Chris Dworczyk

Evaluation

SimplyRugby

Contents

[Project Plan 2](#_Toc515028124)

[Analysis of the current system 2](#_Toc515028125)

[Design of proposed system 2](#_Toc515028126)

[Implementation planning 2](#_Toc515028127)

[Implementation 2](#_Toc515028128)

[Testing 2](#_Toc515028129)

[Documentation 2](#_Toc515028130)

# Introduction

In this report I will be discussing and going over my project that I have been developing the past couple months. It will include an analysis of my own performance as well as a look at the software itself. I have learned many new libraries/techniques throughout the development process and I’m happy in that I’ve provided a solid fully functional product.

# Project Plan

During the project planning phase I approached the problem by starting with the basic user stories, those shortly developed into use cases and the solidified by the requirement specification and its features. I made sure I had the all the client’s main software functionality written down before moving onto the design phase.

To have a good idea of what the program will look like, once I’ve planned out a good portion of the software with the CRC cards and a first-class diagram draft I was ready to create a very early prototype front end UI without any of the real functionalities. This allowed me to see any glaring problems with my plan and help make changes to those problems. With this I was quickly able to see a scope to the program and what type of extra features I might want to add to it.

Overall, I thought my planning stage provided me a good working foundation for the program and made the next stages easier to plan out as now I had a better idea of how the software will look an operate.

# Analysis of the current system

The software is at a fully functional stage and I can safely say that all the required features asked by the client have been successfully implemented. The biggest 3 features that were asked of me to be added are:

* User login page
* Add, remove or edit player forms and profiles
* Display players on screen

There are multiple smaller requirements listed in SRS like displaying errors for wrong passwords but those were small addition made later to the software once the big features were out the way.

As much as I think the software is all done and ready to be given over to the client there are multiple recommendations and future development proposals that I have in mind. These are things that could be added over a longer period as I didn’t have enough time to do so myself.

Changing and handling passwords

Currently the two users (admin and coach) each have their own password. These passwords are in plain text inside the code, this is bad as it could be extracted by someone who has a little programming background. Moreover, the password cannot be changed, if the client wants to do so they will have to come back to me for the password to be changed inside the code and then recompiled. Lastly the password should be hashed and stored using MD-5, there should not be a copy of the password in plain text.

Search function

Currently if a user wants to find a player the quickest option they have is selecting the age group and scrolling through all the players until found. This may take a few seconds if lots of players have been registered. A simple search function could be implemented where the text inside the textbox is matched to the player name, email or any other player information they want to search by.

# Design of proposed system

To begin designing the software I took heavy inspirations from the already existing paper based database. The idea was to create a familiar feeling for the users when they are using the software, for this I recreated the form in which the users enter player information. Once I created these 2 pages I had to think of how to implement another two pages for displaying the players and the other for login users in.

The first few designs I made had an extra window to deal with different jobs for example to add a new player instead of using the already created Player Form window I had a complete separate window just for entering new player details. Overall, I thought the design was simplistic but had all the necessary features to allow the client to store and manage player details.

# Implementation

When implementing the solution I have decided to use an XML file format to store player forms and profiles. This was an unknown library to me which I had to research before fully implementing it into my software. Once I have implemented the code to read and write to XML files I have realised that I will need to create two files, one to keep the player forms and the other to save the profiles. This was to keep the file clean looking and readable as keeping the entire player details and players skills in the same place would be messy. For this I have come with a solution to use the player SRU numbers as their primary keys that would link their player forms with their player profiles, this would also mean I had to make sure the SRU is unique and make it so the user can’t edit it once it’s been created. This entire solution was not planned and came about as I started to get into the nitty gritty of the software and started to program all the validations.

The validation function in my software had easily the most amount of lines compared to any other function. This was due to every field having a different type of validation, some of which I have ever validated for.

## Name

This was the easiest of all the validations I done. I only put a 20 character limit as I didn’t think anyone’s name would have more characters than that.

However I couldn’t fully validate it, when doing all the validations I had to learn another library which was the Regex (regular expressions). This allowed to me filter out certain characters but I couldn’t for some reason get it to work with alphabetical characters A-Z and only with numerical 0-9.

## SRU

For the SRU I used the newly learned library Regex, this allowed me to filter out any character I didn’t want which as anything apart from number 0-9. On top of that I have limited the character limit of 8.

## Email

When validation the mail I learned that it is not easily validated. Emails can look very weird, so weird that some look like they shouldn’t work, but they do. As simple as example@example CAN be an email which was a big surprise to me. To validate this I again had to use another library I’ve not used which was the Mail library. This is used to prepare and send out emails which may come as a surprise to why I’m using it to validate emails. As I said above emails can take lots and lots of different forms and using a simple Regex to filter them out would simply not work. Instead what I done is take the email users have entered and attempt to “prepare” an email for sending, if the email was valid no error would return, if it was invalid windows would throw an exception which I could translate into an invalid email and inform the user.

This was one of the most fun and clever solution I came across when developing my software.

## Emergency Number

For this I again used the already used Regex for SRU to filter out all characters apart from numerical ones and put a 11 character limit on it.

## Date of Birth

For the datepicker I used the already available validation it provides. This was to show no dates further than today and no dates older than 1930. I just assumed there wouldn’t be anyone age 100 signing up for the team.

## Parental Consent

For this I used the combobox which is a validation within itself as the user cannot select anything outside the already created items.

# Testing and Documentation

For the testing I had to come up with a lot of test data to make sure all my validation in player forms were performing correctly. This did surprisingly find me a few errors that I otherwise wouldn’t find myself. For example the regex I used in both the phone number and SRU allowed capital A-Z charter just not lower case, this was fixed shortly after.

Throughout the program I made sure to comment all the difficult section of the code as to provide and easy reading experience for anyone trying to maintain the code or understand how some of the features work. I feel like I went overboard sometimes with the comments even commenting stuff that is self-explanatory. This meant in some cases the code was accrual harder to read with all the green text everywhere.

When creating the user guide I tried to make use of images and text so that the user could understand easily how to perform certain tasks in it. I got one of my friends to read through it to make sure it’s understandable. I feel like I could have made it look more presentable but all the main features were covered inside the user guide.