UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

COS 301 - Software Engineering

🏈 Nimbus 🏈
Functional Requirements

Authors:	Student number:
Jedd Schneier	u13133064
Daniel King	u13307607
Muller Potgieter	u12003672

October 18, 2016

Software Requirements Specification and Technology Neutral Process Design

Nimbus AWS Network Visualiser/Main Project

Version: Version 1.0 Beta For further references see gitHub. October 18, 2016

Contents

1 Functional requirements

1.1 Introduction

The Nimbus Amazon Web Services (AWS) network visualiser will be used to visualise a user's network within the AWS network, through their browser. The purpose of this document is to identify and explain all possible use cases associated with the visualiser and to show how the functional aspects of the visualiser interact with each other.

1.2 Use case prioritiation

Critical

- Log in/Log out
- Scan network
- Visualise network
- Get Node Connections
- Get Node information

Important

- Stop scan
- Resume scan
- Scan Up
- Scan Region
- Scan From
- Scan Instances

Nice-To-Have

- Load scan from local .json
- Save scan to local .json

1.3   Use case/Service contracts

Use Case	Pre Condition	Post Condition	Description
Log in/log out	The visualiser has to be connected to the internet in order to verify the information. Only a registered AWS user with a valid key and secret key may log into the system. Once the user is logged in he/she can then use the logout functionality to log out.	The user is logged in now and may begin making use of the visualisation server.	This use case provides a method for logging in to view a hierarchical representation of their network and log out once done.
Scan network	The visualiser must be connected to the internet in order to load the information from the server. The user must have instances in their network, that the server may scan.	The server continuously sends information of the instances to the browser, which are then stored by the browser.	This use case provides a method for loading the network representation from the AWS network and storing it on a browser.
Visualise Network	The visualiser must be connected to the internet in order to load the information from the server. The user must have instances in their network, that the server may scan. The browser must have a feed from the server, that contain instance information.	As new node information is loaded into the browser, the browser will sequentially display them in a hierarchy.	This use case provides a method for reading information the browser has received and visualising it.
Stop scan	The scan must be active.	The scan's execution is temporarily halted.	This use case provides a method for temporarily halting an active scan.
Resume scan	An active scan must have been stopped.	The scan resumes its execution.	This use case provides a method for resuming a previously halted scan.

Scan Up	The scan must be active.	The direction of the scan is altered	This use case provides a way of changing the direction of the scan.
Scan Region	The scan must be active.	The scan refocuses and only scans instances that fall below a certain region.	This use case provides a way to only scan instances that belong to a specific AWS region.
Load scan from local .json	A .json file with the correct formatting must be stored on the local device.	The browser processes the information on the .json and visualises it, as it would with a normal scan.	This case provides a way load network visualisations, without the need to scan it from the server.
Save scan to local .json	The scan must be active/ finished.	A window appears that will save a .json file to the local device. The file is named with a time stamp, in order to avoid issues if multiple files are saved	This case provides a way to save a representation of the current scan, that can be loaded at a later date.
Security Groups	The scan must be active or finished. The browser must have a connection to the server.	The information box on the browser is loaded with information related to the selected instance.	This case provides a way to see information related to a specific instance.
Node information	The app must be aware that a match is currently being played and which teams are currently playing (individual players as well).	The name of the player who performed the turn-over must be logged.	This use case provides a way of quickly logging information about turn-overs.

1.4 Required functionality

- **Log in/log out**

  The login/logout will be used to let people log in that is an
  administrator/privileged user of a rugby page on the onlyRugby site. This will
  verify if the user has the right credentials to be scoring for a team. Also check
  the rugby page to see if there are any upcoming games the team will be playing
  to record the details of the match.
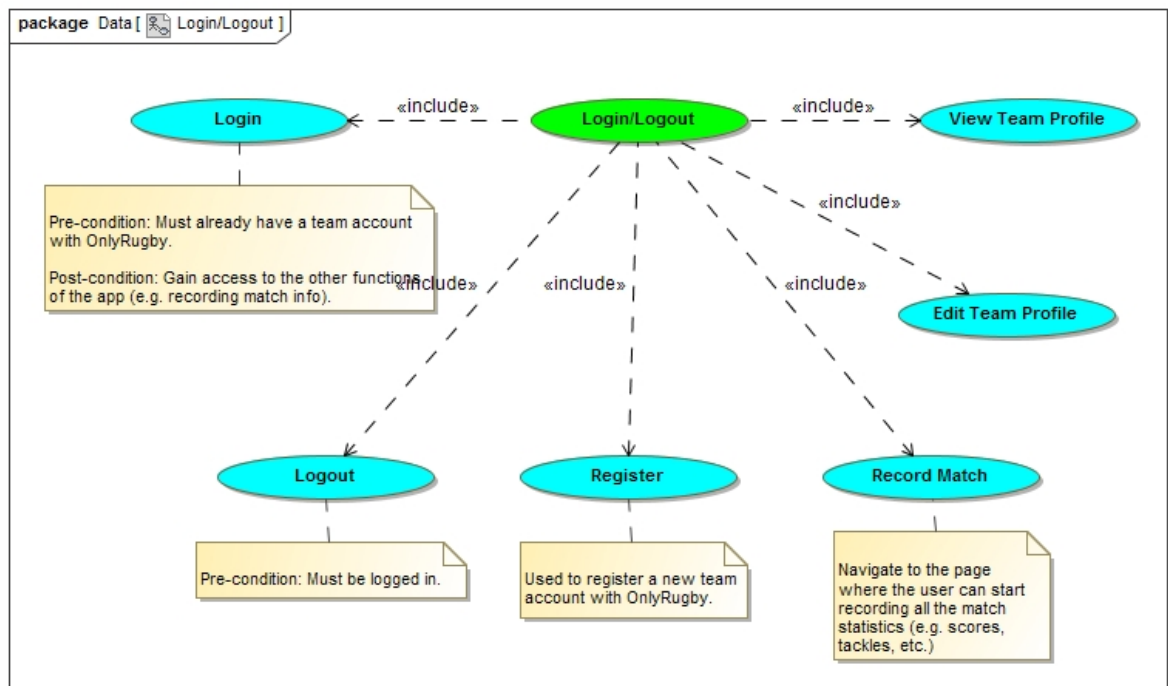
- **Load info**

  The Load Info module will be used to transfer information to and from the
  database, using the server. All destinations are to be verified before attempting
  to access them and incoming connections to the server need to be verified that
  they are from a trusted source.
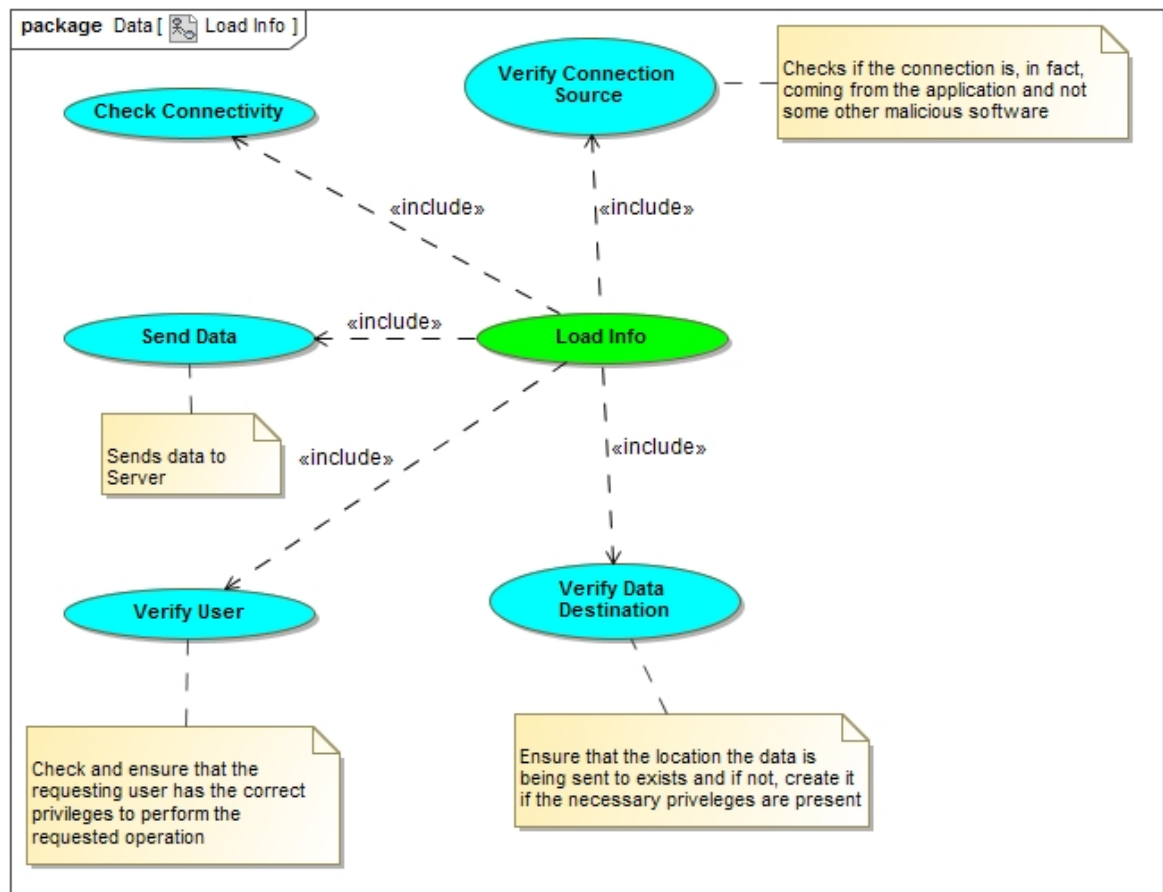
- **Game time**

  This use case will be used to log the start and end time of each half of a match, as well as any intervals during which time was lost (the game was paused) and a reason for this time loss (injury, substitution, referee consultation, replacement of damaged player clothing).
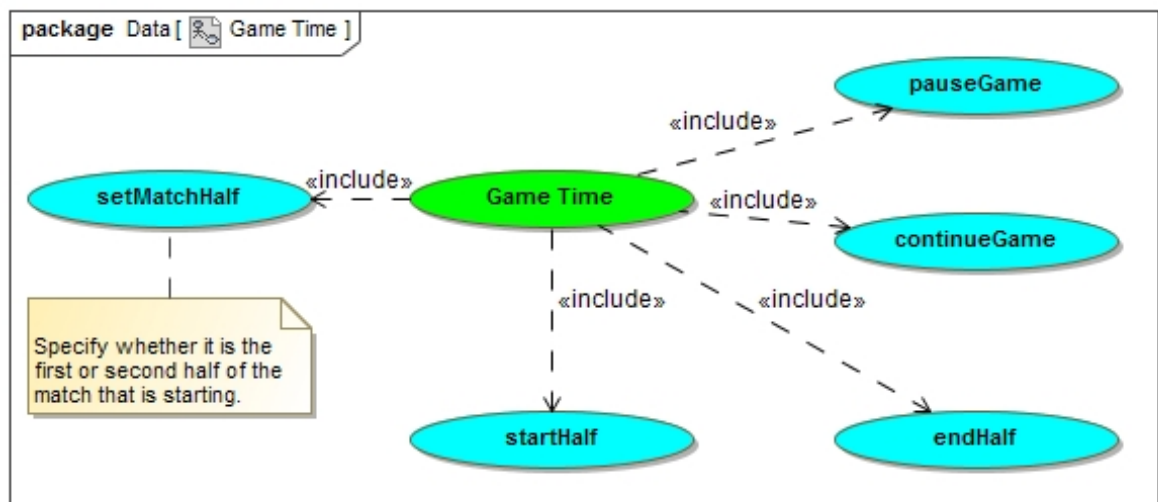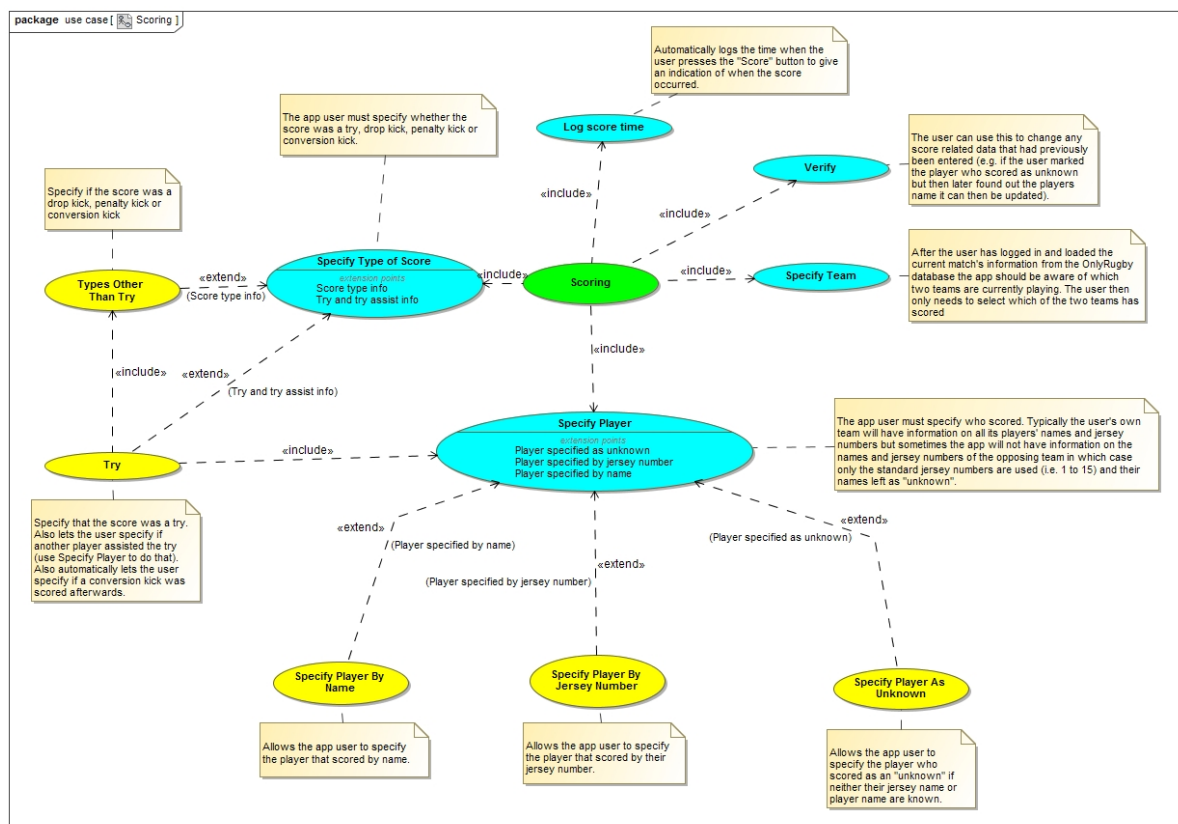
Time.jpg

- **Scoring**

  This use case deals with logging, verifying and storing all the information related to scoring points during a match. When a player scores during a match the app user must use this use case to specify which team scored, the individual player who scored, if the player scored with a try, drop kick, penalty kick or conversion kick and (in the case of the try) if any other players should be credited with a try assist. This use case will also automatically log when the points were scored. All this information is then verified by the app user before it is stored in the database. Once stored this information can then be used to give team- and player statistics.

package use case [Scoring]

Automatically logs the time when the user presses the "Score" button to give an indication of when the score occurred.

Log score time

The user can use this to change any score related data that had previously been entered (e.g. if the user marked the player who scored as unknown but then later found out the players name it can then be updated).

Verify

The app user must specify whether the score was a try, drop kick, penalty kick or conversion kick.

«include»

«include»

Specify if the score was a drop kick, penalty kick or conversion kick

Specify Type of Score
extension points
Score type info
Try and try assist info

«extend»
(Score type info)

«include»

Scoring

«include»

Specify Team

After the user has logged in and loaded the current match's information from the OnlyRugby database the app should be aware of which two teams are currently playing. The user then only needs to select which of the two teams has scored

Types Other Than Try

|«include»

«extend»
(Try and try assist info)

|«include»

Specify Player
extension points
Player specified as unknown
Player specified by jersey number
Player specified by name

The app user must specify who scored. Typically the user's own team will have information on all its players' names and jersey numbers but sometimes the app will not have information on the names and jersey numbers of the opposing team in which case only the standard jersey numbers are used (i.e. 1 to 15) and their names left as "unknown".

Try

«include»

Specify that the score was a try. Also lets the user specify if another player assisted the try (use Specify Player to do that). Also automatically lets the user specify if a conversion kick was scored afterwards.

«extend» /
/ (Player specified by name)

«extend»
(Player specified by jersey number)

«extend»
(Player specified as unknown)

Specify Player By Name

Specify Player By Jersey Number

Specify Player As Unknown

Allows the app user to specify the player that scored by name.

Allows the app user to specify the player that scored by their jersey number.

Allows the app user to specify the player who scored as an "unknown" if neither their jersey name or player name are known.

- **Substitutions**

This use case deals with logging and updating any changes made to either teams on-field teams. When a team makes a substitution the user of the OnlyRugby app will use this use case to specify which team made the substitution, which players were swapped and if there was any special reason for the substitution. This use case will also automatically log when a player has been swapped. All this information is then verified by the app user before it is stored in the database. Once stored this information can then be used to give team- and player statistics.
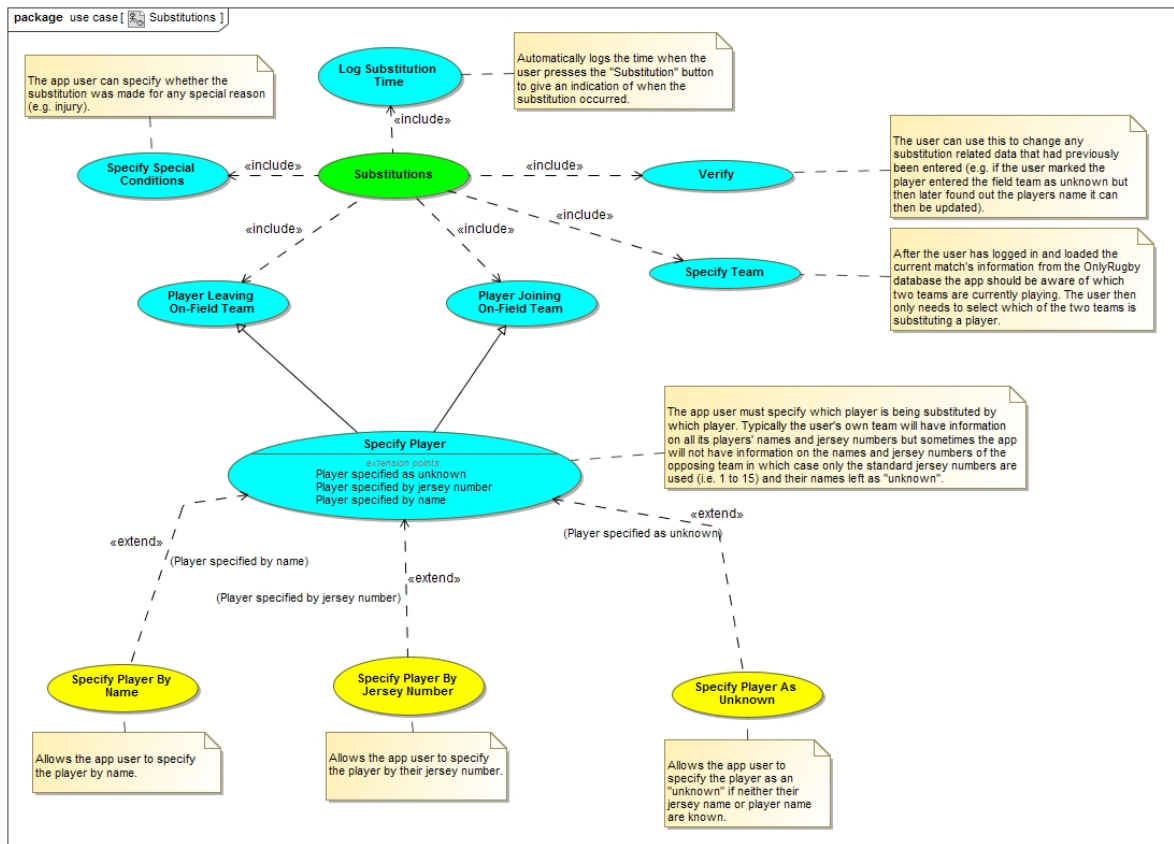
- **Discipline**

  This use case will be used to log any infractions and subsequent punishment for players. If a player is given a card, the reason for the card, as well as the colour of said card. This data will then be logged to the player's profile.
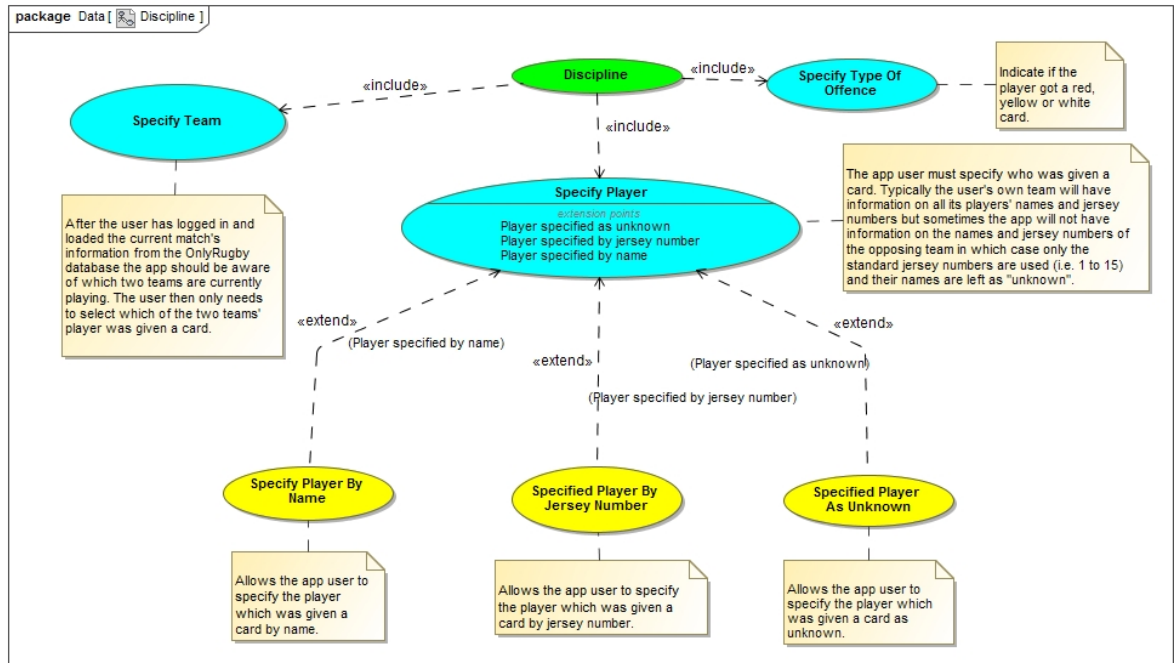
- **Lineouts**

This use case will be used to log information on when a lineout occurred, which team was responsible for throwing in the ball, the identity (name or player number) of the player that threw in the ball, whether or not the lineout was successful, if successful which team won the lineout, and if unsuccessful a reason why the lineout failed.
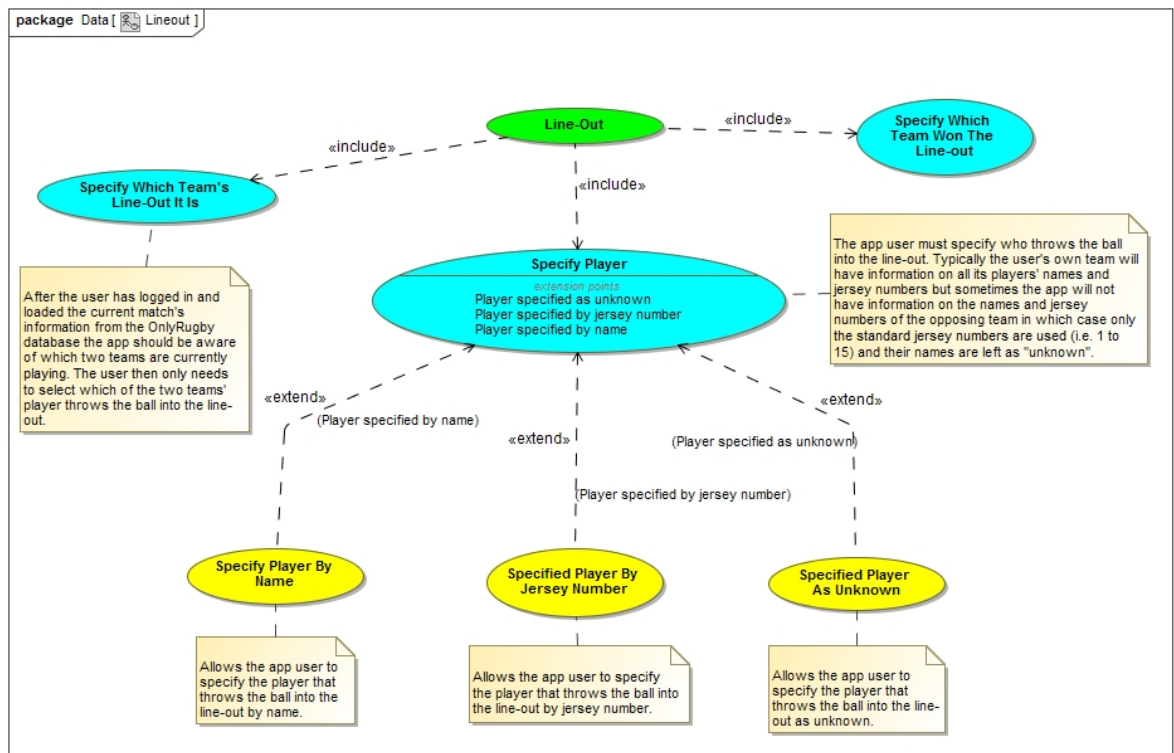
- **Scrums**

  This use case will be used to log information on when a scrum occurred, which team the scrum belonged to and which team won the scrum. The app can check if there are any current reserves on for any forwards or scrum half, and with this info record who was in the scrum.
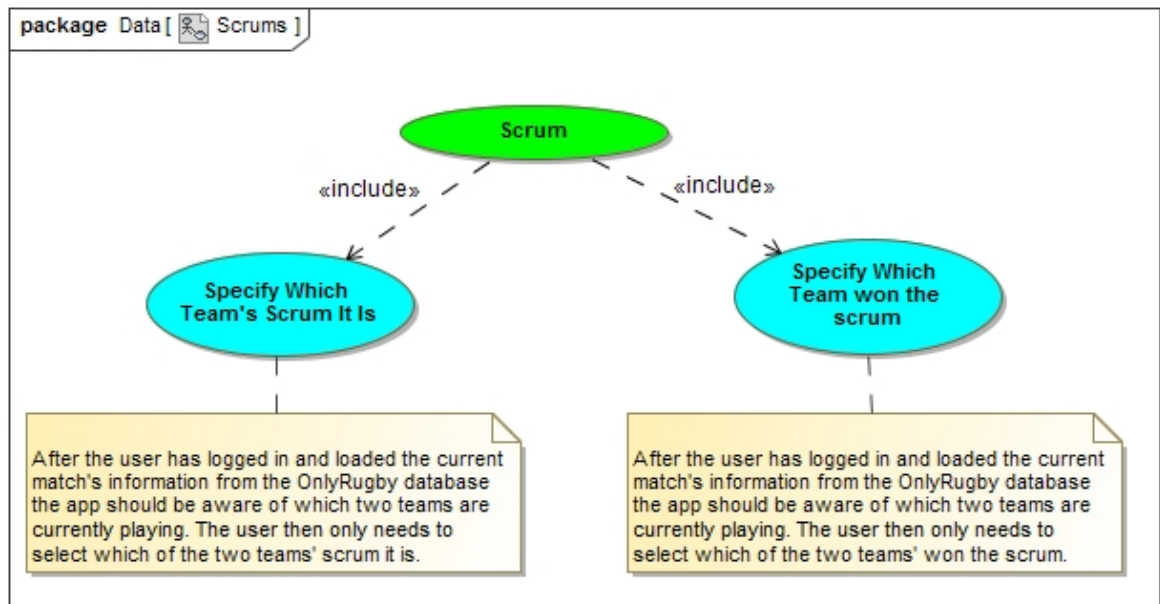

- **Tackles**

  This use case will be used to log the amount of tackles made, by which team member of which team and when it was made. Knowing who was tackled is not required, since it will not be recorded in their statistics.
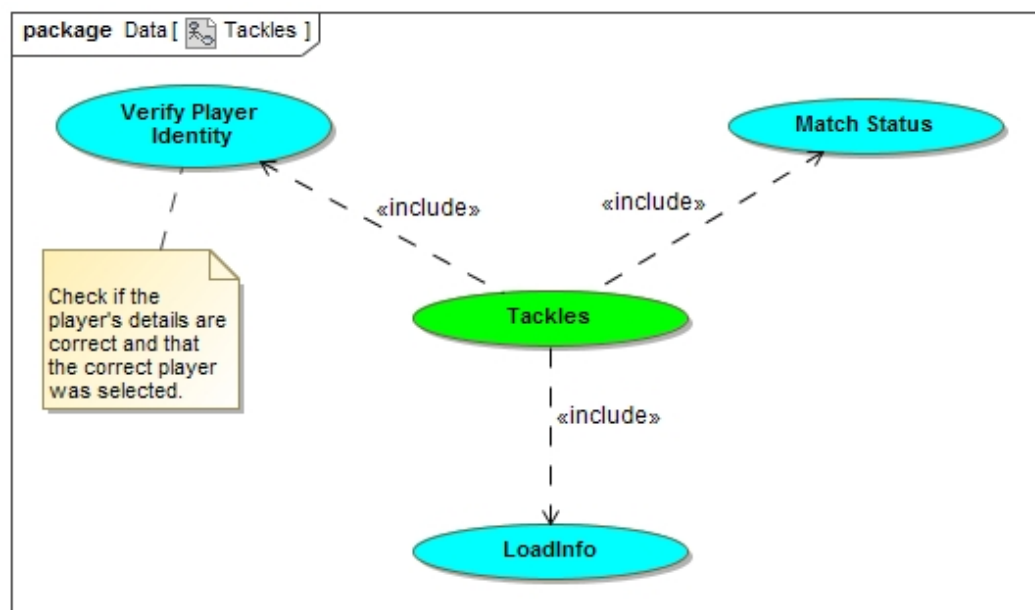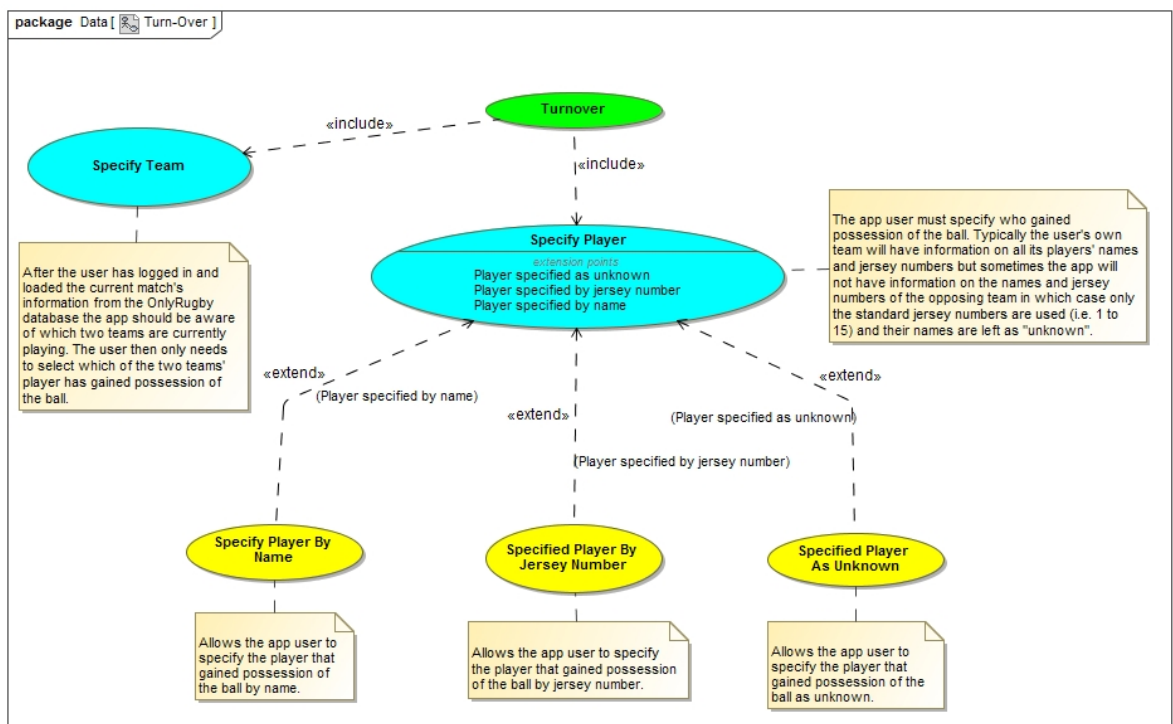
- **Possession**

  This use case will be used to log the percentage of each team's possession for a match. The application will ensure that the total percentages cannot exeed 100 percent.
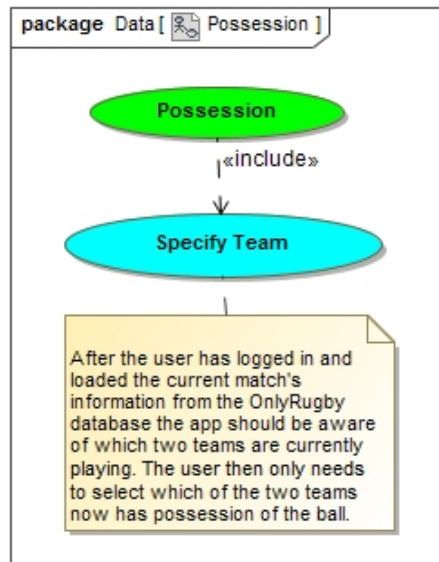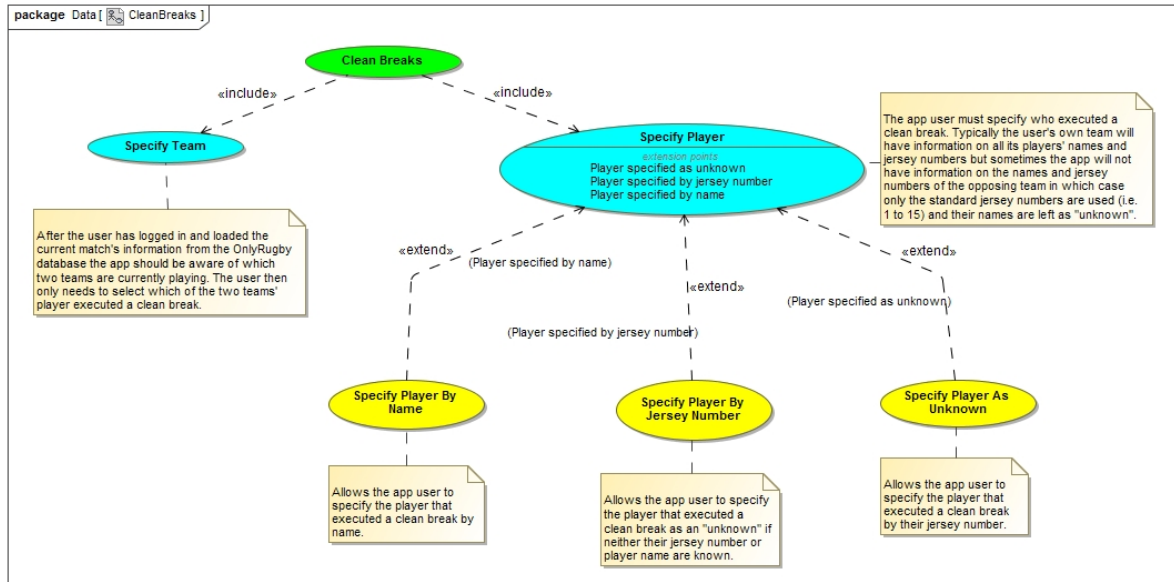

- **Turn-overs**

  This use case will be used to log the occurrence of a turn-over, which team got the ball, which team lost the ball and the players who were involved in the turn-over.
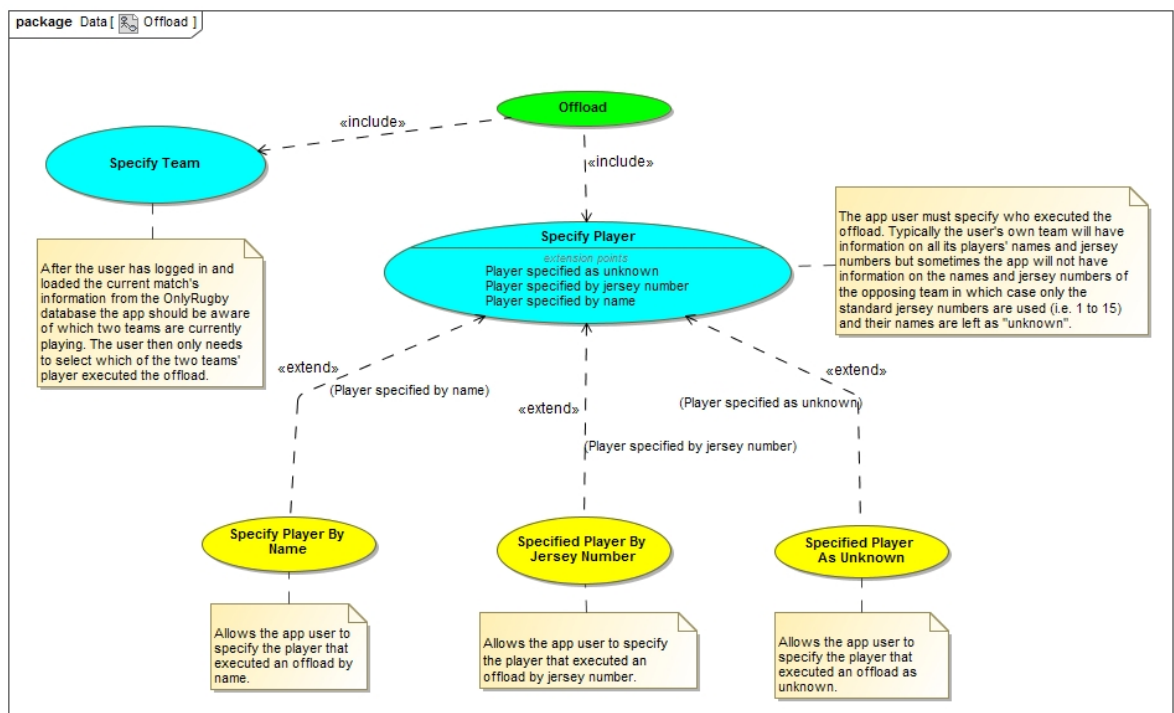
- **Clean breaks**

  This use case will be used to log when clean breaks occur, the player that got the clean break will also be recorded.

  package Data [CleanBreaks]

  Clean Breaks

  «include»   «include»

  Specify Team   Specify Player
                                               extension points
                                               Player specified as unknown
                                               Player specified by jersey number
                                               Player specified by name

  After the user has logged in and loaded the current match's information from the OnlyRugby database the app should be aware of which two teams are currently playing. The user then only needs to select which of the two teams' player executed a clean break.

  The app user must specify who executed a clean break. Typically the user's own team will have information on all its players' names and jersey numbers but sometimes the app will not have information on the names and jersey numbers of the opposing team in which case only the standard jersey numbers are used (i.e. 1 to 15) and their names are left as "unknown".

  «extend» (Player specified by name)   «extend» (Player specified as unknown)
                                               «extend» (Player specified by jersey number)

  Specify Player By Name   Specify Player By Jersey Number   Specify Player As Unknown

  Allows the app user to specify the player that executed a clean break by name.

  Allows the app user to specify the player that executed a clean break as an "unknown" if neither their jersey number or player name are known.

  Allows the app user to specify the player that executed a clean break by their jersey number.

- **Offloads**

  This use case will be used to log information on when an offload occurs, which team performed the offload, and the identity (name or player number) of the player that performed the offload.

  package Data [Offload]

  «include»   Offload

  Specify Team   «include»

                     Specify Player
                     extension points
                     Player specified as unknown
                     Player specified by jersey number
                     Player specified by name

  After the user has logged in and loaded the current match's information from the OnlyRugby database the app should be aware of which two teams are currently playing. The user then only needs to select which of the two teams' player executed the offload.

  The app user must specify who executed the offload. Typically the user's own team will have information on all its players' names and jersey numbers but sometimes the app will not have information on the names and jersey numbers of the opposing team in which case only the standard jersey numbers are used (i.e. 1 to 15) and their names are left as "unknown".

  «extend» (Player specified by name)   «extend» (Player specified as unknown)
  «extend» (Player specified by jersey number)

  Specify Player By Name   Specified Player By Jersey Number   Specified Player As Unknown

  Allows the app user to specify the player that executed an offload by name.

  Allows the app user to specify the player that executed an offload by jersey number.

  Allows the app user to specify the player that executed an offload as unknown.

15

- **Rucks**

  This use case deals with logging any instance of a ruck during match time. When a ruck occurs during a match the OnlyRugby app user can then use this use case to specify which team was defending during the ruck and whether possession of the ball changed during/after the ruck. This use case will also automatically log when a ruck occurs. All this information is then verified by the app user before it is stored in the database. Once stored this information can then be used to give team- and player statistics.
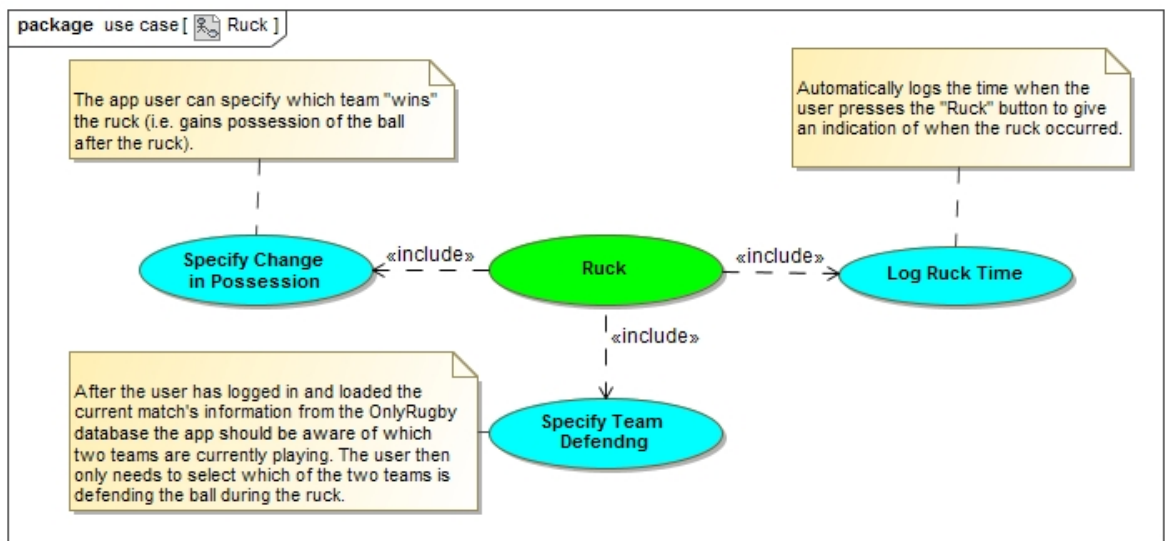
- **Mauls**

  This use case will be used to log how many mauls occurred throughout the match. It will record how many occurred, when they occurred and who won the outcome of the maul (whether there was a turnover ball or not, or if a penalty was conceded).
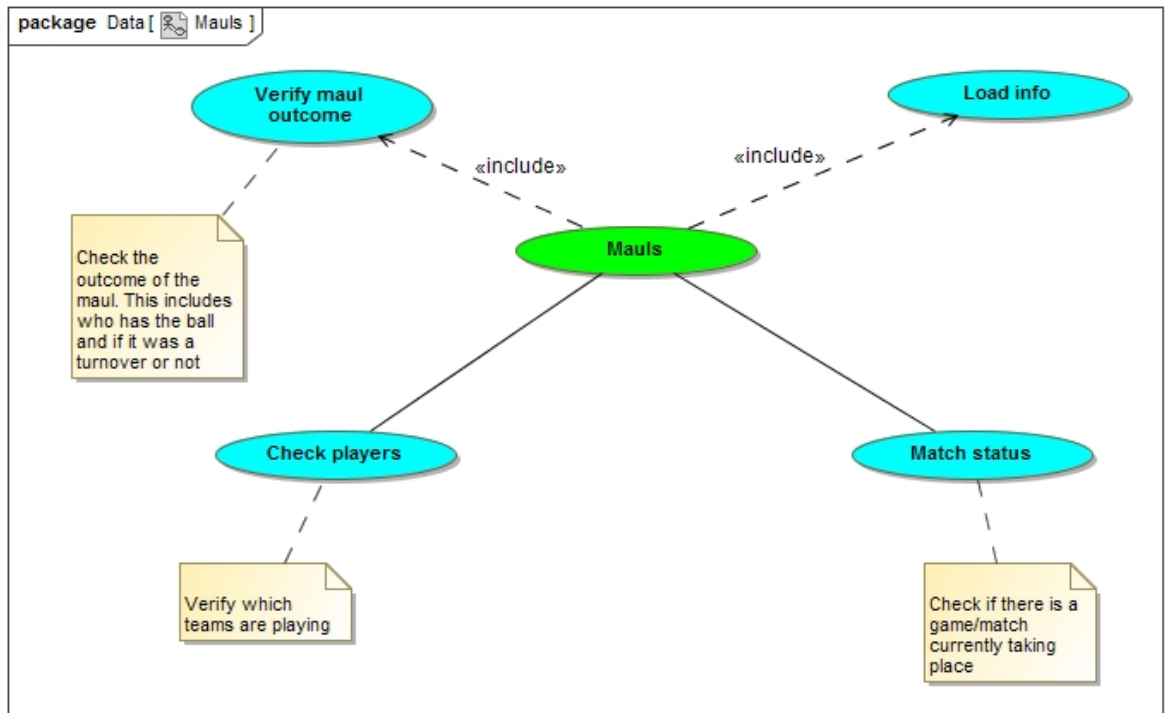
- **Events List**

  This use case will allow the user to view a time-sorted list of events that have so far been logged in the match. It will also facilitate the alteration and deletion of these events as well as the addition of new events to the list. Once an event has been deleted or altered all the previously logged information of that event will be erased (and in the case of alteration, be replaced with new log info).
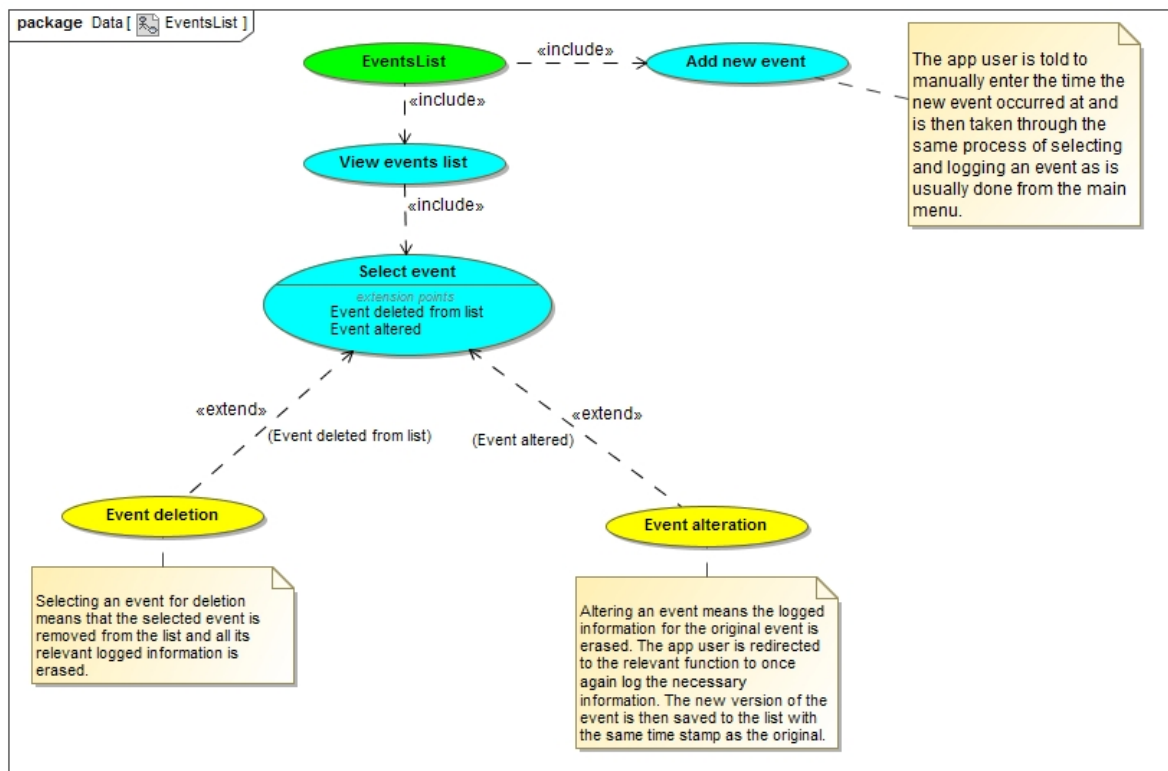
1.5 Process specification

The processes followed when using some of the more important functions of the OnlyRugby app system are displayed below:
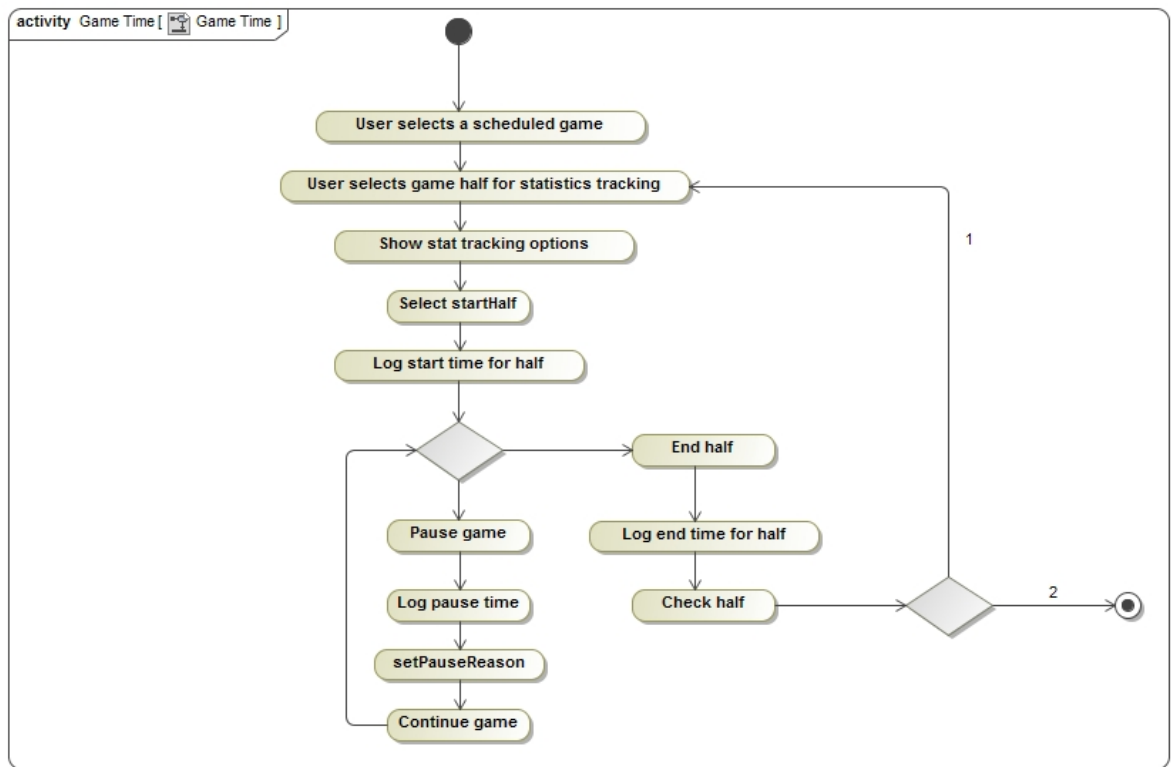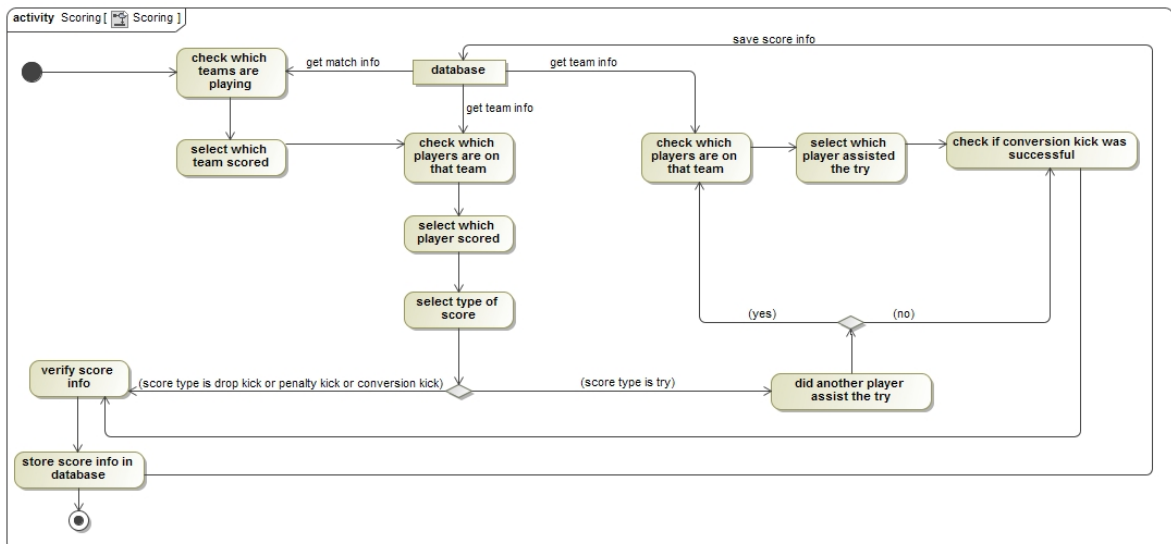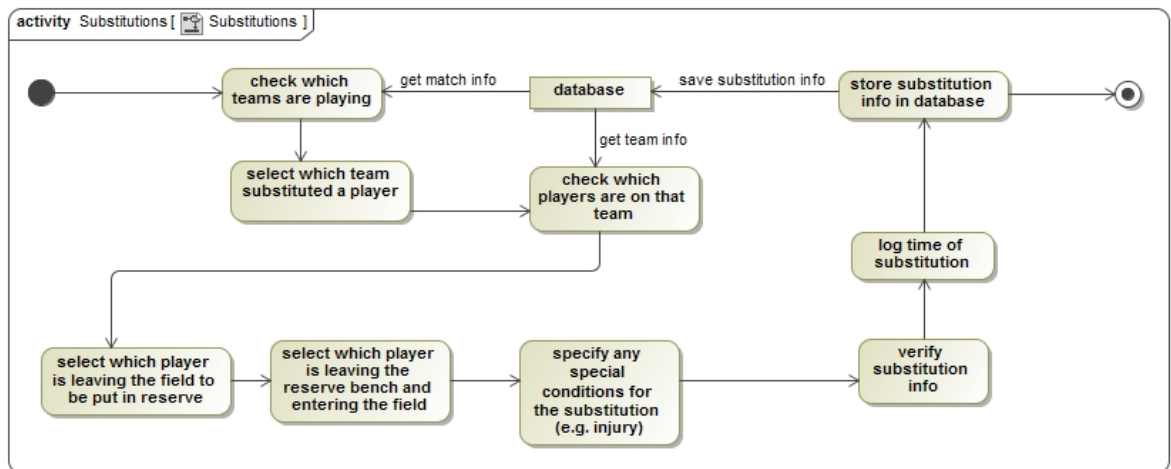
- Log in/ log out
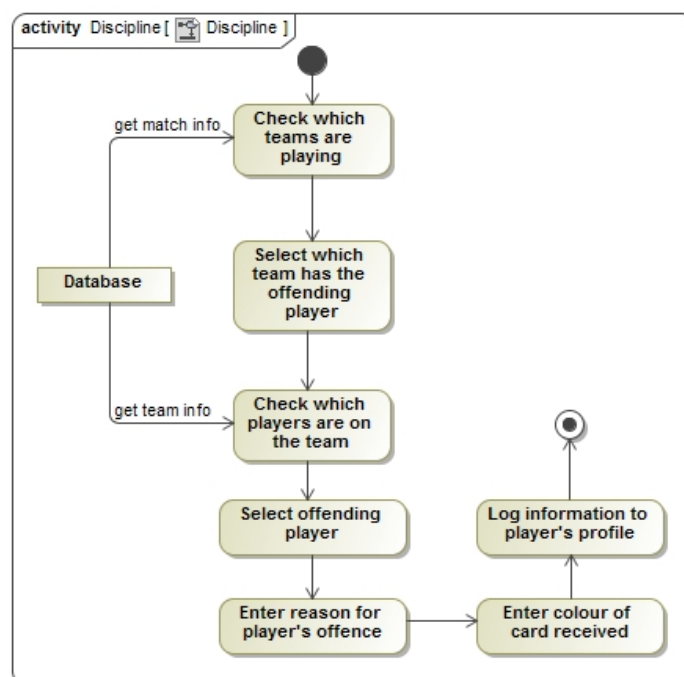

- Load info

- Game time

- Scoring

- Substitutions

- Discipline

21

1.6 Domain Model

Log in/log out

Load Info

Game Time

Scoring

Substitutions

Discipline

Line-outs

Turnovers

Only Rugby App System

Scrums

Offloads

Clean Break

Tackles

Events List

Possession

Ruck

Maul