Chris Dworczyk

Implementation report

Contents

[2 Introduction 2](#_Toc513047850)

[3 Business Model Introduction 2](#_Toc513047851)

[3.1 Main Window 2](#_Toc513047852)

[3.2 Roster window 2](#_Toc513047853)

[3.3 Player Profile Window 3](#_Toc513047854)

[3.4 Player Form Window 3](#_Toc513047855)

[4 View Model 4](#_Toc513047856)

[4.1 Main Window 4](#_Toc513047857)

[4.2 Roster Window 4](#_Toc513047858)

[4.3 Player Form Window 5](#_Toc513047859)

[4.4 Player Profile Window 5](#_Toc513047860)

[5 Unfamiliar libraries 6](#_Toc513047861)

[6 Error handling and data validation 6](#_Toc513047862)

[6.1 Main Window 6](#_Toc513047863)

[6.2 Player Form Window 6](#_Toc513047864)

[6.2.1 Name 6](#_Toc513047865)

[6.2.2 SRU 6](#_Toc513047866)

[6.2.3 Email 7](#_Toc513047867)

[6.2.4 Date of Birth 7](#_Toc513047868)

[6.2.5 Emergency Phone 7](#_Toc513047869)

[6.2.6 Parental Consent 7](#_Toc513047870)

[6.3 Player Profile Window 7](#_Toc513047871)

[7 Internal Documentation 8](#_Toc513047872)

[8 Supporting User Documentation 8](#_Toc513047873)

[9 Bibliography 8](#_Toc513047874)

# Introduction

In this report I will be going over the implementation of customers software. I will go over all the important features and how they were implemented as well as any features that had be removed or changed due to unforeseen circumstances.

I will also provide the ways in which I took to prevent any user errors and to provide a robust and reliable software.

# Business Model Introduction

When implementing the solution, I already had an idea that there will be 4 main classes responsible for login users in, displaying the member names and 2 others for displaying of player profiles and player forms.

## Main Window

The main window was the most basic of them all. This took care of 2 main things, logging the users in depending on their user type choice and creating player details and skills files if they don’t already exist.

First when the application is launched, the constructor checks if there are 2 XML documents which hold the player details and player skills respectively, if either of them is missing, they are created by their own functions (CreateNewMemberXMLFile(); and CreateNewSkillXMLFile();) with 1 example entry to show it was successful. This entry can be removed once admin logs into application.

Once the user presses the login button and a correct password has been used the Roster window page is initialised with the admin Boolean parameter passed on. This will tell the player form and profile windows if the current user can edit them.

## Roster window

When creating this page, I had to create some way of filtering the members as it would be very messy to display all the member. One solution was simply to filter using the column heading names which would work but it would still display all the other members and if the clubs grows to be in the hundreds spread across all their teams it could influence the responsive of the application.

To fix this, I have added a drop-down menu where the user can select an age group they want to see. Once that change is made an event will triggered (ComboBox\_SelectionChanged())

With this technique when the user logs in there will not be any players displayed until an age group is chosen.

To display member on the screen I had to use the new Data library in System. This is achieved by binding a DataView of the XML member details file to the DataGrid. The file gets read into a string variable then once we instantiate a DataSet we use its ReadXml function to turn it into something we can extract a table from, then that table is turned into a DataView ready for viewing.

The software is made so that you only need to call the DisplayMembers() function and provide it parameters with the DataView and an Integer for age. From there the software will filter out the age and hide all the other unnecessary columns except the player name.

From this menu the user can select and entry in the DataGrid and press button to see members details or the profile with skills. There are some restrictions however that had to be made:

The player form can be opened even thought a member has not been selected yet, this will simply open a blank member form from which the user can either add a new member or go back to select a player details they want to see.

The player profile needs to have a member selected first as it would make no sense to show up blank player profile page.

Lastly a remove function is also present and only available to the secretary (admin). Once a player is selected he can be removed by the click of a button, this will also trigger the deletion of the matching players skill sheet in the other XML file.

Removed players cannot be brought back, you will have to create a new member form.

## Player Profile Window

The player profile takes care of displaying and saving player skills to its own separate XML document.

When creating this page, I had to take into consideration that the player details and skills are in 2 different files. They are only kept linked up using the SRU, this meant that when opening this page a player member had to be selected and a DataRowView had to be passed in. Inside the DataRowView is all the details of the selected player (name, sru, email, date of birth and parental consent) which allowed me to take the sru and find a matching one inside the player profile XML document.

Once we find a match we display the player skills as well as any comments and date of when this page was last edited.

Splitting the documents made the documents much easier to read when viewing outside of the program in a editor.

When saving the options inside the combo boxes are written back into the file along with any comments.

## Player Form Window

The player form window was tricky to finish as it has the most complicated logic as it has to valid a significant amount of user input. The type of validations it dose will be discussed in the [Error Handling](#_Error_Handling) section of the document.

Firstly, there are 2 modes to this window, one is editing which allows to change and save selected players details and the other is adding of new members. The user will not be able to add a new member unit the New Member button is pressed, and an alert will show on label signifying they’re about to create a new member, the fields will have been cleared at this point. The user cannot undo the change into “New Member Mode” and once it has been pressed the user will have to click the back button to return to the roster window to select other players details to view.

When the page is loaded, and a player was previously selected then a DataRowView must have been passed. From that we can extract all the current players details an display them on the screen. From here the admin/secretary would be able to change any of the current data. Once they are finished they just press the save button which will trigger multiple if statements to validate the form fields.

# View Model

Throughout the application I have chosen to go with a green colour theme displayed on the various button controls and outlining some input fields.

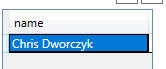
## Main Window

For the UI of the main login window I wanted to keep it simple and functional. As seen in screenshot above you are presented with a window simply displaying a drop down menu for the user type (admin or coach) and underneath a text box for password and again underneath that a login button to verify the login. All this is tied together nicely with in-theme background of rugby balls.

The text has been placed on a white background as it was rather difficult to make out the text on the background.

## Roster Window

Once user is logged in he will see a blank datagrid. This is because an age group has not been selected yet. The user can either use the drop down menu to see a list of age groups (15, 16, 18, 20, Senior) and select one to display. However, I have also made an addition to include arrows next to the dropdown menu, they can be used to either go an age group down (left arrow) or go an age group up (right arrow).

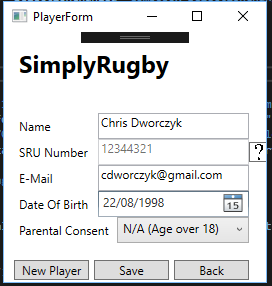
As seen in screenshot they UI looks basic, but you can rest assured the functionality is there. Once the user clicks on a name to select it they can either press the form button or the profile button, both will load their respective windows and displaye the details on there.

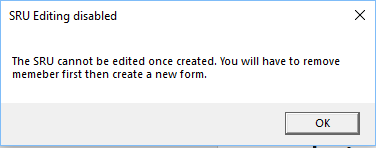
A remove button can also be seen where with that a member entry can be easily removed.

There is a refresh button visible if once the user makes edits to player profiles they can be seen in the main datagrid.

Finally, the logout button will take user back to the main window if a change of user is required.

## Player Form Window

The player form window is made up of a form-like layout. Then player details will be displayed in each field if a member was selected before clicking the player form button, if one wasn’t then a blank form will appear.

From here the user has the option to change any details like name, email, date of birth or parental consent however the SRU is greyed out as it cannot be changed once set, only deleting the member will allow for that change. A help button can be seen (question mark button) next to the field which when clicked will display a message informing the user of this.

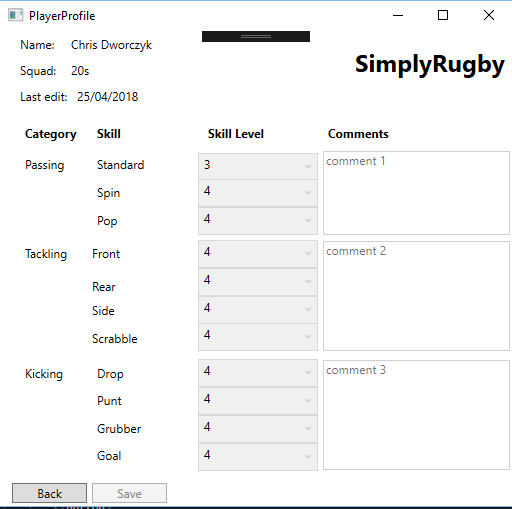
The interface is simple to navigate and everything is right in front of the user to save any confusion.

If a player presses the New Player button they will be alerted that they’re adding a new member to the team. This means when clicking save a new member will be added to the team.

A message will also display of successful save once completed.



## Player Profile Window

The player profile displays all the players skills levels on a drop-down menu scaling 1-5. At the top of the page information about the player is displayed like the name, squad and last time their skills were edited.

The dropdown menus are very easy to use a make sure the user cannot enter a wrong answer. The UI closely resembles the paper-based system previously used and should feel familiar to the customer.

Rich text boxes are also used to allow multiline comments to be saved and displayed.

At the bottom of the page 2 control buttons are placed one to save the current skills and comments and the other to go back to roster window.

# Unfamiliar libraries

During the development I’ve had to use an unfamiliar library to me. The first was the System.XML library which I had to do research on online. This library allowed to me to create/modify XML files which is where I have chosen to store the player details and skills. I had to look up online various syntaxes to better understand how to create elements, their attributes and add it on to nodes. I found useful help on the website (How To Read XML, n.d.)

Using this as my database turned out to be a fairly good and simple choice to the problem. It allowed to read in XML files and turn it into an easily viewable table to be bound with the datragrid. It also provided me with a useful class XmlNodeList which I used to collect all the SRUs and find the player I’m looking for.

Another smaller library I had to use was the Net.Mail library which I used to validate the email. This can be read in the [Email](#_Email) validation section.

# Error handling and data validation

As the application handles a lot of the user input it was very important to make sure all the errors are handled correctly and all the data is validated.

The PlayerForm class has the most complex logic to try catch out all the errors. There are 5 fields and each has its own way of checking if the data entered is correct. The form will not perform the save until all the fields have been validated.

Before any of these fields are validated on their own I check if any of them are empty first. If yes then we throw a message up on screen asking to fill I all the fields.

## Player Form Window

### Name

For the name there are little restrictions. The user can enter any string of characters with a 20 character limit.

### SRU

The SRU servers as a primary key between the member details and skills files. For this reason, I made the choice to disable the ability to edit the SRU once it’s been created.

If a new player is being added 8 numbers are required. An error message will show up if the length is not met.

The field also uses regular expression to ignore any characters that is not a number. This will only allow numbers to be entered in the field.

### Email

The email field was tricky to validate as when I was researching this I found out a lot of emails I thought wouldn’t be allowed, are in fact allowed and valid. The useful blog post (Haacked, n.d.) talks about just how many strange characters an email can have. All the online solutions are using the Regex solution which were simply too strict and wouldn’t allowed otherwise legitimate emails. Now this is not to say that these emails are common but nonetheless they are valid.

Abc\@def@example.com

Fred\ Bloggs@example.com

Joe.\\Blow@example.com

"Abc@def"@example.com

"Fred Bloggs"@example.com

[customer/department=shipping@example.com](mailto:customer/department=shipping@example.com)

all these above are correct emails and for that reason I had to take a different approach.

Instead of checking to make sure the email has an “@” symbol followed by a dot and then maybe by another dot ([exmaple@google.co.uk](mailto:exmaple@google.co.uk)) I instead use the try {} catch {} and inside them I use the Net.mail library to prepare an email as if it was to be sent. If it doesn’t throw an error, we can be somewhat sure that it is a valid.

The only real way to fully validate an email is to send an email with a code or verification link which the user would have to go into their email to verify.

### Date of Birth

This was an easy field to validate as I used an already prepared component. The Date Picker shows the user dates and from there they have no option to type something in wrong. I have also disabled future dates and dates starting before 1940. This was a very easy and clean solution.

### Emergency Phone

For the emergency phone I used the same technique as for the SRU to validate inputs. I made sure only numbers are allowed and a minimum and maximum of 11 numbers is required for it to work.

### Parental Consent

I used a combo box as this again restricts the user’s choice to type in something wrong. They have an “yes” “no” and “N/A (Age is over 18)” and can select whichever.

However, I do another check to see the age of member, I calculate the age first and if its 18 or above and the combo box is not selected to be “N/A (Age is over 18)” then I inform the user once they press save button that the options was changed that as the member doesn’t need their parents’ consent.

## Player Profile Window

For the player profile I have decide to use drop down menus to restrict the user’s choice only allowing numbers to be chosen 1-5. The comments have a character limit of 200 which allows to save a lengthy comment. That is all the validation that was required on this page.

# Testing

For testing I have produced 4 documents with various tables inside with test cases, expected results and the actual results.

# Internal Documentation

The code has been commented for easy reading and understating of various variables/functions. This will allow for easier code maintenance and give a more in depth understanding of various ways Approached the problem.

# Supporting User Documentation

The User Guide document has been created showing off various features in the program as well as answering ant common questions.

# Bibliography

*Haacked*. (n.d.). Retrieved from I Knew How To Validate An Email Address Until I Read The RFC: https://haacked.com/archive/2007/08/21/i-knew-how-to-validate-an-email-address-until-i.aspx/

*How To Read XML*. (n.d.). Retrieved from CSSharp: http://csharp.net-informations.com/xml/how-to-read-xml.htm