Waterford Institute *of* Technology

INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

GAA Game Management, Referee Wear & Live Scores

HIGHER DIPLOMA IN SCIENCE

IN COMPUTER SCIENCE (NFQ – LEVEL 8)

2019-2021

INTERIM REPORT

Name:

Student Number: Sarah Barron

Supervisor: Caroline Cahill

Abstract

GAA game management, Referee android wear application and live scores application. The live scores and team sheets will be displayed on the homepage of a web application. The second web application for GAA game management will allow central, provincial and county secretaries create games, assign referees to a game, view team sheets and match reports from games. Club secretaries and team officials will be able to view their teams, view upcoming games and add team sheets for upcoming games. Referees will be able to view their upcoming games and add match reports to a game they have refereed. The web application will be created with a headless Svelte frontend connected directly to Firebase authentication, database, functions and hosting.

The Referee android wear application will allow a referee to keep a record of time with a stop watch, scores, cards, substitutes and injuries during a game, they will be able to record extra information required for the match report and be able to view team sheets. This will be created in android studio using the Kotlin programming language.

This project is a high-level project which would require a lot of further work to move the project from prototype to production. It would require further security and testing. GDPR requirements would have to be met due to the volume of personal information that would be stored in relation to members. I would like to expand the web application with full management system that would allow a CRUD (Create, Read, Update, Delete) system for clubs, teams, members, official roles, players, registration, fundraising, competitions, tournaments, statistics, recording of courses, garda vetting etc. I would extend the live scores application, the management application and referees android wear application to all have their own mobile applications, this would allow fans to view live scores, team officials to submit team sheets and would give the referee the choice to use a mobile or watch during a game. I will be storing GPS co-ordinates for venues so I would like to include google maps and directions for each game venue. I would also like to link the live results to social media platforms and extend the functionality to LGFA and Camogie or adapt the project to other sports.

Keywords: Android, Kotlin, JavaScript, Svelte, Firebase Authentication, Firestore, Firebase Functions

Declaration of authenticity

Higher Diploma in Computer Science 2019-2021

Name of Student: Sarah Barron

Module: Project

Date: June 2021

Assignment: Final Report

I declare that the work that follows is my own, and that any quotations from any sources (e.g., books, journals, the internet) are clearly identified as such by the use of 'single quotation marks', for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (date, author) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award.

Signed: Sarah Barron

Statement of Copyright

Table of Contents

Table Of Figures

1. Introduction

1.1 Background of the problem

Currently GAA Referees entire process is done with pen and paper, they receive written team sheets from team officials, they record all game details in a small note book, such as scores, cards, substitutions and injuries. After the game they have to transfer the information from the notebook to the match report sheet which takes time and finally they have to organise the safe delivery of the match report to the secretary of the committee-in-charge.

Team Officials such as trainers, selectors or managers currently have to organise getting team sheet booklets from their club/county secretary. They must remember to bring the team sheets booklet to a game along with all the other things they need to remember to bring. They then have to try fill the team sheet out on the side line, making changes can be messy and often a fresh team sheet is needed which is wasteful. The team official has to remember to give the team sheet to the referee and the opposing team, which is time consuming when they could be spending that time preparing their team for a game.

The broadcast of live or final results is also random. Some clubs are great to update their twitter accounts with live scores and some clubs don't post any results, leaving GAA fans wondering where they might find their teams live results today, will it be on Facebook, Twitter or nowhere.

1.2 Description of the project aims

- I aim to create a basic management web application for secretaries to create, view, update and delete games, view team officials team sheets and view the referees match reports. Team-officials will be able to view upcoming games and create, view, update and delete team sheets. Referees should be able to view their upcoming games, create, view, update and delete match reports and view team sheets linked to games they have been involved in.

- I hope to provide the Referee with an android wear application that they can create, view, update and delete:
  - Scores – points & goals, the team who scored and optional added information of the player number for statistics.
  - Cards – red, yellow and black cards and the team and player number.
  - Substitutes – team, player number going off and player number coming on and if it is a blood substitute or black card.
  - Injuries – team, player number and a note
  - Time – stop watch, with functions to start, pause, reset and stop the game.

- I also hope to include in the web application live scores. The GAA fan will be able to filter these results by selecting their county/counties and club(s) they wish to follow.

2. Tools and technologies

2.1 Hardware

- Laptop to write code and do research
- Laptop / Tablet / Mobile to view the web application
- Smart Watch for the referee watch

2.2 Software

- Kotlin - a static type, object-oriented programming (OOP) language that is compatible with the Java virtual machine, Java libraries and Android. (Kotlin, n.d.)
- HTML – a standard markup language for creating website pages. (HTML: HyperText Markup Language, n.d.)
- CSS – Cascading Style Sheets describes how html elements should be look or be displayed on a website page. (CSS: Cascading Style Sheets, n.d.).
- JavaScript – an object-oriented computer programming language that allows interaction within web browsers (JavaScript, n.d.)
- Svelte - Svelte writes code that surgically updates the DOM when the state of your app changes. It requires writing less code. Compiles your code to tiny framework-less vanilla JavaScript, which means it starts and stays fast and it is reactive (Svelte, n.d.)
- Google Firebase – backend application development software that supports android and web applications
  - Authentication – used to authenticate users with email and password and google for the watch (Firebase Authentication, n.d.)
  - Firestore –
    "a flexible, scalable database for mobile, web and server development".
    "It keeps data synchronised across client apps using real-time listeners". Used for responsive apps that works regardless of network latency or internet connection. Integrates with other firebase and google cloud products and functions. (Firebase Firestore, n.d.)
  - Realtime Database
    "Store and sync data with our NoSQL cloud database. Data is synced across all clients in realtime, and remains available when your app goes offline." (Firebase Realtime Database, n.d.)
  - Functions – allows you run backend code when events are triggered by a function. The function is stored in Google cloud in a managed environment. (Firebase Cloud Functions, n.d.)
  - Hosting "Firebase Hosting provides fast and secure hosting for your web app, static and dynamic content, and microservices." (Firebase Hosting, n.d.)
  - Storage – used to store images of club and county crests.

- Android Studio – IDE for android development (Android Studio, n.d.)
- Visual Studio Code – Source Code Editor for development of the svelte web application. (Visual Studio Code, n.d.)
- Bootstrap – Responsive mobile first frontend framework (Bootstrap, n.d.)

3. System Analysis and Design

3.1 User stories

Referee

I want to eliminate having to transfer information from my note book to the match report after the game as this takes up a lot of time. This android wear application will solve this problem as the information will be transferred automatically.

I want to eliminate having to organise the delivery of match report booklets and notebooks from the county board as it takes time and effort and it will also save the county board money on expenses. The web application will solve this as I will not need these items any longer.

I want to track time so that I can track the amount of time played in a game. The android wear application will solve this as it will include a stopwatch that I can start, pause, reset and stop.

I want to track scores so that I can report an accurate score at the end of a game to my match report. The android wear application will solve this as it will allow me to record scores.

I want to track cards so that I can report which players received cards at the end of the game to my match report. The android wear application will solve this as it will allow me to record a card colour, team and player number.

I want to track substitutes so that I can report substitutions at the end of a game and also to know if the maximum number of substitutes have been used during a game.

I want to track injuries so that I can report injuries at the end of a game to my match report. The android wear application will solve this as it will allow me to record an injury

I want to have instant accurate information during a game when asked for the time or score, I don't want to have to count up scores from a notebook each time. The android wear application will solve this as it will display the time and score on the home page.

I want to view what upcoming games I am due to attend so that I don't have to keep track of the games in my diary any longer. The web application will solve this problem as I can view this information clearly all in the one place from my computer or mobile phone.

I want to be able to submit the final match report instantly after a game so I don't have to organise the delivery of match reports to the secretary any longer as it takes a lot of time. The android wear application will solve this problem as I can input the information directly and submit it instantly.

I want to avoid having to remember to gather or mind team sheets any longer as it takes time and team sheets can often be lost during a game when it falls out of a pocket. Having the team management input their team sheet digitally will solve this problem.

Team Official

I want to eliminate having to organise the delivery of team sheet booklets and save the club money having to use funds to purchase these booklets. The web application will solve this as team sheets will no longer be required.

I want to be able to submit my team sheet digitally so that I don't have to remember to bring team sheets to a game along with all the other thing I need to remember to bring. The web application will solve this as I will no longer need to bring these items.

I want to spend the time prior to the match preparing my team and not trying to locate the referee or the opposition manager to give them copies of this team sheet. The web application will solve this as I will be able to submit the team sheet digitally.

I want to be able to submit a draft team sheet the day or days before and only have to update any changes that are needed prior to the game, without having to rewrite a new team sheet it will also save me time and help me to be more prepared for the game. The web application will solve this as I will be able to submit the team sheet at any time before the game and update the team sheet whenever it is needed.

Secretaries

I want to be able to view all upcoming games. The web application can solve this problem as it will allow me to view upcoming games.

I want to be able to create games, so that I have a record of upcoming games and the teams and referees can have information about these games on their devices too. The web application will solve this as I will be able to create, view, update or delete a game.

I want to be able to assign a referee to a game. The web application will solve this as I will be able to add a referee to a game once one has been organised.

I want to be able to view referee match reports and team official team sheets as soon as possible. The web application will solve this as I will be able to view these directly after the team-official or referee has inputted them.

GAA Fan

I want to be able to view all live results for games in one place, without having to scroll through Twitter or Facebook looking for the results. This web application will solve this problem as I can go to the one place and view all the results in real time.

I want to be able to filter which results I see by my club or county. The web/mobile application will solve this problem as I can filter results by club or county.

I want to be able to view the starting team and subs. The application will solve this as I will be able to view the teams online.

3.2 Use Cases

3.2.1 Referee Android Watch

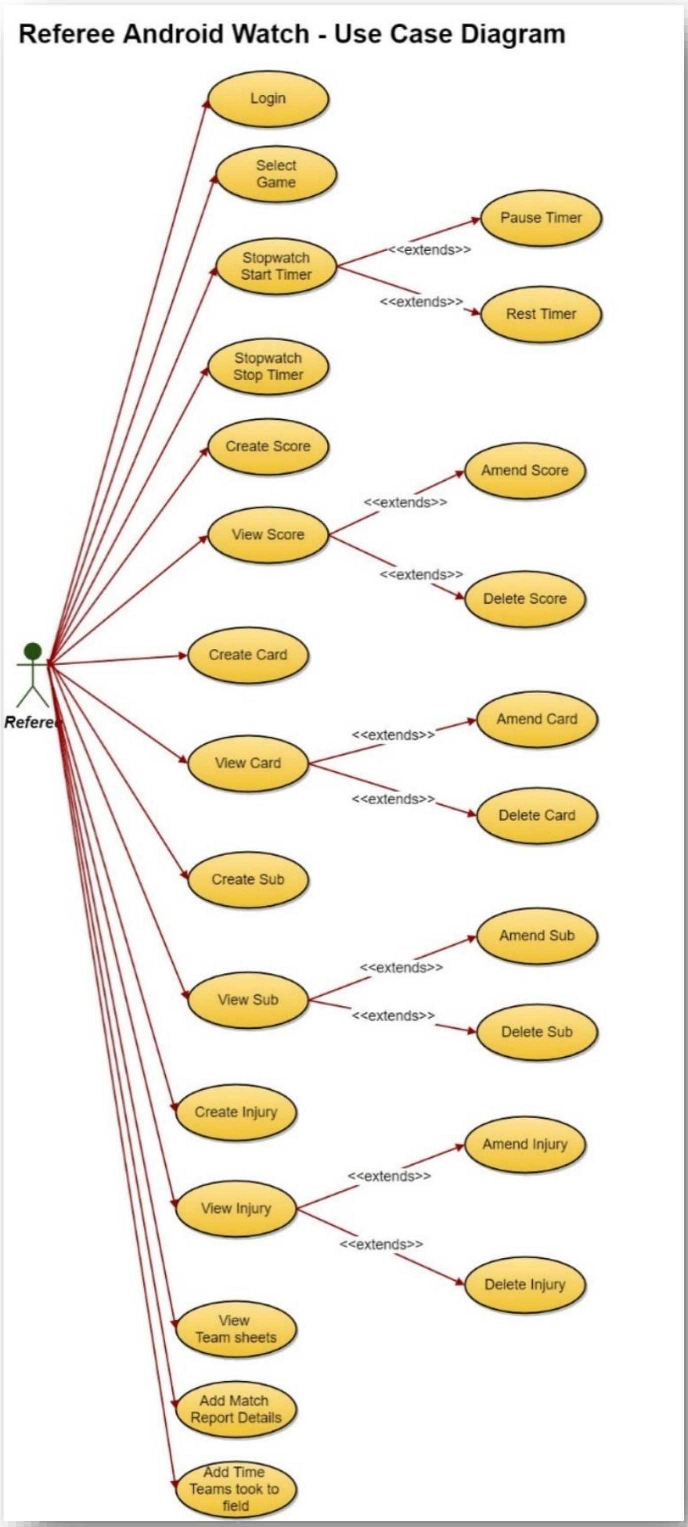


Figure 1 – Use Case Diagram – Referee Android Wear (Draw, n.d.)

3.2.2 Secretary

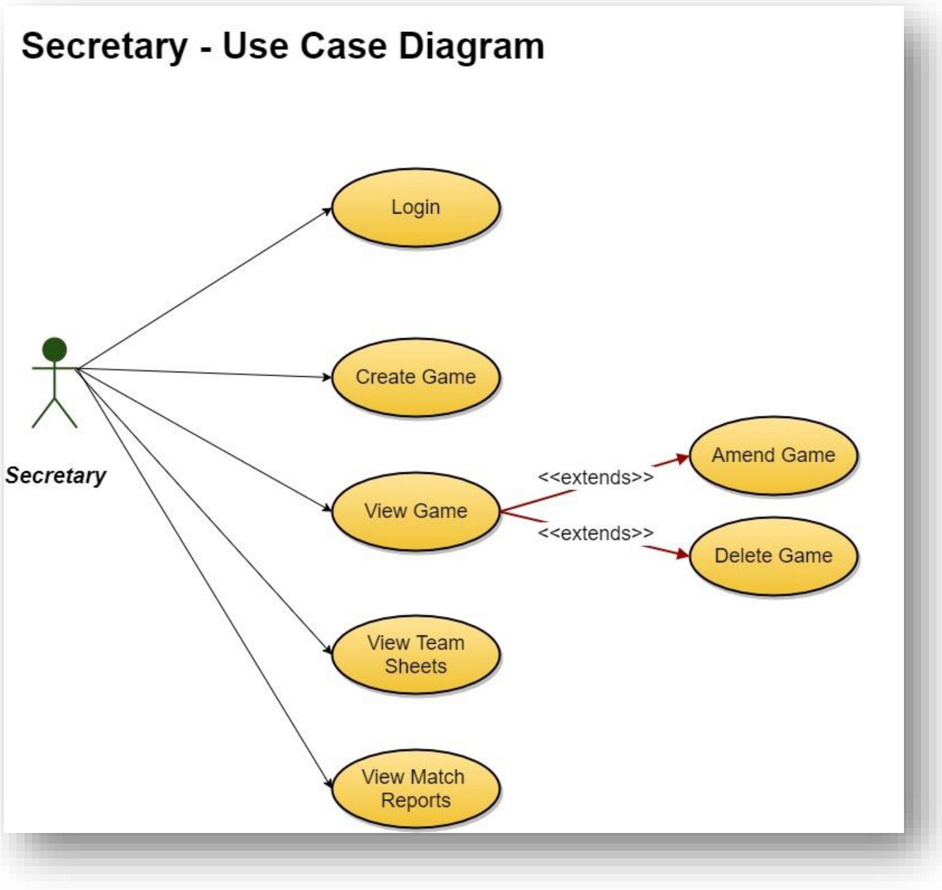


Figure 2 - Use Case Diagram - Secretary Web Application (Draw, n.d.)

3.2.3 Team Official

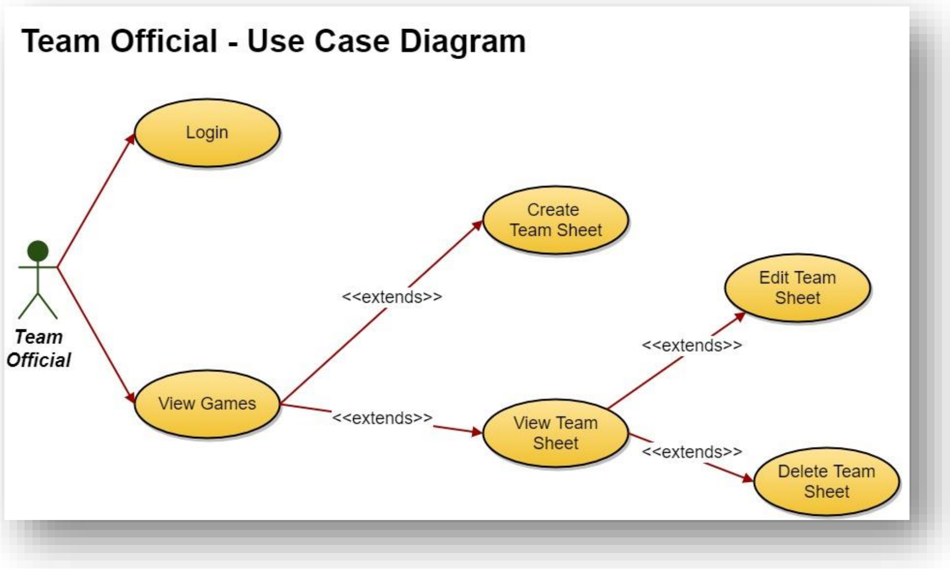


Figure 3 - Use Case Diagram – Team Official Web App (Draw, n.d.)

3.2.4 Referee – Web Application

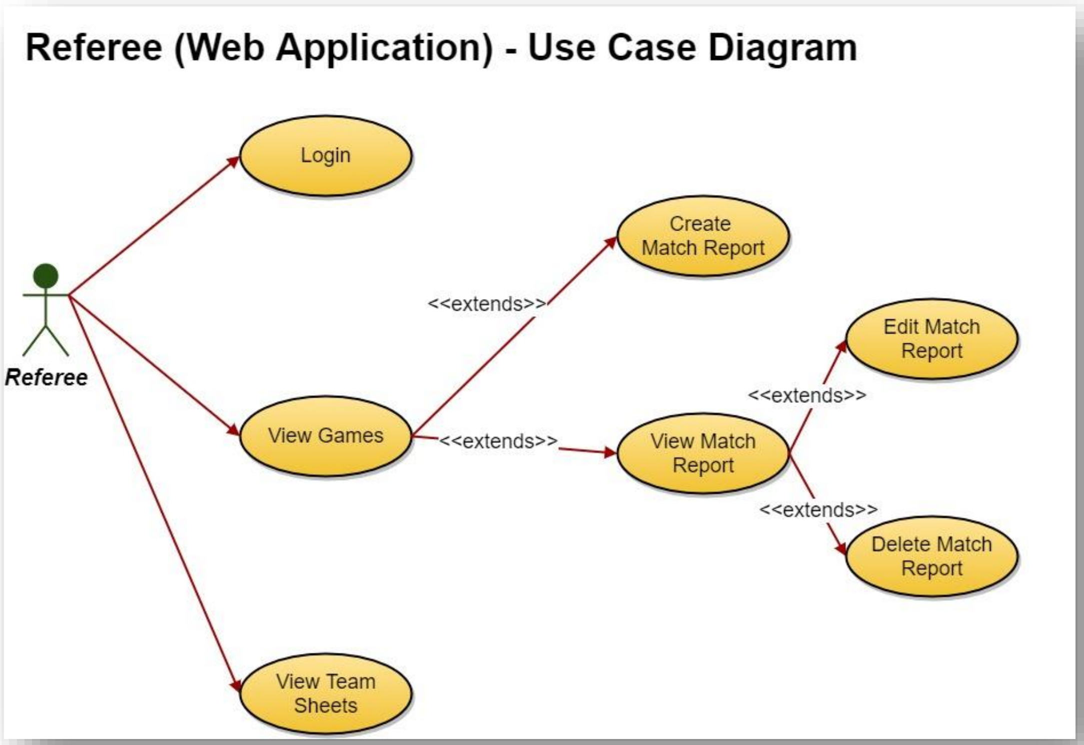


Figure 4 - Use Case Diagram - Referee Web App (Draw, n.d.)

3.2.5 GAA Fan

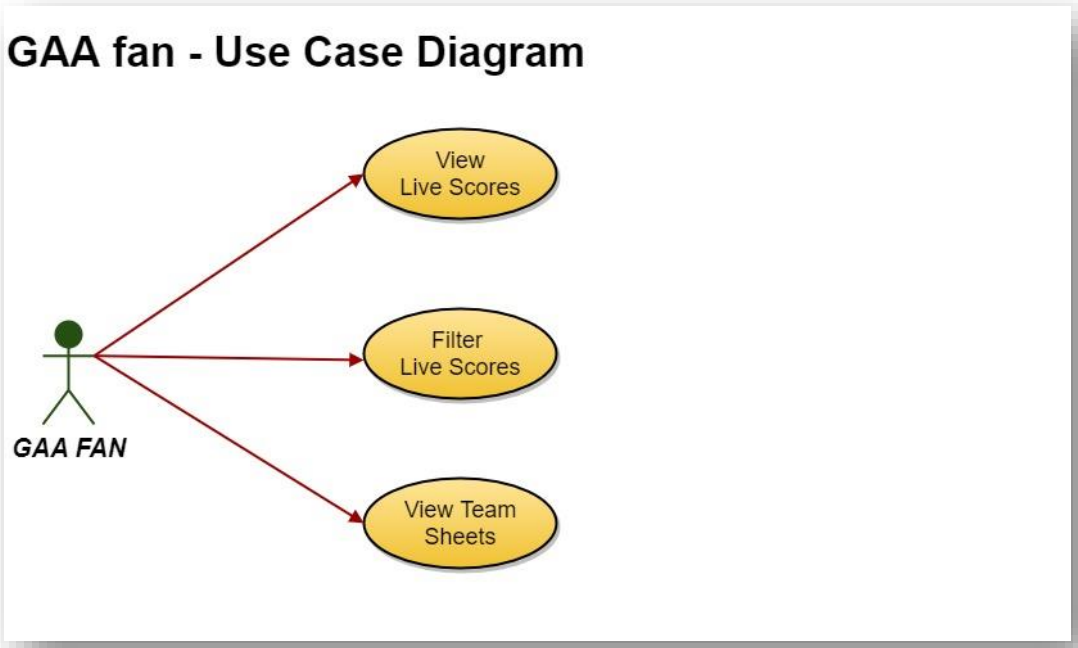


Figure 5 - Use Case Diagram – GAA Fan Web App (Draw, n.d.)

3.3 Database Design

3.3.1 Conceptual Database Design

A Secretary is a member and can create a game.

A Province is made up of counties, it has one secretary and has referees to referee county games.

A County is made up of clubs, it has one secretary, teams and referees. They have at least one venue where games are played.

A club has at least one venue where games are played. The club has a minimum of 15 members. The club has at least 1 team.

A member is a member of one club, they may or may not be a player, secretary, team official or referee. A member has one membership type.

A player is on at least one team. A player can play a game, score, be substituted, get an injury or get a card.

A team official is a member of a club. They are over at least one team which doesn’t have to be their club or county. They can submit team sheets.

A referee can referee club games and may or may not be able to referee county games. A referee is a member. A referee has at least one sport type that he referees. In a game a referee can issue a card, record substitutes, scores and injuries. A referee submits one match report after every game.

A competition has a grade and a sport type.

A team has one grade, one sport type, at least one team official, at least 7 players and a team can play games. When a player scores the team is awarded the score.

A game has a competition, venue, 2 teams, at least one referee and sometimes there will be one more standby referee. A game may or may not have scores, cards, substitutes or injuries. Every game has 2 team sheets and one match report.

A score is scored by a player in a game. This score is given to the team. Scores must be recorded in the match report

A card is given from a referee to a player in a game. The card must be recorded in the match report.

A substitute occurs in a game and involves 2 players from one team. the substitution must be recorded in the match report

A player can receive an injury in a game. An injury must be recorded in the match report

A team sheet is submitted by a team official and includes details about the game, competition, the 2 teams, venue and a list of at least 7 players.

A match report is submitted by a referee and includes details about the game, competition, venue the 2 teams, scores, cards, substitutes and injuries

1.1 Identify entity types

Province, County, Club, Venue, Member, Player, Secretary, Team Official, Referee, Membership Type, Grade, Sport Type, Competition, Team, Game, Score, Card, Substitute, Injury, Team Sheet, Match Report.

1.2 Identify Relationship types

secretary	1..1	creates	0..*	game
province	1..1	contains	1..*	counties
province	0..1	has	1..1	secretary
province	0..1	has	1..*	referees
county	1..1	has	1..*	clubs
county	0..1	has	1..1	secretary
county	1..1	has	1..*	venue
county	0..1	has	0..*	team
club	1..1	has	1..*	venue
club	1..1	has	15..*	members
club	0..1	has	1..*	team
member	1..1	can be a	0..1	player
member	1..1	can be a	0..1	referee
member	1..1	can be a	0..1	team-official
member	1..1	can be a secretary	0..1	secretary
member	0..*	has	1..1	membership-type
player	7..*	is on	1..*	team
player	14..*	plays	0..*	game
player	1..1	can	0..*	score
player	2..2	can be a	0..*	substitute
player	1..1	gets an	0..*	injury
player	1..1	gets a	0..*	card
team-official	1..*	is over	1..*	team
team-official	1..1	submits	0..*	team-sheet
referee	1..2	referees	0..*	game
referee	0..*	has	1..1	sport-type
referee	1..1	issues	0..*	card
referee	1..1	records	0..*	substitutes
referee	1..1	records	0..*	injuries
referee	1..1	records	0..*	score
referee	1..1	submits	0..*	match-reports
competition	0..*	has	1..1	grade
competition	0..*	has	1..1	sport-type
team	0..*	has	1..1	grade
team	0..*	has	1..1	sport-type
team	2..2	plays	0..*	game
team	1..1	is awarded	0..*	score
game	0..*	has	1..1	competition
game	0..*	has	1..1	venue
game	1..1	has	0..*	score
game	1..1	has	0.*	card
game	1..1	has	0..*	substitute
game	1..1	has	0..*	injury
game	1..1	has	2..2	team-sheet
game	1..1	has	1..1	match-report
score	0..*	is recorded in	1..1	match-report

card	0..*	is recorded in	1..1	match-report
substitute	0..*	is recorded in	1..1	match-report
injury	0..*	is recorded in	1..1	match-report
competition	1..1	is recorded in	0..*	team-sheet
competition	1..1	is recorded in	0..*	match report
team	1..1	is recorded in	0..*	team-sheets
team	2..2	is recorded in	0..*	match-reports
player	7..*	is recorded in	0..*	team-sheets
venue	1..1	is recorded in	0..*	team-sheets
venue	1..1	is recorded in	0..*	team-sheets

Figure 6 – Identity Relationship types table

1.3 Identify and associate attributes with entity

Central Council: name, address (addressLine1, addressLine2, town, county, Eircode, country), email, telephone.

Province: name, address (addressLine1, addressLine2, town, county, Eircode, country), email, telephone.

County: name, address (addressLine1, addressLine2, town, county, Eircode, country), email, telephone, colours

Club: name, address (addressLine1, addressLine2, town, county, Eircode, country), email, telephone, colours

Venue: name, address (addressline1, addressline2, county, Eircode, country), latitude, longitude.

Sport Type: name

MembershipType: name, description

Member: name (firstName, lastName), address (addressLine1, addressLine2, town, county, Eircode, country), email, telephone, mebershipDate, firstClub, ownClub

Player: DOB, name (firstName, lastName), address (addressLine1, addressLine2, town, county, Eircode, country), email, telephone, sportType(name)[ ], registeredDate

Team Official: name (firstName, lastName), address (addressLine1, addressLine2, town, county, Eircode, country), email, telephone.

Secretary: name (firstName, lastName), address (addressLine1, addressLine2, town, county, Eircode, country), email, telephone

Referee: name (firstName, lastName), address (addressLine1, addressLine2, town, county, Eircode, country), email, telephone, canRefCounty

Team: name

Grade: level, name

Competition: name, isNational.

Game: dateTime(date, time), linesmen[], umpires[], teamATotalGoals, teamBTotalsGoals, teamATotalPoints, teamBTotalPoints

MatchReport: dateTime(date, time), substituteRef, linesman[], umpires[], teamATookToField, teamBTookToField, gameStart, gameEnd, finalScore(teamAGoals, teamAPoints, teamBGoals, teamBPoints), delayInStart(onTime, note), matchProgram, jerseyNumCorrect(correct, note), linesmenAttire(good, note), pitch(marked, note), grass(cut, note), additionalComments.

Card: timestamp, cardColor, note.

Substitute: timestamp, bloodsub.

Injury: timestamp, note.

Score: timestamp, goal, point

TeamSheet: timestampSubmitted, lineout [players[fieldPosition, jerseyNumber]]

1.4 Determine attribute domains
View on the ERI diagram (figure 19)

1.5 Determine candidate, primary, and alternate key attributes
Unique Entities will have their name as the reference all other entities will have generated references

1.6 Consider use of enhanced modelling concepts
Member is a superclass and player, team official, secretary and referee are subclasses of Member

Everyone is a member but a member may or may not be a player, team official, secretary or referee and they can also be one, multiple or all of these roles, therefore it has an {optional, and} relationship. The guidelines for representation of superclass and subclass with an {optional, and} relationship is to have two relations, one relation for the superclass and one relation for the subclass with one or more discriminations to distinguish the type of each part.

Therefore, I will create Member and MemberRole. I can therefore remove the Player, Secretary, Team Official and Referee Entity.

Member: memberReference, name (firstName, lastName), address (addressLine1, addressLine2, town, county, Eircode, country), email, telephone, membershipDate, firstClub, ownClub.

MemberRoles: secretaryOfClub, secretaryOfCounty, secretaryOfProvince, secretaryOfCouncil, refereeOfCounty, refereeOfClub, teamOfficial, clubPlayer, countyPlayer, DOB, registeredDate

1.7 Check model for redundancy

- Figure 7 shows redundancy between the province, county and member. The province can be found through the county therefore the relationship between the member and the province can be removed

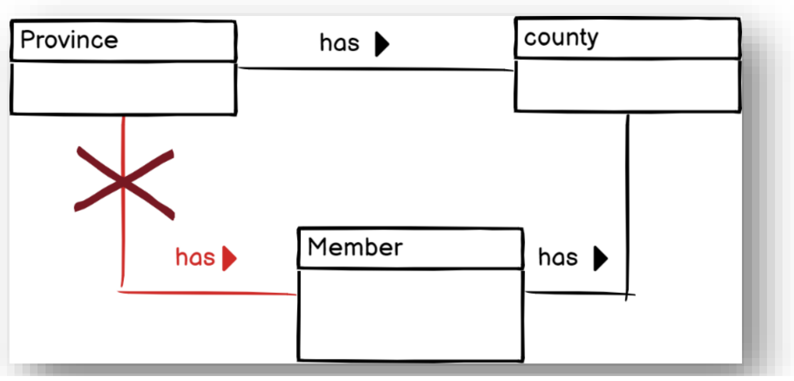


Figure 7 -Model Redundancy – Province and Member (Balsamiq Cloud, n.d.)

- Figure 8 shows redundancy between the county, club and member. The county can be found through the club therefore the relationship between the member and the county can be removed.

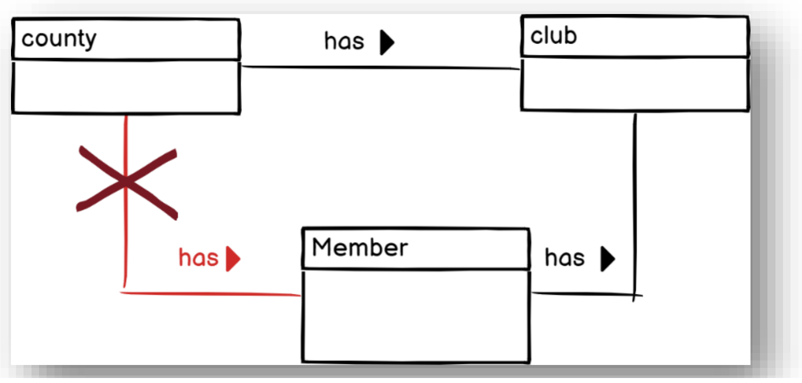


Figure 8 – Model Redundancy – County – Member (Balsamiq Cloud, n.d.)

- Figure 9 shows redundancy between the game, team sheet and competition. The team sheet can find out the competition from the game therefore the relationship between team sheet and competition can be removed.

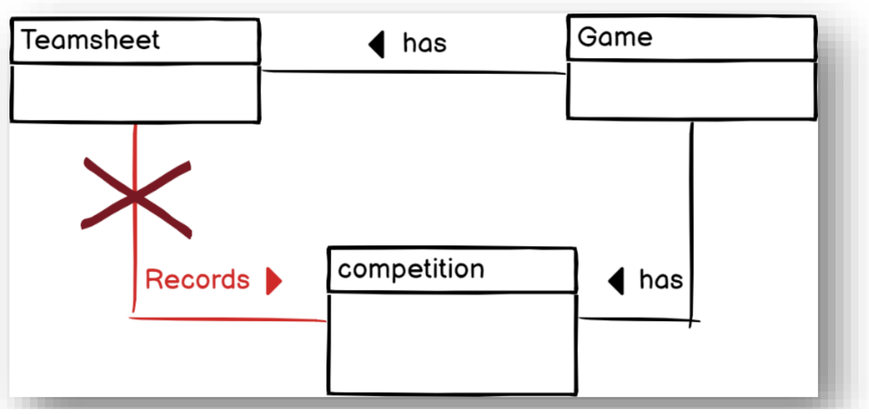


Figure 9 – Model Redundancy – Team sheet- Competition (Balsamiq Cloud, n.d.)

- Figure 10 shows redundancy between the game, match report and competition. The match report can find out the competition from the game therefore the relationship between the match report and competition can be removed.

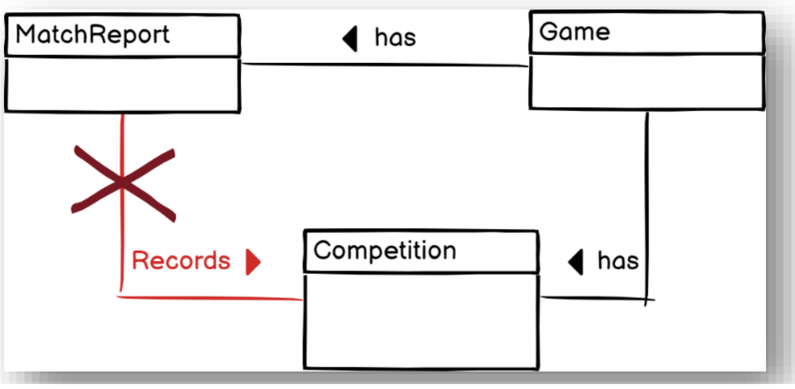


Figure 10 – Model Redundancy – Match Report – Competition (Balsamiq Cloud, n.d.)

- Figure 11 shows redundancy between Team, Player and a Score. The Team can be found through the Player therefore I can remove the relationship between the Team and the Score

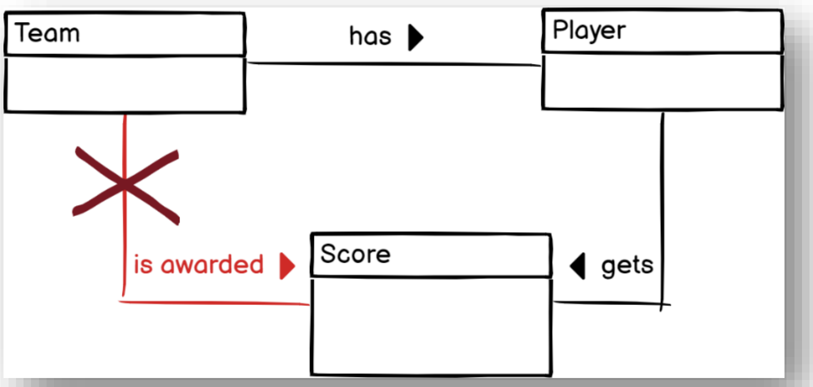


Figure 11 – Model Redundancy Team – Score (Balsamiq Cloud, n.d.)

- Figure 12 shows a redundant relationship between the game, team sheet and the venue. The venue can be found via the game therefore the relationship between the venue and the team sheet can be removed.

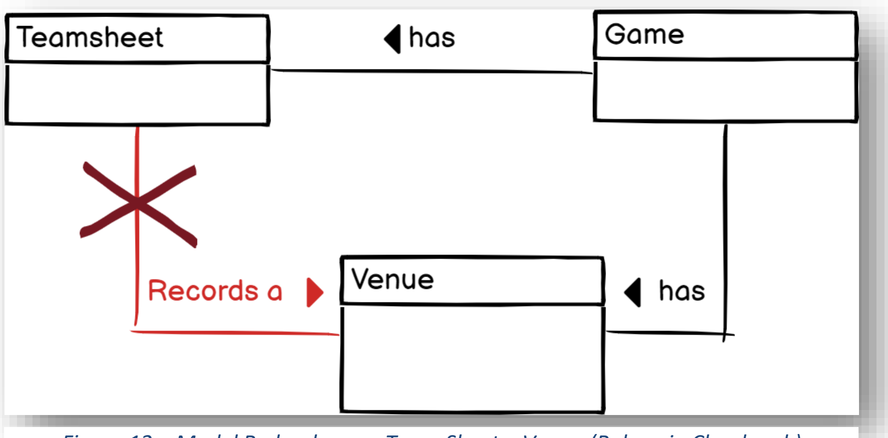


Figure 12 – Model Redundancy – Team Sheet – Venue (Balsamiq Cloud, n.d.)

- Figure 13 shows a redundant relationship between the match report, team and the game. The match report can get the team from the game, therefore, the relationship between the match report and the team can be removed.

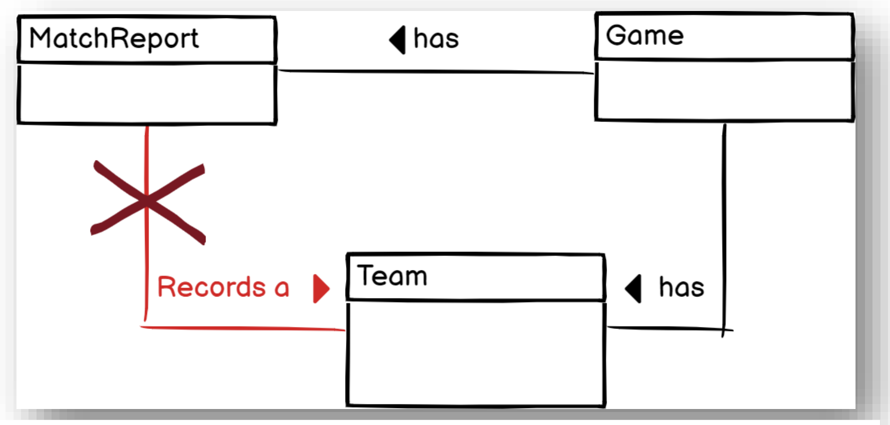


Figure 13 – Model Redundancy – Match Report – Team (Balsamiq Cloud, n.d.)

- Figure 14 shows a redundant relationship between the match report, score and the game. The match report can get the score from the game, therefore, the relationship between the match report and the Score can be removed.

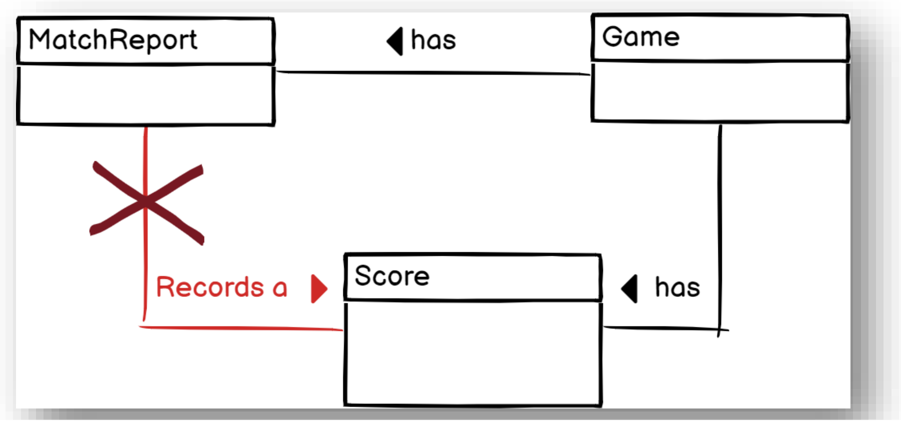


Figure 14 – Model Redundancy – Match Report – Substitute (Balsamiq Cloud, n.d.)

- Figure 15 shows a redundant relationship between the match report, card and the game. The match report can get the card from the game, therefore, the relationship between the match report and the card can be removed.

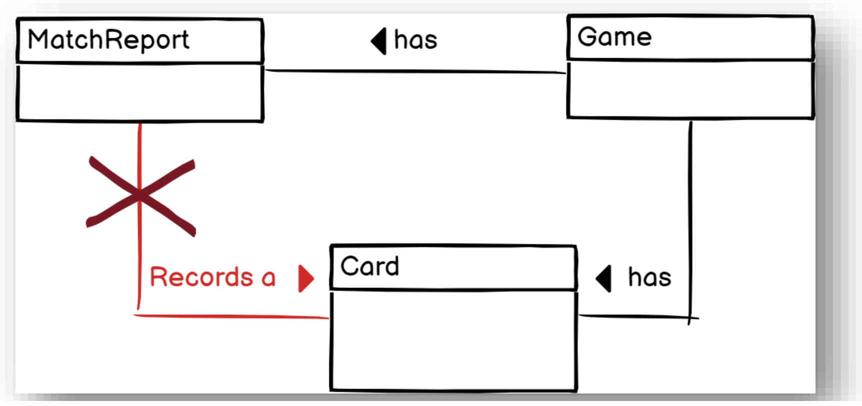


Figure 15 – Model Redundancy – Match Report – Card (Balsamiq Cloud, n.d.)

- Figure 16 shows a redundant relationship between the match report, substitute and the game. The match report can get the substitute from the game, therefore, the relationship between the match report and the substitute can be removed.

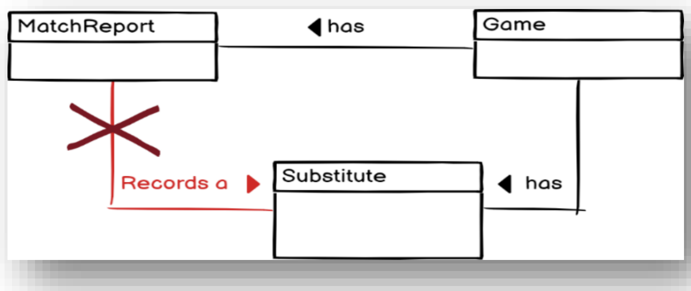


Figure 16 – Model Redundancy Match Report – Substitute (Balsamiq Cloud, n.d.)

- Figure 17 shows a redundant relationship between the match report, injury and the game. The match report can get the injury from the game, therefore, the relationship between the match report and the injury can be removed

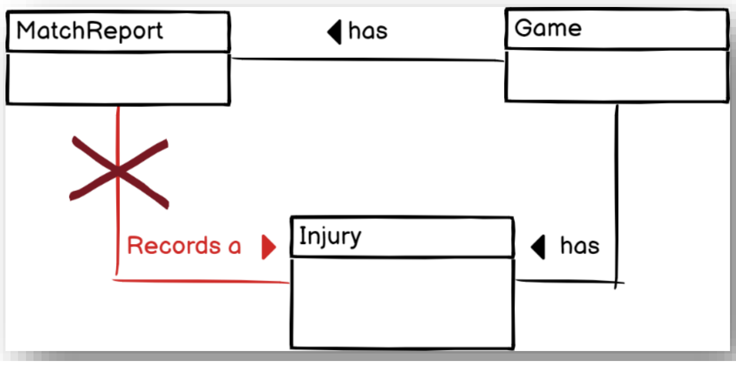


Figure 17 – Model Redundancy – Match Report – Injury (Balsamiq Cloud, n.d.)

- Figure 18 shows a redundant relationship between the match report, member and the game. The match report can get the member from the game and the games relationship with the team sheet, therefore, the relationship between the match report and the member can be removed

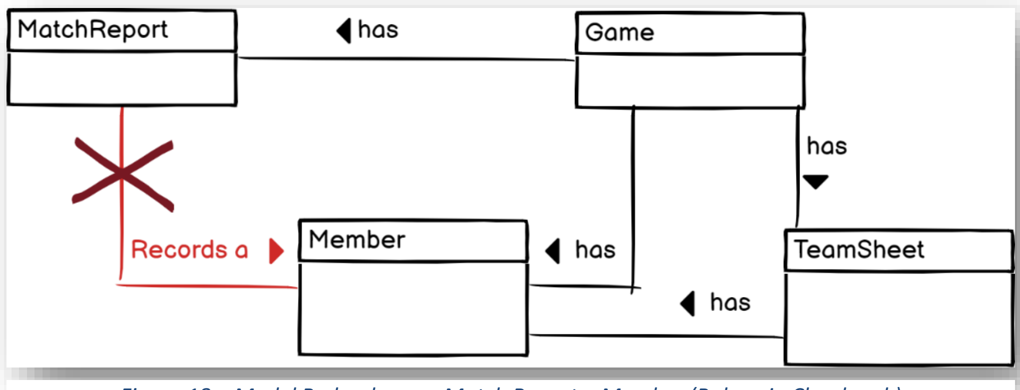


Figure 18 – Model Redundancy – Match Report – Member (Balsamiq Cloud, n.d.)

1.8 Review conceptual data model with user
I did Continuous reviewing with referee and the GAA Official Guide (Referee Handbook)

1.9 Main user transactions
View live scores – live score will include competition name, score time, score club name, score type, score player name, TeamA name, TeamB name, teamA total goals, teamA total points, TeamB total goals, TeamB total points.

CRUD operations for game, card, substitute, injury, team sheet and match report.

3.3.2 Entity Relationship Diagram

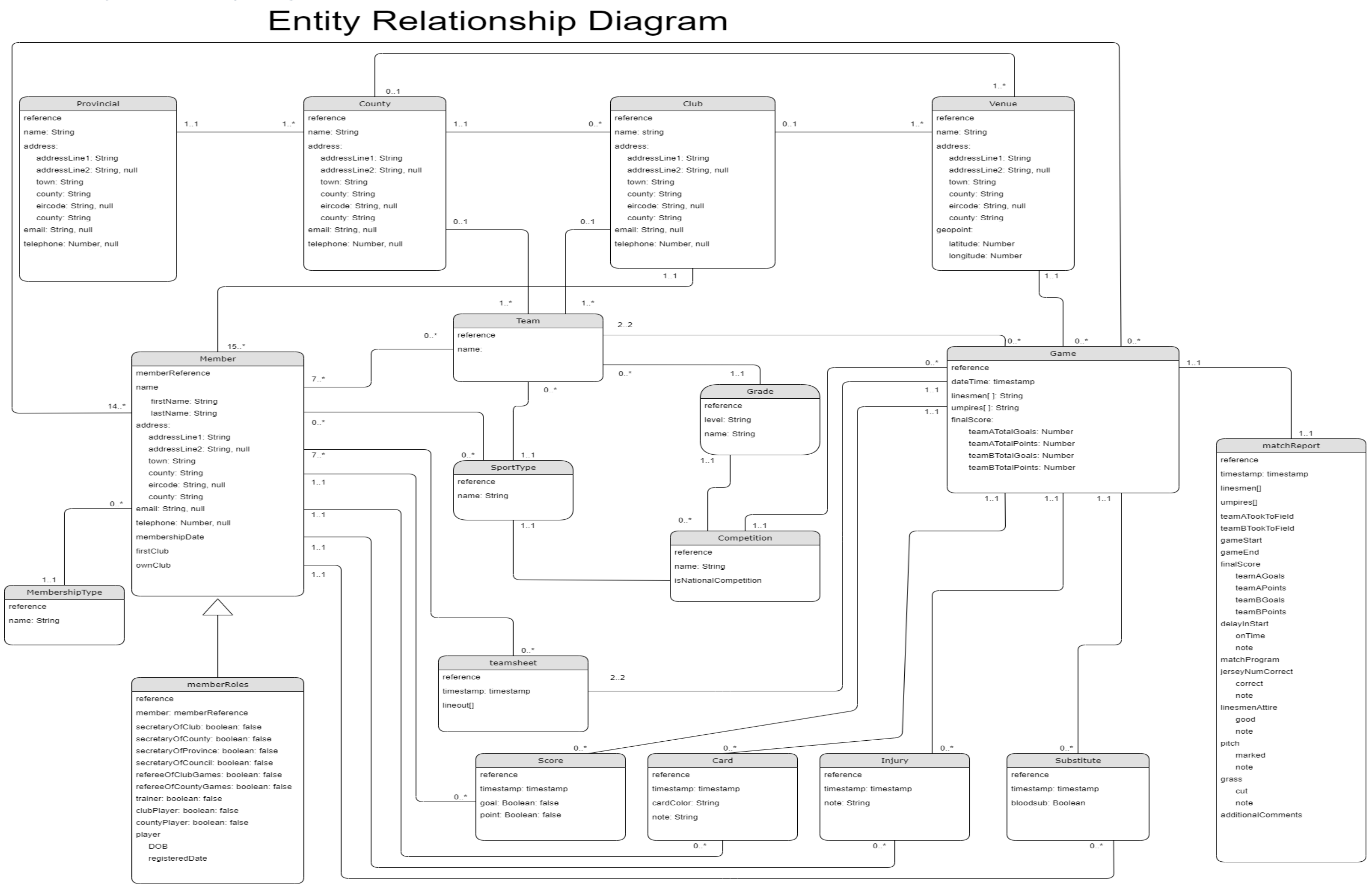


Figure 19 – ERI Diagram (Draw, n.d.)

3.3.3 Database Model

I compared weather to use SQL or NoSQL after my research I decided I would use a NoSQL database.

The GAA organisation has such a big hierarchy with many councils, committees, clubs, competitions, tournaments, officers, members it would require a lot of splitting to store different entities in a relational database and then a lot of joins to retrieve attributes from different tables when information is required.

As there are so many games, scores, cards, substitutes, injuries, team sheets, and match reports which occur each year the database will need to support large volumes of data in years to come. NoSQL databases can handle this better as they can run efficiently on clusters.

NoSQL allows for easier to retrieve data as it is all stored in the one place.

NoSQL is more flexible and scalable. This current model is a simplified model to support the creation and querying of games, scores, cards, substitutes, injuries, match reports, team sheets. If I want to add features such as statistics, registration, or an extended management system for the entire GAA organisation I can take advantage of the NoSQL database and add fields to documents, embed objects and add subcollections. I can scale the model easily.

Atlas MongoDB V Firebase Databases

Atlas MongoDB

One of the most popular NoSQL databases is MongoDB, it is cloud based and supports horizontal scaling using shared clusters and includes fault tolerance using replica sets, storing the replicate at another site, it is also a consistent database. Atlas is fully managed and guarantees

“Availability, scalability and compliance with most demanding data security and privacy standards”. (MongoDB, n.d.)

Cost: shared $0 dedicated $57.00, serverless from $0.30/million reads. (MongoDB Pricing, n.d.)

firebase

Firebase is also cloud based but it has more focus on mobile and web applications. Firebase is scalable. It can be used as a full back-end as a service with minimum effort with connections to authentication, messaging, functions and hosting. It is very user friendly, it supports real-time data, updates in data are recognised without refreshing the database, and real-time data is synchronized across devices and browsers. It has offline mode features meaning the app can write, read, listen to and query data even if the device is offline and has no network. When the device comes back online the data stored locally is synchronized with firebase.

Cost – Pay as You Go (Firebase Pricing, n.d.).

Firestore

(Firestore Pricing, n.d.)

Free for 1GB of storage and costs $.108c per Gb thereafter.

Allows 20K writes per day for free and costs $.18c per 100K thereafter.

Allows 50K reads per day for free and costs $.06c per 100K thereafter.

Allows 20K deletes per day for free and costs $.02c per 100K thereafter.

Realtime Database

(Firebase Pricing, n.d.)

Free for 1Gb of storage and costs $5.00 per Gb thereafter

Free for 10GB downloaded per month and costs $1 per GB thereafter

I have chosen to use firebase over Atlas as it has more focus on android development. Firebase also gives extra functionality to use it as a full back end as a service allowing me to connect to features such as authentication, messaging, functions and hosting. it supports real time data which will be used to update live scores on the database. The data will also be synchronised across the various devices that will be used to connect to the database. It also has the added feature of offline mode which will benefit referees when they are in venues with limited to no network. As I will be storing personal data that will need to be secured and compliance with GDPR is absolutely necessary the fact that firebase encrypts data in transit and at rest in Authentication, Firestore, Realtime Database and Functions is an added bonus as I won't have to carry out further encryption of my own.

Firebase – Realtime Database V Firestore

I compared Firestore and Realtime Database. Both databases have security rules to protect data. integration with functions, authentication, messaging, storage etc. Both have mobile first real-time SDKs. Both support the storing of data locally. Both can work off line. Both can be used in the same app.

Realtime Database

It is easy to store data in the Realtime database, however complex hierarchical data is harder to deal with at scale, it has limited sorting and filtering functionality, both sorting and filtering can't be done together. A query will return the entire subtree, queries don't need an index but this can cause problems if the data set scales. scaling requires sharding. It charges for bandwidth and storage but at higher rates to Firestore.

Firestore

Stores data as collections or documents. Is good for more complex hierarchical data and is easier to scale and organise, it allows the use of subcollections within documents. It doesn't need as much denormalising than the Realtime Database would. Firestore has slightly higher uptime to Realtime Database. Firestore has better filtering and sorting functionality, it allows you to chain filters, and you can do filtering and sorting at the same time. Queries return whole documents in a collection, it doesn't return subcollection data. scaling is automatic. Charges are based on transaction for read, writes and deletes but at a lower rates, bandwidth and storage.

I decided to use Firestore for the hierarchical data model that is needed for the GAA organisation, it has better uptime, better filtering and sorting, scaling is automatic and it has cheaper rates.

I decided to use the real-time database to store the daily live score details, this will take advantage of the two free tiers, reduce the number of queries needed to retrieve the required information, and Live Scores can be deleted daily/weekly as they don't need to be stored long term as the main information is being stored in the Firestore documents that can be used for future information and statistics.

3.3.4 Document Oriented Modelling

Embedding

A game has a one-to-one relationship with match report and team sheets, they are frequently accessed together, and the game and match report have a lot of duplicate fields. I have therefore decided I will use embedding. I will embed the match report and team sheet into the game.

The member has a one-to-one relationship with member role, they are also frequently accessed together, therefore I have decided to embed member role into member.

Referencing

The Province, County, Club and Member all have contact details such as address, telephone and email which will not be acessed often in this case I have decided to use referencing. I will add a Contact Details entity and store a reference to the contact details in the Province, County, Club and Member.

For all other relationships between entities, I will add a reference to one side of the entity.

3.4 Functional Requirements

- A Referee can login to the web application and the android watch.
- On the web application a referee can:
  - View upcoming games they are refereeing.
  - View past games that they have refereed.
  - Create a match report.
  - View a match report.
  - Edit a match report.
  - Delete a match report.
  - View Team Sheets for games they have refereed.
- On the android wear application, a referee can:
  - Start a game.
  - Pause a stopwatch.
  - Start a stopwatch.
  - Reset a stopwatch.
  - End the game.
  - Create a score.
  - View a score.
  - Edit a score.
  - Delete a score.
  - Create a card.
  - View a card.
  - Edit a card.
  - Delete a card.
  - Create a Substitution.
  - View a Substitution.
  - Edit a Substitution.
  - Delete a substitution.
  - Create an Injury
  - View an Injury
  - Edit an Injury
  - Delete and Injury
  - Add the time teams entered the field
  - Add match report additional comments.
  - View team sheets
- A Team Official use the web application to:
  - Login
  - View upcoming games for their team(s)
  - Create a team sheet for an upcoming game
  - View a team sheet for an upcoming game.
  - Edit a team sheet for an upcoming game.
  - Delete a team sheet for an upcoming game.
- A Secretary use the web application to login
- Secretaries can:
  - View games.
  - Create a game.
  - Edit a game.

    - Delete a game.
    - View team sheets for a completed game.
    - View match reports for completed games
- A GAA fan can use the web application to:
    - View live scores.
    - Filter live scores
    - View team sheets

3.5 Non-functional requirements

- Both the web application and the android application should be secure.
- There should be access control.
- It should have good UI/UX so that it is clear and easy to use.
- The android watch should be quick and easy to navigate.

3.6 User interface design

3.6.1 GAA Fan Web Application

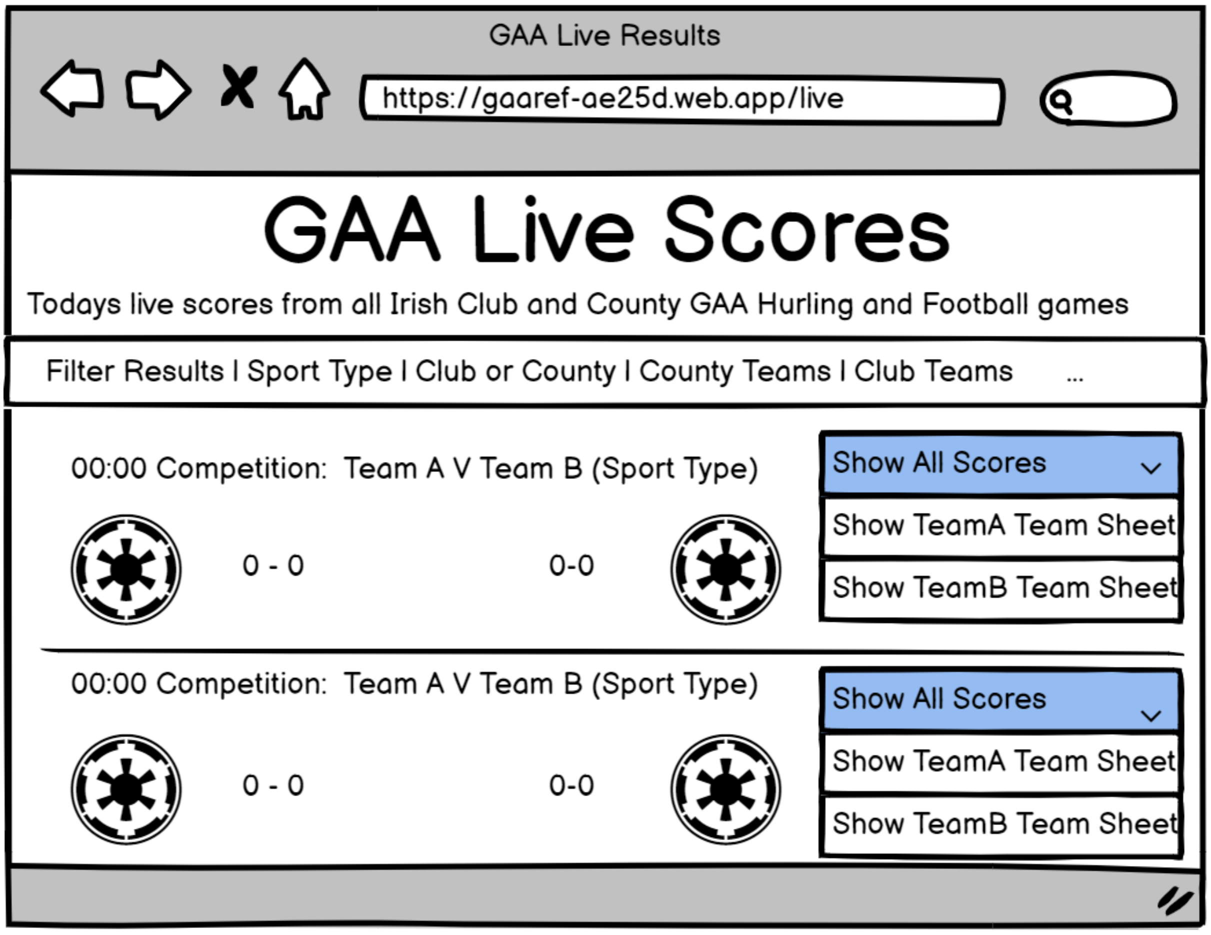

GAA Live Results
https://gaaref-ae25d.web.app/live
GAA Live Scores
Todays live scores from all Irish Club and County GAA Hurling and Football games
Filter Results I Sport Type I Club or County I County Teams I Club Teams ...
00:00 Competition: Team A V Team B (Sport Type)
0 - 0
0-0
Show All Scores
Show TeamA Team Sheet
Show TeamB Team Sheet
00:00 Competition: Team A V Team B (Sport Type)
0 - 0
0-0
Show All Scores
Show TeamA Team Sheet
Show TeamB Team Sheet

Figure 20 – User Interface Design – GAA Fan Web App (Balsamiq Cloud, n.d.)

3.6.2 GAA Game Management

Login – Home Page

GAA Secretaries, Team Management and Referees must first be authenticated by firebase in order to login.

Depending on the Role you will be redirected to the corresponding dashboard

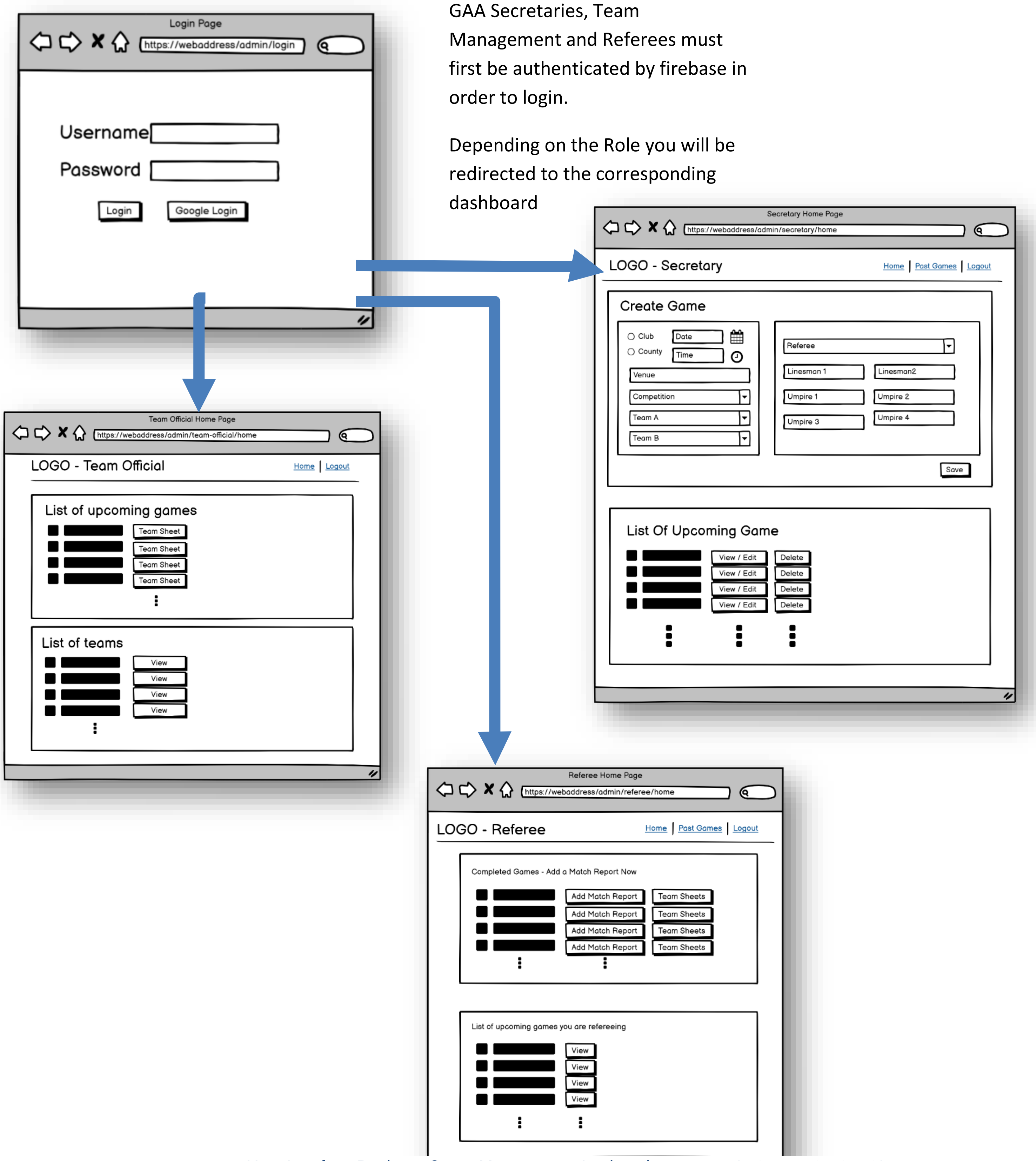


Figure 21 – User Interface Design – Game Management Login to home page (Balsamiq Cloud, n.d.)

Secretary

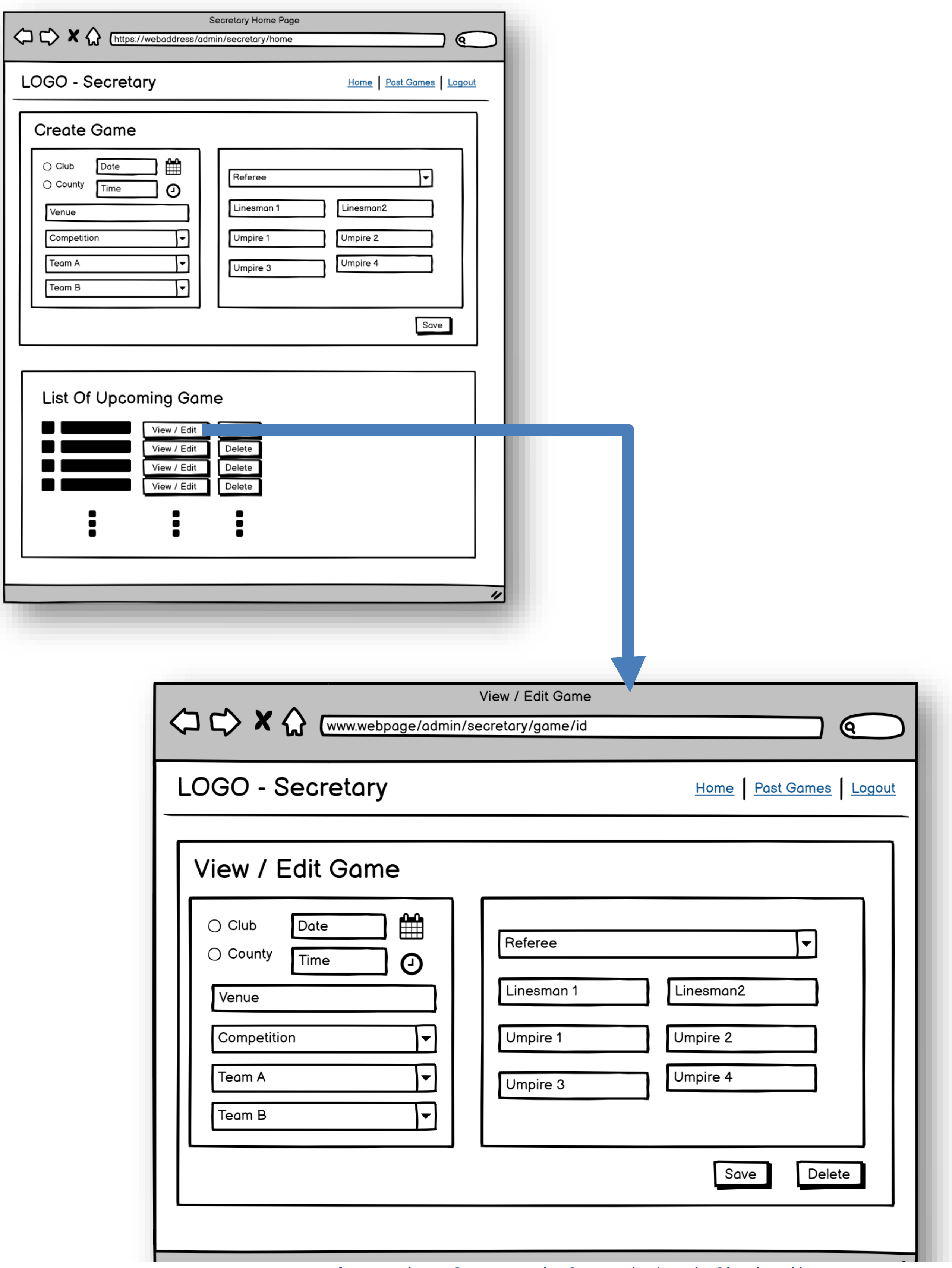


Figure 22 – User Interface Design – Secretary List Games (Balsamiq Cloud, n.d.)

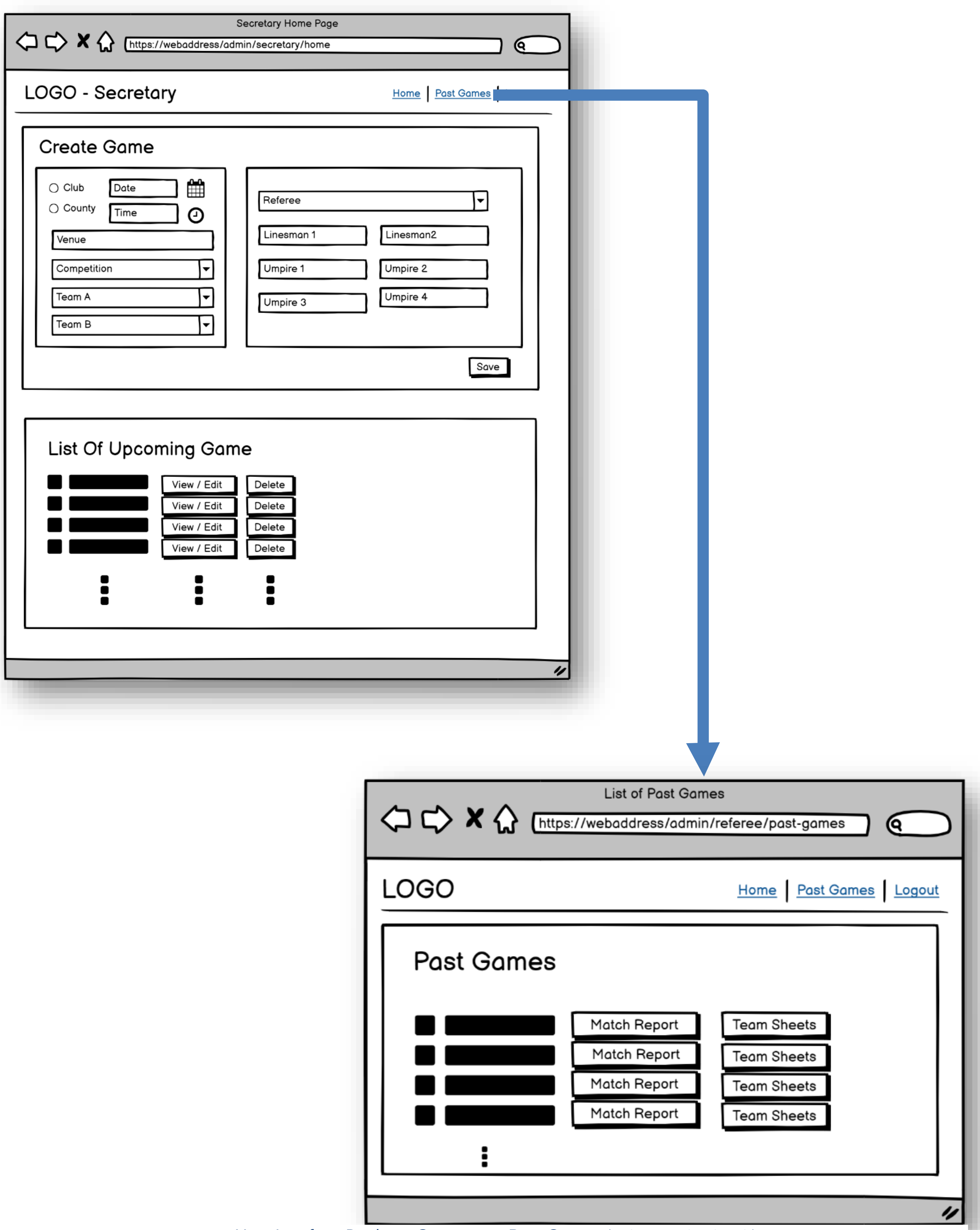


Figure 23 – User Interface Design – Secretary – Past Games (Balsamiq Cloud, n.d.)

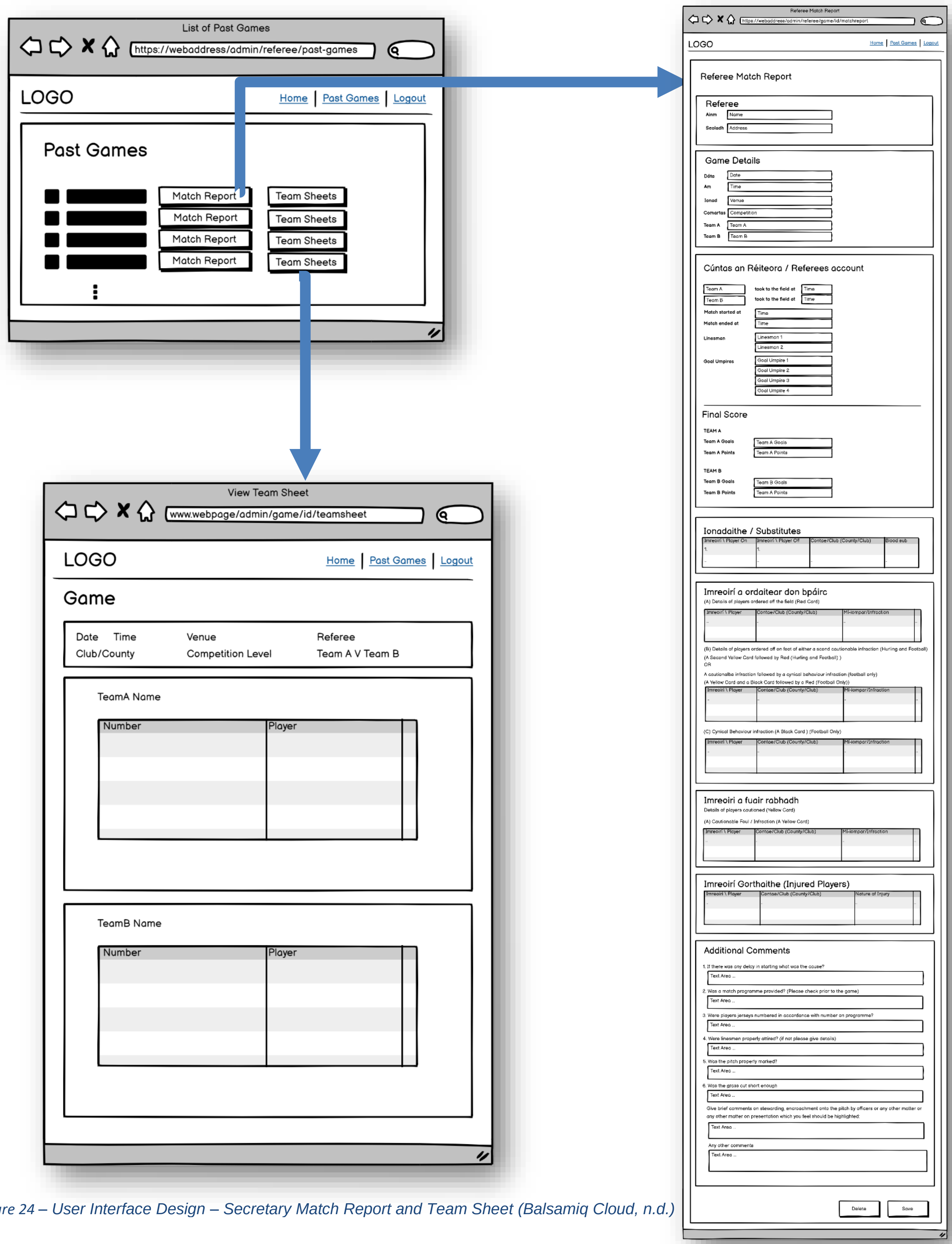
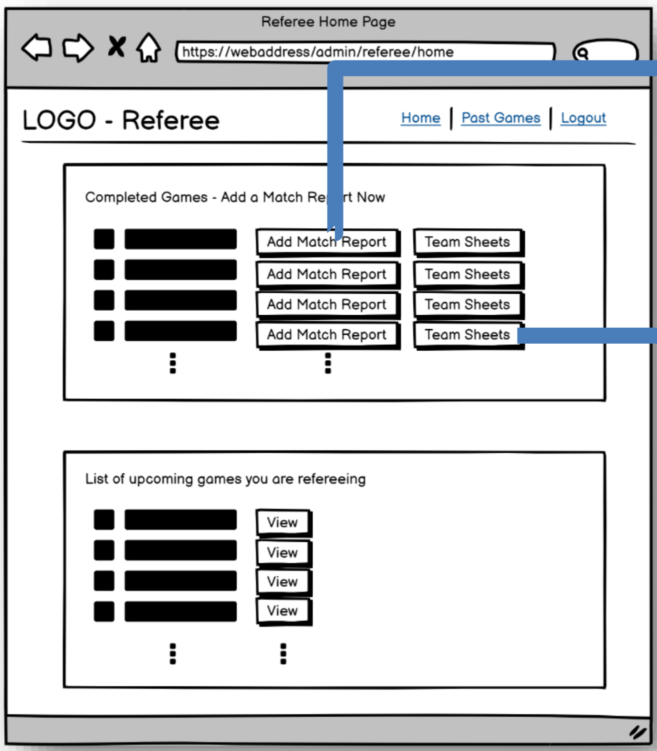

List of Past Games
https://webaddress/admin/referee/past-games
LOGO
Home | Past Games | Logout
Past Games
Match Report
Team Sheets
Match Report
Team Sheets
Match Report
Team Sheets
Match Report
Team Sheets
View Team Sheet
www.webpage/admin/game/id/teamsheet
LOGO
Home | Past Games | Logout
Game
Date Time
Venue
Referee
Club/County
Competition Level
Team A V Team B
TeamA Name
Number
Player
TeamB Name
Number
Player
Referee Match Report
https://webaddress/admin/referee/game/id/matchreport
LOGO
Home | Past Games | Logout
Referee Match Report
Referee
Ainm
Name
Seoladh
Address
Game Details
Dáta
Date
Am
Time
Ionad
Venue
Comartas
Competition
Team A
Team A
Team B
Team B
Cúntas an Réiteora / Referees account
Team A
took to the field at
Time
Team B
took to the field at
Time
Match started at
Time
Match ended at
Time
Linesman
Linesman 1
Linesman 2
Goal Umpires
Goal Umpire 1
Goal Umpire 2
Goal Umpire 3
Goal Umpire 4
Final Score
TEAM A
Team A Goals
Team A Goals
Team A Points
Team A Points
TEAM B
Team B Goals
Team B Goals
Team B Points
Team A Points
Ionadaithe / Substitutes
Imreoirí \ Player On
Imreoirí \ Player Off
Contae/Club (County/Club)
Blood sub
Imreoirí a ordaitear don bpáirc
(A) Details of players ordered off the field (Red Card)
Imreoirí \ Player
Contae/Club (County/Club)
Mí-iompar/Infraction
(B) Details of players ordered off on foot of either a scond cautionable infraction (Hurling and Football)
(A Second Yellow Card followed by Red (Hurling and Football))
OR
A cautionalbe infraction followed by a cynical behaviour infraction (football only)
(A Yellow Card and a Black Card followed by a Red (Football Only))
Imreoirí \ Player
Contae/Club (County/Club)
Mí-iompar/Infraction
(C) Cynical Behaviour infraction (A Black Card) (Football Only)
Imreoirí \ Player
Contae/Club (County/Club)
Mí-iompar/Infraction
Imreoiri a fuair rabhadh
Details of players cautioned (Yellow Card)
(A) Cautionable Foul / Infraction (A Yellow Card)
Imreoirí \ Player
Contae/Club (County/Club)
Mí-iompar/Infraction
Imreoirí Gorthaithe (Injured Players)
Imreoirí \ Player
Contae/Club (County/Club)
Nature of Injury
Additional Comments
1. If there was any delay in starting what was the cause?
Text Area ...
2. Was a match programme provided? (Please check prior to the game)
Text Area ...
3. Were players jerseys numbered in accordance with number on programme?
Text Area ...
4. Were linesmen properly attired? (if not please give details)
Text Area ...
5. Was the pitch properly marked?
Text Area ...
6. Was the grass cut short enough
Text Area ...
Give brief comments on stewarding, encroachment onto the pitch by officers or any other matter or any other matter on presentation which you feel should be highlighted:
Text Area ...
Any other comments
Text Area ...
Delete
Save

Figure 24 – User Interface Design – Secretary Match Report and Team Sheet (Balsamiq Cloud, n.d.)



Referee Home Page

https://webaddress/admin/referee/home

LOGO - Referee

Home | Past Games | Logout

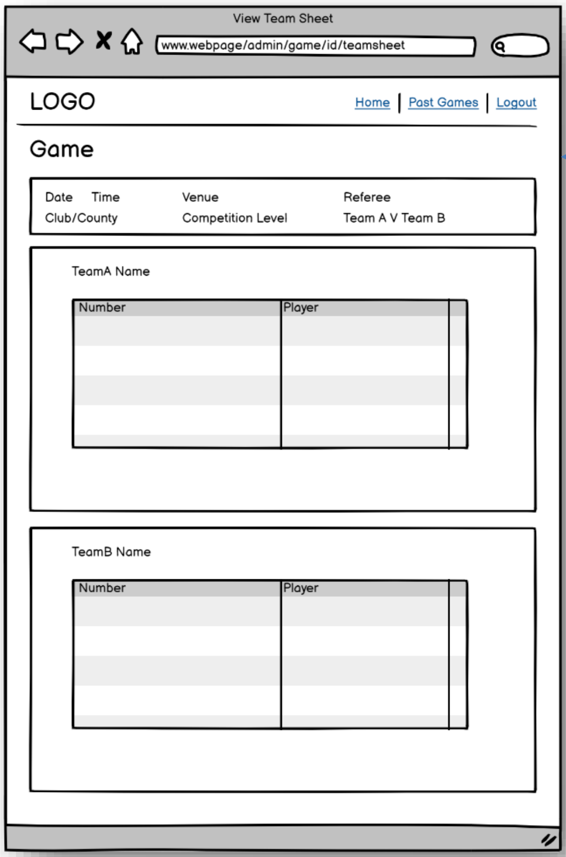
Completed Games - Add a Match Report Now

[redacted]	Add Match Report	Team Sheets
[redacted]	Add Match Report	Team Sheets
[redacted]	Add Match Report	Team Sheets
[redacted]	Add Match Report	Team Sheets
⋮	⋮	

List of upcoming games you are refereeing

[redacted]	View
[redacted]	View
[redacted]	View
[redacted]	View
⋮	⋮

Referee



View Team Sheet

www.webpage/admin/game/id/teamsheet

LOGO

Home | Past Games | Logout

Game

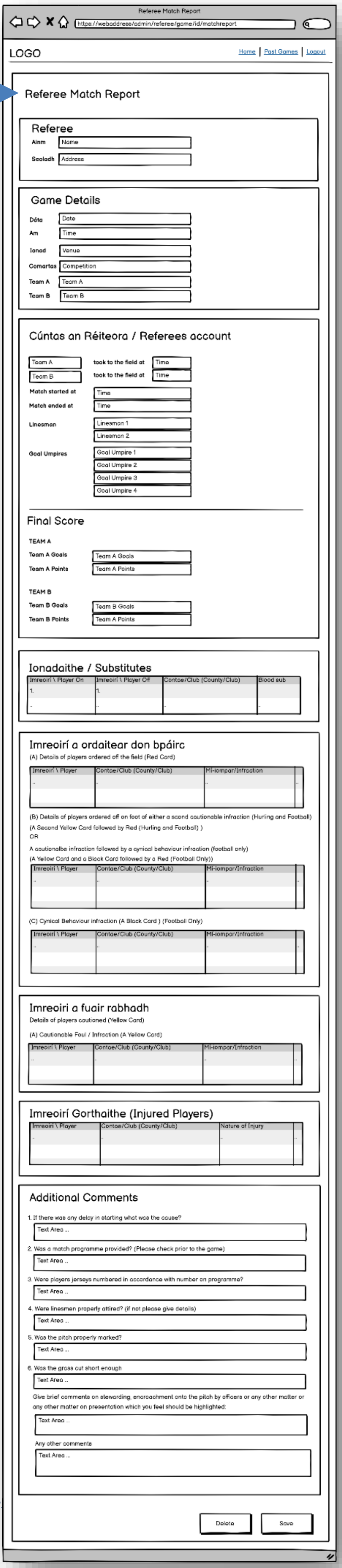
Date Time	Venue	Referee
Club/County	Competition Level	Team A V Team B

TeamA Name

Number	Player

TeamB Name

Number	Player



Referee Match Report

https://webaddress/admin/referee/game/id/matchreport

LOGO

Home | Past Games | Logout

Referee Match Report

Referee

Ainm	Name
Seoladh	Address

Game Details

Dáta	Date
Am	Time
Ionad	Venue
Comortas	Competition
Team A	Team A
Team B	Team B

Cúntas an Réiteora / Referees account

Team A	took to the field at	Time
Team B	took to the field at	Time
Match started at	Time	
Match ended at	Time	
Linesman	Linesman 1	
	Linesman 2	
Goal Umpires	Goal Umpire 1	
	Goal Umpire 2	
	Goal Umpire 3	
	Goal Umpire 4	

Final Score

TEAM A

Team A Goals	Team A Goals
Team A Points	Team A Points

TEAM B

Team B Goals	Team B Goals
Team B Points	Team A Points

Ionadaithe / Substitutes

Imreoirí \ Player On	Imreoirí \ Player Off	Contae/Club (County/Club)	Blood sub
1.	1.		
..	..	..	..

Imreoirí a ordaitear don bpáirc

(A) Details of players ordered off the field (Red Card)

Imreoirí \ Player	Contae/Club (County/Club)	Mí-iompar/Infraction	
..	..	..	..

(B) Details of players ordered off on foot of either a scond cautionable infraction (Hurling and Football)

(A Second Yellow Card followed by Red (Hurling and Football))

OR

A cautionalbe infraction followed by a cynical behaviour infraction (football only)

(A Yellow Card and a Black Card followed by a Red (Football Only))

Imreoirí \ Player	Contae/Club (County/Club)	Mí-iompar/Infraction	
..	..	..	..

(C) Cynical Behaviour infraction (A Black Card) (Football Only)

Imreoirí \ Player	Contae/Club (County/Club)	Mí-iompar/Infraction	
..	..	..	..

Imreoiri a fuair rabhadh

Details of players cautioned (Yellow Card)

(A) Cautionable Foul / Infraction (A Yellow Card)

Imreoirí \ Player	Contae/Club (County/Club)	Mí-iompar/Infraction	
..	..	..	..

Imreoirí Gorthaithe (Injured Players)

Imreoirí \ Player	Contae/Club (County/Club)	Nature of Injury	
..	..	..	..

Additional Comments

1. If there was any delay in starting what was the cause?

Text Area ...

2. Was a match programme provided? (Please check prior to the game)

Text Area ...

3. Were players jerseys numbered in accordance with number on programme?

Text Area ...

4. Were linesmen properly attired? (if not please give details)

Text Area ...

5. Was the pitch properly marked?

Text Area ...

6. Was the grass cut short enough

Text Area ...

Give brief comments on stewarding, encroachment onto the pitch by officers or any other matter or any other matter on presentation which you feel should be highlighted:

Text Area ...

Any other comments

Text Area ...

Delete | Save

Figure 25 - User Interface Design – Referee Match Report and Team Sheet (Balsamiq Cloud, n.d.)

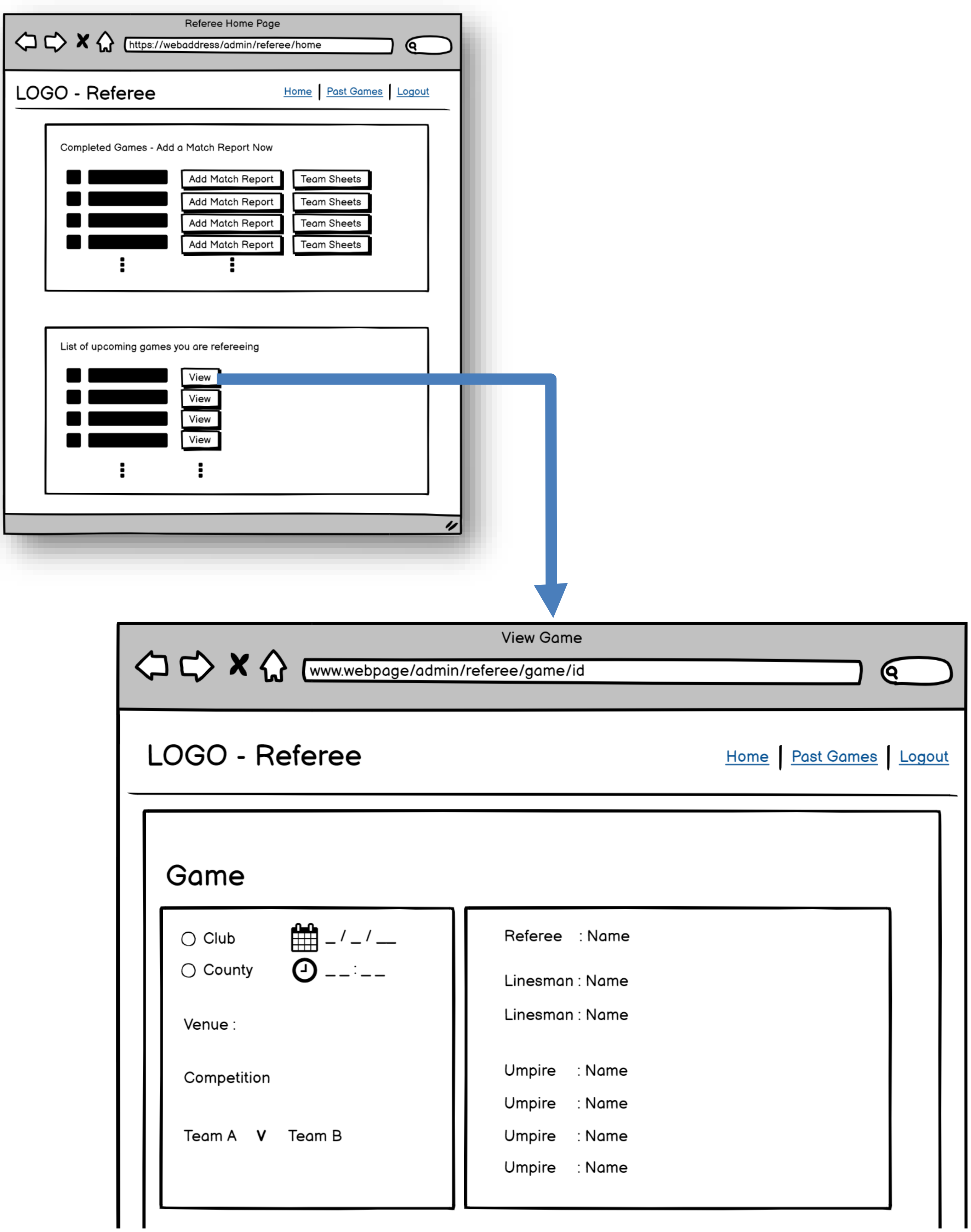


Figure 26 - User Interface Design – Referee – View upcoming games (Balsamiq Cloud, n.d.)

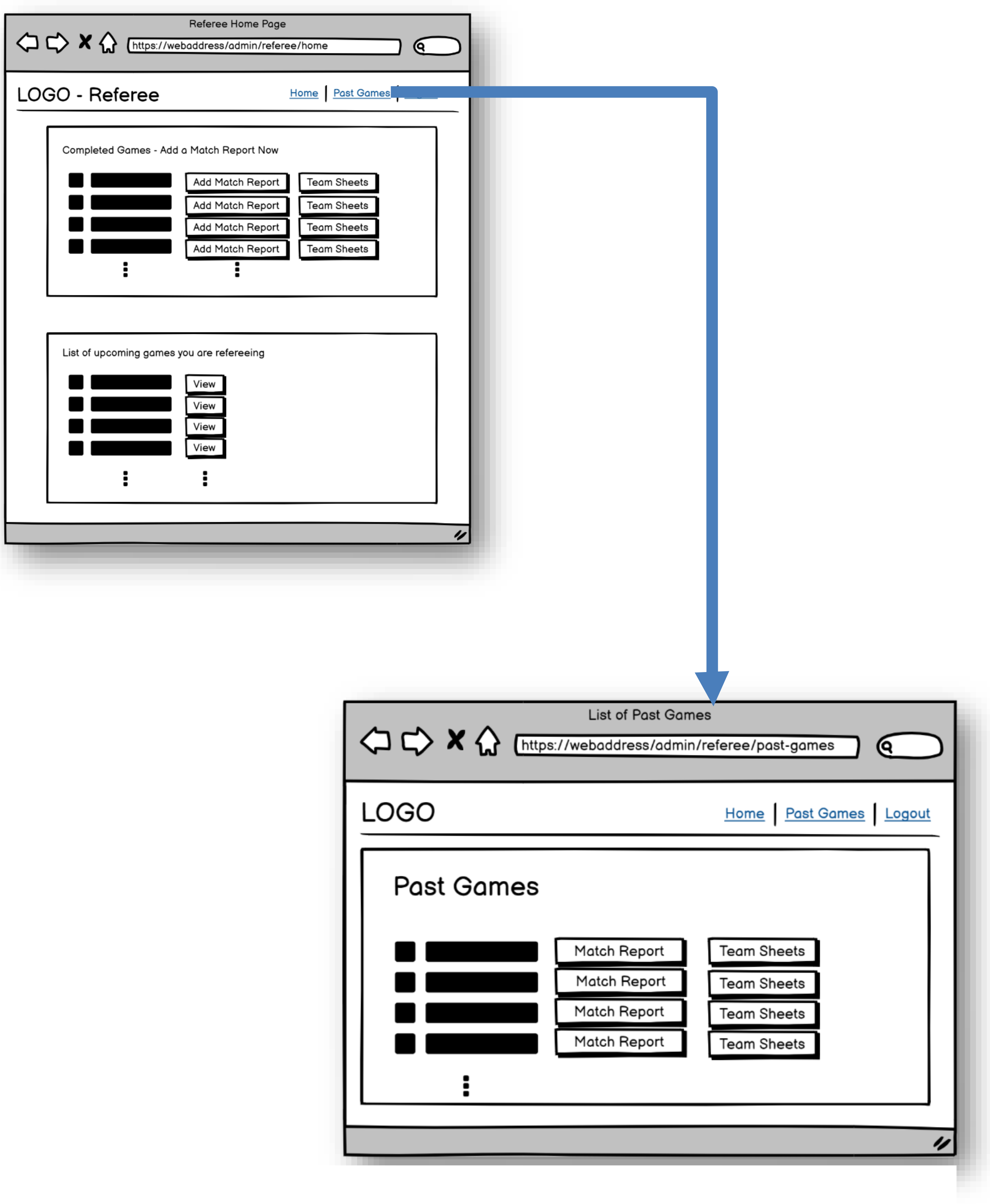

Referee Home Page
https://webaddress/admin/referee/home
LOGO - Referee
Home
Past Games
Completed Games - Add a Match Report Now
Add Match Report
Team Sheets
List of upcoming games you are refereeing
View
List of Past Games
https://webaddress/admin/referee/past-games
LOGO
Home
Past Games
Logout
Past Games
Match Report
Team Sheets

Figure 27 - User Interface Design – Referee – View Past Games (Balsamiq Cloud, n.d.)

Team Official

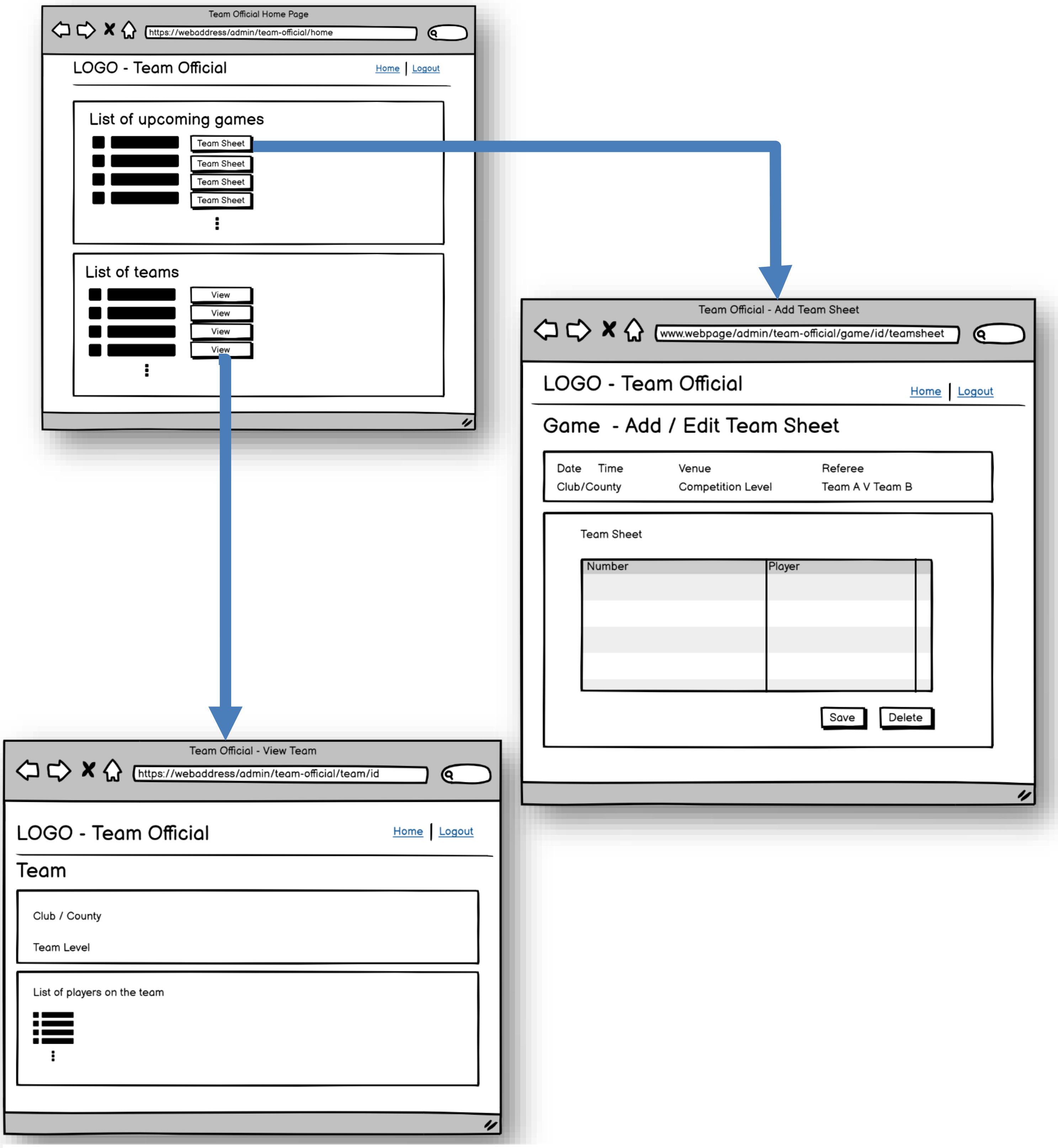


Figure 28 - User Interface Design – Team Official View Past Games and Single Games (Balsamiq Cloud, n.d.)

Android Wear Application

Google Login

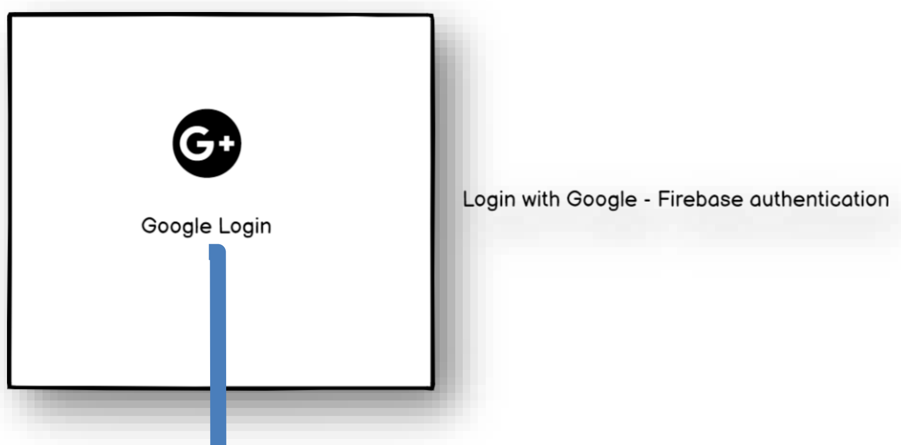


Figure 29 – Android Wear Google Login

List of games

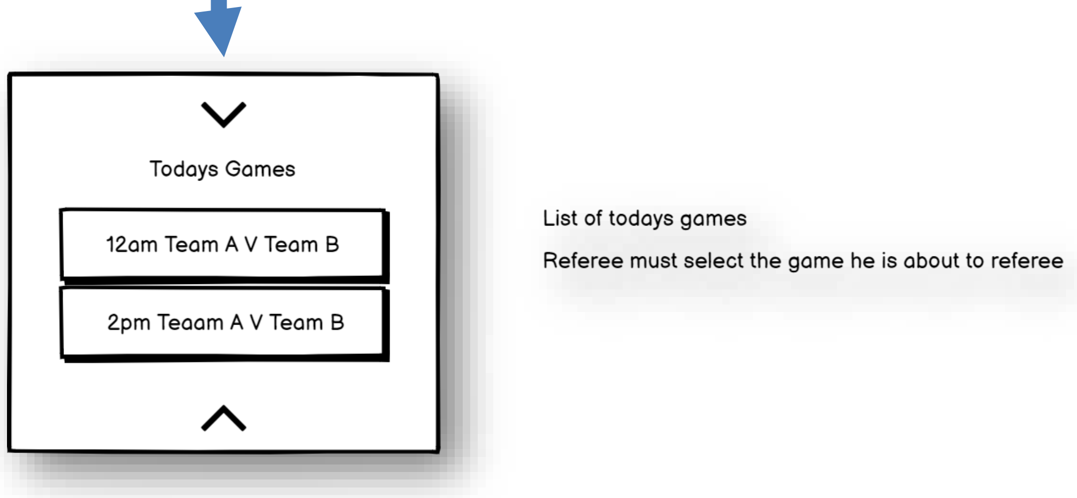


Figure 30 - User Interface Design – Android Wear View list of games (Balsamiq Cloud, n.d.)

Menu layout

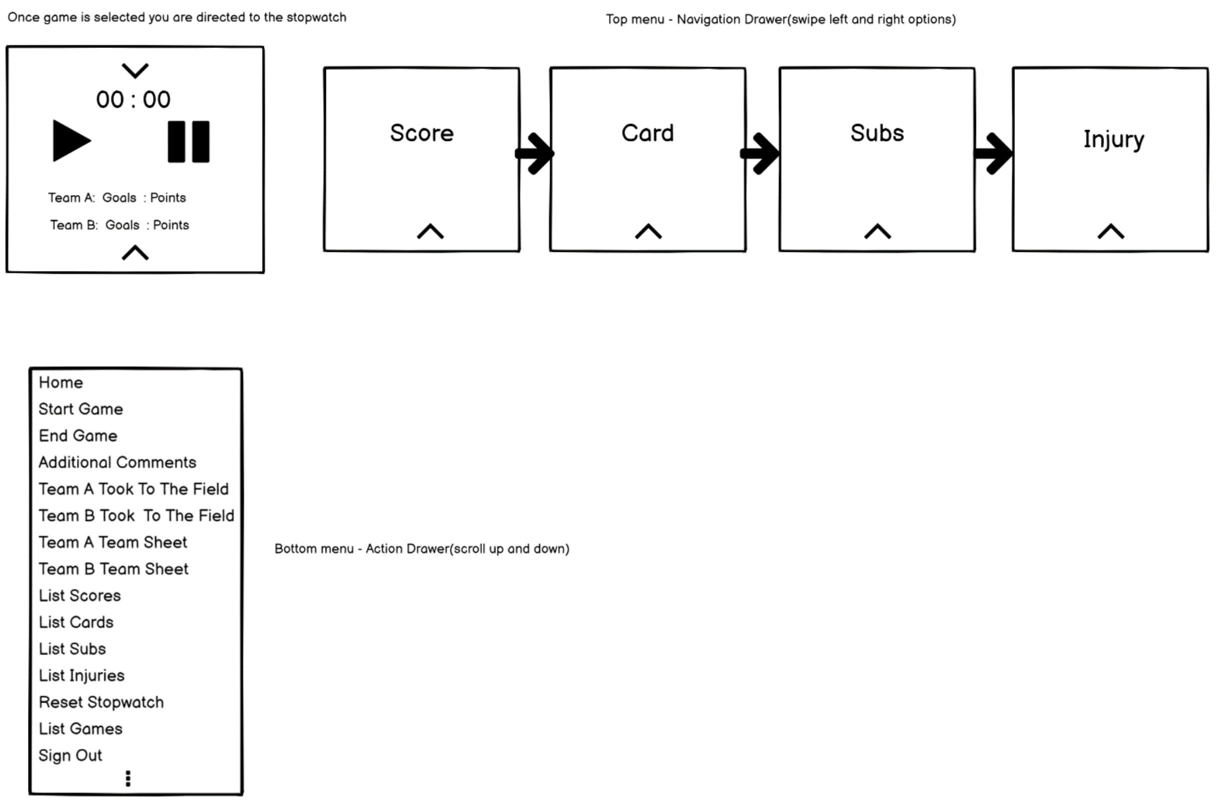


Figure 31 - User Interface Design – Android Wear Menu System (Balsamiq Cloud, n.d.)

Create / Edit Score

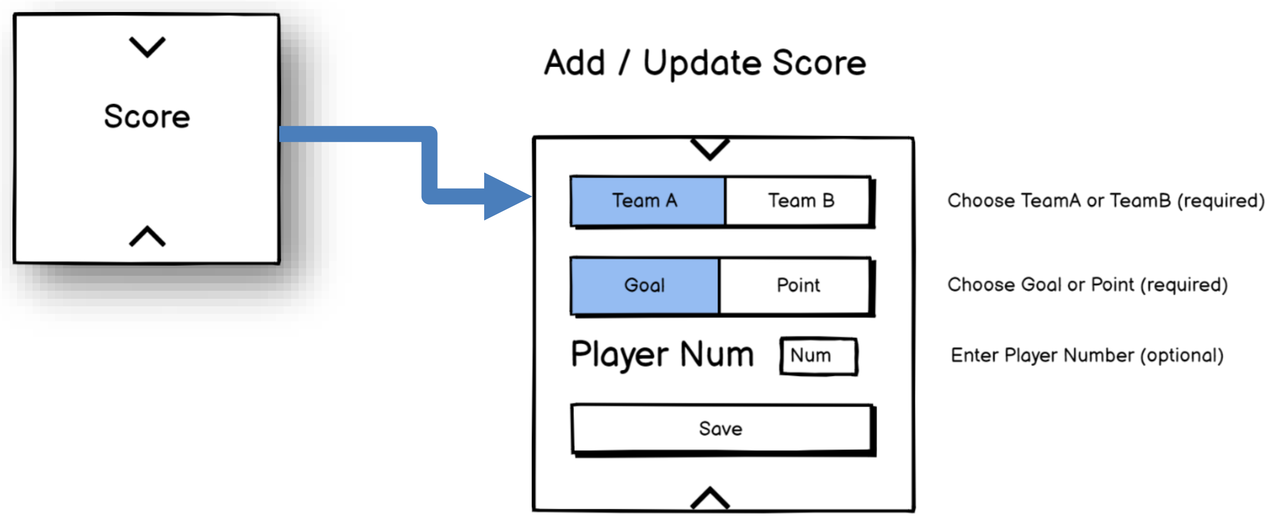

Score
Add / Update Score
Team A
Team B
Choose TeamA or TeamB (required)
Goal
Point
Choose Goal or Point (required)
Player Num
Num
Enter Player Number (optional)
Save

Figure 32 - User Interface Design – Android Wear Add/Update Score (Balsamiq Cloud, n.d.)

Create / Edit Card

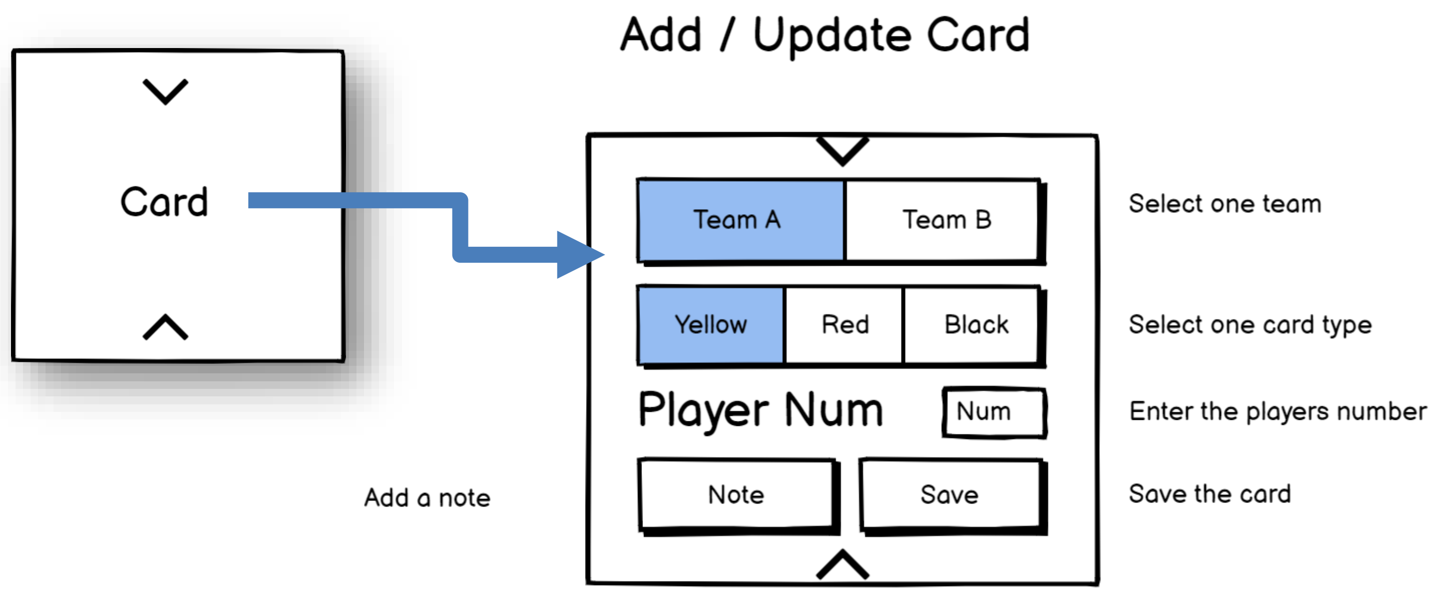


Figure 33 - User Interface Design – Android Wear add/update card (Balsamiq Cloud, n.d.)

Create / Edit Substitutes

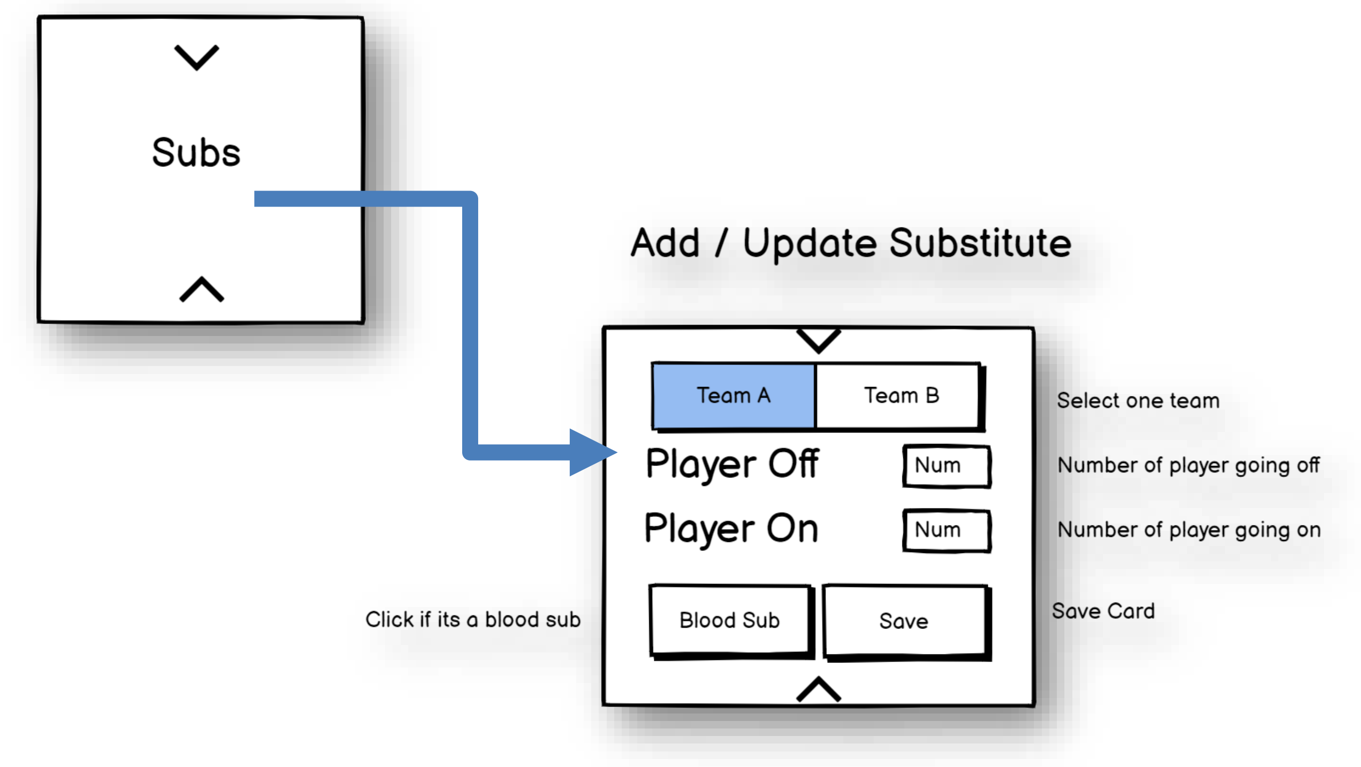


Figure 34 - User Interface Design – Android Wear add/update substitute (Balsamiq Cloud, n.d.)

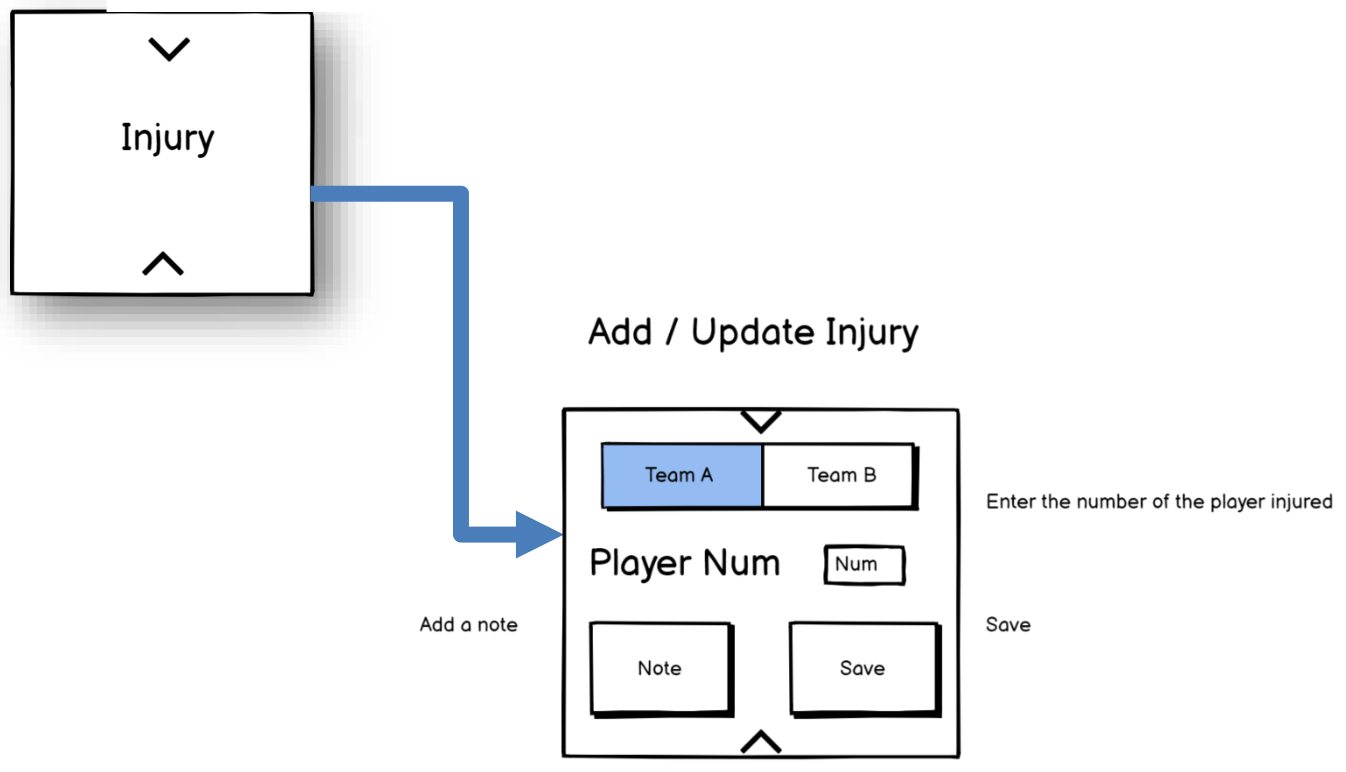


Figure 35 - User Interface Design – Android Wear add/update Injury (Balsamiq Cloud, n.d.)

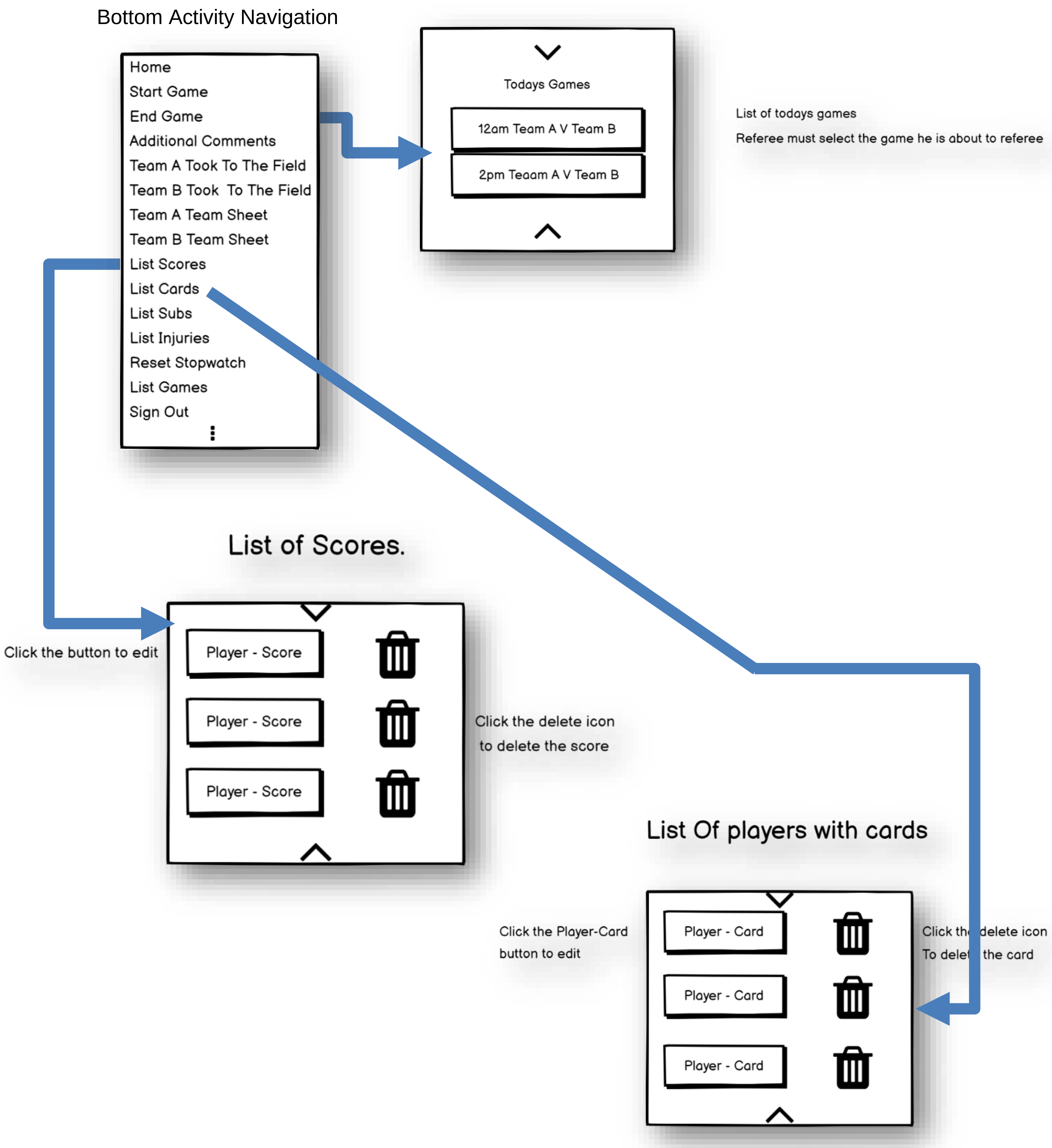


Figure 36 - User Interface Design – Android Wear Bottom Menu (Balsamiq Cloud, n.d.)

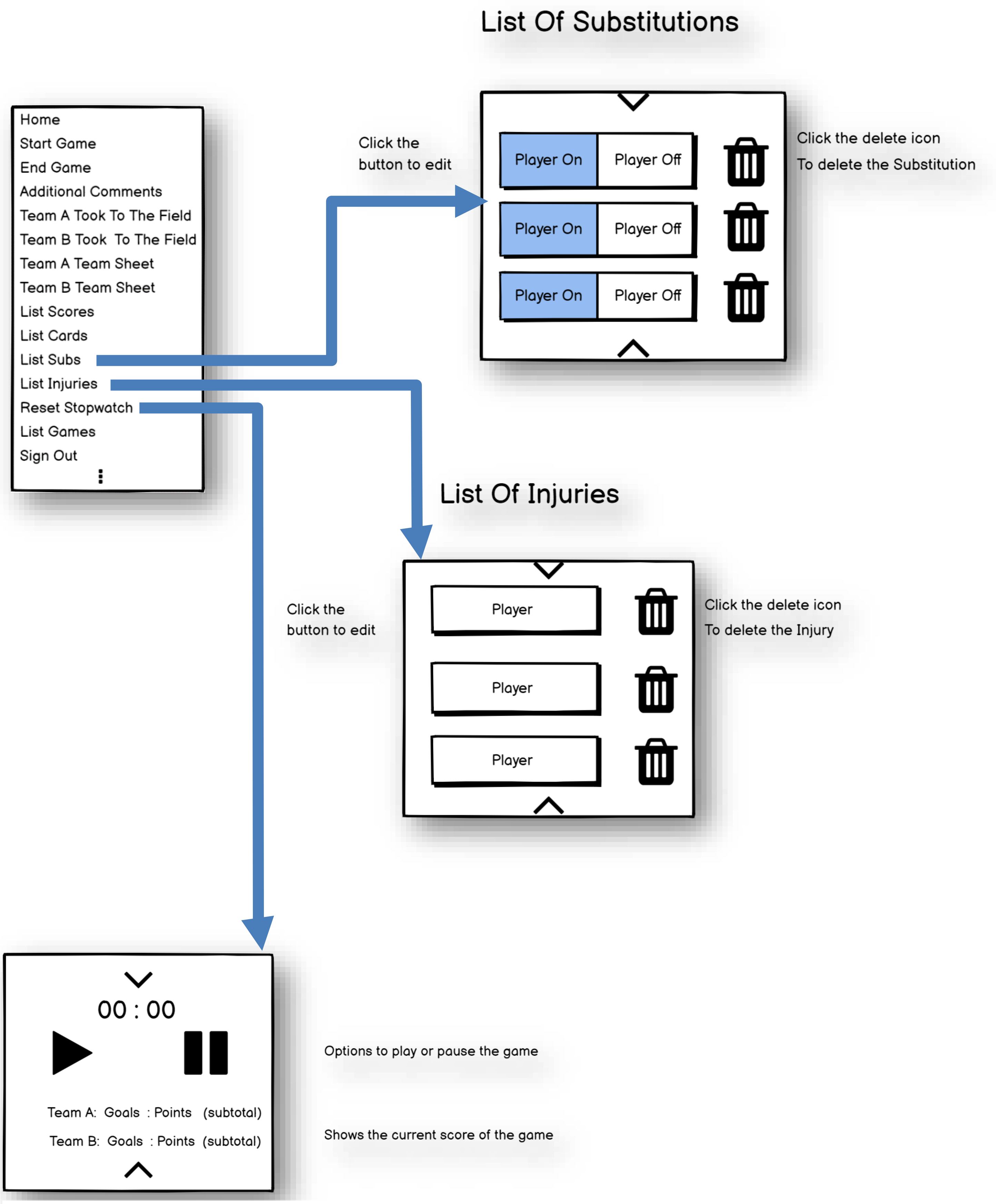


Figure 37 – Android Wear – Bottom Menu (Balsamiq Cloud, n.d.)

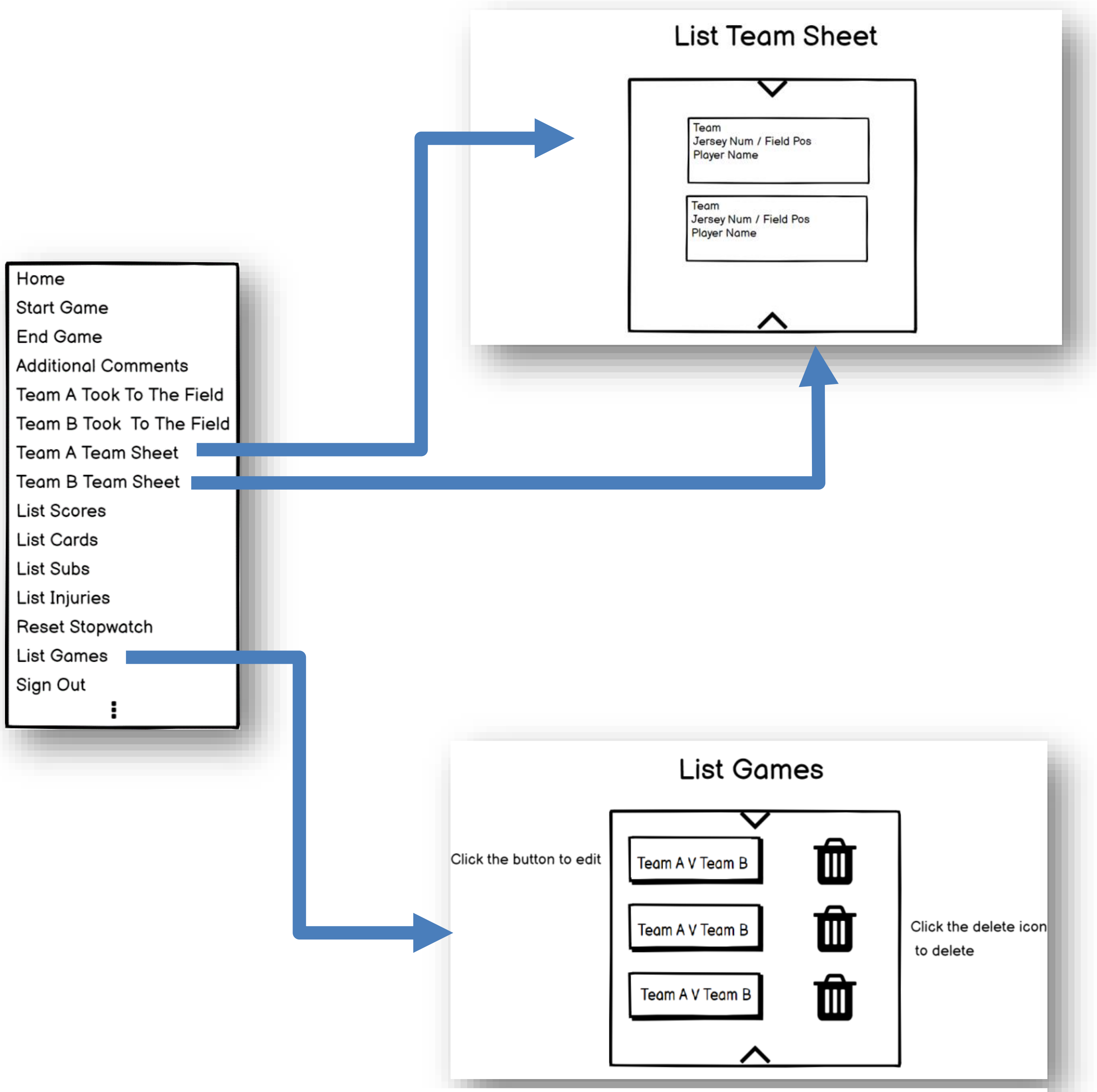


Figure 38 - Android Wear – Bottom Menu (Balsamiq Cloud, n.d.)

4. Methodology

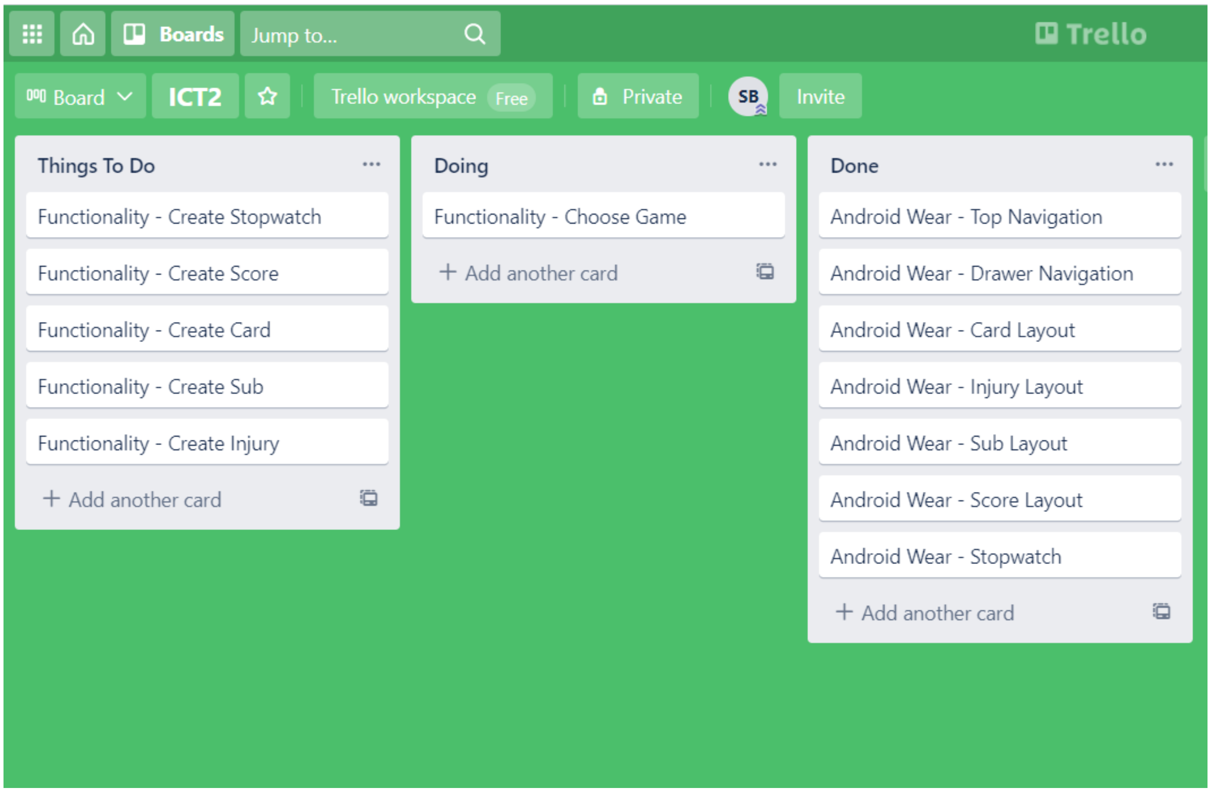
4.1 Methodology selection

I found as the project had so many different areas to complete looking at the project as a whole was overwhelming so I decided to use an agile methodology in order to plan and develop the project. I used weekly sprints to break the project down into smaller sections and set goals for the week ahead.

Using an agile approach allowed me to break all parts of the project in sections and then split each section into smaller parts, this really helped as it allowed me to focus on just one part or a few smaller parts for the week ahead.

4.2 Process outline

I used a software tool called Trello (Trello, n.d.) to create lists of things I needed to do and things that I had completed. This allowed me to see my progress and what my plan for the week was clearly and visually. Each day I worked on the project I also used Trello to complete a goals list for the day. This boosted motivation each time you completed something on your Todo list and moved it to the completed list.



Things To Do	Doing	Done
Functionality - Create Stopwatch	Functionality - Choose Game	Android Wear - Top Navigation
Functionality - Create Score	+ Add another card	Android Wear - Drawer Navigation
Functionality - Create Card		Android Wear - Card Layout
Functionality - Create Sub		Android Wear - Injury Layout
Functionality - Create Injury		Android Wear - Sub Layout
+ Add another card		Android Wear - Score Layout
		Android Wear - Stopwatch
		+ Add another card

Figure 39 – Trello (Monosnap, n.d.)

4.3 Project Schedule

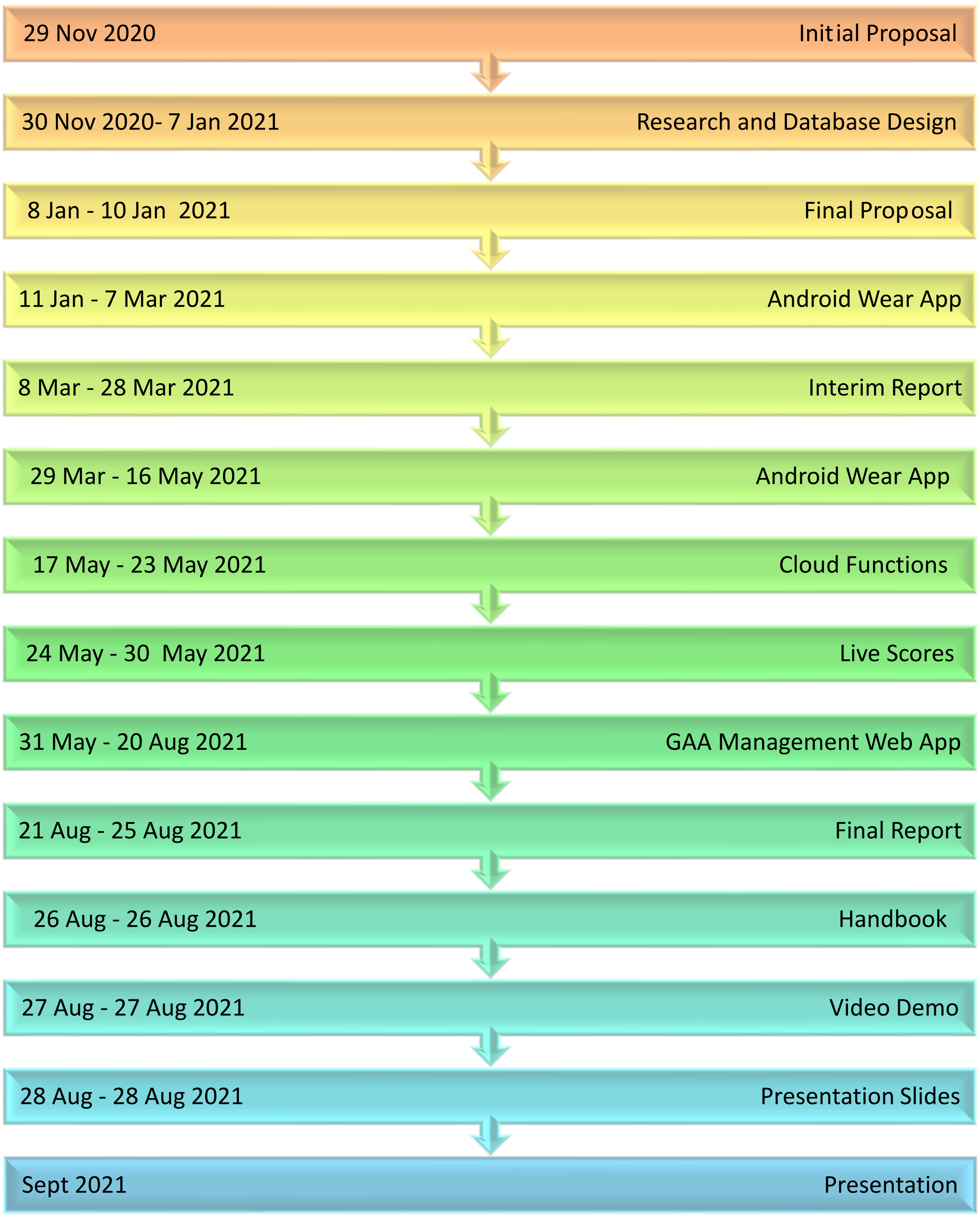


Figure 40 – Project Schedule

5. Implementation

#	weeks	Start Date	End Date	Action
1	1	22/11/2020	29/11/2010	Initial proposal – background, aims, technologies, tools and frameworks, proposed process,
2	2-8	30/11/2020	10/01/2021	Mobile Development Module and assignment
3	9	11/01/2021	17/01/2021	Decide on and design the database needed for the application
4	10	18/01/2021	24/01/2021	Design use cases and wireframes for the project
	11-16	25/01/2021	21/02/2021	Break due to covid and home schooling
5	17	22/02/2021	28/02/2021	Create the base web application with Node JS
6	18	01/03/2021	07/03/2021	Create a Firebase project and connect to the Node JS project
7	19	08/03/2021	14/03/2021	Setup Firebase authentication in the Node JS Web App
8	20	15/03/2021	21/03/2021	Research Android Wear Development, read android wear documentation, do some of their code labs
9	21	22/03/2021	28/03/2021	Setup the starter android wear application and setup google authentication with firebase
10	22	29/03/2021	04/04/2021	Setup Layouts for the android wear app – Navigation, Fragments for Stopwatch, Score, Card, Substitute, Injury
11	23	05/04/2021	11/04/2021	Create Database, JSON objects for all entities provinces, counties, clubs, venues, contact details, grades, competitions, sports type, members, teams, games, scores, substitutes, injuries and cards. Write a program to populate Firestore with the JSON objects
12	24	12/04/2021	18/4/2021	Start the Interm Report – use case diagrams, wireframes and database design
13	25	19/4/2021	25/4/2021	Create List views for Scores, Cards, Substitutes and Injuries on the wear app. Using fragments and recycler views
14	26	26/4/2021	2/4/2021	Create a Stopwatch for the wear application and – Complete Interm report
15	27	3/5/2021	9/5/2021	Create Score, Card, Substitute and Injury
16	28	10/5/2021	16/5/2021	Add voice notes for cards, injuries and general notes
17	29	17/5/2021	23/5/2021	Create Cloud Functions to schedule a daily retrieval of all games happening on that day and a function that listen for scores from the referees watch and updates the Realtime database.
18	30	24/5/2021	30/5/2021	Create a Svelte Web App to listen for live scores and update the UI. Add a Filter and team sheets
19	31	31/5/2021	6/6/2021	Setup the GAA Officials App and Login to Firebase
20	32	7/6/2021	13/6/2021	Setup the GAA Officials App navigation, persistence of svelte store to local storage, retrieve upcoming games for a referee, retrieve all the documents for the games firebase reference and retrieve the information needed to be viewed on svelte frontend.
21	33	14/6/2021	20/6/2021	Retrieve upcoming game details for club and county secretaries from Firestore. Change the layout of the dashboard as the user can have more than one role in the GAA. Add an Accordion of upcoming games for each Role.
	34-35	21/6/2021	4/7/2021	Family Bereavements
22	36	5/7/2021	11/7/2021	Retrieve upcoming game details for provincial and council secretaries and team official's games from Firestore. Add a button for each game to redirect to a single game.
23	37	12/7/2021	18/7/2021	Create a single game page layout, retrieve the game from the local storage, if it doesn't exist there retrieve it from Firestore. Display the details. Add buttons to link the match report and team sheets and add authentication to who can view and use this link.
	38-40	19/7/2021	8/8/2021	Holidays
24	41	9/8/2021	15/8/2021	View Match Report – retrieve details from Firestore and create the view in the web app with svelte. Council Secretary. Provincial Secretary, County Secretary Create game and add the details to Firestore.
25	40	16/8/2021	22/8/2021	Team official – add team sheet and save to firestore. Setup hosting for the game officials web application. Add times for game start, end, teams entering the field and extra information to the android watch
26	41	23/8/2021	29/8/2021	Final Report Video, Presentation Slides, and Submit project.

Table 1 Implementation

6. Project Evaluation

6.1 What were the Outcomes of the project

The objectives of this project were

- To create a basic management web application for secretaries to create, view, update and delete games, view team officials team sheets and view the referees match reports, for team officials to view upcoming games and create, view, update and delete team sheets and for referees to view upcoming and past games, create, view, update and delete match reports and view team sheets linked to games the referee was involved in.
- To provide the referee with an android wear application that can create, update, delete and view scores, cards, substitutes, injuries and run a stopwatch functionality.
- To provide the GAA fans with a web application for live scores that could be filtered to the user's preference.

6.1.1 Referee Android Wear Application

The following is a list of the objectives I met for the referee's android wear application

- Google Login and Logout.
- Retrieve and view today's games.
- Start, stop, pause, reset and view stopwatch.
- View stopwatch, teams and total score on the home layout.
- Create a score (team, score type and player).
    - Check to make sure the player being inputted for the score is on the field.
- Create a card (team, card type, player and notes).
    - Checks if the player being given the card is on the field.
    - If a yellow card is given a check is made to see if the player is already on a yellow card. If they are a message is shown to the referee to send player off.
    - If a black card is given a check is made to see if the player is already on a black card. If they are a message is shown to the ref to send the player off and another player can-not replace this player
    - If a red card a message is shown to the referee to send the player off
- Create a substitute with team, player on, player off, blood sub and black card options.
    - A check is made to make sure the player going off is actually on the field.
    - A check is made to make sure the player coming on is not on the field of play already.
    - A check is made to check how many subs a team has used if they have used 5 substitutions or 3 black card substitutions a message is shown to tell the referee the substitution can't happen as the team is at its maximum.
- Create an injury (team, player and a note about the injury).
- Add additional information for the match report.
- Add the times both teams took to the field for the match report.
- Start and end a game storing the times for the match report.
    - Game can't be started until both team sheets have been submitted.
    - Cards, Scores or Substitutes can't be created until the game is started.
- View both teams team sheets.
- View a list of all scores, cards, substitutes or injuries created during the game.
- All data is stored locally and in Firestore.

6.1.2 Live Scores Web Application

The following is a list of the objectives that were met for the Live Scores web application.

- List all of the current day's games.
- Update the total score for a game live.
- Update the latest score live (time, team, score type and player name).
- Update all of the games scores live.
- View every team's team sheet.

6.1.3 Game Management Web Application

The following is a list of the objectives that were met for the Game Management web application.

- List all upcoming games and past games associated to the user.
- Council, Provincial and County Secretaries can create a game.
- Team Officials can submit a team sheet.
- View team sheet.
- Secretary associated with a game and the Referee of the game can view a match report.
- Information retrieved and stored in Firebase Firestore

6.2 Were the Objectives I set out for the project met

Not all objectives were met that I set out for the project, this was due to time constraints, with more time I could have met all objectives.

6.2.1 Referee Android Wear Application

The following is a list of objectives that were not met for the Referee Android Wear Application.

- Update and delete scores, cards, substitutions, injuries and additional notes.
- End first half.

6.2.2 Live Score Web Application

I met all objectives set out for the Live Score Web Application.

6.2.3 Game Management Web Application

The following is a list of objectives that were not met for the Game Management Web Application.

- Secretaries to be able to update and delete games.
- Secretary associated with a game to be able to submit a team sheet.
- Secretary and team official associated with a game to be able to update and delete and team sheet.
- Referees to be able to create, update and delete a match report.

6.3 Problems that occurred and how I solved them

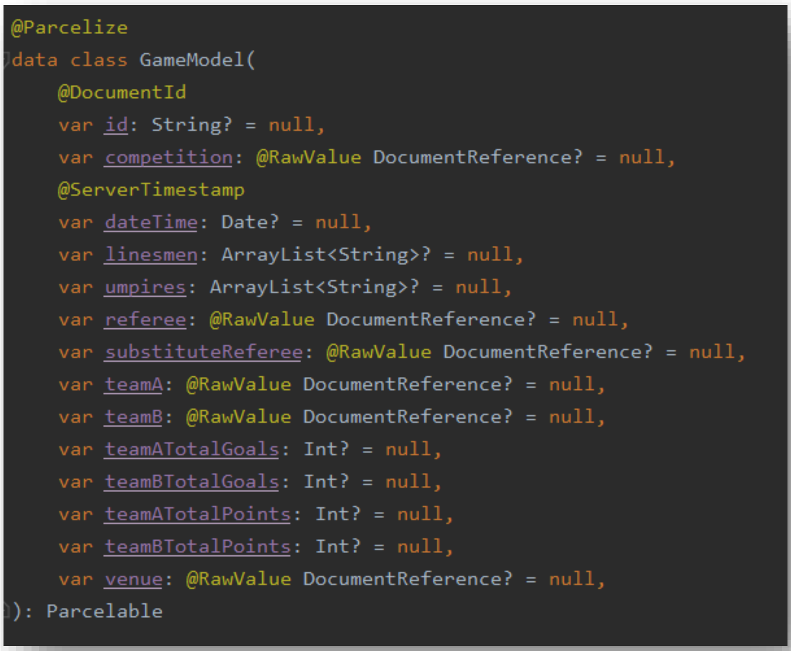
6.3.1 Using Node for the web application.

After setting up the initial NodeJS web application part of the project, I began to look into how to connect a NodeJS web application to Firebase authentication and Firestore. Through my research I learned that by using Firebase I didn't really need to have my own back-end server, I could create a serverless web application with Firebase using their authentication, databases, security rules, functions and hosting. I had spent three weeks working on building the NodeJS web application so it was hard to scrap the work, but I felt it was the right thing to do as I had all the functionality that I needed within Firebase without adding the extra overhead of a backend server, which I would have had to manage on an ongoing basis.

In order to solve this problem, I decided to use Svelte for the serverless frontend and then connect the frontend to Firebase for the backend of the web application.

6.3.2 Converting a Firestore Document into a local data class model

After retrieving a document from Firestore I was having difficulty converting it to a data class object. The first thing I learnt was that the fields in the data class model must be of type var and must be initialized with a value or null. I also needed to use Firestore annotations (Annotations, n.d.). For the Documents Id I needed the annotation @DocumentId, for timestamps I needed the annotation @ServerTimestamp, and for document references the type required was DocumentReference and it needed to be parcelled with the tag @RawValue. In figure 41 you will see how the GameModel needed to be setup in order for the Firestore Document to be converted to a Kotlin object.

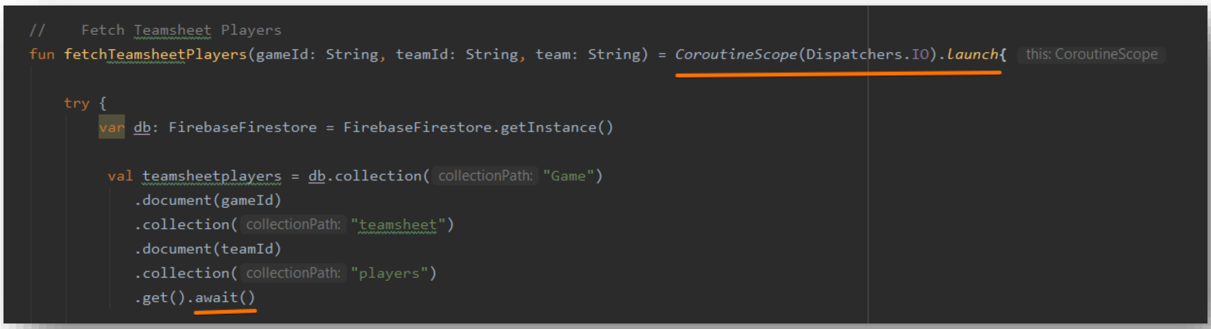


```kotlin
@Parcelize
data class GameModel(
    @DocumentId
    var id: String? = null,
    var competition: @RawValue DocumentReference? = null,
    @ServerTimestamp
    var dateTime: Date? = null,
    var linesmen: ArrayList<String>? = null,
    var umpires: ArrayList<String>? = null,
    var referee: @RawValue DocumentReference? = null,
    var substituteReferee: @RawValue DocumentReference? = null,
    var teamA: @RawValue DocumentReference? = null,
    var teamB: @RawValue DocumentReference? = null,
    var teamATotalGoals: Int? = null,
    var teamBTotalGoals: Int? = null,
    var teamATotalPoints: Int? = null,
    var teamBTotalPoints: Int? = null,
    var venue: @RawValue DocumentReference? = null,
): Parcelable
```

Figure 41 – Kotlin Model code for converting Firestore Document to a Kotlin Object

6.3.3 Fetching data from Firestore

While fetching data from Firestore, data was being run asynchronously and was being returned out of sync. I required data about the team sheet from Firestore, in order to use in the next query to Firestore for the team's player documents, therefore I need these calls to be sequential. This was causing errors and null pointer exceptions. In order to solve this problem, I needed to find a way to write asynchronous code sequentially. I looked at RX and Threads but, in the end, I decided to use coroutines as it eliminated the use of call-backs and a lot of extra code. Coroutines are easier to write, are sequential, non-blocking and don't cause unnecessary memory overhead. They don't block the main thread; therefore, the user will not see any difference, they won't be waiting for the UI to load or have their application crash unexpectantly.



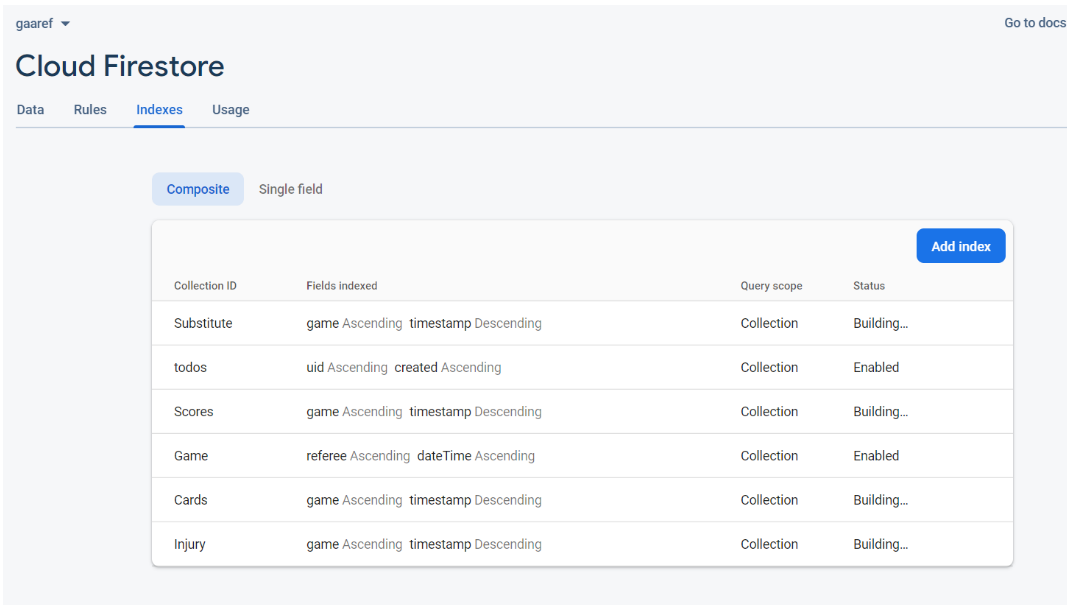
```
//    Fetch Teamsheet Players
fun fetchTeamsheetPlayers(gameId: String, teamId: String, team: String) = CoroutineScope(Dispatchers.IO).launch{  this: CoroutineScope

    try {
        var db: FirebaseFirestore = FirebaseFirestore.getInstance()

        val teamsheetplayers = db.collection( collectionPath: "Game")
            .document(gameId)
            .collection( collectionPath: "teamsheet")
            .document(teamId)
            .collection( collectionPath: "players")
            .get().await()
```

Figure 42 – Kotlin Coroutines

6.3.4 Firebase Cloud Firestore Index

On occasion when I was fetching Documents from Firestore, errors were occurring in relation to needing indexes for Firestore queries. Indexes in Firestore are used for performance. When I send a query from my application, if Indexes for that query are setup in Firestore, Firestore can use the index to quickly lookup the locations of the items that I have requested from my query.

Once I setup each query and had them retrieving the correct documents, I was able to setup a composite index for the query in the Firestore console, which stores a sorted mapping of all the documents in a collection based on the order I declared in my queries



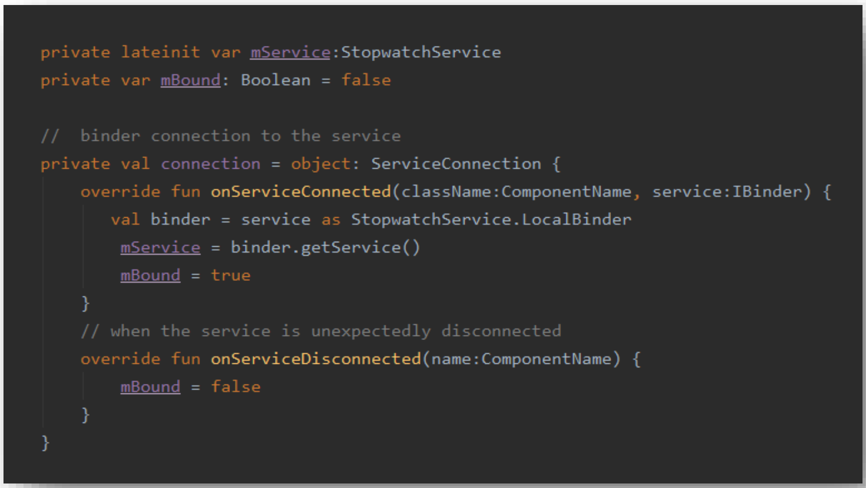
Collection ID	Fields indexed	Query scope	Status
Substitute	game Ascending timestamp Descending	Collection	Building...
todos	uid Ascending created Ascending	Collection	Enabled
Scores	game Ascending timestamp Descending	Collection	Building...
Game	referee Ascending dateTime Ascending	Collection	Enabled
Cards	game Ascending timestamp Descending	Collection	Building...
Injury	game Ascending timestamp Descending	Collection	Building...

Figure 43 – Firebase Cloud Firestore – Indexes (Monosnap, n.d.)

6.3.5 Running a Stopwatch for a long period of time without killing the process when changing between fragments.

When I first created the stopwatch, I did this in the Stopwatch Fragment, when I was in this fragment the stopwatches time incremented as it should but when I navigated to a different fragment the stopwatch functionality was killed and the time was reverted back to zero on return to the Stopwatch Fragment.

I needed a way for the stopwatch to run for over 70 minutes without blocking the main thread and without being cancelled or killed when changing between fragments. I decided to use android Services to complete this task (Android Services, n.d.). Services allow for long running operations. These operations can be run in the background. It had the added functionality of allowing me to bind from the main activity to the Service allowing me to send a signal to the Service to tell it to run or pause the stopwatch.

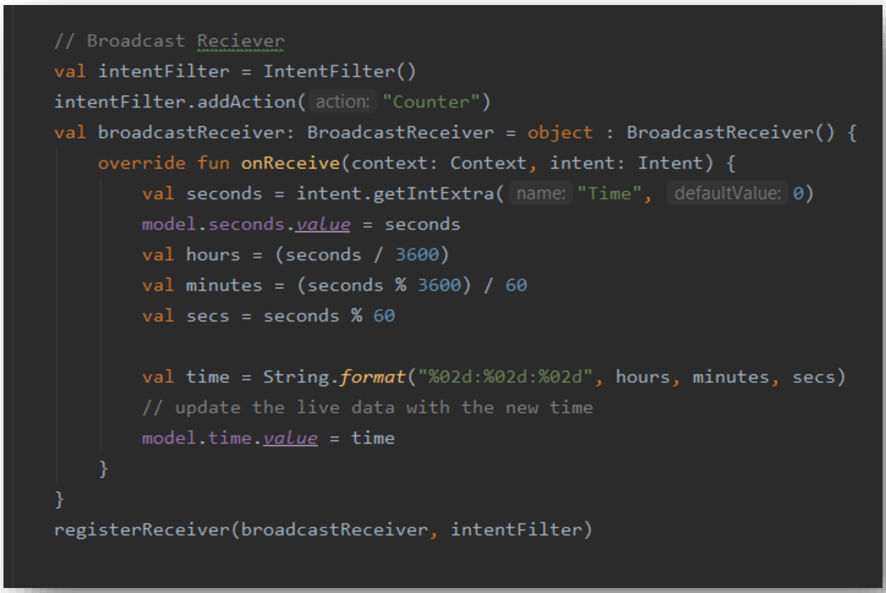


```kotlin
private lateinit var mService:StopwatchService
private var mBound: Boolean = false

//  binder connection to the service
private val connection = object: ServiceConnection {
    override fun onServiceConnected(className:ComponentName, service:IBinder) {
        val binder = service as StopwatchService.LocalBinder
        mService = binder.getService()
        mBound = true
    }
    // when the service is unexpectedly disconnected
    override fun onServiceDisconnected(name:ComponentName) {
        mBound = false
    }
}
```

Figure 44 – Kotin – Creating a connection to the stopwatch service (Monosnap, n.d.)

The one problem with Services is it doesn’t allow you to update the UI it is completely separate. I needed to update the UI with the current stopwatch time. To solve this problem, I used Broadcasts (Android Broadcast, n.d.).

Broadcasts allowed me to be able to send a message from the service to the main activity with the current seconds. In the main activity I could then use this value to work out the hours, minutes and seconds and format them into string format.

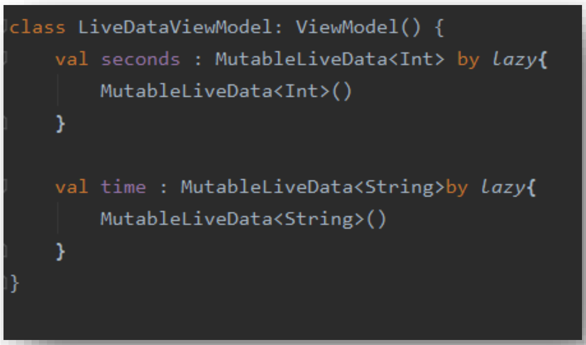


```kotlin
// Broadcast Reciever
val intentFilter = IntentFilter()
intentFilter.addAction( action: "Counter")
val broadcastReceiver: BroadcastReceiver = object : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        val seconds = intent.getIntExtra( name: "Time",  defaultValue: 0)
        model.seconds.value = seconds
        val hours = (seconds / 3600)
        val minutes = (seconds % 3600) / 60
        val secs = seconds % 60

        val time = String.format("%02d:%02d:%02d", hours, minutes, secs)
        // update the live data with the new time
        model.time.value = time
    }
}
registerReceiver(broadcastReceiver, intentFilter)
```

Figure 45 – Kotin – Broadcast Receiver for the stopwatch (Monosnap, n.d.)

As the Service and the BroadcastReciver were communicating the current time. I needed a way to communicate this to the UI of the stopwatch fragment. To do this I decided to use android LiveData.

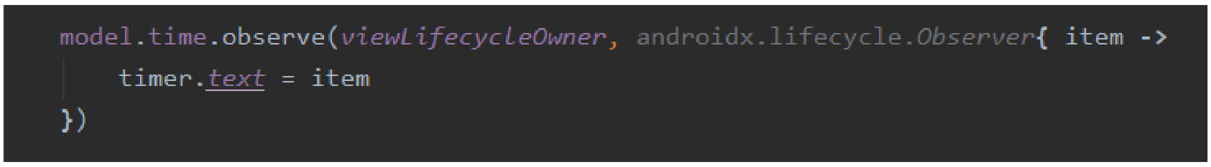
I created a ViewModel class to hold the MutableLiveData objects time. When the BroadcastReceiver receives a message with the current seconds. Using the seconds to calculate the time String I was then able to update the Live Data object for time.



```
class LiveDataViewModel: ViewModel() {
    val seconds : MutableLiveData<Int> by lazy{
        MutableLiveData<Int>()
    }

    val time : MutableLiveData<String>by lazy{
        MutableLiveData<String>()
    }
}
```

Figure 46 – Kotlin - LiveData for the stopwatch (Monosnap, n.d.)

In the Stopwatch Fragment I was able to create an observer of the LiveData Object time, therefore every time the LiveData Object time changes the Stopwatch Fragments UI is updated with the new value for time.



```
model.time.observe(viewLifecycleOwner, androidx.lifecycle.Observer{ item ->
    timer.text = item
})
```

Figure 47 - Kotlin – LiveData observer for changes in stopwatch time (Monosnap, n.d.)

6.3.6 Firebase Functions running slow

I set up Firebase cloud function to run every day deleting the previous days games and retrieving the upcoming days games, as well as a function for listening for score updates. On initial setup the functions were running really slowly they were taking approximately 5-6 minutes for a function to run. The code was running asynchronously therefore the return success message was being returned before the data was returned from the databases.

In Firebase functions when a response is received from the function the function is then considered stale and this effects performance considerably (Firebase Firestore functions working slow or returning too quick, n.d.)

Adding an async await function before returning a response from the cloud function meant the function ran in a few seconds rather than minutes.

6.3.7 Svelte Stores Persistence

I used svelte stores to store the user’s information so that the values could be accessed by multiple unrelated components. I set this up with the help of the svelte tutorials ((Writable Stores, n.d.)), however whenever I refreshed the browser the values were not persistent and were reset back to their initial default values. In order to persist the data, I needed to store the values to local storage. I installed a package called svelte-local-storage-store (Svelte Local Storage Store, n.d.) which did all the hard work in the background to save the store values locally. Now whenever you refresh the page the store values are persistent.

7. Reflection on Learning

7.1 Android Wear Development

While I had completed the module Mobile Development, for this project I was required to do extra research and learning in order to create the android wear project. I began my learning with the Android wear online documentation and followed on with some of their code labs and training (Android Wear, n.d.).

7.1.1 Authentication

The first hurdle was setting up authentication, I decided to use Google Sign-in as it offers the best user experience and it is easy to support (Android Wear Authentication, n.d.)

I needed to integrate the Android Wear Emulator with my Google Account. This was a time-consuming task but once I got it setup, I didn't have any issues with it again (Start Integrating, n.d.).

7.1.2 Layouts and navigation

Other issues I faced and had to learn more about was the wear layouts and the biggest learning curve was the navigation. I found the Android wear-os samples an excellent resource which helped me see the code and how it all worked (Wear Drawers, n.d.).

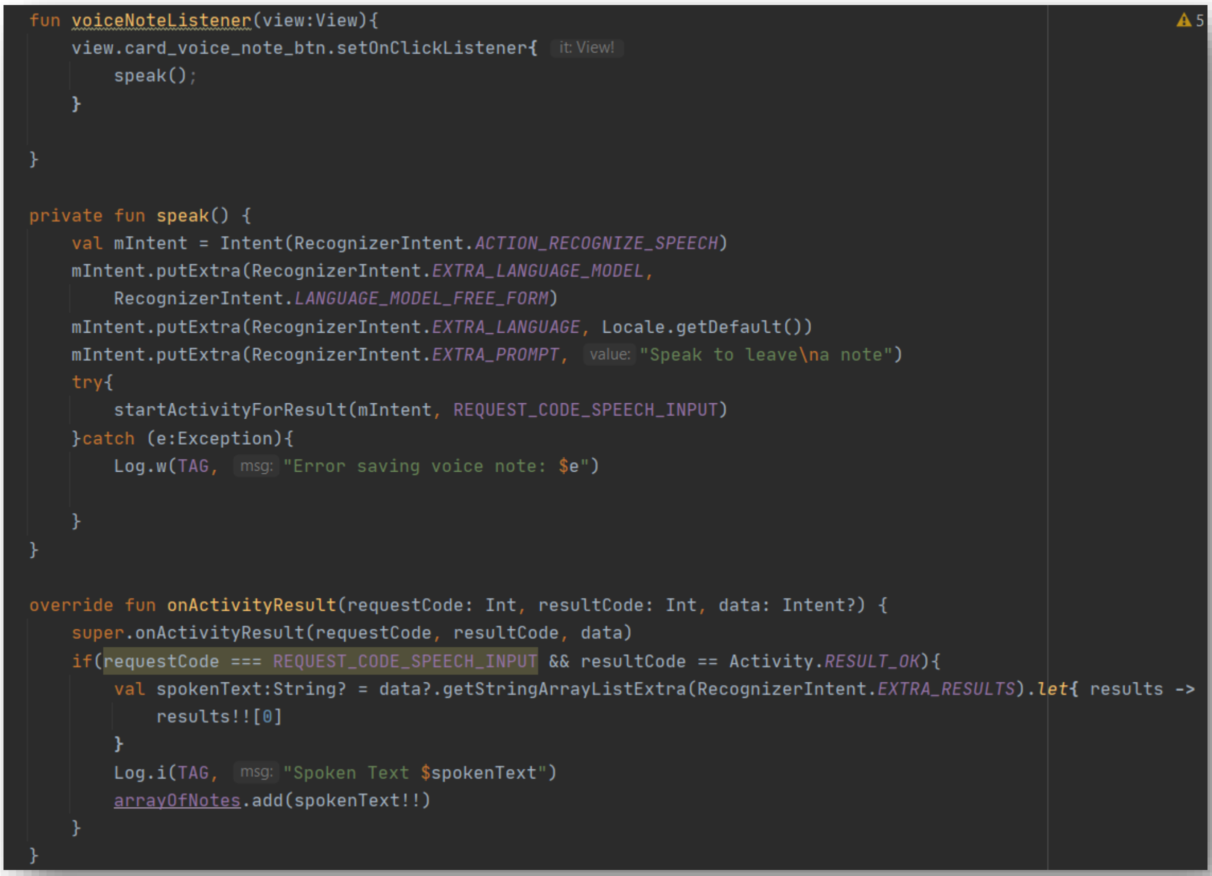
7.1.3 Retrieving voice notes and converting them into text

As the watch is such a small device and would be difficult to enter long amounts of text, I felt it was necessary to add voice input for any notes. Again, this was new to me but it was well documented on the Android website and it was an easy functionality to add (User Input/Voice, n.d.).

Every wearable OS device has a microphone, I used the emulators microphone to record voice. I was then able to use the systems build in Speech Recognizer to retrieve the voice input from the referee.

When the microphone button is pressed, an activity which returns a result is started and uses the ACTION_RECOGNIZE_SPEECH action. The ACTION_RECOGNIZE_SPEECH prompts the referee to speak. the voice is sent to the speech recognizer and the result is returned. I added the required intent EXTRA_LANGUAGE_MODEL which signifies which speech model to use. The model I add was LANGUAGE_MODEL_FREE_FORM which is used for dictation rather than short commands for web searches. I set the language to the local language using EXTRA_LANGUAGE_MODEL.

The referee can add as many notes as they want, I add them to an array of notes for cards, injuries, substitutes and additional notes.



```
fun voiceNoteListener(view:View){
    view.card_voice_note_btn.setOnClickListener{ it: View!
        speak();
    }

}

private fun speak() {
    val mIntent = Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH)
    mIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM)
    mIntent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, Locale.getDefault())
    mIntent.putExtra(RecognizerIntent.EXTRA_PROMPT, value: "Speak to leave\na note")
    try{
        startActivityForResult(mIntent, REQUEST_CODE_SPEECH_INPUT)
    }catch (e:Exception){
        Log.w(TAG, msg: "Error saving voice note: $e")

    }
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if(requestCode === REQUEST_CODE_SPEECH_INPUT && resultCode == Activity.RESULT_OK){
        val spokenText:String? = data?.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS).let{ results ->
            results!![0]
        }
        Log.i(TAG, msg: "Spoken Text $spokenText")
        arrayOfNotes.add(spokenText!!)
    }
}
```

Figure 48 -Android Wear Voice Input Program (Monosnap, n.d.)

7.1.4 Live Data

During our Mobile Development module, we were encouraged to do extra learning on Live Data, I didn’t get around to it for that project, but I needed it for this project. To learn more about it I followed the Live Data code lab (kotlin android training live data, n.d.)

LiveData is an observable data holder which was required for the stopwatch time. I needed the time to continue in the background and not be killed when the stopwatch fragment was destroyed. I also used it to keep track of total points and goals for each team.

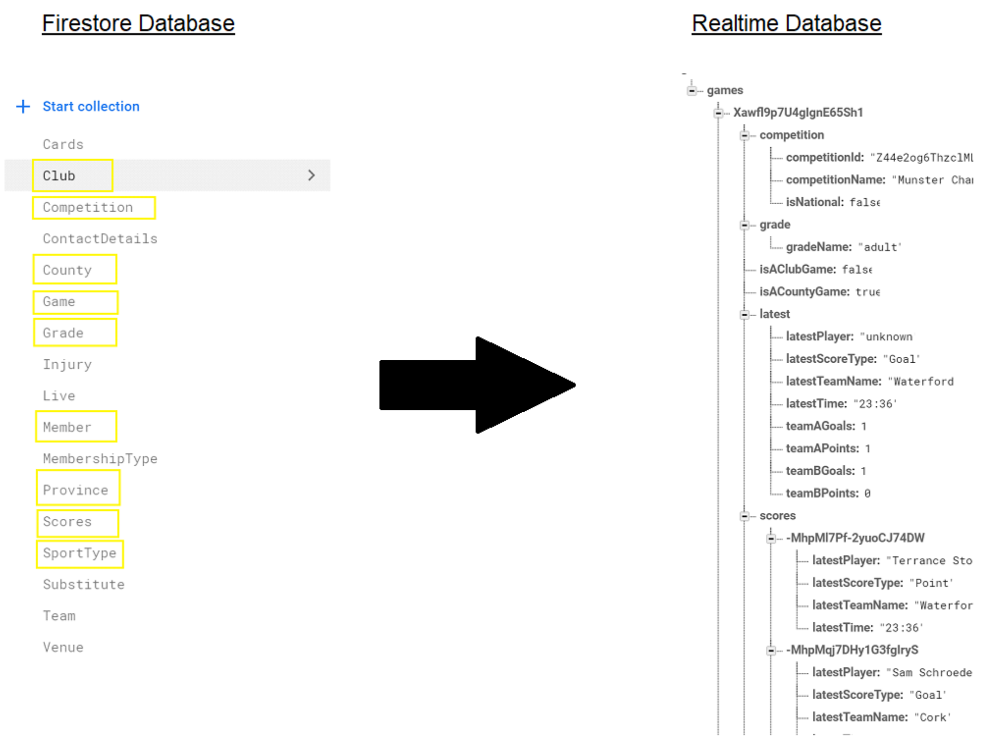
7.2 Firebase Cloud Functions

For the Live Scores web application, I decided to use Firebase Realtime Database. I choose to do this as the information for the Live Scores web application is data that could be accessed thousands of times every day. I needed a database that I could access the information quickly and without too many queries.

I was storing all the information from the GAA game management web application and from the Referees android wear application in Firestore. If I had used Firestore for the live scores it would have required hundreds of queries every time someone accessed the Live Scores web application to show the games, this would slow down the Live Scores web application considerably.

I needed a way to run these queries only once per day to get the game information from Firestore and create the games in the Realtime database. The Answer to this was Firebase Cloud Functions.

Cloud Functions in Firebase is serverless, and allowed me to create functions that run in the background. I setup 2 scheduled functions to run just after midnight every day. The first function deletes all the previous days games from the database and the second function gathers all the information needed for the game from the various documents and collections in Firestore and creates this game in the Realtime database.

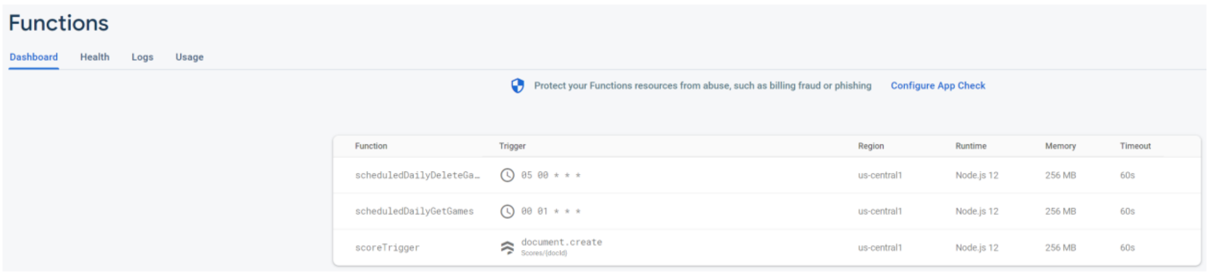


```
Cards
Club
Competition
ContactDetails
County
Game
Grade
Injury
Live
Member
MembershipType
Province
Scores
SportType
Substitute
Team
Venue
```

```
games
  Xawfl9p7U4glgnE65Sh1
    competition
      competitionId: "Z44e2og6ThzclMl
      competitionName: "Munster Chai
      isNational: false
    grade
      gradeName: "adult'
    isAClubGame: false
    isACountyGame: true
    latest
      latestPlayer: "unknown
      latestScoreType: "Goal'
      latestTeamName: "Waterford
      latestTime: "23:36'
      teamAGoals: 1
      teamAPoints: 1
      teamBGoals: 1
      teamBPoints: 0
    scores
      -MhpMl7Pf-2yuoCJ74DW
        latestPlayer: "Terrance Sto
        latestScoreType: "Point'
        latestTeamName: "Waterfor
        latestTime: "23:36'
      -MhpMqj7DHy1G3fglryS
        latestPlayer: "Sam Schroede
        latestScoreType: "Goal'
        latestTeamName: "Cork'
```

Figure 49 - Transferring data from many collections and documents in Firestore to one quick access Realtime Database Reference (Paint) (Monosnap, n.d.)

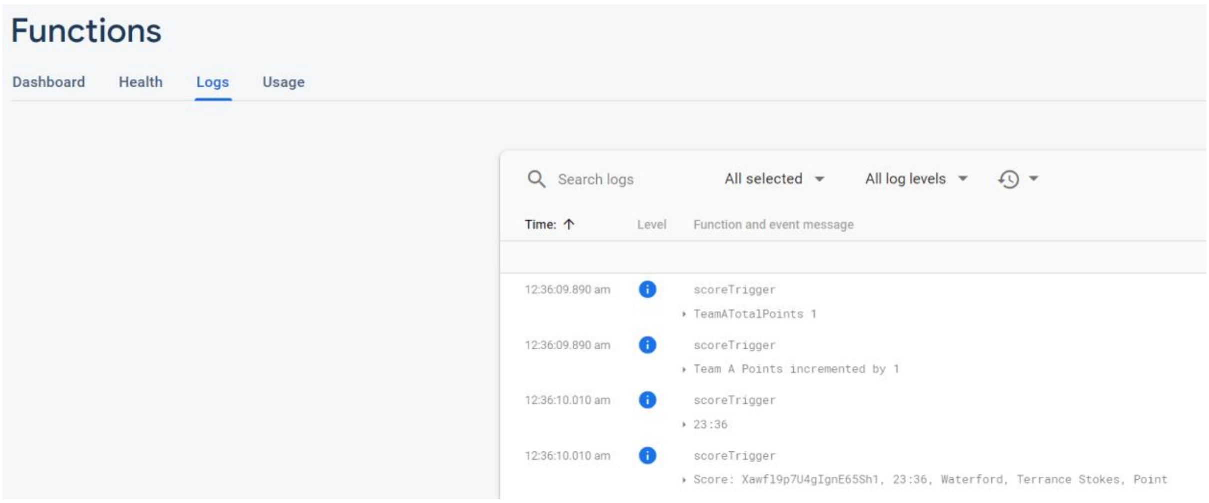
I also took advantage of Firebase Functions to listen for new scores in the Firestore Database. When a new score is saved the Firebase Function queries all the information needed about that score and creates an entry into the Realtime Database for that game which is then shown on the Live Score web application.



Protect your Functions resources from abuse, such as billing fraud or phishing

Function	Trigger	Region	Runtime	Memory	Timeout
scheduledDailyDeleteGa…	05 00 * * *	us-central1	Node.js 12	256 MB	60s
scheduledDailyGetGames	00 01 * * *	us-central1	Node.js 12	256 MB	60s
scoreTrigger	document.create Scores/{docId}	us-central1	Node.js 12	256 MB	60s

Figure 50 - Firebase Functions (Monosnap, n.d.)



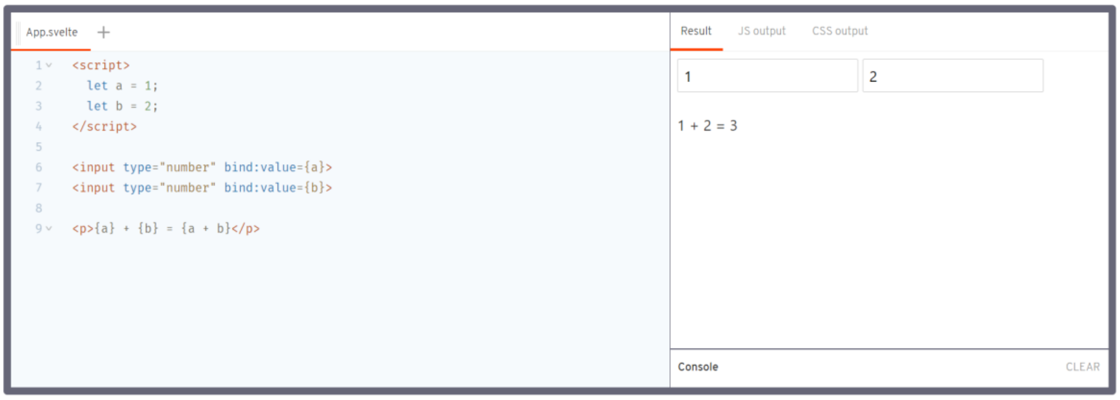
```
Time: ↑          Level   Function and event message

12:36:09.890 am    ●     scoreTrigger
                         ▸ TeamATotalPoints 1
12:36:09.890 am    ●     scoreTrigger
                         ▸ Team A Points incremented by 1
12:36:10.010 am    ●     scoreTrigger
                         ▸ 23:36
12:36:10.010 am    ●     scoreTrigger
                         ▸ Score: Xawf19p7U4gIgnE65Sh1, 23:36, Waterford, Terrance Stokes, Point
```

Figure 51 - Firebase Logs (Monosnap, n.d.)

7.3 Svelte

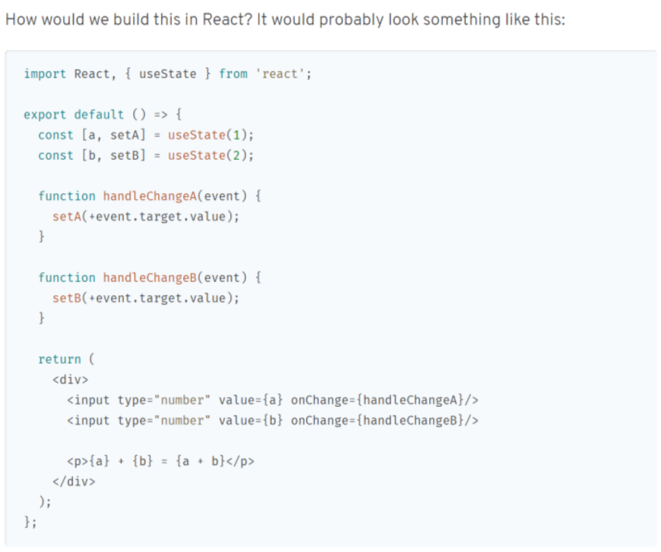
As I previously mentioned, I decided to change from using a node server web application to a serverless web application. To do this I needed to decided what frontend framework I would use. I had already learnt React during our ICT module; however, my lecturer Eamonn had mentioned a new framework called Svelte that would reduce the amount of code needed compared to React. I had to decide whether to complete the frontend of the project using with the knowledge I already had with React or to take on the challenge to learn this new framework. To make my decision I decided to complete some of the tutorials on the Svelte website (Svelte Tutorials, n.d.).

I learnt that svelte was similar to React however the main difference is that Svelte converts the app into JavaScript at build time rather than at run time, which meant Svelte would have a faster loading time. While Svelte had similarities with React with components, props, Lifecycles and events. I found Svelte much easier to learn and use. The amount of code needed for Svelte to React was much less.



```
<script>
  let a = 1;
  let b = 2;
</script>

<input type="number" bind:value={a}>
<input type="number" bind:value={b}>

<p>{a} + {b} = {a + b}</p>
```

```
1    2

1 + 2 = 3
```



How would we build this in React? It would probably look something like this:

```
import React, { useState } from 'react';

export default () => {
  const [a, setA] = useState(1);
  const [b, setB] = useState(2);

  function handleChangeA(event) {
    setA(+event.target.value);
  }

  function handleChangeB(event) {
    setB(+event.target.value);
  }

  return (
    <div>
      <input type="number" value={a} onChange={handleChangeA}/>
      <input type="number" value={b} onChange={handleChangeB}/>

      <p>{a} + {b} = {a + b}</p>
    </div>
  );
};
```

Figure 52 - Sample Difference between code needed for Svelte versus React

(Write Less Code, n.d.) (Monosnap, n.d.)

Overall, I am delighted I took the plunge to learn and use Svelte over React for the headless frontend of the Live Scores and Game Management web applications project. I really enjoyed learning it and using it and it is definitely a Framework, I will continue to learn and practise and hopefully use again in future projects.

7.4 Bootstrap Framework

As a GAA fan or official a lot of your time is spent on the side line of a field so it was important to me to have fully responsive web applications, so that live scores or team sheets could be checked or updated from the mobile phone on the side line or from the laptop at home.

A few years back I learned and used the Bootstrap front-end framework for some projects. I decided I would use the Bootstrap Framework for this web application as I found it had excellent responsiveness, the web application could look great across multiple devices easily using this framework. I needed to go back and refresh my memory of Bootstrap and also learn a lot of new things due the changes since I last used it, but it came back to me easy enough and I am delight with the fact users can use the web applications on mobile, tablet, laptops or desktops.

7.5 Database Design

The courses module in Database and Design was a great benefit to me for designing the database, as there were so many components and hierarchy involved it took a lot of work to design the database and the courses information was vital in me completing this. This project was a great way for me to put into practice what I had learnt, it made me realise how important the initial design is and I thoroughly enjoyed the process.

One of the things we learned was that sometimes it is of more benefit to store all the data for an entity rather than storing references to other entities and this was one mistake I think I made when designing the Database. For the Game Management web application, I needed to use a lot of queries to retrieve the data needed to build upcoming or past games. I only had a few games stored in the Firestore Database but it took approximately 17 seconds to retrieve the data from the various collections and re-build these games. This in my opinion was very slow.

If I was back again, I would not store the game with References to collections such as Members, Competition, Scores etc. I would store all the information needed for a game in the Game document without any references. This would make the retrieval of a games data quicker and easier and it would require a lot less code.

8. Conclusion

The aim of this final project for me was to create software to solve a problem. From my experience as a GAA fan, LGFA Secretary, Registrar and Trainer, I could see a big problem and gap in the digital world of the GAA especially at club level. Currently GAA Referees entire process is done with pen and paper, written team sheets are being handed over by team officials, and trying to find live scores for a game is difficult. Therefore, I decided this would be a good and unique project to complete.

While I didn't meet all objectives set out, with more time, I would definitely have been able to meet those objectives. In the future I would like to expand on this project as follows.

- For all applications I would add Google Maps GPS functionality to allow users view where a venue is and get directions to that venue.
- For the live score's web application, I would like to authenticate users allowing them to have a profile where they can set their favourite teams, counties, clubs or sport that they want to follow and filter the results they see to match their favourites. I would add a listener to the team sheets so that the team sheets would update live when a change is made or a substitution. I would include further information for each game showing information for the substitutions and cards given by the referee.
- For the Game management I would have functionality for the secretary or referee of a game to be able to download a PDF file for the match report or be able to email the match report.
- I would add a public/private view setting to team sheets so that the team official could set the team sheet to private until the game has started. This would be important to the team official as in general they don't want their team to be known prior to a game.
- For the android wear application, I would like to improve the appearance of layouts, add notifications to notify the referee if they have been scheduled for a game and add alarms to a game to notify the referee when time is up.
- Future additions to the project could include a full GAA Management web application that could be used for everything GAA related all in the once place for example registration, fundraising, statistics, recording of courses, garda vetting, covid tracking etc.

Along the way I used many skills that I learned during the Hdip in Computer Science including web development, android development, ICT, Database Design, Programming with JavaScript, Kotlin, HTML and CSS, Git and GitHub as well as expanding on that learning with new topics such as Android Wear, Firebase Cloud Functions and Firestore and Hosting, Svelte Framework and re-learning the frontend Bootstrap Framework.

Overall, I am delighted with what I have achieved, back in January when I was looking at android wear, I didn't know how or where I was going to start with the project but with a lot of research, I got there slowly but surely. I also made many mistakes along the way and wasted a lot of time on some of those mistakes which made me realise that taking the time to prepare and make decisions at the outset of a project is vital, my eyes were also opened to how important it is to spend time designing your database as it can have huge effects on the performance of a project and the amount of code that may be needed. I am especially delighted with all the new things that I have learnt along.

References

Android Broadcast. (n.d.). Retrieved from Android Developer: https://developer.android.com/guide/components/broadcasts

Android Services. (n.d.). Retrieved from Android Developer: https://developer.android.com/guide/components/services

Android Studio. (n.d.). Retrieved from Android Developers: https://developer.android.com/studio

Android Wear. (n.d.). Retrieved from Android Developer: https://developer.android.com/wear

Android Wear Authentication. (n.d.). Retrieved from Android Developer: https://developer.android.com/training/wearables/overlays/auth-wear#Google-Sign-in

Annotations. (n.d.). Retrieved from Android Developer: https://developer.android.com/reference/java/lang/annotation/Annotation.html

Balsamiq Cloud. (n.d.). Retrieved from Balsamiq Cloud: https://balsamiq.cloud/

Bootstrap. (n.d.). Retrieved from Bootstrap: https://getbootstrap.com/

CSS: Cascading Style Sheets. (n.d.). Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/CSS

Draw. (n.d.). Retrieved from Draw: https://draw.io

Firebase Authentication. (n.d.). Retrieved from Firebase: https://firebase.google.com/docs/auth/

Firebase Cloud Functions. (n.d.). Retrieved from Firebase: https://firebase.google.com/docs/functions

Firebase Firestore. (n.d.). Retrieved from Firebase: https://firebase.google.com/docs/firestore

Firebase Firestore functions working slow or returning too quick. (n.d.). Retrieved from Stackoverflow: https://stackoverflow.com/questions/67472446/firestore-firebase-functions-working-slow-or-returning-too-quick

Firebase Hosting. (n.d.). Retrieved from Firebase: https://firebase.google.com/docs/hosting

Firebase Pricing. (n.d.). Retrieved from Firebase: https://firebase.google.com/pricing/

Firebase Realtime Database. (n.d.). Retrieved from Firebase: https://firebase.google.com/docs/database

Firestore Pricing. (n.d.). Retrieved from Firebase: https://cloud.google.com/firestore/pricing

HTML: HyperText Markup Language. (n.d.). Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTML

JavaScript. (n.d.). Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript

Kotlin. (n.d.). Retrieved from Kotlinlang: https://kotlinlang.org/

kotlin android training live data. (n.d.). Retrieved from Android Developer: https://developer.android.com/codelabs/kotlin-android-training-live-data#0

MongoDB. (n.d.). Retrieved from MongoDB: https://www.mongodb.com/cloud/atlas

MongoDB Pricing. (n.d.). Retrieved from MongoDB: https://www.mongodb.com/pricing

Monosnap. (n.d.). Retrieved from Monosnap: https://monosnap.com/

Paint. (n.d.).

Referee Handbook. (n.d.). Retrieved from GAA: https://www.gaa.ie/api/pdfs/image/upload/ogd9kegh1slxnvvuwmll.pdf

Start Integrating. (n.d.). Retrieved from Android Developers: (https://developers.google.com/identity/sign-in/android/start-integrating

Svelte. (n.d.). Retrieved from Svelte: https://svelte.dev/

Svelte Local Storage Store. (n.d.). Retrieved from npmjs: https://www.npmjs.com/package/svelte-local-storage-store

Svelte Tutorials. (n.d.). Retrieved from Svelte: https://svelte.dev/tutorial/basics

Trello. (n.d.). Retrieved from Trello: https://trello.com/

User Input/Voice. (n.d.). Retrieved from Android Developers: https://developer.android.com/training/wearables/user-input/voice

Visual Studio Code. (n.d.). Retrieved from Visual Studio: https://code.visualstudio.com/

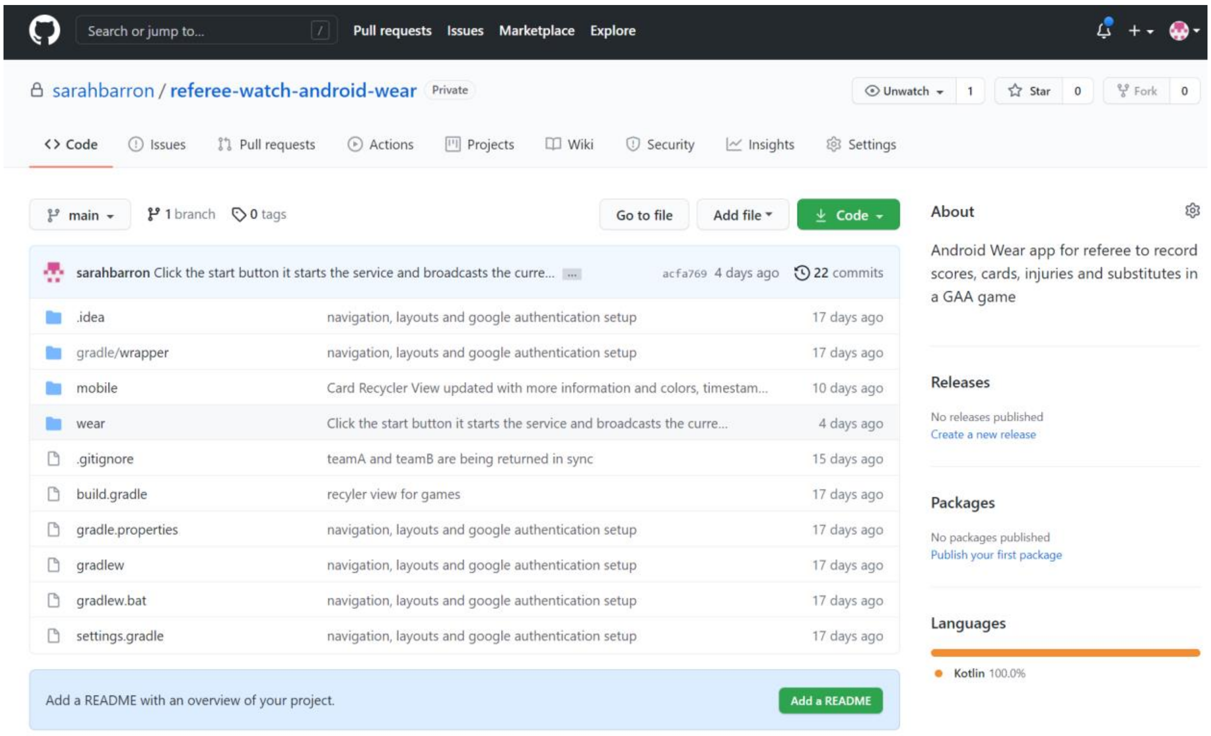
Wear Drawers. (n.d.). Retrieved from Android Developer: https://github.com/android/wear-os-samples/tree/main/WearDrawers

Writable Stores. (n.d.). Retrieved from Svelte: https://svelte.dev/tutorial/writable-stores

Write Less Code. (n.d.). Retrieved from Svelte: https://svelte.dev/blog/write-less-code

Appendices

Appendix A – GitHub Repository for the referee android wear watch



sarahbarron Click the start button it starts the service and broadcasts the curre... acfa769 4 days ago 22 commits

.idea	navigation, layouts and google authentication setup	17 days ago
gradle/wrapper	navigation, layouts and google authentication setup	17 days ago
mobile	Card Recycler View updated with more information and colors, timestam...	10 days ago
wear	Click the start button it starts the service and broadcasts the curre...	4 days ago
.gitignore	teamA and teamB are being returned in sync	15 days ago
build.gradle	recyler view for games	17 days ago
gradle.properties	navigation, layouts and google authentication setup	17 days ago
gradlew	navigation, layouts and google authentication setup	17 days ago
gradlew.bat	navigation, layouts and google authentication setup	17 days ago
settings.gradle	navigation, layouts and google authentication setup	17 days ago

Add a README with an overview of your project. Add a README

About

Android Wear app for referee to record scores, cards, injuries and substitutes in a GAA game

Releases

No releases published
Create a new release

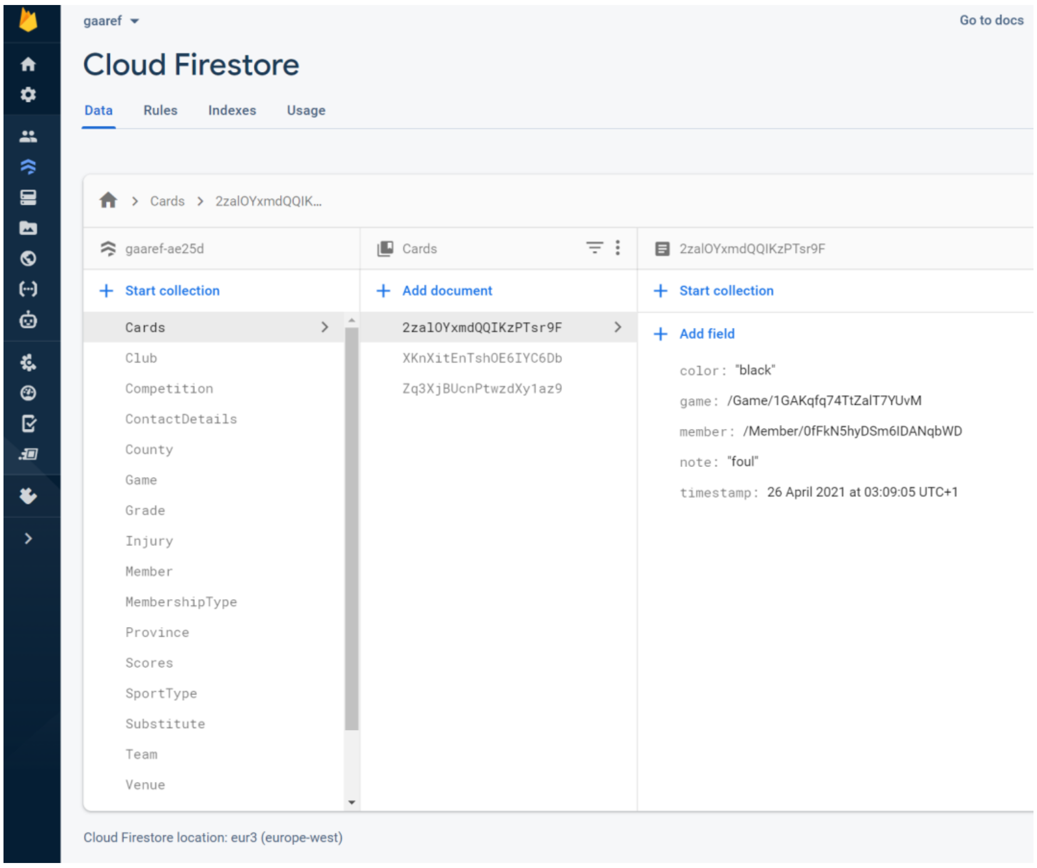
Packages

No packages published
Publish your first package

Languages

● Kotlin 100.0%

Appendix B – Firebase Firestore database setup



Collections:

```
Cards
Club
Competition
ContactDetails
County
Game
Grade
Injury
Member
MembershipType
Province
Scores
SportType
Substitute
Team
Venue
```

Documents in Cards:

```
2zalOYxmdQQIKzPTsr9F
XKnXitEnTshOE6IYC6Db
Zq3XjBUcnPtwzdXy1az9
```

Fields of 2zalOYxmdQQIKzPTsr9F:

```
color: "black"
game: /Game/1GAKqfq74TtZalT7YUvM
member: /Member/0fFkN5hyDSm6IDANqbWD
note: "foul"
timestamp: 26 April 2021 at 03:09:05 UTC+1
```

Cloud Firestore location: eur3 (europe-west)