Syntax and Semantic Learning Tool Codex

Thomas Hutchinson

May 2018

Computing Science

Supervisor: Dr Lindsay Marshall

Word Count: 14,908

Abstract

This paper wishes to explore and build upon already available Syntax and Semantic Learning tools currently available on the market (such as Stack Overflow) [Stack Overflow 2018] to create an application. This application will then provide a better learning experience to the end user allowing them to find and pick up required syntax and semantics for any required language.

The motivation behind this Dissertation and application was an attempt to apply some of the advances in the pedagogy of programming/scripting languages to a real learning tool. The new application was meant to be exploration of what can be possible when the pedagogical theory is applied

Declaration

"I declare that this document consists wholly of my own work except where explicitly stated"

Thomas Hutchinson

Acknowledgements

Table of Contents

1. Introduction

This section provides a general overview of the Dissertation covering the Dissertation structure itself as well as the general aims and objectives.

1.1. Aims and Objectives

Aim: To create a tool that can be used by both intermediate and expert level programmers to aid in their acquisition and learning of new [programming/scripting] languages syntax and semantics.

1. Research existing products currently available on the market and how my tool can offer a service that is different from competitors while still allowing its users to learn new syntax effectively.
    a. Efficiency will be marked by the user being able to find their required information with only a single search and to have found a suitable post within 5 minutes *If a suitable post exists*
2. Research the Pedagogy of [programming/scripting] languages to ensure the tool teaches its information in the most effective way possible.
    a. Effective learning will be defined as a user being able to acquire that knowledge then apply it in a project/ work environment.
3. Conduct a review into the user perception/usage of the currently available tools as well as my hypothetical tool.
4. Develop said tool using the collected information and research to influence design decisions and the overall user experience.
5. Evaluate the performance of the tool in relation to the research from the previous objectives once completed by measuring various statistics such as user activity and user retention.

1.2. Table of Figures

1.3. Table of Tables

2. Background Research

2.1. Market Research

2.1.1. Reasoning

When creating the new application, it was important to review the current applications available and used on the market. It would be a waste of time and resources to create a product that's main feature is already encompassed by another developer. Hence, why it was important to conduct a throughout review of the currently available products taking note of their primary features, positives and negatives.

2.1.2. Method

To ensure each application is reviewed thoroughly and only the points of interest are examined each application will be subjected to a specific process. The reviews will primary focus on how the platform delivers, sorts and presents its content to the end-users. On top of this since the created application will be crowd sourced only currently available platforms that also make use of this content creation system will be reviewed. This is done to ensure that all conclusions are related to the created application and to also help reduce the scope of related applications.

Once a valid product to review had been identified the main features would then be carefully examined to see how the audience used and reacted to those features. A valid product once again is anything that teaches programming/scripting languages to an audience with that content being crowd sourced. A main feature can then also be defined as any feature that greatly helps the learning experience or any feature that is present in that application but no others on the market.

2.1.3. Findings

For a full detailed list of findings see appendix A for the report.

2.1.4. Summary

While searching the market for products like the one outlined in this paper one product stood out from among the rest and that was Stack Overflow [Stack Overflow 2018]. Due to its wide spread use and popularity it boosts one of the widest ranges of content on the internet. This directly feeds into its popularity as it provides tutorial material on just about every topic. It's also helped by having very good SEO (Search Engine Optimisation) allowing users to easily find posts through popular search

engines such as Google. In the audience research this was one of the most popular points of the system *(see chapter2.2).*

So, it can be concluded that Stack Overflow [Stack Overflow 2018] simply works by being one of the most popular systems on the market. However, it does fail in the fact it doesn't adequately reward posts that fully explain their topic. As mentioned in the full market report *(see appendix A)* users creating content are incentivised to provide straight solutions to responses. This does not encourage learning of the background syntax and semantics and this is where my system could provide additional functionality.

By learning from the downsides of big sites such as Stack Overflow [Stack Overflow 2018] it is possible to further improve the outlined application. In this case by focusing on giving the user tools to upvote a post on a range of qualities that have been found to improve learning. The application can perform better than these other services. That said It will be incredibly difficult to ever compete with these services simply due to their large content and user base. That is why when it comes to the final evaluation it will be important to remind users to judge on the potential of the new system. With that in mind it will be easier to judge how the altered focus on learning and improving over simply solving issues will improve programmers greatly.

2.2. Audience Research

2.2.1. Survey Rationale

Another important part of determining which direction my system should go in is by asking my target demographic a few questions relating to their current tool usage and learning habits. I can achieve this through focus group testing but also an audience survey. This survey would be targeted at both student/" junior" developers and senior developers alike to gauge opinion about my possible tool. I primarily want to find out how developers learn new syntax and how/what tools they currently use to solve problems. Below is a list of possible questions for the survey alongside a short purpose of asking that question.

2.2.2. Survey Results

After hosting the survey online for one month a total of 16 answered it. While this response number is not brilliant for forming conclusions about my hypothesis. Using it in conjunction with my academic research as well as the follow up survey after the completion of the app will help to give more reliable data about the validity and correctness of my hypothesis.

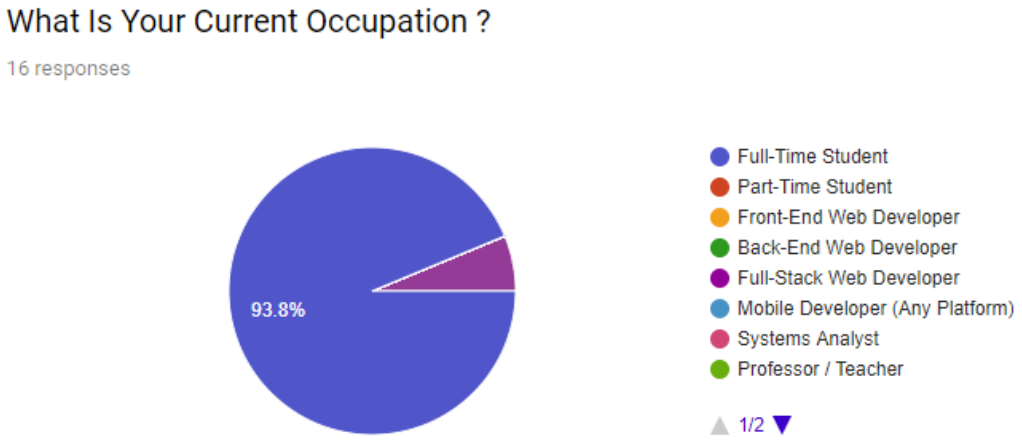

Figure 1: User Survey - Current Occupation

Another point of note is the distribution of occupations. Due to the limited reach of the survey it was primarily completely by Full-Time Students. While this isn't necessarily a negative point it does mean that my conclusions must be adjusted to account for the primary demographic of my application being students.

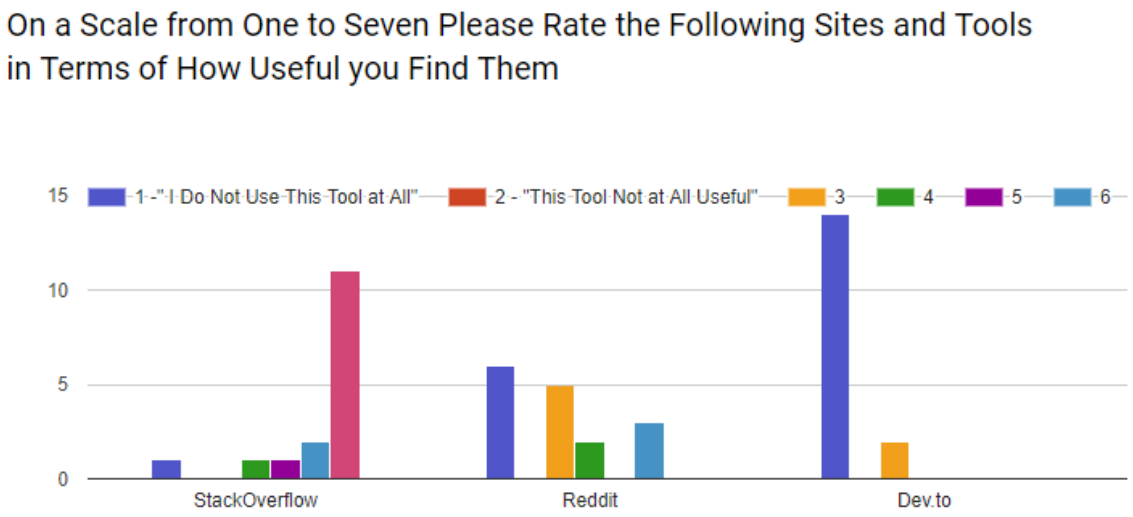

Figure 2:User Survey - Other Tool Usage

The survey also revealed that Stack Overflow [Stack Overflow 2018] is by far the most popular language learning tool amongst the student demographic. This means that the market research should primarily focus on what makes Stack Overflow so popular with this demographic.

On top of this it will also be worthwhile to explore why the other tools do not hold as much favour with this selection of students. As you can see from the above graph Stack Overflow was marked as a 7 in usefulness by most students who answer this section. That is directly compared to an Article based site like Dev.to which received a high selection of 1 responses. This data will be useful in future conclusions regarding the effectiveness and popularity of certain applications over others.

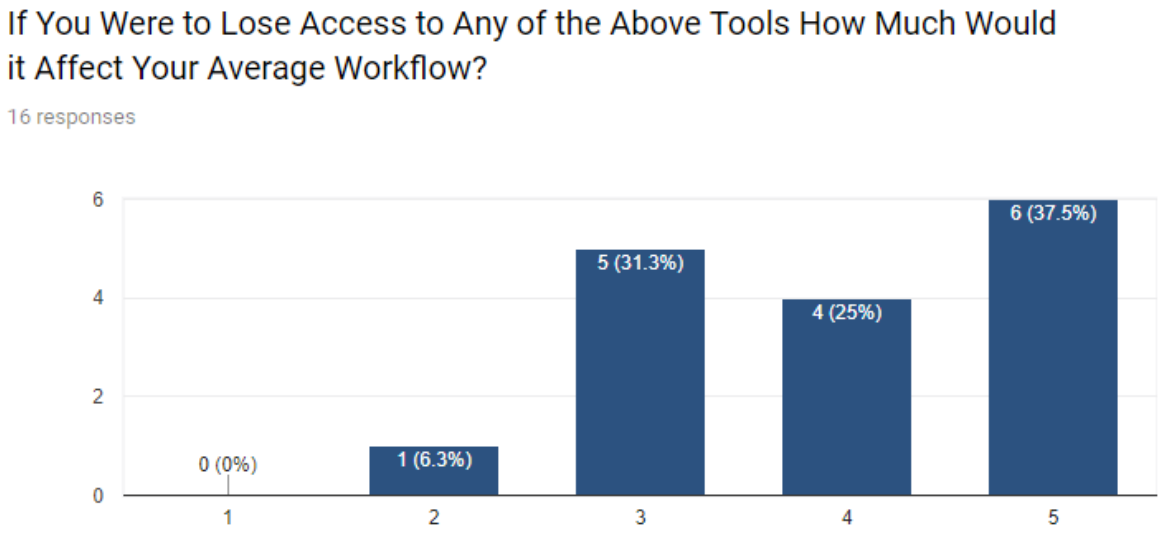

Figure 3: User Survey - Workflow Usage

Another interesting response is trend towards tool dependency. Out of the 16 people who answered the survey 62.5% of the respondents voted 4 or above claiming that losing access to their preferred tool would hold a detrimental effect on their normal workflow. Once again this is useful information

for later as it shows that users may not be learning or retaining information which requires them to make multiple visits to their tools. On the other hand, it could just show that students are required to develop in wide range of languages which means they are in constant need of additional learning materials.

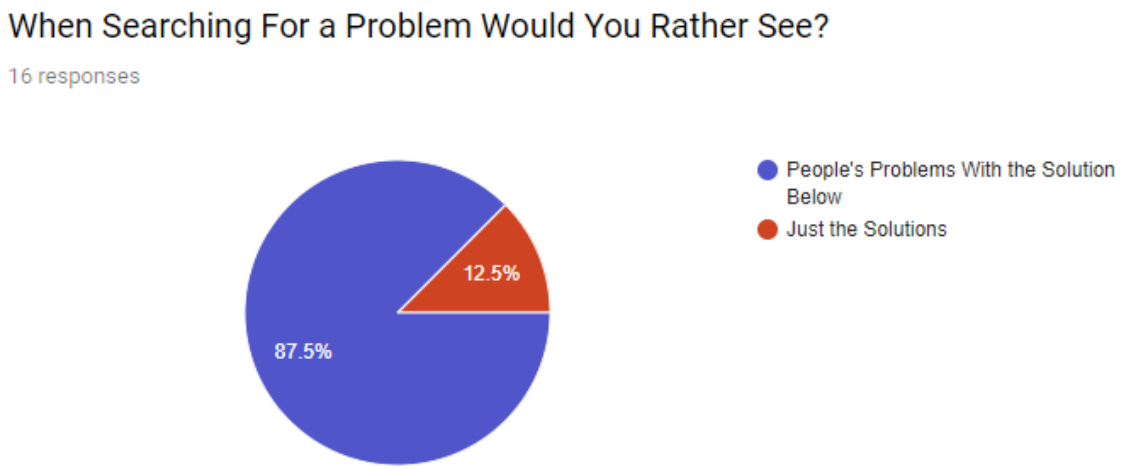

Figure 4: User Survey - Solution Based

Regarding the layout and development of the app one surprise was the number of respondents who prefer a solution-based search system. The respondents answer clearly show that it is greatly preferred when a tutorial is given with a specific real-world example. When it comes to developing the application, it will be important to attempt to implement this idea and to allow the linking of tutorials and specific examples. This could be a link to the idea that high level theory is difficult to learn by itself so showing the application of that theory makes it easier for a wider range of programmers to learn and apply that concept. Therefore, for the tool to be successful with this user base it needs to rely on linking theory to concreate examples.

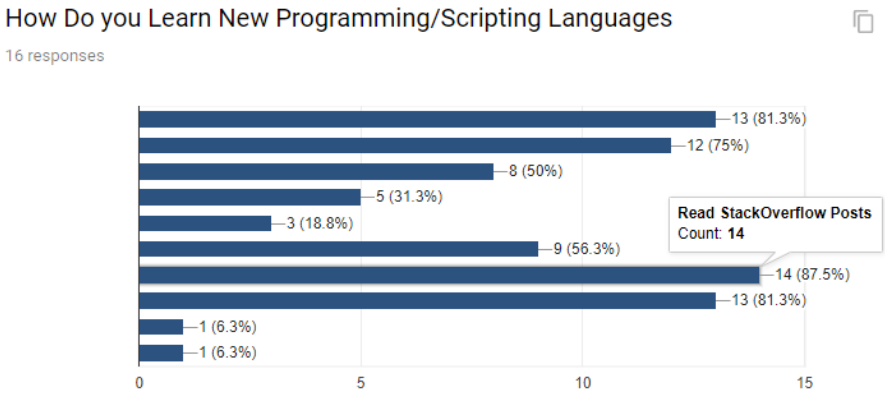

Figure 5: User Survey - Language Learning

Yet another interesting point of note is the number of respondents who rely on Stack Overflow [Stack Overflow 2018] posts to acquire and learn new languages. Once again this speaks volumes about Stack Overflow's ability to provide tutorial material to its users who on average return whenever they need to learn a new language. The second highest response for this question with 13 answers is "taught by a teacher/expert". So, the best way or at least the most favoured way to learn is by example or by expertise. This will be an important element when it comes to my application as It needs to allow the experts of a language to fully demonstrate and create tutorials for specific topics. By looking at these survey results it can be concluded that the respondents prefer learning from experts and being given examples of the theory in motion.

See full results in provided Excel Spreadsheet found in the supplementary material.

2.3. Language Research

2.3.1. Reasoning

Another aspect important to the background research is the study of Computing language itself. By studying this area, it will give some direction to how the application should be developed and what types of features would be useful. Giving the users the correct tools to create posts that help to teach a range of languages and language features will be incredibly useful in keep users retained. On top of that making sure the created application is robust enough to support a range of languages will be important in getting plenty of users creating content for the application.

2.3.2. Method

Simply research languages through available material such as: official documentation, online forums, unofficial documentation and created tutorials. All this data will be gathered, compared and reviewed to determine conclusions and ideas about specific languages. Those conclusions will then be reviewed and used to help determine how the Syntax and Semantic Learning tool should be created and what types of features it should have.

2.3.3. Findings

For full findings please read the full report in Appendix B

2.3.4. Summary

Languages have a wide range of features and an even wider range in which those features are defined and implemented. For specific examples please refer to the *appendix* but it is clear than even from a brief review of various popular languages that they can differ rather largely. That produces the conclusion that users must be able to search via specific languages and able to filter content based on the language they are currently interested in.

Users wanting to learn how to perform a specific technique in one language need specific syntax for that language so by putting a specific tag for languages. Users will be able to more easily find specific content to their liking. That said many of the languages share the same semantical features such as if statements or for statements. These posts could be created with specific syntactically definitions which then link into a single semantical post that explains the background workings. This is because over many of the languages the semantics remain the same however the syntax can change.

2.4. Academic Research

{For full research see Appendix D}

2.5. Implementation Technology

The system will be implemented as a progressive web application. This decision is based off the audience survey *(result below).* As you can see all platforms was selected as the most popular option closely followed by Desktop Application (windows). Due the large different in application types this Dissertation does not have the resources to develop on all platforms. It is therefore decided that a Web Application will be created as it services the most users with the available resources.

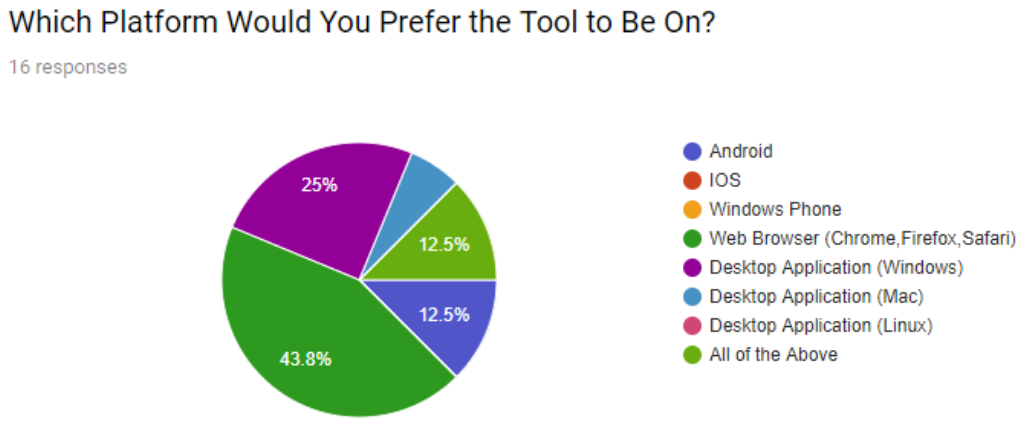

Figure 6: User Survey - Platform Preference

2.5.1. Web Applications

A Web Application is rather like a normal website though it attempts to mimic the behaviour of a native application. This means that the user has a constant state that is linked to their account, so they can use the application across various computers and even in some cases offline. This is achieved through extensive caching both on the user end and the server end storing any settings or posts they create. That means that when they re-connect that content is readily available.

Another advantage is that the software is Operating System free. As outlined in the above results the users want the application on a range of platforms however there is only so much development time. A Web Application can perform on several platforms at once simply because it is a normal website. Any platform that has a modern browser such as Google Chrome, Firefox or Edge can view and use the application.

What makes a Web Application stand out compared to a normal website is a cross between its cache ability, platform independence and real time nature. Users do not need to press buttons to load content it is simply loaded on demand and then cached so it may be viewed at any point. This will be achieved through Google's Polymer [Google Polymer 2017] as well as Google's Firebase platform [Google Firebase 2018]. These two technologies allow real time database calls and updates which means the user

can stay constantly update to date with the latest content. This platform has also been chosen due to prior experiences which will help to speed up development.

Other platforms are available such as Re-act [reactjs 2018] from Facebook however, I have not had as much experience of this platform or any others. The project will be held to a tight deadline, so all advantages must be taken to ensure the application is made to the highest possible standard in the shortest time possible.

3. Methodology and System Design

3.1. Overview

The aim of the Syntax and Semantic learning tool is to facilitate the learning of new syntactical features of programming/scripting languages. On top of that the tool will also allow users to gain a greater understanding of the background semantics of that specific syntax. This is done to help increase the programmer's overall proficiency as they will have a fuller understanding of that syntax and how it works. All of this will be achieved through a crowd sourcing model. The reason the application will rely on crowd sourcing is because there are far too many languages that are used by professionals. Creating content for these languages would require more effort than is currently available for this project.

Crowd Sourcing allows the application to rely upon its community to keep it alive. This is a double-edged sword however as if the community does not support the application the application will fail due to a lack of content. That is why this prototype aims to get an idea of what the community would like from a syntax and semantic learning tool. This information could then be used to fully develop the application in a manner that the general programming community would approve of. This in turn will increase community retention and therefore support the life of the application itself.

3.2. Functional Requirements

Number	Name	Description
FR1	User can create tutorial posts	A singed in user can create a post detailing a specific syntactical/semantical feature of a language.
FR2	Users can tag other posts to their created post	A signed in user can tag other posts (creating a link between them) to help the explanation of specific topics by linking to other user's (or their own) content.
FR3	Users can view other users posts and rate specific attributes of that post.	A user will have access to a feed where they may view other users posts and either upvote or downvote those posts based on various criteria such as: well explained, well linked, and general quality.
FR4	Users can Search for other user's posts based on specific criteria such as username, post title, post language and sort that search data based on its attributes.	A user can search for posts using the username, post title, the language that post is explaining and then sort the post based on different attributes (well explained, well linked etc)
FR5	Users can create bounties, rank other bounties and mark their own bounties as fulfilled	A bounty is a request for a specific tutorial to be made. Other users can then view these bounties and create a specific post for them. Users can also rank these bounties to help improve its visibility to other users.

FR6	Users may view their own created posts and delete them	If a user is unhappy with their post they may delete it from the website.
FR7	Users can save posts for offline viewing.	By viewing a post, a user can save it to an offline cache where they may view it later.
FR8	Users can comment on posts	By viewing a post, a user can place a comment on that post discussing its contents. Other users can then view this comment.
Documentation		
FR9	The code will be thoroughly documented	The code base will contain a reasonable amount of comments to explain it's working as well as to credit any externally created code.
FR10	A user manual will be created	A public use user manual will be created to allow users to understand how to use the application.

Table 1: Functional Requirements

3.3. Non-Functional Requirements

Number	Name	Description
NFR1	The site should be inherently easy to understand and use.	A user should be able to use the site without any prior guidance. This should be achieved by basing the interface of currently available applications to reduce the time to learn.
NFR2	Posts that encourage semantical learning over simple solution-based responses should appear at the top of feeds and searches where possible.	Where available posts that thoroughly explain concepts and don't just provide solutions should be favoured in results. While this can't be directly controlled post, attributes should help sort this content.
NFR3	Creating a post should be simple and easy to understand not requiring any additional knowledge.	This will be achieved through keeping the text editor and post submission form as simple as possible.

Table 2: Non-Functional Requirements

3.4. High Level Design

The Web Application itself will be created with a cross of Google Firebase [Google Firebase 2018] Technologies as well as Google Polymer 2.0 [Google Polymer 2017]. Polymer focuses on the combination of multiple web components which are individually created and defined in custom DOM scheme. This helps to abstract away complex HTML, CSS and JavaScript which could become quite difficult to manage in a larger project.

This works by creating an element much like a normal HTML element would be created. For example, you would create a form using the standard HTML 5.0 mark-up. You can then import some elements from the Polymer library and define that HTML snippet as a custom element. That element can then be imported into another HTML document and used much like a standard HTML tag. So that form could be called “my-Form” and now you can implement simply by creating a set of <my-form> tags. Once again, the main advantage of this is to abstract complexity away when it is not being worked on.

In my specific use case. I can create separate pages and elements and then import them into a single application element (see diagram below). This means I can easily focus on one element at a time rather than creating a single huge page. It also makes it very easy to swap elements in and out if I want to experiment or try different implementation strategies.

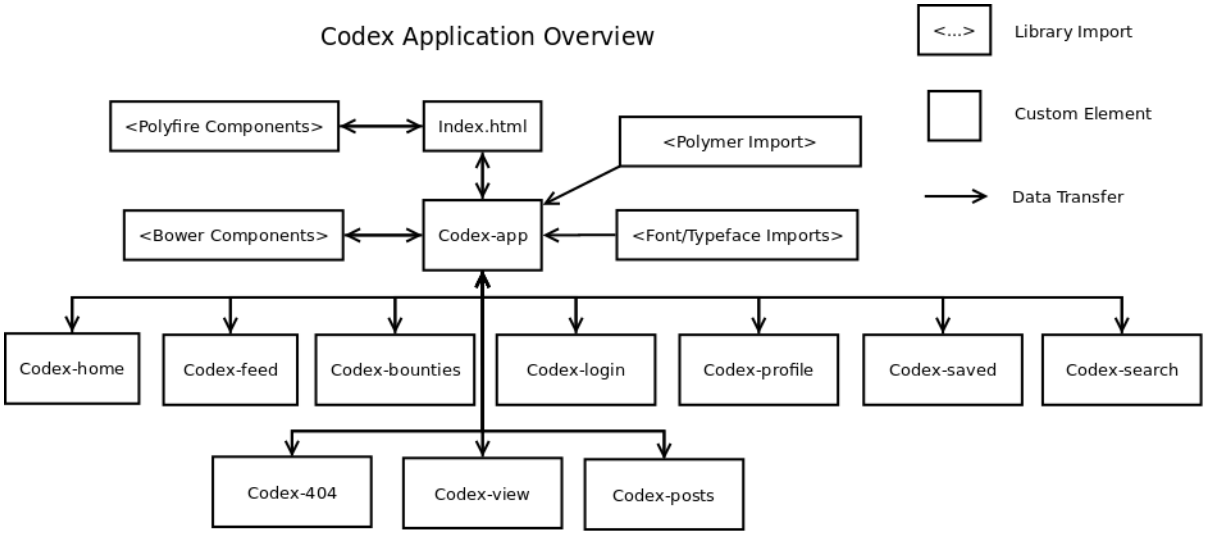

Figure 7: System Design - Application Overview

On top of the custom created elements there will also be the Bower components. Bower is a component management software that assists in the installation and management of dependencies of all the additional third-party components I will be using. Most of these components are constructed by Google or Google afflicted companies and include such things as: layout assistance, loading bars, view swaps and other smaller components. These will help save development time by cutting down on the development of items which have already been created by others.

The Polyfire [Polyfire 2018] components are the components which allow the combination of the Polymer and Firebase. These components handle elements such as user authentication and interaction with the real time database. By importing them through the index page rather than codex-app the entire site can have access to the data from the Firebase connection. This data is then protected by Polymer’s

shadow-DOM. This is a ratification DOM structure created to handle all the various components. It also helps to prevent attacks by making it incredibly difficult to poll data from the website itself.

3.5. Backend Server-Side Design

Firebase will be used to handle all back-end data processing. This primary involves user authentication as well as requests and posts to the database. Thankfully, Firebase directly supports both functions and has cross-over components that allow the communication between polymer and firebase. These are included in a library called Polyfire [Polyfire 2018].

Starting with the database. The application will make use of Firebase's "Real Time Database" [Real-Time Database 2017] structure. This is very similar to a SQL-less database. That means it does not rely on common SQL style functions and it is not a relational style database. Meaning that it does not create relations between records. This has both its advantages and disadvantages. The disadvantages being that having relations between records becomes difficult/impossible and the repetition of data can easily happen. That means careful consideration must be taken when designing the database.

"Real Time Database" [Real-Time Database 2017] from Firebase works on a Document foundation. This means that the database consists of objects. These objects then consist of fields and those fields can also be arrays/lists of objects within themselves. Through this mechanic one may achieve a relation between data. So, when creating the database relations will be displayed through these internal lists. For example, the post document will be a list of posts and within those posts each post will have another list representing its comments.

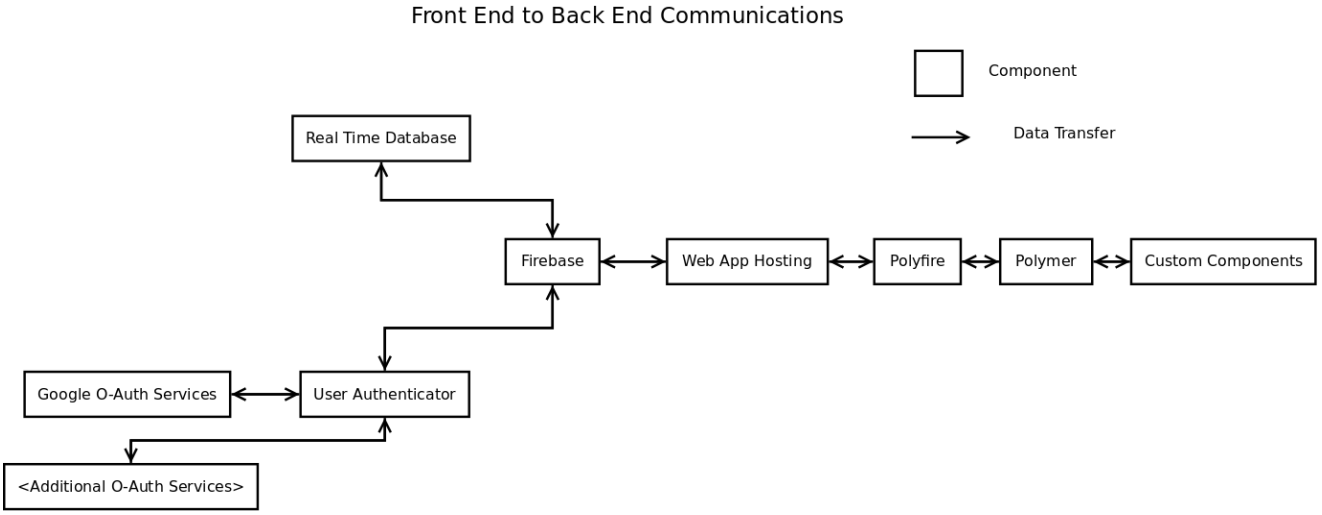

Figure 8: System Design - Communication Overview

The primary advantage to using this approach is that is becomes faster and easier to process single objects. Codex (the application) will often be dealing with a single post or a small number of posts (less than 50). That means when pulling a post, it will be much easier to handle of its data together in an object style paradigm. This is because all the data is present upon call avoiding the resource intensive procedures such as table joins. Real Time Database avoids this by having all the objects essentially being pre-joined saving the server from performing this expensive operation. This in turn

speeds up processing allowing the site to process more users at once which is important in a site that may need to scale to a large user base quickly.

3.6. Database Design

As mentioned in the previous section to increase the speed of the web server and therefore reduce user-end load times a document style database will be implemented. This foregoes costly operations such as table joins for storing “objects” within a database. These objects can then be individually queried from the application which will return all the data held within that specific object. With that in mind certain design constraints must be considered when designing the database.

For a start if not handled properly the database could suffer from a great deal of anomalies. Anomalies can be created when there are duplicate versions of data present and one item of data is edited or deleted which creates inconsistencies across the database. These inconsistencies can then escalate resulting in wasted memory space from storing the same data twice or confusion when it becomes difficult to tell which item of data is the most current. On top of that anomalies can also occur if two data items rely or reference one another and one of those items is deleted. This now means that an object has a “Null” field and is missing data making that record invalid. So, when it comes to developing the database all objects must contain all data relating to them where possible.

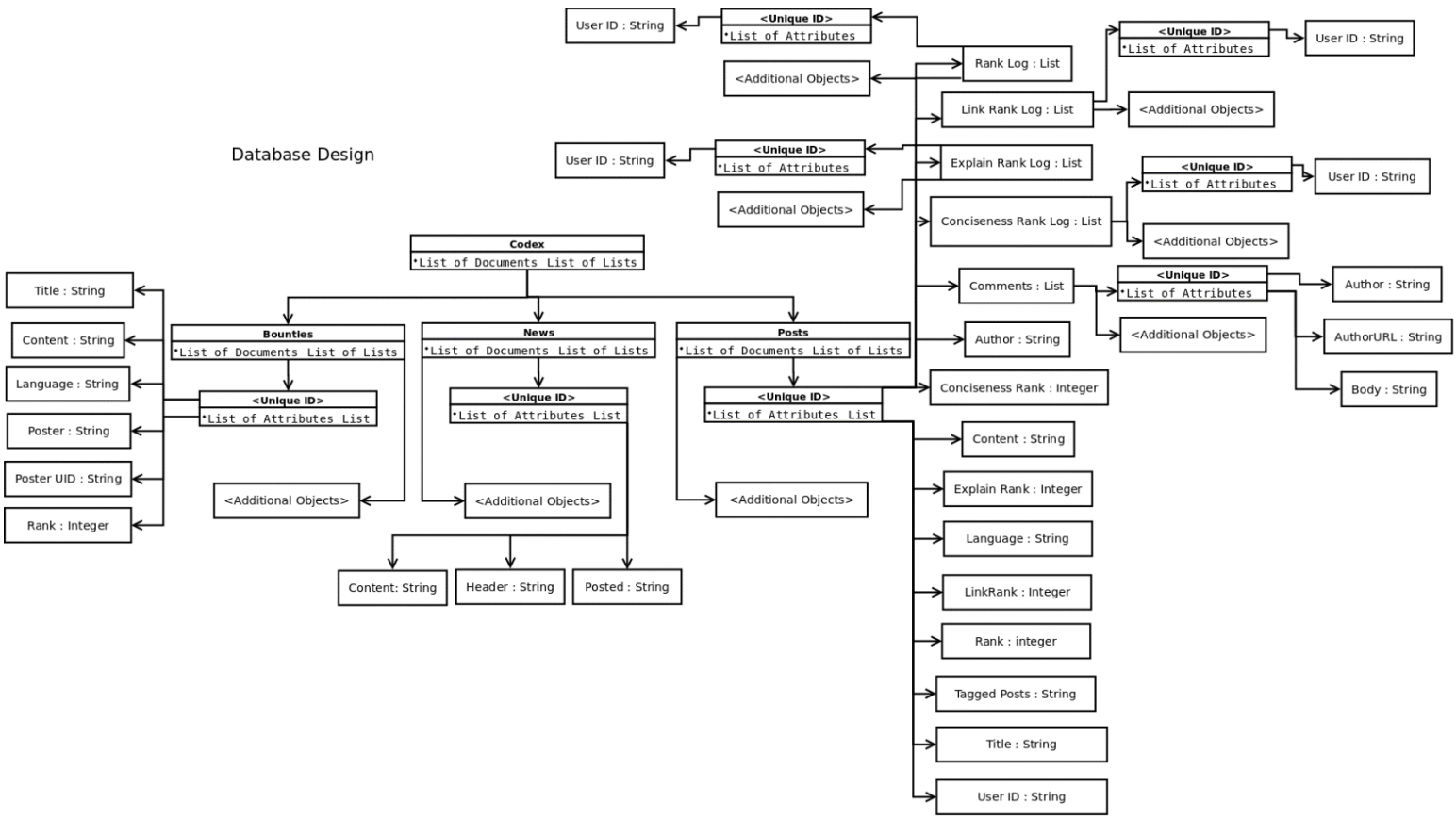

Figure 9: System Design - Database Design

See above figure for full design of the database. As you can see the post object is by far the most complex out of all the objects and that is simply because it is the centre of the application's design. All aspects of the system rely on posts being managed correctly and that means that the post must contain a lot of data so that I can do its job successfully.

The log objects attached to the post objects are lists that are utilised in the up-voting procedures. When a user upvotes a specific attribute of a post their unique identification number is added to this log object. This means that in the future if the user attempts to vote on the same post again the server can check to see if that user has already voted. If they have voted it will simply disallow them from voting further. A separate running number total of total votes is also kept, and this is to save the server performing a count operation on the list every time the rank is requested. It can be expected that the rank will be requested far more than the upvote procedure, so it makes sense to store that number for easy retrieval rather than calculating the number of votes from the log.

Another aspect of note is how the comments are stored alongside the post they are related to. Once again, this document style database does not support conventional relations so to emulate that behaviour comments are stored within the post object. Meaning when a user views post the comments are passed alongside that post saving additional queries and complex joins. This is done to reduce load on the server to allow the application to serve as many users as possible on limited hardware and bandwidth.

3.7. User Interface Design

Linking back to the market research in chapter 2.1 all the successful learning tools implement a simple yet powerful interface. This style of interface can be defined as having a navigation bar with minimal options upon it and relying upon the user to utilise a search feature to narrow down content. As seen with Stack Overflow the search bar takes up prominent space at the top-centre of the page.

This is something that should be directly applied to the application. These sites are successful because users can quickly and easily navigate them to find their required content. Any site that impedes the users progress will quickly fall out of favour. Once again linking back to Geiger [Geiger 2011] on Crowdsourcing systems. The system must retain users can work together with their community to encourage a healthy life cycle. The cycle consists of new users discovering the site through search engines and word of mouth. They then begin to integrate that tool into their daily work routine and eventually they will begin to make content for the site encouraging more users to join.

With that in mind the interface needs to be easy to grasp. The initial designs for said interface are as follows in the figures below:

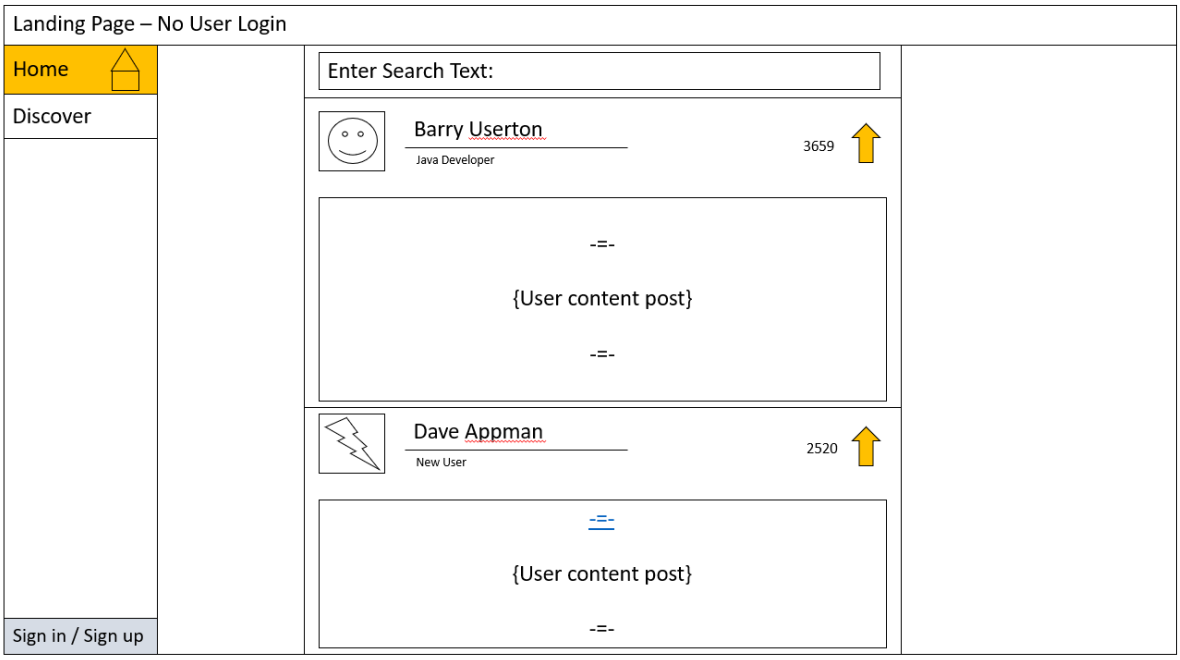

Figure 10: Interface Design - Homepage

The above figure shows the Homepage/feed page. This is the initial design on what the user will see when they first login and it is an attempt to encourage them to explore and view new content they may not otherwise see. What is important though is the prominence of the search bar. The home page aims to give users an idea of what types of content will be present on the site meanwhile, not obstructing them if they are looking for something specific. If a user wants a specific post they can quickly and easily search for it.

This feature has been directly borrowed from Stack Overflow as makes the site easy to navigate. This featured was also praised in the audience survey *(see appendix bleh)*. So, it's an important feature to add to the site.

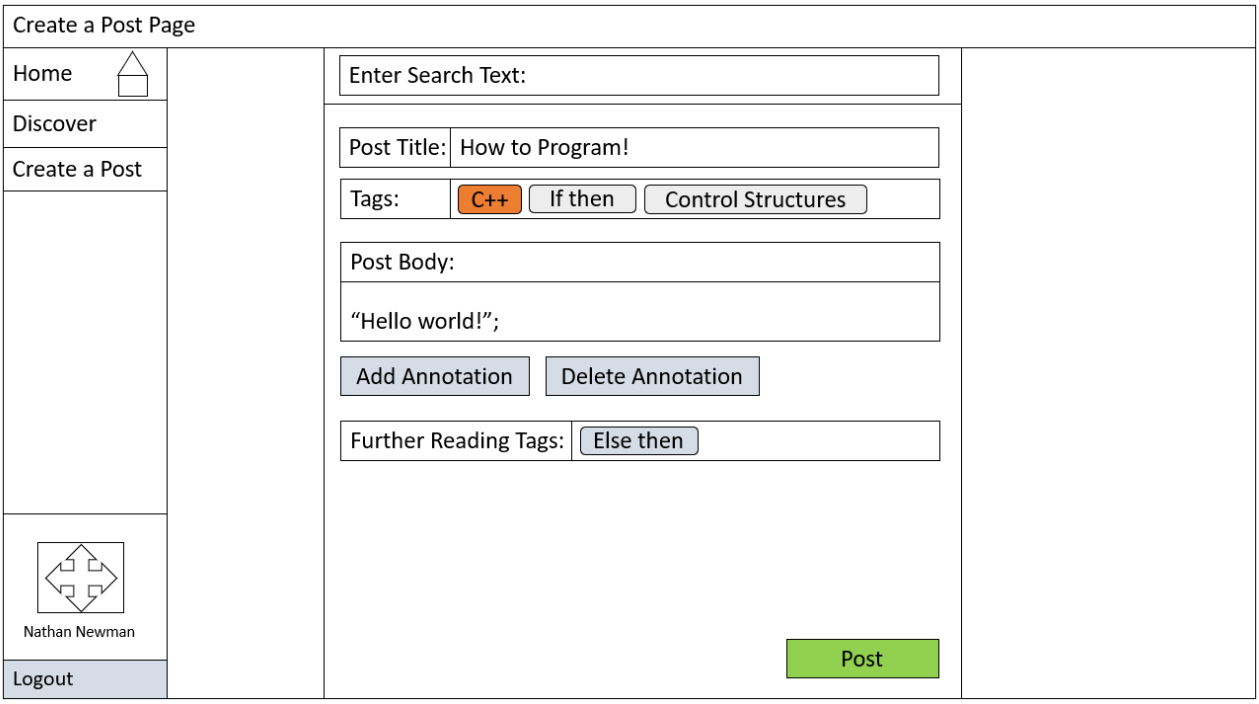

Figure 11: Interface Design - Post Page

Following the homepage is the create a post page. This is the second most important part of the site as it is where more dedicated users will spend most of their time. Here users will be able to create new posts, tag over posts and push that to the public feeds. Once again, it's important to make this

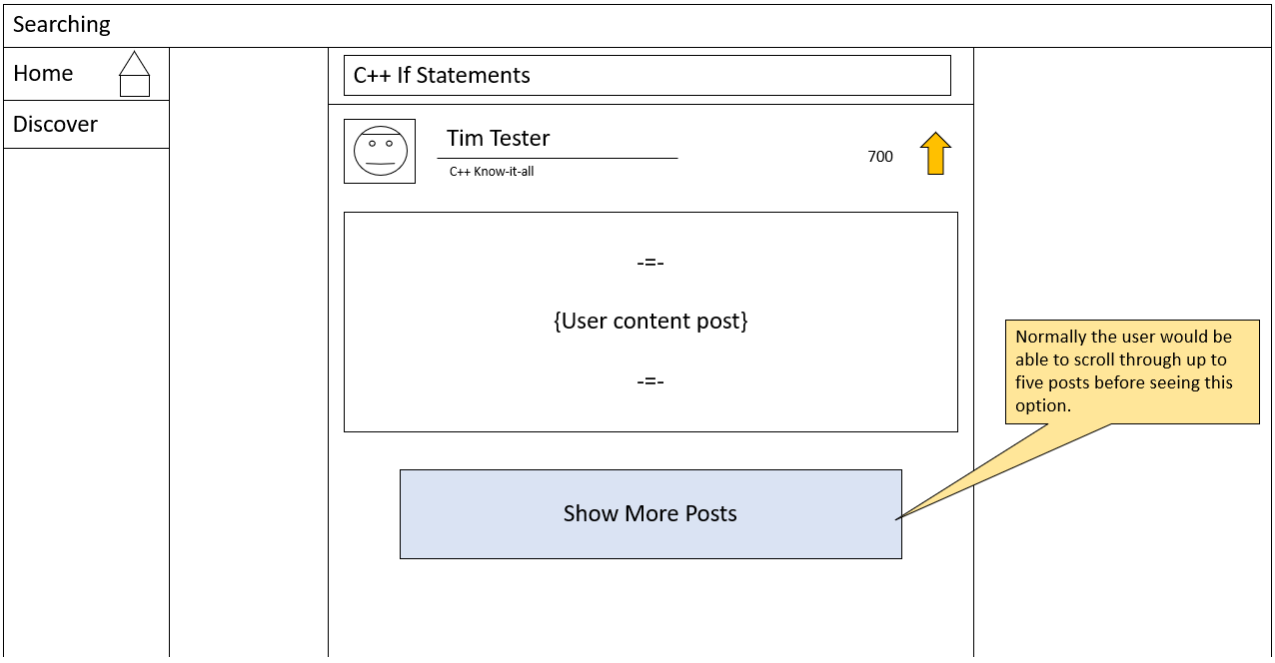

Figure 12: Interface Design - Searching

part of the site easy to learn and use. That is achieved by keeping the interface minimal and not over cluttering it with buttons. Between this design and the final product some of the buttons may even be removed to help keep this page clean.

Finally, there is the search page. This after the home page and the post page is the most important page of the site. It is where most users will spend most of their time as they search for specific content. In the audience survey *(chapter 2.2)* users favoured sites with powerful search tools that allowed them to not only search for posts to but search in different manners. This could take the form of sorting posts based on specific criteria. It could also take the place of allowing users to search by programming/scripting language or maybe be a favourite author. Experimenting with this search tool then gauging user response will be a key aspect of this Dissertation to see how users utilise search tools.

3.8. Design Reasoning

All the previous design features have been directly influenced by the research detailed in chapter 2. This sub chapter aims to provide a little more detail on the exact links between the research aspect and the design aspect.

Crowd Sourcing is a key word of this project as the system will succeed or fail based on how the audience perceives and uses the application. Zwass *[Zwass 2010]* spoke a great deal about communities and what impact they have on the crowd sourcing model. With that in mind during the research of various market products *(Chapter 2.1)* and the audience research *(Chapter 2.2)* a focus was placed on how the application developed their community. Stack Overflow became quickly clear that it's community was based around solving problems and gaining points to increase one's ranking. Meanwhile sites like Dev.to focused around personalities creating articles. This is then reflected in how the audience survey *(Chapter 2.2)* perceived these applications. The audience had integrated Stack Overflow [Stack Overflow 2018] into their work routine meanwhile far fewer users made use of Dev.to on a regular basis. This could just be down to Stack Overflow being a far more popular site, but it could also be down to users preferring solutions and direct learning over interest articles.

Taking that into account it is important that every aspect of design of the application leans into this research. The entire site should focus directly on teaching user's new topics they want to find out about and stay away from opinion articles and interest articles. One way of encouraging solution and tutorial posts is through the rating system. By allowing users to rate posts based on their conciseness, well explained, well tagged then allowing them to filter posts based on those ranks it should encourage posters to adopt those qualities. Once again this is no certainty until the application is passed onto testing and the end users return their feedback on the feature.

The primary focus of the site is enabling users to teach themselves so when it comes to evaluating the successfulness of the rating and search systems a great deal of care will placed into getting throughout feedback. Testers will be questioned on how they used the rating system as well as to what extent they used the sorting system on the search page. Using this feedback, it can be hypothesized how users prefer to learn. As mentioned previously, it will be extremely difficult to gauge if the users have improved their skills through the usage of the application, so this paper must rely on the opinion and views of the users/testers time with the application.

3.9. Project Plan

Due to the limited development resources the project will generally take the form of a Waterfall approach. This means that tasks will be tackled sequentially and as sections become complete new tasks will become available. If the development team was larger an Agile development style could be adopted where tasks were committed to in stages and the order of completion in those stages wouldn't be of importance.

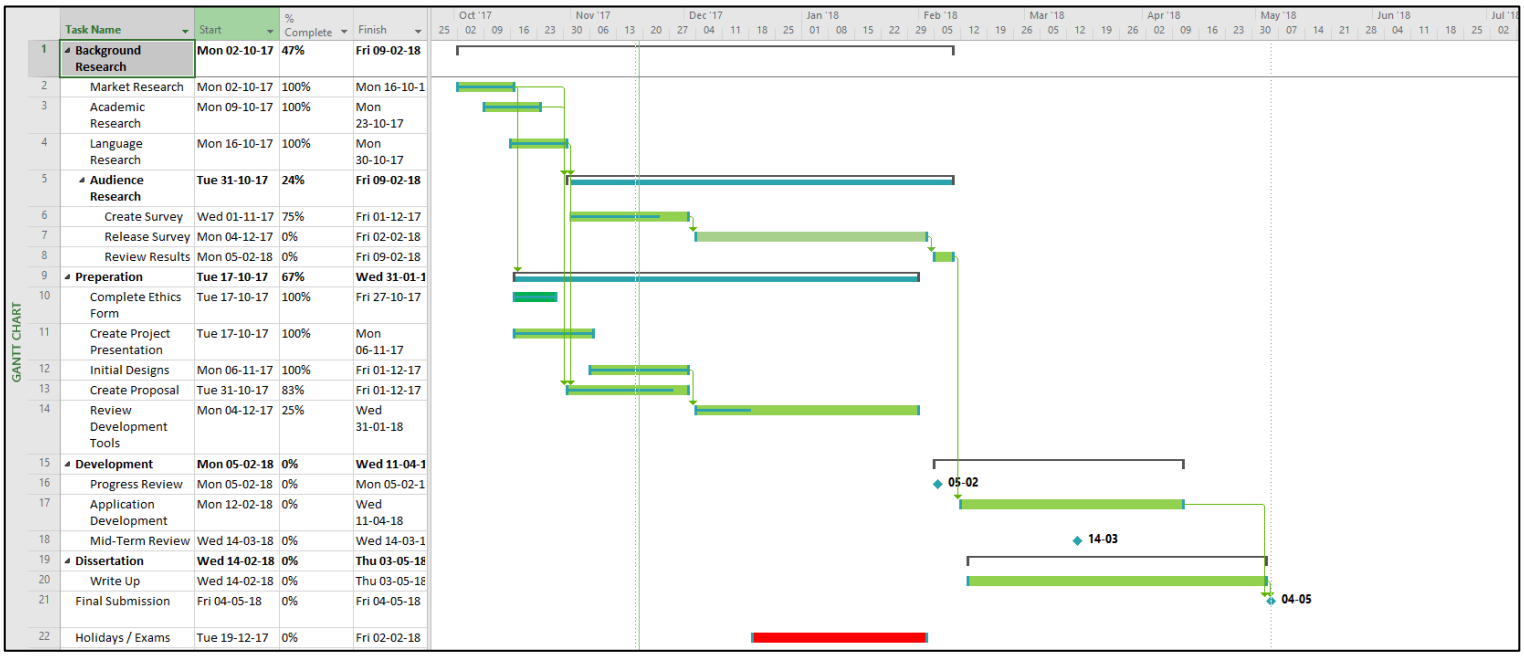

Figure 13: Project Plan

However, the development team only consists of one developer, so the development schedule will fall under a waterfall mentality. Development will begin with the design and research section. During these phases a focus will be applied to deciding what features of the application should be considered critical and what features will be less critical.

In the above figure the application development has been condensed into a single task. This is to save space on the plan and to help to keep development simple and on track. The application development will be broken down further as follows:

3.9.1. Application Development Plan

Task Name – *(Component Name)*	Description	Estimated Duration
Workspace Setup	This includes setting up the server, downloading and installing all the required packages and setting up version control	2 Days
General Layout Construction *(codex-app)*	Creating the structure of the site. Creating the skeleton of each of the components and connecting those components together.	2 Days
Resolving Initial Dependency Issues	With so many packages working together there is a high risk some packages may conflict with one another.	1.5 Days
Implementing O-Auth *(codex-login)*	Implement the ability for users to log in with Google O-Auth	1 Day
Implement Polymer Real Time Attribute Binding	For the site to work correctly it is important that all the components can access the data they require. This task involves ensuring that all components have access to the data/objects they require.	1.5 Days
Implement Basic Post Creation *(codex-posts)*	Implement the basic ability for users to create posts (does not involve post tagging)	1.5 Days
Implement the Basic Post Feed *(codex-feed)*	Create a page where users can view all the created posts in one feed	1 Day
Implement Post Voting	Creating the functions and procedures that allow users to vote on the various attributes of a post.	3 Days
Implement Bounty Posting *(codex-bounty)*	Give users the ability to create bounties, upvote other bounties, view bounties and mark their own bounty as fulfilled.	2 Days
Implement Post Search *(codex-search)*	Implement the ability for users to search for specific posts based on: author name, post title, post language and then sort results based on the posts attributes	3 Days
Implement Basic Profile Screen *(codex-profile)*	Add a page where the user can view their posts	0.5 days
Implement a 404 Page *(codex-404)*	Add a page that will display if the user enters an invalid address or attempts to access resources they do not have privilege for.	1 Day
Create Home Page with News Feed and general site information *(codex-home)*	Add a news feed to the homepage to keep testers up to date with the progress on the application.	1 Day
Implement View Post Page *(codex-view)*	Add a view details page for posts. Users will be able to eventually view tagged posts and comments on this page	1.5 Days
Implement Post Tagging *(codex-post) (codex-view)*	Add the ability for users to tag posts in their post and then to view those tagged posts on the more detail screen	3 Days
Implement Offline Caching *(codex-saved)*	Add the ability for users to save posts from the view post screen to an Offline cache so they may view posts offline.	1.5 Days

Add Styling to the Navigation Bar *(codex-app)*	Add some additional styling and colour to the navigation bar to make it more appealing	1.5 Days
Add Styling to the Entire Site	Add some additional styling and layout features to the entire site to make it more appealing.	4 Days
Implement Real Time Post Commenting *(Codex-view)*	Create a discussion section on the view post page where users can create and view other people's comments	2 Days
	Total Days:	48

Table 3: Application Development Plan

With the above plan that means development will take roughly 48 days. Within the allotted time on the project plan being 59 days. The stated days on the application development do not equate 1:1 to actual days and instead refer to average work days so the required time to develop the application may be considerably less. With that said problems can arise and that is why ample slack time has been provided just in case.

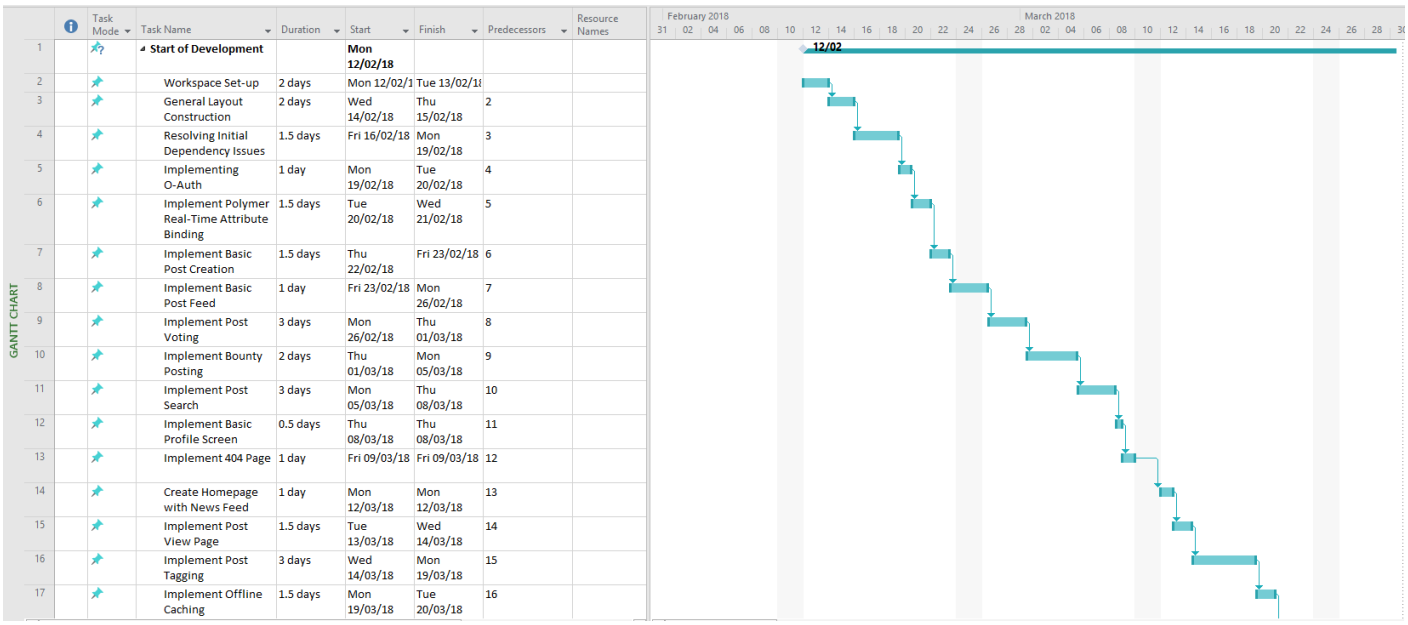

Figure 14: App Development Plan Part 1

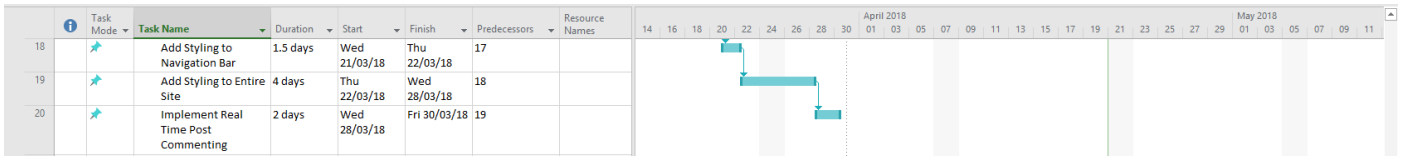

Figure 15: App Development Plan Part 2

The above application development also outlines the minimum viable product. All the above features and tasks must be complete by the deadline of the project to achieve the minimum functional product. This is a concept commonly used in industry where optional features are added but the developers and client decide on what features are critical to the application. The above table outlines those critical features without them the application cannot be declared finished. That said if develop finishes earlier than expected some optional features could be added.

3.9.2. Optional Application Features

Feature Name	Description	Estimated Duration
Additional O-Auth Options	Add additional o-auth options such as Facebook, Twitter, Email and Password etc	2 Days
Author Following and Advanced Feed Control	Give the users the option to customise what appears on their feed through following other users and defining preferences	3 Days
Detailed Profile Page	Add more information and statistics to the profile page about the user's activities	2 Days
View Other Users Profile Pages	Give the ability for users to view each other's profile pages if the user has enabled it in their privacy settings	2.5 Days
Settings Page	Create a settings page where the user can adjust their experience with the application such as enabled accessibility options or changing the applications theme	4 Days.

Table 4: Optional Application Features

4. Implementation

4.1. Overview

This section will cover the implementation of the application. As mentioned in chapter 3.9 development will follow a mostly waterfall method. Development will start with the set-up and installation of all the required dependencies and requirements. Installation involves setting up the hosting and back end server so that requests can be processed and sent onto the database were new data can be created and pulled.

This set up also involves integrating version control. Version control is a very important aspect of any large development project. In this case the project will be utilising GitHub to manage all the version of the application. As Codex is developed those developments will be pushed to GitHub [Github 2018] where those changes can be managed and integrated into a public build. On top of that GitHub can be used to revert any changes that have unforeseen negative consequences. Version control is an incredibly useful tool that allows the easy management of large project through the logging of development commitments. Hence why it will be heavily utilised in this project.

4.2. Implementing the Voting System

A system which contained a surprising amount of complexity was the voting system. The difficulty arose when it came to ensure that users could only vote once. Creating a system that allowed users to vote many times was very simple but allowing the user to only vote once and to enforce that became tricky.

Any attempts to stop the user voting multiple times on the front end would be useless as it would be very easy for a malicious user to circumvent those barriers. So, an approach that validated the user's vote on the backend would be required. In the end a log approach was implemented.

Viewing figure *bleh* you can see that the post object contains several lists called <attributename>log. These are lists of user ID's who have voted on a post. When a user votes their unique ID is added to that list. That paired with front end preventions can prevent users from maliciously voting multiple times. On the front-end a check is passed that disables the functionality of the up-vote button after a single press. This could be circumvented by manually editing the page but that would not allow the user to make multiple votes due to the server-side check. If the user re-enabled the button and submitted another up vote that vote would then be discarded by the server when it checked the vote log and found that user ID is already attached to that post.

The downside of this approach is that it increases the complexity of the post object and therefore requires more memory to store each post object. It also means that at larger sizes voting may take longer as the system checks for a user ID within a huge list of user IDs. If there was more development time was available this system could be improved by creating a user object and attaching the post ID to a list within the user object. That means when making a upvote the server would check a shorter list on the user to make sure they weren't voting multiple times.

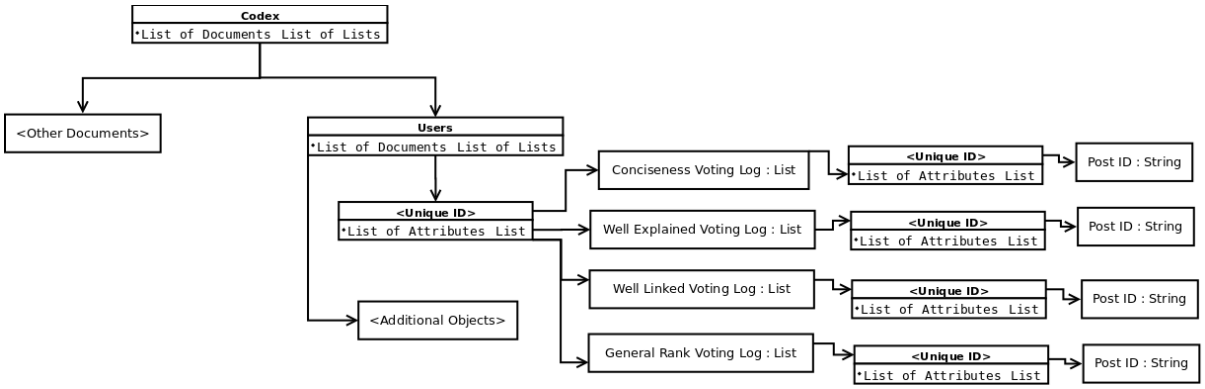

Figure 16: Database Design - New Child Object

The above figure is the proposal for the new user object. If there was more available development time this user object could be implemented and whenever they voted the post ID would be added to the users list. This would avoid the possible huge list of user IDs currently present in the current application. These huge lists could be a potential performance bottleneck so if development were to proceed this implementation should be taken forward.

4.3. Implementing the Post Tagging System

Another tricky aspect is that of the post tagging system. Due to the current layout of the Codex application posts do not have unique address. This was done to close certain avenues of attack. With open external links it opens the possibility of malicious users attempting to provide invalid address that have the possibility to send corrupted data to the server. Therefore, to increase security individual posts were not given unique links.

Unfortunately, that means that tagging posts becomes somewhat difficult as users cannot simply copy and paste a link into their posts. In future development it might be worth using a hashed version of the post's unique ID as a HTML GET request to provide a unique link to the post and provide that functionality. However, as currently implemented a simplified version of the search component is included in the post creation suite. Here users can search for posts and instead of viewing them they can tag them. This will then connect the tagged posts ID to the new posts tagged post list. This is then stored in a string array and attached to the new object.

```
http://www.codexapp.reivew/home?post=abcdefg123
```

If unique post links were to be implemented, they would be implemented through get requests as you can see above. This does however open the site up to various attack such as editing the data on the user end. This could possibly lead to malicious users attempting to query data they are not meant too. That would be why the site would have to carefully vet the hashed post ID before processing it. If the key was invalid the user would be returned to the home page and given an error.

It will be a point of note in the final audience survey to see how users react to the post tagging system. There is a chance they would prefer to just include links in there post in which case the above method would have to be used. However, they may also like the post tagging system as it does all users to search for posts while remaining in the post creation suite. So, a combination of the two approaches might be best for the application. There is no way of concluding on this issue until the final audience survey is completed.

4.4. Implementing the Search and Sort System

As mentioned throughout the Market Research *(chapter 2.1)* and the audience research *(chapter 2.2)* the ability to quickly and easily search for specific content is incredibly important to the average user. That is why in Codex it must be implanted thoughtfully and be extensive in its usage. Plenty of tools and options must be available to the user to allow them to search through a possibly huge range of content to find something specific to their needs.

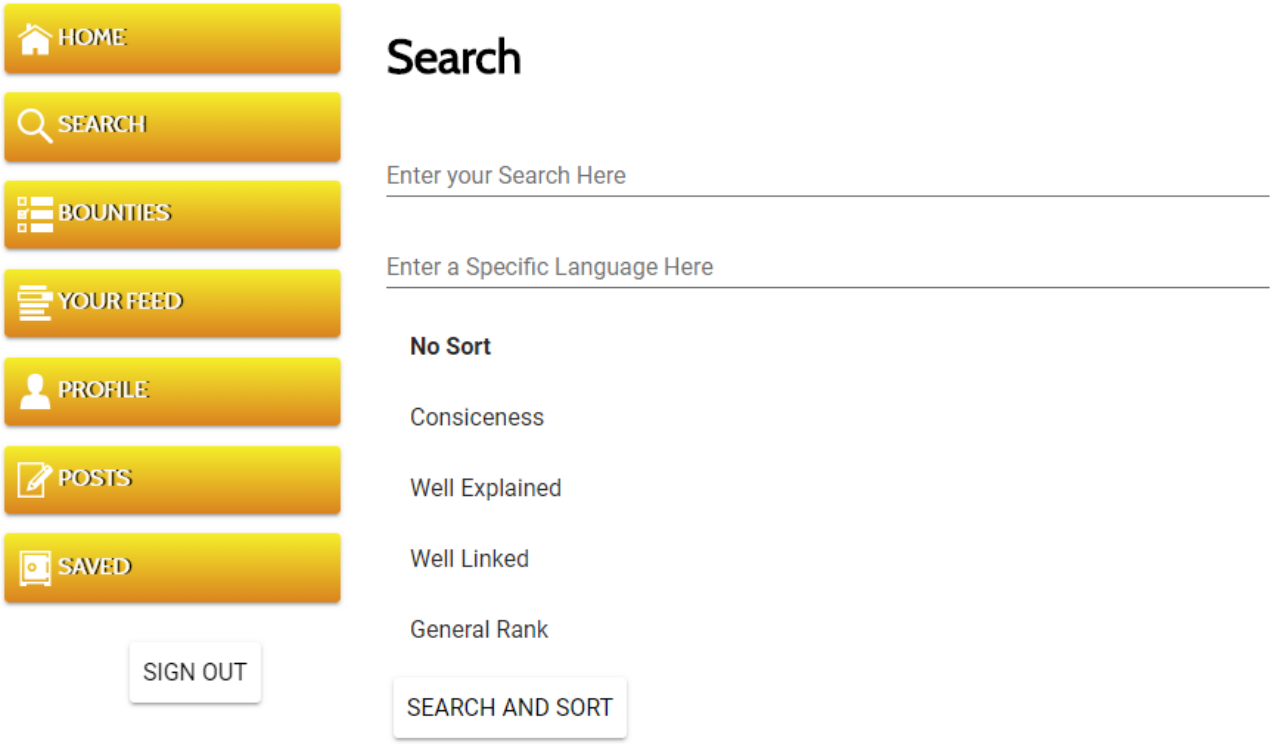

Figure 17: Current Design - Search Page

The above figure shows the final design of the search page. Here the user has multiple options of how to search and how to order the data they receive. Linking to Stack Overflow which provided to be the most popular tool in the Audience Research *(chapter 2.2)* having a search function which queries multiple properties is more popular. That could be down to the fact it makes it easier for a user to find specific content which they may not fully know how to search for.

In the of the codex search bar it will look for partially linked tags to post titles. It will also at the same time look for author names. On top of that it also makes the search case insensitive. One issue that can appear is content not being displayed because of formatting. Search results can be greatly increased by making it as simple as possible for users to find content via known tags. The language is a separate field as a syntax or semantic may be present in multiple languages. However, users may only care about that syntactical feature in a single language. This feature gives that user to search for a topic then limit it down to a single language. Once again giving the user more choice and power when searching for specific posts.

Finally, we have the sort feature. This is a new feature to Codex that hasn't been directly implemented in any other major applications currently available. Other sites like Reddit have a similar feature when users can sort based on "controversial" or "hot". These are not perfectly defined elements though with their underlying algorithms being hidden from the user. The goal with Codex though is to make that functionality clear to the user. If a user sorts based on conciseness they can see the conciseness rank and easily understand why one post was sorted other another post. This function the same with the other criteria too.

4.5. Implementing the Real Time Data Aspect.

These huge crowd sourced sites when they finally develop a community are constantly being updated with new content every second. YouTube for example is estimated to be updated with over 300 hours of video every minute *[YouTube-Press 2018]*. While it is highly unlikely Codex will receive that level of traffic it is an example of how quickly content can be produced. So, with that in mind the application needs to be able to serve the user new content as and when it is created.

```
<firebase-query
  id="query"
  path="/posts"
  data="{{results}}">
</firebase-query
```

Figure 18: Code Snippet - Query

This Real Time Data is enabled through the Polyfire library. Within that library there is a component called firebase-query. The Query component is its own self-contained collection of HTML5 and JavaScript will directly communicate with the Firebase database. Starting at the top attribute of the element the ID tag works the same way as a normal HTML ID attribute. It allows for easily access by JavaScript. This in turn allows the easy alteration of the path and data attributes. Speaking of which, the path attribute refers to the location within the Real Time Database. Finally, the data attribute is the location of any pulled results. This is then exposed to the Polymer databinding using the double curled brackets. Meaning when the data is updated the Polymer engine will be notified allowing other elements access to that data.

The real power behind this system is that all this complexity is given at a great level of abstraction. That means it does not take very long to get complex real time functionality up and working. Using this element alongside some simple JavaScript and Polymer data bindings. The application now can query the database and self-update when it detects a chance.

Once again, this was implemented to make a truly responsive system. As users create posts that content will be immediately pushed to other users creating the feeling that the site is constantly

changing. This does however have a minor setback that the site must regularly check for updates to the database. That means the application may use slightly more bandwidth than other apps. That is offset though by the Realtime functionality and that most browsers will auto limit traffic if the user is on a limited bandwidth.

5. Testing

5.1. Black Box Testing

5.1.1. Method

Black box testing refers to a specific method of verification where the tester only has access to the end user product. From that point onwards, the tester will attempt to verify the functionality of the application with reference to the original functional requirements. For Codex the Black Box testing will consist of a range of test that verify each of the functional requirements are met.

The below test plan is broken into several sub sections with each sub section covering the tests related to a different functional requirement. On top of that each section will include three different types of test. These types being: valid, boundary and erroneous. Valid tests aim to verify the correct and expected functionality of the application by providing an expected, correct input. Boundary tests aim to verify the application does produce anomalous behaviours on the edges of valid inputs. For example, this could be providing 101 to a field that only expects numbers between 0 and 100. Finally, there is erroneous tests which verify the site does not provide anomalous behaviour when given a clearly false or incorrect input. These tests also aim to ensure the site is not open to common attacks such as SQL injections.

Non-functional requirements cannot be fully tested using this method because non-functional requirement express concepts that the system must have. These concepts can only be verified through a combination of testing and audience evaluation. These additional requirements will be verified in the final audience survey found in chapter 6.1

All the functional requirements can be found in chapter 2.2

Tests with Green shading are valid tests

Tests with Yellow shading are boundary tests

Tests with Red shading are erroneous tests

Test Results with Green shading and passed tests

Test Results with Yellow shading are not invalid however the resulting behaviour could be improved

Test Results with Red shading are failed tests and represent bugs in the system.

5.1.2. Report

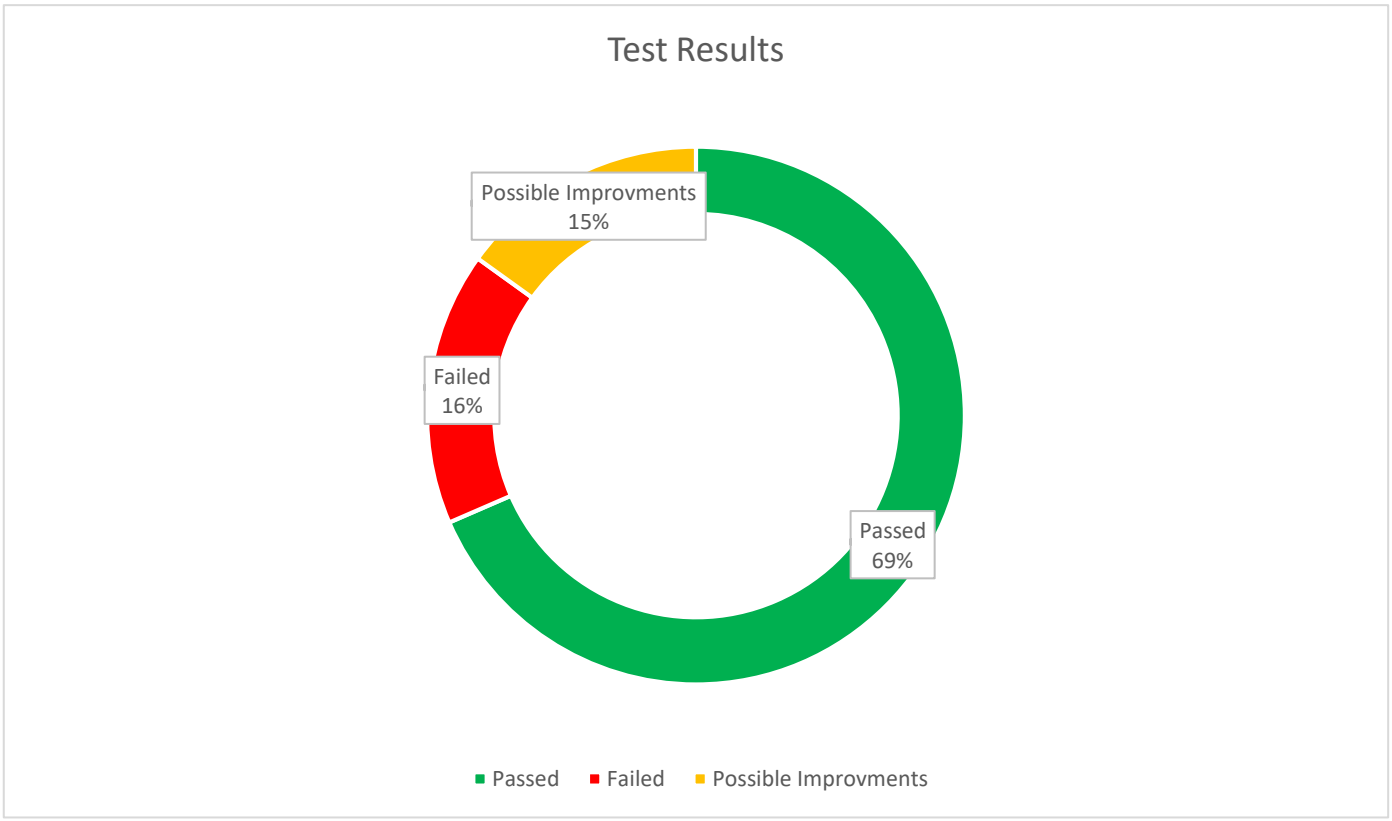

Figure 19: Test Report Graph

{For full report see Appendix C}

5.2. Backend Server Testing

5.2.1. Method

Another important aspect to ensure that the system is performing to specification and the functional requirements is ensuring that when data is saved or queried it is stored correctly on the server side. This can be done by a simple post and check system. A specific action will be carried out on the user end then using the firebase console that data can be reviewed on the server side to ensure it has been stored correctly.

This type of error can also be caught on the user end when reviewing created data. If an error was created when posting that data, it would show upon the user end as well. However, there may be some circumstances where discovering what has happened from just a Black Box approach might be difficult. So, in the case of being thorough all interactions with the database will be reviewed from the backend console as well.

On top of this bandwidth usage will also be monitored when performing certain tasks. This is to ensure there is no aspect of the application that is over using resources. Over usage can be viewed using the Firebase hosting toolkit which provides helpful graphs depicting traffic at any time.

Once again when a front-end application heavily relies on a database and hosting aspect it must be tested on all fronts. This is to ensure the application will run smoothly in a live public environment. Issues such as overuse of bandwidth which may be caused by an inefficient function sending too much data could be very costly for a company when the user base starts to grow. That means it would be best to catch it in the testing phase and fix it before it becomes s costly problem.

5.2.2. Database Usage

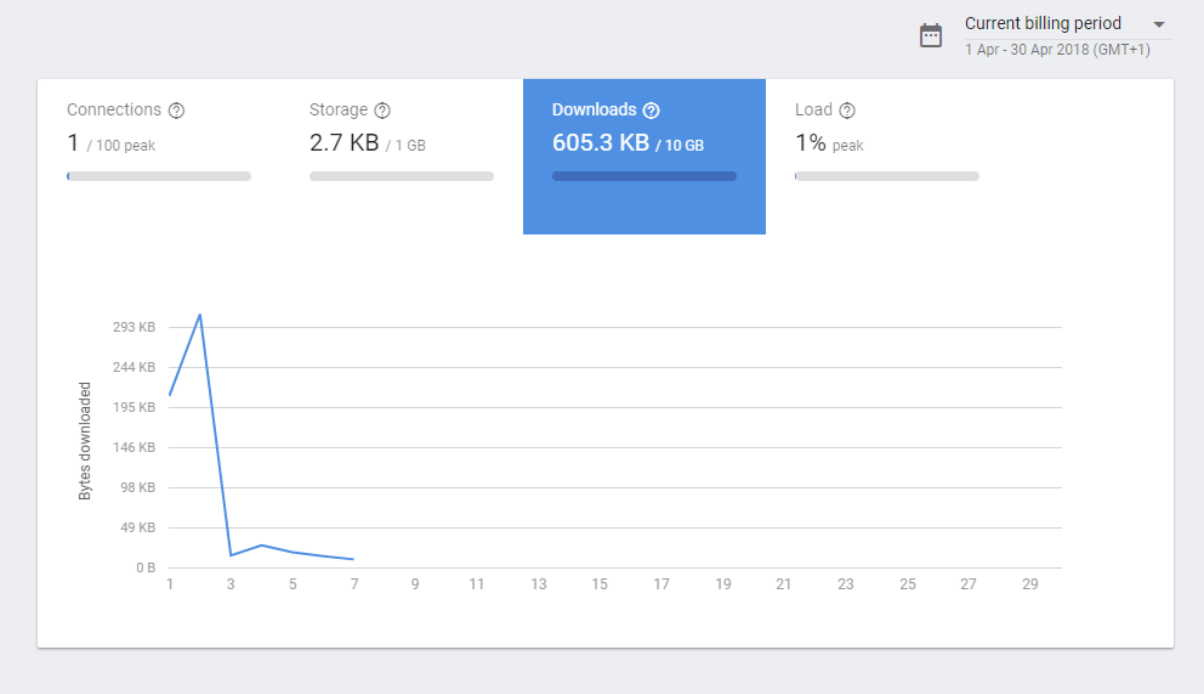

Figure 20: Database Initial Usage Graph

Figure *bleh* shows the database usage of only a single user over the course of 7 days. As you can see there is a huge initial spike that relates to the amount of content currently available on the site. During this time there was a larger amount of content available during the 1st to the 3rd and then on the 3rd a large amount of that data was deleted. This could be an issue when user numbers increase as it would be feasible that the application is pulling too much data. As part of development it would be very important to go back to the application and find areas that could be improved. For example, it might be of worth to implement better caching systems so that the database is queried as much therefore reducing traffic.

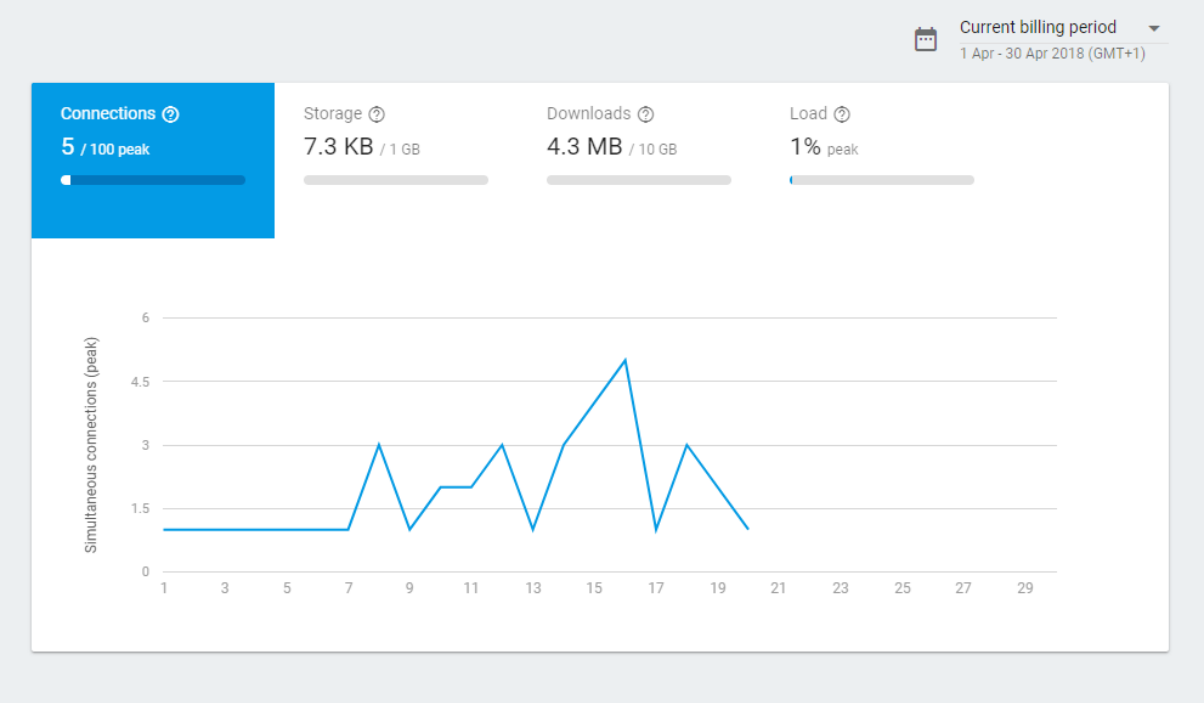

Figure 21: Database Under Use

This graph shows that even with minimal user increase the download size has increased by over a magnitude of seven even though the database has only had approximately 7 more users. While this is not system critical this shows that some aspect of the system is sending more data than truly required and could be optimised. Reducing download rates allows the server to run better by conserving bandwidth. This in turn allows more users to utilise the site at once which for the application to be successful a great deal of users must be on the site at once. If the site were scaled up to a larger user base this inefficient bandwidth usage would have to be optimised.

5.2.3. Database Verification

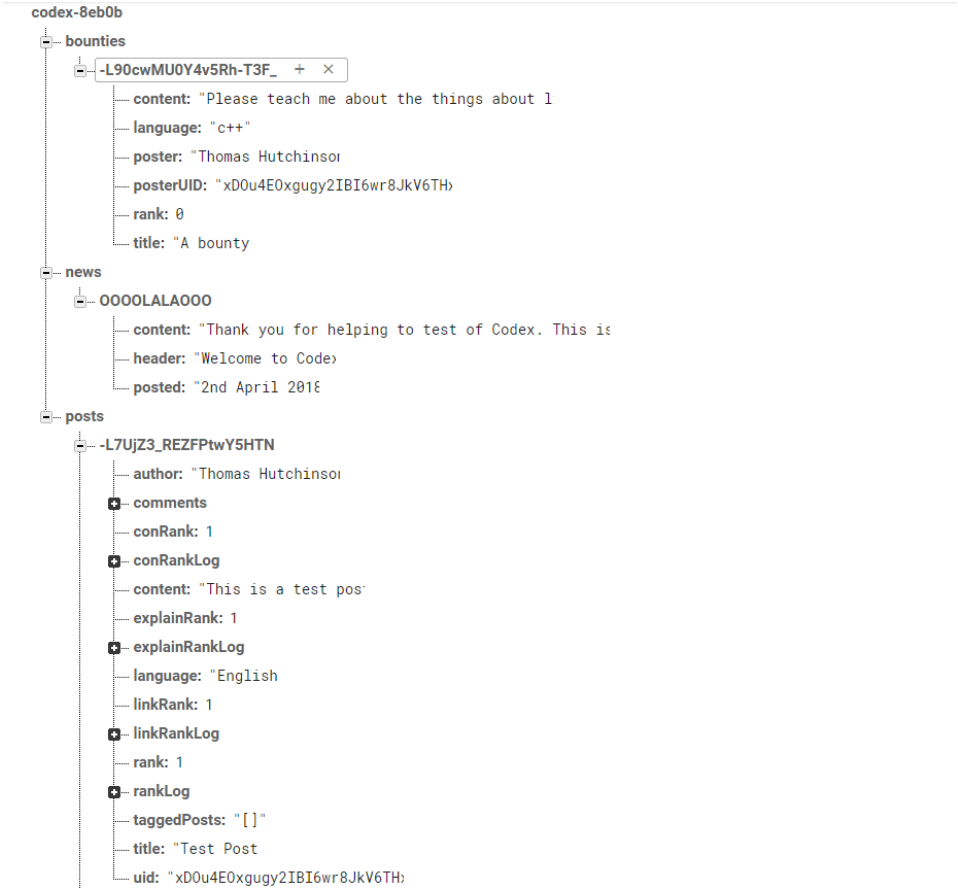

Figure 22: Database View

The above figure *bleh* shows a database after every possible user interaction has been made. In this case a news post has been made. Users have created bounties; multiple posts have been made. Comments have been made on those posts and users have voted on said post. Everything has been stored correctly as according to the database design diagram in chapter 3.6.

This proves that there are no data anomalies being created under normal usage. That said the full Blackbox testing strategy still needs to be carried out to ensure no anomalous data can be created under extreme circumstances.

Following the database design greatly helped in ensuring no excess data was used and that the database itself was simple understand and maintain. Working with new database paradigms proved to be difficult as it required a major change in development strategies compared to classic SQL designs. However, relying on some of the rules SQL databases hold really helped in producing an SQL-less database design that handled the application aptly. For example, creating tables to handle the relationship between objects could be converted into the self-contained lists you see such as comments. By combining my understanding of SQL and SQL-less databases a design that takes advantages of the design features of both could be created. This then has the advantage of providing a faster query time while still providing the level of detail required from the Codex application.

5.3. Miscellaneous Testing

5.3.1. System Stress Test

The Web Application itself is being hosted upon Google's Firebase [Google Firebase 2018] Hosting. This hosting comes with its own limitations on bandwidth usage. These limitations can be found at Firebase's pricing scheme page of which Codex is using the free tier called spark *[Google-firebase pricing 2018].* Spark provides various limitations such as only 10 gigabytes transferred every month. While that may sound like a large amount of available data it can quickly be used up by a large user base. Thankfully, Google makes it very easy to scale up by upgrading to the higher tiers.

Since Codex is currently running on the free plan it would be unwise to attempt major stress tests as this would use up large amounts of this allotted data. While, it would be possible to move Codex to other platforms this would use up precious development time making it not worthwhile. Hence why, not extensive stress tests will be completed on Codex at this point.

If the application where hosting on a private server then the stress test could be completed by simulating or generating a large user input. This could be done by arranging a large group of users to simultaneously use the site at a specified time. Stress testing can be a useful tool in finding possible holes in development that can be patched up. During this test developers would monitor the statistics of the server such as bandwidth usage and overall performance of the server machine. If it was found that the server was lacking, additional resources could be dedicated to the project such as renting additional server machines to handle the increased load.

Overburdened servers can lead to increased load times for users. High load times then lead to users becoming frustrated and possibly leaving the site if things carry on. Disgruntled users go against the sites main objective of retaining those users, so it would be of vital important to ensure that application hosting can handle the predicted user growth.

5.3.2. Storage Tests

Another aspect that can easily sneak up and become a huge problem is running out of storage on server machines due to the database growing too quickly. Thankfully, Codex does not have to worry about file uploads as it pulls all its user avatars (the only user-submitted multimedia used) from external services such as Google O-Auth[Google O-Auth 2017]. Figure *bleh* shows the storage usage of a few posts. As you can see it minimal which is a desired outcome. Low storage usage is important as it helps to save costs on hosting due to the company hosting not needing to buy large amounts of storage.

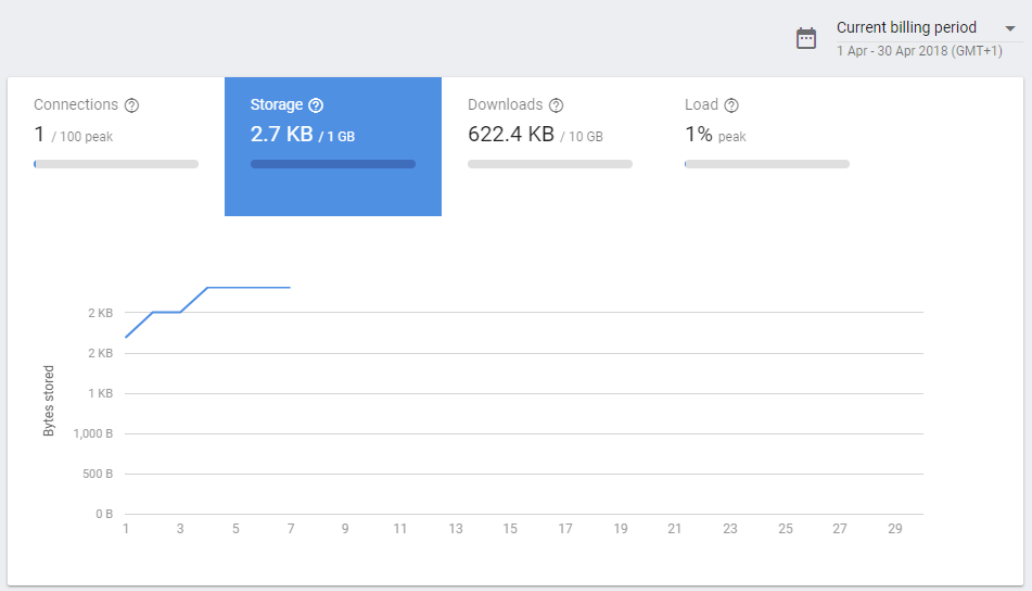

Figure 23: Storage Usage Graph

When it comes to user submitted content platforms things can quickly scale up though. The below figure *bleh* shows how much storage is used after a few more posts have been created by a wider range of users. It's important to remember that as a service becomes more popular it requires additional resources to ensure it's up keep. That is why is important to regularly monitor a live service to ensure it is running smoothly.

5.3.3. Compatibility Testing

All the tests made on in the Black Box testing found in chapter 5.1 were made on a Linux version of Google Chrome version 65.0.3325.181 (64-bit). Of course, this does not encompass all possible users but due to time constraints performing full tests on a range of different platforms is not possible. That is why minor run throughs of the application will have performed on different platforms. This will aim to test the core functionality of the site.

These functional run through will take the following plan: login to the site, create a post, view the feed, create a bounty, mark the bounty as complete, up vote a bounty, down vote a bounty, up vote a post, down vote a post, view a post, make a comment, view other comments, vote on all attributes of a post, save a post locally, view that post without internet connection then log out.

This section will detail the success of the test on a range of platforms.

Platform Details	Results of Test
Windows 10 – Google Chrome Version 65.0.3325.181 (64 Bit)	All test areas succeeded
Android Version 7.0 – Google Chrome Version 65.0.3325.109 (Mobile Version)	All tests succeeded though users are unable to log out due to sign-out button being pushed off page. Also, on the left hand-side of the screen a thin white bar is present.
Android Version 7.0 – Google Chrome Version 65.0.3325.109 (Mobile Version) *(Retest)*	All tests succeeded. Users with smaller screens are now able to log out thanks to the navigation bar being scrollable now. The white bar is also now removed.

Table 5: Compatibility Testing

5.4. Bug Log

Bug #	Test #	Description	Follow Up Test #
1	1-3	User can create a post without a language	1-6
2	4-6	Back to front sorting on attribute sort. Posts with highest score appear at the bottom with lowest at the top	4-13
3	4-7	Back to front sorting on attribute sort. Posts with highest score appear at the bottom with lowest at the top	4-13
4	4-8	Back to front sorting on attribute sort. Posts with highest score appear at the bottom with lowest at the top	4-13
5	4-9	Back to front sorting on attribute sort. Posts with highest score appear at the bottom with lowest at the top	4-13
6	5-3	User can create a bounty without any title	5-16
7	5-4	User can create a bounty without any language	5-16
8	5-6	User can create a bounty without any content	5-16
9	5-7	User can create a bounty without any data at all	5-16
10	5-15	Bounties are not sorted at all	5-17
11	8-3	User can create empty comments	8-5
12	{User found}	Users can mark other user's bounties as complete. *(Test did not catch all possible conditions)*	5-18
13	{User found}	Other user's posts appear on the created posts list in the post page. Also means that users can delete one another's posts.	6-5
13	{User found}	Can't view the news or search for content when not logged in	M-3
14	Chapter 5.3.3	A white bar appears on the left-hand side of the screen on mobile displays or when the screen is scaled down.	M-4
15	Chapter 5.3.3	The side navigation bar is non-scrollable meaning that when a user logs in they cannot log out because there is too many options and the sign out button is pushed off screen	M-5

Table 6: Bug Log

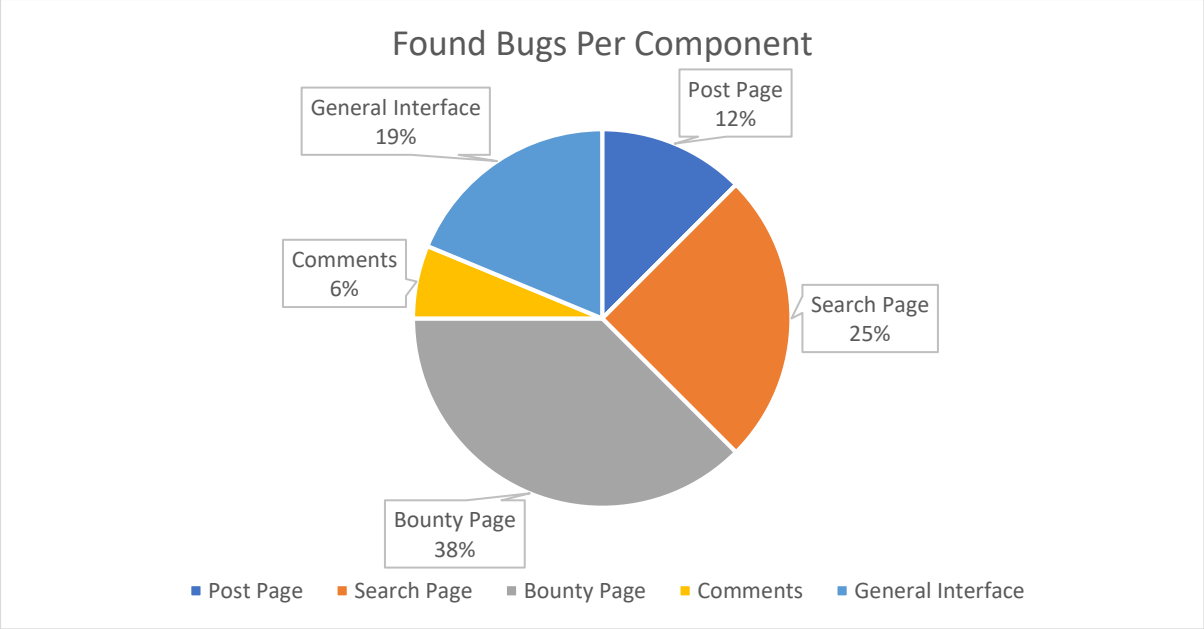

Figure 24: Found Bugs Per Component

6. Evaluation

6.1. Final Audience Survey

6.1.1. Reasoning

To truly test the successfulness of the application regarding the original specification and the original functional requirements the end users must be involved. It is possible to test the presence of features such as “is a user able to create a post?”. That aspect can be quantified however elements such as “is the system easy to use?” and “does this system enable me to learn and improve myself” can only be answered in a qualitative domain. Meaning, the user base must be surveyed again to gain greater insight into the successes and failures of the Codex System.

6.1.2. Method

In a similar fashion to the background research survey the questions will hosted upon Google’s Form service. This service was chosen as it provides and adequate selection of question types and allows for in-depth control of user progression. Meaning that if the user answers a question in a certain manner they can be taken to other sections that allow them to expand upon that answer.

This survey will be more open than the previous. Meaning that the users will have more free answer questions where they can enter as much, or as little, as they want. Free questions can be a double-edged sword however as it can lead to in-depth responses that provide an insight into the user perception of the system. Then on the other hand, some users may enter minimal responses that provide very little utility. I believe that the latter is a minimal risk though as the survey will be targeted directly at the students who answered the first survey. This group is from a highly educated, computing related background and if they have feedback it should be given if the first survey is a standard.

Questions will mainly focus on the various functional requirements and how they have been met. Users will be prompted to rate various features such as the ability to create posts. They will also be queried about their thoughts on the more experimental systems such as the attribute rating. In that space they will also be given the opportunity to give input in how the system could be improved. That information can then be used to deduct what direction Codex needs to go in and how users best learn new content.

6.1.3. Findings

Findings can be found in appendix bleh

6.1.4. Summary

Thanks to the audience survey various immerging properties about the application and the market were possibly discovered. Due to the low user response these results cannot be taken as concreate evidence of any hypothesis or theory. With that said they can be used to infer and begin the construction of theories of the usefulness of the application.

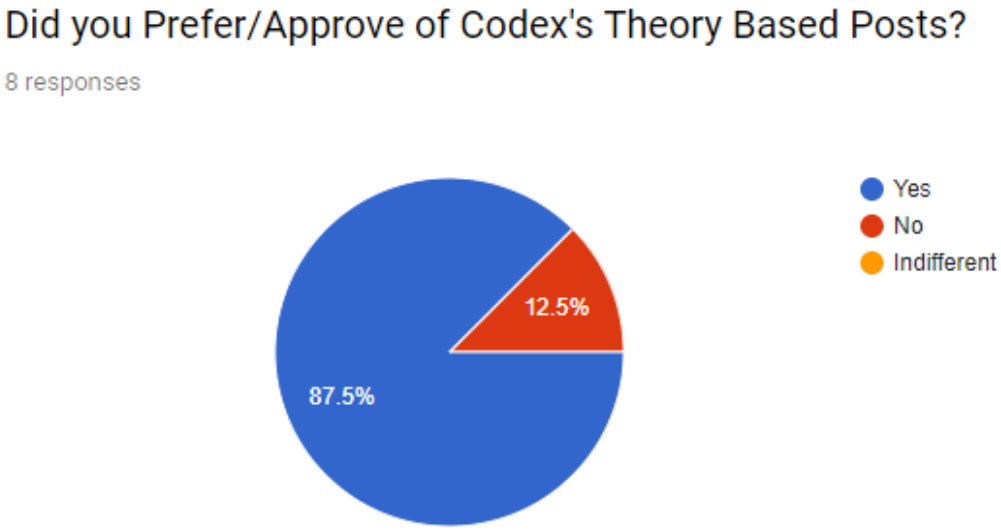

Figure 25: Final Survey - Theory

This question was asked alongside a preface describing the idea of a theory-based post in relation to the currently available posts on other market products such as Stack Overflow [Stack Overflow 2018]. Given the small response number the results of this question can be used as a possible gauge of how users think about theory-based posts. It should also be considered the fact that a primary demographic of the respondents was Full-Time Students. By the response for this question it could be theorised that users approve of posts containing additional information and background behind a specific topic. This would help achieve the aim of the Codex platform. They have responded well to the goal of helping users improve their development skills through the increase of semantical knowledge.

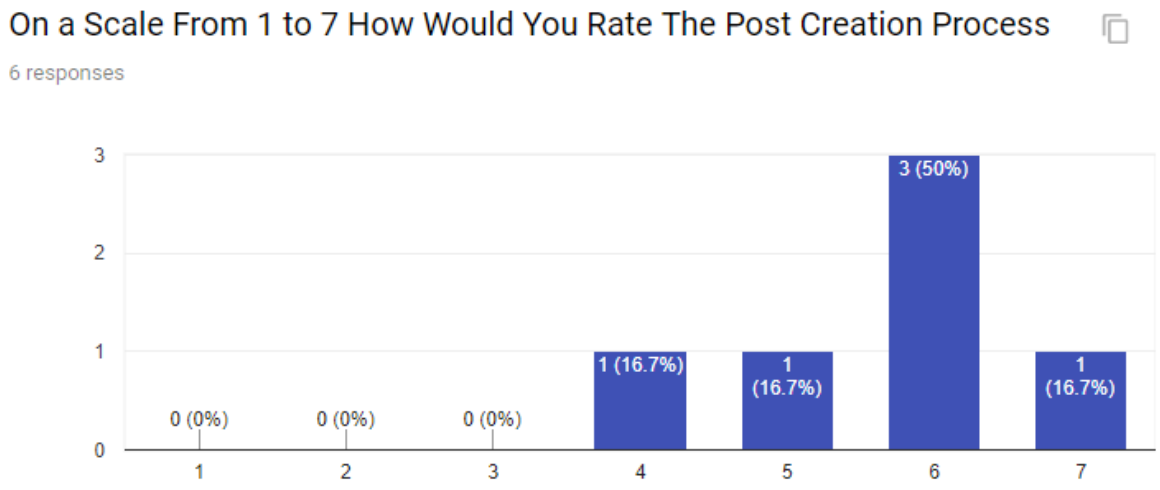

Figure 26: Final Survey - Post Creation

Overall, users responded positively to the post creation system. Some users who voted lower provided comments such as: "Kinda feel like a bit too much options available there." and "Improve the post tagging system. It can be difficult to understand and use.". This is a good sign that the primary aim of Codex has been met in a manner. One of the goals of Codex is to facilitate the learning of new programming and scripting languages. Due to the ever-changing nature of the development industry new languages are constantly being created. That means that new content would have to be constantly created to keep up with that. So that is why the post creation system receiving positive feedback is good because it can be a sign that users are more likely to use it to create new content.

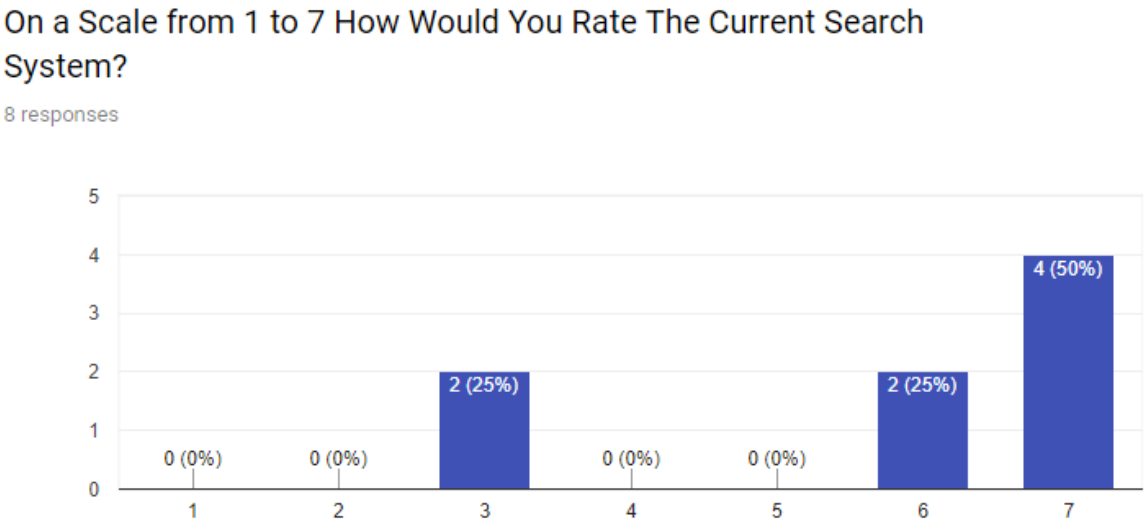

Figure 27: Final User Survey - Search System

On the other hand, the post search system divided the user base in a greater fashion. As you can see several of the users gave a high vote while a smaller group gave a lower vote. Unfortunately, no comments were present explaining this lower rank, so it is hard to draw any formal conclusions to why this pattern has occurred. This could be down to the low respondent base, but it could also be down to poor question design.

At the end of the search section there is a simple “Any More Comments” question referring to the search feature. Upon reflection this is a very poor choice of question. It is far too open which may result in many users simply not giving any comment. The question could be greatly improved by asking the user why they gave such a low score. Or by asking specific questions about each of the components of the search engine.

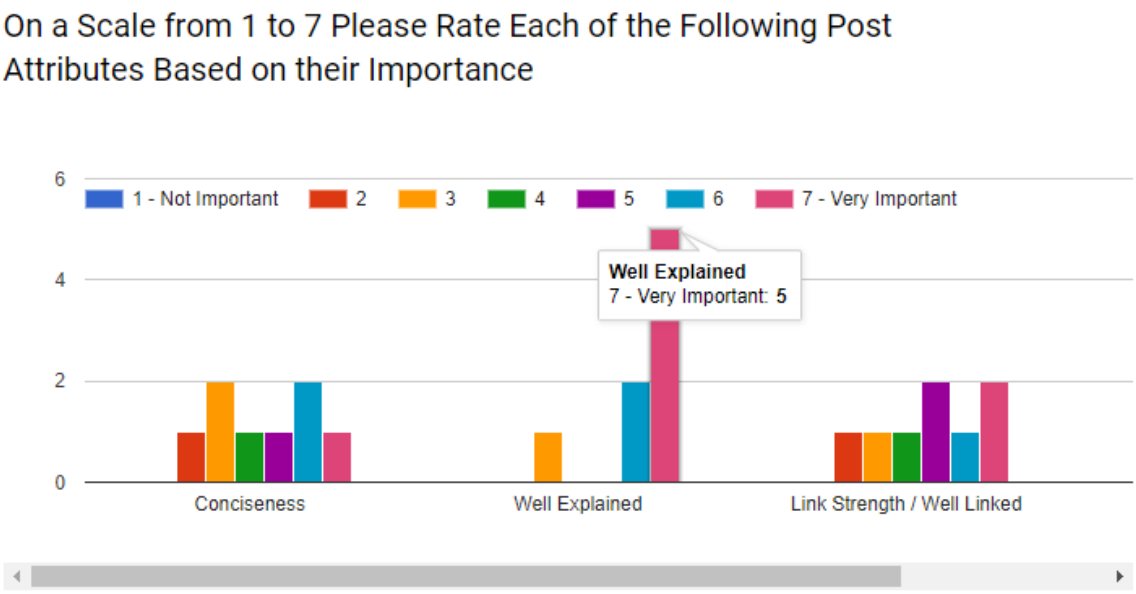

Figure 28: Final User Survey - Attribute Strength

Following on, we have the general user perception of the various attributes a post can embody. By far it is clear that from these results that the users favour a well explained post. Meanwhile, the other attributes have a more even distribution of votes. This could be a sign that the other attributes represent user preference with certain users like more attributes than others. It is clear though that the well explained attribute stands out from the rest. As mentioned, it could be a sign that all users generally prefer a post to embody this attribute.

This then links back to the idea of users preferring posts that contain additional content and take the time to thoroughly demonstrate and explain a concept. From these user responses it could be deduced that users prefer posts that contain a well explained description of the concept. They also like the idea of a system that emphases theory over simple examples. Linking this back to the aim of helping developers improve their skills through learning more theory and background semantics Codex has been a success in this respect.

Figure 29: Final Survey - Use Codex?

Finally, there is the question about whether users would use Codex once it's development concluded. While responses to all the other sub-systems have been *mostly* positive these responses show the protentional downside to Codex. A great deal of respondents referenced the need for an active community who regularly creates high quality posts. It could be said that no matter how good the systems are if it does not foster a strong community then the tool will quickly fail. This again links into the academic research on crowdsourcing *(Chapter 2.4)*. "Communities are built around a common goal" *(Zwass 2010)* and for Codex to succeed in its aim a strong community must be developed.

6.2. Focus Group Testing

6.2.1. Reasoning

To be as thorough as possible multiple avenues of user base testing will be explored. With the background research a simple survey sufficed as it gauged user opinion to an adequate level. However, the development and successfulness regarding the original specification is much more complex. Therefore, having a more rounded and comprehensive review of the application from multiple fronts will help in gauging any trends and overall views in the user base. Focus group testing is simple another method for gauging user opinion that differs from the standard survey. This is done to get different or more detailed responses from users. Those detailed responses can then be cross examined with the final audience survey and together they can be used to form a base argument for the overall successfulness of Codex.

6.2.2. Method

Due to time and resource constraints the focus group cannot be as thorough or as non-biased as preferred. A more optimal way to conduct a focus group would be to have an unaffiliated individual host the focus group. This keeps the questions unbiased and allows the attendees to fully express themselves without worry of upsetting the people directly involved with the project. However, due to resource constraints I will have to directly work with the focus group. This is not optimal due to the reasons, but the feedback gained will still be extremely useful to the overall project. Enabling a directed discussion of the application will allow in-depth exploration of specific elements and those responses can be used to gauge the overall success of Codex.

The target demographic of the focus group will be full-time university students aged 19-30. These will be students studying Computing Science or other related fields. Sessions will last for approximately 40 minutes and generally consist of 4 individuals. These focus groups will be repeated with different groups until an adequate response has been gathered that can be used to either prove the initial hypothesis or refute it.

Questions and discussions at the focus group will be as un-biased as possible. With that said on biases may seep into the discussion and that will be accounted for in the conclusion and focus group report. This data cannot be a perfect proof. It is instead another tool that combined with all the other feedback and responses can be used to get one step closer to proving or refuting the hypotheses.

Discussion will revolve around the various features of the site and how the users responded to them. Much like the final audience survey in chapter 6.1 this will focus around the functionality rather than the user experience. Meaning that discussion will be guided to how effective a learning tool Codex is rather than how it’s interface performs or what little features could be added to make the tool better.

6.2.3. Findings

{Find Focus Group Reports in Appendix F}

6.2.4. Summary

Focus group testing led to a few interesting properties of the Codex system being discovered. Primarily, it would appear from the focus group testing that the quality of the content has little effect on the user perception of system quality. Within certain thresholds. When the content provides working examples of solutions that appears to be adequate for users to integrate that system into their normal work process.

This then means that Codex adapting a more complex theory-based approach to tutorials may have minimal effect on retaining users. This is important to the hypothesis because the platform heavily relies on users consistently creating new content to keep users interested as also stated in the focus group report. If users aren't interested to the platform and return to other options, then Codex will slowly lose users due to out dated posts which then leads to the platform directly becoming less effective at teaching user's new syntax and semantics.

6.3. Project Requirements

The overall success of a project can be evaluated by the effectiveness in which it achieved its objectives or functional requirements. A project that creates a seemingly high-quality product but meets none of its requirements can be deemed as more of a failure than a lower quality product that meets some of its requirements. It is important to perform a final review of each of the functional and non-functional requirements to evaluate how they have been achieved and to what degree.

{For complete project requirement testing see chapter 5.1.2 for the full black box testing.}

{For Full Project Requirement Review Table See Appendix E}

7. Conclusion

7.1. Overview

The purpose of this section is to explore how effective the Codex platform was at completing the primary aim of:

To create a tool that can be used by both intermediate and expert level programmers to aid in their acquisition and learning of new [programming/scripting] languages syntax and semantics.

After that each objective will also be evaluated to see whether those were completed and to what extent while referring to all the content in the previous chapters. Finally, this section will explore how the findings from this study can be further used to discover more in the pedagogy of programming languages and how they can be applied in a real-world situation such as an application.

7.2. Aims and Objectives

1. **Research existing products currently available on the market and how my tool can offer a service that is different from competitors while still allowing its users to learn new syntax effectively.**

This objective has been accomplished to an acceptable standard as seen in chapter 2. The research performed in pursuit of this objective was extremely useful in directing the application to ensure it was suitable in performing it's aim of teaching developers' new syntax. The market research helped to direct Codex to ensure it learnt from the possible mistakes of other platforms while remaining unique and different. Without the research into site such as Stack Overflow [Stack Overflow 2018] there could have been a high chance that the development resulted in an application very similar to other currently available products. Codex was able to differ from the competition by implementing the attribute system which allowed users more control over the posts they viewed. The post tagging system also encouraged the community to work together and link to one another's work.

2. **Research the Pedagogy of [programming/scripting] languages to ensure the tool teaches its information in the most effective way possible.**

The academic and language research also greatly aided in the understanding of the elements that go into making a crowd sourced learning platform. Crowd sourcing is incredibly complex and very easy to implement in a manner that will result in unfavourable results. Building a community is very difficult but thanks to those areas of research the idea to create comments sections and the bounty board came forth. These features allow the community to work and talk together to help everyone work together towards the common aim of learning new syntax and semantics.

3. **Conduct a review into the user perception/usage of the currently available tools as well as my hypothetical tool.**

Then finally the audience research really helped in getting a general idea of what the users wanted to see from an application. It is very easy to forget primary demographics. So, getting their opinions on matters aided in ensuring the application was directly aimed at an audience. This led to the application being a website where the most users could gain access to the site. Once again for the application to be successful as many users as possible need to use the site and the audience survey helped in directing the application to a web platform.

4. **Develop said tool using the collected information and research to influence design decisions and the overall user experience.**

Developing an application can be extremely difficult, time consuming and require a lot of planning. All aspects of this have been documented in the previous chapters from the planning of the system architecture to the implementation of the specific components to the final testing of the finished system.

5. **Evaluate the performance of the tool in relation to the research from the previous objectives once completed by measuring various statistics such as user activity and user retention.**

Deciding how well the developed application meets the original aim was yet another difficult task. This process started with the construction of several functional and non-functional requirements detailed out in chapter 3.2 and 3.3 respectively. These requirements were features or systems that had to be implemented to ensure the basic version the aim could be accomplished. This was then followed up with the requirement report in chapter 6.3. Here each of the requirements were reviewed to see if they had been implemented. The requirements were determined to be important by the research phase.

Another manner of testing how effective the application met it the aim was through focus groups and another audience survey. These gauged user opinions of the final application in relation to the objectives, functional requirements and the original aim. The evaluation of the application could have been improved with further user testing to see how well the application improved user's skills. This would have been extremely difficult to normalise though as according to my academic research users learn in a range of different manners. So, differentiating between the user's growth while using the application and any growth from non-related activities would be extremely difficult and out of the scope of this research.

7.3. Performance of the Final Application

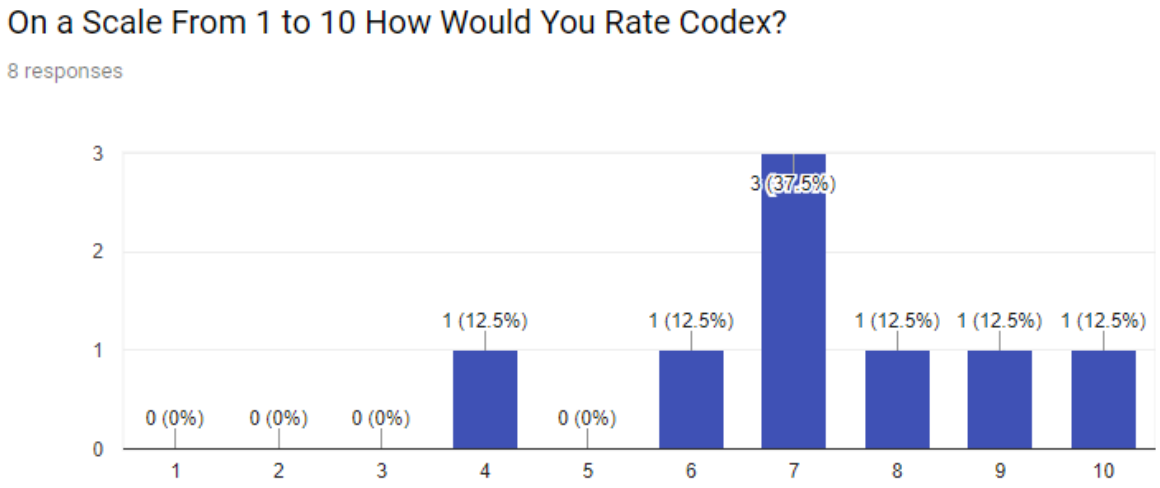

Figure 30: Overall Rating

Codex succeeded in many areas, but it also failed in several and there would be several concerns if the application was taken to market or deployed in a real environment. Relating back to the original aim of allowing users to learn new syntax and semantics through the Codex tool. Codex would have to adapt from it's current version to ensure it would be able to perform and continue performing over a prolonged period.

As detailed in the final audience survey in chapter 6.1 and the focus group testing in 6.2 users generally found the application to be interesting and would possibly use it once it had been finished. The focus on theory-based posts and the detailed post attribute system really performed with users. They linked the additional control over the content they searched for and found it different from other competitors on the market. So, in this manner Codex succeeded at its original aim. Users were able to connect to the site and learn new syntax and semantics. However, this positive praise did come with several caveats.

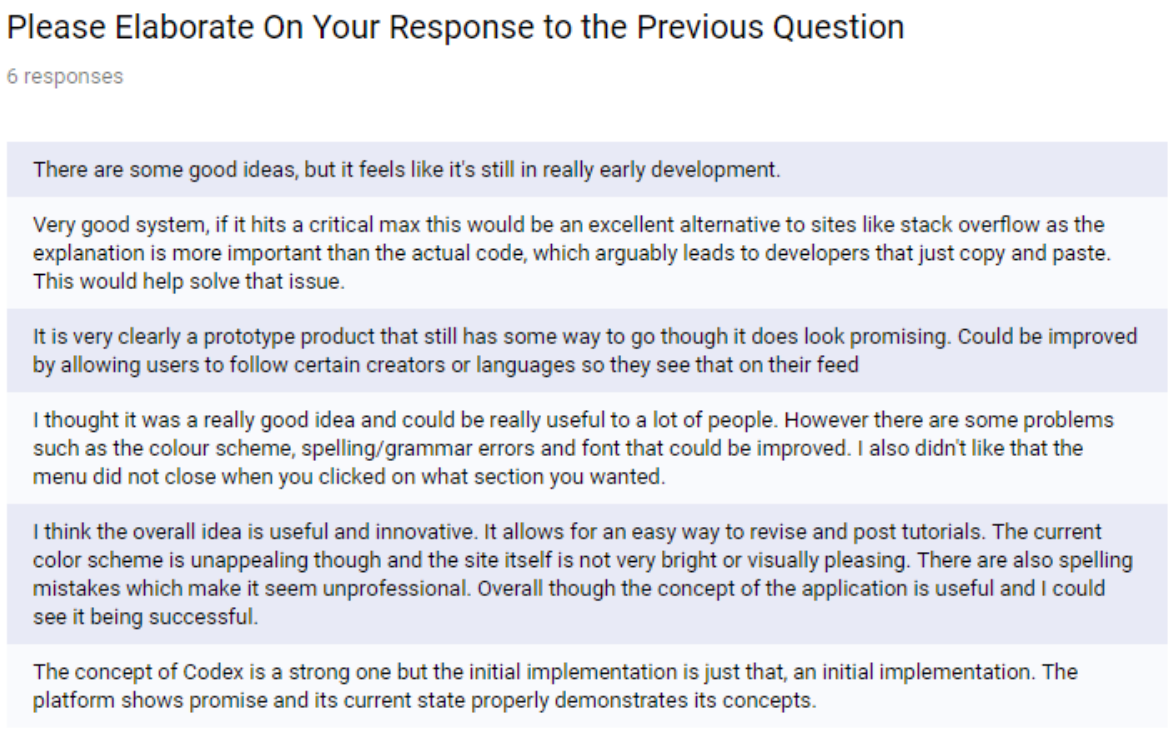

Figure 31: Final Question Elaboration

A great number of the user base who tested the application commented on how they would use the site but only with a larger community. This is one of the most intersecting aspects of the crowd sourcing content delivery platform. It is a double-edged sword as on one side it can help keep an application alive for many years with new content. Then on the other hand it can lead to the direct downfall of the application as it's clear here that the users would not visit the site without new content.

While Codex has met the aim of allowing users to learn new Syntax and Semantics it would have to be extremely careful in the management of the community. If new posts stopped being created the user base would very quickly leave the platform leaving it barren. That is why Codex cannot be truly successful in it's aim from this study alone.

To truly gauge the effectiveness of the developed application in terms of it's original aim the platform would have to be deployed to a real-world environment. The successfulness of Codex is completely dependent on it's community so when users are artificially generated as they were with this research that cannot act as a gauge. Codex has met all of its objectives but whether it is a success or failure can only be determined by publishing the platform.

7.4. Future Improvements

As mentioned in the previous section the application does several things “right”. Users enjoy the theory-based posts and attributes. But as mentioned the Crowd Sourcing could become a major issue that hinders Codex’s ability to meet its goal. One way the platform could be altered is by converting it into a creator focused platform were a specific number of individuals created content for the site. Then the audience would come for those specific creators. This is untested though and the closest market example is Dev.to [dev.to 2018]. This would be an untested venture, but it could lead to a more reliable platform.

Another further avenue for study would be the true effectiveness of the tool as a learning aid. Due to the limitations of the study full examinations of any possible performance increases were unable to be gathered. An individual with more resources and time could use this study as a stepping stone to see how effective this type of tool is on improving developer’s programming skills. Such as being able to apply specific syntaxes and features successfully *(Allan G. Bateson, Ralph A. Alexander, Martin D. Murphy 1987)*. This could be done through the examination of the developer’s skills through various tests such as debugging trials and coding examinations.

Finally, the tool itself could be greatly improved through several requested features.

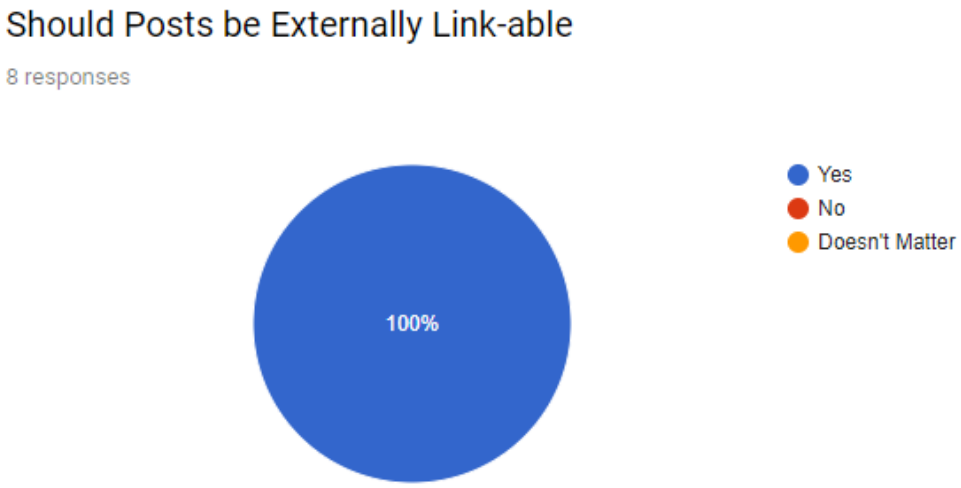

Figure 32: Externally Link-able Posts

As described in the second user survey every single user requested an externally likable post system. This would allow users to be able to generate a URL for specific posts and then be able to share that link with other users. This would have a two-fold utility allowing users to save posts for later but also develop the community by directly sharing posts with other users. A possible implementation of this feature is described in chapter 4.3.

Users also mentioned in the focus group that a difficulty attribute would also be greatly useful. This would allow users to further filter content into posts that were applicable to their current skill level. There could be author defined difficulty levels then user defined difficulty levels. This level would describe how much prior knowledge would be required to understand a topic. This could then link into the post tagging system to give users a content chain to help them understand a specific topic.

8. References

- *(Winslow 1996)* – "Programming Pedagogy – A Psychological Overview", Leon E. Winslow, ACM SIGCSE Bulletin – Volume 28 Issue 3, Published: September 1996, Pages 17 to 22
- *(Allan G. Bateson, Ralph A. Alexander, Martin D. Murphy 1987)* – "Cognitive Processing Differences Between Novice and Expert Computer Programmers", Allan G, Bateson, Ralph A. Alexander, Martin D. Murphy, International Journal of Man-Machine Studies – Volume 26 Issue 6, Published: June 1987, Pages: 649-660
- *(Wikipedia*-Myspace 2010) – Wikipedia Myspace (2017) Available at: https://en.wikipedia.org/wiki/Myspace (Last Accessed: 12th October 2017)
- *(Luis von Ahn 2008) – Luis von Ahn, Laura Dabbish 'Communications of the ACM Volume 51 Issue 8' (August 2008), Pages 58-67*
- *(Zwass 2010)* – "Co-Creation: Toward a Taxonomy and an Integrated Research Perspective", Vladimir Zwass, International Journal of Electronic Commerce – Volume 15 Issue 1 Published 2010, Pages 11-48
- *(Geiger 2011)* – 'Crowdsourcing information systems: a systems theory perspective' *In proceedings of the 22nd Australasian Conference on Information Systems (ACIS 2011)* Sydney, Australia, Sydney: ACIS Pages 0 – 12
- (rigaux.org 2017) Available at: http://rigaux.org/language-study/syntax-across-languages.html#VrsMnlMmrllc (Last Accessed: 12th October 2017)
- (YouTube-Press 2018) – Google, YouTube (2018) *YouTube Press* Available At: https://www.youtube.com/yt/about/press/ (Last Accessed: 06th April 2018)
- (Google-Firebase Pricing 2018) – Google (2018) *Firebase Pricing Schemes* Available At: https://firebase.google.com/pricing/ (Last Accessed: 07th April 2018)
- (Stack Overflow 2018) – Joel Spolsky, Jeff Atwood (2018) Available at: https://stackoverflow.com/ (Last Accessed: 11th April 2018)
- (Google Polymer 2017) – Google (2017) Available at: https://www.polymer-project.org/ (Last Accessed: 11th April 2018)
- (Google Firebase 2018) – Google (2018) Available at: https://firebase.google.com/ (Last Accessed 11th April 2018)
- (Reactjs 2018) – Facebook Open Source (2018) Available at: https://reactjs.org/ (Last Accessed 11th April 2018)
- (Polyfire 2018) – Firebase *Google* (2018) Available at: https://github.com/firebase/polymerfire (Last Accessed 11th April 2018)
- (Real Time Database 2017) – Firebase *Google* (2017) Available at: https://firebase.google.com/docs/database/ (Last Accessed 11th April 2018)
- (Github 2018) – Github (2018) Available at: https://github.com/ (Last Accessed 11th April 2018)
- (Google O-Auth 2017) Google (2018) Available at: https://developers.google.com/identity/protocols/OAuth2 (Last Accessed 11th April 2018)
- (Dev.to 2018) Dev.to (2018) Available at: https://dev.to/ (Last Accessed 12th April 2018)
- (Reddit r/programming 2018) Reddit (2018) Available at: https://www.reddit.com/r/programming/ (Last Accessed 12th April 2018)
- (Hacker News 2017) Hacker News (2017) Available at: https://news.ycombinator.com/ (Last Accessed 12th April 2018)

9. Appendices

9.1. Appendix A: Market Research Report

9.1.1. Review of Stack Overflow's Downsides

Stack Overflow [Stack Overflow 2018] is an online platform that allows developers to crowd source solutions to problems ranging from basic bug fixing to complex system architecture solutions. The system works by allowing users to post "questions" which can then be answered by other users on the platform. The original poster can then select the best answer from all the responses. That answer will then be displayed below the question so that any subsequent users searching for that problem will hopefully find the "correct" answer faster.

A problem arises however when you consider the nature of the platform. The Initial user often posts a question alongside snippets of their own code relating to the problem. This often results in the best answer for that question being the solution to that specific problem which often is provided with minimal explanation. While I am unable to produce hard statistics; during, my time with the site I have noticed the trend of if the initial user posts code snippets and asks for corrections the top response will be a bare bone correct code snippet. By bare bone I mean all the post will include is the "correct" code and a sample of what the result should look like. While this may be good for the initial user as their problem has been solved any subsequent users may have issues as their problems may not be the same or they will not develop any understanding of the underlying system they are using. Instead they will rely on copy and pasting which develops little to no understanding of the system they are using, creating an unhealthy dependency on Stack Overflow.

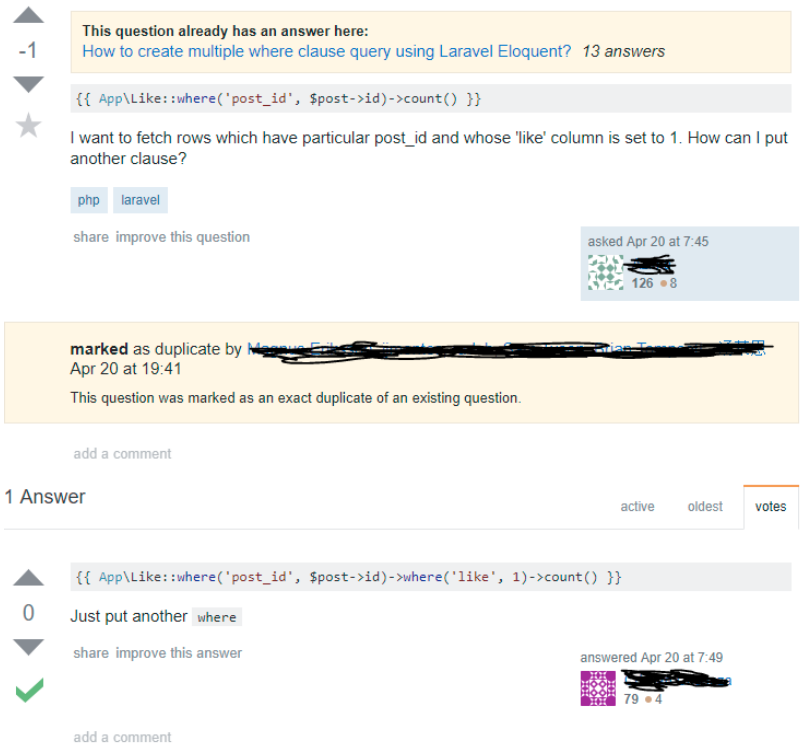

Figure 33: Stack Overflow Post View

Here is a basic example of the above-mentioned problem. Stack Overflow is littered with questions like these relating to basic syntax and rather than explaining or linking to external resources a pre-made solution is posted. This answer was rated as the "best answer" meaning that the solution most likely satisfied the original user. It is possible that the asker figured out their problem and understood the issue but there is also a great chance that they simply copied and pasted the solution and continued with their day.

Another issue that readily plagues Stack Overflow is post duplication (an example shown above). Due to the problem orientated nature of Stack Overflow a user searching for a specific solution can be greeted by hundreds of posts all asking the same thing and hundreds upon hundreds of responses all linking and explaining the solution in different ways. Rather than creating a set number of resources for a specific issue an uncapped number of posts can be created. These can cover the same topic and while duplicates can be marked as such they can also be incorrectly marked making it even more difficult to find a suitable solution.

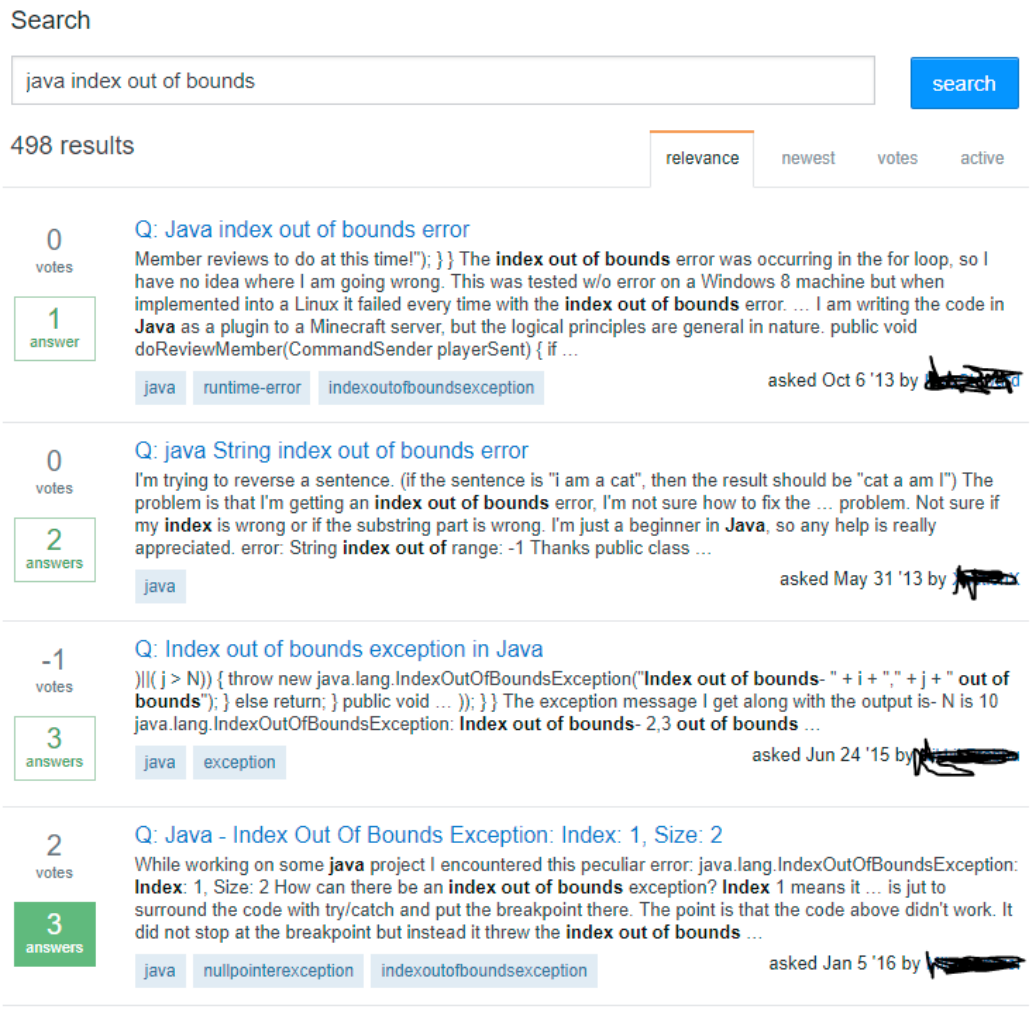

Figure 34: Stack Overflow Search

As you can see, searching for a common problem such as an index out of bounds exception returns just under 500 results and that is for the Java language alone. While some of these posts may contain useful and different information quite a lot of them are simply people pointing out minor errors in other's code.

This is what I was referring to earlier when I spoke about Stack Overflow [Stack Overflow 2018] being "Problem Orientated". Often, a lot of users are drawn towards asking similar questions while other users are equally drawn towards providing straight solutions. Stack Overflow rewards users who post many simplistic answers over users who post a few but well explained responses. Rather than creating focused resources to help users everything is broken up in many different threads many of which are often duplicates of one another.

Talking about the user reward system. Users are rewarded through the badge and medal system. By completing certain milestones and tasks users are given a gold, silver or bronze badge (respective by task difficultly) and this adds to their reputation. This is a number that can be increased through frequent and popular posts. For a user wishing to game the system and gain large amounts of medals and reputation quickly it would be best for them to hunt out duplicate, easy to answer, questions and answer the same questions repeatedly. While they may gain a decent amount of reputation by posting

a long and thought-out answer that takes time and effort. Instead, they could answer simple questions all day to the same, if not better, effect.

This then creates an information economy that favours quick responses which provide bare bone solutions over in-depth posts. Users who ask questions are far more likely to select a response as the “best answer” if it simply gives them the solutions while longer posts that explain the solution thoroughly can sometimes deter users who have little knowledge of the field they are in.

That is not to say that all users on Stack Overflow game the system by posting short responses to all the duplicate questions. There are users who take the time to produce detailed responses (sometimes a little too detailed) to questions but those posts can be few and far between. Stack Overflow is a tool for developers like any other site or app. In some regards it works well. For example, being a site to crowd source developers to give your code a second look over for basic bugs or allowing developers who already know their craft to quickly get a solution to a rare problem. However, it does leave a hole in the market as it can be found lacking in its ability to explain various concepts in an easy to find manner. Searching for a specific topic will often find a provided solution that gives no explanation to its working and while an experienced developer can often use their intuition to decipher the solution many languages and technologies are over complex or confusing making this task far more difficult.

9.1.2. Conclusion on Stack Overflow

The pitfalls and downsides of Stack Overflow are a great direction of what to stay clear from in terms of my own project. My tool and Stack Overflow will be separate entities created for different purposes. Stack Overflow is a site to be used to solve specific problems and to assist in basic debugging and that specific use creates a hole in the market for my application. For my syntax application to be successful it must focus on specific areas which aren't already covered in today's market.

My application should focus on gathering a crowd's response to a certain problem or issue then attempt to trim the excess posts and make it easy for users to find and search for specific syntaxes. One issue with crowd sourcing your material is that you end up will a lot of duplicates of varying quality and detail. My system should focus on taking the best of the best and hiding the rest so that when the user searches for a specific feature they find a post that covers the topic to the best possible extent.

This could be achieved through a limited result search. For example, if the user where to search for "how to declare an array in C++" they will only be shown the top five responses depending on their preferred criteria. The rest of the responses would then be hidden under a "show more" button. This means that should the top five responses be poor the user can choose to continue searching. However, if the user finds their wanted information within those five posts they will have not been flooded with pointless data.

A duplicate system like Stack Overflows could also be implemented. Their system works by allowing users to flag certain questions as duplicates and then link to the original or highest voted question. Unfortunately, when searching for solutions you sometimes still find these duplicate threads. In my system I could "trim the fat" by allowing users the option of hiding duplicate threads showing only the original and the most highly rated instead. Once again, allowing users control over their results and giving them the option of a "straight to the point" search system will increase their efficiency in finding intended results.

So, for my application/system to be different enough from Stack Overflow to stand a chance of being its own entity I must focus on the "Solution Orientated" economy compared to Stack Overflow's "Problem Oriented" economy. By this I mean the best solutions should be pushed to the top of search results rather than the best questions. In my system it might even be a good idea to forgo questions as they appear in Stack Overflow all together. Instead relying on user posted bounties for certain information and when a user searches for a specific topic they are only shown solutions instead. This would help encourage users to create the best possible responses that convey the required information. Then attempt to fully explain the topic so that everyone may learn that information and apply it in their own environment where needed.

9.1.3. Dev.to

Dev.to [Dev.to 2018] is an online community and article board for developers from various backgrounds and professions. It allows them to create, post and share both long and short form articles about specific topics. These topics can be anything from web development to software engineering as well as questions about the industry itself and people's lives within it. Anyone with an account can create an article and then sort it into one or more of the pre-created tags. These tags can then be used by other users to sort and navigate through the huge range of available posts.

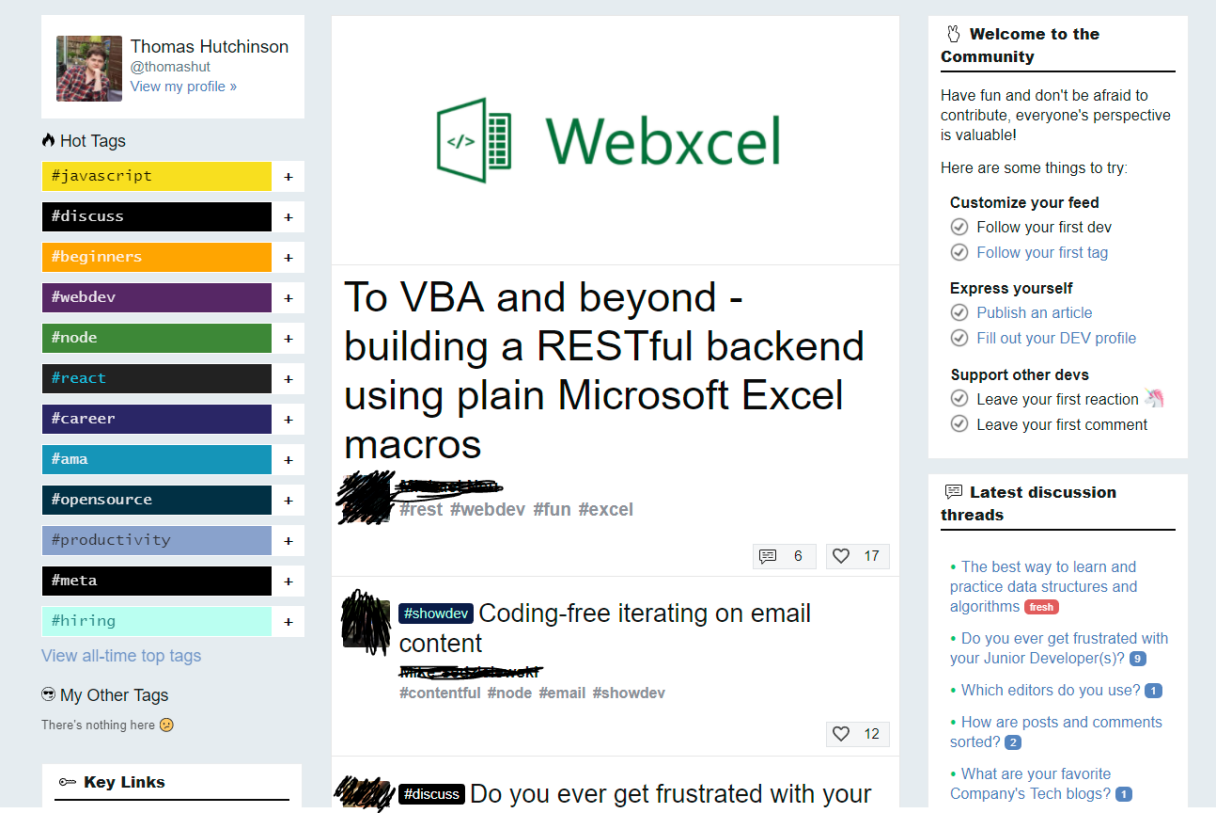

Figure 35: Dev.to Homepage

Above is a screenshot of the Dev.to landing page. Like Stack Overflow [Stack Overflow 2018] you are first greeted with a dynamic feed of the most active posts. By following tags, a user may influence the content of the feed but if no tags are followed then a generic list of content from across all the tags will be displayed. This appears to be an effective way of encouraging user interaction as it keeps them updated with a constant revolving list of new content to consume. By encouraging users to constantly consume content you also stand a better chance at increasing user retention as they become invested within the platform and may even make it a part of their daily routine. This is the ideal outcome as it means you will have a reliable flow of users every day which ensures a strong platform.

You can also see on the left-hand side there's a "Hot Tags" section which suggests various tags that a user may be interested in. Suggesting new sections of the site to go too may help push developers and users to research different topics and maybe even pick up new skills and knowledge. The use of colour also pulls their attention across to those tabs as the vibrant contrast makes that interaction stand out

on the page. The use of colour is an interesting technique to direct user flow as if done right it can pull a user's eye around the page and if done wrong it can make the site look cheap and unprofessional. So, the use of colours outside of the site's natural pallet must be controlled and used sparingly as overuse can distract from the overall design.

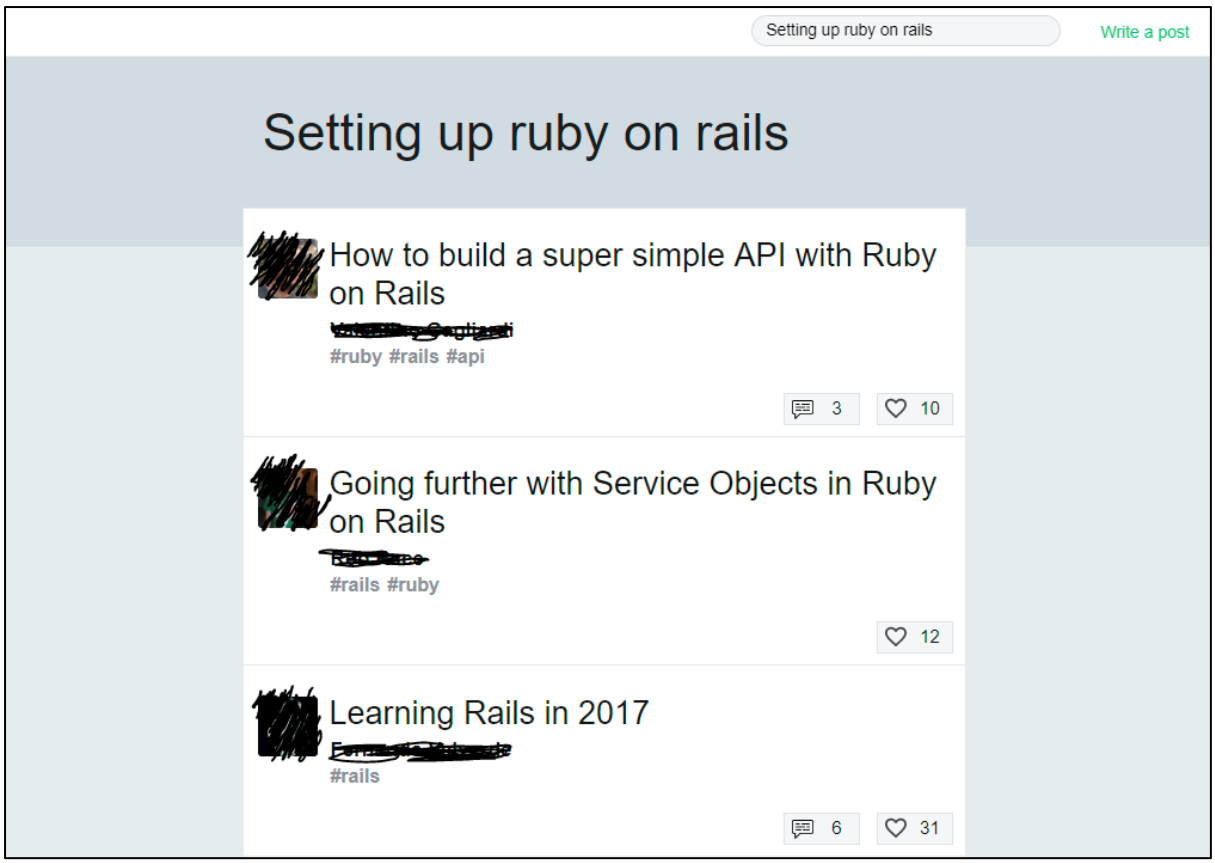

Figure 36: Language Page

Searching for a topic on Dev.to is a rather easy exercise compared to Stack Overflow. The site makes use of a universal navigation bar at the top of the screen which contains a logo and a search bar. This in a single button and a text field allows the user to effortlessly navigate the site and find specific content they are looking for. As an example, I searched for an article on how to set up Ruby on Rails and the top result turned out to be a comprehensive guide on creating a simple API. For someone wanting to create their first Ruby on Rails project this would be brilliant as it took a matter of seconds to find an article that explained the entire concept in-depth alongside code snippets and discussion.

One downside I would throw at this screen though is it is quite sparse on information. To find out more about a specific article I would have to navigate to that page and if it turned out to be something I wasn't interested in I would have to navigate back. From this screen you are only given a small handful of information: the title, author, it's tag, number of comments and number of likes. While the number of comments and likes can be a good indicator of an active and useful post more information would be very useful. For example, one of these posts here makes mention of a date which is very important. Technology is something that is always adapting and evolving however when searching for

a topic most users will want to have the most up to date information. It is impossible to determine at a glance at the article's creation date unless the user was nice enough to include it in the title. To find out you must navigate to the article's page itself which can be quite a pain if you need to navigate through several articles.

Another issue that plagues the entire site is a lack of filtering options. This affects both the search page and the tag pages. When browsing to these sections you are simply given a list of articles without any apparent rhyme or reason. If I were to hazard a guess it appears that search results are determined by the amount of activity on an article (comments and likes) while tag pages appear to just display most recent content. Thankfully, this is most likely the preferred set-up for most users. However, having the tools present to alter the display of search content or content within a topic would be extremely helpful. For example, it might be interesting for a user to be able to see the highest rated post of all time within a specific tag section. Reddit allows this functionality but allowing users to filter and sort their content by several parameters such as: top, controversial, recent and hot. This gives the user more control over their content and helps to maintain retention as they can alter their content feeds as their mood and needs change.

Finally, an issue that only appears with exceedingly long articles is navigation. Often if an article is discussing a rather complex topic it is not uncommon for that article to be quite long in length. The problem arises however when attempting to jump between various sections. There are no in article navigation options, not even a jump to top at the bottom of the page. This means that scrolling through and finding specific sections of a long article can be particularly tedious and quickly jumping to a required section demands the user undertakes a fair bit of scrolling. Solving this would be as simple as including a side navigation bar that hot-linked onto headers within the text helping to solve this issue and making article browsing far easier.

9.1.4. Conclusion on Dev.to

As mentioned in the Stack Overflow conclusion I believe my system should follow a “solution orientated” crowd sourcing method. Posts should be focused on the solution and not the question and when a user searches for a topic they should be given solutions, not other people’s problems and questions. Dev.to is a perfect example of a “solution oriented” design as when you search for a topic or click on one of the tags you are greeted with a list of articles. These articles often are completely about discussing and describing a topic in detail. When implementing my system, I believe a similar style would greatly benefit the user as it allows them to find their solution quicker and abstract away a specific problem or implementation. Abstracting user’s problems and questions away assists in promoting learning over simple task completion. As when users see a problem and solution chances are they will simply take the solution and run with it.

A new problem now arises though as my system now must be sufficiently different enough from Dev.to to warrant its creation and existence. Users who are already invested in one platform are difficult to transfer to a new platform even if there are clear advantages because they are invested in the current solution. So, my tool must be different from Dev.to in a major way. The main way this can be done is to change the focus from writing comprehensive articles and focus more on specific, small scale, learning resources. Dev.to does a great job of explaining a specific topic and educating users about all the details. What it doesn’t do is provide a quick and easy fashion to learn about one specific sub-topic or specific syntax. My tool should focus on separating and categorising its resources into specific sections and creating a greater depth of organisation. Rather than cover a whole specific topic it would be better to focus on one specific technique and how to not only perform it but learn it.

Because of this greater focus on specific topics and techniques my system must also learn another lesson from Dev.to and that is providing filters and search tools that allow users to find their specific solution. My tool if it gains traction with an audience will have a higher post count and more separate bites of information available to the user. Therefore, they need a way to search through that data and find what they want. Providing something as simple as a “relevant” filter to search results which orders posts on based on their matching properties to the user’s criteria will go a long way. Then providing additional filters such as “top” will allow them to alter result lists if the initial list is not to their liking.

By encouraging smaller and more concise resource creation as well as providing more tools that allow users to search and filter my tool’s purpose will be different to that of Dev.to. It must completely focus on the learning of specific techniques and syntax over the learning of whole topics and then make that content easily accessible. By providing a wider range of tools and filters this larger range of smaller content should be accessible by the user meaning that my syntax tool will operate in a different fashion.

9.1.5. Community Forums – Reddit and Hacker News

Apart from the major programming help websites you also have various smaller forums for developers to get together and share information. For this I will analyse some of the largest of these forums those being Reddit's r/programming [Reddit r/programming 2018] and Hacker News [Hacker News 2017].

Figure 37: Reddit Subreddit Page

Staring with Reddit, as you can see upon navigating to the sub reddit for programming you are greeted with a list of "hot" posts. Hot referring to the post sudden burst of activity or recent creation. This method of filtering is great for ensuring that your sites content does not stagnate and is always changing. On top of that, you also have additional filters for things like: new, rising, controversial and top. As mentioned in the Dev.to [Dev.to 2018] section having this range of choices allows the user to alter their content stream as needed.

Reddit itself is more focused around the individual posts. When a user creates a post, it will be added to the new section and outside of a few filtering options there is no more categorisation. It is simply a long list of various topics from across the "programming" eco-system. This can range from web development to system architecture and anything in-between. It is a Hodge podge of various posts and topics and for general viewing and reading that is great. However, if a user wanted to find out about a specific topic that would be rather difficult.

While there is a search bar with various tools and options attempting to find information about a specific topic can be a gamble. If you are specific with what you want and make use of all the search tools, then you might find a post you are looking for. Changes are though you will not. I would describe

Reddit as a brilliant tool for finding out about new topics. It is something that opens doors to new areas but if a user wanted to continue down that path they would need to make use of separate sites and systems. This is down to the fact that there is no categorisation. You can search using keywords but again you are going to get a very long list of posts and not all of them are going to be what the user is looking for.

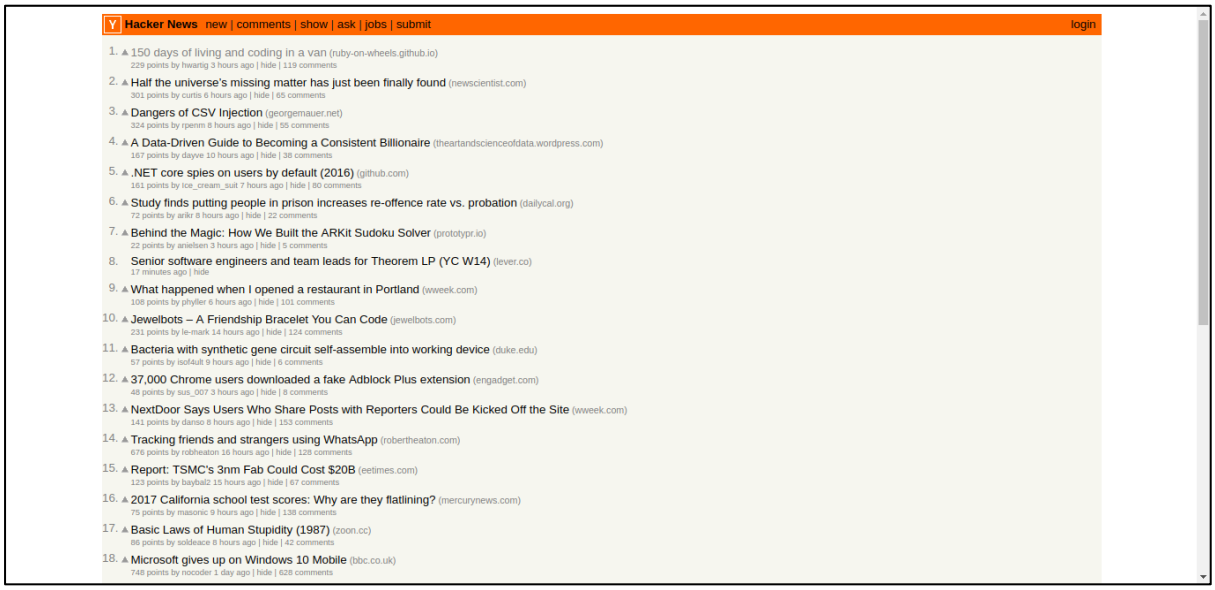

Figure 38: Hacker News Homepage

Hacker News is incredibly like Reddit but without many of the filtering and searching tools. It also cuts out the middle man so to speak. Hacker News is simply a forum of straight links to various articles and other sites. It works more as a news board or a list of interesting places rather than a forum for discussion. While there is a comments section it is not the first thing you see and it the comments do not relate to specific posts and are instead more general. The site seems completely focused on just providing an ordered list of interesting news articles.

An interesting point of note is the lack of a downvote option. On reddit a user may “downvote” a post if they believe it to be incorrect or of poor quality. This will then be factored into how the post is displayed to other users. However, hacker news does not make use of this feature instead only allowing up voting. This could be an effort on behalf of site creators to stop certain groups of users from dominating the boards and removing all content they do not like the look of.

Reddit gets around this problem through the concept of “post decay”. This refers to the idea that the longer a post sits at the top of a list the more difficult it becomes for that post to remain there. Eventually, all posts will drop down in the listings on certain filters. Achieving this can be done through a variety of ways but the most common is attributing a score to a post. You then sort the posts based on that score. A post can gain score by receiving comments and upvotes but as time continues a negative score is applied to that post. For example, say if a post reaches the top ten spots, after one day the post could receive a negative point. This would then continue day after day until that post as

been removed from the top of the hot category. You can also always provide a “top” board that shows the highest rated content but with sites like Reddit and Hacker News that wold not be favourable.

This is because these sites rely on entertaining and enticing their audiences to gain user retention. Having a feed that never changes or is only filled with poor content would get boring and users would gradually leave. But by using a system that favours new highly voted posts and discards older posts you can keep a “fresh” feed. This in the case of these forums will increase user retention.

9.1.6. Conclusion on Forum Based Sites

While there is a lot to learn from these forums they differ quite largely in their final goal from my tool. These sites aim to entertain and intrigue users and while my tool does hope to grab the attention of users it is not attempting to strictly "entertain them". The concept of a "hot" filter is not something that would apply fully to my system. If a user were to look for a solution or resource on a specific language most of the time they are going to want to find that specific solution. They don't care about constantly changing content, they want the correct answer.

As mentioned previously, technology does change though so I could take certain aspect from these sites in terms of that. My own site could possibly allow users to filter posts based on their date of creation. A user learning a new language may want to ensure they are getting the most up to date information and simply relying upon a top system could drown out newer, more accurate, posts. So, with that in mind I could offer a "relevant" filter. This filter will consider: rating, user activity, posters statistics and the date of creation. It will not include a decay feature though. It simply means that a newer post may be favoured in certain circumstances over an older post.

The idea of also linking to external resources like Hacker News appeals to me and could be a powerful feature within my system. While it works best to keep everything self-contained to increase speed allowing users to link to external material to provide other perspectives or extra detail could be quite useful. My tool's goal is to educate users on specific syntaxes and language features. Once the user picks up the basics of the feature an external resource could be linked, and the user would have the choice of continuing their study.

That said, I still want to retain the core of the content with the application. This means that posts that fully rely on links will be downvoted and won't be as favourable in search results. Keeping users within the application helps to save time as they do not need to worry about switching sites or applications. It also helps the user keep track of resources in case they should need them again in the future. If the tool was completely reliant on external content it would be quickly devastated if that external content happened to become unavailable, one day. That is why that original content will be favoured in the new system.

9.2. Appendix B: Language Research

9.2.1. Thoughts on Differences Between Languages

My initial thought is the idea that moving between languages can differ in perceived difficulty depending on the learner's first language and the new intended language. By this I mean the acquisition of a new language can be a trivial matter or an enormous trial depending on the learner's current knowledge and a few other properties. I found the major factor of moving between languages is the learners understanding on generic programming concepts and structures. The more thoroughly the learner is familiar with these concepts the easier it will be for them to spot those features in other languages and transfer their skills and knowledge

As well as understanding of generic language concepts the new language can also play a large role in the ease of learning and acquisition. All programming languages fall into set levels, that being Level 1: Machine Code, Level 2: Assembly Language, Level 3: High Level Languages and the debated level 4: intelligent languages. Learners who possess a great knowledge and understanding of structures and concepts of a specific level may have an easier time learning a language on the same level. On top of this languages are often grouped into "families". These families are defined by a few factors, but they are primarily grouped together based on a common starting point. Many languages are either direct forks of a baser language or directly borrow features from that language to entice developers into transitioning.

A learner who decides to acquire a new language which resides on a level they are already comfortable with as well as choosing a language within the same family then transitioning should be as simple as picking up minor syntactical differences. That is where my tool should step in. For example, a developer may want to implement a specific structure or control loop and the new language differs ever so slightly in its implementation compared to their already learned language. My tool could then step in and allow that learner to search for their structure and quickly pick up the syntactical differences.

Most of the major programming languages share many of the same concepts and just about all popular Object Orientated languages share the same features and structures. It is just a matter of learning the specific syntax. Sites like Dev.to and Stack Overflow are overburdened in information. Trying to find out one specific small item such as a single syntax for a structure can be difficult. Therefore, my system should provide these smaller snippets and lessons on syntax. The user will need one or two lines and the tool should provide them that.

9.2.2. Differences in Common Features Between Languages

9.2.3. If Statements:

```
if (Boolean Logic) { // Code to be executed }
```

The generic if statement syntax holds across most of the popular languages such as Java, C#, C++, JavaScript, PHP and C. One interesting caveat though is that some languages support multiple version of the if then syntax. This could be down to the more commonly used syntax being added to the language after the fact to ease learners into the language.

For example, PHP supports another If then construct:

```
if (Boolean Logic): /* Code to be executed */ endif
```

As mentioned above this is most likely the language developers offering multiple syntaxes to support developers from a range of backgrounds. Unfortunately, not all languages offer the same helpful features.

```
If (Boolean Logic): // Code to be executed
```

Above is the if statement for the popular language python proving that not all the main languages follow the same syntax. In fact, there can be some small details that developers from other languages take for granted. This is where a lot of developers would get caught up as the syntax is almost the same. My tool would come in effect here as when a developer tries and fails the if statement they could simply look up “Python if statement” and they will get a short and concise solution.

9.2.4. Declaring a function

This is where languages really start to separate from one another in a basic manner. The concept of a function differs from language to language. Generally, it's defined as a block of code that can be executed on call and can take parameters and return an output. Once again though, different languages can have widely different implementations of functions. Some languages even separate them into functions and procedures where one can mutate variables and data and the other can only perform calculations.

```
Def func($par1, $par2, …): …
def func(para1,para2) … end
```

You then come across situations like the above. Python and Ruby's function declarations are rather similar outside of one or two minor differences. Once again this is something that can trip up a newer developer. Another feature that will be important to my system is allowing the accurate and detailed dissection of code snippets. For example, the above snippets show how to declare a function, but a user may wonder what the dollar signs are for. The user may also wonder what all the keywords in another snippet stand for. By allowing users to dissect their code and explain it fully hopefully new learners can understand the keywords and syntax and hopefully start piecing the language together in their heads.

9.2.5. Semantic Naming

Languages can also vary quite widely on what they call specific concepts or semantics. Thankfully, most of the popular languages stick to the same naming convents such as calling a body of code that can be executed in another location a "function". However, some languages don't play quite as nice and can employ different naming schemes that throw off newer users.

```
Free(x);
Delete x;
```

Above we have the methods of freeing memory in C and C++. These two languages share a few similarities since C++ was based on the classes of C. However, their methods of freeing memory are both different and this isn't the only case of methods and operators changing names across languages. The syntax tool could support for this by allowing posters to compare a specific syntax to other language's semantics. Then when a user searches for a specific syntax's equivalent in another language that tag could be pulled displaying the content that user is looking for.

9.2.6. Comments

An often-underused feature of every language is the comments. These snippets of user entered text can be fundamental in ensuring the readability of a piece of code and every programmer should make use of them. Every language has their own scheme for allowing programmers to enter text that won't be factored in by the compiler/interpreter and it may be on the areas that varies the most from language to language.

```
// Hello World!
/*
* Hi World!
*/
```

One of the most common manners of creating a comment is the double forward slash. This syntax is used across: Java, JavaScript, C++, Swift and a few others. Any user who learned their first language in the C family of programming languages will often fall for this when they attempt to create a comment in another programming/scripting language.

Ruby:

```
# Hello World!
=begin
Hello World!
=end
```

BASIC:

```
REM Hello World!
```

Haskell:

```
{- Hello World! -}
-- Hello World
```

XML / HTML:

```
<! —Hello World! →
```

Python:

```
“”” A Hello World Block Comment “””
#Hello World
```

As you can see comments differ widely between languages and in some languages, comments operate quite differently. For example, in a lot of interoperated languages such as XML and HTML the comments are not parsed out, so an end user can view those comments using any kind of web editor. Other languages also miss various commenting features such as block commenting. Python for example has no block level comment. Instead users can create a string literal that is not assigned to anything and that the compiler will just ignore.

Commenting is an important part of any development project and many newer developers often skip out on this step. This is especially detrimental as often a newer developers code is harder to read than a veteran because at that stage newer developers tend to glue bits of code together. This leads to difficult to read code that really needs the commenting. That's why learning the commenting scheme of any language is an important step that shouldn't be overlooked.

9.2.7. External Compiling

While these days most developers work within an IDE that handles building and compiling as part of an automatic procedure sometimes a developer must compile their own code. Again, this aspect of development is often overlooked, and it can be quite troublesome to find out the exact steps to compile and run a piece of code. So, it may also be a good idea to cover compiling code in various operating systems and for various languages.

Java Compiling:

```
Javac ./filename.java

Java  ./filename
```

C Windows Compiling:

```
Cl filename.c
```

C Linux/Mac Compiling:

```
Gcc filename.c
```

C++ Windows Compiling:

```
cl /EHsc filename.cpp
```

C++ Linux Compiling:

```
G++ filename.cpp
```

Over only a few different languages so many different keywords and dependencies are required to compile a section of code. Including a manner for users to post and inform others about how to compile code in terminal/command line will be extremely important. It will also be useful to allow those posters to inform users of what dependencies and programs they will require to compile various languages.

9.2.8. Memory Management

Management of memory is a feature that differs on a massive scale from language to language. Some languages such as Java abstract away all management and instead rely on seemingly automatic garbage collectors. On the other side, you have languages such as C, C++ and C# which all demand that the user looks after their own memory by allocating and freeing items as required. A user moving from a language such as Java to a language such as C can quickly encounter memory related issues. This is down to them being accustomed to an environment that cleans up after them. So, when they must finally deal with memory management some developers can encounter various problems.

```
Malloc(...)
```

```
Calloc(…)
```

The above snippets are both from the language C. A user moving to a new language may not have any knowledge or even understand the concept of memory allocation, so the tool must be able to accommodate that kind of user. For example: when the user searches for variable declaration in C at the bottom of the post it could include a tag line such as “explore memory allocation”. This will ensure that users have a trail to follow through with.

```
System.gc(); // Java

GC.start(); // Ruby

VM.garbageCollect(); // JavaScript
```

On top of that, some languages provide the support to manually trigger automatic garbage collection. These exercises would be pointless in most circumstances though as it is performed automatically in most systems anyway. This is another thing my tool could provide notes on. By allowing posters to provide additional comments such as “unnecessary” or “redundant in most circumstances” that could ensure developers aren’t falling down the pitfall of adding redundant code. It also helps to encourage learning of a language at a system level. By understanding how garbage collection works in a specific language the learner can hopefully begin to understand other features.

9.3. Appendix C: Testing Report

FR 1: User can Create Tutorial Posts			
Test #	Description	Expected Result	Actual Result
1-1	User can create a valid post when entering test in all the post fields.	Once entering a valid post (A Title, A Language and some content in the post body) and pressing submit the post will be added to the post	Post created successfully and immediately shows up in users created post list without refresh
1-2	User cannot create a post without a title	Once entering a post (NO title, a Language, and some content) and pressing the submit button the post is not saved	User was unable to create a post without a title. *(No on-screen error message, just a console log)*
1-3	User cannot create a post without a language	Once entering a post (A title, NO Language, and some content) and pressing the submit button the post is not save	User was able to create a post without a language. *Incorrect behaviour – Bug # 1*
1-4	User cannot create a post without content	Once entering a post (A title, a language and NO content) and pressing the submit button the post is not saved	User was unable to create a post without any content *(No on-screen error message, just a console log)*
1-5	User cannot create a post without any content in any of the fields	Leaving the fields empty and pressing the submit button the post is not saved	User was unable to create a post with no fields *(No on-screen error messages, just a console log)*
1-6	*(Re-test)* User cannot create a post without a language	Once entering a post (A title, NO Language, and some content) and pressing the submit button the post is not save	User was unable to create a post without a language (No on-screen message, just a console log)
FR 2: User can Tag Other Posts to Their Post			
2-1	User can search for another post and tag that post to the currently created post	Upon searching for a valid post (using an author name, title or language) and pressing the tag button the post will be tagged	User can tag another searched for post.
2-2	User cannot tag the same post twice to their current post	Upon searching for a valid post and tagging it the user cannot tag the same post a second time	User is unable to tag another post as one tagged the button turns into a remove tag.

2-3	User can tag a post then remove that tag	Upon searching for a valid post and tagging it the user can remove that tag by pressing the remove tag button.	User is able to remove the tag by pressing the remove tag button
2-4	User can tag a post, remove that tag then re-add the same post.	Upon searching for a valid post, tagging it then removing it the user can then re-tag that same post again.	User can re-tag a previous removed post
2-5	A user can tag multiple posts	Upon search for a valid post the user can tag as many posts as they like and those will be added to the current post	User can tag multiple posts
FR 3: User can View Other Posts and Rate Those Posts			
Test #	Description	Expected Result	Actual Result.
3-1	Upon connecting to the feed page, the user will see all currently available posts	With post added to the database upon connection to codex-feed the user will see a list of posts.	Upon connecting to the feed page, the user can see all the currently created posts *(limited to 50 posts)*
3-2	User can click the view post button to see a detailed post page.	Upon clicking on the view post button for a specific post the user will see a full detailed page for that post.	User can view a detailed post page by clicking on the view post button for a specific post
3-3	User can up-vote conciseness rank for a post	Upon clicking on the conciseness rank for a post that rank will be increase by 1	User can increase the conciseness rank by one by pressing the corresponding button.
3-4	User can up-vote well explained rank for a post	Upon clicking on the well explained rank for a post that rank will be increase by 1	User can increase the well explained rank by one by pressing the corresponding button.
3-5	User can up-vote well linked rank for a post	Upon clicking on the well linked rank for a post that rank will be increase by 1	User can increase the well linked rank by one by pressing the corresponding button.
3-6	User can up-vote general rank for a post	Upon clicking on the general rank for a post that rank will be increase by 1	User can increase the general rank by one by pressing the corresponding button.
3-7	User can revoke an up-vote for conciseness rank	By clicking on the vote button again after upvoting the user can revoke their upvote for conciseness	User can revoke a conciseness upvote by pressing the upvote button again
3-8	User can revoke an up-vote for well explained rank	By clicking on the vote button again after	User can revoke a well explained upvote by

		upvoting the user can revoke their upvote for well explained	pressing the upvote button again
3-9	User can revoke an up-vote for well linked rank	By clicking on the vote button again after upvoting the user can revoke their upvote for well linked	User can revoke a well linked upvote by pressing the upvote button again
3-10	User can revoke an up-vote for general rank	By clicking on the vote button again after upvoting the user can revoke their upvote for general rank	User can revoke a general rank upvote by pressing the upvote button again
3-11	User cannot double vote through pressing the vote button twice or refreshing the page and voting again.	If the user presses the vote button again they cannot give +2 to any rank. They cannot also refresh the page to vote again.	User simply revokes their upvote by pressing the button a second time
3-12	User cannot manipulate their local cache to vote on a specific post twice	The user cannot vote twice on a single post by deleting their local cache.	Any manipulation of the cache results in an error giving telling them they have already voted if they have previous voted *(console error)*
3-13	User cannot vote twice on a single post by manipulating their network connection.	The user cannot vote twice on a single post by activating and deactivating their internet connection.	User receives a console error if their connection drops and they are unable to
FR 4: User Can Search for Other's Post Using Specific Criteria			
Test #	Description	Expected Results	Actual Results
4-1	User can search for a post based on just a post title	When entering a post title of a post that is present in the database that post will be returned in the results.	User can search for posts with just a title
4-2	User can search for a post based on just an author name	When entering a author name of a post author in the database that post will be returned in the results.	User can search for posts using an author's name which returns all posts by that author
4-3	User can search for posts based on just a language	When entering a programming language and pressing search the user will be returned all posts that apply to that language.	User can search for a for posts using just a language which returns all post related to that language.
4-4	User can search for a post tile alongside a language	When entering a post title and a language the	When searching using both a title and

		user will be returned all posts that share similar titles and relate to the entered language.	language posts related to that title and language are returned below
4-5	User can search for a post author alongside a language	When entering a post author and a language the user will be returned all posts by that author that are for that specified language.	When searching using both an author and language posts related to that title and language are returned below
4-6	Given a search criterion that returns posts the user can sort those posts based on Conciseness rank.	When searching for posts present in the database and selecting the “sort by Conciseness option” the posts with the highest conciseness will appear at the top of the search list.	User can sort on a specific post, but the posts appear to be back to front with the highest rank post being at the bottom and lowest rank at the top – Bug # 2
4-7	Given a search criterion that returns posts the user can sort those posts based on well explained rank.	When searching for posts present in the database and selecting the “sort by well explained option” the posts with the highest well explained rank will appear at the top of the search list.	User can sort on a specific post, but the posts appear to be back to front with the highest rank post being at the bottom and lowest rank at the top – Bug # 3
4-8	Given a search criterion that returns posts the user can sort those posts based on well linked rank.	When searching for posts present in the database and selecting the “sort by well linked option” the posts with the highest general rank will appear at the top of the search list.	User can sort on a specific post, but the posts appear to be back to front with the highest rank post being at the bottom and lowest rank at the top – Bug # 4
4-9	Given a search criterion that returns posts the user can sort those posts based on general rank.	When searching for posts present in the database and selecting the “sort by well linked option” the posts with the highest general rank will appear at the top of the search list.	User can sort on a specific post, but the posts appear to be back to front with the highest rank post being at the bottom and lowest rank at the top – Bug # 5
4-10	If the user enters no data nothing is returned	Upon entering no data into any of the search fields nothing is returned.	No data is returned if no information is entered
4-11	If the user searches for something that is not present within the database nothing is returned.	Upon entering a post title, an author or language that is not present within the	If no present linked to the search criteria is searched for nothing is returned.

		database nothing will be returned.	
4-12	Attempting to sort a result list that has only one result has no anomalous effects	If the user selects a sort option when a search criterion only returns a single result nothing strange happens.	If no data is present and a sort is used nothing happens.
4-13	*(Retest)* Test to ensure that all sorting criteria not sorts posts correctly	Test each of the 4 sorting criteria in turn to ensure they sort the posts in the correct order with highly voted posts at the top and low posts at the bottom.	Posts are now sorted in the correct order based on the selected sort criteria
FR 5: Users can Create Bounties and View Other Users Bounties.			
Test #	Description	Expected Result	Actual Result
5-1	User can view all currently active bounties	Upon connecting the bounties page, the user can view all the currently created bounties.	Upon loading the bounty page all the currently active bounties are loaded
5-2	User can create a bounty when providing valid information to all fields	Upon entering a valid: title, language and bounty content then pressing submit that bounty will be created and added to the list	Upon entering valid information, the post is created and added immediately to the bounty list.
5-3	User cannot create a bounty without a valid title	Upon entering a valid: language and content but no title then pressing submit the bounty will not be submitted.	User can create a post without a title *(Incorrect behaviour) Bug # 6*
5-4	User cannot create a bounty without a valid language	Upon entering a valid: title and content but no language then pressing submit the bounty will not be submitted.	User can create a post without a language *(Incorrect behaviour) Bug # 7*
5-5	User cannot create a bounty without valid content	Upon entering a valid: title and language but no content then pressing submit the bounty will not be submitted.	User can create a post without any content *(Incorrect behaviour) Bug # 8*
5-6	User cannot create a bounty with no valid information	Upon leaving all the fields empty and pressing submit nothing will be saved.	User can just press the submit bounty button and it creates an

			empty post *(incorrect behaviour) Bug #9*
5-7	User can up vote other bounties will which increase their rank	Upon pressing the upvote button for a bounty that bounties rank will increase by 1	User can up-vote bounties by pressing the upvote button
5-8	User cannot up-vote a post twice by pressing the up-vote button twice	Upon pressing the up-vote button for a second time after voting once the first vote is revoked causing a net increase in the bounty rank of 0	User is unable to upvote twice as the second press revokes the first vote.
5-9	User cannot vote twice on a single post by manipulating their local cache or connection to the internet	User cannot vote twice on a single post but manipulating their connection to the internet or deleting their local cache.	IF the user manipulates their local cache or internet connection they are given a console log *(should be on screen error)*
5-10	A user can down-vote a bounty by pressing the down-vote button.	When pressing the down-vote button on a specific post that post will have its rank reduced by 1	Upon pressing the down vote button, the post loses one point
5-11	A user cannot down-vote a bounty twice by pressing the down-vote button twice	When pressing the down-vote button a second time their first down vote will be revoked causing a net change of 0 in the bounties rank.	User is unable to down vote twice as on the second press the original down vote is revoked
5-12	A user cannot down vote twice by manipulating their connection to the internet or manipulating their local cache	By altering their connection to the internet or deleting their local cache the user is still unable to down vote twice on a single post.	Nothing happens, and user is given a console error *(Should be on screen)*
5-13	When a user discovers a satisfactory post, they may mark their bounty as complete therefore removing it from the bounty list.	Pressing the mark as complete button a post owned by that user the post will be removed from the list.	User can mark posts as complete at which point the post is removed from the list.
5-14	The mark as complete button will only display for users who have created that bounty.	Users who have not created the bounty will not be able to remove a bounty from the board.	Mark as complete button does not appear for non-author users
5-15	Bounties will be ordered based on their rank with bounties with the highest rank being at the top.	Upon connecting to the bounties page, the user will find the bounty with the highest rank at the top.	Bounties are not ordered on their rank *(incorrect behaviour) Bug # 10*

5-16	*(Retest)* Test to ensure that no fields of the create a bounty form can be left empty. Then test to ensure that the entire form cannot be empty.	Upon attempting to leave a form empty the user should be stopped and the bounty not created.	User cannot leave any of the fields empty. No on-screen error message – only console log.
5-17	*(Retest)* Bounties are displayed in the correct order with highest ranked bounties at the top of the page.	Upon connecting to the bounties page, the user will see the bounty with the highest rank at the top.	Bounties are displayed in correct order with highest ranked at the top and lowest ranked at the bottom
5-18	*(Retest)* Users cannot mark other users bounties as complete	The button to mark as complete should not appear on other user's bounties	Mark as complete button does not appear on bounties created by other users.
FR 6: Users May View and Delete Their Own Posts			
Test #	Description	Expected Result	Actual Result
6-1	User can view all their created posts by going to the Codex-Post page	Upon connecting to codex-post the user will see all the posts they have created.	User can view all their created posts from the posts page
6-2	If the user has created no posts nothing will be present	Upon connecting to the posts page of a user who has created no posts nothing will be present.	If nothing has been created, then no posts are displayed
6-3	User can delete their own posts	Upon pressing the delete button for a post that post will be deleted and removed from the database.	User can delete their own post
6-4	Attempting to delete a post while offline has no adverse effects	If the user attempts to delete a post while offline there are no side effects and the post is not deleted.	Nothing happens
6-5	*(Retest)* Users cannot view other users posts in their created list. Nor can they delete other users posts.	When connecting to the posts page the user should only see their own posts	User can only see their own posts. Therefore, they cannot delete other users post
FR 7: Users can Save Posts Offline for Viewing Without an Internet Connection			
Test #	Description	Expected Result	Actual Result
7-1	By pressing the save post button on the view detailed post page that post is added to the saved list	Upon pressing the save post button that post is saved and can be viewed on the saved post page while offline or online.	User can save posts to their list
7-2	The user can save as many posts as they wish without issue	Save a wide range of posts to ensure the user	User can save as many posts as they want

		can save a range of posts without issue.	within reason (50 posts)
7-3	The user tampering with their local cache should only affect their local view of the post and not the server side one.	By altering the local cache of the user, they should only change their view of the post while online. This tampering should not be reflected onto the server side.	Upon local tampering only, the users local view is changed, and no effect is had upon the database
FR 8: Users can Comment on Posts			
8-1	User can view all comments made by other users when viewing the detailed page for a post	Upon connecting to the detailed post view the user will see all comments made by both themselves and other users.	User can view comments made by others when viewing the detailed page
8-2	User can create a comment	User enters a comment in the post body and presses save. This comment will then be submitted and viewable by both the author and other users.	User can create a comment when giving valid details
8-3	User cannot create a comment without any content.	User enters nothing into the comment section and presses save. At which point nothing will be posted.	User can create empty comments Bug # 11
8-4	Pressing the load comments button will refresh the comments section	Upon pressing load comments, the section will be refreshed adding any new comments made.	Comments are refreshed after pressing the load comment button
8-5	*(Re-test)* User cannot create a comment without any content.	User enters nothing into the comment section presses save. At which point nothing will be posted	Comments are not saved if no content is added. (No on-screen error message)
Miscellaneous Tests			
Test #	Description	Expected Result	Actual Result
M-1	User can connect to the website via https://codex-8eb0b.firebaseapp.com/	User is taken to the home page upon connecting to the URL	User can connect using the URL
M-2	User can connect to the website via https://codexapp.review/	User is taken to the homepage upon connecting to the URL	User can connect though they are given a security warning and it takes significantly longer than dev URL (30 seconds)

M-3	*(Retest)* Non-logged-in users can search the site and view the news feed	Upon connecting to the site while not logged in the user should be able to read the news feed and search using the search page.	Non-logged in user can read the news feed and search for content using the search page.
M-4	*(Retest) – Mobile.* Header navigation bar should display correctly.	The header bar should span the entire width of the page on mobile displays.	The header bar spans the entire width of the page on mobile displays
M-5	*(Retest) – Mobile.* Mobile users will smaller screen sizes and resolutions should be able to log out without the button falling off screen.	If the user is using a smaller display and the navigation buttons take up all available space the user should be able to scroll the navigation bar to find the sign-out button	User is able to scroll the navigation bar on smaller screens to find the sign-out button

Table 7: Black Box Testing Report

9.4. Appendix D: Academic Research

9.4.1. Language Pedagogy

Teaching languages can be a difficult and complex topic to discuss because every single learner will be subject to different parameters and situations. A learner's previous knowledge and understanding of programming paradigms and concepts has been found to play a big role in their aptitude for understanding new language. Leon Winslow concluded that an expert can create various mental models and implement generic programming concepts easier than novices. He also states that a novice may only become an expert through years and year of practice and refinement *[Winslow 1996]*. It then stands to reason after these years of training and practice they have an understanding and skill that allows them to acquire these languages faster.

There is another aspect to this practice though. The simple act of constant practice isn't the only element that increases one's aptitude for language acquisition. Semantics also plays a huge role in determine a "programmer's general skill/ability" found Allan G. Bateson *[Allan G. Bateson, Ralph A. Alexander, Martin D. Murphy 1987]* A programmer's understanding of core fundamentals and topics will aid them far more than specific syntax knowledge when understanding new languages. Bateson also concluded that the greatest gauge of a programmer's general aptitude is their semantical knowledge over their syntactical knowledge.

Both Bateson's and Winslow's work combined help paint a more detailed picture of how one learns a programming language and learns it effectively. A novice will simply focus on surface features such as control structures and simple iteration while an expert will dive deeper and focus on more complex aspects of a language much faster *[Winslow 1996]*. This is because of their ability to understand complex features such as object orientation and they can manipulate that to solve problems much faster and to a much higher standard. Their code will consist of less errors, be more efficient and reach specifications much easier. A new language shouldn't affect an expect as much as a novice because they understand and can apply their knowledge regardless of the environment.

Regarding my own application these papers have helped to direct my own tool in the direction of providing syntactical assistance while rooting that assistance in semantic information. By this I mean, users will be rewarded for thoroughly explaining code snippets and referring them to generic programming structures and concepts rather than just writing the syntax with no explanation. This benefit both experts and novices as the experts get the code syntax they require, and novices can expand their knowledge of programming semantics. This over time will help develop them into better programmers by improving their semantical knowledge rather than just their syntactical knowledge. Which in turn improves their overall programming aptitude.

9.4.2. Crowdsourcing and Gamification.

For the syntax/semantic tool to be of use to anyone it is going to require a large amount of data. This data will have to be created covering a different topic and be sourced in a short time-frame. That is why crowdsourcing will be an indispensable technique in helping to build up the resource base and to sustain its growth. By gathering a community based around the application and encouraging accurate content creation the potential of the application can be realised.

What is crowdsourcing though and how can it be used effectively to gather a large quantity of data in a short time span (1 to 2 months)? Well David Geiger summarised crowdsourcing into 4 major categories: Crowd Rating, Crowd Solving, Crowd Processing and Crowd Creation. These categories where devised on the type and complexity of contributions required from the user as well as how those contributions were used and evaluated *(Geiger 2011)*. After analysing the requirements of my system, I devised that the best styles of crowdsourcing for me would be the Crowd Creation and Crowd Rating.

These two styles would be separated across the two primary user groups. The posters and the searchers. Users may travel between these groups freely but when completing a task, they will be a part of either one group or the others. The posters will follow a Crowd Creation crowdsourcing. This involves the users being given the freedom to create certain content within the specification of the site *(Geiger 2011)*. Using a form that requires them to enter certain information but then allows them to create whatever relevant text-based content they want they will have the freedom to provide high quality content. Searchers can then make use of a Crowd Rating system to assign value to these posts and ensure high quality content is rewarded. Geiger himself commented on this relationship between creation and rating saying that a larger audience of creators requires a larger audience of people rating *(Geiger 2011)*. This is because after a certain point and a large enough community managing in-coming content becomes impossible and a larger crowd is required to keep on top of it.

Unfortunately, crowdsourcing is useless without a crowd and for my system I would require a constant healthy stream of new content meaning the tool not only needs a crowd but a consistent one at that. Rather than amassing a crowd it would instead be better for the system if a community was constructed. Communities often develop around a mutual point of discussion or common goal and a healthy community remains as bonds and trust are developed *(Zwass 2010)*. By encouraging user interaction through the voting system and comments sections a community will develop around the application. This in turn encourages further user involvement and increases the chances of the user's continued use of the application. Which is what the application requires to provide up to date syntactical resources to the searching users.

On the topic of community retention. Keeping a community focused and attached to a specific application or concept can be the most difficult aspect of any crowdsourced project. It is the natural way of humans to start of enthusiastic about topics and activities and then gradually over time become board or disinterested. This then leads to them leaving the platform and migrating somewhere else. This would be categorised as a steady increase in users then over the span of several weeks and

months the user count slowly stops increasing and then starts decreasing. This then induces a death spiral in users as the content begins to stagnate and more users leave because of the other users leaving. This then concludes in the platform only being left with a skeleton of its former self and it can be nearly impossible to revive a platform after that. See the History of Myspace as a perfect example of this. A more interesting platform appeared, and a clear majority of the users quickly migrated. *(Wikipedia-Myspace 2017)*

So, to improve user retention the application requires something else. An extra system needs to be developed and deployed to attract new users and keep the old, invested, users around. This system could be based around the theory of Gamification. Gamification is the process of adding features and elements of games to non-game tasks. This has been shown to increase user retention as well as increasing the chance of attracting users in the first place *(Luis von Ahn 2008)*. There are several ways of implementing game features and these ways can have very levels of effectiveness based on how they are implemented and for what purpose. Luis spoke about user ranking in his 2008 magazine article. This is the process of taking a user's input then quantifying that contribution in terms of effort and quality and giving them a score. That score can then be applied to ranks and leader boards *(Luis von Ahn 2008)*. This could be a brilliant feature for my site as it can be applied to both the searchers and the posters.

Posters can be awarded based on the communities' evaluation of their post. Searchers can then be awarded for providing frequent comments and rating on these posts. The level of reward would have to be based on level of contribution as if users are rewarded for simple tasks such as giving rating they have been shown to give more spurious ratings *(Zwass 2010)*. So, a smaller score could be given when a user rates and a larger score given to high quality posts. This score can then be acquired over the user's lifespan and as they reach certain milestones they can "rank up". This is a badge they can display in comment boards as well as their profile page. It can be deduced that a user who has spent a long time acquiring a high rank will be less likely to leave the site.

With the techniques of Crowdsourcing and Gamification combined the syntax tool should be able to gather a large data set of learning resources and then maintain a steady stream of new data as required. Users will be encouraged to join in and stay around because of the community of fellow posters and researchers who can communicate in comment sections. They can then strive together to gain more points by posting and completing tasks that benefit the site such as creating posts and rating effective content. This in turn will then attract new users to the site due to the range of high quality content. If all goes to plan it should be steady upwards spiral of user numbers as the invested community rises thanks to word of mouth and recommendations.

9.5. Appendix E: Project Requirement Review

Project Requirement	State of Completion
FR-1	Users can create tutorial posts – Black Box test series 1 tests the ability to create and manage posts from a user level. It was found during these tests that it is in fact possible to create posts as a user and to save those to the database as proved by the Database Verification in chapter 5.2.3. However, this feature could be improved by implementing a more in-depth text editor and allowing more advanced formatting. At the current moment the user only has access to the most basic of aspects of text formatting. Allowing the users to use more advanced formatting features would open the system up to abuse as users could create malicious posts using edge case formatting. For example, they could create posts with oversized text that used up a large amount of screen space. So, if additional formatting features were added to the text editor care would have to be taken to ensure users could not abuse the additional power. On the other hand, it would give users the ability to express themselves more and to create more interesting posts. This then improves the application by making it more interesting and less uniform which should help to retain users and improve the create-view life cycle as more users stay connected.
FR-2	Users can tag posts to their current post – Black Box series 2 tests the ability to tag posts to the currently created post at a user level. The tests prove that this functionality works though the overall effectiveness of this feature can only be gauged by the final audience survey. It yet to be seen if the users would prefer the currently implemented style of post tagging or if they would prefer a different style of post tagging such as simply including links in the post body. *Complete rest of once audience survey is completed.*
FR-3	Users can view other posts and rate specific attributes on them – Black Box series 3 tests show that users can both view and vote on specific attributes of a post. One-way Codex differs from its mainstream competitors is by allowing users to vote on attributes of a post rather than just assigning a general rank to that post. This is done to allow the users additional control over the types of content they want to see. Different users learn in different way as shown by the audience survey in chapter 2.2. Different users use tools in different ways and by allowing them to search for content that hold values they prefer more should increase overall user satisfaction. Additional criteria may also be added in the future depending on the results of the final audience survey in chapter 6.1
FR-4	Users can search for other user's posts based on specific criteria – Black Box series 4 tests show that users can search for other posts via the language, author name and post title. They can then sort that content based on the various attributes that post holds. Linking to FR-3 the idea that posts holding multiple attributes that can all be rated separately and sorted separately allows users more control over the types of

	content they want to see. The Market Research in chapter 2.1 shows that other competitors while doing similar features have not exactly implemented this style of feature. Sites such as Reddit allow users to filter posts based on “controversial” or “hot” but these phrases are somewhat vague to the user. While the average user can assume what these phrases mean it is not 100% obvious. By assigning a number to specific attributes like Codex does users can easily understand how posts are ranked and they can then easily search for content that appeals to them. All of this is builds up to an experience where the user can easily find the content that most appeals to them. When the site builds up a reliable community of content creators it would be completely possible for two content creators to create a post about the same content but in different ways that embody different attributes. Users then have the choice of which post to view based on the qualities they prefer. Once again, based on the result of the final audience survey in chapter 6.1 attributes may be added or removed depending on their popularity with the testers.
FR-5	Users can create bounties, view other bounties, rate other bounties and mark their own bounty as complete – Black Box series 5 tests prove that this feature has been implemented to an acceptable level. Users can perform all the actions outlined in the functional requirement. The bounty board/request feature was added because of the initial audience research detailed in chapter 2.2. Users noted that they often like the ability to be able to view problems next to solutions to help them learn and they like the ability to being able to request tutorials be made. While Codex does not aim to simply provide solutions to given problems as the hypothesis behind the app states that one can improve their learning by increasing the semantical knowledge. So, the app itself is more focused on learning the theory behind programming structures and concepts rather than simply solving issues. Though this feature can help direct content creators to areas that the user base is most interested in therefore helping to increase user retention. Which in turn helps to keep the platform alive as it holds a healthy user base who requests, views and creates content.
FR-6	Users can view their own posts and delete them – Black Box series 6 tests shows that the user can view and delete their own posts. While this is a simple function it can be helpful to users who may want to update and change their content as they learn more and progress with their own studies. Giving the users the ability to manage their own content will help to keep the site updated and to remove out dated content. It’s important to ensure that content is as up to date as possible and through the ability to delete posts as well as upvote and downvote posts the user base should help to ensure that content that is factually correct rises to the top. Meanwhile, content that is incorrect or outdated is removed or falls from the top of the searches. Having a site full of out dated content can be a real problem as it can be stated that users may leave a site if it is full of outed or otherwise useless content. Which stated in the crowdsourcing research in chapter 2.4.2 can be detrimental for a crowd sourced website. If the community leaves then the site will most likely lose popularity very quickly as content creators will move to other platforms if there is no audience and the audience will leave if there is no new content.

FR-7	Users can save posts for offline viewing - Black Box test series 7 proves that this functionality has been implemented. Like some of the other functional reality the popularity and effectiveness of this feature will be mostly judged from the user feedback from the final survey in chapter 6.1. Offline post caching is simply another feature for the users to make use of. It also doubles up as a manor for users to keep track of posts they may want to view later. By adding it to the saved list they can easily find that post later and view it. Or if it is a post they frequently ready it can also be helpful to keep it in an easy to find place. Creating simple quality of life improvements like offline caching is an example of polish and it can help improve the user retention on a site. As found in the Market Research of chapter 2.1 big sites like Stack Overflow have dedicate development teams who's focus is to provide an *enjoyable* user experience. This is achieved by making it as easy as possible for users to find and create the content they want. In turn this might be one of the reasons these types of sites become the most used sites compared to their peers. Is the accumulation of small features they simply make the users time on the site easier and faster?
FR-8	Users can comment on posts – Black Box test series 8 confirms that users are, in fact, able to post comments on specific posts. Comments and discussion can be a great way to add additional content to a topic that the original poster may have missed. It can also be a great avenue in which to share additional content and to help the community as a whole's learning. The Market Research shows that all these popular sites have some form of comments or discussion section attached to every post so there must be some reason for this. It could possibly be another way these sites build up a community among the users. Having the users communicate with one another helps to create this sense of community as it isn't just faceless posts taking up the entire site. People can converse with one another and in turn this may help to lead to more detailed and better-rounded post. Which finally leads to better posts as the users can help one another learn and with that information new posts can be created with that new content. The academic research on crowd sourcing shows that communities that work together towards a common goal tend to produce higher quality products. So, the comments facilitate that line of communication and betterment of the content.
FR-9	Code will be thoroughly documented. This functional requirement is a little more difficult to verify than the others as thorough documentation can differ from developer to developer. The primary reason behind documenting code bases is to ensure that the developer and future developers can understand how an application fundamentally works so that it may be maintained and upgraded in the future. In code comments have been made throughout the entire code base in areas of complexity of core functionality. These comments help to explain the importance of specific snippets so that any future developers can understand why certain design decisions were made and how the system works

	This Dissertation paper also services as documentation (see chapter 4) to how the system works from a high-level perspective. So, in a way this functional requirement has been met though it can be said more documentation can always be made and it is completely dependent on the company or developer what level of documentation is acceptable.
FR-10	User manual will be created *User manual is here check the attached materials* The website should be easy to learn by itself but lending a helping hand to users can always be useful. This ensures that all users can begin to use the site which helps increase numbers. Those numbers will then hopefully simulate the growth of the application through the consumption and creation of new content. This also helps NFR-1 by making the website easier to use by giving an explicit user manual.

Table 8: Requirement Review

9.6. Appendix F: Focus Group Reports

Date	No of Participants	Lead Researcher	Platform Tested on
16/04/2018	2	Thomas Hutchinson	Android – Google Chrome 64 Bit Version

Outline of Session

- Started with providing the participants with the URL to the Codex website. From here the participants logged into the system using their own credentials.
- Participants where then instructed to go from page to page testing any functionality. This instruction was intentionally left vague to see how users interact with the site under little instruction.
- When the users had questions, they were answered but as little instruction or direction was given as to not influence their opinion or time with the application.
- If the users had a question about how to operate the application, they were directed to the user manual.
- When all the participants felt as if they had fully used the application the testing phase of the focus group was over.
- This then led to the discussion phase. Here the participants were asked questions about specific features of the application that directly linked into proving or disproving the initial hypothesis. These questions included:
    - What is your opinion on the theory-based posts and solutions? Do you prefer posts that contain many examples and less explanation or posts that contain less examples but a greater deal of explanation about the examples and any related content?
    - What features of the application would make you more interested in creating posts for the sites?
    - Would you use the site once it is finished? What feature or properties of a site make you want to use it frequently.
    - How effective is the comments section in getting you involved with the community and directly the direction of content posts?
- These questions were rolled into a free form discussion where the participants could ask counter questions about the system though once again minimal description of the final goal of Codex was given to avoid altering the opinion of the participants.

General Feedback

- Colour scheme is unappalling *(mentioned multiple times)*
- Can be difficult to read the title text found on the navigation bar and app toolbar. A change of colour would be required.
- Site contains several spelling mistakes which makes it seem unprofessional.
- Request/bounty submission is good as it encourages communication between users helping to develop a community and keep the content that people want to see appearing.
- It would be good if users could filter their feed.
- Another possibly good feature would be difficult attributes on post. This attribute could be set both by the author and voted on by the community. This attribute would then describe how difficult the post could be to learn. This can then link into the post tagging system. Those tagged posts could then act as prerequisites for users to go learn first before they tackle a specific topic.
- Saved Posts are a good feature as it lets users come back and revise a specific topic repeatedly until they learn it.
- Manual is another helpful feature though it might be good to have two-way links. Meaning that the manual itself links to the areas it is explaining and that there is some manner for

users to quickly be sent to a specific part of the manual from the area they are trying to operate.

Main Area of Discussion

- Participants noted the important of a wide range of content must be present for them to become invested in the website. On top of that content must be regularly updated. Without those two properties it would be highly unlikely they would use the site for any extended period.
    - In response to this the participants were questioned about what is more important. A smaller amount of high quality posts or a large range of acceptable quality posts. By quality it was referred to as having clear examples and some explanation about those examples.
    - All participants agreed that having a wider range of possibly lower quality posts would be far more preferred than a few higher quality posts. This is because during developer they often must learn many new elements within a short span of time. Sometimes they do not have time to commit to a lengthy post that goes into the background of a feature or semantic. Meaning they would just prefer a clear example of how to use the feature. They may later then go back and learn more about the background if it interested them.
    - What was more important though was finding the content they needed at a level clear enough for them to at least implement the solution. So, in its current conception Codex would be more useful as a "on the side" learning tool rather than a dedicated platform. For Codex to become more useful it would also have to encourage more example-based posts.
    - It was suggested that maybe the example could be given at the top of the post then the user could scroll down for the background and thorough explanation. This could then be incorporated into a viewing history system. That way a user could implement a solution quickly and then review the semantic later when they are not against the clock.

Conclusion

- It will require corroborating with the audience survey to ensure that this focus group wasn't anomalous, but it seems that the content quality has little effect on user retention after a certain threshold of quality. That threshold is having clear examples that are mostly system independent so that the user can quickly copy and paste the solution into their own project.
- While the participants see the advantage of a more theory-based system that thoroughly explains topics rather than just supplying examples it simply wouldn't fit into their busy schedule.
- Review final audience survey to gauge opinion on the importance of example solutions.

Table 9: Focus Group 1 Report

9.7. Appendix G: Glossary

Word	Definition
Userbase	Refers to a specific target demographic at which the application is targeted for. For the case of Codex, the general userbase is Full-Time University Students.
Web Application	A website that aims to provide similar functionality to that of a native application. The site will also use local memory to retain the users current “session”
Backend	Refers to the server structure running the application. This encompasses the hosting and database management.

Table 10: Glossary

10. Supplementary Material

The supplementary material for this dissertation includes:

- 2 Excel Documents containing the full results from the 1st and 2nd audience surveys
- The file base for the final Codex Application. This includes all dependencies and all written code.
    - A live version of the site can be found at: https://codex-8eb0b.firebaseapp.com/ *(As of writing: 22/04/18)*
- 2 .mpp files containing the final updated project plan and final application plan
- 1 .PDF document containing the user manual.
- 1 .zip file containing all the testing screen shot images from the Black Box Testing Phase.