

A Support (Workflow) Management System

Final Report

Submitted for the BSc in Software Development (with Industrial Placement)

May 2016

by

Adam Boyce

Word Count: 14,459

Table of Contents

Tables and Figures	3
1 Introduction	5
2 Aims and Objectives	5
3 Background.....	7
3.1 Problem Context.....	7
3.2 Required Components.....	7
3.3 Alternative Solutions and Feature Comparison	8
3.4 Comparison of Core Technologies.....	10
4 Project Specific Extensions	12
4.1 How to Progress	12
4.2 Where to Progress.....	12
4.3 Future Options.....	13
5 Technical Development.....	14
5.1 Chosen Technologies.....	14
5.2 Processes and Methodologies	15
5.3 System Design	16
5.4 UI Design.....	17
5.5 Test Design and System Testing	21
6 Implementation	22
6.1 Database	22
6.2 Accounts.....	23
6.3 Roles (Permissions).....	24
6.4 Tickets.....	26
6.5 Text-Based Chat.....	29
6.6 Notifications.....	31
6.7 SMS Messaging	32
6.8 Mobile Browser Optimisation	34
6.9 Mobile Application	34
6.10 Web API	35
6.11 Social Media	39
6.12 Hosting	43
6.13 Configuration	43
6.14 Testing	43
7 Conclusion	45
7.1 Project Achievements	45
7.2 Further Work with More Time	45
7.3 Evaluate Specific Tools and Defects.....	45
7.4 What Has Been Learnt	46

7.5 What Would Be Done the Same	46
7.6 What Should have been Done Differently	46
Appendix A: Model (Entities) Class Diagram	47
Appendix B: Controllers Class Diagram.....	48
Appendix C: Role Groups and Access to Controllers.....	49
Appendix D: Administration Panel	50
Appendix E: Generated Organisational Structure	51
Appendix F: Full Single Ticket View	52
Appendix G: Web Application Configuration File Sample.....	53
Appendix H: Testing Document for the Web Application.....	54
Appendix I: Testing Document for the Mobile Application.....	63
References	67

Tables and Figures

Table 1: Comparison of Support Systems	9
Table 2: Comparison of the Possible Application Types.....	11
Table 3: SMS Message Protocol Requests and Responses.....	34
Table 4: Web API Methods with Parameters and Results	37
Figure 1: Worldwide Smartphone Sales to End Users by Operating System (4Q15).....	14
Figure 2: Tasks Completed, Hours Invested and Commit Count Per Sprint (Two-week Iteration).....	15
Figure 3: Template One Design, Welcome/Home Pages	17
Figure 4: Template Two Design, Authorised User	18
Figure 5: Use of Icons to be More Efficient with Available Space	18
Figure 6: Favicon Design and Use in Google Chrome.....	18
Figure 7: Bootstrap Default Colour Palette	19
Figure 8: Initial Design of the Welcome (Index) Page.....	19
Figure 9: Initial Design of the Register Section Popup/Overlay	20
Figure 10: Initial Design of the Main Ticket View	20
Figure 11: Initial Design of the Individual Ticket View.....	21
Figure 12: Data Annotations on Entity Properties.....	23
Figure 13: Applying Roles Based Restrictions on Controllers.....	24
Figure 14: Checking for Applied Roles	24
Figure 15: Administration Options Dropdown and the Generated Organisational Structure.....	25
Figure 16: Additional Icons for Social Media	25
Figure 17: Additional Icons for Text Messaging.....	25
Figure 18: Adding/Removing Roles from Users.....	26
Figure 19: The Tickets Index Page.....	26
Figure 20: Tickets Index Page, User Information.....	27
Figure 21: Tickets Index Page, General Information	27
Figure 22: Tickets Index Page, Date Information.....	27
Figure 23: Ticket Sorting for an Internal User	27
Figure 24: Creating a Ticket	28
Figure 25: Creating a Ticket - Date Picker.....	28
Figure 26: Ticket Summary Information.....	29
Figure 27: Ticket Text-Based Communication.....	29

Figure 28: Ticket Reply for an External User	29
Figure 29: Ticket Reply for an Internal User sending an Internal Reply	30
Figure 30: Ticket Reply for an Internal User sending an External Reply	30
Figure 31: Ticket Reply with an Accompanying File	31
Figure 32: Notification Icon Difference	31
Figure 33: No Notifications for a User	31
Figure 34: User Notifications	32
Figure 35: Role Notifications	32
Figure 36: The Text Message View, with Remaining Balance	33
Figure 37: Text Message Delivery States	33
Figure 38: Web Application Optimisation on Mobile Browser	34
Figure 39: Mobile Application, Welcome Screen, Requesting Permission to Send Authorisation SMS, Adding a Reply to a Ticket	35
Figure 40: The Mobile and Web Application View for the Same User	36
Figure 41: Mobile Application Workflow	38
Figure 42: User Account External Login Options	39
Figure 43: Redirection from Twitter to Gain Application Permissions	40
Figure 44: The Application Links on Facebook	40
Figure 45: Posting as a Page/User to Facebook	41
Figure 46: Page Post from Facebook	41
Figure 47: Twitter Timeline Nesting	41
Figure 48: Detected Tweets that can be converted to a Ticket	42
Figure 49: Detected Tweet that also Exists as a Ticket	42
Figure 50: Social Media Suggestion Options	42
Figure 51: Example of Suggestions that can be Posted to Social Media Platforms	42
Figure 52: Unit/Integration Test Results	44

1 Introduction

A 'Support Ticket (Workflow) Management System', also known as a Ticket System, Ticketing System or Issue Tracking System (will be referred to a Ticket System for the remainder of this report) is a system that's primary aim is to aid support teams in interacting and managing requests from the public regarding products and/or services that they offer.

An effective Ticket System should provide a consistent means of communication for clients to raise issues (Tickets) that they have with products, this enables the support team to efficiently manage these requests internally, and to promptly return feedback to the client with information regarding their raised issue.

Although there are currently various options available, the benefits and disadvantages of some of these will be assessed, resulting in decisions of what to include in this product, what is not required and how Tickets Systems can be improved to suit the changes with how users are now interacting with support teams.

The process of this will be covered in the report, from assessing and dissecting current options, planning/designing a more suited variation, development processes, and evaluation of the process and end result.

2 Aims and Objectives

The aim of the project is to develop a Ticket or Support Management System that could be used by a company/organisation to manage the customer support work they currently have, to improve communication with the clients who raised the problems and keep them up to date on the progress of their issues, whilst innovating the category of Ticket Systems.

The project could be regarded as a success if the following three objectives are met;

Objective 1 – Web Application

A web application (Nations 2014) that can be logged into, with multiple views depending on the user type, i.e. client, team member, administrator. Primarily to be viewed on a desktop browser with ideally some functionality being viewable on smaller mobile browsers for checking up and/or replying when out of the office.

Storage

A reliable and organised database structure to keep everything stored safely in a practical and easy to interpret manner.

Create, Update, and Complete

The application should allow users to sign up and create new tickets, then interact and update them as required, with internal staff having more privileges. When the issue has been solved, the ticket should be complete/closed/done but not deleted as they can often be used for reference in the future.

Assignment and Organisation

It should be possible for tickets to be assigned to users or teams for better management, this requires the teams to be organised into these groups. This functionality should be relatively customisable but not restrictive, more as an aid for distributing work efficiently.

Objective 2 – Communication

Text-Based Chat

A primary method of communication between the client and support team, through a text-based chat that is incorporated into the ticket timeline. Contrastingly there should be a way for staff to communicate internally, to share findings, discuss the problems and consolidate information with the option of not informing the external customer. External users should be restricted to viewing only the tickets that they raised.

SMS Messages

Alongside the built-in chat functionality of the tickets, the ability to manually send, have SMS messages sent out automatically at certain events and be able to use incoming messages as an input for events would provide a lightweight interface that should enable remote use with poor network reception.

Objective 3 – Project Specific Extensions

Social Media

To incorporate aspects of Social Media (*Boyd & Ellison 2007*), including manual posting, have outward posts occur on events (that would be configured as milestones/achievements) and to hopefully use inward bound interactions as methods to raise events, for example, create new tickets with specific keywords. To expand the audience who could use the service and help the support team build a positive, friendly and active online presence.

Mobile Device

As “*the market of mobile devices is one of the fastest growing*” (*Petsas et al. 2013*), the addition of a Mobile Application would be beneficial to keep up with the expectation that consumers have, but would be considered a supplementary addition to the main project, depending on the progress.

API

Providing a usable API (Application Programming Interface) (*Dun & Bradstreet Corporation 2014*) enables the expansion of the system, whether it be to enable the use of a Mobile Application, link into other systems or have a lightweight way of getting and receiving data. An API enables user customisation and bespoke implementations, outside the realms of the Web Application itself.

These objectives will be the concluding factors to the success of the project overall depending on the extent to which they are implemented. The following chapter will summarise the reasoning behind the project concept and why these objectives were determined to be the measure of success.

3 Background

To further develop the objectives, a Ticket System can and has, taken many forms and still be a usable tool for a support team and improve customer service. This could be as simple as a paper-based system (*Rouse 2007*), mobile (*Federal Trade Commission 2012*), desktop (*Microsoft 2010*), web-based (*Pop 2000*) application, or a combination of all.

3.1 Problem Context

With companies and organisations having a vastly increased product reach than ever before (*Young 2013*), especially with software-based services, the global reach now means that companies and organisation who provide these services have to deal with the complications and problems from a larger and more varied customer base. In some companies, the demand and process of customer service can easily overcome previous technologies such as telephone, email and live chat, and improved ways of sorting and structuring these requests are essential to handling the demand and pressure that consumers are starting to expect (*Adobe Systems Incorporated 2010*). This is especially relevant as customer demand continues to increase with the growth of text and voice based service questions up fourteen and 1.4 times respectively since 2006 (*Kodak Alaris Inc. 2015*).

With some of the previous technologies used, there is a problem of individualism, in that certain members of a support team may do things differently and provide inconsistencies with the service the consumer receives. There are also problems with specific instances being solely assigned to individual users, contact via email, for example were absences or mistakes of a single support member could affect individual cases because of this (*BSI Group 2014*). A solution to this is to use a standardised Ticket System throughout the support team, this provides consistency of responses, opens up the work to all team members instead of issues only being restricted to a single user's account, the work can be prioritised and organised as a team, dormant/forgotten problems can be brought to the team's attention, so can excessive demand from specific users/organisations.

3.2 Required Components

For any Ticket System that is not paper-based, and to cover the objectives previously mentioned, the most suitable architecture would be a primary server with numerous clients (*Gunther 2011*). These clients could be a variety of different application types, for example, web, mobile and desktop, and the server should handle each client as required.

3.2.1 Server

The primary task of the server in this configuration would be to provide the clients with the information they require to saturate their current view, and then handle any changes that are requested. This is then reflected within a central data store (*Buffalo Technology 2010*) so that future requests are relevant.

3.2.2 Clients

The responsibilities of the clients would be to authenticate (*GlobalSign 2016*) the user, then present the information requested from the server, whilst updating the server with any user interaction that requires retention. This should be done in a way that is relevant to the user depending on the client type, for example, less information may be beneficial on a small mobile device in comparison to a full-screen desktop web application as “people who use both desktop applications and mobile device applications tend to use the two in different but complementary ways” (*Salmre 2004*), this should be reflected accordingly with what and how the information is displayed.

3.3 Alternative Solutions and Feature Comparison

Currently, there are numerous options that provide a similar experience to each other. It can be seen from *Table 1: Comparison of Support Systems* that there are some similarities and differences between the offerings, and now a large proportion are progressing and adding features onto the standard Ticket System to provide a fuller service to the support team/s and provide a competitive edge (Albrecht et al. 2013).

A modern Ticket System in its simplest form would consist of a means of users creating new tickets and then teams accessing these, update the clients on the progress and then inevitably close the ticket once the problem has been addressed. The features described below cover the required aspects and then optional features that can be incorporated.

3.3.1 Cost

As expected, the paid options offer a greater amount and more progressive features, with some offering free tiers that have a subset of features in comparison to the premium offering. The low-cost/free tiers have limited functionality that addresses the core attributes of helping teams by creating tickets and basic management.

3.3.2 Hosting

As with any system depending on the use of a central server, the two main approaches of hosting are self-hosting (Molnar & Schechter 2010) and cloud-hosting (NIST 2011). The best solution usually depends on the end user and what is most suitable for them. Self-hosting offers far more control, enables the information to be contained in-house (this may be essential for some organisations who deal with sensitive information (Pearson 2013)), and even the possibility of altering the source code. This does mean that the company themselves are responsible for uptime, backups, security and maintenance (this again could be a positive attribute). Alternatively, there is cloud-hosting, usually, this comes with an increase in price as the provider is offering a continued service. This continues the movement of services being cloud hosted to take advantage of saving money with processes that other companies specialise in, as from surveying 1,300 companies '74% of large enterprises and 64% of SME¹s agree that cloud computing has reduced their IT costs', (Nicholson et al. 2011).

3.3.3 Mobile Applications

Despite the primary method of accessing the Ticket System being via a web browser (Grosskurth & Godfrey 2006), mobile applications can be more appropriate when not in close proximity to a desktop, and a dedicated mobile application can often provide an improved experience over a mobile site. A dedicated application removes some of the problems when developing and testing for multiple browsers (Hewlett-Packard 2014) (especially with the vast amount of different web browser combinations now available across all platforms). It should also provide a better experience in areas of low quality/speed internet connections as there is considerably less information to transmit between the server and client. There are also some caveats to dedicated applications, there is the impact of the resources to develop, test, deploy and maintain the application on multiple platforms, then develop the functionality on the server to interact with these applications, and even then compatibility of different devices is not guaranteed (Wasserman 2010).

¹ Small and Medium-Sized Enterprises.

Table 1: Comparison of Support Systems

Name	Type	Attributes	Positive Attributes	Negative Attributes
Request Tracker (Best Practical 2015)	Web.	Self-hosted.	<ul style="list-style-type: none"> - Free. - Basic reporting. - Email - API for Development 	<ul style="list-style-type: none"> - Limited functionality. - Complicated to use.
Spiceworks (Spiceworks 2015)	Web. Mobile.	Self-hosted or Cloud-hosted.	<ul style="list-style-type: none"> - Free. - 3rd Party Applications. - Provided popular software in the past. 	<ul style="list-style-type: none"> - Limited functionality, still in progress.
Hesk (Hesk 2015)	Web. Mobile on paid tier.	Self-hosted or Cloud-hosted.	<ul style="list-style-type: none"> - Free (standalone). - Paid tier is a per licence. 	<ul style="list-style-type: none"> - Paid tier. - Add supported free tier.
OsTicket (OSTicket 2015)	Web. Mobile (unofficial).	Self-hosted or Cloud-hosted.	<ul style="list-style-type: none"> - Free (standalone). - Basic customisability. - Email. - Internal notes. - Basic reporting. 	<ul style="list-style-type: none"> - Paid tier (£6 to £11 per month per agent). - Has to be manually upgraded when on the free tier. - No official support on the free tier.
Vision Helpdesk (Vision 2015)	Web. Mobile.	Self-hosted. Cloud-hosted.	<ul style="list-style-type: none"> - Time tracking - 3rd Party Applications. - Email - Social Media Integration. 	<ul style="list-style-type: none"> - Paid (from £7 to £30 per month per agent).
Zendesk (Zendesk 2015)	Web. Mobile.	Cloud-hosted.	<ul style="list-style-type: none"> - Triggers. - API for Development. - Customisability. - Social Media Integration. 	<ul style="list-style-type: none"> - Paid (from £5 to £79 per month per agent). - Free edition limits to 3 users.
Freshdesk (FreshDesk 2015)	Web. Mobile.	Cloud-hosted.	<ul style="list-style-type: none"> - Team Inbox. - Team working. 	<ul style="list-style-type: none"> - Paid (from £15 to £79 per month per agent).
Helpspot (HelpSpot 2015)	Web. Mobile.	Self-hosted. Cloud-hosted.	<ul style="list-style-type: none"> - Triggers. - Customisation. - API for Development. 	<ul style="list-style-type: none"> - Paid (from £11 to £32 per month per agent).
JitBit Helpdesk (JitBit 2015)	Web. Mobile.	Self-hosted. Cloud-hosted.	<ul style="list-style-type: none"> - Reporting. - Customisation - API for Development. 	<ul style="list-style-type: none"> - Paid (from £600 to £1700+) and £17+ per month for cloud-hosting.

3.3.4 APIs

The addition of APIs (Application Programming Interface) (*Dun & Bradstreet Corporation 2014*) being accessible adds an element of expandability to the whole system, that can be used to access the services through more than a web browser, including mobile applications, and enables users to develop bespoke features/services for themselves that address problems not covered by the standard product (or unique to them), providing that these APIs are made public and documented. These custom use cases add extra functionality to the product without the cost of development and help build a community of users.

3.3.5 Social Media

With 90% of young adults (18-29) and 65% of older adults now using social media (*Pew Research Center 2015*) and 63% of these users expecting companies to offer customer services via social media, with one-in-three preferring it as a means of contact (*Cognizant 2015*). There are now a lot of people using social media for companies and organisations to interact with, and now not only do companies/organisations have social media accounts but in increasing cases, support teams also have, to address the accessibility issues with waiting for email/phone responses. 'Facebook and Twitter are increasingly important channels for interacting with and serving customers.' (*Cognizant 2015*), and support teams who interact via social media can often feel more accessible. This form of support also puts the customer in greater control as 'At the same time, one unhappy customer can be a potential PR disaster — making it critical to ensure that every customer experience begins and ends on a positive note' (*Cognizant 2015*).

3.3.6 Reporting

With busy support teams, over a period of time, there is the opportunity of large amounts of useful data to be collected very easily as a by-product of day-to-day operation. This data can be very useful to team leaders or managers to see how teams are performing (*Iveta 2012*). Data on total tickets opened/closed, time to respond, work done per user/team etc. is all useful in improving a team's performance and provide practical motivation/targets.

3.3.7 Customisation

Simple customisation of software can often make a service appear bespoke to the company/organisation from the external user's point of view, giving a confidence that they are using specially created systems and not just purchasing off the shelf implementations (*Kumar 2013*). Changes like styling to match the company it represents, use of logos or similar characteristics help it to appear as part of the company and not an external component.

3.4 Comparison of Core Technologies

The primary technology to provide the Ticket System will be a web application with a single web server that is associated with a database for storage. Discussed below are the alternative options including the ones decided on to compare and justify the decisions made.

3.4.1 Application Type

For the purposes of this project, the three main options of user interaction would be through mobile, desktop or web application, or a combination of all three. *Table 2: Comparison of the Possible Application Types* covers the advantages and disadvantages of each application type and some key summary attributes.

Table 2: Comparison of the Possible Application Types

Type	Advantages	Disadvantages	Summary
Mobile	<ul style="list-style-type: none"> - Portable. - The majority of people have mobile devices. - Mobile devices are now cheap enough to purchase for single tasks. 	<ul style="list-style-type: none"> - Requires a different application for each mobile operating system. - Not compatible with desktop operating systems. - Bug fixes/updates require an update and re-deployment to all test devices. - Cross-device compatibility problems. - Can be low powered. - Costly to get released in mobile application stores. - Can be hard to do complex tasks. 	<ul style="list-style-type: none"> - Portable. - Suitable for a secondary application.
Desktop	<ul style="list-style-type: none"> - Portable (if on 'desktop tablet'). - Generally more resources available. - Useful for complex tasks. 	<ul style="list-style-type: none"> - Requires a different application for each operating system type (Windows, Linux, Mac). - Not compatible with mobile operating systems. - Bug fixes/updates require an update and re-deployment to all test devices. 	<ul style="list-style-type: none"> - Can be portable. - Access to greater resources (if required). - Better for complex tasks.
Web	<ul style="list-style-type: none"> - The client can be lightweight. - Some compatibility on all devices with a web browser. - Bug fixes/updates only require an update in a single location. - Discoverability via search engines. 	<ul style="list-style-type: none"> - Problems with browser compatibility. - Most current instances are implemented this way. - Require a substantial data connection. 	<ul style="list-style-type: none"> - Lightweight. - Most clients reached for the least input. - Easier for testing during development.

The conclusion from *Comparison of the Possible Application Types* (Table 2) is that for this implementation a web application as the primary means of interaction would be most suitable. It covers the widest range of devices and means that functionality is available to every device with a web browser, with around 86% of adults in the UK using the internet regularly (*Prescott 2015*) giving this method the greatest accessibility. It also presents the least overhead when testing, as it can quickly be updated and retested, this will be advantageous for a first real application build, where more problems are assumed to be included than normal as most of the development will be undertaken for the first time. As the requirements/objectives for this project also correlate well to what a web application can provide, i.e. no need for direct hardware access, high-performance workloads or any major graphics work etc.

As it is commonplace for a lot of web accessible services to provide mobile applications, and *'today, mobile applications have become an integral part of nearly every organization's [sic]*

business strategy' (Mobility Practice Happiest Minds 2012), the addition of a simplified mobile application is seen as a worthwhile use of project time.

3.4.2 Storage

To supplement any of the application types, reliable persistent storage is essential for the functionality of the project. As the intention would be for multiple users to access the same data, a centralised solution is necessary. A document by IBM (IBM Corporation 2005) covers advantages and disadvantages of some current ways of storing data and some major summary points. From the analysis of this document, database storage is the most appropriate solution. Although it would be possible to use both text and XML (Extensible Markup Language) (Tutorials Point 2014) files as the means of storage, there are problems that would need to be manually overcome, the concurrency, reliability and security of a database is required for this type of project (Böttcher & Türling 2002). With the continued development of Object-Relational Mapping tools (ORM) (Michael Kifer et al. 2005) such as Entity Framework (Anderson & Dykstra 2014) for use with C# and LiteSQL (Oracle 2010) for use with C++, database to application interaction is now relatively straightforward.

4 Project Specific Extensions

4.1 How to Progress

As the number of companies continue to grow, with '*5.2 million small firms in the UK and they account for 48% of employment*' (Young 2015), there has been opportunity for an increase in the demand for Ticket Systems of varied configurations, offerings and prices that have features that are more suitable for the individuals and the product/service that they offer.

Despite the now vast array of different systems, from the ones researched and mentioned in *Table 1: Comparison of Support Systems* there has been a disappointing progression of the category. Only the top tier offerings are attempting to make any significant improvements and only 25% of the systems looked at had implemented social media aspects, even though 65% of American adults are using social networking sites (Pew Research Center 2015).

4.2 Where to Progress

As there is only a limited amount of time to spend developing the project, some key areas have been chosen to try and extend a standard Ticket System.

4.2.1 Social Media

As part of Objective 2, from *Aims and Objective*, the incorporation of social media into the project is one aspect believed to be a requirement now to properly reach the customers in ways that work for them. A study by Simply Measured found that in 2014, '*Mentions of dedicated customer service handles² increased by 44% year-over-year*' (Simply Measured Inc 2014), but '*average response time has slowed down by 10% YOY³*' (Simply Measured Inc 2014). Over the 31 days, 186,013 Tweets were analysed in total, from this information it can be seen that there is a demand for social media in support teams, but they cannot keep up with the requests from the customer base, with a large proportion of these requests (40%) not being responded to, '*Brands have increased their response rate by 43% YOY, now averaging 60%*' (Simply Measured Inc 2014).

² Twitter usernames/identifier.

³ Year on Year

4.2.2 SMS Messaging

Although the amount of mobile messaging applications are becoming more popular and with now 50 billion messages sent daily compared to 21 billion SMS (Short Message Service) messages (*Magnet Forensics 2014*). The use of SMS messages still has a large part to play with many more users having access to SMS compared to mobile messengers, 3.5 billion compared to 1.3 billion respectively (*Magnet Forensics 2014*). With the rise of Two Factor Authentication (*Goode-Intelligence 2012*), and the bombardment of services like email and messengers with irrelevant information, approximately 50% of email sent is spam, and that has drastically reduced from around 70% in mid-2014 (*Statista 2015*). SMS is still a relevant tool for companies, with *'about 90% of all text messages are read within 3 minutes of their delivery – and over 99% of all text message are read by the recipient'* (*Mobile Squared 2010*) causing companies to start sending important information via SMS. SMS can also be a useful means of communication for support teams as some end users of the system may depend on remote workers that do not have reliable access to the internet, and having SMS as a lightweight way of communication between users and the system may be necessary depending on the location.

4.2.3 API

A publication from (*3 Scale 2011*) claims that;

Exposing APIs can lead to a combination of business benefits that include: additional revenue channels or extension of existing channels, wider reach (e.g., increase of an organization's brand awareness), external sources of innovation (facilitating the idea of open innovation), or increase in efficiency.

As the most expensive services from the Ticket Systems looked at in *Table 1: Comparison of Support Systems*, have chosen to implement an API, it seems to be a commonality that the successful implementations have is the availability of an API, and access for users to expand the use of the service overall could be a contributing factor to their success.

4.3 Future Options

Options to be considered in the future of Ticket Systems are potentially endless. Full integration of social media channels, so that staff do not have to use the native applications for any day-to-day functionality, would be optimal. Complete automation would assumably be the end goal, where the input from the support teams can be minimised, and their efforts can be solely spent on addressing the issues raised. For example, automated answering, across all platforms, social media, phone, SMS and forwarding this on to the original source, i.e. the Twitter conversation is on the ticket and Twitter Timeline (permitting that no sensitive data is present), as this acts as a live FAQ (Frequently Asked Questions) that may prevent the recurrence of similar queries.

Detection of words that are commonly used and offering predictions of specified phrases, to try and unify responses to keep the service consistent. Detection of similar issues and prompting the user that two tickets may be in fact the same issue, that can be combined and reduce the repetition of responses.

5 Technical Development

5.1 Chosen Technologies

Covered below are some key areas that have had thought to best achieve the objectives previously set.

5.1.1 Application Types

The initially planned application type was a single web application, as from research mentioned previously, a web application is a suitable way to target the most devices and in combination with a thought for mobile resolutions, it can cover the majority of devices and provide a service to all of these through a web browser. As time and development progressed, it became a realisation that further development other than the work originally planned was attainable to extend the project and address some of the features that had been raised from *Table 1: Comparison of Support Systems*. The extra features undertaken was SMS functionality, for mobile authentication and with the addition of a simple protocol (Britannica Encyclopedia 2015) so that users with a low signal can have basic functionality via a series of SMS messages. A simple mobile application, and consequently a public API so that the mobile application (and any other device) can request and receive information in a condensed format.

5.1.2 Languages/Frameworks

There are numerous Web Frameworks (WF) or Web Application Frameworks (WAF) available to aid with the development of the main Web Application. As it would be implausible to cover every option and language, a selection of popular options at the time of writing are CakePHP (Cake Software Foundation 2016) developed using PHP (The PHP Group 2016). django (Django Software Foundation 2016) developed using Python (Python Software Foundation 2016). ASP.NET (Microsoft 2016a) to be used with any CLR (Common Language Runtime) language (Microsoft 2016c), for example, C# or VB.NET. Or Ruby-on-Rails (Rails Core Team 2016) written in Ruby (Matsumoto 2016).

With regards to the mobile application, at present, the three leading options are Google Android (Schmidt et al. 2009), Apple iOS (Apple Inc. 2016) and Microsoft's Windows Phone (Nishimura & Ueda 2013), or a combination of the three. Android is currently the most popular amongst consumers with 80.7% of the devices purchased, compared to IOS at 17.7% and Windows Phone at 1.2% (4Q15). This can be seen in *Figure 1: Worldwide Smartphone Sales to End Users by Operating System (4Q15)*, and Android growing 4.7% from 4Q14, whilst every other mobile operating system has seen a decrease in sales (Woods & Meulen 2016).

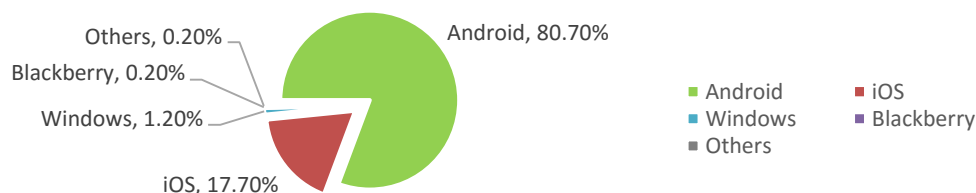


Figure 1: Worldwide Smartphone Sales to End Users by Operating System (4Q15)⁴

As the addition of a mobile application was an extension to the main project and added alongside, it did not benefit from extensive planning, and the decision to develop it solely for Android was a process of elimination. Regarding Windows Phone, the current market share is not as substantial as the other two and hard to justify spending time with the platform that

⁴ Data from <http://www.gartner.com/newsroom/id/3215217> [Accessed April 14, 2016]

is missing the reputation of the others. Developing for Apple iOS requires two resources that were not available (without substantial cost), a device running OSX (Apple Inc. 2013b) for development and an Apple iPhone (Apple Inc. 2015) for testing alongside the fact that they also have a lesser share of current consumers, the ability to develop for Google Android with the resources available resulted in an obvious decision that was achievable and economically viable.

5.2 Processes and Methodologies

5.2.1 Methodology

As the nature of this project meant that there could be continued unexpected changes at any given time, possibly due to new research developments, oversights, or a decision made on the end result. As a way to handle these fluctuations, the incorporation of a suitable methodology from the onset would provide the best chance of managing these and continue to progress forward with minimal extra overhead that can be added by using some methodologies when changes occur part way through planned work.

The methodology chosen was Scrum (SCRUM Alliance 2012), a 'holistic approach to product development' (Takeuchi & Nonaka 1986) that is a branch of the broader term Agile (Szalvay 2004). Work that required completion would be split into Product Backlog Items (Rising & Janoff 2000), these are larger items that generally would not be completed at once, and these can be broken down into manageable Tasks. The Sprints were split up into two-week iterations, where the work planned for this time was selected from the Product Backlog (the storage for the Backlog Items). The amount of planned work varied depending on the expected time designated to the project, past experience with what work could be completed per iteration and the estimated time to complete each Backlog Item. When a Task was selected to be worked on it could be migrated from three different states, 'to do', in progress' and finally 'done', upon completion the parent Product Backlog Item would then be marked as done. If the scheduled work was completed prematurely, then more Backlog Items could be included from the Backlog into the current Sprint, and if the work had taken longer than expected or less time could be spent on the project, then the remaining Tasks would generally be put forward for completion in the upcoming Sprint. *Figure 2* below shows the number of Tasks completed and the time (hours) spent on the project for each two-week iteration (Sprint) from late October 2015.

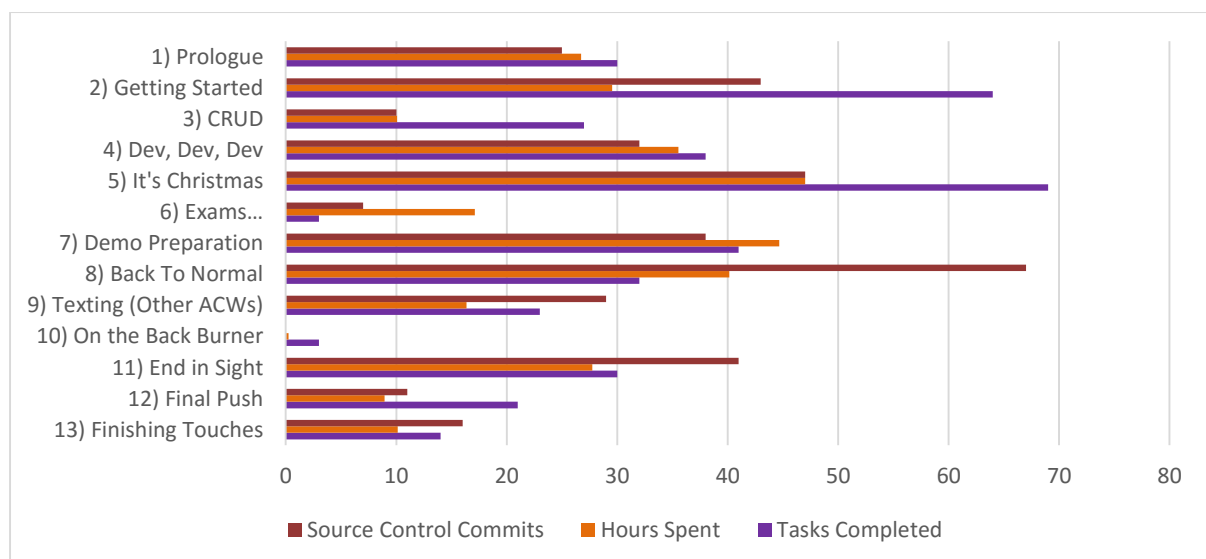


Figure 2: Tasks Completed, Hours Invested and Commit Count Per Sprint (Two-week Iteration)

5.2.2 Source/Version Control

Source control was used extensively throughout the project to help tackle the risks covered in the previous reports. The use of source control not only adds another layer of safety but also adds convenience when working from multiple devices.

There were three source code management systems considered, Git (Git 2016), Microsoft's Team Foundation Version Control (Microsoft 2016b) and Visual SVN (VisualSVN Limited 2015), hosted by the University.

The source code management system used for this project was Git, provided by Microsoft's Visual Studio Online Version Control (Microsoft 2016e). This conveniently groups everything together and combines the Scrum style features mentioned above, with the ability to commit/pull/push completed work. The Commits per Sprint can be seen above in *Figure 2: Tasks Completed, Hours Invested and Commit Count Per Sprint (Two-week Iteration)*.

5.3 System Design

5.3.1 Project Structure

As ASP.NET (Active Server Pages with and .NET language) (Microsoft 2014) was chosen to create the web application, this has the convenience of implementing the Software Architectural Pattern MVC (Model-View-Controller) (Burbeck 1992) and is fully supported in the IDE (Integrated Development Environment) Visual Studio. This enables the parts of the application to be separated out into different aspects that all have their own roles.

The Models are primarily entities that are created, modified and deleted, a class diagram can be found at *Appendix A: Model (Entities) Class Diagram*. This shows each entity, the inheritance and relationships present between them.

A Controller is usually required for each entity, and then often more to cover tasks that use more than one (or no) entities, for example, base classes or the Admin Controller that enables specific users to control multiple entities. A class diagram of the Controllers can be found at *Appendix B: Controllers Class Diagram*. These have significantly fewer relationships and are more independent as there is no need for these to interact with each other, as each Controller relates to a URL (Uniform Resource Locator) path, for example, a 'Ticket Controller' will be *'/Ticket/'* in the URL.

Although the Controllers relate to the URL paths, the end segments represent specific methods within the Controllers, and parameters can be passed as the final part of the URL or as parameters. For example, *'/Tickets/Details/5'* and *'/Tickets/Details?id=5'* would both call the 'Details' method within the Tickets Controller, and pass in parameter '5'.

Using Entity Framework and a 'Code First' approach results in a database design that replicates the design of the entities/models mentioned previously and maps the C# class instances as database tuples. This is done by declaring collections (Database Sets) that are created as Tables and the relationships declared within the classes are mirrored within the database. Due to this, the database diagram replicates the Class Diagram at *Appendix A: Model (Entities) Class Diagram*.

5.4 UI Design

5.4.1 Frameworks

To prevent time being wasted with tweaking and adjusting specific user interface components, the decision was made to incorporate a front-end library to simplify the user interface implementation. A suitable framework to assist in this is Bootstrap (Bootstrap 2016), a free and open-source set of resources (HTML, CSS, and JavaScript) that provides a series of different classes that enables the developer to incorporate these into the design that assists with aspects, such as layout, size and theme. This frees up time for building upon this with customisation and additional features.

5.4.2 Templates

The application design incorporates two main templates that are then applied to each page and the inner content is generated within them, this prevents repeated code and reduces the risk of errors from having to update multiple pages with the same change.

One template is used in the external views of the application, where a User does not have to be logged in or even a member. This has been made to look simple but attractive, with a full image background filling in for these parts of the application that, by nature have very little content, as *'Pages do have to look immediately interesting and attractive if people are to spend time, effort and, because of the communication costs, money, in viewing them'* (Dix et al. 2004). This would certainly be more applicable if it was a commercial product, so, in this case, a well-selected image would be preferred over whitespace to attract the User in, a design for this template can be seen below in *Figure 3*.

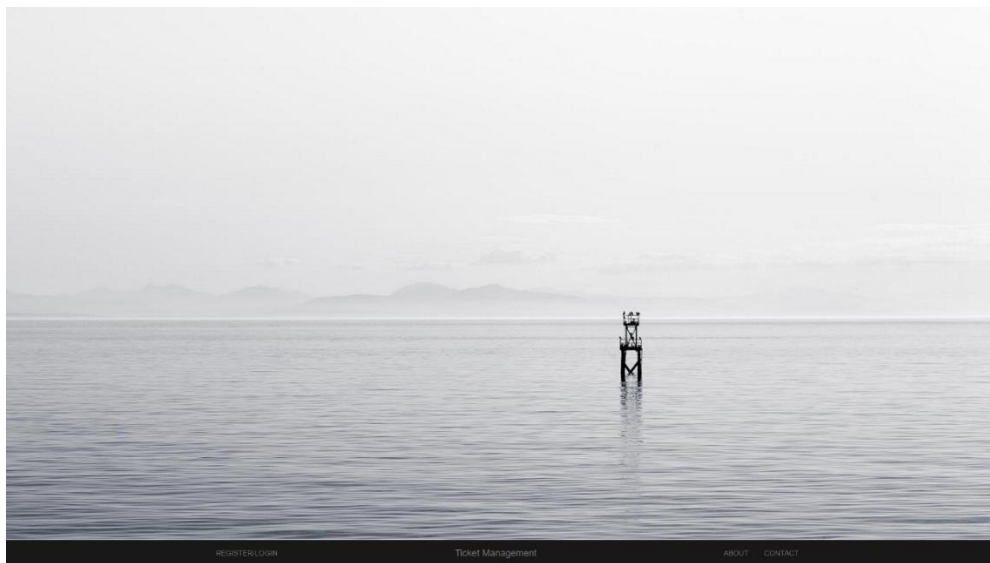


Figure 3: Template One Design, Welcome/Home Pages

The second template is the main layout of the application when the Users are logged in and using the primary functionality, this is a more minimalist approach with only the necessary components that are frequently used/required. This is mainly done through the use of Icons that *'are used in much the same way as in a standard WIMP⁵ interface to represent aspects of the functionality of the underlying pages'* (Dix et al. 2004). To ensure this is done effectively, *'the group of icons has to be designed together, with a coherent and recognizable style'* (Dix et al. 2004), this can be achieved by using a standardised font and icon toolkit such as Font

⁵ Windows, Icons, Menus, Pointers

Awesome (Gandy 2016), that helps to keep the icons consistent, recognisable and authorised. This can be seen below in *Figure 5*, and the full secondary template can be seen below in *Figure 4*.

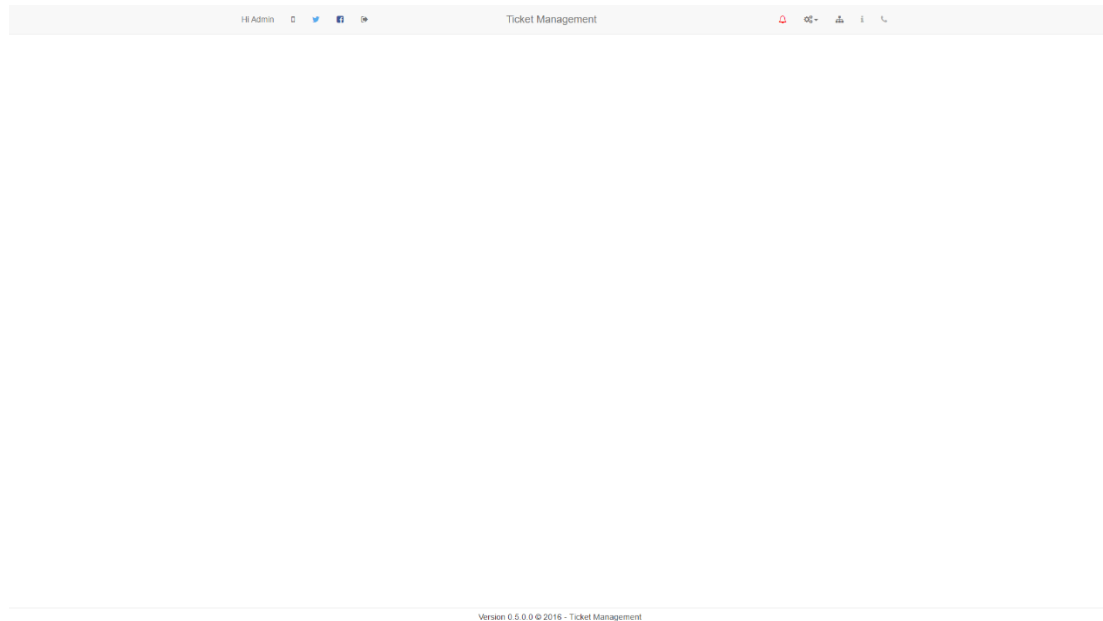


Figure 4: Template Two Design, Authorised User

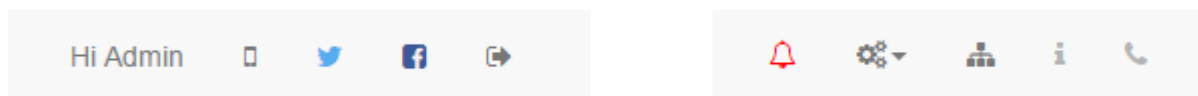


Figure 5: Use of Icons to be More Efficient with Available Space

5.4.3 Favicon Design

Although a very small design aspect, the favicon is seen whenever the web application is used within a Web Browser, for this reason, a small amount of time was spent creating a custom Favicon that can be used when each instance of the application is loaded.



Figure 6: Favicon Design and Use in Google Chrome

5.4.4 Design

The primary theme is to stick with a minimalist design that is well spaced out, with simple, consistent colours encapsulated by whitespace. *‘White space makes important content and functionality more noticeable. Also, white space can impart a sense of calm and tranquility, and it can make an app look more focused and efficient.’* (Apple Inc. 2013a). A select choice of colours will be used throughout the application design to try and keep it simplistic and use the colours to symbolise the type of action or state, the colours incorporated are part of the Bootstrap Framework that can be seen in *Figure 7: Bootstrap Default Colour Palette*. For example, with Ticket state indication or on action events (buttons) to help indicate to the User the type of action they are about to execute. With red/amber indicating possibility of danger, green for success or the intended goal and grey/blue for neutral actions that stand out from the white background, to try and re-enforce *‘that consistent use of color [sic] can help the user understand the role of particular elements more intuitively’* (Dix et al. 2004). Whilst trying to

prevent unnecessary use and inevitably diluting the meaning and irritating the User, as 'color [sic] used for no clear purpose is often distracting' (Dix et al. 2004).



Figure 7: Bootstrap Default Colour Palette

Figure 8 below shows a template of the default home/landing page that the User will see upon visiting the application, it ideally should be very simple with only what the User requires at this point in the visit. This could be their first impression of the application and they could have no previous experience, ideally, a full-screen suitable image will also be present.

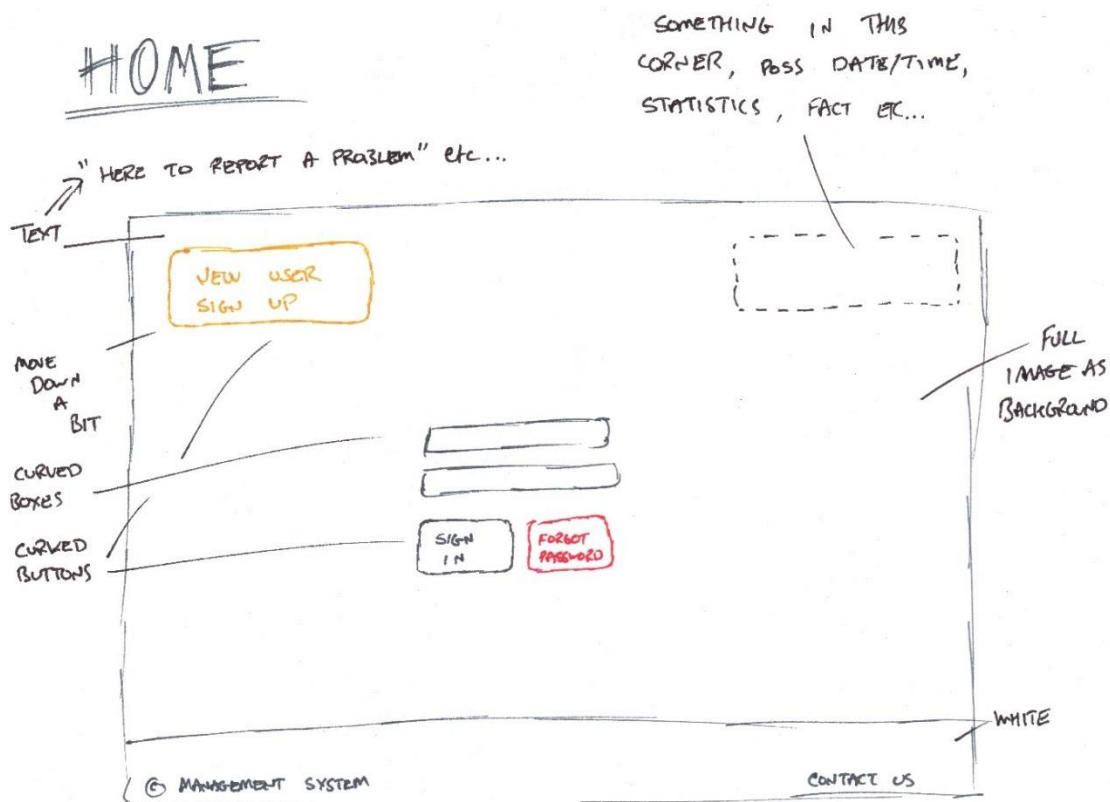


Figure 8: Initial Design of the Welcome (Index) Page

Figure 9 below shows the initial concept for a popup/overlay that will appear when the User intends to create themselves an account for the system. The decision to hide this under a button click is to try and improve the experience for the majority of Users as they should only have to do this once, then upon future visits to the application, this is hidden from view by default.

Figure 10: Initial Design of the Main Ticket View also below, shows a design of the main landing page when a User is logged in and authenticated. As the purpose of this application is Ticket management, it will default the User back here rather than the initial home page whilst still authenticated. This should show a list of the Tickets that are in the system and the User has access to. It will be using the secondary template mentioned previously that contains more whitespace and more compact information, with the use of icons/symbols compared to text so that experienced day-to-day Users have the most efficient experience.

NEW USER

APPEARS AS A POP UP ON TOP OF HOME PAGE, BACK ALL GREYED OUT AND UNCLICKABLE.

Hand-drawn sketch of a 'NEW USER' registration popup. The form includes a 'welcome' header, input fields for first name (FN), last name (LN), username (UN), password (PW), telephone (TP), and email (EM). There are two buttons: a green 'SIGN UP' button and a red 'CANCEL' button. At the bottom, there is a checkbox labeled 'INTERNAL'.

Figure 9: Initial Design of the Register Section Popup/Overlay

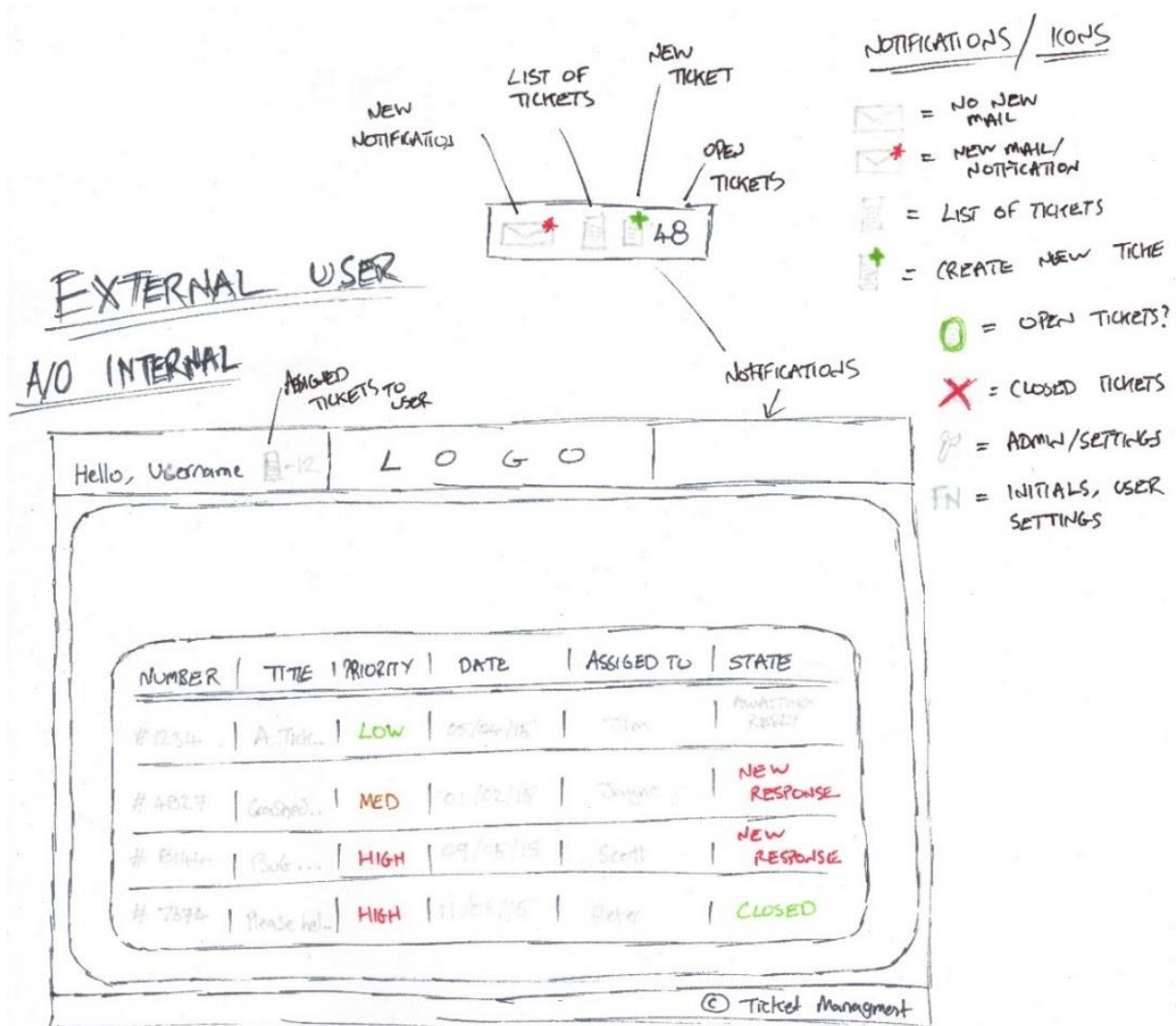


Figure 10: Initial Design of the Main Ticket View

Figure 11: Initial Design of the Individual Ticket View below illustrates the view of a single Ticket when selected, this should show a summary of the Ticket information followed by a list of text-based communication in chronological order to that it can easily be followed to the most recent event, and then further communication can be added at the bottom. Ideally, this should be of a similar style to the full Ticket view but just have the information for the Ticket that the User has selected.

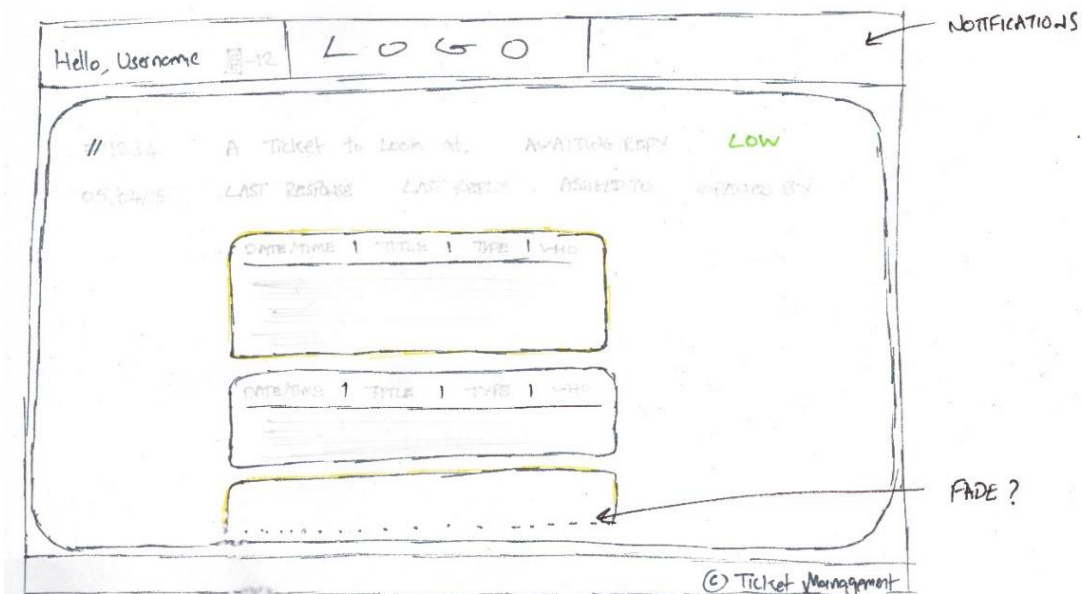


Figure 11: Initial Design of the Individual Ticket View

5.5 Test Design and System Testing

ASP.NET MVC has been designed with the ability to test relatively easily due to the separation of concerns into Models, Views, and Controllers as mentioned above. The intention is to automate testing where applicable, but as the application has a considerable amount of User input, a large amount of testing will have to be User Interface based.

5.5.1 Continued Testing

As development progressed Smoke Testing (*Chauhan 2014*) occurred naturally to see if the changes implemented had made the modifications as expected. Although this testing generally has no structure or record, it ensures that functionality remains, post change and that the area being modified continues to function as expected, so that when documented testing occurs there is some confidence that recent changes should raise minimal unexpected problems.

5.5.2 Unit Testing

When interacting with the Web API Controllers and contained methods, there is no default user interface and the output is text generally in the form of JSON (*EMAC International 2013*), that can be very hard to properly interpret for small errors, with (for the context of this project) the only user interface that uses these JSON results being the Mobile Application, and relying on an external application to test another separate project could introduce extra unknown issues. This also aids when building the Mobile Application as it is less prone to errors knowing that the data it receives has been tested previously. For all of these reasons, the Web API functionality is a suitable use case for Unit Testing (*Apple Inc. 2008*). The testing functionality

will be present in a separate specific testing project that can be run within Visual Studio with reports of failures and execution time.

5.5.3 User Acceptance Testing (Testing Document)

As both the Web and Mobile Application have a large proportion of user interaction, automated testing can often prove time-consuming and difficult. For the purpose of this project, the decision to test the User Interface and the separate functionality beneath each interaction via a series of User Acceptance (*Cognizant et al. 2015*) tests seems adequate. Provided by a Testing Document that in this case will describe the test intention, the expected result, the actual result and if these two can be considered equivalent. This approach helps to work alongside the fact that some implementation of the application may change and improve as the project progresses and technologies are learnt or adopted, as the intended functionality can be decided, and then the end functionality can be compared against this. In comparison, some forms of testing would have to be re-written/implemented as the project improved/changed, and this could waste a considerable amount of time. This type of testing can be carried out internally initially to some degree, to both ensure the testing document is comprehensible and that any failures that could prevent further testing are avoided when costly external testing is ongoing. This type of testing also forces the process of working towards a release that is worthwhile the expenditure of documented testing.

6 Implementation

6.1 Database

6.1.1 Entity Framework

As covered in System Design previously, the mapping between the database and the application (including APIs) was implemented with the use of Entity Framework in a code-first configuration, an open source object-relational mapping framework developed by Microsoft. With code-first, the database design is done via C# classes (models/entities), the properties of each type can be declared, any relationships between them, collections or shared features can all be decided within the code. A database context is created and configured that declares the collections of entities that will be translated to database tables, along with any other functionality that is related to the database interaction. For example, in this case, the Saving and Asynchronous Saving methods were overridden with logic to extract more detailed error messages to the output, to aid debugging when changes were saved to the database.

6.1.2 Design

To start with, the database design was sketched out on paper with the first considerations and assumptions of what properties each entity would require. This can then be replicated within the web application via C# and when an appropriate/usable design is implemented, it can be pushed to a previously configured Microsoft SQL Server, via a connection string that directs Entity Framework to the intended Server. Multiple connection strings can be configured to enable local databases to be used for development with increased performance as the network latency of a remote server can be removed.

To help the translation, Data Annotations are added via C#. These annotations replicate those type of restrictions generally seen within a database schema but also can be application specific, examples of these can be seen in *Figure 12* below.


```

[Required]
[ForeignKey("OpenedBy")]
[DisplayName("Opened By")]
[StringLength(250, ErrorMessage = "Opened By must be less that 250 characters but more than 2", MinimumLength = 2)]
12 references | aboyce, 132 days ago | 2 authors, 3 changes
public string OpenedById
{
    get { return _openedById; }
    set { _openedById = value; Updated(); }
}

```

Figure 12: Data Annotations on Entity Properties

6.1.3 Migrations

After the database has been created and has working data, changes may be required to alter or add functionality. This can be handled via Migrations, this functionality has to be enabled for the project and this will create a Configuration class and an initial Migration for the current state. Within the Configuration class, logic can be added to seed the database with data every time a database update is performed, for this project a selection of User Roles (covered later), default Admin User, initial Organisation and Team, some Ticket Categories, Ticket Priorities and Ticket States would all be populated to ensure the core aspects would always be present.

After a series of structural changes have been implemented, an update to the database has to take place before attempting to run the application. Declaring a new Migration can be done in Visual Studio via the Package Manager Console (a PowerShell console within Visual Studio) and given a name/identifier, this will automatically create the SQL (Structure Query Language) (Connolly & Begg 2005) required to both upgrade and downgrade the proposed changes, these can then be chained to take the database to any previous state. These can then be pushed to the database again via the Package Manager Console to update/rollback.

6.2 Accounts

The User Accounts for the application are inherited from Identity User that is part of ASP.NET Membership (Microsoft 2013a), by doing this it enables the use of the features that are already implemented as part of Identity User, such as storing passwords, basic User information, and a usable way to check for an authorised User whilst developing the application.

6.2.1 Implementation

Initially, there were some problems implementing this into the initial database design and early on there were both instances of Identity User with links to a custom User with the supplementary information required for use within the application. This approach became very unreliable when creating and editing Users, as both had to be updated and retrieved from the database together for use. A decision to invest some time on researching and combining the two into an inherited custom User resulted in some minor changes to the project, that ultimately made the User more robust and accessible.

6.2.2 User Types

Accounts are very strongly interlinked with Identity Roles (Microsoft 2016d), mentioned in further detail below, but with regards to Accounts, there are two main types of User (handled by roles), internal and external. An internal User would be anyone considered part of the support team using the software as part of their profession, as appose to external Users, these would be anyone who creates an account to raise an issue in the system to be addressed by the support team.

Although both of the User types are the same account, their type impacts the functionality that is made available to them. For example, non-internal Users cannot be assigned roles of responsibility (mentioned below) without being made Internal first, external Users can only see

and access Tickets that they have created, contrastingly internal Users can see all of the Tickets within the system. There are also restrictions on what they can add to the Tickets, as an external User can only reply to the Tickets, where internal Users can reply 'internally', this means that only internal Users can see these messages, resulting in extra information being added to the Ticket that may not be appropriate for external consumption.

6.3 Roles (Permissions)

The application has five Identity Roles, these can be checked during execution to see if the current User has access to requested information, these Roles can also be applied to both individual methods or full Controllers to limit access. For example, the Social Media Controller can only be accessed by those Users with the applicable Role. A diagram of the Roles and the Controllers/pages they have access to can be seen at *Appendix C: Role Groups and Access to Controllers*. *Figure 13* and *Figure 14* show how Roles can be applied to a full Controller and how the presence of Roles can be checked.

```
[Authorize(Roles = MyRoles.Social)]  
0 references | aboyce, 9 days ago | 1 author, 25 changes  
public class FacebookController : FacebookErrorHandlerController
```

Figure 13: Applying Roles Based Restrictions on Controllers

```
if (User.IsInRole(MyRoles.Internal))
```

Figure 14: Checking for Applied Roles

6.3.1 Approved Role

When a User creates an Account, as a security measure, it has been implemented that they will be refused access to the application until they have been assigned the Approved Role. To aid with this process it has been implemented as part of a Notification (mentioned below) that can be done with minimal interaction. This Role also has the advantage that an account can be de-authorised, but the account will remain, this could be upon suspicion of foul play, Users who no longer are part of the organisation, or as part of banning a User without removing them for misuse.

6.3.2 Internal Role

As mentioned as part of Accounts, the Role that applies whether or not a User has access to features reserved for the Internal Users (viewing all Tickets, adding Internal only replies) is with the Internal Role. This can be requested upon sign up if you are looking to be an Internal User, and again it will appear as a notification for Admin Users to quickly accept or deny the new request.

6.3.3 Administrator Role

The Administrator Role is reserved for those who are entrusted with the responsibility of full control. They have access to application configuration via an Admin Panel (this can be seen at *Appendix D: Administration Panel*) that can be used to configure all aspects and entities of the application, they also have a specific dropdown menu that can be used to quickly navigate to the area they want to change, this can be seen to the left of *Figure 15*. They can view totals of the entities present in the system (Organisations, Projects etc.) and can then edit all of these. They have access to an Organisational Structure which hierarchically shows each of the Organisations, the Teams within these and the Users/Projects associated with the Teams, a sample of this can be seen to the right of *Figure 15*, but a more complex use case be viewed at *Appendix E: Generated Organisational Structure*.

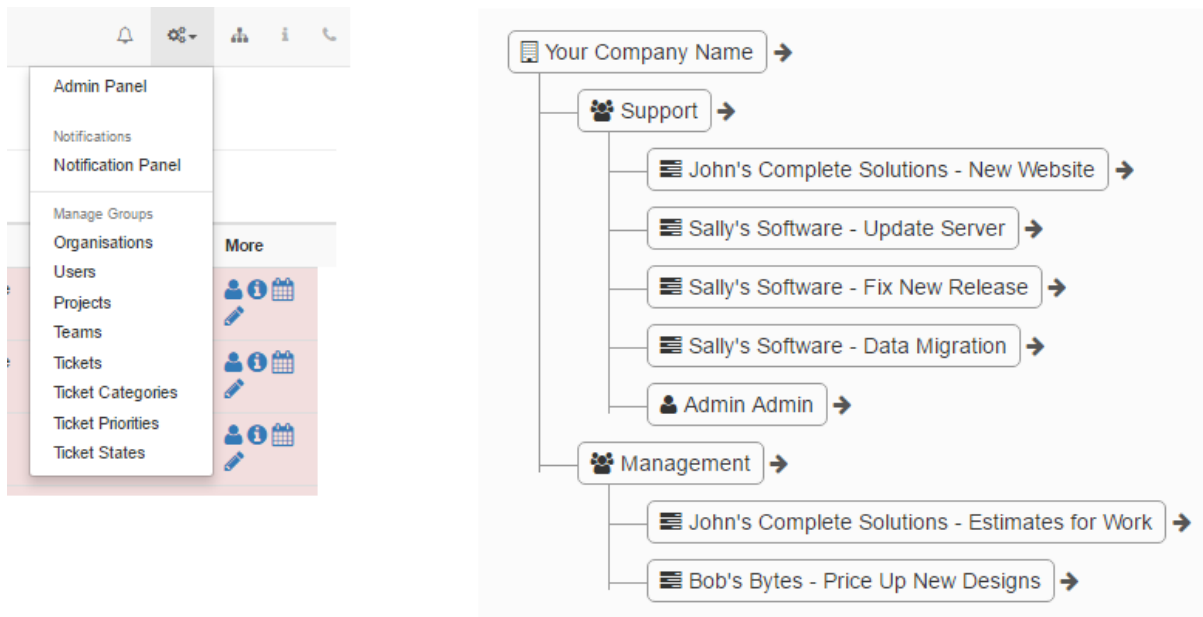


Figure 15: Administration Options Dropdown and the Generated Organisational Structure

6.3.4 Social Role

The Social (Social Media) Role is intended for use for those Users (internal) who have access to the Social Media aspects of the application, this may not be trusted to all employees, as it has the potential for reputation damage if used incorrectly. When assigned to the Social Role, they will see options for Facebook and Twitter links as seen in *Figure 16: Additional Icons for Social Media*. These additional links will take them to the respective page, discussed further below.

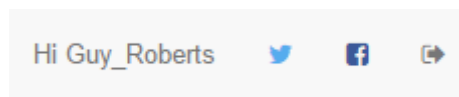


Figure 16: Additional Icons for Social Media

6.3.5 Text Message Role

For the context of this section, the Text Message Role is very similar to the Social Role mentioned above. It adds an extra link that will direct the User to the page that contains the information and functionality for handling/sending Text Messages. This addition can be seen as a small Mobile Device icon as seen in *Figure 17* directly below.

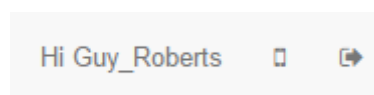


Figure 17: Additional Icons for Text Messaging

Administrators also have the ability to add/remove Roles from Users, this can be seen in *Figure 18* below.

Add Role

Role

Administrator
Administrator
Approved
Internal
Social
TextMessage

Remove Role

Role

Administrator

Remove

Figure 18: Adding/Removing Roles from Users



















6.4 Tickets

The Tickets are the focal point of the system and the default homepage for any logged in User.

6.4.1 Colouring/Styling

To aid in distinguishing the state of the Tickets quickly without having to work out dates/time manually, a colour scheme is automatically applied to the individual Ticket in which the time duration for these is User configurable. Input will be loaded from the configuration files for three values and applied when the page is rendered on the server. These times are used to calculate the time from the last external message in a traffic light style format, where Tickets that are waiting on external input are green, closed Tickets are greyed out, and those waiting on internal input, are coloured from blue, amber through to red depending on the configured durations. *Figure 19: The Tickets Index Page* shows a sample of various Tickets and how the addition of colour easily shows the state of Tickets at only a glance.

Tickets

<div> <div>Open</div> <div>Closed</div> <div>Unanswered</div> <div>Pending Approval</div> <div>Mine</div> <div>All</div> </div>								
Id	Category	Priority	Title	Team	Organisation	State	More	
1	Bug	Medium	The Edit page is not displaying correctly	Support	John's Complete Solutions	Open	  	
2	Question	On Hold	Chasing up Estimates	Support	John's Complete Solutions	Awaiting Response	  	
3	Question	Medium	Problem turning the Server on	Support	Sally's Software	Open	  	
4	Feature	Low	Adding work to the initial specification	Management	Bob's Bytes	Awaiting Response	  	
5	Question	On Hold	Can you look at this issue for me please? #AskTicketSystem	Support	John's Complete Solutions	Awaiting Response	  	
6	Question	Medium	Testing the Last Update, Last Response.	JCS - Support	John's Complete Solutions	Open	  	
7	Bug	Low	Continuing with the Ticket statistics	Management	John's Complete Solutions	Awaiting Response	  	
8	Bug	High	Another problem with the GUI	Support	Bob's Bytes	Closed	  	

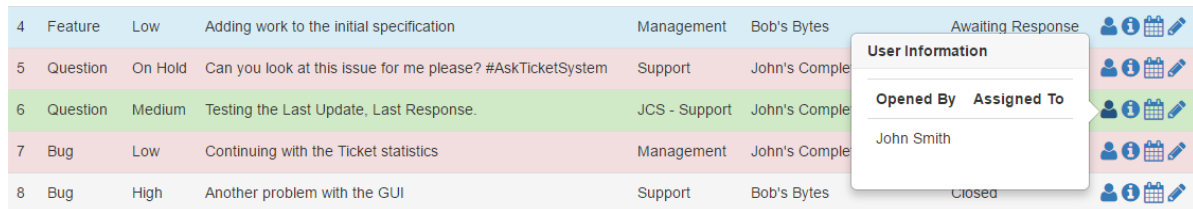
Create New

Figure 19: The Tickets Index Page

6.4.2 Hidden Information

To prevent the User having to click onto the Ticket to get more detailed information, extra categorised information can be viewed by hovering over relevant icons. *Figure 20: Tickets Index Page, User Information* shows the User Icon information, who originally opened the Ticket and if anyone is currently assigned to it. *Figure 21: Tickets Index Page, General*

Information shows extra Ticket information that is not required to distinguish the Tickets from others but users may want to quickly access if the system became more populated. *Figure 22: Tickets Index Page, Date Information* shows date related information for the Ticket such as when it was created, its deadline and information on the Tickets last messages/updates.

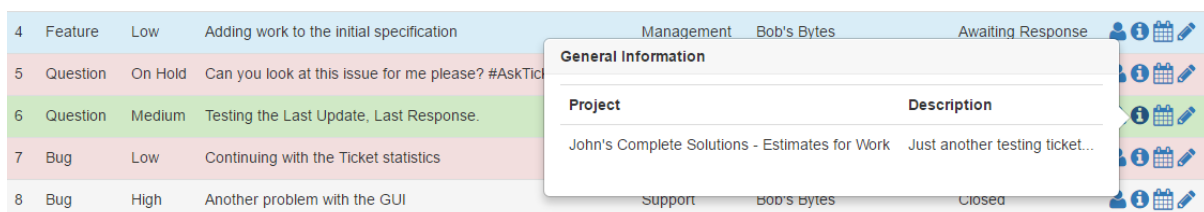


ID	Category	Priority	Description	Manager	Assignee	Status	Actions
4	Feature	Low	Adding work to the initial specification	Management	Bob's Bytes	Awaiting Response	[Info] [Calendar] [Edit]
5	Question	On Hold	Can you look at this issue for me please? #AskTicketSystem	Support	John's Complete Solutions	Completed	[Info] [Calendar] [Edit]
6	Question	Medium	Testing the Last Update, Last Response.	JCS - Support	John's Complete Solutions	Completed	[Info] [Calendar] [Edit]
7	Bug	Low	Continuing with the Ticket statistics	Management	John's Complete Solutions	Completed	[Info] [Calendar] [Edit]
8	Bug	High	Another problem with the GUI	Support	Bob's Bytes	Closed	[Info] [Calendar] [Edit]

User Information

Opened By	Assigned To
John Smith	

Figure 20: Tickets Index Page, User Information

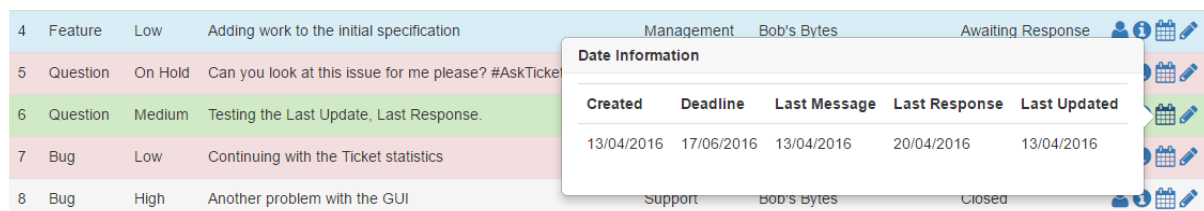


ID	Category	Priority	Description	Manager	Assignee	Status	Actions
4	Feature	Low	Adding work to the initial specification	Management	Bob's Bytes	Awaiting Response	[Info] [Calendar] [Edit]
5	Question	On Hold	Can you look at this issue for me please? #AskTicketSystem	Support	John's Complete Solutions	Completed	[Info] [Calendar] [Edit]
6	Question	Medium	Testing the Last Update, Last Response.	JCS - Support	John's Complete Solutions	Completed	[Info] [Calendar] [Edit]
7	Bug	Low	Continuing with the Ticket statistics	Management	John's Complete Solutions	Completed	[Info] [Calendar] [Edit]
8	Bug	High	Another problem with the GUI	Support	Bob's Bytes	Closed	[Info] [Calendar] [Edit]

General Information

Project	Description
John's Complete Solutions - Estimates for Work	Just another testing ticket...

Figure 21: Tickets Index Page, General Information



ID	Category	Priority	Description	Manager	Assignee	Status	Actions
4	Feature	Low	Adding work to the initial specification	Management	Bob's Bytes	Awaiting Response	[Info] [Calendar] [Edit]
5	Question	On Hold	Can you look at this issue for me please? #AskTicketSystem	Support	John's Complete Solutions	Completed	[Info] [Calendar] [Edit]
6	Question	Medium	Testing the Last Update, Last Response.	JCS - Support	John's Complete Solutions	Completed	[Info] [Calendar] [Edit]
7	Bug	Low	Continuing with the Ticket statistics	Management	John's Complete Solutions	Completed	[Info] [Calendar] [Edit]
8	Bug	High	Another problem with the GUI	Support	Bob's Bytes	Closed	[Info] [Calendar] [Edit]

Date Information

Created	Deadline	Last Message	Last Response	Last Updated
13/04/2016	17/06/2016	13/04/2016	20/04/2016	13/04/2016

Figure 22: Tickets Index Page, Date Information

6.4.3 Sorting

To pre-emptively accommodate for handling large amounts of Tickets, Ticket sorting is available as a tabbed display above the Ticket list, this functionality is available to both internal and external Users except for the 'Mine' category, that only returns the Tickets assigned to the User and this would only be applicable for internal Users with assigned Tickets. This tabbed sorting can be seen at *Figure 23* below.

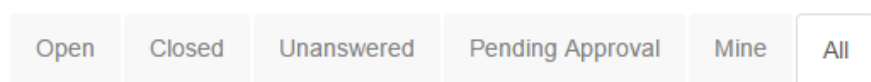


Figure 23: Ticket Sorting for an Internal User

6.4.4 Ticket Creation

Tickets can be created with a standard form, where possible drop-down selectors have been provided to remove errors with string values and to take advantage of the database structure. This adds structured data to the system as relationships between entities are used which can be edited by an administrator, in comparison to 'magic' string/values. For example, Ticket Categories can be applied to a Ticket, and this will be a database entry, not just a string such as "Question". *Figure 24: Creating a Ticket* shows the default view when creating a Ticket,

only the Title and Description have raw string inputs, the other fields are forced drop-down options, except the Deadline which uses a Date Picker that converts the value into a C# DateTime object, as seen in *Figure 25: Creating a Ticket - Date Picker* to try and prevent incorrect values.

Create

Ticket

Title

Description

Ticket Priority

On Hold

Team Assigned To

Support

Organisation Assigned To

Your Company Name

Ticket State

Pending Approval

Project

John's Complete Solutions - New We

Ticket Category

Question

Deadline

Click for datepicker

Create

[Back to List](#)

Figure 24: Creating a Ticket

Ticket Priority

Team Assigned To

Organisation Assigned To

Ticket State

Project

John's Complete Solutions - New We

Ticket Category

Deadline

20/05/2016

Create

« May 2016 »

Mo	Tu	We	Th	Fr	Sa	Su
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Clear

Figure 25: Creating a Ticket - Date Picker

6.4.5 Individual Tickets

When a User selects a Ticket from the list, it will redirect them to a separate Ticket page that is specific to that Ticket. A full example Ticket view can be seen at *Appendix F: Full Single Ticket View*. The Ticket view is broken down into a series of separate parts, or panels. The top panel takes its colouring from the colour that it had in the full Ticket list, and although only a subtle change, it reminds the User the current state of the Ticket when checking through the

information. *Figure 26: Ticket Summary Information* shows a sample Ticket summary with some information present, and the panel border colour, empty values are left vacant.

Title: Chasing up Estimates	State: Awaiting Response	Created: 19/04/2016
Description: Looking for an update on the estimates we discussed	Priority: On Hold	Opened By: Barry Thompson
Project: John's Complete Solutions - Estimates for Work	Assigned to: Clair Angon	Category: Question
Organisation: John's Complete Solutions	Team: Support	Deadline: 09/02/2016
Last Response:	Last Message:	

Figure 26: Ticket Summary Information

6.5 Text-Based Chat

The text-based chat is implemented with separate coloured panels per message, the colouring for each of these symbolises the origin of the message, with light green being from internal Users and a dark blue for the external User. Messages that have been submitted by the current User are indicated as such via a '(You)' before the Users name. This can all be seen in *Figure 27* below. The conversation is progressed with more recent messages being added beneath previous ones.

Message

John Smith @ 2/4/2016 8:31:32 AM

Hello, sorry to bug you, but we could do with an update on this matter. Thanks

Message

(You) Guy Roberts @ 2/4/2016 9:31:32 AM

No problem, we have emailed the new estimate, we will now close this ticket. Thanks

Figure 27: Ticket Text-Based Communication

6.5.1 Adding a New Reply

The functionality to add a new reply varies on the type of User who is viewing the page, with internal Users having access to more advanced functionality. The most restricted reply is for external Users, this can be seen at *Figure 28: Ticket Reply for an External User*, it only allows them to post a new reply, with the option of including an accompanying File (covered in more detail later).

New Response

Select File: (Valid types are .txt, .pdf or an image)

Choose File

No file chosen

Send

Figure 28: Ticket Reply for an External User

For internal Users, they have access to toggle the type of reply, from internal (only internal Users can see) or external, when the external reply is toggled, there is also the option to send a notification to the external User via SMS message as a reminder/notification, this is only functionality trusted with internal Users as it could easily be abused and lose the benefit of an SMS message if internal Users were getting copious text messages from external Users. To ensure the User is aware of the intention of the reply, the 'New Response' panel changes

colour in real time to replicate the decision, hiding/showing the option to send an SMS. The internal reply panel can be seen in *Figure 29* and the external reply panel can be seen in *Figure 30* (both below), with the toggle changing to red to remind the User that this is an externally published message.

Figure 29: Ticket Reply for an Internal User sending an Internal Reply

Figure 30: Ticket Reply for an Internal User sending an External Reply

Internal Users also have the option to 'Close on Reply', this will change the Ticket State to 'Closed' after the reply has been sent. This encourages Users to post a reason why they have closed the Ticket (as the send button only becomes enabled when text is present) when doing so. Adding new replies also updates the Ticket meta-data; an external User adding a new reply will update the 'Last Message' property (what the Ticket colouring is based off) and change the state of the Ticket to 'Awaiting Response'. An internal User's reply will update the 'Last Response' time and change the state to 'Open' (this is only the case if the messages are external, it would not look professional if the external Users could see messages getting updated but no reply was made to them). This all helps prevent outdated information in the system as closed Tickets should not be left marked as closed if they are still active, and this automatic functionality will prevent misinformation from human error.

6.5.2 Accompanying Files

With each reply, the User has the option to attach an accompanying file to try and help demonstrate the situation or just provide useful resources, that again are contained within the Ticket so that all of the information is contained in a single location. Upon clicking the 'Choose File' button, an Operating System based window will appear prompting the User to select a file which is uploaded with the reply, this will then be converted into a byte array to enable it to be saved within the database as a custom File entity. In the Ticket view, replies that are

associated with a File display slightly differently indicating that a File is attached. The File name is displayed as a hyperlink that will open the File in a new browser tab/window and the responsibility of displaying it is handed over to the browser, this can be seen at *Figure 31: Ticket Reply with an Accompanying File*. With testing, Google Chrome will display images, text files, PDFs (Portable Document Format) without any complications automatically.

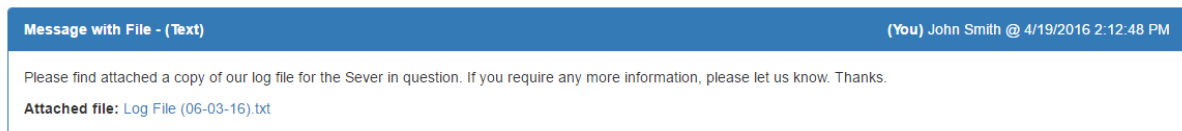


Figure 31: Ticket Reply with an Accompanying File

6.6 Notifications

Notifications are applicable to all types of Users, although there are Social Media notifications displayed with the notifications page, they will be covered as part of Social Media below. For the context of this section, the notifications refer to information that the User should be aware of in response to something happening within the system.

They are checked when the User loads a page and if a notification relates to the User the notification symbol will indicate this by changing colour from grey to red. The difference can be seen in *Figure 32*. If a User has no notifications, their notification view will be displayed as *Figure 33* below.

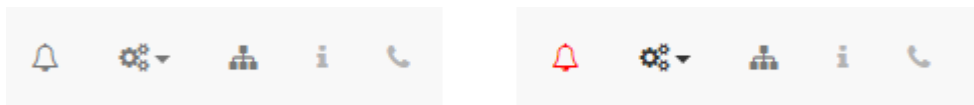


Figure 32: Notification Icon Difference

Notifications

For You

You have no notifications!

For the Roles you are in

You have no notifications for any of your roles!







Figure 33: No Notifications for a User

There are two primary types of notification, User, and Role. User notifications are any notification that is specific to the logged in User, compared to Role notifications that are sent to every member of a Role (mentioned previously), as anyone with the privileges inherited with the Role should be capable of handling the raised event.

At the time of writing, the only implemented User notification is if a change has been made to a Ticket that they are assigned to, or if a Ticket does not have an assigned User, a notification will be sent to every member of the Team that is associated with that Ticket. Example User notifications for both individual and team Ticket updates can be seen in *Figure 34: User Notifications*, for the case of a 'New Ticket Log' notification, there is no User interaction required so the only option is to 'dismiss' it.

Notifications

For You

Id	Notification About	Type	Message	Authorise	Decline/Dismiss
7	Admin Admin	New Ticket Log	New message on your teams ticket 'Chasing up Estimates' by Barry Thompson at 4/18/2016 8:14:24 PM.		
9	Admin Admin	New Ticket Log	New message on your ticket 'Problem turning the Server on' by Sally Jones at 4/18/2016 9:54:16 PM.		
11	Admin Admin	New Ticket Log	New message on your ticket 'Can you look at this issue for me please? #AskTick' by John Smith at 4/18/2016 10:05:51 PM.		
12	Admin Admin	New Ticket Log	New message on your ticket 'Can you look at this issue for me please? #AskTick' by John Smith at 4/18/2016 10:08:29 PM.		
13	Admin Admin	New Ticket Log	New message on your ticket 'Continuing with the Ticket statistics' by John Smith at 4/18/2016 10:09:38 PM.		
14	Admin Admin	New Ticket Log	New message on your teams ticket 'The Edit page is not displaying correctly' by John Smith at 4/19/2016 2:12:48 PM.		

For the Roles you are in

You have no notifications for any of your roles!

Figure 34: User Notifications

Role notifications are slightly more complex as numerous people may get the same notification if they are in the same role, and can manage it for everyone. Again at the time of writing, the role notifications implemented are for the Administrator. When a new User creates an account they have to be authorised, this is done via notifications. Any Administrator will receive a 'Pending Approval' and/or 'Pending Internal Approval' (if the new User also wanted to be internal), this can be seen in *Figure 35: Role Notifications*, from here the User can 'Authorise' or 'Decline' the request, and this will either apply the Role or ignore it, with the possibility of being added manually. These sort of notifications can aid in semi-automating tedious repetitive tasks and helps to prevent mistakes or missed events.

For the Roles you are in





Id	Associated Role	Notification About	Type	Message	Authorise	Decline
5	Administrator	Paul Green	Pending Internal Approval	The user is awaiting approval to be confirmed as an internal user.		
6	Administrator	Paul Green	Pending Approval	The user is awaiting approval before being allowed to login.		

Figure 35: Role Notifications

6.7 SMS Messaging

The sending and receiving of SMS messages is implemented with the assistance of an external service called Clockwork SMS (*Mediaburst 2016*), they provide a useful API that links up applications with Mobile Network Providers. An account has to be created with Clockwork and a generated API (Application Programming Interface) key is used to link up the application with the Clockwork account.

The way that Clockworks pricing structure works, they charge a small fee to send messages and the cost of receiving messages is the sender's responsibility. As a lack of funds would prevent the system sending messages, the current balance is retrieved and displayed when a User requests the Text Messaging views, this can be seen in *Figure 36* below.

Text Message Management

Balance £4.95	Sent Messages		
New Message	Recipient	Message	Sent DR
	Admin Admin	This is a test message from Azure hosting.	2/8/2016 11:38:03 PM
	Admin Admin	From 304	2/9/2016 11:43:26 AM
	Admin Admin	This is test from Azure, testing the new way of sending messages.	2/29/2016 6:08:56 PM

Figure 36: The Text Message View, with Remaining Balance

6.7.1 Sending

Sending a message through the application is done by using Clockwork's C# library, a new instance of a Text Message is created and simply sent via a series of method calls and checks. A custom instance of the message is also created to save within the system to have local copies of all SMS messages sent. Clockwork also offers functionality to report the delivery information of the sent messages, to receive this information from them, a publically available HTTP Post is made available that Clockwork sends a Post request to with message information. When information is received in this way, the information is extracted and then the message information can be updated and displayed on the Text Message view as symbols. Figure 37 shows an example of three delivery states being displayed, 'sent', 'in progress' and 'failure'.

Admin Admin	You have 2 new notifications!, NewTicketLog, PendingApproval	2/29/2016 8:42:14 PM	✓
John Smith	You have 1 new notifications!, PendingApproval	2/29/2016 8:42:57 PM	✗
Admin Admin	You have 2 new notifications! New Ticket Log Pending Approval	2/29/2016 10:29:34 PM	⊕

Figure 37: Text Message Delivery States

6.7.2 Receiving

Receiving a message is at no cost to the system but the sender incurs a small fee. To set up a way for Users to contact the system, Clockwork offers free keywords at specific numbers that can be chosen (depending on availability), so for example in this case by sending a message prefixed with 'TICKETSYSTEM' to 84433, Clockwork will receive this and send an XML-based Post request to a specific URL that has to be set up. The requests are then handled by the system, the information is validated and then extracted and inserted into a custom C# object in the systems database for reference to later on if required. If the system is down or unreachable, Clockwork will keep trying to send the message at decremental intervals.

6.7.3 Messaging Protocol

To try and have a system that has some core functionality in areas with a poor mobile reception, a simple messaging protocol is implemented. The concept is that if the system receives an SMS message from a number that it has associated with a User account, it will respond accordingly with dynamic information. These can also be manually sent out by a User in the Text Message Role at any time. This should help Users quickly check for critical information in areas with unreliable mobile signal. Table 3: SMS Message Protocol Requests and Responses highlights the implemented protocol requests available by the system.

Table 3: SMS Message Protocol Requests and Responses

(All messages will require the prefix mentioned above, in this case, TICKETSYSTEM)

Description of Request	Message Sent from Mobile Device	Response from Ticket System
Help request, to remind the User of the available options.	HELP_TEXT	The available options, e.g. HELP_TEXT CONFIRM_USER_TOKEN GET_NOTIFICATIONS EXTERNAL_TICKET
Notification request, any notifications that the User has.	GET_NOTIFICATIONS	The notifications associated with the User, e.g. You have 2 new notifications! Pending Internal Approval Pending Approval
Confirm User Token, this is the type of message sent by the Mobile Application to add a layer of security (will be covered below).	CONFIRM_USER_TOKEN:c53af08e-4443-4af4-9073-236536126065	No response, the Mobile Application can re-attempt access of the APIs over the network, reduces the cost of sending SMS messages.

6.8 Mobile Browser Optimisation

Although a Mobile Application has been developed, by using the Bootstrap Framework it makes developing for browsers on mobile devices much easier as they have considered how it should display with various resolutions. Because of this, throughout development, functionality has been implemented to change the layout depending on the size of the requesting browser. This is done via HTML classes to alter the positioning on the section on the screen, as the framework uses a *'mobile first fluid grid system that appropriately scales up to 12 columns as the device or viewport size increases'* (Bootstrap 2016) so changing the layout for different resolutions can be done by adding extra classes for each screen category. An example of this can be seen to the right in .

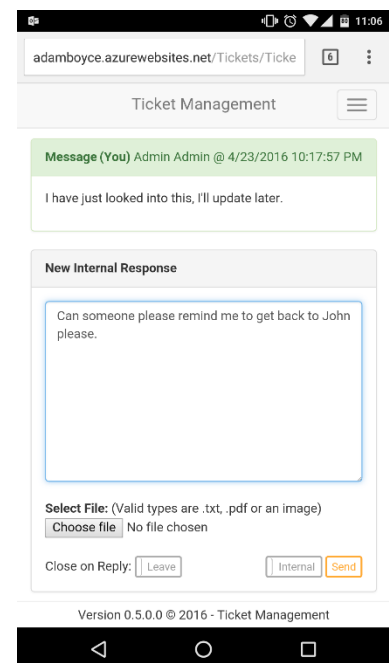


Figure 38: Web Application Optimisation on Mobile Browser

6.9 Mobile Application

The Mobile Application is developed for Android devices that is intended to be a supplementary means to aid in the day-to-day use of the Web Application. Ideally, this would gradually have increased features to enable Users to have the option of using it as a primary measure.

All of the interaction with the system is done via the Web API to get the information it requires as text/JSON and then displays that appropriately depending on the device. The requests are made using core Java HTTP requests to set up connections and read the data using Readers, this information is then passed through a custom JSON helper that converts the data into Java objects to use.

Authentication is also done via the Web API as mentioned in detail in that section below, the phone only stores the User's username and user token, and sends the SMS if required. Upon start, the connection to the server is tested, if this is successful the API call to check the credentials is attempted, upon the result of this, it will try to load the list of Tickets for the User.

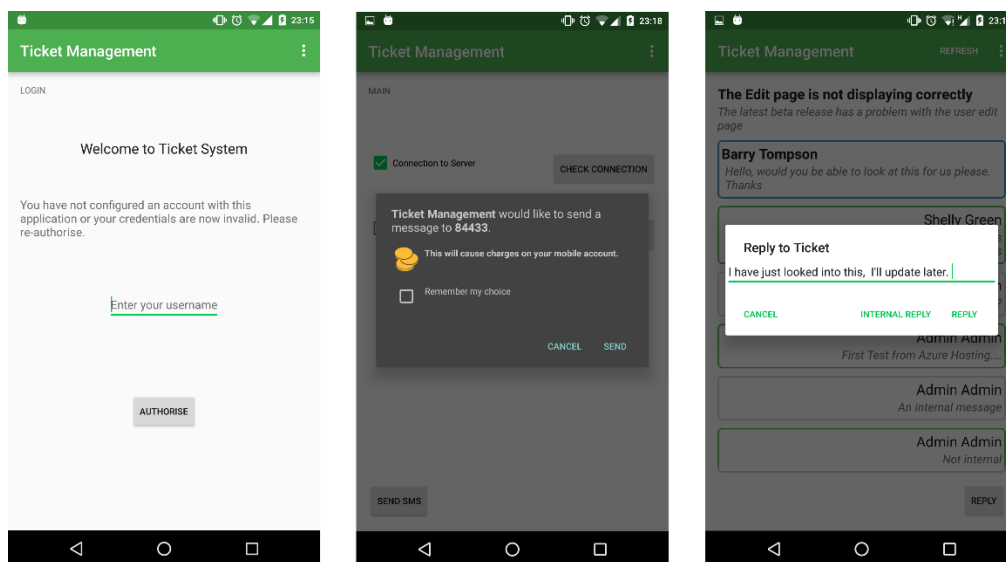


Figure 39: Mobile Application, Welcome Screen, Requesting Permission to Send Authorisation SMS, Adding a Reply to a Ticket

The colour scheme used on the Web Application is also replicated across the Mobile Application to keep consistency and prevent cluttering the limited display when extra information can be added with colouring, this can be seen at *Figure 40: The Mobile and Web Application View for the Same User*. From the Application settings the User can see the stored information and can disassociate themselves from the phone, this will also inform the Server so that it can de-authorise the User Token for that account.

A simplified diagram of the Application workflow and functionality can be seen at *Figure 41: Mobile Application Workflow* which shows the tasks that occur as separate stages of the application.

6.10 Web API

A Web API provides similar functionality to a standard web request, except instead of HTML, data in a predetermined format is returned, often as JSON (this is covered more above) that the recipient can extract and used as required.

Implementing a Web API was a prerequisite for a well implemented mobile application, and from analysis of *Table 1: Comparison of Support Systems* previously, it was noticed that a few of the available Ticket Systems had Web APIs available to build against, so with the additional time available for development, a simple Web API has been achieved.

The addition of API calls on a pre-existing web application is relatively straightforward, they just require specialised Controllers that are separated from the other Controllers. All of the API functionality is accessible with the additional path onto the standard URL, for example *'/api/Tickets/GetAllTicketsForUser'*.

Tickets

Ticket Management						
Tickets						
1	Bug	Medium	The Edit page is not displaying correctly	Support	John's	The Edit page is not displaying correctly The latest beta release has a problem with the user edit page
2	Question	On Hold	Chasing up Estimates	Support	John's	Chasing up Estimates Looking for an update on the estimates we discussed
3	Question	Medium	Problem turning the Server on	Support	Sally's	Problem turning the Server on The server beeps when turned on
4	Feature	Low	Adding work to the initial specification	Management	Bob's	Adding work to the initial specification We would like to add on some more work for the quote
5	Question	On Hold	Can you look at this issue for me please? #AskTicketSystem	Support	John's	Can you look at this issue for me please? #AskTicketSystem From Twitter User: TicketSystem
6	Question	Medium	Testing the Last Update, Last Response.	JCS - Support	John's	Testing the Last Update, Last Response. Just another testing ticket...
7	Bug	Low	Continuing with the Ticket statistics	Management	John's	Continuing with the Ticket statistics More testing...
8	Bug	High	Another problem with the GUI	Support	Bob's	

Figure 40: The Mobile and Web Application View for the Same User

As these are externally accessible, and to simplify requests that cannot easily handle the Cookie based authentication used with the Web Application, a simple authentication protocol has been implemented for the Users to add a layer of protection with regards to security.

For any request that gets/publishes important data, the request must be authenticated with an identification token (referred to as a User Token or Token), in this case, a newly generated GUID (Globally Unique Identifier) that is part of the User profile, and managed with a boolean to indicate authentication. The User can request a new User Token via the API or it can be done manually via the web interface, if the User already has a User Token and the boolean indicates it has been verified, this request for a new Token will be dismissed as malicious, if not, a new Token will be returned to the User. At this point, the Token can be authorised in one of two ways, manually via the web interface, or by using the SMS Protocol mentioned above, this is implemented automatically in the Mobile Application. *Table 3: SMS Message Protocol Requests and Responses* mentioned previously shows an example message. If the system receives an SMS Message from a number associated with a User profile, and the Token sent matches this same Users Token, we can confirm that the request was sent genuinely, all further API calls now require this specific Token as a confirmation that they are legitimate.

Although this is not fool proof, for this project it has been considered sufficient. At any point, the Token can be deactivated from either the Web Application or the Mobile Application and requests with this Token will be disregarded. *Table 4: Web API Methods with Parameters and Results* shows the available API calls and the results returned depending on what parameters are passed in. As the Web API Controllers don't have logged in Users, they are just stateless requests (restless), it has to be determined if the User is allowed to do what they are requesting, i.e. can they update a Ticket if they are an external User, so this has to be checked for each API request.

Table 4: Web API Methods with Parameters and Results

Description	URL Path	Parameters	Return Result
Default request.	/api/	N/A	A string response with all of the available API requests.
Default request, parameter test.	/api/	(int) id	A string response with all of the available API requests, and the id parameter.
Connection Test	/api/CheckConnection	N/A	True, there is no point returning anything else, if they can receive a True, they have a successful connection.
To try and get a new Access Token for a User	/api/User/GetNewUserToken	(string) username	If an error occurs or the User already has an authorised Token, null. If successful a new GUID is returned.
Check to see if the current User Token is valid for a User.	/api/User/CheckUserToken	(string) username (string) usertoken	False if incorrect parameters or the User Token is not valid. True, if the User Token is valid for the provided Username.
Deactivates a User Token for a User.	/api/User/DeactivateUserToken	(string) username (string) usertoken	False if an error occurred or there are invalid parameters. True is the User Token is deactivated.
Get all of the Tickets, that are applicable to the User (External User will only get their own Tickets).	/api/GetAllTicketsForUser	(string) username (string) usertoken	Null if invalid parameters are provided. JSON result explain the error message, if applicable. JSON Array of Ticket instances, that are simplified for API use, for example, the relationships are removed, and it is just string values, i.e. the name of the User who opened the Ticket not an instance of a User. If the User is not internal, it will check to see if they actually have access to it before returning the Tickets.
Gets details of a specific Ticket.	/api/Ticket/GetTicketForUser	(string) ticketid (string) username (string) usertoken	Null if invalid parameters are provided. JSON Object of a Ticket instance that again is simplified for API use, with string values rather than related instances. It will check that the User has access to the Ticket, either because they are internal or external and opened the Ticket.
Gets the Ticket Logs for a Ticket (messages).	/api/Ticket/GetTicketLogsForUser	(string) ticketid (string) username (string) usertoken	Null if invalid parameters are provided. JSON Array of a Ticket Logs again simplified for the user via APIs with string values over other JSON objects.
Adds an External reply to a Ticket.	/api/Ticket/AddExternalReplyToTicket	(string) ticketid (string) username (string) usertoken (string) message	False if the parameters are invalid. It will check that the User is able to add to the Ticket Id provided, and if so will add the Log to the Ticket, and return True if successful.

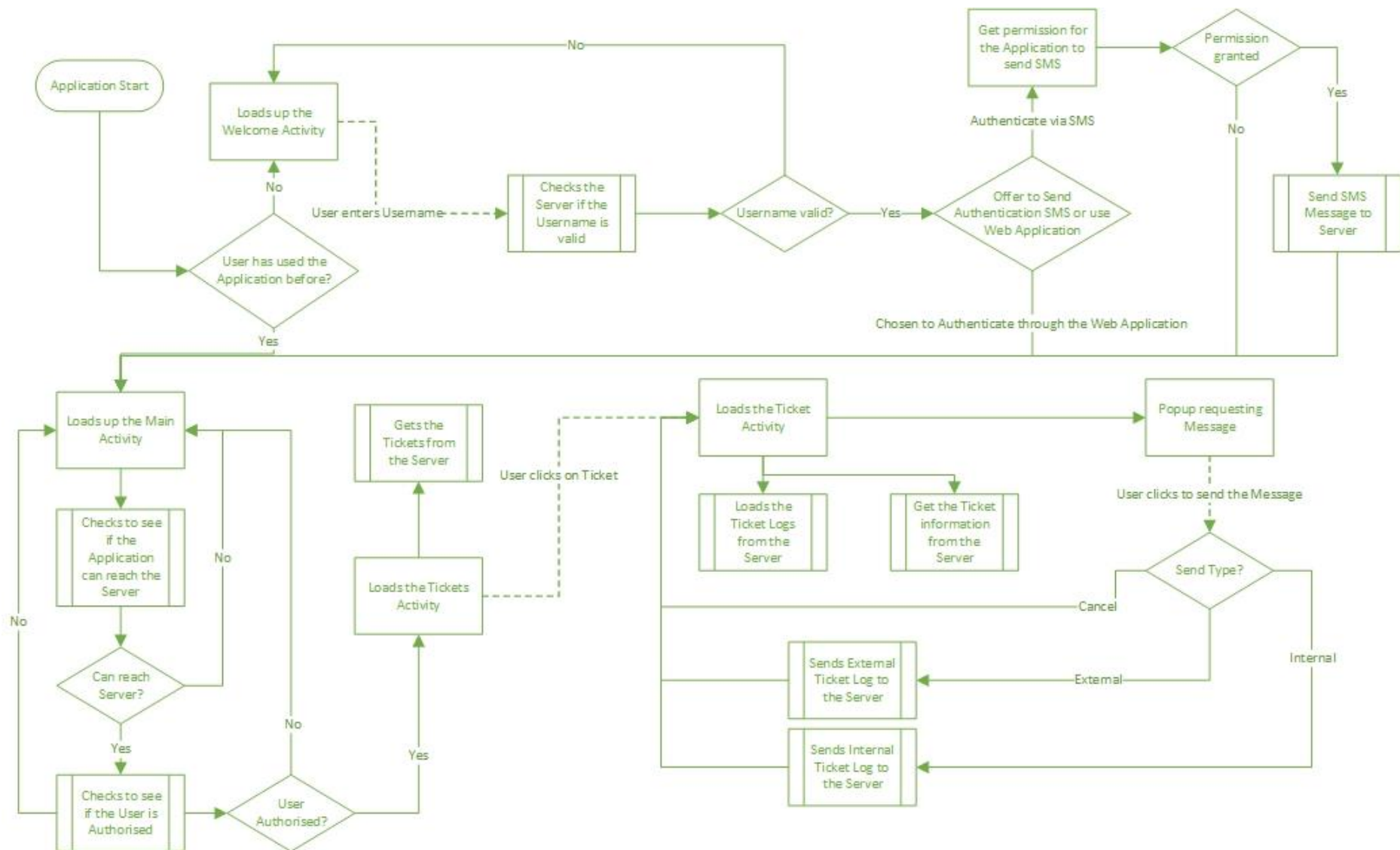


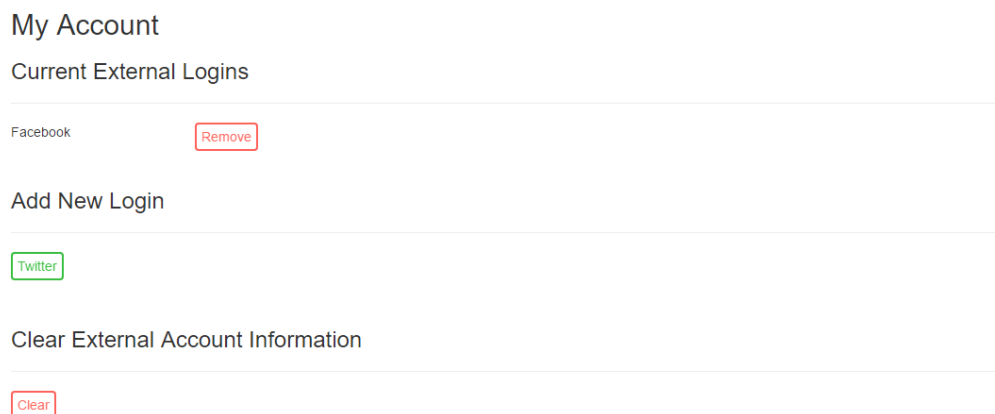
Figure 41: Mobile Application Workflow

6.11 Social Media

The Social Media integration is intended to help support teams automate their social presence with minimal User interaction. The two currently implemented Social Media platforms are Facebook and Twitter.

Interaction with these services requires authorisation, just as required for interaction normally. This is done in a similar way for both platforms. A system administrator would have to set up a virtual application (often linked to the developer part of the account), these applications can be configured depending on its type and what it is required to do, i.e. its permissions and how it will be used. These configured applications are associated with a couple of identification keys/Ids (called slightly different things for each platform).

When the User wants to use these features, they have to confirm that they are willing for the application to access and use the information it requires. This is done by directing the User to the associated page from Facebook and/or Twitter, the User can log into these accounts and then they will be redirected to the application and User Tokens can be saved to prevent this occurring for every request. These stored values are checked for each request and if not present will advise the User that they have to associate their Ticket System account with the Social Media suppliers. *Figure 42: User Account External Login Options* shows the view that is part of the User account to add/remove associations with external providers. *Figure 43: Redirection from Twitter to Gain Application Permissions* shows an example page that the application will redirect to, to acquire permissions from Twitter (if the User is not logged into Twitter within the browser this would have to be done first, as normal), this will then redirect the User back to the system.



The screenshot shows a web interface titled "My Account". Under the heading "Current External Logins", there is a list of connected accounts. One account, "Facebook", is listed with a red "Remove" button next to it. Below this section is a heading "Add New Login" followed by a green "Twitter" button. At the bottom of the visible section is a heading "Clear External Account Information" with a red "Clear" button.

Figure 42: User Account External Login Options

6.11.1 Facebook

A way that Users of Facebook can interact with companies/organisations is via Pages, Users can post to them, and see other interactions related to the topic. The Pages are generally controlled/managed by administrators, these are assigned and can edit/change the page information. The idea for this implementation is that the company/organisation would set up a Page that represents them, and then the team members that are already trusted to be page administrators can use the Ticket System to more effectively manage the page. When Facebook has been linked with the application, it can be seen in Facebook's User settings and revoked if required, this can be seen below in *Figure 44: The Application Links on Facebook*.

The interaction between the system and Facebook is via Facebook's Graph API (identical concept to the implemented API previously mentioned, just far more complex and developed).

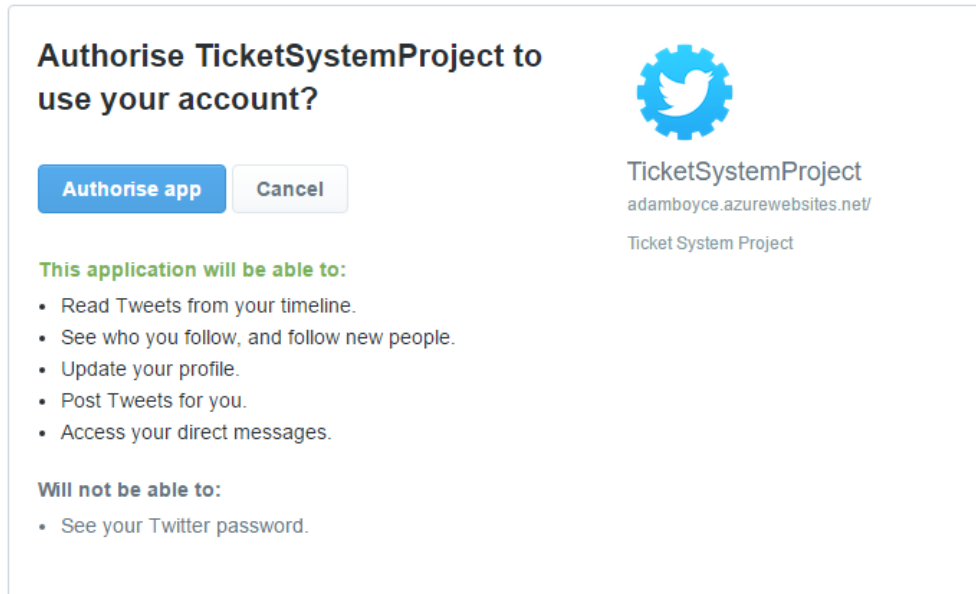


Figure 43: Redirection from Twitter to Gain Application Permissions

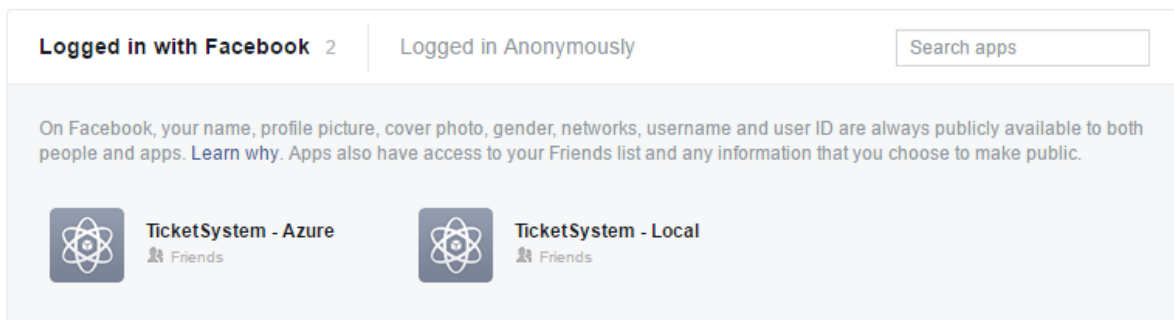


Figure 44: The Application Links on Facebook

The information requested from Facebook and then displayed within the application includes a Profile summary of the logged in User, a summary of the Page that is linked to the system (inputted via configuration), and a list of posts on the Page, that will reload in the next amount (again configurable) when the bottom of the list is reached. The posts will be detected and handled appropriately depending on the type, for example if an image/link is part of the post. There is also functionality to post to the linked Page, this can be done in two slightly different ways that have different outcomes. These are posting as the User logged in with, or posting as the Page itself, the option of including a URL is also available. The posting box can be seen in *Figure 45* below. An example post requested from Facebook can be seen in *Figure 46: Page Post from Facebook*.

6.11.2 Twitter

Twitter is perhaps more suitable than Facebook for use as a support based platform, it has the ability to direct Tweets towards other Users and add keywords (Hashtags) that can group/categorise messages.

The implementation is slightly different to Facebook; with Twitter the concept of Pages/Groups does not really exist, so a single Twitter account would be used by all of the Users of the system. The User will associate their Ticket System account with the Twitter account, and can

again be seen in *Figure 43* how this looks to the User. Once associated with each other, the User can visit the Twitter Management Page.

New Post

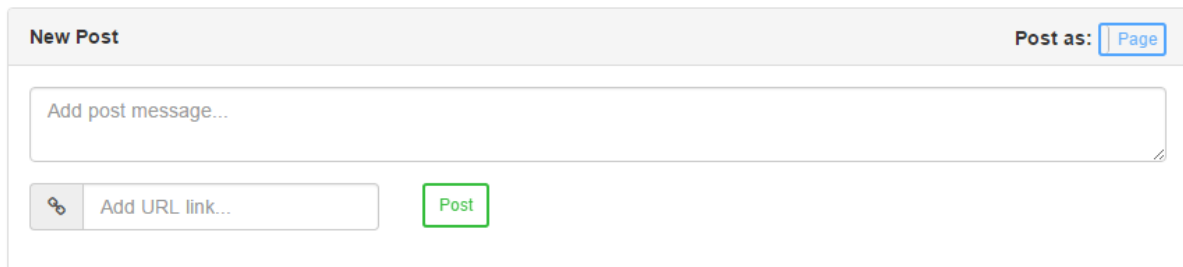


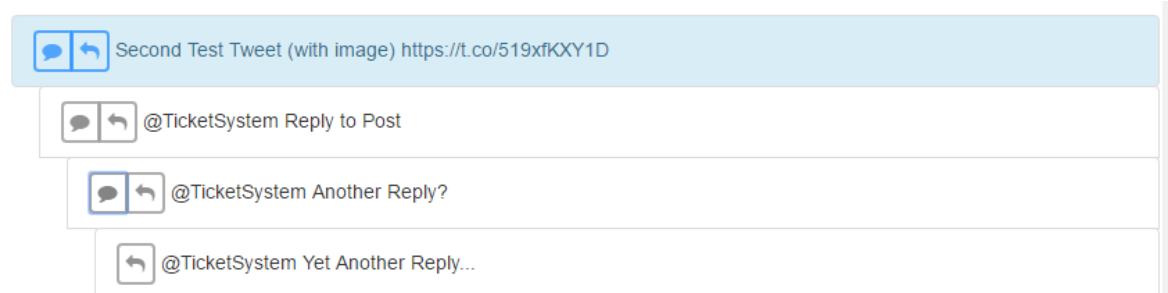
Figure 45: Posting as a Page/User to Facebook



Post Type: Link	Created: 1/30/2016 5:56:27 PM
About: Lincolnshire County Council's computer systems are closed for four days after being hit by computer malware demanding a £1m ransom.	Updated: 1/30/2016 5:56:27 PM
View Item on Facebook	Hidden: False
	Published: True

Figure 46: Page Post from Facebook

The layout of this page is very similar to the Facebook page, with a Profile summary on the left, and Timelines to the right. By default, the Home Timeline is not loaded as this by nature includes a lot of posts from Accounts that the Twitter User follows, but this can instantly be loaded/hidden with a toggle switch. The display of the User Timeline is more refined than the Facebook implementation, with reply based nesting that can be expanded or directly replied to. *Figure 47* shows an example of this, the speech bubble, will show/hide the child elements, the reply icon will prompt the User to directly reply to that Tweet.



Second Test Tweet (with image) <https://t.co/519xfKXY1D>

@TicketSystem Reply to Post

@TicketSystem Another Reply?

@TicketSystem Yet Another Reply...

Figure 47: Twitter Timeline Nesting

As the system is populating the Tweets it will check to see if any of them contain the configurable hashtag that the system detects as a directed Tweet that can be turned into a Ticket (part of the configuration in *Appendix G: Web Application Configuration File Sample*), this is indicated in *Figure 48* below with the 'plus' button, clicking this will direct the User to the create Ticket functionality in a new tab, and will auto populate data from the Tweet. Tweets that have already gone through this process and also exist in the system as a Ticket are indicated as such with a Ticket symbol (seen in *Figure 49*), this will take the User to the Ticket

instance. This functionality makes converting Tweets to Tickets very simple and removes accidental Ticket duplication, as all Users can see these associations within the system.

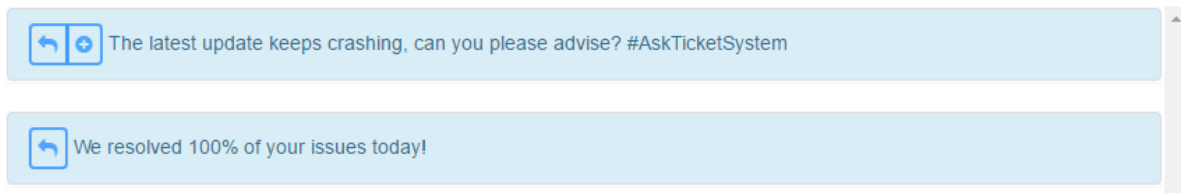


Figure 48: Detected Tweets that can be converted to a Ticket

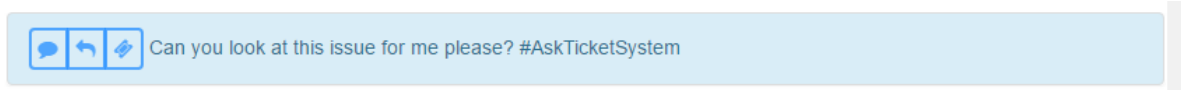


Figure 49: Detected Tweet that also Exists as a Ticket

6.11.3 Automation

As part of the Social Media automation, there is functionality alluded to previously that can help to suggest content to post to Social platforms, and then post it, upon confirmation. Areas of interest can be selected and searched, and depending on the results, the system will suggest suitable posts that reflect the support team's performance. There are search criteria that can be manually included/excluded, or all options can be toggled on/off, seen in *Figure 50: Social Media Suggestion Options* and the system will query to see if there is any significant information that is worth posting.

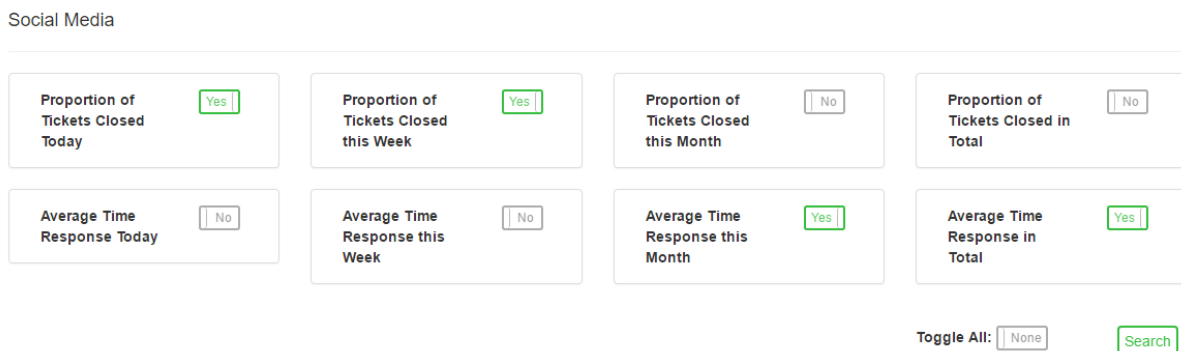


Figure 50: Social Media Suggestion Options

When the querying has completed, a pop-up window will display showing the proposed suggestions, these have three options, post to Twitter, post to Facebook, or dismiss. Upon posting, the option to post again to the same platform will disappear to prevent accidental reposting. This can be seen below in *Figure 51*.

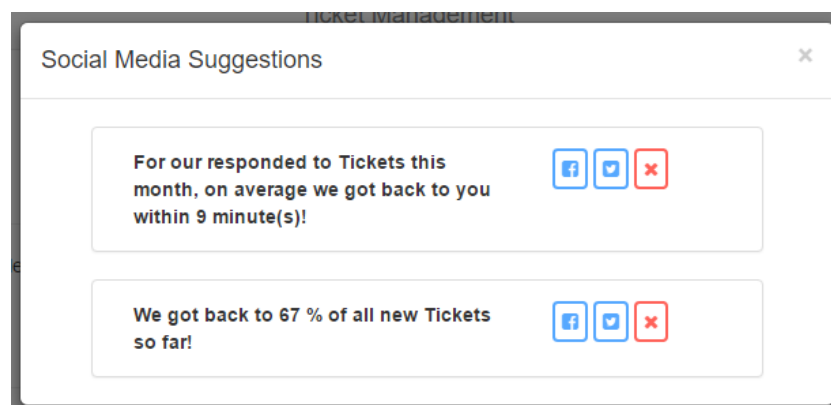


Figure 51: Example of Suggestions that can be Posted to Social Media Platforms

6.12 Hosting

Hosting has been covered in some detail previously, the best solution found was a combination of both Cloud and Self-hosting, in the sense that the end application is hosted in the Cloud, but done manually and configured by the user.

6.12.1 Previous Attempts

The final solution was not the intended implementation from the design, ideally, it was going to be completely self-hosted on a local machine running Windows Server 2012 R2 (*Microsoft 2013b*) with Microsoft SQL Server (*Mistry & Misner 2014*) and Microsoft IIS (Internet Information Services) (*Microsoft 2012*). The deployment was successful to this platform, and would still be a recommended way of the final implementation, but it had limitations with the server location as it had insufficient network transfer rate to provide a satisfactory service.

Another approach to try averting the network issues was hosting the application on an external hosting service, again this was unsuccessful. Although the actual deployment was yet again successful, due to the external service not updating their IIS version, some of the features of C# 6.0 that had been implemented throughout the system was not supported and would result in errors on pages that had taken advantage of these.

6.12.2 Final Solution

The implementation that proved to overcome the previous problems was hosting with Microsoft's Azure (*Tulloch & Windows Azure Team 2013*). Due to its integrated links with Visual Studio, publishing to Azure can be handled with a simple wizard making the process of deployment straightforward enough that regular publishing could be scheduled.

Two web applications were created, primary and testing (beta) sites, this meant that any builds that could cause problems were first published to the testing site to be tried, ensuring a functioning version was always available.

As a large proportion of the site configuration is loaded at start up from a web.config file, when the site is running in Azure these configuration keys can be overridden automatically. This resulted in local development having a different configuration, i.e. database connections can point to a local database during development, but upon publishing it would automatically use the configured database also running in Azure.

6.13 Configuration

As just mentioned above, a lot of the specific values are loaded in from configuration files, this makes the site much more dynamic and if use commercially would prevent different releases for each user. A sample of the web.config file can be seen at *Appendix G: Web Application Configuration File Sample* to show what values can be configured.

6.14 Testing

Testing the full system has been done in various ways to try and best suit the application format.

6.14.1 Testing Document

As large parts of the project are made up of user interface interaction, automated testing can be difficult and time-consuming, especially as sections can change as development progressed, and the time to automate testing of user interfaces was decided not to be a worthwhile investment.

To cover these areas, the use of User Acceptance Testing was carried out. This required documentation to be written that specifies what functionality is required to consider the application in an acceptable state to be implemented and considered complete.

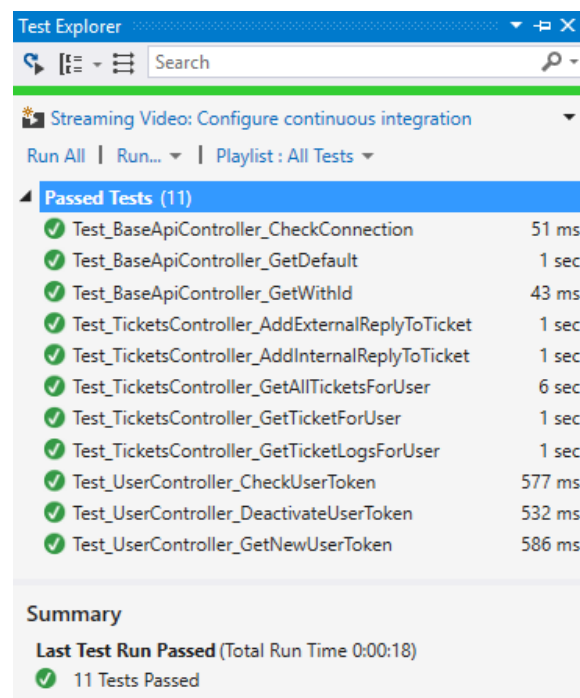
The documentation was run through by a small group of testers using provided data/accounts to check the functionality against the requirements and then recorded if the test was a pass or fail from the predetermined expected results, and if a failure, the severity of the problem. Both the Web and Mobile application were tested in this way to check the end result against the objectives. The testing document for the Web Application can be seen at *Appendix H: Testing Document for the Web Application*. The testing document for the Mobile Application can be seen at *Appendix I: Testing Document for the Mobile Application*.

This is not an efficient way of testing large applications but does ensure that the application under test is of a set standard that has been set before completion, and a way of deciding if the project incorporates the original objectives.

6.14.2 Automated Testing

As the Web API methods only return string/JSON, it is harder to see if they are returning what is expected, as incorrectly formatted JSON would often result in failure with the receiving entity. To ensure these were returning the correct data, a style of Unit Testing was decided to cover these. This also meant that during the development of the Mobile Application it could be assumed that the Web API was functioning and any problems were from the Mobile Application itself.

The way automated testing was implemented was by using Visual Studio Unit Testing Framework (Microsoft 2015). This enables a separate testing project to be created that can be used to test each of the Web API methods. Due to the extensive use of the Microsoft component LINQ (Language Integrated Query) (Calvert & Kulkarni 2009) to query the database, when testing against static data such as Lists/Arrays that is typically done when Unit Testing, LINQ can change the way it retrieves data as it doesn't have to generate SQL. This can provide different results and missed problems as the Unit Tests are not replicating what exactly would happen. There is also the problem of database specific attributes that are not present with Lists/Arrays such as 'not null' and value length restrictions that would not be caught with some data structures. Because of these reasons the decision was made to simulate the environment as much as possible and for each test that interacts with a database, an imitation database is created (as a file), populated with data and then the test is run against this, ensuring that the same conditions are always present. This would arguably cross the realms of Unit Testing into Integration Testing as there is now external interaction outside of the method, but the tests are executed and managed as Unit Tests with the testing framework. The only disadvantage is that these tests usually take a series of seconds to execute compared to milliseconds, but as the Web API methods are relatively small, this compromise is acceptable to benefit from automated Integration tests.



Test Name	Duration
Test_BaseApiController_CheckConnection	51 ms
Test_BaseApiController_GetDefault	1 sec
Test_BaseApiController_GetWithId	43 ms
Test_TicketsController_AddExternalReplyToTicket	1 sec
Test_TicketsController_AddInternalReplyToTicket	1 sec
Test_TicketsController_GetAllTicketsForUser	6 sec
Test_TicketsController_GetTicketForUser	1 sec
Test_TicketsController_GetTicketLogsForUser	1 sec
Test_UserController_CheckUserToken	577 ms
Test_UserController_DeactivateUserToken	532 ms
Test_UserController_GetNewUserToken	586 ms

Summary
 Last Test Run Passed (Total Run Time 0:00:18)
 11 Tests Passed

Figure 52: Unit/Integration Test Results

7 Conclusion

7.1 Project Achievements

Regarding the first objective, a Web Application, the end result is more advanced than was originally planned. It has functionality rivalling existing products and has been reliable and consistent throughout testing. The database design has proven easy to interact with without requiring refactoring to support an increased amount of functionality. Microsoft SQL Server is mature enough to not provide any problems for such a small application and Entity Framework has been an intuitive form of interaction. The structure has covered the subtask of 'Assignment and Organisation' with Users, Teams, Projects and Organisations being created and assigned assets to, all being implemented.

The second objective, Communication, has progressed further than discussed in previous reports, the Text-Based Chat is feature rich with internal and external messages, limited use for less privileged Users, reliable file attachment, ability to close Tickets on reply and to send the messages as an SMS message to external Users. The SMS functionality can be used to send messages manually as planned, and expanded to a simple SMS protocol that, effectively provides Two-Factor Authentication (*Goode-Intelligence 2012*) to securely authenticate the Mobile Application, and request simple data from the system.

The third objective, Project Specific Suggestions, has also exceeded the planned work from the two previous reports. A Mobile Application was considered an extra that should be implemented if the project was a commercial project and the time was found to implement this to a usable standard, although with limited functionality compared to the Web Application. An API was never originally intended but was ideally required for the Mobile Application, this has proved to add useful functionality that again could be expanded further, but is substantial enough to be beneficial to the project overall. The Social Media has been implemented to the extent that it was planned/hoped but is also implemented to a fraction of the potential that it could be with further time.

7.2 Further Work with More Time

With extended time there are a few areas that it would be preferable to improve;

- Increased functionality on the Mobile Application, as at present, it only can read and reply to Tickets and as a consequence the API functionality.
- The SMS protocol could be expanded more, with notifications being automatically sent out via SMS and Ticket response functionality.
- The Social Media aspects have potentially unlimited potential, up to having a full client that can manage settings, adding new Admin users to pages etc.
- As the Web Application is using MVC, it has the potential for every Controller method to be Unit Tested, with more time this would reduce the manual testing to UI components only, as the logic could be automatically covered.

7.3 Evaluate Specific Tools and Defects

The use of Clockwork SMS, although would not have been changed if the project was to be restarted, if it was to be a commercial product would most likely have to be re-thought. The cost to the individual User to send SMS messages to the system, although low, are not included with standard network plans. This could result in some Users not using this form of communication as the alternatives are effectively free. A possible option is to invest in specialised hardware that can be used with a standard SIM card, making it cheaper to all Users in the long term, or another alternative is to set up a number with Clockwork SMS that

includes additional cost as well as a monthly surcharge, but would alleviate the costs to the external Users.

The only real decision that caused complications and a significant amount of wasted time was the attempts at various hosting options, as covered in detail above. Even in hindsight, some of the problems could not have been avoided, but some (version discrepancies) could have been investigated earlier on, and handled more efficiently.

7.4 What Has Been Learnt

Below are some of the noticeable improvements and skills that have been drastically improved;

- Knowledge of ASP.NET MVC 5.
- Restful APIs, and how they differ from standard web development.
- Database design implementation and using Entity Framework.
- Using Scrum for a long term, self-managed project.
- Use of JavaScript and jQuery for client side scripting.
- HTTP Get and Post use cases.
- AJAX calls for asynchronous web development.

7.5 What Would Be Done the Same

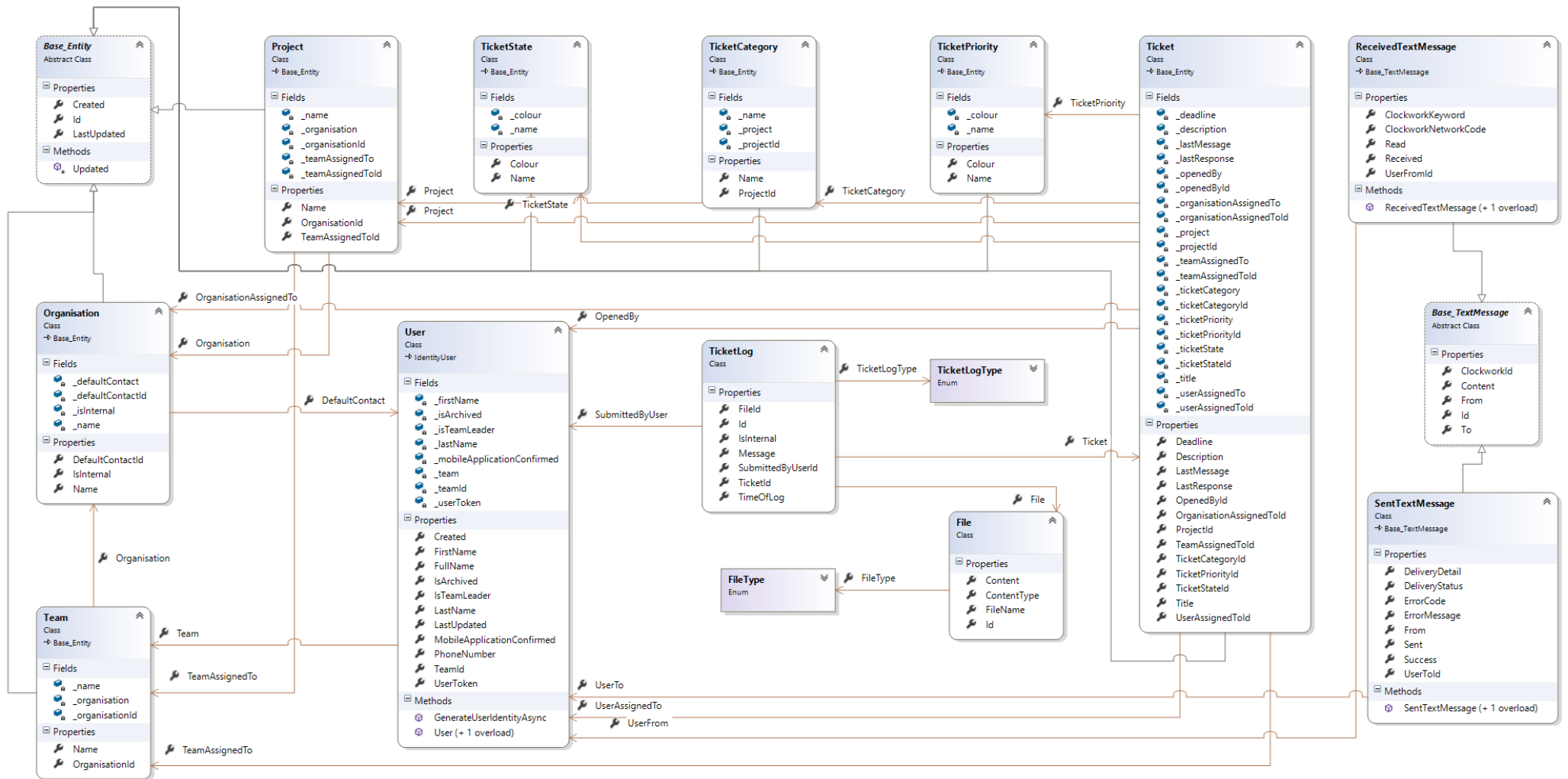
If this was to be completed again, the approach of putting in as much time as possible from very early on would be highly recommended. As a large proportion of the core functionality was completed at around the halfway point, this meant that the project from then on could be handled in a more relaxed manner. Additional features could be considered and implemented building on the core components. As with all projects, even more time could be invested to take the project further, but the magnitude of time spent has been worthwhile.

7.6 What Should have been Done Differently

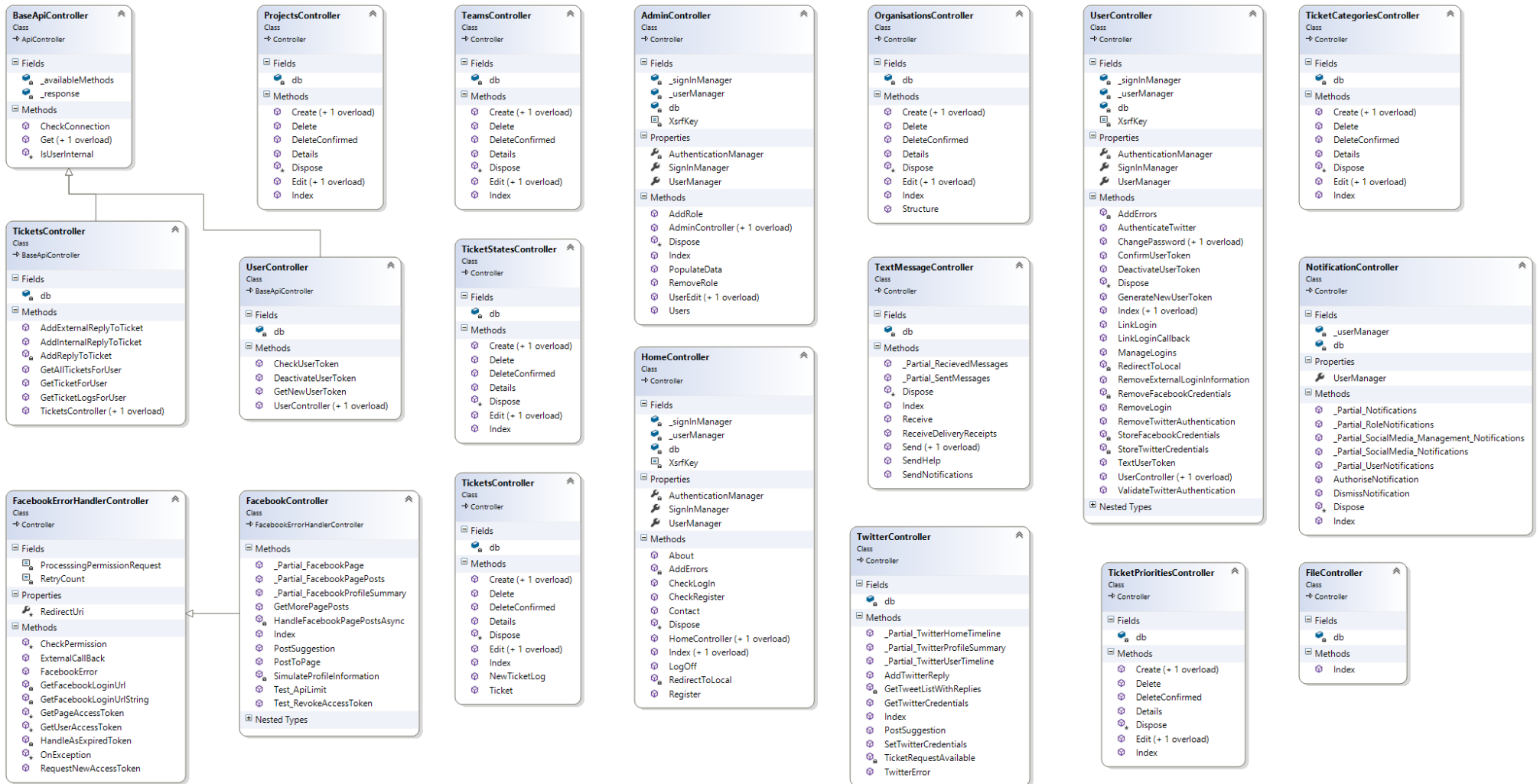
There are a few instances where the project is suitable for a relatively small amount of Tickets, but if used commercially, a slowdown in responsiveness would likely be noticeable. It was only mid-way through the project that measures were taken to combat this, with partial views being implemented more frequently (smaller sections of HTML loaded inside other pages) and asynchronous AJAX calls, that can be sent to the server and the returned HTML is inserted in rather than reloading the page. Some time should be invested in making as much as possible asynchronous to have a template load, and then the separate components load in when appropriate. This is implemented for some parts like the Social Media views, as the response time from these requests is in the control of external providers, but tasks like Ticket sorting should also implement this to accommodate larger Ticket numbers.

Paging is also not implemented for the Tickets (they are all loaded on one page) this is not a problem short term, but for example, 100+ Tickets would not be efficient to load, instead this should be split up into smaller manageable chunks that can be browsed through more easily.

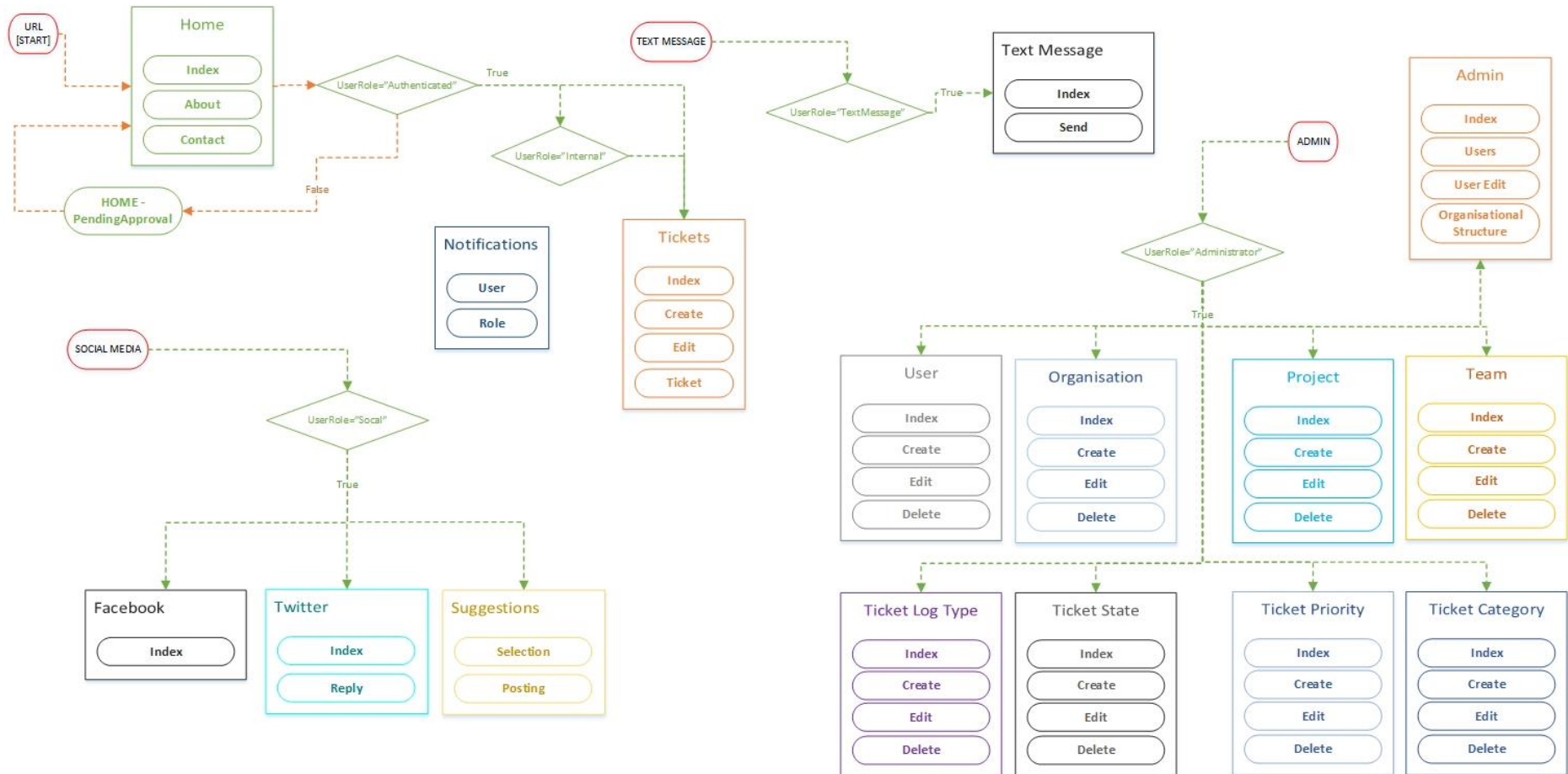
Appendix A: Model (Entities) Class Diagram



Appendix B: Controllers Class Diagram



Appendix C: Role Groups and Access to Controllers



Appendix D: Administration Panel

The Administration Panel, with links to the areas to edit the Entities with the current number of that type of entity present in the system. Functionality to add demo data (can only be ran once). Information about some of the configuration loaded from the configuration file.

Admin Panel

Edit Groups

Organisations	4
Users	10
Projects	6
Teams	6
Tickets	7
Ticket Categories	3
Ticket Logs	30
Ticket Priorities	5
Ticket States	4

Add Demo Data

View Loaded Configuration

Configuration	Value
Green Time Span	24 Hours
Amber Time Span	48 Hours
Red Time Span	730 Hours

Appendix E: Generated Organisational Structure

The Organisational Structure will work its way through the entities stored in the database, and map them out into trees to easily see what is happening within the system.

Below is a screen shot of the body of the 'Organisations/Structure' page.

Organisations

Structure



Appendix F: Full Single Ticket View

An example of a full Ticket view. Detailed Ticket information at the top, followed by a series of text based conversation, and then an option to add another reply to the Ticket log.

Tickets

Title: Chasing up Estimates Description: Looking for an update on the estimates we discussed Project: John's Complete Solutions - Estimates for Work Organisation: John's Complete Solutions Last Response:	State: Awaiting Response Priority: On Hold Assigned to: Clair Angon Team: Support Last Message:	Created: 19/04/2016 Opened By: Barry Tompson Category: Question Deadline: 09/02/2016
--	--	---

Message

John Smith @ 2/4/2016 8:31:32 AM

Hello, sorry to bug you, but we could do with an update on this matter. Thanks

Message

Guy Roberts @ 2/4/2016 9:31:32 AM

No problem, we have emailed the new estimate, we will now close this ticket. Thanks

Message

(You) Admin Admin @ 4/18/2016 9:12:13 PM

Trying to update the Last Response Time

Message

Barry Tompson @ 4/18/2016 8:14:24 PM

This should update the Last Message Time for you chaps!

New Internal Response

Select File: (Valid types are .txt, .pdf or an image)

Choose File

 No file chosen

Close on Reply:

Leave

Internal

Send

Appendix G: Web Application Configuration File Sample

```
<!--USER CONFIGURATION-->

<!--DATABASE CONFIGURATION-->
<add key="DatabaseConnectionString" value="TicketManagementAzure" />

<!--TICKET DURATION-->
<add key="TicketTimeSpanGreen" value="1" /> <!--Hours-->
<add key="TicketTimeSpanAmber" value="24" /> <!--Hours-->
<add key="TicketTimeSpanRed" value="48" /> <!--Hours-->

<!--TEXT MESSAGES-->
<add key="Clockwork_APIKey" value="bb3b7849d789b905c1sh7dkt9366d8d1ff18d832" /> <!-- The ClockworkSMS API Key to associate the application with the user account. -->
<add key="Clockwork_Receive_Number" value="84433" /> <!-- The shared number configured in ClockworkSMS that is used to contact the system. -->
<add key="Clockwork_Receive_Keyword" value="TICKETSYSTEM" /> <!-- The keyword configured in ClockworkSMS that is used to direct the messages within the shared number to the system. -->
<add key="TextMessage_YourName" value="Ticket System" /> <!-- Used to convert the 'to' part of a recieved text message, if it matches the 'Clockwork_Receive_Keyword'. -->
<add key="TextMessageFrom" value="TS" /> <!-- Used as the 'from' value when received on a mobile device. A maximum of 11 characters or 12 digits. -->
<add key="TextMessageMaxLength" value="160" /> <!-- Characters, specified by ClockworkSMS -->

<!--EXTERNAL LOGINS-->
<!--FACEBOOK-->
<add key="Facebook_GraphAPIVersion" value="2.5" />
<add key="Facebook_AppId" value="1685325635412903" /> <!-- From Facebook Developers Page for the Application. -->
<add key="Facebook_AppSecret" value="cdfel1fh7s95ld77db9bf9f9d2e56ea83" /> <!-- From Facebook Developers Page for the Application. -->
<add key="Facebook_Permission_Scope" value="email user_about_me user_birthday user_location user_friends manage_pages publish_pages publish_actions" /> <!-- The permissions required from Facebook. -->
<add key="Facebook_Admin_Page_Id" value="1661946230689519" /> <!-- Used to identify the Facebook page to be used (can be found with a service like http://findmyfbid.com/). -->
<add key="Facebook_Admin_Page_Posts_Batch_Size" value="4" /> <!-- When loading the new posts, how many to pull down at once (less than 50) -->
<!--TWITTER-->
<add key="Twitter_ConsumerKey" value="l9gnPI0cjCCa1STlJ8ysNP75b" /> <!-- From Twitter Developers Page for the Application. -->
<add key="Twitter_ConsumerSecret" value="fQhKIM2sVFas84d0mvAOUuZsIx84Kg9jXGwiwPPmsuWqeCGIgw" /> <!-- From Twitter Developers Page for the Application. -->
<add key="Twitter_Hashtag" value="AskTicketSystem" /> <!-- The Twitter hashtag that should be included with a Tweet for the system to mark it as a Ticket opportunity. -->

<!--END USER CONFIGURATION-->
```

Appendix H: Testing Document for the Web Application

Testing Document – Ticket System Web Application

i – Test Information

Web Application Version	
Web Browser	
Web Browser Version	

P/F = Pass/Fail

S = Severity (1-5, Critical to Minor Issue)

1 – Default Behaviour, Authorisation and Registering

#	Description	Prerequisites	Anticipated Result	P/F	S
1.1.1	Check the default home page loads.	Ensure the application URL is known and the site is online.	The default web page loads, with a background image, welcome information, options to log in, register button, links to other pages.		
1.1.2	Check the Register Modal loads.	On the home page, click 'Register'.	A modal (pop-up) appears on top of the original website. The modal can only be closed via the 'X' or the 'Close' button.		
1.1.3	Check Register Form Validation.	Fill in various invalid/missing fields.	When the form is submitted, it reports back the problem(s) with the provided information.		
1.1.4	A User can register with the System.	Valid information is provided.	The modal closes and a dismissible alert appears informing the user that their account has been created, but they cannot log in until authorised.		
1.2	Trying to log in when not permitted.	A user account has to be created that does not have rights to log in. (Internal Role)	A dismissible alert appears informing the user that they have an account, but they cannot log in until authorised.		
1.3	Testing log in validation.	On the home page. A valid, authorised user.	Enter invalid information		
1.4	Logging in.	On the home page. A valid, authorised user.	After entering the valid information, they user is directed to the Ticket Index, that shows a list of relevant Tickets.		
1.5	Checking the Links.	On the home page, click 'Ticket Management'.	It should return to the home page, or refresh the page if already on it.		
1.6	Checking the Links.	On the home page, click 'About'.	The About page loads, with extra information.		

1.7	Checking the Links.	On the home page, click 'Contact'.	The Contact page loads, with contact information.		
1.8	Checking the Links.	On the home page, click 'Register/Login'.	The Homepage loads, or refresh the page if already on it.		
1.9	Logging out.	A valid, authorised user, that is logged in.	Click the 'exit/logout' icon, this will redirect the user to the home page, with an alert informing the user that they have been logged out.		
1.10	Addition Icons, information.	A valid, authorised user, that is logged in. Click the 'information' icon.	The user will be redirected to the Information page, with the user logged in, clicking the 'Ticket Management' link will redirect the user to the Tickets index, not the Welcome page.		
1.11	Addition Icons, contact.	A valid, authorised user, that is logged in. Click the 'telephone/contact' icon.	The user will be redirected to the Contact page, with the user logged in, clicking the 'Ticket Management' link will redirect the user to the Tickets index, not the Welcome page.		

2 – Ticket Behaviour

#	Description	Prerequisites	Anticipated Result	P/F	S
2.1	The Tickets load correctly.	A valid user is logged in, and on the Ticket Index page.	A list of Tickets displays for the User, depending on what they created or if internal all of the Tickets display.		
2.2	Ticket 'quick access information' loads correctly.	A valid user is logged in, and on the Ticket Index page. Hover over the 'Person' icon, the 'information' icon, and the 'calendar' icon.	When hovering over each one respectively, additional information expands that is relevant to the topic. Opened by and assigned to information, the associated project and description and important Ticket dates, respectively.		
2.3	Sorting re-orders results.	A valid user is logged in, and on the Ticket Index page. Click on each of the Ticket sorting tabs.	Depending on the tab clicked, the Tickets will re-load with the correct Ticket Type if there are Tickets of that type available.		
2.4	Ticket sorting variation, depending on User Roles.	A valid user is logged in, one with the Internal Role applied and one without, and on the Ticket Index page.	With the Internal User an extra sort category should be present, 'Mine', that only shows Tickets assigned to that User. External Users should not see this option.		
2.5	Ticket Editing as an Admin User.	A valid user is logged in, and on the Ticket Index page. Click on the 'pencil/edit' icon.	As an Admin User, the User should be able to edit all of the Ticket attributes, except who Opened the Ticket, and the times of the last messages from each User.		
2.6	Ticket Editing as a non-Admin User.	A valid user is logged in, and on the Ticket Index page. Click on the 'pencil/edit' icon.	As a non-Admin User, the User should only be able to edit the Ticket title, and only see who opened the Ticket and the times of the last messages, but not change these.		
2.7.1	Ticket Creation.	A valid user is logged in, and on the Ticket Index page. Click on the 'Create' button.	The User is redirected to a Ticket Create page that displays the available Ticket attributes.		

2.7.2	Ticket Creation, validation.	A valid user is logged in, and on the Ticket Create page. Attempt to create a new Ticket with invalid data.	Warnings should appear near the area that is not valid, and will not create a new Ticket.		
2.7.3	Ticket Creation, date validation.	A valid user is logged in, and on the Ticket Create page. Click on the Deadline input box.	A Date Picker appears, that prevents the User from selecting a weekend, the current date is highlighted, the current date hovered over is highlighted appropriately.		
2.7.4	Ticket Creation.	A valid user is logged in, and on the Ticket Create page, with valid Ticket information present, click 'create'.	The User is redirected to the Ticket Index, the new Ticket is created and has the information present that was previously entered.		
2.8.1	Individual Ticket View.	A valid user is logged in and has selected an individual Ticket. Ideally, the Ticket will have numerous Ticket Logs (Chat messages).	A Ticket summary is present at the top of the page, with information relevant to the Ticket. Chat based messages will be present, with the oldest at the top and newest towards the bottom. Each message has information on the type, who created it, and the date/time it was sent.		
2.8.2	Individual Ticket View, log specific properties.	A valid user is logged in, and has selected an individual Ticket, and the Ticket has at least one internal message, an external message from an external User and external message from an internal User.	The internal messages are greyed out; these should not be visible to external Users. Messages that are external should vary in colour depending on the type of User who sent them, from an internal User, they should be green, from an external User, they should be blue.		
2.9.1	New Ticket response. External User.	A valid external user is logged in and has selected an individual Ticket.	At the bottom of the Ticket view the 'New Response' panel should be blue (to match the message colour). It should have the option to attach a File as part of the message. The send button should not be enabled when no text is present in the box.		
2.9.2	New Ticket response. External User.	A valid external user is logged in and has selected an individual Ticket. Enter Text into the 'New Response' box.	When text is present, the 'Send' button should enable. When text is removed, the 'Send' button should not be enabled.		
2.10.1	New Ticket response. Internal User.	A valid external user is logged in and has selected an individual Ticket.	At the bottom of the Ticket view the 'New Response' panel should be grey (to match the message colour). It should have the option to attach a File as part of the message. The send button should not be enabled when no text is present in the box. A toggle should be present to change the Message type, toggling this will change the reply box colour and relevant text. On the external response, an option to send an SMS message should also be present, as well as the option to close the Ticket on reply.		

2.10.2	New Ticket response. Internal User.	A valid external user is logged in and has selected an individual Ticket. Enter Text into the 'New Response' box.	When text is present, the 'Send' button should enable. When text is removed, the 'Send' button should not be enabled.		
2.11.1	New Ticket response.	A valid external user is logged in and has selected an individual Ticket. Enter Text into the 'New Response' box. Press 'Send'.	The Message should be added to the Ticket; the Ticket Log Type should be as expected.		
2.11.*	New Ticket response. Repeat for all of the Ticket Log combinations, ensuring the colouring is correct. Internal messages are only visible to internal Users. Responses from External Users should update the 'Last Message' time, and change the Ticket State to 'Awaiting Response'. External responses from Internal Users should update the 'Last Response' time, and change the Ticket State to 'Open'.				
2.12	New Ticket response. Close on Reply.	A valid external user is logged in and has selected an individual Ticket. Enter Text into the 'New Response' box. Enable 'Close on Reply'. Press 'Send'.	The Message should be added to the Ticket, the Ticket state should change to Closed, and the Ticket Log Type should be as indicated by the toggle (internal or external).		
2.13.1	New Ticket response, the addition of Files.	A valid external user is logged in and has selected an individual Ticket. Enter Text into the 'New Response' box. Click 'Choose File'.	An Operating System window should appear so that the user can select a file.		
2.13.2	New Ticket response, the addition of Files.	A valid external user is logged in and has selected an individual Ticket. A file has been selected to upload.	When the file has been selected, the file name is added to the Ticket Response box.		
2.13.3	New Ticket response, the addition of Files.	A valid external user is logged in and has selected an individual Ticket. A file has been selected to upload. Click 'Send'.	The Ticket Log should be added to the Ticket. The log should indicate that a File is attached and the type of file if recognised. The Filename should be a hyperlink, that upon clicking is opened up in a new browser tab/window.		

3 – Administrator Role

#	Description	Prerequisites	Anticipated Result	P/F	S
3.0	To check the accessibility of the Admin Users.	A valid user with the Administrator Role. A valid user without the Administrator Role.	The user with the role should see additional icons along the navigation bar. These include a 'cogs' icon that drops down to see multiple options and a 'tree' icon. If the non-admin tries to access the Admin controller (/Admin/), they will be informed an error has occurred and it is because of the lack of permissions.		
3.1.1	The admin panel.	A valid user with the Administrator Role. Click on the 'cogs' icon.	A dropdown should appear with links to an Admin Panel, and the controllers of each of the Entities.		

3.1.2	The admin panel.	A valid user with the Administrator Role. Click on the 'cogs' icon, then 'Admin Panel'.	The Admin Panel page should display, with a list of the Entities, if they can be modified they should appear as buttoned links and a count of the total number of each entity. Beneath that is functionality to add demo data. And beneath that is some of the configuration loaded in from the configuration file.		
3.1.3	The admin panel.	A valid user with the Administrator Role. Click on the 'cogs' icon, then 'Admin Panel'. Click on an Entity.	The user should be redirected to the Index for the selected entity, that displays a list of the current entities. For each one, the user can be directed to a page to edit the information, to view the full information about it, or delete it completely.		
3.1.*	This functionality should be repeated for each of the entities. This includes Organisations, User, Projects, Teams, Ticket Categories, Ticket Priorities, and Ticket States. In each case, where applicable, an instance should be checked, edited created and deleted.				
3.2	Organisational Structure.	A valid user with the Administrator Role. Click on the 'tree' icon.	This will redirect the user to a generated Organisational Structure that shows how the system entities are associated with each other. Arrows are present to direct the administrator to edit each of the entities.		
3.3.1	Addition of Roles.	A valid user with the Administrator Role, on the User Admin page, select a user to edit. Add a role to a user who is not internal.	An error alert appears informing the user that a functionality role cannot be applied to an external user.		
3.3.2	Addition of Roles.	A valid user with the Administrator Role, on the User Admin page, select a user to edit. Add a role to a user who is internal.	The role is applied to the User, log in as them and check this has been applied.		
3.3.3	Removal of Roles.	A valid user with the Administrator Role, on the User Admin page, select a user to edit. Remove a role from the user.	The role is removed from the User, log in as them and check this has been applied.		

4 – SMS Role and Functionality

#	Description	Prerequisites	Anticipated Result	P/F	S
4.1	Access to SMS functionality.	A valid user with the Message Role. A valid user without the Message Role.	The user with the Message Role should see an additional 'phone' icon. This will take the user to the message index. Any user attempting to get to this functionality via the address bar will be redirected to an error page, informing them that they do not have the required permissions.		

4.2.1	SMS functionality.	A valid user with the Message Role. On the Text Message index page.	The page should contain a list of sent SMS messages, the contents of the messages, the time they were sent, user recipient information and delivery information. It should also contain a list of received messages, who sent the message, the message body, and the time it was received.		
4.2.2	SMS functionality, balance.	A valid user with the Message Role. On the Text Message index page.	The current balance of the Clockwork account should be displayed; this should be updated when the page is refreshed.		
4.3	New Message Functionality.	A valid user with the Message Role. On the Text Message index page. Click the 'New Message' button.	The Send Text Message page should have the option to send an SMS message to any of the registered users, via a dropdown. There is also the option to send a 'Notifications' message (any of the notifications the user has) and/or a 'Help' message, that sends them a list of the Text Message protocol options.		
4.4	Send an SMS Message.	A valid user with the Message Role. On the Text Message index page. Sufficient credit to send a message. Send one of the available messages.	The user should receive an SMS message to the number associated with the user account. The message should appear on the Text Message index page.		

5 – Social Media Role and Functionality

#	Description	Prerequisites	Anticipated Result	P/F	S
4.1	Access to Social Media functionality.	A valid user with the Social Media Role. A valid user without the Social Media Role.	The user with the Role should see an additional 'Twitter' icon. This will take the user to the Twitter index. The user with the Role should see an additional 'Facebook' icon. This will take the user to the Facebook index. Any user attempting to get to this functionality via the address bar without the Social Media Role will be redirected to an error page, informing them that they do not have the required permissions.		
4.2.1	Access to Social Media functionality. Account association.	A valid user with the Social Media Role. Go to the Twitter index.	The user should be informed that their account has not been associated with Twitter, a link to the Account Association page should be present.		
4.2.2	Access to Social Media functionality. Account association.	A valid user with the Social Media Role. Go to the Facebook index.	The user should be informed that their account has not been associated with Facebook, a link to the Account Association page should be present.		

4.2.3	Access to Social Media functionality. Account association.	A valid user with the Social Media Role. Go to the Account Association page.	The options of the available Social Media platforms should be available to link up with the system account.		
4.2.4	Access to Social Media functionality. Account association.	A valid user with the Social Media Role. Go to the Account Association page, click on an Account to associate.	This should redirect the user to the external site (Facebook or Twitter), asking the user to log in if they haven't already, and then allowing the system to access the information it requires. Upon confirmation it will return the user to the application and the account should be associated.		
4.2.*	This will need to be done for both Facebook and Twitter, both processes are the same, just with different providers.				
4.3	Access to Social Media functionality. Twitter.	A valid user with the Social Media Role. The account should be associated with a valid Twitter account.	The Twitter page should load, pulling down profile information, a list of the User Timeline, a section to create a new post, and an area for the Home Timeline that is toggle-able, to show/hide the information.		
4.4	Access to Social Media functionality. Twitter, posting	A valid user with the Social Media Role. The account should be associated with a valid Twitter account. Enter a valid message to Tweet.	The message should be sent; a notification should appear the user alerting them the Tweet has been sent. If the page is refreshed, it will appear in the Timeline.		
4.5	Access to Social Media functionality. Twitter, timelines.	A valid user with the Social Media Role. The account should be associated with a valid Twitter account.	The Timelines should default to being compacted, the nested conversations should be expanded/collapsed by clicking on the 'chat' icon. The Tweets should also have a 'reply' icon.		
4.6	Access to Social Media functionality. Twitter, replies.	A valid user with the Social Media Role. The account should be associated with a valid Twitter account. Click on a 'reply' icon.	A popup box should appear that the user can enter text into, and a send button. Upon sending the Tweet, the dialogue should disappear, the new reply should then be seen in the Timelines.		
4.7.1	Access to Social Media functionality. Twitter, Ticket Detection.	A valid user with the Social Media Role. The account should be associated with a valid Twitter account. A Tweet with the predetermined Hashtag.	If a Tweet has the hashtag, it will be detected as such and a 'plus' icon will be present on the Tweet. Pressing this will direct the user to the Ticket creation page, and prepopulate the information.		
4.7.2	Access to Social Media functionality. Twitter, Ticket Detection.	A valid user with the Social Media Role. The account should be associated with a valid Twitter account. A Tweet with the predetermined Hashtag, that has been through test 4.7.1.	If a Tweet has the hashtag and has been converted to a Ticket, it will be detected as such and a 'ticket icon' will be present on the Tweet. Pressing this will direct the user to the Ticket instance that the Ticket is related to.		
4.8	Access to Social Media functionality. Facebook.	A valid user with the Social Media Role. The account should be associated with a valid Facebook account, that has administrative rights on the preconfigured Page.	The Facebook page should load, pulling down profile information, the page Timeline, a section to create a new post.		

4.9	Access to Social Media functionality. Facebook, posting	A valid user with the Social Media Role. The account should be associated with a valid Facebook account, that has administrative rights on the preconfigured Page.	The message should be sent; a notification should appear the user alerting them the Post has been sent. If the page is refreshed, it will appear in the Timeline.		
-----	---	--	---	--	--

5 – Notifications

#	Description	Prerequisites	Anticipated Result	P/F	S
5.1.1	Notification Icon changes.	A logged in user, without a notification.	The 'bell/notification' icon is visible but grey.		
5.1.2	Notification Icon changes.	A logged in user, without a notification. On the Notification index page.	The page informs the user that they don't have any notifications, User or Role based.		
5.2.1	Notification Icon changes.	A logged in user, with at least one notification.	The 'bell/notification' icon is visible and red.		
5.2.2	Notification Icon changes.	A logged in user, with at least one notification. On the Notification index page.	The page informs the user of the notifications they have, either User and/or Role based. It has the type of notification, a description and the actions that can be performed.		
5.3.1	Notification generation.	A logged in, internal user, without a notification, who is an assigned to a Ticket.	Update the ticket, and the user assigned should get a new notification informing them that an update to the Ticket has been made.		
5.3.2	Notification generation.	A logged in, internal user, without a notification, who is part of the Team assigned to the Ticket.	Update the ticket, and the user (and the rest of their team) should get a new notification informing them that an update to the Ticket has been made.		

6 – Generated Suggestions

#	Description	Prerequisites	Anticipated Result	P/F	S
6.1	Suggestions basic functionality.	A logged in user that is assigned the Social Media Role, on the Notifications index page.	There is a list of options beneath the standard notifications that can be toggled on or off, and a search button. There is also a 'Toggle All' button that will change them all in either direction.		
6.2.1	Suggestion generation.	A logged in user that is assigned the Social Media Role, on the Notifications index page. Some criteria are selected, and the 'Search' button is clicked.	A modal/popup appears that has a selection of panels with suggested posts to make. Each post can be posted to Facebook, Twitter or dismissed.		

6.2.2	Suggestion generation functionality.	A logged in user that is assigned the Social Media Role, on the Notifications index page. Some criteria are selected, and the 'Search' button is clicked. The Facebook button is clicked for a suggestion.	The message is posted to the Facebook page, as it is displayed, and the option to post that message to Facebook is removed.		
6.2.3	Suggestion generation functionality.	A logged in user that is assigned the Social Media Role, on the Notifications index page. Some criteria are selected, and the 'Search' button is clicked. The Twitter button is clicked for a suggestion.	The message is Tweeted to the Twitter page, as it is displayed, and the option to post that message to Twitter is removed.		
6.2.4	Suggestion generation functionality.	A logged in user that is assigned the Social Media Role, on the Notifications index page. Some criteria are selected, and the 'Search' button is clicked. The 'close/dismiss' button is clicked for a suggestion.	The suggestion is removed from the list.		
6.3	Suggestion generation functionality.	A logged in user that is assigned the Social Media Role, on the Notifications index page. Some criteria are selected, and the 'Search' button is clicked. The 'close/dismiss' button is clicked on the model.	The model/popup is removed.		

Appendix I: Testing Document for the Mobile Application

Testing Document – Ticket System Mobile Application

i – Test Information

Mobile Application Version	
Mobile Phone Model	
Android OS Version	

P/F = Pass/Fail

S = Severity (1-5, Critical to Minor Issue)

1 – Associating the Device with a User Account

#	Description	Prerequisites	Anticipated Result	P/F	Severity
1.0	Checking the application loads.	Ensure the application is not associated with an account. Open up the application.	A welcome screen is displayed, asking for a username.		
1.1.1	Checking if the User is valid.	Leave the username input empty and click 'Authorise'.	A popup should appear asking the user to enter a value.		
1.1.2	Checking if the User is valid.	Enter an invalid username value and click 'Authorise'.	A popup should appear informing the user that an incorrect value has been entered.		
1.2.1	Authorising an Account.	Enter a valid username value and click 'Authorise'.	The user should be directed to a new page (activity) informing them that the credentials have been saved, with the option to wait for authorisation from the web application or send an SMS message.		
1.2.2	Authorising an Account.	Enter a valid username value and click 'Authorise'. Choose to authorise via SMS.	An Android notification will appear asking the user to confirm that the application can send an SMS message on their behalf. (Also that the number to send the message may charge them).		

1.2.3	Authorising an Account.	Wait for the SMS message to be sent (this can be checked on the web application) and then click authorise.	The user should be taken to the Tickets page (activity).		
The application has to be taken back to 1.2.1 for an alternative approach to be tested.					
1.2.4	Authorising an Account.	Enter a valid username value and click 'Authorise'. Choose to authorise via the web application.	The popup will be dismissed, as now the User Token has to be authorised via the web application. This can be accessed from the User Edit page.		
1.2.5	Authorising an Account.	Enter a valid username value and click 'Authorise'. Press the authorise button.	It will check the unauthorised User Token and report that it is not authorised.		
1.2.6	Authorising an Account.	Enter a valid username value and click 'Authorise'. Have the User Token authorised via the web application. press the authorise button.	It will check the User Token and then redirect the user should be taken to the Tickets page (activity).		
1.3	Authorised Account.	A user has been associated with the device and has the User Token confirmed.	When the application is loaded, it should direct the user to the main page, it will check the details with the server, then automatically redirect the user to the Tickets page (activity).		

2 – Settings, Removing User Association

#	Description	Prerequisites	Anticipated Result	P/F	Severity
2.1	The settings page (activity) availability.	Click the 'Settings' option at some point in the application.	The Setting page (activity) should load, displaying the stored user information (if applicable) and the option to remove the user from the phone and to deactivate the user from the phone.		
2.2.1	Removing the user from the device.	The phone has some user information stored on the device. Click the 'Remove User from Phone' button.	A popup informing the user that the user information has been removed and then the user will be redirected to the welcome page (activity).		

2.2.2	Removing the user from the device.	The phone has some user information stored on the device. Click the 'Deactivate User from Phone' button.	A popup informing the user that the user information has been deactivated from the server and removed from the phone, the user will be redirected to the welcome page (activity).		
-------	------------------------------------	--	---	--	--

3 – Tickets

#	Description	Prerequisites	Anticipated Result	P/F	Severity
3.1	The correct User Tickets are displayed.	A user has been associated with the device and has the User Token confirmed.	The user should be redirected to the Tickets page (activity) and the Tickets matching those populated on the web application should be displayed, the colouring should also be the same.		
3.2	User input, scrolling.	A user has been associated with the device and has the User Token confirmed.	The user should be able to scroll through the Tickets, with fading at the top and bottom, and more Tickets appearing depending on the scroll action.		
3.3	Refresh functionality.	A user has been associated with the device and has the User Token confirmed. On the Ticket page, the 'Refresh' button should be clicked.	This will reload the Tickets from the server, this can be checked by changing the Tickets via the web application and refreshing.		
3.4	Individual Ticket View.	A user has been associated with the device and has the User Token confirmed. Click on an individual Ticket.	The individual Ticket should load from the server, with the type of messages being replicated as it would display on the web application. The origin of the message should also be emphasised with right/left alignment to illustrate a conversation. A reply button should also be present.		
3.5.1	Reply functionality.	An internal user has been associated with the device and has the User Token	A popup should appear with a place for a message input. There should be the		

		confirmed. Click on an individual Ticket. Click 'Reply'.	option to post an internal or external message.		
3.5.2	Reply functionality.	An external user has been associated with the device and has the User Token confirmed. Click on an individual Ticket. Click 'Reply'.	A popup should appear with a place for a message input. There should only be the option to post the message.		
3.5.3	Reply functionality.	An internal user has been associated with the device and has the User Token confirmed. Click on an individual Ticket. Valid input entered and click 'Reply'.	The reply should be posted to the Ticket and the user redirected to the single Ticket view with the new reply present at the bottom of the Ticket.		

References

- 3 Scale, 2011. *What is an API ? Your guide to the Internet Business (R)evolution*,
- Adobe Systems Incorporated, 2010. *Exceptional customer experience : How to turn customers into advocates*,
- Albrecht, M., Lesser, E. & Ban, L., 2013. *IBM Institute for Business Value; The Software Edge*,
- Anderson, R. & Dykstra, T., 2014. *Getting Started with Entity Framework 6 Code First using MVC 5*, Microsoft Corporation.
- Apple Inc., 2016. About the iOS Technologies. *About the iOS Technologies*. Available at: https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898-CH1-SW1 [Accessed April 14, 2016].
- Apple Inc., 2013a. iOS Human Interface Guidelines: Human Interface Principles.
- Apple Inc., 2015. iPhone User Guide. In pp. 1–152.
- Apple Inc., 2013b. *OS X Mavericks - Core Technologies Overview*,
- Apple Inc., 2008. Xcode Unit Testing Guide.
- Best Practical, 2015. Support Ticket System. Available at: <https://www.bestpractical.com/rt/features.html> [Accessed December 21, 2015].
- Bootstrap, 2016. Bootstrap. Available at: <http://getbootstrap.com/> [Accessed January 16, 2016].
- Böttcher, S. & Türling, A., 2002. Access Control and Synchronization in XML Documents. In *XML Technologien für das Semantic Web*. p. 14.
- Boyd, D. m. & Ellison, N.B., 2007. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, 13(1), pp.210–230.
- Britannica Encyclopedia, 2015. Protocol - Computer Science. Available at: <http://www.britannica.com/technology/protocol-computer-science> [Accessed April 8, 2016].
- BSI Group, 2014. British Standard for Customer Service (BS 8477:2014).
- Buffalo Technology, 2010. Technical Brief; Central Vs Local Storage, What is Right for Your Business? , pp.1–3.
- Burbeck, S., 1992. Applications Programming in Smalltalk-80 (TM): How to use Model-View-Controller (MVC). *Smalltalk-80 v2, 80(Mvc)*, pp.1–11.
- Cake Software Foundation, 2016. CakePHP. Available at: <http://cakephp.org/> [Accessed May 3, 2016].
- Calvert, C. & Kulkarni, D., 2009. The Essence of LINQ. In *Essential LINQ*. Pearson Education Limited, pp. 39–64.
- Chauhan, V.K., 2014. Smoke Testing. *International Journal of Scientific and Research Publications*, 4(1), pp.2250–3153.
- Cognizant, 2015. *Power to the People : Customer Care and Social Media*,
- Cognizant, Nashikkar, N. & Garg, R., 2015. User Acceptance Testing.
- Connolly, T. & Begg, C., 2005. SQL: Data Manipulation. In *Database Systems: A Practical Approach to Design, Implementation, and Management*. Pearson Education Limited.

Dix, A. et al., 2004. *Human-Computer Interaction* Third Edit., Pearson Education Limited.

Django Software Foundation, 2016. django. Available at: <https://www.djangoproject.com/> [Accessed May 3, 2016].

Dun & Bradstreet Corporation, 2014. *What is an API?*,

EMAC International, 2013. The JSON Data Interchange Format. *EMAC International*, 1st Editio(October).

Federal Trade Commission, 2012. *Understanding Mobile Apps*,

FreshDesk, 2015. Support Ticket System. Available at: <https://freshdesk.com/ticketing> [Accessed December 21, 2015].

Gandy, D., 2016. Font Awesome. Available at: <https://fontawesome.github.io/Font-Awesome/> [Accessed April 27, 2016].

Git, 2016. Git. Available at: <https://git-scm.com/> [Accessed January 3, 2016].

GlobalSign, 2016. *Client Authentication - Prevent Unauthorized Access & Protect Business Assets*,

Goode-Intelligence, 2012. *Two-factor Authentication Goes Mobile*, London.

Grosskurth, A. & Godfrey, M., 2006. Architecture and Evolution of the Modern Web Browser. *Preprint submitted to Elsevier Science*, (June), pp.1–24.

Gunther, N.J., 2011. Client/Server Analysis with PDQ. In *Analyzing Computer System Performance with Perl*. pp. 331–354.

HelpSpot, 2015. Support Ticket System. Available at: <https://www.helpspot.com/help-desk-ticketing-software> [Accessed December 21, 2015].

Hesk, 2015. Support Ticket System. Available at: <http://www.hesk.com/> [Accessed December 21, 2015].

Hewlett-Packard, 2014. Cross-Browser Functional Testing Best Practices.

IBM Corporation, 2005. Comparing XML and Relational Storage; A Best Practices Guide. , (March).

Iveta, G., 2012. Human Resources Key Performance Indicators. *Journal of Competitiveness*, 4(1), pp.117–128.

JitBit, 2015. Support Ticket System. Available at: <https://www.jitbit.com/helpdesk/> [Accessed December 21, 2015].

Kodak Alaris Inc., 2015. *Harness the Flood of Email for Exceptional Customer Service Manage the Incoming Wave of Email Messages*,

Kumar, R., 2013. Efficient Customization of Software Applications of an Organization. *International Journal of Business and Social Science*, 4(11), pp.36–42.

Magnet Forensics, 2014. *The Rise of Mobile Messaging: The Use of Mobile Messenger Apps has Exploded in the Last 18 Months*,

Matsumoto, Y., 2016. Ruby Programming Language. Available at: <http://www.ruby-lang.org/en/> [Accessed May 3, 2016].

Mediaburst, 2016. Clockwork SMS API. Available at: <https://www.clockworksms.com/> [Accessed April 20, 2016].

Michael Kifer, Bernstein, A. & Lewis, P., 2005. Object-Relational Databases. In *Database Systems*. pp. 1–27.

Microsoft, 2016a. ASP.NET 5 Documentation.

- Microsoft, 2014. ASP.NET Web Pages Using the Razor Syntax.
- Microsoft, 2012. ASP.NET Website on IIS. Available at: <https://technet.microsoft.com/en-GB/library/hh831550> [Accessed April 21, 2016].
- Microsoft, 2016b. Choosing the Right Version Control for Your Project. *Developer Network*. Available at: <https://msdn.microsoft.com/en-us/library/vs/alm/code/overview> [Accessed April 16, 2016].
- Microsoft, 2016c. Common Language Runtime (CLR). *Developer Network*. Available at: <https://msdn.microsoft.com/en-us/library/8bs2ecf4> [Accessed May 3, 2016].
- Microsoft, 2010. Desktop Development. *Developer Network*. Available at: <https://msdn.microsoft.com/en-gb/ff380143.aspx> [Accessed January 20, 2016].
- Microsoft, 2013a. Introduction to ASP.NET Membership. *Developer Network*. Available at: [https://msdn.microsoft.com/en-us/library/yh26yfyzy\(v=VS.100\).aspx](https://msdn.microsoft.com/en-us/library/yh26yfyzy(v=VS.100).aspx) [Accessed April 17, 2016].
- Microsoft, 2016d. Understanding Role Management. *Developer Network*. Available at: <https://msdn.microsoft.com/en-us/library/5k850zwb.aspx> [Accessed April 18, 2016].
- Microsoft, 2015. Unit Testing Framework.
- Microsoft, 2016e. Visual Studio Online. *Visual Studio Online*. Available at: <https://www.visualstudio.com/features/version-control-vs> [Accessed December 22, 2015].
- Microsoft, 2013b. Windows Server 2012R2 Datasheet.
- Mistry, R. & Misner, S., 2014. *Introducing Microsoft SQL Server 2014 Technical Overview*, Microsoft Corporation.
- Mobile Squared, 2010. *Conversational Advertising*,
- Mobility Practice Happiest Minds, 2012. *Dealing with the Dilemma : Mobile App Development Approach & Decisions*,
- Molnar, D. & Schechter, S., 2010. Self Hosting vs. Cloud Hosting: Accounting for the Security Impact of Hosting in the Cloud. *Self*, pp.1–18.
- Nations, D., 2014. What is a Web Application? *About Tech*. Available at: http://webtrends.about.com/od/webapplications/a/web_application.htm [Accessed January 4, 2016].
- Nicholson, D.B., Owrak, D.A. & Daly, D.L., 2011. *Cloud Computing Research - Manchester Business School - Commissioned by Rackspace*,
- Nishimura, S. & Ueda, A., 2013. *Development of Windows Phone Devices at Fujitsu*,
- NIST, 2011. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. *Nist Special Publication*, 145, p.7.
- Oracle, 2010. *Oracle Database Lite SQL Reference*,
- OSTicket, 2015. OSTicket. Available at: <http://osticket.com/> [Accessed December 21, 2015].
- Pearson, S., 2013. Privacy, Security and Trust in Cloud Computing. *Privacy and Security for Cloud Computing*.
- Petsas, T. et al., 2013. Rise of the planet of the apps. *Proceedings of the 2013 conference on Internet measurement conference - IMC '13*, pp.277–290.
- Pew Research Center, 2015. Social Media Usage. Available at: <http://www.pewinternet.org/2015/10/08/social-networking-usage-2005-2015/> [Accessed January 19, 2016].

- Pop, P., 2000. Comparing Web Applications with Desktop Applications: An Empirical Study. , pp.1–6.
- Prescott, C., 2015. *Office for National Statistics: Internet Users 2015*,
- Python Software Foundation, 2016. Python. Available at: <https://www.python.org/> [Accessed May 3, 2016].
- Rails Core Team, 2016. Ruby on Rails. Available at: <http://rubyonrails.org/> [Accessed May 3, 2016].
- Rising, L. & Janoff, N.S., 2000. The Scrum Software Development Process for Small Teams. *Software, IEEE*, 17 , Issue(August), pp.26 – 32.
- Rouse, M., 2007. Trouble Ticket (Trouble Report). *Tech Target - Software Applications*. Available at: <http://searchcrm.techtarget.com/definition/trouble-ticket> [Accessed January 19, 2016].
- Salmre, I., 2004. Characteristics of Mobile Applications. In *Writing Mobile Code: Essential Software Engineering for Building Mobile Applications*. pp. 19–36.
- Schmidt, H. et al., 2009. *Google Android - A Comprehensive Introduction*,
- SCRUM Alliance, 2012. Core Scrum. , pp.1–14.
- Simply Measured Inc, 2014. *Customer Service on Twitter*,
- Spiceworks, 2015. Support Ticket System. Available at: <http://www.spiceworks.com/free-help-desk-software/> [Accessed December 21, 2015].
- Statista, 2015. Global Spam Volume as Percentage of Total Email Traffic from January 2014 to December 2015, by Month. Available at: <http://www.statista.com/statistics/420391/spam-email-traffic-share/> [Accessed April 8, 2016].
- Szalvay, V., 2004. An Introduction to Agile Software Development. *Danube Technologies*, (November), pp.1–9.
- Takeuchi, H. & Nonaka, I., 1986. The New Product Development Game. *Harvard Business Review*.
- The PHP Group, 2016. PHP: Hypertext Preprocessor. Available at: <https://secure.php.net/> [Accessed May 3, 2016].
- Tulloch, M. & Windows Azure Team, 2013. *Introducing Windows Azure*,
- Tutorials Point, 2014. *XML Tutorial*,
- Vision, 2015. Support Ticket System. Available at: <https://www.visionhelpdesk.com/apps> [Accessed December 21, 2015].
- VisualSVN Limited, 2015. VisualSVN. Available at: <https://www.visualsvn.com/> [Accessed December 22, 2015].
- Wasserman, A., 2010. *Software Engineering Issues for Mobile Application Development*,
- Woods, V. & Meulen, R. van der, 2016. Worldwide Smartphone Sales Q4 2015. Available at: <http://www.gartner.com/newsroom/id/3215217> [Accessed April 14, 2016].
- Young, D. (Lord), 2013. *Growing Your Business: A Report on Growing Micro Businesses*, Available at: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/197726/bis-13-729-growing-your-business-a-report-on-growing-micro-businesses.pdf.
- Young, D. (Lord), 2015. *The Report on Small Firms 2010-2015*, Available at: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/402897/L

ord_Young_s_enterprise_report-web_version_final.pdf.

Zendesk, 2015. Support Ticket System. Available at: <https://www.zendesk.com/zendesk-insights/> [Accessed December 21, 2015].