COM814: Project 2015-16

# Dissertation

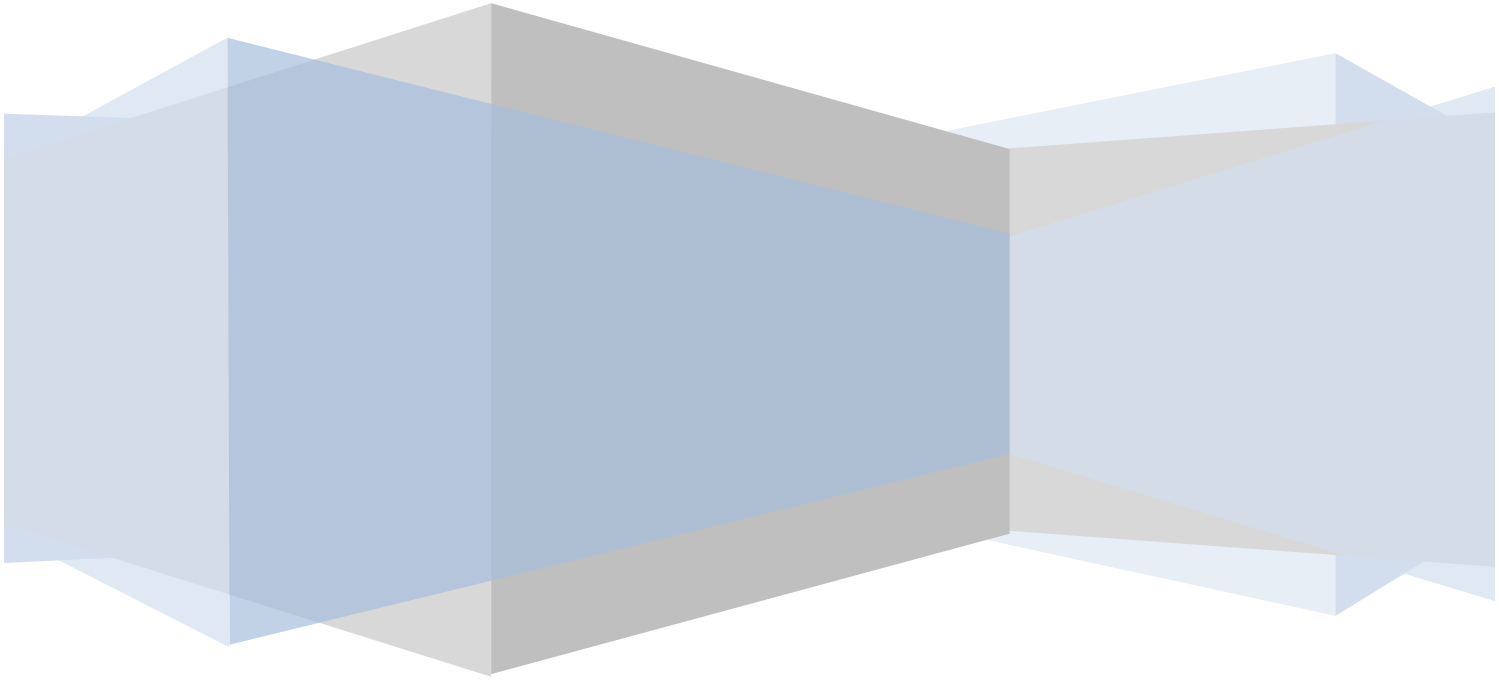
## School of Computing & Information Engineering

**Seán Óg O'Hara – B00606812**

**RESTFUL WEB DEVELOPMENT IN JAVA – ICE HOCKEY CLUB MANAGEMENT**

**Supervisor: Dr Tom Lunney**
**Second Marker: Dr Aidan McCaughey**

**1st September 2016**

## Plagiarism Statement

I declare that this is all my own work and does not contain unreferenced material copied from any other source. I have read the University's policy on plagiarism and understand the definition of plagiarism. If it is shown that material has been plagiarised, or I have otherwise attempted to obtain an unfair advantage for myself or others, I understand that I may face sanctions in accordance with the policies and procedures of the University. A mark of zero may be awarded and the reason for that mark will be recorded on my file.

SOH.


## Acknowledgements

For my grandmother, Mary Bernadette.

# Contents

# Abstract

The world of sport is ever evolving. In the past ten to fifteen years, catalysed by the introduction of the Belfast Giants, ice hockey has begun to rise up the ranks of popular sport in Northern Ireland and the United Kingdom as a whole. It only requires a glance to North America to understand the passion and loyalty which can be evoked from Ice Hockey. Considering that Northern Ireland has generally been partisan in its sports club, the Belfast Giants offered the first non-denominational sports team to the people of Belfast, and from here, interest in ice hockey spiked. However, with a rise in interest for a sport, so too comes the twenty-first century demand for sports technology. There existed no technological means of managing and keeping track of an ice hockey team.

The idea for this dissertation arose in addressing this problem. It is proposed that a solution to this problem be the development of an ice hockey club management web application. This solution will allow managers and players alike to effectively manage their team, be it a well-known league team or a newly established amateur team. The key areas of management addressed will be player management, fixture management and staff management. It will place particular emphasis on the ability to track the statistics of each player in order to make educated decisions on team selection.

These key features were the primary objective of the project. However, the fact remains that further development of the system could promote not only its effectiveness as a technological tool, but also its effectiveness as an economically viable technological product for the sports technology market. The prototype proposed at this point is the result of analysis into the requirements of the system, which has subsequently been designed and implemented in order to draw conclusions from to utilise for further development.

# CHAPTER 1: Introduction

Ice hockey has recently emerged as a fierce competitor for sports fans' time and money. In 2015 in fact, there was over 2.6 billion dollars of revenue accumulated by the National Hockey League. (Statistica, 2016.) A long established sport in North America – and often an affront to Canadian fans in not officially being their national sport as that title is held by lacrosse – it has only developed in the United Kingdom since the early 2000s. While Ice Hockey has been growing exponentially in the UK since the early 2000s, its roots can be traced back to the early 1900s when it was first played in Britain. (EIHA, 2016) The fact remains however, that is has only been in the last fifteen years that it has become a larger player in the sports scene.

It is not however the length of time it has been here, but the impact it has made in that short duration. At the time of writing there are only twenty professional teams across the two leagues in the UK, the top tier Elite Ice Hockey League, and the second tier English Premier Ice Hockey League. While this may seem insignificant in the face of the thousands of soccer teams and hundreds of rugby and cricket teams in existence, it brings an important note to the forefront. When there are less teams, fans have less choice, and thus these teams have larger followings than the miniscule amount of fans who attend Sunday league football games across the country for example. Of course the followings of the mammoths of soccer will never be rivalled in the UK, but ice hockey has surely been campaigning for attention in the last ten years.

Sports analysist and management tools have recently become an invaluable weapon in any team's arsenal. (Medeiros, 2014.) In the past we would see football managers furiously thumbing through pages of tactics and team reports on the touchline, but now we see them sending their backroom staff forth with an iPad. The reason for this is the availability of management and analysis systems on the market, which allow club staff to view analysis material and make decisions thereafter. We can also observe them showing tactics on the screen to the players on the field, allowing them the opportunity to exploit weaknesses in their opponents. This phenomenon is rampant in the United States, but has recently made its way to our shores. It only takes one viewing of Monday Night Football to see the extent to which technology has made its way into the sport, with presenters interacting with massive touchscreens without even being involved in team management. (Sky Sports, 2013)

It is unfortunate that when the magnitude of the sport is scaled back to be in line with the relative size of ice hockey, that no tool is tailored to the game. Indeed sports management apps are available on various app stores, but they are much more geared towards the management of football or rugby clubs. Ice hockey is fundamentally different in the way it is played. Being the only sport without an 'out of bounds' touchline, it is much faster paced. This can making tracking the game difficult for new viewers (ETHW, 2016), so it certainly must be difficult for team staff to maintain the events and scoring aspects of each game. Herein lies the problem.

## 1.1 The Problem

### Problem Statement:

*There exists no technological means of team management for ice hockey teams. Ice Hockey has been developing rapidly in the United Kingdom for close to twenty years, and as it progresses it is all but certain that more teams will be established. In order for teams to improve their game and stand a chance at progressing to a professional or semi-professional league it would be helpful if a team management application existed. To provide a tool to allow a team to maintain a record of their players, update their statistics as games progress, manage their upcoming fixtures and handle the details of their staff would be beneficial to the team as a whole.*

As aforementioned, there are a number of team management applications available on the market. However, none of these are tailored for ice hockey, and while some applications tailored to other sports do exist, most are primarily geared toward the management of actual sports facilities rather than team management. Capterra lists a number of these options, but the overriding aspect of many is the ability to book halls out or to keep track of a sports league rather than a team. (Capterra, 2016.)

Subsequently there exists a gap in the market for a tool to manage an ice hockey team. The overall benefit of its development would be ability for managers to improve the performance of their team and their club as a whole by staying on top of the day-to-day running of a sports club and keeping track of player statistics. If this proposed tool is created, it is fair to say that those who benefit from it have a good chance of improving their team's

game and possibly improving their standings in any league in which they compete. In order to successfully create a web application to fill this gap in the market, a solution must first be devised theoretically. Once a solution has been considered, the developer can then begin work on creating an application which will not only offer a fix to the problem, but also generate revenue as a commercially viable system.

## 1.2 The Aim

The aim of the Ice Hockey Team Management prototype will be to provide an online web application to allow a manager or a team themselves to create, update and maintain details relating to their team. This includes maintaining players, fixtures and staff. It will allow for the presentation of data to the user from which they can make informed decisions for team selection as well as financial decisions such as wage and contract extensions. Further development would include tailored reports based on number of goals scored, assists awarded and time spent in the penalty box.

## 1.3 Objectives

Objectives are a good way of setting goals for a project, and if met can improve customer satisfaction and help to remedy defects. (ITSQ, 2016.) From the outset of the project it is important for the developer to have in place a structured and time-realistic schedule on which to base the development process. In order to facilitate this, the following list of objectives has been drawn up and is intended to be used as a starting point for each step.

- Complete detailed analysis of current solutions available for this problem, whether they be loose fitting or closer to the overall direction the proposed application will take.

- Investigate the needs of the market through detailed questionnaires which will allow for an insight into the features which are desired from such an application. With this in mind, coupled with knowledge of existing web applications available, it will be possible to put in place system requirements on which to form the foundation of the application.

- To complete an aesthetic and user friendly design for the application which will promote its usability by being sleek and easy to navigate.
- To consider and identify the best means of creating the back-end of the application, including the environment in which it is created, the language with which it is programmed and the framework on which it is built.
- To implement an attractive and functional prototype which possesses the key features identified in the aim of the project.
- To test the working prototype in order to identify strong aspects and areas which are lacking in functionality or usability.
- To consider the best means of fixing any weak features identified and propose changes to them which will promote the overall usability of the application in future development.

## 1.4 Outline

The chapters following this introduction will begin to work through each of the objectives listed above. In doing this, it is hoped that the final prototype is created in a timely fashion and will meet all of the system requirements, which are detailed later. The structure of this report will be as follows:

**Chapter 2 – Analysis**

In the analysis stage, detailed analysis into solutions already in existence will be carried out. Decisions will be made at this point as to how best to go about the requirements gathering process. This analysis stage should produce a series of these in depth requirements on which to build the system.

**Chapter 3 – Design**

At the design stage, the primary objective will be to create and user friendly and aesthetically pleasing design for the web application. At this stage a graphical user interface will be created. Furthermore, particular emphasis will be placed on creating the data model for the solution,

as it is forecasted that a team management application will require the creation and maintenance of various types of data. The content in design will include mock ups and database diagrams.

**Chapter 4 – Implementation, Testing and Evaluation**

The implementation stage will follow on from design. It will be the most technical stage for the developer at which code will be written and preparation consolidated. It will flag up areas of success and areas of concern, where perhaps a certain requirement could not be met, and a solution or a compromise will have to be drawn up. Furthermore, it will consist of a testing section which will outline the process of testing the application and what conclusions can be drawn thereafter. The evaluation stage will set out the roadblocks which were faced and make suggestions for further improvements which would have to be made to the application before it could be released to the market.

**Chapter 5 – Conclusions**

This report will finish with a conclusions section which will note the successes and the failures of the project. It will tie together all the loose ends and present a retrospective of the project – what went well, what didn't, and how this could be improved.

With the aim of the project set out and the objectives in place on which to structure its development, it is now time to begin working through the stages of software development, beginning with analysis.

# CHAPTER 2: Analysis

It has been noted thus far that the problem at hand is a lack of technological assistance for ice hockey team management, and therefore a solution to this problem is the development of a team management web application.

Analysis is a key component of the software development cycle. It allows the developer an insight to what exists, what doesn't and what should. (Lea, 1995) There are the three key questions when working through the analysis stage. The first question will be the first issue addressed – what applications already exist, and how are they suited to the problem at hand? Further to this, after considering the problem in depth and setting out the objectives of the project, it has been decided that the waterfall method of software development will be utilised to see the project through. The waterfall method provides a means of 'everything flowing logically from the beginning of a project through to the end.' (Hughey, 2008.)

## 2.1 Existing Applications

There are frankly not a great deal of team management tools in existence. Of those which do exist, only a handful could be considered close enough to providing a solution to the problem in discussion. A closer view of these applications can be found in appendix A1. They include EZFacility, Sportlyzer, Manage Your League and Team Sports Admin. A brief case study of each of these systems will follow, and will be accompanied by a table to present the findings.

**EZFacility**

EZFacility is one of the highest rated sports management tools on the internet. However, its main features are more directed towards managing a sports leisure centre rather than managing a team. Its most prominent features include facility scheduling, trainer and instructor scheduling, membership management, fitness assessment, payroll and commission tracking and employee time clock. (EZFacility, 2016.) While this application is a fairly robust option and does include some features which would suit the proposed solution, it could not be considered tailored enough to be a viable solution to the problem.

**Sportlyzer**

Sportlyzer is an online community which boasts a number of tools for team management. Unlike EZFacility, Sportlyzer's main tool is a mobile application. Its main feature is one which would be key to the proposed solution, an athlete's database. However, Sportlyzer is geared towards the coaching of youth teams for which most of its features are tailored. It includes tools to keep track of parents contact information, as well as the ability to hold details regarding medical conditions of children. These features would be irrelevant to the problem at hand. Furthermore Sportlyzer is an extremely expensive option, with a month's membership costing in excess of £90. (Sportlyzer, 2016.) For these reasons, and the fact that using the system for one year would cost over £1000, it would not be a suitable solution.

**Manage Your League**

Manage Your League has a couple of features vital to the problem, however its main drawback is the fact that it is not at all tailored for team management, but instead for league management. It includes player evaluations, such as details about player ratings and statistics, as well as fixture scheduling. However, the web application is honed specifically to the management of small sports leagues, wherein it allows the user to schedule referees, register teams and look through league history. (Manage Your League, 2016.) Furthermore, Manage Your League also has an expensive price point, at $550 per year, and as yet, is not available in the UK. It would not be a suitable solution.

**Team Sports Admin**

Team Sports Admin is another similar product which has a few desirable features, but geared towards the wrong problem. It could be consider an amalgamation of Sportlyzer and Manage Your League, as it allows the user the ability to register teams and schedule fixtures. However, the main focus of the application is administration, and it places a lot of emphasis on its secure payment system which is used to accept fees from players and coaches. It includes the ability to mass communicate with players which would certainly be an advantageous feature to consider adding to the proposed solution. It is too hard to look over

the administrative spin the application has, as its price point is $1.99 per player registered. (Team Sports Admin, 2016.) Again, it would not be a suitable solution to the problem.

The analysis into these already existing applications has found that none at all are specifically tailored for ice hockey, and most are in fact geared towards administration. As such it is found that an ice hockey club management tool would be a viable solution to the problem, as at present, no suitable alternatives exist.

| Application | Player Management | Fixture Management | Staff Management | Tailored for Ice Hockey | Suitable overall? |
|---|---|---|---|---|---|
| **EZFacility** | ✘ | ✘ | ✔ | ✘ | ✘ |
| **Sportlyzer** | ✔ | ✔ | ✘ | ✘ | ✘ |
| **Manage Your League** | ✔ | ✔ | ✘ | ✘ | ✘ |
| **Team Sports Admin** | ✘ | ✘ | ✔ | ✘ | ✘ |

(Table 1. – Table of desirable features.)

## 2.2 Surveyed Analysis

Having understood from existing application analysis that the proposed solution would be a viable alternative in the face of a number of apps which are not closely enough tailored to the problem, it is imperative to collect some first-hand analysis, i.e. from the horse's mouth, so to speak. As ice hockey is a fairly niche phenomenon in Northern Ireland, an in person focus group would not be an effective means of collecting data. The main reason for this is that the data required would have to come from two primary groups: firstly, those involved first-hand with ice hockey, such as players or staff, and secondly those involved in any form of team management, regardless of the sport. The most appropriate means therefore would be the distribution of questionnaires comprised of questions carefully selected to achieve the most relevant answers, and in turn, the most suitable data. The results

of questionnaires can be ‘quickly and easily quantified by the researcher’ meaning it is a fast and reliable tool for information gathering. (University of Surrey, 2010.)

The main function of the questionnaire distribution is to collate the resulting data into a number of requirements. The questionnaire will generate the bulk of the functional requirements of the proposed solution. Moreover, due to the nature of software development, the data collected from the questionnaires will also contribute in part to a number of non-functional requirements. Questionnaires are much less intrusive than telephone or face-to-face surveys, (Walonick, 1993.) making them ideal for this scenario.

As aforementioned, ice hockey is a fairly niche sport in the UK and because of the nature of respondents required, the project will rely on only a few contactable individuals. It was decided that the most efficient means of distributing the questionnaire would be through google docs, as a single hyperlink allows access to the questionnaire for the respondent and keeps their identity hidden. The respondents were all contacted via social media, with the higher profile ice hockey players being contacted via direct message on Twitter. The seven respondents to the questionnaires were:

- 4 Belfast Giants ice hockey players.
- 2 Sports team coaches.
- 1 ice hockey fan / current professional software engineer.

The balance of respondents was luckily in the developer’s favour as the participants were mostly from the two desired groups, allowing for data essential to development to be returned. It was important that the questions distributed met a number of requirements. They could not be too technically worded. It was not intended that those with an acute knowledge of software development were to respond. Conversely in fact, the aim of this analysis was to gain real life information from people directly related to the field of ice hockey. It would have been counter-productive to ask non-technically minded people what they desired in a graphical user interface for example. This kind of information could be sought from academics and from years of evolution in design. It would however be appropriate to ask which kind of features would deter a person from using a website, and if the answers tended to relate to issues such as poor layout or unattractive design, they could then be used

as such. Secondly it was important to make the questions as simply answered as possible. No one wants to spend thirty to forty minutes answering an in depth questionnaire. Complacency would set in after a few questions, and the quality of data collected would drop thereafter. It was therefore decided that a simple 1 to 5 scale would be coupled with yes/no questions and a small number of open questions in order to maximise response quality. (CVent, 2009.)

The questions included in the questionnaire are listed in the table below, and their responses analysed in the pages following. A more in depth graphical analysis of the results can be observed in appendix A2.

| No. | Question. | Type of Question. |
|---|---|---|
| **1** | Current "paper" team management could be improved upon. | 1 to 5 scale. |
| **2** | A web application could improve team management. | 1 to 5 scale. |
| **3** | I am aware of existing applications which aid team management. | Yes/No. |
| **4** | The ability to add players and maintain their information is important. | 1 to 5 scale. |
| **5** | The ability to add fixtures and maintain their information is important. | 1 to 5 scale. |
| **6** | The ability to add staff and maintain their information is important. | 1 to 5 scale. |
| **7** | A web application would make it easier to manage a sports team. | 1 to 5 scale. |
| **8** | An ice hockey management system may encourage new teams to form. | 1 to 5 scale. |
| **9** | What in particular do you feel an ice hockey management system should include? | Open question. |

| | | |
|---|---|---|
| **10** | What should an ice hockey management system not include? | Open question. |
| **11** | Would you be willing to pay a monthly subscription fee to use an ice hockey management system? | Yes/No. |
| **12** | Would you be willing to pay a higher fee to own an ice hockey management system outright? | Yes/No. |

(Table 2. – List of questions included in the questionnaire. )

The responses to the questionnaire provided an insight into the views of players and other sporting personnel regarding the current state of team management, and their opinions on the features proposed for the solution. In gaining this information, a detailed analysis of the answers can take place in order to more fully understand the requirements of the system. While appendix A2 holds graphical representations of the answers given, this short section will describe the outcomes of the question.

*NB a scale of 1 = strongly disagree and 5 = strongly agree was used.

**Question 1**

51% percent of those surveyed noted that they strongly agreed current non-technological forms of team management could be improved on. None of the respondents voted lower than a "3" for this question, meaning that all of them agreed that team management could be improved upon.

**Question 2**

Again, no respondent voted lower than a "3" when asked whether a web application could improve team management. A smaller percentage, 43% strongly agreed that a web application could make improvements.

**Question 3**

A fairly staggering 85% noted in question three that they were not aware of any existing team management applications.

**Question 4**

All respondents voted higher than a "3" in noting that the ability to add player information is important. 57% strongly agreed.

**Question 5**

71% of respondents strongly agreed that the ability to add and maintain fixtures was important. All but 2 respondents voted strongly agree, and again no lower than a "3" was recorded.

**Question 6**

The same amount of respondents were indifferent to staff details being maintained as those who strongly agreed.

**Question 7**

71% of respondents voted "4" that they agreed a web application would make it easier to manage a sports team.

**Question 8**

There was an overall consensus that an ice hockey management system would not encourage new teams to form, with the results even across "1", "2", and "3".

**Question 9**

Only three written responses were recorded as suggestions for what the system should include. They included the ability to add the score to a match already played, keep track of upcoming matches, add a goals statistic to a player, and the ability to make a team selection.

**Question 10**

Only two written responses were recorded as suggestions for what the system should not include and included personal contact information and "hidden fees."

**Question 11**

85% of respondents said they would be willing to pay a monthly fee to use an ice hockey management system, with only 15% saying they would not.

**Question 12**

100% of all respondents selected that they would not be willing to pay a higher fee to own an ice hockey management system with no further subscription fee.

The results of the questionnaire give an insight into what features a user will consider fundamental to the operation and value of the system. If these issues are not taken into consideration, the developer runs the risk of creating an application which they may think is more useful than it actually is. In understanding this data, the developer is able to put into plan a system which will be deemed valuable to the user, rather than what the developer may think is valuable to the user. Some of the answers to the questionnaire highlight this. For example, an initial plan was to have the system be available at a one-time only fee, whereas the survey shows zero interest in this plan. A monthly subscription business model may be the best business plan as the questionnaire determined interest as such. In order to maximise the effectiveness of the development, it is now important to compile a list of system requirements which adhere with the data collected.

## 2.3 System Requirements

An amalgamation of research and survey responses now give the developer the ability to organise the overall specification for the system itself, by putting into place a list of system requirements. System requirements can be divided into functional requirements and non-functional requirements.

### Functional Requirements

Functional requirements are those which 'describe the core functionality of the application.' (University of North Carolina, 2012.) By coupling earlier research with the survey response, the following functional requirements have been identified:

- The application must allow for the creation of players, i.e. the user must be able to create a list of the players in their team. It must also allow players to be updated and deleted.
- The application must allow for the creation of fixtures, i.e. the user must be able to create a list of their upcoming games. It must also allow for these fixtures to be updated and deleted.
- The application must allow for the creation of staff members, i.e. the user must be able to compile a list of the team staff. It must also allow for staff to be updated and deleted.

These are the requirements considered fundamental to the performance of the website. It is possible that following implementation and feedback, further features of the web application could be implemented to more robustly fit the needs of the user. However, these are the functional requirements considered necessary for the application to be of worth, and address the problem identified earlier. Typically, functional requirements specify what the system should do. Therefore, the data collected from the questionnaires has suggested that the intrinsic features of the application should be the ability to create, update and delete data relating to the team. This would allow the user to successfully manage the day to day operations of their team.

In order for the developer to fully understand the functional requirements, it is beneficial to compile these requirements into a set of simple user stories. User stories are hypothetical representations of what a user with a specific requirement might say. In creating user stories, the developer is afforded the ability to understand why a function is required and the criteria which must be met in order to fulfil this requirement. In other words, it shifts the focus from writing about requirements to talking about them. (Mountain Goat Software 2016.) While user stories are normally a feature of agile methodology, the developer still believes they are a useful tool of analysis. The table below will set this out.

| User Story | Why? | Criteria |
|---|---|---|
| I want to be able to create a team. | In order to maintain my team in real life, I must be able to keep track of them. | Provide a team section. |
| I want to be able to add the players in my team to the application. | Be sure of who is eligible for team selection. | Allow the details of each player on the team to be added to a database. |
| I want to be able to maintain my players' information. | In order to keep the data current. | Allow for the updating and deletion of player records. |

| | | |
|---|---|---|
| I want to be able to add fixtures | In order to keep on schedule. | Allow for fixtures to be added. |
| I want to be able to edit fixtures. | In case any details such as opponent or date changes. | Allow for updating and deletion of fixture records. |
| I want to be able to maintain my staff members. | Staff are subject to change. | Allow for the creation, updating and deletion of staff members. |
| I want to be able to find everything I need. | The application would only be useful if easy to use. | Put in place an effective menu. |

(Table 3. – User Stories )

## Non-Functional Requirements

In relation to the developer, it is really the non-functional requirements of the site which cause the most concern. The non-functional requirements 'describe system attributes such as security, reliability, maintainability and usability.' (SAFE, 2015.) While functional requirements provide an idea of what the user wants, the non-functional result of this can often be constraining and even possibly detrimental to the integrity of the system. Functional requirements can be thought of as what the application should do, while non-functional requirements set out how it does something. Non-functional requirements tend to be similar for most projects of this calibre, and some information regarding each one is set out below.

The Volere template is an industry standard template for system requirements. It outlines within six main non-functional requirements, which are as follows.

### USABILITY AND HUMANITY REQUIREMENTS

This regards the client's wishes for how easy the system should be to operate. Generally this should cover efficiency, ease of remembers, how many errors the user makes, overall satisfaction and feedback. This may require 'special consulting sessions with the client.' (Volere, 2012, s.11)

**PERFORMANCE REQUIREMENTS**

There are a number of performance requirements set out in Volere also. Speed and latency requirements regards the time the system takes to carry out a specified task. Speed issues tend to vary from project to project so it is important to understand what is required of the system. Precision requirements set out how accurate the system should be, i.e. the scale of a map that may be required. Capacity requirements set out the volume of data that the system should be able to process. (Volere, 2012, s.12)

**OPERATIONAL REQUIREMENTS**

Considerations here include the requirements of interfacing with other systems, such as other applications that the solution may need to operate successfully. This heading also contains release requirements which set out the release cycle to be followed for the software. (Volere, 2012, s.13)

**MAINTAINABILITY REQUIREMENTS**

The major requirement under this bracket is the requirement of maintenance, which set out the desired time necessary to make changes to the solution. It also includes supportability requirements which note how much support should be available, such as a help desk. (Volere, 2012, s.14)

**SEQURITY REQUIREMENTS**

Under this heading comes access requirements which relates to who is authorised to access the product. Integrity requirements relate to the databases within the software and the software itself. In the same vein comes privacy requirements which must ensure that the individuals about whom the system stores data is kept secure and within line of the Data Protection Act. (Volere, 2012, s.15)

**LEGAL REQUIREMENTS**

Finally, legal requirements set out the degree to which the software must obey certain laws. The objective of these requirements is generally to avoid delays due to legal disputes. The most obvious consideration here is again the Data Protection Act. So long as the Act is not undermined, no problems should arise. (Volere, 2012, s.17)

## 2.4 Professional Issues

### Business Case

The business case for this project is fairly straight-forwarded. The intention is not necessarily to benefit financially from the proposed application, but instead to offer a solution to the problem at hand. However, as can be observed in data retrieved from the questionnaires, there may be a market to make the application subscription based. An initial plan for monetising the venture should the opportunity arise was to charge a one off fee for access to the application. As was found though, none of the select group would be willing to pay in such a way. In contrast, a high percentage of respondents noted that they would be willing to pay a smaller monthly subscription fee to avail of the service. If this were to be put into motion, the business case of this project would be satisfied.

There exist also a number of stakeholders who could be seen as beneficiaries should the application be monetised, and there role is most important in this analysis stage. (Enfocus Solutions, 2011.) The developer of course would benefit from the actual building and launch of the application. Others may benefit indirectly. Managers of teams may find the tool helps them improve the overall performance of their team, and thus would benefit from increased profit. Players themselves may benefit from an increased wage if the system were to improve their game. The general advancement of the team would also lead to increased merchandising and ticket sales for ice hockey clubs.

In a business scenario there would of course be a cost to creating the solution as well. Larger projects which span several years are likely to have more recurring cost than a project of this calibre which would only really be subject to start-up costs. (Williams, Smith, 2003.)

### Risk Assessment

Risk management strategies give the developer the opportunity to identify ‘strengths, weaknesses and threats’ to the project’s success. (Duggan, 2015.) The developer does not foresee many potential risks, however there are a few factors which could lead to the project being delayed or not delivered on time. Firstly, an overrunning of time could be considered a foreseeable risk. To mitigate this risk, the helpful scheduling tool Trello will be used through

the course of the project to ensure tasks are allocated and scheduled in a timely manner. Furthermore, the loss of data could make a large impact on the project were it to occur. The project will be backed up in three locations: the developer's computer, an external hard-drive, and in the cloud via Dropbox.

### Ethical Considerations

The project has been deemed category Z by the ethics committee as it is completely non-invasive, and the only interaction with the public involved the filling out of questionnaires.

## 2.5 Summary

The analysis stage has allowed for an insight into what means can be used to combat the project's problem. Valuable feedback from a number of distributed questionnaires has allowed the developer to compile a number of functional and non-functional user stories and conceptualise them as user stories. These requirements will subsequently be implemented into the system as best as possible. It is foreseeable that issues may arise if certain aspects are unable to be implemented. However, for now the developer is happy that a useful end product can be created and subsequently built upon if any feedback deems it necessary.

# CHAPTER 3. Design

The next stage in the development of the proposed solution is design. There are a few important design features to consider in this chapter. Design is an aspect fundamental to software development. It is important to consolidate as much of the information gathered as possible and implement it into a solid and finalised design plan for the application.

Design does not only relate to how the application will look. While HCI factors are considered in this chapter, so too are software concerns and database design. It is intended that the design process will allow for the creation of an aesthetically pleasing and functional prototype.

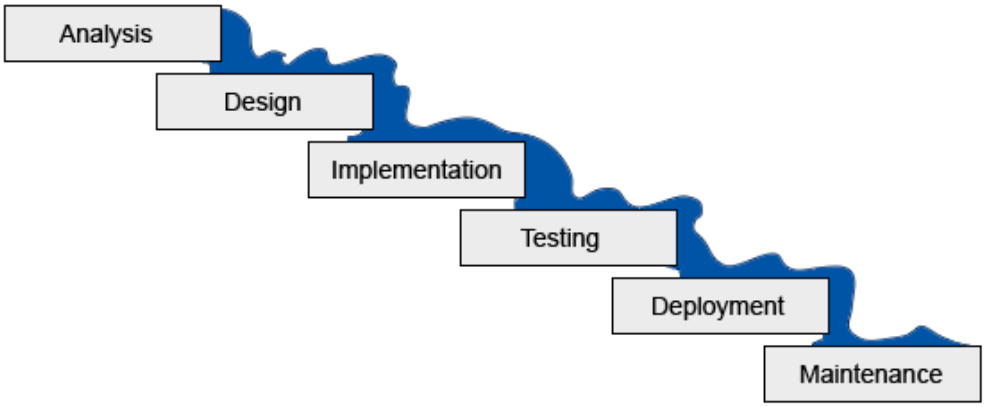
This chapter will address all of the issues above and contain a number of template and prototype designs. Consideration will also be given to the design of the graphical user interface - Gokjo Adzic noted in a 2007 article that 'if the GUI is not polished it is best to delay the project to clean it up.' (Adzic, 2007.) With this in mind, it is important the GUI implemented is sleek. Furthermore, this chapter will explore the architectural design of the system which will be used in order to perform the three main functions of the application. The chapter will conclude with information on the design of the database before summarising the key aspects of the subject.

## 3.1 Methodology

As aforementioned the methodology used throughout this project has been the waterfall method. Methodology is a fairly important factor in software development as it allows the developer to closely hone their work under a number of various headings. Another option explored was the use of the agile methodology. However, as the developer has previously developed using the waterfall method, and has little experience of the agile approach, it was decided that the approach used would be the one with which they were most familiar.

The waterfall method consists of six key steps in the software development lifecycle, namely analysis, design, implementation, testing, deployment and maintenance. (Royce, 2003.) As the analysis stage has already been carried out, the developer will now follow on to

the next stage of the waterfall in design. In following this structure, the developer is afforded the ability to focus on each individual aspect of the project's lifecycle, as well as foresee which stage will be coming next, and plan ahead. The following diagram illustrates the key components of the waterfall methodology.



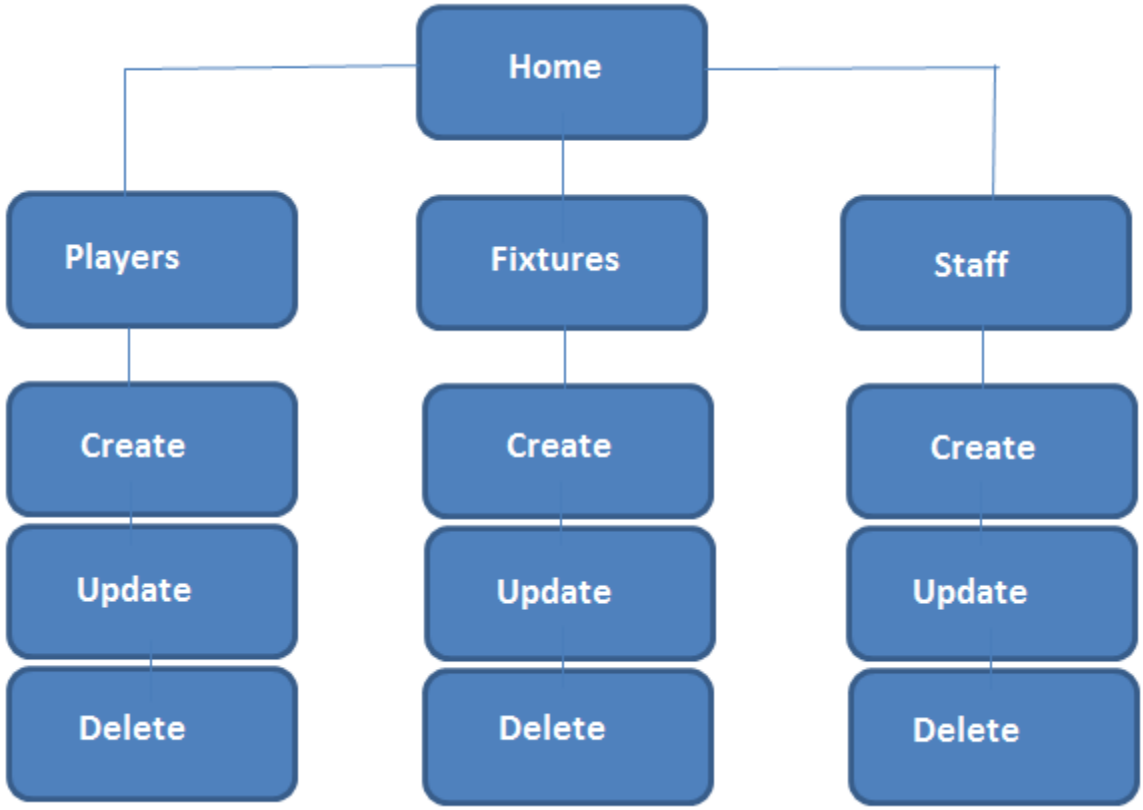

(Fig. 1 – Waterfall methodology.)

## 3.2 Architecture

At this stage in development there are a number of vital technological decisions to make. While the design process does include the aesthetic development of the solution, so too does it include architectural considerations. These considerations will be presented in the pages following and will describe the means and technology which will be used to create the web application.

At this point it is valuable to set out how the site will flow, so to speak. At present the solution is comprised of three key features: player management, fixture management and staff management. These are the key aspects required to meet the functionality objectives of the site. It is intended that when the user comes to use the web application they will be greeted by a sleek and stylish homepage which will be easily navigable. The home page will contain links to the three main areas of the site. Upon clicking a link they will be taken to the desired page which will house the content for each section. As the three main functions are similar, so too will be the navigation on each page. Links will then be available to allow the user to add, edit or delete a member in either field. An ever present menu bar will be sandwiched between the site header and the main body of the page and will allow the user

to either return to the homepage, or navigate to another page on the site. The site map overleaf presents a graphical representation of how the site will flow for the user.




(Fig 2. Site Map)

It would be easy to create a simple HTML website without functionality, but in order to correctly address the problem at hand it is important that the proposed application be a functional solution to the problem at hand. Karl Wiegers notes that 'Establishing each chunk of functionality lets you sequence construction to provide the greatest product value. (Wiegers, 1999) There are various ways in which the site could be given functionality, but there is one particular means of web development which has been chosen to facilitate the application.

**REST**

REST stands for representational state transfer and is an architectural style used for web development. The main advantages of REST are the ability to create a very fast and reliable web application which has the ability to be scaled to suit any number of user. The

REST architectural style relies on a client-server uniform interface. It essentially decouples the architecture and allows for independent function. (Fielding, 2000.) HTTP verbs, such as GET, PUT, POST and DELETE are used in order to handle the data. It presents it in both XML and JSON, with the latter being the most commonly used. Furthermore, uniform resource identifiers are used in order to access resources. An early example of this could be illustrated as follows. If the user has created a team of players within the application, unbeknownst to them they could travel to a page such at http://examplesite/REST-API/players and be presented with all of the data on their team in JSON or XML format. The REST architecture is a quickly evolving and frequently used architecture in modern day web development, and so it is an ideal choice for the project. The process of making this application a RESTful web services is expanded upon in the implementation chapter.

**IDE**

The Netbeans development environment has been selected in which to create the application. The project will be written in Java and make use of the Java EE7 platform in order to create a functional RESTful web service. Netbeans has a number of advantages, but the primary reason for which it was chosen is it's built in server and database services. (NetBeans, 2016.) This will allow for the creation of the application in an environment in which every resource required is readily available.

**Front-End**

It is intended at this point that HTML5 and CSS will be used to develop front end web pages in which to house the main functional aspects of the website. The decision was made to use these two technologies as they are proven to work well together and allow for the creation of sleek web pages. The web application will be designed to have a fairly simple layout to aid the user in their use of it.

**Back-End**

Glassfish 4.1 will be used at the web server for the project as it interacts well within the Netbeans IDE. For similar reasons, the Derby database has been chosen with which to create the required database tables to allow the application to function.

**Model – View – Controller**

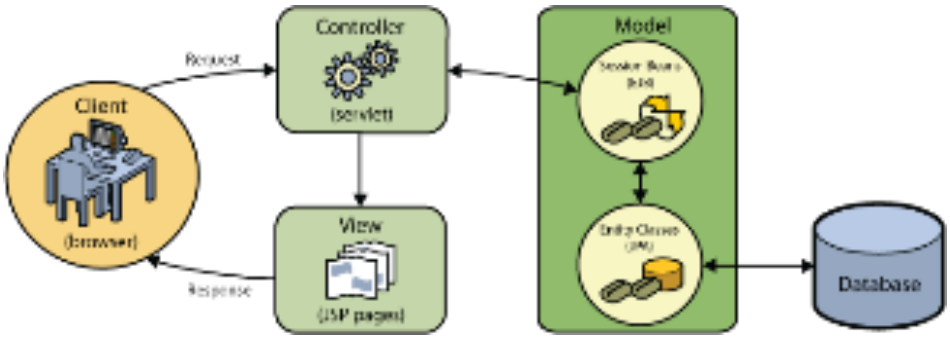
The application will be built with the Model – View – Controller design pattern in mind. MVC is an architectural style which works by splitting the development process into three main components. These are:

- The Model – which represents knowledge, either as an object or a series of objects.
- The View – which creates a visual representation for the user of the model. It retrieves data from the model by querying it.
- The Controller – which is the link between the user and the system. Depending on the user's input, it will create tailored views to be displayed. (Coding Horror, 2008.)

Terence Parr notes that MVC is the most successful and simplistic architectural framework:

> The controller in a web app is a bit more complicated, because it has two parts. The first part is the web server (such as a servlet container) that maps incoming HTTP URL requests to a particular handler for that request. The second part is those handlers themselves, which are in fact often called "controllers." So the C in a web app MVC includes both the web server "overlord" that routes requests to handlers and the logic of those handlers themselves, which pull the data from the database and push it into the template (Terence Parr, 2008 (Transcribed by Bill Venners.))

The following diagram presents a graphical illustration of MVC:



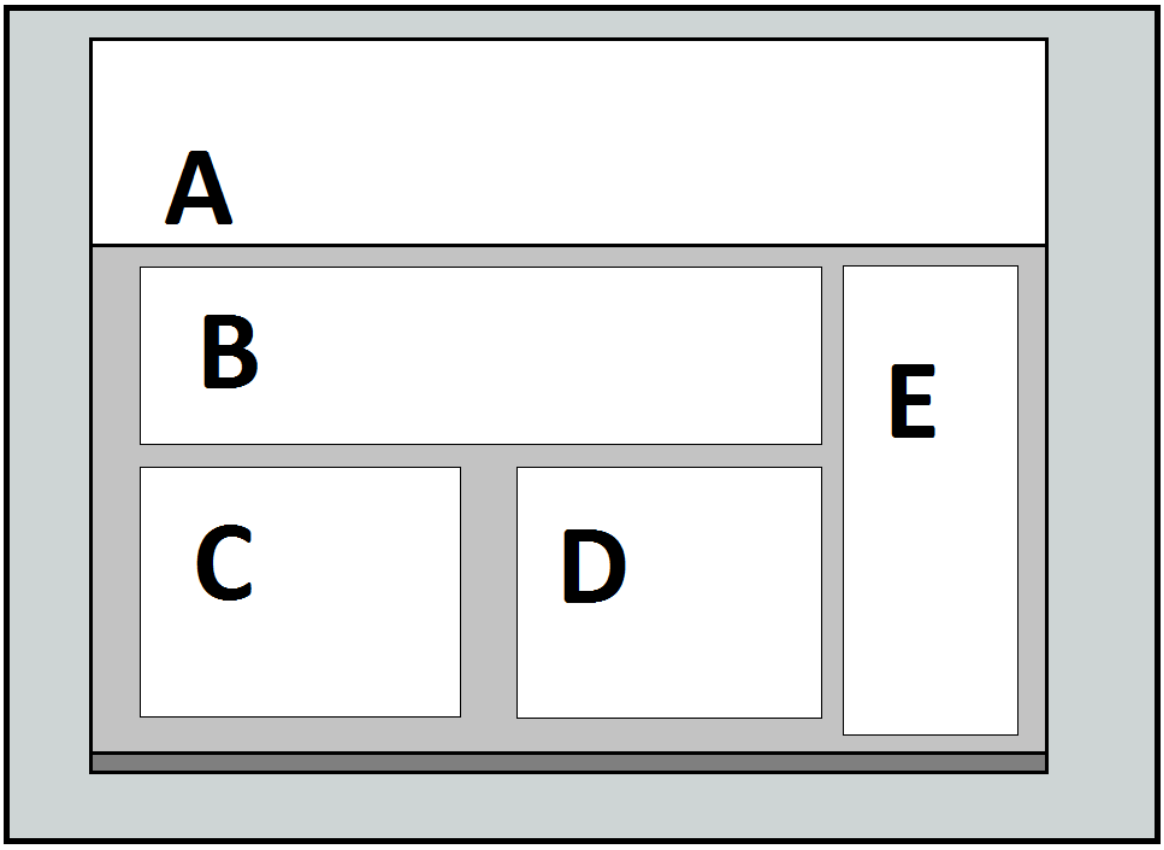

(Fig. 3 – Model – View – Controller diagram.) (Netbeans, 2015.)

## 3.3 Template Design

In order to get a good idea of where to take the finalised design of the web application it is important to go through two initial development stages. The first of these is template design where the intention is to create a very simple layout design of the pages to be included in the web application. These designs are intended to be very basic and simple illustrate where the key features of the application will go. Each of these designs will now be discussed in detail.

**The Home Page**

The home page will be the first port of call for the user. The index page is incredibly important as it sets the feel of the user experience from the outset. The home page should present the user with a number of options detailing the options they can choose in order to use the site effectively. The following template design is proposed:

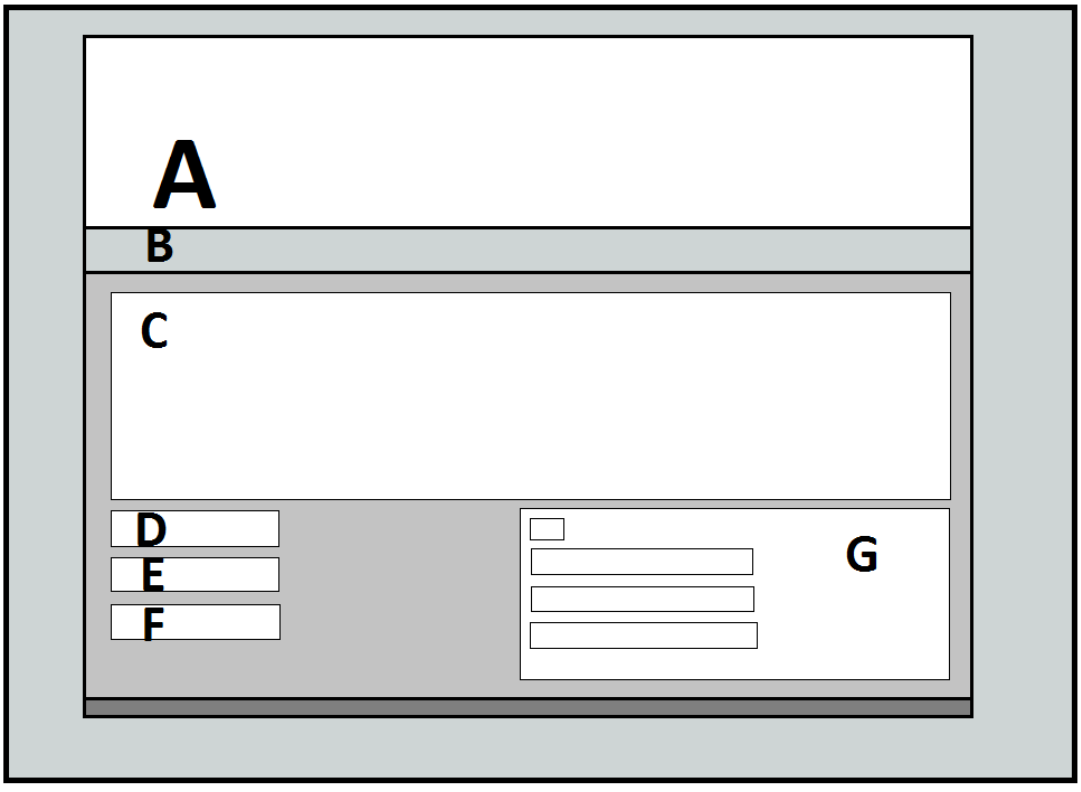



(Fig. 4 – Homepage template design.)

The layout of the homepage is intended to stay consistent with the other pages in the website. The five elements of the homepage are as follows:

- **A –** The header, which will consist of a banner bearing the title of the application overlaid on an image. This will remain consistent through-out.
- **B –** The "players" image which on click will navigate the user to the team management section of the application.
- **C –** The "fixtures" image which on click will navigate the user to the fixture management section of the application.
- **D –** The "staff" image which on click will navigate the user to the staff management section of the application.
- **E –** A twitter widget which will display tweets from the official Elite Ice Hockey League twitter feed which is the hub for all UK based ice hockey news and updates.

Upon navigation to one of the key areas of the application, the user will be met with a consistent design tailored to the feature requested.




(Fig. 5 – Players/Fixtures/Staff template design.)

As can be observed in figure 5, the subsequent pages are at this stage designed to be very similar in feel to the homepage. The user therefore will not come across any unfamiliar elements of design as they navigate through the application. There are however more elements to the subsequent pages in order to facilitate their functionality:

- **A –** The header, again bears the banner and is consistent throughout.
- **B –** When navigating away from the homepage, a menu comes into play in order to ensure the user is just as easily able to use the system to its full extent without having to constantly return to the homepage.
- **C –** This element will house the relevant database table for either players, fixtures or staff. While it is placed at the top of the body of the site, it will change in size depending on how many records the table holds. For example if the user has not yet added any players to their team, the table will be blank and they will be invited to add players to their team accordingly. The table can be manipulated to show ten players and allow the user to leaf over to the next page. The purpose of this is to ensure the table doesn't drag the whole way down the page if the user were to add hundreds of records.
- **D –** A button to allow the user to add a player.
- **E –** A button to allow the user to update a player.
- **F –** A button to allow the user to delete a player.
- **G –** The form which appears when the user wishes to update their team.

While this template relates to the players section of the web application, so too will it apply to the fixtures and staff management sections. The developer feels that this will promote the usability of the site by having common elements across the pages. While the subject matter and data input may change depending on what section the user is in, the process of doing it will remain the same. Of course the data held for each individual table will vary from page to page, however this is not a concern for the template design. These matters will be addressed in the database design section of this chapter which will note the entities required per database table.

## 3.4 Graphical User Interface Design

Before progressing to the prototype design stage it is vital that the developer take into consideration the graphical user interface. Many believe that GUIs were invented by Apple or Microsoft, but their roots can be traced to the early 60s. (Baykov, 2013) Modern GUIs are by contrast very sleek, so it is important to develop one for this system. While the previous template designs create a layout for the pages of the application, they do not give any insight as to the actual presentation of the site.

**Colour Scheme**

A clear and consistent colour scheme is an excellent means of ensuring the application is easy to use for the user. It is important that the colour palette used for the application works well together, i.e. that there are no harsh colours, or any colour text on a coloured background. The best way to ensure this is to make all of the text in the body black, and any required text on background colours, such as in the menu for example, white. In doing this the accessibility of the application is promoted, as it shouldn't prevent anyone with impaired sight from viewing the text. Furthermore, the Arial font family is used throughout; a standard font which is stylish and easily read. The following colour scheme was chosen to be used throughout the application.

(Fig. 5 – Colour Scheme.)

**Navigation**

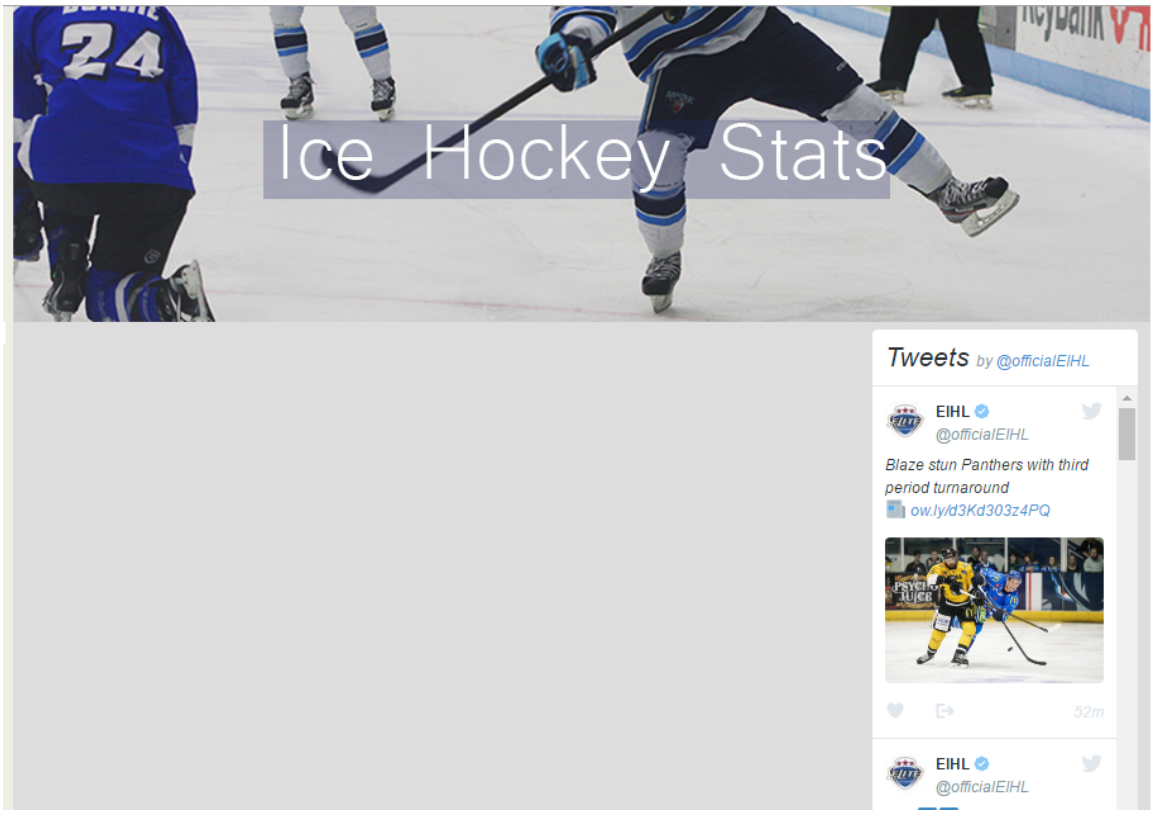
As navigation is intended to be carried out via the menu bar, no subsequent buttons were required to allow the user to get around the website. As this project is being developed as a web application is does not require certain GUI tools that a mobile application might require for example.

**Layout**

As HTML5 and CSS are being used to control the aesthetic features of the web application, certain lines of code can be used to ensure the web application will display well on any computer, regardless of display resolution. Therefore, it is not foreseen that there should be any issues in regards to ensuring the layout of the application remains consistent.

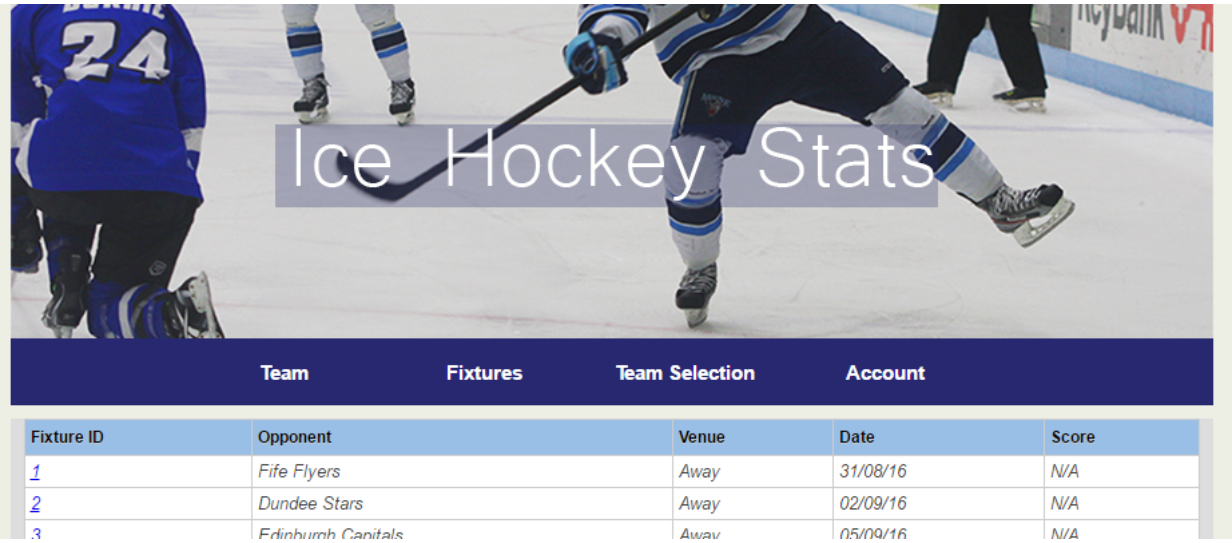
## 3.5 Prototype Design

'Creating prototypes allows a developer to improve a design concept quickly. They can interactively revise and refine an idea in a matter of minutes.' (Gube, 2013.) With a colour scheme in mind and a number of template designs to work from, it is possible to create a number of prototype designs for the application. These designs are used to give the developer a more robust idea of what the final product should look like. With the colour scheme and template implemented, the prototype design for the homepage is as follows.




(Fig. 6 – Homepage prototype design)

As can be observed from figure 6, the prototype design has implemented both the layout from the template design as well as the colour scheme noted under the graphical user interface section. The main body of the page is left blank, but will include images which will link to the main sections of the application. The twitter widget too has been included. While the image design in the banner is not final, it gives the developer an opportunity to decide what best to replace it with. Crucially the final banner will have to match with the design of the web application, as the current placeholder does. It is also important to note that the developer opted for a soft grey colour for the pages background, as bright white backgrounds can often cause strain on the eyes. At this point in the development of the prototype, the developer was content that they had chosen a colour pallet which not only suited the feel of the application, but was also easy on the eye and cause no problems in terms of accessibility.




(Fig. 7 – Fixtures prototype design)

Figure 7 illustrates the page view of one of the functional pages, in this case the fixtures page. The darkest colour from the colour pallet was chosen for the menu as it acts in breaking the page up. Furthermore, the eye is easily drawn to the menu bar for this reason. The menu bar is comprised of placeholder links which will display various pages in the final product. There is not option in the menu for home, but as the user may infer, the banner at the top navigates back to the home page when clicked. This design will be implemented into all of the subsequent pages of the application.

## 3.6 Database Design

Another important factor in ensuring the application maintains functionality is to ensure that the database design is correct. For the purposes of the three main features, a database will have to be made and three table created to hold the details of the three relevant aspects, players, fixtures and staff.

**Players**

The players section of the web application will be utilised for the manager to keep track of their players and their statistics thus far. This table will have to hold information on the player's name, their age, their position, their contract expiry dates, their wage, their goals, assists and penalties. This information can in turn be used by the user to keep track not only of the personal details relating to each player, but also of their performance on the ice.

**Fixtures**

Similarly, the fixtures table will be vital to the application's functionality. It must hold the opponent for the fixture, the date, the venue and the score line which can be updated as games take place.

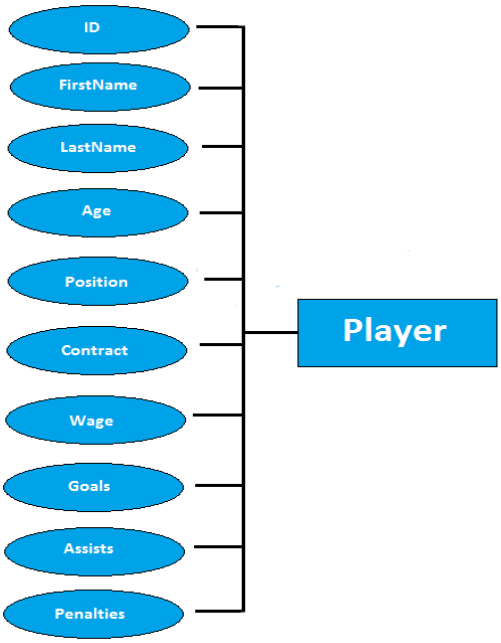
**Staff**

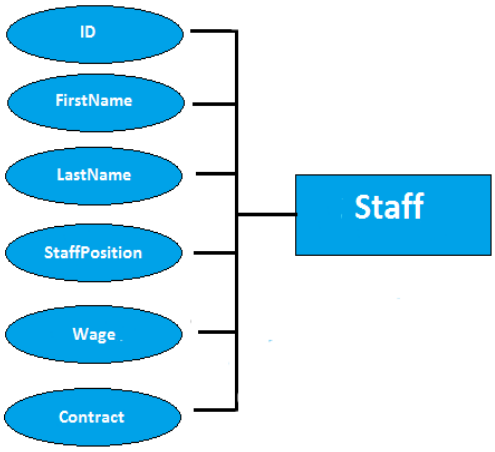
The user of the application will be able to manage their staff members by utilising the staff table. It must hold the name of the staff member, the position they hold as well as their wage and contract expiry date.

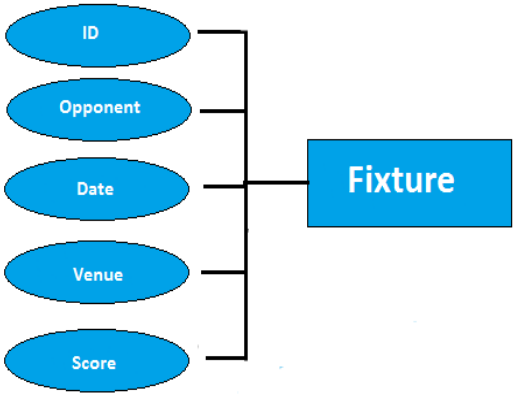
The information which will need to be stored in these tables has been summarised in the following table of database entries. ER diagrams follow on the next page.

| | |
|---|---|
| **Players** | **ID,** FirstName, LastName, Age, Position, Contract, Wage, Goals, Assists, Penalties |
| **Fixtures** | **ID,** Opponent, Date, Venue, Score. |
| **Staff** | **ID,** FirstName, LastName, StaffPosition, Wage, Contract. |

(Table 4. – Database entries.)




(Fig. 8 – The entity-relationship diagrams.)

These three diagrams show the relationship between the entities and their attributes for each table in the database. Entities are identified by a rectangular box, and attributes are identified by an oval box.

## 3.7 Summary

The design stage has allowed the developer the opportunity to finely hone the overall design of the system, from its look and feel, to its technological foundation in java and to its data model design. The developer has placed particular emphasis on maintaining the consistency of the application's design which promotes its usability and accessibility for the user. Furthermore, the navigation system for the application has been finalised as a series of buttons on the homepage and a menu bar on any pages subsequent. The next stage of development will focus on the actual coding of the application to ensure it means its functionality objectives. This, along with other important factors will be discussed in depth in the implementation chapter.

# CHAPTER 4: Implementation

With designs firmly in place it is time for the project to progress to implementation. The implementation stage is a crucial part of the development process as it is the time at which the coding and functional development of the solution will take place. The design stage was the building block for implementation. With the bare bones prototype which was created to finalise the look and feel of the system, as well as the data model designs in place, it is possible for the developer to now begin coding the application in the Netbeans development environment.

This implementation chapter will be comprised of a step by step look at all of the packages and java classes used throughout the system. It will look at these classes in detail and provide an explanation for why certain technical decisions were made.

## 4.1 Project Management

It was important from the outset for the developer to have in mind a number of key issues. Project management is vital to the development process. Its fundamental issues include how changes will be handled, how time will be managed, and how communication will be facilitated. (Hallman, 2011.) The first project management key issue to bear in mind was the fact that this project is undertaken by only one person. The individual nature of the project required the project management to be as effective as possible, while also maintaining adaptability. It was foreseeable from the outset that a few areas of concern could arise throughout the project. These concerns were attributed to the developer's inexperience in certain areas. Firstly, as the project was to be developed as a RESTful web service, the developer would have to be allotted time to achieve an understanding of the REST approach. Furthermore, as the developer was unfamiliar with front-end technologies, it was foreseeable that problems could arise. In order to mitigate this a time schedule was drawn up to try and remedy these two concerns. In making the schedule adaptable, the developer was able to ensure the strict deadline could still be met should any of the aforementioned problems be encountered.

A key aspect of the project's development was constant feedback and evaluation. As the project was overseen by a project supervisor, the developer was able to voice any

concerns relating to the development of the project and receive sage advice in return. Furthermore, a series of meetings with the supervisor ensured that constructive feedback was given to the developer as to the progress of the application and the additional reports. With this to their advantage, the developer was able to resolve problems and move forward with development. Feedback was also sought from those who answered questionnaires as well as qualified peers. This advice was also utilised by the developer to settle certain design queries as well as identify issues relating to code.

## 4.2 Tools Used

It is important to discuss the individual tools which will be used for the development of the project. There are a multitude of options for each technological aspect of the project, so it is important to have in place a number of firm decisions before advancing with the creation of the application. The following points note the technology of choice for each aspect of the solution's development.

**The Operating System**

The application is to run on the Microsoft Windows operating system. The main reason for this is the fact that it is the OS most readily available to the developer. With access to a Windows computer for all key stage of development, it is the most obvious choice. Furthermore, while the developer is also confident using IOS, the bulk of their computer using experience has been with Microsoft Windows.

**The Language**

There are a number of very popular programming languages with a plethora of support for online, but the application will be coded primarily in Java for its backend. Again, the developer is most proficient in the Java language, making it the most appropriate for development. Furthermore, HTML, JavaScript and CSS will be used for the front-end of the application.

**The IDE**

The prototype will be built within the Netbeans IDE. Netbeans provides access to a number of key features which will be beneficial to the development of the application. The most important of these for the project is the integration of server and database tools.

**Back-End**

Glassfish 4.0.1 will be the web server deployed in the project. It integrates seamlessly with the IDE and connects seamlessly to the database. Java EE 7 will be used as a platform for integrating a REST API as well as the Jersey libraries included in Netbeans.

**The Database**

Derby will be used as the database for the project. The developer has experience in its use within other projects, and again, its integration with the IDE makes in the ideal choice.

**The Testing Environment**

The use of Glassfish as the web server will allow for the application to be viewed and tested on any web browser by navigating to the local host on port 8080. The browser of choice is Google Chrome.

## 4.3 Back-End

In order to properly evaluate the implementation of the project, this chapter will look at the features of the application in three sections, first the back-end development of the system will be considered as it holds the bulk of the code, followed by the less intensive front-end, as well as the data model.

Back-end development consists of using an 'array of programming languages and frameworks when building the server-side software.' (Wodehouse, 2016.) The back-end of this project has been created using the Java EE 7 platform which is one means of supporting RESTful development. It has been primarily coded in Java, however to a lesser extent, JavaScript has been used in order to compliment certain RESTful features on the client side. The bulk of these Java classes will now be examined in order to give an understanding the way in which the application operates.

The back-end of the project itself has also been split up into sections. The three main sections include domain, service and utilities, with the front-end held in a client section. The main classes contained under these headings will now be examined in course.

**Domain Java Classes**

Three java classes are housed within the domain package, each relating to a vital function of the web application, i.e. players, fixtures and staff. As these classes are fairly similar, except for information necessary for each section, only one will be examined. Players.java will be the class considered.

The Players.java class is the class which controls information relating to the players of the user's team. The class opens with a number of imports which allow for the creation of a number of different functions. These same imports are found across a variety of the Java classes used in the system as they are required for each. Figure 9 shows the list of imports.

```
import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;
```

(Fig. 8 – Players.java imports.)

The first of these imports, 'Serializable' is used for the process of converting an object into bytes for transmission across the network. (Oracle, 2016.) This is followed by a number of imports relating to Java persistence. These included column, entity, Id and table. As may be inferred, Java persistence is used in this context in serving the relational database included within the system. This use of the Java persistence API allows for the information submitted by the user to be sent to the relevant database table.

```
@NamedQueries({
    @NamedQuery(name = "Players.findAll",
            query = "SELECT p FROM Players p"),
    @NamedQuery(name = "Players.findByPlayerid",
            query = "SELECT p FROM Players p WHERE p.playerid = :playerid"),
    @NamedQuery(name = "Players.findByFirstname",
            query = "SELECT p FROM Players p WHERE p.firstname = :firstname"),
    @NamedQuery(name = "Players.findByLastname",
            query = "SELECT p FROM Players p WHERE p.lastname = :lastname"),
    @NamedQuery(name = "Players.findByNumber",
            query = "SELECT p FROM Players p WHERE p.number = :number"),
    @NamedQuery(name = "Players.findByPosition",
            query = "SELECT p FROM Players p WHERE p.position = :position"),
    @NamedQuery(name = "Players.findByDateofbirth",
            query = "SELECT p FROM Players p WHERE p.dateofbirth = :dateofbirth"),
    @NamedQuery(name = "Players.findByHometown",
            query = "SELECT p FROM Players p WHERE p.hometown = :hometown")})
```

(Fig. 9 – Players.java queries.)

Following the imports is a very important section which allows for the querying of the database. As can be observed in figure 9, a number of these named queries are listed. These include the ability to search for all of the players, for players by ID, by first name, last name,

and so on and so forth. This is intended to provide a vital tool for the user, allowing them to search for the players in their team effectively. The ability to search for players by number of goals for example allows the user to make team selection decisions.

```
@Id
@Basic(optional = false)
@NotNull
@Column(name = "PLAYERID")
private Integer playerid;
@Size(max = 20)
@Column(name = "FIRSTNAME")
private String firstname;
@Size(max = 20)
@Column(name = "LASTNAME")
private String lastname;
@Column(name = "NUMBER")
private Integer number;
@Size(max = 9)
@Column(name = "POSITION")
private String position;
@Size(max = 20)
@Column(name = "DATEOFBIRTH")
private String dateofbirth;
@Size(max = 25)
@Column(name = "HOMETOWN")
private String hometown;
```

(Fig. 10 – declarations.)

As figure 10 shows, the class then implements a number of lines of code which control the input of data to the tables. These lines of code make a series of important variable declarations, such as 'private String firstname.' Furthermore, the names of the columns in the database tables are also defined – @Column (name = "FIRSTNAME"). This ensures that the data entered by the user is sent to the correct place in the database table. The class holding the code in figure 10 also crucially implements the serializable class.

The remainder of the Players.java class is then made up of a number of getters and setters. These getters and setters are used to retrieve the variables initialised previously as well as set their contents to new values. An example list of these getters and setters are laid out in figure 11 on the following page.

```
public Players() {
}

public Players(Integer playerid) {
    this.playerid = playerid;
}

public Integer getPlayerid() {
    return playerid;
}

public void setPlayerid(Integer playerid) {
    this.playerid = playerid;
}

public String getFirstname() {
    return firstname;
}

public void setFirstname(String firstname)
    this.firstname = firstname;
}

public String getLastname() {
    return lastname;
}
```

(Fig. 11 – getters and setters.)

As aforementioned all of the classes under domain are similar and therefore do not require in depth analysis.

**Service Java Classes**

The service package is another collection of classes which are fundamental to the operation of the application. More specifically, it is within the service package where the classes which allow the system to function as a RESTful web service lie. These classes include AbstractFacade.java, ApplicationConfig.java as well as three specific classes relating to each function: PlayersFacadeRest.java, FixturesFacadeRest.java and StaffFacadeRest.java. The contents of these classes will now be described.

**AbstractFacade.java**

The abstract facade class is vital in the operation of the application as a whole. The main function of this class is to manage the entities for all of the accompanying service classes. As can be seen in figure 12, the abstract facade class is comprised of a number of methods, create, edit and remove. Furthermore, the abstract facade also has methods for finding an object by ID as well as a find all method.

```
public abstract class AbstractFacade<T> {
    private Class<T> entityClass;

    public AbstractFacade(Class<T> entityClass) {
        this.entityClass = entityClass;
    }

    protected abstract EntityManager getEntityManager();

    public void create(T entity) {
        getEntityManager().persist(entity);
    }

    public void edit(T entity) {
        getEntityManager().merge(entity);
    }

    public void remove(T entity) {
        getEntityManager().remove(getEntityManager().merge(entity));
    }
```

(Fig. 12 – AbstractFacade.java)

The term ‘facade’ as used throughout these classes relates to a design pattern which helps to structure the code. The developer made use of this façade to allow objects within the application to use other objects. This method is used to achieve loose coupling which is always a desired outcome when programming.

**ApplicationConfig.java**

In order to fully implement the RESTful architecture, JAX-RS libraries are used. These are implemented within ApplicationCongfig.java. Another interesting point of note, is that the

path which can be followed in order to access data in JSON format is defined within ApplicationConfig.

```
@javax.ws.rs.ApplicationPath("REST-API")
public class ApplicationConfig extends Application {
```

(Fig. 13 – Path.)

As figure 13 shows, the developer has opted to name the path "REST-API." What this essentially means is that end points in the system can be reached by using the path /Rest-API/. For example in order to retrieve all of the information on players in JSON format, the path /REST-API/players/ can be used.

ApplicationConfig.java also ties together all of the resource classes within the service package as can be observed in the image below.

```
private void addRestResourceClasses(Set<Class<?>> resources) {
    resources.add(service.FixturesFacadeREST.class);
    resources.add(service.PlayersFacadeREST.class);
    resources.add(service.StaffFacadeREST.class);
    resources.add(util.CrossOriginResourceSharingFilter.class);
```

(Fig. 14 – Resource Classes.)

**PlayersFacadeRest.java**

The PlayersFacadeRest.java class, similar to the AbstractFacade class, follows a design pattern which allows for the functioning of the players section of the application. It is the class which allows the RESTful nature of the players function. We also define the path at which the players' endpoint exists within the system in this class.

```
@Stateless
@Path("players")
public class PlayersFacadeREST extends AbstractFacade<Players> {
    @PersistenceContext(unitName = "IceHockeyStatsEnginePU")
    private EntityManager em;
```

(Fig. 15 – Players path.)

As figure 15 shows, we define the Players path as being “players”, and thus, as aforementioned this can be reached by navigating to /REST-API/players.

This class also holds a number of key methods which allow for functions to be carried out. The REST architecture requires that HTML terminology be used, (Zuzak, 2010) and it is in the PlayersFacadeRest.java class in which this terminology is held.. These can be viewed in figure 16 below.

```
@POST
@Override
@Consumes({"application/xml", "application/json"})
public void create(Players entity) {
    super.create(entity);
}

@PUT
@Path("{id}")
@Consumes({"application/xml", "application/json"})
public void edit(@PathParam("id") Integer id, Players entity) {
    super.edit(entity);
}

@DELETE
@Path("{id}")
public void remove(@PathParam("id") Integer id) {
    super.remove(super.find(id));
}

@GET
@Path("{id}")
@Produces({"application/xml", "application/json"})
public Players find(@PathParam("id") Integer id) {
    return super.find(id);
}
```

(Fig. 16 – PlayersFacadeREST.java)

The terms “POST,” “PUT,” “DELETE,” and “GET” are used throughout the class to ensure that the correct function is attributed to the required query. The methods in figure 16 also provide some of the intrinsic functionality of the player’s class. “Create” for example is the method used in order to add a new player to the team. “Edit” is used to update their information and “Remove” is used to delete them from the system. Other methods included

in this class include the ability to find all of the players, find them based on a specified criteria and find a range of players.

Another important feature to note from figure 16 is "@Produces." This is the part of the code which allows the user to define which format they want to return data in. Within the code in figure 16 it has been specified that both XML and JSON be returned, though if only JSON is required, the term "application/xml" can be easily removed.

A final package within the back-end of the project is the utilities package, which holds only one Java class – CrossOriginResourceSharingFilter.java. This java class simply allows the API to be called from any domain.

## 4.3 Front-End

While the back-end of the system houses all of the service side code, the client side code can be found in the front-end of the project. Front-end development has been described as the desire to have some control over the data you are receiving. (Coyier, 2015.) HTML, CSS and JavaScript were utilised in the front-end of the application, and some of these classes are described below.

**Players.html**

The Players.html class gives us a good insight into the HTML which was used in the creation of the webpages. Standard HTML language was used consistently throughout the system. An example of this can be seen in the menu bar implemented into Players.html which allows the user to navigate through the application. The code for this can be observed in figure 17.

```
<div id="menu">
        <ul>
        <li class="menuitem"><a href="players.html">Team</a></li>
        <li class="menuitem"><a href="fixtures.html">Fixtures</a></li>
        <li class="menuitem"><a href="staff.html">Staff</a></li>
    </ul>
```

(Fig. 17 – Menu bar HTML)

One of the most important features of the Players.html file, and for the relate HTML files for fixtures and staff is its presentation of the table to which data is entered by the user. An example of the code used for the “First Name” column can be seen in figure 18 below.

```
<tr>
<td>First Name</td>
<td><input type='text' id='firstname' name='firstname' value='<%= firstname %>'/></td>
</tr>
```

(Fig. 18 – First Name HTML.)

**PlayersRestClient.js**

JavaScript files are also used to create client side code for the three main functions of the application. An important technology used here is Backbone. Backbone is the technology used to structure the code to enable the aforementioned Model – View – Controller pattern.

A good example of Backbone in work can be seen in the PlayersRestClient.js class where the creation of a new model is undertaken. This can be observed in figure 19.

```
    // This view is used to create new model element
    views.CreateView = Backbone.View.extend({
        initialize: function() {
            this.render();
        },
        render: function(eventName) {
            $(this.el).html(this.template());
            return this;
        },
        template: function(json) {
            /*
             *  templateName is element identifier in HTML
             *  $(this.options.templateName) is element access to the element
             *  using jQuery
             */
            return _.template($(this.options.templateName).html())(json);
        },
        /*
         *  Class "new" is used on the control to listen events.
         *  So it is supposed that HTML has a control with "new" class.
         */
        events: {
            "click .new": "create"
        },
        create: function(event) {
            this.options.navigate();
            return false;
        }
    });

})(app.module("views"));
```

(Fig. 19 – PlayersFacadeRest.js)

## 4.4 Testing

Testing is perhaps the most fundamental aspect of software development. An application which is not thoroughly tested is possibly full of bugs and not delivering its intended functionality. (Johnston, 2014.) In order to remedy this, the developer undertook a number of key tests during the implementation stage.

Testing primarily took place within a web browser as the application is intended to be used at this point only on a PC. It is important to note that the developer's testing was carried out in two stages. The first stage of testing was informal and was carried out during the creation of the application. It was mainly used at this point in ensuring individual modules worked as they should at the time. This allowed the developer to ensure that one thing had been completed before moving on to the next. However, while this testing was helpful at the time, it is not a strong enough foundation on which to declare that the system has been tested. In order to be sure of the application's functionality, the main testing stage was a second, formal testing stage.

The following sections will detail the results of the three main issues tested. A table holding all of the test data can be located in appendix A3.

**Testing Across Browsers**

Many internet browsers exist. Certain browsers have maintained dominance over certain time periods, however at present a multitude of browsers are used by web surfers across the world. An inherent issue with this is the fact that things tend to be different across browsers. On certain occasions a web page will not load one browser but will load in another, certain forms may not be completed or content loaded. It is often a strange occurrence but it does happen. As the application was developed in the Netbeans IDE, it is important to note that Internet Explorer was the default testing browser. However, as this was changeable it was altered to Google Chrome, which was the developer's browser of choice. As the informal testing to place, it took place solely on Google Chrome.

In order to ensure that the application maintains its proportions and remains functional, it was decided to test it across a number of popular browsers. These included

Google Chrome, Internet Explorer, Microsoft Edge, Mozilla Firefox and Apple Safari. While full results can be viewed in the test suite, everything was deemed to be in place by the developer. The website remained proportional and all elements in place across the variety of browsers and no elements were unduly non-functional. In this instance therefore, the application was found to be successfully tested across multiple browsers.

**Navigation Testing**

Testing was also carried out on the navigation of the website in order to ensure that the user could reach every page of the application. This included the homepage buttons for players, fixtures and staff. It also included the menu bar throughout the website and the banner. All of these test were carried out successfully except for the banner test. It was found that the banner did not link back to the homepage. This was subsequently remedied by the developer.

**Functionality Testing**

The key area of testing for this stage was the functionality testing of the application. It was paramount that the functionality of the website be tested to ensure that the main features worked as they were supposed to. As there were three main functions of the application, testing was carried out on each page. On the player's page, the first test ensured that the table of players was successfully loaded. Secondly the buttons to allow for data entry were tested about found to be working sufficiently. A player was then added to the database and appeared in the table as expected. When they were selected and edited, their information was updated in the table. Furthermore, when selected and the delete button clicked, the player was successfully deleted from the table. The same tests were carried out on the fixtures page and the staff page to the same level of success. These tests were carried out using the browsers' refresh feature as well as exiting and re-visiting.

In light of the three areas of testing carried out, the developer is happy that the system is suitably functional.

## 4.5 Evaluation

As the end of the implementation stage approaches it is useful for the developer to evaluate their performance on the project as a whole. 'A well planned and carefully executed evaluation will reap benefits for all stakeholders.' (Zint, 2015)There are a few means of doing this, however the best is to weigh up performance against the set list of objectives identified in the analysis chapter. These objectives at the time acted as a jumping point for the developer, giving them a number of goals to reach through the performance of the task. At this point however, they can be reflected on in such a way that they provide a valuable insight into the success of the task as a whole. Each of these objectives will now be considered following the implementation of the solution.

1) **Complete detailed analysis of current solutions available for this problem, whether they be loose fitting or closer to the overall direction the proposed application will take.**

   The analysis stage allowed for a deep insight into what alternatives currently exist and what they lack. In doing this the developer decided that current market solutions were not apt to meet the requirements necessary to solve the problem. These were due to a number of reasons, but the overbearing disadvantage of all the alternatives considered was the fact that none of them were tailored enough to suit an ice hockey manager. Furthermore many of them did not have all of the features identified as being crucial to the solution. This objective was sufficiently addressed.

2) **Investigate the needs of the market through detailed questionnaires which will allow for an insight into the features which are desired from such an application. With this in mind coupled with knowledge of existing web applications available, it will be possible to put in place system requirements on which to form the foundation of the application.**

   Through use of questionnaires the developer was able to deduce what features the application should have at its core in order to best meet the needs of those who manage ice hockey teams and play in them. From this a number of strict system requirements were established. As the requirements from the analysis stage were carried forward and used as a foundation, it is fair to say that this objective was met.

3) **To complete an aesthetic and user friendly design for the application which will promote its usability by being sleek and easy to navigate.**

   The design process, though fairly lengthy did allow the developer to come out at the end with a detailed and final idea of how the look and feel of the application should be. By taking into account the information discussed in the chapter and implementing a final design into the application, this objective was met.

4) **To consider and identify the best means of creating the back-end of the application, including the environment in which it is created, the language in which it is programmed and the framework on which it is built.**

   These technical considerations were discussed in both the design and implementation chapters and options for all were decided upon. The most prominent decision, to make the application a RESTful web service, was one which dictated the choices for a number of the other technical concerns. As the application has been created using all of the decided technologies, this objective was met.

5) **To implement an attractive and functional prototype which possesses the key features identified in the aim of the project.**

   After a number of template designs were created in the design chapter, it was necessary to further create a prototype to finalise a number of design features. More work was put into this prototype and eventually, through the choice of an evolutionary approach, it became the end application. In this respect therefore, this objective was met.

6) **To test the working prototype in order to identify strong aspects and areas which are lacking in functionality or usability.**

   As can be shown from the testing section directly proceeding this section, the application was successfully tested to confirm the functionality of its main components. As this testing didn't flag any large issues, it is fair to say that this objective has also been met.

7) **To consider the best means of fixing any weak features identified and propose changes to them which will promote the overall usability of the application in future development.**

   This is the only objective laid out in the analysis chapter which will require any additional work from the developer. While all of the previous objectives have been successfully met at this point, further discussion of further developments will take place in the final chapter.

## 4.6 Summary

The implementation stage has been the most important stage in the actual development of the application. It has allowed for a RESTful web service to be created for an ice hockey club management system, as was the aim of the project as a whole. Furthermore, this chapter has laid out information regarding the thorough test of the application as well as an evaluation into the work which has been completed so far. The final chapter of this dissertation will follow.

# CHAPTER 5: Conclusions

In concluding this dissertation, there are a number of issues to discuss. These include what went well, what could have went better, and recommendations for further development. These issues will be summarised in this chapter, and a final conclusion given at the end as to the overall success of the project.

## 5.1 Positives

Creating a RESTful web service for an ice hockey club management system was an overall enjoyable experience for the developer. It allowed them to identify the current strengths in their software development as well as weakness which need to be worked upon. The following positives can be taken away from the project.

- The project followed through the waterfall methodology very well which was a pleasing outcome. As development can often bring up a number of problematic areas, it was positive that no major roadblocks were encountered along the way. Minor issues with code were come across a few times, but nothing that affected the timescale of the project.

- Similarly to the point above, the developer handled the project management aspect of the project well. Due to a strict deadline, the developer implemented a flexible schedule which kept them on course. The flexibility of the schedule was relied upon a few times as certain internal and external factors at times meant that the developer was unable to stick to the plan completely. However, as an adaptable time scale was in place, these issues were mitigated by the moving around of certain aspects of development, and as such the project will be delivered on time.

- Thorough analysis gave the developer a very solid foundation upon which to base the remainder of the project. Analysis is totally fundamental to the development process and the developer feels that the information they retrieved during this stage allowed them to progress through the project without any major issues.

- The overall appearance of the web application is pleasing to the eye. This is due to the hours spent by the developer in the design stage in designing rough ideas for layout and colour scheme, and ultimately implementing them into the final system. The developer believes that the appearance of the website is one of the factors which makes it a viable solution to the problem.

- The developer also believes that the application is easily usable. There are no 'clunky' features, and all of the pages are consistent in design and layout. It is a fairly intuitive system to use, and thus accessible to many possible users.

- The business case for the application could well be met. Upon further development, it is reasonable to say that a user would be willing to pay a small subscription fee in order to make use of the website, and thus the developer would profit for their efforts.

- Working on the practicality of coding the application allowed the developer to increase their experience in their preferred area, back-end development. While the developer was formally inexperienced in RESTful development, the project has allowed them to learn a new skill which they will be able to build upon and possibly implement into future projects.

- The application works as it was intended to in the beginning. While the features at present are fairly simple, they carry out their intended tasks. In this respect the system can be considered as functional for the means for which it was created.

## 5.2 Negatives

While many aspects of the project were created without issue, there were, as the developer expected, a number of small failures which if implemented would have improved the effectiveness of the application. These issues of course will be used as points to improve upon – and as the saying goes, we either win or we learn. Some of these issues are listed below.

- The application is more simplistic than the developer would like. This is the overbearing negative point to be brought up in this conclusion. At the beginning of the project the developer was keen that the ice hockey club management tool

would implement into its arsenal a number of features which would vastly improve the application. Firstly, the developer was unable to implement a feature which would allow for statistical analysis of team players. It was hoped that this feature would allow the user to make team selection decisions based on their player's stats. However after much deliberation it was decided that this feature would be dropped in order to ensure that other vital features of the application would work.

- Secondly, the developer is somewhat underwhelmed with the front-end of the system. While there are positives attributed to the client side of the application, such as its aesthetic design and functionality, there is a negative also. The developer had hoped that a feature would be implemented which would allow the user to 'drag and drop' their players into a team. However, as the developer has little to no experience with front-end development, the feature could not be implemented with the short time period available for project implementation.

- Finally the developer felt that the implementation stage as a whole could be improved upon in future projects. The main issue here was that the developer found themselves to be 'learning as they went' with certain technologies. This however was partly due to time constraints and other commitments. In the future, it is intended that this will be remedied with more precise study of unfamiliar technologies before the implementation stage.

## 5.3 Further Development

While the negatives provide some disappointing truths – and all projects have them – it is important that they are used for good. In doing this we can take a negative and turn it into a positive. The best means of doing this is making recommendations for the future development of the project. The developer is happy with the end result, but feels that improvements could certainly be made to the system to make it more robust and therefore more valuable to the end user. In order to achieve this, the follow recommendations are made.

- Further development of a statistical analysis tool for the application would improve its overall appeal to a typical end user. As mentioned earlier, the

developer was unable to implement this feature in the time frame permitted, however if implemented into a future version of the application, its value would increase.

- As was also mentioned, the inclusion of an interactive feature to allow for team selection would make the application more appealing to an end user.
- It is intended that future development of the application will include a login feature which will allow the user to access their team information securely.
- A final feature which would improve the viability of the application as a economical success would be the ability for the user to manage several teams, for example if the user had a main team but also a reserves team or an under 18s team, the ability to manage them as separate entities would be desirable.

These points, if implemented into a future version of the system, would complete what the developer would describe as the perfect tool for ice hockey club management.

## 5.4 Summary

The project as a whole has been a learning experience for the developer. It has been the first solo project they have completed as a software developer, and so has given them both the confidence to continue and the assurance that as more projects are undertaken their ability to create viable technological solutions to problems will improve. Furthermore, the developer is content that they have created a fairly decent web application on their first attempt. As mentioned above, certain improvement can surely be made in the future, but as a simple tool for ice hockey club management, the current version does its job.

The main thing the developer will take from this experience is that software development is often much more intensive than it may appear, but it is just as rewarding. This dissertation has followed the progression of this project from an initial idea to a final application, and the developer is proud of what they have achieved.

# References

Adzic, G. (2007). *Blinded by the user interface.* Available: https://gojko.net/2007/01/31/blinded-by-the-user-interface/. Last accessed 10th August 2016.

Baykov, A. (2013). *45th Anniversary of the Graphical User Interface.* Available: https://uxpa.org/article/45th-anniversary-graphical-user-interface. Last accessed 14th August 2016.

Capterra. (2016). *Sports League Software.* Available: http://www.capterra.com/sports-league-software/. Last accessed 21st July 2016.

Coding Horror (2008) Available from: http://blog.codinghorror.com/understanding-model-view-controller/. Last Accessed 20th January 2016.

Coyier, C. (2015). *The State of Front-End Development.* Available: http://alistapart.com/event/front-end-dev. Last accessed 26th August 2016.

CVent. (2009). *Guide to the Five Types of Survey Questions.* Available: http://survey.cvent.com/blog/cvent-survey-blog/guide-to-the-five-types-of-survey-questions. Last accessed 24th July 2016.

Duggan, T. (2015). *Why Is Risk Management Important to Project Success?.* Available: http://smallbusiness.chron.com/risk-management-important-project-success-56920.html. Last accessed 2nd August 2016.

Enfocus Solutions. (2011). *Stakeholder Role in the Software Development Lifecycle.* Available: http://enfocussolutions.com/stakeholder-role-in-the-software-development-lifecycle/. Last accessed 2nd August 2016.

ETHW. (2016). *Tracking the Ice Hockey Puck.* Available: http://ethw.org/Tracking_the_Ice_Hockey_Puck_-_FoxTrax_(Glow_Puck). Last accessed 21st July 2016.

EZ Facility. (2016). *Features.* Available: https://www.ezfacility.com/features. Last accessed 21st July 2016.

Fielding, RT. (2000). *Architectural Styles and the Design of Network-based Software Architectures.* Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. Last accessed 12th August 2016.

Gube, J. (2013). *Why Prototyping is Essential to Your Design Process.* Available: http://designinstruct.com/web-design/prototyping-is-essential/. Last accessed 22nd August 2016.

Hallman, B. (2011). *10 Key Success Factors For Application Implementation Projects.* Available: https://www.projecttimes.com/articles/10-key-success-factors-for-application-implementation-projects.html. Last accessed 23rd August 2016.

Hughey, D. (2008). *The Traditional Waterfall Method.* Available: http://www.umsl.edu/~hugheyd/is6840/waterfall.html. Last accessed 22nd July 2016.

ITSQ. (2016). *Software Testing Objectives.* Available: http://istqbexamcertification.com/what-is-the-software-testing-objectives-and-purpose/. Last accessed 22nd July 2016.

Johnston, M. (2014). *The Importance of Testing Software Before Launch.* Available: https://www.utest.com/articles/the-importance-of-testing-software-before-launch. Last accessed 26th August 2016.

Lea, D. (1995). *The Analysis Process.* Available: http://gee.cs.oswego.edu/dl/oosdw3/ch12.html. Last accessed 22nd July 2016.

Manage Your League. (2016). *Features.* Available: http://manageyourleague.com/MYL/features. Last accessed 21st July 2016.

Medeiros, J. (2014). *The Winning Formula.* Available: http://www.wired.co.uk/article/the-winning-formula. Last accessed 21st July 2016.

Mountain Goat Software. (2016). *User Stories.* Available: https://www.mountaingoatsoftware.com/agile/user-stories. Last accessed 24th July 2016.

NetBeans (2015) Available from: https://netbeans.org/kb/docs/javaee/ecommerce/design.html. Last Accessed 20th January 2016.

Netbeans. (2016). *Features.* Available: https://netbeans.org/features/index.html. Last accessed 14th August 2016.

Oracle. (2016). *Serialization.* Available: https://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html. Last accessed 24th August 2016.

Roberts, S. (2016). *EIHC History.* Available: http://eiha.co.uk/history/. Last accessed 20th July 2016.

Robertson, J & Robertson S. (2012) Volere Requirements Specification Template. Available from: https://www.cs.uic.edu/~i440/VolereTemplate.pdf. Last Accessed 18th January 2016.

Royce, W. (2000). *Managing the development of large software systems.* Available: https://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf. Last accessed 10th August 2016.

SAFE (2015) Available from: http://scaledagileframework.com/nonfunctional-requirements/

Sky Sports. (2013). *MNF Makeover.* Available: http://www.skysports.com/football/news/11096/8878157/mnf-makeover. Last accessed 21st July 2016.

Sportlyzer. (2016). *Features.* Available: https://www.sportlyzer.com. Last accessed 21st July 2016.

Statistica. (2014). *Statistics and facts on the National Hockey League (NHL).* Available: https://www.statista.com/topics/960/national-hockey-league/. Last accessed 20th July 2016.

Team Sports Admin. (2016). *Features.* Available: http://www.teamsportsadmin.com/. Last accessed 21 July 2016.

University of North Carolina. (2012). *System Requirements Specification.* Available: http://www.unc.edu/~stotts/comp523/USDA-funcSpecs.pdf. Last accessed 24th July 2016.

University of Surrey. (2010). *The advantages and disadvantages of questionnaires.* Available: http://libweb.surrey.ac.uk/library/skills/Introduction%20to%20Research%20and%20Managing%20Information%20Leicester/page_51.htm. Last accessed 23rd July 2016.

Venners, B. (2008) The Importance of Model-View Separation: A conversation with Terence Parr. Available from: http://www.artima.com/lejava/articles/stringtemplate.html Last Accessed January 20th 2016.

Walonick. (1993). *Questionnaires.* Available: http://www.pgce.soton.ac.uk/IT/Research/Doctorate/Questionnaires/Walonick.pdf. Last accessed 23rd July 2016.

Wiegers, K. (1999). *First Things First: Prioritizing Requirements.* Available: http://www.processimpact.com/articles/prioritizing.html. Last accessed 12th August 2016.

Williams, G, Smith, C. (2003). *Making the Business Case for Software Performance Engineering.* Available: http://www.perfeng.com/papers/buscase.pdf. Last accessed 2nd August 2016.

Wodehouse, C. (2016). *The Role of the Back-End Web Developer.* Available: https://www.upwork.com/hiring/development/back-end-web-developer/. Last accessed 23rd August 2016.

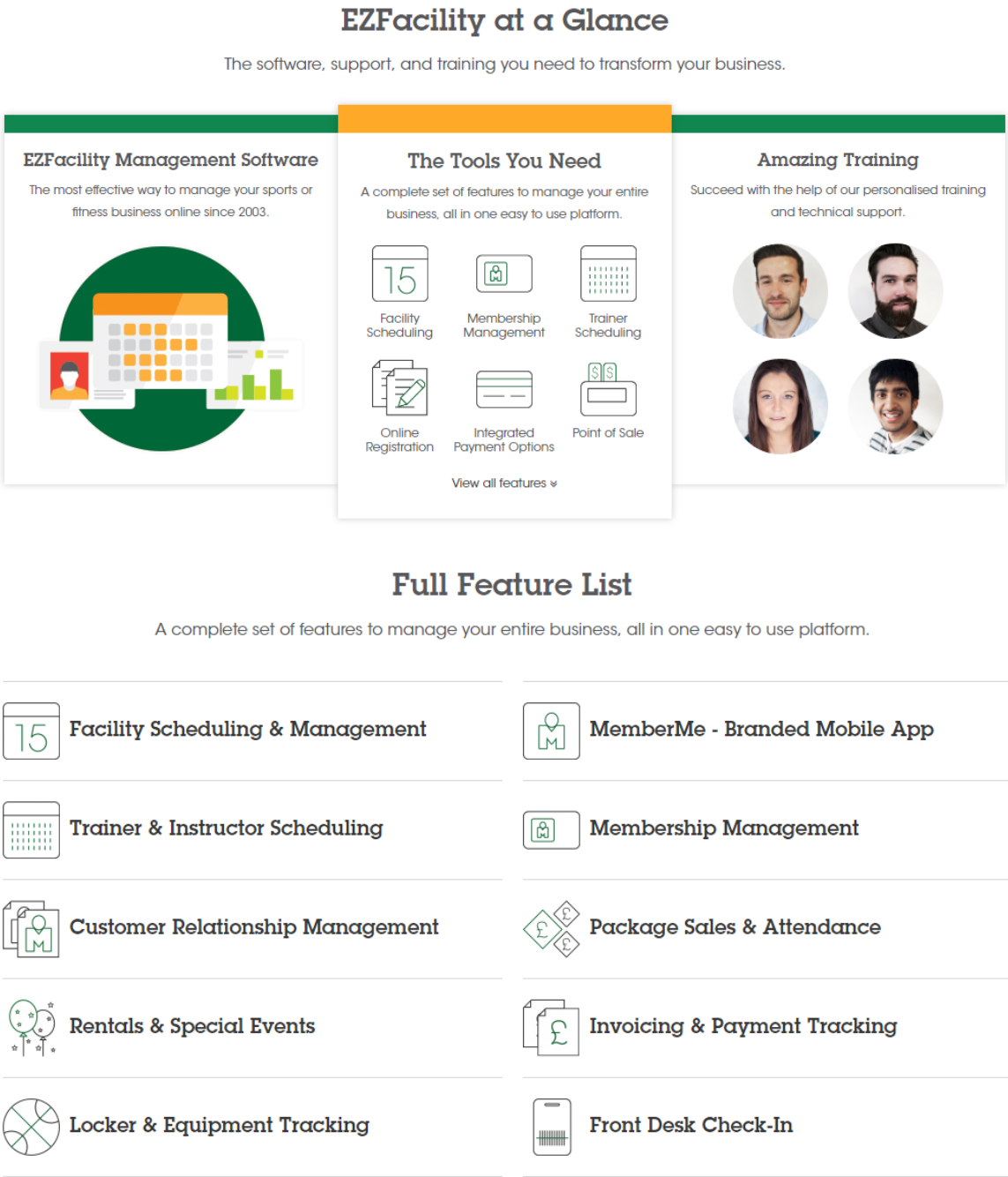
Zint, M. (2015). *Evaluation: What is it and why do it?* Available: http://meera.snre.umich.edu/evaluation-what-it-and-why-do-it. Last accessed 27th August 2016.

Zuzak, I. (2010). *Why understanding REST is hard and what we should do about it.* Available: http://ivanzuzak.info/2010/04/03/why-understanding-rest-is-hard-and-what-we-should-do-about-it-systematization-models-and-terminology-for-rest.html. Last accessed 24th August 2016.

# Appendices

## A1. Current Applications Available

**EZFacility:**



**Sportlyzer**


You will love it
Mark attendance on your phone
Mark athletes' attendance at a practice using your phone or tablet. No internet connection needed.
Always know in advance who's coming
Send out invitations to know for sure who can or can't make it to a game, practice, or a season end party.
Share schedules with your team
Enter your schedules once and they will be shared with everyone in your team - via email or athlete apps. Athletes don't even have to create accounts.
You will also get
AS A COACH
AS A CLUB
Athletes database
Handle all your athletes, contacts, and team information in one place. Everything is accessible 24/7 on web or on your mobile devices.
Automated group email & SMS
Notify your athletes or parents about any changes in training times, about urgent meetings, etc. Automatic notifications can be sent to shared schedules, track who's coming to a workout, etc.
Reports
In case you want to analyze attendance in a specific period or need to send your athletes list to a third party, you can easily export the data you need.
Mobile apps
Use Sportlyzer on your phone and leave your notebook at home. Our apps don't need constant internet connection and therefore can be used at training facilities with poor network coverage.

**Manage Your Team**


**Team Sports Admin**

## Save Time and Money


*"I am SO loving using your system and only kick myself that we didn't find you sooner! I'm going to recommend you to our soccer club. They just went online this year with the system we were using, but I know they would find this one so much better! Over 2500 kids are part of the club, and it's growing.*

*Roxane Lee*
*Vancouver Sports Club*

*No more searching through piles of paper. No more wasting time having to check everything by hand. No more boxes and boxes of old signed documents and forms gathering dust. You'll have everything you need to run your organization in one place using our easy-to-use online sports registration software we call Team Sports Admin.*

*Plus, we offer the best pricing in the industry - whether you're a national organization with tens of thousands of kids, or a small league with a few teams - you'll be saving time and money.*

## Simplify Your Registration Process and Stay Organized

**Parents or Guardians Registering Their Player(s) for Your Event Only Takes Minutes**

### 1. Register

*Players and/or Guardians quickly fill out their information online using your custom registration link.*

### 2. Sign Legal Documents

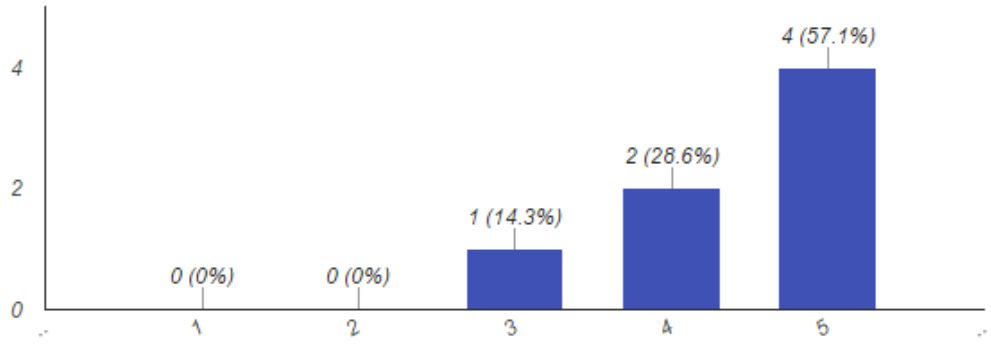
*Any required documents are instantly signed electronically using DocuSign™, the global standard for online signatures.*

### 3. Pay Securely

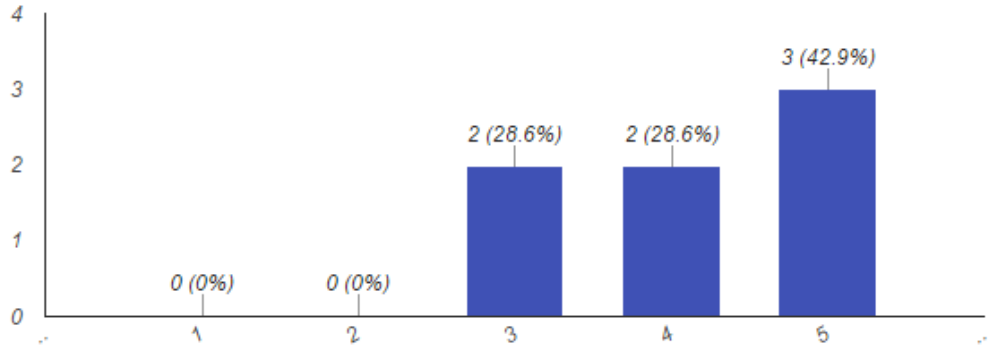
*Any payments needed can be instantly paid online using our 100% secure online payment processor.*

## A2. Questionnaire Analysis

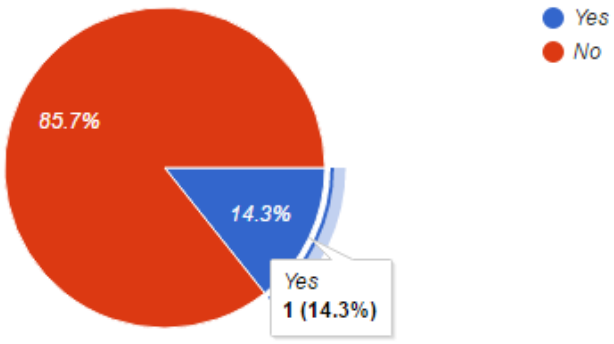
Current "paper" team management could be improved upon. (7 responses)



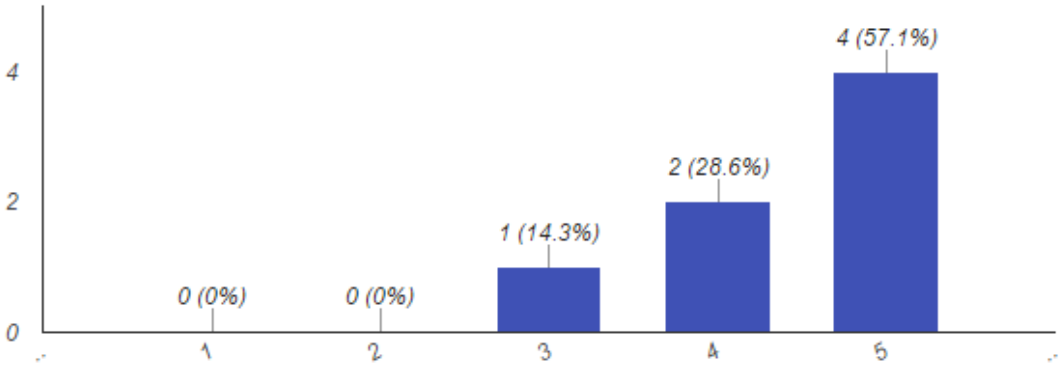

A web application could improve team management. (7 responses)



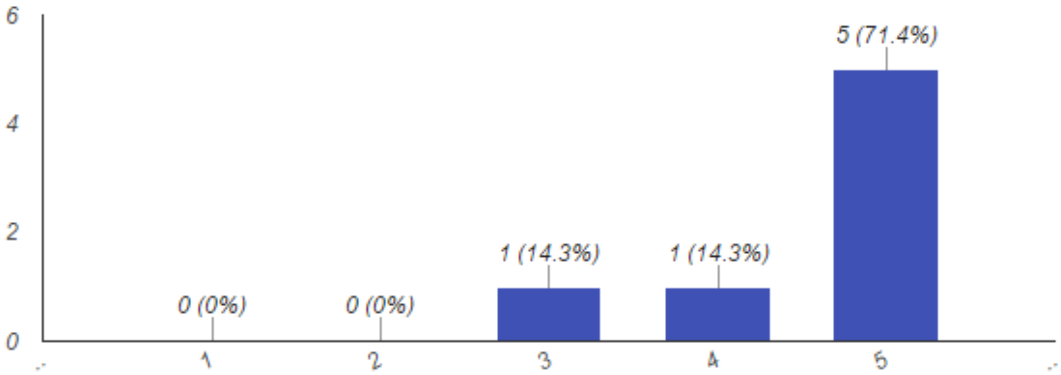

I am aware of existing applications which aid team management. (7 responses)



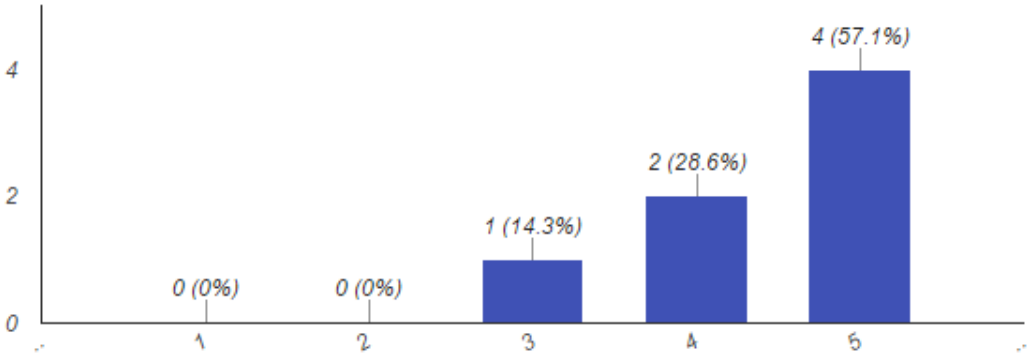
The ability to add players and maintain their information is important.
(7 responses)



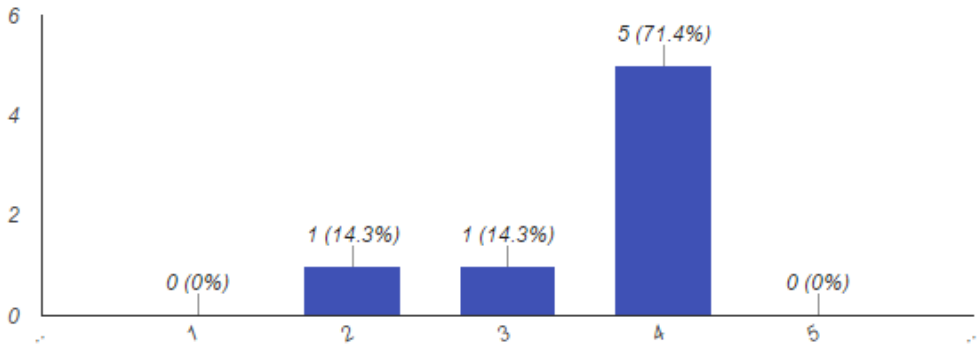

The ability to add fixtures and maintain their information is important.
(7 responses)



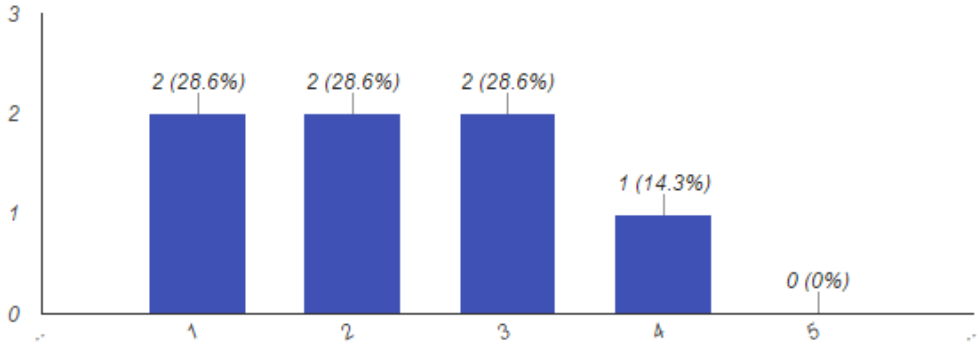

The ability to add players and maintain their information is important.
(7 responses)



A web application would make it easier to manage a sports team. (7 responses)




An ice hockey management system may encourage new teams to form.
(7 responses)



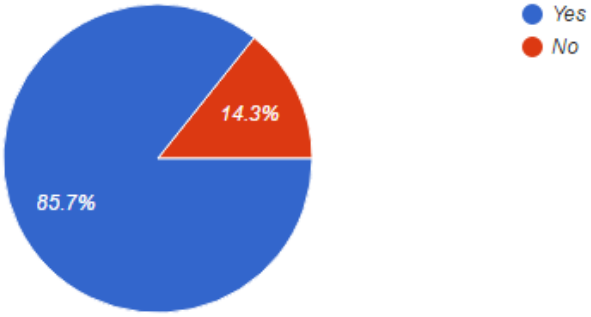

What in particular do you feel an ice hockey management system should include?
(3 responses)

| |
|---|
| add the score to a match already played, keep track of upcoming matches |
| add my goals scored and and when the next fixture is |
| team selection |

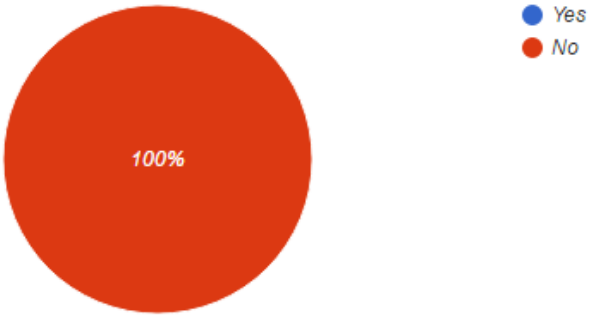
What should an ice hockey management system not include? (2 responses)

| |
|---|
| personal contact information |
| hidden fees |

Would you be willing to pay a monthly subscription fee to use an ice hockey management system?
(7 responses)




Would you be willing to pay a higher fee to own an ice hockey management system outright?
(7 responses)



## A3. Test Suite

| Test # | Item Tested | Worked as expected? | Notes |
|---|---|---|---|
| **1** | Application running on Google Chrome. | Yes. | All proportions correct and elements in place. |
| **2** | Application running on Internet Explorer. | Yes. | All proportions correct and elements in place. |
| **3** | Application running on Microsoft Edge. | Yes. | All proportions correct and elements in place. |
| **4** | Application running on Mozilla Firefox. | Yes. | All proportions correct and elements in place. |
| **5** | Application running on Apple Safari. | Yes. | All proportions correct and elements in place. |
| **6** | Image on homepage directing user to management pages. | Yes. | |
| **7** | Players button on menu directing to Players page. | Yes. | |
| **8** | Fixtures button on menu directing to Fixtures page. | Yes. | |
| **9** | Staff button on menu directing to Staff page. | Yes. | |
| **10** | Banner directing to homepage when clicked. | No. | Remedied, tested again and found to be working. |
| **11** | Twitter feed appearing on homepage. | Yes. | |
| **12** | Table of Players appearing on load of Players page. | Yes. | Table loaded correctly displaying players. |
| **13** | Create button displaying add a player form. | Yes. | |

| | | | |
|---|---|---|---|
| **14** | Player being added to the table following the form being filled out and save being clicked. | Yes. | |
| **15** | Player's details being displayed when their ID is clicked. | Yes. | |
| **16** | Selected player's information being updated when altered and save clicked. | Yes. | |
| **17** | Player being deleted from the table when selected and delete clicked. | Yes. | |
| **18** | Table of Fixtures appearing on load of Fixtures page. | Yes. | Table loaded correctly displaying fixtures. |
| **19** | Create button displaying add a fixture form. | Yes. | |
| **20** | Fixture being added to the table following the form being filled out and save being clicked. | Yes. | |
| **21** | Fixture's details being displayed when their ID is clicked. | Yes. | |
| **22** | Selected fixture's information being updated when altered and save clicked. | Yes. | |
| **23** | Fixture being deleted from the table when selected and delete clicked. | Yes. | |
| **24** | Table of Staff appearing on load of Staff page. | No. | Error in the HTML file, stylesheet not referenced. Remedied, tested again and worked. |

| | | | |
|---|---|---|---|
| **25** | Create button displaying add staff form. | Yes. | |
| **26** | Staff member being added to the table following the form being filled out and save being clicked. | Yes. | |
| **27** | Staff details being displayed when their ID is clicked. | Yes. | |
| **28** | Selected staff member's information being updated when altered and save clicked. | Yes. | |
| **29** | Staff Member being deleted from the table when selected and delete clicked. | Yes. | |