

**Solution Planning Document**

**By Renat Oosthuizen**

**1988820**

The reader is advised to view the document at 200% zoom level to improve the readability of the diagrams. Larger copies of the diagrams are provided in PDF form.

**Contents**

Introduction …………………………………………………………………………………………………………………..……. Page 1

Class Diagram ………………………………………………………………………………………………………….…………… Page 3

Use Case Diagram ……………………………………………………………………………………………………..…………. Page 4

Use Case Descriptions ……………………………………………………………………………………………….…………. Page 5

Wireframes ………………………………………………………………………………………………….………………………. Page 9

Entity Relationship Diagram ……………………………………………………………………………..………..………… Page 24

Sequence Diagrams …………………………..…………………….…………………………………………………………… Page 25

Unfamiliar Constructs and Libraries …..…………………….…………………………………………………………… Page 27

**Introduction**

After the submission of the Inception Planning Document, I was notified that the client no longer wished for the prototype to track games and training sessions. Hence, my Solution Planning begun with the modification of the Use Case Diagram in order to reflect the client’s new needs. Now knowing what the prototype needed to do, I made some Use Case Descriptions to give me a general idea for how the prototype would do those things.

I knew quite early on that I wanted the prototype to utilise a database. As the data that I chose to store and the way in which I stored it would have wide reaching influence on the rest of the application, I decided that my next task should be to create an Entity Relationship Diagram. The data stored by the Entity Relationship Diagram was chosen based on the analysis of the forms provided by the client. I also filled out the data object classes in the Class Diagram to mirror that of the database. From previous experience, I knew that being able to access and modify the data from internal data objects would be much simpler than doing that with externally stored data. For this reason, I wanted the application to fetch and store the data from the database, utilise the data as needed and then write the data back into the database if it was modified.

Now that I knew what data the prototype would collect. I needed to figure out how the prototype would collect and display this data. So, I made Wireframes for every screen in the application. During this process, I frequently referred to both the Use Case Diagram and the Entity Relationship Diagrams. By the end, I had a clear idea of how the front end of the prototype should function.

At this point my I stalled in making further designs as I was unsure in my ability to make the initial designs work and felt that making further designs that built upon unproven concepts could end up being wasteful.

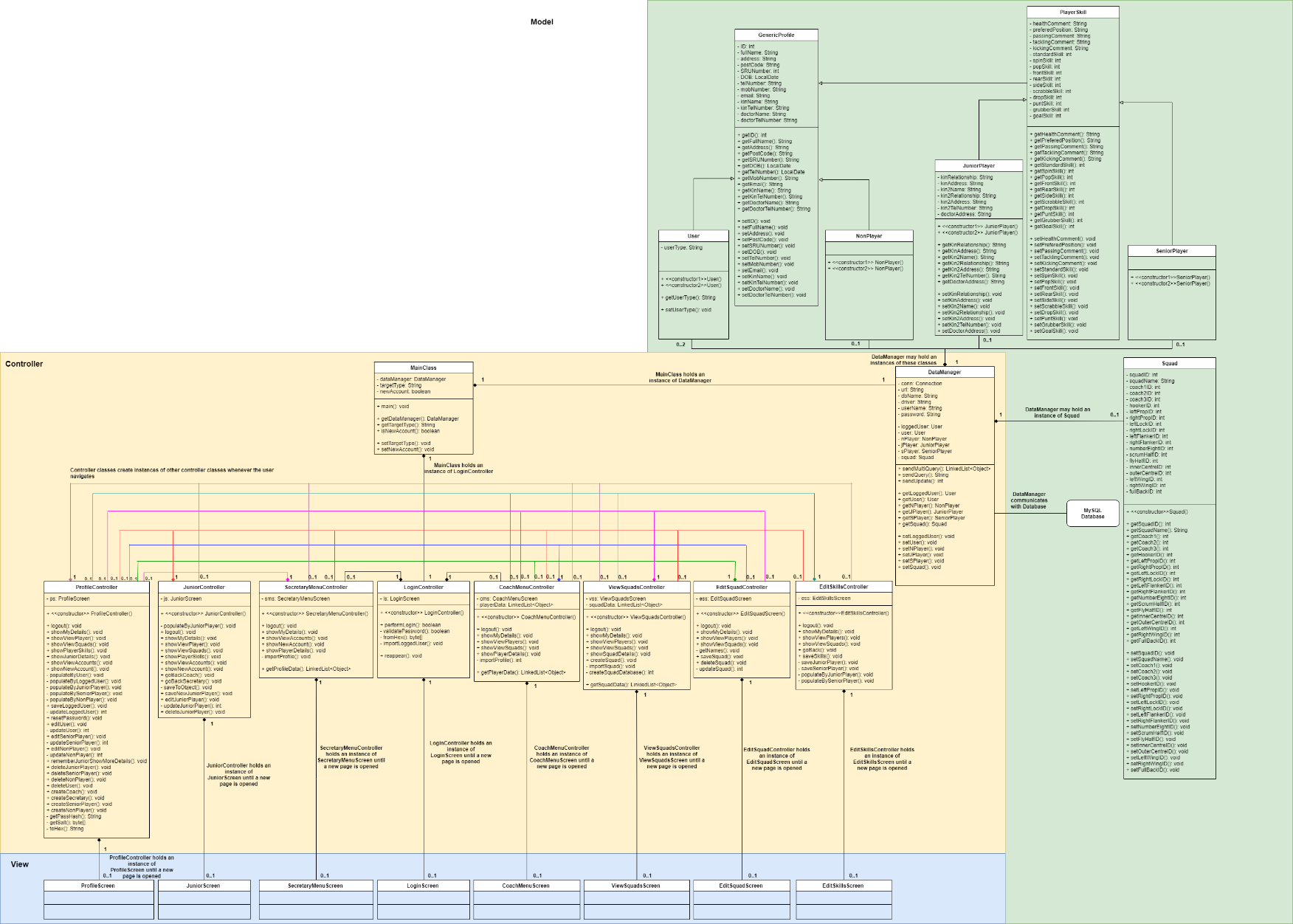
After developing approximately half of my application I realised that some of my designs wouldn’t work and had to be changed. I ended up modifying some of my Wireframes. Because of the changes to the Wireframes, I also had to change my Use Case Descriptions (see Deviation from Initial Design document for a full list of changes).

Once I had programmed these changes into the application and verified that everything could work as needed, I made the Sequence Diagrams. These diagrams were made to complement my Use Case Descriptions.

When I had a complete application, I filled out the Unfamiliar Constructs and Libraries section of the Solution Planning document. This section contains a list of online resources I used during the development of the application. I had kept track of them during the development but didn’t write them down in an organised fashion until the completion of the Implementation. I also finished my Class Diagram as I now knew of every variable and method that the application contained.

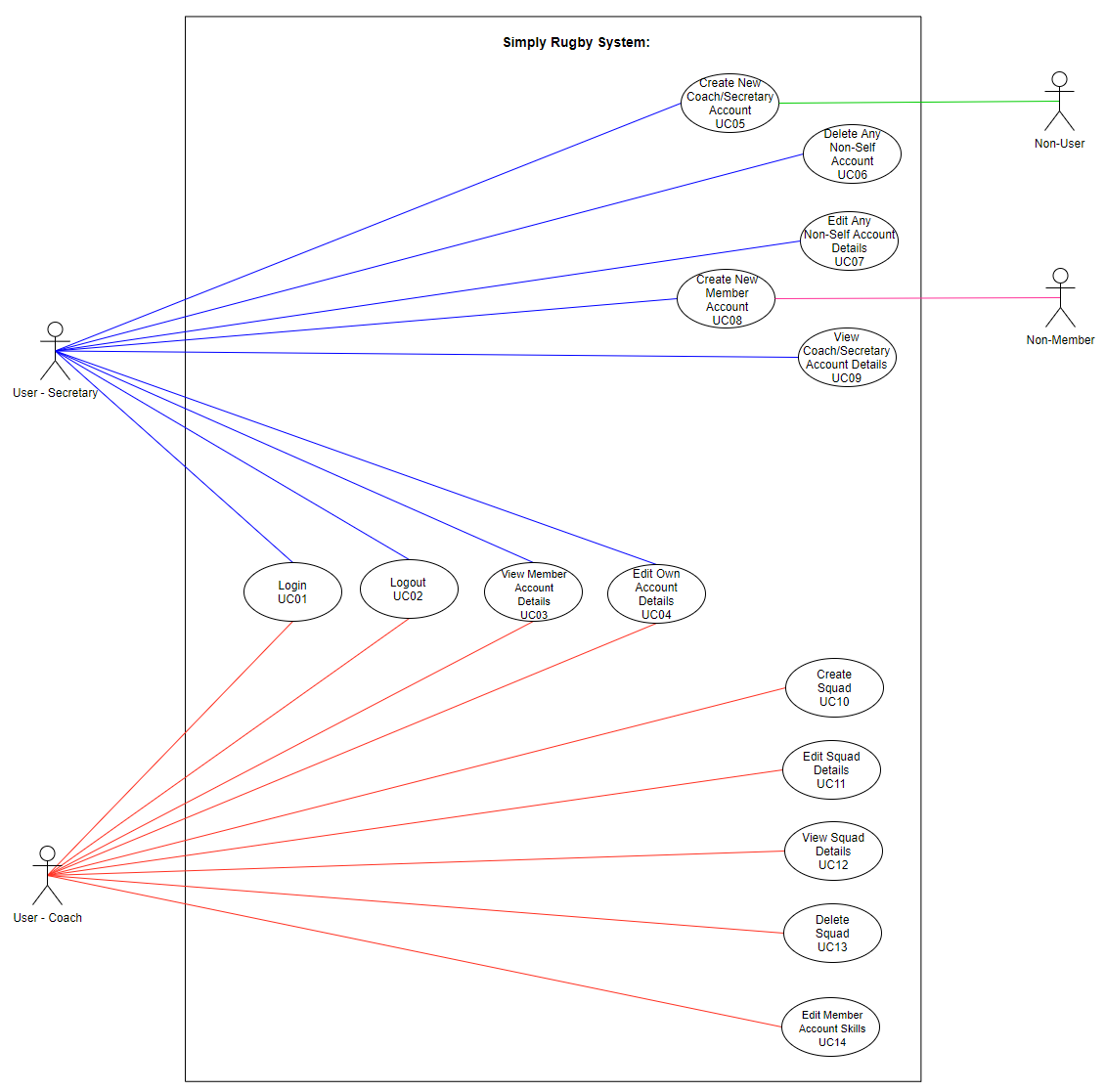
Overall, I found some parts of solution planning to be very useful for designing the application while other parts were almost completely useless. The Use Case Diagram and Concept Class Diagrams were useful for keeping me informed of the general structure and functionality of the application. The Wireframes and the Entity Relationship Diagrams were also extremely useful. I referenced them on a regular basis. The Wireframes told me how the application would function while the ERD let me know where the data was contained in the database at a glance. On the other hand, I did not get any use from the Use Case Descriptions as they did not provide me with any information I did not get from the Wireframes but they took a lot of time to create. Similarly, I did not get any use from the Sequence Diagrams as I had to be late into development before I knew how those systems would work. By the time I knew how to make those systems work I no longer needed the Sequence Diagrams.

**Class Diagram:**

This diagram is a modified and fully updated version of the Concept Class Diagram submitted with the Inception Planning Document. A version of this diagram was used as a reference for coding the application. It contains information on which classes should exist, how they should relate as well as what data and methods they contain.

**Use Case Diagram:**

This is a modified version of the Use Case diagram submitted with the Inception Planning document. A number of use cases related to games and training sessions were removed as the client no longer wished for them to be in the prototype. This diagram can be used to keep track of which functionalities the software should have and which user should have access to those functionalities.



**Use Case Descriptions:**

These descriptions were made to provide further context to the Use Case Diagram. These particular use cases were chosen either due to their complexity or because they were thought to be especially informative of how the application should function in general.

|  |  |
| --- | --- |
| Use Case ID: | UC01 |
| Use Case Name: | Login |
| Created by: | Renat Oosthuizen |
| Date Created: | 27/03/2021 |
| Actors: | 1. Coach or Secretary |
| Trigger: | Coach or Secretary wants to log in. |
| Pre-conditions: | Coach or Secretary has been registered with the system. |
| Post-conditions: | 1. The User is logged into the system and the appropriate view is displayed. 2. User fails to log into the system. |
| Normal Flow: | User enters an Email into the Email field.  User enters a Password into the Password field.  User presses the Submit button.  If Email is not found in database:   * Error text field (Email or Password is incorrect!) is made visible. **End.**   Else if Email is found in the database but hash of text in Password field does not match the hash associated with Email in database:   * Error text field (Email or Password is incorrect!) is made visible. **End.**   If Email is found in the database and hash of text in Password field matches the hash associated with Email in database:   * User is logged in and appropriate view is displayed. **End.** |
| Extends (Alternative flows): | None. |
| Includes: | None. |

|  |  |
| --- | --- |
| Use Case ID: | UC03 |
| Use Case Name: | View Member Account Details |
| Created by: | Renat Oosthuizen |
| Date Created: | 02/05/2021 |
| Actors: | 1. Secretary or Coach |
| Trigger: | A Coach or a Secretary wishes to view the details of a member account. |
| Pre-conditions: | 1. Secretary/Coach is logged in. 2. Secretary/Coach knows the Name or ID of the account they wish to view. |
| Post-conditions: | 1. The profile of interest is found. 2. The profile of interest is not found. |
| Normal Flow: | Coach clicks on “View Players” navigation button / Secretary clicks on “View Accounts” navigation button.  Coach/Secretary looks at the table, scrolling down if necessary, while looking for the target Name or ID.   * If Coach/Secretary finds the target Name/ID in the table then they click on the line containing the target data and then click on the “Select Player”/”Select Profile” button. Page with the Member’s data is then displayed. **End.** * Else if Coach/Secretary reaches the end of the table without seeing the target Name/ID then:   + If the Member they are searching for is nearby then ask them to confirm details.     - If they provide different details then search again with new details.     - If Member does not provide new details or search with new details is unsuccessful: search for similar Names in case it has been misspelled.       * If a possible match is found then they click on the line containing the target data and then click on the “Select Player”/”Select Profile” button. A page with a Member’s data is then displayed.         + If the Member provides information that matches that of the page, then their account has been found. **End.**   + Conclude that the Member has not been registered and stop searching. **End.** |
| Extends (Alternative flows): | None. |
| Includes: | None. |

Note: An identical flow would be used for UC09 (View Coach/Secretary Account Details).

|  |  |
| --- | --- |
| Use Case ID: | UC04 |
| Use Case Name: | Edit Own Account Details |
| Created by: | Renat Oosthuizen |
| Date Created: | 02/05/2021 |
| Actors: | 1. Secretary/Coach |
| Trigger: | A Coach or a Secretary wishes to edit the details of their own account |
| Pre-conditions: | 1. Secretary/Coach is logged in. |
| Post-conditions: | 1. Coach/Secretary makes changes to their account. 2. Coach/Secretary does not make any changes to their account. |
| Normal Flow: | Secretary/Coach clicks on “My Details” navigation button which opens a page displaying their data.  Secretary/Coach edits the text fields they wish and press the “Save” button. The current contents of the text fields become permanently saved.   * If all text fields contain valid data, then they are notified that their data has been saved. **End.** * Else if at least one of the inputs is not valid then they are notified of that the changes have not been saved due to an invalid input. **End.** |
| Extends (Alternative flows): | None. |
| Includes: | None. |

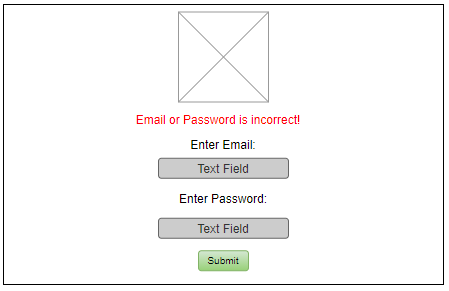
|  |  |
| --- | --- |
| Use Case ID: | UC07 |
| Use Case Name: | Edit Any Non-Self Account Details. |
| Created by: | Renat Oosthuizen |
| Date Created: | 27/03/2021 |
| Actors: | 1. Secretary and 2. Coach, Secretary or Member |
| Trigger: | A Coach, another Secretary or a Member asks for account details to be changed. |
| Pre-conditions: | The account that needs to be changed is found. |
| Post-conditions: | 1. The profile that needs to be changed is changed. 2. No changes are made to the profile. |
| Normal Flow: | Secretary asks 2nd actor what information needs to be changed.  If the 2nd actor asks for a password reset and is a Coach or a Secretary:   * Secretary presses the Password Reset button and clicks Confirm on the subsequent dialogue box. **End.**   Else if the 2nd actor asks to delete their account:   * Secretary presses the Delete Account button and clicks Confirm on the subsequent dialogue box. **End.**   Else if the 2nd actor asks to change any of the standard account information:   * Secretary enters the new information into the relevant field and presses the Save button. **End.**   Else if the 2nd actor is a member that is a Junior Player and asks to change some of the junior details:   * Secretary clicks the More Details button, enters the new information into the relevant field and then clicks the Save button. **End.**   Else the Secretary informs 2nd actor that they cannot make the requested changes and explains the reason why. **End.** |
| Extends (Alternative flows): | None. |
| Includes: | None. |

**Wireframes:**

I made wireframes of each of the view screens in the project in order to plan out how each of the view screens should look and function.

The layout of my pages is inspired by the design used by the Scottish Rugby Management System (SCRUMS) which can be seen in their tutorial videos, found [here](https://www.scottishrugby.org/clubs-and-schools/support-for-clubs/scrums). My reasoning for this is that this design is already being actively used by a large organisation and is therefore likely to be an effective design. Additionally, it is possible that the users of this application have previous experience with using SCRUMS. Having a similar user interface between the two systems would likely make it easier for the users to adjust to using this system after previous exposure to the other one.

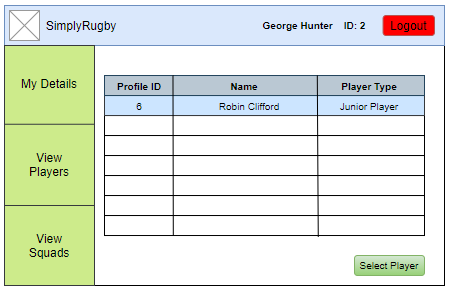
I tried to incorporate a clean, consistent design into my interface. Colour is used to split the page up into three sections. The three sections are the top information bar, the side navigation bar and the central section that displays key information. Buttons are placed at the bottom of the page as much as possible to make the user interface intuitive. Button colours were selected based on general cultural association with the colours. Dark green buttons are used for positive actions such as saving data and selecting profiles. Light green buttons are used for navigation. Red buttons are used for dangerous actions that can result in loss of data, such as deleting profiles. Orange buttons are used for resetting passwords. This colour was chosen due to the need for caution as it could result in significant inconvenience for other users. Large fonts and high contrast between text and background is used to increase readability whenever possible.



This is the login screen use by both Coaches and Secretaries. If the login is successful the system will present the user with view screens appropriate for their user type.

There will be a club logo at the top of the page.

There is a text field for an Email and a text field for a Password. The Submit button will send the contents of the field to be processed by the system. If they do not match the internal records then an error message will be made visible below the logo.



This is the Coach Menu screen shown to Coaches. It can be navigated to by logging in with a Coach account or by pressing View Players in the Navigation Bar.

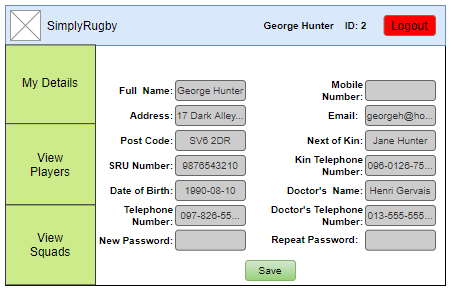
Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

On load the page will display a Player ID, Name and Player Type for all players.

The Coach can click on a line in the table to select it and the press the Select Player button to view the details of the player.



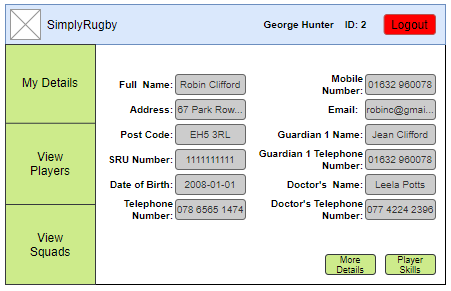
This is the Profile Screen that is shown to Coaches when they click the My Details button in the Navigation Bar.

Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

The Screen contains text fields that display all their profile details excluding their ID and password. The fields can be edited and then the Save button can be pressed to permanently save the changes. A popup will inform the Coach if their details were saved successfully or not.



This is the Profile Screen that is shown to Coaches when they click the Select Player button on the Coach Menu Screen.

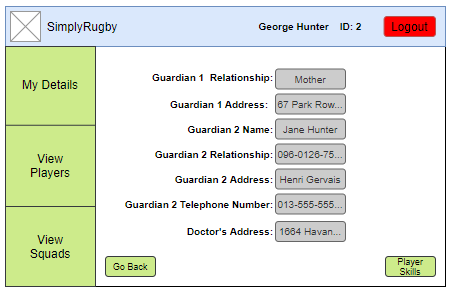
Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

The Screen contains text fields for all profile details excluding their ID. There is a Player Skills navigation button and a More Details navigation button that only appears when viewing a Junior Player Profile.

Note: For Junior Players, “Kin Name” is replaced with “Guardian 1 Name” and “Kin Telephone Number” is replaced with “Guardian 1 Telephone Number”.



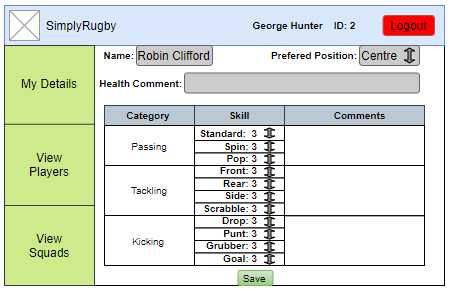
This is the Junior Profile Screen that is shown to Coaches when they click the More Details button on the player Profile Screen.

Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

The Screen contains text fields for all profile details unique to Junior Players. There is a Player Skills navigation button and a Go Back navigation button that will take the coach back to the player Profile Screen.



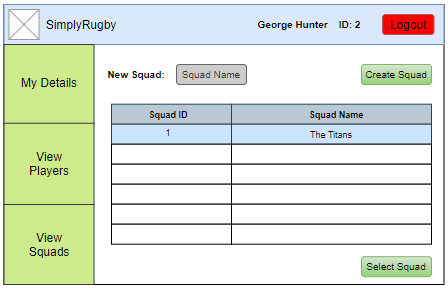
This is the Edit Skills Screen that is shown to Coaches when they click the Player Skills button on the player Profile Screen or the Junior Profile Screen.

Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

The Screen contains a non-editable Player Name text field. There are Health Comment, Passing Comment, Tackling Comment and Kicking Comment text fields which are editable. There are also dropdown menus for various skills with options ranging from 1 to 5 as well as a preferred position dropdown menu. The Save button can be pressed to permanently save the changes and selections. A popup will inform the Coach if the changes were saved successfully or not.



This is the View Squads Screen that is shown to Coaches when they click View Squads in the Navigation Bar.

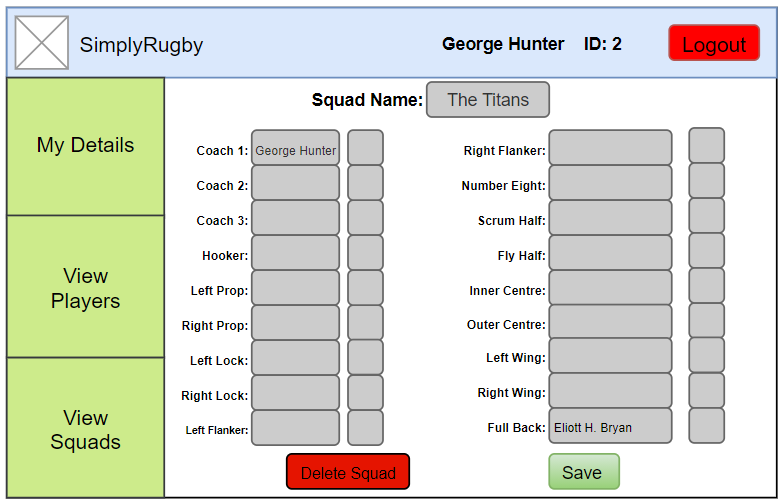
Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

On load the page will display the Squad ID and Name for all squads that the logged Coach is a part of. The Coach can click on a line in the table to select it and the press the Select Squad button to navigate to the Edit Squads Screen.

There is a New Squad text field and a Create Squad button at the top. If the New Squad text field is not empty then this button will create a new squad with that name in the table.



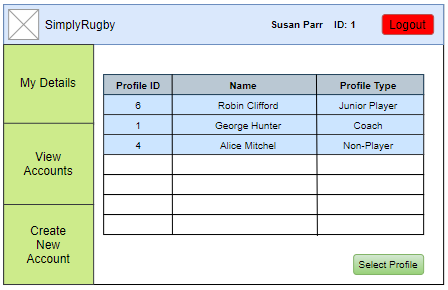
This is the Edit Squads Screen that is shown to Coaches when they click Select Squad or Create Squad buttons on the View Squads Screen.

Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

On load the page displays then Squad Name and the names of all members of the Squad in non-editable text fields. There are also text fields that accept Profile IDs for Coaches and Players. Pressing the Save button will save the profiles for their respective Squad positions and will reload the page to display their names. A popup will inform the Coach if the changes were saved successfully or not. The Delete Squad button can be pressed to permanently delete the squad. A popup message will be displayed, asking the Coach to confirm squad deletion. If the Coach confirms, the Squad is deleted and they are returned to the View Squads page.



This is the Secretary Menu Screen shown to Secretaries. It can be navigated to by logging in with a Secretary account or by pressing View Accounts in the Navigation Bar.

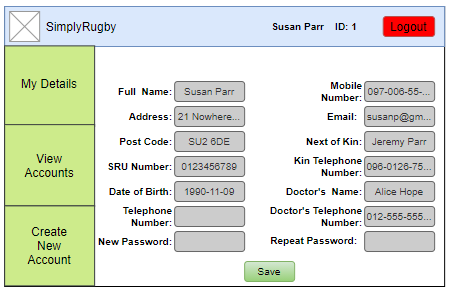
Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

On load the page will display a Profile ID, Name and Profile Type for all profiles.

The Secretary can click on a line in the table to select it and the press the Select Profile button to view the details of the profile.



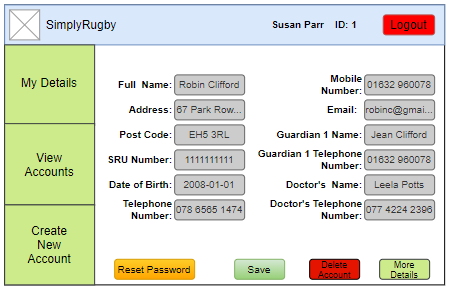
This is the Profile Screen that is shown to Secretaries when they click the My Details button in the Navigation Bar.

Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

The Screen contains text fields that display all their profile details excluding their ID and password. The fields can be edited and then the Save button can be pressed to permanently save the changes. A popup will inform the Secretary if their details were saved successfully or not.



This is the Profile Screen that is shown to Secretaries when they click the Select Profile button on the Secretary Menu Screen.

Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

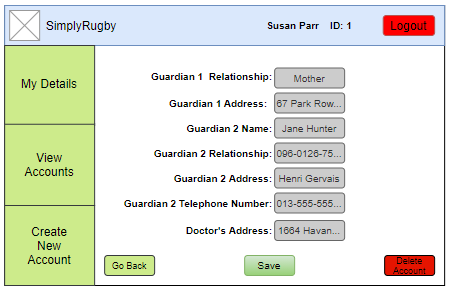
The Screen contains text fields for all profile details excluding their ID. The fields can be edited and then the Save button can be pressed to permanently save the changes. A popup will inform the Secretary if their details were saved successfully or not. The Save button does not appear here for Junior Players.

The Reset Password button only appears for Secretary and Coach accounts. It allows the password for that account to be reset to “root” after Confirm is clicked on the subsequent dialogue box.

The Delete Account button allows that account to be permanently deleted after Confirm is clicked on the subsequent dialogue box. The Secretary would then be taken back to the Secretary Menu Screen.

The More Details button only appears for Junior Players and takes the Secretary to the Junior Player Screen (edits to text fields are remembered).

Note: For Junior Players, “Kin Name” is replaced with “Guardian 1 Name” and “Kin Telephone Number” is replaced with “Guardian 1 Telephone Number”.



This is the Junior Profile Screen that is shown to Secretaries when they click the More Details button on the Profile Screen.

Layout:

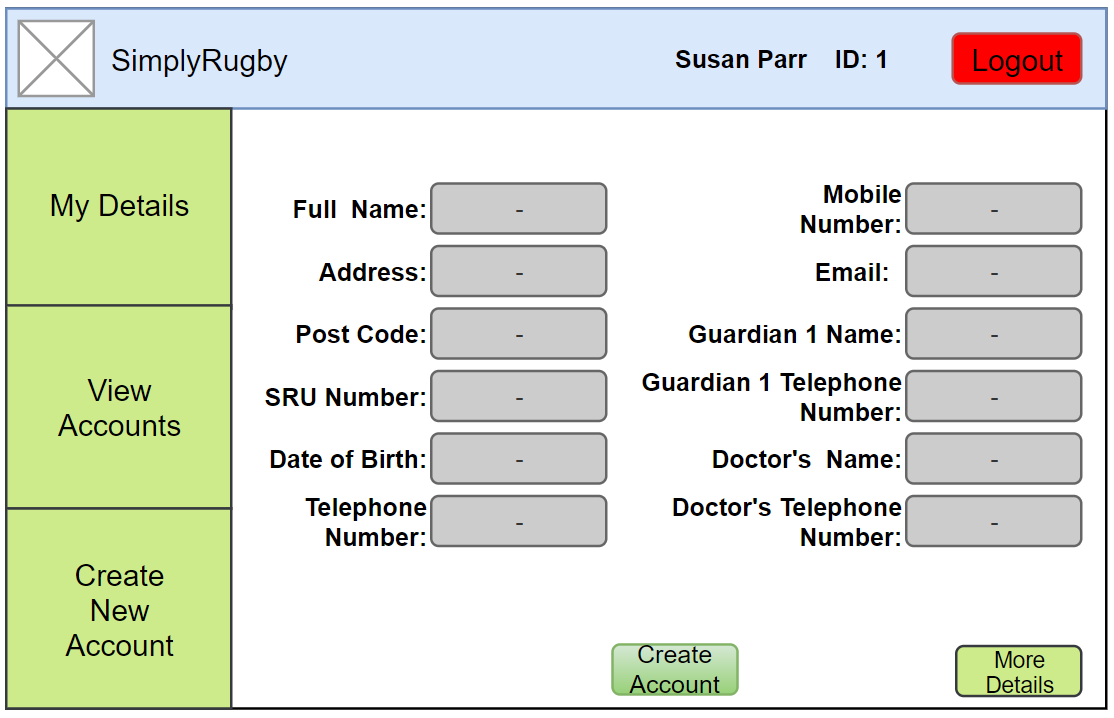
Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

The Screen contains text fields for all profile details unique to Junior Players. The text fields can be edited and the changes saved by pressing the Save button. A popup will inform the Secretary if their details were saved successfully or not (this will save the changes for both this page and the previous page).

There is a Go Back navigation button that will take the Secretary back to the previous Profile Screen (edits to text field are remembered).

The account can be deleted by pressing the Delete Account button and clicking Confirm on the subsequent dialogue box. Secretary will then be taken back to the Secretary Menu.



This is the Profile Screen that is shown to Secretaries when they click Create New Account in the Navigation Bar. Note: They will first be presented with a popup box asking them what kind of account they wish to create.

Layout:

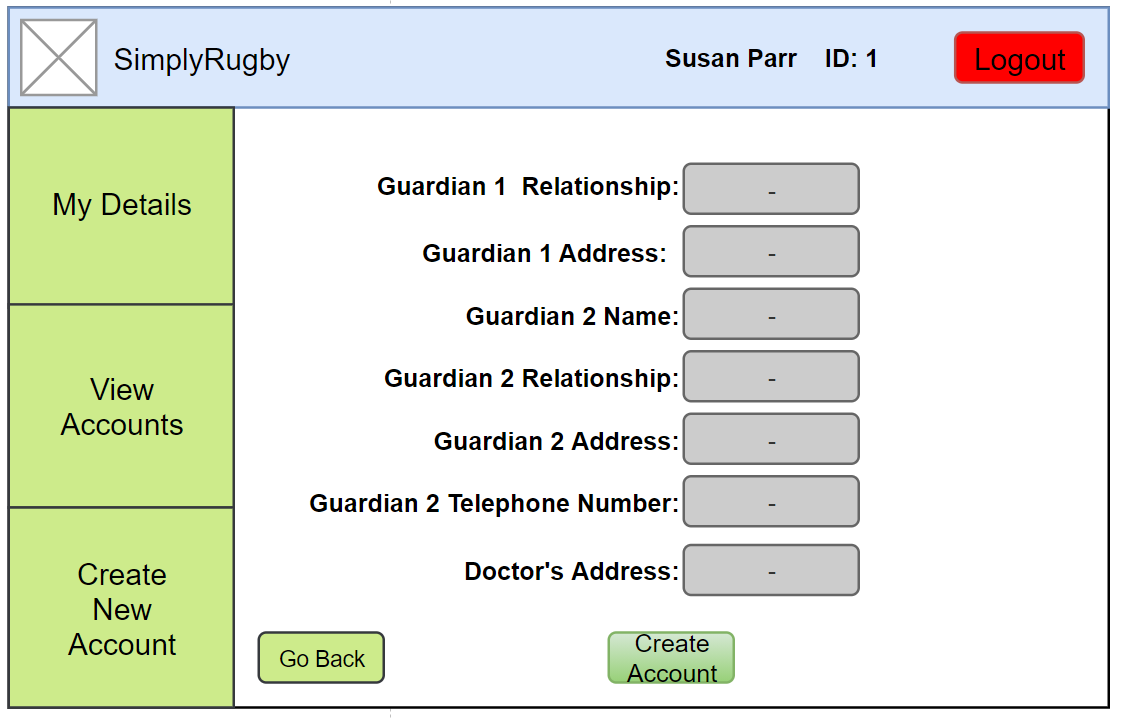
Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

There are a number of text field that can be edited. A new account can be created with the entered data by pressing the Create Account button. A popup will inform the Secretary if the account has been created successfully or not. The Create Account button does not appear here for Junior Players.

More Details button only appears if a Junior Player account type is selected. It takes the Secretary to the Junior Player Screen (edits to text fields are remembered).

Note: For Junior Players, “Kin Name” is replaced with “Guardian 1 Name” and “Kin Telephone Number” is replaced with “Guardian 1 Telephone Number



This is the Junior Profile Screen that is shown to Secretaries when they click the More Details button on the player Profile Screen during account creation.

Layout:

Top of the screen contain the club logo, name of the coach, ID of the coach and the logout button.

Left of the screen contains the navigation bar with 3 options.

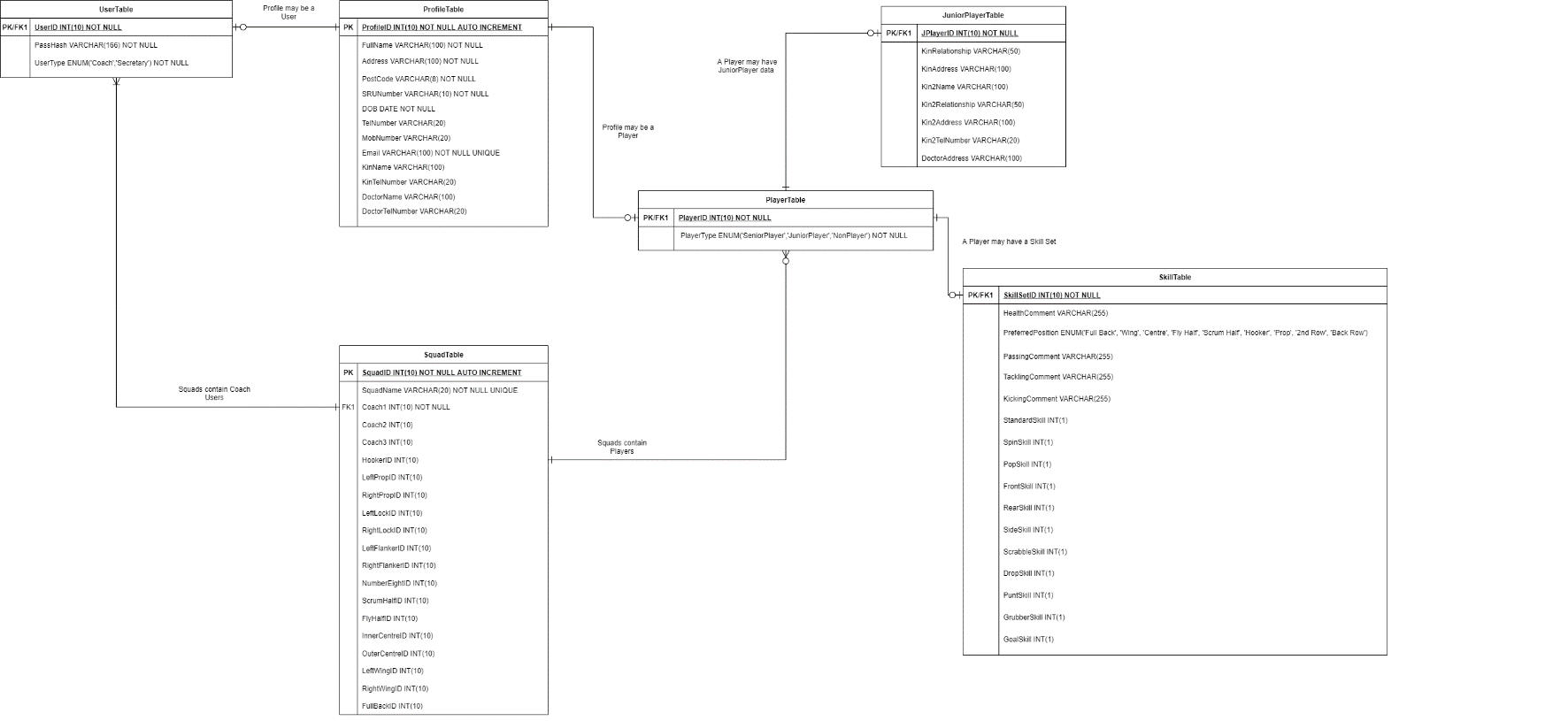
The Screen contains text fields for all profile details unique to Junior Players. The text fields can be edited and a new account can be created with the entered data by pressing the Create Account button (this will also save data entered in the previous page). A popup will inform the Secretary if the new account was created successfully or not (this will save the changes for both this page and the previous page).

There is a Go Back navigation button that will take the Secretary back to the previous Profile Screen (edits to text field are remembered).

**Entity Relationship Diagram:**

I decided quite early on to use a Database in the prototype. My reasons for this were as follows:

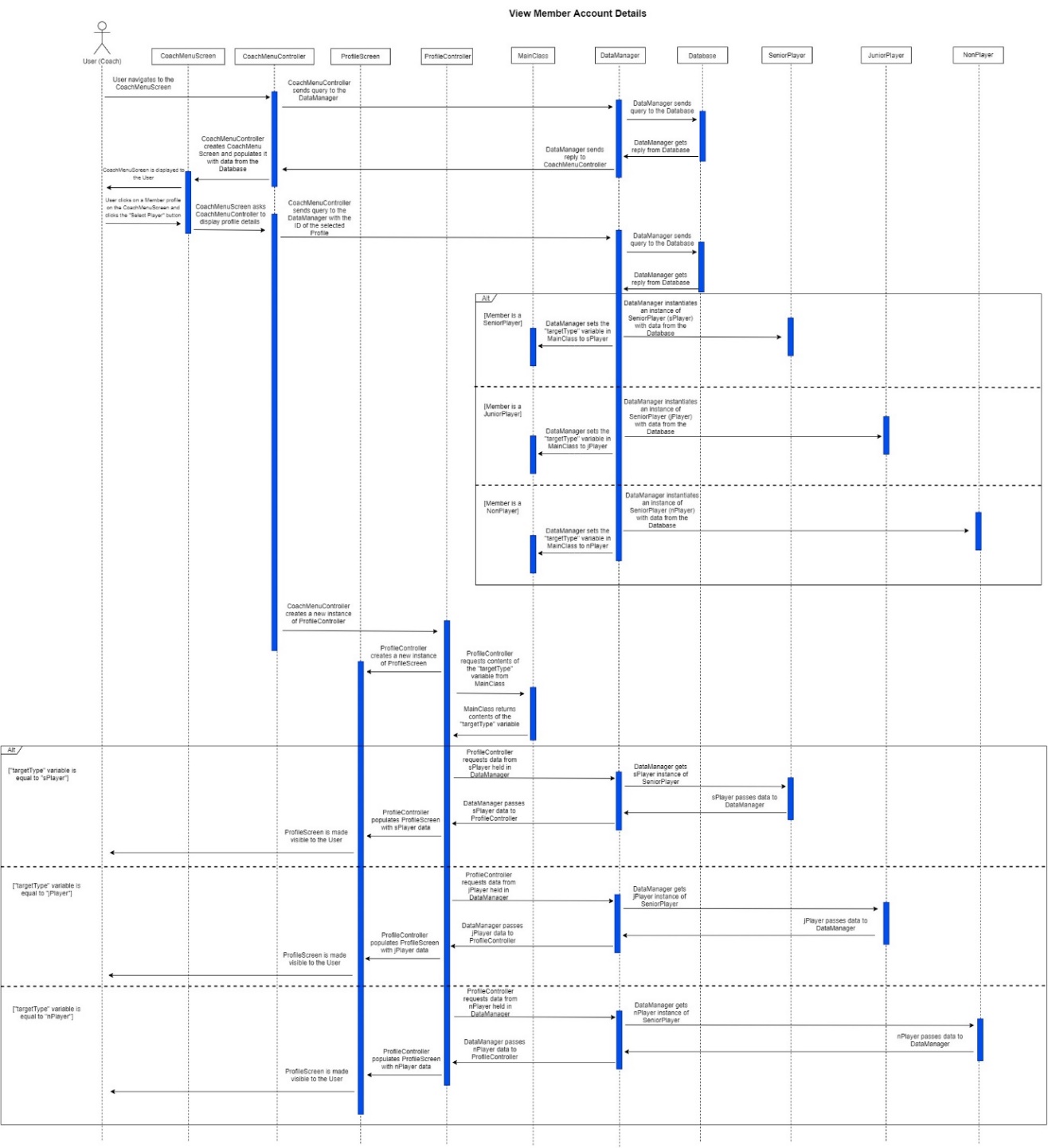
1. The client is interested in having the final version of the software being connected to the database. Data management is an integral part of the software design and adding it later would require a significant redesign of the prototype.
2. By utilising a database, I would eliminate the issue of data desynchronisation between users.
3. Using a database ensures data integrity. Improper data input should never result in total system failure.

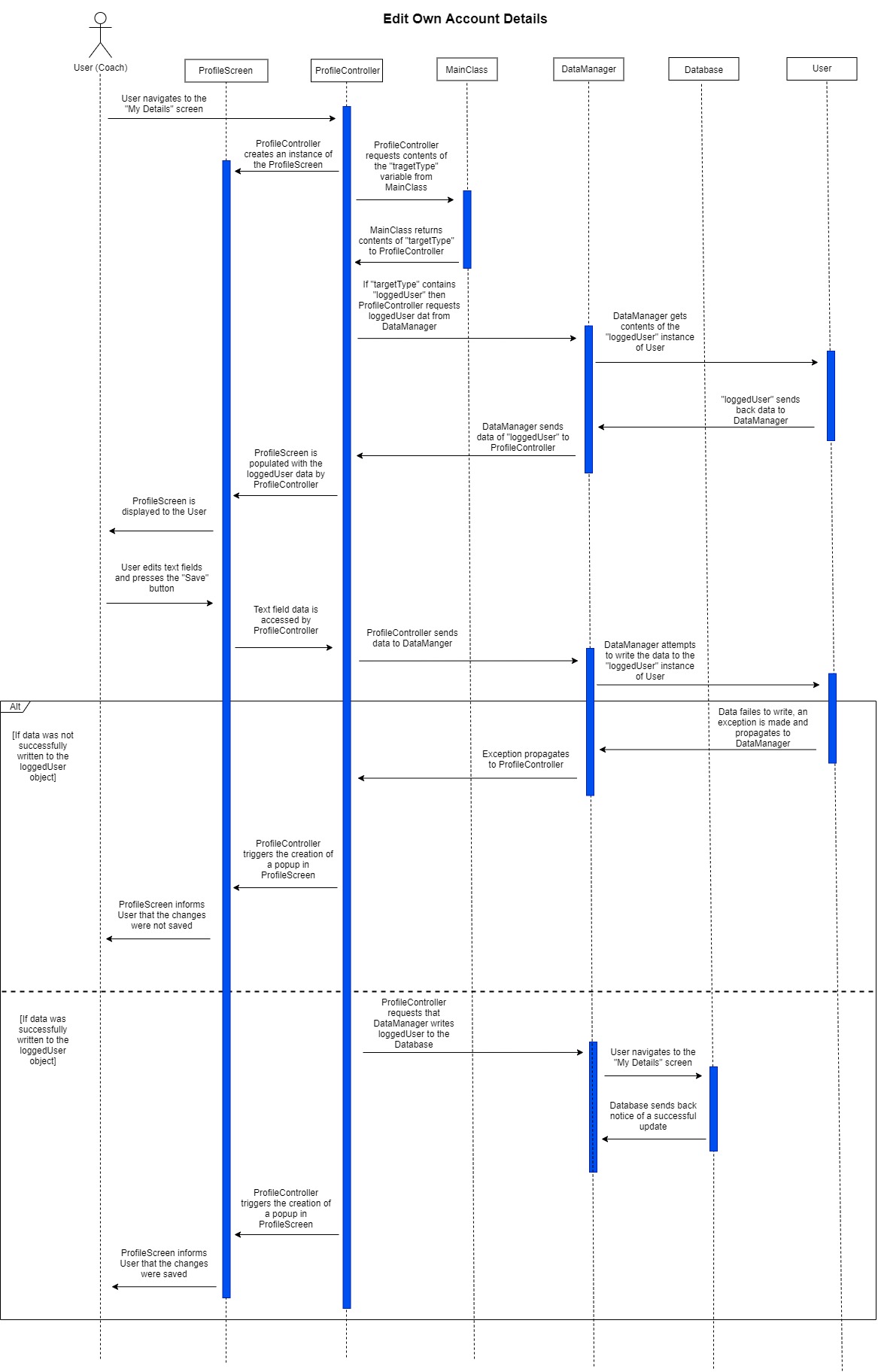
This diagram provides information on how the database is structured which is useful when building the database. It is also useful to refer to while coding the Java software as it provides information on how the data should be structured, which Tables it should be stored and retrieved from as well as which data is required and which is optional.

**Sequence Diagrams:**

The sequence diagrams were created very late in the developments process as the system design needed to be finalised and a lot of prerequisite functionality needed to be completed before the later functionality could be designed. Below are the sequence diagrams for the View Member Account Details (UC03) and the Edit Own Account Details (UC04) use cases. Those were selected as they built upon the Use Case Descriptions provided earlier and are representative of how most of the use cases would function.

In both of these diagrams the User is a Coach. While both of these use cases apply to both Coaches and Secretaries, Secretaries have separate Menu classes and additional functionalities. I choose to only display the Coaches in order to make the diagrams more readable. I felt that there was very little additional utility that would be gained by also displaying Secretary functionalities.

****

****

**Unfamiliar Constructs and Libraries**

During the course of the project, I utilised a number of unfamiliar libraries and constructs.

**Databases**

I used java.sql.\* for communicating with the database. I had previous experience with using databases for web development and thought that this would be a useful experience for me personally as well advantageous for the application itself. These were the resources I used to figure out how to use the library:

|  |  |
| --- | --- |
| **Resource Number** | **Resource Link** |
| 1 | <https://www.youtube.com/watch?v=2i4t-SL1VsU> |
| 2 | <https://dev.mysql.com/downloads/connector/j/> |
| 3 | <https://www.infoworld.com/article/3388036/what-is-jdbc-introduction-to-java-database-connectivity.html> |
| 4 | <https://stackoverflow.com/questions/2839321/connect-java-to-a-mysql-database> |
| 5 | <https://stackoverflow.com/questions/1905607/cannot-issue-data-manipulation-statements-with-executequery> |
| 6 | [www.hostinger.com](http://www.hostinger.com) |
| 7 | <https://stackoverflow.com/questions/37779418/how-to-map-sql-date-to-localdate> |
| 8 | <https://support.hostinger.com/en/articles/1583546-how-to-set-up-a-remote-mysql-access> |

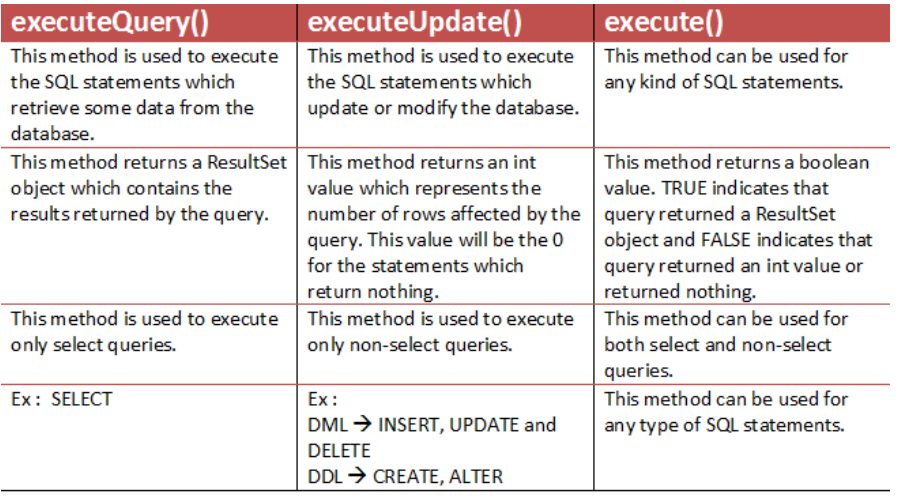
Before I could use a database in Java, I had to set one up. I used Hostinger (resource 6) to host my database. I had SQL from a previous web development project which I used to quickly build a database for testing purposes.

I used resource 3 and 4 to understand the code for sending and receiving queries to the database. I also used resource 1 to figure out how to setup the JDBC driver necessary for communicating with the database. I got the JDBC driver from resource 2. Once I had created a database and written all the code for communicating with it, I still failed to connect to it. Using resource 8, I learned that this was because by default the database was set up to only accept connections from local host. This resource also taught me how to configure the database to accept all external connections.

Once I had created the database for the SimplyRugby application I run into a problem importing date data from the database into my application. Thanks to resource 7 I learned that this was due to the way MySQL stores dates in an SQL DATE data type. I also learned how to convert SQL-DATE to LocalDate datatype.

Later on, I run into a problem sending update queries to the database. Using resource 5 I learned that this was because the JDBC driver uses different methods for sending different query types.

This image from resource 5 was particularly useful:



**Cryptography**

Since I was already using a database, I decided to also use password hashing. I had previous experience with password hashing and though that it would make a useful, simple to implement addition. I used javax.crypto.SecretKeyFactory and javax.crypto.spec.PBEKeySpec libraries for password hashing. I learned how to use them from the resources below.

|  |  |
| --- | --- |
| **Resource Number** | **Resource Link** |
| 1 | <https://howtodoinjava.com/java/java-security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/> |
| 2 | <https://www.baeldung.com/java-password-hashing> |

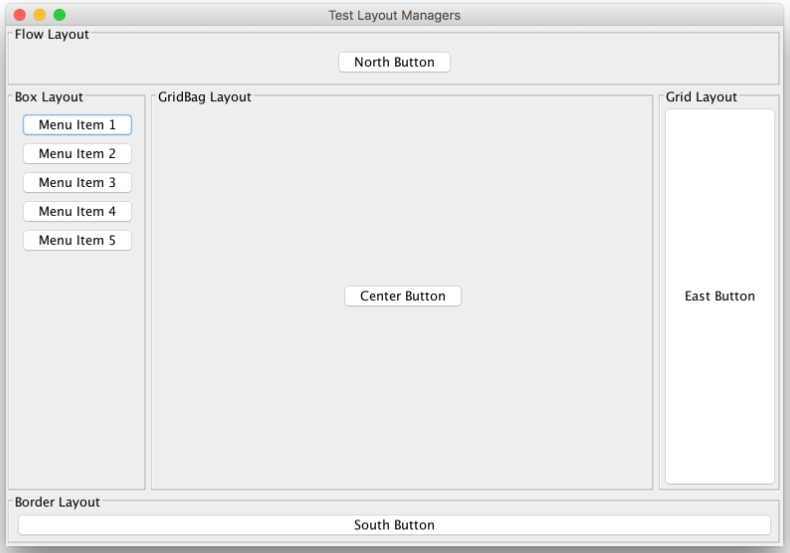
Resource 1 provided me with the code needed to hash and salt a password as well as to check if the entered password matched the hashed password. The code was modified to fit my needs and improve readability. Resource 2 provided me some additional information regarding different hashing algorithms. Overall, I had very little trouble with cryptography.

**Java Graphical Elements**

One of the requirements for this project was for the application to use a graphical interface. My previous experience with graphical elements in Java was limited, however, I was determined to create a good-looking graphical interface that also scaled to the screen resolution. I used javax.swing.\* for this purpose. In order to make the interface scalable I had to learn a lot about templates. Below are the resources I used to make that happen.

I used resources 1 to 13 to learn about different layouts managers. During this process I made many unsuccessful attempts to use a layout before figuring out that it wouldn’t be able to do what I wanted and moving on to a different layout. Ultimately, I ended using a Border Layout as my main page layout. I also used a Box Layout for the left panel containing a navigation bar on each page and a Grid Bag Layout for the central panes containing the main content of each page. I also ended up using Box Layout Glue that I learned about from resources 8 and 14 to dynamically separate elements as the screen size changes.

This diagram from resource 1 was particularly useful to me:



One of the problems I encountered was that some elements would only scale when both their Preferred Size and their Maximum Size were manipulated. Resource 15 helped me with this.

I used resources 16 and 17 to figure out how to use Dialogue boxes.

Lastly, I needed to figure out how to implement JLists. I used them both for displaying lists of players/profiles/squads on pages as well as inside dialogue boxes to select which account the user wished to create. I accomplished this thanks to resources 18, 19, 20 and 21. I also used resources 20 and 22 to create scroll bars around the JLists on my pages so that they could still display all their content regardless of whether it exceeds the current dimensions of the JList.

|  |  |
| --- | --- |
| **Resource Number** | **Resource Link** |
| 1 | <https://stackoverflow.com/questions/21375255/jpanel-positions-and-sizes-changes-according-to-screensize/21376596#21376596> |
| 2 | <https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html> |
| 3 | <https://www.youtube.com/watch?v=pDqjHozkMBs&list=PLZPZq0r_RZOMhCAyywfnYLlrjiVOkdAI1&index=57> |
| 4 | <https://www.youtube.com/watch?v=PD6pd6AMoOI&list=WL&index=66> |
| 5 | <https://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html> |
| 6 | <https://www.youtube.com/watch?v=g2vDARb7gx8> |
| 7 | <https://www.youtube.com/watch?v=ohNqQagkDDY&list=PLZPZq0r_RZOMhCAyywfnYLlrjiVOkdAI1&index=58> |
| 8 | <https://docs.oracle.com/javase/tutorial/uiswing/layout/box.html> |
| 9 | <https://stackoverflow.com/questions/24780480/gridbaglayout-how-to-set-fixed-column-width> |
| 10 | <https://stackoverflow.com/questions/27391049/java-change-jtextfield-size-inside-a-gridlayout> |
| 11 | <https://stackoverflow.com/questions/23855974/multiple-panels-in-jframe> |
| 12 | <https://docs.oracle.com/javase/tutorial/uiswing/layout/group.html> |
| 13 | <https://docs.oracle.com/javase/tutorial/uiswing/layout/grid.html> |
| 14 | <https://stackoverflow.com/questions/29353004/how-to-have-one-component-on-the-left-and-one-on-the-right-by-flowlayout> |
| 15 | <https://stackoverflow.com/questions/3692987/why-will-boxlayout-not-allow-me-to-change-the-width-of-a-jbutton-but-let-me-chan> |
| 16 | <https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html> |
| 17 | <https://www.javatpoint.com/java-joptionpane#:~:text=The%20JOptionPane%20class%20is%20used,JOptionPane%20class%20inherits%20JComponent%20class> |
| 18 | <https://www.javatpoint.com/java-jlist> |
| 19 | <https://www.codejava.net/java-se/swing/jlist-basic-tutorial-and-examples> |
| 20 | <https://docs.oracle.com/javase/tutorial/uiswing/components/list.html> |
| 21 | <https://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html> |
| 22 | <https://www.javatpoint.com/java-jscrollbar> |