INVESTIGATION INTO THE EXTENT AND LIMITATIONS OF DYNAMISM AND INTERACTION THROUGH WEB APPLICATIONS

By

Gerard Byrne

Supervisor: Dr Luke Raeside

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF B.SC IN COMPUTING
AT

THE INSTITUTE OF TECHNOLOGY BLANCHARDSTOWN
DUBLIN, IRELAND
MAY 2014

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of **B.Sc. in Computing** in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Dated: May 2014

Author: _____

Ger Byrne

Abstract

As new approaches to web development continue to appear, the World Wide Web as we know it is constantly evolving. Web applications are improving on their dynamic capabilities and web-user interaction is almost a standard feature on every contemporary website. The extent of this web app dynamism is not yet defined but some trends have emerged that offer an alternative approach to the conventional three-tier system. One noticeable trend involves more and more utilization of the client-side in a single-tier system. Interactive web applications have also progressed to become ubiquitous and this is mainly due to the growth in mobile technology and development. This recent lean towards the client-side approach and achievements made with web app interactivity is nowhere more emphasised that with the arrival of newly updated web technologies such as HTML5 and CSS3. With the introduction of these technologies the debate now rages over which type of application is best suited for the development, deployment and implementation of mobile applications. Which is the preferred option, native apps or web apps?

Table of Contents

Table of Figures

Table of Tables

1 Introduction and Background

1.1 Project Description

This project relates to current web development technologies in the area of Human Computer Interaction (HCI). It will examine the different techniques available when developing interactive web-based applications. The project will consist of two related approaches. Firstly, it will seek to find answers to all relevant research questions associated with proposed topic and secondly, it will involve the creation and development of an interactive web application for the purpose of highlighting the project's hypothesis. This web app will be developed to for both the desktop and mobile environments.

The aim of the web app will be to test users on mathematic equations in the form of a game. Its target demographic will be children who are in the process of learning and improving their mathematical skills. With this in mind there will be three levels of skills easy, normal and hard. As the user ascends up the three levels the degree of difficulty will also rise. The goal of designing an application will be to incorporate some of the new interactive technologies and APIs associated with HTML5 and CSS3 while also implementing current ones such as Javascript, JSON and the Document Object Model (DOM).

As these new technologies and APIs are still relatively in their infancy, it is estimated that a few problems may be encountered along the way. As it stands there are numerous issues rumoured to be found with some new APIs such as HTML5's Drag and Drop and Local storage. Most issues however can be overcome with the implementation of polyfills, fallbacks or shims. Polyfills or browser fallbacks, as they are also known, are Javascript implementations that aid HTML5 and CSS3 web application functionality in unsupportive browsers. Polyfill HTML5 compromises are also known as HTML5 shims and HTML5 shivs. Originally the purpose of these workarounds was to implement modern web apps in previous versions of browsers but they have become much more prevalent recently working in tandem with HTML5 APIs. In fact this is one of the issues explored in this thesis as it is believed that some new HTML5 APIs and CSS3 technologies will not fully function without the aid of one of the many, now available polyfills. How much this is the case and whether or not polyfills are needed for a fully functioned web app will indicate the state of interactivity with HTML5 and CSS3. Polyfills are not a HTML5 standard and if possible the desktop and mobile web apps will be intended to be developed without polyfills

or as few as possible. Similar to polyfills is the jQuery framework and it is also an aim to use as little of this as possible in the development process for fear that it may prevent any adequate analysis of HTML5's interactivity.

This projects development will aim to incorporate technologies such as HTML5, CSS3, JavaScript, JSON, and maybe more. It may also utilize techniques such as jQuery and the Document Object Model.

1.2 Background

In today's modern computing small-sized, user-friendly applications, or apps for short, have become much more prevalent. This is due to the ever growing availability and use of mobile and portable devices. Recent research by Cisco forecasts that "By the end of 2013, the number of mobile-connected devices will exceed the number of people on earth, and by 2017 there will be nearly 1.4 mobile devices per capita" [1]. This growth in mobile computing brings with it a great demand for applications that are simple to use, easy to navigate and up to date in content.

Other technological advances in the area of application development include the recent introduction of Microsoft's Windows 8 touchscreen operating system. This latest release of Windows has introduced Windows Store applications (Windows apps) to the desktop environment.

One important aspect to note is that in this field of development, various devices, on which apps operate, can support different platform architectures. This is a major factor when creating native platform-specific applications. For example, to develop and run an application, which is aimed at reaching a high volume of users, it must be capable of running on numerous independent platforms. These platforms include Android, IOS, Blackberry OS, Windows Phone 8 and Windows 8. This issue of cross-platform support can involve the re-coding of an app so as to be compatible with multiple architectures.

Another factor to consider is that as there are so many different types of mobile devices available, then there may be issues with application and hardware compatibility. For example, Google's Android operating system is designed to run on a wide range of devices such as mobile phones or tablets. These devices in turn may be constructed by various different manufactures. This implies that native applications must be compatible with a diverse range of device hardware regardless of its manufacturer.

In contrast to native apps, web-based application development is entering a new and exciting phase. It is currently witnessing new and improved revisions in technologies such as HTML5 and CSS3 and with the many of the available Javascript frameworks. This new chapter in development forms the basis for this project by prompting some interesting questions such as, can the advent of these enhanced web technologies provide web–based apps with the functionality that is able to equal and even surpass that of native applications?

1.3 Main Research Questions

As the title suggests this project is an exploration into interactive dynamic web development. This exploration will prompt many questions and all will be relevant to its outcome. One such question that the research will concern itself with is:

Can or will interactive web-based applications equal or surpass native applications in all areas such as functionality, performance and results achieved?

Whatever the outcome to this question, the thesis aims to describe the current status of both native and web-based applications and their relation to each other. If one form of application performs better than its counterpart, the project will investigate as to why this is so and it will analyse what the low-performing app needs to achieve better. The thesis will also weigh up the advantages and disadvantages of each format and will arrive at its conclusions based on this informed research.

As stated previously interactive programming through the web is reaching new heights with advancements in technologies such as HTML5 CSS3 and Javascript. These web technologies along with PHP, Java, SQL and the Document Object Model (DOM) programming interface will require further research. This will be for the purpose of making an informed decision as to which will be best applied when developing an interactive web-based application. This part of the research will ask:

What adequate technologies are required to develop an interactive dynamic web application?

What Interactive Web Application Programming Interfaces (APIs) will be required?

What is the scope and limitation of the technologies involved?

Other pertinent questions may include:

What issue are there with cross-browser compatibility?

What other similar interactive web-based applications are currently available in this field?

1.4 Aims & Objectives

The aim of this project is to acquire a better understanding of interactivity and dynamism in web development. It also has the intent of producing a user-friendly web-based application that will demonstrate these behaviours. It is also an objective to prove that with the aid of new technologies such as HTML5 and CSS3, web application performance can now rival that of its platform specific counterpart. It is expected that these result of this projects findings will pose the question as to where native application development is heading.

The completed web application will be hosted on a server and readily available to all visitors to the site. It is a goal of this website to demonstrate a fully functional, interactive, dynamic web-based app that is capable of running in a wide variety of web browsers.

1.5 Feasibility

This project does not seek funding or require any financial support so there will be no budget constraints. The feasibility of the project will therefore only focus on its completion within a designated timeframe and the scope and limitations of all the appropriate technologies involved.

At this preliminary stage it appears highly probable that the majority of this projects desktop and mobile application development will be implemented in HTML5, CSS3 and Javascript. Other technologies will be investigated but as one of the main advantages of HTML5 interactivity and web storage is its ability to cut down on the need for server-side technologies such as PHP and SQL, it is unlikely that they will be implemented. If this is possible HTML5 will be chosen as the app's core technology and this will require additional research so as to check its feasibility in relation to cross-browser support and performance in the mobile arena. As alluded to previously there may be some need for the implementation of workarounds such as polyfills and javascript frameworks. These will only be implemented as a last resort when no other option is available or feasible to achieve complete functionality of the application.

Another area of concern is the feasibility of the work involved within the given time frame. This is in terms of the volume of work required to bring the project to completion. This can be further broken down and outlined in the Project Plan & Timetable section. This section will also answer the question, what are the tasks required within this project and understanding these duties will also help determine the projects viability.

One more important question relating to feasibility is that of what are the overall objectives of this project? As it is an exploration and investigation, it is not completely clear at this stage how this will manifest itself. It is estimated that the best course of action is to evaluate interactivity, first-hand with the development and implementation of a web application and this undoubtedly seems to be the most likely course of action. This manifestation will take the form of a use-friendly, web-based, game application called MathMagic Basketball. MathMagic Basketball will be aimed at children who are eight years of age or over, who are currently learning about mathematical equations but it is also a goal that it can be seen as a tool for learning by all ages. As some schools are now being equipped with iPads or Tablet PCs this application could be seen as an adequate tool to assist in the education of young students.

The goal of the MathMagic Basketball app is to provide the user with the experience that combines the subject of mathematics with the interactivity of the World Wide Web. This will form the basis for the study into the extent and limitations of dynamism and interaction in web-based applications. Once these objectives are well-defined the scope of the project can now be better understood and so too can its overall feasibility.

1.6 Proposed Methodologies

The structure of this projects research will be based on the Software Development Life Cycle (SDLC).
There will be a number of steps involved from the inception and project definition, through analysis and development to the testing and evaluation stages.

At the beginning a clear goal or objective is defined. Following on from this is the system analysis and design stage. This includes a study of the feasibility of the task ahead and the examination of all the available technologies suitable for the apps creation. The analysis will involve:

- The identification of which HTML5 APIs are suitable for implementation. Such as Drag and Drop, Web Storage and other APIs. These are the most problematic of all the APIs as there

are some known issues with both. Because of such issues it is for this reason that they will both most likely be included in this development. It is expected that they could give an indication of the current status of interactivity through HTML5 web applications.

- The implementation of workarounds or polyfills to compromise development issues and to what extent will their impact be on the rest of the program code.
- What is the best method for creating a mobile application? Is pure HTML5 a preferred method over the use of a javascript framework such as PhoneGap?

Various methods of research will be undertaken to discover solutions to these proposed questions and this research can be categorised as follows:

- Analysis of relevant research papers and published journals.
- The study of topical literature and other material made available through various accessible libraries.
- Reading online blogs and reviews about the web technologies involved.
- Seeking out and examining similar available applications.

The next stage in the SDLC is the implementation where the web app in both desktop and mobile form will come to live. As this phase nears to an end then so begins the testing phase. The purpose of this phase is to isolate any problems with the code and then rectify them. This employs various techniques for testing the product and in doing so identifies and eliminates any existing bugs that may exist. Once carried out correctly the application should have a noticeable improvement in performance and it is now ready for the next stage of evaluation. Evaluation is the final stage of the SDLC where the application is evaluated based on the overall objectives and targets of the project.

1.7 Expected Results

It is expected that this project's analysis of web and native applications will provide various advantages and disadvantages associated with both. However I am confident that the research will show that interactive and dynamic web development has progressed with the introduction of HTML5. It is believed that research will show that the arrival of HTML5 will

have a profound effect on all application development. Results should show that it is now at a stage that it rivals native app development. It is also expected that the technological analysis will indicate that HTML5 is the appropriate technology required for this project's application implementation.

Implementation of an interactive web application that has cross-browser support is a major concern. However, this should be proved achievable especially with the release of Internet Explorer 11. The latest edition of this browser is contained within Microsoft's Windows 8.1 [2] which has just recently been released. Internet Explorer has been one of the last major browsers to fully support HTML5 and other new technologies such as WebGL [3]. WebGL is a JavaScript API for implementing interactive 3D and 2D graphics through a browser interface. It is hoped that the project's research will reflect any issues and tackle any such concerns.

1.8 Preliminary Project Plan

1.8.1 Milestones

1.8.1.1 Task 1. Research

This initial phase of the project will begin with extensive research into the project's subject matter and all information pertaining to the research questions outlined above. The research stage is expected to take 7 weeks to complete and will conclude with a literary review.

1.8.1.2 Task 2. Feasibility Study

The second task of the project will involve the drafting of a Feasibility study. It will determine to what extent each task, and the project as a whole, is achievable and realistically feasible. It will also define the scope and the limits of what is proposed. This task is expected to be completed within 3 weeks and will overlap with the research task.

1.8.1.3 Task 3. System Analysis and Design

This will involve the study and the analysis of the web application's design. The creation of UML diagrams will also occur at this stage providing a better view of the web application's structure and its proposed design. Analysis and Design has a timeframe of 2 weeks.

1.8.1.4 Task 4. Project implementation

This task is concerned with the construction and development of the hosting website including the web-based application. It will involve the actual coding of all files in all of their appropriate languages. All of the relevant technologies will be implemented in this phase. How they interact and coordinate with each other will also become clearer. The duration of this task is expected to be 8 weeks.

1.8.1.5 Task 5. Project Testing

It is at this stage where all the code implementation from the last task will be rigorously tested for errors time and time again. The purposes of this is to isolate any problems within the application development and then to rectify them. This task is estimated to last 2 weeks.

1.8.1.6 Task 6. Evaluation

Unlike in the Project Testing phase, this task will involve the evaluation and assessment of the entire project. It will take into account all the tasks performed and assess how well they reached their conclusion. The purpose of this task is to analyse these results to ensure that the project's goals have been reached. This task will take 2 Weeks for completion.

1.8.1.7 Task 7. Maintenance and Monitoring

This task involves the carrying out of any maintenance that the web application requires in its initial period of existence. The application will be monitored for any issues or problems to ensure it operates at the desired level of performance. It will be analysed so that it meets such functional requirements as cross-browser compatibility, ease of accessibility, and general smooth running of all operations. The duration of this task will last 6 weeks.

1.8.1.8 Task 8. Presentation

The final stage of the project and will involve the presentation of all research findings in conjunction with a demonstration of the completed web application. Preparation for this will take a minimum of 2 weeks.

1.8.1.9 Timetable and Gantt chart

The timetable for each activity is shown in the table below:

Task	Week beginning	Duration in weeks	Activity
1	0	7	Research
2	6	3	Feasibility Study
3	9	2	System Analysis and Design
4	11	8	Project Implementation
5	18	2	Project Testing
6	20	2	Evaluation
7	20	6	Maintenance and Monitoring
8	24	2	Presentation
9	0	26	Project Thesis

Table 1 A timetable of the project's schedule.

Note: The duration of some tasks will overlap.

All tasks and activities are depicted in a Gantt chart in Fig. 1.

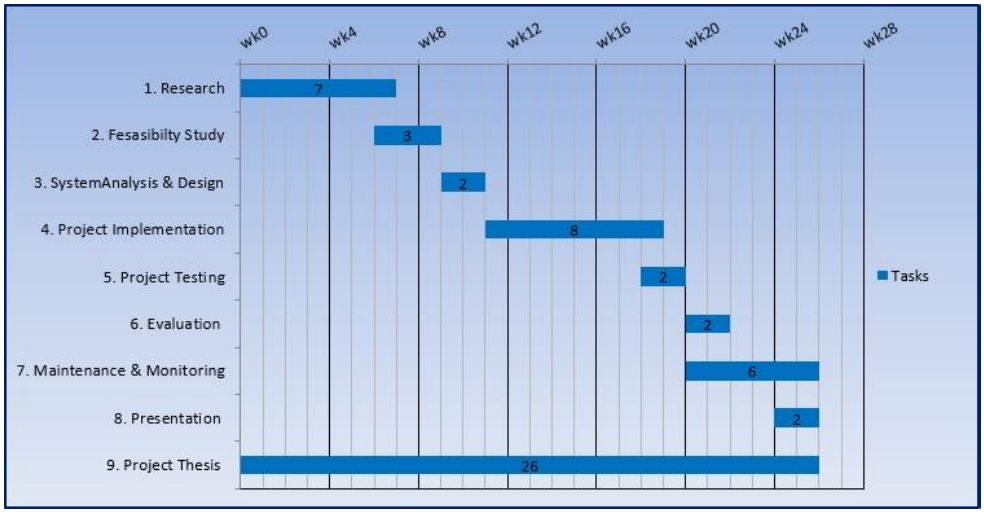

Fig. 1. A Gantt chart representation of the project schedule.

2 Literature Review

2.1 Web Dynamism

2.1.1 Programming through the Web

In the research paper Some Trends in Web Application Development [4] by Mehdi Jazayeri of 2007 many comparisons are made between conventional approaches to software engineering and the development of web applications. Even though this paper was published six years ago, and six years being a long time in the area of development, it is still very relevant today. A lot of the predictions and findings made by Mehdi Jazayeri regarding future development trends have been proven to come true. For example the author correctly foresees the technological advances that were to be made with protocol standards, web browsers, software engineering methods and application trends. These prediction have mostly come to bear as all of the main browsers have since cycled through many editions in their quest to support new web technologies such as HTML5, CSS3 and WebGL [5]. Web browsers have also adapted to incorporate new means of client-side storage such as HTML5's local storage [6]. Trends in application development have also seen a swing towards the mobile environment.

This paper highlights how users can collaborate and cooperate with each other more efficiently as web applications become more interactive and dynamic. A matter of interest is the relationship between the development of these web apps and software engineering. Web app development can be implemented, like with software development, with the aid of the Model-View-Controller (MVC) [7] structural pattern. The model in the MVC pattern represents functions that interact with the database or data store, the controller embodies the application logic and the view reflects the HTML web pages that output to the user. There is also a wide range of available web app development frameworks which are based on programming languages such as the Java Enterprise Edition (Java EE) [8]. Java EE is based on the Java programming language. These frameworks also employ the MVC design pattern and use the Object Relational Mapping (ORM) [9] technique to map objects to databases.

Managing and maintaining web applications differs to software development when it comes to application improvements such as updates and version upgrades. Patches, bug fixes and

new features can be applied to web apps directly and usually without the participation of the end user. This is in sharp contrast to desktop applications where significant changes come in the form of new version releases which occur periodically.

Web developers now have access to a variety of Application Programming Interfaces (APIs) which are provided by frameworks and web-based software companies such as Google, Twitter and Flicker. In addition to these, HTML5 brings a whole new range of APIs to Web application development.

Challenges exist for web developers in designing applications that can run on different client devices such as mobile phones, tablets and even Smart TVs. One such challenge is the development of apps to avail of user location and context-aware services which are provided by these devices.

This paper correctly forecasts some of the future trends in Web app development. Bearing in mind that it was published in 2007, it indicates that at this time there is work in progress with HTTP and HTML by the Web Hypertext Application Technology Working Group (WHATWG) [10] and the World Wide Web Consortium (W3C) [11]. This could almost certainly be referring to the advent of the new HTML standard, HTML5 which has involved the cooperation of the two organisations since 2006.

The example web applications given by Jazayeri do adequately support his arguments as of this period in time, However dynamic and interactive web apps have progressed to an entirely new level due to the arrival of new and improved technologies such as HTML5 and CSS3.

The author, Mehdi Jazayeri provided me with a great insight and confirmed my already established believes about the correlation between software and web development. This is the main reason why I chose to review it. The technologies and methodologies discussed were very beneficial to my current research. The document was well worded and I believe very well written. All of the technological explanations and examples were expressed clearly. I found them straightforward and easy to understand. The paper was also well-structured and laid out adequately. I found the author, Jazayeri, to be insightful and informative. The only drawback that I had with this paper was the fact that it was written such a long time ago. I would have liked to have seen an updated edition of this document. This, however has no reflection on the good work

accomplished here by the author or the overall quality of the paper. It was ideal as an introductory piece for my research as it covered the fundamentals of Web dynamism and interactivity. I used this paper as a benchmark for further research material.

2.1.2 Web Development Trends

There has been a trend in web application development which is leaning towards the implementation single-tier architectures. This is important as it shows that there is now an alternative approach taken to dynamic web development which employs the traditional three-tier system. This system embodies the Presentation layer, the Logic layer and the Data layer.

These alternative approaches are highlighted in a 2011 article in the IEEE magazine, IT Professional. This article titled Toward Unified Web Application Development [12] begins by classifying programming languages in to three patterns or paradigms; Imperative, Declarative and Hybrid. The Imperative style of programming deals with statements and instructions that are given to a computer. It defines how results are achieved i.e. the how-to-do. The Declarative style specifies the set of rules as to what outputs should result from the inputs i.e. the what-to-do. The Hybrid style, as expected, is a combination of the other two styles and does not feature in any further analysis in the article. Examples of Imperative languages include C, Java and JavaScript while examples of Declarative languages include CSS, HTML and SQL.

The article continues by giving a brief layer by layer explanation of the three-tier system. Information is also provided about the various languages and techniques that occur at each layer. The author goes on to report that there is a problem with this architecture. This is that each specific tier a level of specialised expertise is required by web developers. The article attempts to resolve this by implementing a variety of frameworks to accomplish a single-tier system. These framework tools are applied to a specific tier where they work best. Their classified style, imperative or declarative, is also a factor in their implementation.

According to this article the Presentation layer, usually associated with Declarative languages, can be expanded, with a Declarative-based framework such as XFormsDB [13] to incorporate server-side and data sore functionality.

The Logic layer can be expanded with the aid of a framework based on an imperative programming language. The author suggests the Google Web Toolkit (GWT) [14] which was introduced by Google in 2006. This toolkit enables developers to construct a client-side user-interface coded in Java (which is then compiled to JavaScript and HTML) that would normally be implemented at the Presentation layer in a Declarative language. The Logic layer can become a fully functional single-tier architecture with the additional functionality of a Java-based ORM library such as Hibernate [15] for database mapping.

This article proposes that the Data layer can also be expanded by implementing a framework such as the Sausalito XQuery application server. This can extend this tier‘s declarative query language to adapt client-side and server-side functionality.

When these three aforementioned frameworks were evaluated it was the imperative styled GWT that came out on top as the best implementation. It was proven to be the most reliable and best supported with libraries, documentation and access to an online community. All three approaches to a single-tier architecture had convincing and positive results. It was successfully demonstrated by the author that there is an alternative model to the traditional three-tier system.

This research paper finishes with a statement declaring that due to the recent popularity of cloud distributed systems, declarative based programing frameworks will feature more and more in the future as this trend continues. I was unsure about the authenticity of the article and the impartiality of one of the authors. This is because one of the frameworks selected for review, XFormsDB, was developed by the author Markku Laine. This however turned out not to be the case as it was ultimately outperformed by GWT. With this in mind this article gave me an insight into the use of declarative languages and frameworks. Its relevance to my own thesis is based on the belief that the declarative languages HTML5 and CSS3 should feature heavily as Web interactivity and dynamism is explored.

2.2 Modern Web Interactivity

2.2.1 HTML5 Limitations

A main characteristics of distributed web applications is the use of client-side scripting such as JavaScript and server-side scripting such as Perl or PHP. As referred to previously, for some time now there has been a trend in development to maximize the use of the client-side. This involves the browser being employed as an applications user-interface. This has been hampered by the fact that browsers have had limited access to resources on a host computer. The introduction of HTML5 provides a basis for tackling this issue as it provides its own APIs and it makes use of Web storage [6] within the browser.

There are a number of concerns surrounding the implementation of HTML5 especially when it is being compared to native apps in the mobile development environment. A number of these specification limitations are identified by Peter Wayner in a 2011 article for InfoWorld.com magazine titled 11 hard truths about HTML5 [16]. This article addresses issues such as those relating to performance, security, local storage limitations, synchronization capabilities and browser compatibility. Although it predicts that HTML5 will be a viable alternative to native applications, it also states that its true capabilities are questionable.

The article lists eleven areas of concern with the first being that of security. It states that by handling much of the programming code on the client-side, this code may be edited and altered by way of a debugging tool. Brief examples of this flaw are given ranging from a simple playful geolocation edit on Facebook to criminal abuse for the purpose of financial gain.

Another area pinpointed by Wayner is that of the local storage. This is a new feature provided by HTML5 were web apps can avail of local and session storage otherwise known as Web storage. This storage allows web apps to store data on the client-side. Positives of this feature are that it boosts performance and enables web apps to work offline. Although data may also be synchronised to a server, if it is not then this form of storage becomes vulnerable. Possible reasons given for this are that the data is not portable and can only be accessed on the client. This article continues to assert three further negatives associated with this localised data. It states that

vulnerabilities exist because there is a lack of security control provided by a centralized server and because that the data cannot be moved or backed up.

An issue is also identified with HTML5's Web Workers. These are JavaScript's that run independently of each other while separating the workload. While running in the background they have no adverse effect on the web applications performance. The article states that other than API messages passed in and out of these scripts, the workload is unmanaged and only dealt with by the browser. It asserts that at some stage these scripts may become vulnerable to malware attacks.

Other notable hard truths suggested by the author refer to:

- **Data synchronisation.** This can be problematic especially in relation to changes which have been made while offline.
- **Application upgrades.** These can occur universally and without exception. Issues may arise with software compatibility for some users.
- **Browser compatibility.** This can hinder web app development. For example the new audio and video tags may suffer with this browser compatibility while implementing some media formats.
- **Hardware compatibility.** Like with native app development, Web app developers have to be aware of hardware challenges.

The author concludes this article with a political analogy. He declares that like in any democratic process, for the successful development and implementation of HTML5 as a standard, it is necessary to allow browser developers to strive to outdo each other. As they do this in the hope of gaining a competitive advantage over each other, browser compatibility issues will continue to arise. However the long term positive effects of this competitiveness will eventually far outweigh these issues.

I highlighted only some of the eleven points made in this article. I did this because I felt some of these points overlapped (for example two separate points were made regarding the limitations of local storage). I also did not fully agree with all of the conclusions reached by the author. In fact one point referred to a possible problem surrounding cloud data storage, its long-

term availability and its ownership. I felt it would be unfair to apportion blame for this issue to HTML5. I thought some issues explained had more to do with browser compatibility rather than that of HTML5's implementation. I also believed that a number of the limitations expressed, although all were worth emphasising, did not warrant the advice suggested like that there are "reasons to steer away from HTML5" [16].

2.3 Mobile Development

2.3.1 Testing Web Apps in the Mobile Environment

As the growth in mobile web development continues so too does the growth in competition between the various development platforms. This competition is outlined in the 2012 paper by Ngu Phuc Huy titled Developing Apps for Mobile Phones [17]. This paper emphasizes the importance of HTML5 in the development of interactive and dynamic mobile web apps. It is a great source of information relating to mobile computing. It defines the status HTML5-based Web applications in relation to alternative mobile app types or paradigms, most notably native mobile applications.

This paper begins by stating that the growth in smartphones is directly due to the ability of these devices to run applications. The problem for developers is which application paradigm is the best to implement. Five paradigms are identified and evaluated; native apps, platform-based apps, mobile widgets, Web apps and HTML mobile apps. The aim of the paper is to aid developers in their choice of paradigm and to do this an app is constructed for the purpose of evaluation using two of the most popular paradigms, platform-based and HTML5.

NOTE: Throughout this paper the author divides what are commonly known as native apps into two paradigms, native apps and platform-based apps. The difference being that the latter are specific to a mobile platform (e.g. Android apps). This will continue while referring to this paper and for the purposes of its review. Other than this any reference made to native apps will encompass both these two similar paradigms i.e. the terms native and platform-based are used interchangeably.

2.3.1.1 The Developer Community Evaluation

The author puts application paradigms into two categories client-side apps and server-side apps. With client-side apps the main elements of the app are located on the mobile device. These client-side apps can be further classified as:

- **Native apps**. These are coded in compilation programming languages such as C or C++ and usually have access to a devices APIs and hardware.

- **Platform-based apps**. These are apps specific to an operating system platform such as Google's Android, Apple's IOS or Microsoft's Windows Phone. They are written in a language that is specific to a platform such as Java for Android or Objective C for IOS. They then access APIs and hardware by utilizing frameworks and libraries associated with these platforms.

- **Mobile widgets**. These are lightweight apps that utilize Web content using Web technologies.

The author continuous to define server-side applications and these are classified as:

- **Mobile Web apps**. These apps are deployed by means of the three-tier architecture where the mobile device is seen as the thin layer.

- **HTML5 mobile apps**. These apps are described in the paper as having the most modern approach to mobile development. The author explains that HTML5 comes with advanced technological features such as Geolocation [18], Scalable Vector Graphics (SVG) [19], Audio and Video tags [20], Canvas [21], Context-aware services [22], Web storage [6] and Web workers [23].

NOTE: Throughout this paper HTML5 mobile apps (although technically are mobile web apps) are categorised differently than mobile web apps (which are HTML4-based apps) as they provide newer improved features.

For the evaluation of the five paradigms. Questions were asked regarding a number of characteristics that related to both the development environment and an applications overall functionality. These characteristics included:

For the development environment:

- **Programming Language.** How challenging was the programming language?
- **Community support.** How much help was available from the developing community?
- **Code editor.** The performance of the Integrated Development Environment (IDE).
- **Testing and Emulator.** How an app performed when tested on an emulator?
- **Debugger.** How capable was a debugging tool?
- **Deployment.** The ease of operation when deploying and updating an app.
- **Distribution.** How accessible is an app to users within the marketplace.

For the functionality:

- **Software development kit (SDK).** Ease of installation and use of the SDK.
- **Hardware interface.** Ease of access to device hardware.
- **Built-in app interface.** The adaptation of built-in apps, such as a phone book, on devices.
- **Web service interaction.** Evaluation of web service implementation or access such as Facebook.
- **User-interface builder.** How efficient a user-interface builder is for development.

These characteristics were evaluated from feedback gathered from a number of online developer communities and also the analysis of a selection of scientific articles. Each paradigm

was graded as per category or criteria from 1 to 5 with 1 being a low grade and 5 being a high grade. The results can be seen in below in Table 1:

Criteria	Sub-criteria	Native app	Platform-based app	Web app	Widget	HTML5
Development environment	Programming language	2	4.33	3	5	4
	Support from community	3	4.67	3	3	5
	Code editor	4	5	3	3	4
	Debugger	3	4.33	3	4	3
	Testing	2	4	1.67	3	4.33
	Deployment	3	3	3	3	3
	Distribution	3	3.33	5	4.5	5
Functionality	SDK	4	4.67	4	4	4.5
	Hardware interface	5	5	1	3	4
	Built-in app interface	1	2	1	1	5
	Web service interaction	2	5	4	3	4
	User interface builder	2	5	2	2	3
	Average points	*2.83*	*4.2*	*2.81*	*3.21*	*4.07*

Table 2 Evaluation of Mobile Paradigms (table sourced from Developing Apps for Mobile Phones [14]).

2.3.1.2 The Application Testing

This section of the study is performed to substantiate the findings of the previous evaluation. An object recognition application was developed as a platform-based app, then as a HTML5 app and then it was also created with the aid of a mobile development framework called PhoneGap [24]. The PhoneGap app was written in HTML5, CSS3 and JavaScript. These apps were implemented and tested on Android and IOS mobile operating systems. Their function was to capture an image by means of a mobile device's camera and then send it to a server for recognition. The author states that a major advantage gained from the utilization of the PhoneGap framework was that it provided efficient cross-platform support. This addresses one of the main concerns surrounding the deployment of HTML5 web apps, platform-independence.

The results of the application testing coincided with the findings of the previous community-based evaluation. They show that platform-based apps still perform the best and this is largely due to their compatibility with on-board device hardware such as a camera and storage. The HTML5 app built with the aid of the PhoneGap framework scored very high in the tests and this was due to having such added benefits as access to a device's native APIs. It had sufficient interaction with a phone's built-in apps. The HTML5 app developed without the PhoneGap framework performed well also but the test seemed to highlight some of its limitations. These limitation were associated its own APIs and with cross-platform support.

In concluding this research the author finds that an apps fundamental requirements are important factors when considering the right paradigm. Such elements that influence the design process include hardware compatibility needs with which platform-based apps perform the best, access to web services with which HTML5 apps lead the field and access to native APIs with which is best supported by a framework such as PhoneGap. As a recommendation, the author suggests using such a framework. He alludes to the benefits of its write once run anywhere capabilities although he asserts that there may be some adverse effect on performance due to its overheads. It is also worth noting that HTML5 is incomplete as of this papers publication and the author reiterates this.

The reason I reviewed Developing Apps for Mobile Phones [17] was because I believed that the results of the evaluation performed were very informative and very relevant to my own research. I valued the integrity of this research because it is based on a survey of developers.

What is of particular interest to me about this evaluation is how close HTML5 apps rate when compared to platform-based apps and how high they are graded for all of the other applied criteria. In my ongoing research I have noticed many references to areas where HTML5 is lagging behind other application formats. One is in the area of distribution as HTML5 lacked a dedicated distribution outlet like an app store. Amazingly in the evaluation HTML5 apps get a maximum grade of 5 for distribution. Another area of development that I have seen called into question is that of Hardware interface or compatibility. Surprisingly here also HTML5 apps receive a high grade of 4 for this. I did not really gain any further knowledge from the application testing other than that of the benefits obtained when using a development framework such as PhoneGap. The results of these tests were largely what I would have expected.

2.3.2 Interactive Web Apps. An Alternative to Native Apps

The interactive capabilities of languages such as HTML5 and JavaScript are further explored in the mobile environment in Mobile Web Apps - The Non-programmer's Alternative to Native Applications [25], a 2012 paper by David Sin et al. This research compares HTML5 Web apps to native applications. It is important to note that within this research, and as alluded previously, the term native apps encompasses all platform-specific applications including the aforementioned platform-based apps. The authors declare that this study will show that Web apps can imitate and perform as equally efficient as native apps. Comparisons between these two

popular paradigms are derived from the development and evaluation of a HTML5 mapping web application.

An Html5 web app was selected for its ease of development and it was tested on the iPhone's IOS. Map-based apps are ideal for showcasing the many new features of HTML5 as they usually implement a number of its featured APIs. Two techniques were applied for the purpose of testing. One technique was to test an app with the aid of geolocation and the other was to test an app without it. The app without geolocation would be tested for its offline capabilities which utilises the client-side storage. The author suggests that the use of client-side storage saves on the use of resources such as bandwidth. As it is continuously being stated that HTML5 is yet to reach completion, it is for this reason that the geolocation API is constantly under discussion about issues relating to its functionality.

It is asserted in this research that one major advantage that HTML5 apps has over native apps is that they are platform-independent and therefore can easily be accessible on any mobile device. This, however, is only true if that device supports a HTML5 compatible browser. As in the previous reviews the author here talks enthusiastically about the advantages gained when implementing a framework during the development process. Frameworks such as JQuery mobile [26] and Sencha Touch [27] can be a great aid to developers. They provide valuable libraries especially for use in the construction of graphical user interface (GUI) objects.

In the evaluation the app was tested by a number of users. The results were based on feedback received which was very positive. Any corrections to the app that were needed were done so and these were mainly in relation to the GUI.

This paper concludes with the author predicting that web applications will play a major role in the future of app development and depending on how they are distributed they may even become more preferable than native applications.

In the Non-programmer's Alternative to Native Applications [25], I found the comparison between native apps and HTML5 app very interesting. The author points out that a benefit to Web apps is the fact that because they are hosted on a server they do not have to incur all of the restraints normally associated with an app store. It is also suggested that the type of application paradigm depends on its operational requirements i.e. native apps for hardware compatibility and web apps for server-side services. This review was short and less detailed than that of the other papers as it basically corroborated most of the information and conclusions that were reached

previously. I was disappointed with the lack of information supplied about the evaluation and I found the testing process a little confusing.

3 System Analysis

3.1 The Project Overview

As part of this projects exploration and investigation into web application interactivity it will feature the creation of simple mathematics web-based game. It will showcase the technologies involved is creating such an implementation highlighting their advantages and disadvantages. It can also be used as a new and innovative tool in the teaching of mathematic equations.

The project's implementation will involve the development of a game application called MathMagic Basketball. MathMagic is aimed at children who are eight years of age or over who are currently learning about mathematic equations and who also enjoy the fun of games. The goal of MathMagic Basketball is to combine the subject of mathematics with the interactivity of modern web applications through a user-friendly browser interface. The game will also be developed for both mobile and desktop environments. This game will test users with a number of simple random mathematical equations. As the user progresses through the game their skill, knowledge and understanding will also progress. This progression is achieved by ascending up the levels of difficulty of which there are three. As the user progresses so too does the degree of difficulty.

This game will aim to introduce young children to computer interaction giving them a better understanding to the technological devices that they see all around them in today's world. The hope is that they will learn in their early stages of growth that information technology can be an aid to their advancement through life. In the case of MathMagic Basketball, they can see how a game application can be used for fun and also as a tool for learning. MathMagic Basketball is not just intended for children. Its simplistic design can able users of all ages to brush up on their maths skills.

On a personal note, I have acquired some accumulated knowledge with these associated technologies involved in this project. However as with the case of some, such as Javascript, this was very limited. It is hoped by incorporating this technology into the project I can expand on this knowledge.

3.2 The Web Application

The project will involve the creation of a web-based system that will test users on the fundamental mathematical equations in a game format. Users can choose a level that they are comfortable with and when they feel they are ready they can then select a more advanced level. When the game begins a countdown clock appears and they have 60 seconds to complete a set of equations that will randomly appear on the screen. To answer these equations the user must drag their answer, which is in the form of a basketball to a basketball net. If correct they will receive a score of 10 points. When the time is up and the game is over they are prompted to enter a name to be entered into a scoreboard list.

3.3 Goals and Benefits

The goals of the project are:

- To showcase new developments in interactive and dynamic technologies such as HTML5 and CSS3.
- To Implement and test such APIs which have been designed to make the development process simpler while still being sufficient enough to produce a more efficient application.

- To develop an application that will provide users with a fun way to improve their knowledge of mathematics.

- To provide a sample implementation that is in the form of an eye-catching, well-designed, user-friendly web application which incorporates these new and improved development technologies.

- To store and record users accumulated data in the form of the highest scores achieved locally within a browser web storage rather than on a server-side database.

- To provide greater knowledge and valuable experience to the applications developer in relation to the technologies used in the applications implementation.

3.4 The Scope

This web app will provide the user with a new way to experience mathematics. They will take part in an interactive learning activity that takes the form of a user-friendly game. It will take into account a user's current knowledge of mathematics offering various degrees or levels of difficulty that most users will find suitable. MathMagic Basketball will not cover any areas of mathematics that would be deemed to be too advanced or complex. It will only include positive whole numbers and there will be no use of decimal numbers, negative numbers or fractions. However this may change depending on the wishes of users and such numbers may be included in later editions of the software. When the program begins a simple graphical interface design greet the user and following on from the prompt to choose a level the game will begin. During the game the user will interact by dragging their answer in the form of a basketball into the net until the countdown clock reaches zero. Upon completion the user will be further prompted to enter a user name to see their position on a scoreboard. On the desktop version this will all appear in one simple user-interface. On the mobile version there will be a series of screens that automatically appear to facilitate this interaction.

The scope of the in-game user interaction will be comprised of:

- Select a level. Easy, Normal or Hard.
- Drag a basketball representing a chosen solution to the basketball net.
- Enter a username to see position on scoreboard.

3.5 Walkthrough Scenarios

User:

To begin MathMagic Basketball the user will select a set of equations to solve based on a degree of difficulty. This is accomplish in the same way in both the desktop application and in its mobile equivalent. The choices are Easy, Normal or Hard and this selection is done by clicking on one of three buttons indicating an option. Once selected a continuous stream of equations will appear one by one center screen and the user will be prompted for a response.

The user will have one minute to answer each set of equations. When the first equation is displayed the user will then be required to enter an answer. If the answer is correct then a score

of 10 points will be awarded. If the answer provided is incorrect then the game will stall until the correct answer is given. The timer will still decrement while the game is stalled. The aim of the game is to accumulate as many score points as possible as these will be entered into a table of scores when the time is up. When the game is complete in the desktop version the scores are recorded and displayed in a table as the game is automatically reset. When the game is over in the mobile edition the screen automatically switches to a screen which will display the newly updated scoreboard. This screen will provide an option in the form of a button to return to the game. The game will reset before upon returning to it. This switching will not be achieved by means of linking to and from multiple HTML files but by the implementation of Javascript and the Document Object Model to amend one single HTML index file.

System:

The system will be programmed to respond to the user's entries, selections and interactions. The speed of these responses will be a very important factor in evaluating the performance of the game. This will ultimately depend on the technologies chosen for the apps implementation. This responsiveness should be a major factor in the projects findings as it almost certainly could define the limits of the technologies involved. Responsiveness may also be of particular importance when developing the app for the mobile environment. The system will basically interact through, and be represented by a web browser. Regardless of which of the main available browsers that are chosen for this purpose, the actions of the system or browser will be the same in the walkthrough scenario.

3.6 Preliminary Software Requirements Analysis

3.6.1 Functional Requirements

User:

Select a difficulty level:

The user should be able to initiate the game by choosing a degree of difficulty. When selected a set of equations should be displayed on screen that represent the chosen level. To achieve this the user should be giving the option of three buttons with each one representing a level categorised as easy, normal or hard.

Enter an answer:

Once prompted with an equation to solve, the user should be required to enter an answer from a choice of four possibilities that should appear on screen in the form of four basketballs. These basketballs should appear marked with a randomly generated number of which one of these is the correct answer. The user's choice of basketball should have the functionality to enable it to be dragged and dropped into a basketball net that should also appear on screen. The net should have the ability to capture this event and trigger a response.

Equations:

The equations should contain numbers that will be limited within various range to accommodate each level. For example with regards to division the second number in the equations, the denominator should not be greater than the first number, the numerator. The second number should not be too large and therefore should be within a well-defined range. The result of a division equation should also be and have no remainder. Similar appropriate factors should also be considered when formulating equations with the addition, subtraction and multiplication operators. The equations for the 'hard' level should be the most difficult to implement so this may need to be researched further.

Countdown timer:

The countdown timer should be able to function in descending order from 60 to 0 seconds with 0 triggering the events for the game over and enter username for scoreboard.

Reset settings:

The game should reset to its start state or welcome screen when it is over and the username and current score have been stored. The desktop application should provide a reset button but this is not required in the mobile app.

Scoreboard:

The scoreboard should not be reset at the end of the game however there should be a button to provide the user with the option to clear the scoreboard at any time. It is also worth noting that since the scoreboard only appears on screen when the game is over then it should be a requirement to provide an option in the form of a link or a button to access it from elsewhere within the application. Possibly located on the welcome screen.

System:

Progress:

This application should provide a score for each set of equations correctly solved.

Storage:

At the end of each game the system should provide the functionality to communicate with a browsers local storage to so as to store and record a username and password.

3.6.2 Non-Functional Requirements

User interface/Graphics:

The application should be displayed with a friendly, fun and colourful interface. The purpose of which is to provide the user with a less formal approach to mathematics.

Random numbers:

Each number that appears in the displayed equations should be chosen at random by the application.

Countdown timer:

Each difficulty level should have a set of equations that are timed for a period of 60 seconds.

Processing:

The web application should process the answer provided (dragged and dropped) by the user by checking if it is correct or incorrect.

Incorrect answer:

If the user enters the wrong answer then score points should not be awarded and the game will not progress until the correct answer is provided. The game should effectively stall and should not proceed to the next equation. The countdown timer should continue and not be interrupted or paused. This should be seen as a form of punishment for entering the wrong answer.

Scoring:

Each correct answer should be registered in the form of a score, i.e. ten points for each correct entry or drop. This should be displayed on the user-interface.

3.7 Technical Requirements and Feasibility

Development Language:

This web application will be developed and implemented using HTML5, CSS3, Javascript, JSON and the Document Object Model. These technologies are readily available to work with as is with no software development kit being required for their implementation.

Frameworks:

If some frameworks such as jQuery are required in this development then the latest versions of each should be sourced and appropriately linked to by their path locations within this applications HTML files. This is also true of the Modernizer framework which provides support for previous versions of browsers among other functionalities.

Browser compatibility:

As this application utilizes web technologies it should be compatible with all leading browsers and it should be expected to run in previous versions.

Hardware:

As the graphical interface and functionality of this software is web-based. It should only need network access and a compatible browser installed to be operated on a device. This requirement applies to the app when it is accessed in both the desktop and the mobile environments. At a later stage in the project an app should be developed for the mobile medium using a Javascript framework such as PhoneGap. This should take the form of a web app wrapped as a native application and the purpose of this development is to provide functional comparisons with the HTML5 implementation. The PhoneGap- developed app should require installation on an Android mobile device for usage. This would then require sufficient amounts of RAM and CPU processing capabilities but this is expected to be minimal.

The storage space on the device should store usernames and low-value integer scores locally on the client side. This is achieved by means of HTML5s Web storage or Local storage. This

storage is similar to web browser cookies but with a larger capacity. Cookies traditionally had a storage space of about 4 kilobytes while local storage now has the ability to store up to 5, 10 or 15 megabytes of data depending on the model of browser.

A large amount of storage should not be required for this app as the scoreboard list should only list the last ten scores. If this storage size becomes problematic then stored usernames and scores outside the list of the top ten should be removed with Local storage's remove method (localStorage.removeItem()).

Testing:

This application will be rigorously tested and debugged prior to its final release. These techniques are readily available.

3.8 Operational Feasibility

Although this application and how it should function are still at the inception stage of development. It should be seen as a new and different approach to easy-learning mathematics by users or students in education.

A student's progression through math education should be able to accelerate with the aid of MathMagic Basketball.

Although accessible to all, this application should be able to be implemented by educators or teachers on a student by student basis. This independent usage should be able to make the teachers monitoring of a student's abilities and progress somewhat easier.

3.9 Software Development Process

The model for the software development life cycle that will be adhered to in this project is the Waterfall model. However it will be a modified version of this model. As stated previously in section *3.7 Technical Requirements and Feasibility* under *Hardware,* the system requirements are understood and this stage need not be defined. In the Waterfall model a stage must be completed for the next stage to begin. This will not be the case in this modified version, shown in Fig. 2, as some stages near completion other phases will have begun and so therefore stage will slightly overlap. This is better illustrated in section *1.8. Preliminary Project Plan* of this documentation.

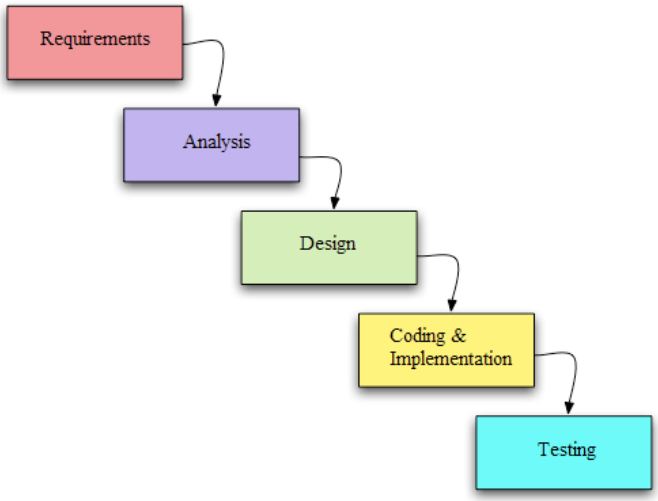

Fig. 2 A diagram of the Waterfall model.

3.10 Risk Analysis

Complex equations: There may be some difficulty in the comprehension of advanced division equations by users. The division of some numbers unevenly by another number which may result in a remainder will also cause problems not only for the user but for the apps displayed output.

Solution: This application will be programmed to only use composite numbers for the division equations. Complex division equations by large numbers should also be avoided.

Numerals: The use of Fractions, Negative numbers and Decimal numbers may also be deemed too complex for users especially young students at an early stage of learning.

Solution: The software will only make use of positive whole numbers but the option to include more complex equations as another level of difficulty may be taken at a later date.

API Compatibility: The MathMagic Basketball game will utilise some of the new available HTML5 APIs and CSS3 technologies.

Solution: All leading browsers which are intending to run the application must be initially checked and tested for HTML5 and CSS3 compatibility.

Personal Skill Base: As some of these technologies are new revision of current standards my experience with these may be somewhat limited. As it stands I possess only a very basic understanding with limited knowledge of such technologies as Javascript and jQuery.

Solution: Further research will need to be undertaken into all the development technologies, frameworks and techniques, and programming languages that are required to successfully bring this project to fruition.

3.11 Visibility Plan

Meetings with the project supervisor will be arranged whenever necessary to discuss the projects development. Such topics that may be discussed at these meetings may include:

- What is the current status of the project?
- In what direction is the project going in?
- Is it still progressing according to the project proposal and plan or have any unforeseen issues arisen with the proposed technologies?
- As a result of these issues have any changes being forced upon the project or has changes been made simply upon reflection?
- Is the project development still on schedule i.e. is the proposed timeline being adhered to?

Communication with the supervisor will also be done by email and pre-arranged meetings. The email channel of communication will remain open although it will mainly be used for the purpose of arranging the time and location of meetings.

4 System Design

4.1 Design Structure

This section aims to describe the design of the project's web app implementation and the overall design phase of the project. It describes how the system will be structured in preparation for the implementation stage. Its contents will be made up of a series of UML diagrams that adequately describe the proposed structure and how a user will interact with the system. The design phase will encompass the layout of the web application's game interface, the functional design that incorporates the user-interaction and the proposed formulas or algorithms for the mathematical equations.

4.2 Interactive Interface Design

This project has two main aims, the first being to implement an application that exhibits human computer interaction through web-based technologies, in doing so comparing these technologies against any alternatives. The second being to ensure that this application is fully functional and user-friendly. With these goals in mind a lot of importance should be placed on the web apps user-interface. It should be straight-forward and simplistic in design. It also should be designed to accept user interactions and handle this event responsively.

The interface should:

- Test users interactively by posing mathematical equations and accept their input by implementing HTML5's drag and drop API.
- Allow users the option of choosing a degree of difficulty in the form of three buttons. One for each level representing easy, normal and hard.
- Test users speed of response and performance by invoking an onscreen countdown timer.
- Provide users with information about their current score and ranking within a scoreboard.

- Allow users the options to reset the game and clear the scoreboard table by mean of two buttons.

- Allow users to share the application with friends on social media websites such as Facebook, Twitter and Google plus.

The design of the website will be structured with the HTML5 markup language, its design will incorporate the CSS3 styling technology and the functionality will be based on Javascript scripting technology. How these technologies interact will not only play a pivotal role in the successful outcome of this apps development but it will ultimately form the basis for this projects findings.

The interface design for the desktop application:

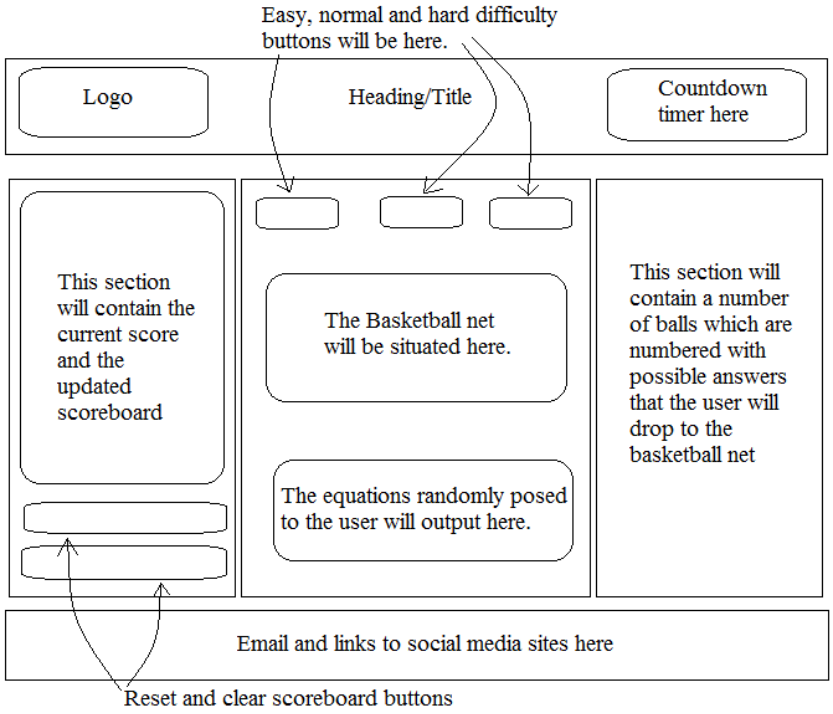

Fig. 3 The proposed interface for the desktop application.

The layout or structure of the mobile application will be different than that of its desktop counterpart. This is because every bit of space of such small-screen devices is at a premium and very valuable. Clutter is also worth avoiding and this is why the use of multiple screens is the best way of implementation. MathMagic Basketball will have three functional screens. A welcome screen with limited functionality except for the three level of difficulty buttons and various links.

A game-play screen which will implement the game and only functions that are game related and a scoreboard screen which will display a top ten list of the highest recently achieved scores.

The interface design for the mobile application:

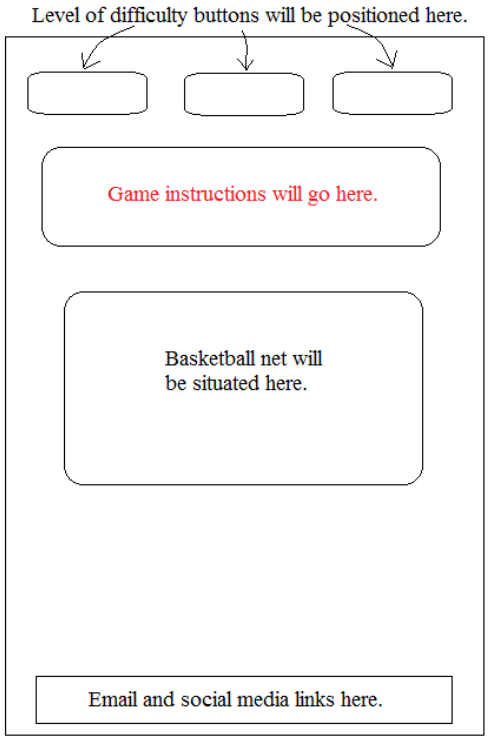

Fig. 4 The welcome screen layout of the mobile app.

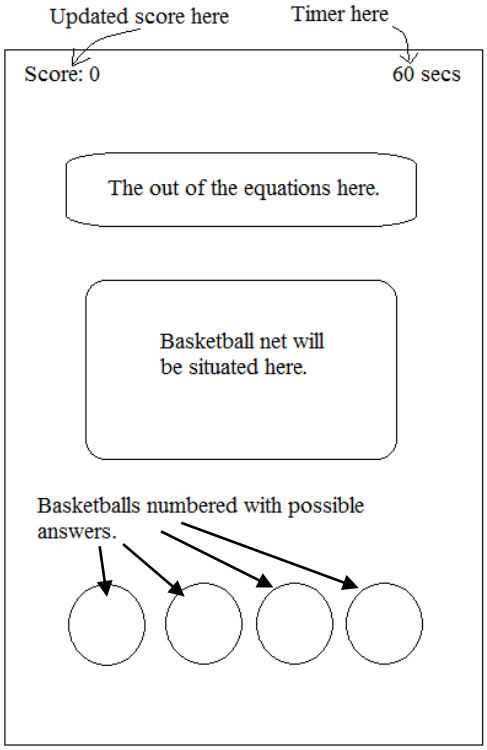

Fig. 5The game-play screen layout of the mobile app.

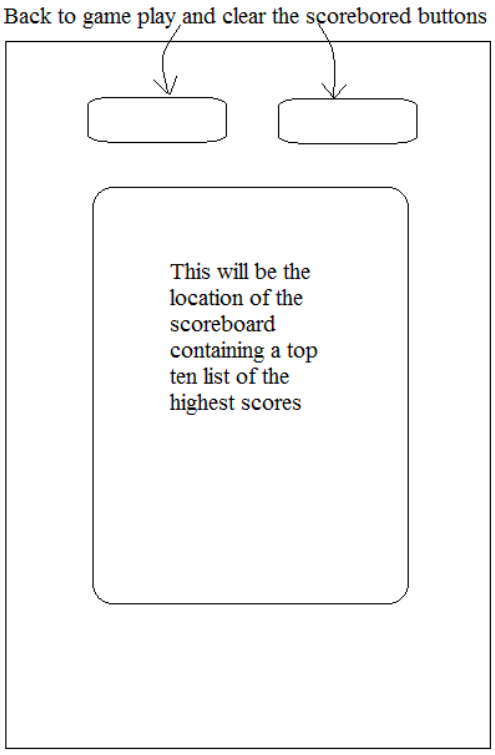

Fig. 6 The layout of the scoreboard screen on the mobile app.

4.3 The Functional Design

4.3.1 The Use Case Diagram

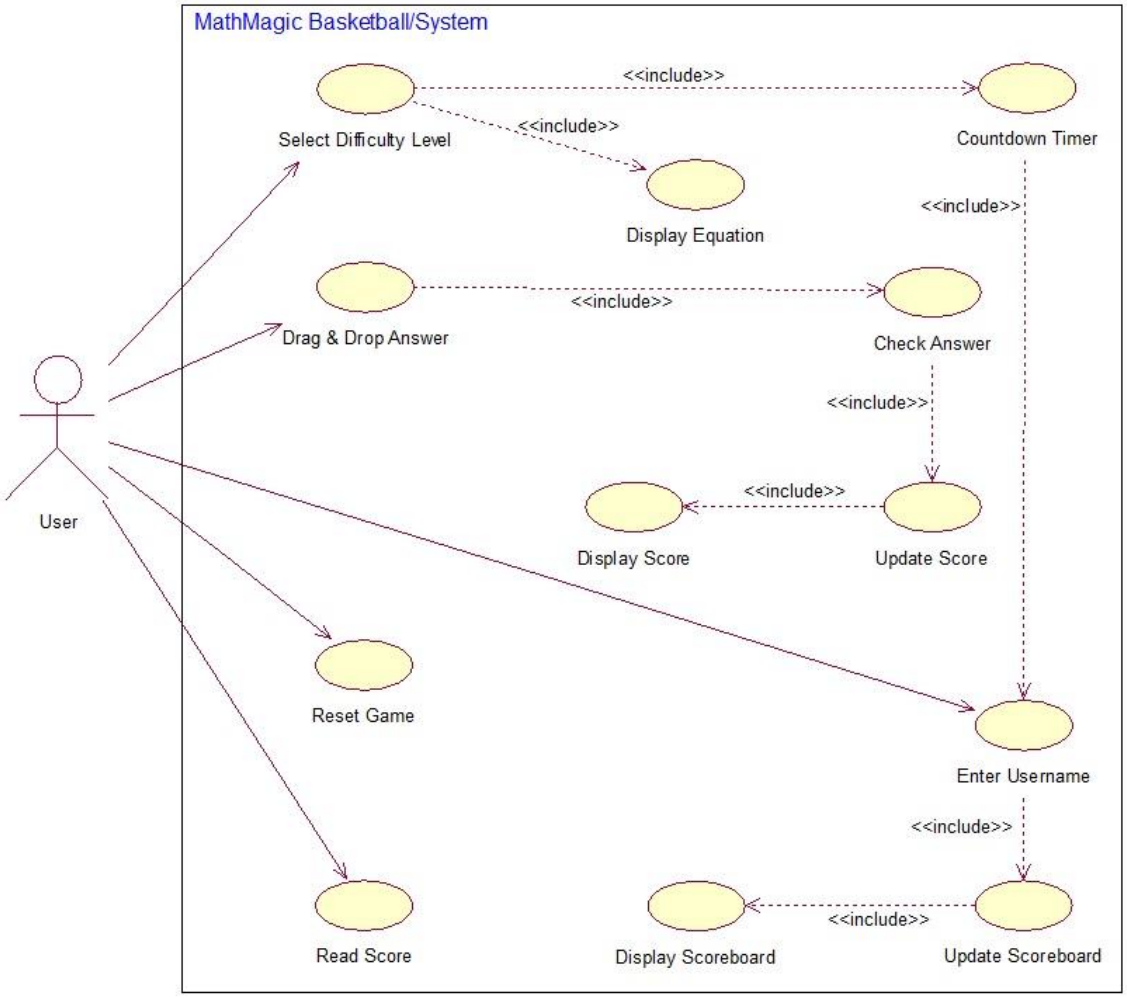

Fig. 7 The game interaction depicted in a Use Case diagram.

Use Case Specifications

The following are UML screenshots image of the Use Case Specification for each function of the application.

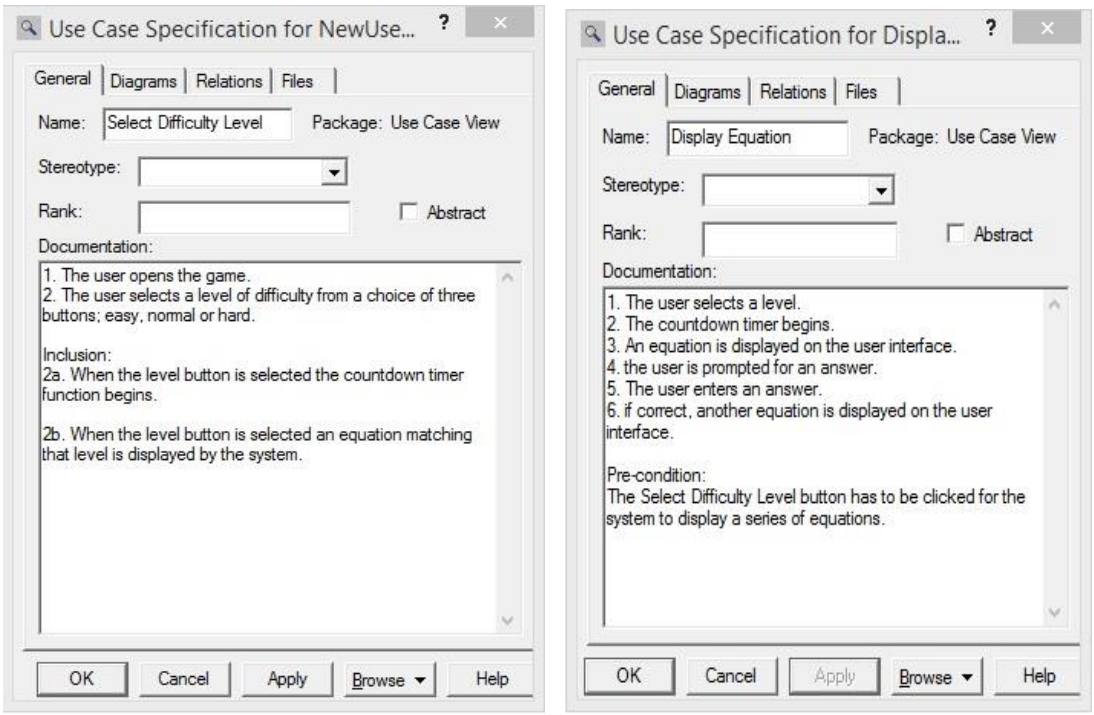

Fig. 8 The Select Difficulty Level and Display Equation Use case specification.

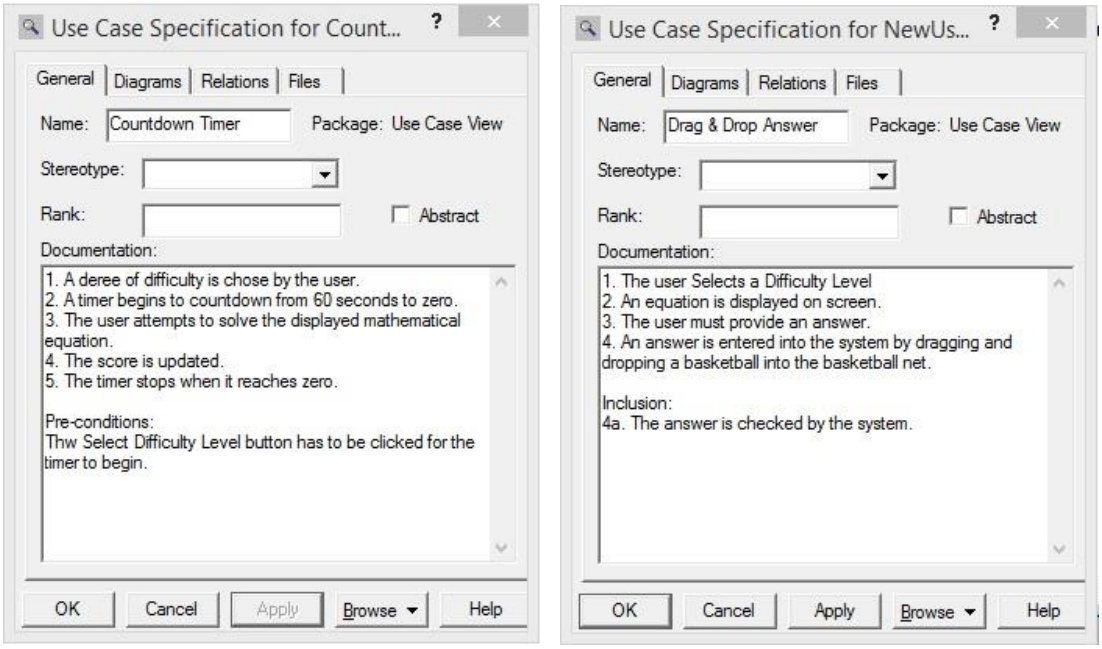

Fig. 9 The Countdown Timer and Drag and Drop Answer Use case specification.

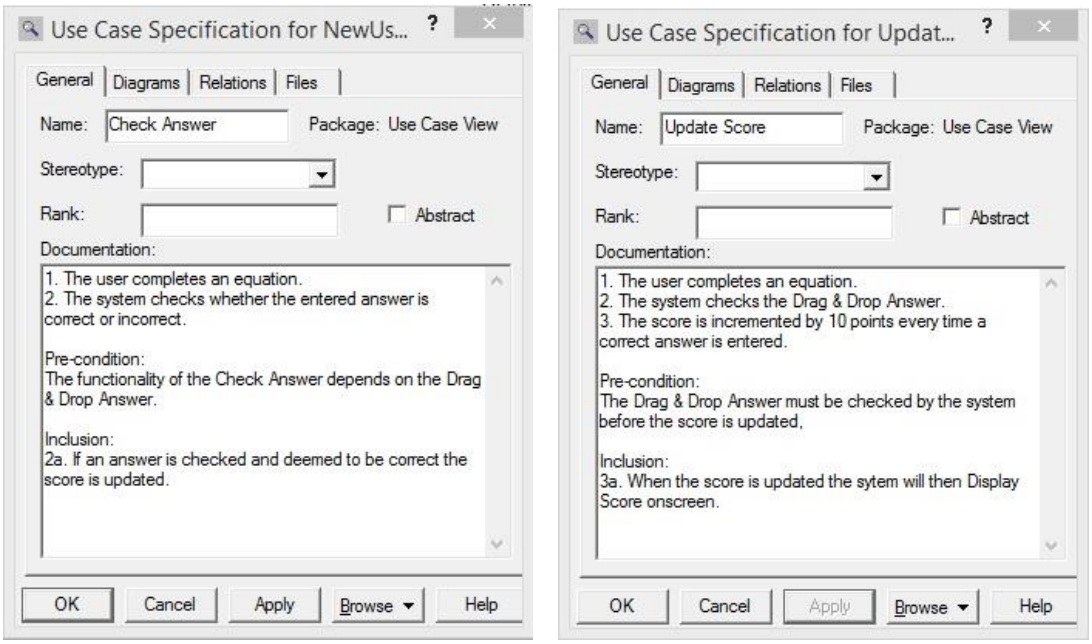

Fig. 10 The Check Answer and Update Source Use case specification.

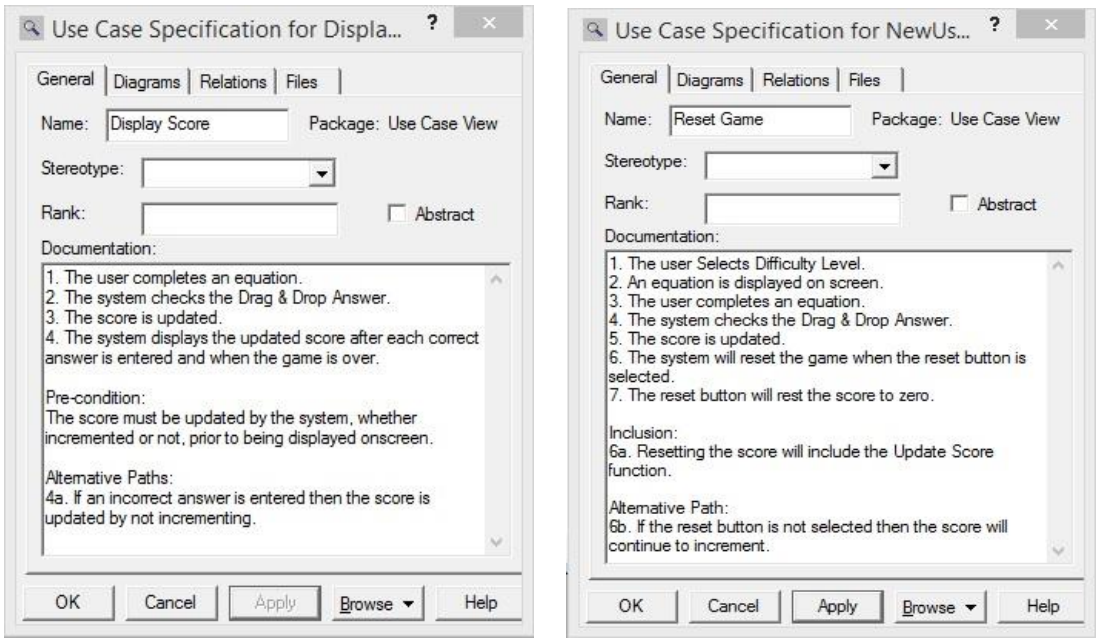

Fig. 11 The Display Score and Reset Game Use case specification.

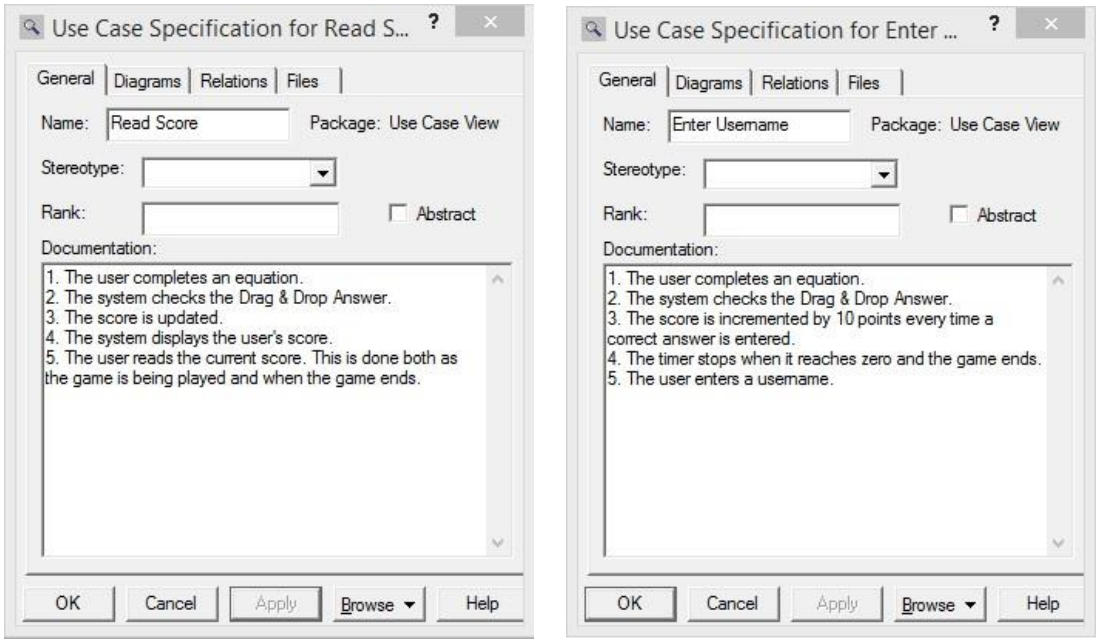

Fig. 12 The Read Score and Enter Username Use case specification.

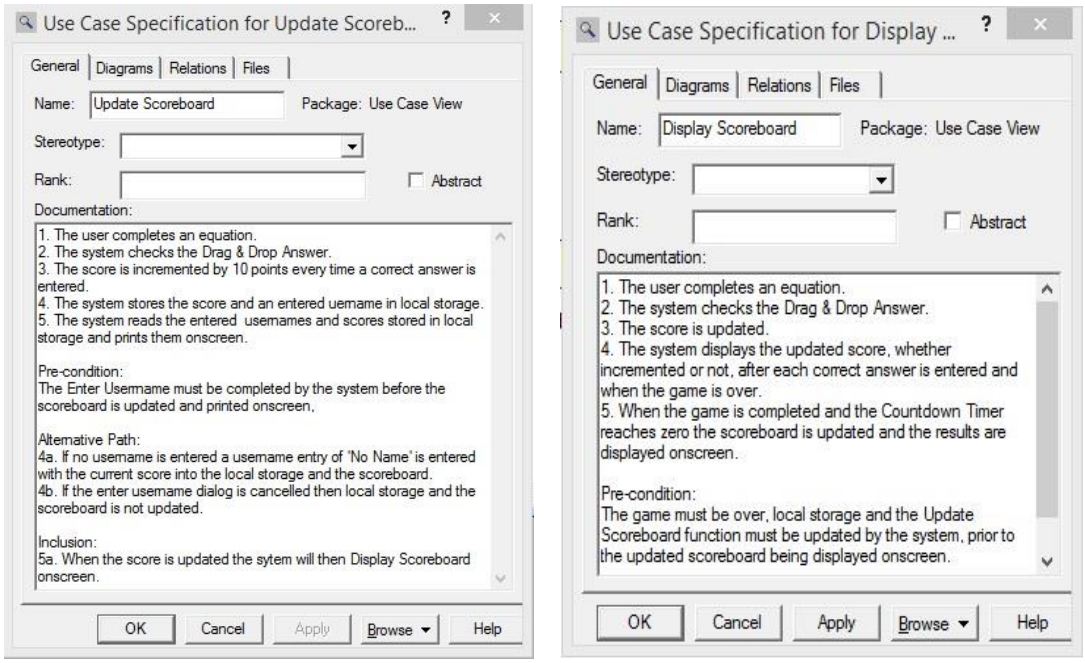

Fig. 13 The Update Scoreboard and Display Scoreboard Use case specification.

4.3.2 Class Diagrams

What follows below is a Class diagram depicting the overall functionality of the Javascript programming classes and their associated attributes and functions for the game application.

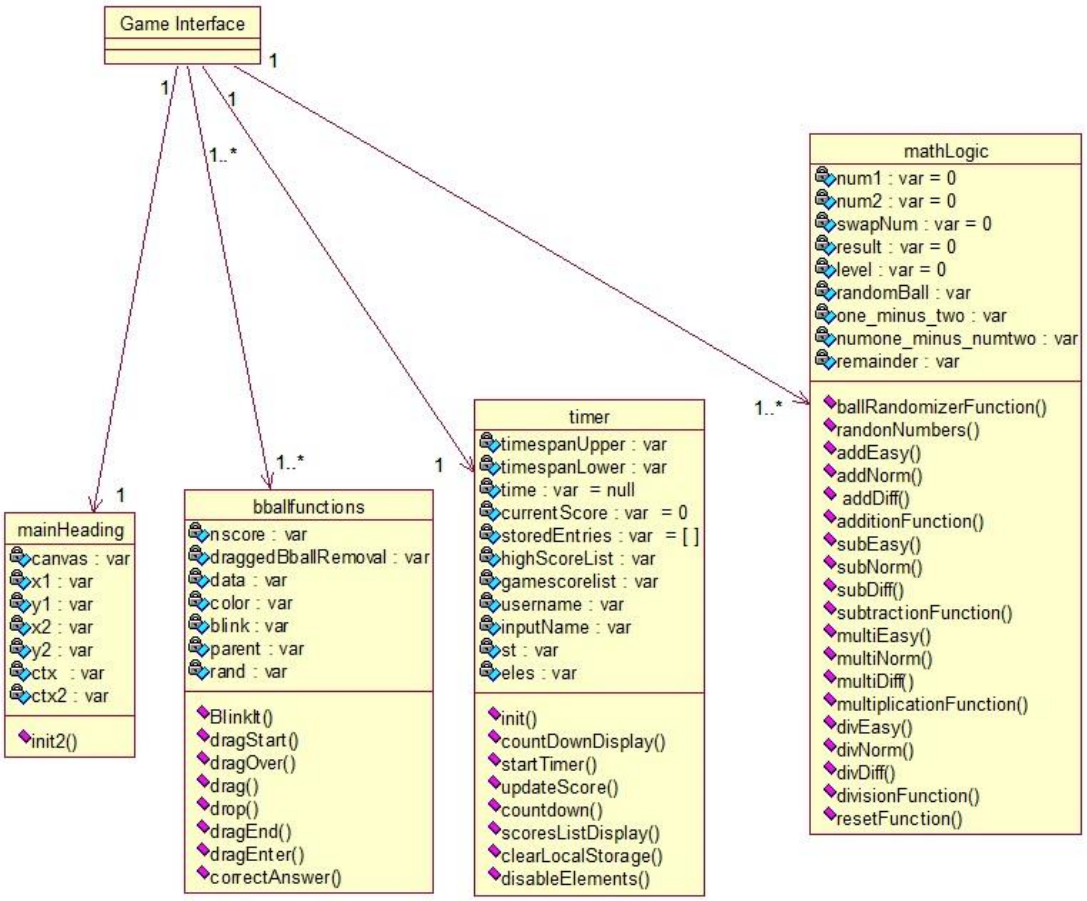

Fig. 14 The overall game functionality shown in a Class diagram.

Fig. 14 illustrates a Class diagram of the functionality of the mathLogic.js class or script. Although it may initially appear complex it is however an adequate representation of how the mathematical equations will be incorporated into the application.

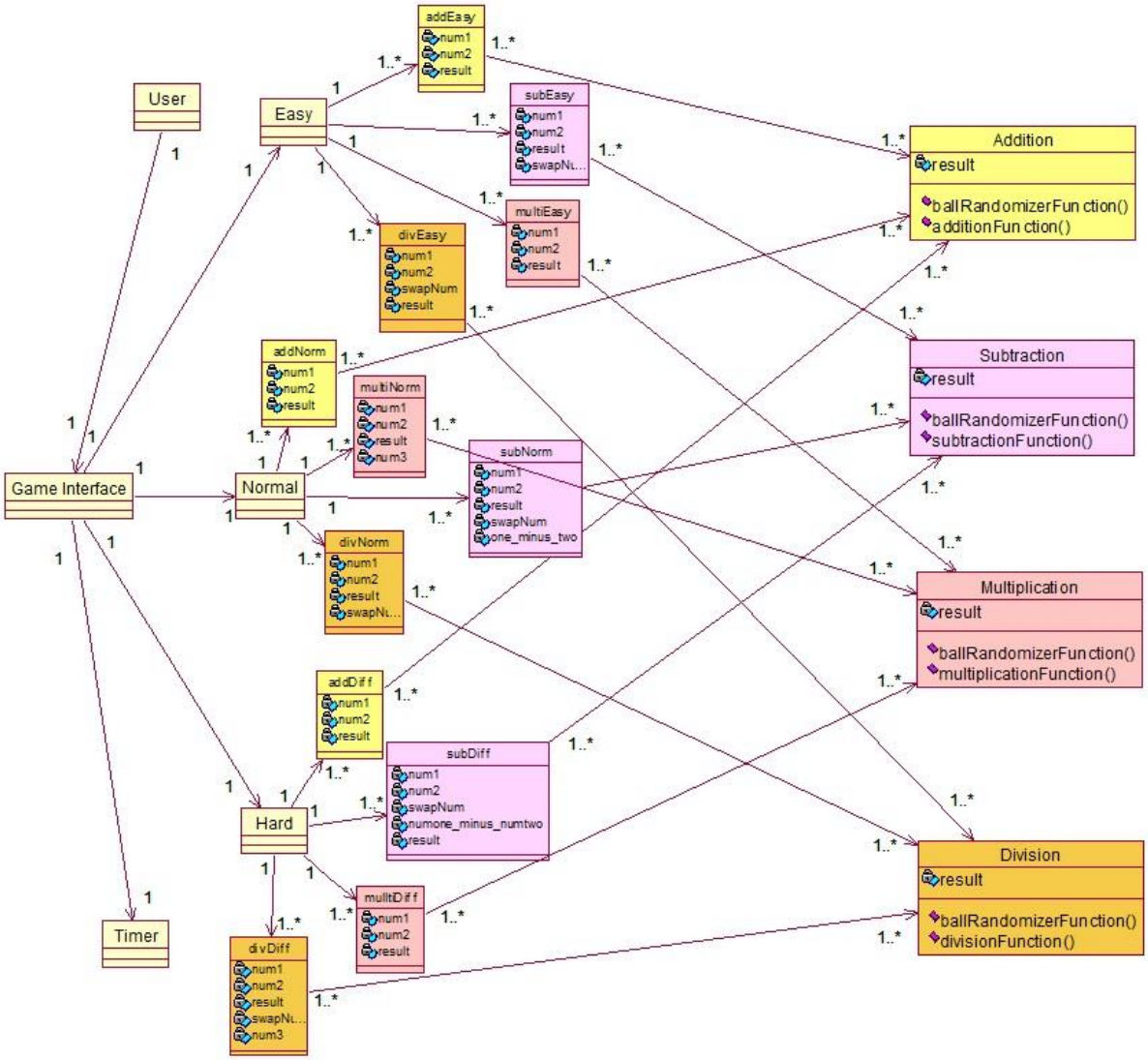

Fig. 15 The mathematical logic functions illustrated in a Class diagram.

4.3.3 Sequence Diagrams

As this is a web app and a class diagram may not do justice to the sequence of events the following sequence diagram illustrate the events or actions as the take place within the game.

Note: As these images illustrate the sequence of events in a web app which has its basis in the HTML5 markup language the labels on the events or messages does not necessarily equate to methods or functions within the code.

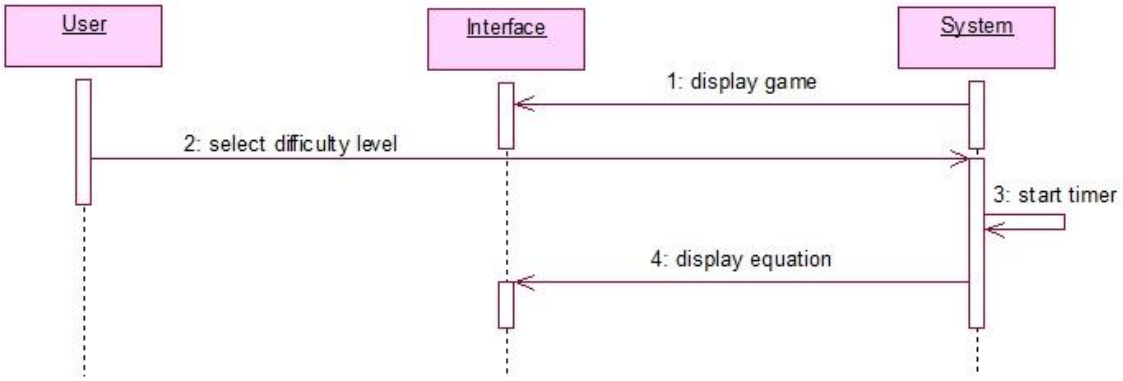

Fig. 16 Sequence diagram for the Select Difficulty Level

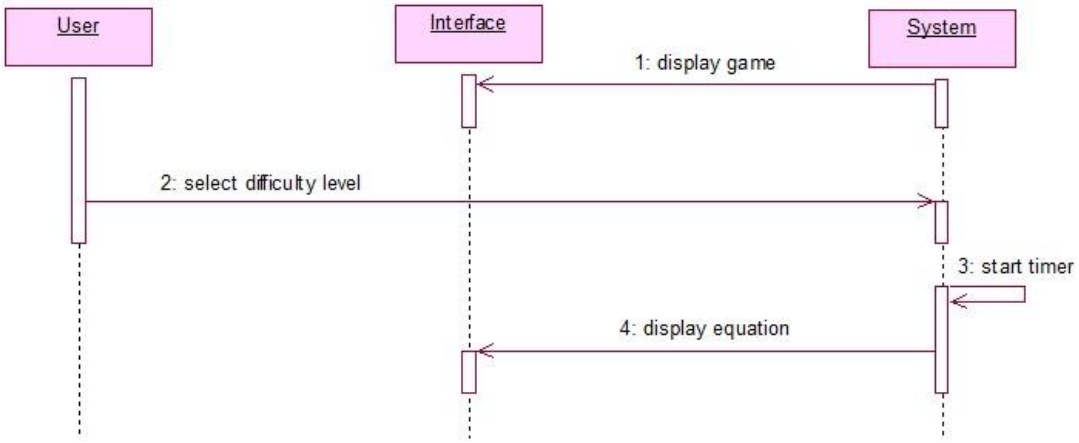

Fig. 17 Sequence diagram for the Countdown timer.

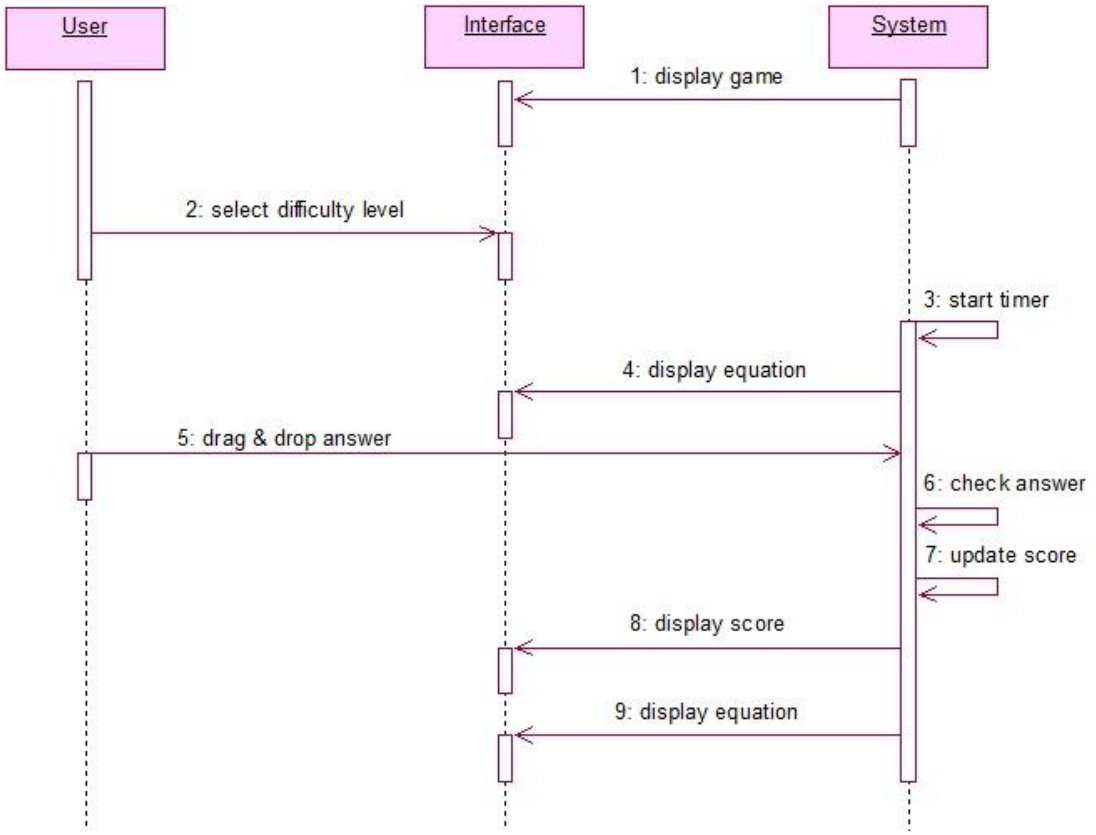

Fig. 18 Sequence diagram for Display equation

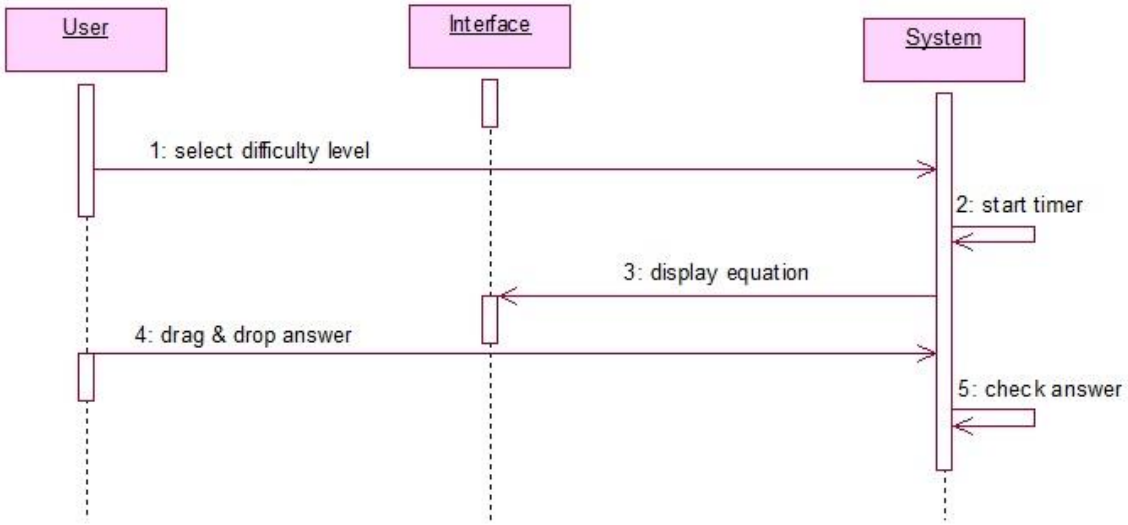

Fig. 19 Sequence diagram for Drag & Drop answer.

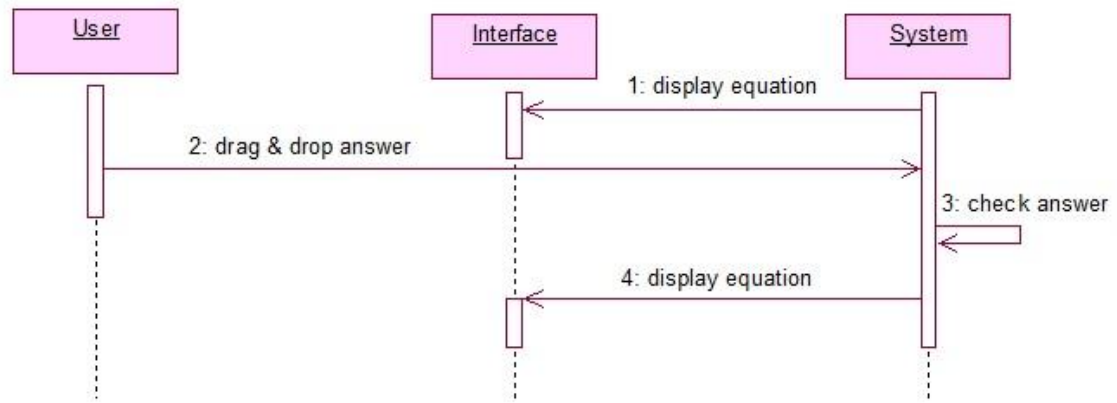

Fig. 20 Sequence diagram for Check answer.

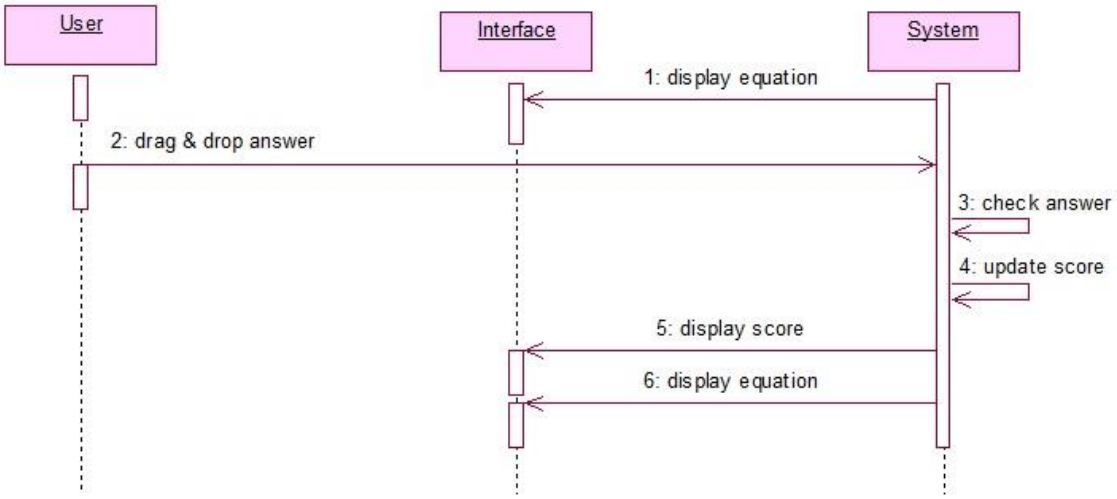

Fig. 21 Sequence diagram for the Update score.

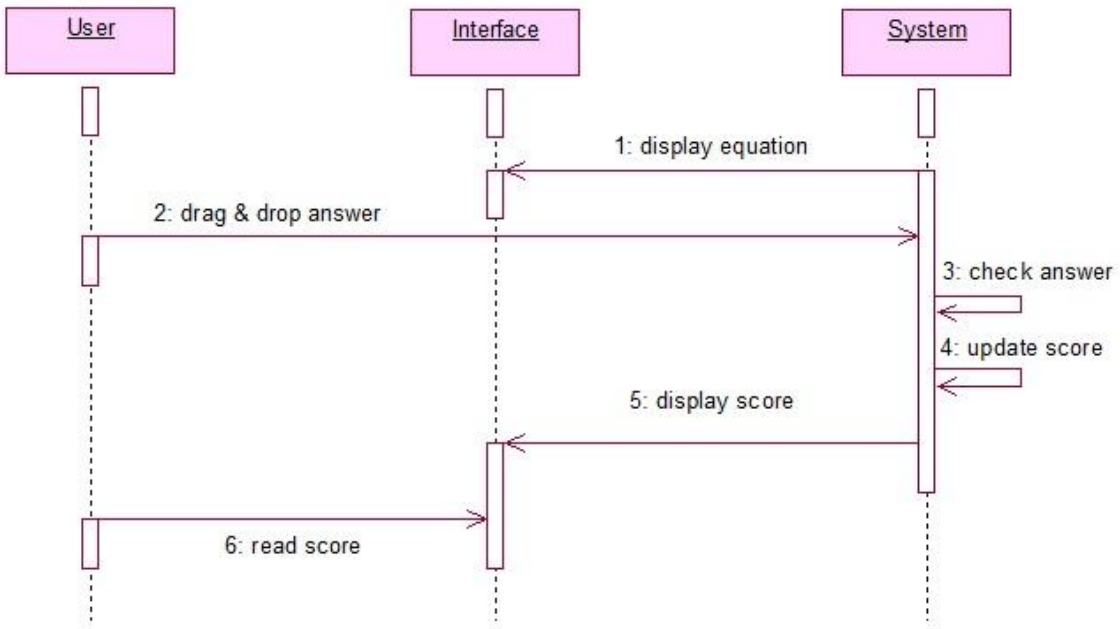

Fig. 22 Sequence diagram for Display score.

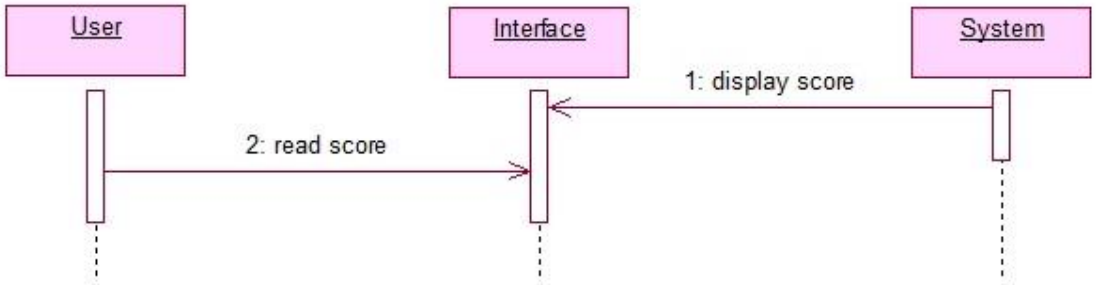

Fig. 23 Sequence diagram for Read score.

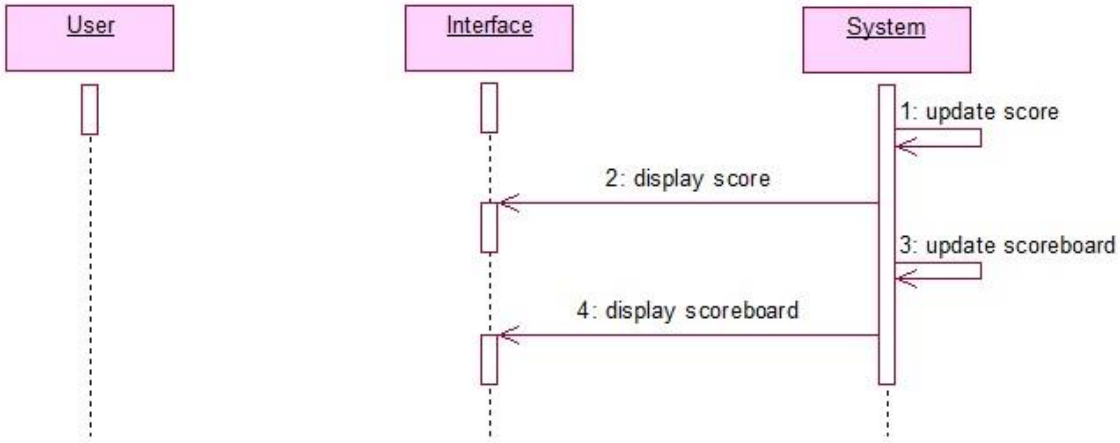

Fig. 24 Sequence diagram for the Update scoreboard.

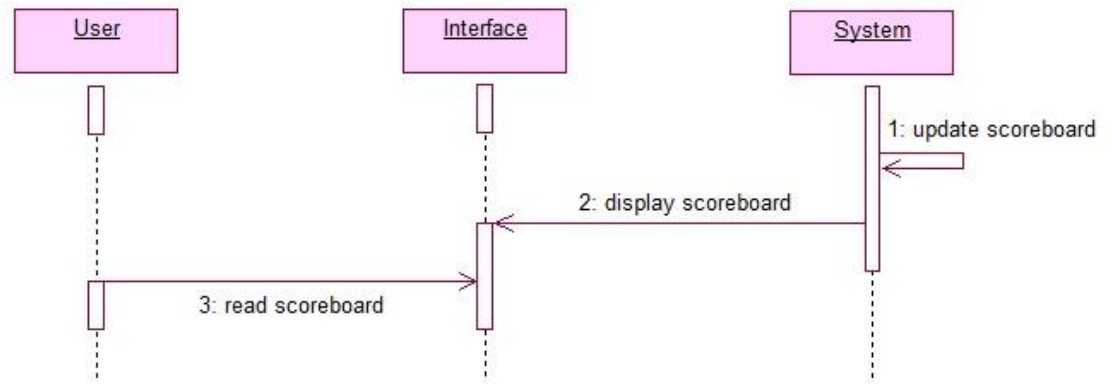

Fig. 25 Sequence diagram for Display scoreboard.

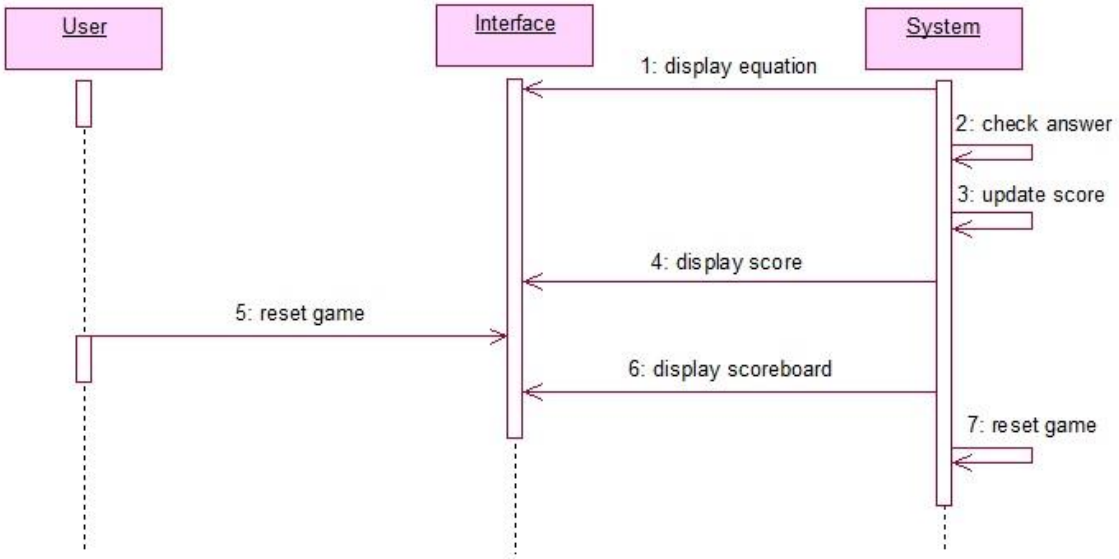

Fig. 26 Sequence diagram for Reset Game.

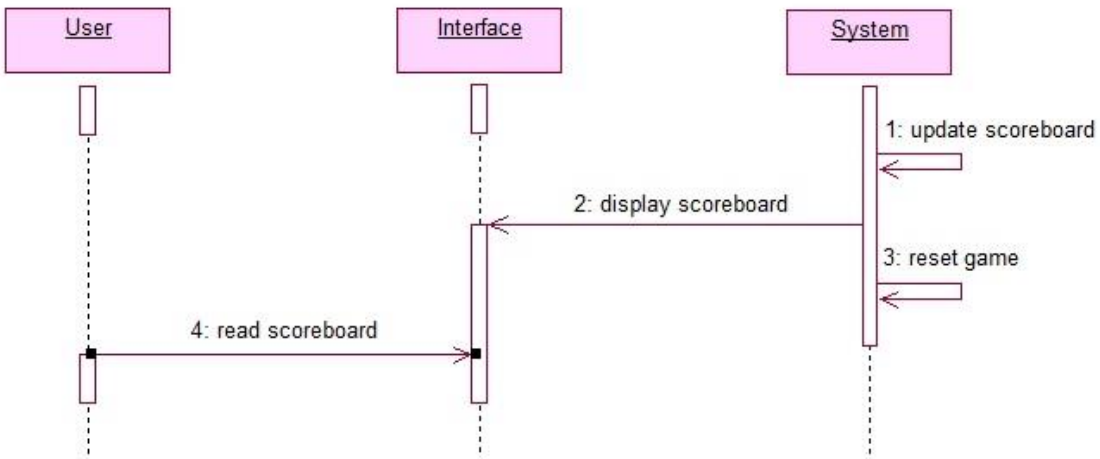

Fig. 27 Sequence diagram for Read scoreboard.

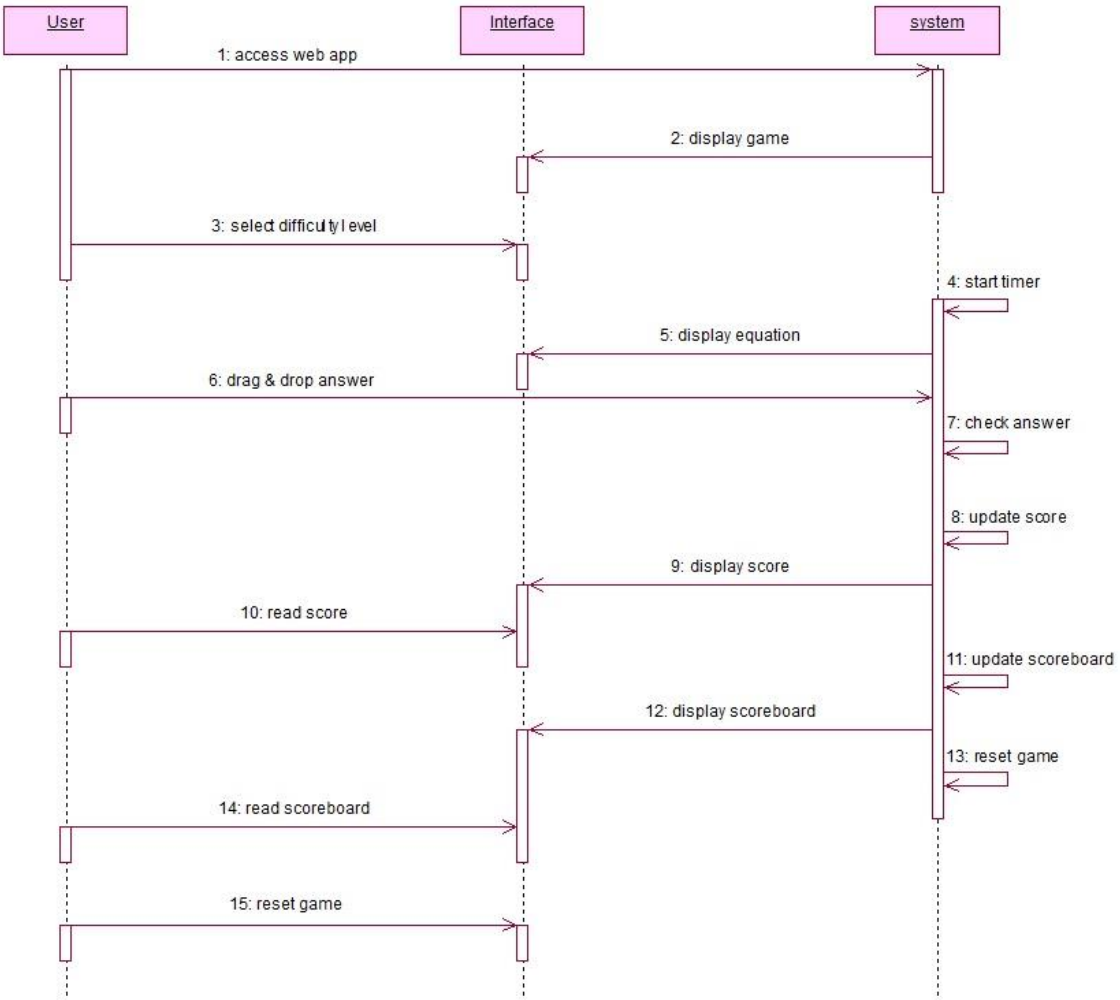

Fig. 28 Sequence diagram for all of the game events.

4.4 Mathematical Formulas & Algorithm Design

4.4.1 Algorithm Design for Addition Equations

The equations that will be prompted to the user will be made up randomly chosen whole numbers. This will be achieved by means of the *Math.floor()* and the *Math.random()* Javascript functions. What follows in this section are examples of game outputs of equations and how these may appear onscreen followed by examples of how their design may be structured within the programming code.

4.4.1.1 Easy Level Addition

An example of an addition equation which may be displayed onscreen when the 'Easy' level mode is selected:

$$12 + 4 = ?$$

num1 = 12,

num2 = 4,

result = (num1 + num2);

result = 16.

System should accept 16 as the correct answer.

4.4.1.2 Normal Level Addition

An example of an addition equation which may be displayed onscreen when the 'Normal' level mode is selected:

$$12 + 4 + 2 = ?$$

num1 = 12,

num2 = 4,

num3 = 2,

result = (num1 + num2 + num3);

result = 18.

System should accept 18 as the correct answer.

4.4.1.3 Hard Level Addition

An example of an addition equation which may be displayed onscreen when the 'Hard' level mode is selected:

$$4^2 + 2^2 = ?$$

num1 = 4^2,

num2 = 2^2,

Numbers will be squared by means of the Math.pow() function and these numbers will not be high in value.

result = (num1 + num2);

result = 20.

System should accept 20 as the correct answer.

4.4.2 Algorithm Design for Subtraction Equations

Note: As alluded to previously all subtraction functions will require that the second randomly generated number must be less than the first randomly generated number. To ensure this a condition will have to be implemented within all functions that swaps these numbers if this is the case.

4.4.2.1 Easy Level Subtraction

An example of a subtraction equation which may be displayed onscreen when the 'Easy' level mode is selected:

$$12 - 4 = ?$$

num1 = 12,

num2 = 4,

If *num2 equals num1* ***then***

num2=num2+3; *//to ensure number values are not the same*

If *num1 is less than num2* ***then***

Swap num2 with num1;

result = (num1 – num2);

result = 8.

System should accept 8 as the correct answer.

4.4.2.2 Normal Level Subtraction

An example of a subtraction equation which may be displayed onscreen when the 'Normal' level mode is selected:

$$(12 - 4) + 2 = ?$$

num1 = 4,

num2 = 12,

If *num2 equals num1* ***then***

num2=num2+3; *//to ensure number values are not the same*

If *num1 is less than num2* ***then***

Swap num2 with num1;

num3 = 2,

result = (num1 – num2 + num3);

result = 10.

System should accept 10 as the correct answer.

4.4.2.3 Hard Level Subtraction

An example of a subtraction equation which may be displayed onscreen when the 'Hard' level mode is selected:

$$(12 - 4)\ x\ 2 = ?$$

num1 = 12,

num2 = 4,

If *num2 equals num1* ***then***

num2=num2+3; *//to ensure number values are not the same*

If *num1 is less than num2* ***then***

Swap num2 with num1;

num3 = 2,

result = ((num1 – num2) x num3);

result = 16.

System should accept 16 as the correct answer.

4.4.3 Algorithm Design for Multiplication Equations

4.4.3.1 Easy Level Multiplication

An example of a multiplication equation which may be displayed onscreen when the 'Easy' level mode is selected:

$$12 \; x \; 4 \; = ?$$

num1 = 12,

num2 = 4,

result = (num1 x num2);

result = 48.

System should accept 48 as the correct answer.

4.4.3.2 Normal Level Multiplication

An example of a multiplication equation which may be displayed onscreen when the 'Normal' level mode is selected:

$$2 + (3 \; x \; 7) = ?$$

num1 = 2,

num2 = 3,

num3 = 7,

result = (num1 + (num2 x num3));

result = 23.

System should accept 23 as the correct answer.

4.4.3.3 Hard Level Multiplication

An example of a multiplication equation which may be displayed onscreen when the 'Hard' level mode is selected:

$$3^2 \; x \; 2 = ?$$

num1 =3^2,

num2 = 2,

The first random number will be squared by means of the Math.pow() function and it will not be high in value.

result = (num1 x num2));

result = 18.

System should accept 18 as the correct answer.

4.4.4 Algorithm Design for Division Equations

4.4.4.1 Easy Level Division

An example of a division equation which may be displayed onscreen when the 'Easy' level mode is selected:

$$12 / 4 = ?$$

num1 = 12,

num2 = 4,

If *num2 equals num1* ***then***

num2=num2+3; *//to ensure number values are not the same*

If *num1 is less than num2* ***then***

Swap num2 with num1;

If *num1 divided by num2 does not have a remainder of zero* ***then***

num1 equals num1 minus the remainder;

result = (num1/num2);

result = 3

System should accept 3 as the correct answer.

4.4.4.2 Normal Level Division

An example of a division equation which may be displayed onscreen when the 'Normal' level mode is selected:

$$(12 / 4) + 6 = ?$$

num1 = 12,

num2 = 4,

If *num2 equals num1* ***then***

num2=num2+3; *//to ensure number values are not the same*

If *num1 is less than num2* ***then***

Swap num2 with num1;

If *num1 divided by num2 does not have a remainder of zero* ***then***

num1 equals num1 minus the remainder;

num3 = 6,

result = ((num1/num2)+num3);

result = 9

System should accept 9 as the correct answer.

4.4.4.3 Hard Level Division

An example of a division equation which may be displayed onscreen when the 'Hard' level mode is selected:

$$(12 / 4) \, x \, 6 = ?$$

num1 = 12,

num2 = 4,

If *num2 equals num1* ***then***

num2=num2+3; *//to ensure number values are not the same*

If *num1 is less than num2* ***then***

Swap num2 with num1;

If *num1 divided by num2 does not have a remainder of zero* ***then***

num1 equals num1 minus the remainder;

num3 = 6,

result = ((num1/num2) x num3);

result = 18

System should accept 18 as the correct answer.

5 Implementation

5.1 Overview

The implementation of this project's web application had two strands. One involved the creation of an app suitable for the desktop environment and the other involved the development of an app capable of running on mobile devices with the Android operating system. As expected these two developments would be produced differently utilizing different techniques but ultimately the core technologies will remain the same. Both strands implemented the same proposed technologies from the system design and as this projects aim is to explore the extent and limitations of web apps, the desktop app will be tested with all of the leading browsers and the mobile app will be tested against an app constructed with the aid of a development framework.

As this projects research showed, there are a number of these frameworks now available and for this implementation PhoneGap was chosen. Both Apps will incorporate a number of HTML5's APIs and CSS3 techniques. These API's and techniques are relatively new and as HTML5 has yet to be officially standardised, some revisions are still under review. This has led to various reviews and reports stating that they are not completely ready for standardisation due to various problems that arise with their implementation. Two of these troublesome APIs are Drag and Drop and Web Storage or Local storage. These will be implemented in both the Desktop and Mobile applications for examination. As the mobile app is expected to be a condensed incarnation of the desktop app it will therefore only feature APIs relevant to its core functionality. Some APIs such as Geolocation may not be pertinent to the games operations.

Operating Environment	APIs	Technologies, Techniques, Frameworks and Polyfills.
Desktop	Drag and Drop, Local storage, Canvas, ClassList, Geolocation, CSS3 Transitions.	HTML5, CSS3, JavaScript, JSON, jQuery, DOM, Modernizer.
Mobile	Drag and Drop, Local storage, Canvas, ClassList, CSS3 Transitions.	HTML5, CSS3, JavaScript, JSON, jQuery, DOM, Hammer.js, Touch and Punch, Modernizer, PhoneGap, Fastclick.

Table 3 Table of tasks and the technologies involved.

5.2 The Desktop Application

The implementation phase of the project began with the creation of the MathMagic Basketball web application for desktop computing. This would also form the basis for the mobile app's development providing an oversight of what may and may not be applicable to that implementation. Because of this both apps were not fully developed side by side, the development of the desktop app was near completion before the mobile app development began.

5.2.1 The Drag and Drop API

The development of the desktop app began with the creation of a prototype which was developed with basic HTML and CSS design. The Drag and Drop API was then incorporated into this design. When its basic functionality was in operation then the rest of the design of the app continued by building around the drag and drop API. The images in Fig. 29 and Fig. 30 illustrate early prototypes of the desktop web application with a makeshift and barebones drag and drop implementation.

In Fig. 29 Numbers are dragged from the source area to a dropzone as can be seen with the number 17 example. In Fig.30 balloon images are implemented as the dragged objects. These two procedures of dragging balloons and numbers would later need to be merged to achieve the desired output of basketballs numbered with possible solutions to the game's proposed equations.

Fig. 29 Prototype with numbers being dragged.

Fig. 30 Prototype with balloons as dragged objects.

It can also be seen from Fig. 30 that the game was originally tested with four buttons each used to initiate a game session categorised by a designated math operator, addition, subtraction, multiplication and division. In the early stages of development this method of selection was deemed unacceptable as it did not offer the user a choice of any difficulty level.

The Drag and drop API was very troublesome to implement and difficult to get working correctly. It had issues with cross-browser compatibility and this is probably why it is known as the black sheep API. A javascript titled *bballfunctions.js* was created for the purpose of the Drag and drop implementation. The Drag and drop development began by first specifying an object or in this case, each of the four objects, as being draggable. This is done with the Javascript draggable method which is assigned with a Boolean value of true. This enables a number of events to be specified and assigned to trigger when the object is dragged. The available events are:

- dragstart
- drag
- dragenter
- dragleave
- dragover
- drop
- dragend

The main events that are commonly used are dragstart, dragover, drag and drop and these are the events that are implemented in this application. These events are then handled by ondragstart, ondragover and ondrop. The dragged object is dragged by its id which is attached as Text data to

the ondragstart handler. This was achieved by adding the following line of code to the dragstart method.

```
ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
```

Text is the data type and the value is represented by ev.target.id, which is the id of the element being dragged.

Next the ondragover handler cancels or prevents the default behavior or handling of the drop element which allows the dropping of the object to take place. There are various ways to achieve this but in this implementation the following line was inserted into the dragover and drop methods.

```
ev.preventDefault();
```

When inserted into the drop event function it prevents the default handling of the dropzone element since the default behavior is not to allow drag and drop.

What happens when the object is dropped over the dropzone is defined by the ondropover event handler and when it is dropped onto the dropzone the ondrop event handler is implemented.

The full list of event handlers is:

- **draggable** – this specifies a an element as draggable or not with the values of true, false or auto.
- **ondrag** – fires while dragging.
- **ondragstart –** fires when dragging starts.
- **ondragstart –** fires when dragging starts.
- **ondragenter** – fires when an object is dragged in the dropzone area.
- **ondragover** – fires while the object is being dragged over the dropzone.
- **ondragleave** – fires when an object is dragged outside of a dropzone.
- **ondragend** – fires when dragging stops and the mouse is released.
- **ondrop** – fires when data is dropped on it.

The ondrop handler which fires the drop event handler function is where the operations are set for what happens when an object has been dropped.

In this case it handles and defines the functions for when a basketball has been dropped into the net. The first main task of the drop event function is to clone a copy of the basketball with the DOM *cloneNode()* method and drop it into the net. This is achieved with the inclusion of the following line of code:

```
ev.target.appendChild(document.getElementById(data).cloneNode(true));
```

Once a clone of the basketball had been set to drop in to the net the line of code was appended to have its opacity set to 0.7 with the following line to give it the appearance of being inside a net.

```
ev.target.appendChild(document.getElementById(data).cloneNode(true)).style.opacity = '0.7';
```

The reason for cloning the object was so that the original copy of the basketball could remain in its source position within a HTML table element as seen in the following snippet of code.

```
<table>
  <tr>
    <td></td>
    <td><div id="bball1" class="number_font" draggable="true" ondragstart="return dragstart(event)"
     ondragleave="return dragleave(event, this.id)"></div></td>
    <td></td>
  </tr>
  <tr>
    <td><div id="bball2" class="number_font" draggable="true" ondragstart="return dragstart(event)"
     ondragleave="return dragleave(event, this.id)"></div>
    </td>
    <td></td>
    <td><div  id="bball3" class="number_font" draggable="true" ondragstart="return dragstart(event)"
    ondragleave="return dragleave(event, this.id)"></div></td>
  </tr>
  <tr>
    <td></td>
    <td><div id="bball4" class="number_font" draggable="true" ondragstart="return dragstart(event)"
   ondragleave="return dragleave(event, this.id)"></div></td>
    <td></td>
  </tr>
</table>
```

What happens to the source of the dragged object or basketball when it was dragged was a major issue for the development. At first it seemed that the best way to attempt this was to remove it in the dragstart event function as it was initially dragged then appended back to position from within the drop function with DOM methods such as in the following:

```
function dragStart(ev) {
ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
var node = document.getElementById(ev.target.getAttribute('id'));
if (node.parentNode) {
    var parent = node.parentNode;
    parent.removeChild(node);     //remove the original child
   }
}

function drag(ev) {
ev.dataTransfer.setData("Text",ev.target.id);
ev.dataTransfer.dropEffect = 'move';
var node = document.getElementById(ev.target.id);
if (node.parentNode) {
    var parent = node.parentNode;
    parent.appendChild(node);     //append it back
   }
}
```

This however was not appropriate as it only appended the basketball to the end of the table with the *appendChild()* function and conflicted with the mathematics logic that was to rely on the positions of the objects. The next attempt at a solution involved the implementation of the ondragleave handler which would fire a newly created dragleave event function. This worked perfectly until it was noticed that when a basketball was dragged over another basketball they then also disappeared. Eventually the best approach to this was to go with what was originally implemented and that was to set the opacity of the source basketball tp zero. It seemed the best approach as this original source basketball remained in position for the next round of possible answers. Although it was a less cumbersome method, I am not sure if it was the best efficient but it was achievable in a much straightforward way with much less coding involved.

Although removing the original basketball form the display when dragged was important this does not completely describe the full operation of the drop event. Its main function is to accept the dropped basketball and identify if it was marked with the correct answer to an equation or not. This was achieved by seeking to find if each of the basketballs were located (i.e. dropped) on the

target dropzone. This was done by issuing numerous jQuery-coded if statements testing for each of the four basketballs within each of the three levels of difficulty. An example snippet of the code for the Easy level can be seen in the following:

```
if($('#drop_zone').find('#bball1').size() == 1 && randomBall  == 1 && level == 1) {
    correctAnswer(level);
   }
   else if($('#drop_zone').find('#bball2').size() == 1 && randomBall == 2 && level == 1) {
   correctAnswer(level);
   }
   else if($('#drop_zone').find('#bball3').size() == 1 && randomBall == 3 && level == 1) {
   correctAnswer(level);
   }
   else if($('#drop_zone').find('#bball4').size() == 1 && randomBall == 4 && level == 1) {
    correctAnswer(level);
   }
```

What exactly happening here is that in the first line, the if statement is seeking to find *bball1* and only *bball1* in the target dropzone and if it does and the *randomBall* variable equals 1 and if that does and the *level* variable equals 1 then execute the *correctAnswer()* function by passing the *level* variable with a value of 1 to it.

The *randomBall* variable and the *level* variable are part of the mathematical functionality of the game and they are defined in the mathLogic script. Their purpose is to aid in the identification of whether or not a dropped ball is a correct answer or a wrong answer.

Whether a correct or incorrect answer in the form of a basketball is dropped on the dropzone two game occurrences should happen next. One such occurrence for a correct answer and one for an incorrect answer. However, with regards to the development only one event is programmed to be handled. This is the event for when a correct answer is dropped and this is handled by a call to a *correctAnswer()* function. If an incorrect or wrong answer is dropped then no event takes place or is handled. The games response to an incorrect answer drop materializes by displaying a 'Wrong Answer Try Again' onscreen message followed by a wait for the equations correct answer to be dropped in the net. This is handled in the program with the *if* statement described above to

find the correct answer being simply appended with an else condition for the 'Wrong Answer …' notification message.

This *correctAnswer()* function could be seen as the most important and slightly complex function within the game. It will trigger the display of a 'Correct' onscreen message for a period of 800 milliseconds and when this time period ends, the game will move on with this message disappearing, the user's score will increment by 10 points and a new equation will be displayed. Javascript's *setTimeout()* function was used for the purpose of setting the time delay period.

In its declaration this function takes in a *levelnumber* variable which is the level number of the *level* (representing the difficulty level) argument passed in from the function call in the drop event. This is used in conjunction with another generated random number between 1 and 4 which is assigned to a variable called *rand*. The purpose of this randomized value is to dynamically select which form of equation comes next in the game and this function will assign numbers 1 to 4 to the operator's addition, subtract, multiplication or division respectively.

The selection of the next equation also takes into account the *levelnumber* so as to ensure that the randomly selected equation is maintained within a set level of difficulty i.e. an 'Easy' level subtraction equation should appear and not a 'Hard' level equation when a user selects the easy level of difficulty. When the random value of *rand* is considered with the *levelnumber* value passed in from the function call in the drop event, the resulting action is to invoke the assigned functions from the *mathLogic.js* script. For example, if the *levelnumber* is 1(i.e. easy level) and the *rand* value is 1 (i.e. addition) then easy addition functions are called. This is demonstrated in the following snippet of code from the *correctAnswer()* function:

```
setTimeout(function() {
    var rand=Math.floor((Math.random()*4)+1);
    if (rand === 1 && levelnumber===1) { addEasy(); additionFunction(); }
    else if (rand === 2 && levelnumber===1) { subEasy(); subtractionFunction(); }
    else if (rand === 3 && levelnumber===1) { multiEasy(); multiplicationFunction(); }
    else if (rand === 4 && levelnumber===1) { divEasy(); divisionFunction(); }

    if (rand === 1 && levelnumber===2) { addNorm(); additionFunction(); }
    else if (rand === 2 && levelnumber===2) { subNorm(); subtractionFunction(); }
    else if (rand === 3 && levelnumber===2) { multiNorm(); multiplicationFunction(); }
    else if (rand === 4 && levelnumber===2) { divNorm(); divisionFunction(); }

    if (rand === 1 && levelnumber===3) { addDiff(); additionFunction(); }
    else if (rand === 2 && levelnumber===3) { subDiff(); subtractionFunction(); }
    else if (rand === 3 && levelnumber===3) { multiDiff(); multiplicationFunction(); }
    else if (rand === 4 && levelnumber===3) { divDiff(); divisionFunction(); }
  }, 800);
  updateScore();
```

It can be seen from the code snippet that the call to the new equation functions only occurs after a period of 800 milliseconds. Finally in the *correctAnswer()* function the score is incremented by 10 and this happens when the *updateScore()* function is invoked from the *timer.js* script.

5.2.2 Mathematical Logic

The mathematical logic for the games equations is implemented in the *mathLogic.js* script. In a function called *ballRandomizerFunction()* a random number is generated between 1 and 4 and assigned to a variable called *randomBall* . If *randomBall* has a value of 1 then the correct answer to an equation will be placed at position 1 in the basketball source area (see Fig. 31). If *randomBall* has a value of 2 then the correct answer to an equation will be placed at position 2 and so on.

Fig. 31 The positioning of the basketballs

This occurs in all of the top-level functions associated with the mathematical operators *additionFunction(), subtractionFunction(), multiplicationFunction()* or *divisionFunction()*. I call these top-level operator functions because they contain instructions that are more globally specific to a mathematical operator.

The *ballRandomizerFunction()* function is invoked at the beginning of each top-level operator function. The *randomBall* variable is then checked in the *drop* event function so as to identify if the basketball dropped by the user matches the basketball assigned with the result by the system. A snippet of the code from the top-level *divisionFunction()* can be seen from the following:

```
function divisionFunction() {
   var remainder;
   ballRandomizerFunction();

   if(randomBall == 1)
      {
      document.getElementById("bball1").innerHTML=result;
      document.getElementById("bball2").innerHTML=result+2;
      document.getElementById("bball3").innerHTML=result+4;
      document.getElementById("bball4").innerHTML=result+1;
      }
   else if(randomBall == 2)
      {
      document.getElementById("bball1").innerHTML=result+2;
      document.getElementById("bball2").innerHTML=result;
      document.getElementById("bball3").innerHTML=result+4;
      document.getElementById("bball4").innerHTML=result+1;
      }
   else if(randomBall == 3)
      {
      document.getElementById("bball1").innerHTML=result+2;
      document.getElementById("bball2").innerHTML=result+4;
      document.getElementById("bball3").innerHTML=result;
      document.getElementById("bball4").innerHTML=result+1;
      }
   else
```

```
    {
    document.getElementById("bball1").innerHTML=result+2;
    document.getElementById("bball2").innerHTML=result+4;
    document.getElementById("bball3").innerHTML=result+1;
    document.getElementById("bball4").innerHTML=result;
    }
}// end function
```

This snippet of code illustrates how a correct answer or result is randomly assigned in the *divisionFunction()* by means of the *randomBall* variable.

The top-level operator functions are each supported by three low-level functions that represent the three difficulty levels. These are outlined in the following table and their hierarchical structure is illustrated in the class diagram that follows:

additionFunction()	**subtractionFunction()**	**multiplicationFunction()**	**divisionFunction()**
addEasy()	subEasy()	multiEasy()	divEasy()
addNorm()	subNorm()	multiNorm()	divNorm()
addDiff()	subDiff()	multiDiff()	divDiff()

Table 4 The operator functions and their associations.

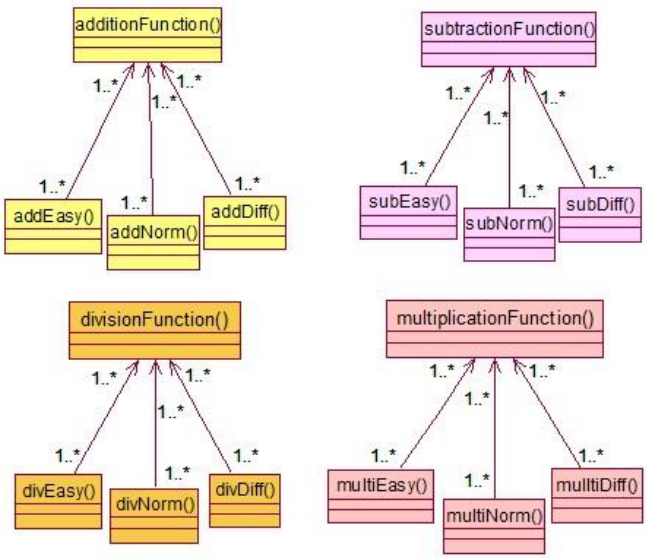

Fig. 32 A class diagram of operator functions and their associations.

When called upon an operator function will work in conjunction with one of the difficulty level operator functions. For example, the ***divisionFunction()*** will be invoked along with either the ***divEasy()*** or the ***divNorm()*** or the ***divDiff()*** function. This is because these three low-level functions hold code logic that is locally specific to each associated difficulty level for a selected operator.

What follows is an snippet of the code from the hard division function (*divDiff()*). I chose this example as it is probably the most complex function of all the functions:

```
function divDiff(){
   num1 = ~~((Math.random()*30)+2);
   num2 = ~~((Math.random()*14)+2);
   num3 = ~~((Math.random()*10)+1);
   if (num2 === num1) {
      num2 += 3;
   }
   if(num1<num2) {
            swapNum=num1;
            num1=num2;
            num2=swapNum;
   }
   if (num1 % num2 !== 0) {
      remainder=num1%num2;
      num1 -= remainder;
   }
   result=(num1/num2)*num3;
   document.getElementById("question_output").innerHTML=
                              "(" + num1 + " &#247; " + num2 + ") &#215; " + num3 + " = ?";
   level = 3;
   }
```

There are a number of points of interest that be observed from the divDiff() function code. Firstly, as mentioned earlier, this is code only specific to the hard division function. Three numbers are defined for the equation output. These are generated randomly and within an acceptable range for their position in the equation output so as to avoid becoming too large or complex for the game player. The conditional if statements ensure that:

(a) The second number is not equal to the first number to avoid the first number being divided by itself as this is deemed too easy for an equation.

(b) The second number cannot be greater than the first number as division would not result in a whole number.

(c) When the first number is divided by the second number then there is no resulting remainder left over.

The equation is formulated and displayed to screen when the function is called and finally when it is called the *level* variable is set to 3 to reflect a difficult level of 'Hard'.

Note: This function is in keeping with the pseudo code examples that were outlined in section *4.4 Mathematical Formulas & Algorithm Design* under part *4. System Design* of this thesis.

5.2.3 The Canvas API

As this web application was a game showcasing interactivity and dynamism associated with HTML5 and CSS3 implementations all of the games functionality will operate from one web page. This will be the Home or index page. The website will also contain two other pages that are relevant to the application. These are a Mobile page for access to the Android app and a Help page that contains step-by-step instructions for playing the game.

The Home page or Game page featured a heading which displayed the title 'MathMagic Basketball'. This was created with the implementation of the HTML5 Canvas API [21]. The functionality for the heading was placed in an external script called *mainHeading.js* which was linked to within the head tag of the index or Home page. Within this document the Canvas text was accessed with the implementation of a canvas HTML5 tag such as that seen in the following code snippet:

```
<canvas id="canvasLogo" width="490" height="110"></canvas>
```

The script references the canvas area by the *canvasLogo* id and this script is initialised when the web application runs. The canvas tag encases the canvas area allowing the *mainHeading.js* script to paint text in 2d onto it. At present HTML5 only supports a rendering setting of 2d [21] although it is expected that a 3d context will be defined in later revisions. Text can be placed between the canvas tags as a browser fallback if the script has problem loading.

In the script a two canvas context objects are created using the *getContext()* method. One object *ctx* for MathMagic and one object *ctx2* for Basketball which is positioned on a slightly lower level as seen in Fig. 33.

Fig. 33 The canvas text heading.

The place that will be painted on within the canvas is defined as *ctx* and *ctx2*. These variables have properties added to them to draw the actual text. The text for both MathMagic and Basketball is drawn or written using the *fillText()* method and the text outline for Basketball is drawn using the *strokeText()* function. These two functions are issued with three arguments; the text, the x-coordinates and the y-coordinates. In the script I found the best approach was to create variables for these coordinates assigning the *canvas.width* and the *canvas.height* before adding them as arguments. The benefit to this was that the variables could easily be altered when seeking to find the best position for the text. The font property was added to *ctx* and *ctx2* along with other properties that assigned effects like shadow effects. One important property that was added to *ctx2* for the Basketball text was the rotate property. This enabled the Basketball text to rotate or appear at an angle by utilizing the Javascript Math.PI property. The core of the *mainHeading.js* script can be seen in the following code snippet:

```
var ctx = canvas.getContext("2d");
var ctx2 = canvas.getContext("2d");
var x1 = canvas.width / 1.85;
var y1 = canvas.height / 1.8;
var x2 = canvas.width / 1.49;
var y2 = canvas.height / 1.0;

ctx.shadowOffsetX = -5;
ctx.shadowOffsetY = 5;
ctx.shadowBlur = 10;
ctx.shadowColor = "#ffff00";
ctx.textAlign = 'center';
ctx.fillStyle = "#FF0";
ctx.fillText( "MathMagic", x1, y1);

ctx2.shadowOffsetX = -4;
ctx2.shadowOffsetY = 4;
ctx2.shadowBlur = 10;
ctx2.shadowColor = "#FF99CC";
ctx2.font = " italic 60px Comic Sans MS";
```

```
ctx2.fillStyle = "#F30";
ctx2.rotate(-Math.PI/55);
ctx2.fillText("Basketball",x2, y2);
ctx2.lineWidth=0.8;
ctx2.strokeText("Basketball",x2, y2);
```

5.2.4 The Main Content

The overall layout of the Home page aimed to be as appealing to the user as possible. In the header section and below the MathMagic Basketball canvas heading there are links to the Mobile and Help pages. To the left of the heading is the MathMagic logo and to the right is the game's countdown timer. The main content is situated below the header section and this is divided into three panels, a left panel, a center panel and a right panel.

The left panel contains a current in-game score display, a scoreboard displaying the previously obtained highest scores, a 'Game Reset button' to refresh the game at any point and a 'Clear Table' button to completely clear the scoreboard of all entries. The functionality of the scoreboard was achieved with the implementation of HTML5's Local storage API and the clear table button implements the Javascript *clear()* and *reload()* functions.

The center panel is where the user will focus most of their attention. At the top it will feature the three inline difficulty level buttons of Easy, Normal and Hard as displayed in Fig 34.

Fig. 34 The center panel.

The job of these buttons is exactly as expected, they will initiate a new game relevant to the user's choice of difficulty level when selected. When a game is in session all of the games buttons

except for the game reset button are disabled. When a game is not in session full functionality is restored to all of the buttons. When a level is selected and the game begins, depending on the level selected, an addition equation is first displayed before the random process of loop through equations is implemented. This is implemented in the index HTML5 file with the following code:

```
<div style="float: left;">
	<button id="addEasy" class="smallButton"
		onclick="addEasy();
		additionFunction();
		startTimer();">Easy
	</button>
</div>
<div style="float: right;">
	<button id="addDiff" class="smallButton"
		onclick="addDiff();
		additionFunction();
		startTimer();">Hard
	</button>
</div>
<div style="margin-left: 25px;">
	<button id="addNorm" class="smallButton"
		onclick="addNorm();
		additionFunction();
		startTimer();">Normal
	</button>
</div>
```

The process of looping through the equations and displaying the next random equation is defined in the *setTimout()* method within the *correctAnswer()* function in the bballfunction.js script. A snippet of code for this method is displayed in section 5.2.1 The Drag and Drop API.

Below the level of difficulty buttons is where the basketball net or dropzone is positioned. This was constructed with the insertion of a basketball net in a png image format that required some extensive work in Photoshop to achieve full image transparency. Transparency was required as this image was to be placed onto a white board and this white background board was developed by implementing CSS code.

As mentioned previously, the net is where the target dropzone is located and this is implemented in the Home or index page document with the following HTML5 code that includes the ondrop and the ondragover event handlers:

```
<section id="behindBoard">
    <div id="innerBoard">
     <div class="dropzone-image">
       <div id="drop_zone" ondrop="drop(event)" ondragover="dragover(event)" ></div>
     </div>
    </div>
</section>
```

The position of the dropzone can be seen in Fig. 35 where its location is within the highlighted blue dotted border. This is positioned so as the center of the dropzone will be the place where the basketball is dropped i.e. near the top of the net.

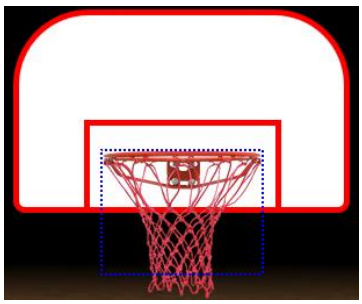

Fig. 35 A blue dotted outline of the actual dropzone.

The most difficult part of the net building process was how to determine the position of all the net's parts. The reason for this was to facilitate the appearance of the dragged ball being dropped into a net which is on top of the white board then the net must be placed on top of the dropzone which in turn must be placed on top of the white board. This was ultimately achieved with CSS styling by assigning the white board div block with the *innerBoard* id (as seen in the code snippet) as *position: relative*, the dropzone div block with the *drop_zone* id as *position:absolute* and the *dropzone-image* class as *position relative*. When fully constructed the displayed result is as depicted in Fig. 36.

Fig. 36 An example of a ball dropped in the net.

Below the net and dropzone within the center panel is the output where the mathematical equations will be displayed and this was developed to resemble a school blackboard. Above this output is a notification panel that is hidden from the user. Its purpose is to display a blinking 'Correct' answer message if the user drops the ball with the correct answer into the net or a 'Wrong Answer! Try again' message if an incorrect answer is dropped. An example of this can be seen in Fig. 37.

Fig. 37 An example of a correct notification message.

The right panel of the main content displays the source area where the user selects a basketball from a choice of four to be dragged. One ball contains the correct answer and the other three contain incorrect answers that are relatively close to the correct answer. When the web application is first opened this panel displays a brief set of instructions on how to begin the game. An example of the two occurrences of the right panel can be seen in Fig. 38.

Fig. 38 The right panel

Lastly, in the interface is the footer. This simply provides the Institute of Technology Blanchardstown (ITB) copyright status, the year of publication along with the author's name and contact email information. In addition to this, links to share the application on social media are provided. These links are provided for Facebook, Twitter and Google plus. A screenshot of the footer is displayed in Fig. 39.

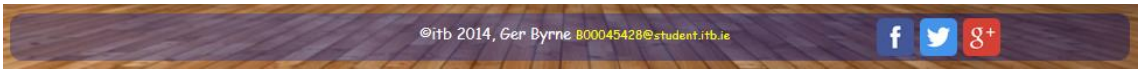

Fig. 39 The footer containing the social network links.

5.2.5 Game Countdown Timer

As stated in section 5.2.3.2 The Main Content, The right hand side of the header features a countdown timer. This is implemented from a script called appropriately timer.js and its main function is to declare and implement functions related to this game timer. The countdown is implemented with a global variable being declared as *count.* This is initialised to 600 and as the countdown is decremented in milliseconds, 600 sets the *count* to begin at 60 seconds. A variable named *time* is also declared and initialised to null. A *startTimer()* function for the timer is implemented and this is called from the difficulty level button onclick event. Within the *startTimer()* function If time is null and count is greater than 0 then a *countdown()* function is invoked. The *countdown()* function implements various if and else statements with some of these nested to implement events on conditions related to the timers countdown. Such statements can be seen in the following code snippet:

```
if (count == 0) {
      if(currentScore<10) {
        alert('Time is up');
      }
      else {
        var username;
        var inputName=
        prompt("\t\tTime up\n\tYou Scored "+currentScore+"\nEnter Your Name Here \n\t","");
            if (inputName!=null) {
                username=inputName;
                if(inputName==="null" || inputName===""){
                    username="No name";
                    location.reload();
                }
                storedEntries = JSON.parse(localStorage.getItem("localstoragekey"));
                if(storedEntries == null) {
                  storedEntries = [];
                  }//end if(storedEntries == null)
                  storedEntries.push(currentScore+" "+username);
                  storedEntries = storedEntries.sort(function(a,b){return parseInt(b)-parseInt(a)});
                  localStorage.setItem("localstoragekey", JSON.stringify(storedEntries));
                  storedEntries.length=10;
          }//end if (inputName!=null)
      }//end else currentScore<10
location.reload();
} //end if count == 0
else {
count--;
```

```
/* Diable all buttons during a game session except game reset button */
document.getElementById("addEasy").onclick=false;
document.getElementById("addNorm").onclick=false;
document.getElementById("addDiff").onclick=false;
document.getElementById("clearButton").onclick=false;
document.getElementById("addEasy").setAttribute("style", "background-color:#0000FF; text-shadow: 0
-1px 0 #000;");
document.getElementById("addNorm").setAttribute("style", "background-color:#0000FF; text-shadow: 0
-1px 0 #000;");
document.getElementById("addDiff").setAttribute("style", "background-color:#0000FF; text-shadow: 0 -
1px 0 #000;");
document.getElementById("clearButton").setAttribute("style", "background-color:#0000FF; text-shadow:
0 -1px 0 #000;");
document.getElementById("countdownClock").style.backgroundImage=
"url('images/basketball_game_dark.jpg')";
time = setTimeout("countdown()", 100);
   }

/* Diable all draggable objects so as not to conflict with any end game alerts */
if (count == 20) {
document.getElementById("bball1").draggable=false;
document.getElementById("bball2").draggable=false;
document.getElementById("bball3").draggable=false;
document.getElementById("bball4").draggable=false;
}
```

The multiple if and else statements in the code snippet are best described in the following pseudo code:

```
If count equals zero then          //i.e. if the game is over
{
  If the current score is less than 10 then          //i.e. if player has not scored
    Display "Time is up";
    Else if the player has scored
    {
       Display "the player's final score and Enter Your Name Here message";
       If there is a score and the user has not entered a name then
         {
          The string "No Name" is to be entered into the scoreboard with the final score;
          Refresh the game page;
         }
      Store the final score and username in Local storage;
     }//end else if
Refresh the game page;
}
Else If count is not equals zero
{
Decrement count from 600 down to zero; // i.e. count--;
Disable all other game buttons except the game reset button by setting onclick to false;
}

If count equals 20 then  //i.e. when the countdown has only 2 seconds left.
```

```
{
Disable the dragging of all basketballs by setting draggable to false;
}
```

The last if statement is implemented to avoid conflicts that occur with alerts while dragging when the game ends. If a user or player is dragging a basketball while count reaches 0 and an alert occurs the browser freezes requiring the browser process to be stopped and restarted. Setting the draggable handler to false on all basketballs two seconds before the game ends is a workaround for this.

5.2.6 The Local Storage API

There are other functions that are declared in the *timer.js* script and these are done for the purpose of implementing the Local storage API which is part of HTML5's Web storage. Web storage is comprised of Local storage and Session storage. Both have the capability to storing user data within a supported browser. However Session storage only stores data for the duration of a browser's session or a browser tab's session. It is lost when the bowser or tab is closed. Local storage on the other hand is much more persistent as it stores data continuously and it is not deleted when a browser is closed. Web storage is commonly referred to as Local storage or Dom storage. This project implements the Local storage API and refer to it as such.

Local storage was incorporated into the project as it is one of HTML5's APIs that has renowned issues with its implementation. These issues are chiefly related to its cross-browser compatibility. This provided an ideal opportunity to test the extent and limitations of dynamic web application content even though this API stored data on the client-side. Another reason for its inclusion was because it seemed to provide the best technique for implementing a scoreboard that stored usernames and scores. Local storage stores data as string types in key/value pairs. The maximum size of a browsers storage that is available differs from browser to browser. This limit could be anywhere from 5mb to 50mb or even Unlimited storage depending on the browser.

The Local storage methods are:

setItem() – stores data - Requires key and value to be stored as arguments.

getItem() – retrieves data. Requires just the key as an argument.

removeItem() – removes data. Requires just the key as an argument.

clear() – clears all data in local storage. No argument required.

5.2.6.1 Creating the Scoreboard

In the *timer.js* script a function named *scoresListDisplay()* is initialised with windows.onload. As the name suggests this function creates and displays an ordered list of previous scores in the form of a scoreboard on the left panel of the Home page. The data that is stored in Local storage with the key *localstoragekey* is retrieved with the *getItem()* method and parsed as a JSON string called *storedEntries,* otherwise *storedEntries* is initialised as an empty array.

Note: Local storage at this point in time cannot hold data types other than strings. However it is possible to store data as an array by converting it to JavaScript Object Notation (JSON) with the methods *JSON.stringify()* for to store and *JSON.parse()* for to retrieve.

The HTML5 code for the left panel on the Home page is shown in the snippet below. This implements interaction with the scoreboard's creation and update operations in the *timer.js* script is shown in the snippet below:

```
<div class="left_panel">
<section>
        <button class="resetButton" onclick="resetFunction()">Game Reset</button>
</section>
<section id="game">
        <div id="inGameScore">Score: 0</div>
        <span> <ol class="high-scores"></ol></span>
</section>
<section>
        <button id="clearButton" class="resetButton"
        onclick="clearLocalStorage()" style="margin-top:15px;">Clear Table</button>
</section>
</div>
```

The *scoresListDisplay()* function from the *timer.js* script that contains the operation for populating the HTML5 scoreboard list is shown below:

```
var storedEntries = [];  //This is a global variable
function scoresListDisplay(){
        storedEntries = JSON.parse(localStorage.getItem("localstoragekey")) || [];
        var highScoreList = game.querySelector("section#game ol.high-scores");
        for(var i = 0; i < 10; i++){
          var st = storedEntries[i];
          var gamescorelist = document.createElement('li');
```

```
            gamescorelist.innerHTML = (typeof(st) != "undefined" ? st : "--" );
            highScoreList.appendChild(gamescorelist);
        }
}
```

The snippet of code defines the creation of the scoreboard and as the *gamescorelist* variable references the a list (li) element or scoreboard entry then if the new entry is defined (i.e. *typeof(st) != "undefined"* is true) then return the given value of the variable st to the scoreboard (i.e. gamescorelist.innerHTML), which is the new user or player otherwise return a value of "--" which represents a non-entry. The *highScoreList* variable which updates the ordered list with the *high-scores* id by referencing the section element with the game id so as to append the child li tags with the string values stored in the array.

5.2.6.2 Updating the Scoreboard

When updating the scoreboard Local storage is accessed from within the *countdown()* function in *timer.js* to store a players final score and username when a game has finished. This javascript code is implemented when the countdown reaches zero (i.e. when the *count* variable decrements to 0). Scores and usernames are only stored in Local storage if the player has achieved a score of 10 or greater. If a player has not scored then Local storage is not accessed or stored to. A snippet of the code from the *countdown()* function can be seen in the following:

```
storedEntries = JSON.parse(localStorage.getItem("localstoragekey"));
if(storedEntries == null) {
storedEntries = [];
}
storedEntries.push(currentScore+" "+username);
storedEntries = storedEntries.sort(function(a,b){return parseInt(b)-parseInt(a)});
localStorage.setItem("localstoragekey", JSON.stringify(storedEntries));
storedEntries.length=10;
```

The code states that when a game has finished and a player has 10 points or more scored then local storage is retrieved by the key named *localstoragekey* using the *getItem()* function. This is parsed to a JSON string named *storedEntries* using the *JSON.parse()* function. Next the code addresses the issue of what would happen if the array is empty. If it is empty then *storedEntries* is declared as an array *storedEntries[]*. Whether the array is empty or not the *currentScore* which is an integer and the username which is a string are appended to the array using the Javascript *push()* function. Next the *sort()* function is implemented and this converts the values to integers

with ‘a’ coming before ‘b’. After this the Local storage object with the key *localstoragekey* is set or updated with the new array *storedEntries* which has been converted to JSON string or stringified using the *JSON.stringify()* function. Finally the array has a limit to its size set at 10.

The scoreboard can now be updated with the highest scores sorted to appear in ascending order. A demonstration of its functionality can be seen in Fig. 40.

Fig. 40 The score, scoreboard and reset buttons.

The images in Fig. 40 illustrate two side-by-side instances of the High scores scoreboard. On the left is an image of an in-game instance for a player named ‘Ger’. Above the scoreboard is the current score for the player while in a game session. Below is a list of the recorded high scores which are currently set or stored in Local storage. As can be seen from the list, ‘John’ is at the top of the list, in first place with a score of 150 points and ‘Mike’ is at the bottom in tenth place with a score 50 points. The image on the right displays the scoreboard for when the game is over for the player named ‘Ger’. It shows how the final score for ‘Ger’ has been inserted into the scoreboard at the correct position of second place with a score of 140 points behind ‘John’ who is in first place. The scores for all the players below ‘Ger’ have now dropped down one position and the player named ‘Mike’ has dropped out of the scoreboard with ‘Janis’ now being positioned in tenth place with a new lowest score of 70 points.

The full user-interface for the desktop web application can be seen in Fig. 41 & 42:

Fig. 41 The Welcome screen

Fig. 42 The interface in during a game session.

5.2.7 The Mobile Page

MathMagic Basketball also contains a Mobile webpage for providing information about, and access to, the game's mobile application. This page contains none of the games core functionality and the design of the page focuses on the center panel which provides the ability to download the game's Android app straight to a mobile phone via a Q-code image. A warning note is also given on the right panel to inform users that some security settings may need to be altered to download apps from sources other than the Google play store. The center panel also provide the users with the ability to download the app in the format of an apk file for the purpose of self-installation.

This page maintains the applications main style and design and it also features a list of links to other popular mathematical, educational websites on the left panel.

The Mobile page is illustrated in Fig. 43:

Fig. 43 The Mobile page.

5.2.8 The Help Page & the Geolocation API

A Help page is provided within the application for step-by-step instructions for playing the game. Like with the Mobile page this page contains none of the games core functionality while

maintaining the same overall design and style of the game. Along with the instructions a panel is provided that notifies the user of their location pinpointed on Google maps. This is provided with the implementation of HTML5's Geolocation API and Goole maps API.

The Help page is illustrated in Fig. 44:

Fig. 44 The Help page.

The HTML5 code for the implementation of the Google Map API is shown in the following snippet:

```
<div class="help_center_panel">
        <p id="msg"></p> <p id="map" class="myinfo"></p>
</div> <!-- end center_panel -->
```

A snippet of the Help page's internal Javascript code for the implementation of the Geolocation API can be seen in the following:

```
document.getElementById("timespanUpper").innerHTML ="";
document.getElementById("timespanLower").innerHTML ="";
document.getElementById("game").innerHTML="";
$('#game').removeAttr('id');                //jQuery was the best option here

if (navigator.geolocation)
{
```

```
document.getElementById('msg').innerHTML = "Geolocation service is trying to find you...";
navigator.geolocation.getCurrentPosition(successFunction, errorFunction);
}

else { document.getElementById('msg').innerHTML = "Your browser does not support
Geolocation service"; }

function errorFunction(position)
{
document.getElementById('msg').innerHTML = "Geolocation cannot find you at this time";
}

function successFunction(position)
{
var lat = position.coords.latitude;
var lng = position.coords.longitude;

document.getElementById('msg').innerHTML = "Found you at...<br>Latitude:
"+lat.toFixed(2)+", Longitude: "+lng.toFixed(2);

var lating = new google.maps.LatLng(lat, lng);
var options = { zoom: 18, center:lating, mapTypeId: google.maps.MapTypeId.ROADMAP };
var map = new google.maps.Map(document.getElementById('map'), options);
var map = new google.maps.Marker( {position: lating, map: map, title:"You Are Here"} );
```

To correctly implement the Google API, It must be sourced from within the Help page document's header with the following script:

```
<script src="http://maps.google.com/maps/api/js?sensor=false"></script>
```

5.3 The Mobile Application

The mobile application had a two stranded approach to its development which aimed to investigate the extent and limitations of HTML5-built apps in the mobile environment. One approach was to develop a MathMagic Basketball app as a pure HTML5 web application hosted on, and accessed through a web server. The other approach was to develop the same application wrapped as a client-side native app built with the aid of PhoneGap, the framework that creates apps from HTML5, CSS3 and Javascript with web APIs for various platforms such as Android.

For this project's mobile implementation, the Android platform was chosen for the PhoneGap development process.

5.3.1 The ClassList API

The development of the mobile application followed the same theme and style of design as its desktop counterpart. Some subtle differences however do separate the desktop and mobile applications. One of these differences is the dynamic features of the mobile implementation. Dynamism is one of the main characteristics of the mobile app. This manifests itself with the entire application comprising of one HTML5 web page yet when the game is running it appears that there are three separate screens within the application. This was basically made possible with the implementation of HTML5's classList API which allows for the manipulation of HTML element classes.

Previous methods of accessing and changing classes included the use of the className property. With the className property an elements class or group of classes took the form of a string. Looking at an example of the HTML code for the first basketball with the id of bball1 with the three classes' *number_font bballStyle* and *hover*.

```
<div id="bball1" class="number_font bballStyle hover"></div>
```

When using the className property to remove one of the classes, for example *bballStyle*, the three classes are retrieved as a string which then is required to be split or tokenised into three individual classes which is then looped through to find the position of the *bballStyle* class that requires removing. When the *bballStyle* class is found and removed a string is then constructed from the remaining classes, *number_font* and *hover*. This process to simply remove a class involves a large block of code and this applies also to when a class is replaced and a similar process is implemented when adding a class.

With the introduction of HTML5's classList property the process stated in the previous example is simplified and there is no need for such a large block of code involving the process outlined in the example. The removal of the bballStyle can simply be achieved with the following line:

```
document.getElementById("bball1").classList.remove('bballStyle');
```

The methods associated with the classList API are:

- **add(value)** – adds a class to the element. If it already exists then it will not be added.
- **contains(value)** – checks if a class already exists. Returning true or false.
- **remove(value)** – removes a class from an element.
- **toggle(value)** – if the class already exists then it is removed if not then it is added.

The incorporation of the classList API made the addition and removal of the apps content a much more simple process enabling the content to change dynamically as the user switches from screen-to-screen. In fact it made the application operate with such efficiency that I then implemented the classList property in the development of the desktop application wherever it was required. This also proved to be beneficial to that version of the application.

The game also incorporate another form of dynamism within the app. This comes in the form of CSS3 transitions. The implementation of transitions such as shake and hover enable elements to shake when a shake class is applied and hover when a hover class is applied. The hover transition enables a basketball to hover up and down when it is placed into the net.

5.3.2 The Mobile User-interface

An example of the mobile app's initial welcome screen can be seen in Fig. 45. As can be seen the page resembles the desktop application. An example of how the content changes from screen-to-screen is with the scoreboard, which occupies its own screen, this can be accessed only from the Welcome page as there is no need to access it during a game or from anywhere else. When a game ends the scoreboard screen will appear. The Welcome page provides the user with basic information about the game and offers three difficulty level button options. Any of which starts the game. The MathMagic Basketball title logo remains basically the same as the desktop app and is also constructed with the Canvas API. One major difference here is the font and this difference in font applies to the entire mobile app. A free font package was obtained from the Font Squirrel website named 'Action Man' and this was added to the application by way of linking to a CSS stylesheet in the HTML header [28]. This font was then applied wherever required throughout the application.

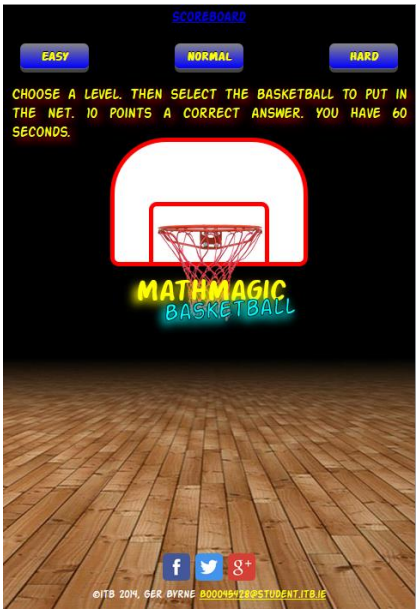

Fig. 45 The mobile Welcome screen.

When a difficulty level is selected the game begins by switching to the Game screen this can be viewed in Fig.46. This shows the initial Game screen which waits for a user input which answers the displayed equation. The user inputs an answer by tapping on their desired basketball marked with their answer and this selection is placed in the basketball net. Numerous attempts were made to implement Drag and drop for this selection process but these were unsuccessful. Drag and drop is renowned for its difficulty when implementing in the mobile environment. It is not supported by any of the mobile browsers except Internet Explorer mobile 10.

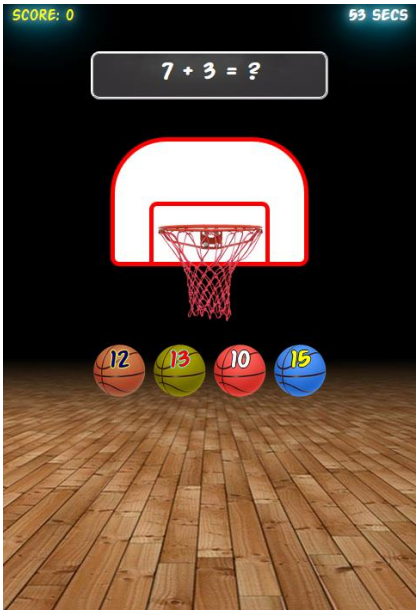

Fig. 46 The mobile interface in a game session.

The current score for the user and the countdown timer are dynamically displayed on the game page and here only. The current score, the countdown timer and the correct/wrong answer notifications (see Fig. 47) were implemented with similar code to that which was used in the development of the desktop app but with some alterations. Work on these took a lot more time than was originally expected but the desired result was achieved and they now function appropriately in the mobile environment.

Fig. 47The in-game notifications

The last screen in the mobile app is that of the scoreboard (see Fig. 48). This is also coded dynamically and appears automatically when the timer reaches zero and a game ends.

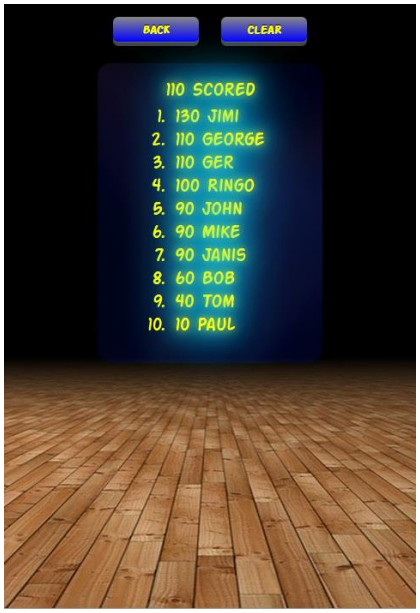

Fig. 48 The mobile scoreboard and reset buttons.

The content for the scoreboard screen was constructed in much the same way as the desktop application. The scoreboard is created and updated with the implementation of Local storage and the code for this is identical to that of the desktop application with no variations. What did vary here was the display of the user's score. This displays a user's last achieved score above the scoreboard when a game ends. This is a totally different implementation of the score from that which appears at the top left of the Game screen. The Game screen score and the score in the desktop application reset when a game ends but as this scoreboard screen automatically appears when a game ends it was felt that the score above the scoreboard in (see Fig. 48) should display the score for the last user that played the game.

Finally on the scoreboard screen two buttons are center positioned at the top of the display, one labelled 'Back' and one labelled 'Clear'. The 'Back' button is for to return to the Welcome screen and the 'Clear' button is for the purpose of clearing the scoreboard which in fact implements the Local storage *clear()* method in exactly the same way as the desktop app.

The main difficulty with the development and implementation of the mobile app was with that of the design and styling of the interface. Success here was ultimately made possible with the use of appropriate CSS and CSS3 properties and attributes. The HTML5 classList API also played an important role in this outcome. Although staying true to the design of the desktop app was important some changes were essential to convert the games functionality to the mobile arena. As mentioned previously, one such change was with the Drag and drop implementation. This had to be completely revamped to suit the new environment.

5.3.3 Polyfills, Shims and Fallbacks

Most issues with HTML5 and Javascript web applications can be overcome with the implementation of polyfills, fallbacks or shims. Polyfills are Javascript implementations that aid HTML5 and CSS3 web app functionality in unsupportive browsers. They can reproduce the functionality of HTML5 APIs and CSS3 techniques on previous versions of browsers. Polyfill compromises are also known as HTML5 shims and HTML5 shivs or browser fallbacks. Originally the purpose of these workarounds was to implement modern web apps in previous versions of browsers but they have recently become much more prevalent working in tandem with HTML5 APIs.

5.3.3.1 Modernizr

Modernizr [29] is a feature detection implementation that provide a Javascript library to test a browser when a web application is loaded for compatibility issues. The features it tests for are implementations of new APIs and techniques provided with new HTML5 and CSS3 implementations. To implement Modernizr a script is downloaded, added to the app and referenced in the header of the index.html file. This script comes in two forms, there is a script for development purposes and one for production purposes. The development script is a more general purpose script as it may be unsure at an early development stage what features are to be implemented. The development script contains all features. The production script, on the other hand is a condensed, customised to only detect for features that have been implemented into the web app as these should be known once the app goes into production. In addition to the link place within the header element of the index file Javascript code was added for the purpose of testing browser compatibility for the Drag and drop API and the Local storage API. A snippet of this can be seen in the following:

```
<script>
   if(!Modernizr.draganddrop){
     alert("Sorry, This browser does not support drag and drop!");
   }

   if (!Modernizr.localstorage) {
     alert("HTML5 Local storage is not available!");
   }
</script>
```

Modernizr was used in this project for both the desktop app and the mobile app. However the script depicted in the code snippet is embedded in the desktop web application. In the mobile app the test for Drag and drop functionality was omitted as it is already known that it is not supported by mobile browsers except for IE 10 on the Windows platform [30].

As can be seen from the *if* statements in the code snippet. Modernizr can also be used to provide the ability to load polyfills or shims if browser compatibility has failed, hence the name Modernizr. It also can achieve this with a resource handler called Modernizr.load that can implement polyfills if it finds an API that is unsupported.

5.3.3.2 FastClick

FastClick is a Javascript, click handler library whose sole aim is to improve the responsiveness of web applications with in mobile browsers. In this environment click events may appear to take too long to register. This is because click events within mobile web apps have approximately a 300 millisecond time delay from when a button is tapped to when the event is handled. This default delay takes place so as to wait and see if a user is trying to double tap. There are a number of polyfills available to combat this time delay but the easiest to implement is fastClick. This polyfill was created and developed by FT Labs which is part of the Financial Times [31]. It was incorporated into the mobile application with the implementation of the following script into the index files header element:

```
<script type='application/javascript' src='js/fastclick.js'></script>
```

FastClick was then instantiated on the mobile web apps body with the use of the following line of code and once applied its improved effects were immediately noticeable.

```
window.addEventListener('load', function() { FastClick.attach(document.body);  }, false);
```

The 300 millisecond time delay had one more side effect. When the ‘Clear’ button was selected on the scoreboard screen, it was then wrapped in an orange border for 300ms. This is a known issue and fortunately a fix or workaround is also known for this. The fix was applied to the apps stylesheet and the orange border was eradicated and the issue was corrected with the following CSS webkit code:

```
* {       -webkit-tap-highlight-color: rgba(255, 255, 255, 0);       }
```

5.3.4 Drag and Drop on Mobile

As much as the implementation of the Drag and drop API was problematic and troublesome in the development of the desktop application this was nothing when compared to its application within the mobile environment. As it is not supported by any of the main mobile browsers except for IE 10 and its conflict with mobile default touch gestures the only way it can be incorporated into an app is with the addition of a Javascript library or framework dedicated to the task. There are a number of these dedicated libraries available for the purpose of Drag and drop and for this project two were chosen to aid in its implementation in the mobile app. These were Hammer.js [32] and jQuery Touch Punch [33].

5.3.4.1 Hammer.js

Hammer.js incorporates Drag and drop and other multi-touch gestures into a mobile applications functionality with the addition of a few lines of javascript code and the inclusion of dedicated library scripts. While implementing this code I found that the process began to get more and more complex as the implementation progressed. When it was eventually implemented, albeit only slightly in a haphazard way, the Drag gesture had minimal effect. It did not seem best suited for interaction over client-server communication. The worst thing, however that I found about Hammer.js was its support for the library. There seemed to be little or no online support for the product and although it had what it called a wiki this was very limited and below standard.

5.3.4.2 Touch Punch

Touch Punch is a jQuery hack that recognises touch events on mobile devices and maps them to their equivalent mouse events. The implementation of the Touch Punch Polyfill went a lot more smoothly than that of Hammer.js and this was with the addition of less complex code. The Touch Punch script was added to the mobile apps index page and as it was a jQuery hack, a number of jQuery minified scripts were also required. There was much better support available for Touch Punch with the availability of a simple set of instructions which can be found on its dedicated website and a well-supported community base.

The results of the Touch Punch implementation was not that much better than that of Hammer.js. When dragging any of the basketballs from the source area to the dropzone they appeared to move sluggishly and in some instances would not move at all. They seemed to adopt a position mid-drag and lock there. A screenshot of the first basketball locking in the Firefox browser with jQuery's Touch Punch implemented on an Android device can be viewed in Fig. 49.

Fig. 49 A ball being dragged with Touch Punch

5.3.5 PhoneGap and Hosting

Once the web application was fully developed a mobile version, constructed with the aid of Adobes PhoneGap was sought. PhoneGap is a free open-source distribution of Apache Cordova and is powered by its engine [34]. Cordova is a framework for mobile development which wraps HTML5, CSS3 and Javascript into native mobile applications. It does this for cross-platform development while at the same time it does not utilize these platform's languages. A major advantage to PhoneGap is that it can blur the distinction between an applications developed for the web and native apps. There are a number of ways to use PhoneGap in the development process. The framework can be downloaded to a computer and implemented with the Eclipse Integrated development environment and the Android SDK or it can be utilised from the Adobe PhoneGap build which packages applications wrapped as native apps in Adobe's creative cloud service.

As the MathMagic Basketball mobile app was now already developed the Adobe PhoneGap build was deemed to be the best method for the creation of the native app. To construct the app a short sign up process was completed to obtain a free account. This provides the service for the development of only one app and if more are required then a fee is imposed. With the account now open the next task was to upload the already developed HTML5, CSS3 and Javascript code to the service. After this upload was completed and a very short period of time had passed the application was ready in the form of an Android .apk file format. The mobile web application of MathMagic Basketball was now ready for installation as a native app on any Android device.

A number of installations with corrections made to the development were needed to finally achieve a satisfactory result with the app. This was made easy with the PhoneGap cloud service as new revisions could be uploaded at any time overwriting previous versions of the build.

Another benefit to using the PhoneGap build cloud service was that it provided the ability to easily share the application with links to download and a Q-code image for download which was incorporated into the desktop application on the Mobile web page.

With the client-side mobile native application built ready and available for installation the last task required to complete the projects implementation was to host the desktop and mobile HTML5 web applications. This was a straightforward process and the hosting service was provided by x10hosting. X10hosting host web applications on its Apache servers and among its tariffs there is basic free hosting service and a premium service. The basic service was availed of for this project's implementation.

I have used x10hosting services before and was aware of all that was required to successfully host the two applications. Two separate accounts were opened one for the desktop application and one for the mobile application. While working with x10hosting any alterations that were required with the two forms of application were done so, by means of a desktop FTP client. This process is made easy by the x10hosting service and it is one of the reasons for which I use it.

Now with implementation successfully completed, all versions of MathMagic Basketball were ready for testing in a variety of environments.

6 Testing

6.1 Testing the Desktop Application.

6.1.1 Testing the Math Logic

In the initial phase of testing the desktop web application, first on the agenda was the testing of the math logic to see if everything concerned was in proper working order. To achieve this, the *randomBall* variable was displayed to screen along with each equation. This variable is implemented in the mathLogic.js script and its purpose is to position the basketball with the result of an equation. This is dealt with in more detail in section 5.2.2 Mathematical Logic. While testing *randomBall* identified the position of the correct answer and if the basketball which occupied this position was flagged as correct when placed in the net then the mathematical logic of the game was working correctly. An example of the randomBall variable displayed for testing purposes can be seen in Fig. 50.

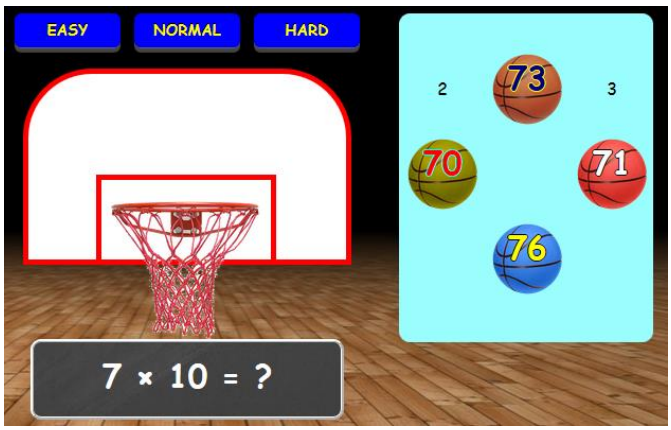

Fig. 50 The output of the app while under testing.

In Fig. 50. The number 2 represents the randomBall output. This correctly identifies that the answer is in the second basketball at position 2. This is marked 70 and this is correct.

The number 3 on the right of the first basketball (marked 73) is an output of the *rand* variable which is implemented in the *correctAnswer()* function within the bballfunctions script (see section 5.2.1 The Drag and Drop API). The *rand* variable identifies which equation is currently being displayed on screen i.e. 1 for addition, 2 for subtraction, 3 for multiplication and 4 for division.

This was also displayed onscreen for the purpose of testing and when used in conjunction with the randomBall variable they were both very useful for this purpose. The use of these variables meant that the game could be tested multiple times at a fast pace and this was done to any errors reported. This form of testing was also implemented for the mobile application and no errors were reported there either.

6.1.2 Browser Compatibility

The next stage in the testing phase involved checking the application for cross-browser compatibility. This was to ensure that the application could be accessed by a user from within any of the main leading browsers, Mozilla Firefox, Google Chrome, Microsoft Internet Explorer and the Apple Safari browser. For this purpose a list made up of API's, functions and other techniques were examined. These were given a rating from 1 to 10 based on their performance and compatability. These ratings were derived from my own personal experience gained from having developed and worked with the app for the last 28 weeks or so.

	Mozilla Firefox	Google Chrome	Internet Explorer	Apple Safari
Drag & Drop	10	8	4	0
Local Storage	10	10	10	0
Canvas	10	10	10	10
ClassList	10	10	10	0
Geolocation	3	10	3	0
Timer	10	10	10	10
CSS Transitions	6	10	4	10

Table 5 Browser Compatibility Table

The Drag and drop API performed flawlessly in Firefox. When implemented on other browsers such as Chrome it performed reasonably well although it was expected to perform much better. In Internet Explorer it performed badly sometimes there would be an issue with the selection of a basketball for dragging and this would freeze or lock albeit for a couple of seconds but it still locked. Apple's Safari browser for the Windows platform was a disaster. Safari has very poor HTML5 support and I can safely say that Drag and drop does not work on Safari for Windows.

As this game app rely heavily on the Drag and drop API to function without some of the other APIs could not be tested. Another issue that was discovered with all browsers except Firefox was that dragging a transparent item with a transparent background would somehow drag with an imprint of a div block background.

Local storage was implemented successfully on all browsers although it could not be tested on Safari. The Canvas text appeared well and was compatible with all browsers including Safari. The ClassList work on all also although it could not be tested on Safari. Geolocation worked perfectly on Google Chrome but had issues with the other browsers where sometimes it just would not load. The Timer was fully functional with all browsers including Safari. And Lastly the CSS transitions were temperamental when it came to which browser that they worked. They worked flawlessly when implemented on Google Chrome and Safari but only an estimated 60% of their implementation was successful on Firefox and 40% was successful on Internet Explorer.

6.1.3 Adaptability

The desktop web application was constructed to be adaptable to work on any computer regardless of their screen resolution. After careful CSS development for this purpose the web app was tested to work on a variety of resolutions including Laptop and large screen desktop monitors. Some alterations needed to be made to the CSS stylesheet but when this was completed the web application was fully adaptable.

6.2 Testing the Mobile Application.

In the mobile environment the game functionality followed the same analysis with the same methods as in the desktop environment and it concluded without any errors reported. The testing of the application differed from the desktop approach in other areas. Although browser compatibility was an important consideration the testing also involved the mobile web app being pitted against the native app built with PhoneGap. As the platform being tested on here is Android, the app was tested to run in Android's default browser and it was not tested on Internet Explorer's mobile browser or Apple' Safari as these are not available for Android.

	Web App			Native App
	Mozilla Firefox	Google Chrome	Android Browser	PhoneGap App
Drag & Drop	0	0	0	0
Hammer.js	3	3	3	3
Touch Punch	6	6	6	6
Local Storage	10	10	10	10
Canvas	10	10	10	10
ClassList	10	10	10	10
Timer	3	3	10	10
CSS Transitions	3	3	6	10

Table 6 Mobile app performance table

HTML5's drag and drop is the worst performer when it comes to the mobile environment not just because it's one implementation is not supported by any of the mobile browsers but even with the inclusion of various polyfills such as Hammer.js and Touch Punch it is a nightmare to work with. There was no problems discovered when it came to implementing the Local storage, Canvas and ClassList APIs on any browser or even on the PhoneGap built native app. The Timer seemed to have some issues in Firefox and Chrome where on occasion it would count down very slowly. This was not the case when the app was implemented in the Android browser or in the PhoneGap native app. The CSS transitions incorporated into the mobile app worked in much the same way as they did on the desktop app. In some browsers they had full-functionality while in others only worked 30-60% of the time.

The results of the mobile testing were surprising the hosted HTML5 app performed equally as well as the PhoneGap with both applications being very responsive. When comparing browsers the Android default browser was most efficient with the HTML5 hosted app outperforming Firefox and Chrome.

7 Conclusion

On a personal level I find it bizarre that HTML5 and CSS3 are being offered as new technologies, marketed as being ready to compete with native platforms but when implemented cannot seem to function without numerous polyfills or shims. The addition of these polyfills has previously always been the way, providing workarounds or fixes for such implementations as Drag and drop. I would prefer to have these polyfills as an option and not a necessity. As can be seen from the mobile application loading a web application with multiple scripts may eventually get some desired implementations to work but it must come at a cost. It does not seem an efficient way to develop applications and it must impact on a user's experience. When, as in this case more scripts were added for the Hammer or Touch Punch implementations the mobile application appeared to become slow and more sluggish. Maybe there was something wrong or lacking in the development process to cause this but the last thing that was needed was another polyfill to combat or overcome the shortcomings of other polyfills. Although I am being slightly sarcastic here this does seem to be where web app development seems to be going as it continuously strives to equal or even outperform mobile native applications.

Implementing drag and drop with either Hammer.js or Touch and Punch was not an impossible task but it sure seem laborious. In normal circumstances a developer would surely be tempted to implement a scenario where tapping buttons could achieve a similar result as a drag and drop implementation with far less of a struggle in much less time and with the incorporation of less scripts leaving less room for errors with Javascript conflicts.

When it comes to interactive and dynamic, desktop web apps there fundamentally seems to be no real major issues with HTML5 and CSS3. Most of the available APIs work flawlessly bar one or two which suffer from a few well-known issues. These issues however can be overcome with the implementation of added packaged javascript code in the form of frameworks or techniques such as Modernizer and jQuery.

As HTML5 is seen as an alternative to native applications in the mobile arena this comparison has also being called into question. This project found that apps developed in pure HTML5 were sluggish and unresponsive when hosted and accessed on a server by a mobile device. In comparison they performed much more efficiently when they were incorporated into a framework such as PhoneGap. PhoneGap essentially converted these apps into native apps which effectively lessens the argument that HTML5 is a better performing alternative. For improved functionality and performance in mobile devices the addition of numerous frameworks is almost a certainty.

It is my believe that some new HTML5 APIs and CSS3 technologies will not fully function without the aid of one or more of the many, now available polyfills. How much this is the case and whether or not polyfills are needed for a fully functioned web app will indicate the true state of interactivity and dynamism with HTML5 and CSS3 was a viable alternative to both mobile and desktop native applications. This remains to be seen when these web app technologies are finally, officially standardised by the W3C, as expected later this year in 2014.

8 Appendix – Project Diary

Semester 1

September 2013

Week 4
Discussed and considered proposed project topic with lecturers in charge.
Briefly researched ideas in the area of web development.

October 2013

Week 1
Researched Project ideas.

Week 2
Sought out possible project supervisor and discussed some of his ideas.
Luke Raeside agreed to be the project supervisor with a variation of one of his ideas.
Begun work on the project plan with the aim to providing a clearer view as to what work lied ahead.

Week 3
Started a diary and outlined a plan for the project.
Second meeting with supervisor. Started to work on the project proposal.

Week 4
Started to seek out relevant research material relating to web interactivity and dynamism.

November 2013

Week 1
Gathered research ideas and began the research.
Began work on the Literature review.

Week 2
Research now fully focused on web interactivity with a lean towards HTML5

Week 3
Begun the development of a prototype web-based game app.
This involved work with HTML5, CSS3 and JavaScript.

Week 4
Looking to incorporate HTML5 APISs into the web app to showcase interactivity.
Seeking to implement Drag and drop and Local storage into the web app.

December 2013

Week 1
Continue to construct an adequate web app in relation to the projects topic. Fully focused now on an interactive Maths game app called MathMagic.

Week 2
Plan and prepare for the projects prototype presentation.

Week 3
Continued to work on the projects prototype presentation.
Presented a working prototype of the application.

Semester 2

January 2014

Week 3
Started to develop the MathMagic app.

Week 4

Change the idea for the web app and it is now MathMagic Basketball

February 2014

Week 1

Completed the drag and drop implementation of the game.

Week 2

Research the possibility of constructing a basketball net within the app.
Begin implementation of Local storage to construct a scoreboard.

Week 3

Continued working with the Local storage API. This implementation is taking longer than expected.

Week 4

Re-evaluated the application and decide what other APIs or technologies will in fact be implemented. More research was needed here.

March 2014

Week 1

Successfully implemented Local storage and constructed a Basketball net as part of the game for the app.

Week 2

Begun the development of the mobile app.

Week 3

Completed the first draft of a fully functional desktop application.

Having trouble implementing HTML5's Drag and drop in the mobile app even with Polyfills

Began to prepare the final draft of the system analysis and design documentation.

April 2014

Week 2

Still having trouble with mobile drag and drop. Worried that it might impact on other work.

Continued work on the thesis. System Analysis and Implementation sections.

Week 3

Abandoned drag and drop in mobile. Now working on tap gestures.

The web application is now fully functional as a completed working model.

Created the first version of the PhoneGap app

Week 4

Finished the first draft of the mobile web application.

Organised the final build of the applications.

Corrected any outstanding issues with the HTML5 and CSS and Javascript.

Run final tests.

Check for any errors and issues.

May 2014

Week 1

Presented the final build of the application.

Completed work on the Thesis documentation.

Week 2

Completed and delivered the Thesis.

9 References

[1]. Cisco. (2013). Cisco Visual Networking Index: Global Mobile Data. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.pdf Last accessed 6th Oct 2013.

[2]. Kaelin Mark. (2013). Get ready for your upgrade to Windows 8.1. Available: http://www.techrepublic.com/blog/windows-and-office/get-ready-for-your-upgrade-to-windows-81/ Last accessed 5th Oct 2013.

[3]. Catuhe David, Rousset David. (2013). *WebGL Hello World in IE11*. Available: http://devhammer.net/webgl-hello-world-in-ie11 Last accessed 5th Oct 2013.

[4]. Jazayeri, Mehdi. "Some trends in web application development." Future of Software Engineering, 2007. FOSE'07. IEEE, 2007.

[5]. Jackson, Dean. "WebGL specification." Khronos WebGL Working Group (2011).

[6]. Hickson, Ian. (2013). W3C's Web Storage. Available: http://www.w3.org/TR/webstorage/. Last accessed 6th Nov 2013.

[7]. Trygve Reenskaug. Models - views - controllers. Technical report, Technical Note, Xerox Parc, 1979.

[8]. Oracle. (2012). Your First Cup: An Introduction to the Java EE Platform. Available: http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html. Last accessed 20th Oct 2013.

[9]. Hibernate. (2011). What is Object/Relational Mapping?. Available: http://www.hibernate.org/about/orm. Last accessed 21th Oct 2013.

[10]. Wilson, Chris. (2007). You, me and the W3C. Available: http://blogs.msdn.com/b/cwilso/archive/2007/01/10/you-me-and-the-w3c-aka-reinventing-html.aspx. Last accessed 6th Nov 2013.

[11]. W3C. About W3C. Available: http://www.w3.org/Consortium/. Last accessed 5th Nov 2013.

[12]. Laine, Markku, Denis Shestakov, Evgenia Litvinova, and Petri Vuorimaa. "Toward Unified Web Application Development." IT Professional 13, no. 5 (2011): 30-36.

[13]. Laine, Markku Pekka Mikael. "XFormsDB-An XForms-Based Framework for Simplifying Web Application Development." (2010).

[14]. Google. (2007). GWT. Available: http://www.gwtproject.org/?csw=1. Last accessed 19th Sept 2013.

[15]. Gavin King, Christian Bauer, Max Rydahl Andersen, Emmanuel Bernard, Steve Ebersole, and Hardy Ferentschik. (2004). HIBERNATE - Relational Persistence for Idiomatic Java. Available: http://docs.jboss.org/hibernate/stable/core/manual/en-US/html/. Last accessed 4th Nov 2013.

[16]. Wayner, Peter. (2011). 11 hard truths about HTML5. Available: http://www.infoworld.com/d/html5/11-hard-truths-about-html5-169665. Last accessed 1st Nov 2013.

[17]. Huy, Ngu Phuc. "Developing Apps for Mobile Phones."

[18]. Popescu, Andrei . (2012). Geolocation API Specification. Available: http://dev.w3.org/geo/api/spec-source.html. Last accessed 15th Oct 2013.

[19]. Dahlström, Erik et al. (2011). Scalable Vector Graphics (SVG) 1.1 (Second Edition). Available: http://www.w3.org/TR/SVG/. Last accessed 15th Oct 2013.

[20]. W3C. (2013). Audio and Video. Available: http://www.w3.org/standards/webdesign/audiovideo. Last accessed 15th Oct 2013.

[21]. Rik Cabanier, Eliot Graff, Jay Munro, Tom Wiltzius, Ian Hickson, . (2013). HTML Canvas 2D Context. Available: http://www.w3.org/TR/2dcontext/. Last accessed 16th Oct 2013.

[22]. Abowd, Gregory D., et al. "Towards a better understanding of context and context-awareness." Handheld and ubiquitous computing. Springer Berlin Heidelberg, 1999.

[23]. Hickson Ian. (2012). Web Workers. Available: http://www.w3.org/TR/workers/. Last accessed 16th Oct 2013.

[24]. PhoneGap. PhoneGap Documentation. Available: http://docs.phonegap.com/en/3.1.0/index.html. Last accessed 5th Nov 2013.

[25]. Sin, David, Erin Lawson, and Krishnan Kannoorpatti. "Mobile Web Apps-The Non-programmer's Alternative to Native Applications." Human System Interactions (HSI), 2012 5th International Conference on. IEEE, 2012.

[26]. JQuery. JQuery Mobile. Available: http://jquerymobile.com/. Last accessed 4th Nov 2013.

[27]. Sencha. (). Sencha Touch Build Mobile Web Apps with HTML5. Available: http://www.sencha.com/products/touch/. Last accessed 5th Nov 2013.

[28]. *The Font Squirrel API.* Available: http://www.fontsquirrel.com/blog/2010/12/the-font-squirrel-api. Last accessed 11th March 2014.

[29]. Author Unknown. (2009-13). *Modernizr.* Available: http://modernizr.com/docs/. Last accessed 22 Feb 2014.

[30]. Author Uknown. (2014). *Can I use... Drag and drop.* Available: http://caniuse.com/#feat=dragndrop. Last accessed 22th March 2014.

[31]. Author Uknown. (2014). *FT Fastclick.* Available: http://labs.ft.com/articles/ft-fastclick/. Last accessed 2nd April 2014.

[32]. Unknown. (2014). *Hammer.js.* Available: https://github.com/EightMedia/hammer.js/wiki. Last accessed 5th Jan 2014.

[33]. David Furfero. (2011). *jQuery UI Touch Punch.* Available: http://touchpunch.furf.com/. Last accessed 10th Feb 2014.

[34]. Brian. (2012). *PhoneGap, Cordova, and what's in a name?.* Available: http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/. Last accessed 22nd April 2014.