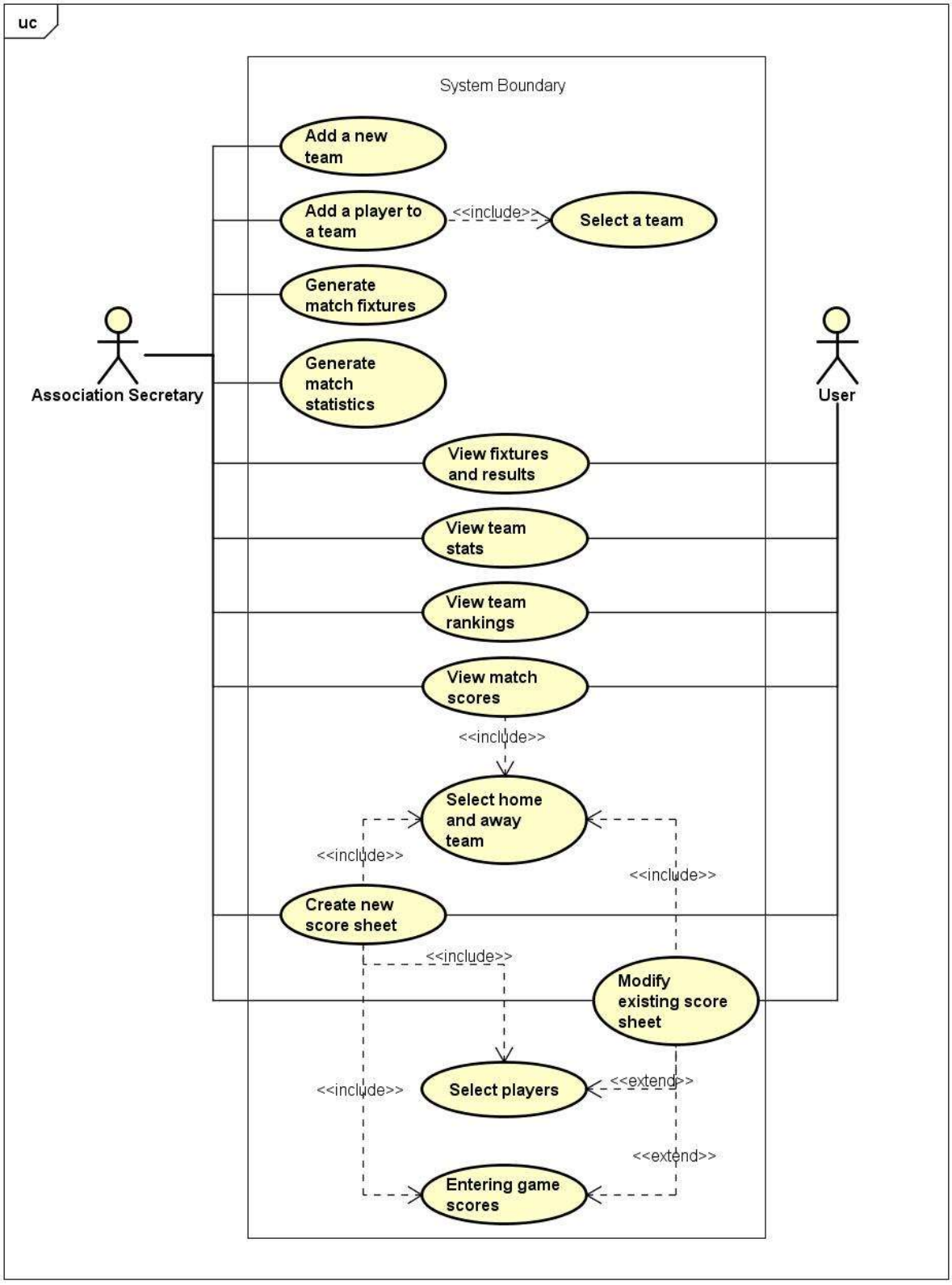
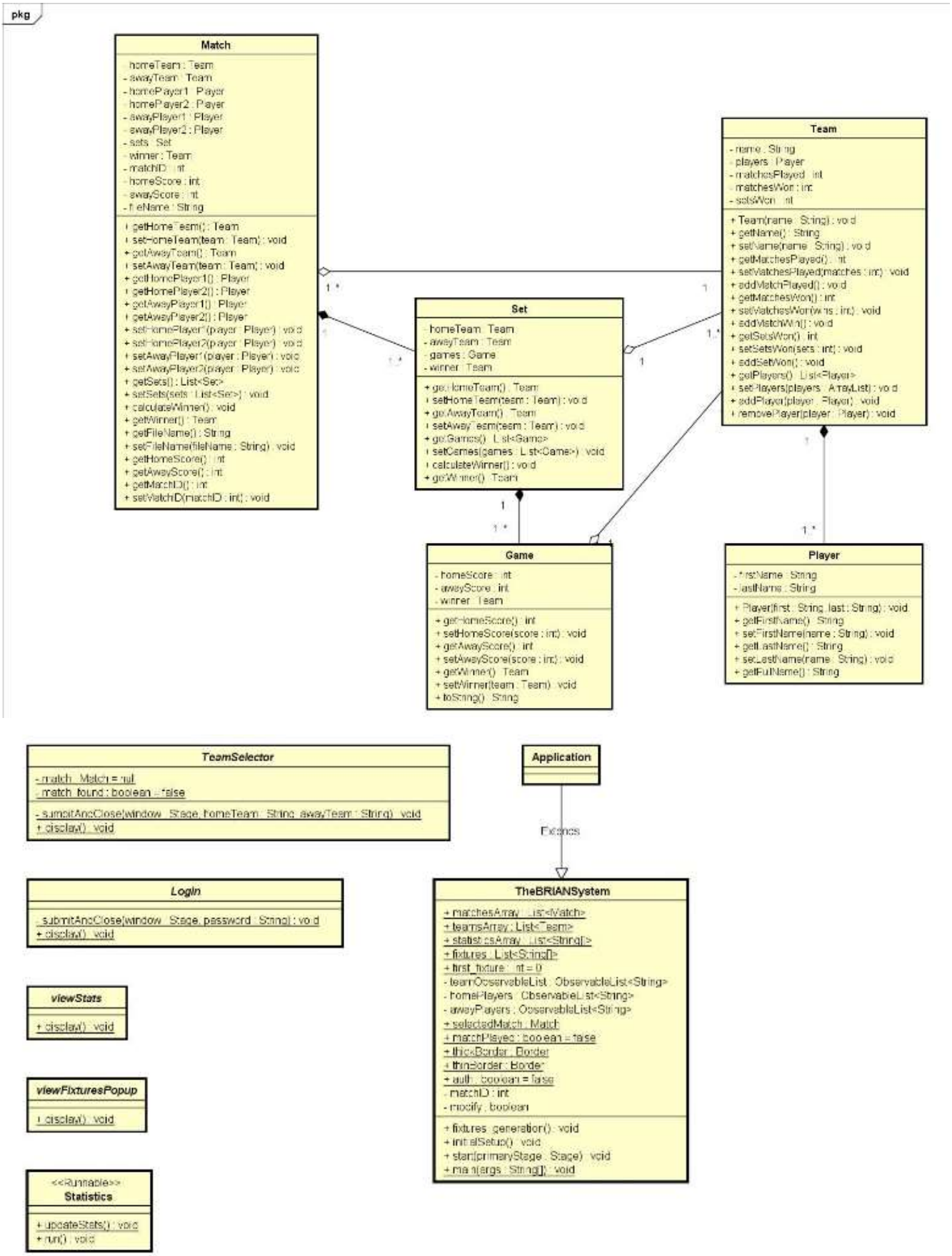
Design Diagrams

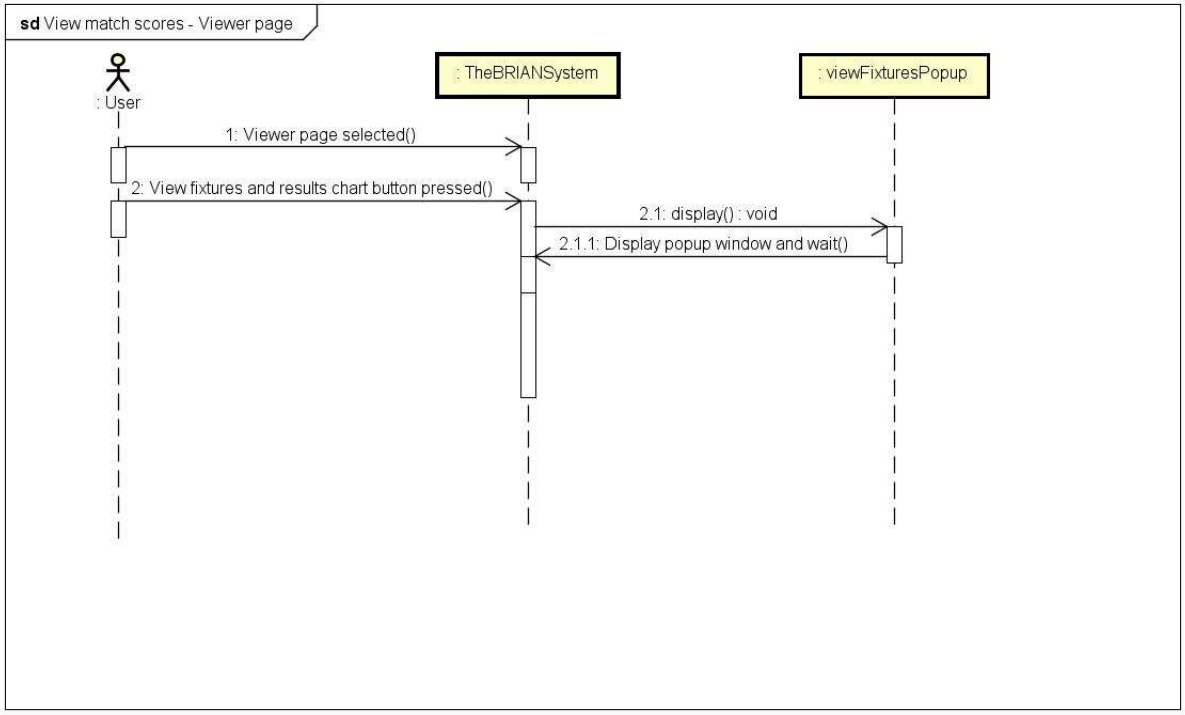
Use Case diagram:

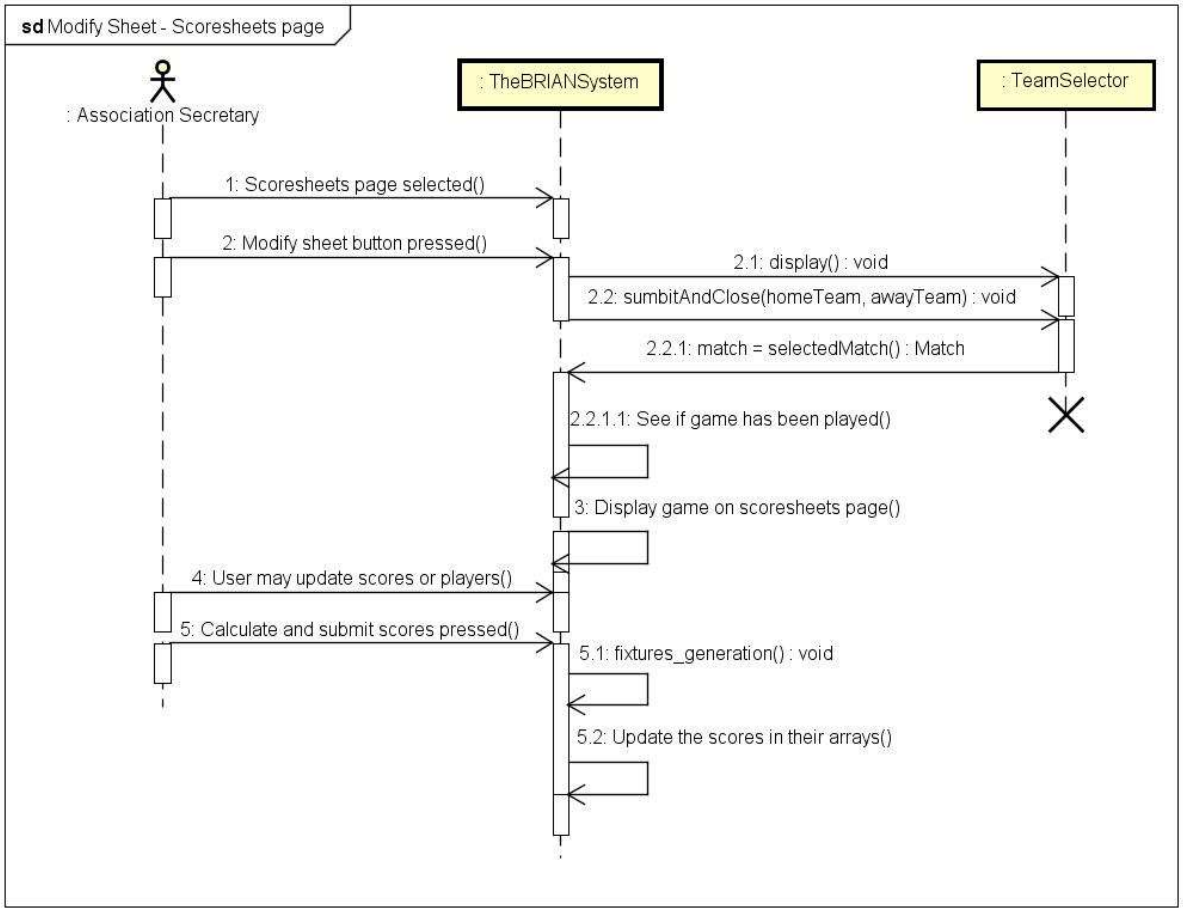


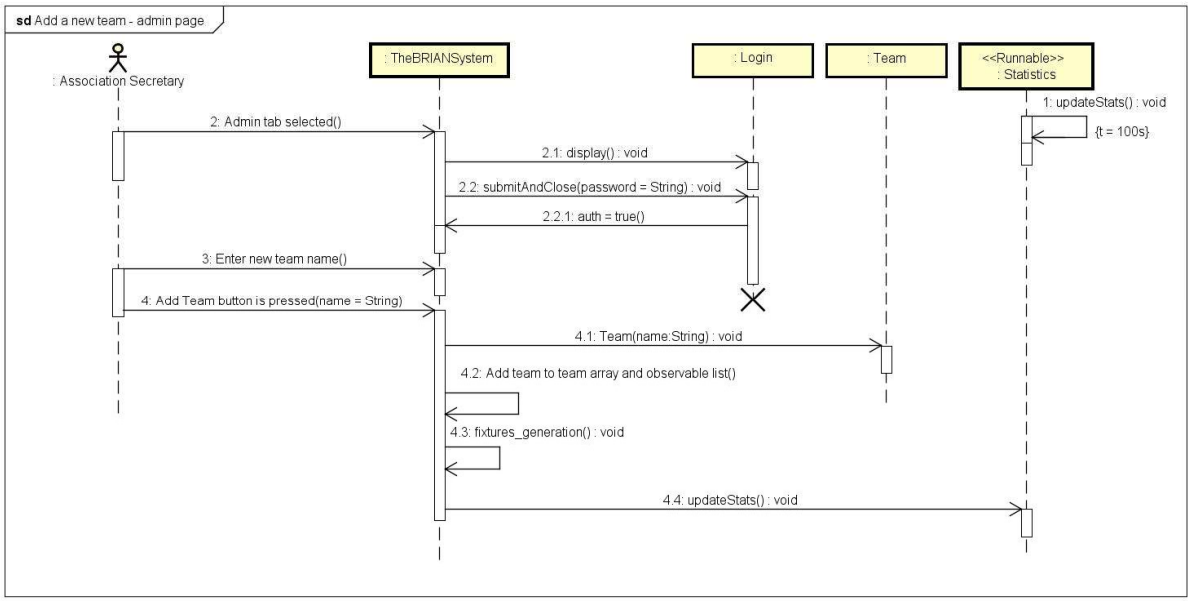
Class diagram:



Sequence diagrams:






sd Add a new team - admin page
: Association Secretary
: TheBRIANSystem
: Login
: Team
<<Runnable>>
: Statistics
1: updateStats() : void
{t = 100s}
2: Admin tab selected()
2.1: display() : void
2.2: submitAndClose(password = String) : void
2.2.1: auth = true()
3: Enter new team name()
4: Add Team button is pressed(name = String)
4.1: Team(name:String) : void
4.2: Add team to team array and observable list()
4.3: fixtures_generation() : void
4.4: updateStats() : void

Agile Development Episode Outline

1. Feasibility and requirements

At first we looked at the project specification provided to us and reviewed how we could implement it. We identified the users who were going to be using the system, which boiled down to simply be the administrator and the users/players. We saw it needed to fulfil the main criteria, with 3 main tabs of Viewer, Scoresheets and admin.

2. Planning

For this part of the process, we started to plan how we were going to implement the data in their various classes. We met up in person and discussed using UML diagrams, how we were going to represent the data in code form from what is specified in the specification. Eventually we settled on an overall design and began work on this area. After this, we discussed what needed to be done in terms of prioritisation, as no particular area needed to be done first, we decided that functionality was going to be more important than aesthetics and that creating it from scratch in JavaFX would better than using the FXML editor.

3. Development and iterations

At this stage, we start our development iteration process, this includes finalising our design plans and beginning to code up the project. As we worked with the project and discovered issues, we talked amongst ourselves, be it online or in person to see what we could do about it to sort the issue. For example, we needed to include an ID for the matches so we could identify which match was which, therefore updating the plan after we had begun. This more fluid system of agile development allowed us to develop the program at our own pace whilst also allowing for changes as we implemented it. We maintained daily communications on what we worked on and what our issues were in a verbal and online manner.

4. Adaptation

As we developed the product, we looked back at the specification to see whether we were creating something in line with what we needed to make and adjusting where needed to suit this. At one point we had a standing meeting on our landing to discuss how we were getting on and what we had left to complete before the menu was complete.

5. Simplicity

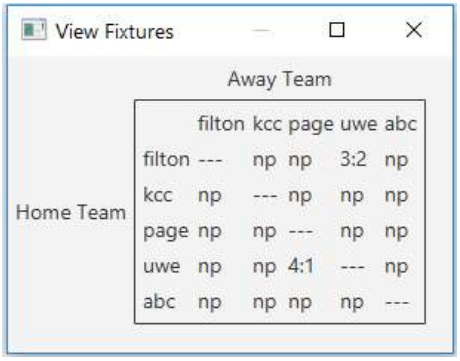
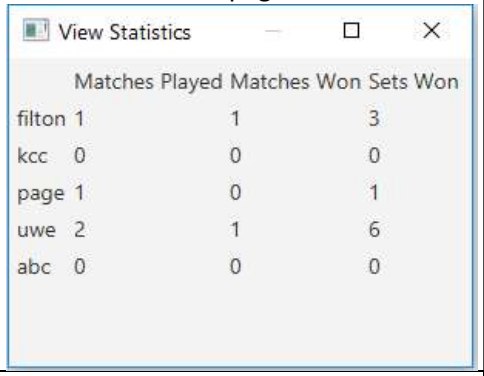
One part of our project we have found that having a simple GUI for the user makes things a lot easier as they are able to identify any issues which the design may have. We experimented with other designs and found that keeping it minimalistic was preferable. Further to improve the adaptability of the code we have kept it as simple as possible in order allow for simple and easy changes to our code. We have modularised the classes in our code as much as possible, making use of the object orientated process by encapsulation in order to split our classes from the main code to make them easily maintainable.

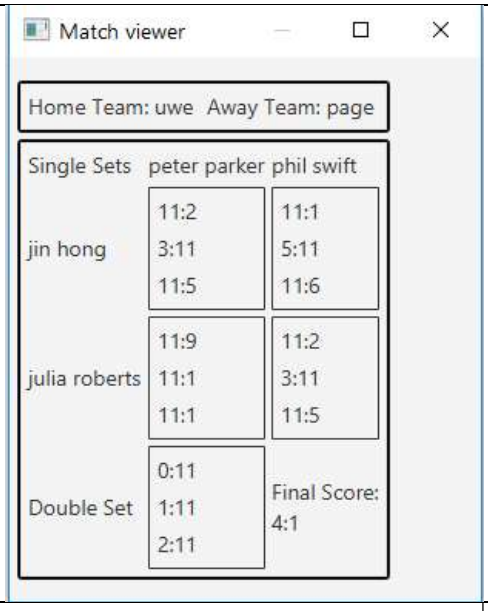
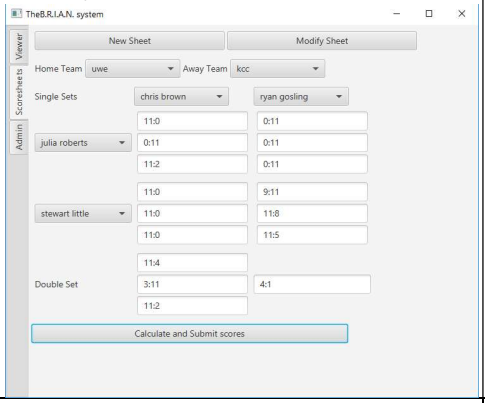
6. Deployment

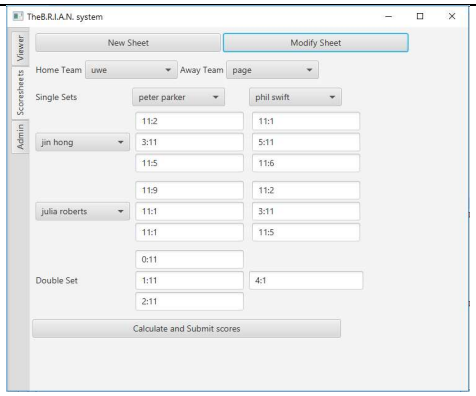
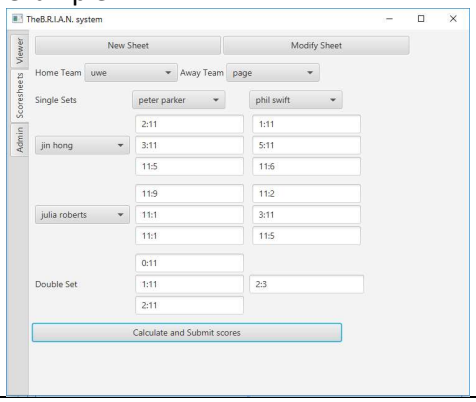
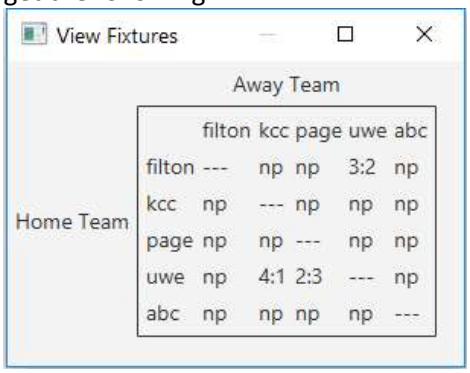
As we reached the end of our development cycle, we looked back at the specification one last time to make sure we had included everything we needed to implement, whilst also making sure

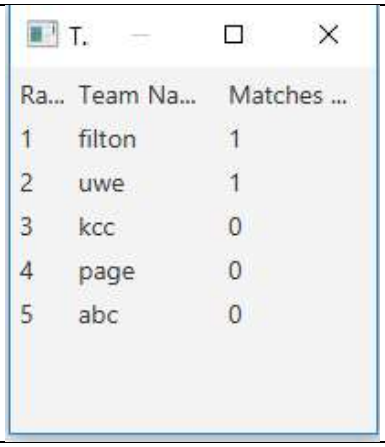
everything was working as intended by using test cases to test various areas of the code. Furthermore, we looked back at our UML diagrams, making sure the use case, class and sequence diagrams were all correct and displaying what was actually going on in the project correctly.

Test Cases

Test Case	State of the System	Input	Expected Output	Actual Output
1. Add a new team	The required test data are in the system	Enter a new team name, (e.g. team “abc”) and click the add team button	No error, confirmation message	As expected
2. Add a new player to a team	The required test data are in the system	Enter a player name, (may need to enter the return key) Select an existing team, Click the register player button	No error, The select team drop down box should have all the pre-entered team names + plus the team added test case 1, Player added to team successfully	As expected
3. Generate fixtures	The required test data (teams) are in the system, a new team was added (e.g. team “abc”)	Click the generate fixtures button, And then go to the viewer’s page and click on “view fixtures and result chart”	The fixture should be generated as shown: 	As expected
4. Show all team stats	The required test data are in the system, a new team was added (e.g. team “abc”), Fixtures is generated	Click the show all team stats button	The following is displayed in the text area in the viewer page: 	As expected
5. View a match		Enter home team and away team, e.g. uwe and page	The following is displayed in the text area in the viewer page:	As expected

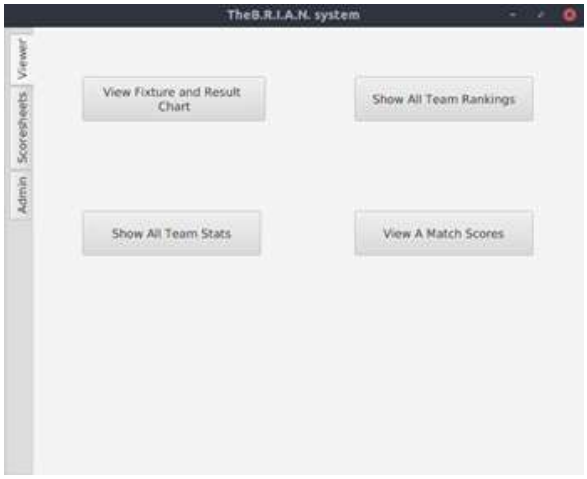
				
6. Enter new scores		Click the “New sheet” button in the “Score sheet” tab, Select a home team and an away team from the drop down boxes Select suitable players from the drop down boxes Enter the points	All registered teams should be in the team dropdown boxes. Players from the right team should be listed in each of player drop down box.	As expected
7. Calculate and submit scores	Test case 6 is successfully completed	Click the “Calculate and submit scores” button	The final score should be displayed in the text field at the bottom right corner, as such: 	As expected
8. Modify an existing score sheet	Modify an existing score sheet	Click the “Modify sheet” button in the “score sheet” tab System prompts for	It should bring up the score sheet as follows:	As expected

		home team name, user enter home team System prompts for away team name, user enter away team System brings up the score sheet (example shown on the right) Change some scores and click the "Calculate and submit scores" button	 It should revise the scores, for example:  	
9. View fixtures and result chart	Test cases 7 and 8 are completed	Click on "View fixtures and result chart" button	The result chart should show the changes, for example if we use the data entered in test cases 7 and 8, we get the following:  	As expected
10. Show team ranking	Team stats need to be generated since the changes made in test cases 7, 8 and 9. This can be done in two ways: click the "Generate team stats" button in the Admin tab or wait for the		It should display the teams in the order of the number of sets won:	As expected

	reporting timer thread to generate it automatically.				

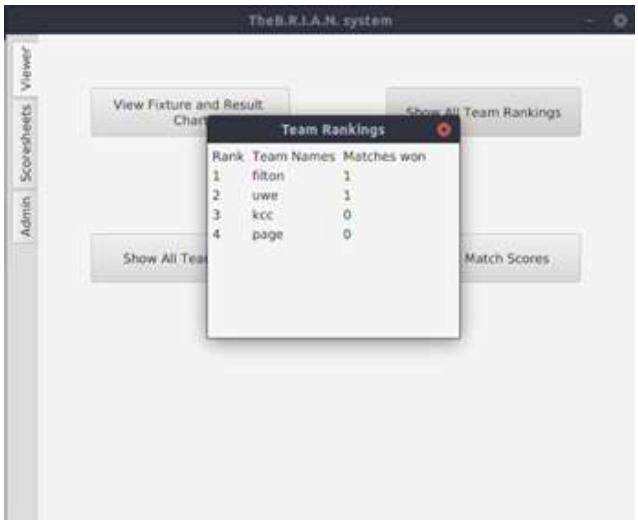
Screenshots of Success and Failure

Viewer Page:



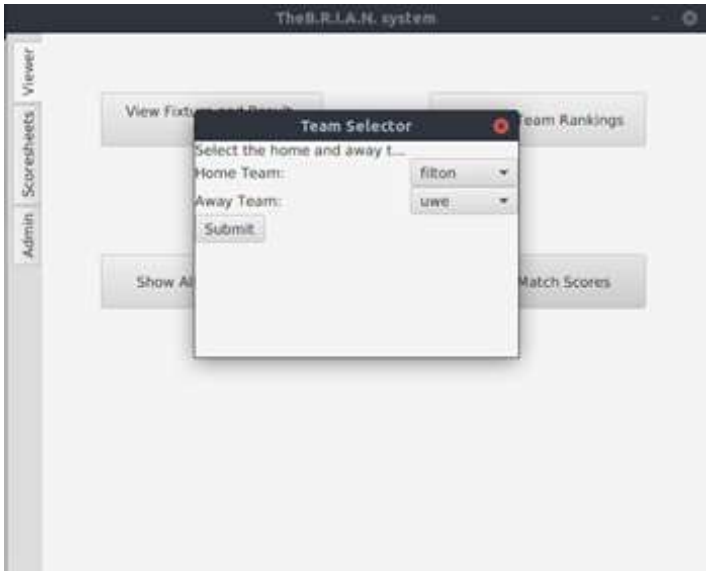
View fixtures:

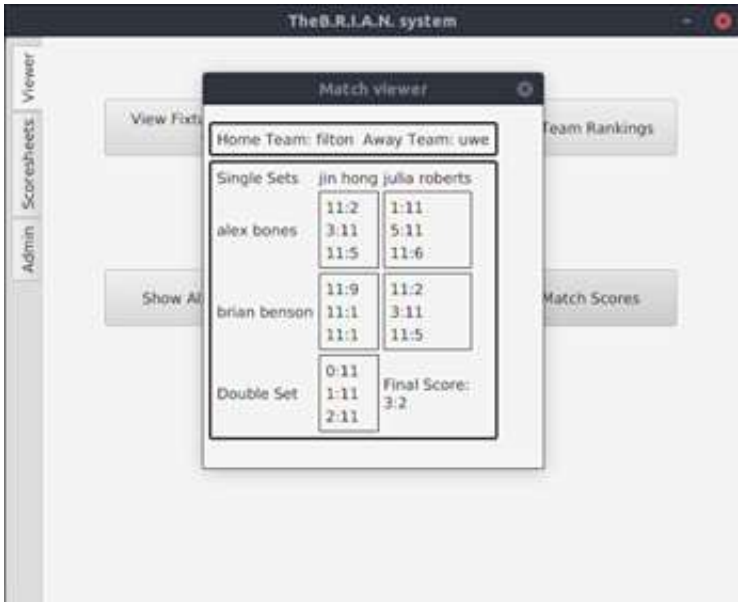
Team rankings:



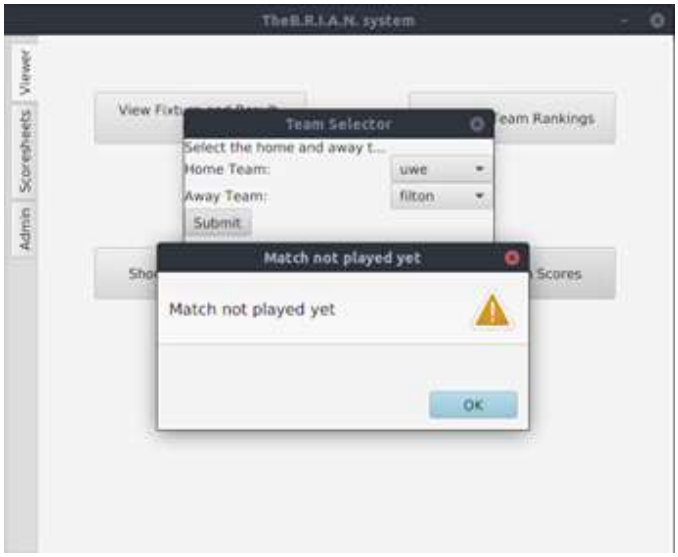
View statistics:

View match scores (using team selector):

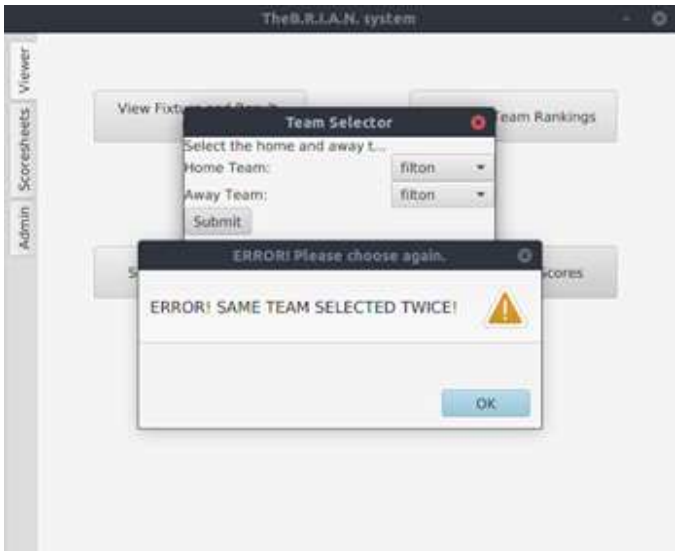




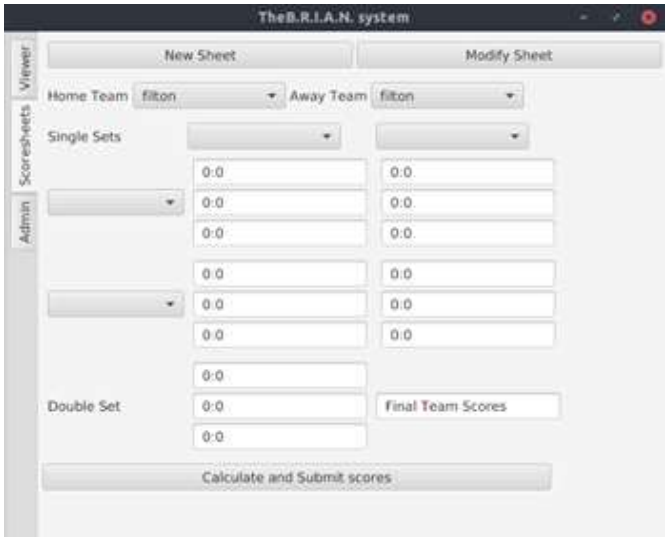
Error if match not played yet:



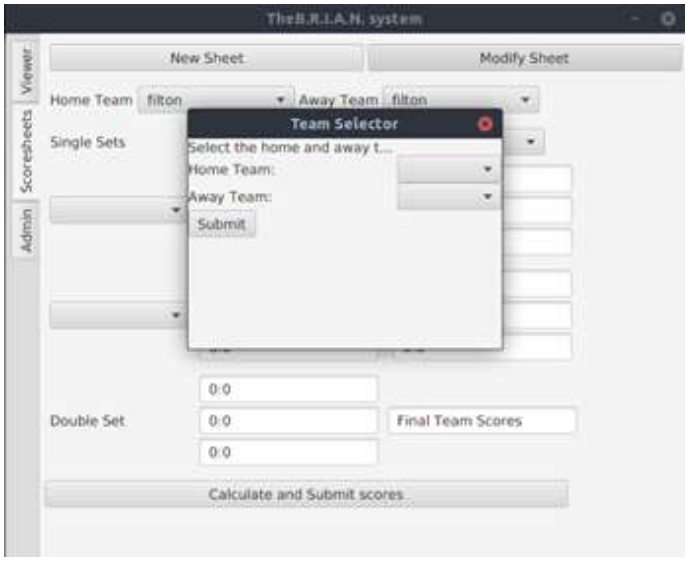
Error if same team selected:

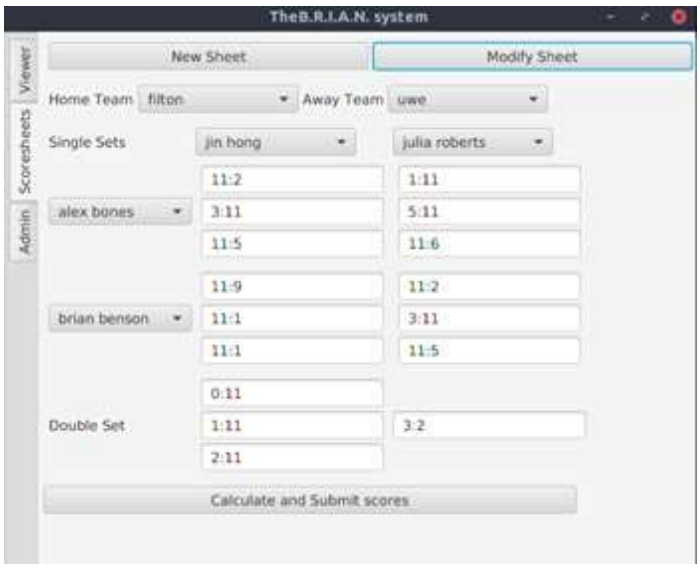


Scoresheets Page:

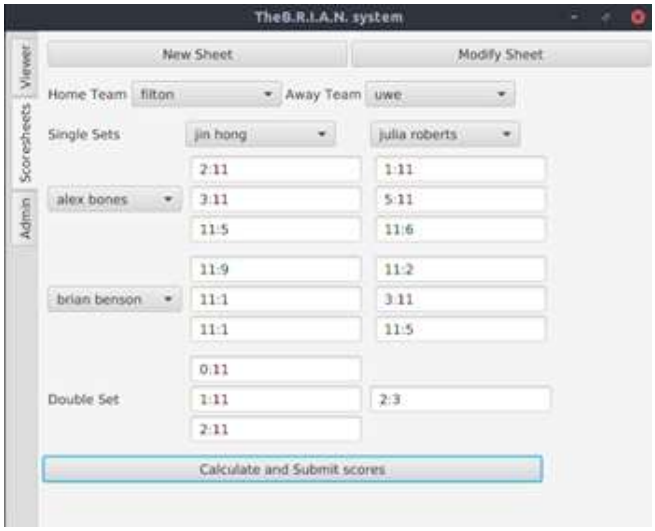


Modify sheet:

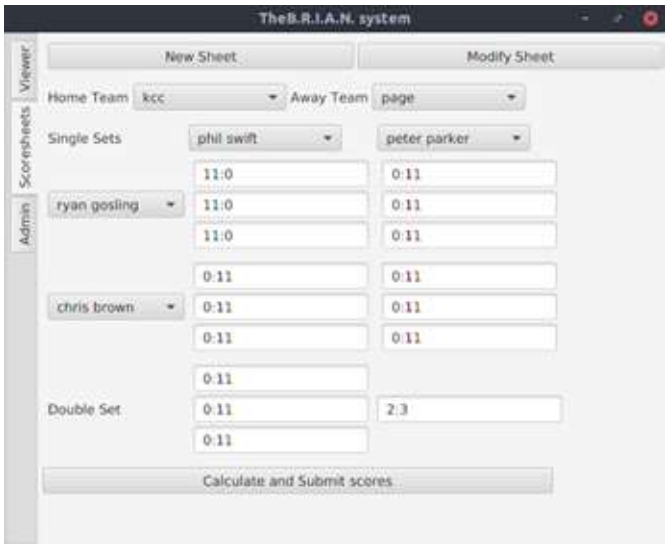




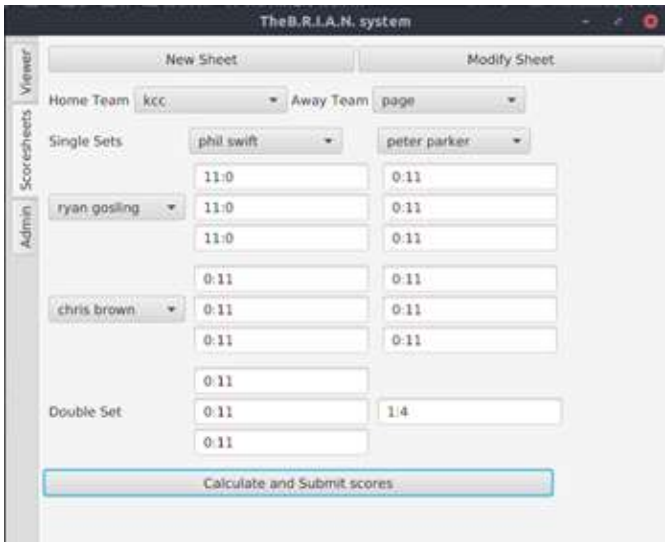
Change sheet and submit:



New sheet:

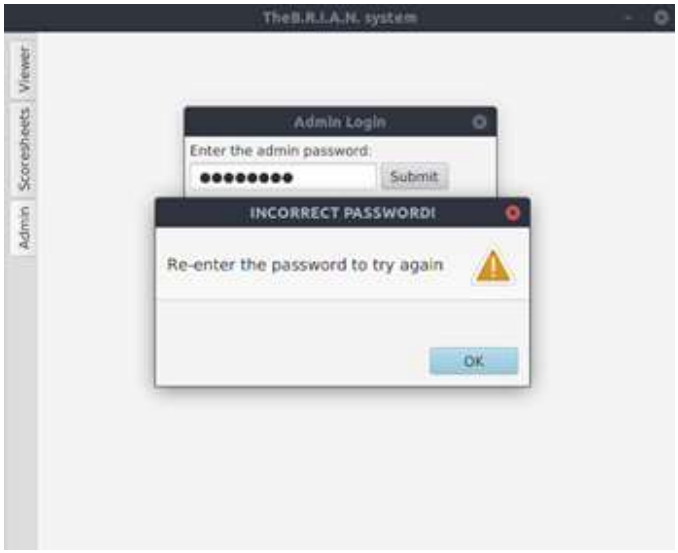


Test calculate and submit button again:



Admin page login:

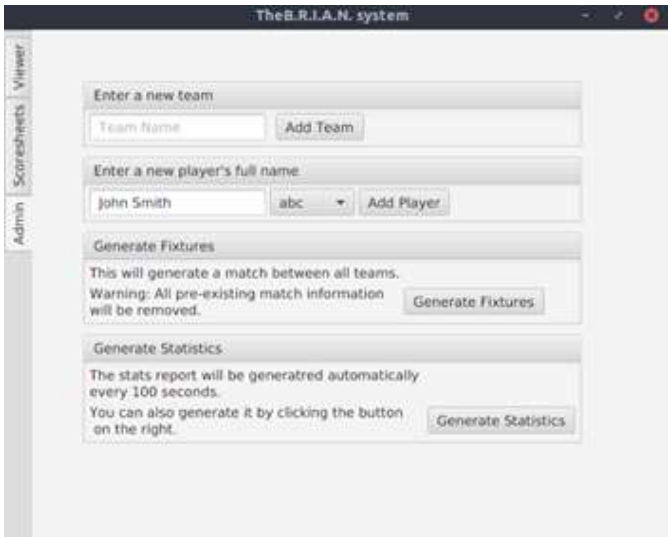
Admin page incorrect login:



Admin page:

Adding team:

Adding player:



Error in name: