).
  o Any extra testing on top of this may require a User sample and thus more time management, coordination and ethics forms.
  o If not the above, then extra testing involves a full comprehensive run through of all possibilities for ConnectIt.
• How much maintenance will be necessary?
  o There is a possibility that most of the site will be self-maintaining or maintained by those registered to the site.
  o A skilled administrator/moderator is mandatory.
• Legalities that need researching/Terms and Conditions of Use.
  o See Appendix 3.

2.3.7 Limitations/Possible Issues

Some of the issues that may be faced, and limitations that exist are:

• Limited amount of time
• Scope creep
• Testing and maintenance
• Usage of any new software/languages
Chapter 3 - Design

This chapter will deal with shaping of the web platform, and show exactly how all features can be implemented on ConnectIt, as well as the specifications surrounding them.

3.1 Shaping the Website

This section focuses on the design of visible and usable elements that will be included in the front-end of the website.

3.1.1 Pages

There will need to be a certain amount of pages, each displaying different types of content relevant to Users and/or Developers.

- A "Home page"/"Splash page"/"Welcome page".
- An "About" page describing what the aims of the site are, who created it, and what it can be used for.
- A page that explains to those accessing the site the difference between Users and Developers.
- A page for submitting posts, as well as uploading content and details about registration.
- A posts feed. This would be a page that would contain all of the posts (probably in chronological order).
- A contact page for contacting the moderator/administrator.
- A page that outlines the Terms and Conditions of using the site. This page must be accessible and visible to anyone who accesses or uses any part of the website.

3.1.2 Posts

There must be a few options for the posts too:

- They will have their own page/feed (as in 3.1.1).
- They will be able to be accessed via this feed, and also the search function.
- Default sorting will be chronologically, however they can be filtered by the search function, by author, or by category.
• Posts can be created by the Developers using the content that they have submitted. One way of doing this will be by editing the Press This plug-in (see 4.3.3), so that they can access it via the sidebar.
• The "edit" function will be enabled for authors on their own posts (i.e. they will be able to see it beside each of their posts).

3.1.3 Discussion

The discussion will come about in three different forms: Comments, Facebook Comments, Forum Discussions.

• Comments will be set so as to allow anybody to comment as long as they have a valid email address.
• To avoid spam (although there is a spam filter plug-in activated), anyone commenting must first have a previous comment approved.
• Users will also be able to log in using their Wordpress account and comment with that.
• Facebook commenting will be enabled for posts too (if a User does not wish to give their email, it would defeat the aims of the project to dissuade them).
• Forums operate with the same security parameters as the comments.
• Forum topics can be created by anyone with a query (again so as not to dissuade Users from asking their questions).

3.1.4 Sidebar

The sidebar will exist to provide easy access to some of the most used functions on the website.

• Login and Registration links, for those who need to quickly login and those who haven't yet registered.
• The aforementioned (3.1.2) modified Press This plug-in. This will allow for quick creation of a post by those registered for the site.
• Links for the most recent posts. This is important to keep everyone as up to date as possible.
• Links to forums (as with the posts).
3.1.5 Footer

The footer will exist to provide access to some of the lesser used functions on the website.

- Terms and Conditions page. This will be visible on the footer of every page, thus showing anyone who access what their rights and responsibilities are.
- Categories of different types of posts.
- A calendar of all posts created (for a particular month).
- The "Meta" section. This will include quick links for the administrator to access what he/she needs.

3.1.6 Header

The header will be the navigation bar for ConnectIt. It will contain links for all of the main pages (3.1.1) as well as a search function implementation.

3.2 Shaping the Back-end

3.2.1 File Management

File management is important to this project, because it allows an administrator to access, update, edit, change, fix, add and delete all files associated with the system, whether they are visible on the front end or not (2.3.3 point 7).

As such, through the lightweight file manager within the control panel, access is available to all the files contained within the standard Wordpress install (and subsequent changes via Wordpress Dashboard front-end). Through said file manager, it is possible to also add extra files that will be needed for anything extra to be added to a Wordpress install. In this case, that will be PHP scripts that must be included in order that they can be called from the front-end when needed.

Extra folders can also be created in order to store files that are not directly associated with Wordpress. To expand, this means that content that is uploaded can be directed to be stored in a separate folder, and pulled from there once the content is needed again. This ties in with some of what this project will seek to achieve.
3.2.2 Database Management

Database management is another important aspect of the project. This is because it may be necessary to edit, change, fix, delete, update and access details from the back end (i.e. not via Wordpress). These details can range from registered User details, to different posts and pages.

The control panel offers links to the phpMyAdmin tool for handling the administration of the MySQL database (2.3.3 point 8).

From here, Users can be deleted, settings for posting can be changed, comments can be edited and other important functions. This would be very useful for instance in the event that a User lost their login details, or a post had to be removed remotely for example. All in all it is a very useful tool.

3.2.3 Email

Another useful tool provided for dealing with options in the back-end is the email functionality (see 2.3.3 point 9). The web hosting plan includes up to 1000 email addresses for the domain name (e.g. admin@ruairikel.com). A third party email service, Roundcube, is a tool provided (through the control panel once again), in order that emails can be sent and received. This can be done by accessing Roundcube directly, or through the email functions provided through the Wordpress dashboard.
Chapter 4 - Implementation: Part 1

This section of the report will deal with how the project was set up to be built - following from Chapter 3. Focus will be on the software set-up, language usages, development of plug-ins, specific features for the website, and tackling all the initial aims set out in the above requirements documentation (2.3).

4.1 Wordpress

4.1.1 Download and Local Host Test

To begin, Wordpress was downloaded from the website and installed on a laptop, using a local host through XAMPP. This means that the site was viewable as if it were live from the local machine, but it was not actually online. This also meant that it could be controlled through locally stored files and the XAMPP control panel.

The reason this was done was in order to become more familiar with using Wordpress, and as such be prepared for setting up online.
4.1.2 Move to Online Hosting

Once this was done, the natural progression was to move to online website hosting. After minimal research, AgilityHoster was chosen. This was the obvious choice for a number of reasons: it had an automatic Wordpress install, fees were cheap, and the control panel interface was easy to use.

![Hosting Tools](image)

After much deliberation, and assessment of what was required, it was decided to opt for a year's hosting plan at $9.99. This included the necessary: unlimited disk space, unlimited traffic, 2 domains, 5 subdomains, 7 MySQL databases (3.2.2) and 1000 email address (3.2.3) (2.3.3 point 8, 9). Added to this, it also allowed for editing of the `php.ini` file, which would definitely be needed later on when PHP would be used for the project.

To finish off setting up the hosting, the domain [http://ruairikell.com](http://ruairikell.com) was registered, and Wordpress was installed for this domain.

4.1.3 Advantages of Using Wordpress

As have been previously mentioned, there are numerous reasons for using Wordpress for this project. A major reason is how it allows for the use of plug-ins (explained further in 4.3). This is a distinct advantage. Added to this, it cuts down on the amount of coding necessary to build a website (the aims of this project are not to build the website, but to create a
space that bridges User and Developer communities, and as such many hours can be saved by using a Wordpress platform). It is also so widely used, that the interface will be relatively familiar to many Users.

**4.2 Languages**

Certain languages would have to be used to code in, in order to create the features required for ConnectIt.

**4.2.1 HTML**

HTML is key for building any website as it is the basis of everything that we see (of course this is tied in with CSS formatting). However, it is dually important in Wordpress, as there is built in functionality for creating posts or pages that include HTML. This can allow for the creation of forms etc. *within the posts on the site itself*. This would be important for the site as it developed.

**4.2.2 PHP**

PHP is a server-side scripting language (PHP, 2014), which is what was needed in order to access the servers from the Wordpress site. In order to allow PHP work within Wordpress there had to be some adjustments made front-end and back-end. The front-end involved a plug-in to allow for PHP inside Wordpress (see 4.3.1 for more details). The back end involved editing the `php.ini` file supplied via the control panel on AgilityHoster. In order to be able to push content to the servers and to pull that content again, `fsockopen()` must be enabled within this file (this is disabled by default, and thus supports the reasoning in 4.1.2 that access was needed to the file).
PHP would also be used as directly uploaded files to the server (that could then be accessed through the front end). The web hosting file manager allowed very well for this, and also allowed for online editing of those files, which was important.

4.2.3 JavaScript

A third language that was needed was JavaScript. This too is not automatically supported within Wordpress and ergo needed a plug-in to work (see 4.3.2 for more details). JavaScript would allow for certain plug-ins to be then integrated into the website. These plug-ins would make the site easy to use and also allow for the integration of JavaScript where it would not otherwise be able to go.

4.3 Plug-ins

As has been mentioned above, there were plans for a lot of plug-ins to be included in this site for various reasons. Here is an analysis of those that were installed.

4.3.1 Allow PHP in Posts

The title of this plug-in is very self-explanatory. It allows for PHP code to be run from inside a Wordpress post or page. Instead of writing it traditionally `<php ... ?>` the plug-in allows you to write it as `[php]...[/php]` and translates this to traditional format itself. This plug-in will allow for those creating posts to pull their content from the website directly to the post for display. This is a significant part of what will be in play for the completed website.

4.3.2 HTML JavaScript Adder

The HTML JavaScript Adder is a plug-in that allows the creation of widgets anywhere on the Wordpress site. This means that it can be used to display other plug-ins for use by any Users on the site. Particularly it will tie in with the Press This plug-in (4.3.3) to allow others to use it.

4.3.3 Press This
Press This is a bookmarkable plug-in used within Wordpress for creating quick posts from any webpage on the internet. It normally directly pulls links from the website currently being viewed, and is normally only available for the administrator. However, it can be tweaked (with the help of JavaScript, 4.3.2) so that registered users of ConnectIt will be able to use it to create posts of their own with their content (2.3.3 point 3).

4.3.4 bbPress

bbPress is a plug-in that allows for the creation of forums within Wordpress. This will not be edited from what it supplies, as the software is all-encompassing and it helps to facilitate one of the main aims of the project - to have discussion (2.3.3 point 1). This is vastly important.

4.3.5 Other

- Acutnetix Secure Wordpress - built in Wordpress security
- Broken Link Checker - checks all links are working on the site
- Contact Form 7 - allows for simple contact form creation
- Akismet - used to help block spam comments

4.4 Wordpress Features

There are also additional features of Wordpress that will be put to good use for this project.

4.4.1 Posting

Posting has already been mentioned above, but it is important to clarify what exactly it is. Each Wordpress site can allow for registered users to create new posts on a site (akin to blog posts). This is what is envisaged for Developers on the site (2.3.3 point 3). Therefore, it is a feature that will be tapped into and used to its fullest potential.
4.4.2 Commenting

Commenting is another important feature **(2.3.3 point 1)**. It is important because it helps tie into the aims of the project for facilitating discussion. The important part about Wordpress commenting, is that it can be edited so no sign in is necessary (just a valid email address), which is something that has been mentioned above as important for the **Users**.

4.4.3 Sidebar/Footer/Header

By editing the widgets that come with a Wordpress standard set-up, it is possible to create very professional sidebars, footers and headers **(3.1.4, 3.1.5, 3.1.6)**. A lot of the information, links and links to specific pages etc. for the site will contained within the sidebar/footer and the header will be a navigation bar for ConnectIt. Thus, it is important to make the best use of all of these features too.

4.5 Fulfilling Requirements

It is important at this point to note why the **Design** and **Implementation** aspects are being discussed in such detail here and Chapter 3. This is because it is imperative to show that it will fulfil the requirements set out above in **2.3** (with specific reference to 2.3.3). As the design is set-up as per this Chapter (4) as well as somewhat in Chapter 3, it should allow for all aspects of those requirements to be realised.

It is also important to take note that all of these software choices were made through careful consideration with the forethought that they would be the most beneficial to completing the aims of this project (their benefits being shown throughout this chapter). However, were somebody else to tackle this same issue, they may implement a similar style approach using different technology (for example using AJAX [Kyrnin, 2014] instead of PHP, using Joomla & Drupal instead of Wordpress, or choosing different plug-ins).
Chapter 5 - Implementation: Part 2

This Chapter will take from all that has been outlined in Chapter 3 and 4 regarding the initial design and beginning of implementation, and show how each part was then further implemented and specific features completed.

5.1 Front End Full Implementation

This section deals with the front-end of the website. The dashboard is used for the entire build here, and plug-ins are woven throughout all features to be discussed. This will be shown clearly within this part (5.1).

5.1.1 Dashboard

The Wordpress dashboard is a place where the creator can login to change front-end settings, create posts, pages etc. and monitor Users contributions (as well as approve, delete and edit comments/posts). The files responsible for the appearance of the site can
also be edited from the dashboard, as well as management of all plug-ins (installing and setting editing). For 5.1.2 - 5.1.7 the dashboard's many functions were used in order to build and change the front end.

5.1.2 Pages

Following from 3.1.1, there were numerous pages to build. Some of these were simple information pages, while others required more information or a more technical aspect that needed to be included.

- The first page visible to visitors was to be some kind of splash page. Here, a simple Welcome Page was opted for, that has two lines explaining what ConnectIt is, and how to get started with the site. This required some simple text, and some links to relevant pages.
- The next page was a Posts page. Within Wordpress settings, this was made to be the default page for posts to be placed in a chronological form (see 5.1.3 for more details).
The third page was one of the most important to create for ConnectIt. The Submit page involved creating a HTML form to create uploading buttons that in turn called from a PHP upload script from the back-end (see 5.1.3 for full details!). It also contained a list of instructions on how to upload, create posts and submit them to the site (as well as simple registration/login instructions). The text here took some refining in order to keep it as short as possible with as much information as necessary.

All of the above pages were featured in the header. The last pages to be featured in the header were informational pages: explaining the difference between Users and Developers, explaining the features contained in the site, and a contact page for anybody to use.

A final page to be included was the Terms and Conditions page. This was to be featured on every page able to be accessed on the site and as such was included in
5.1.3 Uploading and Creating Posts

As mentioned in 5.1.2 (and before), posts need to be created by those registered for the site. Thus, the Submit page was created. As previously mentioned, this involved a HTML form created:

```html
<html>
<form action="upload_file.php" enctype="multipart/form-data" method="post">
<label for="file"></label>
<input id="file" type="file" name="file" />
<input type="submit" name="submit" value="Upload" />
<p style="padding-left: 30px;">
</p>
</form>
</html>
```

The call to the PHP upload script "upload_file.php" will be discussed in detail in 5.2.1. However, this is an example of what it returns:

```
Please copy the following text:

[php] echo file_get_contents('upload/htmlform.txt');[/php]

Click here to return to Submit Page
```
The text that is copied acts as explained in 4.3.1 and will echo the contents of the uploaded file. The next step in submitting is to put this copied text into a post. The way in which this is done is by pasting it into the modified Press This plug-in (see 4.3.3) (located in the sidebar). This was modified by using HTML JavaScript Adder and adding it as a widget to the sidebar:

This pop's up a new Press This box, that is completely blank (unlike a normal Press This), into which the copied text is inserted, as well as any other details, such as title, links to code repository, back story for the code or whatever the poster decides to include:
From this, the Developer will get the contents of their file - i.e. the code - and everything else they have added, in a post. They then have the ability to edit/delete this post when necessary, as well as being notified by email when a comment is made on their post.

5.1.4 Discussion

Above, there are three different forms of discussion that have been decided on to include for this site and its purposes (see 3.1.3). To reiterate, these three are: Comments, Facebook Comments, and Forum Discussions. Following from 3.1.3, these discussions must be facilitated.

The easiest of these to set up was the comments section. In order to follow with the aims of this project, and not dissuade the Users from interacting and contributing to the discussion, it was set - via the dashboard - so that all that was necessary for the User to comment was an email address and to input a name (this does not have to be a full name necessarily). They could also use their Wordpress login if they had one too. Following from this, it was decided that these Users must have one previously approved comment on the site (verifiable by email or by Wordpress details) in order to comment. Once their first comment had been approved by the moderator (i.e. not spam!) they could freely comment on any post that they desired.
The second implementation was that of the Facebook comments. This was set up in case a User was not comfortable giving their email address, and if they were already logged in to Facebook, they could simply type and post their comment with no fuss. To attempt this first, two different plug-ins were tried - LoginRadius and OneAll Social Login. These would integrate Facebook comments (as well as other social media websites) and allow Developers to login via those platforms rather than providing an email address. However, when these were set-up and put in place, the login did not work for either plug-in. After troubleshooting, it seemed that the problem rested in the *php.ini* file residing within my control panel, and some settings had to be changed within that. After tweaking those settings though, nothing changed, and it was decided to give up on LoginRadius and OneAll and go direct to Facebook commenting. Firstly, two pieces of code were placed in the *header* file, and the *comments* file respectively for the Wordpress installation:

```
<script>(function(d, s, id) {
    var js, fjs = d.getElementsByTagName(s)[0];
    if (d.getElementById(id)) return;
    js = d.createElement(s); js.id = id;
    js.src = '//connect.facebook.net/en_GB/all.js#xfbml=1&appId=624693737579931';
    fjs.parentNode.insertBefore(js, fjs);
})(document, 'script', 'facebook-jssdk'));</script>
```

```
<div class="fb-comments"
data-href="http://ruairikell.com" data-numposts="10" data-colorscheme="light">
</div>
```

This now meant that a box for Facebook commenting would show up at the bottom of the page too.
The third part of the discussion is forums. These are powered by a great Wordpress plug-in called bbPress (see 4.3.4). As with the comments, it was not desirable to have anything that could possibly exclude Users from the discussion, so therefore it was set so that there would be no registration necessary to: comment on the forum, create a new topic on the forum, or create a new specific forum. Again, the forums are moderated.

5.1.5 Sidebar

The sidebar (as in 3.1.4) contains a lot of the most used functions on the website. This means that they are visible at all times, and easily accessible. As such there is a Get Started section and this contains links to both login to the site, and register for the site (the login and register are both dealt with by Wordpress, and operate on very simple email/username/password terms).

There is also a Create New Post section, into which was placed the modified Press This plug-in (see 5.1.3).

Then there are two sections covering generally well used links - Recent Posts and Forums.

5.1.6 Footer
The footer contains four things as seen above. A simple calendar widget has been placed here for quick easy viewing of when something has been posted. The same can be said for the Categories.

The Meta exists for ease of use for the administrator.

There is also the permalink to the Terms and Conditions as outlined in 5.1.2.

5.1.7 Header

Finally, we have the header, which is the navigation bar for the site. This has links to all the pages (as covered in 5.1.2).

It also contains a slide-out search plug-in, which allows anybody accessing ConnectIt to search the entire site's contents.

5.2 Back End Full Implementation

While most of the work for this was done in the front-end, as shown in section 5.1, there was still some work to be done in the back-end, via the control panel or otherwise. In this section, that work will be discussed.

5.2.1 Upload Script and File Storage

To relate back to 5.1.3, there was a need for a PHP script that could upload files to the server. In order to achieve this, the first step was to create a new folder in the file manager, called "upload", and put it inside the folder for the website (i.e. the website folder called ruairikell.com). Then a file called upload_file.php was uploaded into the main directory folder for the website (to allow it to be accessed by the HTML form in 5.1.3).
To explain the code: when this file is called (in this case, by the HTML file in 5.1.3), it will first check if the chosen file is a text file (by checking against the allowed file extensions), and if it is under 1mb (by checking against the file size limiter). If this is true, and there are no other standard errors, it will check if the file already exists. If this too is true, it will then upload the file, remove all punctuation and spaces (as it was found that these could often cause parsing errors), and move the file into the "upload" folder that had been created earlier. Then it would echo the text that was above (5.1.3). If the file is over 1mb, not a text file, or already exists, the correct error will be shown instead of the text.

By storing the file in the "upload" folder, it can then be accessed from the Wordpress post creator via PHP code (the copied text in 5.1.3). This is allowed in Wordpress only because of the installation of Allow PHP in Posts plug-in (discussed in 4.3.1). That means that the content can be pulled from the server to that post.

5.2.2 PHP.INI File

In theory this all should have worked fine as it was. However, it was throwing up errors within Wordpress that had to do with the file_get_contents() part of the PHP code. Troubleshooting resulted in finding out that for this to work, fsockopen() has to be enabled in the php.ini file contained within the web hosting control panel. Therefore, it was
necessary to go into that file and edit it so that this was enabled. After this was done, the errors stopped occurring.

5.2.3 Emails

A final, minor part of implementing everything in the back-end was to enable emails. This would allow for user registration, emails sent to the administrator via contact form, and any other emails (automatic or user generated) that needed to happen (see 3.2.3 also). This was easily completed through the control panel, and the email system set up (with Roundcube). This all went without problem, and was a great bonus to the project.

_A larger selection of screenshots from across the website can be found in Appendix 4._
Chapter 6 - Evaluation

This chapter is in relation to how the site was evaluated. This will be looking at how the site runs overall, as well as the testing that was performed on it.

6.1 Agile Testing

As mentioned previously (2.3.2), the project was to take a part-Agile approach, and as such, testing would occur as each iteration of a new piece of the site was completed.

6.1.1 Beginning the Project

The beginning of the project was not too difficult to assess. Simply put, the setting up of Wordpress, web-hosting, database etc. was done on a trial-and-error basis. While there was a learning curve with the new software, after spending the time to work out issues such as where to upload files, or how to edit themes, the project was set-up and eventually (after much tweaking) the platform ran smoothly.

6.1.2 During the Project

During the project, testing had to be continuous. As each new aspect was added it had to be tested to see if each part worked. If it did not, a new iteration occurred, with some changes. To give two examples:

- The *Press This* modification:
  - Firstly it was attempted to copy and place it directly into a sidebar text widget. This did not work when tested, instead returning the code, so then it was placed inside a specific JavaScript widget on the sidebar. This managed to run when tested, but it was not yet modified for the purposes of the site. The code was then edited, saved, and tested again, until it worked.

- The PHP upload script:
  - This was first tested simply to see if it uploaded. It was tested by checking the "upload" folder. This worked, and then followed numerous iterations, each with different added lines of code: limiting by extension, limiting by file size, checking if it already existed, removing punctuation, echoing back text. Most
of these ran smoothly, with the exception of removing punctuation, as that removed the full-stop before the extension, which was not desired, so that required a new iteration. This meant after all of this, the code was fully tested.

6.2 Full Site Testing

Come the end of the project, the end product had to be tested in its entirety. This meant deciding how to evaluate it, as well as evaluating comprehensively.

6.2.1 External Evaluation

The first option was to get it evaluated by those external to the project, i.e. through a sample group of "customers", let them rate the site via a number of metrics, and provide feedback. This was rejected for a number of reasons:

- Due to time constraints, setting up testing, finding subjects, and passing ethics approval forms, it would have taken too long and hindered the development of the project.
- There was a possibility that the lack of familiarity with the site would mean that they would miss something out.
- The test subjects would not necessarily be from the groups that this project is targeting.

6.2.2 Internal Full Site "Run-Through"

Therefore, it was decided to test it internally. While this did mean that there would be no "customer feedback", it did allow for a complete familiarity with the site, an ability to make the needed changes on-the-go, and no time constraints save the time set aside for testing.

The site "Run-Through", consisted of a number of checkpoints:

- Check that every page on the website is coherent and accessible
- Check that the Submit page does exactly as it says
- Check that discussion is available through:
  - Commenting logged in
Commenting not logged in
Facebook commenting
Forum topic creation and replies

- Check that posts can be edited and deleted both by an external registered user, and the administrator
- Check that emailing works
- Test the contact form
- Test the search function

### 6.3 Problems Encountered

For both the Agile testing methodology, and the full site test (as well as after this), there were of course problems encountered.

#### 6.3.1 Coding Errors

One of the common problems with any project is the errors that occur within the code. This happened very often over the course of this project. However due to the Agile approach that was in place, it did not cause any hindrance other than time consumption. These coding errors ranged from incorrect formatting, to parsing errors, to compiling errors, to server restrictions. Fixing these required a vast amount of effort over the course of the project, yet it was necessary and worthwhile in order to achieve a finished working product.

#### 6.3.2 Site Restrictions

There were also some restrictions that happened during this project. These ranged from not being able to access the `php.ini` file (thus restricting PHP pull requests), to firewalls blocking incoming RSS feeds (this was going to be a simple aesthetic addition to the site). While not all of these were resolved, and while they were again time consuming, they were hurdles that could not simple be ignored, and had to be overcome for the sake of the project.

#### 6.3.3 Spam

One major issue that was not foreseen was that of spam. The security measures put in place by Wordpress do a wonderful job blocking spam from comments. Added to this, one must have a previously approved comment in order to be able to comment without moderation.
Therefore, comment spam was able to be dealt with efficiently. It also appeared, after many weeks of the site being live, that there were no other spamming problems. However, on one recent day, there began to be a lot of spam users registering and creating posts. This was unexpected as it should have been dealt with by Wordpress, and had not been a problem in the past. **However, with no current solution to this recent problem, registration has been closed on the site, and as such it is requested that if you wish to log in and use the site, you do so with one of the usernames and passwords supplied in Appendix 5.**
Chapter 7 - Conclusion: Part 1 - Critique

The first part of the conclusion for this project analyses and critically assesses the project as a whole, in relation to how the whole process went, the aims that were or weren't fulfilled, and the personal development effect that it had.

7.1 Fulfilling Project Aims

This section is relatively simple to assess. Previously (in 2.3.3), there were nine points outlined that needed to be achieved in order to consider this project successful. Here it can be seen if these points were all completed:

1. There is an availability for both Users and Developers to discuss not only amongst themselves, but with each other - bridging their communities
2. The Developer can upload their original content to the site
3. That Developer is then given a unique piece of text that allows them to create a post around that content
4. There is an ability for the Developer to edit said post
5. A Developer can login, use, and manage their own account
6. While linking capabilities to repositories etc. remain only at the URL level, this still is possible on the site
7. Moderators are able to edit any/all posts or discussions
8. Moderator have access to the database and can manage it simply and effectively
9. Moderators also manage emails, which are sent automatically as well as by users to a site-registered email address

It is clear that the aims of the project were fulfilled, and thus the project overall can be deemed a success.

However, we must add to this by comparing ConnectIt to the comparison chart made in part 2.2.1, and look at where it improves on those attempts.
When this is compared with what is above in 2.2.1, it is very clear that ConnectIt not only fulfils its aims, but also meets the market needs where the other sites and plug-ins do not.

Finally, the title of this project calls for the **bridging** of two communities. This is what has been achieved through the completion of the project.

### 7.2 Critique

While the project was a success, that does not mean that it is beyond critique. There were some aspects that fell short of the expected standard.

#### 7.2.1 Content from Other Sites

One extra addition to ConnectIt that was desired was bringing in content from other sites (e.g. GitHub), and finding better ways to link to it within posts. This was never realised however, as although it was attempt numerous times (and with a significant amount of time and effort involved), nothing came to fruition. In an ideal situation this would have been an aspect that did work within the site and benefitted the project overall. It was a shame that it did not happen.

#### 7.2.2 Coding Issues

There were a number of coding issues that came up over the course of the project. Some (such as 7.2.1 above) were not rectified, while others (such as pull requests) were. This was all detrimental to the project overall, and would have been better avoided. More apt project management, more time spent learning the coding languages, and more specific and refined research into the problem areas would have went a long way to avoiding this problem for the project.
7.2.3 Set-Up Issues

Finally, there were also issues with the set-up. Too much time was spent in learning the set-up of Wordpress, finding a web hosting service, and getting the site up and running. While a large portion of this can be attributed to working with this software and software systems for the first time, an equally large portion should have had time budgeted for that. Added to the issues involved with time management, was perhaps the self-taught aspect of the new software, where seeking help would have been more appropriate and could have aided the project.

7.3 Personal Development

Personal development was also a huge part to this project. Growth was seen in numerous areas that allowed for a vast amount of development.

Firstly, the learning of new languages: PHP, JavaScript etc., and development in other languages: HTML etc., as well as the learning of new software: Wordpress, online web-hosting, phpMyAdmin etc., were all aspects that grew out of this project. Not only had they not been used before, but they all involved a steep learning curve that forced concentration and dedication.

Secondly, project management was not a foreign concept, and had been used many times in the past. However, the sole and complete management of this project was a challenge - one which was successfully overcome. This added to personal growth in the areas of management, project development, time management, resource allocation and content management (among others) was a great bonus to completing the project.
Chapter 8 - Conclusion: Part 2 - Future Work

This chapter takes a look at everything that wasn’t done for the project. It does this by looking at what could be improved on the project, what could be changed and done in a different way, and what developments could be made in the future.

8.1 What Could Be Improved?

For this project, there were two very specific features that were felt could have been implemented, but due to time and coding constraints weren’t. They would be a great improvement to the site.

8.1.1 Download Function

The first of these is a download function for the Developers' uploaded code. This would allow Users to download full copies of executable code without having to go to an external source (such as a GitHub repository). The hopes would be that this would add to both User and Developer experiences for the site.

8.1.2 Login with Other Platforms

Another feature was the ability to login with other platforms (e.g. logging in with GitHub username-password). This would allow greater ease of use for Developers, thus making sure that they use ConnectIt more than they currently do.

8.2 What Could Be Done Differently?

Were somebody else to tackle this project, they could have done some things differently. Here are a couple of examples of that (with the possibility that they would be implemented in the future).

8.2.1 Greater Integration of Other Sites' Content

By having a completely self-contained site, it did not take advantage of using the content that could be gleaned, imported, or linked from other sites (once again, see GitHub, BitBucket). This was a decision made due to the desire to keep the site self-contained, rather than being an amalgamation of other sites and their content. However, if done
correctly, it could be achieved in a manner that would benefit the site and attract more users/content.

8.2.2 Plug-in Rather than Full Website

As per the initial idea of having a plug-in resting over other sites' content, the project could have followed its initial trajectory and developed in full as a plug-in. While this did not happen it was obviously considered as an option. This would once again benefit by linking with other sites and their content rather than simply self-containment, thus potentially expanding the reach of the project.

8.3 What Scope Is There For Future Developments?

8.3.1 Build on What's Mentioned Above

The most obvious move for future development is to look at what has already been mentioned in this chapter. That is to say, the next steps should be to implement: a download function, login with other platforms, greater linking to other sites' content, and development of a plug-in to go alongside the website. All of these options are considered feasible possibilities that should be implemented in the future after careful design considerations.

8.3.2 Teaching Facility

As a secondary requirement (i.e. not completely necessary) for this project, exposure to learning how to code was suggested. Through gradual exposure to code this is something that could happen with ConnectIt. However, to build on this, it could be moved to a primary requirement, and a full teaching facility could be implemented that would keep Users on the site, rather than pushing them to W3schools or Codecademy.
Bibliography

References


List of Websites

- AgilityHoster: http://www.agilityhoster.com/

- BitBucket: https://bitbucket.org/

- Codecademy: http://codecademy.com

- CoderDojo: http://coderdojo.com

- ConnectIt: http://ruairikell.com

- Drupal: https://drupal.org/

- Get Satisfaction: https://getsatisfaction.com
- GitHub: https://github.com
- Joomla: http://www.joomla.org/
- LoginRadius: http://www.loginradius.com/
- Netbeans: https://netbeans.org/
- OneAll: http://www.oneall.com/
- Sourceforge: http://sourceforge.net
- Stack Overflow: http://stackoverflow.com
- UserEcho: http://userecho.com/
- Uservoice: https://www.uservoice.com
- Wordpress: http://wordpress.org
- W3schools: http://w3schools.com
- XAMPP: https://www.apachefriends.org/index.html
Appendices

1: Original Idea

The original idea for this project was to have a communications style plug-in/s that would be integrated with other plug-ins and tools on a CMS (Content Management System) platform. This tool would allow users of that to discuss with and query directly the Developers of whatever feature they were using from the very website where they were using said feature. This would be instead of having to go to a Developer site/repository (for example) where the code would be contained, and attempt to query from there, or try to flag an issue.

The idea was drafted (see Appendix 2 below) to draft requirements stage, and some level of implementation was achieved from this. As can be seen, the draft requirements documentation closely resembles the final requirements document, and ergo, the final implementation did end up achieving some of what was contained in the original idea. What it did achieve however, was in the aims (with regard to discussion etc.) and not the specific architecture.

2: Original Draft Requirements Document

Title & Abstract:

Project: Bridging User and Developer Community Discussions for CMS Plug-ins

The purpose of this project is to look at the community aspect of user generated content, and developers' additions/changes to this content, with the end goal of creating a web-app platform that allows for social connectivity between content users and developers. This social connectivity will allow for community style discussion on the use and development of the content.

Background & Purpose:

This project, as described above, came about due to what I perceived to be a need for a form of social/online interactivity between users and developers. As such, research was done during the period of August - September 2013, and eventually the project title was
conceived, with the aim of creating a plug-in style interface/web-app to allow this to happen.

Introduction:

In order to begin this project, I have had to decide on what the best methodology would be to use, and for this type of project it came down to either using a "waterfall" style approach, or using an "Agile" style approach. Having said this, I believe that it is entirely possible for me to use aspects from both of these styles. I feel that Agile aspects will be in place as I will use numerous iterations of the process in order to further the development of my project. However, I do feel that there will still be more of a focus on one particular aspect each time around, following with a waterfall approach. For example, my project may have four phases, with phase one being Requirements, phase two being Design, phase three being Development (coding and testing), and phase four being Launch (final testing and 'okayeing' the project). This would follow with a waterfall approach. However, during each of these stages I would also be incorporating each of the other aspects, e.g. doing building and testing right through phase one, two and three. This would follow with an Agile approach.

Functionality:

Here I hope to address what exactly I want to have the end product able to do. The end product must by definition be accessible to both developers and users. As we can see from the use case diagram, this means that it must contain the following:

- Commenting functionality for both user and developer
- Ability for developer to link to their content (which is stored elsewhere)
- Ability for both users and developers to edit said content
- Login capabilities for both users and developers
- Linking capabilities for both users and developers
- Capability for storage of user and developer logins
- Capability to store edit and revision logs of content
**Software/Hardware:**
In order that I should be able to develop this properly, I had to decide what I would use to develop the project. In terms of hardware, there is nothing needed for this other than my own laptop. For software, there is a little to be discussed:

- Using Notepad++ to write code.
- Plan to develop as a Wordpress plug-in, to ease the amount of unnecessary coding to be done.
- Most development should take place using JavaScript, jQuery, and PHP, and possibly some HTML5
- Some research has already been done on these languages and a small amount of previous use, however a lot more learning is necessary.

**Performance:**
With links to both an external repository as well as being developed as a Wordpress plug-in, and being relatively lightweight, there should be few problems with hosting, speed etc. Users should be always able to use this product without any hitch.

**Considerations:**
For this project, there are a number of things that need to be taken into consideration, such as:

- Security for logins
- Ethics forms for testing
- How much maintenance will be necessary?
- Are there any other legalities that need researching?

**Limitations/Possible Issues:**
Some of the issues that may be faced, and limitations that we have are:

- Limited amount of time
- Scope creep
- Passing ethics forms
- Usage of any new software/languages
3: Complete Terms and Conditions from Site

Statement of Rights and Responsibilities

By using or accessing this website, you agree to all that is contained in this statement.

1. Privacy

We aim to respect all elements of your privacy. Your email address and personal information will not be shared by us, but may be shared by you.

2. Your content

1. Any post that you make may be edited or deleted by you.

2. If the administration deem any content not following the standards set out in Section 6, they may remove it.

3. Any file that is uploaded will remain on our server and will not be removed when a post is deleted. If you wish to have a file removed, you must email admin@ruairikell.com with your request.

4. If you make a comment or forum post/reply, you have the option of removing or editing it. If there is a problem with it, email admin@ruairikell.com with your request.

3. Your Account

1. If you register for an account, your username and email address is shared with the administrator.

2. You have access to your own account and may edit the settings, but you do not have access to other users accounts.
3. If your account, activity or behaviour does not comply with the standards set out in Section 6 the administrator has the right to remove your account.

4. You will not create more than one account.

5. You will not share your password, or let anyone else access your account.

6. You will not transfer your account to anyone else.

4. Other People’s Rights

1. You will not post content that infringes or violates someone else’s rights, or violates the law.

2. This site is intended for sharing, and as such all content is deemed open source. You do not have the right to violate this by claiming others work as your own, or posting non-open source or illegal content here.

3. If you are collecting information about others, you will make it clear to them that you are doing so, and not obtain any information under false pretenses.

5. Discussion

1. Comments and forum posts/replies must comply to the standards set out in Section 6.

2. Comments and forum posts/replies may be edited, held for moderation, or deleted by the administrator at any time.

3. Any Facebook commenting system issues or problems may be on Facebook’s part, or on our part. As such, contact the administrator at admin@ruairikell.com immediately in order to allow us to solve the problem.
6. Standards

This section outlines all the standards you must keep to in creating, posting, commenting, and discussing on this website. Failure to do so may result in termination of your account and possible legal action from the party or parties affected.

1. Violence and threats are not permitted. You may not credibly threaten others, or organise any acts of violence. When we perceive a genuine risk of harm or safety, we will remove the content, with the possibility of law enforcement involvement thereafter. Any content relating to the discussion of previous acts covered above will also be removed.

2. Any discussion relating to the promotion of self-harm, suicide, or related matters (including drug-abuse, self-mutilation, and eating disorders) will be removed.

3. We do not permit bullying or hate-speech. We encourage critical and challenging discussion on any content on this site, and we make a distinction between humourous and serious speech. We do not permit attacks on others based on their race, ethnicity, national origin, religion, gender, sexual orientation, disability or medical condition.

4. Any content deemed too graphic (covering gore, abuse, extreme violence and sadism), will be removed.

5. Nudity and pornography is not allowed on this site and will be removed.

6. You are only allowed to share content on this site that is open source, creative commons, or that you have expressly asked for the intellectual property rights to be allowed to post it.

7. You may not promote the consumption of goods in a form deemed to be advertising (by the administrator). This content will be removed. You also may not complete any transactions involving currency, trading or bartering on this site.

8. Any attempts at spam or phishing will be dealt with severely, and all content will be removed.
4: Screenshots
5: List of Users and Passwords for Access

- **Username:** User3  **Password:** iYrQgYqDPXBw
- **Username:** User5  **Password:** zHFDsUNTBh6B
- **Username:** User6  **Password:** FkFryEhXEyfJ