Management Science and Information Systems Studies

Final Year Project Report

Health Check System: Data Capturing Application for IRFU Rugby Department

Alan O’Neill

March 2015
Declaration

I declare that the work described in this dissertation has been carried out in full compliance with the ethical research requirements of the School of Computer Science and Statistics.

Signed: ___________________
Alan O’Neill 20th March 2015
ABSTRACT

The aim of this project was to produce a bespoke web application for the Irish Rugby Football Union (IRFU) Rugby Department to enable periodic capture of club data. The complete application consists of three components: the Administration Component, the User Component and the Application Database Component. It was composed using MySQL, PHP, HTML, CSS and JavaScript. The application fully meets the client's requirements while maintaining the adaptability for future extension. In keeping with the Rugby Department’s objective a comprehensive health check reports on club, provincial and national levels the application was integrated with a Health Check Reporting Application.
PREFACE

The Irish Rugby Football Union (IRFU) is the managing body of rugby union in Ireland (both Republic of Ireland and Northern Ireland). The IRFU Rugby Department is responsible for the development of all aspects of the game at both professional and club/amateur levels. The IRFU Rugby Department will hereby be referred to as ‘the client’. The client required an application that would allow the capture of data relating to off field elements of each club in Ireland and would facilitate an integrated Health Check Reporting Application to generate automated reports based on this data.

The final application surpasses the terms of reference as specified by the project’s terms of reference. It remains adaptable to accommodate future changes in the requirements of the client.

The main issues involved in the project arose from the scope and complexity of the system and the need for integration with an application being concurrently developed by Scott Burrell.

The complete application required several hundred hours of development and was fully completed and integrated by the project deadline. The application is composed of five markup, scripting and database management languages.

I would like to include thanks to Scott Walker of the IRFU, Phillip Scott of the Leinster Branch, Ultan O’Callaghan of the Munster Branch, Claudia Carr of BearingPoint and Scott Burrell of Trinity College Dublin for their help and enthusiasm throughout the project. I would also like to thank Denise Leahy, the project supervisor, for her guidance and assistance.
# TABLE OF CONTENTS

1 INTRODUCTION AND SUMMARY ........................................... 6
   1.1 Introduction .............................................................. 6
   1.2 The Client ............................................................... 6
   1.3 The Project Background .............................................. 6
   1.4 Terms of Reference .................................................... 8
   1.5 Summary ................................................................. 9

2 SYSTEM OVERVIEW .......................................................... 10
   2.1 System Objectives ....................................................... 10
   2.2 Application Description .............................................. 11
   1.3 System Overview Diagram ........................................... 13
   1.4 Technical Environment .............................................. 15

3 DESCRIPTION OF WORK DONE .......................................... 17
   3.1 Requirements Analysis ................................................ 17
   3.2 Solution Investigation ................................................ 18
   3.3 Architect ................................................................. 20

4 Design-Build-Deploy ....................................................... 23
   4.2 Integration ............................................................... 26
   4.3 Form Design ............................................................ 26

5 CONCLUSIONS AND RECOMMENDATIONS ............................ 29
   5.1 Conclusions ............................................................. 30
   5.2 Recommendations ...................................................... 30
A INTRODUCTION AND SUMMARY

A.1 Introduction
The aim of this project is to investigate the client’s core data capturing requirements and the objectives driving them; to examine the approaches currently employed and develop an application to better facilitate the achievement of these objectives.

A Health Check Data Capturing Application was developed and integrated with a Health Check Reporting Application to form a complete Health Check system for the client. This system fully meets the client’s requirements and was completed within the specified project timeline.

This chapter describes the client, project background and terms of reference of the project as well as a summary of remaining chapters in the report.

A.2 The Client
The Irish Rugby Football Union (IRFU) is the governing body for rugby union in Ireland. The client for this project is the IRFU Rugby Department. The IRFU Rugby Department is responsible for the development of all aspects of the game at both professional and club/amateur levels.

A.3 The Project Background
A key objective for the Rugby Department is to increase the level of participation in all facets of rugby throughout the country. The aim is to increase the level of participation through creating opportunities for men, women and children, of all ages and abilities, to become involved in rugby and to retain players and members through providing a quality rugby experience. The expectation of the department is that increasing the overall level of participation will widen the pool of players available for selection to regional and provincial squads for elite player development, thus improving the nation’s ability to compete at a national level.

In alignment with the IRFU Strategic Plan (2013-2017) resources are allocated to the Provincial Branches to implement a delivery plan to achieve this objective. These resources include a team of managers and rugby officers who work closely with clubs, schools and local authorities to develop the appropriate infrastructure, support services and training required to achieve this objective.

To support the Provincial Branches in this role it is a principal requirement that their staff have access to up to date and accurate information on a number of on field and off field elements for each club in the country. By monitoring changing trends in these elements rugby officers can proactively identify needs and opportunities at provincial, regional and club levels.

The aim of the client is to develop a comprehensive Health Check system that will allow them to assess the state of Irish rugby at club, regional, provincial and national levels. This
system will then be used to assist in achieving the client’s aforementioned objective. Currently, this system is partially in effect and is referred to as the Club Health Check.

Information for the Club Health Check is pooled from a collection of both internal and external systems. External systems provide information regarding on field elements such as on games played, the number of coaches and the number of players in each club. Internal systems provide information on off field elements such as playing and training programs run. Information on these elements are compiled on a monthly basis in excel spreadsheets which form the Club Health Check.

There are several problems with the Club Health Check system:

- **Complexity**: There a several separate sources of information for the system including complex internal systems and external third-party systems.
- **Reliability**: Information from external systems must be handled manually as excel or .csv files. This can cause data integrity issues resulting from human error.
- **Time-Consumption**: Information from internal systems must be compiled and condensed a number of times from club level to regional level to provincial level to national level on a monthly basis. This process can be a heavily time consuming and can also result in the loss of information.
- **Inadequacy**: There are a number of off field elements contributing to the health of clubs that currently fall outside the existing systems and have no means of data capture.
- **Subjectivity**: Currently the data for a number of club elements is subjectively based on the opinion of rugby officers. This can result in tension between the client, its officers and the clubs they report on.

The aims of this project are:

- To develop forms that can be used to gather information on off field elements of clubs for which there is currently no means of data capture.
- To provide an application that will allow these forms to be used to capture of data relating to the off field elements periodically.
- To integrate this system with another system that is being developed concurrently to provide the client with a Health Check system that will allow them to comprehensively assess the state of Irish rugby at club, regional, provincial and national levels.

The off field elements that the application will be used to gather data on are:

- Finance
- Child Welfare
- Medical
- Volunteers
- Branch Engagement
- Project Clubhouse
- LSP/Council
- Club/School Link

A detailed breakdown of the on field and off field Health Check elements and their sources is included in Appendix C.4.
A.4 Terms of Reference

The terms of reference agreed with the client are:

- To investigate the client’s aims through meetings and workshops with the client.
- To research possible solutions.
- To design and develop forms to capture relevant data on off field elements of a club, for which data are not currently gathered.
- To design and develop an online application from which these forms can be accessed and that is capable of gathering and storing response data for them.
- To integrate the application with a Health Check reporting application being developed concurrently by Scott Burrell of Trinity College Dublin with the aim of providing the client with a complete online Health Check system.
- To develop extensive technical and user documentation.

Additionally, the client requires that a level of adaptability be built into the application to reflect the continuously changing and developing environment of rugby. The project has exceeded the original terms of reference by including in the application:

- A secure online application consisting of a user-friendly interface allowing users to login securely under varying levels of accessibility.
- The ability for users to view, edit and delete their own form submissions within time constraints designated by the client.
- A user administration module that allows administrator-level users to create, view, update and delete users; to view user activity on the portal and to edit user access to forms and reports.
- Levels of user access that automatically determine a users access to content on the application; specifically the level of Health Check a user can view, the forms a user can access and the scope of clubs for which form submissions are viewable.
- The ability to easily search through past form submissions, users using dynamic paginated tables with search and sort capabilities.
- A level of adaptability that enables future functionality to be added easily to the application.

These terms of reference are the final terms of reference agreed with the client. During the early phases of the project opportunities arose to more comprehensively achieve the client’s objective. These opportunities were identified through discussion with the client as well as examination of existing systems and ongoing projects. This allowed for the development of the initial terms of reference through further discussion with the client. The development of the terms of reference is described further in Chapter 3. The original terms of reference are included in Appendix B.
A.5 Summary

A.5.1 Chapter 2:
Chapter 2 describes the purpose and objectives of the application; gives a general overview of its capabilities, describes how it operates and defines the technical environment.

A.5.2 Chapter 3:
Chapter 3 provides a description of the software development life cycle (SDLC) model used and the breakdown of work done in relation to the application development throughout each of its phases. Chapter 3 also provides a breakdown of the work done in relation to the off field element form development and a note on requirement change management relating to aims of this project.

A.5.3 Chapter 4:
Chapter 4 discusses conclusions and recommendations for implementation, general use and further development.
B SYSTEM OVERVIEW
This chapter describes the purpose and objectives of the application; gives an overview of its capabilities; describes how it operates and briefly defines the technical environment.

2.1 System Objectives
The purpose of the overall integrated system is to provide the IRFU Rugby Department officers with a complete Health Check System that will allow them to assess clubs and regions in each of the four provinces. The Rugby Department officers will use the Health Check reports to gain insight into the state of each club’s operations both on field and off field. These reports will be used to:

- Highlight regions and clubs in which to focus their efforts
- Highlight specific elements of a club that may require attention
- Provide officers with objective data on clubs and regions rather than necessitating reliance on subjective data.
- Guide interactions between officers and clubs towards a more beneficial outcome.
- Provide a trigger to both Rugby Department officers and club volunteers to question the state of each pertinent aspect of a club.

Having these reports will allow the client to assist in improving the state of clubs across the country. This is central to the client’s core objective of increasing participation in rugby and preventing the disillusionment of volunteers.

This overall integrated Health Check system consists of this data capturing application and the reporting application being developed concurrently by Scott Burrell of Trinity College Dublin as described in the terms of reference in Section 1.4.

This application provides a data capturing facility that will allow users to periodically input up to date information on each clubs’ off field elements. Previously, the Club Health Check reports were incomplete as there was no data capturing facility for several of the off field elements.

The application has been designed to be adaptable and extensible. The elements that constitute the Health Check reports are expected to change over the long term in order to mirror changes in the IRFU’s strategic goals. This required the system to be capable of accepting further modules in future to allow the client to add and edit Health Check elements and forms and to customise reports accordingly. The manner in which the system is designed meets this requirement.
2.2 Application Description

The application consists of three main components: the User Component, the Application Database and the Integrated Health Check Report Component.

B.1.1 The User Component

The User Component consists of two core modules: the Manage Users Module and the Health Check Forms Module. This subsection will describe each of these modules as well as describing the factors affecting the logical flow and capabilities of the User Component. A more detailed breakdown of the capabilities of each module is included in Appendix C.2.

Manage Users Module

From the Manage Users Module the user can create, manage and monitor users and their capabilities. This module allows users to:

- Create a user
- View all users’ details
- View individual user details
- Update a user’s details
- Manage user access
- Delete a user
- Monitor user activity

Health Check Forms Module

The Health Check Forms Module displays each of the Health Check forms available and allows the user to manage Health Check form submissions. From this module the user can:

- View all available off field Health Check forms
- Submit a response to a Health Check form
- View past Health Check form submissions
- Update past Health Check form submissions
- Delete past Health Check form submissions

To use the application users must log into the User Component. The logical flow of the application and the availability of the user interactions listed above are dependent on the details of the user logging in. There are 5 user attributes that determine the User Component’s logical flow and capabilities: the user’s Level; their User’s Characteristic; their Access to Form Module; their Access to Report Module and their Individual Form Access.

The User’s Level and Characteristic:

As discussed in Section 1.3, this application (and the overall Health Check system) maintains a 5-tier user level. The capabilities of a user depend on their designated level. The Manage Users Module is available only to users at Administrator Level. In this way, the abilities of this module can be restricted to a small number of super users at this level. From the Health Check Forms Module, users at Administrator Level have access to the past submissions of every user relating to every club in the country. Users at this level can switch to viewing past submissions for any club using a set of province, region and club dropdowns. Users at this level also have the ability to edit or delete any users’ past submissions with no time limits as well as submitting form entries themselves.
National Level users have more restricted access. They cannot access the Manage Users Module. They also have the capability to access the past submissions of every user relating to every club in the country; to switch to viewing past submissions for any club using a set of province, region and club dropdowns and to submit form entries themselves. More restrictively, users at this level only have to ability to edit or delete their own past submissions within 30 days of its original entry.

Users at the remaining three user levels: Provincial, Regional and Club, are only more restrictive than National Level in that users at Provincial Level are restricted to viewing past submissions and submitting forms for clubs within the regions in the province set as their User Characteristic; users at Regional Level are restricted to clubs within the region set as their User Characteristic and users at Club Level are restricted to just the club set as their User Characteristic.

Access to Form Module and Access to Report Module:
Each user’s access to the entire Health Check Forms Module can be restricted allowing users at any level to be given access or denied access to the module. Each user’s access to the entire Health Check Report Module can also be similarly restricted. This allows for user accessibility to be broadly managed when necessary.

Individual Form Access:
Users at any level can be given access or denied access to each individual form. This allows for users to have access only to the forms relevant to them. Once given access to the form module a user only has access to the forms that they should be able to access.

B.1.2 The Application Database
The Application Database is a MySQL Server (version 5.5.34) database. The database consists of 25 tables and can be divided into the following three layers: the User Layer, the Health Check Form Layer and the Response Layer. A detailed database breakdown accompanied by an ER diagram is included in Appendix C.1.1.

The structure of the database tables is crucial to the functionality, adaptability and extensibility of the application. When designing each database table’s structure, a common design methodology was used. In each case the aim was to reduce the required tables to 5NF and then introduce a level of redundancy in the form of an auto-incrementing primary-key index column on each table.

Reducing the table to 5NF increased the adaptability of the application while optimising the database performance and memory. Introducing redundancy in the form of indexes improves the capabilities of the applications PHP scripting when interacting with the database. This also increases the speed and simplicity of the application’s PHP scripting as the majority of the database querying is done on these single indexed columns.
B.1.3 Integrated Health Check Report Component

The integrated Health Check Report Component refers to the aspect of the application that has been developed to integrate this application with the Health Check Reporting application. This component consists of 8 PHP scripts, one for each of the current off field Health Check elements, stored in separate files. These scripts all follow the same structure and logical flow described subsequently.

The scripts load the most recently submitted responses of any form relating to the relevant off field Health Check element for a given club. The submission date of the responses is tested to ensure that they fall within the valid submission deadline constraints for each form. These constraints are described in Appendix C.4. If invalid, Health check score of 0 and score breakdown descriptions of ‘No relevant submission’ are passed to the Health Check Reporting application. No changes are made to the Health Check Reporting application database in the case of invalid submissions.

If the submissions are within the correct deadline constraints, the scripts calculate the club’s score for that element and pass this score, along with the breakdown descriptions for each factor contributing to the score to the Health Check Reporting application. A more detailed description of the breakdown descriptions is available in Appendix C.4. The scripts then access database tables in the Health Check Reporting application database and insert the most recent score and the breakdown descriptions into the relevant elements designated tables. For a more detailed description of these PHP scripts see Appendix C.2.

1.3 System Overview Diagram

Figure 1.3.1 shows an overview diagram of the final integrated Health Check system. For applications in the web environment, there are two main entities: the client and the server. Operations run by the client are described as client-side operations and those run by the server are described as server-side operations. The client, in this case, refers to the users computers. The server in this case is a web-hosted server. This is discussed further in Section 5.2.4. The client displays information to users via a web browser (such as Google Chrome, Internet Explorer etc.). User interaction with the web browser causes the client to request information from the server. The server processes this request and returns the required information to the client, which displays it to the user through the web browser.

The diagram illustrates what each of the main components of the overall Health Check system are composed of and how they interact with each other and with the user. Every user request runs through the PHP controller, which controls the logical flow of the application as well as the overall system. It also verifies the user every time a request is received.
Figure 1.3.1: System Overview Diagram. The *Application Database* is shown at the bottom of the diagram; the two red-shaded areas on the left and in the middle represent the *User Component* and the red-shaded area on the right represents the *Integrated Health Check Report Component*. For the sake of simplicity, smaller function files and the majority of the PHP classes have been left out although they play a major role in the system.
The initial landing page is the Home Page. From here, the user can request access to the either of the User Component modules or the integrated Health Check Reporting application. Access the Manage Users Module landing page is only allowed if the user’s level is verified as administrator. From here, the 5 pages composing the module can be requested and returned to the user. Through these pages the user can view data from the database. The corresponding yellow functions represent the actions that can be taken by the user on each page.

From the Home Page, the user can also request access the Health Check Forms Module. Access to the Health Check Forms Module landing page is only allowed if the user has been designated access to it. From here, the 3 pages composing the module can be requested and returned to the user. The accompanying functions are required load the forms and past submissions for each page and to allow the user to take further actions such as submitting a form response.

The core functionality of the Health Check Forms Module and of the overall application is the getForm function. This function is central to the adaptability and extensibility of the application. It consists of over 1600 lines of code and gives the application the capability to load any form stored in the application database, current or future. From the Home Page the user can also access the Integrated Health Check Reporting application. Again, the user must be designated access to the module to access it.

The diagram also demonstrates the flow of information between each of the application modules and the Application Database. The application’s functions and pages are capable of requesting and receiving data from the database as well as requesting and passing data between themselves and others in the same module.

For simplification, several aspects of the overall system relating this application we excluded from the diagram. The application’s PHP classes have all been excluded, with the exception the user class when, in reality, they are used for every external interaction between the application and the user as well as every internal interaction between application components.

### 1.4 Technical Environment

The system was created using several programming languages and an accompanying database. The database used is a MySQL Server (version 5.5.34) database. The combination of a server-side scripting language, a database management system and multiple client-side scripting languages and markup languages were used in creating the application.

The server-side scripting language used was PHP. Server-side scripting refers to processes or operations enacted on the server (as described in Section 1.3). A custom framework was used in programming the server-side scripting of the application.
MySQL is a relational database management system. It is used to handle every interaction with the application database. Most of the operations performed by the application require interaction with the database. MySQL operations are enacted on the server-side; they are embedded in PHP functions in order for database interactions to be included in the logical flow of the application.

The client-side scripting languages used are JavaScript and JQuery. Client-side scripting refers to process or operations enacted on the client (as described in Section 1.3). As an example, JQuery functions are enacted on the client when a form requiring a date response is loaded; they are used to convert the standard input box into a datepicker, which is a visual calendar input.

Hypertext Markup Language (HTML) is used to display the information past to the client by the application. Cascading Style Sheets (CSS) is used to format the HTML to present the application information as a user-friendly interface.

Bootstrap 3 is a HTML, CSS and Javascript framework. It is used as a basis for structuring and styling the information presented by the client.
C DESCRIPTION OF WORK DONE

This chapter outlines the work carried out throughout the duration of the project. The work carried out was divided into two tasks undertaken concurrently: application development and form design. This chapter first discusses work carried out in relation to the software development task and then work carried out in relation to form design.

Throughout each phase of the application development and form design tasks multiple meetings, discussions, and workshops with the client were held. These meetings were attended by a combination of the project owner Scott Walk and the project sponsor Phillip Lawlor as well as a number of other project stakeholders including Scott Burrell of Trinity College Dublin; Claudia Carr of BearingPoint; Ultan O’Callaghan, Munster DRM; Chris Webster, Ulster DRM; Eric Elwood, Connacht DRM, and a number of other IRFU Rugby Department officers.

The software development life cycle (SDLC) model used throughout the project was an incremental development model; it consists of five phases:

- Requirement Analysis
- Solution Investigation
- Architect
- Design-Build-Deploy
- Integrate

The SDLC model is explained further in Appendix E.2.1.

C.1 Requirements Analysis

The initial phase of the project was to identify and define the needs of the client. As mentioned in Section 1.2, the original project requirements differ from the final requirements agreed to by the client. This resulted from identifying an opportunity to more effectively achieve the client’s core objective.

This opportunity was identified during the requirement analysis phase of the SDLC. During this phase the client’s requirements were analysed with respect to their core objectives. This involved several meetings and workshops with the client in which their requirements, current systems, and planned projects were discussed at length. The client’s core objective is to increase participation in all facets of rugby across the country (described further in Section 1.2). To achieve this objective the client requires a system capable of facilitating the capture of data, from every rugby club in Ireland, relating to several off field elements of each club.

The requirements analysis phase involved attending requirement workshops with the client, other stakeholders, and Microsoft SharePoint provider SpanishPoint. These workshops highlighted an overlapping project undertaken by SpanishPoint and a concurrent project undertaken by Scott Burrell of Trinity College Dublin. The undertaking of both of these projects provided the opportunity to reform the client’s original requirements. Further advanced discussions with the client led to the formation of the current terms of reference as outlined in Section 1.4.

This phase also resulted in the necessity for the form design task described later in this chapter.
C.2 Solution Investigation

This phase was broken into four segments:

- Examining existing approaches
- Investigation into alternatives
- Evaluation of possible solutions
- Developing the technical environment

The first segment of solution investigation was the examination of the client’s existing data gathering approaches. These are as outlined in Section 1.3. In discussion with the client it was decided that a new approach be used that was more suitable to the client’s needs. The alternatives investigated included extracting data from existing systems, building forms in Microsoft SharePoint and a bespoke web-based application.

The client was concerned that the existing data gathering approaches were slow and troublesome and created a host of administrative and managerial issues. The client was in the process of having a costly long-term business software solution developed using Microsoft SharePoint that had not yet been completed.

It was decided that a bespoke web-application most suited the client’s needs. The application could be available to any user with access to the Internet; form submissions could be stored in a single online database and collected automatically and reports could be accessed as required by each user.

Overall, a bespoke online application meets the data capturing and reporting needs of the client while minimising the effort required by users while also minimising the costs involved. Deploying the application online allows it to be accessible to any user on almost any device across the country. Overall, the client’s needs were of such a specific nature that a bespoke application was deemed the appropriate solution.

The advantages associated with this approach include:

- A bespoke application could be developed to meet all of the client’s needs without excessive additional features increasing size and complexity.
- Hosting the application online allows the application to be accessed by any user with a device connected to the Internet.
- The application could be designed to allow for future adaptability to be added at the behest of the client.
- The client could have the final say over all aspects of functionality.
- Costs could be kept considerably lower than alternative business software packages.

There are also disadvantages associated with this approach:

- Developing a bespoke online application of this size would prove to be a lengthy and complex task involving several programming and markup languages.
- All bespoke applications have associated maintenance issues due to the requirement-specific nature of their development.
- Creep-scope is a common occurrence in bespoke application development and would have to be monitored and managed.
- Integration of two bespoke applications is complex procedure that can be prone to failure.

Overall, the advantages of building a bespoke solution outweighed the disadvantages. This approach would also provide the client with an opportunity to assess the merits of using bespoke solutions in other areas of business.

Following the decision to provide a bespoke online solution the development environment within which the application would be designed, built and deployed was investigated. An overview of the development environment is included in Appendix E.1. There are a number of programming languages and tools available for building online applications. One of the most commonly used server-side languages is PHP. PHP is open source and was deemed most suitable for this application. A further discussion on PHP is continued in Appendix E.

The client requirements called for forms, responses and user details to be stored which necessitated the application utilising some storage method such as XML files, text files or a database. For this application an online database was the logical choice for storing the required data. Databases provide a structured way in which to store and retrieve data. Several database options were investigated throughout this phase however MySQL, an open-source relational database management system, was chosen as the most appropriate solution. Being open-source it keeps the cost of the application down and it is very commonly used for online applications, particularly when used with PHP. The discussion on MySQL is continued in Appendix E.

One of the benefits of using both MySQL and PHP is that they are both open-source and as such are free and have a huge amount of materials, examples and troubleshooting available online. Also, many modern devices come with preinstalled servers capable of running MySQL and PHP, reducing the complexity of the development environment.

For displaying and formatting web pages, HTML and CSS were employed. These are the standard languages used in web development. Every major browser, with the exception of slightly differing behaviors, fully supports both HTML and CSS.

An investigation into which client-side languages would be employed led to the choice of using Javascript, the most widely supported and commonly used. Further research throughout the project highlighted the benefits of also utilising open-source Javascript libraries to simplify some Javascript syntax. The decision was made to incorporate the use of JQuery in some aspects of the application to reduce complexity. Further description of the client-side languages is included in Appendix E.

The final stage of developing the technical environment involved defining the tools that would be used to develop the application. Several integrated development environments (IDEs) and text editors were evaluated during this stage including Eclipse, Microsoft Web
Expression, Dreamweaver and Sublime Text. Eclipse was chosen as the IDE for the project as it is free; it allows for coding in multiple languages it provides a range of more user-friendly features including in-screen access to workspace files.

C.3 Architect
In order to effectively develop the application, a well-defined system architecture was essential. The third phase of the SDLC was to define the architecture of the application. This involved:

- Defining the scope of the application
- Defining the framework that would be used during development
- Developing a timeline for each design-build-deploy sprint
- Assigning features to each build

The scope of the application refers to the list of features the application incorporates. The initial scope of the application was defined using the terms of reference agreed on with the client. This ensured that the client’s fundamental requirements were provided for. At the most basic level, the application needed to be secure; it needed to allow access to off field forms; it needed to allow form responses to be submitted and it needed to be able to pass these submissions to the Health Check Reporting application. During the requirements analysis phase and after each design-build-deploy sprint the features developed were assessed and the application scope reassessed with the client and additional features were included where possible. The importance of avoiding creep-scope was emphasised throughout these reassessments.

The final application scope was defined as follows:

**General Application Scope Information:**

Forms must be able to incorporate the following types of questions:

- Text questions
- Integer questions
- Date questions
- Checkbox questions
- Dropdown questions
- Multiple choice questions

The application must incorporate a 5-tiered user level system. The 5 levels are:

- Club Level
- Regional Level
- Provincial Level
- National Level
- Administrator Level

**General Features:**

The application must have a database to store the following data:

- Health Check elements
- Forms
- Questions
- Responses
- User details
  - Details
  - Level
  - Accessibility

The application must be secure:

- Externally: Application must be able deny access to unauthorised users
  - Users must have to login
  - Users must be able to log out
- Internally: Application must be able to individually restrict users access to specific features and content:
  - Health Check Report application
  - All Health Check forms
• Clubs
• Regions
• Provinces

- Individual health check forms
- Any administrative features such as adding users etc.

**Features: (Manage Users Module)**

The application must incorporate a number of administrative features to ensure it can be managed and maintained. It must allow for administrator level users to:

- Create users
- View users
- Search users
- Sort users
- Update user details (including accessibility)
- Delete users
- View user activity
- Search user activity
- Sort user activity
- Edit individual form access
- Search user form access
- Sort user form access

**Features: (Health Check Forms Module)**

The application must incorporate a number of features to allow form submissions to be managed and maintained. It must:

- View forms (available to each user)
- View an individual form
- Submit a form for particular club
- Restrict form submissions for clubs depending on the user’s level and the club/region/province they are assigned to (no restrictions for national or administrator levels):
  - Users at club level can only submit for the club to which they are assigned
  - Users at regional level can only submit for clubs within their region
  - Users at provincial level can only submit for clubs within the regions in the province to which they are assigned
  - Users at national level can submit for any club in the country
- View past form submissions
- Restrict viewable submissions to the club selected which is restricted as outlined above
- Search past submissions
- Sort past submissions
- Update past submission
- Restrict updating past submissions depending on user level:
  - Users at club, regional and national level can only update their own submissions and only within 30 days of submission
  - Users at administrative level can update anyone’s submission within any time period
• Delete past submissions
• Restrict the deletion of past submission depending on the user level as outlined above
• Users should be able to view their own details (without editing them)

Features: (Health Check Reporting Application)
The application must be fully integrated with the Health Check Reporting application being concurrently developed. Specifically, the application must be capable of:
• Validating the most recent submissions for each off field Health Check element with respect to the deadline constraints of each of the related forms
• Calculating the Health Check score for each off field Health Check element for any given club
• Generating the score contribution breakdown for each element of each club once its score is calculated
• Passing the calculated score and score contribution breakdown to the Health Check Reporting application
• Inserting the calculated score and score contribution breakdown to the Health Check Reporting application database tables

The framework used in developing the server-side scripting of the application is a custom object-orientated framework. The framework was developed subsequent to the database design and creation of the application database. The framework is based on translating database tables that represent real world entities into comprehensive PHP objects before any server-side scripting is undertaken in a design-build-deploy sprint. There are 25 tables in the application database and each is incrementally translated into a PHP object with a host of functions to interact with the database and the logical flow of the application. For client side scripting Bootstrap 3 and JQuery frameworks were chosen to simplify the application development.

Once the initial application scope and the development framework were defined the timeline of the design-build-deploy sprints was mapped. The design-build-deploy phase was scheduled to begin on the Monday 5th January and conclude on Monday 16th March. This allowed for five 2-week design-build-deploy sprints. This timeline was agreed with the client before the phase began.

The final action of this phase involved assigning features from the application scope the five sprints. This was done in an order that allowed for the application to be incrementally developed. For instance, if the deployment one feature relied on the existence of another, the latter would be included in an earlier sprint, or earlier in the same sprint. This allowed for both the newly added working features and the overall application to be assessed by the client and tested after each sprint. A breakdown the features assigned to each sprint is included in Appendix C.5.
D Design-Build-Deploy

This design-build-deploy phase of the project involved incrementally designing, building and deploying the each feature of the application. This phase was split into five sprints, each two weeks long. Each of these sprints was subdivided into a Design Stage, a Build Stage and a Deploy Stage. This allowed for feature development to be more easily managed and made progress easier to track. This approach also made issues affecting development timelines easier to identify and address and contributed to ensuring that all specified application features were finished and complete before the end of the project.

This section describes the design, build and deploy steps that constitute this phase of the project. It then describes the work done during each of the five sprints.

The Design Stage
Effective feature design at the start of each sprint was imperative to ensure that the application as a whole would meet the client’s needs. The first step of Design Stage for each sprint was to model user interactions with the application. Taking the first sprint as an example, at the most basic level, users needed to be able to log into the application and log out again. For this to take place the application would require user details to be stored in the database, a log in and a log out user interface. Unified Modeling Language (UML) was employed to create Use Case Diagrams for each of the user interactions. These diagrams were compiled sprint-to-sprint into a Use Case Diagram for the completed application. This diagram is included in the Appendix.

The next step of the Design Stage for each sprint was to model the application processes involved in each user interaction. Following from the previous example, a user logging into the application must first access a log in interface or web page. The application must then verify the details entered by the user. A verified login would then direct the user to a landing page or home page whereas a failed verification would return the user to the log in interface. Each of the application processes was modeled using UML Activity Diagrams during each sprint. These diagrams were compiled into a series of complete application Activity Diagrams and included in the Appendix.

The Build Stage
The next stage of each sprint is the Build Stage. During this stage each of the application features were developed. The Build Stage accounted for approximately eight days of each sprint. During this step adaptable and extensible development was considered crucial. This ensured that further incrementally developed features could be efficiently and easily collated into the application.

The Deploy Stage
The third stage of each sprint is the Deploy stage. This stage was composed of testing and collation. During this stage, each new feature developed in the Build Stage was subjected to unit testing in isolation. This was done to ensure that each feature worked as it was expected to before being added to the rest of the application. The new features were then individually collated with the rest of the application and retested.
Before the development sprints began the application controller script, homepage and navigation bar were designed and built. These form the platform onto which the rest of the features are built but at this stage were at their most basic level. The PHP controller script only included configuration scripts and contained just an empty switch statement ready for further features to be added. The home page was simply a HTML landing page and the navigation bar was a template bootstrap navigation bar; both ready for further features to be added.

D.1.1 Sprint 1
The aim of the first sprint was to provide the application with the capability to store, manage and maintain users and to make it secure against unauthorised users. A database table to store basic user details was designed. The table was designed to store a user’s username and hashed password for logging in and out of the application and the user’s name and email address to add more clarity as to who each user is than a username would provide. Tables were also designed to record basic details of user activity on the application. These tables were then built into the application database.

A PHP user class was designed and built to handle all user related functions from logging in and out, verifying user details and hashing user passwords to creating, updating and deleting users and querying the database for user details. Once the database was capable of storing user details a user-friendly log in page and log out button were designed and built using HTML. Corresponding login and log out functions were designed and built as PHP scripts. The logout button was included on the navigation bar so it could be included dynamically along with it. The log in and log out functions were included in the application logic via the controller script.

In order to allow adaptability in terms of the users authorised to use the application and their details, a Manager Users Module page was designed and built. Features to allow users to be created and deleted and their details to be viewed and updated were designed and built and included on the Manage Users Module page.

Further to this, JQuery Datatables were added to allow user-friendly features such as the ability to search and sort users to be added. An extra security feature was added to the Module to allow user activity to viewed and easily searched and sorted. As each new feature was designed and built it was tested in isolation and then tested again one collated with the rest of the application.

D.1.2 Sprint 2
The aim of the second sprint was to provide the application with the capability to store off field Health Check elements and the forms relating to each element as well as to enable users to be able to view these forms. 6 tables were designed and built into the application database to store a Health Check elements details, the title of a form, questions that are part of a form and the type of each question. This enabled the application to be able to store questions of the 6 types of questions specified in the feature scope. Corresponding PHP classes were built for each of these database tables.
Once the database was capable of storing elements, forms and questions two more features were designed and built into the application. The first was to allow users to view a list of the forms in the database. A HTML page to display the form titles and a PHP function to load them were designed for this feature.

The second feature, to allow users to access individual forms, was designed and built next. This feature is one of the core features of the application. It is composed of a PHP script that dynamically loads a form from the database and displays it to the user on a HTML page. This feature was designed and built to incorporate adaptability into the application. As a result, the application can cater for future development of Health Check elements and relating forms. This meets the client’s need for the application to maintain adaptability in order to handle the changing and developing environment of Irish rugby.

D.1.3 Sprint 3
The focus of this sprint was enabling users to submit responses to each of the forms. For this feature to be designed and built, the application needed to be capable of storing responses to form questions. On top of this, the application needed to be able to restrict submission based on the users level. For example, a club level user should only be able to submit for the club they are assigned to whereas regional level users should be able to submit for any club in their region and so on.

To add this functionality to the database, tables were designed and built into the database to store the clubs, regions and provinces as well as the user levels and finally the responses. PHP scripts were then designed and built to allow forms to be submitted by users.

Sprint 4
Restricting the accessibility of users within the application was the main focus of this sprint. The development of framework PHP classes during the previous three sprints allowed for features in this sprint to be developed more quickly and efficiently. Database design and build required less focus as small alterations were all that were necessary. The user tables in the database were modified to incorporate accessibility levels consisting of ‘Yes’ or ‘No’ attributes to both access to the Health Check Reporting application and access to the Health Check Forms module. The capability for users to view past form submissions and manipulate their own was also developed. This featured required a JavaScript function on the front-end of the application to send Ajax calls to the server to reload submissions based on the club selected by the user. Another main feature designed and built during this sprint was the Form Access Module. This module allows administrators to easily edit each user’s access to individual forms.

D.1.4 Sprint 5
The final sprint was devoted almost entirely to designing, planning and preparing for integrating the application with the Health Check Reporting application. The one extra feature added to the application was the ability to allow users to delete past form submissions. This required a PHP script to delete the form submission in a particular order to maintain database integrity. This was due to the foreign structure of the database.
D.2 Integration

The integration phase of this project involved integrating the application with the Health Check Reporting application being developed by Scott Burrell of Trinity College Dublin and was essential to meeting the client’s needs. The Health Check Reporting application was developed to extract data from existing IRFU systems relating to on field Health Check elements and automate Health Check Reports at club, regional, provincial and national levels. Integration of these two applications provides the client with a complete Health Check system fully meeting their requirements. This integration was successfully completed within the project schedule.

Regular periodic discussions were held with the concurrent developer to discuss feature design, font-end design, database design and feature collation. This ensured that all features developed for both the Health Check Data Capturing application and the Health Check Reporting application were consistent throughout development. This contributed considerably to the absence of any delays or problems during the integration phase.

For this project, integration of the reporting application required a function for each off field Health Check element. The same logical flow is common to all of the functions. The functions retrieve the most recent form submissions relating to a given off field element of a club. Depending on the form the function then validates the submission dates against the form’s deadline constraints. The deadline constraints for each form are described in Appendix C.4.

If the submissions are valid, the function retrieves the individual question responses from the database and calculates an off field Health Check element score for the club. The function then generates a breakdown of the score contributions as HTML, based on the responses. The score and the breakdown are then sent to the reporting application’s logical layer for further calculations. Further discussion on the contribution breakdown of each element is included in Appendix C.4. The final action of the functions is to insert the score’s contribution breakdown into especially designated table in the reporting application database.

A final review meeting was held with the concurrent developer to finalise all aspects of integrated features, database design and front-end styling. This was to ensure that the client’s needs were fully met.

D.3 Form Design

The aim of the form design task was to develop effective forms to gather data on the off field elements of the Health Check. The necessity of this task was identified during the requirement analysis phase of the project. This phase included investigation into which elements the client required in the Health Check system. Documentation outlining each of these elements is included in Appendix C.4.
D.3.1 Identifying Health Check Elements
Several elements relating to off field aspects of club were identified as a requirement for the system but without existing data sources. For the client objectives to be achieved, these elements required the data capturing facilities provided by this application. The off field Health Check elements that the application is required to capture data for are:

- Finance
- Medical
- Volunteers
- Branch Engagement
- Project Clubhouse
- LSP/ Council
- Club/ School Link
- Child Welfare

There were originally ten elements that the application was required to capture data for. During the early stages of this task however, two of the elements, Governance and Facilities, were excluded. This was as a result of further investigation into the elements and the client’s requirements. It was deemed, with the agreement of the client, that the evaluation of the governance of a club would be inherently reflected in a clubs scoring across all other off field Health Check elements and so was superfluous to the Health Check. Investigation of the client’s other systems identified that an alternative data capturing tool and reporting application incorporated the Facilities element and so could be excluded.

D.3.2 Information-Gathering Goals
Once the final off field elements were identified, the next stage of this task was defining the client’s information-gathering goal for each. This involved discussing the elements at length with the client to discover what club information would be required to allow for an accurate assessment to be made of the club in relation to each element. The core data required for each element was assessed defined and then reassessed during several subsequent meetings with the client. For example, the core information required to assess the clubs in relation to the Medical Health Check element is:

- The clubs medical equipment
- The medical training of club volunteers
- The clubs adherence to IRFU injury reporting policy

D.3.3 Question Development
The next stage of the form design task was to develop a set of questions for each element that most aptly capture this required information. This stage required substantial discussion with the client over a number of weeks. During this stage, emphasis was placed on developing user-friendly questions. This was in accordance with the client’s objective of retaining volunteer members. Including questions that required lengthy or laborious answers could have risked placing burden on the users, which could result in disenfranchisement or rejection of the application. To avoid this, the questions were developed to incorporate as many checkbox and dropdown questions as possible. These types of question require far less effort than lengthy text responses and also provide more a quantitative level of data.
D.3.4 Users and Timelines

Once the questions required for each element were reviewed and finalised, appropriate users and submission timelines were investigated and defined. It was during this stage of the form design task that the questions for each element were split into individual forms. These forms were defined by the selection of submission user and the timeline for each.

Through consultation with the client, the user most suitable to answer the individual questions for each element were defined. Again taking the Medical element for example, a club’s Club Secretary was deemed to be most suitable to report on the status of medical equipment of their club whereas a branch officer involved with injury reporting procedures was deemed most suitable to report on each club’s adherence to injury reporting. Questions were then grouped by their corresponding element and by their assigned user. These groups formed the basis of the final forms.

The most suitable submission timeline for the submission of each of the questions was then investigated with the assistance of the client. For example, submission of medical equipment status was deemed to be necessary on a yearly basis at the start of each rugby season whereas the client preferred that adherence to injury reporting should be reported on a twice-a-season basis, once near the halfway point of the season and once towards the end of the season. Questions were then grouped by Health check element, user and timeline. These groups represented the finalised forms that would be stored in and accessible through the data capturing application.

D.3.5 Health Check Assessment Score

The next stage of the form design task was to develop a scoring system for each of the off field elements by which to assess each of the clubs. A scoring system was required to generate Health Check reports based on form inputs. This scoring system is used by the integrated Health Check Reporting application to generate reports for the off field Health Check elements. The aim of this scoring system is to assign a score to a club for each of the off field elements. The score is generated based on the most recent submission of the forms for each element.

The first step in developing this scoring system was to assess which parameters, or form data, should contribute to the score and which merely provides a more detailed picture of the state of the club. For example, Finance element forms record the type of accounts submitted by a club. This information provides the client with a clearer picture of the club’s financial setup. However, it doesn’t directly contribute to assessment of the club when compared to another question that records whether or not a club has submitted its accounts at all.

Once the score-contributing parameters had been defined for each off field Health Check element the next step was to review and define the extent to which each parameter contributes to a club’s score. To develop the score contribution of each parameter, the client’s original Club Health Check system was examined. This system was Red-Amber-Green (RAG) system that classifications a club red, amber or green for a number of Club Health Check elements depending on the state of a number of related parameters. The
constraints for these classifications were used as a basis for the scoring system for each of the off field elements for this application.

D.3.6 Scoring System
This final subsection of the form design describes the scoring system developed for the off field Health Check elements with the use applied examples. The scoring system assigns each element a maximum score of 100% meaning this is the maximum a club can score on any given element. Each score-contributing parameter is assigned a contribution percentage score or a series of percentage scores. Constraints were then assigned to the possible values of each parameter in order to establish the level of score contribution that a club has attained from a parameter.

For example, one question, or parameter, relating to the Finance element of a club is whether or not they have submitted their accounts. The score contribution assigned to this question is 20%. The values for this parameter are ‘Yes’ or ‘No’. The constraints on the value of the question are: (‘Yes’=20%, ‘No’=0%). This implies that if a club has submitted their accounts, their Finance element score will increase by 20% and 0% if not.

The sum of the assigned contribution parameter scores for any given element is 100%. For instance, take an element with 5 parameters, each contributing 20% for a ‘Yes’ value and 0% for a ‘No’ value. If the club has a ‘Yes’ value for 4 of the parameters and a ‘No’ value for 1 of the parameters, the club’s score is 80% for that element (20%+20%+20%+20%+0%).

D.3.7 Red-Amber-Green Classification Constraints
The final aspect of the developing the scoring system was to define the classification constraints for each element. The Health Check system requires these classification constraints in order to classify each club as red, amber or green in relation to each of the Health Check elements. Again, the client’s initial Club Health Check’s RAG constraints were used as a basis for developing the constraints. The final classification constraints were defined as a set of three inequalities for each off field Health Check element with one for red, one for amber and one for Green.

For example, if the classification constraints for Finance were (Red, <40%), (Amber, >=40%<=80%) and (Green, >80%); a club scoring less than 40% for the Finance element would be classified as Red, a club scoring more than 80% would be classified as Green and a club scoring anywhere from 40% to 80% would be classed as Amber.

E CONCLUSIONS AND RECOMMENDATIONS
This chapter presents the project conclusions and discusses recommendations for deployment, testing, general use and further development.
E.1 Conclusions

The finished application fully meets the client’s requirements as described in the terms of reference. Additionally, the application incorporates several supplemental features that, while not explicitly included in the original requirements, improve the functionality, adaptability, extensibility and user-friendliness of the application as a whole. The final application markedly reduces several of the problems with the client’s existing system, removing inadequacies, greatly reducing time consumption and providing objectivity.

The application has been successfully integrated with Health Check Reporting application to provide the client with a complete bespoke Health Check system. Deployment of the system online via a web hosting service should remove close to all of the problematic issues with the client’s existing system.

This project presented several considerable challenges throughout. The size of the application, the range of features it incorporates and the number of programming languages required for its development increased the complexity of the project. Compounding this, integrating bespoke applications is a complex procedure in itself and is prone to failure. The number of project stakeholders also complicated the manageability of the project.

The use of a comprehensive framework and well-defined development environment and software development life cycle model reduced the complexity to much more manageable levels. As a result none of the five design-build-deploy sprints ran over scheduled finish times and the application was completed within the specified project life cycle.

An emphasis on consistent communication was maintained throughout the project. Regular review meetings were held with both the client and other stakeholders to discuss work done at each stage. This was central to managing stakeholder expectations and avoiding project creep-scope.

Frequent periodic discussions were held with the concurrent developer to ensure that all features for both applications were consistent throughout development. This contributed considerably to absence of problems and delays during the integration phase.

As a whole, the development project required several hundred hours of development time as well as several hundred more of project management hours over five months. It has been an invaluable opportunity providing countless learning experiences in software application development, project management and stakeholder management. The cooperation and enthusiasm of both the client and stakeholder’s helped a great deal throughout the project.

E.2 Recommendations

E.2.1 Deployment

Currently, the application is hosted on local servers. The aim of the project was to develop the application to be hosted online. It is recommended that it be migrated online. The client will need to source a web hosting service provider that can facilitate MySQL databases and
supports PHP scripting. Due to the popularity of both, there is a large range of possible options.

It is recommended that the client arrange meetings between Rugby Department officers and clubs to assign designated application users for each club. This process will open a dialogue with clubs in relation to the application. It is envisioned that this will improve user acceptance levels as well as facilitating the necessary task of assigning application users.

E.2.2 Testing
To date, the application has only been tested in the development environment on two local servers. Subsequent to migration of the application to online web servers, it is recommended that the application be thoroughly tested. It is also recommended that particular emphasis be placed on testing its security features to ensure that the application is secure in the hosting environment.

E.2.3 General Use
For security purposes, the client is recommended to limit the number of users assigned to administrative level. This is due to the functionality available at this level. A very low number of users should be able to maintain and manage the administrative aspects of the application.

E.2.4 Further Development
The application was designed to incorporate high levels of adaptability and extensibility while still meeting all of the client’s requirements. It is recommended that the client investigate developing the application to add further adaptability features such as the ability for administrator users to be able to add, edit and delete new Health Check elements, forms and subsequent reporting features. The application has been designed to accept such additional future features. This functionality would ensure that the client could keep the Health Check relevant well into the future despite the continuous development of the rugby environment.

APPENDICES

A ORIGINAL PROJECT OUTLINE 33
B INTERIM REPORT 35
C DESIGN DOCUMENTATION 37
C.1 Data model 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.2</td>
<td>Modeling:</td>
<td>39</td>
</tr>
<tr>
<td>C.3</td>
<td>Explained Code</td>
<td>45</td>
</tr>
<tr>
<td>C.4</td>
<td>Form Design</td>
<td>45</td>
</tr>
<tr>
<td>C.5</td>
<td>Design-Build-Deploy Sprints</td>
<td>53</td>
</tr>
<tr>
<td>D</td>
<td>APPLICATION SCREENSHOTS</td>
<td>55</td>
</tr>
<tr>
<td>E</td>
<td>TECHNICAL DOCUMENTATION</td>
<td>63</td>
</tr>
<tr>
<td>E.1</td>
<td>Development environment</td>
<td>63</td>
</tr>
<tr>
<td>E.2</td>
<td>Software Development Life Cycle Model: Incremental</td>
<td>65</td>
</tr>
</tbody>
</table>
A ORIGINAL PROJECT OUTLINE

Client: Irish Rugby Football Union
Project: Rugby Department–Rugby Development Officers Reporting
Location: Lansdowne Rd. Dublin 4
Client Contact: Scott Walker, Director of Rugby Development, Andrew Montgomery, BearingPoint
Dept. Contact: Aideen Keaney

Client Background:
-IRFU
The Irish Rugby Football Union (IRFU) is the governing body for Rugby in Ireland. The IRFU Rugby Department is responsible for the development of all aspects of the game at professional and club level.
A key objective for the department is to increase the level of participation in rugby. This will be through creating opportunities for men and women of all ages and ability, to become involved in the game, and retaining players/members through providing a quality experience. This in turn increases the depth of players available for selection to regional and provincial squads for elite player development.

-BearingPoint
BearingPoint is an independent management and technology consultancy managed and owned by its 120 Partners located in 14 countries across Europe. Employing over 200 technology and management consultants in Dublin, and 3500 internationally, they bring extensive capability and practical experience in implementing business solutions for their clients across the public, commercial and financial services sectors.

BearingPoint has been providing a range of technology and project management advisory services to the IRFU for the last four years.”

Client Requirement:
Each Provincial Rugby Development Manager is currently required to prepare a number of reports for the IRFU on a monthly basis. Reports are based on data submitted by Coach Development Officers (CDOs), Community Rugby Officers (CROs) and Club Community Rugby Officers (CCROs).

Currently these reports are generated manually based on data that is available in existing systems and entries from event diaries.

The IRFU would like to automate this process as much as possible by providing a real time facility to capture data online, and extract reporting data from existing data sources. Data capture will include:

• Schedule of programmes conducted in schools and clubs
• Non-competition games arranged
• Meetings with clubs & schools
• Courses and seminars completed
• Matches observed
• Sessions held with underage development squads

Automating the preparation of these reports will significantly reduce the time spent on administration by Branch officials, allowing more time for interactions with clubs and schools. Where applicable details will also be captured on the category of player/participant, number of participants and any players on the watch list for development squads.
B INTERIM REPORT

Interim Project Report
Management Science and Information Studies

Project: Rugby Department – Rugby Development Officers Reporting Tool
Client: Irish Rugby Football Union
Student: Alan O’Neill
Supervisor: Denise Leahy

Review of Background and Work to Date
Each of the four Irish provinces has assigned a Provincial Rugby Development Manager (RDM) who is required to prepare a number of reports for the IRFU on a monthly basis. The reports are based on data submitted by Coach Development Officers (CDOs), Club Community Rugby Officers (CCROs) and Community Rugby Officers (CROs). These reports are currently generated manually.

The aim of this project is to automate this process as much as possible by providing a real time facility to capture data online and extract reporting data from existing data sources.

Initially the time was spent meeting with the IRFU and BearingPoint. BearingPoint is an independent management and technology consultancy that has been providing a range of technology and project management advisory services to the IRFU for the last four years. The subject of the meetings was requirements analysis – understanding the aim of the project, the business drivers and the parties involved.
To date there have been three meetings and one workshop to clarify the business drivers, requirements and IT systems currently in place.

There is considerable overlap between this project and a project being undertaken by Scott Burrell (MSISS) titled Rugby Department – KPI Reporting Tool for the same client, which will require collaboration between the project managers.

There has also been discussion about the clients preferred delivery platforms as part of the solution investigation phase of the project.

Terms of Reference:
• To investigate the problem through meetings and workshops with the client.
• To research possible solutions.
• To design and develop an online reporting tool that will:
  o Automate the preparation of the RDMs monthly reports
  o Allow the capture of real-time data through forms for CDOs, CRO and CCROs
  o Store the data in a database
  o Output the data in a useable format for the Rugby Development Manager to generate reports
  o Provide levels of detail from Rugby Development Manager to CCRO with restrictions on each level.
• To allow for the questions, content and layout to be edited/updated
• To provide extensive user documentation

Further Work:
There are two meetings arranged for the finalization of the project requirements and the signing of a project charter by the client and other parties involved. From there the application will be developed incrementally. The aim will be to involve the client on the basis of reviewing build stages and testing the application.

The following is an outline of the project schedule:

<table>
<thead>
<tr>
<th>Phase</th>
<th>Deadline:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan</td>
<td>12th December 2014</td>
</tr>
<tr>
<td>Design</td>
<td>9th January 2015</td>
</tr>
<tr>
<td>Build</td>
<td>27th February 2015</td>
</tr>
<tr>
<td>Deploy</td>
<td>13th March 2015</td>
</tr>
<tr>
<td>Operate</td>
<td>27th March 2015</td>
</tr>
</tbody>
</table>

Conclusions:
The overlap between this project and the KPI Reporting Tool Project will require a degree of collaboration with Scott Burrell.
The requirement gathering phase is considerably time consuming as the there are multiple parties involved, multiple IT systems already in place and an overlapping project underway.
The client remains enthusiastic and is eager to see options for
C DESIGN DOCUMENTATION

C.1 Data model
This section briefly contains the data model for the application in the form of an EER diagram. It then describes each of the database layers in terms of their structure and the reasoning behind their design.

Figure C.1.1: Application Database
User Layer:
The User Layer is composed of 10 tables: 1 central table, tblUser, to store all of a user's details; 2 tables to store the details of user log-ins and log-outs; 1 table to store user level descriptions and 6 tables to store user characteristics. The central user table stores all of the information relating to a user including their username, contact details, access statuses and a user’s level and characteristic. The access statuses are two fields, user_access_forms and user_access_reports, which can be set to allow or refuse a user access to all reports and all forms.

The login and logout tables store the date and time a user logs in and logs out as well as the user’s id and the ip address that they logged in from. The application uses these tables to automatically track who is using the system, when they are using it and from where they are using it. This allows a greater level of system security to be maintained.

The user level table stores the description of each level and assigns them an id. The user structure developed through consultation with the client is Club Level, Regional Level, Provincial Level, National Level and Administrator Level.

The user characteristic tables store the names of each of the clubs and the region they belong to, the names of each of the regions and the provinces they belong to, the names of each of the provinces and a separate table with a single entry for the nation. These tables are reverse engineered into a final table called user characteristic. This allows a user, depending on their level, to be assigned to a club, region or province as their user characteristic or be given the national user characteristic while still maintaining the integrity of the database. These user characteristics improve the quality of the logical flow of the application.

Health Check Form Layer:
The Health Check From Layer consists of 7 tables: 1 table to store the Health Check element details, 2 tables to store form details and 4 tables to store question details.

The table tblRagElement stores the title of each of the off field Health Check elements and assigns them an id. This allows the correct data to be found and correct calculations to be made for each Health Check element when generating Health Check Reports and pushing information to the integrated Health Check Reporting application.

The 2 form tables store the details of a form, including its title and the Health Element element it is associated with as well as the users who have access to each form. This allows for users to be given or refused access to individual forms depending on the information they are required to input rather than simply accessing all of the forms or none of the forms.

The 4 tables that store the question details allow for considerable versatility of the questions. The text of each question, its type, the form it is associated with and whether or not it is repeatable within a form are stored in the core question table tblQuestion. The question type allows the integrity of the database to be maintained while still storing all types of question in
a single table. The question types also heavily define the logical flow of the application. The final two tables store any possible options for questions with a defined set of possible answers, such as a question answered using a dropdown box. They store the description of the option and the question they are associated with.

**Response Layer:**
The Response Layer is composed of 8 tables: 1 table to store the details of each form submission and 7 tables to store the individual user responses.

The table tblSubmission stores the data gathered whenever a form is submitted including the time and date of submission; the form that was submitted; the user who submitted the form and the club that the submitted responses refer to. This allows the responses to be bound together as a single submission and also provides the ability to check submissions by date and time and to group submissions by both the submitting user and the associated form.

Of the remaining tables, 6 are to store responses, one for each type of response (corresponding to the types of questions). These tables are reverse engineered into a final table called tblResponse. This table allows each of the different response types to be handled as the same entity while also storing the question the response is associated with, the user who entered the response and the submission that the response is a part of.

**C.2 Modeling:**
**Use Case Diagram**
Figure C.2.1: Application Use Case Diagram

Activity Diagrams:

Displays Update Form Submission Page

User Enters Updated Form Submission

Verifies Form Submission

Includes Invalid Update Message On RAG Form Page

Includes Form Updated Message On RAG Form Page

User Selects Log Out

Logs User Out

Unsuccessful

Successful

Figure C.2.2: Update Form Submission Use Case Diagram
Figure C.2.3: Health Check Forms Module Activity Diagram
Figure C.2.4: Activity Diagram

Figure C.2.3: Login & Home Page Activity Diagram
C.3 Explained Code
The getForm Function
The core functionality of the Health Check Forms Module and of the overall application is the getForm function. This function is central to the adaptability and extensibility of the application. It consists of over 1600 lines of code and gives the application the capability to load any form stored in the application database, current or future.

The getForm function is a PHP scripted function that takes a user selected form’s id from the module-landing page, which displays all forms. The function also takes the logged in user object which is stored as a Session Variable. The function uses the given user object’s level and the form id to load the form title, all of the questions on the form and any JavaScript required from the database and from other php scripts. The function then generates the HTML required to display the form and a set of club selection dropdowns based on the users level and returns them to be displayed to the user.

This function, along with the submitForm function allows for any future form that is inserted in the database to be displayed with the same functionality as existing forms.

C.4 Form Design
This section contains the final summary breakdowns of the form design phase for each off field Health Check element.
### Rag Element: Finance

<table>
<thead>
<tr>
<th>Form:</th>
<th>Finance</th>
</tr>
</thead>
<tbody>
<tr>
<td>User:</td>
<td>Branch Officer (Accounts)</td>
</tr>
<tr>
<td>Frequency:</td>
<td>Yearly</td>
</tr>
</tbody>
</table>

#### Club
- Dropdown

<table>
<thead>
<tr>
<th>Information</th>
<th>Question</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounts</td>
<td>Have they submitted their accounts for last year?</td>
<td>Checkbox</td>
</tr>
<tr>
<td></td>
<td>Date submitted</td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td>Type of accounts submitted</td>
<td>Dropdown</td>
</tr>
<tr>
<td>Cash Flow &amp; Arrears</td>
<td>Competition Fee Paid?</td>
<td>Checkbox</td>
</tr>
<tr>
<td></td>
<td>Registration Fees Paid?</td>
<td>Checkbox</td>
</tr>
<tr>
<td></td>
<td>Insurance Paid?</td>
<td>Checkbox</td>
</tr>
<tr>
<td></td>
<td>Are there any outstanding invoices?</td>
<td>Checkbox</td>
</tr>
<tr>
<td></td>
<td>If yes, what type?</td>
<td>Dropdown</td>
</tr>
</tbody>
</table>

#### RAG Score Constraints:
- Score=100%  
- 60%<=Score<100%  
- Score<60%

<table>
<thead>
<tr>
<th>Questions</th>
<th>Criteria</th>
<th>Score Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Have they submitted their accounts for last year?</td>
<td>value=1</td>
<td>20%</td>
</tr>
<tr>
<td>Competition Fee Paid?</td>
<td>value=1</td>
<td>20%</td>
</tr>
<tr>
<td>Registration Fees Paid?</td>
<td>value=1</td>
<td>20%</td>
</tr>
<tr>
<td>Insurance Paid?</td>
<td>value=1</td>
<td>20%</td>
</tr>
<tr>
<td>Are there any outstanding invoices?</td>
<td>value=0</td>
<td>20%</td>
</tr>
</tbody>
</table>

**Figure C.4.1: Finance Form Design**
### RAG Element: Medical

<table>
<thead>
<tr>
<th>Equipment</th>
<th>Type</th>
<th>Information</th>
<th>Question</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Couch</td>
<td>Checkbox</td>
<td>Club</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pillows (2)</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ambulance Blankets (6)</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Foil Blankets (6)</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clothes Shears</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scissors</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Folding Stretcher</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Penlight</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instant Cold Packs (14)</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPR Face Mask (2)</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Aid Chart</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Aid Room Sign</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Aid Press</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disposable Glove Dispenser</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hands-Free Hand Sanitiser</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kidney Dishes (20)</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clinical Waste Bags (100)</td>
<td>Checkbox</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rugby Safe</th>
<th>Type</th>
<th>Information</th>
<th>Question</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person 1</td>
<td>Text</td>
<td>Club</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date Completed</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Person 2</td>
<td>Text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date Completed</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Form: Medical-Injury Reporting
- **User:** Branch Officer
- **Frequency:** Quarterly

### Injury Reporting
- **Compliant/Non-Compliant**
- **Checkbox**

### RAG Score Constraints:
- Score = 100%
- 60% <= Score < 100%
- Score < 60%

### Questions
<table>
<thead>
<tr>
<th>Questions</th>
<th>Criteria</th>
<th>Score Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Couch</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Pillows (2)</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Ambulance Blankets (6)</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Foil Blankets (6)</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Clothes Shears</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Scissors</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Folding Stretcher</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Penlight</td>
<td>value=1</td>
<td>33%</td>
</tr>
<tr>
<td>Instant Cold Packs (14)</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>CPR Face Mask (2)</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>First Aid Chart</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>First Aid Room Sign</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>First Aid Press</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Disposable Glove Dispenser</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Hands-Free Hand Sanitiser</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Kidney Dishes (20)</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Clinical Waste Bags (100)</td>
<td>value=1</td>
<td></td>
</tr>
<tr>
<td>Rugby Safe Completed</td>
<td>value=1</td>
<td>34%</td>
</tr>
<tr>
<td>Compliant/Non-Compliant</td>
<td>value=1</td>
<td>33%</td>
</tr>
</tbody>
</table>

**Figure C.4.2: Medical Form Design**
### RAG Element: Volunteers

<table>
<thead>
<tr>
<th>Form:</th>
<th>Volunteers - Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>User:</td>
<td>Club Officer (Secretary)</td>
</tr>
<tr>
<td>Frequency:</td>
<td>Yearly</td>
</tr>
</tbody>
</table>

#### Club Dropdown

<table>
<thead>
<tr>
<th>Information</th>
<th>Question</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions</td>
<td>Name of Club President:</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Name of Club Chairman:</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Name of Club Secretary:</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Name of Club Treasurer:</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Name of Director of Rugby:</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Name of Grounds Chairman:</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Name of Club Child Welfare Officer:</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Name of Bar Chairman:</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Name of Health &amp; Safety Representative:</td>
<td>Text</td>
</tr>
<tr>
<td>Committees</td>
<td>Committee Name</td>
<td>Text</td>
</tr>
<tr>
<td>Recruitment</td>
<td>What is the committee structure?</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>What is the recruitment plan?</td>
<td>Text</td>
</tr>
</tbody>
</table>

#### RAG Score Constraints:

- Score > 85%
- 85% ≥ Score ≥ 50%
- Score < 50%

<table>
<thead>
<tr>
<th>Questions</th>
<th>Criteria</th>
<th>Score Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Club President Position Filled?</td>
<td>value=1</td>
<td>7%</td>
</tr>
<tr>
<td>Club Chairman Position Filled?</td>
<td>value=1</td>
<td>7%</td>
</tr>
<tr>
<td>Club Secretary Position Filled?</td>
<td>value=1</td>
<td>7%</td>
</tr>
<tr>
<td>Club Treasurer Position Filled?</td>
<td>value=1</td>
<td>7%</td>
</tr>
<tr>
<td>Director of Rugby Position Filled?</td>
<td>value=1</td>
<td>7%</td>
</tr>
<tr>
<td>Grounds Chairman Position Filled?</td>
<td>value=1</td>
<td>7%</td>
</tr>
<tr>
<td>Club Child Welfare Officer Position Filled?</td>
<td>value=1</td>
<td>7%</td>
</tr>
<tr>
<td>Bar Chairman Position Filled?</td>
<td>value=1</td>
<td>4%</td>
</tr>
<tr>
<td>Health &amp; Safety Representative Position Filled?</td>
<td>value=1</td>
<td>7%</td>
</tr>
<tr>
<td>Number of Committees</td>
<td>value&gt;3</td>
<td>10%</td>
</tr>
<tr>
<td>Is there a committee structure?</td>
<td>value=1</td>
<td>15%</td>
</tr>
<tr>
<td>Is there a recruitment plan?</td>
<td>value=1</td>
<td>15%</td>
</tr>
</tbody>
</table>

**Figure C.4.3: Volunteers Form Design**
### RAG Element: Branch Engagement

<table>
<thead>
<tr>
<th>Form:</th>
<th>Branch Engagement - Officer Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>User:</td>
<td>CRO, CDO or CCRO</td>
</tr>
<tr>
<td>Frequency:</td>
<td>Quarterly</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Club</th>
<th>Dropdown</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Information</th>
<th>Officer Feedback</th>
<th>Question</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Level of Engagement</td>
<td></td>
<td>Dropdown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Form:</th>
<th>Branch Engagement - Committee Meetings</th>
</tr>
</thead>
<tbody>
<tr>
<td>User:</td>
<td>Club Officer (Secretary)</td>
</tr>
<tr>
<td>Frequency:</td>
<td>Quarterly</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Club</th>
<th>Dropdown</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Question</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Committee Meetings</td>
<td>Meeting Description</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Attended By</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Attendee Committee</td>
<td>Text</td>
</tr>
</tbody>
</table>

#### RAG Score Constraints:

- Score>=90%
- 90%>=Score>=50%
- Score<50%

<table>
<thead>
<tr>
<th>Questions</th>
<th>Criteria</th>
<th>Score Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Committee Meetings</td>
<td>value&gt;10</td>
<td>40%</td>
</tr>
<tr>
<td></td>
<td>6&lt;=value&lt;=10</td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>3&lt;=value&lt;=6</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>0&lt;=value&lt;=3</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>value=0</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>value=5</td>
<td>60%</td>
</tr>
<tr>
<td></td>
<td>value=4</td>
<td>45%</td>
</tr>
<tr>
<td></td>
<td>value=3</td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>value=2</td>
<td>15%</td>
</tr>
<tr>
<td></td>
<td>value=1</td>
<td>0%</td>
</tr>
</tbody>
</table>

**Figure C.4.4** Branch Engagement Form Design
## RAG Element: Project Clubhouse

<table>
<thead>
<tr>
<th>Form:</th>
<th>Project Clubhouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>User:</td>
<td>Club Officer (Secretary)</td>
</tr>
<tr>
<td>Frequency:</td>
<td>Yearly</td>
</tr>
</tbody>
</table>

### Club Information

<table>
<thead>
<tr>
<th>Information</th>
<th>Question</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Club officer in charge of maintaining data integrity</td>
<td>Text</td>
</tr>
<tr>
<td></td>
<td>Number of staff trained in data integrity</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td>Has a system cleanup been implemented in the last 12 months?</td>
<td>Checkbox</td>
</tr>
</tbody>
</table>

### RAG Score Constraints:
- Score=100%
- 100%>=Score>=80%
- Score<60%

### Questions

<table>
<thead>
<tr>
<th>Questions</th>
<th>Criteria</th>
<th>Score Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is there an officer in charge of</td>
<td>value=1</td>
<td>40%</td>
</tr>
<tr>
<td>maintaining data integrity?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of staff trained</td>
<td>value&gt;0</td>
<td>20%</td>
</tr>
<tr>
<td>Has a system cleanup been implemented</td>
<td>value=1</td>
<td>40%</td>
</tr>
<tr>
<td>in the last 12 months?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure C.4.5:** Project Clubhouse Form Design
### RAG Element: LSP/ Council

<table>
<thead>
<tr>
<th>Form:</th>
<th>LSP/ Council</th>
</tr>
</thead>
<tbody>
<tr>
<td>User:</td>
<td>Club Officer (Secretary)</td>
</tr>
<tr>
<td>Frequency:</td>
<td>Quarterly</td>
</tr>
<tr>
<td>Club</td>
<td>Dropdown</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Information</th>
<th>Question</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engagement</td>
<td>Level of Engagement</td>
<td>Ranking</td>
</tr>
<tr>
<td>Funding</td>
<td>Date of Application</td>
<td>Date</td>
</tr>
<tr>
<td>Funding Type</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>Received</td>
<td>Checkbox</td>
<td></td>
</tr>
</tbody>
</table>

### RAG Score Constraints:

- Score > 90%
- 90% >= Score > 40%
- Score <= 40%

### Questions

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Score Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level of Engagement value = 4</td>
<td>50%</td>
</tr>
<tr>
<td>value = 3</td>
<td>40%</td>
</tr>
<tr>
<td>value = 2</td>
<td>20%</td>
</tr>
<tr>
<td>value = 1</td>
<td>0%</td>
</tr>
<tr>
<td>Number of funding applications: value &gt; 0</td>
<td>30%</td>
</tr>
<tr>
<td>Number of funding applications approved: value &gt; 0</td>
<td>20%</td>
</tr>
</tbody>
</table>

**Figure C.4.6:** LSP/Council Form Design
**RAG Element: Club/School Link**

<table>
<thead>
<tr>
<th>Form: Club - School Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>User: Club Officer (Secretary)</td>
</tr>
<tr>
<td>Frequency: Quarterly</td>
</tr>
</tbody>
</table>

### Club

<table>
<thead>
<tr>
<th>Information</th>
<th>Question</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>School</td>
<td>Dropdown</td>
<td></td>
</tr>
<tr>
<td>Type of School</td>
<td>Dropdown</td>
<td></td>
</tr>
<tr>
<td>Are they actively playing rugby?</td>
<td>Checkbox</td>
<td></td>
</tr>
<tr>
<td>Do they supply you with pupils</td>
<td>Checkbox</td>
<td></td>
</tr>
<tr>
<td>Number of Boys</td>
<td>Integer</td>
<td></td>
</tr>
<tr>
<td>Number of Girls</td>
<td>Integer</td>
<td></td>
</tr>
</tbody>
</table>

**RAG Score Constraints:**

- Score >= 70%
- 70% > Score > 40%
- Score < 40%

### Questions

<table>
<thead>
<tr>
<th>Questions</th>
<th>Criteria</th>
<th>Score Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of schools supplying club</td>
<td>value &gt; 0</td>
<td>10%</td>
</tr>
<tr>
<td>Level of Engagement</td>
<td>value = 5</td>
<td>70%</td>
</tr>
<tr>
<td></td>
<td>value = 4</td>
<td>50%</td>
</tr>
<tr>
<td></td>
<td>value = 3</td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>value = 2</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>value = 1</td>
<td>0%</td>
</tr>
<tr>
<td>Number of new male students supplied since last submission:</td>
<td>value &gt; 0</td>
<td>10%</td>
</tr>
<tr>
<td>Number of new female students supplied since last submission:</td>
<td>value &gt; 0</td>
<td>10%</td>
</tr>
</tbody>
</table>

**Figure C.4.7: Club/School Link Form Design**
C.5 Design-Build-Deploy Sprints

**Sprint 1: 5\(^{th}\) Jan 15 – 18\(^{th}\) Jan 15**

<table>
<thead>
<tr>
<th>Database:</th>
<th>Security:</th>
<th>Manage Users Module:</th>
</tr>
</thead>
<tbody>
<tr>
<td>User details</td>
<td></td>
<td>Create users</td>
</tr>
<tr>
<td>o Details</td>
<td>Users must have to login</td>
<td>View users</td>
</tr>
<tr>
<td></td>
<td>Users must be able to log out</td>
<td>Search users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sort users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delete users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>View user activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Search user activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sort user activity</td>
</tr>
</tbody>
</table>

**Sprint 2: 19\(^{th}\) Jan 15 – 1\(^{st}\) Feb 15**

<table>
<thead>
<tr>
<th>Database:</th>
<th>Health Check Forms Module:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Health Check elements</td>
<td>View forms</td>
</tr>
<tr>
<td>Forms</td>
<td>View an individual form</td>
</tr>
<tr>
<td>Questions</td>
<td></td>
</tr>
</tbody>
</table>

**Sprint 3: 2\(^{nd}\) Feb 15 – 15\(^{th}\) Feb 15**

<table>
<thead>
<tr>
<th>Database:</th>
<th>Health Check Forms Module:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responses</td>
<td>Submit a form for particular club</td>
</tr>
<tr>
<td>Clubs</td>
<td>Restrict form submissions for clubs depending on the user’s level and the club/ region/ province they are assigned to (no restrictions for national or administrator levels)</td>
</tr>
<tr>
<td>Regions</td>
<td></td>
</tr>
<tr>
<td>Provinces</td>
<td></td>
</tr>
<tr>
<td>User details</td>
<td></td>
</tr>
<tr>
<td>o Level</td>
<td></td>
</tr>
</tbody>
</table>

**Sprint 4: 16\(^{th}\) Feb 15 – 1\(^{st}\) Mar 15**

<table>
<thead>
<tr>
<th>Database:</th>
<th>Manage Users Module:</th>
<th>Health Check Forms Module:</th>
</tr>
</thead>
<tbody>
<tr>
<td>User details</td>
<td>Update user details (accessibility)</td>
<td>View past form submissions</td>
</tr>
<tr>
<td>o Accessibility</td>
<td>Edit individual form access</td>
<td>Restrict viewable submissions to the club selected which is restricted as outlined above</td>
</tr>
<tr>
<td></td>
<td>Search user form access</td>
<td>Search past submissions</td>
</tr>
<tr>
<td></td>
<td>Sort user form</td>
<td>Sort past submissions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update past submission</td>
</tr>
</tbody>
</table>
| access                                                                                                                                                                                                                                                                                                                                 | • Restrict updating past submissions depending on user level  
|                                                                                                                                                                                                                                                                                                                                                                                                      | • Users should be able to view their own details (without editing them) |

| **Sprint 5: 2nd Mar 15 – 15th Mar 15**                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                 |
| Security:                                                                                                                                                                                                                                                                                                                        | Health Check Forms Module:                                                                                                                                                                                                                                                                                              | Health Check Reporting Application:                                                                                                                                                                                                                     |
| • Health Check Report application  
  • All Health Check forms  
  • Individual health check forms  
  • Any administrative features such as adding users etc.                                                                                                                                                                                                 | • Delete past submissions  
  • Restrict the deletion of past submission depending on the user level as outlined above                                                                                                                                                                                                                     | • Validating the most recent submissions for each off field Health Check element with respect to the deadline constraints of each of the related forms  
  • Calculating the Health Check score for each off field Health Check element for any given club  
  • Generating the score contribution breakdown for each element of each club once its score is calculated  
  • Passing the calculated score and score contribution breakdown to the Health Check Reporting application  
  • Inserting the calculated score and score contribution breakdown to the Health Check Reporting application database tables |
D APPLICATION SCREENSHOTS

Figure D.1: User Login Page

Figure D.2: Home Page
Figure D.3: View Forms Module Page

Figure D.4: View Form Page (Finance form)
Figure D.5: View Form Page (Club dropdown selected)

Figure D.6: View Forms Page (Project clubhouse form)

Figure D.6: View Forms Page (Finance past submissions)
Figure D.7: View Forms Page (Past submissions search)

Figure D.8: View User Profile Page
Figure D.8: Manager Users Module Page

Figure D.9: View User Activity Page
Figure D.10: Edit User Access Page

Figure D.11: Create User Page
Figure D.12: View Users Page

Figure D.13: Update User Page

Figure D.14: Delete User Confirmation Box
Figure D.15: Delete User Confirmation Box
E TECHNICAL DOCUMENTATION

The aim of this chapter is to describe the languages and technologies used in the construction of this application.

E.1 Development environment

The above diagram represents the development environment used throughout the project.

Tools Used:

Eclipse

Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java,
Eclipse can be used to develop most applications. Eclipse is an open-source IDE that could facilitate every language required for this project.

**MAMP**

MAMP is a web development environment for Mac OS X computers. Similar to XAMPP for Windows and Linux, MAMP comes free of charge, and is easily installed. MAMP does not compromise any existing Apache installation already running on the system. Apache, PHP and MySQL can be installed without starting a script or having to change any configuration files. If MAMP is no longer needed, deleting the MAMP to everything to its original state. MAMP is a combination of free software and thus it is offered free of charge. MAMP is released under the GNU General Public License. MAMP was deemed the most suitable deployment environment for the development of the application.

**Apache**

The open-source http web server is one of the primary parts of MAMP. Because of its modular structure, it can easily be enhanced with add-ons. MAMP comes with over 70 Apache Modules. Apache supported the client-side layer of the application.

**Programming Languages:**

The main programming and mark-up languages used during the project were MySQL, PHP, HTML, CSS and JavaScript. JQuery was also used in parts.

**MySQL**

A database is at the heart of every modern, dynamic website. MySQL is the most commonly used relational database system. There is a MySQL interface for nearly every programming and scripting language available.

**PHP**

81% of all websites are using PHP as its server-side programming language. It is the most commonly used programming language for creating websites. PHP is preinstalled by nearly every ISP. MAMP comes with 2 versions of PHP. You can switch between the versions in the settings.

**HTML**

The standard markup language used to create Web pages.
CSS
The standard markup language used to create Web pages. Cascading Style Sheets is a style sheet language used for distributing the look and formatting of a document written in a markup language.

JavaScript
JavaScript is an object-oriented computer programming language commonly used to create interactive effects within web browsers. It was used in the application to script functions on the client side and to fire of Ajax calls to the server without reloading the web page.

jQuery
jQuery is a fast, small, and feature-rich JavaScript library. It was used for several aspect of the front end of the application.

E.2 Software Development Life Cycle Model: Incremental
Incremental development, which is a fundamental part of agile approaches, is better than a waterfall approach for most business, e-commerce, and personal systems. By developing the software incrementally, it is cheaper and easier to make changes in the software as it is being developed. Each increment or version of the system incorporates some of the functionality that is needed by the customer.
Incremental development has three important benefits, compared to the waterfall model:
• The cost of accommodating changing customer requirements is reduced.
• It is easier to get customer feedback on the development work that has been done.
• More rapid delivery and deployment of useful software to the customer is possible, even if all of the functionality has not been included.
The following describes the incremental SDLC:

![Diagram](image)

**Figure E.2.1**: Incremental SDLC