Qr code

Description automatically generated

**TEST STRATEGY DOCUMENT**

**Project Name:** SimplyRugbyManagementSystem

**Project Number:** 1.0

Test Plan Identifier: SimplyRugbySystem\_KG

**Prepared For:** Simply Rugby Club

**Prepared By:** OOP Solutions Ltd

**Creation:** 27/04/23

**Last Updated:** 30/04/23

**Date Submitted:** 03/05/23

**Class:** HND: Software Development – Graded Unit 2

**Version Number:** 01

Logo

Description automatically generated

Contents

[Document Control 3](#_Toc133966842)

[Revision History 3](#_Toc133966843)

[Document Detail 3](#_Toc133966844)

[Referenced Documentation 3](#_Toc133966845)

[Team Members and Roles 3](#_Toc133966846)

[Diary/Log of Errors 4](#_Toc133966847)

[Traceability Matrix 5](#_Toc133966848)

[Introduction 6](#_Toc133966849)

[Objectives 7](#_Toc133966850)

[Scope 8](#_Toc133966851)

[Test Items 9](#_Toc133966852)

[Testing Methods 10](#_Toc133966853)

[Testing Strategy 10](#_Toc133966854)

[Testing Levels 11](#_Toc133966855)

[Unit Testing 11](#_Toc133966856)

[Environmental Needs 12](#_Toc133966857)

[Test Schedule 12](#_Toc133966858)

[Testing Deliverables 12](#_Toc133966859)

[Approvals 12](#_Toc133966860)

# Document Control

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision No** | **Revision Date** | **Description of Change** | **Author** |
| *1.0* | *27/04/23* | *Document created* | *Kian Gault* |
| *1.1* | *28/04/23* | *Documented Amended after test logs* | *Kian Gault* |
| *1.2* | *30/04/23* | *Documented is finalised for submission* | *Kian Gault* |
|  |  |  |  |

# Document Detail

|  |  |
| --- | --- |
| **Title** | Test Plan – Graded Unit |
| **Version** | 1.0 |
| **Date** | 27/02/2023 |
| **Electronic File Name** | Test Plan – Graded Unit\_KG |
| **Electronic File Location** | Onedrive – Documents – HND – Graded Unit – Development - Testing |
| **Author** | Kian Gault |
| **Contributors** | West – Lothian College – Template Provided |

# Referenced Documentation

|  |  |  |
| --- | --- | --- |
| **Ref** | **Document Name** | **Electronic File location** |
| Test Logs | Data Validation, Event Validation, Functional Acceptance, Navigation Validation | Onedrive – Documents – HND – Graded Unit – Development - Testing – Logs |
| Simply Rugby Management System | SimplyRugbySystem\_KG | Onedrive – Documents – HND – Graded Unit – Development |
| Software Requirements Specification Document | SRS Doc | Onedrive – HND – Graded Unit - Analysis & Design – SRS Doc |

# 

# Team Members and Roles

|  |  |
| --- | --- |
| **Resource Name** | **Role** |
| Kian Gault | Project Manager |
| Kian Gault | Programmer |
| Kian Gault | Tester |

# Diary/Log of Errors

|  |  |  |
| --- | --- | --- |
| **Date** | **Error Description** | **Action** |
| 23/04/23 | When a record is deleted, the emails list does not update with any new records. | The method we used to display just the name and email fields was not right for this scenario.  We used the FindFirstOrDefault method with the use of the primary key ID. The issue was the IDs are giving automatically and increment when a new record is made. This means deleting record ID 2 means the next record created will still be ID 3. So this method was ending when it couldn’t find the next ID.  Solution:  Instead of using the first or default method we decided instead to use the SQLite method SELECT.  This means we can just select the name and email field from the table and display it to the list.  This solved this issue. |
| 19/04/23 | When creating or updating the 18Plus or 21Plus teams, the admin could still select the parental consent tickbox which then threw an exception error as the 18Plus and 21Plus entities do not contain a parental consent field. | To fix this issue, whenever the 18Plus team or 21Plus team become active, we disable the consent checkbox so it’s impossible for the user to add this.  We then reenable the consent checkbox whenever the team switches back to either Juniors or Under18s. |
| 19/04/23 | The coach is able to click on the team checkboxes switching to a different team from their own. | We have added in a conditional statement to check first who the current user type is before performing a checked box event.  If the user type isn’t an admin, the system will stop the process from happening and display a message to the user. |

# Traceability Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requirement ID** | **Requirement Description** | **Result** | **Source** | **Status** |
| 1.1 | “The admin uses their login details to access the system” | When an admin ID and Password is a match to a stored login for Admin the system navigates the user to the next page with Admin privileges. | SRS Document | PASS |
| 1.2 | “The coach uses their login details to access the system” | When the user enters an ID and Password that matches a stored coach login the user is navigated to the next window with permissions giving to them on what type of coach they are. | SRS Document | PASS |
| 1.3 | “Admin wants to edit/add/delete records for the player database” | The admin can successfully create a new record, update an existing record or delete an existing record.  The admin can also do this with any of the four teams. | SRS Document | PASS |
| 1.4 | “Admin decides to search for specific player information” | The admin can achieve this by entering the current teams player SRU number into the search box.  If the program can find a match, that users record will only be displayed. | SRS Document | PASS |
| 1.5 | “The coach wants to edit the player skill profiles” | The coach can edit the skill profiles for their own teams records. This is done through the use of combo boxes with the range of 0 to 5 | SRS Document | PASS |
| 1.6 | “The coach wants to view their team’s personal records” | The coach can view the records of their team and only their team.  The coach can search for a record within their team, but they are restricted access to updating or deleting any of these records. | SRS Document | PASS |

# Introduction

This document will serve as the full documentation of the testing process of the system commissioned by Simply Rugby and will serve as evidence of the system we have created being tested fully.

The system we have built is a management system, so by nature data is very big part of the system so its crucial data validation is at the forefront of our testing so this is what we have done with having many test logs to test the different data inputs that we might expect or not expect.

The system we have designed also has the possibility to have logical errors as we deal with permission access levels and many different operations we these will be tested also.

# Objectives

Our main objective with this testing process is to delivery the highest possible quality possible to our client Simply Rugby. This means the system is fully robust, flawless to use and has exceptional error handling, so the system is fully functional with zero crashes.

Our honesty at OOP Solutions is our most prided aspect of our corporation. To display this we will revert back to our initial software requirements document we created with out clients and test each functional that was agreed upon with the client. We will achieve this through the use of a traceability matrix which will test each requirement to make sure they are of the highest quality.

# Scope

The application we have created for Simply Rugby is very complex and large so the scope of our testing plan will be centred around two aspects which are the view model (User interface and interaction) and the business domain (The backend servers and use of a database to store data).

* Back-end:
  + The back end of our application handles the logical aspect of the application when it comes to data handling. It was discussed very early on with our client that data would need to be saved externally so the data was contained after sessions ended. We achieved this through the use of SQLite integrated with WPF C#.
    - We will test that when we are uploading data from our windows that the database is being used and read from each time.
    - We will also test to see if the data is being stored within the correct entity (table) that has been chosen.
    - We will also perform excesses exception handling to make sure zero invalid data is stored within the database that would cause the front end of the system to crash.
* Front-end:
  + User experience is at our forefront when delivering the best possible software so it’s important this is thoroughly tested and documented.
    - We will test that all navigational routes are working as intended.
    - We will test to make sure all database operations are working as intended i.e. create, update, delete and search.
    - Testing will also occur for our different user tasks.
      * Admin will be tested to make sure this user can successfully create, update, and delete records from the player database.
      * Coaches will be tested to make sure they can successfully update the skill profiles for their teams.
        + This also means testing to make sure the coach cannot amend data that is outside of their own teams.

# **Test Items**

The main test items I will using for this project is test logs. These are small documents but very effective in picking out errors of the system.

* Event validation test log:
  + The event validation test log has been used to test each event trigger which can be done by the user. In this app this is done through the use of buttons and checkboxes.
    - Buttons – Our buttons serve as the users’ way of starting an operation e.g. create, update, delete, search, move window. Once one of these buttons are clicked the back-end of the program will run methods that handle these operations. Each event should be thoroughly tested to make sure each operation works as designed.
    - CheckBoxes – our checkboxes are used to move around our database on the front-end interface. For example if the admin wishes to start editing the seniors table, they can tick the senior’s checkbox and the table will now be selected for amendment. It’s crucial all these checkbox events work so data is stored in the correct and intended place.
* Functional acceptance test log:
  + This test log is used to check the functionality of our program.
    - Here we will test each user operation one by one to make sure their tasks work as intended. This includes: Admin operations e.g. create a member, update a member, search for a member and a coach operation e.g. updating skill profiles or viewing their own teams data without the ability to edit this data.
* Navigation test log:
  + This test log is used to test the different navigation routes for the application. These are triggered through events which are displayed to the user as buttons. Each window should be opened when called and close the previous window which will help performance.
* Data Input Validation test log:
  + These test logs are the most important for this application because they test the different inputs a user could make throughout this system. The three different types of inputs all have different limits.
    - Normal: This set of inputs is what we expect the user to enter i.e. the correct data types for all textboxes and no exceptional lengths.
    - Boundary: This set of inputs is what the system should still be able to handle but might cause some issues like long lengths for certain data types like integers being set at 34 bit.
    - Extreme: These are the set of inputs that will in most cases cause the system to crash. For example entering a word into a textbox that is designed to be an integer. If this isn’t validated in the back-end the program will crash with an error.

# Testing Methods

This test plan will focus on three methods of testing. These include black box testing, white box testing and the use of a traceability matrix.

Black box testing:

This testing method is designed to test a blind users experience without having knowledge of the intended actions within this application. For this a demo was conducted with a tester of our organisation with them not being told what to do on the system. This led to them entering invalid data or trying to do admin operations as a coach user.

This demo resulted in our exception handling stopping the tester from crashing the system and even offered them insight on how to properly use the operations with the use of user messages being displayed when an error was caught by the program.

White box testing:

This testing method is the method on which we test the program with the intended routes in mind. For this, it formed as our functional acceptance testing log which saw us go through each admin and coach operation one by one making sure each function of our program worked.

Traceability Matrix:

This is our table of requirements that we agreed upon with the client during the start of this project. Here we test each requirement one by one determining if the requirement has been fulfilled.

# Testing Strategy

For this testing process I choose the bottom-up method of testing. Majority of possible errors will come from incorrect data being inserted into the wrong place or variable so back-end validation will be crucial.

This application has been designed in an object-oriented fashion so there is no fixed structure to the program as each module and class work polymorphically so the testing process can follow the same approach. This will help reduce time as we can check each functionality and test it instead of using an older method like desk checking.

# Testing Levels

The testing process will only follow two main areas of the application. This being the user side of the application and the back-end side of the system that handles the logic. The user side testing level will focus on the user experience and the user input that will happen in the system. The back-end side will focus on each logical module of the code and test each one to make sure each unit of the program is tested.

# Unit Testing

The window classes handle the event validation, but they do little more than calling either the admin class or coach class to handle the actual database operations.

The create event saves the entered data into local variables then calls the class Admin and calls it’s method Create\_[teamName]() and passes along the entered data that the user wishes to save as a new record. This method handles the creation of a new record within the database by using EntityFrameWork.SQLite’s Add function. It’s important this creation is saved and works correctly so this operation needs to be tested.

The update event does the same as the create method by saving the entered data and stores it locally before calling either the coach class to update the skill profile or calls the admin class to update a record. Instead of just adding to a new record, the program takes the records currently selected from the list window and checks to see if that selected user’s id matches an existing id on the program. If it can find a match, that user is saved and then passed over to the method called update where it will add the new data to the existing user. It’s crucial the records are updated accurately.

Delete follows the same process as update by finding the selected users id and storing it. It them uses the EntityFrameWork.SQLite’s remove() method to remove the found user from the table. It’s important to thoroughly test that the correct user is being removed from the database.

# Environmental Needs

|  |  |
| --- | --- |
| **Hardware** | **Software** |
| PC- minimum hardware requirements | Visual Studio 2022 (Will not work on previous versions of Visual Studio) |
| Memory – 41.6MB of space for the project files. | Word – Test logs |
|  |  |
|  |  |

# Test Schedule

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Start** | **Milestone/Finish** | **Comments** |
| Create Test Plan | 20/04/23 | 30/04/23 |  |
| Create Event Validation Test Log | 22/04/23 | 20/04/23 | One created for each window |
| Create Navigation Test Log | 22/04/23 | 22/04/23 |  |
| Create Functional Acceptance Test Log | 25/04/23 | 25/04/23 |  |
| Create Data Input Validation Test Log | 26/04/23 | 26/04/23 | All windows with validation have a data test log. |

# Testing Deliverables

|  |  |  |
| --- | --- | --- |
| **Deliverable** | **For** | **Date/Milestone** |
| Test Plan | Tester | 30/04/23 |
| Test Logs | Tester | 27/04/23 |

# Approvals

|  |  |  |
| --- | --- | --- |
| **Prepared By** | **Signature** | **Date** |
| **Kian Gault: Programmer/Tester – OOP Solutions Ltd** | **K.Gault** | **30/04/23** |
| **Mr Muir:**  **Spokesman for Simply Rugby**  **Simply Rugby Ltd** | **M.Muir** | **30/04/23** |