HND Computing:

Software Development

Graded unit 2

H48W35

Solution Planning Document

Brodie Harkins

**Content Page:**1 - [Business model](#BusinessModel)

* 1. – [CRC Cards](#CRCCards)
  2. – [Static Model](#StaticModel)
  3. – [Dynamic Model](#DynamicModel)
  4. - [Use Case Diagram (Coach)](#UseCaseCoach)
  5. - [Use Case Diagram (Admin)](#UseCaseAdmin)
  6. – [Activity Diagram](#ActivityDiagram)
  7. – [Sequence Diagram](#SequenceDiagram)

2 - [UI Design](#Wireframes)

2.1 – [View Model](#ViewModel)

Business Model

Identifying Class methods & Attributes

* 1. **-** **CRC Cards**

CLASS: SimplyRugby

RESPONSIBILITY:

Know its Users (Admin/Coach)

CLASS: JuniorManager

RESPONSIBILITY:

Know its Junior Players

Create a list of Juniors

COLLABORATION: Junior (SuperClass)

CLASS: Coaches

RESPONSIBILITY:

Know all coaches

COLLABORATION: Admin Home (SuperClass)

CLASS: Player

RESONSIBILITY:

Know Skill Categories

Know Player details

COLLABORATION: SimplyRugby

CLASS: PlayerManager

RESPONSIBILITY:

Know its players

Create a player list

COLLABORATION: Player (SuperClass)

CLASS: SkillCatagory

RESPONSIBILITY:

Know list of Skill categories

Know skill category name

Know list of skills

COLLABORATION: Players (Superclass)

CLASS: Skill

RESPONSIBILITY:

Know skill name

Know skill Rating

COLLABORATION: SkillCatagory

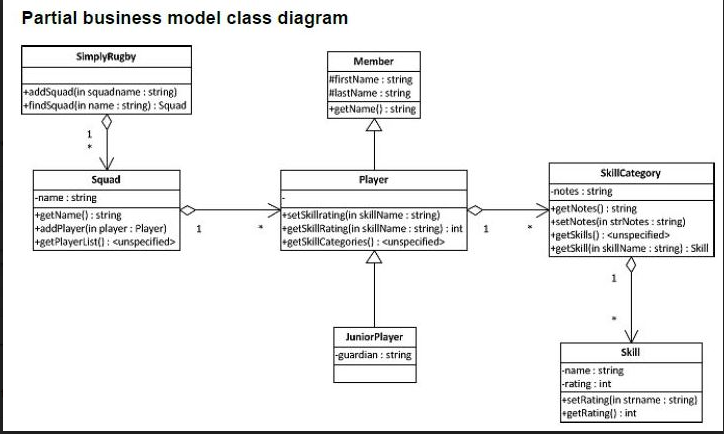
CLASS: Junior

RESPONSIBILITY:   
Know Junior details

Know Junior skills

COLLABORATION: Player (SuperClass)

Static Model

**Partial business model class Diagram provided by the client** 

**Expanded Class Diagram** Diagram

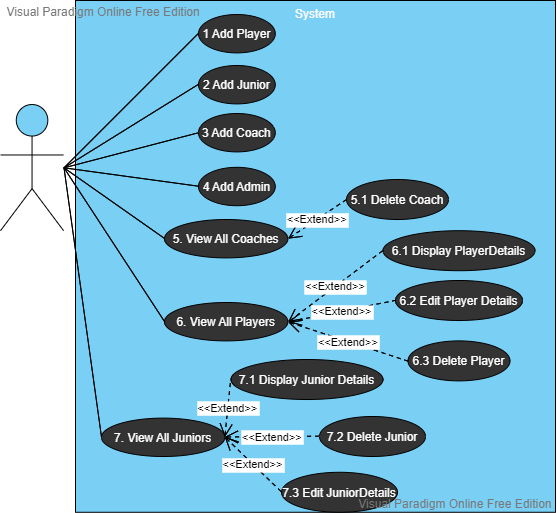
Description automatically generated

Dynamic Model

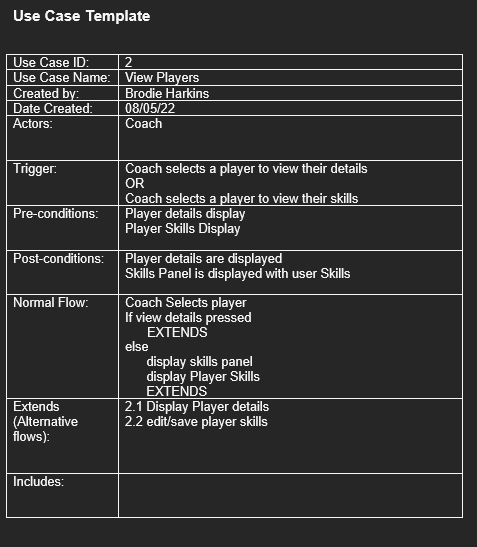
**Use Case Diagram - Coach**Diagram

Description automatically generated

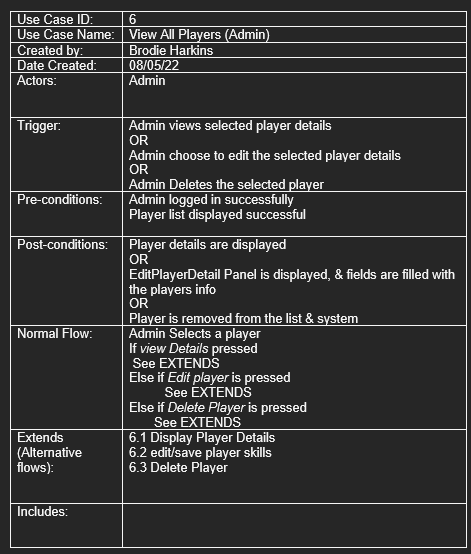
**Use Case Diagram - Admin Page**



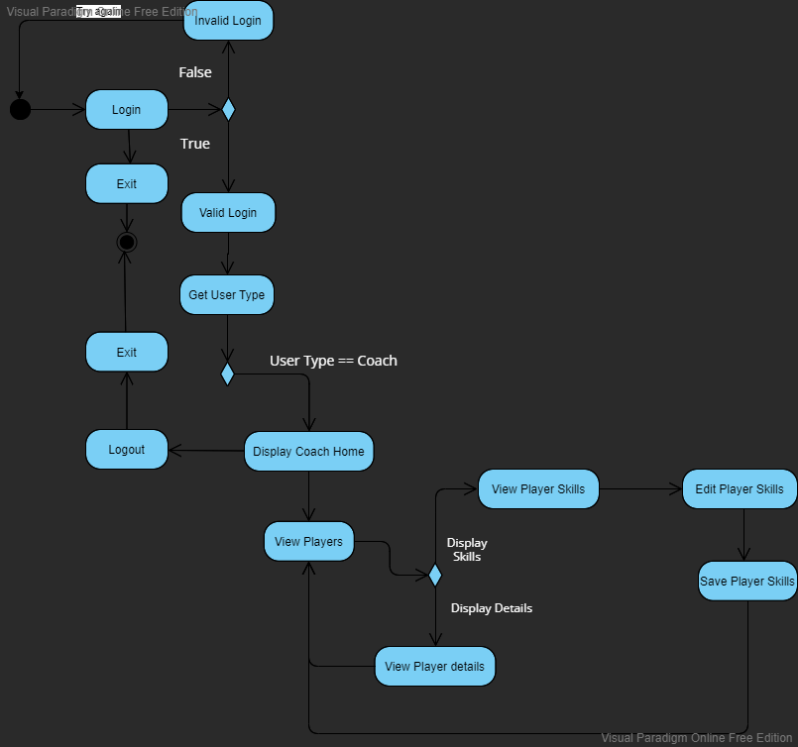
**Use Case Description – View Players**



Use Case Description – Editing Player Details (Admin)



**Activity Diagram – Coach Home Page**



**Sequence Diagram – Coach**

Diagram

Description automatically generated

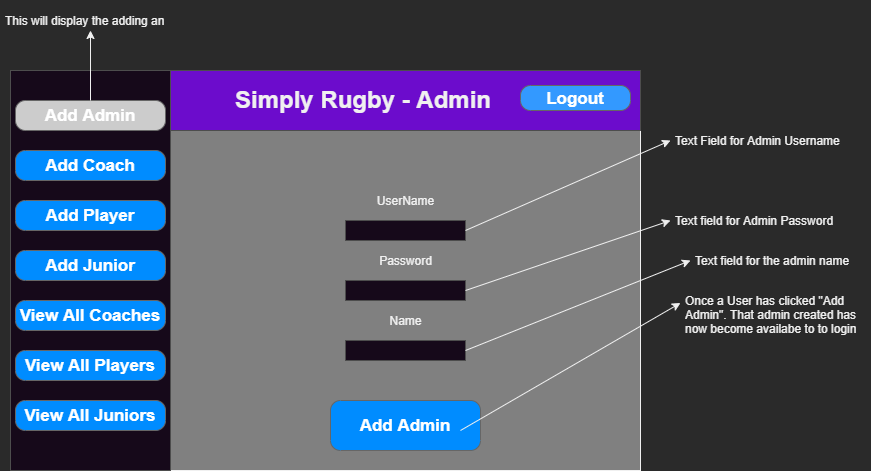
UI Design

**Wireframe – Login**



The overall design of this application will have any buttons on the left-hand side of the page. Keeping the overall design consistent for the user to get comfortable with. When *Login* is pressed, it will check to see whether an admin or coach is attempting to login into the system. If it is a coach, it will display the correct display based on which coach is a part of which team i.e., Under 18’s or Over 18’s. If it is an admin user, it will display the administrator view.

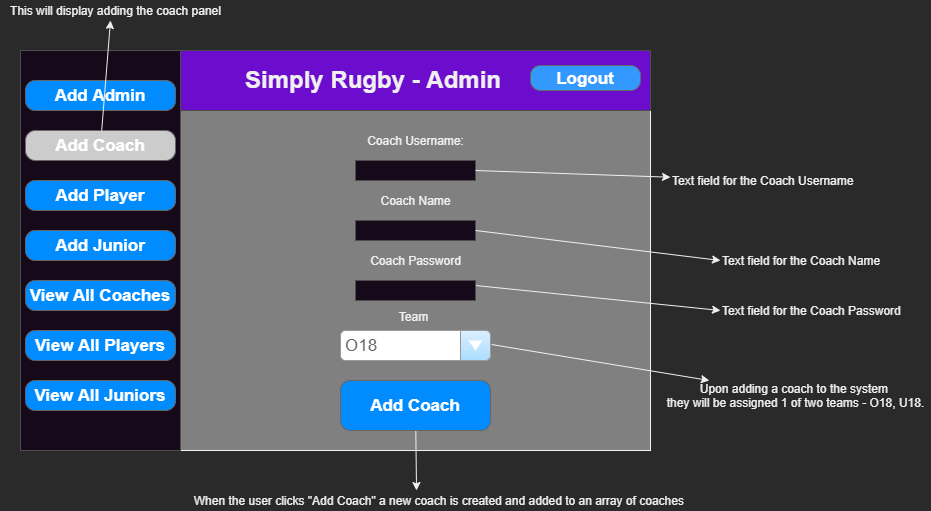
**Wire Frame - Add Admin**



When adding any type of person to the SimplyRugby Application, text fields will accept all forms of data – Integers, Strings etc... If the client decides to take this prototype further, implementation of stopping incorrect characters into a field will added.

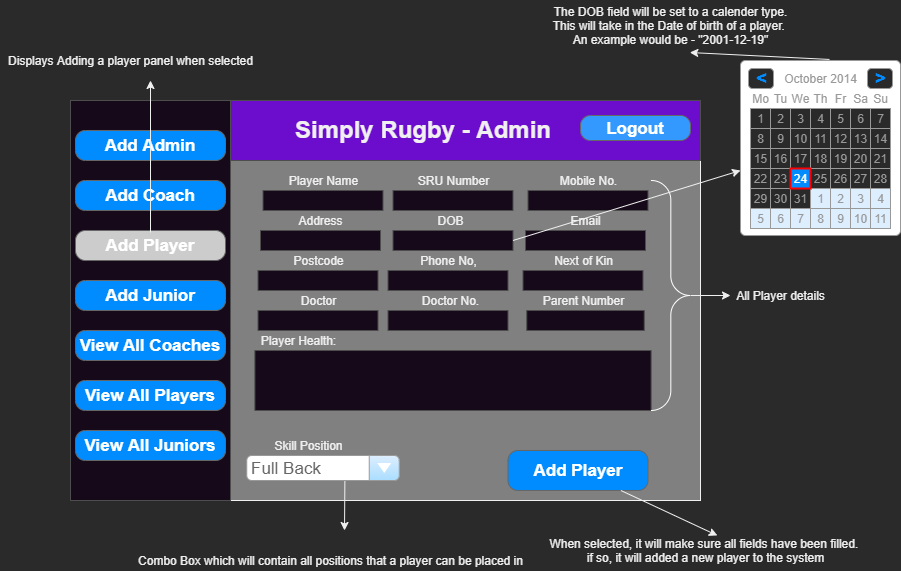
When adding an Admin to the system, it will be added to an ArrayList of Users which will be given a type of ***Admin***.

**Wireframe – Add Coach**

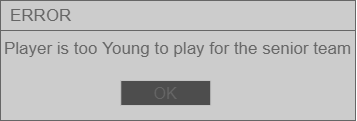
****

When adding a Coach to the system, it will work the same way as adding in an Admin. It will assign the type of ***Coach.*** The new coach will also be assigned to one of two teams. O18 or U18. If the client decides to go ahead with the prototype, add players and coach to a specific team can be added. How the implementation if it will work is noted below under [Design Decisions](#Coach)

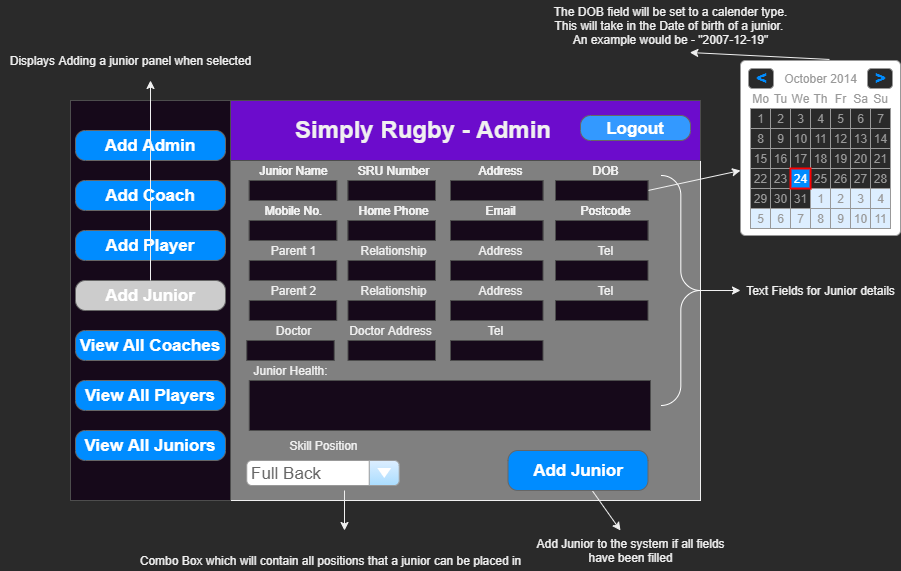
**Wireframe – Add Player**

****

When adding a player, the date of birth will be added through a calendar which the player can select the appropriate DOB for the player. If the DOB makes the player seventeen or younger, it will output an error message looking like this:

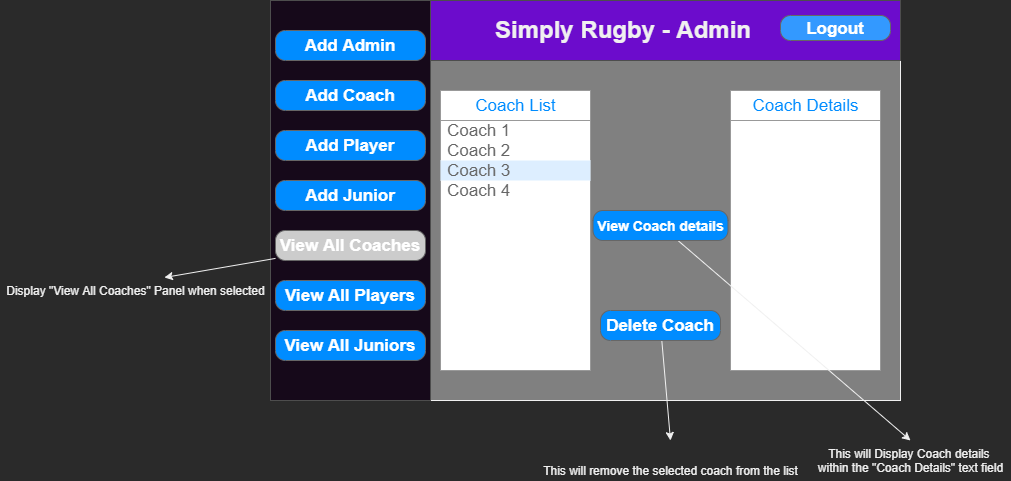


**Wireframe – Add Junior**

****

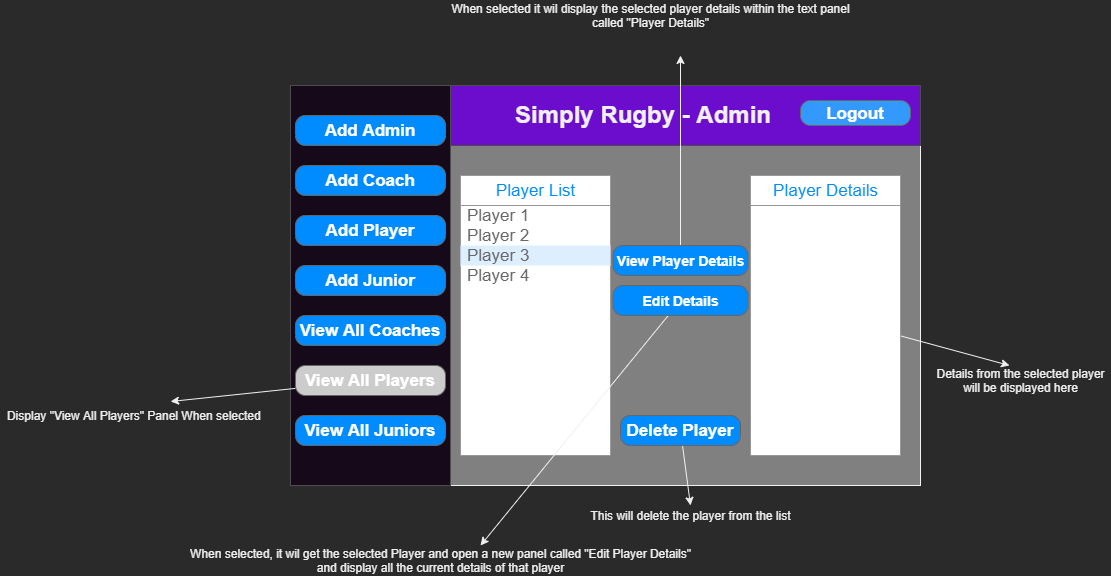
When adding in a junior, the DOB field will check to make sure the junior player is young enough to play for the junior team. The error message displayed will be the same as the player but set too “Junior is too old to play for the Junior team.”

**Wireframe – View All Coaches**

****

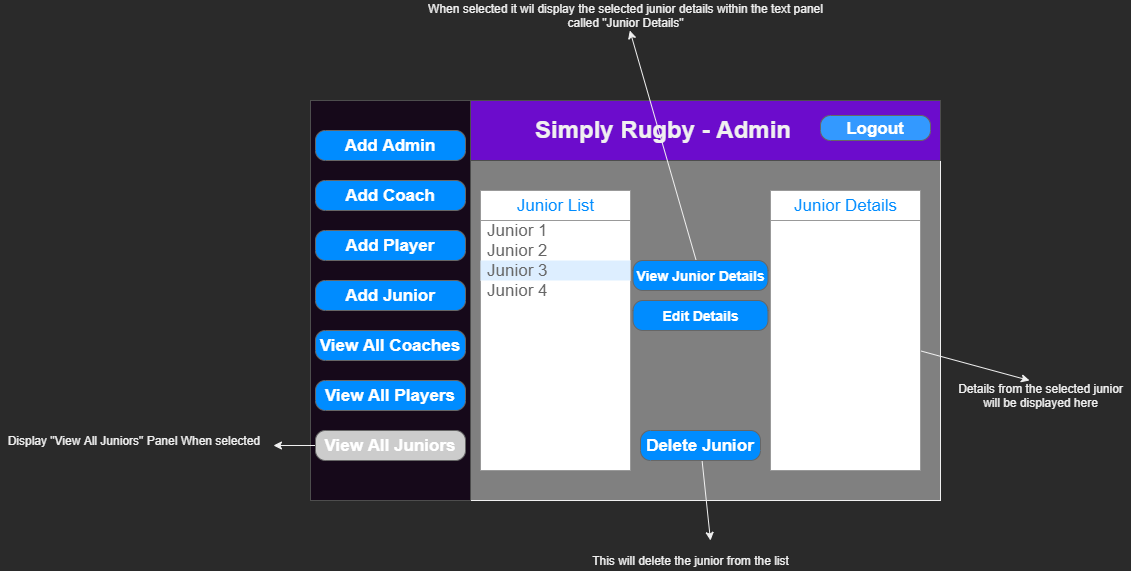
When viewing all Coaches within the system, the user can choose between viewing the coaches details or deleting that selected coach.

**Wireframe – View All Players**



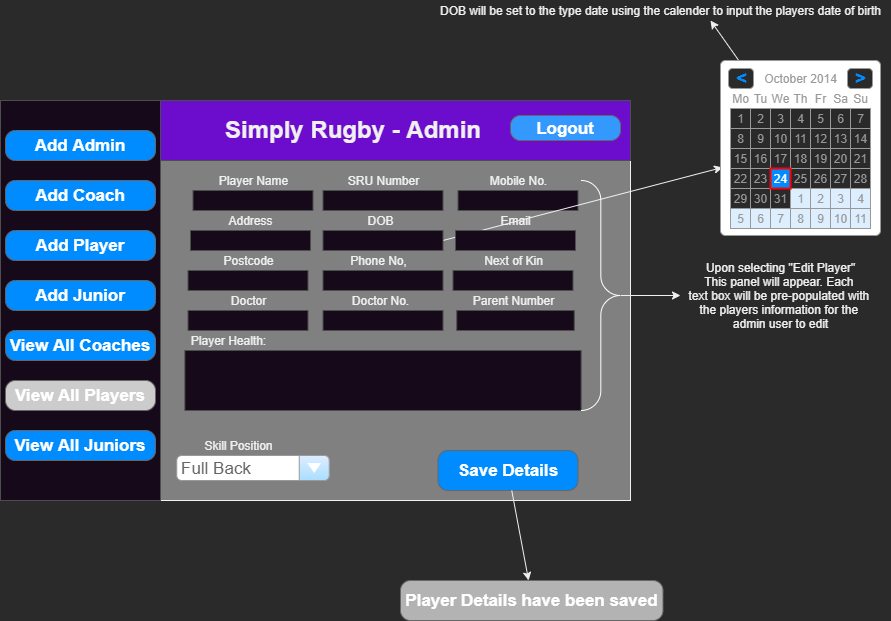
The functionality of viewing all players is the same as the coaches but the user is able to edit the player details.

**Wireframe – View All Juniors**

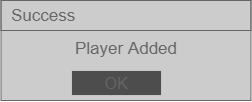
****

Functionality of view, editing and deleting a junior will work the same as the Player.

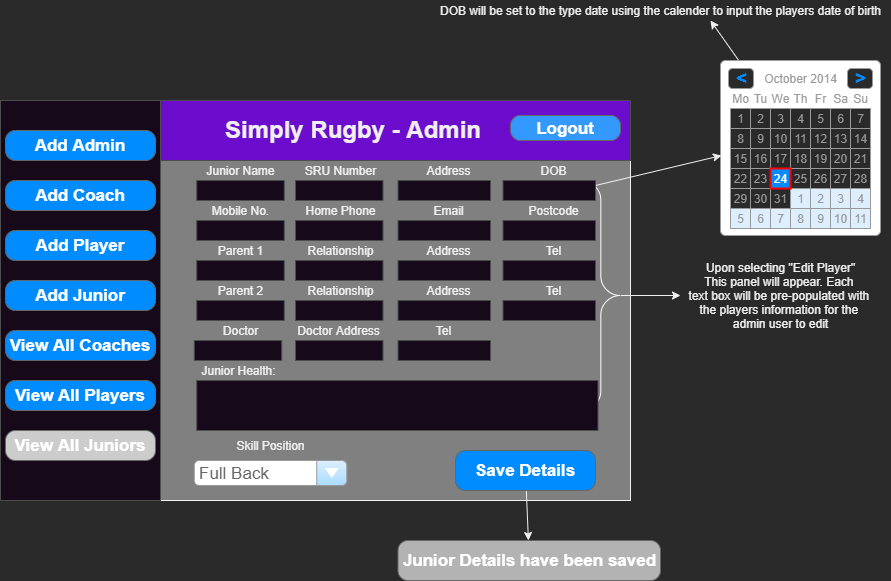
**Wireframe – Edit Player**

****

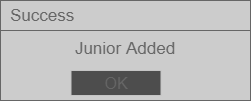
Upon selecting a player to edit all fields will be populated with the players data that was originally entered. Once the player details have been updated, it will save the details and output a message box like this:



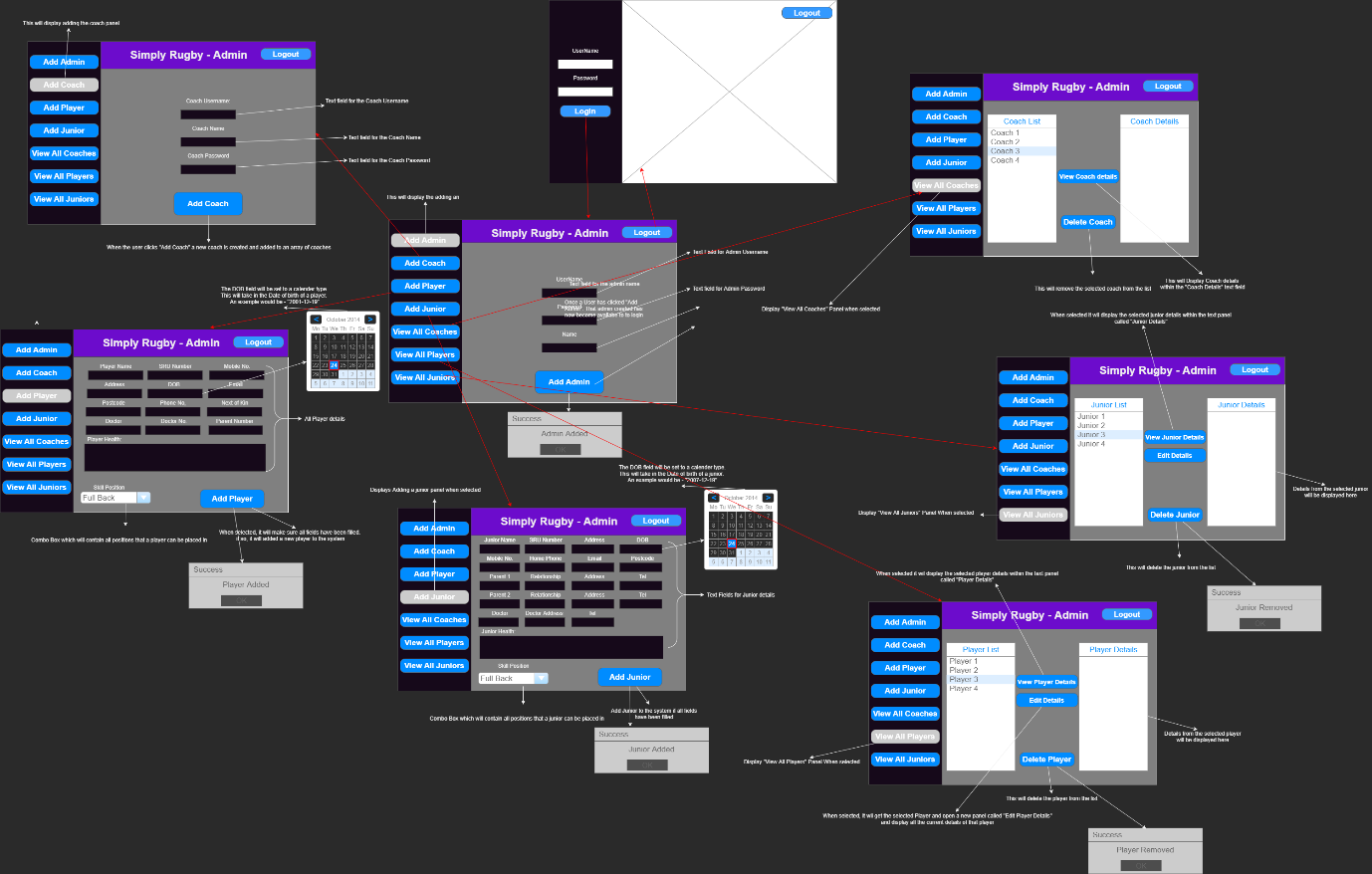
**Wireframe – Edit Junior**



The functionality of editing a junior is the same for editing a player. The message box shown if all the details have been updated correctly will look like this:

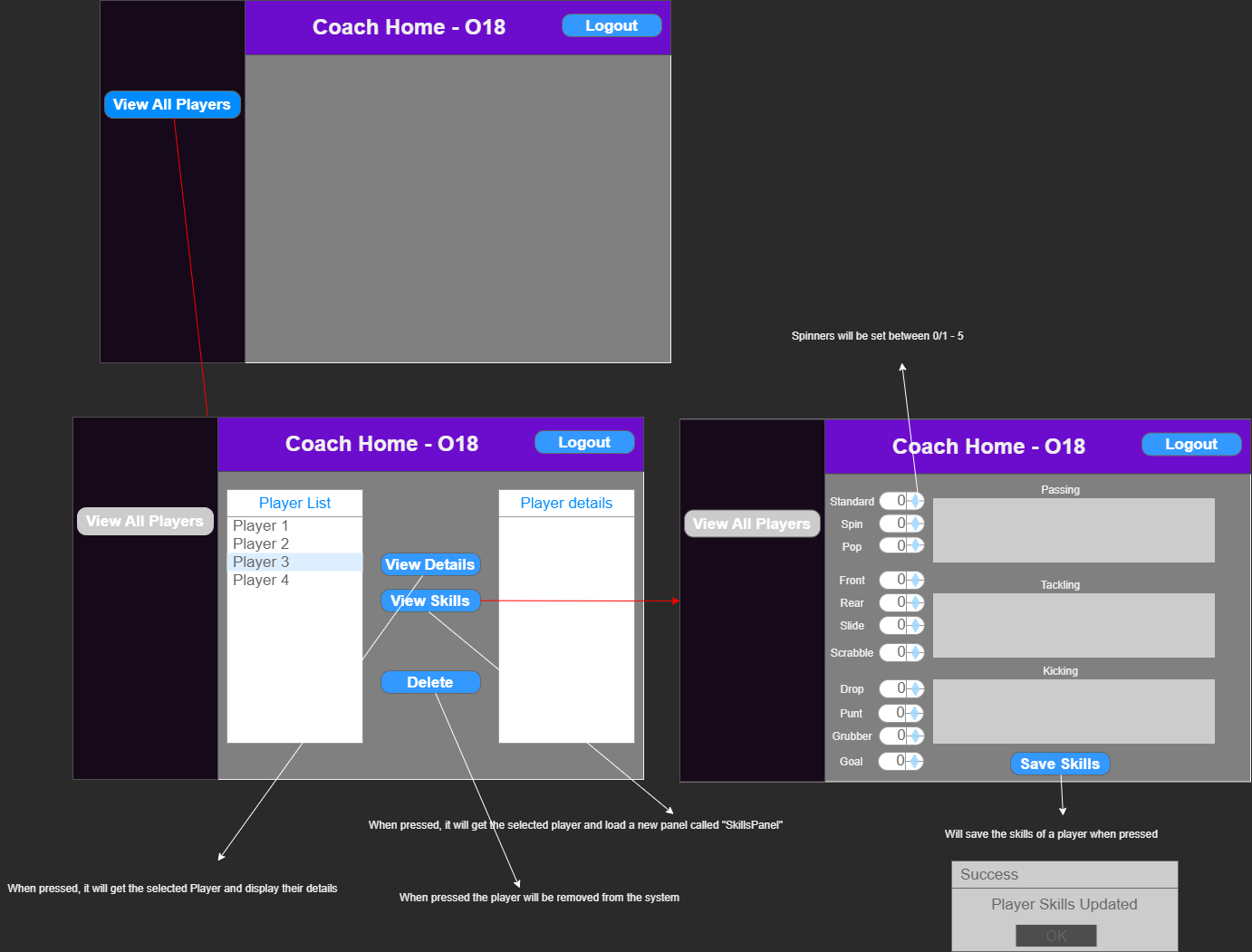


**Screen Flow Diagram – Admin Page (View Coach/Edit Players & Juniors)**

****

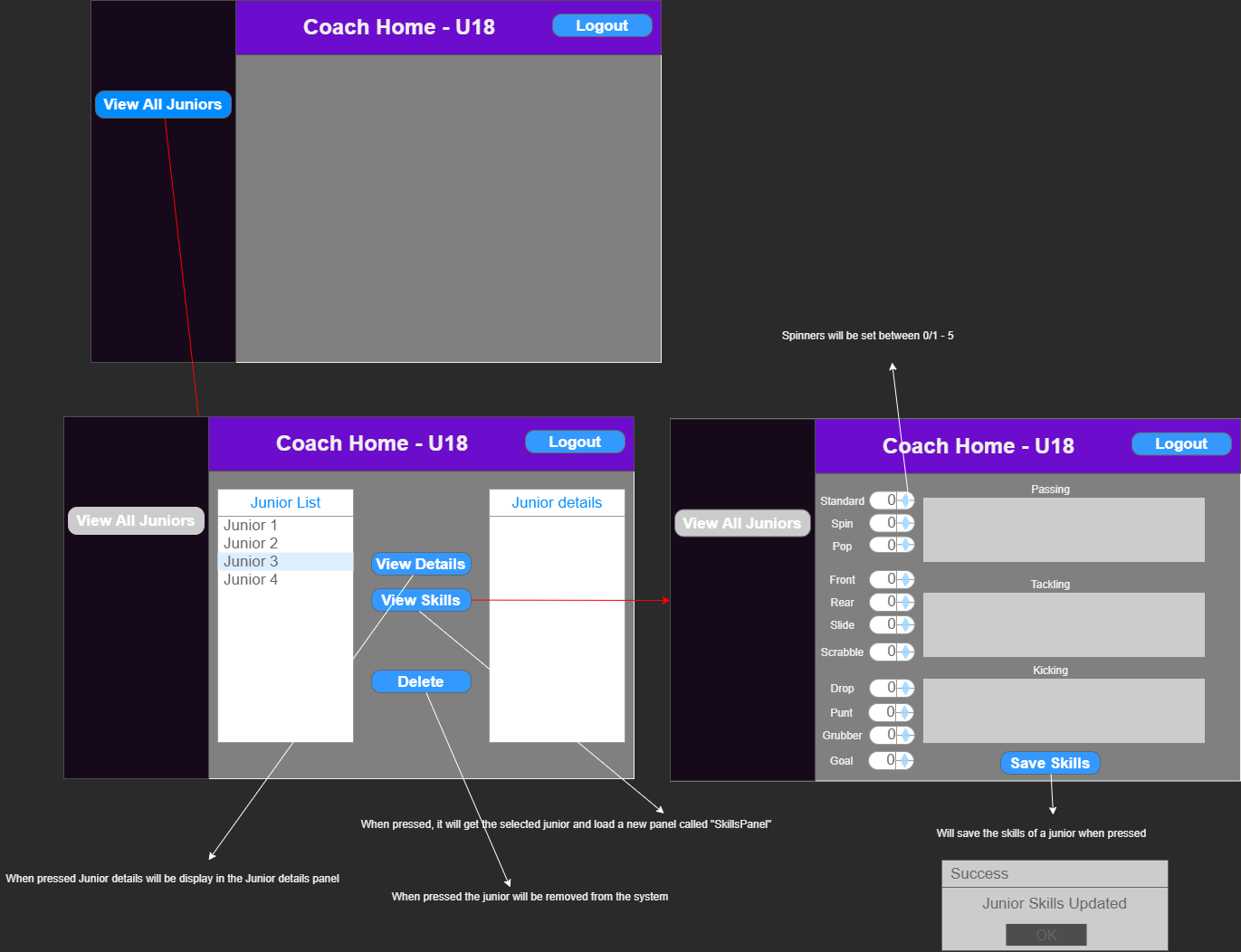
The flow diagram above shows how a user might intend on using the application and the appropriate screens showing which button will do what in the app.

**Screen Flow Diagram – Coach Page (Viewing details & Skills)**



When the Coach has selected a player, they would like to edit their skills, a new panel is displayed with all spinners set to 0 if the player is new or if they have not completed training just yet. The Coach can add notes if they would like to mention any new developments or uncertainties.

**Screen Flow Diagram – Coach Page (Junior)**



when a coach that is a part of the Junior squad. The display will change to display all the current juniors within the club. Coaches will then have a choice to either View details or their skills or delete that selected Junior from the system.

View Model

**Analysis of appropriate Techniques**

**Research**

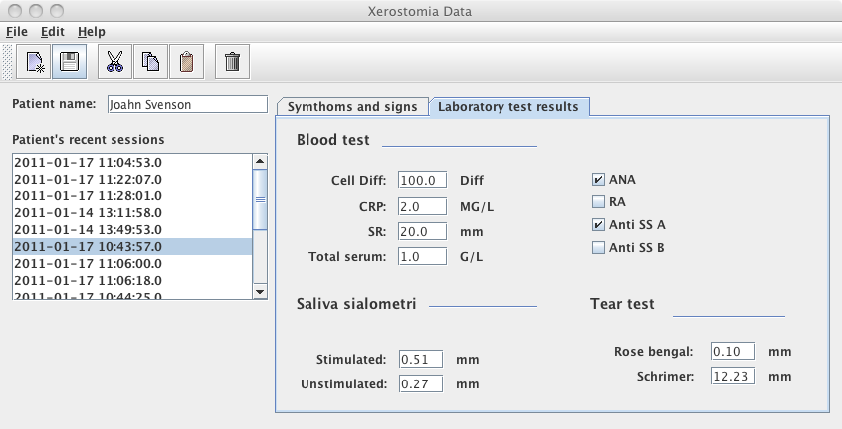
When given the project of designing an application for coaches and admin users to operate alongside their club, keeping track of players and their development while at the club. I decided that the best toolkit to use would be Java swing. Java Swing is GUI development kit which allow the development of GUI applications which can manage and manipulate data based on what the user is doing.

In order to fully understand Java swing and how it might be implemented I had to do some research into the development kit. This involved reading documentation from Java websites which can be found here: <https://docs.oracle.com/javase/tutorial/uiswing/components/index.html>

I also used some educational resources such as iLearn. A website which contained examples on Swing and how to implement items like a jList or a login system. This provided the proper information regarding Java Swing giving me confidence in knowing that this toolkit will help me deliver on what the client/Users require.

**Design Decisions – GUI Layout**

I decided to display all our data (Players, Juniors & Coaches) within jLists. Using a jList will allow me to use different jButtons like: “*View Player Details,” “Edit Player”* etc…

****This decision will allow users full freedom to select what they want to do with a player, Coach or Junior details or skills. This also makes it clear on which jButton does what within the system.

When Researching ideas on how I will display information to our users. I liked the idea of selecting a player from a list which would then display all the information about that player in fields allowing the admin to edit the player details. The example above shows this idea in how we can utilize jLists to view and selected data.

**Design Decisions – Admin**

I decided that the system will require two types of users. An Admin and a Coach user. An Admin user will be able to add, delete & edit players, Juniors. This will keep in line with data protection function as Coaches will only be able to view the data and not change anything about a particular player.

Once an Admin user has logged in, it will display numerous buttons which can be selected depending on what the administrator wants to do. This admin page will contain a consistent layout with buttons on the left side of the page and all the information will be displayed in the middle of the screen. This allows for a familiar layout allowing users to become comfortable using the system.

**Design Decisions – Coach**

When designing the coach layout, I decided that each coach will be assigned to an Over 18 or an Under 18 squad. When a coach logs into the app, it will display the appropriate squad. If the coach is assigned to the U18’s it will display all Juniors in the club where the coach can view their details but not edit them and also view and edit that player skills and add notes to each skill category. Another design decision was to only have coaches assigned to two types of teams. U18 and O18.

If the client decides to take forward this prototype adding a team can be added to the definitive version of the app. How this will be done is a new button which will take the admin user to a new panel which will contain a list of teams. The admin user can edit those team names if needed. Once a new team has been added it will be added to the list of teams. When adding a new player or coach, they can be assigned to that new squad. Then, once the coach of that team logins in, it will display all the players apart of that squad.

Due to Data protection act, I have decided to allow only certain users edit player/junior data. Only admin users within the system will be allowed to edit player data. This will stop coaches from being able to tamper with confidential information.

**Design Decisions – Adding User/Player/Junior**

When adding in a player, Junior, or User (Admin/Coach) any type of data can be placed into the fields. Due to keeping in line with HCI – Human Computer Interaction – Techniques. I have decided that if the client decides to take this prototype forward, error handling will be put into place in order to stop inappropriate or incorrect information from being entered and added/Saved