

Mark Innes

1008377

Mobile/Web Application for storing event information

***Submitted to Robert Gordon University for the requirements for the
Degree of Multimedia Development***

School of Computing

2013

Authorship Declaration

I, Mark Innes can confirm that this report and the implementation of this entire project is work that I have completed on my own.

Where I have quoted other pieces of work then I have provided a referenced source.

Where surveys have been completed for the purpose of research the data from surveys have been stored on Google servers. The surveys were anonymous, but the participant had to confirm that storing answers via Google was acceptable.

I have read and understood the University's procedures regarding plagiarism.

Signed:

Mark Innes

Date:

Matriculation No: 1008377

Project Supervisor: Richard Glassey

Abstract

In the past few years, the amount of content being generated and shared online has exploded. Events in particular are spikes in the activity of content generation, such as a concert, a festival or other public event.

This report concerns the development of the Web application called Tickitbox. By using the knowledge gathered during the literature review and information analysed during a study on online human behaviour, a web application was designed and implemented that created a service for a user to have digital relationships with events they have attended by using Tickitbox.

The applications concepts were fully tested and proven to be bug-free before evaluations were conducted on the resulting application with positive results.

Tickitbox reads the information from the image of a ticket and returns data about the event based on the artist venue and date of the event.

Table of Contents

1	Introduction	6
1.1	Motivation for the project	6
1.2	Scope	7
1.3	Objectives for the project	7
1.4	Meeting these objectives	7
1.5	A Start	9
2	Literature Review	10
2.1	Web Applications, a brief history	10
2.2	Human factors, the digital world we live in	22
2.3	Web Applications	27
3	Problem and Data Analysis	36
3.1	Problem overview	36
3.2	Research survey	37
3.3	Survey Findings	39
3.4	Survey Conclusions	40
3.5	Survey Recommendations	41
3.6	Application Requirements	42
3.7	Summary	45
4	Application Design	46
4.1	System	46
4.2	Data Modelling	52
4.3	Interface	57
4.4	Technologies	61
4.5	Moving On	69
5	Implementation of the Application	70
5.1	Creating an event	70
5.2	Viewing an event	83
5.3	User Management	90
5.4	Application functionality	95
5.5	User interface design	99
5.6	Security	101
5.7	In summary	103

6	Testing and Evaluation	104
6.1	Testing.....	104
6.2	User Evaluation	110
6.3	Performance Evaluation.....	114
6.4	Quality of the application	120
6.5	Evaluation Summary	121
7	Conclusions and reflections	123
7.1	Project contributions	123
7.2	Conclusions	125
7.3	Further work	127
7.4	Personal Reflections.....	129
8	References	131
	Appendix A: Survey Questions.....	135
	Appendix B: Detailed Survey results	138
	Appendix C: Page recommendations from Google Chrome Page Speed	150
	Appendix D: Google Chrome Audit tab recommendations	152
	Appendix E: Full list of HTML Errors by page	154

1 Introduction

In the past few years, the amount of content being generated and shared online has exploded. Events in particular are spikes in the activity of content generation, such as a concert, a festival or other public event. The content is then uploaded to many generators and is often lost amongst a sea of unfiltered content or content from the same event has no way of being matched with another.

The purpose of this project is to create a system/service that allows users to interact with events they have been to by capturing information on a ticket. The application will create a multimedia experience for the user using the application programming interfaces for various content generators to bring the content into a single application.

1.1 Motivation for the project

I have collected and kept tickets for events I have been to since as long as I can remember. I would keep these in an old shoe box hidden away, which I would call my ticket box. The ticket box has tickets from the first football, concert and festival I attended. It also has tickets from events that have been some of the best moments of my life so far.

I began keeping the tickets as I wanted to remember the events after it was over and to have something to represent the experience I had witnessed. The first time I went to a music festival was a defining moment in my life and I truly believe that these amazing events should be allowed to live on.

When I first started keeping event tickets I could have had no idea that the Web would be such a large part of society and now everyone has a camera phone in their pocket, content is being generated at every single event possible. I can now search YouTube for my favourite event and be reminded of that special atmosphere that existed at that particular moment. I accept that nothing could ever compare with the real thing, good or bad, the experiences of these events will live in the mind.

The project is a take on what could be very useful to a lot of people, using current skills in Web development and furthering experience with the subject. The collection of tickets is a common act and to create a service that people not be aware of now but would use in the future greatly appeals.

Furthering experience in Web Development is also a goal for this project. New technologies that are created every single day in this field are impossible to keep up with, but the project's aim is to look at recent popular methods of creating Web applications in order to implement the project.

1.2 Scope

The goal of this project is to deliver a seamless Web application that stores information on events the user has attended and uses this information to send to Media generators to return results relevant to the event the user has attended. The proposed application will be a digital representation of the physical box of tickets.

Also the project should look at new methods of implementation with emerging technologies, for both functionality and research purposes.

1.2.1 What will be the benefit?

Surveys will be carried out with members of the public to see how they feel about events, tickets, social media and web applications and will use that as a platform and evidence to create the application.

Due to exploration into newer technologies the literature review in chapter 2 will be beneficial to web developers are looking at new methods of creating applications.

1.3 Objectives for the project

The objectives that follow have been identified as necessary for the completion of the project.

1. Analyse existing online multimedia services and web development technologies;
2. Conducting a study of online social behaviour;
3. Design and implement an application that interacts with existing web applications to create a social experience.
4. Explore new technologies of Web Application Development.
5. Proper Testing and Evaluation of the application.

Upon the completion of the project, how well these objectives have been met and what has been learned as a result of the project's completion will be discussed. Future work regarding the application will be also stated.

1.4 Meeting these objectives

The objectives will be met through gathering information on all aspects of web development and human behaviour towards the digital world, by gathering information about and end user and deeper analysis of the problem. The application will be designed and implemented before testing and evaluating the final solution.

1.4.1 Literature Review

Research will be carried out on the history of Web applications and looking at what factors have been decisive in creating the desktop like programs available today. Next the research will analyse the Human factors that affect Web design and development. Thirdly, looking at web applications in general and what properties define them.

1.4.2 Problem & Data analysis

The problem will be analysed and before creating a public survey to understand further what user's experiences with web applications are like. The survey will be created using Google docs which has a feature to review the results.

Based on the survey results and the research carried out an approach to solving the problem will be created. This is also a point where functional & non-functional requirements for the application will be stipulated.

1.4.3 Application Design

The design section will look at the requirements of the application in detail and how best to meet them. This section will include architectural views, use cases and sequence diagrams.

1.4.4 Implementation of the Application

Based on the design three prototypes will be created, each with a different creation method. This will be a basic application with only a few features and an investigation on which method is most suitable for the full project will be carried out.

1.4.5 Testing and Evaluation

To test the project the same people who filled in the survey for the Problem chapter to use the application and review it will be offered the chance to use the application. The project will also be subject to web development testing with static analysis of code and unit testing.

1.4.6 Conclusions and reflections

When the web application is completed and has been tested the process will be reviewed by stating how the objectives have been met. This is where the future work required will be acknowledged as well as a personal reflection of the project.

1.5 A Start

The research will begin with the history of web applications and how they have become the powerful desktop like programs we use today.

2 Literature Review

This section will begin by looking at a history of web and web applications and then looking at the human aspect of the digital world we now live in. Next up is looking at specific web application creation methods in detail before finally reviewing the research and discuss the relevant findings.

2.1 Web Applications, a brief history

It would be impossible to acknowledge every important advance in web development that has occurred in history but this section will give a brief account of what are historical moments or technically important landmarks of a specific year that have made it possible to create the type of applications that are possible today.

The research will start with a background of how the internet works and move through the years to 2013 and the state of the web today.

2.1.1 In the beginning

The Internet as we know it today is a collection of computers that are linked together via the TCP/IP set of network protocols. TCP/IP is the abbreviation of Transmission Control Protocol / Internet Protocol and this specifies how computers can communicate data between one another (W3C, 2013). Each computer has its own unique IP address, and this connection created between two IP addresses has expanded into an amazing array of interlinked social networks, customers, clients, email, phone calls, video streams and many other uses who are all connected together via phone line all over the world, however the basic principal is still the same.

The Internet began on the orders of President Dwight D. Eisenhower, who created the Advanced Research Projects Agency (ARPA). The purpose of this agency was to gain a technological edge over the Soviet Union during the Cold War (How stuff works, 2013) who had just launched the famous satellite, Sputnik, into space.

The ARPA created a computer network called ARPANET, which was 4 computers running on 4 different operating systems. This network created the protocols still used to connect us today, and in 1983 this lead to the creation of TCP/IP.

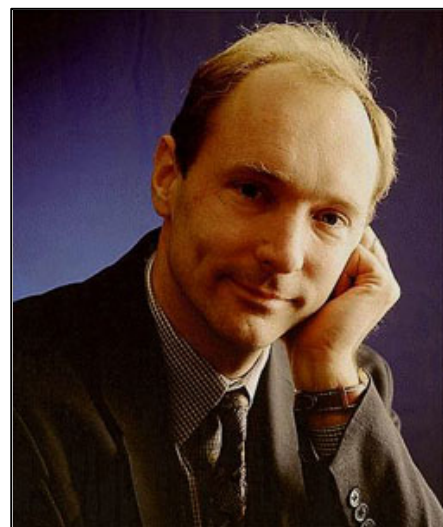


Figure 2.1 Tim Berners-Lee
(<http://cache.ohinternet.com>)

In 1989, a British computer engineer called Tim Berners-Lee, shown in Figure 2.1, invented what would go on to be known as the World Wide Web (W3C - History, 2013). While at the European Laboratory for Particle Physics (CERN), in Geneva, Switzerland he observed that particle research often involved joint work with input from all over the world. He wanted to allow this research to take place between the researchers but from faraway sites. The idea of a 'Web' was that documents sent between two computers could be linked together, and where one research paper contained information of another's, the two documents would be linked together via buttons on the screen. To do this he would create his own Hypertext system.

2.1.2 1991

HTTP

With the creation of a hypertext system, Tim Berners-Lee also created a protocol, Hypertext Transfer Protocol (HTTP). HTTP is the definition of the way that messages are formatted and sent, and also defines how web browsers and web servers should respond to the instructions they receive (Webopedia, 2013). An example of this today would be typing the address for Google (<https://www.google.co.uk/>) into the address bar of a Web browser. This sends an HTTP request to the Web server of Google. The Web server of Google then sends back the homepage of Google to the user. In 1991 however, there were no web browsers as they had not been invented yet.

2.1.3 1992

HTML

To allow the linked computers to 'speak' to each other via HTTP a text format was now required. This format was named Hypertext Mark-up Language (HTML) (W3C - History, 2013). The language is written with HTML elements which have tags with brackets around them so that a browser is aware what part of the web page each element is. An example web page is shown below:

```
<html>
  <head>
    <title>Honours project report</title>
  </head>

  <body>
    <h1>This is my report</h1>
    <p>Welcome to TICKITBOX</p>
  </body>
</html>
```

The text between the <html> tag defines the web page and the text between the <body> tag shows what would be visible in a web browser.

The 1st ever web page created Tim Berners-Lees local machine (W3 – History 2013) and simply contained information on the project of the web.

2.1.4 1993

Mosaic

HTTP and HTML both need something very important to make any sense, a web browser. The requirement of a web browser is to examine and display the HTML documents.

Mosaic was the 1st web browser to become popular near the end of 1993 (NSCA, 2013). It had a user friendly interface, recognisable icons, bookmarks and contained pictures and became popular with 1st time users who were not of the academic class of users the web. It was available to download from the creator NCSA's website in November 5,000 copies were being downloaded per month.

By the end of 1993 there were estimated to be 9.99 million web users (The Evolution of the Web, 2013).



Figure 2.2 Mosaic 1st Web Browser (<http://www.evolutionoftheweb.com>)

2.1.5 1994

Cookies

With the early Web increasing in popularity, a computer programmer called Lou Montulli took what were known as “magic cookies”, and apply the same logic to HTTP communication (BBC – Webwise, 2013). The programmer was trying to create an e-commerce website for his client, where information about the user of the site could be stored so that upon return to the site it would know who the user was and have the same information ready to be displayed. What the programmer created was data object, normally a short piece of text, which was stored in the user’s browser. This data object stored information about what the user of the site was doing so that if they left the site and came back, the data stored could be retrieved. This was an important development of the web for two reasons, functionality and security.

Cookies allow users to remain logged into websites when they revisited, and were important for e-commerce during the creation of online shopping carts. They create a greater overall experience for the end-user and make online shopping easier.

There are different types of cookies however, and there are seen as an invasion of privacy by some web users in the present day. Sophisticated cookies can track a user’s long term records and browsing history and this user information is stored and can then be sold on. New laws were brought created in the UK in May 2012 stating that ‘sites must obtain "informed consent" from visitors before saving cookies on a machine’ (BBC News – Cookies, 2013) or they can face fines of up to £500,000.

Number of estimated web users 20.38 million (The Evolution of the Web, 2013).

2.1.6 1995

1995 was an extremely important year for the Web, the invention of SSL, PHP and JavaScript which are as important today as they were upon creation.

SSL

Secure Socket Layer (SSL) allows the encryption of data between the browser of a user and the server that it is communication with (SSL Explained, 2013). This encryption allows credit card information so be sent to a website or e-commerce application safe in the knowledge that the information cannot be compromised. This security has been instrumental in allowing online retail

sales to go above £50bn in 2011 (Money | theguardian.co.uk, 2013) and without SSL online consumers would not have spent an average of £1,500 for that calendar year.

PHP

PHP (Hypertext Pre-processor) is a very important scripting language originally created in order to produce dynamic websites. Rasmus Lerdorf created the language and it became quickly popular with web developers as it can be placed into HTML documents. It has proved crucial in creating web applications for two important reasons, it can be deployed on nearly every operating system and due to the fact that it is an open source project (PHP EXX, 2013). When used in tandem with Structured Query Language (SQL), a database enabled dynamic application can be created. A typical setup would be a web page sending information to a PHP script. The PHP script will then send the information to a web server respond. This response is then sent back to the original web page.

JavaScript

The most popular client side scripting language is JavaScript. Scripts are placed into an HTML document and the script included is then sent to the browser, and it is left for the browser to execute it. Jumping to the present day, JavaScript is one of the most important technological advances in web development and the possibilities are seemingly endless. There are thousands of JavaScript libraries available which all perform functions that would have seemed impossible in the beginning. With JavaScript now being taken seriously on the server-side when creating Web Applications through Node.js, it seems limitless as to what the language can do on the web (Understand JavaScript, 2013).

2.1.7 1996

Hotmail launches

Hotmail revolutionised electronic mail in 1996 and was one of the 1st web-based email services. It was purchased in 1997 by Microsoft for roughly \$400 million (W3C – History, 2013). In late 2012, Microsoft re-branded the service to Outlook and by the end of 2013 all Hotmail users will be upgraded to Outlook.

Number of estimated web users 73 million (The Evolution of the Web, 2013).

2.1.8 1997

XML

In 1997 Extensible Mark-up Language (XML) was starting to become popular. It is a language that can be used to stipulate other mark-up languages and is used to describe data rather than format it like HTML. XML “doesn’t do anything” (W3C, 2013) and is seen as being as important for the web as HTML due to its ability to transfer data between one application to another. The HTML of a document then takes that data and displays it. An example of this being used would be the data that is returned from an Application programming interface (API) which is then parsed into readable data on the web page via HTML.

Number of estimated web users 117.70 million (The Evolution of the Web, 2013).

2.1.9 1998

In 1998 there were another three major changes or breakthroughs that would affect the Web for many years to come

HTML4

In 1998, a major change was made to the standards of which the HTML language adhered to in order to satisfy the needs of the ever expanding Web (HTML Source, 2013). New tags were included to give developers far more options; like the <iframe> tag allowing the input of other websites into a page or the <button> tag that enabled more user friendly interactions. New attributes were also available to HTML, like an elements class or id which would allow elements to be singled out and described by Cascading Styles Sheets (CSS).

CSS

The information that a web page contained was placed on a page via HTML, but the way that information was presented would now have the ability to be changed by CSS. The entire look of a page can be changed, from the pages font, layout or colour. As mentioned, with the new attributes an HTML tag could be given like its class or id any part of the document could be changed and alter with a cascading style sheet. A major benefit of this would be that if a style sheet was included in multiple pages then they would all have the same properties described, so any major changes made to the look of a website would be updated across all pages instead of one (W3C, 2013). This idea of placing external scripts in a page and it being effective for many pages allowed web designers to create far greater looking sites.

Google launches

Google is a company that requires no introduction. When Larry Page and Sergey Brin were attending Stanford University they would begin the most successful internet company in the history of the Web (The Evolution of the Web, 2013). The company's plan was to "organise the world's information and make it universally accessible and useful" (Company – Google, 2013) and regardless of where it has been via the search engine or the amazing web application Google Maps, they have succeeded with their plan. Newer web applications like Google Docs, Gmail, Google Drive or Google earth or the Mobile operating system Android have been "game changers" as far as software development is concerned.

Number of estimated web users 183.91 million (The Evolution of the Web, 2013).

2.1.10 1999

AJAX/XMLHttpRequest

Two more very important technologies used when creating dynamic and highly responsive web applications and methods that will be used in the creation of the application. XMLHttpRequest (XHR) is an application programming interface (API) that uses most frequently JavaScript in order to send HTTP or HTTPS requests to a server. It then will receive a response in data forms like JavaScript Object Notation (JSON) or XML and that response can then be placed in a web page.

Ajax is a Web Development technique that uses these XMLHttpRequests in an asynchronous way. Asynchronous requests allow the web page to send and receive data without it refreshing. This method has been vital in creating applications like Google Maps, which load in data in the background and when the user requires it, the information is presented with minimal lag time (Google – Company, 2013)

These techniques are very important in creating the desktop like programs available on the Web.

Number of estimated web users 276.95 million (The Evolution of the Web, 2013).

2.1.11 2000

Dot-Com bubble bursts

One of reasons why there has been a great advancement of web technology in the years leading to 2013 is due to the 'Dot-Com Bubble' crash at the turn of the millennium. There was a ridiculous inflation placed on many internet companies who brought no real value to the table, companies like

Pets.com (NY Times, 2013), and the market crash that resulted in from this bubble allowed truly innovative companies and technologies to be created in the years that followed as they could learn from the mistakes that the dot-com companies had made. From the period between 1996 and 2000, the stock index of NASDAQ grew from 600 to 5,000 points (The Stock Bubble, 2013) however it wasn't long before the bubble burst. Trillions of dollars was wiped from the market in a very short period of time and by 2002 the NASDAQ's index had fallen to 800.

Number of estimated web users 395.09 million (The Evolution of the Web, 2013).

2.1.12 2001

Wikipedia launches

Larry Sanger and Jimmy Wales launched Wikipedia in January 2001 and it has become one of the most popular websites available on the Web. Articles on information are written by volunteers and verified by contributors. This is included as the most important advancement in 2001, not for an advancement of technology, but due to the online collaboration that has linked millions of people together. Tim Berners-Lee wanted to enable 'researchers from remote sites in the world to organize and pool together information' (W3C-History) and who could argue that Wikipedia is not the site that has the greatest effect of empowering web users with information.

Number of estimated web users 497.58 million (The Evolution of the Web, 2013).

2.1.13 2002

WEB 2.0

A term to describe the 'new version' (W3C, 2013) of the web was coined in 1999 but it was around 2002 before major changes to the way that web developers and users used the web. This vision was that a virtual community could exist between web sites via user generated content, rather than simply viewing content. The thought process of user contribution paved the way for some of the most popular and influential web applications that still exist in 2013

Number of estimated web users 658.66 million (The Evolution of the Web, 2013).

2.1.14 2003

Wordpress launches

With the vision of Web 2.0 firmly in minds of web developers, it wasn't long before web applications reflected the vision. WordPress is a popular content management system (CMS) and blogging site

that is used in 16% of all websites on the Web (WordPress Statistics, 2013). The site has allowed people who may not be highly skilled in Web Development coding to quickly have their say online.

Number of estimated web users 770.74 million (The Evolution of the Web, 2013).

2.1.15 2004

Facebook launches

2004 would see Mark Zuckerberg launch 'the Facebook' at Harvard University which is a social network that has grown to have over 900 million members (Facebook Statistics, 2013), half of those members use the site on a daily basis, with the average user spending 700 minutes on the site per month. There is no question of its overwhelming popularity, but why has this network become a massive part of daily life where others have failed? Facebook capitalised on blogging and user generated content and has connected the world together.

Number of estimated web users 879.35 million (The Evolution of the Web, 2013).

2.1.16 2005

YouTube Launches

YouTube is a very popular video hosting and sharing website that was launched in February 2005. With the technological advances in camcorders and video on camera phones, the user was again enabled, this time to gain the ability to share video content with who ever wanted to watch. In every minute, 72 hours of video is uploaded to the site (YouTube, 2013). In 2006, Google purchased YouTube for \$1.6 billion despite the site having never turned any profit.

Number of estimated web users 1.02 billion (The Evolution of the Web, 2013).

2.1.17 2006

JQuery

jQuery is a JavaScript library of coded functions that has made difficult scripting functions far quicker to implement. It is defined in the jQuery documentation as 'a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development (jQuery, 2013)'. The functions contained can animate, validate forms, create date pickers, manipulate data and make AJAX queries simpler to name a few. This is seen as a necessity to deliver user friendly applications and it is no coincidence that 55% of the Web's most popular websites 10,000 sites contain jQuery.

Number of estimated web users 1.15 billion (The Evolution of the Web, 2013).

2.1.18 2007

1st iPhone released

In what may have been the most important release of them all, the 1st iPhone was released in June 2007. Mobile computing existed before the iPhone, however the ease of use and marketing tactics that Apple employed has sold over 192 million units since its release (Statistics Brain, 2013). This profitable success has pushed application development for both web and mobile into very high demand. The iPhone could take photos on the move and link them with services on the web allowing sharing of moments instantly. The camera phones with internet capabilities that exist nowadays are now as standard phone requirement, meaning more people are able to capture events or moment that they can keep or share.

Number of estimated web users 1.37 billion (The Evolution of the Web, 2013).

2.1.19 2008

HTML5

HTML5 was launched in 2008 with its main aims being to improve the language by supporting new forms of media but maintain a standard that would be universally understood by humans, computers and devices. It contained support for new tags like <video>, <audio> and <canvas> elements that allow multimedia to be placed directly into the web page without the use of Flash or other video or audio player. Sections of the <body> tag were also more defined into separate elements. <section>, <article>, <header>, <footer> and <nav> were introduced. The <canvas> element has allowed developers to create drawing applications for the web. Overall there is no longer a requirement for 3rd party applications like Flash, Shockwave or Java.

Number of estimated web users 1.57 billion (The Evolution of the Web, 2013).

2.1.20 2009

Node.js

Node.js is a server-side development method of creating web applications. JavaScript is used to write the program on the server side and the goal of the software is to “provide an easy way to build scalable network programs (node.js, 2013).” The Node.js server is then combined with JavaScript library’s like Backbone.js, and applications built with node.js respond to events quicker than other

types of applications. A benefit of creating node.js applications would be if the application has to be scaled quickly then this can be done easily.

Number of estimated web users 1.80 billion (The Evolution of the Web, 2013).

2.1.21 2010

Instagram launches

Launched in 2010, Instagram is a mobile photo-sharing application that allows users to apply filters and borders to photos before placing them within the social network of Instagram. The application is a photo only social network that allows the posting to other networks like Facebook and Twitter. The application has made it easy to take professional quality photos by simply adding effects.

Number of estimated web users 2.01 billion (The Evolution of the Web, 2013).

2.1.22 2011

WebGL

In 2011, a JavaScript API called Web Graphics Library (WebGL) was launched. This allows web browsers to display interactive 3D graphics without the use of plugins. This library now gives web applications the potential to have high performance effects and image processing capabilities.

Number of estimated web users 2.27 million (The Evolution of the Web, 2013).

2.1.23 2012

SOPA

The Stop Online Piracy Act (SOPA) is a bill that has been produced in the United States of America to try and combat copyright breaches and counterfeit goods being sold (Sopa | The Guardian, 2013). The act was generally seen as an attempt to censor or control the Web and was met with protests from Google, Wikipedia and Reddit. The laws that were proposed would have allowed the United States government to close down websites if they infringed on copyright issues. On the 20th of January 2012 the bill was postponed.

Number of estimated web users 2.405 billion (New Media trend, 2013).

2.1.24 2013

Vine

Vine was released in early 2013. This application allows its users to create 6 second videos and share them with the world on Twitter or Facebook (Vine, 2013). Owned by Twitter, the app is unique as the 6 seconds of the video do not have to be in order. What that means is that the user holds down to record, then can change the shot and record again until the 6 seconds have elapsed. The human stop motion animations that can be created are unlike anything else available and the potential of this application is very exciting.

2.2 Human factors, the digital world we live in

In just over 20 years, the Web has changed the way the world works forever. We now use technology in nearly every aspect of our life, to organise our day, to do our shopping, to socialize and we have transformed into a society that is dependent on the web. If the internet was unplugged for just one day there would be a 'serious panic in financial markets around the world (Brandswatch, 2013)' and companies like Amazon and Google would have billions wiped off their value. This section will look at the statistics concerning who uses the Web and what we use it for. Factors effecting design will be examined before looking at how the Web has improved our quality of life. Finally, the negative effects the Web is having on our physical relationships.

2.2.1 How many?

The most recent study of World Internet use was published in June 2012. This study looked at the most recent censuses of countries and internet usage data collectors like Nielsen online and the International Telecommunications Union (WIUSIWPS, 2013). The World's population was shown as being 7,017,846,922 people and that 34.3% (2,405,518,376) of the world would be classed as Internet users. Considering that in June 2000, a similar study showed just over 360 million internet users. In the 12 years between the studies there has been a 566.4% increase in the amount of internet users.

The average user is now spending 16 hours per month connected to the internet (The Evolution of the Web, 2013) and with mobile internet devices now becoming more affordable, this figure will again be certain to rise. Hotmail, Google, Wikipedia, WordPress, Facebook, YouTube, Instagram and Vine all have one thing in common; they empower the user to do something with technology via the web.

2.2.2 How the Web has improved our quality of life

As mentioned, our civilised society is deeply integrated with the web and has made so many aspects of life far easier before its existence. Below are 3 ways that the web has improved the quality of life to its users:

Access to information

With a search engine like Google, a user can search on how to do or learn just about anything. This unfiltered access to information has allowed people to learn in unconventional ways. University's standard practice is to offer online degrees and the recent popularity in massive online open courses (MOOC)'s have allowed millions of people to access online distance education without tuition fees.

Connecting people

Social networks and blogging have allowed the world to connect to each other and voice their opinion. Other methods like Voice of Internet Protocol (VoIP), Video calls with Skype and instant messaging services like WhatsApp have both improved and reduced the cost of long distance communication.

Online Shopping

The choice of products available on the Web is greater than any high-street in the world and the products purchased are delivered straight to the door. This convenience has saved us many hours walking about shopping centres looking for the correct purchase. Amazon and eBay alone have revolutionised online retailing by offering a consumer an easy, cheap service. We can have purchases shipped from all over the world for minimal cost at the click of a button on computer, laptop, smart phone or tablet.

2.2.3 How the Web is affecting our physical relationships

As the world becomes more dependent on the web, there has unfortunately been a developing negative undercurrent in relation to time spent online. Given that we are spending an average of 16 hours online per month, every month (WIUSIWPS, 2013); this also means we are spending 16 hours less in the physical world. Author Sherry Turkle looked at the break down of human relationships in favour of technology in her book *Alone Together*, 2011. She argues that technology has replaced the necessity for face-to-face contact where people “Don’t have time to make a phone call, so send a text message instead”, and we as a society now prefer the technology communication option. With the recent explosion of social media and worldwide “smart phone” devices in use moving passed 1bn in October 2012 (Telegraph, 2013), the way the world converses with each other was bound to develop side-effects.

Communication

Before the internet, aside from having a conversation, there were two methods of reaching someone that wasn’t immediately in your presence. Writing a letter or making a phone call. It’s true the Web has allowed people to send important e-mails instantly or video call a friend on the other side of the world, but at what cost has the dependence on these mediums had on the face-to-face conversations.

Social media has allowed us to “share thoughts, music and photos” to an ever growing group of acquaintances, (average number of Facebook friends for all users in May 2012 was 229 (Pew

Internet, 2013)), to boost our egos and brand our own online “personalities”. It has allowed users to be connected to so many online despite being connected in a similar way in the real world. A reason for this as Turkle argues, is that people like the idea of many relationships, but at lengths they can control with the secondary online persona allowing users to escape from the real world to be whoever they want.

By carrying smartphones around, this connection to the internet is now all day, every day. Before advances in mobile communication, a user would be required to sit stationary in front of a desktop computer to enter the virtual world. This can now be done from any position that has 3G or Wi/Fi reception via internet enabled mobile phone or tablet. Turkle recalled a graduate student of hers who was talking to a friend, when that friend took an incoming call on his mobile phone. The graduate student has effectively “been placed on pause” until that incoming call had been completed. These constant interruptions from our physical conversations to receive calls, text messages, twitter updates, e-mails, push notifications or friend requests are now accepted within society as being the new etiquette.

Addiction

What is slightly more worrying than the breakdown in communication affecting some is that people are becoming addicted to being online. Internet addiction disorder (IAD) is becoming a more common diagnosis for psychologists (AOA, 2013) and although there is sparse definitive research on the problem, a loose definition of the term would be that excessive time spent online is interfering with daily life. A recent study by the University of Swansea (PLOS ONE, 2013) has shown that when heavy users of the web log off they are starting to experience similar withdrawal symptoms to that of narcotics users. One of the Professors involved in the study was quoted as saying that “we do not know exactly what Internet addiction is, our results show that around half of the young people we studied spend so much time on the net that it has negative consequences for the rest of their lives.” As Turkle puts it, “we have a neurochemical response to every ping” and these reasons to return and go back online are having serious effects on peoples wellbeing.

2.2.4 Users and Computers

Having considered how society has benefited by using the web and also areas that side effects are being developed, it would now be a good point to consider the science behind the interaction of computers and web applications.

Human Computer Interaction

The study of Human-Computer Interaction (HCI) involves analysing and improving by design the way that users interact with computers. With regards to Web development, how well the design of the site has considered HCI can be a major factor in how 'sticky' the site is. Web stickiness is considering features of the site that will encourage return visits (What is, 2013), however these features have to be designed in a way that considers the communication, cognitive psychology or user satisfaction of the site content.

In order to create a functional, usable Web application it is important to understand that I will be developing for people to use and that should be the 1st priority. The fundamental part of a web application to achieve successful HCI is to develop the interface of the site.

Web Usability

How successful an interface is can be measured by how easy the site is to use for the end user to use, whatever that user may be, having had no extra help to learn how to use the site (Ratner, 2002). Implications that can be drawn from my reflection of Julie Ratner's Human Factors and Web Development would be that the goals of any successful application in terms of web usability would:

1. Take into consideration that users can have physical or mental disabilities and that the information should be simple and clear.
2. Consider the Visual Hierarchy of a page, how important specific content is and its place on a page should reflect the user's needs. It goes without saying that the navigation of a page should be very accessible, however different pieces of content can have differ in weight in terms of relevance to the remainder of the site.
3. Intuitively give users options to move from one stage of the site to another without confusion.

These 3 important aspects compliment the 10 Usability Heuristics developed by Jakob Nielsen in 1995 (Nielsen, 2013). It will be important for this project to consider these important heuristics at the design stage and I will be reflecting later how well the application created fairs with these.

User Experience

Jakob Nielsen's company the Nielsen Norman Group state that "User experience encompasses all aspects of the end-user's interaction with the company, its services, and its products. In order to achieve high-quality user experience in a company's offerings there must be a seamless merging of

the services of multiple disciplines, including engineering, marketing, graphical and industrial design, and interface design (Nielsen, 2013)". The users experience with a web application is differentiating from user to user as there can be very different perceptions of the service. Effective UI, 2010 states that there are 3 factors that must complement each other in order to build great user experiences and they are Science, Art and Craftsmanship. The goal of all Web applications should be an engaging application that ensures a user wants to use the service, and this project is no different. The JavaScript library jQuery UI is a library built on top of jQuery and the application will take advantage of its features to improve the user experience.

2.3 Web Applications

By looking at the characteristics of Web applications and what properties define them, a greater understanding will ensure that this project is successful.

2.3.1 What is a web application?

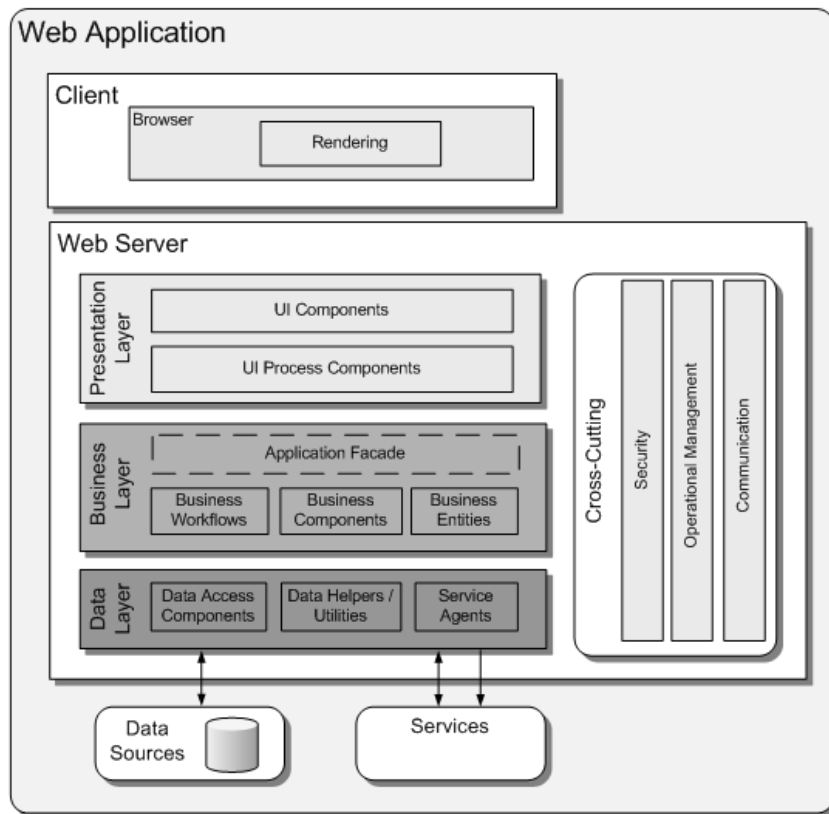


Figure 2.5: A typical Web Application structure
(<http://www.webappers.com/img/2008/09/web-app-testing.png>)

The term web application can several definitions but broadly is accepted as a website that allows a user to complete specific tasks and what these tasks are defines the category of application, which can vary from an online store, social network or e-mail service. A standalone website that is not an application will simply provide the user with a form of information.

As shown in Figure 2.5, a typical web application has separate characteristics that separate the logic from the presentation (G Kappel, 2006).

2.3.2 Web application architecture

In order to gain an understanding on the principles of web application architectures, reference was sought from the book Web Engineering, 2006. The standard of a web application can be measured by how effective the underlying architecture is and how the analysis of the applications

requirements is translated into its architecture, which is very important to the success of the project. In order to achieve the highest quality of application, there are important properties that the web architecture must contain:

1. The applications architecture must describe the structure;
2. The architecture forms the transition from analysis to implementation;
3. The architecture makes a system understandable.

When this project reaches the design stage, aspects that will be considered when developing the applications architecture include the Functional requirements of the users, previous experience, performance and technical abilities of available technologies. Along with these attributes, patterns and frameworks will be looked at to gain understanding.

2-tier Architectures

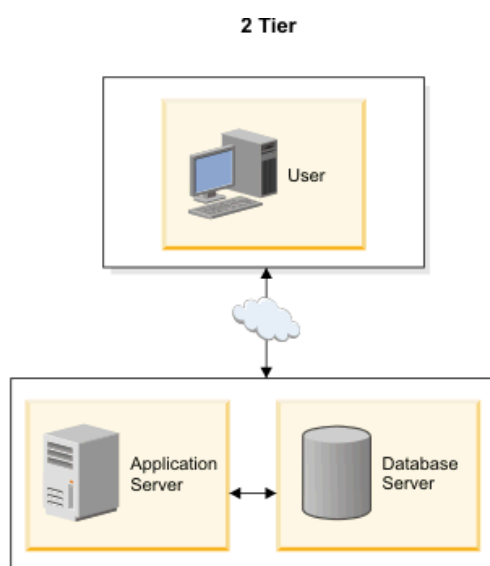


Figure 2.6 2-tier architecture

(<http://pic.dhe.ibm.com/infocenter/>)

2-tier architectures are used for simple web applications where a client requests are handled by dynamic HTML pages.

2-tier architectures are used for simple web applications where a client requests are handled by dynamic HTML pages.

N-tier Architectures

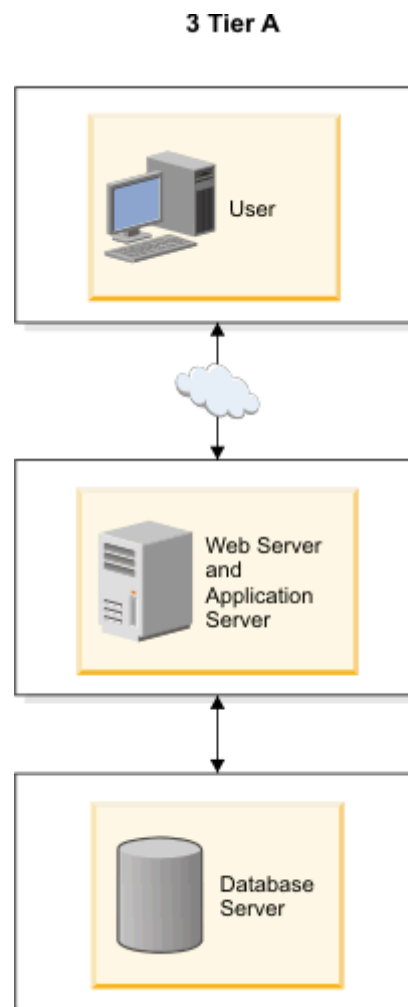


Figure 2.7 3-tier architecture (<http://pic.dhe.ibm.com/infocenter/>)

In contrast, multi-tiered architectural approaches are used when the application required to be developed when the amount of users will be larger or the logic behind the application may be complicated. There are usually 3 types of tiers in the application, the data tier, the business tier and the presentation tier.

Patterns

Patterns convey common design problems that regularly appear in a particular design circumstances and suggest an answer. A solution “describes the participating component, their responsibilities, the relationship between these components, and the interplay of these components within the specific problem,” meaning that patterns will allow the use of tried and tested methods and previous knowledge that has already been part of a high quality software architecture. According to (Web Engineering, 2006) the most popular architectural pattern is the Model-View Controller (MVC).

Model-View-Controller (MVC)

MVC splits the application into three parts: the data (Model), the presentation layer (View) and the user interaction layer (Controller). A user will firstly interact with an application before the controller's event handler's trigger. The controller will then request data from the model before passing it to the view to then present the data to the user.

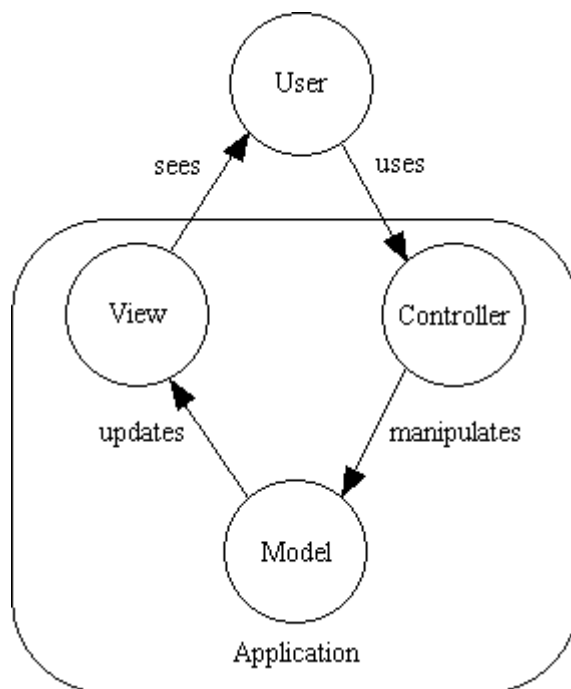


Figure 2.7 shows the process that the application goes through with this pattern.

Frameworks

Another architectural option for creating web applications would be to use a framework. A framework is a template for the application of sorts where the main functionality is already implemented. This basic template blueprint allows the application to have a strong platform to build on top of. Frameworks have a lot of documentation to help with the implementation as they are normally used by many users; however, some can prove to be a steep learning curve.

2.3.3 Web Application technologies

The method or language that web applications can be created using virtually the same architecture can vary. Two popular methods of creating web applications at the moment, PHP and Node.js, will be looked at in order to grasp what makes other developers choose these methods. The pros and cons of each method will be researched along with the available frameworks to create the applications.

PHP

Having already given an introduction into PHP in 2.1.6 I will discuss reasons for using the language as based on the information from two books, one by Ralph Mosley called Developing Web Applications (2007) and G. Kappel's Web Engineering (2006). As Mosley mentions, "PHP is well suited for Web Application Development particular because of its ability to allow dynamic interaction".

- PHP is an easy to learn and very quick server side scripting language
- The support available is extensive, partly due to the popularity of the language.
- PHP works very well with databases, file systems and images.
- It can be the P in but the LAMP (Linux, Apache, MySQL, and PHP) and WAMP (Windows, Apache, MySQL, PHP) which are combinations of software allowing a development setup. This allows for cheap hosting.
- The PHP online manual is "acknowledged" as the best in the world for any language.
- There are a large amount of functions that are built into PHP which can be extended.
- The integration with MySQL is very easy.
- PHP runs on most platforms without having to change anything about the way the language works

Available Frameworks:

-Code Igniter

-Bones

Node.js (Node.js, 2013)

The system is designed for creating highly scalable web applications but using JavaScript on the server-side to create event driven asynchronous applications that are very fast. Some of the reasons for using Node.js are below, and this information is based on the Node.js website.

- Asynchronous events allow node.js applications react to events instantly to send messages to the client or database
- JavaScript is used very heavily on the client side with almost universal usage so by using this on the server side it is easier for a developer who may have high knowledge to code on the server side, without having to learn another language.
- Packages available in node are very accessible. The node.js package management system (npm) allows the download or replication of packages through command line to allow

greater features to be placed or integrated into the application like socket.io which aims to make real-time applications like an instant messaging service to be possible.

- Active Development within the node.js community is being supported and the materials available are starting to increase.

Available Frameworks:

-Meteor

-Express

2.3.4 Libraries

Applications can have additional libraries inserted into the application for a certain purpose, to improve functionality or improve the look. Below are important libraries used in application development that will come in useful during the implementation of this project:

JQueryUI (jQuery UI, 2013)

Having already mentioned jQuery earlier and mentioned its importance, an extension of that is jQuery UI. Built on top of jQuery, UI is a predefined set of user-interface themes, effects, buttons or interactions that allows a richer user experience. UI contains Autocomplete, Date pickers, Progress bars, a slider, tabs, tooltips, animations, accordion navigation as well as various pre-set effects like fade-in or out.

An example of jQuery UI in use would be the date picker on an airline site, where the date picker pops up to save the user typing the information into the box.

Bootstrap (Bootstrap, 2013)

A popular front-end framework, Bootstrap is a collection of tools designed for web application that is responsive to the device it is shown on. Created by Twitter, the responsive design of bootstrap allows the pages of a site to dynamically adjust to the PC, tablet or mobile device it is being used on. The way this works is that the layout is in grid format with 9 sections, where each section of content is placed within a span class. The containers individual div's are then given spans, and each span corresponds to a certain width. For example a div with span1 would take 1/9th of the entire width of the screen, whatever size that may be. Divs can also be made to be visible depending on the width of the grid, or the width of the screen. So a large logo could appear on PC but an icon would replace it on mobile.

2.3.5 Databases SQL/NO SQL

Since the 1970's relational databases have been the most efficient way of storing data generated and these databases would use Structured Query Language (SQL) to allow the finding of data stored. With any technology, innovation is driving force that pushes boundaries and it wasn't long before Relational Database Management Systems (RDBMS) were created. Examples of the popular RDBMS systems in use today are MySQL, Microsoft SQL Server and Oracle (Juravich, 2012 p8).

RDBMS gain popularity due to the simplicity of the model which was easy to learn and easy to manipulate. It also had a high level of ACID properties, which are at the heart of relational databases. ACID is defined by four main properties:

1. Atomicity – All of the functions or code must be committed to the database or none of it will atoll.
2. Consistency – Data should be validated to remain the same type.
3. Isolation – Database transaction will be individual and not affect others.
4. Durability – When the data is committed to the database it must be safe regardless of circumstances that could corrupt it.

These properties make the RDBMS's very powerful; however they also are one of the factors that make reading and writing become tediously slow.

After the rise of Social networks, companies wanted to realise the potential of the large amount of data they stored and very quickly. Unfortunately relational models couldn't provide answers to all of the questions being asked, most commonly about scalability and availability. This allowed the term NoSQL to gain popularity.

NoSQL actually stands for "not only" SQL and is designed to complement the relational model but not replace it. The 1st thing that makes NoSQL different from the traditional approach is that they have a different data structure that can have 4 variations, Key-Value stores, Document Stores, Graph databases or Column stores. Whichever structure is preferred, the scaling performance in terms of size and complexity is far greater when compared with a traditional relational database (Juravich, 2012). Another big difference between NoSQL and traditional means is that it avoids the previously mentioned concept of ACID, and by having less restrictions, NoSQL data writing speeds are incredibly fast. By abandoning ACID, NoSQL goes back the concept known as the CAP Theorem, which acknowledges the defining characteristics of all databases or distributed environments. Eric Brewer, a computer science professor from the University of Berkeley, first introduced the CAP theorem in

2000 (Wired.com, 2013). The theorem specifies that for any distributed environment, it is not possible for three of the following guarantees to be made:

- Consistency: The systems servers will all contain the same data meaning that regardless of who is accessing the data; it will all be the latest version.

- Availability: There will never be a server that will not return data.

- Partition-tolerance: If part of the system gets an error and fails, the remainder of the system will still operate as normal.

NoSQL databases take two of the three options from the CAP theorem and then develop ways of controlling the third option.

As this project aims to explore new technologies from an exploratory perspective, a NoSQL database is the preferred option to be implemented in this project.

2.3.6 NoSQL solutions

Although there are many options available when it comes to NoSQL solutions, there are two far more popular in terms of usage and who have a higher quality of documentation available. I will describe both before giving examples of use.

Mongo DB (Mongo DB, 2013)

Mongo DB defines itself as the “leading NoSQL database” and it is easy to see why it has become the most popular. It retains some of the SQL properties like index and querying so would prove to be an easier stepping stone from the traditional relational methods. The most common use would be building a similar database as would be done with MySQL that would require dynamic queries on a large database.

CouchDB (Juravich, 2012) (ApacheCouchDB, 2013)

The homepage of CouchDB gives an excellent quick summary of what it’s all about. “Apache CouchDB is a database that uses JSON for documents, JavaScript for Map Reduce queries and regular HTTP for an API”. That simple yet fully describing piece of information reflects a lot on how this database is intended to be used. By using HTTP/RESTful JSON API, the data is very easily accessible. CouchDB’s two guarantees with regards to the CAP theorem are the availability and partial tolerance allowing vast amounts of changing data to be accessed by pre-defined queries. CouchDB allows administration duties to be performed via its own built in web application called Futon. Within each

CouchDB database, a document with a unique id is created. These documents are stored as JSON objects and are totally schema-less meaning that each document can be completely different to the next.

Cloudant is a CouchDB hosting service which replicates the setup locally into the cloud. The database is queried in the same way, and the service provides exactly the same Futon administration.

2.3.7 Mobile Applications

With the explosion in the requirement for mobile applications and emergence of tablet computers, the need to replicate the same service on the web in many places has never been so necessary. The Bootstrap framework as mentioned does a very good job of creating responsive web design for every device available however a mobile application can be created without having to learn operating system code for Android or iOS.

jQuery Mobile

Another JavaScript library developed by the jQuery developers, jQuery Mobile allows a creation of unique apps for a mobile device by using HTML, CSS and JavaScript (jQuery Mobile, 2013). Its features include touch optimised controls built on top of jQuery so if a developer has learned jQuery the transition period is very quick. Its website also contains a theme builder, allowing custom themes for an application to be created. The application can then be exported to PhoneGap.

PhoneGap (PhoneGap, 2013)

PhoneGap is a framework for creating mobile applications that allows developers to use HTML, CSS and JavaScript to create applications for mobile devices without having to learn the Operating systems languages. For example, an Android application is created in HTML like a normal website, before being placed through PhoneGap via Eclipse. The resulting application is a fully optimised Android application.

2.4 What Next?

Having considered the History, the Human considerations and web application methods, this gives good grounding to begin to tackle the problem that this application will aim to solve. The first step is to analyse the problem, before creating a survey to generate data from users.

3 Problem and Data Analysis

This chapter looks at how the research findings and data analysis will motivate the design and implementation of the project. As discussed in section 1.1, the implementation of the project should be in an experimental way with the technologies that are available.

In order to build on the research that has been conducted, this project will establish a survey on the general public regarding their online behaviour towards the web before looking at the external services that are available as API's that will be included within the application. The project hopes to take on board the responses from the survey before creating application requirements.

Firstly however, in order to create questions to ask the general public in a survey, the problem faced will be looked at in greater detail.

3.1 Problem overview

In order to fully understand the problem, the objectives of this project are stated below:

1. Analyse existing online multimedia services and web development technologies;
2. Conducting a study of online social behaviour;
3. Design and implement an application that interacts with existing web applications to create a social experience.
4. Explore new technologies of Web Application Development.
5. Proper Testing and Evaluation of the application.

Having already looked at application methods used on the web in chapter 2, as well as some of the factors that affect our relationships with computers, the project will require more depth into online social behaviours. What needs to be established is what services people use and what they use them for. It also needs to be determined how people interact with events, what they do with the tickets from the events. Keeping or collecting items can become a very private matter so any questions presented in a survey would have to be well-structured to extract information. The problems at this stage could be summarised by the following three points:

- There is a lot of information available regarding events in the future and the past from an artist or organiser perspective but rarely does the information reflect how people feel about these events, if they enjoyed them, and what they remember about them. There is also little or no information about how people would like to interact with events after they have passed.

- There is not currently a service that would allow the storing of event tickets that would then interact with content generators to deliver information about the event. The types of content generators need to be explored.
- The tickets from events would need to be computer images via photo from some source i.e. a camera, so there needs to be a method of extracting the information from the ticket before storing the image itself and information in a database.

3.2 Research survey

The main purpose of the survey created will be to get opinions on the web, social media, what events they attend, attitudes towards these events and the potential ticket information storage application.

3.2.1 Survey Design

The survey will contain questions that are of similar type. They are shown below with a list of possible answers before reasons as to why the questions are being asked. The types of questions are categorized into four groups; General Questions, Events, Event Behaviours and Social Media and Web Services.

The questions asked to the survey participants are shown in the table below Survey Questions:

Table Name: Survey Questions

Question	Possible answers
Gender	Male or Female
Age	Under 16, 16 to 18, 19-21, 22-25, 25-30, Over 30
How frequently do you attend ticketed events?	Weekly, Fortnightly, Monthly, Never, Other
After attending an event, what do you do with the ticket from that event?	Keep every ticket, Keep Some, Throw away, Other
If you keep your tickets, where do you keep them?	Open answer
What is the 1st event you can remember attending?	Open answer
What do you remember about it?	Open answer
What the best or most memorable event the participant has attended	Open answer

Which music festivals have been attended this year	Rockness, T in the Park, Creamfields, Belladrum, None or Other
Most memorable moment of the festival.	Open answer
Type devices the participant takes to an event	Smartphone, Digital Camera, Camcorder, Sound recording equipment or other.
Type of mobile device the user has.	iPhone, Android Device, Blackberry, Windows Phone or Other
Type of content created at an event.	Photos, Videos, Sound Recordings or None.
The reasons that trigger the need to create content.	Song being played, People the participant is with, To remember what it was like or other.
Invitation to voice concerns about Facebook and any issues the participant has with Facebook's service.	Open answer
If the participant has a YouTube or Google account.	Yes or no
If the participant has a Spotify account	Yes or no
If the participant has a Flickr account.	Yes or no
Invitation to voice opinion on Facebook events	Open answer
How the user most frequently checks Facebook	Laptop/Desktop, Mobile device, Tablet or Other
Find the users thoughts on using Facebook Connect to register or login to a site	More, Less or Don't know
Invite the user to say why they would use Facebook Connect or not	Save time filling in login details again, Trust Facebook with data or Other

Appendix A has a more detailed list of the questions which includes reasoning behind why the questions are being asked.

The questions developed have been created in a way to make it easier to justify design decisions at a later stage, but also to gauge if this application would be a service that the general public has a need for.

3.3 Survey Findings

The survey software used was a Google Doc form where participants were told that the results would be stored on Google servers. In order to keep the responses as honest as possible, the participants were told this was an anonymous survey.

The method of delivering this survey was via Facebook where it was shared. It was also placed on efestival's (<http://www.efestivals.co.uk/>) forum, Skiddle (<http://www.skiddle.com/community/>) and drowned in sound (<http://drownedinsound.com/community>). This gave a demographic of people who share similar interests and who are interested in going to music events.

The survey had 52 replies and the important results are show below in the table below Survey Results, with Appendix B having a more detailed version of these results including the open answered questions.

Table Name: Survey Results

Question	Answer	Percentage
Gender	Male	67%
	Female	33%
Age	Under 16	0%
	16 to 18	12%
	19-21	29%
	22-25	38%
	25-30	12%
	Over 30	10%
How frequently do you attend ticketed events?	Weekly	10%
	Fortnightly	29%
	Monthly	54%
	Never	0%
	Other	8%
After attending an event, what do you do with the ticket from that event?	Keep every ticket	40%
	Keep Some	35%
	Throw away	23%
	Other	2%
Type of mobile device the user has.	iPhone	52%
	Android Device	35%

	Blackberry	0%
	Windows Phone	6%
	Other	8%
If the participant has a YouTube or Google account.	Yes	87%
	No	13%
If the participant has a Spotify account	Yes	42%
	No	58%
If the participant has a Flickr account.	Yes	21%
	No	79%
How the user most frequently checks Facebook	Laptop/Desktop	54%
	Mobile device	40%
	Tablet	6%
	Other	0%
Find the users thoughts on using Facebook Connect to register or login to a site	More	56%
	Less	35%
	Don't know	10%
Invite the user to say why they would use Facebook Connect or not	Save time filling in login details again	56%
	Trust Facebook with data	35%
	Other	10%

Appendix B also has analysis commentary of the results as well as charts of the data.

3.4 Survey Conclusions

Indictors show that there may be a requirement for this service within the social group tested, with 75% of participants surveyed stating that they keep tickets from events they have been to. It is necessary to remember that the participants share similar interests so the results are not reflective of the general public, however the people questions could certainly be described as a target market for the application. Important information was gained about the types of devices this selection of people use, with 94% taking smartphones to events and 52% of the participants phones were iPhones. With an every changing web, the fact that 46% of participant's most use Facebook on a mobile device more often than personal computer, the design of the application should take that

into consideration. With 87% stating they had a YouTube/Google account; this would make integrating the application to YouTube far easier than Spotify or Flickr.

The quality of the responses when the participants were asked what they remember about events is very interesting. These memories only exist through photos, videos and thoughts at the moment and this project will aim to change that. There is no graphical scale or statistic that can represent a feeling that is stored in someone's head about the "atmosphere" at a gig or event; however a service to allow people to express what they remember or what it was like should remain the core of this project. This can be quite a personal memory like a ticket box is a personal collection of memories and these should remain private.

Overall the survey has garnered some valuable knowledge about potential users and this will be reflected in the design stage of the project.

3.5 Survey Recommendations

Having reflected on the survey results, below are some recommendations that this project should consider:

- There should be a way to store "memories" about an event.
- The application should respond to the device it is using to create a seamless user experience.
- The login system should not solely rely on a third party like Facebook connect.
- The external APIs should not require a login from the user.
- User experiences differ depending on the type of event the user is at and the application should reflect that.

The original idea for this application was to interlink into other types of content generators like YouTube for videos, Flickr for photos and Spotify for music. With the relatively low percentage of participants from the survey describing themselves as Spotify or Flickr users, interlinking with these services that require registration would not be beneficial for creating a multimedia based application that everyone can use. Clearly required are simple accessible APIs which do not require users to sign up and having learned this, the approach will be modified to take that into account.

It is clear that the survey has been beneficial in order to back the research stated during chapter 2. A key statistic is that just under half of the participants do not use a desktop or laptop when they visit

Facebook and the design of the application should reflect that. The next stage of this project is to create a set of requirements document for the application.

3.6 Application Requirements

Based on the research of chapter 2 and survey results, the requirements of the application are to be stated in order to progress to the design stage. Each requirement will be from a table corresponding to the part of the application, for example, requirement 1.1 will be the 1st requirement of the 1st table. Each table of requirements will share common characteristics. Each requirement will be given an importance rating from 0 (lowest) to 5 (highest) and this will state how necessary that requirement is to the project.

Table ID: 1

Table Name: Handling users

Id	Name	Description	Functional/ Non-Functional	Importance Rating (0-5)
1.0	Login	-Users must be able to login to the application	Functional	5
1.1	Sign-up	-Users must be able to sign-up for the application	Functional	5
1.2	User Details storage	-The data stored about the user will be in a secure location	Non-Functional	3
1.3	Confirm Password	-The application should ask the user to confirm a password	Non-Functional	2
1.4	Validation Sign up	-The form used to sign up to the application should be validated	Non-Functional	2
1.5	Logout	-The application must have a means of logging out	Functional	5

Table ID: 2**Table Name: Data**

Id	Name	Description	Functional/ Non-Functional	Importance Rating (0-5)
2.0	Database	-The application must have a database	Functional	5
2.1	Encryption	-Users passwords will be encrypted	Non-Functional	3
2.2	Events	-The information of about the event will be able to be stored in the database	Functional	5
2.3	Deletion	-The information entered about an event must be able to be deleted	Non-Functional	4
2.4	Update	-The information about an event should be able to be updated	Non-Functional	1
2.5	Intelligence	-The application should learn about types of data that can be entered to improve the overall experience	Non-Functional	1
2.6	Type	-Each event should have a type	Non-Functional	2
2.7	Comment	-The application should store users thoughts of the event	Functional	5
2.8	Artist	-The application will store the artist of the event	Functional	5
2.9	Venue	-The application will store the venue of the event	Functional	5
2.10	Date	-The application will store the date of the event	Functional	5

Table ID: 3**Table Name: Application Features**

Id	Name	Description	Functional/ Non-Functional	Importance Rating (0-5)
3.0	Videos	-The application will return videos from an external source	Non-Functional	4
3.1	Setlists	-The application will return a setlist of the event from an external source	Non-Functional	4
3.2	Information	-The application will return information regarding the event from an external source	Non-Functional	3
3.3	Reader	-The application will read the photo and take the text information from it to a database.	Non-Functional	5
3.4	Upload photo	-The application will allow upload of photos to within the application.	Non-Functional	5
3.5	Crop	-The application will allow users to crop tickets and store the cropped image	Non-Functional	3
3.6	Social	-The application will allow users to connect via social network	Non-Functional	1
3.7	Social share	-The application will allow users to share events via social network.	Non-Functional	1
3.8	How to use	-The application will explain to users how to use it.	Non-Functional	2
3.9	Camera	-The application should allow users to upload photos via camera if accessed on that type of device	Non-Functional	1
3.10	Webcam	-The application should allow users to upload photos via Webcam if available.	Non-Functional	1

Table ID: 4

Table Name: Web Design

Id	Name	Description	Functional/ Non-Functional	Importance Rating (0-5)
4.0	Interface	-The interface of the application will be aesthetically pleasing	Non-Functional	3
4.1	Responsive	-The application should respond to whatever device the user is using.	Non-Functional	5
4.2	Style	-The application should consider all aspects of HCI for the design	Non-Functional	3
4.3	Standards	-The application should conform to standards defined by W3C	Non-Functional	0
4.4	Usability	-The application should consider the 10 usability heuristics as defined by Nielsen	Non-Functional	1

3.7 Summary

In this chapter analysis of the problem was carried out in detail, which resulted in a study of online behaviour through the medium of a survey. The survey results were analysed and based on the study and the literature review of Chapter 2, the applications requirements document was created.

4 Application Design

The design chapter will cover an overview of the system as a whole, the data modelling concerns, the user interface concerns, and review of technologies suitable for implementation before being concluded with a summary. For the remainder of the report the application will be referred to as Tickitbox or the Tickitbox application. The design document will be description of the Tickitbox application that will be used to guide the implementation of the project.

4.1 System

This section will give a brief overview of the system to be created, the characteristics of the system and the systems architecture before looking at the components that will make up the system.

4.1.1 System outline

The outline will give a guide as to what the desired goal is of the design stage will be by considering the applications requirements. The Features the application must have are as follows; a login/logout system with the possibility of the user signing up for the application. The application will also obviously require a database, which will store information about the user and information about the event. These are simply the functional requirements of the application will be considered in detail in this design stage. Figure 4.1 is a low level diagram of the basic functional application:

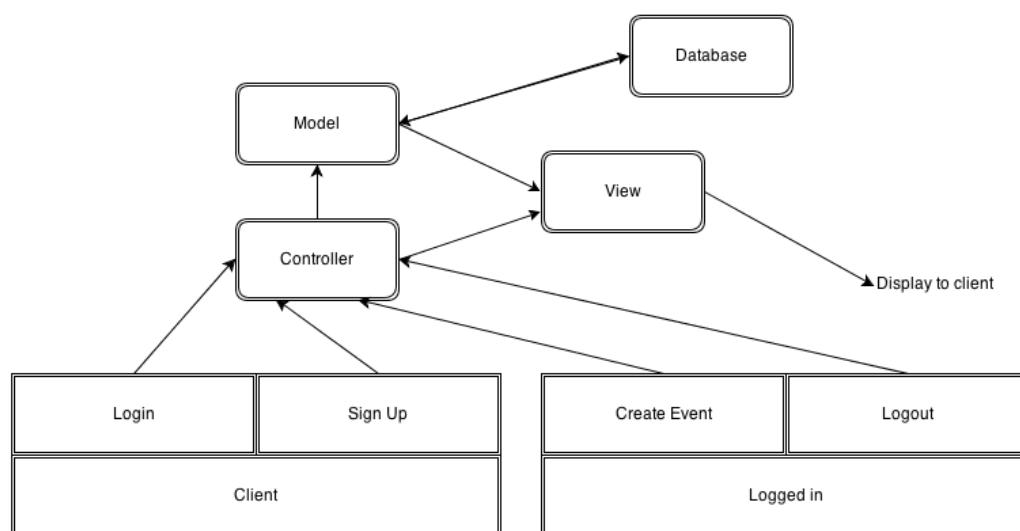


Figure 4.1 Low level Tickitbox

The research of chapter two, section three showed that the most popular architectural method of creating web applications is via the design pattern of Model-View-Controller (MVC) and it is the intention to create a MVC application. The architecture of a web application is very important to how the application performs and the choice is based on how understandable the pattern is to

develop with and the documentation available sub seeds that of similar architectures. Also, with the intention to use a framework for the application, a high percentage of the frameworks available are in MVC form.

The client will make requests to the model before having a two way conversation with the database before the model will pass this information to the view which will be displayed to the client.

The above diagram considers all functional requirements from chapter three, section six and becomes a good starting point for the design of the application. Having established a very basic outline for Ticketbox, the applications system will now be looked at.

4.1.2 System Goals

The goals of the system are as follows:

1. The goal of the Ticketbox application is that it will be easy and intuitive to use and should support users of all level of expertise. Ticketbox will take into consideration the design conventions that applications like Facebook have, not only aesthetically but systematically. As the application is aimed for a younger user with a keen interest in going to events with an aim of accompanying the physical ticket box a user may hold, there will be parts of the design that reflect that.
2. The second goal of the application is to design in a way that gets the end user served with the requested information is the shortest amount of time possible. The design will use a “NoSQL” database not only for the exploration into newer methods, but mainly as these data stores are very quick in the transferring and saving of information.
3. A third goal is to have a way of transferring the information held on the users physical ticket straight to the application via image. The idea is that a user will upload a photo and the application will read the information stored on the ticket before placing that data within the database.
4. A final goal of the application is that it will operate by itself; there will be absolutely no requirement for the user to need help from a human on how to operate the application and how it works.

4.1.3 System Components

Figure 4.2 shows the components that Tickitbox will require in becoming a functional application.

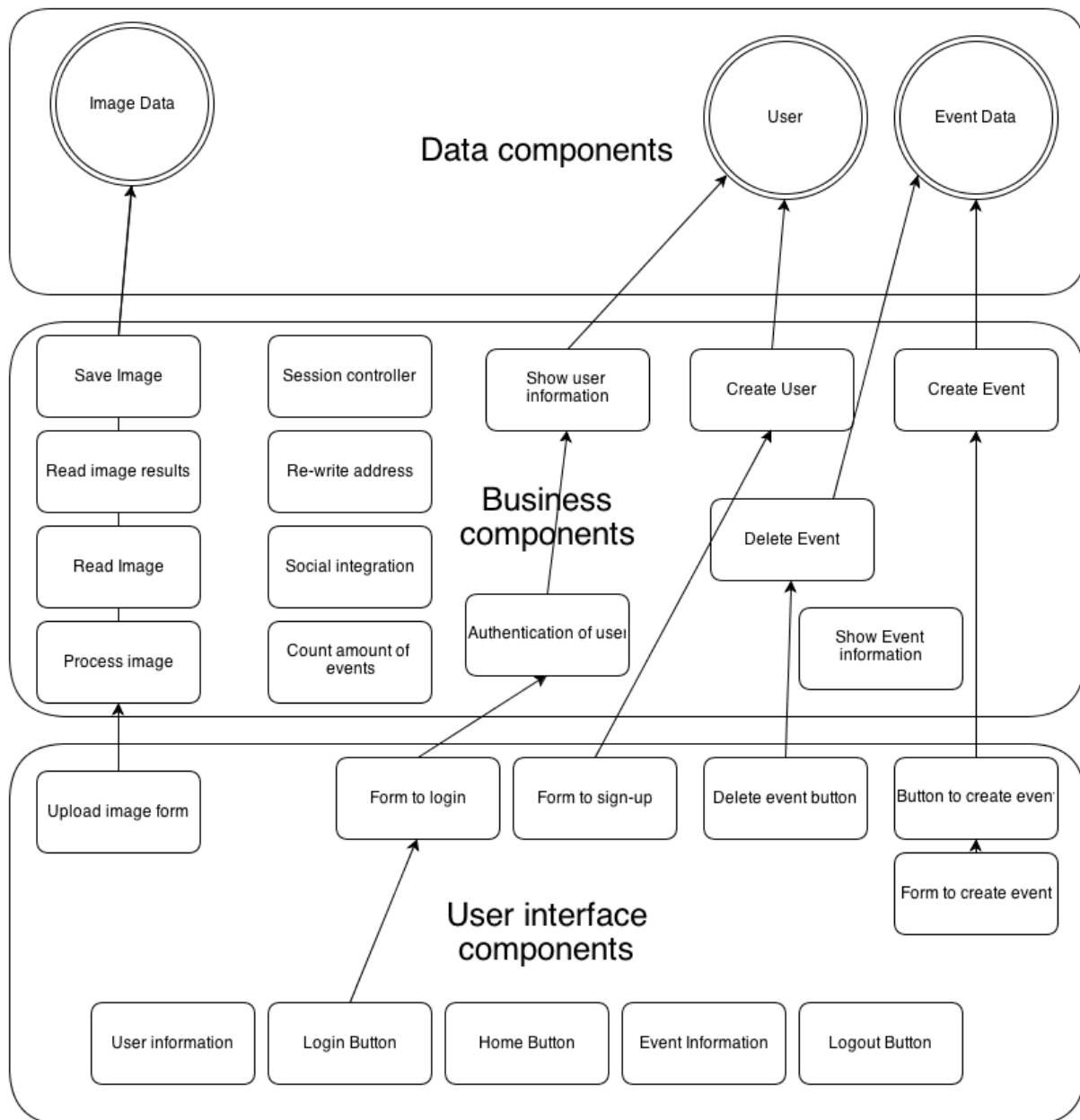


Figure 4.2 Tickitbox components.

There is a clear n-layer architecture being formed using a data, business and presentation layer. This architecture will be defined in greater detail later in the design stage, but these components stipulated are a solid basis for an application system.

The systems components are application objects that will change data from one form to another. In order to define the components required for the Tickitbox application, the requirements document from chapter three will be revisited to determine functions that the application will need to

complete. The components will be categorised into three types Data-Centric, Business Logic and User-Interface. Each component will be given an ID, name, a description of what its function will be and a reference to the requirement that it will help satisfy.

In addition the components of the application will be split into tables and the tables will be given an ID and a name.

Table ID: C1

Table Name: Data Centric components

Id	Component Name	Function of Component	Requirement(s) Satisfied.
C1.0	Users information store	-The component will store the users information	1.2
C1.1	Events information store	-The component will store the information of the events	2.2
C1.2	Image information store	The component will store the images the application creates	3.4

The data centric components shown above in table C1 are part of the data layer of the application. These are the areas of the application that hold the data that the business components will access or create.

Table ID: C2

Table Name: Business Logical components

Id	Component Name	Function of Component	Requirement(s) Satisfied.
C2.0	Authentication of user	-Function so that application knows which user is logged in	1.0
C2.1	Session controller	-Function to start new sessions of a user being logged in or out.	1.0, 1.5
C2.2	Re-write address	-Function of rewriting the address of the applications views from index/hello/world to app/world	N/A

C2.3	Create User	-Function to create a new user record in the database.	1.1
C2.4	Create Event	-Function to create a new event record in the database.	2.2
C2.5	Delete Event	-Function to delete the event record from the database.	2.3
C2.6	Show Event information	-Function to access the event records information for a logged in user.	2.2
C2.7	Show user information	-Function to access the events information record of a logged in user.	1.2
C2.8	Count amount of events a user has been to.	-Function to count the amount of events a user has attended.	N/A
C2.9	Save Image	-Function to save image that have been uploaded	3.4
C2.10	Read image	-Function to read the information off of the image	3.3
C2.11	Read image results	-Function to place the image results into a destination	3.3
C2.12	Process image	-Function to process the image i.e. crop before uploading	3.5
C2.13	Social integration	-Function to interact with Facebook i.e. SDK	3.6, 3.7

The business components that make up the application will provide functionality to for the application. These components will complete the tasks that the user interface components request.

Table ID: 3

Table Name: User Interface components

Id	Component Name	Function of Component	Requirement(s) Satisfied.
C3.0	Form to sign-up	-The component where the user will enter the sign-up information	1.1
C3.1	Form to login	-the component where the user will enter their	1.0

		login details	
C3.2	Logout Button	-the component that will allow the user to logout from the application	1.5
C3.3	Login Button	-the component that will allow the user to login to the application	1.0
C3.4	Home Button	-The component that will return the user to the home of the application	N/A
C3.5	Form to create event	-The component that will allow the user to enter the event information	2.2
C3.6	Button to create event	-The button that will send the event information	2.2
C3.7	Delete event button	-The button that will trigger the delete function	2.3
C3.8	Upload image form	-The button that will open the file upload dialog.	3.4
C3.9	Event Information	-This component will display the event information	2.2
C3.10	User information	-This component will display the users information	1.2

The user interface components of Tickitbox are the components that the user will interact with in order to execute business logic and retrieve data.

4.2 Data Modelling

This section will consider the data the application will store and how the data generated will be interacted with. As this project will aim to explore newer methods the decision has been made to use a NoSQL database. Having reviewed the various types of NoSQL databases in Chapter 2.5.3, the conclusion is that the most appropriate database management system for this application is CouchDB.

The main reason for this choice when compared with MongoDB is that the quality of the documentation available for CouchDB is very high and considering it has never been used before the learning process needs to be as easy as possible. Also the integration with 3rd party services like Cloudant is explained in detail in documentation. The choice of the database is being decided now due to the fact that the document structure will be different when compared with a relational database management system.

Having already defined data centric components that will make up the data layer of the application, UML object, class, content and hypertext models will be created before moving to the Interface stage.

4.2.1 Entities

The first step is to define the entities and attributes that the data models will manipulate. By looking at the data components the entities and attributes of each component can be stipulated as shown in the entities table.

Table Name: Entities Table

Images	Users	Events
Images have names e.g. image.jpg.	Users have IDs e.g. 001	Events have IDs e.g. E001
Images have types e.g. .jpg, .png or .gif.	Users have Usernames e.g. innesm4	Events have Artists e.g. Primal Scream
Images have sizes e.g. 2.4MB.	Users have first names e.g. Mark	Events have Venues e.g. Music hall
Images have dimensions e.g. 480px x 360px	Users have last names e.g. Innes	Events have dates e.g. 05/06/2012
	Users have passwords e.g. helloworld	Events have types e.g. Festival, sport, music

	Users have types e.g. user, admin	Events have comments e.g. This event was very enjoyable
	Users have email addresses e.g. innesm4@gmail.com	Events have images e.g. image.jpg
		Events have users e.g. innesm4

4.2.2 UML Modelling

Unified modelling language (UML) is a popular standardized modelling language used for developing software and includes a set of graphic notation techniques that create visual models of software systems. The process of create models using UML has consulted the 2003 book Learning UML by Sinan Si Alhir. Having already identified the data components, entities and attributes of the application, UML models will now be completed in order to show a static structure of the system.

Use case diagram

Figure 4.3 is a Use case diagram of the Tickitbox system:

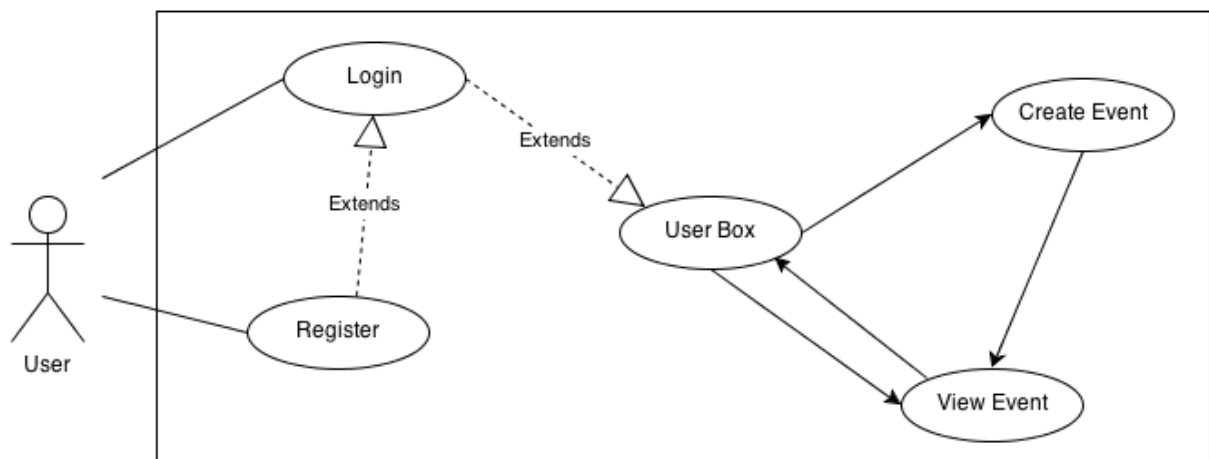


Figure 4.3 Use case diagram

The use case diagram introduces the idea that a user has a “box” within the application. This is where the user will create and store events.

Activity diagram

Figure 4.4 is an activity diagram of the event creation process:

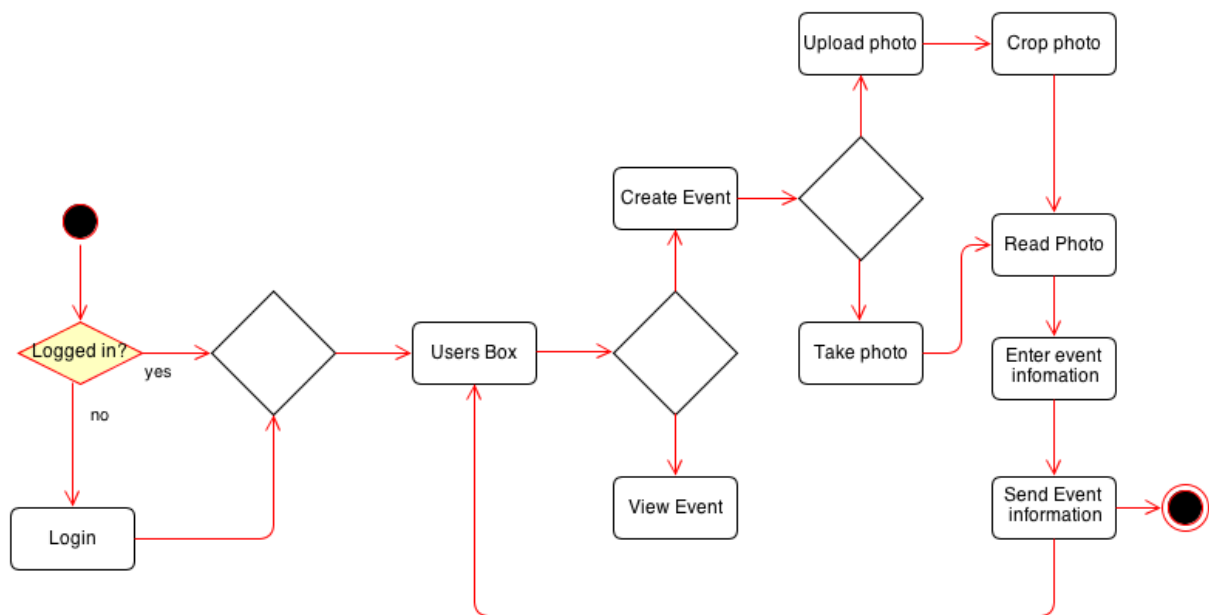


Figure 4.4 Tickitbox Activity diagram

The diagram shows the process that a user will go thru to create an event.

4.2.3 Content Modelling

A content model shows all the types of content that the application will have. It will contain the definitions of each content type's elements to each other. A popular way of showing content is through the user of an ER Diagram and will expand on the entities already defined.

ER Diagram

Although there will not be a RDMS, it is still necessary to create an ER model, as shown in figure 4.5:

E-R Diagram

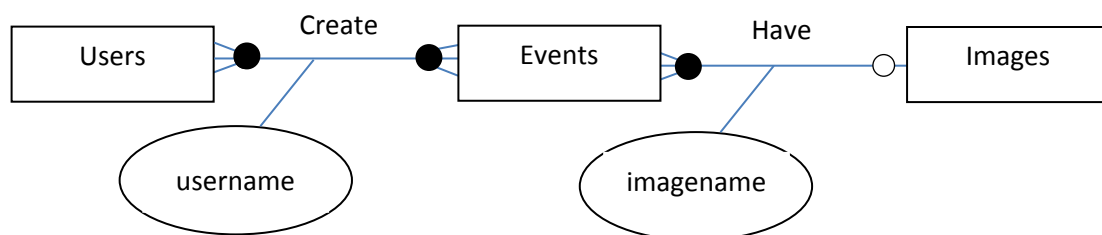


Figure 4.5 Tickitbox ER model

Entity Sets

Users (user_id, username, first_name, last_name, password, type, email)

Events (event_id, artist, venue, date, type, comment, imagename, username)

Images (imagename, type, size, dimensions)

Relationships

Create (username) –Many users create many events [M:N] [m:m]

Have (imagename)– Many events have one image[M:1] [m:o]

Constraints/Assumptions

In order to create an event there must be a user

Each event must have an artist, data and venue

The ER model shows the entities and relationships these entities have with each other. The events entities will link the image and user by the image name and the username. With entities established the process will move onto higher level modelling.

Class diagram of the proposed system

Figure 4.6 shows a class diagram of Tickitbox:

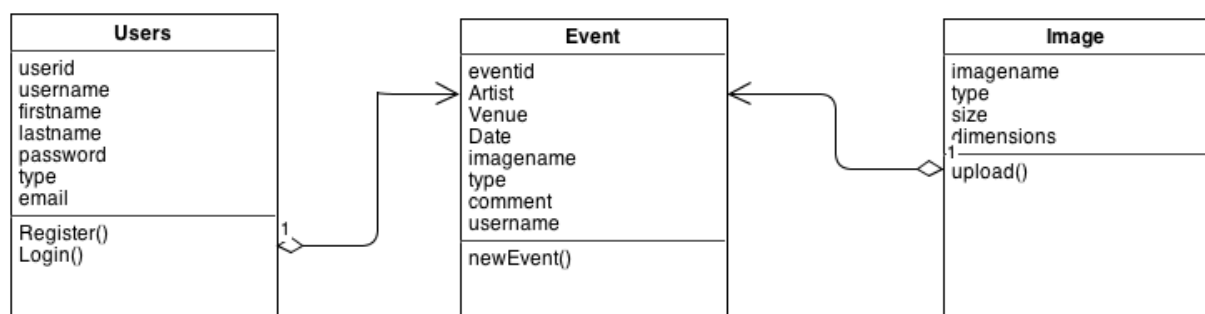


Figure 4.6 Tickitbox class diagram

The diagram shows the classes that Tickitbox will have and the methods they will require to create.

4.2.4 Hypertext Model

The hypertext model will define the navigation and composition of the site before it is placed into an MVC mapping at a later stage. The systems composition describes the pages that the application will

hold and what contents these pages will have.

The navigation of the application will be specified by showing links. These links can be hyperlinks to another page or within the same page. As this will be an MVC application, the hypertext segments created will be reflecting the applications views.

Hypertext structure model

Figure 4.7 shows a hypertext structure model of the Tickitbox Application:

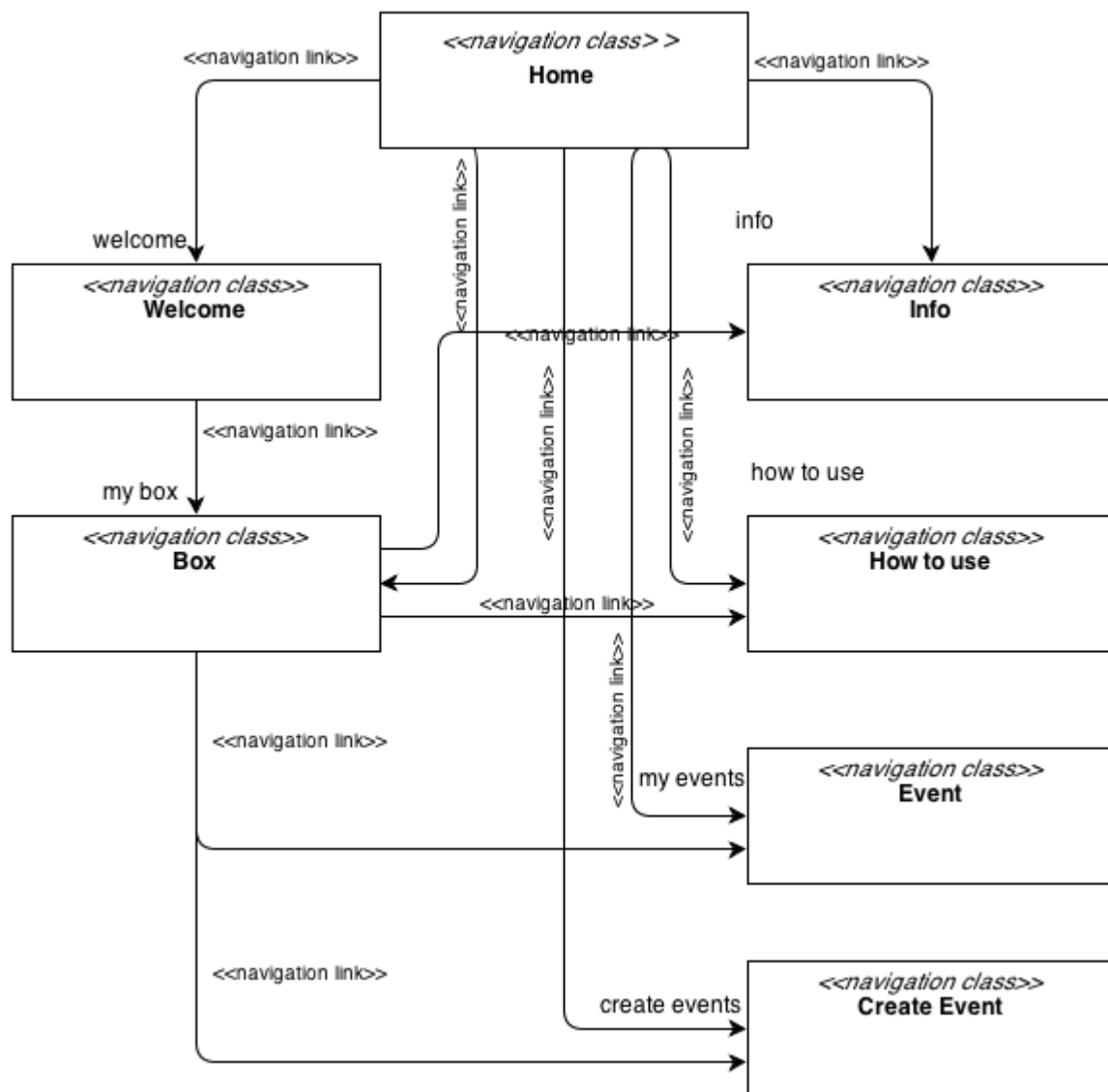


Figure 4.7 Tickitbox hypertext structure model

The model shows seven pages, Home, Welcome, Box, Info, How to use, Event and Create Event. The model has taken into consideration the requirements document and this hypertext model satisfies each requirement.

4.3 Interface

Also known as the presentation layer, the interface deals with the look and feel of the application. The most crucial hypertext pages or views of the application will now be expanded on in detail and will be given a presentation page, before progressing to wireframe and finally a mock-up of the final design. Also included are notes describing the objects necessary or tasks that page will complete that page.

4.3.1 Generic page (No functionality aside from navigation i.e. Home)

Presentation page

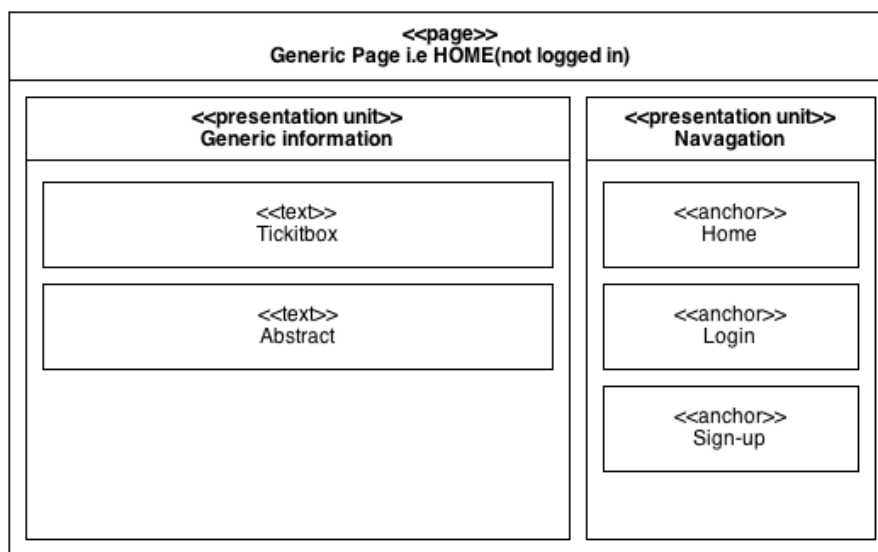


Figure 4.8 Presentation page of Generic page

Wireframe

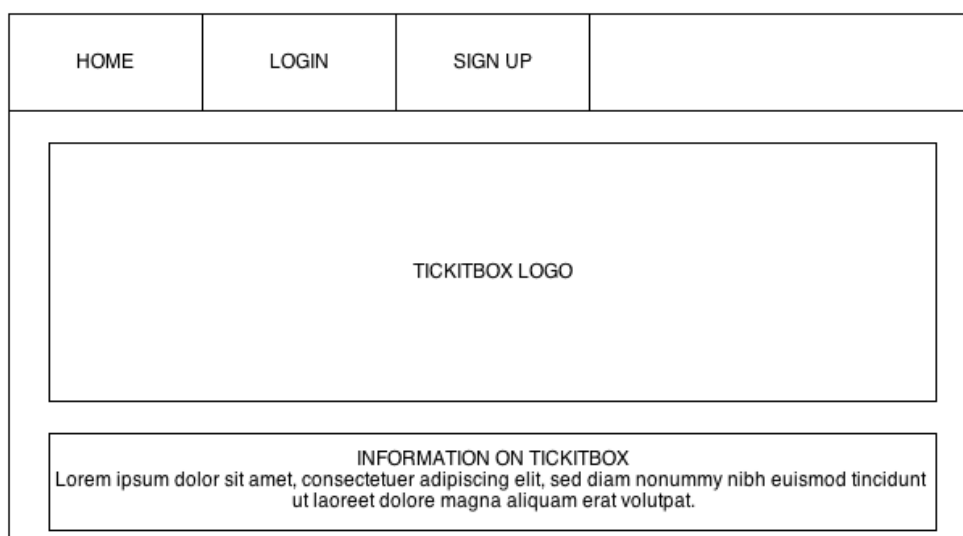


Figure 4.9 Wireframe of Generic page

Additional features/notes generic page

- Interface will concern the Homepage, Info, How to use and Welcome pages.
- The login and Sign up pages will have forms in place of content to perform the functions required.

4.3.2 My Box view

Presentation page

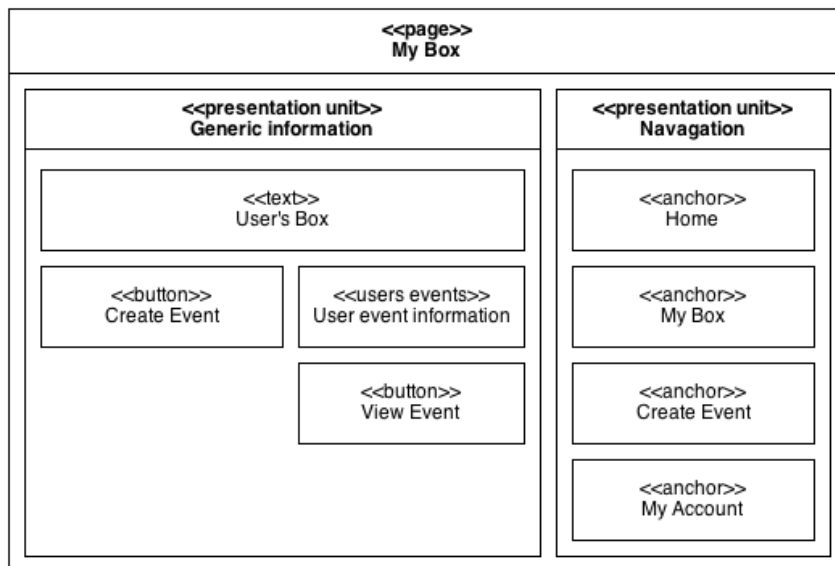


Figure 4.10 Presentation page of My box

Wireframe

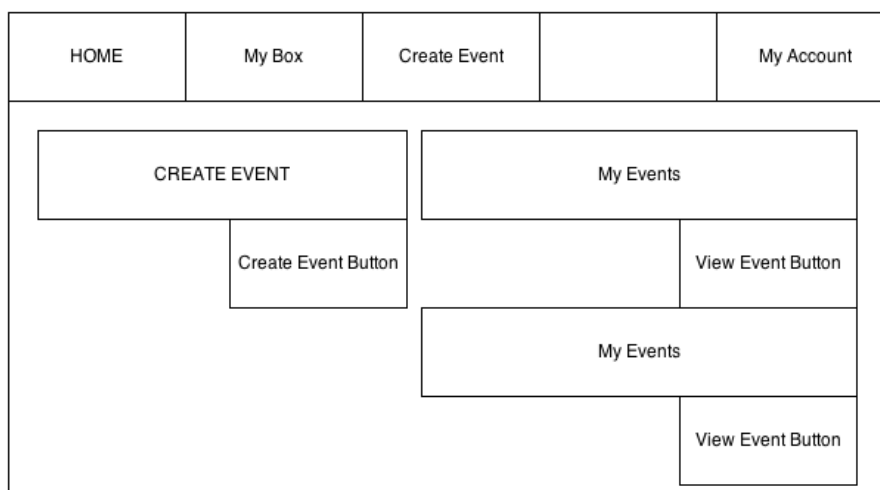


Figure 4.11 Wireframe of My box

Additional features/notes

- The events shown will be that of the user which have already been created
- The My Account will be a drop down menu to the Info and how to use pages

4.3.3 Event view

Presentation page

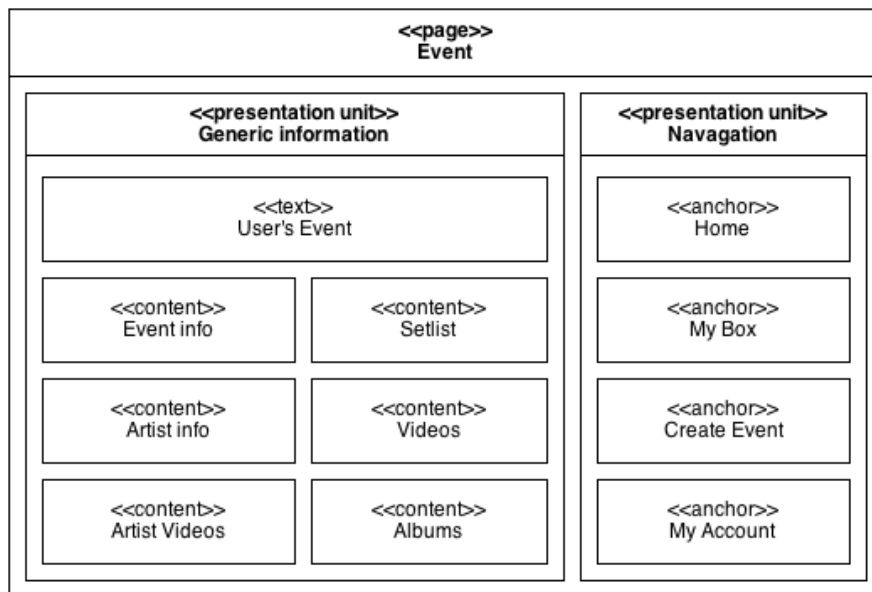


Figure 4.12 Presentation page of an event

Wireframe

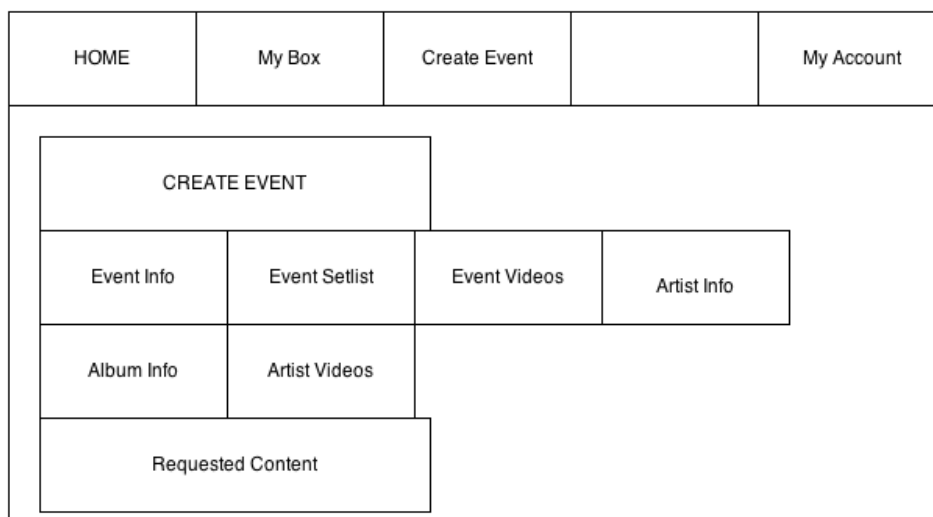


Figure 4.13 Wireframe of Event page

Additional features/notes event view

- The requested content will be shown in an area depending on the type of action required. For example clicking on the Event setlist will show the events setlist
- The actions available will be a tabbed layout.

4.3.4 Create Event view

Presentation page

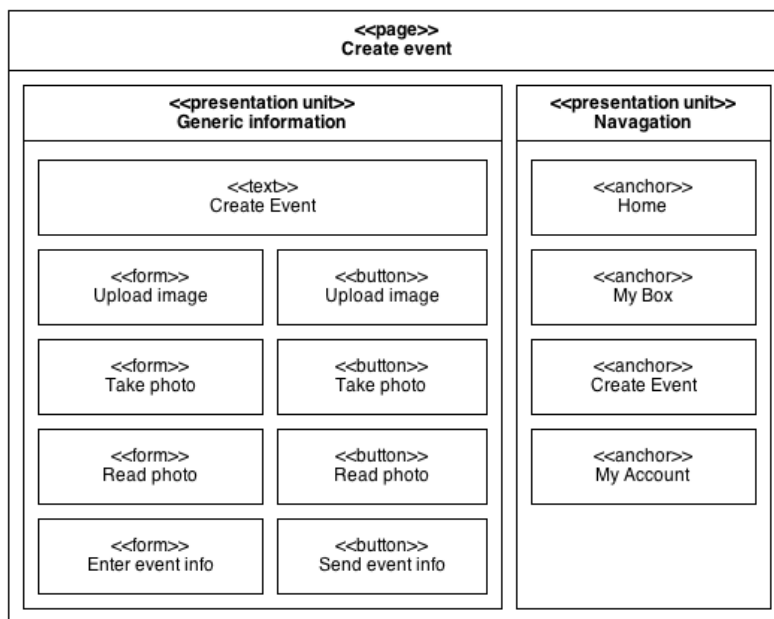


Figure 4.14 Presentation page of creating an event

Wireframe

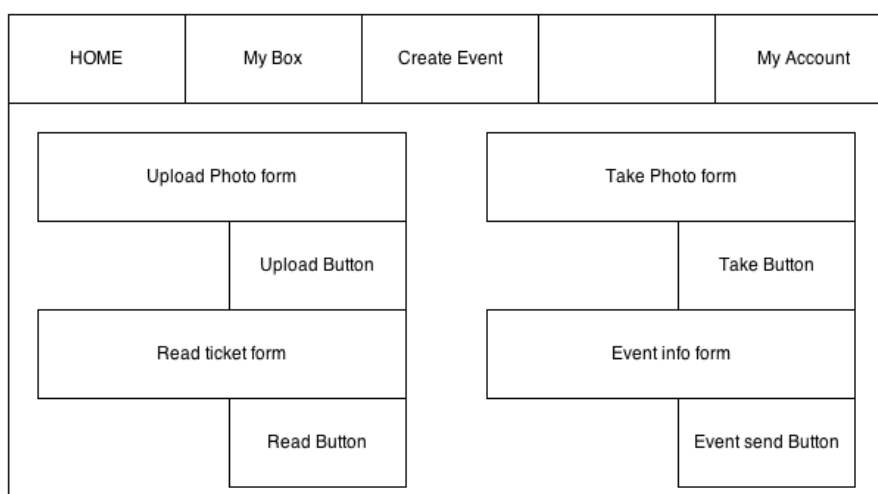


Figure 4.15 Wireframe of Create Event

Additional features/notes create event

- Each form shown will be part of a larger form with the goal of creating an event.
- The upload location of take photo or upload photo will be stored in the Event info form.

The Interface of all the main pages is now complete and the design stage will move to the technologies that will provide the Tickitbox components with functionality.

4.4 Technologies

The choice of the appropriate technologies is important as to how successful the Tickitbox application created will be. Chapter 2.3 looked in detail at various technologies and now comes the time to make the decision on which technologies will interact with the applications architecture.

It is obvious that the Tickitbox application will make use of HTML, JavaScript and CSS during the development but the application will need to make use of more powerful technologies to comply with the features that the components will have.

There are six identified areas that require a design decision to be made. The six areas are listed below and will follow the following format:

- Description of necessity and which components the decision will effect.
- List of the choices for that area that are available.
- Design choice.
- Reason for making this choice.

4.4.1 Web Services

-Description of necessity and which components the decision will effect.

The choice of web service is important due to the large amount of cross domain request that will be required, from accessing API's to accessing the CouchDB. This decision will affect every layer of the application. Simple Object Access Protocol (SOAP) and Representational state transfer (REST) are the two most popular methods of sending and receiving data between client and server.

-List of the choices for that area that are available.

- SOAP
- REST

-Design choice:

REST

-Reason for making this choice.

REST is very simple and doesn't have many strict standards to adhere to. The data that can be sent and received can be JSON, XML or text and CouchDB uses REST services so it is important that the data from the applications database uses the same web service as the rest of the application.

4.4.2 Image Storage

-Description of necessity and which components the decision will effect.

The images of the tickets will be required to be kept somewhere within the application or within a third party service

-List of the choices for that area that are available.

- Store on couch as attachment
- Self creation code using a server side language
- Imgur API
- Amazon S3

-Design choice:

Delay until implementation stage

-Reason for making this choice.

The application will look at implementing all methods before deciding during the implementation stage which is most suitable for the project

4.4.3 Optical Character Recognition

-Description of necessity and which components the decision will effect.

Optical character recognition is the electronic conversion of images into text form and that is what will be required to read the ticket information

-List of the choices for that area that are available.

- Self creation code using a server side language

- Abby OCR
- OCR API service
- OCR Tesseract

-Design choice:

Delay until implementation stage

-Reason for making this choice.

The application will look at implementing all methods before deciding during the implementation stage which is most suitable for the project.

4.4.4 APIs

-Description of necessity and which components the decision will effect.

The API's that will provide the applications features based on the information stored on the ticket.

-List of the choices for that area that are available.

- YouTube
- Soundcloud
- Last.fm
- Grooveshark
- Spotify
- Setlist.fm

-Design choice:

Delay until implementation stage

-Reason for making this choice.

The application will look at implementing all methods before deciding during the implementation stage which is most suitable for the project.

4.4.5 Image Processing

-Description of necessity and which components the decision will effect.

The image will need to be that of a certain size to be passed to the image reader or upload functions. This could be done by cropping or third party service.

-List of the choices for that area that are available.

- Self creation code using a server side language
- Cloudinary
- Transloadit

-Design choice:

Delay until implementation stage

-Reason for making this choice.

The application will look at implementing all methods before deciding during the implementation stage which is most suitable for the project.

4.4.6 Server-side technologies

-Description of necessity and which components the decision will effect.

The language that will be imbedded in the application and will provide the application with most of the business logic is a very important decision concerning the success of this project.

-List of the choices for that area that are available.

- PHP
- Node.js

-Design choice.

In order to decide which language will be most suited for the application development, two prototypes were created, using PHP and Node.js. The prototypes were not fully functional however did have some of the applications features. As shown in figure 4.16, the goal of the prototype was to have a login system and to create an event. The event would then return video from YouTube based on the event.

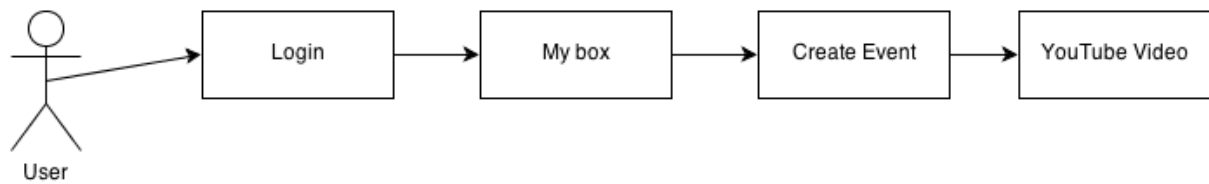


Figure 4.16 Goal of the prototype

The prototypes will contain sample code and screenshots of the prototype and a summary of the creation process.

PHP prototype

Installing PHP to local machine took longer than expected, the php.ini file had to be reconfigured as the local machine was using recently released. After initial teething problems WAMP server was up and running and ready for development.

It is easy to see why PHP has become as popular as it has. The sample framework did effectively all the work in creating the MVC architecture and in order to create a login system the PHP Facebook SDK was used. The create an event function was a simple html form that when submitted calls a script that sends the information to the localhost's CouchDB.

JavaScript was then used to retrieve information from the database before feeding it into the YouTube API. JSON was returned before being parsed using jQuery back onto the page.

Shown below is sample code from the index.php of the prototype:

```
get('/', function($app) {
    $app->render('home');
});

get('/hello/:name', function($app) {
    $app->set('name', $app->request("name"));
    $app->render('hello');
});

post('/hello', function($app) {
    $app->set('name', $app->form('name'));
    $app->render('hello');
});
```

Figure 4.17 shows an example of the PHP prototype:

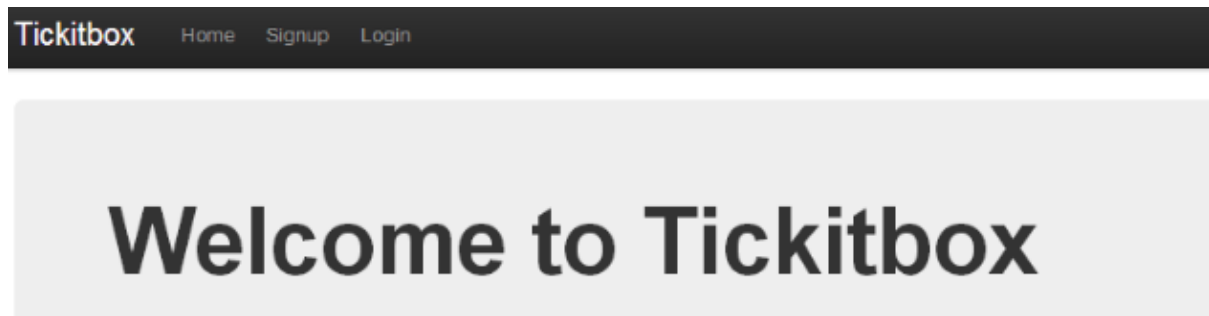


Figure 4.17 PHP prototype

The PHP framework was easy to follow and the integration with HTML and CSS means that a lot can be done in a short space of time.

Node.js prototype

The node.js prototype was simple to create and install. The program was downloaded via the node.js home website and the server was up and running ready for development within minutes. By using the node packet manager, express.js was installed from the command line so very quickly a basic template for the application was created.

In order to create a login system, the Facebook SDK was again used in order to achieve functionality quickly.

Shown below is the example node.js code of the prototype that was made:

```
// JavaScript Document
var express = require("express");
var app = express();
app.get('/', function(req, res){
  res.send('hoya!');
});
app.listen(3000);
var http = require('http');
var times = 50;
while (times--){
  var req = http.request({
    port: 3000
```

```

, method: 'POST'
, headers: { 'Content-Type': 'application/x-www-form-urlencoded' }
});
req.on('response', function(res){
  console.log(res.statusCode);
});
var n = 500000;
while (n--){
  req.write('foo=bar&bar=baz&');
}
req.write('foo=bar&bar=baz');
req.end();
}

```

This shows a server being created at localhost:3000 and including the express.js web development framework for node.js.

Next was the create an event components that linked to the localhost CouchDB server. As everything is in JavaScript the creation of CRUD commands was very simple.

Finally the YouTube API was called and returned into XML via an Ajax request.

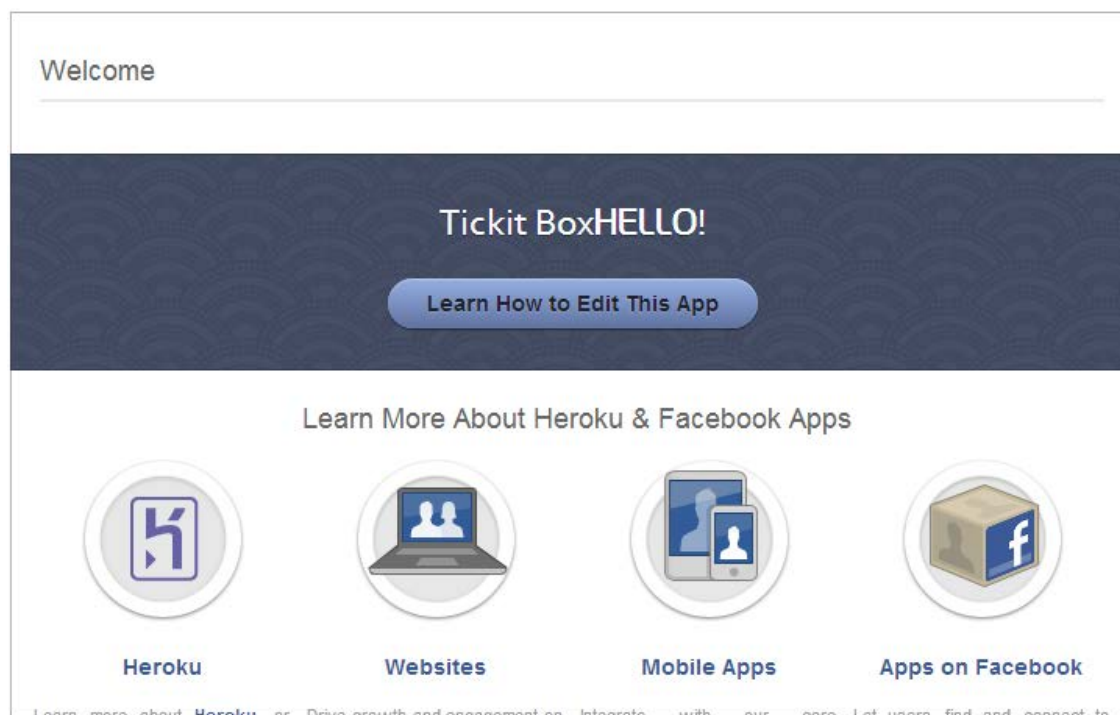


Figure 4.18 Deployed to Heroku

Tickitbox

[Logout](#)



Mark Innes
Your Access Token: AAAGmc1Rd7vUBAIsPT8HOVAGeRT7aKzZB8bG4QVfKyAvAnetY4ySNDzatrwhLxBwZCpPfMdEPK

[Publish Wall Post](#) | [Share With Your Friends](#) | [Publish Stream Using Graph API](#) | [FQL Query Example](#)

Write your status here and click 'Status Set Using Legacy Api Call'

[Status Set Using Legacy Api Call](#)

Welcome to Tickitbox

Events

[Add an Event](#)

Figure 4.19 Homepage of Node.js prototype

The node prototype is very fast and updates in real-time. Concerns about using this are the lack of documentation for larger applications. It seems as well that there is a novelty with the node support and is at a very experimental stage.

-Reason for making this choice.

PHP was chosen mainly due to the functions that will require to be performed. There will be a steep degree of difficulty. As this was an experimental prototype process with node.js it is clear that it has its benefits, but these are simply not what is required for this application. The benefits of PHP's documentation and community are far greater than node.js.

The application will still consider new technologies via a NoSQL solution and PHP's integration with such a database has been very encouraging.

4.4.7 PHP Framework

-Description of necessity and which components the decision will effect.

With the scripting language now decided as PHP, a framework is required to execute the MVC application architecture that has been designed in the early stages of this chapter.

-List of the choices for that area that are available.

- CakePHP

- Codeigniter
- Bones

-Design choice.

Bones

-Reason for making this choice.

The bones framework is a project stored on Github that I was immediately drawn to due to the ease of learning. The framework is a scaled down version of CakePHP or Codeigniter and the learning curve for bones would be far less than learning one of the other frameworks. The bones framework is named that as it is the “bare-bones” of what is required to make an MVC application.

Deeper functionality can be added as later stages but for this project, with the time constraints that will affect it, a simple framework that allows quick progress seems ideal.

4.5 Moving On

Before moving to the implementation of the Tickitbox application it is appropriate to acknowledge the design chapter’s progress. The application has moved from a basic idea to an application in waiting. The design stage has taken decisions to best prepare for the implementation of the project with clear goals for the application defined. There are still a few decisions to be made, for example which OCR to use, but these decisions will be made during the implementation stage based on the most suitable working solution.

5 Implementation of the Application

The implementation phase of the project has certainly been the most challenging. Many functions that seemed straight forward to implement quickly became significantly time consuming. There were occasions that with more than one option were available to create a feature, deciding the most appropriate meant implementing each option for the best results. Overall, the application has extremely high functionality when comparing with the system that was designed in chapter four. All requirements from section 3.6 have been tackled and every requirement bar one has been implemented successfully into the application.

This chapter will discuss the major elements of the Tickitbox application will look at how the application was taken from the design stage to a working application and the challenges encountered and solutions provided for the implementation of this project. The important features of Tickitbox's implementation will now be documented and each section will include a summary, screenshots and relevant code that describes how the problem faced was solved. The applications test plan will also be included and the relevant testing will be documented.

The elements covered in this chapter include the creating an event function, viewing an event, user management, application functionality, user interface design, the security and finally the applications quality.

5.1 Creating an event

This concerns the function that allowed users to create events by entering the Artist, Date and Venue of the event in question before adding further details about the event. The main feature that was implemented was the optical character recognition (OCR) on the ticket, which reads the information on an image and places it back into a format that can be used in the application. In implementing the OCR, problems that could not have been considered during the design stage occurred and solutions were found for each circumstance that arose. The section begins by looking at the OCR implementation in detail, before discussing the remainder of this function.

5.1.1 Optical Character Recognition

Section 4.4.3 describes the choices available for the OCR function that would read the text from the images. After some trial and error, it was decided that the best method of extracting the text from a ticket image was via OCR API Service. Abbyy OCR is the most established of the choices, however they only returned text in XML and the process of reading the ticket should be instant, so for that reason alone an API service was a more attractive prospect.

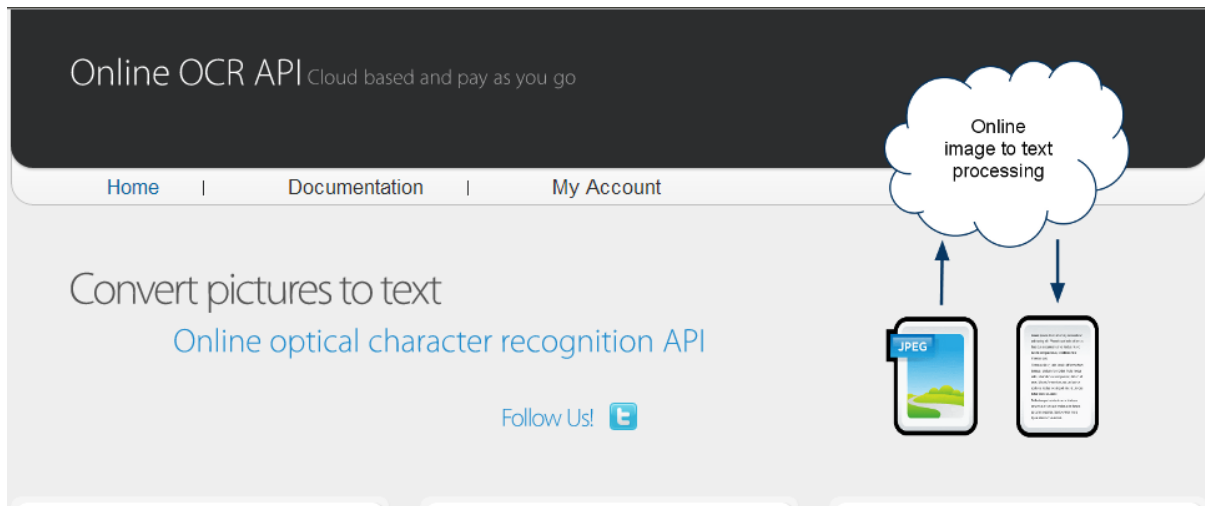


Figure 5.1 OCR API service

OCR API service is a cloud based service that allows uploading to the API and retrieving the results. The image file is simply placed into an HTML form and sent to the API service, which reads the tickets and displays the result. Figure 5.2 shows the form Tickitbox uses to upload images:

```
<form id="upload_form" enctype="multipart/form-data" method="post"
action="http://api.ocrapiservice.com/1.0/rest/ocr" onsubmit="return checkForm()">

<h2>Please select your ticket:</h2><br />
<div><input type="file" name="image" id="image_file" onchange="fileSelectHandler()"
accept="image/*" /></div>
<input type="hidden" name="language" value="en" />
<input type="hidden" name="apikey" value="qV4rmrmC9K" /><br />
<input type="submit" class="btn" value="Read my ticket" />
</form>
```

Figure 5.2 OCR upload form

The hidden commands sent to the API service include the pre-registered API key. During the first attempts to create this system, the problem faced was that the OCR service would open the results of the process into a new window as shown in figure 5.3.

```
SATURDAY 16th FEBRUARY
9PM TILL LATE
LIVE AT THE MJOITERS
JUSTIN MARTIN
PLAYING A WIDE RANGE OF MUSIC FROM THE BEATLES, JOHNNY CASH,
OASIS, PINK FLOYD, STEREOPHONICS, BILL WITHERS, THE JAM AND MANY MORE
```

Figure 5.3 Returned text from an image poster opened in a new page.

Due to it being image uploading, an AJAX method of uploading images to a different domain and retrieving the result in this way was not allowed by web browsers due to the same origin security policy, so another method of uploading was required. Figure 5.2's form has an `onsubmit="return checkForm()"` function attached which solves this problem.

Cross-origin resource sharing (CORS) is a mechanism that allows cross domain image uploads to another domain via an XMLHttpRequest. The solution to this problem was the iframe method. A jQuery plugin located at the address shown in figure 5.4 provide a library that takes the information received by the OCR service and places it back into defined area on the page.

```
<script src="http://malsup.github.com/jquery.form.js"></script>
```

Figure 5.3 CORS jQuery plugin.

By using this library, the problem of placing the text returned from the OCR API service found a solution. As shown in figure 5.4 the text received from the OCR service was displayed into a div via JavaScript, and a method of pulling relevant text from that div was required. Also shown in figure 5.4 is the original image that I used to get that text.

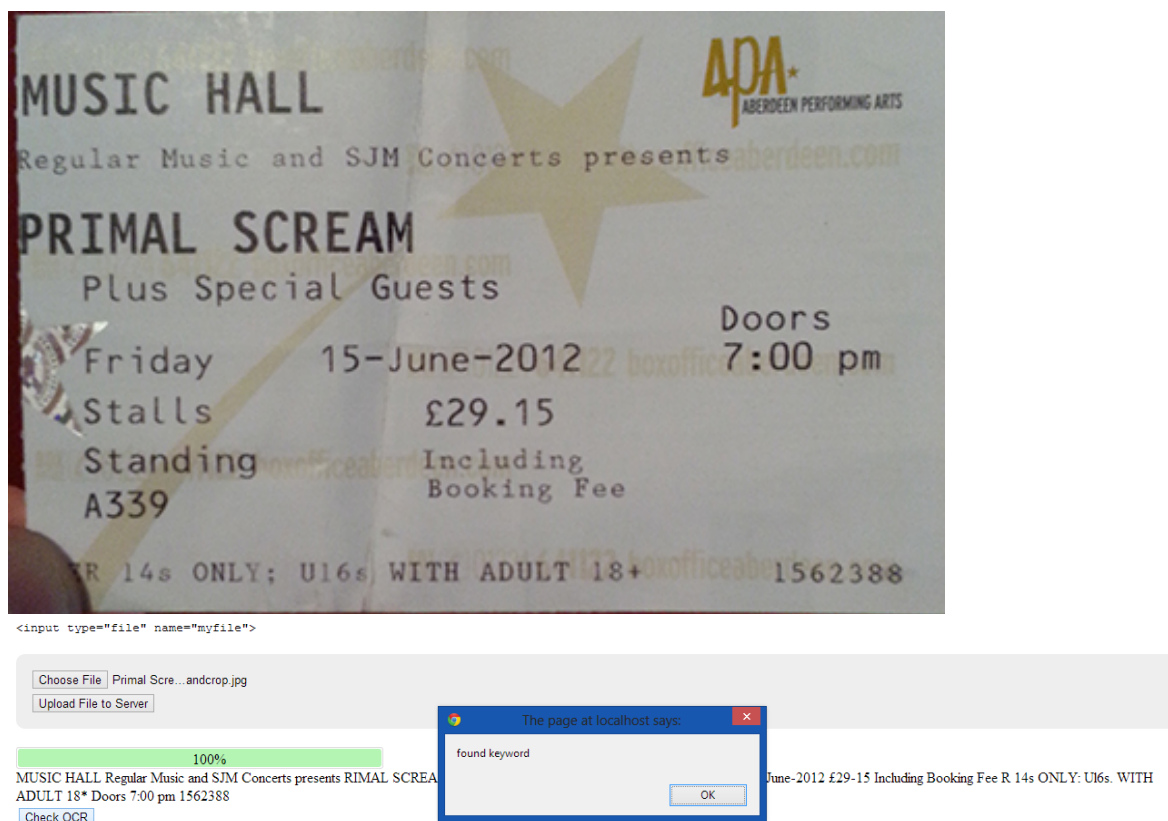


Figure 5.4 Image uploaded and text returned.

As a camera function had not been implemented yet, the test image shown in figure 5.4 had been reduced in size and had no redundant areas that were not part of the ticket information. When the camera function is built in or an image is uploaded then the images would not be edited however. Figure 5.5 shows the original image:

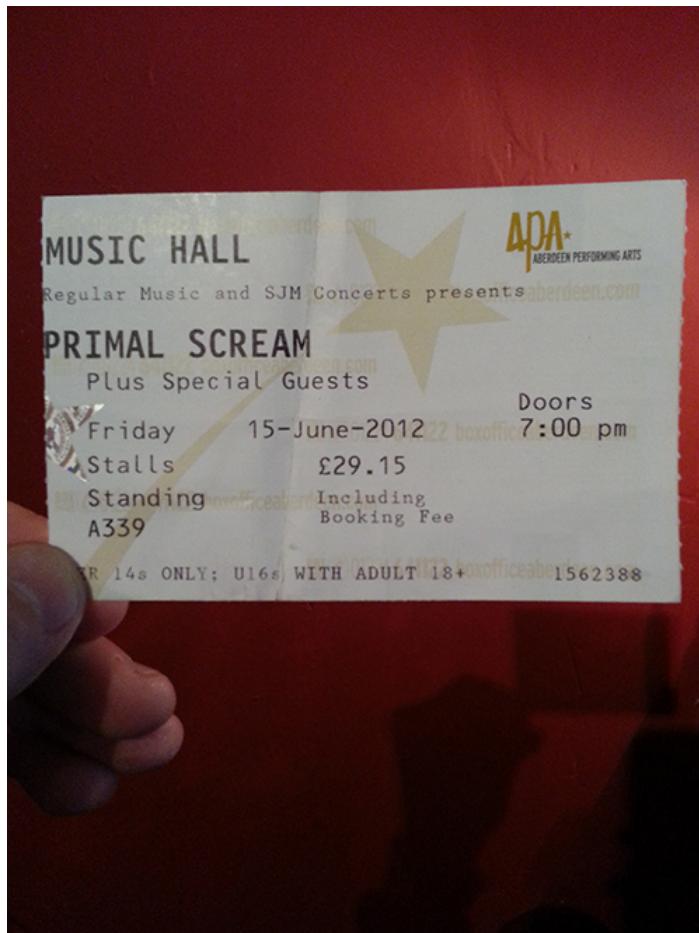


Figure 5.5 Image uploaded before editing.

When the original image shown in figure 5.5 was uploaded to the OCR, the service completed the reading of the ticket and no text was returned. The image dimensions shown in figure 5.5 were 2448 x 3264 pixels, or the equivalent of an 8 megapixel camera if the file size of this raw image was 2.28MB with a 8bit colour depth. If the application is going to be storing the images, the space required on a server or third party service would quickly expand depending on the amount of events the user has. If this data was to be stored on a third party service the hosting charges would rise considerably the more users the application has. Clearly required was a form of image processing in order to get the most amount of text from the OCR service and also to ensure the file size and dimensions were as low as it possibly could be. In order to find out what the ideal size of an image to be uploaded was and to return relevant text from the OCR an experiment was carried out.

The experiment would involve sending different sized images to the OCR service until a low file size was established that still returned the amount of text shown in figure 5.4.

Table 5.1 shows the results of what text was returned and what edits were placed to the original image shown in Figure 5.5:

The following constants remained for each treatment-

- Same image shown in Figure 5.5,
- Bit depth: 8Bit (256 colours)

Table ID: 5.1

Table Name: Effect image size has on data returned from OCR

Treatment	Image Uploaded Size	Image uploaded dimensions	Cropped around ticket?	Text returned from OCR
1	2.28MB	2448 x 3264 pixels	No	Nothing
2	1.09MB	1224 x 1632 pixels	No	Nothing
3	1.21MB	2248 x 3264 pixels	Yes	Nothing
4	0.69MB	1224 x 1632 pixels	Yes	Few words
5	0.65MB	612 x 816 pixels	No	Few words
6	0.39MB	612 x 816 pixels	Yes	Main body of ticket most details, not everything
7	0.39MB	500 x 500 pixels	No	Main body of ticket most details, not everything
8	0.12MB	500 x 500 pixels	Yes	Full information on ticket

The experiment clearly showed that the OCR API service was better at picking up information from the ticket as the dimensions of the file were reduced. As the image has redundant data removed from it by cropping the text returned from each ticket also improves.

The design stage had already considered a cropping feature for aesthetic purposes in displaying the ticket, however it seems that cropping the ticket before it is uploaded to the server or third party is now essential to ensure enough text is returned to the application from the external OCR service.

As the text returned also improved due to the size of the file, dimensions of the image were recommended to be altered by the application before uploading. The next section looks at the solutions to how the application handles images from processing to storage.

With a working image reader, attention was turned to how to extract the correct information from the ticket. Named entity recognition was considered, however the difficulty of implementing a fully optimized entity recognition system was not possible considering the time constraints of the project.

The implemented application analyses the text that is returned from the OCR into the 'status' div. If the text returned matches any of the artists or venues that are stipulated, then the elemart variable will become the artist from the image text. This artist value is then placed back onto the page where required.

```
$('#upload_form').ajaxForm({  
  beforeSend: function() {  
    status.empty();  
    var percentVal = '0%';  
    bar.width(percentVal)  
    percent.html(percentVal);  
  },  
  uploadProgress: function(event, position, total, percentComplete) {  
    var percentVal = percentComplete + '%';  
    bar.width(percentVal)  
    percent.html(percentVal);  
  },  
  complete: function(xhr) {  
    status.html(xhr.responseText);  
    if ($('#status').html().indexOf("SCREAM" || "Scream" || "scream") != -1){ (elemart.value =  
"Primal Scream");}  
    if ($('#status').html().indexOf("U2" || "u2") != -1){ (elemart.value = "U2");}  
    if ($('#status').html().indexOf("Bombay" || "BOMBAY") != -1){ (elemart.value = "Bombay  
Bicycle Club");}  
    if ($('#status').html().indexOf("BLACK KEYS" || "Black Keys") != -1){ (elemart.value = "Black  
Keys");}  
    if ($('#status').html().indexOf("MUSIC HALL" || "Music Hall") != -1){ (elemven.value =  
"Music Hall");}  
    if ($('#status').html().indexOf("WEMBLEY" || "Wembley") != -1){ (elemven.value =  
"Wembley");}
```

```

    if ($('#status').html().indexOf("HAMPDEN" || "Hampden") != -1){elemven.value =
"Hampden");}

    if ($('#status').html().indexOf("SECC" || "S.E.C.C") != -1){elemven.value = "SECC";}

}
});

```

Another consideration for the entities that the application could read from was to give the application a form of intelligence. The intention was to have the application learn itself what different types of artists and venues are and as more users create events, the amount of choice of these entities would increase.

A CouchDB Design document was attempted to create a document store of different types of entities is shown below:

```

{
  "musicartists": {
    "map": "function(doc) {\n  if(doc.artist) {\n    emit(doc.artist);\n  }\n}"
  }
}

```

The resulting output of this design document:

```

{"_id": "_design/musicartists", "_rev": "1-
269efb56269f016991ab9210ecdcd27c", "language": "javascript", "views": {"musicartists": {"map": "fun
ction(doc) {\n  if(doc.artist) {\n    emit(doc.artist);\n  }\n}"}}}

```

The application would add more artists to this document store as more were entered and the application would use this to check if they existed in the data returned from the OCR. Unfortunately, time constraints did not allow this function to be fully implemented, but will be a priority in the applications future work to be carried out.

5.1.2 Handling images

The images have a working solution to reading the ticket via OCR, but what was required was a way to get an image to the optimum size of around 500x500 pixels and cropping of the redundant parts of the photo that were not on the ticket. Section 4.4.5 deals with the design of image processing, it was recognised that there would have to be some form of file size reduction due to the large raw file sizes from camera phones. The options included using Cloud based processing through Transloadit or Cloudinary. These methods would both work, but added the fact that the images now required to be cropped before uploading, processing alone would not be enough.

The implemented function that Tickitbox uses for this is a combination of JavaScript and PHP. The ticket image has the option of being uploaded or being taken from a web cam as depicted in Figure 5.6:

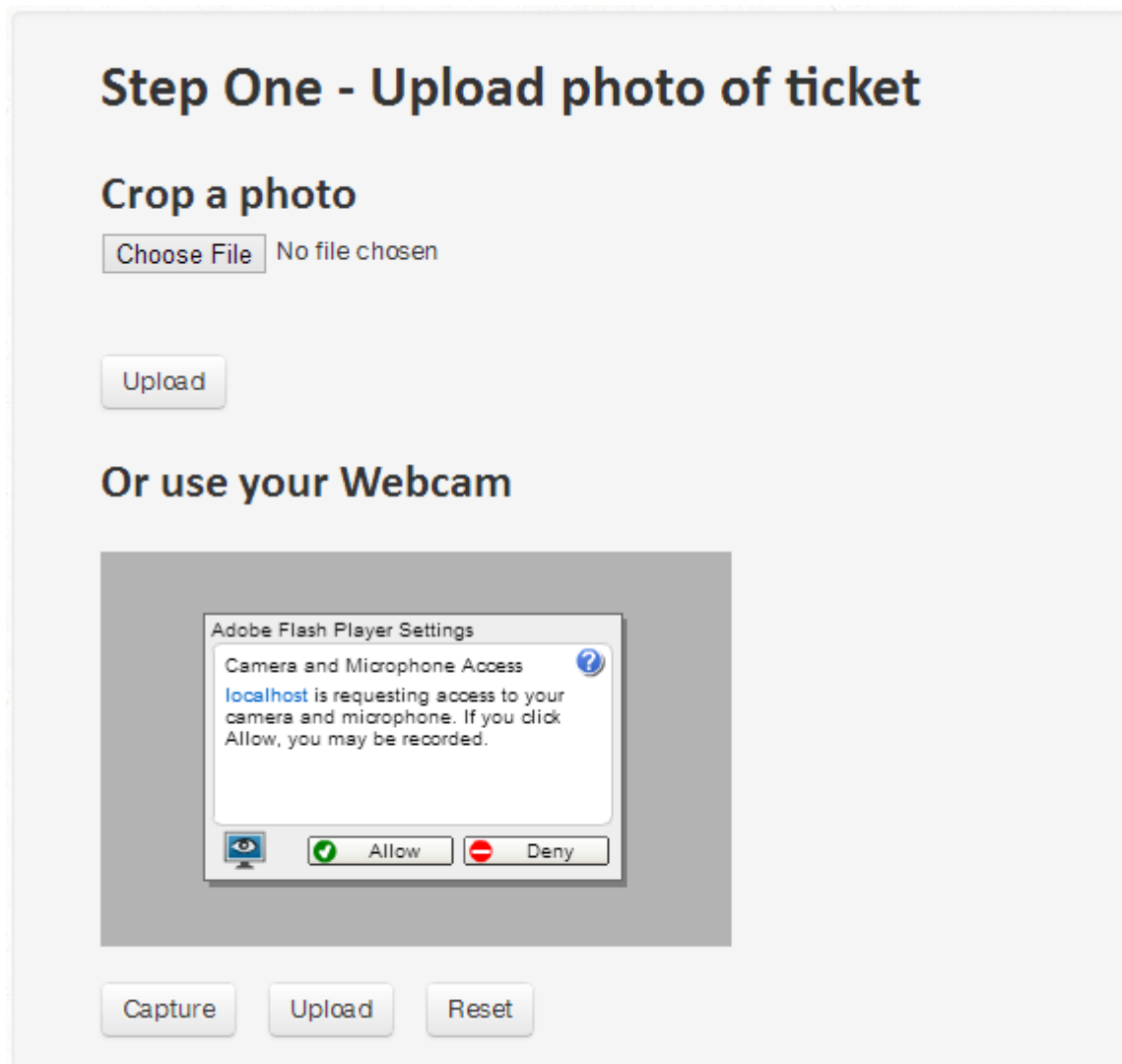


Figure 5.6 Uploading an image.

Behind the scenes of these two methods of uploads are two modified scripts, a library known as Jcrop for cropping and a script called webby.js. Both are used to power uploads, but Tickitbox takes these libraries and joins them together to make one function.

As mentioned in section 5.1.1, the dimensional size of the image and the data size are very important and the upload functions reduce the size before they are stored.

```
<script language="JavaScript">
    document.write( webcam.get_html(320, 200) );
</script>
```

The implemented webcam restricts the image sizes by defining the height and width of the webcam that will appear in the application. The webcam will be 320 x 200. Although the optimum images sizes were found to be 500x500 for the OCR, the decision was taken to further reduce the size of the image. At 320x200, there is a further increase in the data returned from the ticket and the file size is considerably less. Cloudant charges for database space in terms of amount of data used, so any reductions that will save money on database hosting without affecting the applications features are important.

For the cropping feature, the image is firstly uploaded to the screen via an HTML form:

```
<form enctype="multipart/form-data" target="AXframe" method="post"
action="http://localhost/appticko/cache/upload.php" onSubmit="return validateForm();" >
  <input type="hidden" id="w" name="w" />
  <input type="hidden" id="h" name="h" />
  <input type="hidden" id="x1" name="x1" />
  <input type="hidden" id="y1" name="y1" />
  <input type="hidden" id="x2" name="x2" />
  <input type="hidden" id="y2" name="y2" />
  <h1>Step One - Upload photo of ticket</h1><br />
  <h2>Crop a photo</h2>
  <input type="file" id="image_file" name="image_file" style="outline:none">
  <br/>
  <div class="error" style="display: none;"></div>
  <br/>
  <div id="image_div" >
    <img id="load_img" />
  </div>
  <div id="profile_pic" style="display:none;">
    <img src="" id="thumb"/>
  </div>
  <input type="submit" value="Upload" class="btn"/>
  <iframe id="AXframe" name="AXframe" frameborder="0" width="100%"
style="display:none;"></iframe>
</form>
```

The hidden elements store the points where the image will be cropped to, and the library creates a new image with these coordinates as shown in figure 5.7:

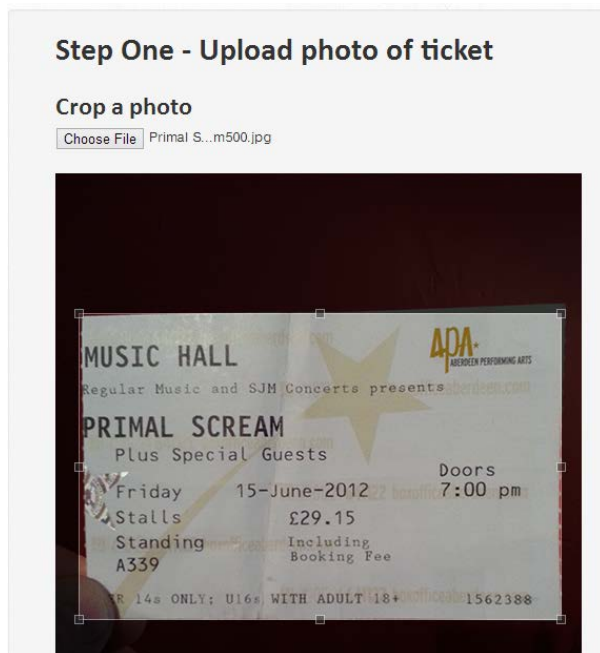


Figure 5.7 Image cropping.

The Jcrop plugin then takes over, and then saves the dimensions of the new image that is created from removing the redundant information.

```
$('#load_img').Jcrop({
  minSize: [32, 32], // min crop size
  // aspectRatio : 1, // keep aspect ratio 1:1
  bgFade: true, // use fade effect
  bgOpacity: .3, // fade opacity
  onChange: showThumbnail,
  onSelect: showThumbnail
}, function(){
  // use the Jcrop API to get the real image size
  var bounds = this.getBounds();
  boundx = bounds[0];
  boundy = bounds[1];
  // Store the Jcrop API in the jcrop_api variable
  jcrop_api = this;
```

5.1.3 Storing the images

Images that have now been processed need to be stored somewhere so they can be accessed at a later stage. Having considered Amazon S3 to store the images, a far simpler solution was to store the images within the application itself.

The solution was to not place images into the CouchDB, only the filename of the image which was randomly generated in stored on Couch. The reason for this was that when an image was being uploaded to the CouchDB was taking between 10 and 20 seconds to upload, but the final solution to the problem was instantaneous.

Upon sending the form for either the Crop or Webcam methods, the image is placed in the applications cache folder.

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
    /*Configuration part*/  
    extract($_POST);  
    $desired_width = 323; //desired image result width  
    $desired_height = 200; //desired image result height  
    $img_quality = 100;  
    $upload_dir = 'cache/';
```

The code above shows an extract from the upload.php file. Shown is the extraction of the image file to the desired sizes and the directory the files are placed in.

By storing the images on the server within the application, they are easily accessed if the filename is known. The application takes the filename and the directory and stores that in the CouchDB

5.1.4 Storing data

The final part of the Create an Event component is the storage to the database. The form shown in figure 5.8 includes the Type, Artist, Venue, Date and the event memory as well as a hidden element of the image location in text form.

Fill in the event details below:

Type:

Artist:

Venue:

Date:

What do you remember about this event?

Figure 5.8 Create event store

Below shows the opening HTML line of the final form:

```
<form class="form-horizontal" id="eventHere" action="<?php echo $this->make_route('/post')?>"
method="post" enctype="multipart/form-data">
```

The PHP action to the route ('/post') sends the information to the receiver on the index of the site:

```
post('/post', function($app) {
    if (User::is_authenticated()) {
        $post = new Post();
        $post->cacheimage = $app->form('cacheimage');
        $post->artist = $app->form('artist');
        $post->venue = $app->form('venue');
        $post->eventtype = $app->form('eventtype');
        $post->memory = $app->form('memory');
        $post->eventday = $app->form('eventday');
        $post->eventmonth = $app->form('eventmonth');
        $post->eventyear = $app->form('eventyear');
        $post->create();
        $app->set('success', 'Your event has been created');
        $app->redirect('/user/' . User::current_user());
    }
})
```

```

    } else {
        $app->set('error', 'You must be logged in to do that.');
```

```
    }
});
```

The function above is creating a new document on the CouchDB. It takes the form id's for example artist and creates a new variable for the database.

The post class recognises these variables created:

```

<?php
class Post extends Base{
    protected $date_created;
    protected $cacheimage;

    protected $artist;
    protected $venue;
    protected $memory;
    protected $eventtype;
    protected $eventday;
    protected $eventmonth;
    protected $eventyear;
    protected $user;
```

The final step is to send the information to the CouchDB and create a new document as shown in

Figure 5.9:

```

$post = new Post();
    $post->_id = $_post->id;
    $post->_rev = $_post->value->_rev;
    $post->date_created = $_post->value->date_created;
    $post->memory = $_post->value->memory;
    $post->eventtype = $_post->value->eventtype;
    $post->artist = $_post->value->artist;
    $post->venue = $_post->value->venue;
    $post->eventday = $_post->value->eventday;
    $post->eventmonth = $_post->value->eventmonth;
    $post->eventyear = $_post->value->eventyear;
```

```
$post->user = $_post->value->user;
```

```
array_push($posts, $post);
```

Overview > verge > 3a0e06bf3811980ab6f33f594000566	
✓ Save Document + Add Field ⬇ Upload Attachment... ✕ Delete Document...	
Field	Value
_id	"3a0e06bf3811980ab6f33f594000566"
_rev	"1-0893f61be9a14cc3e498be209de443e3"
✕ artist	"Primal Scream"
✕ cacheimage	null
✕ date_created	"Sat, 06 Apr 2013 16:16:34 +0000"
✕ eventday	"17"
✕ eventmonth	"04"
✕ eventtype	"Music"
✕ eventyear	"2013"
✕ memory	""
✕ rating	null
✕ type	"post"
✕ user	"innesm"
✕ venue	"Music Hall"

Figure 5.9 Documents in the Futon of CouchDB

5.2 Viewing an event

In order to view an event created with the application, the user must navigate to their Box.

5.2.1 My Box

Figure 5.9 shows the implemented box area. As designed, the box will show all the tickets that the user has in the box, and has an option to create more. The event can be viewed by clicking the View Event Button or deleted by selecting delete.

mark's Box Signed in

Create a New Event

MUSIC HALL

Regular Basis and SIM Concerts presents

PRIMAL SCREAM

Plus Special Guests

Friday 15-June-2012

Stalls £29.15

Standing A359

including Booking Fee

1562388

Artist: Primal Scream

Venue: Music Hall

Date: 15-06-2012

View Event

(Delete)

Load More...

Figure 5.9 My Box

Functions created to display the events a user has is made possible by what is called a map/reduce function of CouchDB. A map/reduce function is what could be described as querying the database.

The queries are created within the Futon of CouchDB and then saved as a design document.

Ticketbox then pulls the information from that specific design document.

Get_posts_by_user is the name of this design document hosted on Couch and is shown below:

```
posts_by_user
map
function(doc) {
  if (doc.type == 'post')
    emit(doc.user, doc);
}
reduce_count
```

This function created finds all the documents in the database that are of the type 'post'. The emit part simply means that whatever is placed in the emit function will be displayed. The information displayed in this function is all of the posts by a user. The reduce function simply counts how many of these cases exist.

This design document is now given the URL of:

http://localhost:5984/verge/_design/application/_view/posts_by_user

Which outputs:

```
{"rows":[
  {"key":null,"value":1}
]}
```

As there is only one event in the entire application, the JSON returned show a value of 1. The next part of this function that was created was the method to get a specific users events, and place that back onto the box's page.

```
public function get_post_count_by_user($username) {
  $bones = new Bones();
  $rows = $bones->couch->get('_design/application/_view/posts_by_user?key="'. $username .
  '"&reduce=true')->body->rows;
  if ($rows) {
    return $rows[0]->value;
  } else {
```

```

    return 0;
}
}

```

This function `get_post_count_by_user` both retrieves specific users events and counts the amount of events they have. The information is now placed back into the My Box view:

```

<h2>My Events (<span id="post_count"><?php echo $post_count; ?></span></h2>
<div id="post_list">
    <?php include('_posts.php'); ?>
</div>
<div id="load_more" >
    <a id="more_posts" href="#">Load More...</a>
</div>
</div>

```

The PHP include for the `'_posts.php'` contains all of the event information as a template for each event to use and uses `<?php foreach ($posts as $post): ?>` to cycle through each event the user has.

The information displayed is simply pulled from the database document. For that event the images location is placed into the `` tag and is displayed below:

```

<strong></strong>
Artist: <strong><?php echo $post->artist; ?></strong>

```

5.2.2 The event

In order to move to the specific events page to see the data returned from the API's, the link had to also be dynamically placed into the box:

```

<a href="<?php echo $this->make_route('/user/' . User::current_user() . '/posts/' . $post->_id) ?>">
<button class="btn btn-success btn-large">View Event</button></a>

```

This pastes the button view event to the view `/user/currentuser/posts/postid`. In order to get the information from the event, the post id from the URL is grabbed by JavaScript, and the CouchDB is then queried for that document id:

```
// this will give you your path ie /light/saber/
urlPath=window.location.pathname;
//this splits your url in pieces using the / as a separator
urlPathArray = urlPath.split('/');
//get the first section of your path ie light
urlPath1=urlPathArray[5];
var jsid = urlPath1;
```

Variable jsid is the event id from the View Event function. URL is split into 6 sections and urlPathArray[5]; and returns that event id to the page.

Now with this event id, the events information is retrieved with JavaScript:

```
$(document).ready(function()
{$("#target").html("");

// GET INFO FROM COUCH //

var cd_url='http://127.0.0.1:5984/verge/'+jsid+'?callback=?';
$.ajax({
  type: "GET",
  url: cd_url,
  dataType: "jsonp",
  success: function(data){
var cachediv=data.cacheimage;
var artistdiv=data.artist;
var venuediv=data.venue;
var memorydiv=data.memory;
var eventdaydiv=data.eventday;
var eventmonthdiv=data.eventmonth;
var eventyeardiv=data.eventyear;
var eventtypediv=data.eventtype;
$("#target1").html(artistdiv);
$("#target2").html(venuediv);
$("#target3").append("<p>" +eventdaydiv+"-"+eventmonthdiv+"-"+eventyeardiv+"</p>");
$("#target4").append("<img src='http://localhost/appticko/cache/'+cachediv+'">");
```

So the information from that specific events id is posted back into the page and this view is shown in Figure 5.10:

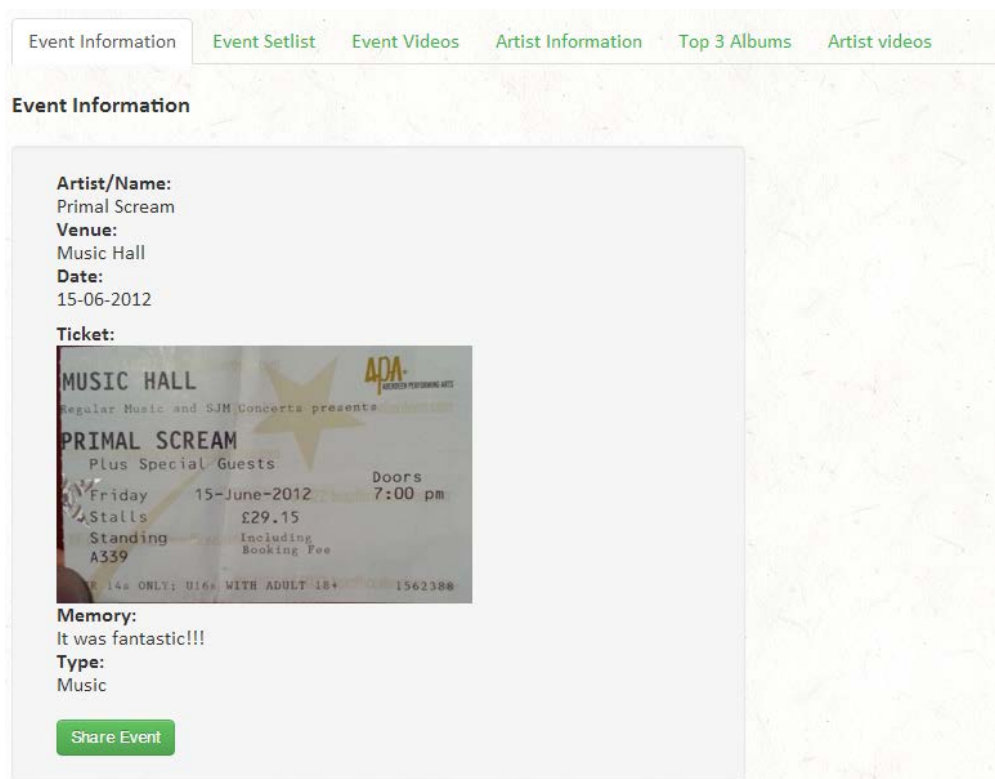


Figure 5.10 View Event

Having successfully retrieved the information from the database, the API's that could use this information to retrieve data were then implemented.

API's for the View event components are listed below with a screenshot of the working function, a sample snippet of code and an explanation of the API.

Event Setlist

The event setlist function used the Setlist.fm API to return data about an event as shown in figure 5.11:

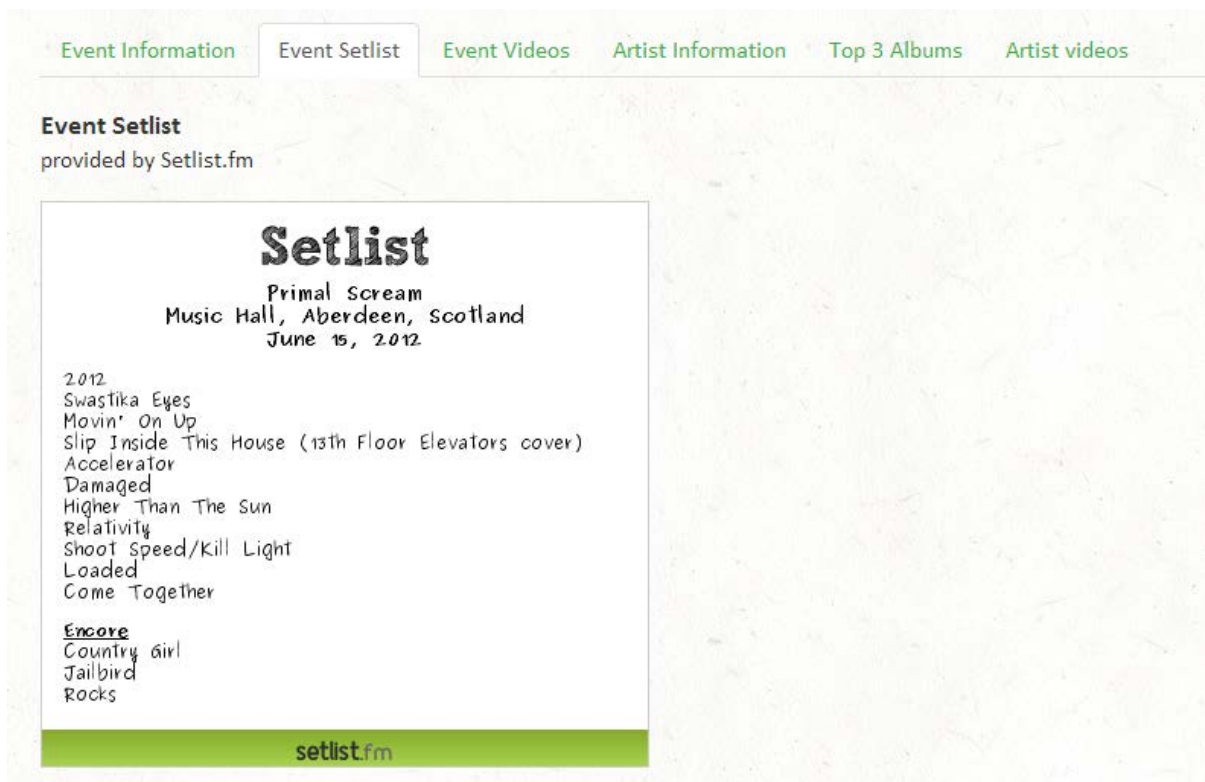


Figure 5.11 Event setlist

Code:

```
// GET INFO FROM SETLIST//  
  
var  
ss_url='http://api2.setlist.fm/rest/0.1/search/setlists.json?&artistName='+artistdiv+'&venueName='  
+venuediv+'&date='+eventdaydiv+'-'+eventmonthdiv+'-'+eventyeardiv+'&callback=?';  
$.ajax({  
  type: "GET",  
  url: ss_url,  
  dataType: "jsonp",  
  success: function(data)  
  {  
    var song1 =(JSON.stringify(data.setlists.setlist["@id"]));  
    var resultmbid =data.setlists.setlist.artist["@mbid"];  
    var result = $.parseJSON(song1);
```



```

var lasteid = data.setlists.setlist["@lastFmEventId"];
$('#setlist2').append("<iframe width='400' height='500' src='http://www.setlist.fm/widgets/setlist-image-v1?id="+result+"&fg=000000' frameborder='0' scrolling='no' type='text/html'></iframe>")
// END SETLIST//

```

The setlist.fm API is called by placing the events artist, venue and date into the URL and creating an AJAX GET request and returning JSON data. The data is then parsed before being placed back into the page.

Event Videos

The other API selected to be shown is the function that returns videos and data about an event from YouTube as shown in figure 5.12:

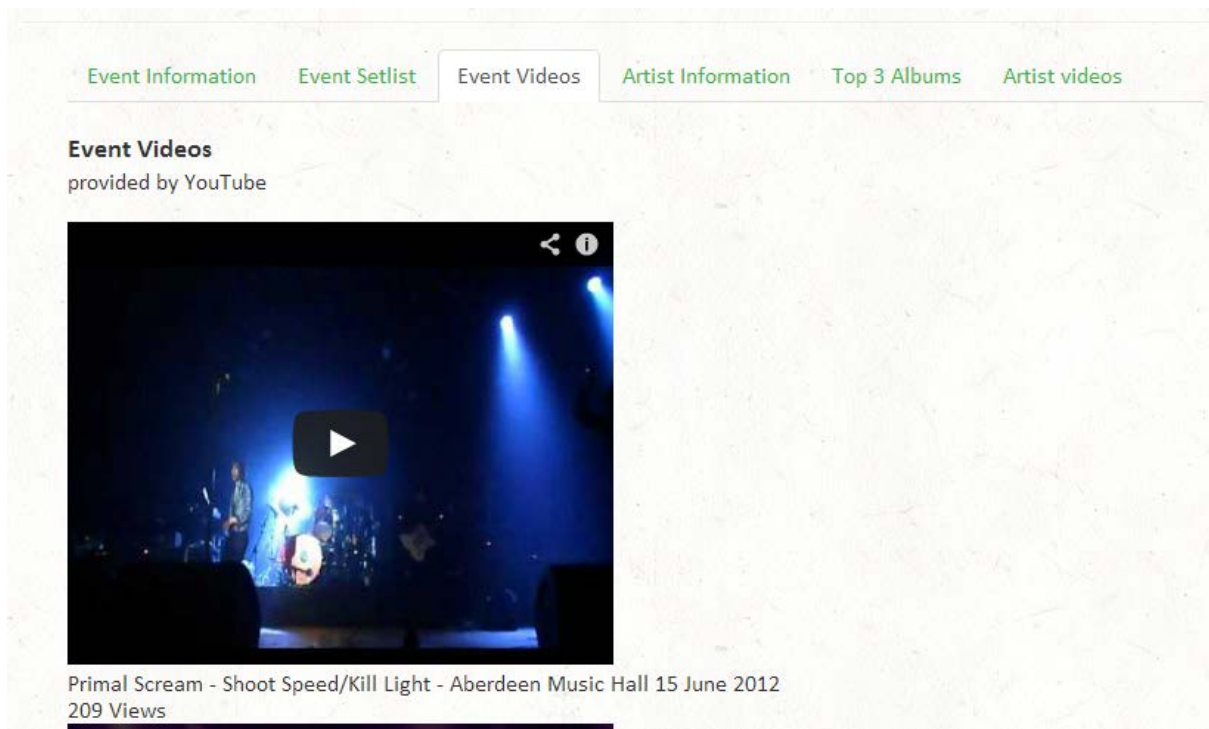


Figure 5.11 Event Videos

Code:

```

// GET INFO FROM YOUTUBE//
$("#video").html("");
var yt_url='http://gdata.youtube.com/feeds/api/videos?q='+artistdiv+'
'+venueid+'&format=5&max-results=2&orderby=relevance&v=2&alt=jsonc';

```

```

$.ajax({
  type: "GET",
  url: yt_url,
  dataType: "jsonp",
  success: function(response){
    if(response.data.items){
      $.each(response.data.items, function(i,data){
        var video_id=data.id;
        var video_title=data.title;
        var video_viewCount=data.viewCount;
        var video_frame="<iframe width='370' height='300'
src='http://www.youtube.com/embed/'+video_id+' frameborder='0' type='text/html'></iframe>";
        var final="<div id='resultvideo'>"+video_frame+"<div id='title'>"+video_title+"</div><div
id='count'>"+video_viewCount+" Views</div></div>";
        $("#video").append(final);});
    }
  }
// END YOUTUBE //

```

The Youtube API is called by placing the events artist and venue into the URL and creating an AJAX GET request and returning JSON data. The date is not included into this, as the API returns the newest videos and most relevant videos.

5.3 User Management

The user management system is controlled in a similar way to the creating an event. A form for signing up to Tickitbox is presented to the user and the data is posted to the `_users` database rather than the Tickitbox events database

```

<?php
class User extends Base
{
    protected $name;
    protected $email;
    protected $full_name;
    protected $salt;

```

```
protected $password_sha;
```

```
protected $roles;
```

The user class defines the attributes that the signup form will pass to it.

```
public function signup($username, $password) {  
    $bones = new Bones();  
    $bones->couch->setDatabase('_users');  
    $bones->couch->login($bones->config->db_admin_user, $bones->config->db_admin_password);  
    $this->roles = array();  
    $this->name = preg_replace('/[^a-z0-9-]/', '', strtolower($username));  
    $this->_id = 'org.couchdb.user:' . $this->name;  
    $this->salt = $bones->couch->generateIDs(1)->body->uuids[0];  
    $this->password_sha = sha1($password . $this->salt);
```

The signup function then passes the form variables to the _users database on CouchDB:

```
post('/signup', function($app) {  
    $user = new User();  
    $user->full_name = $app->form('full_name');  
    $user->email = $app->form('email');  
    $user->signup($app->form('username'), $app->form('password'));  
    $app->set('success', 'Thanks for Signing Up ' . $user->full_name . '!');  
    $app->render('user/login');  
});
```

This is the basic mechanics of how information is stored in the database. A user will enter information into a form, the forms action will send to the index page where defined posting methods are waiting, before the class is called and deliver the information to the database. `$app->render('user/login');` tells the application display the login form when the new user has been created.

Similarly if the login button is clicked anywhere on the site it will perform the following:

```
get('/login', function($app) {  
    $app->render('user/login');  
});
```

Now the user is given the presentation layer of the user/login view as shown in Figure 5.12:

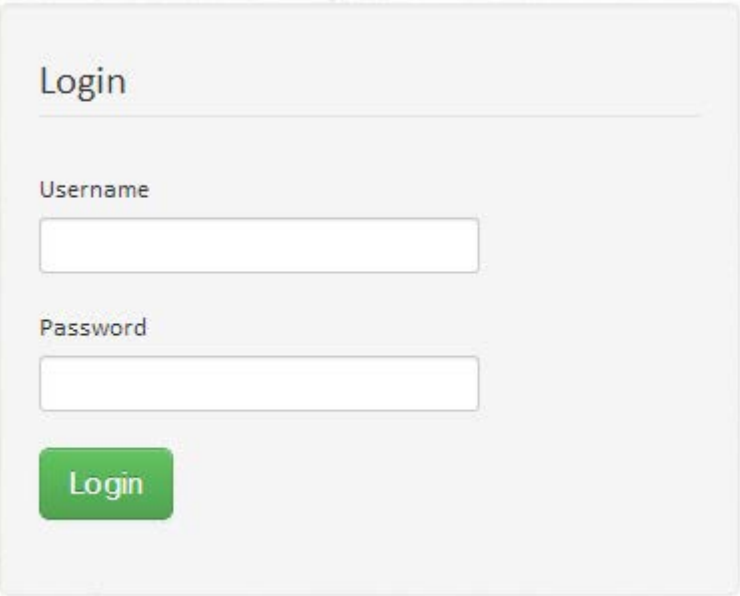
The image shows a login form with a light gray background. At the top, the word 'Login' is displayed in a dark gray font. Below it, there are two input fields: one for 'Username' and one for 'Password'. Both fields are white with a thin gray border. Below the password field is a green button with the word 'Login' in white text.

Figure 5.12 Logging in

The HTML form displaying this login is shown below:

```
<legend>Login</legend>
<form id="loginHere" action="<?php echo $this->make_route('/login') ?>" method="post">
  <fieldset>
    <div class="control-group">
      <label class="control-label" for="input01">Username</label>
      <div class="controls">
        <?php Bootstrap::make_input('username', 'Username', 'text'); ?>
      </div>
    </div>
    <div class="control-group">
      <label class="control-label" for="input01">Password</label>
      <div class="controls">
        <?php Bootstrap::make_input('password', 'Password', 'password'); ?>
      </div>
    </div>
    <button class="btn btn-success btn-large">Login</button>
  </fieldset>
</form>
```

Again on the forms action the route is set and corresponds to the post route of /login:

```
post('/login', function($app) {  
    $user = new User();  
    $user->name = $app->form('username');  
    $user->login($app->form('password'));  
    $app->set('success', 'You are now logged in!');  
    $app->render('welcome');  
});
```

The users details are retrieved from the database and the welcome page is shown. Now a user is logged in, this would be a good time to explain how Tickitbox handles users being logged in or out or what could be described as session management.

The application uses a library client especially created for connections to CouchDB and handling these types of functional issues that the application has to deal with. The name of the library is SAG for CouchDB and handles all of the applications required connections. An example of this shown below is an extract of the code from the Sag.php library:

```
/**  
 * Get current session information on the server with /_session.  
 *  
 * @return stdClass  
 */  
  
public function getSession() {  
    return $this->procPacket('GET', '/_session');  
}  
  
* @param bool $decode True to decode, false to not decode.  
* @return Sag Returns $this.
```

The code has excellent documentation, and with heavily commented code it became easy to interact with the required elements of the application. This library was integrated with the bones framework and handles session management for the application. During the implementation, all that was required was to link the sag library into the application and match the variables of the framework with the required variables of the library that the Sag documentation required.

As mentioned before, when a user is logged in, they are presented with a welcome screen into the application as shown in Figure 5.13 below:

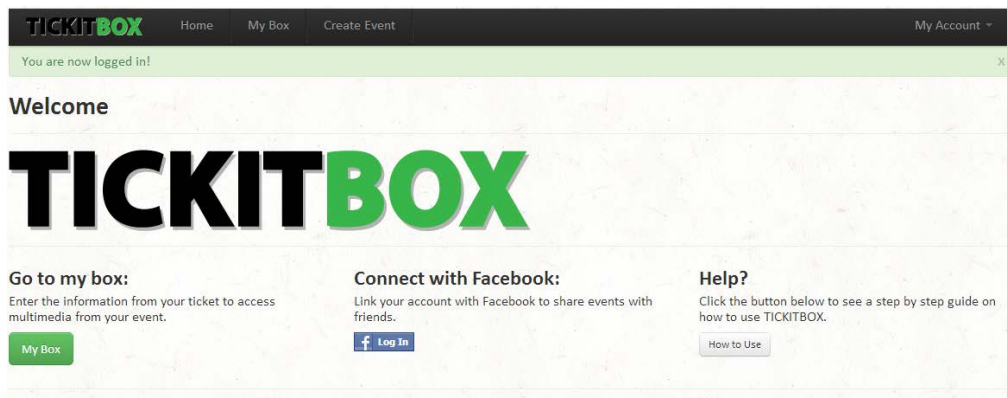


Figure 5.13 Welcome Screen

This is only available to users who are logged in and are presented with the full navigation and the option to be taken to their box. As the Sag client registers that a user is logged in, this effects which buttons are displayed in the navigation. Without going into great detail about the framework, as that will be spoken about in section 5.4, the applications navigation is predefined depending on the user being logged in or out.

```
<div class="nav-collapse collapse">
  <ul class="nav">
    <li><a href="<?php echo $this->make_route('/') ?>"><i class="icon-
home"></i>Home</a></li>
    <?php if (User::current_user()) { ?>
    <li class="divider-vertical"></li>
    <li><a href="<?php echo $this->make_route('/user/' . User::current_user()) ?>">My
Box</a></li>
    <li class="divider-vertical"></li>
    <li><a href="<?php echo $this->make_route('/event/' . User::current_user()) ?>">Create
Event</a></li>
```

The layout.php of the site is where the navigation is determined. The `if (User::current_user())` allows the application to display the My Box and Create event navigation if the user is logged in. The Sag client handles the sessions entirely with this current user variable. When an event is created, the current user variable is placed into the username part of the database. Having already discussed

how the application retrieves specific user events earlier in this section, another function exists to get the users profile displayed in this login system:

```
function get_user_profile($app) {
    $app->set('user', User::get_by_username($app->request('username')));
    $app->set('is_current_user', ($app->request('username') == User::current_user() ? true : false));
    $app->set('posts', Post::get_posts_by_user($app->request('username'), ($app->request('skip') ?
app->request('skip') : 0)));
    $app->set('post_count', Post::get_post_count_by_user($app->request('username')));
}
```

The application checks to see that the user is logged in via the current user variable and displays the information.

5.4 Application functionality

This section will discuss the applications functionality and explain how the framework manages the application. This section will also include how the important libraries included have allowed Tikitbox to become a working application.











 cache	15/04/2013 20:21	File folder	
 classes	25/03/2013 10:43	File folder	
 lib	24/03/2013 23:05	File folder	
 public	24/03/2013 23:05	File folder	
 views	25/03/2013 17:15	File folder	
 .htaccess	07/03/2013 13:40	HTACCESS File	1 KB
 download.php	25/03/2013 00:31	PHP Script	4 KB
 index.php	25/03/2013 10:43	PHP Script	4 KB
 shutter.mp3	16/03/2008 14:28	MPEG Layer 3 Aud...	14 KB
 webcam.swf	05/06/2010 21:56	SWF File	6 KB

Figure 5.14 Tikitbox folder structure

The bones framework used separates the logic from the presentation of the application by using the index.php as a hub for controlling every aspect of the application. Some of the functions index.php performs have been covered already in this chapter however the index file will now be given more detail explanation.

```
<?php
include 'lib/bones.php';
```

The first line of the index.php is to include the bones.php file. The bones.php is the library that essentially powers this framework. Here all of the major functions that allow the framework to make get and post requests, define the documents or declare global variables. Not many modifications were made to this file bar error handling and changing key variables but there were important additions to the file as shown below:

```
<?php
define('ROOT', __DIR__ . '/../');
require_once ROOT . '/lib/bootstrap.php';
require_once ROOT . '/lib/sag/src/Sag.php';
require_once ROOT . '/lib/configuration.php';
```

The additional libraries added were the Sag client, a bootstrap route and the database configuration. These libraries extended the bones framework in order to manage sessions and communicate with the CouchDB. The Sag client has already been discussed in section 5.3 and explained why it is required so there is no need to explain the library again.

The file configuration.php file is used to create a new class of Configuration. This class contains all of the variables that connect the application with the database:

```
<?php
class Configuration {
    private $db_server = '127.0.0.1';
    private $db_port = '5984';
    private $db_database = 'database';
    private $db_admin_user = 'username';
    private $db_admin_password = 'password';
    public function __get($property) {
        if (getenv($property)) {
            return getenv($property);
        } else {
            return $this->$property;
        }
    }
}
```


In the bones.php file, sessions are created with the Sag library using the following code:

```
public function __construct() {  
    $this->route = $this->get_route();  
    $this->route_segments = explode('/', trim($this->route, '/'));  
    $this->method = $this->get_method();  
    session_start();  
    $this->config = new Configuration();  
    $this->couch = new Sag($this->config->db_server, $this->config->db_port);  
    $this->couch->setDatabase($this->config->db_database);  
}
```

In order to achieve the addresses of <http://localhost/ticketbox/> or <http://localhost/ticketbox/login> as oppose to <http://localhost/ticketbox/index.php> or <http://localhost/ticketbox/login.php> there were modifications made to the hta.access file of the application:

```
<IfModule mod_rewrite.c>  
    RewriteEngine On  
    RewriteCond %{REQUEST_FILENAME} !-f  
    RewriteCond %{REQUEST_FILENAME} !-d  
    RewriteRule ^css/([^\s/]+) public/css/$1 [L]  
    RewriteRule ^js/([^\s/]+) public/js/$1 [L]  
    RewriteCond %{REQUEST_FILENAME} !-f  
    RewriteCond %{REQUEST_FILENAME} !-d  
    RewriteRule ^(\.*)$ index.php?request=$1 [QSA,L]  
</IfModule>
```

The code shown will remove index.php from each URL request made and this allows the applications routes to be defined.

The application is also connected to Facebook using a JavaScript SDK:

```
window.fbAsyncInit = function() {  
    FB.init({appId: '419639651461622', status: true, cookie: true, xfbml: true});  
    /* All the events registered */  
    FB.Event.subscribe('auth.login', function(response) {  
        // do something with response  
    })  
}
```

```

    login();
  });
  FB.Event.subscribe('auth.logout', function(response) {
    // do something with response
    logout();
  });
  FB.getLoginStatus(function(response) {
    if (response.session) {
      // logged in and connected user, someone you know
      login();
    }
  });
};

```

The application uses a Facebook connect login and allows sharing of events as shown in Figure 5.15

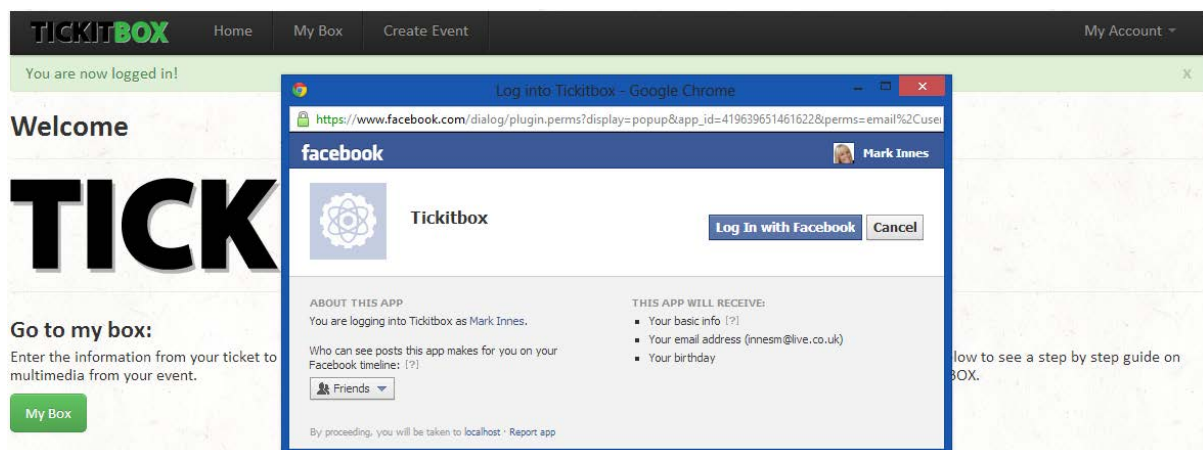


Figure 5.15 Facebook connect

The application is firstly created on Facebook and the applications ID is matched up in the Id placed in the code of Tickitbox.

This completes the functionality of the application. The most complex and important to the applications solutions have been covered and the chapter will now move onto Tickitbox's user interface design.

5.5 User interface design

The user interface of the application was created with responsive web design in mind. This section will look at how responsive design was implemented before looking at ways the user experience was improved by improving the interface.

The Twitter bootstrap was used to create a responsive site. In figure 5.16 the homepage is shown, but in figure 5.17 a mobile version is shown. The bootstrap creates a grid system for the user interface and depending on the width of the screen, it reacts to that size.

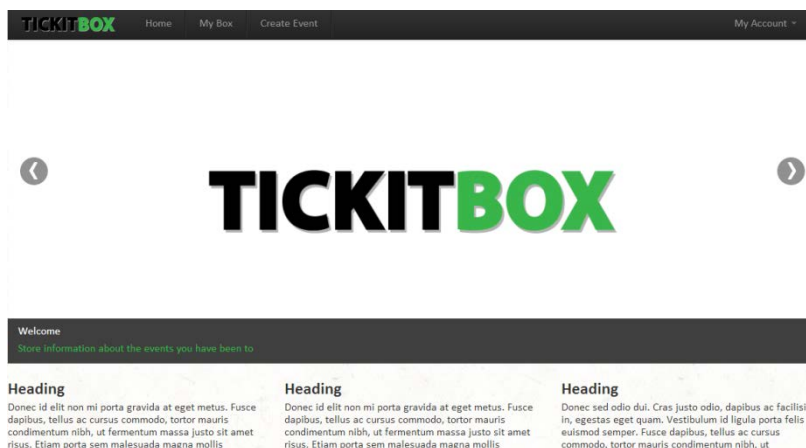


Figure 5.16 Tickitbox Homepage

The page features the navigation, a slideshow and areas for content at the bottom.

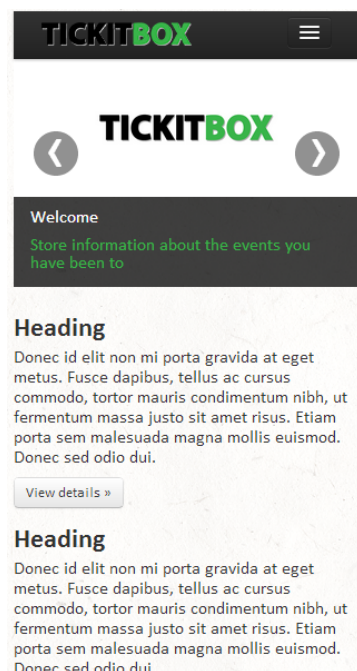


Figure 5.17 Tickitbox homepage at mobile screen width

Tickitbox was branded with the colours shown in figure 5.18:



Figure 5.18 Logo and icon

Features of Tickitbox tries to make the experience as easy for the user to navigate to where they want to go as easily as possible, and where information is required pre-emptive design decisions have been made to improve the users experience.

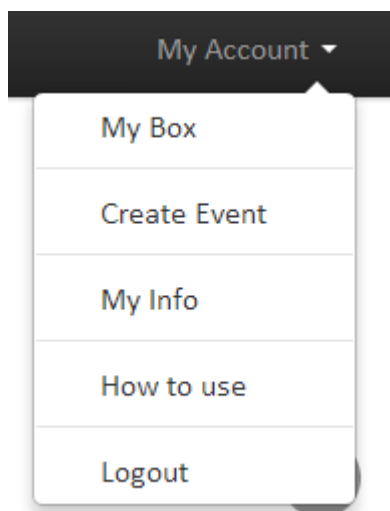


Figure 5.19 Drop down menu of the options for the user.

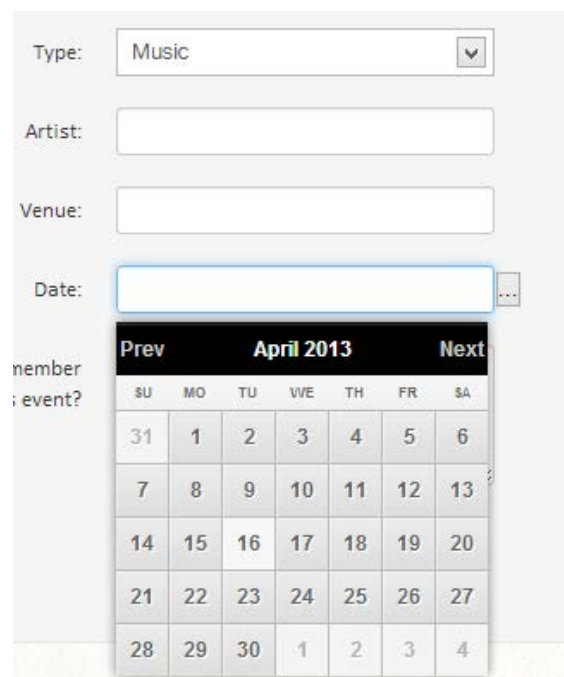


Figure 5.20 Datepicker implemented

The screenshot shows a sign-up form for 'TICKITBOX'. The form has five input fields: Name, Email, Username, Password, and Confirm Password. The Name field contains the letter 'f'. The Email field also contains 'f', and below it is a red error message: 'Enter a valid email address'. The Username field is empty. The Password field contains a single dot, and below it is a red error message: 'Password must be minimum 6 characters'. The Confirm Password field also contains a single dot, and below it is a red error message: 'Password and Confirm Password must match'. At the bottom of the form is a green 'Sign Up' button.

Figure 5.21 Form validation

Figure 5.19 shows the drop down menu on the logged in navigation, this was implemented to de-clutter the navigation bar.

Figure 5.20 shows the datepicker implemented in the create event form. This was used to speed up the process of the user selecting a date

Figure 5.21 shows the form validation that is in place within the application. This is used to quickly inform the user that the information they have entered is not correct.

5.6 Security

Security is very important to the application's solution and because Tickitbox stores user passwords, the user would expect their data and details would have some form of protection.

The application has some created security features and also takes steps to ensure user information is secure. One of the ways Tickitbox protects its user information is by Salt password encryption.

<?php

```
class User extends Base
```

```

{
    protected $name;
    protected $email;
    protected $full_name;
    protected $salt;
    protected $password_sha;
    protected $roles;
    public function __construct()
    {
        parent::__construct('user');
    } public function signup($username, $password) {
        $bones = new Bones();
        $bones->couch->setDatabase('_users');
        $bones->couch->login($bones->config->db_admin_user, $bones->config->db_admin_password);
        $this->roles = array();
        $this->name = preg_replace('/[^a-z0-9-]/', '', strtolower($username));
        $this->_id = 'org.couchdb.user:' . $this->name;
        $this->salt = $bones->couch->generateIDs(1)->body->uuids[0];
        $this->password_sha = sha1($password . $this->salt);
    }
}

```

The sag client has built in salt encryption, and handled the process. The calls to the Bones function allow the salt encryption to take place. The password_sha is the SHA-1 encrypted value after it has been combined with the salt. A salt is a random sting of data.

SHA-1 is a hash function created by the National Security Agency (NSA) that combines the salt string with a password to make it unreadable.

CouchDB has its own form of security. The server admin is defined and therefore the only user who can update the documents of the database without use of the application is the Administrator of the database. New admins can be added to gain the same privileges as the main Administrator, however ever other user that sign up for the application only the functions that are available as a result of the application. If a user tried to login to the CouchDB Futon, they would be able to gain access to their data, but they would not be able to make any changes to in from within the Futon.

The sag client also has preventive measures to protect unauthorised access to important files or interfering with the applications data.

```

try {
    $document = $bones->couch->get('org.couchdb.user:' . $username)->body;
    $user->_id = $document->_id;
    $user->name = $document->name;
    $user->email = $document->email;
    $user->full_name = $document->full_name;
    return $user;
} catch (SagCouchException $e) {
    if($e->getCode() == "404") {
        $bones->error404();
    } else {
        $bones->error500();
    }
}
}

```

If any part of the application is attempted to be accessed that is not allowed, the Sag client will catch the errors and display error messages as shown in Figure 5.22

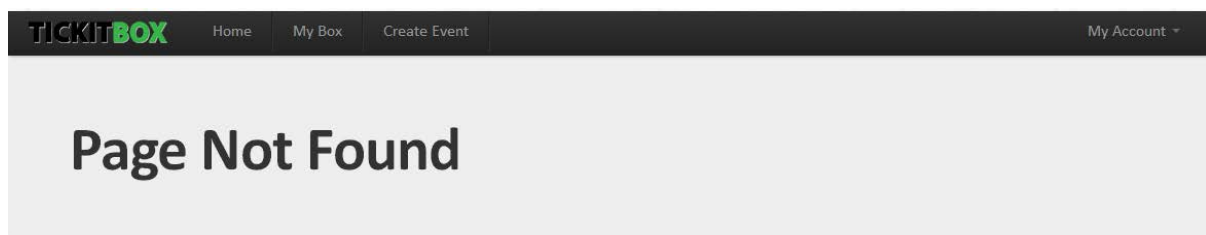


Figure 5.22 Page not found

5.7 In summary

This chapter has covered the most complex and import aspects that define the created application. The main components of the application have been included in detail on how they work. Ticketbox will now be evaluated for quality, quality user experience and performance of the components that characterize the implemented solution.

6 Testing and Evaluation

Having developed the Tickitbox application, testing and evaluations were both important to ensure the technical aspect of the system were to a high standard. Evaluations were carried out to find how well the Tickitbox application solves the problems acknowledged in chapter 3. Evaluations and testing were carried out on the following criteria, the applications features, User experience, Performance and the Quality of the application.

6.1 Testing

The testing goals of the Tickitbox application are to make sure the developed application meets its requirement's created in section 3.6 and to discover any faults that may exist in the applications code.

In order to establish if the application has managed to meet the requirements by testing, the requirements were placed into scenario-based tests. Scenario-based testing involves groups of features or functions that are tested all at once, rather than testing each individual line of code. Each test was conducted with the intention of having as many requirements' included as possible. Each test will have an ID, description, ID of requirements tested and a result of the test. Although the application has obviously been tested during the development of the application, a formal testing method was required before any users came in contact with the application. The scenarios tested will include the major applications function and the testing results are shown below:

Scenario testing

Table Name: Scenario ST1

| | |
|---------------------------------|--|
| Test ID: | ST1 |
| Test Description: | A new user sign-up for Tickitbox |
| Requirement's satisfied: | 1.1, 1.2, 1.3, 1.4, 2.0, 2.1 |
| Test Results: | As expected, in-line with test description |
| Test Errors: | None |

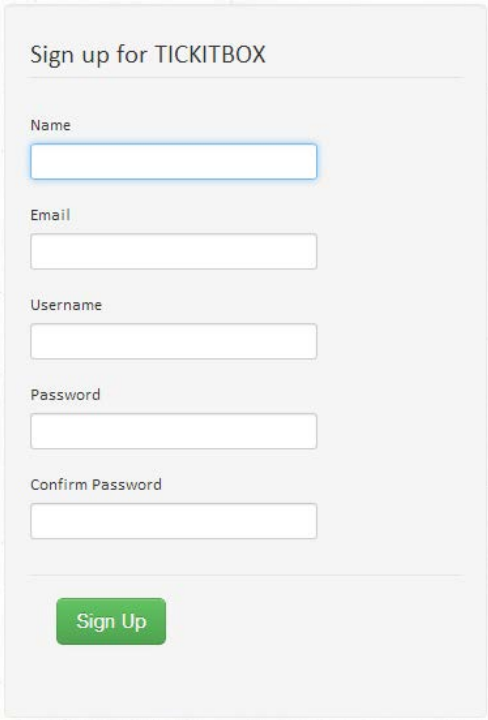
| | |
|--------------------|---|
| Screenshot: |  |
|--------------------|---|

Table Name: Scenario ST2

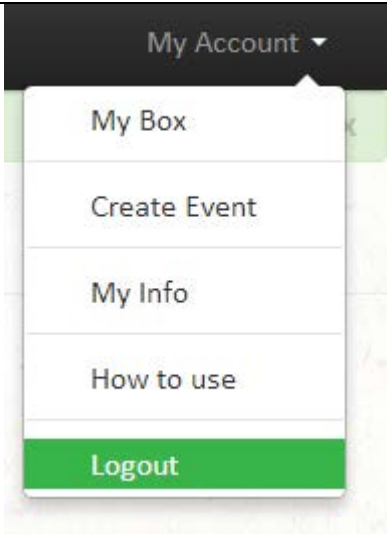
| | |
|---------------------------------|---|
| Test ID: | ST2 |
| Test Description: | A signed up user can login then logout |
| Requirement's satisfied: | 1.0, 1.2, 1.3, 1.5, 2.0 |
| Test Results: | A signed up user can login then logout. |
| Test Errors: | None |
| Screenshot: |  |

Table Name: Scenario ST3

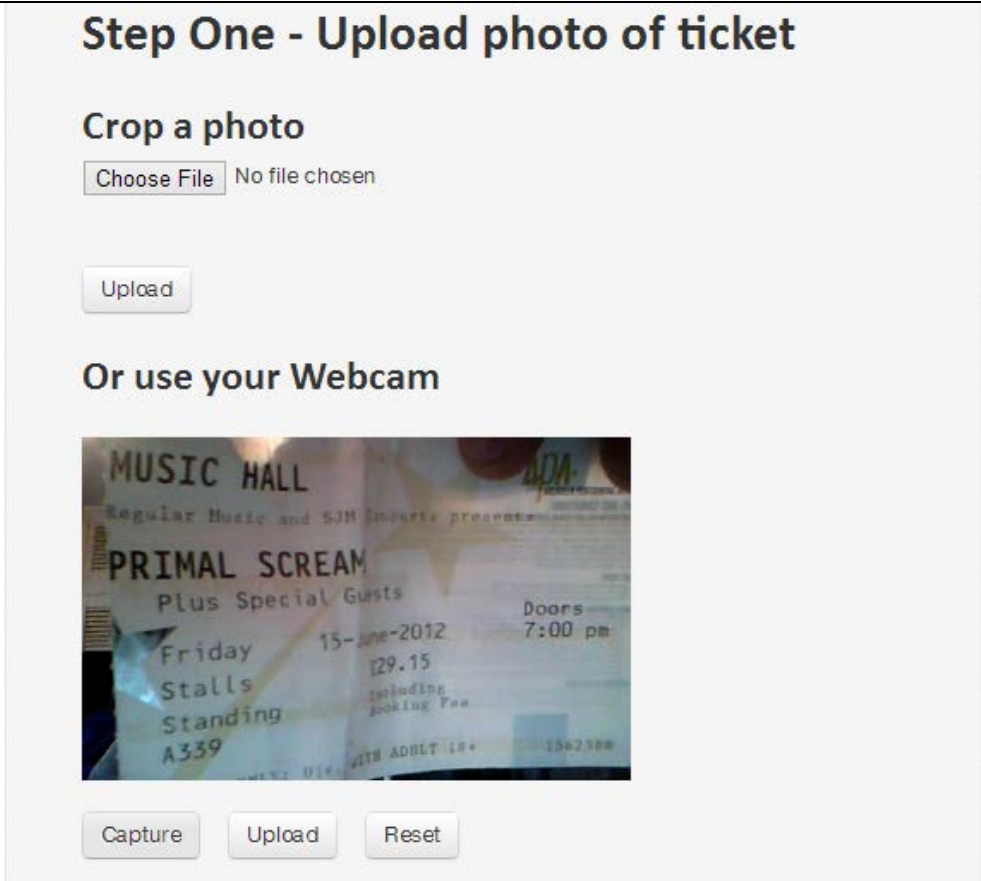
| | |
|---------------------------------|---|
| Test ID: | ST3 |
| Test Description: | A logged in user can create a new event |
| Requirement's satisfied: | 1.0, 1.2, 1.3, 1.5, 2.0, 2.2, 2.6, 2.7, 2.8, 2.9, 2.10, 3.3, 3.4, 3.9, 3.10 |
| Test Results: | Each function of the Create an Event Function was tested here and an event was created. |
| Test Errors: | The OCR's API key had to change as the maximum number of requests had been made. This was replaced and all other functions of the create event worked fine. |
| Screenshot: |  |

Table Name: Scenario ST4

| | |
|--------------------------|--|
| Test ID: | ST4 |
| Test Description: | The created event can be viewed in the My Box presentation area and manipulated. |
| Requirement' | 2.3 |


| | |
|----------------------|--|
| s satisfied: | |
| Test Results: | As expected, The created event is deleted. |
| Test Errors: | None |
| Screenshot: |  |

Table Name: Scenario ST5


| | |
|---------------------------------|--|
| Test ID: | ST5 |
| Test Description: | The specific event returns information from API's |
| Requirement's satisfied: | 3.0, 3.1, 3.2 |
| Test Results: | The application returns information about the event, setlist and Videos. |
| Test Errors: | None |
| Screenshot: |  |

Table Name: Scenario ST6

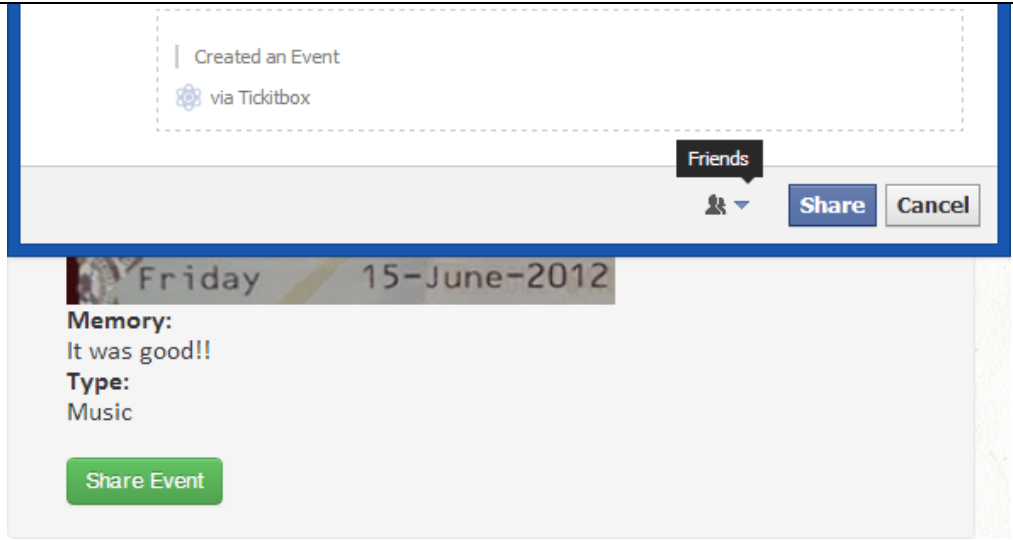
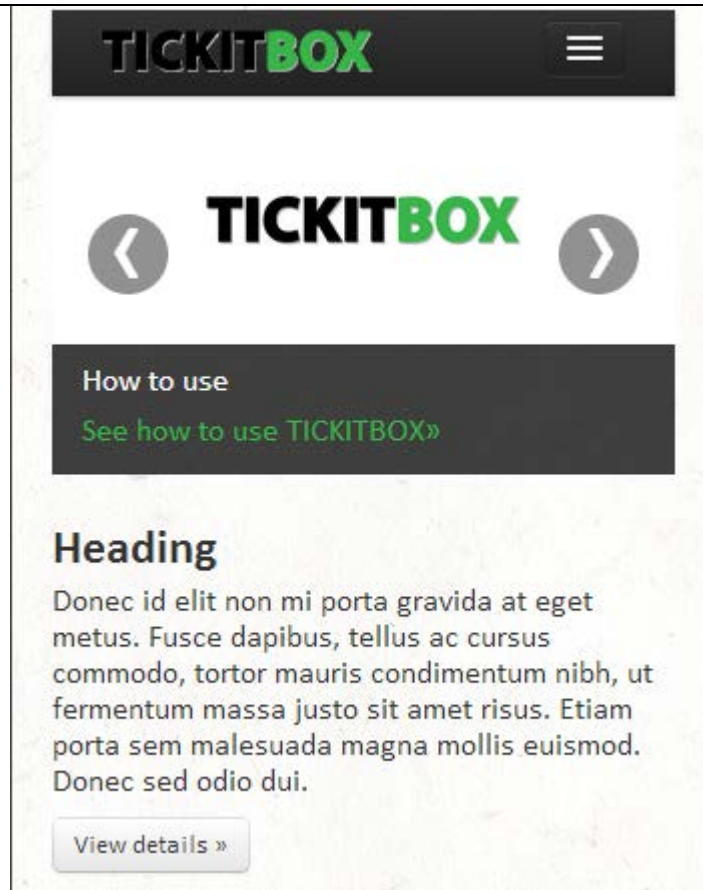
| | |
|---------------------------------|--|
| Test ID: | ST6 |
| Test Description: | The applications social media functions |
| Requirement's satisfied: | 3.6, 3.7 |
| Test Results: | The Application allows the user to login with Facebook and share the event they have created with Facebook |
| Test Errors: | None |
| Screenshot: |  |

Table Name: Scenario ST7

| | |
|---------------------------------|---|
| Test ID: | ST7 |
| Test Description: | Test to ensure the application is responsive |
| Requirement's satisfied: | 4.1 |
| Expected Results: | The application responds to the size of the browser. If the site was being viewed on a mobile device then the screenshot below is what that user would see. |
| Test Results: | As expected |
| Test Errors: | None |

Screenshot:



Scenario Testing Summary

Testing of each of the applications main functions was completed and no errors were found. The OCRs API key had to be changed during the test however, the API's development key only caters to 100 requests, so if the application was to be fully released then this API key would require a greater amount of requests.

The testing asked which requirements the tested functions would satisfy and the only requirements not covered by this test were 2.4 - Update and 2.5 – Intelligence as they have not been implemented. This also takes into account the interface and heuristic evaluations that will be conducted later in this chapter section 6.2.

Overall the scenario tests found every part of the application to be working as planned for during the design stage.

6.2 User Evaluation

In order to evaluate the user's experience and gain important feedback on as many aspects of the application as possible, a survey was created to be completed by a user who has used the application. As shown in Figure 6.1 the survey was made available from within the application:

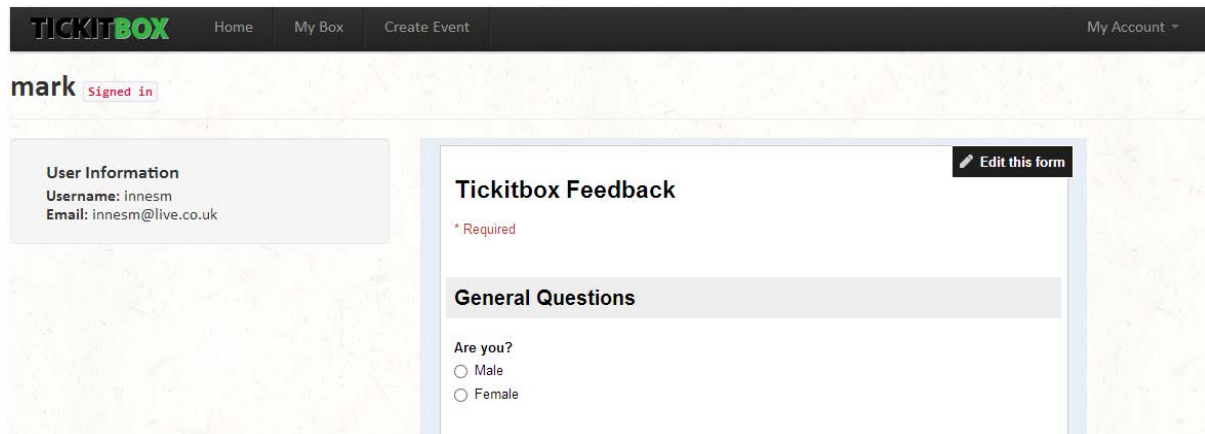
The image is a screenshot of a web application interface. At the top, there is a dark navigation bar with the 'TICKITBOX' logo on the left and links for 'Home', 'My Box', 'Create Event', and 'My Account' on the right. Below the navigation bar, the user is logged in as 'mark' with a 'Signed in' status. On the left side, there is a 'User Information' box containing the username 'innesm' and email 'innesm@live.co.uk'. The main content area is titled 'Tickitbox Feedback' and includes an 'Edit this form' button. Below the title, there is a section for 'General Questions' with a question 'Are you?' and two radio button options: 'Male' and 'Female'. A red asterisk and the word 'Required' are visible above the question.

Figure 6.1 Tickitbox's evaluation survey

The method of delivery of this survey was to set up a computer in a busy computing room at the Robert Gordon's University and to invite as many students and staff to use the application and fill out the survey. This method managed to recover 32 responses.

This method is also a way of user acceptance testing and observations made during the participants' use of the application are included.

The participants in the user evaluation survey are asked general questions for statistical purposes before asking question regarding the user interface and the usability of Tickitbox. Questions were asked on usefulness, ease of use, ease of learning, satisfaction, user interaction and aesthetics.

The goal of the survey is to gain honest feedback about how the application performs and to provide evidence that the requirement's 4.0, 4.2 and 4.4 have been met and provide an interface and heuristic evaluation through the choice of questions in the survey.

6.2.1 The survey results

The survey results are shown in the table 6.1 below:

Of the 32 participants:

Gender:

68% Male

32% male

Age:

21% 16-18

42% 19-21

21% 22-25

11% 25-30

5% Over 30

Table 6.1

Table Name: Usefulness

| Question | Positive % | Negative % |
|---|------------|------------|
| Would you use this application to store your tickets? | 79 | 21 |
| Tickitbox does everything I would expect it to do? | 84 | 16 |
| It meets my needs... | 84 | 16 |
| Tickitbox is useful | 95 | 5 |

Table Name: Ease of use

| Question | Positive % | Negative % |
|--|------------|------------|
| Easy to use? | 100 | 0 |
| Tickitbox is simple to use | 100 | 0 |
| Tickitbox is user friendly | 100 | 0 |
| Using Tickitbox is effortless | 100 | 0 |
| I don't notice any inconsistencies as I use it | 100 | 0 |

| | | |
|---|-----|---|
| Both occasional and regular users would like it | 100 | 0 |
| I can use it without written instructions | 100 | 0 |

Table Name: Ease of Learning

| Question | Positive % | Negative % |
|--|------------|------------|
| I learned to use Tickitbox quickly | 100 | 0 |
| I easily remember how to use Tickitbox | 100 | 0 |
| Tickitbox speaks my language...no jargon | 100 | 0 |
| It is easy to learn to use it | 100 | 0 |

Table Name: Satisfaction

| Question | Positive % | Negative % |
|----------------------------------|------------|------------|
| I am satisfied with it | 100 | 0 |
| I would recommend it to a friend | 100 | 0 |
| It is pleasant to use | 100 | 0 |

Table Name: User Interaction

| Question | Positive % | Negative % |
|---|------------|------------|
| Tickitbox's navigation made it easy to move through the application | 100 | 0 |
| Organization of information was clear | 100 | 0 |
| Straightforward task performance was successful | 100 | 0 |
| It was clear what part of the | 100 | 0 |

| | | |
|-----------------------------------|--|--|
| application I was on at all times | | |
|-----------------------------------|--|--|

Table Name: Aesthetics

| Question | Positive % | Negative % |
|---|------------|------------|
| The design style is consistent throughout | 100 | 0 |
| Colours used are appropriate and pleasing | 75 | 25 |
| Tickitbox is visually appealing | 90 | 10 |

Overall rating out of 10:

5/10 3%

7/10 9%

8/10 15%

9/10 27%

10/10 45%

6.2.2 User acceptance testing observations

Notable observations of participants:

All participants seemed to have a positive experience

Some struggled to figure out how to use the Create an Event function

One user pointed out a spelling error on Welcome page

Some of the participants seemed very surprised with the OCR feature in a positive way and asked questions on how it worked.

6.2.3 User Evaluation reflection

From the positive observations of the participants, and the overwhelmingly positive results of the survey from the same participants, it would be possible to argue that Tickitbox has met its design requirements for usability, style, and user experience. The questions were structured to cover the requirement and also to provide evidence that the application meets the 10 usability heuristics as developed by Nielsen.

With a near 100% positive response for each question, the application pleased a diverse group of people of a selection of ages. Clearly the idea behind the Tickitbox application is not for everyone (21% negative on if they would store their tickets in this box) but the people who said they may not use it still were able to recognise that the application meets the goals it was asked.

Overall, 88% of participants would give Tickitbox an 8/10 or better.

The evaluation will now look at the performance of Tickitbox.

6.3 Performance Evaluation

In order to evaluate the Applications performance, development add-on Firebug was used as well as the built in developer tools for the Chrome browser. The goal of the performance evaluations is to show the application has been built effectively. The performance results will reflect on the design of that application as well as the implementation.

Firebug

Firebug is the add-on to the Firefox browser and is a set of tools that allow web developers to edit, debug and monitor an application from the browser. Firebug was used to evaluate Tickitbox by using its error console log to monitor issues that arise from using the application. The console log will show how quickly functions load.

Error Console Report

Table 6.2

Table Name: Error Report

| Page | Error Shown | Resolved | How |
|------|--|----------|--------------------------------------|
| Home | Error: TypeError:
document.getElementById(...) is
null
Source File:
http://localhost/appticko/js/fbss
.js
Line: 27 | No | Cannot be resolved |
| Home | Error: The character encoding of
the HTML document was not | Yes | Added the following to
layout.php |

| | | | |
|--------|---|-----|--|
| | <p>declared. The document will render with garbled text in some browser configurations if the document contains characters from outside the US-ASCII range. The character encoding of the page must be declared in the document or in the transfer protocol.</p> <p>Source File:
http://localhost/appticko/
Line: 0</p> | | <pre><meta content="text/html; charset=utf-8" http-equiv="Content-Type"></pre> |
| Home | <p>Error: ReferenceError: jsid is not defined</p> <p>Source File:
http://localhost/appticko/js/master.js
Line: 64</p> | No | Cannot be resolved |
| Login | <p>Error: The stylesheet http://localhost/appticko/css/normalize.css was not loaded because its MIME type, "text/html", is not "text/css".</p> <p>Source File:
http://localhost/appticko/login
Line: 0</p> | Yes | <p>Spelling error in normalize.css name.</p> <p>Stylesheet not required.</p> |
| Login | <p>Error: TypeError: \$(...).validate is not a function</p> <p>Source File:
http://localhost/appticko/login
Line: 86</p> | Yes | Error in code |
| My Box | <p>Error: TypeError: \$(...).stepy is not a function</p> <p>Source File:</p> | Yes | Removed unnecessary function |

| | | | |
|--|--|--|--|
| | http://localhost/appticko/user/innesm
Line: 330 | | |
|--|--|--|--|

The two errors not resolved cannot be done so due to the way the application is created. The JSID variable has to be on a different page from the script as it refers to the post id which is only pulled from the URL bar on the events page.

The Facebook error is for the same reason, variables that are not global have to be used in other places.

Chrome Developer Tools

By using the Network, PageSpeed and Audit sections of the developer tools, it is possible to see how quickly the pages of the application load and gain recommendations on how to improve this time. Table 6.3 shows how quickly each page fully loads from the Network tab, table 6.4 shows the PageSpeed Score and recommendations to improve the PageSpeed and 6.5 shows the Audits tab results. The goal of this evaluation is to reduce load times to as low as possible. The load times will be taken using the Network tab and recorded before taking into consideration the recommendation's made in the PageSpeed and Audit tabs. The load times will then be taken again and a speed score for each page to see the difference made in table 6.6.

Table 6.3

Table Name: Google Chrome Network tab

| Presentation layer | Requests made | Time taken |
|--------------------|---------------|--------------|
| Home | 18 | 244ms |
| Login | 34 | 536ms |
| Sign-up | 32 | 690ms |
| Welcome | 27 | 791ms |
| My box | 33 | 1.14 seconds |
| Create Event | 36 | 1.20 seconds |
| Event page | 69 | 3.2 seconds |

The Google documentation states that the maximum page loading times should be 2 seconds or less before a user becomes frustrated or disinterested. The event page taking 3.2 seconds to load is far

too long and with 69 requests being made, that is too many. Table 6.3 shows that as the amount of requests made increases, then the time taken also increases. The PageSpeed and Audit tabs introduce ways of making Tickitbox as fast as possible by providing recommendations on how to do so.

Google Chrome PageSpeed tab

Page speed analyses the entire page and identifies ways to gain optimum performance and gives advice on how to carry out the improvements with the score out of 100 with 100 being the best score. The Google homepage was tested to see how quick it was for comparative purposes, and it scored 98 with no recommendations, showing that they have considered everything to load the page faster

Table 6.4

Table Name: Google Chrome PageSpeed

| Presentation layer | Page Speed Score |
|--------------------|------------------|
| Home | 58 |
| Login | 59 |
| Sign-up | 65 |
| Welcome | 72 |
| My Box | 69 |
| Create Event | 68 |
| Event page | 81 |

The recommendations made by Google varied for each page and all problems come from the same sources; however the most common recommendations were:

- **High priority.**
Enable compression, Leverage browser caching
- **Medium priority.** Minimise redirects
- **Low priority.** Defer parsing of JavaScript, Minify CSS, Minify JavaScript, Optimise images, Minify HTML, Specify a character set, Specify image dimensions, Specify a Vary: Accept-Encoding header

Appendix C shows a list of the recommendations for each page of the application. Each recommendation was considered and followed as closely as possible and the new scores are displayed in Table 6.6.

Google Chrome Audit tab recommendations:

1. Combine external CSS
2. Combine external JavaScript
3. Enable gzip compression
4. Leverage browser caching
5. Leverage proxy caching
6. Minimize cookie size
7. Serve static content from a cookie less domain
8. Specify image dimensions
9. Remove unused CSS rules

Appendix D shows the recommendations made, but also lists the reasoning behind them and names the files that could be improved.

The Audit tab made recommendations mainly regarding how the CSS and JavaScript are placed into the application, showing that the changes recommended can make large improvements to the load times. All of the recommendations were implemented as much as possible and again the difference these recommendations made is reflected in Table 6.6.

Table 6.6

Table Name: Page speed and load scores having considered recommendations

| Presentation layer | Requests made | Time taken | Page Speed Score |
|--------------------|---------------|-------------|------------------|
| Home | 8 | 161ms | 91 |
| Login | 22 | 301ms | 79 |
| Sign-up | 15 | 176ms | 88 |
| Welcome | 17 | 368ms | 89 |
| My box | 22 | 277ms | 81 |
| Create Event | 33 | 691ms | 81 |
| Event page | 57 | 1.33seconds | 91 |

Table Name: Averages

| | | | |
|---------------------|----|-----------|----|
| Before Optimisation | 35 | 1.1second | 67 |
| After Optimisation | 24 | 472ms | 85 |

The recommendations and modifications proposed to the application were carried out as closely as possible. Table 6.6 shows the reduction in load times and the increase in Page Speed Score. The average load time dropped from 1.1 second down to 472ms and the average Page Speed Score raised from 67 up to 85.

The change that yielded the greatest improvement was Enabling compression on the server.

```
# compress text, html, JavaScript, css, xml:
AddOutputFilterByType DEFLATE text/plain
AddOutputFilterByType DEFLATE text/html
AddOutputFilterByType DEFLATE text/xml
AddOutputFilterByType DEFLATE text/css
AddOutputFilterByType DEFLATE application/xml
AddOutputFilterByType DEFLATE application/xhtml+xml
AddOutputFilterByType DEFLATE application/rss+xml
AddOutputFilterByType DEFLATE application/javascript
AddOutputFilterByType DEFLATE application/x-javascript

# Or, compress certain file types by extension:
<files *.html>
SetOutputFilter DEFLATE
</files>
```

The code above was added to the .htaccess file and the deflate_module was turned on within the server. This code allows the output from the server to be compressed before being sent to the client over a network.

By combining the JavaScript and CSS files into one, the amount of requests made dramatically fell and this vastly increased the Page Speed Score.

6.4 Quality of the application

The final part of the evaluation is to measure the quality of the code and make changes based on the recommendations of validators. Requirement 4.3 states that the application should conform to the standards defined by the World Wide Web Consortium (W3C) however this was not deemed important to the success of the project. The code will be checked using a Chrome add-on called Kingfisher HTML5 Validator, which conforms to W3C standards and the new regulations as defined in the HTML5 remit.

As Ticketbox relies on a large amount of JavaScript, the quality of the JavaScript code will also be checked using JS Hint, which detects errors and problems in JavaScript code.

The errors each of these methods found are explained below:

Kingfisher HTML5 validator

There were around 100 errors found by the HTML5 validator that included missing titles, no alt tags on images, missing div tags and not closing tags. A full listing of each error found by the validator broken down by page can be found in Appendix E.

Each page's errors and warnings were evaluated and re-coded until the message was shown explaining the page contained valid HTML. The warning page has to stay with two warnings due to the insertion of the Facebook login button. This uses Facebook's own mark-up therefore would not pass as valid HTML

JS Hint – Master.js



Figure 6.2 JS Hint

The applications code that has been created during this application is all contained in the Master.js file. Other JavaScript exists; however Master.js is the most important and contains the most functions. Clearly Figure 6.2 shows that there are many errors within the Master.js file with around 400 present.

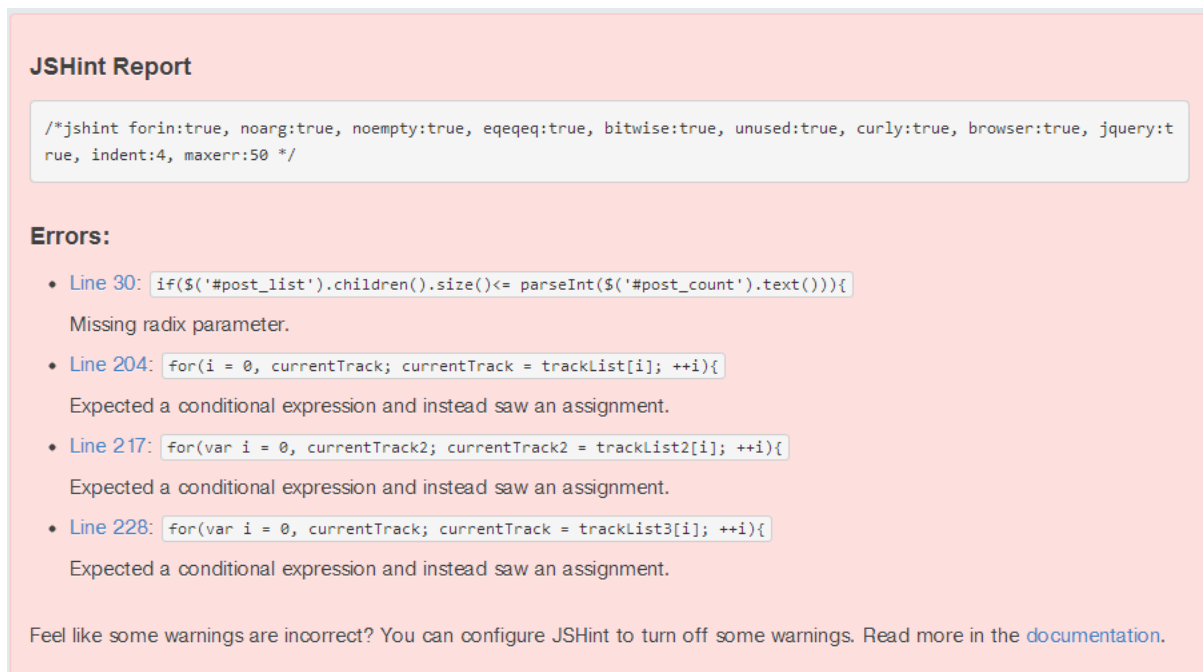


Figure 6.3 JS Hint after re-coding

By using JS Hint to find errors, the application was re-coded and the amount of errors within Master.js was taken from over 400 to 4. The errors that remain are necessary to the applications functionality and cannot be resolved.

This completes the quality evaluation of the application with a greater enhancement of the applications code.

6.5 Evaluation Summary

Tickitbox has now been scenario tested on its main features, evaluated by users measuring their experience, performance and the quality it possesses with each of these evaluation criteria supported by meeting the applications own requirements. With more time, a larger study could have been created to assess further the development tools Chrome and Firefox has to optimize the application further, and to gain a larger number of participant's for the evaluation survey. Based on the evaluations that have been completed, it can be said that Tickitbox has performed admirably, with a few issues arising from the evaluations. The performance optimisation achieved positive results in terms of reducing the speed of the application, the quality of the applications code

has been improved in a way that it is now HTML validated on each page. The user experience evaluations were very useful, and the observations made during participant's use of the application highlighted areas in which Tickitbox could be improved further:

- The create an event page should explain how to use the function far better.
- Overall the explanation on how the site works should be present throughout the site, not just within the how to use guide.

7 Conclusions and reflections

7.1 Project contributions

In chapter 1, it was stated that public events, whether the event is a concert, festival or sporting event, generated content in various ways. It could be the sets list of a gig or the winning goal at a football match, but it is more than likely that some form of information about an event exists somewhere online. A problem exists where content can come from many places and there is nothing linking that content with similar content based around the web.

This problem formed the basis of this project, and with that in mind, this projects purpose was to create a system that provides a service to allow users to interact with an event they have been to by pulling the information from many places into one single system.

This problem could have many interpretations as to what a system could be like that provides this service, but system created called Ticketbox certainly provides the following solutions to the problem that existed, by doing the following:

1. Providing a place to store individual event information online.
2. Using that information created to go to many content generators and return information about that event into a single application.

Based on the problem that is at the heart of this project, the following objectives were created:

1. Analysis of existing online multimedia services and web development technologies;
2. Conducting a study of online social behaviour;
3. Design and implement an application that interacts with existing web applications to create a social experience.
4. Explore new technologies of Web Application Development.
5. Proper Testing and Evaluation of the application.

Objective 1 was met through chapter 2's literature review where a full analysis of the history of the online services was compiled to the present day and through the research carried out on web application methods in section 2.3

Objective 2 involved analysing the problem at hand in chapter 3 before creating a study of potential users of an application. A survey was created in order to gain an understanding how people interact with the Web. Facebook was used as a common application for the survey in order to see

participants views on different parts of the application which, along with section 2.2 allowed for an in depth study into human online behaviour.

Objective 3 has been met in the development of Tickitbox. Tickitbox's design and implementation has successfully met each requirement of the application as detailed in the requirements document of in section 3.6 and the application created is a solution to the problem posed in chapter 1.

Objective 4 was met in various ways. The literature review has looked at many modern web development technologies and libraries, and the application has a NoSQL CouchDB database within the application. The design stage also had a Prototype created in Node.js and although Tickitbox was eventually made in PHP, the Node.js prototype explored what benefits that method could have to more suited applications. Evidence this objective was met is further enhanced by the work with Optical Character Recognition. The experiment conducted with the OCR found characteristics that affect the OCR's performance may not have been considered before.

Objective 5, in regards to the testing and evaluation of the application, Tickitbox was evaluated using scenario based testing, User evaluations, Performance testing and evaluation of the applications code. Each of these evaluations produced positive results.

As Tickitbox has met all of its objective's and satisfied each requirement it would appear to be suitable solution to the problem faced by the project and identified in Chapters 1 and 3.

7.2 Conclusions

Tickitbox is a web application that allows a user to upload and store information about an event they have been to, and then returns information about that event. The application built has been shown to have been tested and evaluated as the solution to the problem that chapter 1 introduced. As the further work section shows, there are many ways to extend Tickitbox further and given more time, the further work section would have been less, or rather other features would have been created to add on to what is a potentially very useful application.

A literature review was carried out in chapter 2 in order to prepare for the analysis of the problem in detail and formed the basis for the creation of requirements for the design of the application.

Chapter 3 saw a detailed analysis of the problem faced before a survey was created and received 52 replies. The survey was carried out in order to achieve an understanding of how potential users of Tickitbox interact online. The survey supported the literature review and a requirements document was created.

The applications design was discussed in chapter 4, where the application was taken from a very low level system to the feature heavy presentation layers. This chapter's most significant decision was choosing the applications programming language. After creating a Node.js prototype, it was decided that PHP should be the language that Tickitbox uses.

Tickitbox's implementation was covered in chapter 5. The design blue print was implemented with this chapter including the most complex parts that make up the final application. During the implementation of the OCR, an experiment was carried out showing that as the size of an image increases, the amount of data returned from an OCR decreases and this experiment allowed a recommended image size to be found. The experiment also proved that removing redundant data from an image increases how well the OCR performs. This experiment resulted in Tickitbox having a cropping system being designed into the application in order to achieve the best results from the OCR service. The implementation covered how images are handled by Tickitbox, how information is stored, how the application gets information regarding the event and how the application itself handles issues like security and manages users.

The application was tested and evaluated in chapter 6 by using a variation of techniques. The application was shown to score very positively with users, a reflection on the design of the application considering the users experience to be a top priority. The performance of Tickitbox was dramatically improved by implementing recommendations from web development tools before the quality of the applications code was HTML validated and checked using JS Hint. The testing and

evaluation support the case for Tickitbox being a very user friendly web application which has no known errors and will be suitable for the projects demonstration.

Ticktibox is a solution to the problem found in event content generation and correlates that content into a system that allows users to have digital relationships with the events they have attended.

7.3 Further work

Further work can be carried out to further enhance many aspects of the applications implemented features, and also by evaluating the current application further could also improve Tickitbox.

Some features that would add to the current web application to improve the service to a user will be implemented in the future are as follows:

1. Native mobile applications will be created for iOS and Android:

The application is fully responsive to the type of device that the user will be using; however to further improve on that experience, native mobile applications will be built for iOS and Android. They would link to a deployed Tickitbox, and simply provide the user with a login system and the ability to take a photo of a ticket on the move as oppose to using a mobile browser.

2. The create an event function will be improved to have less steps:

Currently, the Create an Event function has 5 actions it must perform before an event is created, Uploading of a ticket, Cropping, saving the ticket image to a database before then sending it to the OCR reader and filling in the form. The idea is to stop the user having to download the processed image before sending it to the OCR, this should just be done automatically. This was not done in the implementation due to the difficulty in creating an HTML form with two actions for a file upload. The image had to be processed before sending to the OCR and the only way to get the processed image to the second form was for the user to locate it themselves. This issue was looked at during the implementation, but was not considered vital to creating the application as the create an event component still manages to perform all of the tasks it was required to perform.

3. The OCR reader will be improved by using entity recognition:

In order to read the ticket, the string returned from the OCR is compared with artists and venues that have been pre-coded in JavaScript. This method involves comparing the text in the string with every value in the pre-defined code list before finding matched artists or venues. There are other methods in extraction information that would not involve hard coding every possible artist or venue that could come up. This would involve implementing entity recognition tools, which would automatically be aware if Primal Scream was an artist or the Music Hall is a venue. The research into this at the moment seem to have no web based API's for performing tasks like this, but it would benefit the users experience if every possible combination could be found.

4. The application will learn about the user's events to provide a greater experience:

The application should learn about the users as more events are added. This was attempted by creating a design document within the application that stored new artists as they were created, but it was difficult to create a mechanism that would then look up this design document for the artist name. The idea would be that the application automatically recognises artists or venues the user has visited before and suggests them to the user.

5. The event data collected about a user will be displayed in more efficient ways.

Finally the application's display of events currently does not have any order or ranking system. The idea would be to rank the events in some way where a number one event could be defined. The application would also have options to filter the events created by artist, venue, date, year or rating and show the users this information within the application.

7.4 Personal Reflections

I have attended quite a large amount of events, concerts, gigs and festivals. For every single one I have been to, I have kept the ticket for that event as a memento. The tickets all sit in an old Adidas shoe box in my house. The 1st event I went to was a Doves concert at the Music Hall in Aberdeen many years ago and the last was Maceo Plex recently in Edinburgh. In between there have been many events that have been such a big part of my life, that all of this information is stored in my box.

I'm not saying that every event I have ever been to has been brilliant or unforgettable, however the reason I kept these tickets was to remember what it was like being there. A lot of the survey responses shown in Appendix B hint at an "atmosphere" that existed during an event. It would be foolish to try and recreate something that doesn't really exist, one person's atmosphere might be someone else's rubbish night out.

What I wanted to achieve with this project was to create an application which digitizes my box of tickets in a way that allows me to "remember" the event and act as a reminder of that warm feeling of a good atmosphere for an event.

I saw Noel Gallagher at the AECC in February 2012, someone who I really didn't like. I bought a ticket off eBay about 3 hours before it started for half the price of the face value because my brother wanted to go. It ended up being surprisingly good but what I noticed during the event was the sheer volume of photos and videos people take throughout a gig. For one song in particular; "Don't look back in anger" there were more people recording the event on their phones than actually watching it live. These types of moments are placed on YouTube for the world to see but are special to the person uploading.

That is the type of stories I wanted this application to capture, a place to store ticket images and information yes, but as a way to personally reflect on an event that has been attended so that when the user returns later to remember they can instantly be reminded of feelings that existed during that event.

This project has allowed me to express myself through this application. This idea has been with me since I began university, and is something I will carry on and extend when I have left.

There have been many challenges faced in the creation of Tickitbox, none bigger than the Optical Character Recognition that has been implemented in the application. I tried 5 different ways to make this work and I nearly gave up on including it at all. It seemed everything I was trying would just

not integrate with the application. I then had a breakthrough and I have to be honest, the 1st time I successfully pulled the artist and venue from a Ticket image is something I will never forget.

Something I regret, is not managing to get the application live online. I wanted to create a Phonegap application that would link to the website and use the phone camera to take the image of the ticket and upload to the OCR. This would work if accessed by a phone, I have made it so that function is available, but due to problems with Cloudant and AppFog, the database of the application would not deploy online.

If I could start from the beginning again, I would have created the application using Node.js. I am fully satisfied with what the current Tickitbox has achieved and I have tried to push myself as much as possible by using newer technologies, but to have completed this application in only JavaScript would have not only be something to be proud of, but would have been beneficial to the wider web development community who are also exploring new technologies.

The most satisfying thing about this project has been how much I have learned about what is possible in web development and I would like to thank my supervisor Richard Glassey, for guidance throughout the year and introducing me to some of the libraries and technologies that have been implemented in Tickitbox.

Mark Innes

8 References

The History of the Internet. 2013. *The History of the Internet*. [ONLINE] Available at: <http://www.investintech.com/content/historyinternet/>. [Accessed 20 February 2013].

Internet Timeline History of the Internet. 2013. *Internet Timeline History of the Internet*. [ONLINE] Available at: http://www.siliconvalleyhistorical.org/home/internet_timeline. [Accessed 02 March 2013].

WC3. 2013. *TCP/IP Introduction*. [ONLINE] Available at: http://www.w3schools.com/tcpip/tcpip_intro.asp. [Accessed 02 March 2013].

HowStuffWorks . 2013. *HowStuffWorks "How did the Internet start?"*. [ONLINE] Available at: <http://computer.howstuffworks.com/internet/basics/internet-start.htm>. [Accessed 02 March 2013].

W3C - History. 2013. P *Chapter 2*. [ONLINE] Available at: <http://www.w3.org/People/Raggett/book4/ch02.html>. [Accessed 02 March 2013].

Webopedia. 2013. *What is HTTP? - A Word Definition From the Webopedia Computer Dictionary*. [ONLINE] Available at: <http://www.webopedia.com/TERM/H/HTTP.html>. [Accessed 02 March 2013].

NCSA. 2013. *About NCSA Mosaic*. [ONLINE] Available at: <http://www.ncsa.illinois.edu/Projects/mosaic.html>. [Accessed 02 March 2013].

The Evolution of the Web. 2013. *The Evolution of the Web*. [ONLINE] Available at: <http://www.evolutionoftheweb.com>. [Accessed 02 March 2013].

New Media Trend Watch World. 2013. *World Usage Patterns & Demographics - New Media Trend Watch World*. [ONLINE] Available at: <http://www.newmediatrendwatch.com/world-overview/34-world-usage-patterns-and-demographics>. [Accessed 02 March 2013].

BBC – WebWise. 2013. *BBC - WebWise - What are cookies?*. [ONLINE] Available at: <http://www.bbc.co.uk/webwise/guides/about-cookies>. [Accessed 02 March 2013].

BBC News - Cookie law: websites must seek consent from this weekend. 2013. *BBC News - Cookie law: websites must seek consent from this weekend*. [ONLINE] Available at: <http://www.bbc.co.uk/news/technology-18194235>. [Accessed 02 March 2013].

SSL Explained. 2013. *SSL Explained : ScionSSL SSL Certificate Authority*. [ONLINE] Available at:<http://www.scionssl.com/page/ssl-explained>. [Accessed 02 March 2013].

Money | guardian.co.uk . 2013. *Online retail sales hit £50bn | Money | guardian.co.uk* . [ONLINE] Available at:<http://www.guardian.co.uk/money/2012/jan/19/online-retail-sales-hit-50bn>. [Accessed 02 March 2013].

PHP explained. 2013. *PHP explained*. [ONLINE] Available at:<http://everything.explained.at/PHP/>. [Accessed 02 March 2013].

Understand JavaScript 2013. *JavaScript - General Introduction*. [ONLINE] Available at:<http://www.quirksmode.org/js/intro.html>. [Accessed 02 March 2013].

HTML source. 2013. *HTML 4 Explained | A reference of each HTML 4.01 element and attribute*. [ONLINE] Available at:<http://www.yourhtmlsource.com/accessibility/html4explained.html>. [Accessed 02 March 2013].

Statistic Brain. 2013. *Apple Computer Company Statistics | Statistic Brain*. [ONLINE] Available at:<http://www.statisticbrain.com/apple-computer-company-statistics/>. [Accessed 03 March 2013].

New York Times. 2013. *The Dot-Com Bubble Bursts - New York Times*. [ONLINE] Available at:<http://www.nytimes.com/2000/12/24/opinion/the-dot-com-bubble-bursts.html>. [Accessed 03 March 2013].

Company - Google . 2013. *Company – Google* . [ONLINE] Available at:<http://www.google.com/about/company/>. [Accessed 03 March 2013].

WordPress Statistics. 2013. *WordPress » About » Statistics*. [ONLINE] Available at: <http://wordpress.org/about/stats/>. [Accessed 03 March 2013].

The Dot-com Bubble 2013. *The Dot-com Bubble | Stock Market Crash!*. [ONLINE] Available at: <http://www.stock-market-crash.net/dot-com-bubble/>. [Accessed 03 March 2013].

jQuery. 2013. *jQuery*. [ONLINE] Available at: <http://jquery.com/>. [Accessed 03 March 2013].

Vine. 2013. *Vine*. [ONLINE] Available at: <http://vine.co/>. [Accessed 03 March 2013].

Facebook Statistics. 2013. *Facebook Statistics*. [ONLINE] Available at:<http://blog.kissmetrics.com/facebook-statistics/>. [Accessed 03 March 2013].

Sopa | The Guardian . 2013. *Sopa | Technology | The Guardian* . [ONLINE] Available at:<http://www.guardian.co.uk/technology/sopa>. [Accessed 03 March 2013].

YouTube. 2013. *YouTube*. [ONLINE] Available at:http://www.youtube.com/t/about_youtube. [Accessed 03 March 2013].

node.js. 2013. *node.js*. [ONLINE] Available at: <http://nodejs.org/>. [Accessed 04 March 2013].

Brandwatch. 2013. *What Would Happen if the Internet Just... Collapsed? - Brandwatch*. [ONLINE] Available at:<http://www.brandwatch.com/2012/05/what-would-happen-if-the-internet-just-collapsed/>. [Accessed 04 March 2013].

WIUSIWPS 2013. *World Internet Users Statistics Usage and World Population Stats*. [ONLINE] Available at: <http://www.internetworldstats.com/stats.htm>. [Accessed 04 March 2013].

SitePoint. 2013. *Human-Computer Interaction and Your Site - SitePoint*. [ONLINE] Available at:<http://www.sitepoint.com/computer-interaction-site/>. [Accessed 05 March 2013].

WhatIs.com. 2013. *What is stickiness? - Definition from WhatIs.com*. [ONLINE] Available at:<http://searchsoa.techtarget.com/definition/stickiness>. [Accessed 05 March 2013].

Telegraph. 2013. *Number of smartphones tops one billion - Telegraph*. [ONLINE] Available at:<http://www.telegraph.co.uk/finance/9616011/Number-of-smartphones-tops-one-billion.html>. [Accessed 09 March 2013].

Ratner, J. (2002). *Human Factors and Web Development*. 2 Edition. CRC Press.

Effective UI, (2010). *Effective UI: The Art of Building Great User Experience in Software*. 1 Edition. O'Reilly Media.

Nielsen. 2013. *10 Heuristics for User Interface Design: Article by Jakob Nielsen*. [ONLINE] Available at: <http://www.nngroup.com/articles/ten-usability-heuristics/>. [Accessed 09 March 2013].

Sherry Turkle, 2011. *Alone Together: Why We Expect More from Technology and Less from Each Other*. 1 Edition. Basic Books.

Pew Internet 2013. *Facebook: A Profile of its 'Friends' In light of...* [ONLINE] Available at:<http://pewinternet.tumblr.com/post/23177613721/facebook-a-profile-of-its-friends-in-light-of>. [Accessed 09 March 2013].

PLOS ONE, 2013. *PLOS ONE: Differential Psychological Impact of Internet Exposure on Internet Addicts*. [ONLINE] Available at: <http://www.plosone.org/article/info:doi/10.1371/journal.pone.0055162#pone.0055162-Young2>. [Accessed 10 March 2013].

AOA. 2013. *Internet Addiction and Online Addiction*. [ONLINE] Available at: <http://psychcentral.com/netaddiction/>. [Accessed 10 March 2013].

BBC News - 'Internet addiction', 2013. *BBC News - 'Internet addiction' linked to depression, says study*. [ONLINE] Available at: <http://news.bbc.co.uk/1/hi/8493149.stm>. [Accessed 10 March 2013].

G Kappel, 2006. *Web Engineering: The Discipline of Systematic Development of Web Applications*. 1 Edition. Wiley.

Tim Juravich, 2012. *CouchDB and PHP Web Development Beginner's Guide*. Edition. Packt Publishing.

Wired.com. [ONLINE] Available at: <http://www.wired.com/wiredenterprise/2012/09/meet-the-man-whos-rewiring-google-from-the-inside-out/>. [Accessed 20 March 2013].

MongoDB. 2013. *MongoDB*. [ONLINE] Available at: <http://www.mongodb.org/>. [Accessed 20 March 2013].

Apache CouchDB. 2013. *Apache CouchDB*. [ONLINE] Available at: <http://couchdb.apache.org/>. [Accessed 20 March 2013].

jQuery UI. 2013. *jQuery UI*. [ONLINE] Available at: <http://jqueryui.com/>. [Accessed 20 March 2013].

Ralph Moseley, 2007. *Developing Web Applications*. 1 Edition. Wiley.

PhoneGap 2013. *PhoneGap / Home*. [ONLINE] Available at: <http://phonegap.com/>. [Accessed 21 March 2013].

jQuery Mobile. 2013. *jQuery Mobile / jQuery Mobile*. [ONLINE] Available at: <http://jquerymobile.com/>. [Accessed 21 March 2013].

Sinan Si Alhir, 2003. *Learning UML*. 1st Edition. O'Reilly Media.

Appendix A: Survey Questions

General questions:

- Gender of participant.

Possible answers: Male or Female

- Age of participant.

Possible answers: Under 16, 16-18, 19-21, 22-25, 25-30, Over 30

Reasoning – The participants can be separated into user groups, allowing the project to build use cases for the design stage. Also, user profiles can be generated based on the rest of the responses in order to create a digital representation of a person in order to make assumptions that can influence the design. For example, if the response for the service was only positive for Females who are under 16, then the application would have many different factors affecting it when compared with a Male who is over 30.

Events the User has attended:

-Frequency of a participant attending ticketed events.

Possible answers: Once a Week, Once a Fortnight, Once a Month, Never or Other

-Participants attitude to the ticket when the event is over.

Possible answers: I keep every ticket; I keep tickets of events that were particularly good, Throw away or other

-Where the participant keeps there tickets if they are kept.

Possible answers: Open response text box

-What the 1st event the participant can remember going to.

Possible answers: Open response text box

-Invite the participant to explain in detail why this event has been remembered.

Possible answers: Open response text box

-What the best or most memorable event the participant has attended and why.

Possible answers: Open response text box

-Find out about which music festivals have been attended this year.

Possible answers: Rockness, T in the Park, Creamfields, Belladrum, None or Other

-Find out the most memorable moment of the festival.

Possible answers: Open response text box

Reasoning: This is to determine if the participant will use the potential service. Most people will go to ticketed events at least once per year, so if there is a positive response retrieved and they hold on to their tickets then this is the type of profile the application should cater for. Also, if a user is willing to tell a computer what their favourite moment of an event is, the language they use and how in detailed they are could lead to a feature within the application to record or state how they feel about the event.

Event Behaviour

-Type devices the participant takes to an event

Possible answers: Smartphone, Digital Camera, Camcorder, Sound recording equipment or other.

-Type of mobile device the user has.

Possible answers: iPhone, Android Device, Blackberry, Windows Phone or Other

-Type of content created at an event.

Possible answers: Photos, Videos, Sound Recordings or None.

-The reasons that trigger the need to create content.

Possible answers: Song being played, People the participant is with, To remember what it was like or other.

Reasoning: These questions are aimed at what people do at events with regards to content creation which mediums they use to create the content.

Social Media and Web services

-Invitation to voice concerns about Facebook and any issues the participant has with Facebook's service.

Possible answers: Open response text box

-If the participant has a YouTube or Google account.

Possible answers: Yes or no

-If the participant has a Spotify account.

Possible answers: Yes or no

-If the participant has a Flickr account.

Possible answers: Yes or no

-Invitation to voice opinion on Facebook events

Possible answers: Open response text box

-How the user most frequently checks Facebook

Possible answers: Laptop/Desktop, Mobile device, Tablet or Other

-Find the users thoughts on using Facebook Connect to register or login to a site.

Possible answers: More, Less or Don't know

-Invite the user to say why they would use Facebook Connect or not

Possible answers: Save time filling in login details again, Trust Facebook with data or Other

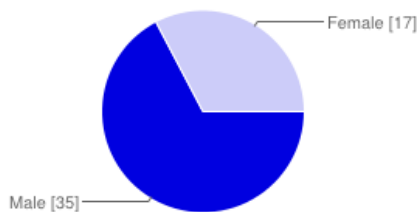
Reasoning: By using Facebook as a base as nearly everyone universally uses this service, it is easy to see participant's opinions on the design and how they interact as a basis for this applications design. The project is open to all options of development and one of these methods is to create a Facebook connect system using the SDK, so it would be beneficial to see what opinions on that service are. By asking what device the participant uses to view Facebook should decide how the web application should be implemented with regards to responsive design. Also by finding out if people have external services then it would allow the application to take advantage of them

Appendix B: Detailed Survey results

The survey had 52 replies and the full results are show below:

- *Gender of participant.*

Are you?



Results:

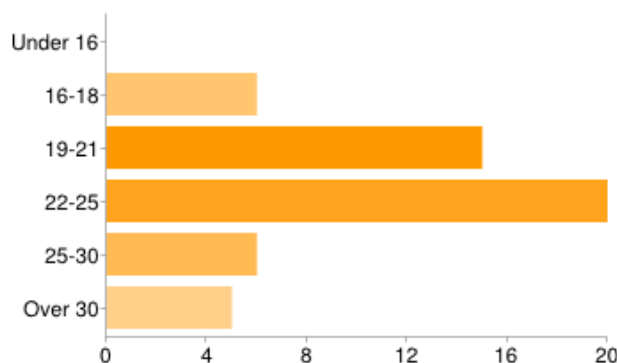
| | | |
|--------|----|-----|
| Male | 35 | 67% |
| Female | 17 | 33% |

Analysis Commentary:

Clearly the survey reached more women than men, however considering where this survey was shared or posted it is not surprising.

- *Age of participant.*

How old are you?



Results:

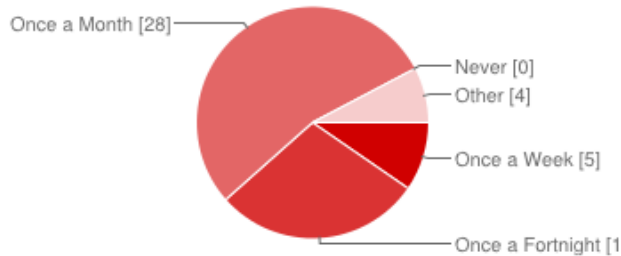
| | | |
|----------|----|-----|
| Under 16 | 0 | 0% |
| 16-18 | 6 | 12% |
| 19-21 | 15 | 29% |
| 22-25 | 20 | 38% |
| 25-30 | 6 | 12% |
| Over 30 | 5 | 10% |

Analysis Commentary:

Again the ages of the people asked are reflected by where the survey was posted, it would have been beneficial to have some Under 16's fill the survey. Overall a good range of ages of people asked.

-Frequency of a participant attending ticketed events.

How frequently do you attend ticketed events?



Results:

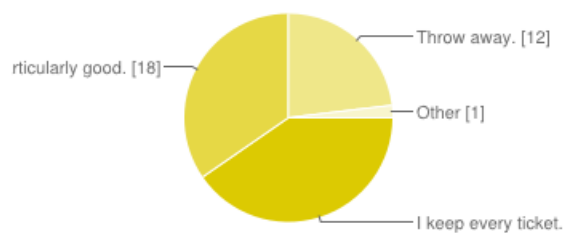
| | | |
|------------------|----|-----|
| Once a Week | 5 | 10% |
| Once a Fortnight | 15 | 29% |
| Once a Month | 28 | 54% |
| Never | 0 | 0% |
| Other | 4 | 8% |

Analysis Commentary:

This shows that 92% of the participants attend an event at least once a month. This question is not greatly accurate as it requires people to estimate when they attend events, but it could be certainly considered that 92% of people asked attend ticketed events often

-Participants attitude to the ticket when the event is over.

After attending an event, what do you do with the ticket from that event?



I keep every ticket.
I keep tickets of events that were particularly good.
Throw away.
Other

Results:

| | | |
|---|----|-----|
| I keep every ticket. | 21 | 40% |
| I keep tickets of events that were particularly good. | 18 | 35% |
| Throw away. | 12 | 23% |
| Other | 1 | 2% |

Analysis Commentary:

Although 23% of the people asked throw away their tickets, this could be for the reason that they have no reason to keep them, as there is currently not a service them into doing so. A rather surprisingly high amount of people said they keep all their tickets, which was not expected.

- Where the participant keeps there tickets if they are kept.

Responses:

Don't keep them, Shoe box, top drawer, a ticket box, a shoe box, pinned to notice board with other tickets, in a box, Folder at home, in a box, in the man drawer.

Analysis Commentary:

A nice selection of responses from people who keep their tickets in a safe place to people who simple don't keep them.

- What the 1st event the participant can remember going to.

Responses:

Franz Ferdinand, Spice Girls, Aberdeen vs. Celtic at Pittodrie, Aberdeen vs. Raith Rovers Scottish Cup 1993, Bring me at the horizon gig at Moshulu, 1998 Scotland vs. Faroe Islands, Scotland vs. Japan Rugby.

Analysis Commentary:

Lots of sporting responses here, the purpose of this question was simply to see if people remember.

-Invite the participant to explain in detail why this event has been remembered.

Responses:

It was at the exhibition centre it was really good. My friend Justin was there and we had a top laugh, Amazing, Posh Spice just got her hair cut, Getting shafted 5-0 but apart from that very little, I remember celebrating the goal but not the goal itself. I also remember having juice in a small coca cola cup being squished at the front, Meeting the sexy boys afterwards, Scotland won 2-1 and it was

to qualify for Euro 2000, brilliant atmosphere and a great game, It was the first underground club night I went to, the crowd and atmosphere is unforgettable, went with mum and sat in the balcony

Analysis Commentary:

These responses show that people have memories of events that are “unforgettable”. They are not always positive, but they exist.

-What the best or most memorable event the participant has attended.

Responses:

Underworld, Rockness, My 1st Rockness, Stone Roses Heaton Park, Rockness, Eric Prydz @ Amnesia, Porter Robinson at Colours @ the archers in Glasgow, Outlook festival 2010.

-Why

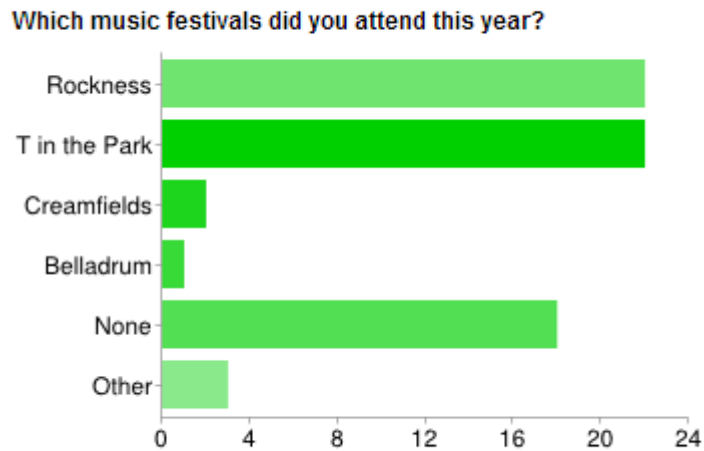
Responses:

The people I was with, Brilliant 1st time at a festival, The whole weekend surrounding the event made the actual event far more enjoyable, The gig was almost a one off as it was a special home coming performance in their home city, The atmosphere, My 1st time in Ibiza and it was unlike anything else I had ever been to before, Cold air ice cannons shooting the crowd on every drop, brilliant music and the atmosphere was amazing thanks to the venue, had seen flux pavilion earlier that night which made the main act even more enjoyable

Analysis Commentary:

Mentions of atmosphere, as well as new experiences. These experiences we have at events are only within memories. If the participants are able to tell a survey what they were like then they would have no problem submitting these feelings and descriptions to an application.

-Find out about which music festivals have been attended this year.



Results:

| | |
|---------------|----|
| Rockness | 22 |
| T in the Park | 22 |
| Creamfields | 2 |
| Belladrum | 1 |
| None | 18 |
| Other | 3 |

Analysis Commentary:

This was presented as a checkbox so participants could select more than one. Out of 52, 34 attended at least one festival.

-Find out the most memorable moment of the festival.

Results:

Finding a chair at Two door cinema club, Afterburner, Seeing Paul Kalkbrenner in the slam tent at T in the Park, The big wheel, seeing the Stone Roses @ T in the Park, 1st time I seen calibre play, Standing on the top of the Afterburner and getting chucked off at Rockness, Silicone Soul playing

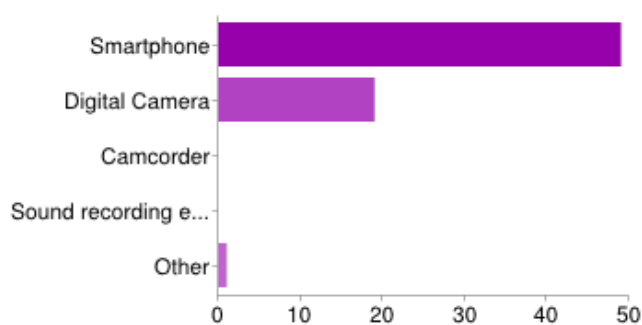
“lost in Music”, Speaking to random people, Glimmer twins closing set of the Sunday night of the Afterburner.

Analysis Commentary:

An array of experiences: Favourite songs, random moments and amazing artists.

-Type devices the participant takes to an event

What type of media devices so you take with you to events?



Results:

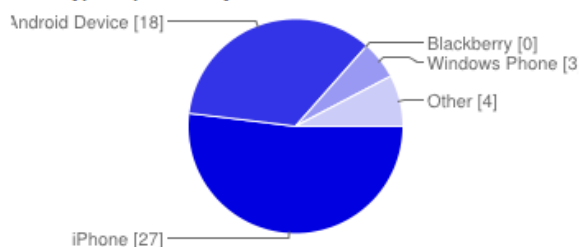
| | |
|---------------------------|----|
| Smartphone | 49 |
| Digital Camera | 19 |
| Camcorder | 0 |
| Sound recording equipment | 0 |
| Other | 1 |

Analysis Commentary:

This again was checkboxes so people could have more than one. The majority of participants take at least one type of device to the event.

-Type of mobile device the user has.

What type of phone do you have?



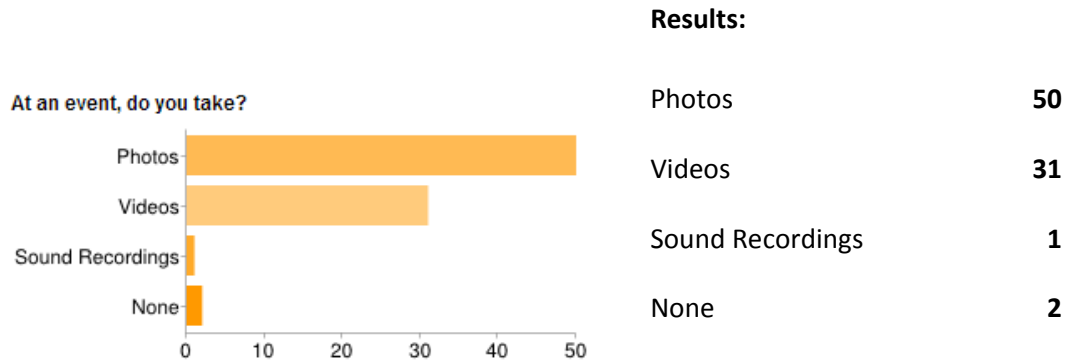
Results:

| | | |
|----------------|----|-----|
| iPhone | 27 | 52% |
| Android Device | 18 | 35% |
| Blackberry | 0 | 0% |
| Windows Phone | 3 | 6% |
| Other | 4 | 8% |

Analysis Commentary:

Nearly all participants in the survey own a smartphone, with over half having an iPhone.

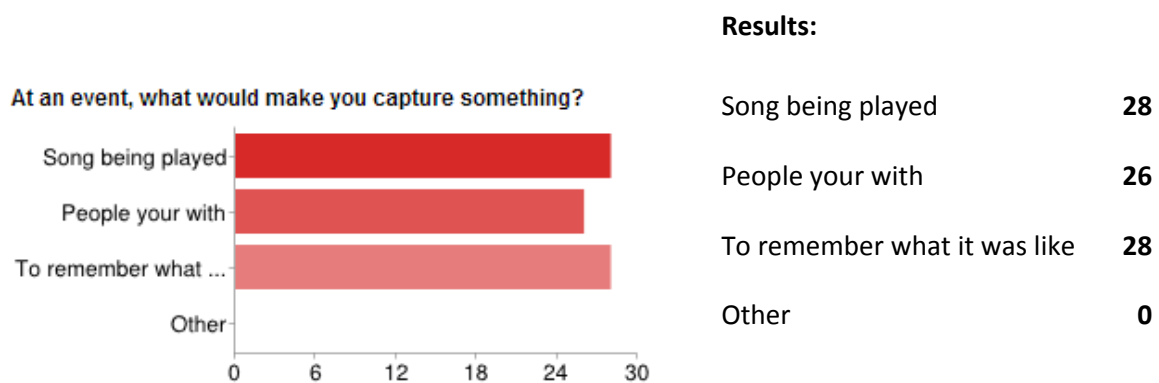
-Type of content created at an event.



Analysis Commentary:

Out of the 52 surveyed, 50 take photos most likely with a smart phone.

-The reasons that trigger the need to create content.



Analysis Commentary:

Participants could select more than one showing there are no specific ways that will trigger content generation universally.

-Invitation to voice concerns about Facebook and any issues the participant has with Facebook's service.

Responses:

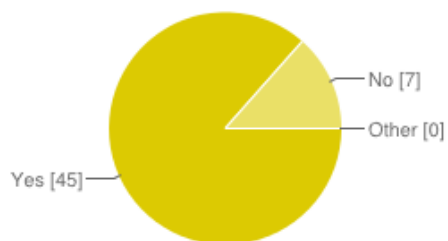
Annoying Like pages, The people, twitters a lot better, it's getting old, it allows people to express their views to hundreds of people very quickly which can cause offence and hurt to people who otherwise may not have seen/heard what has been said, It allows you to see a lot into other people's lives which can cause jealousy and upset, Facebook is an easy way to find/snoop on people which causes bullying in some cases, Too much arguments

Analysis Commentary:

Quite negative remarks regarding Facebook, a service which all of the participants still use even though they feel this way.

-If the participant has a YouTube or Google account.

Do you have a YouTube or Google account?



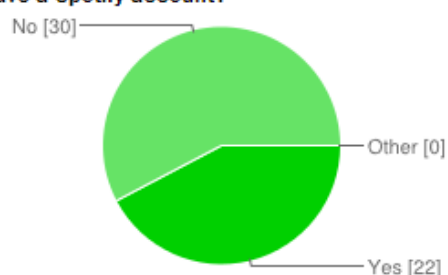
Results:

| | | |
|-------|----|-----|
| Yes | 45 | 87% |
| No | 7 | 13% |
| Other | 0 | 0% |

Analysis Commentary:

A very high percentage of participants have a YouTube account.

Do you have a Spotify account?



-If the participant has a Spotify account.

Results:

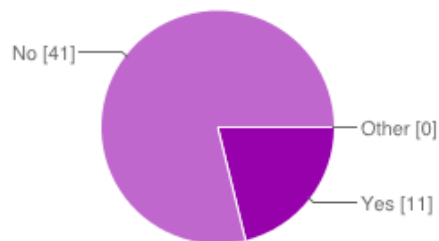
| | | |
|-----|-----------|-----|
| Yes | 22 | 42% |
| No | 30 | 58% |

Analysis Commentary:

It was anticipated that this demographic would have a higher percentage of Spotify users.

-If the participant has a Flickr account.

Do you have a Flickr account?

**Results:**

| | | |
|-----|-----------|-----|
| Yes | 11 | 21% |
| No | 41 | 79% |

Analysis Commentary:

Again a surprising low number of users who have Flickr, considering nearly everyone who participated said they take photos at events.

-Invitation to voice opinion on Facebook events

Responses:

They are good, Annoying, There OK, if invited and forget to decline them they send you notifications which can be irritating, It's a good idea, allowing you to get a group of people together for an activity quickly and allowing everyone to have a say, Handy, Easy to see who else is going, Notifications are

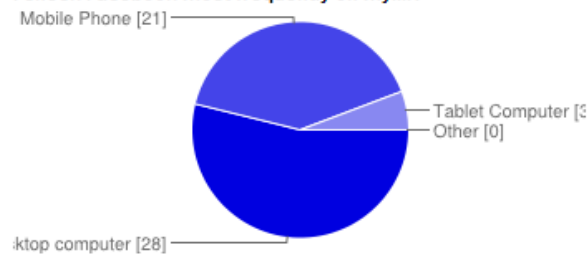
annoying and they don't allow multimedia, they are a good tool for organising nights out and birthdays, useful but not everybody engages with them.

Analysis Commentary:

Generally seen positively, Facebook events are seen as useful. This question was included to see how people felt about interacting with the service at the proposed application would be similar.

-How the user most frequently checks Facebook

I check Facebook most frequently on my....?



Results:

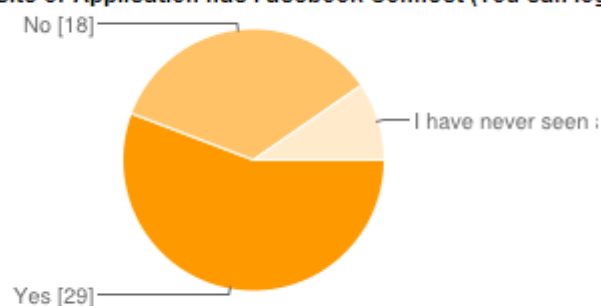
| | | |
|-------------------------|----|-----|
| Laptop/Desktop computer | 28 | 54% |
| Mobile Phone | 21 | 40% |
| Tablet Computer | 3 | 6% |

Analysis Commentary:

Clearly people are not just using PC's anymore to use the web.

-Find the users thoughts on using Facebook Connect to register or login to a site.

If a website or Application has Facebook Connect (You can login with)



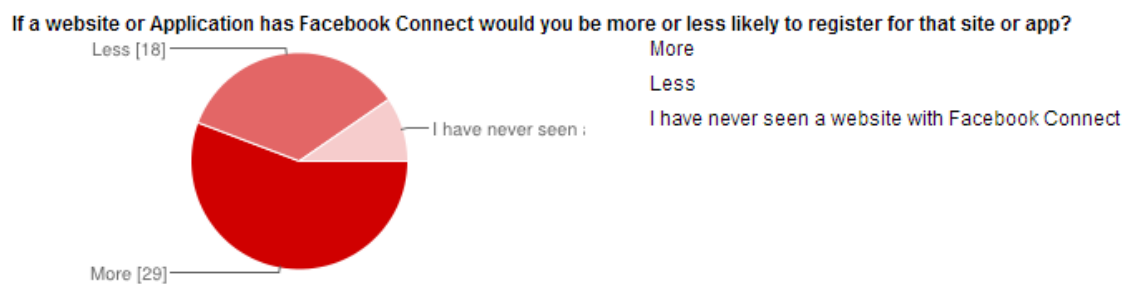
Results:

| | | |
|---|----|-----|
| Yes | 29 | 56% |
| No | 18 | 35% |
| I have never seen a website with Facebook Connect | 5 | 10% |

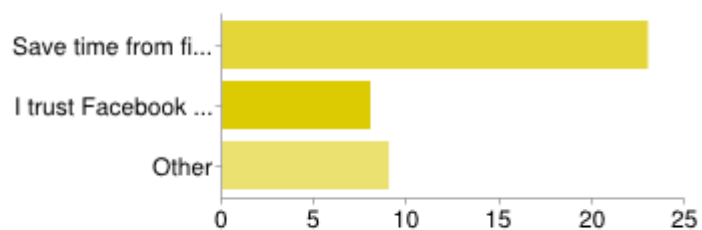
Analysis Commentary:

The purpose of this question was to find out if there was a strong feeling around how people like to register for web services.

-Invite the user to say why they would use Facebook Connect or not



Why?



Results:

| | | |
|---|-----------|-----|
| More | 29 | 56% |
| Less | 18 | 35% |
| I have never seen a website with Facebook Connect | 5 | 10% |

| | | |
|---|-----------|-----|
| Save time from filling in login details again | 23 | 64% |
| I trust Facebook to store my data | 8 | 22% |
| Other | 9 | 25% |

Analysis Commentary:

It is clear that some participants are interested in saving time and can do this with Facebook Connect.

Appendix C: Page recommendations from Google Chrome Page Speed

| Presentation layer | Page Speed Score | Recommendations |
|--------------------|------------------|--|
| Home | 58 | <ul style="list-style-type: none"> ▪ High priority. Enable compression ▪ Medium priority. Leverage browser caching, Defer parsing of JavaScript ▪ Low priority. Minify CSS, Minify JavaScript, Optimise images, Minify HTML, Specify image dimensions, Specify a character set, Specify a Vary: Accept-Encoding header |
| Login | 59 | <ul style="list-style-type: none"> ▪ High priority. Enable compression, Leverage browser caching ▪ Medium priority. Minimise redirects ▪ Low priority. Defer parsing of JavaScript, Minify CSS, Minify JavaScript, Optimise images, Minify HTML, Specify a character set, Specify image dimensions, Specify a Vary: Accept-Encoding header |
| Sign-up | 65 | <ul style="list-style-type: none"> ▪ High priority. Enable compression, Leverage browser caching ▪ Medium priority. None ▪ Low priority. Defer parsing of JavaScript, Minify CSS, Minify JavaScript, Minimise redirects, Optimise images, Minify HTML, Specify a character set, Specify image dimensions, Specify a Vary: Accept-Encoding header |
| Welcome | 72 | <ul style="list-style-type: none"> ▪ High priority. Enable compression ▪ Medium priority. Leverage browser caching, Serve resources from a consistent URL ▪ Low priority. Minify CSS, Minimise redirects, Defer parsing of JavaScript, Minify JavaScript, Optimise images, Minify HTML, Minimise request size, Specify image dimensions, Specify a character set, Specify a Vary: Accept-Encoding header |
| My box | 69 | <ul style="list-style-type: none"> ▪ High priority. Enable compression, Leverage browser caching ▪ Medium priority. Defer parsing of JavaScript, Minify JavaScript, |

| | | |
|--------------|----|--|
| | | <p>Serve scaled images</p> <ul style="list-style-type: none"> ▪ Low priority. Minify CSS, Minimise redirects, Optimise images, Minify HTML, Specify a character set, Put CSS in the document head, Specify image dimensions, Specify a Vary: Accept-Encoding header |
| Create Event | 68 | <ul style="list-style-type: none"> ▪ High priority. Enable compression, Leverage browser caching ▪ Medium priority. Defer parsing of JavaScript, Minify JavaScript, Serve scaled images ▪ Low priority. Minify CSS, Minimise redirects, Optimise images, Minify HTML, Specify a character set, Put CSS in the document head, Specify image dimensions, Specify a Vary: Accept-Encoding header |
| Event page | 81 | <ul style="list-style-type: none"> ▪ High priority. Enable compression ▪ Medium priority. Leverage browser caching, Serve resources from a consistent URL ▪ Low priority. Minify CSS, Minimise redirects, Defer parsing of JavaScript, Minify JavaScript, Optimise images, Minify HTML, Minimise request size, Specify image dimensions, Specify a character set, Specify a Vary: Accept-Encoding header |

Appendix D: Google Chrome Audit tab recommendations

| Recommendation | Reasoning |
|---------------------------------|---|
| Combine external CSS (5) | <p>There are multiple resources served from same domain. Consider combining them into as few files as possible.</p> <p>5 CSS resources served from local host.</p> |
| Combine external JavaScript (8) | <p>There are multiple resources served from same domain. Consider combining them into as few files as possible.</p> <p>8 JavaScript resources served from local host.</p> |
| Enable gzip compression (15) | <p>Compressing the following resources with gzip could reduce their transfer size by about two thirds (~210 KB):</p> <p>local host could save ~4.4 KB</p> <p>bootstrap.css could save ~68.3 KB</p> <p>bootstrap-responsive.css could save ~9.9 KB</p> <p>datepicker.css could save ~2.9 KB</p> <p>jquery.Jcrop.min.css could save ~1.3 KB</p> <p>main.css could save ~848 B</p> <p>jquery.min.js could save ~61.7 KB</p> <p>bootstrap.min.js could save ~13.8 KB</p> <p>master.js could save ~7.0 KB</p> <p>jquery.validate.min.js could save ~14.1 KB</p> <p>jquery.ui.widget.js could save ~9.9 KB</p> <p>jquery.Jcrop.min.js could save ~10.4 KB</p> <p>fbss.js could save ~1.9 KB</p> <p>script.js could save ~2.5 KB</p> <p>star.js could save ~951 B</p> |
| Leverage browser caching (19) | <p>The following resources are missing cache expiration. Resources that do not specify an expiration may not be cached by browsers:</p> <p>bootstrap.css</p> <p>bootstrap-responsive.css</p> <p>datepicker.css</p> <p>jquery.Jcrop.min.css</p> <p>main.css</p> <p>tbsh.png</p> <p>ticko.jpg</p> <p>jquery.min.js</p> <p>bootstrap.min.js</p> |

| | |
|--|--|
| | master.js
jquery.validate.min.js
jquery.ui.widget.js
jquery.Jcrop.min.js
fbss.js
script.js
star.js
lightpaperfibers.png |
| Leverage proxy caching (3) | Consider adding a "Cache-Control: public" header to the following resources:
tbsh.png
ticko.jpg
lightpaperfibers.png |
| Minimize cookie size | The average cookie size for all requests on this page is 35 B |
| Serve static content from a cookie less domain (9) | 315 B of cookies were sent with the following static resources. Serve these static resources from a domain that does not set cookies:
bootstrap.css
bootstrap-responsive.css
datepicker.css
jquery.Jcrop.min.css
main.css
tbsh.png
ticko.jpg
lightpaperfibers.png
glyphicons-halflings.png |
| Specify image dimensions (2) | A width and height should be specified for all images in order to speed up page display. The following image(s) are missing a width and/or height:
tbsh.png
ticko.jpg (2 uses) |
| Remove unused CSS rules (804) | 90.6 KB (77%) of CSS is not used by the current page.
bootstrap.css: 73.6 KB (75%) is not used by the current page.
Bootstrap-responsive.css: 10.4 KB (77%) is not used by the current page.
datepicker.css: 4.3 KB (100%) is not used by the current page.
jquery.Jcrop.min.css: 1.9 KB (100%) is not used by the current page.
main.css: 414 B (67%) is not used by the current page. |

Appendix E – Full list of HTML Errors by page

| Page | Errors | Resolved |
|--------------|--|---|
| Home | line 3 column 3 - Warning: inserting missing 'title' element
line 29 column 72 - Warning: lacks 'alt' attribute
line 54 column 9 - Warning: lacks 'alt' attribute
line 62 column 14 - Warning: lacks 'alt' attribute | Yes |
| Login | line 46 column 5 - Warning: missing </div>
line 20 column 3 - Warning: missing </div>
line 3 column 3 - Warning: inserting missing 'title' element
line 29 column 72 - Warning: lacks 'alt' attribute | Yes |
| Signup | line 20 column 3 - Warning: missing </div>
line 3 column 3 - Warning: inserting missing 'title' element
line 29 column 72 - Warning: lacks 'alt' attribute | Yes |
| Welcome | line 56 column 27 - Warning: discarding unexpected
line 85 column 99 - Warning: inserting implicit <p>
line 91 column 41 - Warning: inserting implicit <p>
line 92 column 14 - Error: <fb:login-button> is not recognized!
line 92 column 14 - Warning: discarding unexpected <fb:login-button>
line 92 column 110 - Warning: discarding unexpected </fb:login-button>
line 92 column 132 - Warning: inserting implicit <p>
line 3 column 3 - Warning: inserting missing 'title' element
line 29 column 72 - Warning: lacks 'alt' attribute
line 70 column 4 - Warning: lacks 'alt' attribute | line 95 column 11 - Error: <fb:login-button> is not recognized!
line 95 column 11 - Warning: discarding unexpected <fb:login-button>
line 95 column 107 - Warning: discarding unexpected </fb:login-button> |
| My Box | line 56 column 27 - Warning: discarding unexpected
line 153 column 9 - Warning: missing
line 260 column 5 - Warning: discarding unexpected </div>
line 261 column 5 - Warning: discarding unexpected </div>
line 3 column 3 - Warning: inserting missing 'title' element
line 29 column 72 - Warning: lacks 'alt' attribute
line 167 column 25 - Warning: lacks 'alt' attribute
line 205 column 25 - Warning: lacks 'alt' attribute | Yes |
| Create event | line 56 column 27 - Warning: discarding unexpected
line 100 column 7 - Warning: missing
line 158 column 88 - Warning: '<' + '/' + letter not allowed here | Yes |

| | | |
|------------|--|-----|
| | <p>line 169 column 182 - Warning: '<' + '/' + letter not allowed here</p> <p>line 204 column 17 - Warning: missing </p> <p>line 320 column 1 - Warning: discarding unexpected </div></p> <p>line 472 column 5 - Warning: discarding unexpected </div></p> <p>line 473 column 5 - Warning: discarding unexpected </div></p> <p>line 3 column 3 - Warning: inserting missing 'title' element</p> <p>line 29 column 72 - Warning: lacks 'alt' attribute</p> <p>line 116 column 17 - Warning: lacks 'alt' attribute</p> <p>line 116 column 17 - Warning: lacks 'src' attribute</p> <p>line 118 column 17 - Warning: lacks 'alt' attribute</p> <p>line 212 column 26 - Warning: <input> anchor 'image_file' already defined</p> | |
| Event page | <p>line 56 column 27 - Warning: discarding unexpected </p> <p>line 104 column 6 - Warning: missing </p> <p>line 112 column 33 - Error: <fb:login-button> is not recognized!</p> <p>line 112 column 33 - Warning: discarding unexpected <fb:login-button></p> <p>line 112 column 129 - Warning: discarding unexpected </fb:login-button></p> <p>line 20 column 3 - Warning: missing </div></p> <p>line 3 column 3 - Warning: inserting missing 'title' element</p> <p>line 29 column 72 - Warning: lacks 'alt' attribute</p> <p>line 157 column 46 - Warning: <div> anchor 'summary' already defined</p> <p>line 158 column 48 - Warning: <div> anchor 'summary' already defined</p> <p>line 185 column 31 - Warning: <div> anchor 'lastfmbuy' already defined</p> <p>line 191 column 27 - Warning: <div> anchor 'lastfmcase' already defined</p> <p>line 194 column 27 - Warning: <div> anchor 'tracks' already defined</p> <p>line 196 column 31 - Warning: <div> anchor 'lastfmbuy' already defined</p> | Yes |