

Analysing the Usability of a Single Page Application created with React

by James Bullock

FILE	18284_JAMES_BULLOCK_ANALYSING_THE_USABILITY_OF_A_SINGLE_PAGE_APPLICATION_CREATED_WITH_REACT_78196_1544328024.DOCX (9.74M)		
TIME SUBMITTED	20-APR-2016 04:12PM	WORD COUNT	19755
SUBMISSION ID	56135989	CHARACTER COUNT	124583



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Analysing the Usability of a Single Page Application created with React

By James Bullock

Supervisor: Kemi Ademoye
Second Marker: Mike Dacey

Project submitted as part of the requirements for the award
of BSc(Hons) Web Development

20th April 2016

Declaration Form

I, ... declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Signed :

Date :

Form of Consent

I hereby consent that my Project, submitted in candidature for the B.Sc.(Hons) Web Development degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Signed :

Date :

Abstract

The aim of this project is to Analyse and evaluate the Usability of a Single Page Application created with React. The objectives include investigating JavaScript MVC's, researching React and alternative frameworks for creating SPA's along with any testing techniques, creating a survey to determine a list of requirements, developing a functioning SPA using React and finally, testing and evaluating the usability of the application using various methods.

In general, the findings are extremely promising for application development using React, the response towards SPA's from the conducted survey was generally popular, the individual questionnaires once the application had been tested were also very positive. Aside from a few shortcomings here and there, overall the project was a relative success and when combined with the research of the relative subjects, the outlook on React as a SPA development framework is incredibly encouraging.

Contents

1. Introduction	1
2. Literature Review	2
2.1. Single-Page Applications	2
2.1.1. Examples of Popular Single-Page Applications	4
2.1.1.1. Discord	4
2.1.1.2. Gmail	5
2.1.1.3. Outlook	6
2.1.1.4. Dropbox	7
2.2. JavaScript Model View Controllers (MVC)	8
2.2.1. React	10
2.2.2. Alternative JavaScript MVC Frameworks	13
2.2.2.1. Node.js	13
2.2.2.2. AngularJS	14
2.2.2.3. Ember	16
2.2.2.4. Backbone.js	17
2.2.2.5. Popularity Trends Comparison	18
2.3. Potential Methods of Increasing Usability	19
2.3.1. React Native	20
2.3.2. Flux	22
2.4. Usability and Testing Methods	24
2.4.1. Usability	24
2.4.2. Testing	27
2.4.2.1. White Box and Black Box Testing	28
2.4.2.2. Unit Tests	28
2.4.2.3. Integration Tests	29
2.4.2.4. System Tests	30
2.4.2.5. Acceptance Tests	30
2.4.3. JavaScript Testing	31
2.4.3.1. Jest	31
2.4.3.2. Mocha	32
2.4.3.3. Karma	33
2.4.3.4. Jasmine	34
2.4.3.5. QUnit	35
2.4.3.6. Chosen Testing Framework	35

2.5. Conclusion.....	37
3. Research and Development Methodology	38
3.1. Research Methodology	38
3.1.1. SPA Usability Survey	40
3.1.2. Testing Feedback Questionnaires	42
3.2. Development Methodology	43
4. Design.....	47
4.1. Paper Prototype	48
4.2. Basic Prototype	51
4.3. Final Design Phase	51
5. Implementation.....	52
5.1. Use Case Diagram	52
5.2. Code Implementation	54
5.3. Software Testing	56
5.3.1. Unit Testing	56
5.3.2. System Testing	56
5.3.3. Usability Testing	58
5.3.4. Acceptance Testing	58
6. Evaluation.....	59
6.1. SPA Usability Requirements Survey Results	59
6.2. Summary	71
7. Conclusions and Recommendations.....	71
8. Reflection	74
9. References	76
10. Bibliography	86
Appendix 1: Testing Feedback Questionnaires.....	89
Appendix 2: Original Project Prototype	93
Appendix 3: List of Requirements	96
Functional Requirements.....	96
Non-Functional Requirements.....	96

1. Introduction

The overall aim of this project is to Analyse and Evaluate the Usability of a Single Page Application created with React. As for the objectives, the first objective is to conduct Research in order to evaluate the concept of JavaScript MVC's, React and a selection of popular alternative JavaScript frameworks. An investigation into conducting both Usability tests and Unit tests for SPA's will also be directed along with a large scale survey in order to determine a list of requirements for the project.

Once these Objectives have been met, a SPA will be developed using the React framework, along with conducting Usability and Unit tests on the created application in order to fully determine the benefit of using React over other frameworks or methods. Finally, a full evaluation of the Usability and overall Functionality of the application will be conducted by providing a selection of testers with a face-to-face questionnaire once the application has been tested.

2. Literature Review

This literature review will make use of assembled research to provide detailed information regarding *Single Page Applications*, *JavaScript MVC's*, *JavaScript Frameworks*, *Potential methods of increasing Usability*, *Usability and Testing methods* and finally *JavaScript Testing methods*.

2.1. Single-Page Applications

Single-Page Applications (SPAs) are becoming more and more popular due to the introduction of smart phones, tablets and other forms of mobile devices capable of accessing the internet, anywhere and at any time [1]. The popularity of SPAs is growing so much so, that the technology is considered to be one of the most important web development technology trends of 2015 and early 2016 [2][3][4].

A SPA is a Web Application that loads a single *Hyper Text Markup Language* (HTML) [5] page offering various features and advanced functionality at the press of a button, without leaving the original page [6]. These pages utilise *Asynchronous JavaScript and XML* (AJAX) [7] and HTML5 to create a fluid, responsive and modernized Web App, minimizing the loading times of the web page whilst being extremely flexible and diverse with the content on offer depending on the complexity of the created application [6].

In February 2005, Garrett [8] devised the name AJAX after the technology had existed for a few years beforehand, this led onto more people adopting the term and further utilising the technology in websites and Web Apps henceforth [8][9]. Google [10] initially pioneered the development of AJAX by investing heavily in the technology and creating multiple, extremely popular applications using the technology from relatively simple to extremely advanced [8].

Although AJAX utilises *EXtensible Markup Language* (XML) [11], creating a SPA does not require an understanding of XML, as the apps are coded in either JavaScript, or a Framework

specific version of JavaScript [7][9]. A SPA created using AJAX is composed of numerous components which can be manipulated in real time independently based on user input, removing the limitation of having to load a new page at each button press [12]. This allows the Application to be used across all devices, increasing the responsiveness, whilst allowing users with limited connectivity or data allowances to use the Web App without any hassle, promoting interactivity and user satisfaction [9][12].

The development of most SPAs is achieved via the front-end of the application, as opposed to traditional web applications that use server-side interactions and engines to achieve the same effect [1]. One of the main reasons for the popularity of SPAs is that they resemble desktop applications as opposed to full websites [1][13]. This again is further supported by the fact that more and more well-known companies are adopting a SPA approach when it comes to designing popular Web Applications for day-to-day uses such as communication, e-commerce, streaming content and viewing news or information [1][13].

2.1.1. Examples of Popular Single-Page Applications

There are a large number of popular websites that are Single Page Applications, the majority of daily internet users will not necessarily realise it, but a number of their favourite sites are most likely SPA's or modelled on the design principles of SPA's [14].

2.1.1.1. Discord

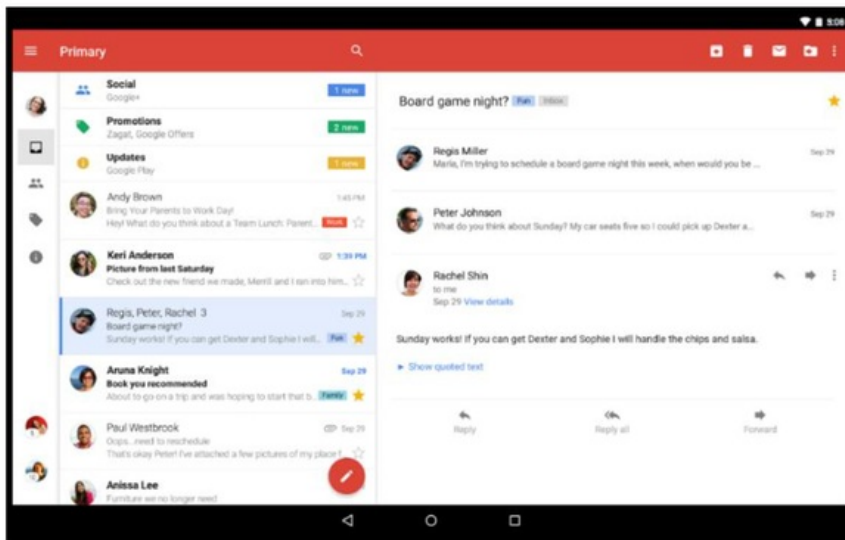


[15]

Discord [16] was founded by Jason Citron of *Hammer & Chisel Inc.* [17] in early 2015 [18] with the idea of being a revolutionary all-in-one text and voice chat application for both regular users and gamers. The application is free, secure and is also available across a large number of platforms including the web, desktop and mobile devices amongst others [16].

Numerous features are available to Discord users with the ability to communicate through both text and speech with friends and family using a high quality, low-latency and low-power application. It is available via web browser or as a downloadable application for multiple devices [16].

2.1.1.2. Gmail

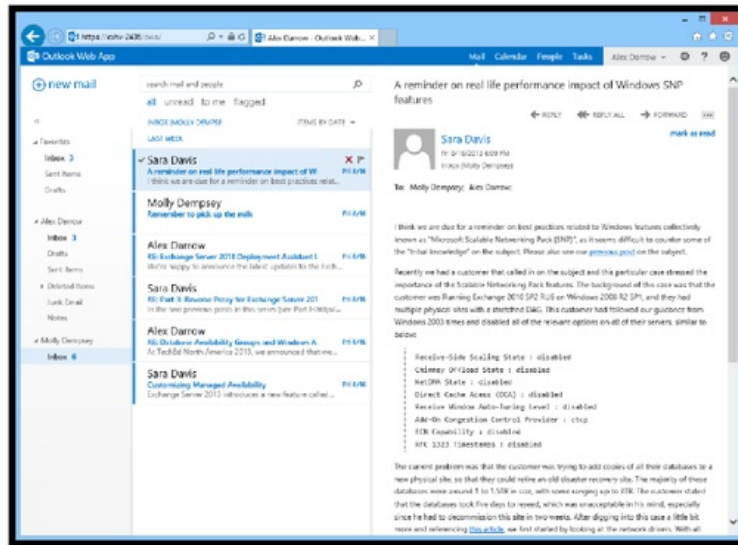


[19]

Gmail [20] is a free email service provided by Google [10], originally developed in 2004 [21], it was officially released in July 2009 along with a number of Google developed applications as part of the *Google Apps* [22] suite [23]. The list of features available in Gmail include sending, managing and receiving emails, a list of contacts, notes and junk email filtering [20].

The *Google Apps* suite includes a range of popular Google developed applications, such as: *Gmail*, *Calender*, *Maps*, *Google+*, *Drive*, *Docs*, *Forms* & *Sheets* [22].

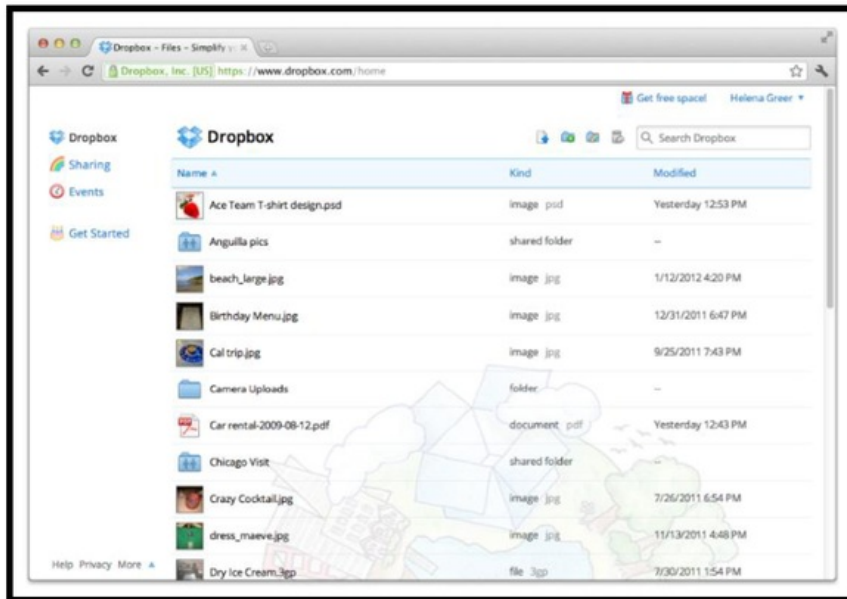
2.1.1.3. Outlook



[24]

Outlook [25] is another free email service with similar features to Gmail developed by Microsoft [26], originally named *Microsoft Mail*, Outlook was added to the Microsoft Office suite Version 4 in November 1993 [27]. The name was changed to Outlook in January 1997 upon the release of Office 97, which became the fastest-selling business application in PC history [27]. The latest iteration of the Microsoft Office suite of applications contains further SPAs such as: Office Online & Mobile, OneNote & SharePoint [28].

2.1.1.4. Dropbox



[29]

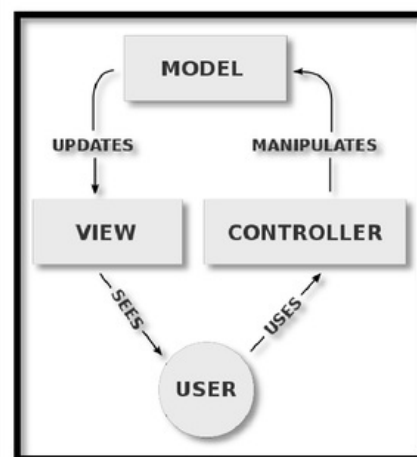
Dropbox [30] is a file hosting and sharing service created by Dropbox Inc. in 2007 [31], allowing registered users to upload documents from a PC, Laptop or mobile device into a shared folder accessible by anyone logged into the same account with Dropbox installed [30]. Dropbox is incredibly popular throughout the world, to the extent that it was named as the world's fifth most valuable startup company in 2011 [32][33].

2.2. JavaScript Model View Controllers (MVC)

The term **Model View Controller** (MVC) [34][35] refers to a software architectural pattern used primarily for building user interfaces [36]. An MVC is made up of three components, a *Model*, a *View* and a *Controller*, each component works in tandem with one another and are manipulated or altered by a user's actions [37][38]. The *Model* is considered as back-end data or code, the *View* is a visual representation of the data from the *Model* that is displayed to the *User*, and the *Controller* is any actions that are taken based on the users' interaction with the application [39].

As is evident from the MVC Task Process shown in *Figure 1*, the *User* is able to choose what is displayed on the application by using a *Controller* to dictate specific functions, for example, clicking a hyperlink or button, pressing a keyboard shortcut, opening and closing a menu or initiating an animation playback [41][42]. When the *User* activates a *Controller*, the *Controller* processes the interaction and manipulates the specific *Model* in the

Model View Controller Task Process (Figure 1)



[40]

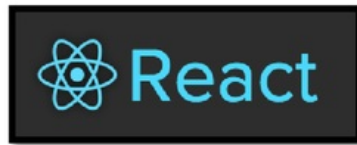
back end of the application, which then updates the *View* displaying the desired outcome [38][41]. This means that the *User* interacts with the Application solely by *Controllers* [35].

There are a large number of JavaScript MVC Frameworks available currently, with the list growing even more in recent years due to the increase in accessibility, flexibility and ease-of-use of these Frameworks. Use of the MVC architecture over standard JavaScript helps promote a more organized infrastructure, increasing the overall standard of JavaScript applications and allowing for maintenance to be conducted far more easily [43]. On top of this, MVC Frameworks allow the use of advanced *Framework-specific* features with the use of a single command line tailor-made by the Frameworks' developers. This opens doors to

less-experienced web developers allowing them to create complicated applications with relative ease [44][45].

The major advantage of using a JavaScript MVC to create a Single-Page Application comes from the flexibility of the Frameworks on offer. As stated by Mesbah & Deurson [12], one of the most important aspects of a SPA is the ability to display numerous different features on the same page, without the need for further page loading [12]. This means that the use of an MVC design pattern is heavily evident in SPA's, as the View is allowed to change based on the User's interaction with the application [37][38]. This allows a Single Web page to contain a vast amount of different features and content depending on the complexity of the design of the application [6].

2.2.1. React



[46]

React, also referred to as *React.js* or *ReactJS*, is an open-source JavaScript Library for building User Interfaces [47][48]. Created by Jordan Walke, a software engineer working at Facebook [49] in 2013 [50], React was designed with the intent of moving the technology of SPAs forward and creating a solid design framework that allowed developers of various skill levels to create complex SPAs [51][52]. The original design influence came from XHP [53], a PHP-based component system utilising XML, used by Facebook [50][54].

As React is owned by Facebook, it is maintained and updated by both Facebook, Instagram and a community of corporations and individuals [55][56]. Whilst being constantly updated and worked on by these developers, React and React Native have steadily established themselves as Facebook's top two open-source projects [57]. To add to this, React is currently being used on a huge number of popular homepages, including *Netflix*, *BBC*, *Facebook*, *Instagram*, *Imgur* & *IMDb* [58].

While React could be mistaken for an MVC Framework, it is in-fact differentiated from MVC Frameworks in that it is a library for building User Interfaces (UI), encouraging the creation of reusable UI components which present data that changes based on user interaction over time [51][52]. This means that real time adjustments can be made to the displayed data as each time the data is changed, a method is called, updating the contents of the application with the updated data [48][52].

Some of React's most notable features are the use of One-way data flow, Virtual Document Object Model (DOM) [59], the JSX syntax extension and the ability to utilise React Native, the native application oriented extension of React [47][60].

Most Frameworks utilise Two-way data binding, where user interaction, such as entering information into a form, updates the application upon each letter entered into the input box [61]. This can cause possible errors both during the development process, and if the Framework is updated [61]. By using One-way data flow, React avoids any potential bugs, scalability challenges or performance issues, whilst still allowing developers to achieve this functionality with a little extra programming [47][61].

React utilizes a feature called Virtual Document Object Model (DOM) which it manages instead of monitoring and updating the browser's DOM on the fly [62][63]. This is due to the Virtual DOM being comprised of data instead of actual rendered content, which allows React to re-render individual changes as opposed to the entire application [62]. This does not just benefit the performance of the application, but also provides a simpler programming model and guaranteed support across all browser types [47][63].

While React is extremely powerful on the client side, both client and server side rendering are supported when using React in conjunction with Node.js [64] which is detailed in *Chapter 2.2.2.1*. This is due to the Virtual DOM being utilised instead of a browser to render content and in turn aids the overall performance of React applications by speeding up initial page loading [64][65].

Development with React is made very simple by the fact that the Programming language used is JavaScript, whilst developers will be able to program an application using JavaScript, they will also be able to use JSX [66] which was created by, and specifically for, React [52][60]. JSX is an optional syntax extension for JavaScript, visually similar to XML, giving web developers an option of using JavaScript that is more similar to the readability of HTML [48][52][66][67].

There are also extra libraries that can be utilised in order to either modify React in a preferred way, or output an application created with React across multiple platforms and device. Packaged with React is React Native, that enables developers to build applications on native platforms using a consistent development experience based on JavaScript and React [60][68]. React Native is again open-source and contains standard industry platform

components in order to keep both the quality of the application at a high level, and the look and feel of the application consistent to the platform operating system [68][69].

The fact that JavaScript or JSX is used as a programming language across both React and React Native allows for a more flexible design and maintenance process. As well as this, React and React Native adopt the 'learn once, write anywhere' motto [48][52]. This means that once a developer has a reasonable understanding of the design language, for example if an application is created using React Native, the development process of creating an application in React will be much easier than learning a new programming method or language from scratch [68][69][70].

Testing with React is also an importantly regarded design philosophy amongst the developers of React, thus *React Developer Tools* was created [71]. The React Developer Tools have been built using React, which allows for an easier application development process and the ability to extend the tools to gain additional important information during the testing process [48][65]. This means that the development period will not only be shorter, but the overall structure of the code within the application will be more robust [48][52][65].

Additional testing methods are available to React through numerous JavaScript testing frameworks, along with including its own built-in test utility '*ReactTestUtils*', the developers of React suggest the use of *Jest* [72] to complete Unit Tests of individual functions of a React application [73]. However, due to the flexibility of both React as a JavaScript framework, and JavaScript as a coding language, developers are free to use whichever form of JavaScript testing framework they prefer [73][74].

Overall, React and React Native bring a new dimension and level of professionalism to the design of both single-page and native application development [48][52][60]. The fact that the technology is backed by a global name as renowned as Facebook also means that the growth of the technologies will only continue to move forwards, with the likelihood of it becoming the front-runner in both Web and Native Application design a very real possibility [75].

2.2.2. Alternative JavaScript MVC Frameworks

2.2.2.1. Node.js



Node.js [76], also known as *Node*, is a JavaScript runtime build utilising Google Developer's Chrome V8 JavaScript engine [77][78]. Node.js uses an event-driven, non-blocking I/O model making it lightweight and efficient [76]. Through the utilization of Node.js, developers can create Web servers and networking tools using JavaScript and a selection of modules that handle numerous core functionality using an API designed to simplify the process of writing server applications [79][80][81].

The first version of Node.js was created by Ryan Dahl and released to the public on the 27th of May 2009 [82][83], this version was only available to Linux platform users and was further maintained by Ryan Dahl with the backing of Joyent, a company specialized in application virtualization and cloud computing [84]. The framework evolved further during June 2011, Microsoft and Joyent came together to announce a partnership with the aim of successfully porting Node.js to Windows [85].

The inspiration behind the creation of Node.js came from Ryan Dahl encountering a demo on Flickr [86], where the progress bar displaying the completion status of the action on the webpage would indicate to the user how far along the current process was by constantly querying the web server [87]. This led to the concept of a JavaScript platform that incorporated non-blocking I/O in order to perform multiple tasks or requests at once [87].

Non-blocking I/O, or *Asynchronous I/O* is a form of input/output processing that allows other actions or requests that aren't dependent on the I/O operation to start or complete before the current request has completed [88][89]. This process allows the server side of a web application to push updates to the client without the need for a client-side query, which reduces memory usage whilst also improving latency and responsiveness, allowing the application to be output to a larger audience [89][90].

Both Chrome V8 and Node.js are primarily deployed using C and C++ [78] to provide increased performance benefits and lower memory consumption [83]. Due to the utilization of the JavaScript engine in Chrome V8, the overall performance of Node and any applications created with the framework is at a very high level, this is again helped by the majority of the features packaged within, or available as extra add-ons [83].

2.2.2.2. AngularJS



applications [93].

AngularJS [91], also referred to as *Angular* or *Angular.js*, is an open-source web application framework maintained by Google and a large community of developers and corporations [92]. AngularJS functions as a client-side MVC framework, similar to React, written in JavaScript to run in a web browser, allowing developers to create single-page, modernized, AJAX based web

Originally developed in September 2009 by Misko Hevery and Adam Abrons [93][94], as the software behind an online JSON storage service for easy-to-create applications. Later, both Hevery and Abrons decided to release Angular as an open-source library backed by Google [45], however, Abrons soon left the project leaving Hevery to develop and maintain the library with fellow Google employees and the help of a large community of developers and corporations [95]. Version 1.0 of AngularJS was released in June 2012 [93] and has since steadily grown with a future release of AngularJS 2.0 [96] scheduled for mid-2016 [97].

Whilst AngularJS is a relatively recent addition to the list of client-side MVC Frameworks, it has succeeded in propelling itself to the forefront due to its selection of unique features, including the use of two-way data bindings, an innovative HTML templating system, ease-of-use for both development & testing, extremely solid development practices, extending the HTML dialect by using directives and *Dependency Injection* [45][98].

AngularJS employs the *Dependency Injection* (DI) design pattern, which is a software design pattern that allows the framework to pass the service to the client rather than waiting for the client to find the service and utilise it [99]. This allows for a strong focus on the *Loose Coupling* [100] design approach, which promotes flexibility when adding to or manipulating the application, increased testability throughout the design process & more easy-to-read code [101]. Through the use of *Dependency Injection*, AngularJS brings traditionally server-side services, such as controllers reliant on the view, to client-side web applications, allowing for improved server-side performance [102][103].

As well as utilising the DI design pattern, AngularJS also uses HTML as the language for its templating system [102][104]. This allows for more design flexibility and a greater understanding for Web Developers during the design process, also meaning that HTML5 features are able to be incorporated into the application's interface [93][98]. Another HTML feature available to developers using the AngularJS Framework is the extensible components subsystem [98][103], allowing web browsers to be taught how to interpret new HTML tags, functions and attributes to futureproof created applications or allow the use of updated HTML features [102][104].

One of the final features bundled within the AngularJS Framework is the ability of tracking user actions, browser events and model changes to figure out when and which templates to use or refresh based on the action in process [93]. This reflects the *MVC* architectural pattern by having user interaction dictate the actions, or view, of the application [93][103]. The advantage of utilising this design method means that each interaction with the application doesn't necessarily require that the Document Object Memory (DOM) be refreshed [103][98]. This is achieved by the AngularJS Bootstrapper employing three phases after the DOM is initially loaded, the first phase being the creation of a new Injector, the second being the compilation of the directives that decorate the DOM and finally, the linking of all directives to scope [98][103][104][105].

2.2.2.3. Ember



Ember [106], or alternatively *Ember.js*, started life initially as SproutCore MVC Framework in 2007 developed by SproutIt [107][108]. The project was later picked up by Apple and lead by Yehuda Katz, a core contributor to the popular jQuery and 'Ruby on Rails' projects [45]. Katz, Y, co-author of the best-selling *jQuery in Action* [109] and *Rails 3 & 4 in Action* books [110][111], is joined on the Ember development team by Tom Dale who was previously part of the SproutCore team after leaving his post as a software engineer at *Apple* [95][112].

Ember 2.0 was released on 13th August 2015, with the mantra 'Stability without Stagnation', this included the introduction of the Glimmer rendering engine [113][114] aimed at improving render and re-render performance, and features such as One way Dataflow by default, automatic application refresh on any change within the application and standard lifecycle hooks for components [107][114][115].

There are specific types of applications that allow Ember to excel, examples of these are applications that dynamically display data and generally require a large amount of user interaction [95][107]. This list could include applications such as Dashboard Interfaces, Social Platforms, Instant Messaging services or Task Managers [107]. However, in contrast, Ember falters when it comes to static websites so should primarily be considered by developers with the intention of creating ambitious web applications [95][107][112].

There are a number of notable companies or organizations that employ Ember within their webpage or projects, such as *Twitch*, *Microsoft*, *Apple*, *Netflix*, *Vine*, *Linkedin*, *Yahoo!*, *Groupon* and *ZenDesk* [45][107][116].

2.2.2.4. Backbone.js

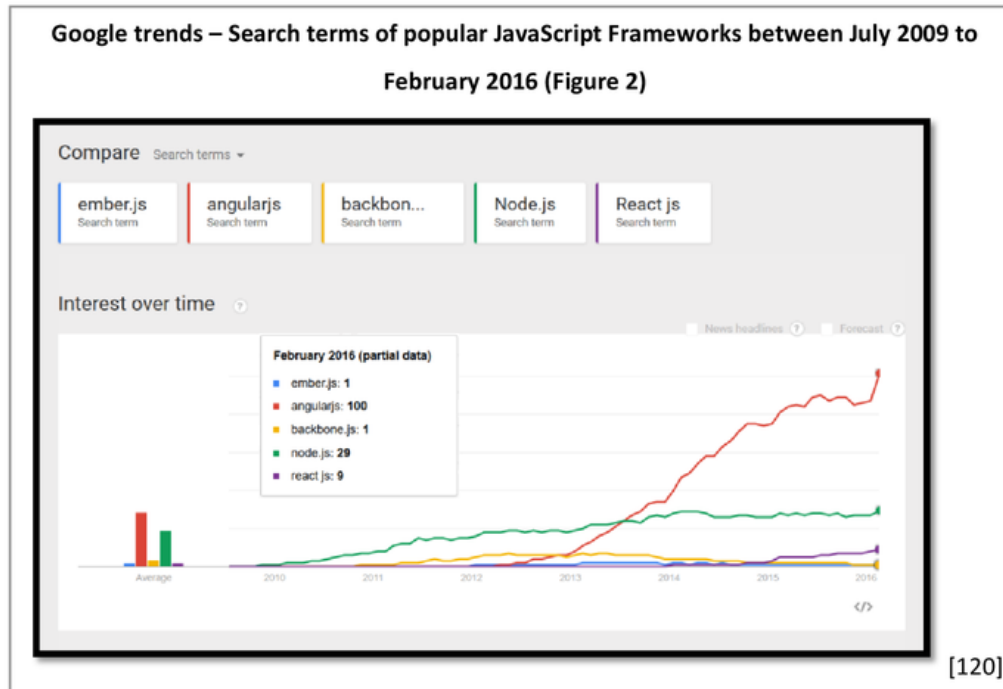


Backbone.js was originally created in 2010 by Jeremy Ashkenas [95][117][118], it is a lightweight JavaScript MVC Framework and

since its creation has been considered as one of the leading libraries available for Single Page Web Application development [118]. Ashkenas also developed both the *CoffeeScript* programming language, along with the *Underscore.js* JavaScript Utility library, which is the only dependency of *Backbone.js* [95][118]. These were both part of the original *DocumentCloud* code base which was an open-source project that allowed journalists to upload, annotate and share documents between each other [117][118].

The main source of the popularity of Backbone.js is due to the fact that it is a simpler alternative to feature-heavy, bloated JavaScript MVC frameworks [45][117]. This is also complemented by the availability of a large amount of extensions and add-ons providing more features when required, and a passionate community of developers propelling the framework forward [118][119]. The list of services utilising Backbone.js includes *Pinterest*, *SoundCloud*, *LinkedIn*, *AirBNB* and *Walmart* [45][117].

2.2.2.5. Popularity Trends Comparison



As is evident by the data displayed in *Figure 2*, the rapid rise in the number of times AngularJS was searched for using google since mid-2012 is impressive, especially when compared to its competitors, which in the case of Ember and Backbone have fallen off since the beginning of 2014. Node has been steadily rising in popularity since 2010, which is evident by the fairly stable increase in searches for the framework, the trajectory shown in the graph is similar to the recent surge in popularity of the React framework. This however is over a shorter timespan due to the relatively short period of time React has existed for.

2.3. Potential Methods of Increasing Usability

In an age of technology where according to *Chaffey*, the number of internet users on smart devices has overtaken that of the amount of users on desktops [121], the demand for scalability and accessibility is bigger than ever. It has become the norm to build applications and websites using the '*Mobile First*' approach in order to maximize the Usability of an application across all platforms [122].

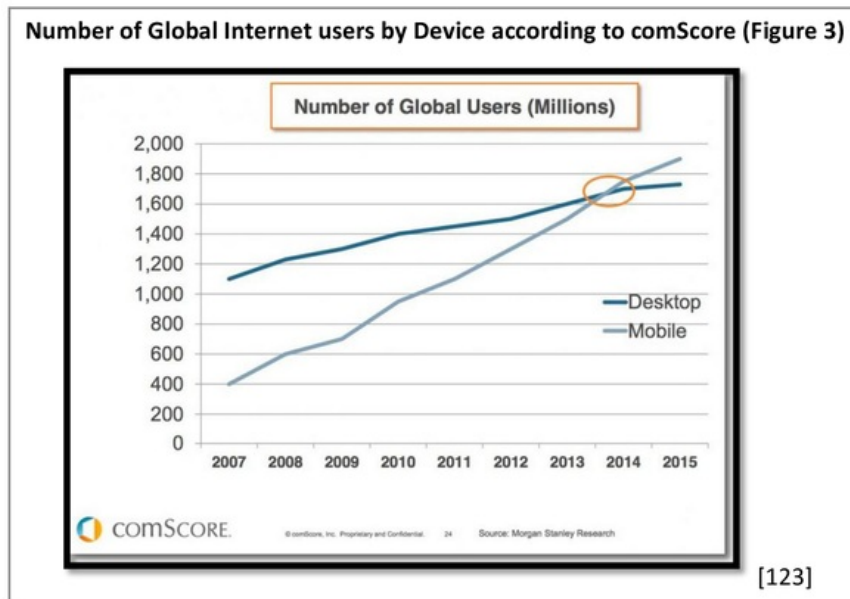
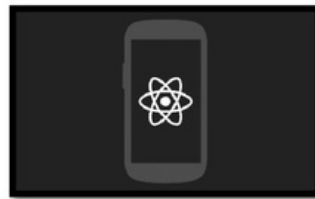


Figure 3 not only reinforces the fact that mobile's internet usage has overtaken that of desktop's, but also suggests that it will continue to rise at an impressive rate. In future application development it is becoming increasingly necessary to either build an application that scales to the requirements of the device, or has an alternative application specifically for mobile [124].

2.3.1. React Native



[125]

React Native [68] is essentially the Native Application oriented little brother of React, a JavaScript framework based on React designed to natively render iOS and Android mobile applications [124][126]. Similar to React, React Native was developed by Facebook and released publicly in January 2015 [127]. The primary goal of React Native is to provide an efficient framework for application developers, with the familiar look and feel of both JavaScript and React [60][68]. This is added to by the ability to further output an application across multiple popular platforms in order to increase the target audience of the application and promote an ever-growing user base [70][124][126].

React Native uses the same programming language as React, which is beneficial as the use of JSX across these two frameworks means that cross platform development is much easier, and does not require further learning of other framework-specific variations of JavaScript [48][126]. This is not the only benefit of making use of the same design language, as the motto '*learn once, write anywhere*' means that the use of JSX allows a React web application to not only be simply rewritten to an iOS native application, but also a native Android application [68][69][70].

As React Native is based on React, operations within an application are all conducted asynchronously, with the added bonus of the native modules utilising extra threads if necessary [68][124]. This allows for the possibility of added depth to a created application with the ability to conduct multiple complex actions, whilst hiding these actions behind the UI [124][126]. The fact that this is possible allows for a much more fluid and responsive user experience regardless of how complicated the functions of the application may be [68].

Another example of React Native catering to an application users' needs and attempting to create a positive user experience is the added ability for developers to include interface design elements based on the device or operating system [68][70]. Included within the React Native library are standard platform specific components such as `UITabBar` for Apple iOS and alternatively `Drawer` for Android. The implementation of these components within a created application will further add to the quality level and give the application a consistent feel to the device it is being used with [60][68].

An application created using React Native will also have access to the relevant devices feature APIs by utilising Objective-C for iOS and Java for Android [124][126]. This not only allows access to the previously mentioned Interface components `UITabBar` for iOS and `Drawer` for Android, but also allows the application to access platform specific features such as cameras, audio features, location tracking, and the accelerometer [60][124][126].

Due to the fact that both React and React Native include the *ReactTestUtils* library, testing is encouraged greatly through both the included library and additional testing frameworks if required [71]. The *React Developer Tools* created by Facebook are also an additional development lifeline, these tools support both Firefox [128] and Google Chrome [129] and offer full support for React Native [71].

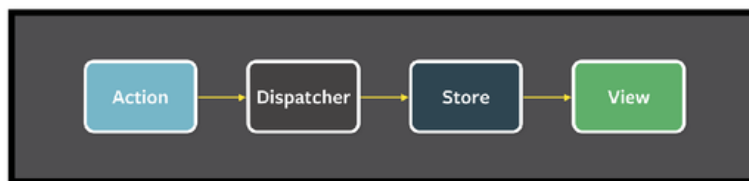
2.3.2. Flux



[130]

Flux [131] is considered an application architecture pattern as opposed to a framework or library [132][133], which means that developers are able to begin using Flux immediately without a large amount of additional coding required [131]. As with React and React Native, a team of developers at Facebook built Flux from the ground up and released it to the public in 2014 [132], it is utilised by developers to deliver scalable client-side applications and promotes cross-platform compatibility [60][131].

Example of the One-Way data flow model used by both React and Flux (Figure 4)



[134]

Contrasting React, Flux differs from the MVC architectural pattern in that it utilises three different components, the *Dispatcher*, *Stores* and *Views* [67][135]. However as is illustrated in *Figure 4*, similarly to React, Flux adopts the same One-Way data flow model which allows Flux to work particularly well with React due to both using a declarative programming style [60][131].

The heart of an application built containing Flux is made up of the Dispatcher, which dictates which action should be taken when a user interacts with a view, these views are made up of React components [133]. The chosen action is then sent to the corresponding

Store containing the relevant data or business logic, which in turn updates any matching views [132][131].

The fact that Flux does not follow the MVC architectural pattern does not mean that Controllers do not exist within a Flux application, they are just considered as '*Controller-Views*' [67][131]. These are generally the first components that receive data from the Stores and proceed to move this data on through the rest of the application. In order to assist this process, Dispatcher helper methods are implemented to support a semantic API and can be considered as the fourth update process of a Flux based application [60][133].

2.4. Usability and Testing Methods

Usability and **Testing** go hand in hand when it comes to both Software and Web Development [136]. Usability is the most important design aspect to consider when developing a project with an interface [137][138], the project will simply fail if its target audience encounters any issues with the Usability of the application [139]. In order to avoid this, Usability tests should be conducted thoroughly before, during and after the design process to determine and eradicate any issues that would affect the ease-of-use of the application [140].

As for Testing, along with testing an application's Usability, there are also various testing methods for an application's code and functionality [141]. There are two methods of application testing, White Box testing and Black Box testing [142], these variations were created in order to completely test every aspect of an application, from the external features to the individual internal components [136][142].

2.4.1. Usability

Generally, the **Usability** of a project or product is defined by its simplicity or how easy it is to use, this can be extended to Web Usability, which measures the ease-of-use of a website or application [138][143]. The Usability of a project is approached during both the design and development process by basing interaction on the prospective users, which means adhering to the requirements and preferences of the target audience [139][144].

Now that User Interfaces are such a major aspect of day-to-day lives, partly due to the introduction of smart devices, reduced prices of smart hardware and increased exposure to the internet and internet-enabled services, the target audience of development geared towards Usability is ever-growing and much more important [137][140].

Idler, states that there are a number of principles that can be adhered to in order to promote Web Usability [138][145][144]. The first of these is *Availability and Accessibility*, this principle is fairly broad as it includes points such as considering a users' disabilities,

internet connection speeds, screen resolution, broken or missing links and possible hosting issues [137][144]. In order to aid developers attempting to increase the Accessibility of their website or application, the World Wide Web Consortium (W3C) [146] developed the Web Accessibility Initiative (WAI) to '*lead the web to its full potential to be accessible, enabling people with disabilities to participate equally on the Web.*' – W3C [139][147]

A website or application designed for web accessibility should take into account possible disabilities or age-related ailments of users by providing alternative methods for accessing or interpreting content [137][148][147], along with this, a users' device resolution or connection speed should be catered for in order to allow mobile browsing or devices with lower resolutions and internet speeds to access the website with relative ease [144][149].

Learnability is a fairly important principle of web Usability as it dictates how easy or difficult it is to determine what actions should be taken by the user when using the website [144][149]. A websites *learnability* can be negatively impacted by poor design choices or an abnormal layout, this is largely why web design trends exist as developers adopt current popular design styles in order to avoid any issues that could arise from users struggling to understand the concepts implemented by developers [137][144].

Another principle mentioned by both Idler and Liebowitz is *Clarity*, which can be considered the core concept of Usability. *Clarity*, in this case, can be achieved by presenting information and possible choices in a clear and concise way, whilst displaying important features in suitable areas [144][149]. This principle can also refer to a website's readability, which is affected by the overall theme, colour palette, design layout or font choice throughout the project [139][144].

In an age of internet anonymity and abundant social media, *Credibility* is a huge principle to adhere to in order to provide users with a reason to trust a website and the developer or community behind it [150][144]. Providing potential users with a form of contact, either direct or through social media can help give a sense of trustworthiness and *credibility* to a website or company. In order to go even further, some information about the website or

company can also go a long way to giving a user piece of mind about interacting or frequenting the website in the future [138][144][149].

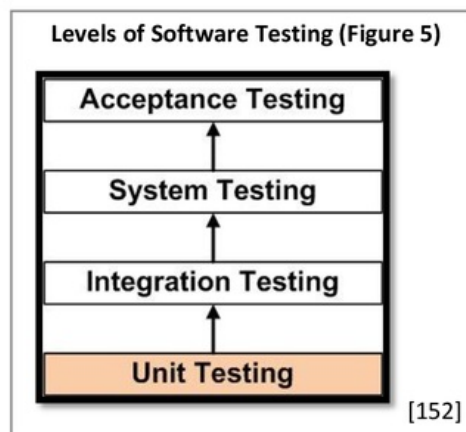
Finally, the *Relevancy* of a website is dictated by the target audience and how the content of the website relates to the requirements and interests of this demographic [138][144]. In order to find the *relevancy* of a potential website or application, the user base must be defined which allows a list of requirements and probable scenarios to be gathered [138][150]. This allows a developer to determine the actions that are required in order to create the most relevant project for the targeted audience [137][138][149].

2.4.2. Testing

Testing, as with *Usability*, is a fundamental process in both Software and Web Development, the testing process allows any development issues to be flagged and fixed prior to the release of the project into the public domain [136][137].

In the case of Software testing, written code is tested against a specification where-in any potential problems are flagged and presented to the developer [137]. There are numerous different methods of testing a created application, some of these methods also provide in-depth testing of specific areas to allow for a thorough overall test [137][136].

Each individual test is conducted by writing a selection of *assertions*, these dictate which values are expected to be returned and what format they should take [151]. If there is an error or the specified value or format type differs from the expected outcome, the test will fail due to the assertion not being met [151][141].



There is a logical order to Software Testing as displayed in *Figure 5*, which if followed, will provide developers with optimal feedback and important data regarding the functionality of the created application [141]. In order to test effectively and efficiently it is wise to break up a testing cycle into the recommended order, this order begins with **Unit Testing**, moves onto **Integration Testing**, followed by **System Testing**, and ending with **Acceptance Testing** [136][142].

2.4.2.1. White Box and Black Box Testing

There are two terms that are used to differentiate between these four sub-levels of testing, **White Box** and **Black Box** testing. Both *Unit* and *Integration* testing are methods of White Box testing, and alternatively, *System* and *Acceptance* tests are techniques of Black Box testing [142][141].

White Box testing is defined as a software testing method where the tester is aware of the structure, design or implementation of the test subject [142][141]. Sometimes referred to as open box, glass box or clear box testing, White Box testing examines the internal code or design of the project [141]. This testing method is generally applied at the lower levels of testing an application and is usually conducted by the developer of the project [142].

Alternatively, **Black Box** testing is a testing method that focuses on the external aspects of an application, this includes the specifications and requirements of the project [141][153]. Utilising the Black Box testing method means that the actual code of a program is not tested [141]. This in turn creates an issue when only using Black Box tests, as internal program errors can go unidentified, therefore requiring that White Box testing be also conducted in order to thoroughly test a project [142][153].

In order to successfully utilise all of these individual testing methods, tests have to be written for each individual aspect of the application, from individual units to overall requirements [137][136][151]. By passing each of these tests, a developer can determine whether or not an application could possibly succeed in achieving the goals that were set at the beginning of the project, and if the created application will be received positively by the target audience [137][151].

2.4.2.2. Unit Tests

Unit testing is a form of **White Box** testing and generally considered as the 'Go to' testing method for JavaScript applications as it provides a much more in-depth testing environment, allowing individual aspects of an application to be tested in great detail

[154][155][156]. The main aim of a Unit Test is to test *units*, which are the smallest testable parts of an application. These are generally functions or components, and during a Unit Test, these units are individually examined in order to determine any improper operations or errors [157][158][159].

Whilst *Unit testing* can be automated in some cases, the option to manually test is also available. This allows for added flexibility for developers, enabling the possibility of testing a single unit in multiple ways, to isolate even the smallest of problems. This testing mode is an element of Extreme Programming (XP), which is a practical development methodology that adopts a thorough approach to the design process by conducting constant testing [154][156][157].

2.4.2.3. Integration Tests

Integration Testing is essentially true to its name in the sense that it extends *Unit Testing* by combining two tested units to form a component and testing the way they integrate together. In the case of *Integration Testing*, which is another method of **White Box** testing, a component is made up of two or more units, meaning that it can vary from being a simple test between two units, or a more advanced test containing a large amount of units [160][161].

The end-goal of *integration testing* is to test the components made up of units with the incentive of larger scale testing between multiple components. This means that eventually all the modules of an application are tested together in numerous ways [160][161].

2.4.2.4. System Tests

The goal of **System Testing** is to evaluate how well the system in question fulfils the identified user requirements. Contrary to *Unit* and *Integration* tests, *System tests* are a form of **Black Box** testing and are used to determine how relevant the application is to the target audience and determine how to address any potential requirements that have not been met [137][162].

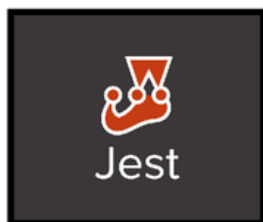
2.4.2.5. Acceptance Tests

The primary purpose of **Acceptance Testing** is to confirm that a scenario is able to be accomplished by matching a potential users' action to the desired outcome [163]. An additional extension to *system testing*, *Acceptance tests* which are another form of **Black Box** testing, are used to evaluate whether a system meets its required goals in order to determine whether it is acceptable for release to the user base [163][164].

2.4.3. JavaScript Testing

There are a large number of tools available to web developers in order to test a project created using JavaScript or any of its substantial range of Frameworks [165]. Among the most popular of these testing tools are **Jest** [72], **Mocha** [166], **Karma** [167], **Jasmine** [168] and **QUnit** [169].

2.4.3.1. Jest



[170]

Jest is considered one of the most popular JavaScript testing frameworks and is the recommended testing utility by the developers of React [67][74]. There is a long list of attractive features listed by Jest, some of these features include the fact that the framework is extremely flexible, modular, configurable and extendible [171][72]. This allows for the installation of further libraries or frameworks to access otherwise unavailable testing features [171]. Additionally, Jest utilises Jasmine assertions by default but also builds upon these with further assertions and the ability to add to and extend these assertions at will [72].

Further features of Jest include the ability to automatically find and run numerous tests at the same time across multiple workers. Additionally, Jest is able to automatically mock JavaScript modules, which allows testing of existing code without having to rewrite large parts of it [60][171]. Add to this the ability to include dependencies, developers are able to include complete libraries into tests in order to completely mimic functionality of the application during tests [72][171].

These are generally the more impressive features of Jest, however there are even more features, these include:

- Test files can be sandboxed and once the test is completed, will have their state reset automatically.
- Asynchronous code can be tested synchronously.
- It is possible to utilise a fake DOM in order to run specific tests.
- Jest works with any compile-to-JavaScript languages and incorporates Babel, a compiler that converts JSX to JavaScript, which is being heralded as the future of JavaScript [75].
- Jest can be configured to create coverage reports if required by the developer.

[72]

2.4.3.2. Mocha



[172]

Whilst not as popular amongst developers for testing React applications, **Mocha** is however, considered one of the strongest testing frameworks especially by *Node.js*, *AngularJS* and *Backbone.js* developers [104][173][174]. Mocha is a JavaScript testing framework that is very feature heavy, it runs on Node.js allowing for simple and flexible Asynchronous testing [166][173].

Tests can be run serially, which can help developers troubleshoot issues easily by giving accurate responses and reports [166][174]. One of the interesting features of Mocha is that it was designed to run in a browser, which removes the limitations that command prompts like Node.js can cause [166][174].

Amongst these features, Mocha also provides many more, these include:

- Mocha gives testers the ability to use any assertion library required, promoting extensibility.
- JavaScript API is included to allow tests to run smoothly.
- If a test is abnormally slow, Mocha will highlight it and display the duration of the test.
- If any uncaught exceptions are detected, Mocha will automatically direct them to the correct test cases.
- Bundled with Mocha are more than nine reporters to allow for extremely detailed test overviews.

[166]

2.4.3.3. Karma



[175]

Unlike other JavaScript testing frameworks, **Karma** is not a framework but a test runner [176], this means that developers are able to run tests automatically across multiple browsers [177]. Through entering one command, a large number of tests can be run to determine any failed tests and allow the tester to fix them quickly [176]. By implementing this method of testing code, Karma utilises a *Test-Driven Development* (TDD) methodology and promotes repeated automated testing in order to remove errors as quickly as possible [177].

Karma was originally named '*Testacular*' and created by Vojtech Jina [177][176], with the main objective of simplifying and speeding up the productive testing environment for developers [167]. The idea is to promote instant feedback in order to hasten the development process and promote the TDD methodology whilst being able to run tests across numerous devices and within various browsers [167][176]. Aside from these impressive features, further features are included and constantly added, these additional features include:

- Karma is completely open source due to being developed for and maintained by the GitHub community.
- Developers are able to control the entire testing process from the command prompt to an Integrated Development Environment (IDE), Karma will automatically run the tests if required.
- Copying a Karma developed test across to another testing framework is simple due to developers having the option to create a simple adapter.

[167]

2.4.3.4. Jasmine



[178]

Originally released in 2010 by a talented group of developers at Pivotal Labs [179][180], **Jasmine** does not rely on any JavaScript frameworks, DOMs or browsers making it ideal for virtually any instance where JavaScript is able to run [165][168].

Jasmine is considered as an '*All-in-One*' JavaScript testing framework as it comes bundled with extensive features without the need to add additional components or libraries [174][179]. Besides being a fairly new testing framework, Jasmine has become one of the frontrunners for JavaScript developers due to the *Behavior-Driven Development* (BDD) principles adopted throughout the framework [168][179].

The utilisation of the BDD development method within Jasmine allows for more precise choices or specifications, whilst also enabling for greater feedback in the early stages of development which could improve the overall polish of the project or application [181].

Example of a simple Jasmine test unit (Figure 6)

```
describe("Player", function() {
  describe("when song has been paused", function() {
    it("should indicate that the song is paused", function() {
      // ...
    });
  });
});
```

[179]

Figure 6 shows a simple Jasmine unit test on a *Player* function, by following a simple set of guidelines, tests can be written in a simple but concise way to reflect how they would actually be interpreted [179][181]. In this case, the **function** is a *Player*, **When** the song is paused, a '*song is paused*' indication **Should** be displayed [179].

2.4.3.5. QUnit



[182]

One of the first JavaScript testing frameworks ever developed, **QUnit** was originally created by John Resig, in tandem with jQuery [109]. It is a testing framework built towards simplicity, whilst still retaining the functionality and range of features available in other testing frameworks [169]. The concept of QUnit helped to influence the assertion methods defined in the CommonJS Unit Testing specification, which it in turn currently follows [109][183].

In 2009, QUnit branched off from jQuery and became a stand-alone framework, this was after an initial year of relying on jQuery in order to function [109]. Once QUnit became a standalone testing framework with its own API documentation, it allowed other JavaScript frameworks to make use of its testing abilities [169][183].

2.4.3.6. Chosen Testing Framework

Based on the research conducted into JavaScript testing frameworks, the Framework chosen for this project is **Jest**. The main reason for choosing Jest over the other available options is

partly due to React recommending it as their testing framework of choice [74], but also because the compatibility between React and Jest is far superior to that of the other testing frameworks mentioned [171]. After experimenting for a short period with Jest, this statement is validated as the integration with React and any required extension libraries makes unit testing much simpler than it initially seems.

2.5. Conclusion

In conclusion, React was chosen as the framework due to its impressive modularity and flexibility. The research reaffirmed this as there is an overwhelmingly positive response to React and its counterparts amongst the web development community [56][65][184]. Aside from this, the vast array of library extensions and open-source material available only adds to the attractiveness of the framework.

In addition, an interesting discovery was made whilst conducting research regarding methods of implementing a React application, this discovery is of a cloud-based backend data storage for web applications called Firebase [185]. Firebase was established by Google in 2014 and offers convenient features such as cloud-based data storage, built-in authentication and additional hosting packages for expansion [185]. This platform is perfect for the project and will be implemented into the final project as a powerful alternative method of storing necessary application data.

3. Research and Development Methodology

3.1. Research Methodology

The research methodology chosen for this project is a **Quantitative Research Methodology**, this method was chosen due to the fact that it utilises feedback and testing heavily, promoting good practice for both determining the Usability experience of the project and encouraging thorough Testing processes throughout the application's development lifetime [186][187].

Further reasons for Quantitative Research methodology being chosen are that the use of Questionnaires or Surveys are encouraged in order to gain a large range of feedback both before and after the actual development phase of the project [188]. This feedback can then be displayed in the form of a graph in order to give a visual representation of the data and the general weighting of specific answers or opinions from a large pool of responses.

Another positive reason to make use of this research methodology is the ability to reuse the implemented methods in order to provide future feedback should the need arise or feedback on any changes made to the project. A key feature of Quantitative Research methodology is replicated research studies in order to obtain a large number of results in order to better generalise

In order to determine a list of requirements for this project, a survey would be an ideal solution to obtain relevant feedback regarding the target audience's opinions on SPAs and how likely they would be to make use of the created SPA. This would give a general idea of the requirements of a large selection of potential users, creating valuable feedback on a large scale.

Once this survey had been conducted successfully and all relevant responses were analyzed in detail to determine common trends, the requirements of the project could be outlined and work on the project could begin with more of a distinct direction. This method of

Quantitative Research methodology is **Descriptive** as the survey participant's responses were only measured once initially.

Numerous iterations of the project would then be made, each with more advanced features than the last, due to the understanding of the framework growing and design principles and ideas being discovered and implemented. This phase of the project took an **Experimental** approach to Quantitative Research due to feedback being measured both before and after the iteration of the project.

The overall aim of Quantitative Research is to determine requirements and their importance in relation to the target audience, whilst being able to construct statistical models of these results to allow observations to be made regarding response patterns and popular opinions [189]. This would help narrow down the most important features that are required to be implemented in the application and could possibly determine how effectively the project will achieve its goals upon completion.

3.1.1. SPA Usability Survey

A brief usability survey was conducted using SurveyMonkey [190] in order to gather a list of requirements. The resulting list of requirements gained from this survey can be found in *Appendix 3*, whilst the analysis of the overall results is found in detail in *Chapter 5.1*.

1. What is the current device type you are using to fill out this survey?

- iOS Phone/Tablet
- Android Phone/Tablet
- Other Phone/Tablet
- Windows Desktop/Laptop
- MacOS Desktop/Laptop
- Other

2. What is the most important thing you look for when using a website? (Text box for response, up to 150 characters)

3. Have you knowingly encountered or used a Single Page Application before?

- Yes
- No
- I don't know

4. If yes, did you prefer the usability of it to that of a normal website? (A Link to an example was included in case the answer to the previous question was 'no')

- Yes
- No
- I'm not sure (User can specify response)

5. Did you notice any improvements in specific areas?

- Performance (Load times etc.)
- Ease-of-use (Simplicity, Navigation etc.)
- Aesthetics (Polish, Animations etc.)
- Features
- Other (Please specify)

6. Would you avoid a website if there was a possibility that you could not access it via a mobile device?

- Yes
- No
- Not Sure

7. What kind of Single Page Applications would you prefer to encounter more often?

- E-Commerce
- News
- Communication
- Entertainment
- Streamable Content
- Gaming
- Other (Please specify)

8. Finally, on average, how would you rate your experiences so far with Single Page Web Apps? (Scale of 1-5, or Poor to Amazing provided)

These questions were chosen due to the fact that a fairly broad set of responses should be gained as there are a large number of options for some of the more important questions. As was outlined in *Chapter 3.1*, the chosen Research methodology is Quantitative, therefore a large pool of data is required in order to narrow down the more important design aspects of the application and make sure they function optimally.

Another reason this set of questions was chosen is because they provide an idea of where the application could be expanded upon in the future. This is an important benefit of a large scale Survey as it can determine the development direction of the application, thus possibly defining the potential success or failure of the project as a whole.

3.1.2. Testing Feedback Questionnaires

An Individual Test was conducted once the application had been designed and implemented, with acquaintances testing the application thoroughly to determine how effective the list of requirements had been adhered to. Once the Initial tests had been completed by the acquaintances, a Face-to-Face Questionnaire was conducted in order to determine the tester's views and opinions of the current iteration of the application.

This Individual Test and Face-to-Face Questionnaire was repeated throughout the development process during each Iteration of the project in order to gain relevant feedback on the progress of the application and what aspects should be changed if any. The questions asked to the testers individually are as follows.

- 1. How relevant is the Project to you?**
- 2. What are your preferred Aesthetics or Theme?**
- 3. Would you prefer that the application be accessible via Mobile or Tablet?**
- 4. What is the most important aspect of the website?**
- 5. Would you use the created application in the future?**
- 6. Do you think the created application would be successful within your demographic audience?**
- 7. How was your overall user experience of the application?**
- 8. Is there anything you would change or add to the application if you could?**
- 9. What was your favourite feature of the application?**
- 10. Where there any issues you encountered during general use of the application?**
- 11. Overall, how would you rate the created application? (On a scale of 1 to 10)**

3.2. Development Methodology

The most efficient Development Methodology for a solo developer creating a web based project with a set deadline would be the **Agile Development methodology** [191]. In more detail, utilising an Agile Development methodology in a projects' development stage allows a developer to step back and assess the trajectory of the project and dictate whether any changes need to be made or not [192][193].

An extension of the Agile Development methodology is **Scrum** which usually goes hand-in-hand with Object-oriented development projects [192]. An advantage to using Scrum is that it allows for an incremental or iterative approach, which allows specific features or stages of a project to be prioritised above others and encourages working on these features in a pre-determined sequence.

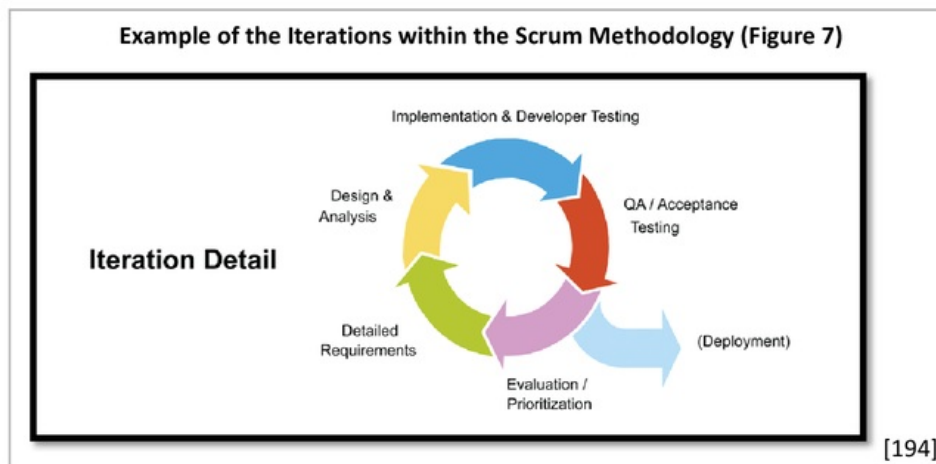


Figure 7 details the individual iteration details of a project utilising the Scrum development methodology [195]. Projects developed using this methodology will be broken up into numerous iterations of which the development process is repeated in order to gain feedback, fix potential issues or change a projects design quickly and effectively during each iteration [192]. These iterations are accomplished in short bursts or '*sprints*' which essentially reduce the time spent in between receiving feedback and reduces overall time spent in development.

The reason this method was chosen for this project is due to the fact that it allowed for large amounts of feedback both prior to the development of the project and through thorough testing once the project had been completed to an acceptable level. Another reason for choosing this methodology is due to the fact that it allowed the project's development trajectory to be assessed or changed at multiple steps of the development phase.

The fact that the direction the project was going in could be assessed and changed allowed for a very flexible development process, whilst also encouraging the use of object-oriented program which is evident through-out the code within the project. This was partly influenced by the feedback and requirements gathered during the initial planning process which helped to determine the importance of object-oriented programming in web development.

There is only one real disadvantage to utilising the Scrum methodology in this project. This consists of the fact that the development team is a singular person as opposed to a group of individual developers, which is generally what Scrum is most suitable for. Due to this, there were no team meetings or group sessions to determine the best course of action or how to tackle problems. The only major problem this posed during the project was the fact that some issues could have been fixed both quicker and more easily if there were other developers involved in the process to confer with.

During the course of the project, other React experienced developers were contacted in order to determine courses of action to fix specific problems or what the industry standard approach was for specific tasks, which helped a large amount and made the process much simpler and relatively smooth considering the task at hand.

Overall, the use of concepts of both Agile Development methodology and Scrum allowed for a fairly quick development process, which was essential after a majority of the time spent on the project was taken up by research and learning the framework language variation to an acceptable level.

During the development of the project, a large selection of software was used to achieve the end goal. Amongst this software were additional frameworks, languages and libraries that were essential to achieving the required outcome of the project. The list of software used in this project includes:

- **Sublime Text** – A sophisticated text editor for a large number of coding languages.
- **Node.js** – A JavaScript runtime for building scalable network applications.
- **Cmder** – Portable console emulator, essentially an upgrade to the command prompt.
- **Mozilla Firefox Developer Edition** – A developer oriented version of Firefox with advanced features and tools.
- Mozilla Firefox, Google Chrome, Microsoft Edge and Safari – The most popular web browsers currently available.
- **React Developer Tools** – A browser plugin specifically for React development.
- **Visual Studios Professional 2015** – Rich, integrated development environment for creating applications.
- **Microsoft Office 2016** – A Software Suite of tools offering numerous programs for achieving different goals.
- **Adobe Photoshop CS6** – Graphics design software for creating 3D visuals and artwork.

The main ingredient of the project was the use of the **React** JavaScript framework and the **JSX** language, these were chosen originally due to having encountered the framework in a website and being extremely impressed with the results. After some research, which was added to further during the writing of this report, it was generally considered by a large number of web developers that JavaScript MVC's along with variations of JavaScript frameworks and syntaxes but more specifically React, were the future of web development [75].

As for extra frameworks, languages and libraries, these were generally made up of additional React components, but also included libraries such as **Babel** [196], which is a

compiler that allows developers to utilise the *ECMAScript5* (ES5) or *ECMAScript6* (ES6) JavaScript standards and automatically converts JSX code into JavaScript. The use of Babel allows developers to write an entire application in JSX and have it automatically converted to JavaScript in order for it to run successfully in the browser [67][197].

Another important library used in the development of this project was **Gulp** [198], Gulp is a slightly newer alternative to Grunt which is an automated Node.js task runner that allows the created project to be rendered and streamed in any web browser whilst also updating on the fly whenever changes are made to any of the project files [42][67]. This allowed for much more agile development due to being able to instantly view any changes made to the project and showcase these changes to others developers for direct feedback.

4. Design

One of the main design principles utilised in this project was **Object-Oriented Programming** (OOP), which was integrated during the coding process of the project in order to promote a flexible coding architecture. OOP is a design principle that allows developers to create essentially modular applications by structuring the application in a way that individual components can be written separately from the main app, and then incorporated with one line of code [199]. This technique facilitates altering of individual components of the application instead of having to go through the entire application code in order to make any changes.

The biggest benefit of utilising OOP in any software development project, but also when designing this application, was that it allowed for any changes to be made via individual components or functions as opposed to altering the entire application code [199]. Another benefit to this design principle is that adding any new components in the future would be extremely simple. The only necessary change to the main application would be to include the newly created component and then implement it anywhere in the application by calling the relevant script.

Another major design principle utilised throughout the project was Iterative development, this is due to utilising the *Scrum* Research methodology. The use of *Scrum* methodology promotes Iterative development practices, this meant that there were numerous versions of the application developed during the project. Each Iteration attempted to improve upon the last by following recommendations from individual testers who were questioned once they had tested the application.

An Iterative approach to development allows for a large amount of feedback to be gathered whilst also ironing out any potential issues that are encountered at each stage of the project's lifecycle. This was mainly implemented during the Final Design phase and benefited the project hugely in the long run as it allowed for a large amount of tuning to be done and new methods of implementing specific features were discovered.

4.1. Paper Prototype

Prior to the coding process, a **Paper Prototype** was created in order to determine a general proof of concept of how the application's interface would be structured. During the initial design of the paper prototype, the responses to the survey in *Chapter 3.1.1* were referred to in order to build an interface that complied with the requirements outlined by the participants.

Welcome Page

The **Welcome** page, the Login form was removed from this page and added it to the Main page in order to remove the necessity of an 'Account' page and create a more streamlined user experience. An information panel was added in its place.

Main Page

The **Main** page, this was split into 3 Sections, Products, Order and Inventory Management in order to reduce actions required to access important features and promote a more streamlined user interface. Also the menu was removed as there was no real need for it.

Account Page

This **Account** page was completely removed as there is no requirement of it, the payment information can be entered at the checkout phase instead, and the only other available account options are Email and Password.

Checkout Page

Product	Quantity	Price
Product 1	1	£0.00
Product 2	2	£0.00

Total: £0.00

Cancel Purchase

There are no major changes to the **Checkout** page, aside from adding in a form for entering payment and shipping details.

As is fairly evident by the analysis of each section of the paper prototype, there were fairly drastic changes between this phase of the Design process and the Final Design phase. This is partly due to referring to the pre-determined list of Requirements, as seen in *Appendix 3*, that were created based on the results of the SPA Usability Survey in *Chapter 3.1.1*.

The user interface elements were refined in order to provide a more fluid experience overall, along with increasing performance by removing unnecessary views and features. This essentially proves that the paper prototype was a fairly good choice as it allowed for design elements to be scrutinized and altered to abide by the outlined requirements.

4.2. Basic Prototype

Before the Final Design Phase, an initial prototype was made that was loosely based on the paper prototype, screenshots of this prototype can be found in *Appendix 2*. The reasoning behind creating this initial prototype was not to test any Usability or Testing functions, but purely to get to grips with the Framework and essentially begin learning a new language in JSX.

The design of the initial prototype was very barebones and lacked any real direction or focus, but displayed a basic knowledge of the framework with a capacity to learn further. Taking this approach of creating a basic prototype in order to make mistakes and learn from them was a good Design choice as it helped dictate what areas needed to be worked on prior to the Final Design Phase.

4.3. Final Design Phase

During the final design phase, the use of *Gulp* was imperative as it allowed for the created application to be streamed to any browser by entering the address '*http://localhost:3000*'. This was incredibly efficient as every individual alteration to the code was updated in real time and instantly gave any indication of errors or individual coding mistakes. This was further helped by utilising the *React Developer Tools* [71] referenced in *Chapter 2.2.1*, which allowed for detailed analysis of individual parameters, references and preferences within the created application.

One of the Design practices used during the Final Design phase was Iterative implementation, which is a core design approach of *Scrum*, as detailed in *Chapter 3.2*. During this phase, numerous iterations of the application were tested by individuals with the feedback received helping to shape the final representation of the application. Each iteration was manipulated based on this feedback with features either added or taken away, and numerous bugs fixed to help mould the application into the final iteration. This final iteration was then tested and evaluated again, with the results available in *Appendix 1*.

5. Implementation

The implementation process was fairly demanding as there were certain limitations and problems encountered that took not just time, but a large amount of effort to overcome. Overall though, the journey to get to the end result was rewarding in that it provided various challenges that required solving to achieve the end goal.

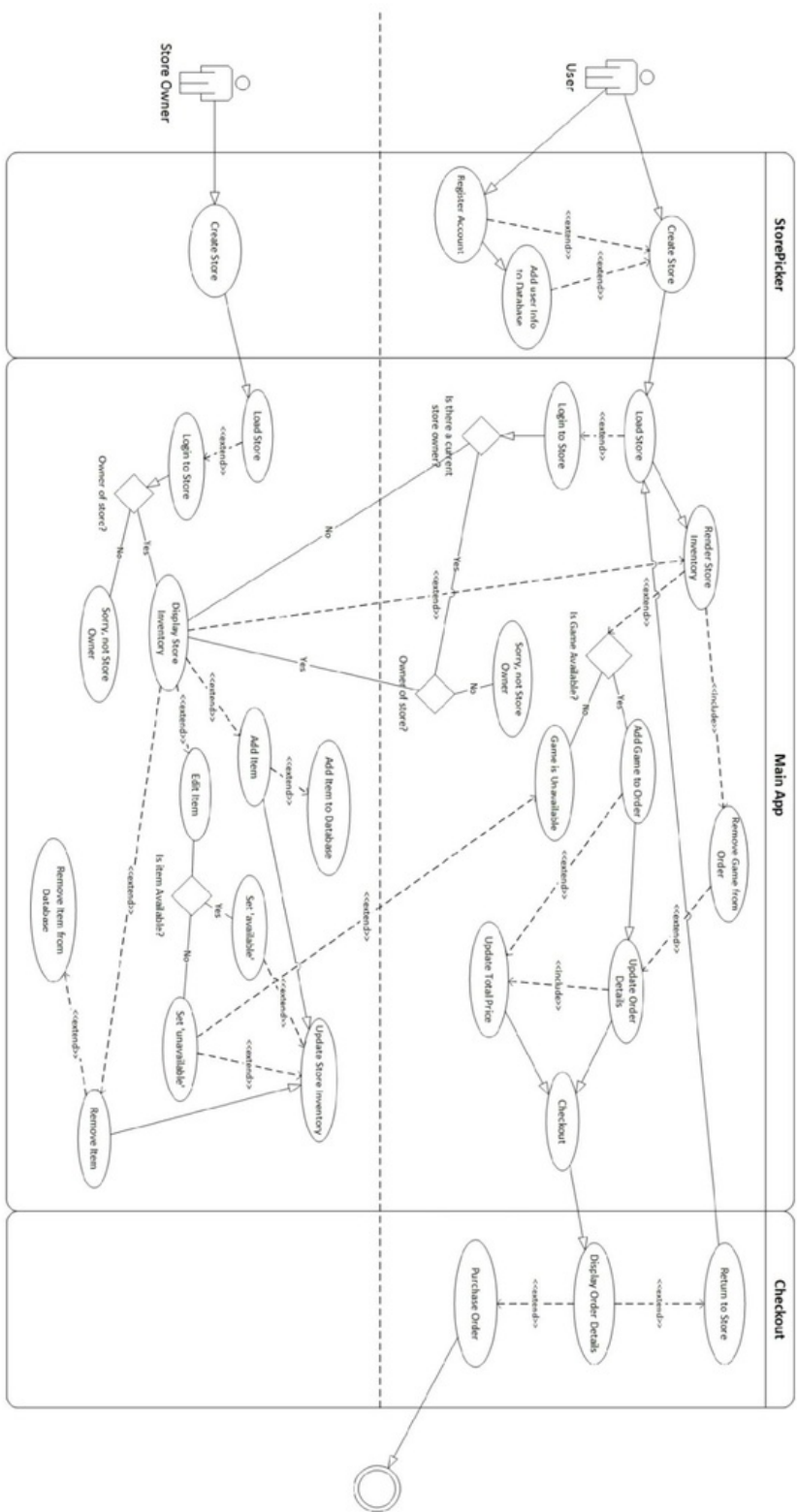
In order to determine a list of requirements for the project, a survey was created, as is detailed in *Chapter 3.1.1*, and the results of which can be found in *Chapter 6.1*. Once a final list of requirements had been determined based on the results of this survey, importance ratings were assigned to each of the Functional and Non-Functional requirements, these can be found in *Appendix 3*.

Based off these requirements, a Paper Prototype was created to demonstrate a rough User Interface design in order for a baseline to be built upon and allow for future refinements to the interface to resemble the user requirements. Once the Paper Prototype had been developed, a Use Case Diagram could then come to fruition in order to display the internal actions of the code and the function patterns the app would follow.

5.1. Use Case Diagram

The created Use Case Diagram was extremely useful because it allowed for a simple approach to coding as it could be approached step-by-step. This really simplified the coding process and exemplified the Agile Development method being utilised in the development of the project.

The two users of the application have various different interactions with the application, although at some points both users are able to perform similar actions. There are however, some actions restricted to the Store Owner, in order to allow that user to perform actions specific to the store of the owner, for example adding or removing products.



5.2. Code Implementation

One of the main design principles attempted throughout the coding process was to follow the Object-Oriented Principle in order to promote flexible code and modularity. This was made fairly simple by the use of JSX, which allows components to be imported into the main application.

Importing Components into the Main App (Figure 8)

```
import React from 'react';
import Header from './Header';
import Game from './Game';
import Order from './Order';
import Inventory from './Inventory';
import Checkout from './Checkout';
import Catalyst from 'react-catalyst';
```

Rendering Imported Components into the Application (Figure 9)

```
<div className="menu-wrap">
  <Header tagline="THE BEST GAMES EVER!" />
  <ul className="list-of-games">
    {Object.keys(this.state.games).map(this.renderGame)}
  </ul>
</div>
<Order games={this.state.games} order={this.state.order} removeFromOrder={this.removeFromOrder} />
<Inventory addGame={this.addGame} loadSamples={this.loadSamples} games={this.state.games} linkState={this.linkState} />
```

Evidence of an *Object-Oriented* approach is visible in both **Figure 8** and **Figure 9**, in *Figure 8* numerous components are imported into the main application allowing specific references, parameters or preferences to be fetched across components. As for *Figure 9*, the imported components have been included within the Application, allowing them to be rendered inside the application's main <div>.

Creating additional content Renders within Functions (Figure 10)

```
renderInventory(key) {  
  var linkState = this.props.linkState;  
  return (  
    <div className="game-edit" key={key}>  
      <input type="text" valueLink={linkState('games.' + key + '.name')} />  
      <input type="text" valueLink={linkState('games.' + key + '.price')} />  
      <select valueLink={linkState('games.' + key + '.status')}>  
        <option value="available">Available</option>  
        <option value="unavailable">Sold Out</option>  
      </select>  
      <textarea valueLink={linkState('games.' + key + '.desc')}></textarea>  
      <input type="text" valueLink={linkState('games.' + key + '.image')} />  
      <button onClick={this.props.removeGame.bind(null, key)}>Remove</button>  
    </div>  
  )  
}
```

Including the Function within the Component (Figure 11)

```
<h2 className="inventory-title">Inventory</h2>  
{logoutButton}  
{Object.keys(this.props.games).map(this.renderInventory)}  
<AddGameForm {...this.props} />
```

Another example of the created code utilising Object-Oriented principles is evident in both **Figure 10** and **Figure 11** respectively. *Figure 10* displays the ability to render additional content within a function, this function can then be called in the main component Render, exhibited in *Figure 11*, to allow for extremely flexible coding.

It is important to note that all the code examples mentioned in this chapter also show the benefit of using the JavaScript Syntax eXtension (JSX) programming language, as it allows for JavaScript and HTML to be incorporated together with ease. This alone is a huge quality of life improvement when coding web applications using JavaScript and HTML.

5.3. Software Testing

In order to thoroughly test the created application, three of the testing levels covered in *Chapter 2.4.2*, were applied along with *Usability testing*. These testing levels are *Unit tests*, *System tests* and *Acceptance tests*.

5.3.1. Unit Testing

During the course of the application development, the React Developer Tools were utilised in order to keep track of any React specific information, changes or errors. This tool was installed across multiple browsers to ensure cross-browser compatibility and consistency. This also meant that the application was tested across four of the most popular web browsers, these are Google Chrome, Mozilla Firefox and Firefox Developer Edition, Microsoft Edge and Safari.

In addition to testing cross-browser compatibility, individual, extremely basic Unit tests were conducted on the components. These unit tests were conducted using the Jest JavaScript Unit testing framework. Due to knowledge limitations, these tests could not be conducted thoroughly, however, the tests that were conducted passed and aside from this, there are no obvious problems with the application when it is in use.

5.3.2. System Testing

In order to test how well the system performed, a selection of acquaintances were invited to test the created application, a face-to-face questionnaire was then completed in order to evaluate each individual testers opinions of the application. The results of the questionnaires based on the final iteration of the application can be found in *Appendix 1*.

Overall the feedback was extremely positive, with users rating the functionality very highly. However, there were also some issues, primarily regarding the unfinished Checkout page of the application. Another issue highlighted by two testers was when returning to the store from the Checkout page, sometimes the store content failed to load, this is rectified by

pressing the *'Refresh Store'* button. Aside from this issue, the application functions extremely well.

5.3.3. Usability Testing

Again, in order to determine the state of the application's Usability, the face-to-face questionnaires in *Appendix 1* need to be referred to in order to gauge testers reaction to the application's functionality and overall ease-of-use.

Generally, the feedback was very positive, especially regarding the aesthetics and performance. Another positive aspect of the Usability that was highlighted by testers is the interesting features and simplicity of the layout, this was especially relevant to the elderly tester.

In summary, everything implemented into the interface works perfectly, all the features and animations work as they should and the interface is mostly considered a success by the testers who all enjoyed the aesthetics of the application.

5.3.4. Acceptance Testing

As with the other testing methods, the questionnaire results from *Appendix 1* are analysed for Acceptance testing. The general theme of these results is that the testers believed the application was functioning as was expected and to a high standard with good performance levels. As was noted, the Checkout is unfinished due to knowledge limitations, which was highlighted by two of the testers as a weak point of the application.

In conclusion, the application passed the acceptance test according to the testers with pretty high praise. The testers were given the option to grade the application out of 10, with the results received between 8 to 10, this shows a high level of satisfaction with the application and translates into a success.

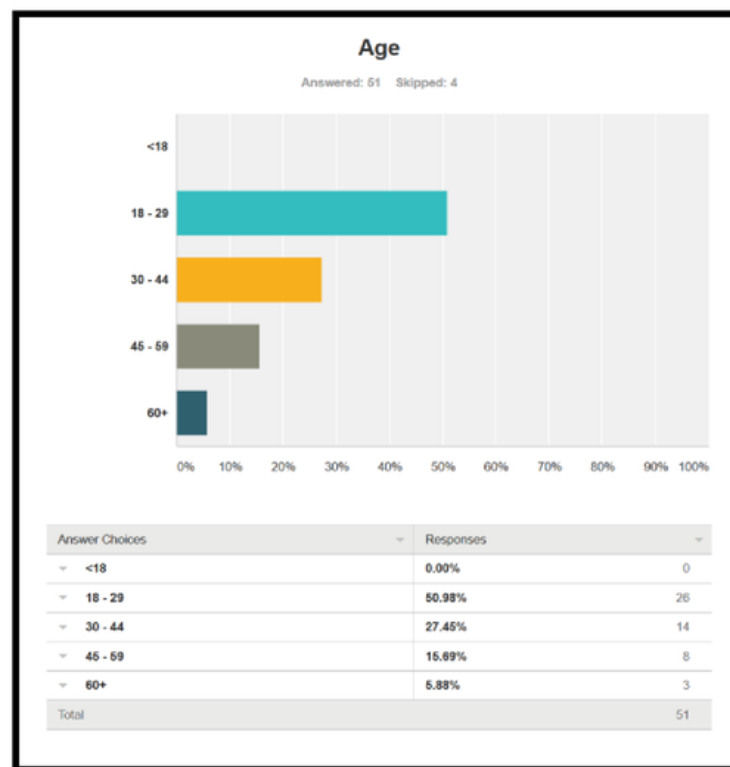
6. Evaluation

For the Evaluation of SPA's and the created application a Survey and face-to-face questionnaires were created and completed. The results of the individual post development testing questionnaires can be found in *Appendix 1*, while the results of the SPA Usability Requirements survey will be analysed in detail.

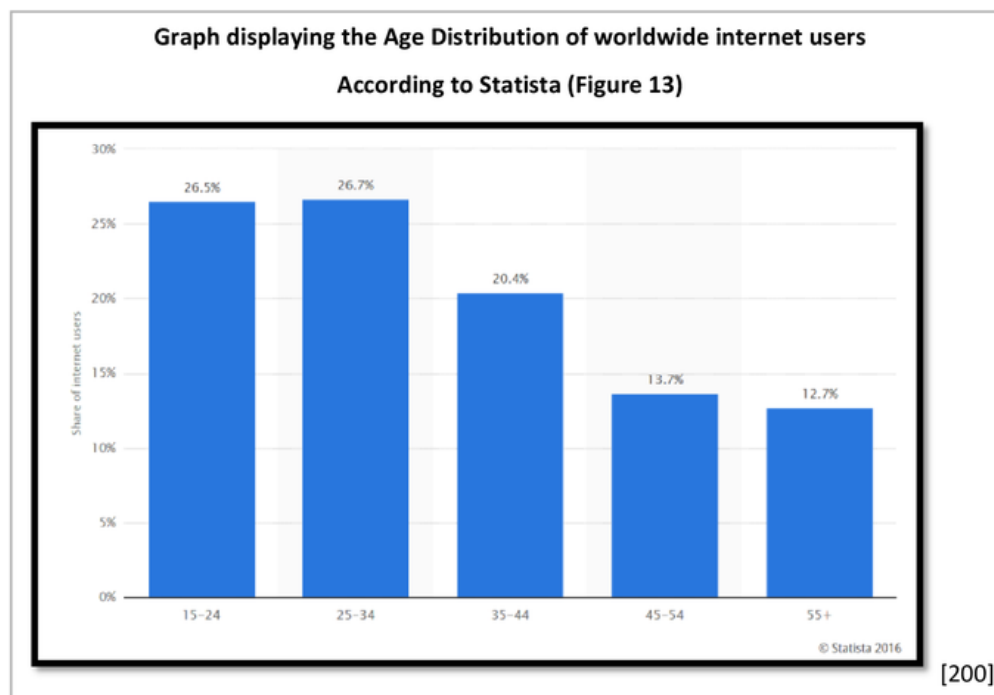
6.1. SPA Usability Requirements Survey Results

A short survey was created using *SurveyMonkey* [190] as detailed in *Chapter 3.1.1* and given to a random selection of users in order to determine a rough list of requirements based on feedback regarding general website usage, Single Page Application usage and sentiments, usability preferences and finally, important features generally expected from a website.

Graph displaying Age range of Survey participants (Figure 12)

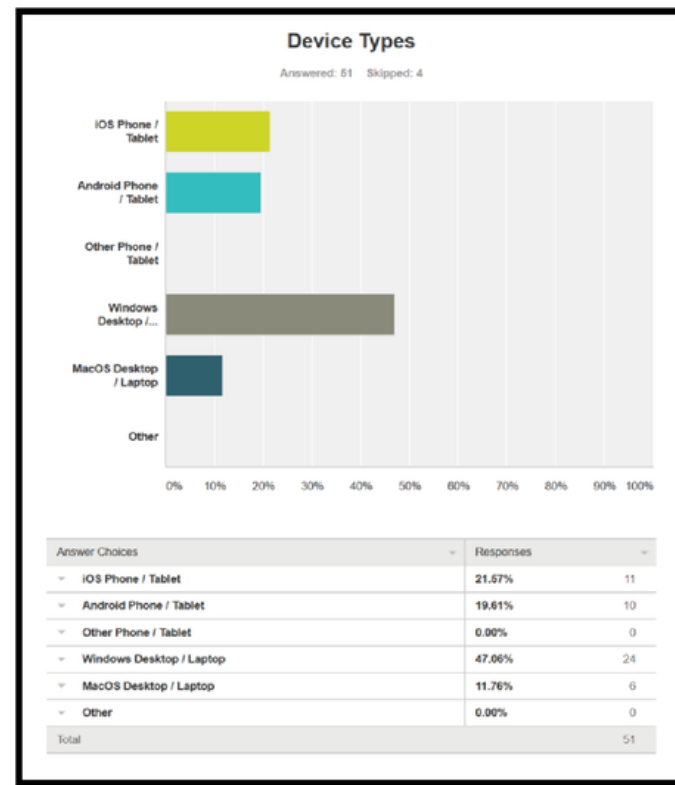


As is displayed in the graph in **Figure 12**, the general age of the demographic is within the 18 to 29 range with more than 50% of the users within this category opting to participate in the survey. With the number of participants above the age of 60 being the least, aside from under-18's, a general consensus of the older generations' views on Single Page Applications cannot really be gauged.



Whilst it is quite unfortunate that there are less responses from elderly users, it is not a major issue as the target audience of the project is aimed more at users between the 15 to 45 age range, which according to Statista [201] and displayed in the graph in **Figure 13**, makes up 73.6% of worldwide internet users as of November 2014 [200].

Graph displaying Device types of Survey participants (Figure 14)



Generally, **Figure 14** shows that the number of desktop or laptop users that responded to the survey is much larger than that of smart phone or tablet users. The fact that 58.82% of the devices in use by users surveyed are desktop or laptop based indicates that the majority of the feedback will be aimed towards websites or content developed towards larger screen resolutions and desktop operating systems.

Alternatively, as the number of responders utilising either an iOS or Android smart device makes up the rest of the user base at 41.18%, which indicates that just under half of the responses will be looking for features that improve a website's usability on either a mobile device or tablet. Due to the ever growing popularity of smart devices and the time spent using these devices, the desire for mobile oriented websites or applications is growing exponentially.

In terms of relating this to the created project, the advantage of creating a web application using React is that both React Native and Flux are an option for expanding the application to mobile platforms. As the responses to this survey are primarily desktop browser based in terms of requirements, the feedback is generally applicable to this project. This does not however mean that the users providing feedback based on requirements for mobile devices is useless as there is potential within this project to expand across multiple platforms in the future.

**Graph displaying the most Important Features required of a Website
by survey participants (Figure 15)**

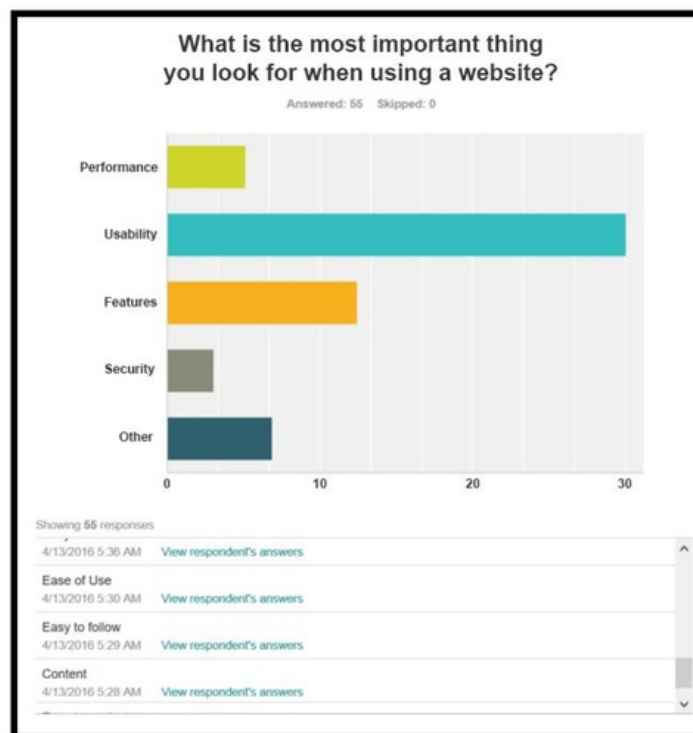


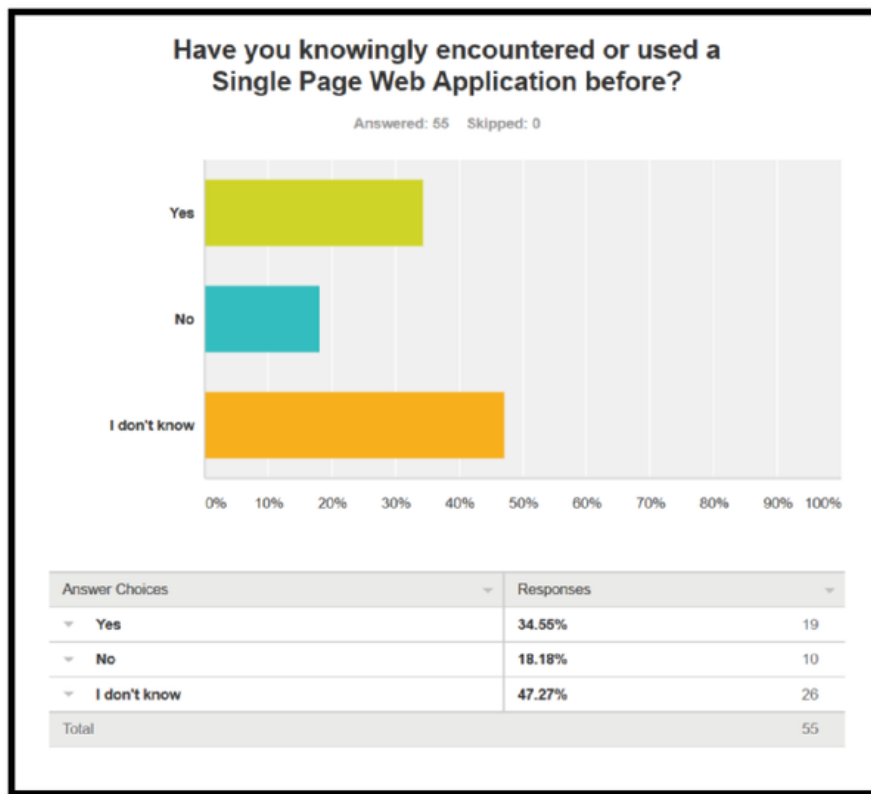
Figure 15 shows the most important features requested by participants of the survey with an extremely dominant amount of responses geared towards the Usability features of a website. A total of 30 responses were gathered that prioritised the Usability of a website as

a key feature, these answers included general Clarity, overall Theme, Navigation options and Organisation or Layout.

Other important feature requirements include Performance and Features, though these were much less prevalent than Usability requirements they are still important design aspects of a website to consider during development. In contrast, there were very few mentions of Security playing a key factor in a website's design which could be quite worrying especially if the website was aimed at e-commerce and in-turn stored valuable personal information of its users.

In a growing age of identity theft and fraud, web users should be especially wary of unsecure websites or any malicious content within said websites. According to *Sullivan and Liu*, up to 70% of internet security attacks come through web applications [202]. This is made even more alarming by a recent survey of IT specialists by the Ponemon Institute who determined that almost 90% of companies do not regard web application security as a serious threat, and thus spend more money on coffee for their employees than protecting their web applications [203].

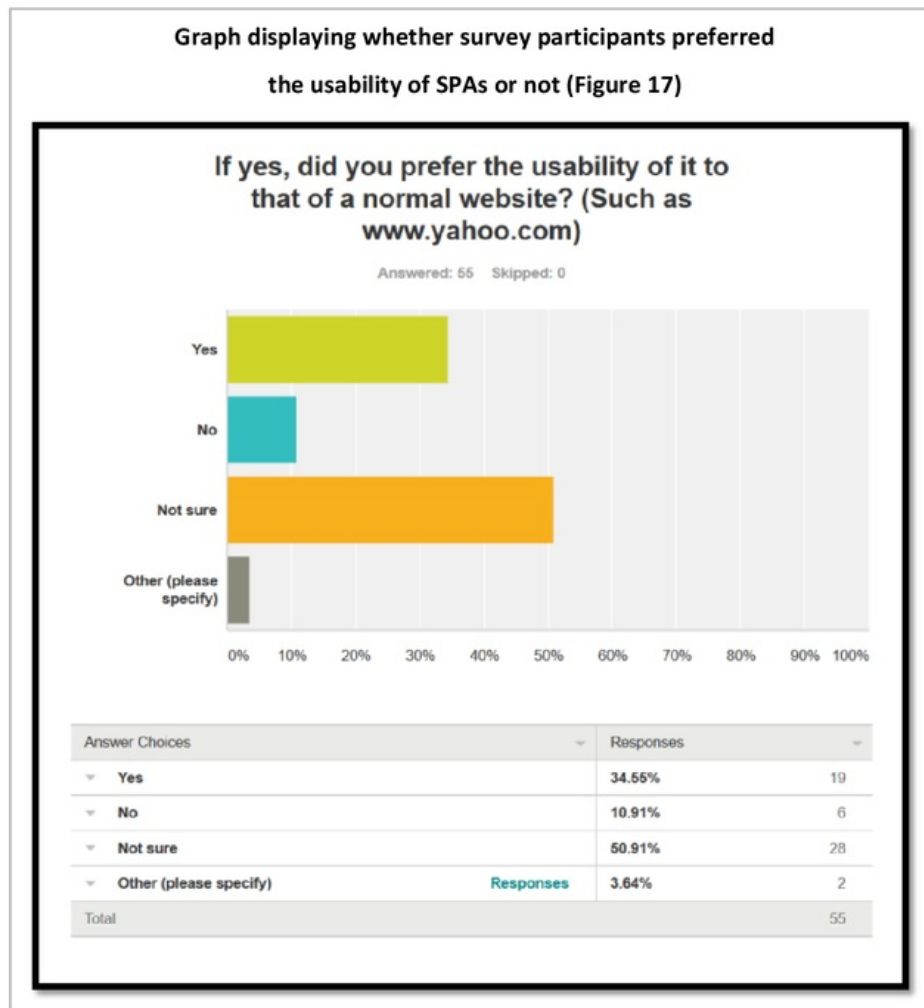
Graph displaying whether Survey participants have encountered a SPA or not (Figure 16)



Generally, most regular internet users will not be aware of the differences between a website and a SPA, or how to determine whether a website is a SPA or not. This is displayed in **Figure 16** by the large amount of responses from survey participants stating that they did not know whether they had encountered or even used a SPA before.

Due to 47.27% of the survey responders not knowing if they have experienced a SPA before, a challenge is represented when gauging responses to further questions within the survey regarding SPAs. However, examples of popular SPAs were given in the description bellow this question in order to allow participants to understand the difference of a SPA to that of a normal website. This allows for actual responses based of previous experiences, whilst still allowing for users to respond to further questions with a better idea of what a SPA is.

Thankfully, a total of 52.73% of the survey participants had an idea of what a SPA is, with 34.55% actually having encountered a SPA prior to this survey. These numbers are encouraging and, with the inclusion of a link bellow to examples of SPAs, any users that had not used a SPA now had the chance to in order to provide accurate results to the rest of the questions.



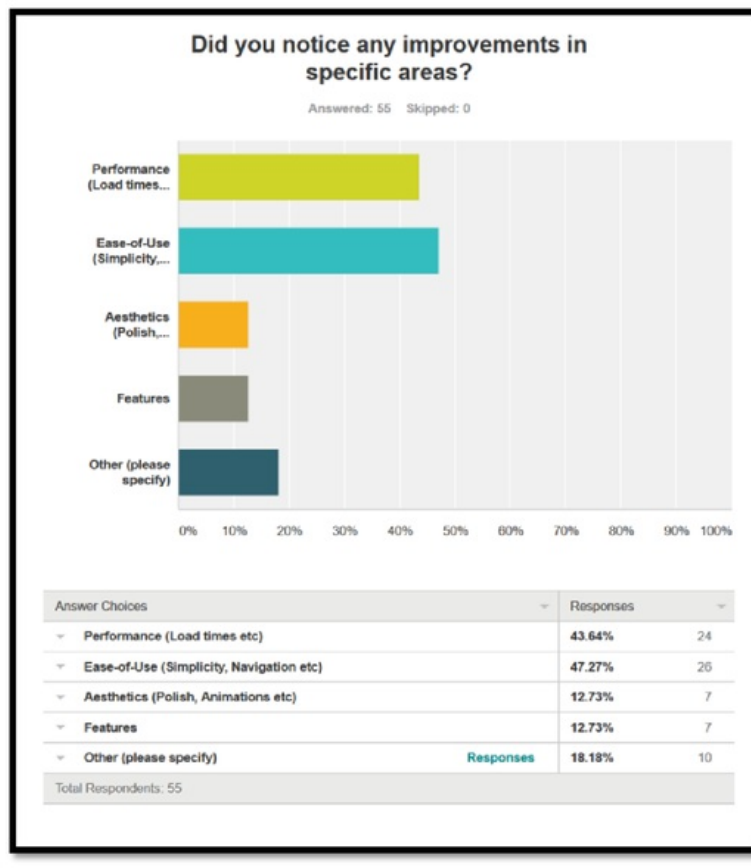
The responses shown in the graph in **Figure 17** display two overwhelming results, the highest amount of responses was 50.91%, and this was that users were not sure if the usability of a SPA was preferred to that of a normal website. This could be due to two

different scenarios, the first is that the majority of survey participants could not have encountered a SPA before so struggled to provide a response based on that fact. However, the second scenario could be that the responses are based on the user experiencing a SPA that was extremely similar to a normal website's functionality.

Either of these scenarios could have arisen, as an example of a SPA was provided earlier in the questionnaire, it is safe to assume that the second scenario is slightly more likely, which is based on user experience but provides valuable feedback none-the-less. This assumption is backed up by the answers provided in the 'Other' option, of which users noted that their usability experience varied on a site-by-site basis.

On the other hand, 34.55% of the responses were positive and are encouraging, indicating a noticeable improvement in Usability across numerous SPAs.

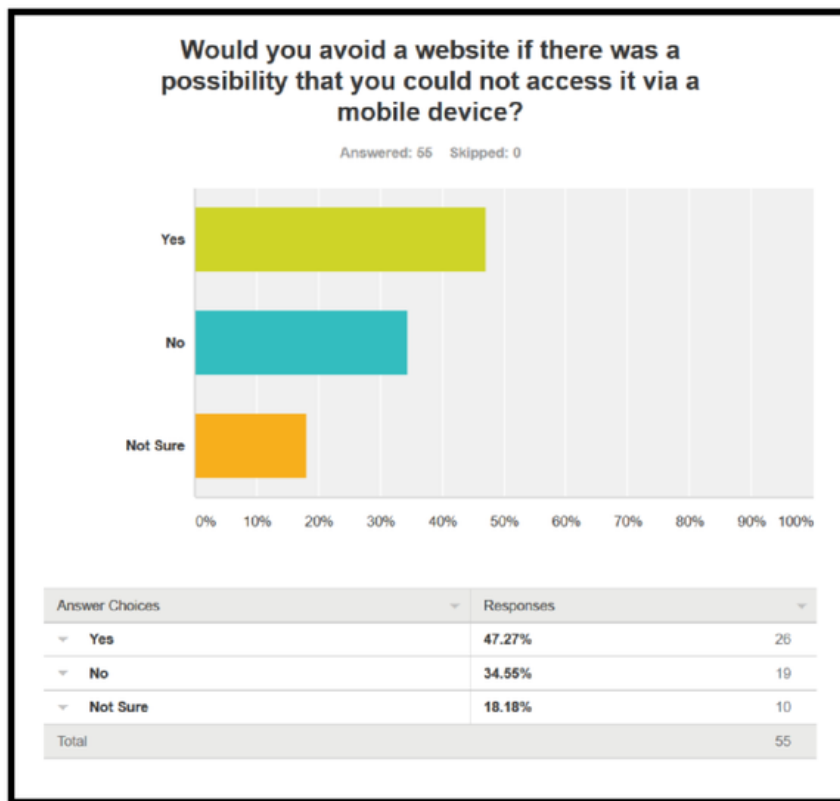
Graph displaying Survey responses to
Possible Improvements in Specific areas (Figure 18)



The most important results from this question of the survey are the overwhelming responses regarding both Performance and Ease-of-use as showcased in **Figure 18**, with 24 and 26 individual choices respectively. The fact that these two improvements were highlighted by survey participants above the other features indicates two of the most important features when deciding to develop a SPA instead of a standard website.

The results could possibly be skewed due to the fact that this question is a multiple choice question, however this still shows strong improvements are noticeable in these areas. The majority of the responses specified in the 'Other' option are mainly users being unsure of their answer due to not experiencing a SPA or not being able to tell.

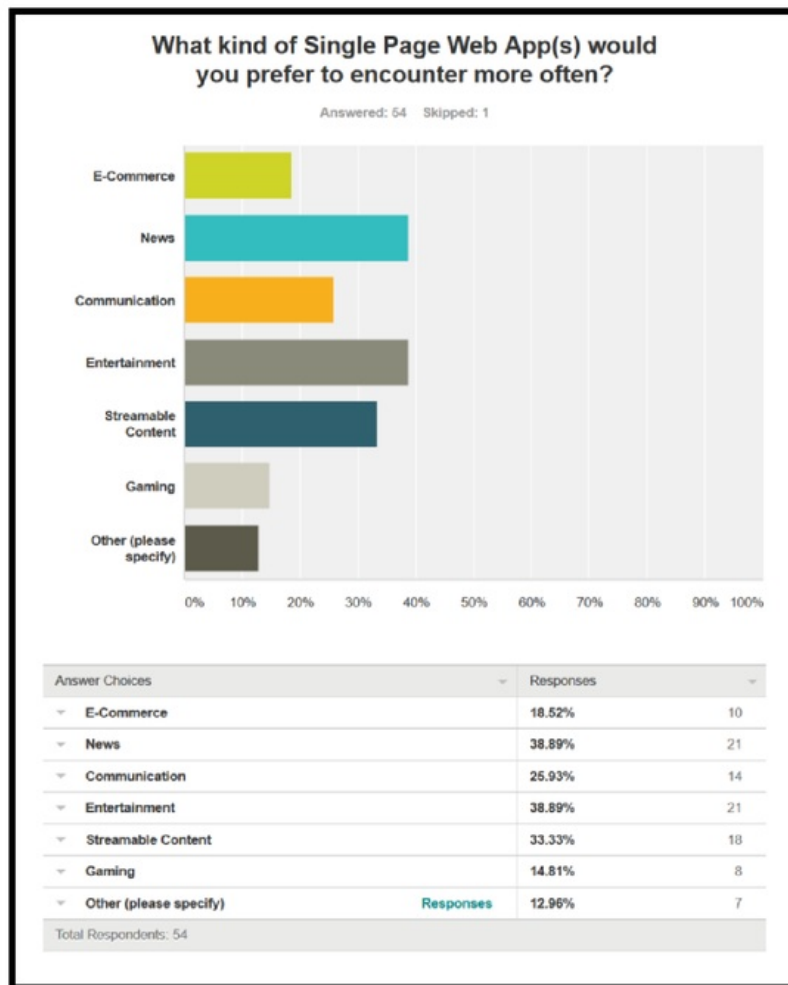
Graph displaying whether Survey participants would avoid a website if it was inaccessible via mobile devices (Figure 19)



As was highlighted in *Figure 14* regarding the large number of smart device users partaking in the survey, the demand for web developers to create projects with a 'Mobile First' design philosophy is becoming more prevalent as the web becomes more accessible as a whole. **Figure 19** emphasises this point further by highlighting that 47.27% of the responses state that a website would be avoided if content was unable to be accessed on a mobile device.

Alternatively, a reasonable amount of survey participants stated that this would not deter them from utilising a website, this is encouraging due to the lack of a mobile specific design for the created project. Whilst being accessible via a mobile browser, the project is not responsive or integrated into a mobile specific app, so functionality is limited in this case.

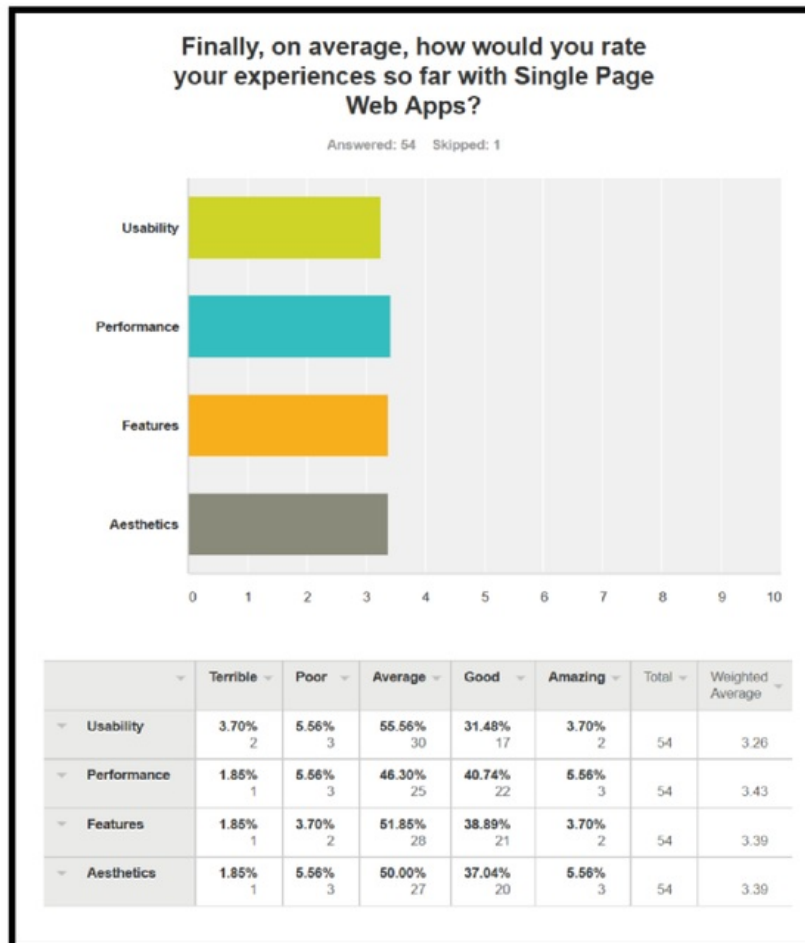
Graph displaying preferred SPAs of Survey participants (Figure 20)



Another multiple choice question, the results for which are shown in **Figure 20**, gives a rather balanced and varied number of responses. Unfortunately, relating to the created project, both e-commerce and gaming are both fairly low on the list of preferred content that survey participants would like to see more often. This could be partly due to the fact that the users featured in this survey do not lie in the target demographic or it could be that they generally prioritise other forms of content over these options.

Alternatively, this type of feedback could be extremely useful for any future attempts at creating a relevant SPA as there are large numbers of positive responses for news, entertainment and streamed content.

Graph displaying average User Experience Ratings for Survey participants (Figure 21)



The results displayed in **Figure 21** can be taken in two ways, on one hand the feedback is fairly poor for development purposes as the majority of responses were for the Average and Good options across all options. On the other hand, this feedback could be positive in that users generally had average or above experiences with SPAs which is a good sign that there are little to no drawbacks of utilising SPAs over static webpages.

6.2. Summary

Overall, the feedback from this survey is very important for determining a list of requirements for the project. As is evident, an emphasis on Usability and Performance of the created SPA is imperative, along with supplying relevant content and generally making the application easy to use. Another requirement that could be possibly implemented in the future is expanding the user base to mobile platforms by utilising React Native and Flux to create an iOS or Android version of the SPA.

Comparing this collected data to the collected research within *Chapter 2*, is quite difficult because a large portion of the responses are based on the user being unaware of what a SPA is in some cases. Although, the collected data that does successfully correlate with the research conducted is that of the overall Usability, Performance and Features being noticeably improved with SPA's as opposed to the alternatives.

In the case of analysing whether the aim and objectives were met by this survey, it is difficult as the conducted survey does not necessarily cover React specifically, it is more oriented towards SPA's in general. This does not mean that no information regarding the Aim is not gathered though, as the feedback suggests that the Usability of SPA's is generally considered superior to generic websites.

Overall, these results are roughly the sort of feedback as was expected from a relatively large scale survey. Whilst not the most conclusive results regarding SPA Usability conceptions as a whole, the results still go some way to providing an insight into the general consensus of SPA Usability on a large scale.

7. Conclusions and Recommendations

In conclusion, the introduction of SPA's has changed the trajectory of web design and propelled it into a very positive direction. The natural progression from an MVC to a VC that React has moved towards is forcing web development trends to adapt and evolve into an

exciting prospect for the future. The level of flexibility, performance and usability that comes with a SPA is far superior to that of any competitors currently.

React is not just a framework for developing SPA's, it can also create incredibly responsive, user friendly websites with multiple pages and a plethora of impressive features. This means that React has the potential to dominate not just the MVC or SPA demographic, but also the majority of web development trends into the future. The disadvantage however is that a large portion of regular internet users do not, or will not realise the difference between a multi-page or static website and a SPA. The positive aspect of this alternatively, is if they are made aware of the difference, they will usually rate it very highly compared to the alternatives.

The remedy to this is that over time, web users will gradually learn to differentiate between a multi-page or static website and a SPA, but until then, React and other similar frameworks will generally continue to go unappreciated. Fortunately, with the backing of a multi-billion-dollar company such as Facebook and a large dedicated community of developers, React will continue to receive the support it needs to become the front-runner in web development that it can, and should be.

For future research, it would be beneficial to research cross-platform application implementation in order to provide potential users with a method to access the application on mobile devices, or even download the application onto a desktop or laptop. This will not only increase both the accessibility and usability, but also increase the target audience of the application.

Another potential important avenue of research would be cloud-based data storage, this is an extremely flexible way to manage the back end of an application in a user-friendly, easy to access form. The implementation of the cloud-based data storage helped make the development process much smoother than it would likely have been otherwise, and provides a secure way to store the application's important information.

Both of these subjects would be extremely relevant for further research as they are both equally important when it comes to the future of website or application design.

8. Reflection

Overall, I feel like this project went fairly well, although at the same time, there are probably numerous things I would approach differently if I were to do it again. One of the biggest issues I encountered during the development process was lack of knowledge, which was in part probably due to taking on a challenge that may have been too much!

However, I am happy that I attempted it, as I enjoy learning and trying to achieve difficult things in order to further myself and increase my knowledge in a field I enjoy. I feel like it was a positive learning experience and will definitely take heed of it in future endeavors.

As for the positives of the experience, I feel like I got to grips with learning advanced JavaScript and, in turn JSX fairly well, this in turn helped me to further my skills by going out of my way to learn something new. Part of the reason I chose to do a project on React and focus on Usability was to force myself into learning something new and challenging. The other reason I chose React was because, whilst doing original research on JavaScript MVC's, I came across a large amount of praise and positive feedback about React and how refreshing it was compared to the alternatives.

To add to the positives, I also managed to find numerous websites and communities of React developers that I could contact if I wanted to troubleshoot an issue and they would provide constructive feedback in order for me to learn more. This helped me hugely as I felt that I had somewhere to turn to if I had a simple problem, which also in turn helped me to learn more in the process.

In addition to these React communities, one of the big benefits of React for me is the accessibility to additional library extensions. There are so many open-source libraries available on github in order to basically achieve any outcome required, I haven't really experienced anything like this before in Web design so it was really refreshing for me.

Finally, another positive was the fact that I ended up pushing myself out of my comfort zone by conducting questionnaires and surveys in order to obtain relevant information to the

project and help further the design of my application. I personally don't enjoy writing surveys and collecting and analysing data, but I do think it was worth doing in this case and I appreciate the feedback much more now.

As for the negative aspects of the project, there were quite a few that I didn't really think about prior to the planning phase. For instance, my biggest issue is, and always has been, time management and dedicating my time to my work. I always mark this as an issue and I think this finally might make me do something about it as it has made the approach to the deadline extremely stressful when it didn't need to be.

The next negative point ties into time management somewhat as I feel like I probably didn't give myself enough time to learn advanced JavaScript **and** an entirely new framework. On top of this, I feel like I let myself down when it came to testing my code. I got to the point where I'd made my application and was happy with it, but then started running out of time to actually begin unit testing the code.

Part of the reason I was quite rushed towards the end of the deadline for this project was due to creating numerous small React projects prior to starting this one as I wanted to learn as much as I could have before implementing my final application. In retrospect, I think I would have started learning React during the summer break in 2015, this would have allowed me a much larger timeframe to become more comfortable with the framework.

Finally, another issue I encountered was that there is currently very little published literature on React. I purchased 4 React books from Amazon that were only released midway through the academic year and they are very basic compared to some of the alternative frameworks' published books.

Whilst in hindsight, I may have bitten off more than I could chew, I think overall I ended up doing a decent job, even just with learning React and creating an application I'm happy with and will probably continue to work on in the future.

9. References

- [1] G. Fink, I. Flatow, and S. Group, *Pro Single Page Application Development: Using Backbone.js and ASP.NET*. Apress, 2014.
- [2] S. Varghese, "Web Development Trends For 2015 And Beyond — Medium," 2014. [Online]. Available: <https://medium.com/@shijuvar/web-development-trends-for-2015-and-beyond-c2d3c1ef5718#.y4wnenvtc>. [Accessed: 13-Jan-2016].
- [3] N. Huening, "Web Design & Development Trends You'll See in 2015 - New Media Campaigns," 2015. [Online]. Available: <http://www.newmediacampaigns.com/blog/web-design-development-trends-youll-see-in-2015>. [Accessed: 13-Jan-2016].
- [4] M. Angelov, "The Languages And Frameworks That You Should Learn In 2015 | Tutorialzine," 2014. [Online]. Available: <http://tutorialzine.com/2014/12/the-languages-and-frameworks-that-you-should-learn-in-2015/>. [Accessed: 13-Jan-2016].
- [5] W3schools, "Introduction to HTML." [Online]. Available: http://www.w3schools.com/html/html_intro.asp. [Accessed: 13-Jan-2016].
- [6] Microsoft, "ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET," 2013. [Online]. Available: <https://msdn.microsoft.com/en-us/magazine/dn463786.aspx>. [Accessed: 13-Jan-2016].
- [7] W3schools, "AJAX Introduction." [Online]. Available: http://www.w3schools.com/ajax/ajax_intro.asp. [Accessed: 13-Jan-2016].
- [8] J. J. Garrett, "Ajax: A new approach to web applications," pp. 1–5, 2005.
- [9] A. Mesbah and A. van Deursen, "An Architectural Style for Ajax," *2007 Work. IEEE/IFIP Conf. Softw. Archit.*, pp. 9–9, 2007.
- [10] Google, "Company – Google," 1998. [Online]. Available: <https://www.google.co.uk/about/company/>. [Accessed: 13-Jan-2016].
- [11] W3schools, "XML Introduction." [Online]. Available: http://www.w3schools.com/xml/xml_what.asp. [Accessed: 13-Jan-2016].
- [12] A. Mesbah and A. Van Deursen, "Migrating multi-page web applications to single-page AJAX interfaces," *Proc. Eur. Conf. Softw. Maint. Reengineering, CSMR*, pp. 181–190, 2007.
- [13] M. Mikowski and J. C. Powell, *Single Page Web Applications: JavaScript End-to-end*. Manning, 2013.
- [14] F. Monteiro, *Learning Single-page Web Application Development*. Packt Publishing Ltd, 2014.
- [15] Discord, "df21d1b0c082d8a3cce1c43b290614f9.png." [Online]. Available: <https://discordapp.com/assets/df21d1b0c082d8a3cce1c43b290614f9.png>. [Accessed: 19-Nov-2015].
- [16] Discord, "Discord - Free Voice and Text Chat for Gamers." [Online]. Available: <https://discordapp.com/>. [Accessed: 19-Nov-2015].
- [17] Discord, "Hammer & Chisel." [Online]. Available: <https://discordapp.com/company>. [Accessed: 13-Jan-2016].
- [18] Discord, "Discord Blog." [Online]. Available: <https://blog.discordapp.com/>. [Accessed: 13-Jan-2016].
- [19] GMail, "unnamed.png." [Online]. Available: <https://lh5.ggpht.com/WK1g2s6NMv0pWJPJPfGBf2tzDgxI0IdSj4YFB2YrZhOI3f3->

- 5sJIYjLLwKF7AEJmfgc=h900. [Accessed: 19-Nov-2015].
- [20] Google, "Gmail - Free Storage and Email from Google." [Online]. Available: https://www.google.com/intl/en/mail/help/about.html#utm_source=en-et-more&utm_medium=et&utm_campaign=en. [Accessed: 19-Nov-2015].
 - [21] GooglePress, "Google Gets the Message, Launches Gmail – News from Google." [Online]. Available: <http://googlepress.blogspot.co.uk/2004/04/google-gets-message-launches-gmail.html>. [Accessed: 13-Jan-2016].
 - [22] GoogleApps, "Google Apps for Work – Email, Collaboration Tools and More." [Online]. Available: https://apps.google.com/#utm_medium=blog&utm_source=us-en-ogb-oob0707&utm_campaign=oob. [Accessed: 13-Jan-2016].
 - [23] Google, "Official Google Blog: Google Apps is out of beta (yes, really)." [Online]. Available: <https://googleblog.blogspot.co.uk/2009/07/google-apps-is-out-of-beta-yes-really.html>. [Accessed: 13-Jan-2016].
 - [24] Outlook, "7345.OWA_delegate_7.png." [Online]. Available: https://officeblogseast.blob.core.windows.net/wp-content/migrated-images/78/7345.OWA_delegate_7.png. [Accessed: 19-Nov-2015].
 - [25] Microsoft, "Outlook - Sign In." [Online]. Available: https://login.live.com/login.srf?wa=wsignin1.0&ct=1447900291&rver=6.1.6206.0&sa=1&ntprob=-1&wp=MBI_SSL_SHARED&wreply=https:%2F%2Fmail.live.com%2F%3Fowa%3D1%26owasuffix%3Dowa%252f&id=64855&snsc=1&cbcxt=mail. [Accessed: 19-Nov-2015].
 - [26] Microsoft, "Microsoft – Official Home Page." [Online]. Available: <https://www.microsoft.com/en-gb/>. [Accessed: 13-Jan-2016].
 - [27] IntoWindows, "Microsoft Office History In Brief." [Online]. Available: <https://web.archive.org/web/20120829165451/http://www.intowindows.com/microsoft-office-history-in-brief/>. [Accessed: 13-Jan-2016].
 - [28] Microsoft, "Buy Microsoft Office 365 | Home & Personal software suites." [Online]. Available: <https://products.office.com/en-gb/buy/office>. [Accessed: 13-Jan-2016].
 - [29] Dropbox, "g0qRy.png." [Online]. Available: <http://i.imgur.com/g0qRy.png>. [Accessed: 19-Nov-2015].
 - [30] Dropbox, "Dropbox." [Online]. Available: <https://www.dropbox.com/>.
 - [31] Dropbox, "Dropbox - About." [Online]. Available: <https://www.dropbox.com/about>. [Accessed: 13-Jan-2016].
 - [32] R. MacManus, "Is Dropbox Really The World's 5th Most Valuable Startup? - ReadWrite," 2011. [Online]. Available: http://readwrite.com/2011/10/03/dropbox_value. [Accessed: 13-Jan-2016].
 - [33] BusinessInsider, "The 100 Most Valuable Startups In The World, Revamped And Revised!," 2011. [Online]. Available: <http://www.businessinsider.com/2011-digital-100?IR=T#5-dropbox-5>. [Accessed: 13-Jan-2016].
 - [34] G. E. Krasner and S. T. Pope, "A Cookbook for Using the Model- View-Controller User Interface Paradigm in Smalltalk-80," *Joop J. Object Oriented Program.*, vol. 1, pp. 26–49, 1988.
 - [35] F. Bushmann, R. Meunier, and H. Rohnert, "Pattern-oriented software architecture: A system of patterns," *John Wiley&Sons*, vol. 1, p. 476, 1996.
 - [36] T. Reenskaug and J. O. Coplien, "The DCI Architecture: A New Vision of Object-Oriented Programming," 2009. [Online]. Available: http://www.artima.com/articles/dci_vision.html. [Accessed: 14-Jan-2016].

- [37] G. E. Krasner and S. T. Pope, "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System," *J. object oriented Program.*, vol. 1, no. 3, pp. 26–49, 1988.
- [38] A. Leff and J. T. Rayfield, "Web-application development using the Model View Controller design pattern," *Enterp. Distrib. Object Comput. Conf. 2001. EDOC '01. Proceedings. Fifth IEEE Int.*, vol. 22002, no. March, pp. 1–30, 2001.
- [39] J. Snook, "JavaScript MVC - An A List Apart Article," 2009. [Online]. Available: <http://alistapart.com/article/javascript-mvc>. [Accessed: 14-Jan-2016].
- [40] MVC, "500px-MVC-Process.svg.png." [Online]. Available: <https://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/MVC-Process.svg/500px-MVC-Process.svg.png>. [Accessed: 14-Jan-2016].
- [41] S. Burbeck, "Applications Programming in Smalltalk-80 (TM): How to use Model-View-Controller (MVC)," *Smalltalk-80 v2*, vol. 80, no. Mvc, pp. 1–11, 1992.
- [42] D. Odell, *Pro JavaScript Development: Coding, Capabilities, and Tooling*. Apress, 2014.
- [43] Google, "MVC Architecture - Google Chrome." [Online]. Available: https://developer.chrome.com/apps/app_frameworks. [Accessed: 14-Jan-2016].
- [44] A. Osmani, "AddyOsmani.com - Understanding MVC And MVP (For JavaScript And Backbone Developers)," 2012. [Online]. Available: <https://addyosmani.com/blog/understanding-mvc-and-mvp-for-javascript-and-backbone-developers/>. [Accessed: 14-Jan-2016].
- [45] U. Shaked, "AngularJS vs. Backbone.js vs. Ember.js." [Online]. Available: <https://www.airpair.com/js/javascript-framework-comparison>. [Accessed: 14-Jan-2016].
- [46] React, "React Logo," 2015. [Online]. Available: <http://codejunkii.com/wp-content/uploads/2015/02/fbreact.png>.
- [47] React, "A JavaScript library for building user interfaces | React," 2015. [Online]. Available: <https://facebook.github.io/react/>. [Accessed: 14-Jan-2016].
- [48] C. Gackenheimer, *Introduction to React*. Apress, 2015.
- [49] Facebook, "Facebook.com," 2015. [Online]. Available: <http://www.facebook.com>. [Accessed: 09-Nov-2015].
- [50] B. Fisher, "React (JS Library): How was the idea to develop React conceived and how many people worked on developing it and implementing it at Facebook?," 2015. [Online]. Available: <https://www.quora.com/React-JS-Library/How-was-the-idea-to-develop-React-conceived-and-how-many-people-worked-on-developing-it-and-implementing-it-at-Facebook/answer/Bill-Fisher-17>. [Accessed: 14-Jan-2016].
- [51] P. Krill, "React: Making faster, smoother UIs for data-driven Web apps | InfoWorld," 2014. [Online]. Available: <http://www.infoworld.com/article/2608181/javascript/react--making-faster--smoother-uis-for-data-driven-web-apps.html>. [Accessed: 14-Jan-2016].
- [52] P. Hunt, "Why did we build React? | React," 2013. [Online]. Available: <https://facebook.github.io/react/blog/2013/06/05/why-react.html>. [Accessed: 14-Jan-2016].
- [53] Facebook, "XHP-Lib." [Online]. Available: <http://facebook.github.io/xhp-lib/>. [Accessed: 14-Jan-2016].
- [54] Facebook, "An Introduction to XHP," 2011. [Online]. Available: <http://facebook.github.io/xhp-lib/2011/02/04/an-introduction-to-xhp.html>. [Accessed: 14-Jan-2016].

- [55] C. Dawson, "JavaScript's History and How it Led To ReactJS - The New Stack," 2015. [Online]. Available: <http://thenewstack.io/javascripts-history-and-how-it-led-to-reactjs/>. [Accessed: 14-Jan-2016].
- [56] Z. Hemel, "Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews," 2013. [Online]. Available: <http://www.infoq.com/news/2013/06/facebook-react>. [Accessed: 14-Jan-2016].
- [57] C. Abernathy, "Open source in 2015: A year of growth," 2015. [Online]. Available: <https://code.facebook.com/posts/1236426443040107/open-source-in-2015-a-year-of-growth/>.
- [58] Facebook, "Sites Using React," 2016. [Online]. Available: <https://github.com/facebook/react/wiki/Sites-Using-React>.
- [59] W3C, "W3C Document Object Model." [Online]. Available: <https://www.w3.org/DOM/#what>. [Accessed: 14-Jan-2016].
- [60] A. Fedosejev, *React.js Essentials*. Packt Publishing, 2015.
- [61] R. Harris, "What's the difference between React and Ractive? | Blog | Ractive.js," 2014. [Online]. Available: <http://blog.ractivejs.org/posts/whats-the-difference-between-react-and-ractive/>. [Accessed: 14-Jan-2016].
- [62] BehindTheCode, "An Introduction to React.js — Instrument — Independent Digital Creative Agency in Portland, Oregon," 2015. [Online]. Available: <http://www.instrument.com/latest/an-introduction-to-react-js>. [Accessed: 14-Jan-2016].
- [63] React, "Working With the Browser | React." [Online]. Available: <http://facebook.github.io/react/docs/working-with-the-browser.html>. [Accessed: 14-Jan-2016].
- [64] D. Nutels, "Server-Side Rendering With React, Node And Express – Smashing Magazine," 2016. [Online]. Available: <https://www.smashingmagazine.com/2016/03/server-side-rendering-react-node-express/>. [Accessed: 16-Apr-2016].
- [65] A. Ivantsov, "Why we use React for building single page applications | Graph," 2015. [Online]. Available: <https://www.graph.uk/insight/why-we-use-reactjs-for-building-single-page-applications>. [Accessed: 13-Apr-2016].
- [66] React, "JSX in Depth | React." [Online]. Available: <https://facebook.github.io/react/docs/jsx-in-depth.html>. [Accessed: 14-Jan-2016].
- [67] C. de S. Antonio, *Pro React*. Apress, 2015.
- [68] React, "React Native | A framework for building native apps using React." [Online]. Available: <https://facebook.github.io/react-native/>. [Accessed: 14-Jan-2016].
- [69] T. Occhino, "React Native: Bringing modern web techniques to mobile," 2015. [Online]. Available: <https://code.facebook.com/posts/1014532261909640/react-native-bringing-modern-web-techniques-to-mobile/>.
- [70] W. Mociun, "Going mobile with React Native - React Kung Fu." [Online]. Available: <http://reactkungfu.com/2015/07/going-mobile-with-react-native/>. [Accessed: 14-Jan-2016].
- [71] B. Alpert, "New React Developer Tools | React," 2015. [Online]. Available: <https://facebook.github.io/react/blog/2015/09/02/new-react-developer-tools.html>. [Accessed: 16-Apr-2016].
- [72] Jest, "Jest · Painless JavaScript Unit Testing," 2016. [Online]. Available: <http://facebook.github.io/jest/>. [Accessed: 11-Apr-2016].

- [73] React, "Tutorial – React · Jest," 2016. [Online]. Available: <https://facebook.github.io/jest/docs/tutorial-react.html>. [Accessed: 16-Apr-2016].
- [74] React, "Test Utilities | React," 2016. [Online]. Available: <https://facebook.github.io/react/docs/test-utils.html>. [Accessed: 15-Apr-2016].
- [75] R. Nicholus, "The Future of Web Development - React, Falcor, and ES6 | Widen Engineering," 2015. [Online]. Available: <http://engineering.widen.com/blog/future-of-the-web-react-falcor/>. [Accessed: 15-Apr-2016].
- [76] Node.js, "Node.js." [Online]. Available: <https://nodejs.org/en/>. [Accessed: 16-Feb-2016].
- [77] Google, "Chrome V8 | Google Developers." [Online]. Available: <https://developers.google.com/v8/>. [Accessed: 16-Feb-2016].
- [78] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to build high-performance network programs," *IEEE Internet Comput.*, vol. 14, no. 6, pp. 80–83, 2010.
- [79] P. Teixeira, *Professional Node.js: Building Javascript Based Scalable Software*. John Wiley & Sons, 2012.
- [80] D. Howard, *Node.js for PHP Developers*. "O'Reilly Media, Inc.," 2012.
- [81] S. Mohan, *Node.js Essentials*. Packt Publishing, 2014.
- [82] Node.js, "v0.0.1," 2009. [Online]. Available: <https://github.com/nodejs/node-v0.x-archive/releases/tag/v0.0.1>.
- [83] R. R. McCune, "Node. js Paradigms and Benchmarks," *Striegel, Gr. Os F*, vol. 11, pp. 1–6, 2011.
- [84] JoyentInc, "Elastic Container Infrastructure - Home - Joyent." [Online]. Available: <https://www.joyent.com/>. [Accessed: 16-Feb-2016].
- [85] Node.js, "Porting Node to Windows With Microsoft's Help | Node.js." [Online]. Available: <https://nodejs.org/en/blog/uncategorized/porting-node-to-windows-with-microsofts-help/>. [Accessed: 19-Feb-2016].
- [86] Flickr, "Flickr - Photo Sharing." [Online]. Available: <https://www.flickr.com/>.
- [87] A. Harris, "The Birth of Node: Where Did it Come From? Creator Ryan Dahl Shares the History | SiliconANGLE," 2013. [Online]. Available: <http://siliconangle.com/blog/2013/04/01/the-birth-of-node-where-did-it-come-from-creator-ryan-dahl-shares-the-history/>. [Accessed: 16-Feb-2016].
- [88] D. Kegel, "Introduction to non-blocking I/O." [Online]. Available: <http://www.kegel.com/dkftpbench/nonblocking.html>. [Accessed: 17-Feb-2016].
- [89] Tornado, "Asynchronous and non-Blocking I/O — Tornado 4.3.dev1 documentation." [Online]. Available: <http://tornadokevinlee.readthedocs.org/en/latest/guide/async.html>. [Accessed: 17-Feb-2016].
- [90] L. Orsini, "What You Need To Know About Node.js - ReadWrite," 2013. [Online]. Available: <http://readwrite.com/2013/11/07/what-you-need-to-know-about-nodejs>. [Accessed: 16-Feb-2016].
- [91] AngularJS, "AngularJS — Superheroic JavaScript MVW Framework." [Online]. Available: <https://angularjs.org/>. [Accessed: 19-Feb-2016].
- [92] AngularJS, "AngularJS: Developer Guide: Introduction." [Online]. Available: <https://docs.angularjs.org/guide/introduction>. [Accessed: 19-Feb-2016].
- [93] P. Kozłowski, *Mastering Web Application Development with AngularJS*. Packt Publishing Ltd, 2013.
- [94] M. Hevery, "Hello World, Angular is here," 2009. [Online]. Available:

- <http://misko.hevery.com/2009/09/28/hello-world-angular-is-here/>. [Accessed: 19-Feb-2016].
- [95] C. McKeachie, "Choosing a JavaScript MVC Framework," 2013. [Online]. Available: <http://www.funnyant.com/choosing-javascript-mvc-framework/>. [Accessed: 19-Feb-2016].
 - [96] C. Hamilton, "A sneak peek at the radically new Angular 2.0 - JAXenter," 2014. [Online]. Available: <https://jaxenter.com/angular-2-0-112094.html>. [Accessed: 19-Feb-2016].
 - [97] D. Iffland, "AngularJS 2.0 Details Emerge," 2014. [Online]. Available: http://www.infoq.com/news/2014/10/angular-2-atscript?utm_campaign=infoq_content&utm_source=infoq&utm_medium=feed&utm_term=global. [Accessed: 19-Feb-2016].
 - [98] B. Green and S. Seshadri, *AngularJS*. "O'Reilly Media, Inc.," 2013.
 - [99] J. Shore, "Dependency Injection Demystified," 2006. [Online]. Available: <http://www.jamesshore.com/Blog/Dependency-Injection-Demystified.html>. [Accessed: 01-Mar-2016].
 - [100] M. Rouse, "What is loose coupling?," 2011. [Online]. Available: <http://searchnetworking.techtarget.com/definition/loose-coupling>. [Accessed: 01-Mar-2016].
 - [101] M. Seemann, "Dependency Injection is Loose Coupling," 2010. [Online]. Available: <http://blog.ploeh.dk/2010/04/07/DependencyInjectionisLooseCoupling/>. [Accessed: 01-Mar-2016].
 - [102] Chandermani, *AngularJS by Example*. Packt Publishing Ltd, 2015.
 - [103] A. Freeman, *Pro AngularJS*. Apress, 2014.
 - [104] V. Karpov and D. Netto, *Professional AngularJS*. Wiley, 2015.
 - [105] AngularJS, "Writing Directives," 2012. [Online]. Available: <https://www.youtube.com/watch?v=Wqmel5fZcho&list=TLJUxRYO87UWA>. [Accessed: 19-Apr-2016].
 - [106] Ember, "Ember.js - A framework for creating ambitious web applications." [Online]. Available: <http://emberjs.com/about/>. [Accessed: 02-Mar-2016].
 - [107] M. Kelonye, *Mastering Ember.js*. Packt Publishing Ltd, 2014.
 - [108] M. Bodmer, *Instant Ember.js Application Development How-to*. Packt Publishing Ltd, 2013.
 - [109] B. Bibeault, Y. Katz, and A. De Rosa, *Jquery in Action*, vol. 91. Manning Publications, 2015.
 - [110] R. Bigg and Y. Katz, *Rails 3 in Action*. Manning Publications, 2011.
 - [111] R. Bigg, Y. Katz, R. Skinner, and S. Klabnik, *Rails 4 in Action*, vol. 28. Manning Publications, 2014.
 - [112] J. Cravens and T. Q. Brady, *Building Web Apps with Ember.js*. O'Reilly Media, 2014.
 - [113] Y. Katz, "Ember.js - Glimmer's In Canary, Test Your Apps!," 2015. [Online]. Available: <http://emberjs.com/blog/2015/05/05/glimmer-merging.html>. [Accessed: 02-Mar-2016].
 - [114] Y. Katz, "Implement Glimmer Engine," 2015. [Online]. Available: <https://github.com/emberjs/ember.js/pull/10501>.
 - [115] Y. Katz, "Ember with Glimmer," 2015. [Online]. Available: <https://www.youtube.com/watch?v=5Xd9cnndB74>. [Accessed: 19-Apr-2016].
 - [116] Ember, "Ember.js - Users." [Online]. Available: <http://emberjs.com/ember-users/>.

- [Accessed: 02-Mar-2016].
- [117] A. Osmani, *Developing Backbone.js Applications*. "O'Reilly Media, Inc.," 2013.
 - [118] J. Sugrue, *Beginning Backbone.js*. Apress, 2014.
 - [119] V. Mirgorod, *Backbone.js Cookbook*. Packt Publishing Ltd, 2013.
 - [120] GoogleTrends, "Google Trends - Web Search interest: ember.js, angularjs, backbone.js, node.js, react js - Worldwide, Jul 2009 - Feb 2016." [Online]. Available: <https://www.google.com/trends/explore?hl=en-US#q=ember.js%2C angularjs%2C backbone.js%2C Node.js%2C React js&date=7%2F2009 80m&cmpt=q&tz=Etc%2FGMT>. [Accessed: 03-Mar-2016].
 - [121] D. Chaffey, "Mobile Marketing Statistics 2016," 2015. [Online]. Available: <http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/>. [Accessed: 20-Apr-2016].
 - [122] J. Gonzalez, *Mobile First Design with Html5 and CSS3*. Packt Publishing Ltd, 2013.
 - [123] MorganStanleyResearch, "Mobile-stats-vs-desktop-users-global.png." [Online]. Available: <http://www.smartinsights.com/wp-content/uploads/2014/03/Mobile-stats-vs-desktop-users-global.png>. [Accessed: 20-Apr-2016].
 - [124] B. Eisenman, *Learning React Native: Building Native Mobile Apps with JavaScript*. O'Reilly Media, 2015.
 - [125] React, "reactnative.png." [Online]. Available: http://fbcdn-dragon-a.akamaihd.net/hphotos-ak-xfa1/t39.2365-6/11057107_805781416175147_684201729_n.png. [Accessed: 16-Apr-2016].
 - [126] E. Holmes and T. Bray, *Getting Started with React Native*. PACKT PUB, 2015.
 - [127] B. Alpert, "Introducing React Native | React," 2015. [Online]. Available: <https://facebook.github.io/react/blog/2015/03/26/introducing-react-native.html>. [Accessed: 16-Apr-2016].
 - [128] Facebook, "React Developer Tools | Mozilla Firefox," 2016. [Online]. Available: <https://addons.mozilla.org/en-GB/firefox/addon/react-devtools/>.
 - [129] Facebook, "React Developer Tools | Google Chrome," 2016. [Online]. Available: <https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en>.
 - [130] React, "flux.png." [Online]. Available: <https://reactweek.com/images/flux.png>. [Accessed: 16-Apr-2016].
 - [131] Facebook, "Flux | Application Architecture for Building User Interfaces," 2014. [Online]. Available: <https://facebook.github.io/flux/>. [Accessed: 13-Apr-2016].
 - [132] B. Fisher and J. Chen, "Flux: An Application Architecture for React | React." [Online]. Available: <https://facebook.github.io/react/blog/2014/05/06/flux.html>. [Accessed: 16-Apr-2016].
 - [133] K. Wheeler, "Getting To Know Flux, the React.js Architecture," 2014. [Online]. Available: <https://scotch.io/tutorials/getting-to-know-flux-the-react-js-architecture>.
 - [134] Flux, "flux-simple-f8-diagram-1300w.png," 2015. [Online]. Available: <https://facebook.github.io/flux/img/flux-simple-f8-diagram-1300w.png>. [Accessed: 13-Apr-2016].
 - [135] A. Banks and E. Porcello, *Learning React: Functional Web Development with React and Flux*. O'Reilly Media, 2016.
 - [136] J. Rubin and D. Chisnell, *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. John Wiley & Sons, 2011.
 - [137] C. M. Barnum, *Usability Testing Essentials: Ready, Set...Test!* Elsevier, 2010.

- [138] J. Nielsen and H. Loranger, *Prioritizing Web Usability*. Pearson Education, 2006.
- [139] J. Hammond, T. Gross, and J. Wesson, *Usability: Gaining a Competitive Edge*. Springer, 2013.
- [140] J. Nielsen, *Usability Engineering*. Morgan Kaufmann, 1994.
- [141] S. Desikan, *Software Testing: Principles and Practice*. Pearson Education India, 2006.
- [142] SoftwareTestingFundamentals, "Differences Between Black Box Testing and White Box Testing," 2010. [Online]. Available: <http://softwaretestingfundamentals.com/differences-between-black-box-testing-and-white-box-testing/>. [Accessed: 12-Apr-2016].
- [143] J. Nielsen, "Usability 101: Introduction to Usability," 2012. [Online]. Available: <https://www.nngroup.com/articles/usability-101-introduction-to-usability/>. [Accessed: 13-Apr-2016].
- [144] S. Idler, "Principles of Website Usability | 5 Key Principles Of Good Website Usability," 2013. [Online]. Available: <http://blog.crazyegg.com/2013/03/26/principles-website-usability/>. [Accessed: 04-Dec-2015].
- [145] K. Chandler and K. Hyatt, *Customer-centered Design: A New Approach to Web Usability*. Prentice Hall Professional, 2003.
- [146] W3C, "World Wide Web Consortium (W3C)." [Online]. Available: <https://www.w3.org/>. [Accessed: 14-Apr-2016].
- [147] WorldWideWebConsortium, "Accessibility - W3C." [Online]. Available: <https://www.w3.org/standards/webdesign/accessibility>. [Accessed: 14-Apr-2016].
- [148] A. Badre, *Shaping Web Usability: Interaction Design in Context*. Addison-Wesley Professional, 2002.
- [149] B. Liebowitz, "Website Usability: Virtual Elephants of the Internet Room | Bosmol Social Media News," 2014. [Online]. Available: <http://bosmol.com/2014/02/website-usability-virtual-elephants-of-the-internet-room.html#.Vw2kzkf979B>. [Accessed: 13-Apr-2016].
- [150] J. Nielsen, *Designing Web Usability, Issue 667*. New Riders, 2000.
- [151] M. McDonnell, "Beginners guide on how to test your code," 2015. [Online]. Available: <http://www.integralist.co.uk/posts/guide-to-js-testing.html>. [Accessed: 11-Apr-2016].
- [152] SoftwareTestingFundamentals, "unittesting.jpg (JPEG Image, 290 x 272 pixels)," 2011. [Online]. Available: <http://softwaretestingfundamentals.com/wp-content/uploads/2010/12/unittesting.jpg>. [Accessed: 12-Apr-2016].
- [153] J. Gao, H.-S. J. Tsao, and Y. Wu, *Testing and Quality Assurance for Component-based Software*. Artech House, 2003.
- [154] M. Rouse, "What is unit testing?," 2014. [Online]. Available: <http://searchsoftwarequality.techtarget.com/definition/unit-testing>. [Accessed: 12-Apr-2016].
- [155] J. Teminio, "Unit Testing Tutorial Part I: Introduction to PHPUnit — Juan Treminio — Dallas based senior web developer," 2013. [Online]. Available: <https://jtreminio.com/2013/03/unit-testing-tutorial-introduction-to-phpunit/>. [Accessed: 11-Dec-2015].
- [156] SoftwareTestingFundamentals, "Unit Testing." [Online]. Available: <http://softwaretestingfundamentals.com/unit-testing/>. [Accessed: 14-Apr-2016].
- [157] K. L. Lee, *JavaScript Unit Testing*. CreateSpace Independent Publishing Platform, 2015.
- [158] H. Saleh, *JavaScript Unit Testing*. Packt Publishing Ltd, 2013.
- [159] M. E. Trostler, *Testable JavaScript*. "O'Reilly Media, Inc.," 2013.

- [160] Microsoft, "Integration Testing," 2003. [Online]. Available: [https://msdn.microsoft.com/en-us/library/aa292128\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/aa292128(v=vs.71).aspx). [Accessed: 12-Apr-2016].
- [161] SoftwareTestingFundamentals, "Integration Testing." [Online]. Available: <http://softwaretestingfundamentals.com/integration-testing/>. [Accessed: 14-Apr-2016].
- [162] SoftwareTestingFundamentals, "System Testing," 2011. [Online]. Available: <http://softwaretestingfundamentals.com/system-testing/>. [Accessed: 14-Apr-2016].
- [163] SoftwareTestingFundamentals, "Acceptance Testing." [Online]. Available: <http://softwaretestingfundamentals.com/acceptance-testing/>. [Accessed: 14-Apr-2016].
- [164] M. Rouse, "What is acceptance test?," 2014. [Online]. Available: <http://searchsoftwarequality.techtarget.com/definition/acceptance-test>. [Accessed: 12-Apr-2016].
- [165] E. Hahn, *JavaScript Testing with Jasmine*. "O'Reilly Media, Inc.," 2013.
- [166] MochaJS, "Mocha - the fun, simple, flexible JavaScript test framework," 2016. [Online]. Available: <http://mochajs.org/>. [Accessed: 11-Apr-2016].
- [167] Karma, "Karma - Spectacular Test Runner for Javascript," 2016. [Online]. Available: <https://karma-runner.github.io/0.13/index.html>. [Accessed: 11-Apr-2016].
- [168] Jasmine, "Jasmine: Behavior-Driven JavaScript," 2016. [Online]. Available: <http://jasmine.github.io/>. [Accessed: 11-Apr-2016].
- [169] QUnit, "QUnit," 2016. [Online]. Available: <https://qunitjs.com/>. [Accessed: 11-Apr-2016].
- [170] Jest, "opengraph.png." [Online]. Available: <https://facebook.github.io/jest/img/opengraph.png>. [Accessed: 15-Apr-2016].
- [171] M. Rowett, "Painless Javascript testing? Surely you Jest! | realestate.com.au Tech Blog," 2015. [Online]. Available: <http://techblog.realestate.com.au/painless-javascript-testing-surely-you-jest/>. [Accessed: 15-Apr-2016].
- [172] Mocha, "image62.png." [Online]. Available: <http://www.tutorialsavvy.com/wp-content/uploads/2015/09/image62.png>. [Accessed: 15-Apr-2016].
- [173] P. Teixeira, *Using Node.js for Ui Testing*. Packt Publishing Ltd, 2013.
- [174] R. Roemer, *Backbone.js Testing*. Packt Publishing Ltd, 2013.
- [175] Karma, "banner.png." [Online]. Available: <https://karma-runner.github.io/assets/img/banner.png>. [Accessed: 15-Apr-2016].
- [176] V. Jina, "JavaScript Test Runner," p. 61, 2013.
- [177] T. Chaplin, *AngularJS Test-driven Development*. Packt Publishing Ltd, 2015.
- [178] Jasmine, "jasmine.jpg." [Online]. Available: <https://tobythetesterblog.files.wordpress.com/2014/09/jasmine.jpg>. [Accessed: 15-Apr-2016].
- [179] P. Ragonha, *Jasmine JavaScript Testing - Second Edition*. Packt Publishing Ltd, 2015.
- [180] D. W. Frank, "Jasmine 1.0 Released | Pivotal P.O.V.," 2010. [Online]. Available: <https://blog.pivotal.io/pivotal-labs/labs/jasmine-1-0-released>. [Accessed: 15-Apr-2016].
- [181] M. Sethi, *Jasmine Cookbook*. Packt Publishing Ltd, 2015.
- [182] QUnit, "qunit.png." [Online]. Available: <https://jordankasper.com/js-testing/images/qunit.png>. [Accessed: 15-Apr-2016].
- [183] D. Sheiko, *Instant Testing with QUnit*. Packt Publishing Ltd, 2013.

- [184] J. Kwok, "The Netflix Tech Blog: Netflix Likes React," 2015. [Online]. Available: <http://techblog.netflix.com/2015/01/netflix-likes-react.html>. [Accessed: 14-Jan-2016].
- [185] Google, "Firebase - Build Extraordinary Apps." [Online]. Available: <https://www.firebase.com/>. [Accessed: 20-Apr-2016].
- [186] M. Balnaves and P. Caputi, *Introduction to Quantitative Research Methods: An Investigative Approach*. SAGE Publications, 2001.
- [187] W. P. Vogt, *Quantitative Research Methods for Professionals*. Pearson/Allyn and Bacon, 2007.
- [188] W. K. Hoy and C. M. Adams, *Quantitative Research in Education: A Primer*. SAGE Publications, 2015.
- [189] R. V. Labaree, "Research Guides: Organizing Your Social Sciences Research Paper: Quantitative Methods."
- [190] SurveyMonkey, "Welcome to SurveyMonkey!" [Online]. Available: <https://www.surveymonkey.com/>. [Accessed: 15-Apr-2016].
- [191] AgileMethodology, "Agile Methodology, Development Methodologies | What Is Agile Methodology?," 2008. [Online]. Available: <http://agilemethodology.org/>. [Accessed: 08-Dec-2015].
- [192] K. Schwaber, *Agile Project Management with Scrum*. Microsoft Press, 2004.
- [193] S. Millett, J. Blankenship, and M. Bussa, *Pro Agile .NET Development with SCRUM*. Apress, 2011.
- [194] ScrumReferenceCard, "scrum-iteration-detail.png." [Online]. Available: <http://li657-9.members.linode.com/scrumreferencecard-content/uploads/2013/03/scrum-iteration-detail.png>. [Accessed: 18-Apr-2016].
- [195] M. James, "Scrum Reference Card | Scrum Reference Card," 2015. [Online]. Available: <http://scrumreferencecard.com/scrum-reference-card/>. [Accessed: 18-Apr-2016].
- [196] Babel, "Babel · The compiler for writing next generation JavaScript," 2016. [Online]. Available: <https://babeljs.io/>. [Accessed: 18-Apr-2016].
- [197] B. McCormick, "ES5, ES6, ES2016, ES.Next: What's going on with JavaScript versioning?" [Online]. Available: <http://benmccormick.org/2015/09/14/es5-es6-es2016-es-next-whats-going-on-with-javascript-versioning/>. [Accessed: 18-Apr-2016].
- [198] Gulp, "gulp.js - the streaming build system," 2016. [Online]. Available: <http://gulpjs.com/>. [Accessed: 18-Apr-2016].
- [199] V. Beal, "What is Object-Oriented Programming? Webopedia Definition," 2015. [Online]. Available: http://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html. [Accessed: 19-Apr-2016].
- [200] Statista, "Age distribution of internet users worldwide 2014 | Statistic," 2014. [Online]. Available: <http://www.statista.com/statistics/272365/age-distribution-of-internet-users-worldwide/>. [Accessed: 17-Apr-2016].
- [201] Statista, "Statista - The Statistics Portal for Market Data, Market Research and Market Studies." [Online]. Available: <http://www.statista.com/>. [Accessed: 17-Apr-2016].
- [202] B. Sullivan and V. Liu, *Web Application Security, A Beginner's Guide*. McGraw Hill Professional, 2011.
- [203] Ponemon, "State of Web Application Security Executive Summary Sponsored by Cenzic & Barracuda Networks," no. February, 2011.

10. Bibliography

Dayley, B. (2014). *Node.js, MongoDB, and AngularJS Web Development*. Pearson Education.

Odell, D. (2014). *Pro JavaScript Development: Coding, Capabilities, and Tooling*. Apress.

Eisenman, B. (2015). *Learning React Native*. O'Reilly Media, Inc.

De Sousa Antonio, C. (2015). *Pro React*. Apress.

Fedosejev, A. (2015). *React.js Essentials*. Packt Publishing.

Holmes, E., & Bray, T. (2015). *Getting Started with React Native*. Packt Publishing.

Banks, A., & Porcello, E. (2016). *Learning React: Functional Web Development with React and Flux*. O'Reilly Media.

Syed, B. (2014). *Beginning Node.js*. Apress.

Teixeira, P. (2012). *Professional Node.js: Building Javascript Based Scalable Software*. John Wiley & Sons.

Mohan, S. (2014). *Node.js Essentials*. Packt Publishing.

Chandermani. (2015). *AngularJS by Example*. Packt Publishing Ltd.

Green, B., & Seshadri, S. (2013). *AngularJS*. O'Reilly Media, Inc.

Karpov, V., & Netto, D. (2015). *Professional AngularJS*. Wiley.

Cravens, J., & Brady, T. Q. (2014). *Building Web Apps with Ember.js*. O'Reilly Media.

Osmani, A. (2013). *Developing Backbone.js Applications*. O'Reilly Media, Inc.

Kozlowski, P. (2013). *Mastering Web Application Development with AngularJS*. Packt Publishing Ltd.

Saleh, H. (2013). *JavaScript Unit Testing*. Packt Publishing Ltd.

Johansen, C. (2010). *Test-Driven JavaScript Development*. Addison-Wesley Professional.

Lee, K. L. (2015). *JavaScript Unit Testing*. CreateSpace Independent Publishing Platform.

Trostler, M. E. (2013). *Testable JavaScript*. O'Reilly Media, Inc.

Chandler, K., & Hyatt, K. (2003). *Customer-centered Design: A New Approach to Web Usability*. Prentice Hall Professional.

Nielsen, J., & Loranger, H. (2006). *Prioritizing Web Usability*. Pearson Education.

Rubin, J., & Chisnell, D. (2011). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. John Wiley & Sons.

Hammond, J., Gross, T., & Wesson, J. (2013). *Usability: Gaining a Competitive Edge*. Springer.

Balnaves, M., & Caputi, P. (2001). *Introduction to Quantitative Research Methods: An Investigative Approach*. SAGE Publications.

Newman, I., & Benz, C. R. (1998). *Qualitative-quantitative Research Methodology: Exploring the Interactive Continuum*. SIU Press.

Vogt, W. P. (2007). *Quantitative Research Methods for Professionals*. Pearson/Allyn and Bacon.

Shore, J., & Chromatic. (2007). *The Art of Agile Development*. O'Reilly Media, Inc.

Gao, J., Tsao, H.-S. J., & Wu, Y. (2003). *Testing and Quality Assurance for Component-based Software*. Artech House.

Desikan, S. (2006). *Software Testing: Principles and Practice*. Pearson Education India.

Gupta, R. K., Prajapati, H., & Singh, H. (2015). *Test-Driven JavaScript Development*. Packt Publishing.

Chaplin, T. (2015). *AngularJS Test-driven Development*. Packt Publishing Ltd.

Monteiro, F. (2014). *Learning Single-page Web Application Development*. Packt Publishing Ltd.

Appendix 1: Testing Feedback Questionnaires

Name	Chloe Dowdeswell
Age	25
Gender	Female
How relevant is Project to you?	Extremely, I love online shopping & gaming
Preferred Aesthetics/Theme?	Anything involving purple is amazing
Accessible via Mobile/Tablet?	Preferably
Most important aspect of the website?	Either loading times or aesthetics
Would you use the created application?	Yes, definitely!
Do you think the App would be successful within your demographic?	Most likely
How was your overall user experience of the application?	Really good, enjoyed giving the application a go
Is there anything you would change or add to the application?	Only thing I'd change is fixing the checkout page
What was your favourite feature of the application?	Definitely adding/removing content from my shop
Were there any issues you encountered through general use?	Only problem was having to refresh page sometimes after visiting checkout
How would you rate the project? (1-10)	9 or 10

Name	Luke Fongenie
Age	25
Gender	Male
How relevant is Project to you?	Very, I'm a big gamer and prefer to shop online as opposed to the high street
Preferred Aesthetics/Theme?	Not too bothered, I'm not a huge fan of purple
Accessible via Mobile/Tablet?	Yes, I love my phone!
Most important aspect of the website?	Performance and Usability

Would you use the created application?	Probably, yes
Do you think the App would be successful within your demographic?	Most likely
How was your overall user experience of the application?	Great, it worked really well
Is there anything you would change or add to the application?	Maybe a direct contact form?
What was your favourite feature of the application?	Adding and removing items from the cart
Were there any issues you encountered through general use?	No, nothing that stuck out to me
How would you rate the project? (1-10)	Probably 9

Name	Nesli Kent
Age	37
Gender	Female
How relevant is Project to you?	I enjoy online shopping!
Preferred Aesthetics/Theme?	Bright and colourful
Accessible via Mobile/Tablet?	Not bothered
Most important aspect of the website?	Visually pleasing and it didn't take AGES to load
Would you use the created application?	Probably if the situation arose
Do you think the App would be successful within your demographic?	Amongst gamers and online shoppers, yes
How was your overall user experience of the application?	I loved it!
Is there anything you would change or add to the application?	I'd probably make it not just gaming related so other forms of products could be added
What was your favourite feature of the application?	Creating my own store!
Were there any issues you encountered	Not that I noticed

through general use?	
How would you rate the project? (1-10)	8

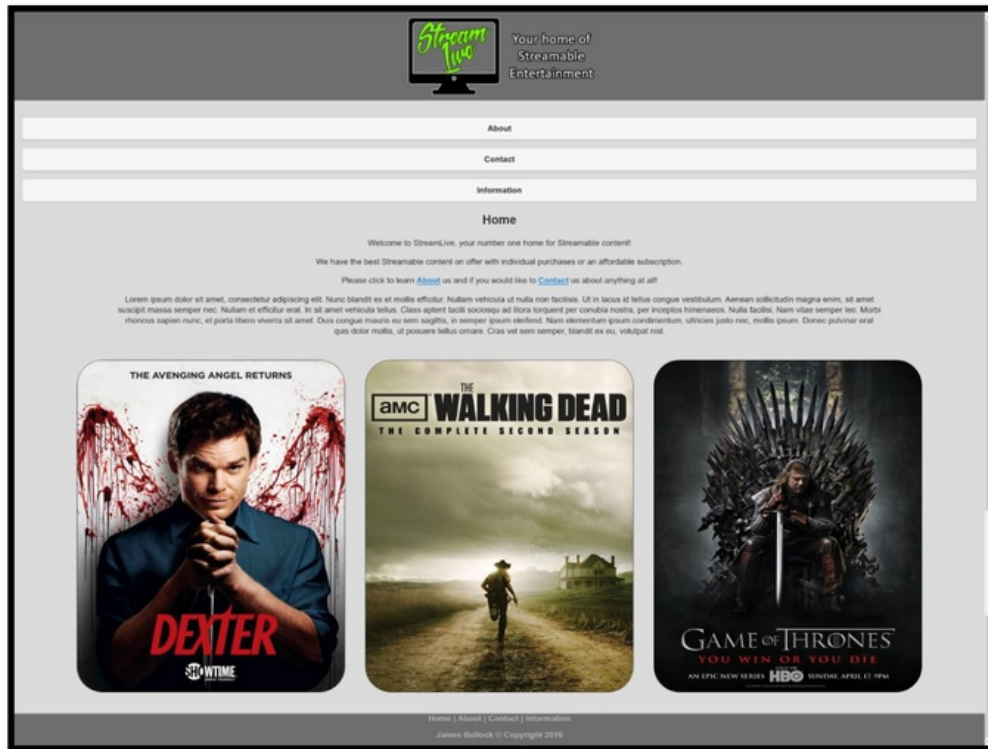
Name	Dennis Bullock
Age	82
Gender	Male
How relevant is Project to you?	Not entirely, I love my PC but not gaming on it!
Preferred Aesthetics/Theme?	Anything besides white is great by me!
Accessible via Mobile/Tablet?	Don't care
Most important aspect of the website?	Features
Would you use the created application?	If I wanted to purchase a gift, yes
Do you think the App would be successful within your demographic?	Probably not
How was your overall user experience of the application?	Was okay, not too confusing and worked well overall
Is there anything you would change or add to the application?	Maybe add a Menu, or a Help/Information button?
What was your favourite feature of the application?	Adding and updating the product information
Were there any issues you encountered through general use?	Was abit confused at first but after a minute of use, everything was fine
How would you rate the project? (1-10)	7

Name	Cerys Gibbins
Age	21
Gender	Female
How relevant is Project to you?	Fairly, I love shopping online
Preferred Aesthetics/Theme?	Perfect, nice and vibrant
Accessible via Mobile/Tablet?	Yes
Most important aspect of the website?	Features or performance

Would you use the created application?	Yes, if I was looking for a birthday present for someone
Do you think the App would be successful within your demographic?	I think so
How was your overall user experience of the application?	It was brilliant, I would definitely recommend it to friends
Is there anything you would change or add to the application?	Possibly product filters or a search bar
What was your favourite feature of the application?	Store creation and login features
Were there any issues you encountered through general use?	I had to refresh the store after returning from the checkout
How would you rate the project? (1-10)	8-9

Appendix 2: Original Project Prototype

Prototype Homepage



Prototype About Page



Your home of
Streamable
Entertainment

[Home](#)

[Contact](#)

[Information](#)

[About](#)

StreamLive has been a registered company since 2015, allowing users to experience all the amazing streamable content we have on offer at an affordable price.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc blandit ex et mollis efficitur. Nullam vehicula ut nulla non facilisis. Ut in lacus id felis congue vestibulum. Aenean sollicitudin magna sem, sit amet luctus massa semper nec. Nunc et efficitur arcu. In sit amet vehicula tellus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Nulla facilis. Nam vitae semper leo. Morbi morous sapien nunc, et porta libero viverra sit amet. Duis congue mauris eu sem sagittis, in semper quam eleifend. Nam eleventhium ipsum cononertum, utroque justo nec, mollis ipsum. Donec pulvinar erat quis dolor mollis, ut posuere tellus ornare. Cras vel sem semper, blandit ex eu, volutpat nisi.

Cras suscipit ligula vitae metus venenatis, et faucibus odio pharetra. Duis a mollis dolor vitae tempor auctor. Pellentesque eget dui est. Curabitur sed rutrum neque. Curabitur sit amet cursus diam, sit congue dui. Ut sodales diam ut cursus rhoncus. Curabitur sit amet nisi quis elit vulputate pellentesque non non velit. Maecenas lacinia augue ante, conmodo consequat purus finibus eu. Donec neque nulla, pharetra id cursus sit amet, loorits ac erat. Etiam sodales consequat justo, a pretium nunc laoreet id. Vestibulum vitae risus eget turpis imperdiet malesuada. Morbi pulvinar ligula non erat bibendum blandit. Maecenas ut orci vitae dolor vehicula pellentesque. Sed eu velit interdum, dictum odio sit amet, accumsan mauris. Phasellus aliquam lacus turpis ut varius.

In hac habitasse platea dictumst. Aliquam consectetur erat eget lorem pretium finibus. Mauris sed cursus ex. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Integer eu velit quam. Quisque ac tortor vehicula, gravida arcu sit amet, auctor eros. Etiam vulputate scelerisque arcu, nec vulputate nulla aliquam laoreet. In hac habitasse platea dictumst. Donec nec ligula et justo blandit tristique. Vivamus volutpat euismodi quam at sodales. Nulla nec mi nulla.

[Back to homepage](#)

[Home](#) | [About](#) | [Contact](#) | [Information](#)
James Bullock © Copyright 2016

Prototype Contact Page



Your home of
Streamable
Entertainment

[Home](#)

[About](#)

[Information](#)

[Contact](#)

Please fill out the form below if you would like to contact us.

Name:

Age:

Email:

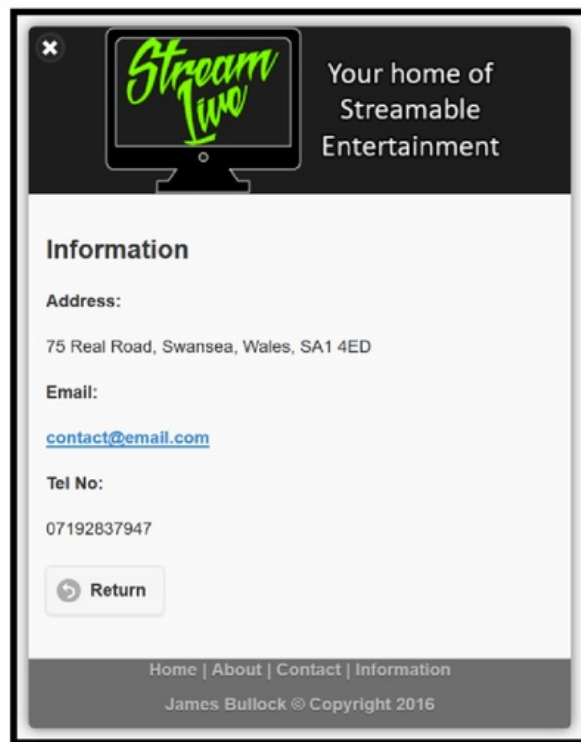
Message:

[Submit](#)

[Back to homepage](#)

[Home](#) | [About](#) | [Contact](#) | [Information](#)
James Bullock © Copyright 2016

Prototype Information Popup



Appendix 3: List of Requirements

List of Requirements determined by both Research and the Survey results detailed in *Chapter 5.1*. Requirement ratings are graded as **Mandatory** (M), **Highly Desirable** (HD) or **Desirable** (D).

Functional Requirements

Requirement	Preference
Allow Users to create an account	M
Allow Users to login to created account	M
Allow Shop owners to add & remove products	M
Allow Users to add products to Order	M
Allow Users to remove products from Order	M
Allow Users to access specific shops	M
Allow Shop owners to alter product information	M
Notify Users if a product is out of stock	HD
Allow Users to review their Order and make changes to it	HD
Allow Users to purchase their Order	HD
Provide Shop owners with sample content to fill store	D
Implement product filters or a search method	D
Account recovery options	D
Refresh store for any reason	D

Non-Functional Requirements

Requirement	Preference
Optimal Performance – Load times,	HD
Reliability – Always available.	D
Security	D
Content	HD

Usability	M
Features	M
Accessibility	D