



**DUBLIN INSTITUTE
of TECHNOLOGY**

Institiúid Teicneolaíochta Bhaile Átha Cliath

Basketball Statistics Application Final Year Project Report

**DT228
BSc in Computer Science**

**Alannah Mullins
Paul Bourke**

School of Computing
Dublin Institute of Technology

13th April 2018

Abstract

Basketball is one of the most popular sports across the world. Its popularity is growing at all levels year on year and it is a fantastic community to be a part of. But with players at the highest levels receiving top end data analysis to improve their skills the same can not be said for lower level players.

While the sophisticated video analysis systems are out of reach, there are alternatives. The purpose of this project is to create a statistical recording app that can be used at any level to benefit from data analysis.

This project is an android based project that works by implementing custom gestures to record statistics. The data is displayed graphically and the overall result is an application that can help improve a teams performance.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

A handwritten signature in brown ink that reads "Alannah Mullins". The signature is written in a cursive style with a large initial 'A'.

Alannah Mullins

13th April 2018

Acknowledgements

Firstly, I would like to sincerely thank my project supervisor Paul Bourke for his help throughout the year and for keeping me and my ideas grounded.

All my testers who took the time to not only give me a hand when I needed it but that also took an interest in my project and became endlessly excited with the possibilities.

A very big thank you to my family and friends who supported me through the college year and who turn up to every match. My sister Chloe for the wonderful art work that made my app look a thousand times better than I ever thought it would. Lastly Aaron for never letting me give up.

1 Table of Contents

2	Introduction.....	7
2.1	Overview and Background.....	7
2.2	Project Objectives	7
2.2.1	Primary Objectives.....	7
2.2.2	Secondary Objectives.....	8
2.3	Project Challenges	8
2.4	Structure of the document	9
3	Research.....	10
3.1	Problem and Solution Identification Research.....	10
3.1.1	Growth of Statistic Analysis in Sport, specifically Basketball	10
3.1.2	Statistical Recording in Ireland.....	11
3.1.3	User Requirements.....	12
3.1.4	Existing Solutions	13
3.2	Technologies Evaluated and Key Decisions.	19
3.2.1	Android	19
3.2.2	Django REST Framework.....	20
3.2.3	DigitalOcean	21
3.2.4	MySQL	21
4	Design	23
4.1	Design Methodology Choice.....	23
4.2	User Interface Design.....	24
4.3	Use Cases Diagrams.....	25
5	Architecture & Development	27
5.1	System Architecture	27
5.2	Project Components	29
5.2.1	Key Implementations	29
5.3	Problems and Challenges Encountered and Resolved	32
5.4	External API's	32
6	System Validation.....	34
6.1	Testing.....	34
6.2	Demonstration	36
6.2.1	Creating a New Fixture	36
6.2.2	Choosing a Fixture	37
6.2.3	Recording Statistics	37
6.2.4	Viewing Match Reports	39
7	Project Plan	39

8	Conclusion	40
9	Bibliography	41
10	Appendix.....	43
10.1	Usability Heuristics for Touchscreen-based Mobile Devices	43

Table of Figures

Figure 1	SportVu Tracking System	10
Figure 2	Recorded Statistics from Basketball Ireland for Players	11
Figure 3	Recorded Statistics from Basketball Ireland for Teams	12
Figure 4	iTouchStats Basketball App Input Screen	15
Figure 5	iTouchStats Basketball App Player Output	15
Figure 6	iTouchStats Basketball App Input Screen for Score Placement	15
Figure 7	Basketball Stat Keeper Action Screen.....	17
Figure 8	Basketball Stat Keeper Shot Placement Screen.....	17
Figure 9	Basketball Stat Keeper Sub Player Screen	17
Figure 10	Basketball Stat Keeper Main Input Screen.....	17
Figure 11	Summary of Heuristics Evaluation.....	18
Figure 12	Smartphone Market Share	20
Figure 13	Showcasing Python's growing popularity	21
Figure 14	Waterfall Methodology	23
Figure 15	Agile Methodology.....	24
Figure 16	Material Design Prototype.....	25
Figure 17	Initial Menu Use Case	25
Figure 18	Stat Input Use Case	26
Figure 19	System Architecture Diagram	27
Figure 20	Sample GET Request for Actions made by Android Volley	28
Figure 21	Sample Insert Request made by Android Volley	29
Figure 22	MPAndroidChart inclusion in build gradle	33
Figure 23	Declaring Bar Chart.....	33
Figure 24	Sample Barchart	34
Figure 25	Image Defining the Gestures	35
Figure 26	DIT Tournament Match Report.....	36
Figure 27	DIT Tournament Match Report.....	36
Figure 28	DIT Tournament Match Report.....	36
Figure 29	Creating a Fixture	37
Figure 30	Demonstrating all Inserts.....	39

2 Introduction

2.1 Overview and Background

Basketball is one of the most popular sports across the world, second only to soccer with more than 200 nations competing against each other in the sport[1]. It's popularity is growing at all levels year on year with recorded increases in casual participation in the United States [2] as well as increasing global tv ratings for college level competitions such as March Madness [3]. The National Basketball Association has also experienced a surge in viewership with the 2017-18 regular season averaging 3,818,000 viewers, up 17% from the previous year[4].

Many attribute this success to the leagues willingness to embrace statistics to improve the standard of play, including the NBA commissioner Adam Silver[5]. However, these clubs are spending large sums of money to achieve an extremely in-dept analysis of each game and training session to maximise their players performance. This level of analytics is far out of reach for amateur clubs which in turn effects their team's ability to improve and their player's abilities to progress past local level. For amateur team's even basic statistical records are not attainable as it typically requires a fan or parent to record these statistics. For some this act distracts too much from the game making it unenjoyable and a chore while for others the complicated nature of the applications input systems create confusion and result in errors in the recordings.

This project is an android application that provides coaches at any level of the sport with a platform to benefit from keeping a statistical record of their team's games. The user records the actions of the players on court as they occur in a game. These actions are stored on an online server and used to generate a post-match report for each individual game. This project improves upon other existing solutions by using a minimalist design and input system of custom gesture patterns. The gestures limit the amount of distraction the user will encounter when recording each statistic and overall improve the quality of their experience.

2.2 Project Objectives

The primary objective of the project is to develop an application that will provide any team or coach with post-match statistical analysis. Below is a higher-level overview of the project objectives.

2.2.1 Primary Objectives

- Create an android application capable of recording individual player statistics during a live game and store them on an online server.
- Implement custom gestures as the method of input for the statistics to reduce the user's distraction from the game and reduce errors in recording.
- Generate individual post-match reports for each recorded fixture containing graphical implementations of the data recorded. Turning a traditional box score into an intuitive design that highlights the benefits of recording.

2.2.2 Secondary Objectives

- Identify core reasons for the lack of statistical recording at amateur and underage level and provide solutions within my application
- Create a better user experience than existing solutions in terms of attention required to be spent on the app versus on the match.
- Implementing android material design aspects to design an app with a modern feel to improve on older solutions.

2.3 Project Challenges

An expected challenge for this project was the hosting and deployment of a server. It is an area of development the author had little experience in prior to beginning the project. While previously working on small class projects for module labs or assignments, never had they developed or hosted a server for a project of this size. Similarly, this was the first time the author had encountered more advanced Python and the Django REST API. Learning to use these new technologies during the course of the project was a requirement. While these technologies are well documented, and tutorials exist online they still became a frustrating source of roadblocks during the lifetime of this project as expected. These challenges were overcome but did take up more development time than was expected.

Another expected challenge was the gesture capabilities of android and the lack of custom gestures being used in apps available on the Play Store. While there are many third-party libraries that can be used with android the author was unable to find any libraries for gesture recording in android. This limited the gesture capabilities to only one option, a gesture overlay view with recorded custom patterns drawn in the gesture builder provided with the Android Studio emulators. While it is still available for use most commercial apps have steered away from this method and instead make use of on touch events. This method has a very limited amount of gestures that can be used and was not suitable for the several types of statistics that needed to be recorded. The initial worry was that any problems with this method could seriously halt the development process as there were no other alternatives if problems arose.

Lastly project management and presenting deliverables became a bigger and bigger issue as the project progressed. Initially the idea for this project was far too ambitious with hopes of implementing an automated system that would track players on court and make note of various actions that occurred. After some initial research it was decided that this idea was too big a task for the final year project, but an app could be created that would solve some of the issues that the existing solutions present. Similarly, again the author had very high hopes of implementing complicated functionality before any core functionality was established. As can be seen in the interim submission where there is a discussion of splitting the workload for one game over two phones. Unforeseen problems and setbacks quickly took up a lot of development time which resulted in more and more functionality becoming a future task rather than a goal that could be achieved in time for submission. The problem here is a combination of unreasonable project management and overpromising which left the author disheartened at times due to the lack of progress that was being made.

In the end features were abandoned to focus on creating core functionality that was of a higher standard rather than many underperforming elements. The author on reflection should have started with a small idea and built out rather than splitting focus over many complicated ideas which resulted in under delivering on various aspects of the project.

These issues will be discussed in more detail in section [Problems and Challenges Encountered and Resolved section](#).

2.4 Structure of the document

The structure of the document matches the progression of the project over the course of the project timeline. It begins with an overview of the project and its primary objectives and challenges. It then moves into research and design followed by the end project architecture and implementation.

The research section gives insight into the current situation of statistical recording and the benefit of post-match statistics for teams. The various current working solutions available and their advantages and disadvantages. Lastly, determining what the target audience valued the most out of the app and finding solutions for their issues.

The design section details the methodologies maintained during this project. Choices for the design of the user interface are explained and visuals are provided. This chapter also lists use cases for the project.

The architecture and development sections explore the implementation of the final project. This section details how communication takes place between all the elements of the project. The development process is explored in detail including problems and their solutions.

The system validation section contains details of the different forms of testing executed both in application and real-world testing. It also includes a demonstration of how the various objectives were met.

3 Research

3.1 Problem and Solution Identification Research

3.1.1 Growth of Statistic Analysis in Sport, specifically Basketball

All sports teams are in the business of winning and many have adopted a technique of intensive data analysis to gain an edge over one another. A story that's credited with changing the way all sports are played today is that of Billy Beane and the Oakland Athletics Baseball Club roster of 2002 which went on a 20-game winning streak to set an American League record and win the AL West with a record of 103-59[6]. Beane, after suffering the loss of three-star players to bigger brand teams partnered with Bill James to develop a system to buy a winning team on a small budget. Bill James discovered that on-base percentage was the key to winning games and they bought a series of players who were undervalued by traditional scouting methods but that the statistics suggested would earn more runs than they would lose. The system eventually payed off with every sport taking note of the value that statistic analysis can bring to a team[7].

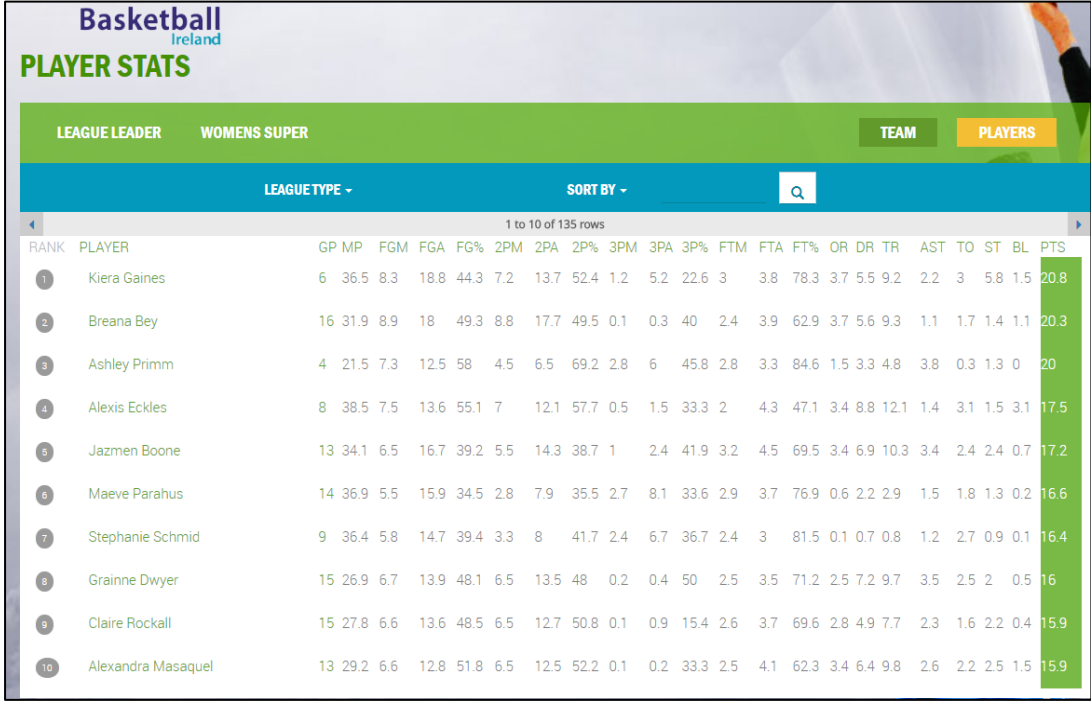
Today basketball and more specifically the NBA are the forefront of the statistical revolution with their implementation of SportVu[8]. Data analysts are now on staffing lists for nearly every team with some being credited by their clubs for the team's success. SportVu is a complicated video analysis system the league began using in 2009 that automates the process of recording statistics. A series of cameras hanging from the rafters track the movements of all players on the court and the ball[9]. The statistics gathered from this system have changed the face of the sport with the three-point shot becoming more prevalent than ever[5].



Figure 1 SportVu Tracking System

3.1.2 Statistical Recording in Ireland

It's clear that even basic shot tracking can improve a team's performance with the rise of the three-point shot but despite the growth of statistics at a professional level many amateur teams have not implemented even a basic version for themselves. Basketball Ireland have recognised the benefits of statistical analysis and have made it mandatory for all teams competing at super league level, the highest in the country. But as it is still initial stages in its implementation for the most part these statistics are ignored. The statistics can be viewed online for each individual player or for a team, but the layout is difficult to read and navigate. Players can not be grouped by team and are instead scattered throughout a list. The stats also only provide total averages over an entire season and not for individual matches. This information while useful to see how a player did in a season doesn't provide any breakdown to help with improvement game after game. In some cases the data can be misrepresentative where a player was out with injury or missed games during the season[10].



The screenshot shows the 'PLAYER STATS' page on the Basketball Ireland website. It features a green header with the site logo and navigation tabs for 'LEAGUE LEADER', 'WOMENS SUPER', 'TEAM', and 'PLAYERS'. Below the header is a blue bar with 'LEAGUE TYPE' and 'SORT BY' dropdowns, and a search icon. The main content is a table of player statistics, with a pagination bar indicating '1 to 10 of 135 rows'. The table has columns for RANK, PLAYER, GP, MP, FGM, FGA, FG%, 2PM, 2PA, 2P%, 3PM, 3PA, 3P%, FTM, FTA, FT%, OR, DR, TR, AST, TO, ST, BL, and PTS. The first 10 players listed are Kiera Gaines, Breana Bey, Ashley Primm, Alexis Eckles, Jazmen Boone, Maeve Parahus, Stephanie Schmid, Grainne Dwyer, Claire Rockall, and Alexandra Masaquel.

RANK	PLAYER	GP	MP	FGM	FGA	FG%	2PM	2PA	2P%	3PM	3PA	3P%	FTM	FTA	FT%	OR	DR	TR	AST	TO	ST	BL	PTS
1	Kiera Gaines	6	36.5	8.3	18.8	44.3	7.2	13.7	52.4	1.2	5.2	22.6	3	3.8	78.3	3.7	5.5	9.2	2.2	3	5.8	1.5	20.8
2	Breana Bey	16	31.9	8.9	18	49.3	8.8	17.7	49.5	0.1	0.3	40	2.4	3.9	62.9	3.7	5.6	9.3	1.1	1.7	1.4	1.1	20.3
3	Ashley Primm	4	21.5	7.3	12.5	58	4.5	6.5	69.2	2.8	6	45.8	2.8	3.3	84.6	1.5	3.3	4.8	3.8	0.3	1.3	0	20
4	Alexis Eckles	8	38.5	7.5	13.6	55.1	7	12.1	57.7	0.5	1.5	33.3	2	4.3	47.1	3.4	8.8	12.1	1.4	3.1	1.5	3.1	17.5
5	Jazmen Boone	13	34.1	6.5	16.7	39.2	5.5	14.3	38.7	1	2.4	41.9	3.2	4.5	69.5	3.4	6.9	10.3	3.4	2.4	2.4	0.7	17.2
6	Maeve Parahus	14	36.9	5.5	15.9	34.5	2.8	7.9	35.5	2.7	8.1	33.6	2.9	3.7	76.9	0.6	2.2	2.9	1.5	1.8	1.3	0.2	16.6
7	Stephanie Schmid	9	36.4	5.8	14.7	39.4	3.3	8	41.7	2.4	6.7	36.7	2.4	3	81.5	0.1	0.7	0.8	1.2	2.7	0.9	0.1	16.4
8	Grainne Dwyer	15	26.9	6.7	13.9	48.1	6.5	13.5	48	0.2	0.4	50	2.5	3.5	71.2	2.5	7.2	9.7	3.5	2.5	2	0.5	16
9	Claire Rockall	15	27.8	6.6	13.6	48.5	6.5	12.7	50.8	0.1	0.9	15.4	2.6	3.7	69.6	2.8	4.9	7.7	2.3	1.6	2.2	0.4	15.9
10	Alexandra Masaquel	13	29.2	6.6	12.8	51.8	6.5	12.5	52.2	0.1	0.2	33.3	2.5	4.1	62.3	3.4	6.4	9.8	2.6	2.2	2.5	1.5	15.9

Figure 2 Recorded Statistics from Basketball Ireland for Players

The layout is a traditional box-score with the data displayed in simple lists. It can be difficult to interpret these numbers to understand their true worth. In some cases, a player that scores less than others can be beneficial in off the ball tactics and may be overlooked with this setup as the numbers for non-scoring statistics are typically lower. Similarly, a team that only averages 50 points per game might seem low but if their defence stops their opponent's ability to score highly then they can still compete at the top levels. The layout of the box-score doesn't read very intuitively for these reasons. Lastly these statistics are not regularly updated, and these tables are out of date. If the tables are not as up to date as possible then the statistics are of no use to teams planning to use them for performance gain.

RANK	TEAM	GP	MP	FGM	FGA	FG%	2PM	2PA	2P%	3PM	3PA	3P%	FTM	FTA	FT%	OR	DR	TR	AST	TO	ST	BL	PTS
1	Ambassador UCC Glanmire	15	200.1	32.9	72.9	45.1	28.3	58.3	48.6	4.5	14.6	31	10.5	15.4	68	15	32	47	19.1	10.3	13.9	5.3	80.7
2	DCU Mercy	13	196.8	30.3	72.5	41.8	25.1	54.4	46.1	5.2	18.1	28.9	8.9	12.9	69	11.2	27.9	39.1	19	11.9	11.9	8.4	74.8
3	Pyrobel Killester	16	200	27.1	70.8	38.3	19.9	50.1	39.7	7.2	20.6	34.9	9.4	14.2	66.5	15.5	31.3	46.8	18.4	18.6	12.9	2.9	70.8
4	Singleton SuperValu Brunell	16	200	27.9	71.3	39.2	23.3	54.4	42.8	4.7	16.9	27.8	9.4	15.5	60.5	11.4	27.9	39.3	8.8	12.2	10.4	3.9	69.9
5	Courtyard Liffey Celtics	16	200.7	26.9	65.8	40.9	21.8	53.6	40.7	5.1	12.2	41.5	9.4	12.3	76.6	12	31.8	43.8	20.5	10.7	8.6	3.7	68.2
6	IT Carlow Basketball	14	201.8	22.7	63.6	35.7	14.9	37.6	39.6	7.9	26	30.2	9.8	15.3	64	8.2	24.2	32.4	14.1	13.4	8.1	4.6	63.1
7	Maxol WIT Wildcats	16	200	27.2	68.9	39.4	25.9	60.7	42.7	1.3	8.3	15.2	7.3	12	60.9	12.4	24.8	37.3	13.7	15.4	10.3	1	62.9
8	NUIG Mystics	15	200	21.5	72.3	29.7	18.7	57.9	32.2	2.8	14.4	19.4	10.7	16.7	64	9.3	22.7	32	6.4	12.3	13.5	1.9	56.4
9	Portlaoise Panthers	9	200	20.8	56.2	37	16.9	40.6	41.6	3.9	15.7	24.8	10.2	16.2	63	8.3	24.6	32.9	14.1	17.4	5.4	3.6	55.7

Figure 3 Recorded Statistics from Basketball Ireland for Teams

3.1.3 User Requirements

While the author has a lot of experience recording statistics both as a table official and as a dedicated stat recorder before beginning the design process it was decided that discussions with the target audience would be essential. While there are big improvements that can be made in the representation of the data a lot of user's key issues stem from the act of recording. The lack of visual representation then adds to the unwillingness to record stats as the effort is not worth the result. But the main advantage the current solutions will have over this project is time. There are two existing solutions that are of a high quality and are used by many teams in the super league. The most popular was first released in 2011 and has had many new features added and updates over time. Due to the short project timeline it will be impossible to implement every feature to the same high standard. But by targeting the specific audience this project can be tailored to address issues that arise with the existing solutions, giving the project an edge over it's competitors.

Discussing the disadvantages with several table officials, referees, coaches and stats recorders was very beneficial. Trends appeared with the conversations with each representative having similar grievances. Firstly, of all the stats recording apps available online, two specific apps stood out to be popular. They were of the highest quality in terms of how much data could be recorded and this was the key feature when choosing an application to use. An application at the very least must be able to record scores, rebounds, steals and blocks to be worth the effort of recording. Any simpler solutions were rejected by most people during these discussions. But too many actions to record and the app becomes confusing to keep track of what is happening on court.

More to this point, solutions that require a lot of screen swapping or multiple clicks for one stat to be recorded are too distracting. The users found they spent more time looking at the phone and trying to catch up with what was happening than enjoying the game. For many users they are recording these stats for a child or other family member and not being able to enjoy the game with other supporters is a turn off. Lastly many coaches do not end up using the data to improve their team or plan training sessions. The displaying of the stats after a game is lacking and they provide little more benefit than the scoresheet used to record the game. Most coaches prefer to save copies of a scoresheet rather ask a user to record statistics.

3.1.4 Existing Solutions

After the above discussions had taken place it was clear that improvements could be made by implementing the gesture control and intuitive graphical post-match reports. Using the Usability Heuristics for Touchscreen-based Mobile Devices an evaluation was completed that considered the authors personal experience as well as that of the various members of the target users. This evaluation was used to determine the necessary project requirements that would result in overall improvement in terms of functionality and user interaction with the new solution.

Usability Heuristics for Touchscreen-based Mobile Devices are heavily inspired by Nielsen's heuristics[11]. This was the best set of heuristics to use for evaluation as Nielsen's are highly regarded and used commonly. This alternative encompasses all the advantages of Nielsen's with a focus on a mobile interface as well.

3.1.4.1 iTouchStats Basketball[12]

- 1) **Visibility of system status:** When an action is completed the text bar at the top of the screen displays the last action that was successfully entered. No other feedback to suggest an action was completed correctly. Small colour coded squares containing numbers to show the user a summary of each player's current statistics. They add more to the cluttered look of the screen and the colours are too similar in some cases to be helpful. 3/10
- 2) **Match between system and the real world:** Speaks the user's language. The status text bar uses full sentences to describe actions logged. It refers to players by number and last name which is the same convention as with the official scoresheet for a match. Some buttons have awkward abbreviations for names due to spacing problems such as Def Reb. 5/10
- 3) **User control and freedom:** An undo method will delete the last entered statistic without having to switch screens. A past statistic can be fixed by viewing a list of previous actions and modifying or deleting them. A delete button exists but it is quite small and easy to miss. To see the extended list, you must click on the text bar, but this isn't very intuitive. 7/10
- 4) **Consistency and standards:** Consistent recording method throughout a match, but the screens are not of a good quality. This app also has a steep learning curve which can prevent familiarity. 6/10
- 5) **Error prevention:** Main input screen layout is very confusing and time-consuming to use which can cause many errors while recording. The number of buttons and colour coded icons causes a cluttered look which can become hard to navigate. Some of the colours used are very similar which makes the icons less useful. The app removes unavailable functionality while in the middle of recording by opening new screens or smaller windows. 1/10
- 6) **Minimize the user's memory load:** Main screen of this app overloads the user with information to remember with the colour coded icons and button

layout. While every option is visible the overall layout cancels out the usefulness of the action buttons being on one screen 1/10

- 7) **Customization and shortcuts:** Customisations in the form of switching the substitutes on and off the bench as well as the order the players appear on the side bar. Experts can add on or take away unnecessary statistics they want to record. 6/10
- 8) **Aesthetic and minimalist design:** Colour coded icons are rarely needed while recording the match statistics. They are designed to help with errors but often can be irrelevant, for example, a blue icon is the number of overall rebounds for a team, but three types can be recorded. The team members listed at the bottom of the screen are also unnecessary for most of recording time. They could easily be moved to another screen to allow more for more space3/10
- 9) **Help users recognize, diagnose, and recover from errors:** The system does not diagnose errors but instead provides users with different delete options to recover from an incorrect record. 5/10
- 10) **Help and documentation:** No in app documentation for the input and instead relies on the user watching a tutorial on YouTube. No direction for the user to find the video. 4/10
- 11) **Physical interaction and ergonomics:** There is no order to how the buttons have been displayed on the main screen. This requires the user to learn this new layout as it is not very intuitive. Due to the number of buttons needed for this system, it is impossible to cater for the natural position of a hand. This requires the user to change their natural input method. This leads to extra time being needed to get accurate statistics recorded as well as increasing the likelihood of errors. 3/10

The app can record every stat needed for a full match report, but it lacks in usability. All the necessary statistics can be recorded but the overall layout and functionality are not user-friendly. While the individual player profiles are useful most coaches during testing preferred report for each match rather than individually selecting each player.



Figure 4 iTouchStats Basketball App Input Screen



Figure 6 iTouchStats Basketball App Input Screen for Score Placement



Figure 5 iTouchStats Basketball App Player Output

3.1.4.2 Basketball Stat Keeper[13]

- 1) **Visibility of system status:** No feedback to suggest an action has been completed correctly for certain inputs on the main screen. Displays the points and fouls statistics for each player on the main screen but to see the rest of the statistics the user needs to click on each individual player. The player must be active in the app or on court to view these extra statistics. 2/10
- 2) **Match between system and the real world:** The app doesn't provide long blocks of text for the user but the little language that is used is simple and straightforward. 8/10
- 3) **User control and freedom:** Accessible undo button that will delete the last entry made. Difficult to see working as the only feedback given other than the change being visible in the player's summary is a very quick toast. If a user wants to edit an entry made for a player that is older than the most recent entry then the user must move that player to active, click the player and change the next screens mode to edit. Fixing errors in this app can be very complicated and time-consuming. 2/10
- 4) **Consistency and standards:** Consistent recording method throughout a match and the screens are of a good uncluttered quality. This app has a steep learning curve which can prevent familiarity. 7/10
- 5) **Error prevention:** Main screen layout has a slimline look as most action buttons are not present but instead, the user clicks through more screens to record basic functionality. Every action requires the player to be selected first and this takes the user to a screen of buttons to choose from which all look alike. This can be confusing when trying to find the correct action quickly as the buttons are all similar. Depending on the action another screen may be

opened to finish the recording. The app removes unavailable functionality while in the middle of recording by opening new screens or smaller windows. Overall this should reduce errors when recording but it increases recording time and demands more of the user's attention. 3/10

- 6) **Minimize the user's memory load:** A lot of screens to navigate through to record actions, there are very few differences between the actions. The user has less information to remember overall. However, it can be difficult at first to understand how to navigate these screens as there is no information provided in the app explaining how to record any actions. 4/10
- 7) **Customization and shortcuts:** Customisation in the form of switching the substitutes from active to inactive. Experts can add on or take away unnecessary statistics they want to record. However, there are shortcuts to features that are unnecessary, for example being able to create a new player profile in the middle of recording a game. 5/10
- 8) **Aesthetic and minimalist design:** Main screen avoids displaying too much information and cluttering. The most important statistics are displayed and keep the screen slim lined. A summary of the total score is displayed at the top of the screen. Rarely needed information like jump balls and turnovers that occur less often for each player can be seen by clicking on a different screen. 6/10
- 9) **Help users recognize, diagnose, and recover from errors:** System does not diagnose errors but instead provides users with different delete options to recover from an incorrect record. There is no timeline of all action entries that have been recorded during a match. This can mean deleting a specific incorrect entry can be very difficult, for example, if the user needs to delete a three-pointer that was scored earlier in the game it is not possible to select which three-pointer to deduct. This renders the shot chart feature irrelevant as a random three-pointer will be deleted rather than being able to pick the one that is incorrect. 5/10
- 10) **Help and documentation:** The app provides no documentation for the input and there is none available online. Blind guessing is required to learn the methods of recording. 2/10
- 11) **Physical interaction and ergonomics:** Due to the number of buttons needed for this system it is impossible to cater for the natural position of a hand. This natural input method. This leads to extra time being needed to get accurate statistics recorded as well as increasing the likelihood of errors. The shot chart feature is not very accurate as a finger can easily place a marker in the wrong place but there is only one chance to place the marker before the score is recorded. Editing on the shot chart is not possible. 3/10

<div> <div>You: 8</div> <div>Points: 8 0 0 0</div> <div>Fouls: 1 0 0 0</div> </div> <div> <div>Opp: 0</div> <div>Points: 0 0 0 0</div> <div>Fouls: 0 0 0 0</div> </div>					
Time	#	Plyr	Fouls	Pts	Eff.
-		Totals	1	8	11
00:00	34	Alanna...	0	4	4
00:00	33	Chante...	1	4	4
00:00	8	Erica S...	0	0	-2
00:00	4	Katie Fox	0	0	2
00:00	12	Olivia ...	0	0	3
Inactive players (tap to bring on court)					
Move active players to bench					
00:00	34	Alanna...	0	0	0

Figure 10Basketball Stat Keeper Main Input Screen

<div> <div>You: 8</div> <div>Points: 8 0 0 0</div> <div>Fouls: 1 0 0 0</div> </div> <div> <div>Opp: 0</div> <div>Points: 0 0 0 0</div> <div>Fouls: 0 0 0 0</div> </div>					
Time	#	Plyr	Fouls	Pts	Eff.
00:00	4	Katie Fox	0	0	2
00:00	12	Olivia ...	0	0	3
Inactive players (tap to bring on court)					
Move active players to bench					
00:00	34	Alanna...	0	0	0
00:00	16	Sara G...	0	0	0
<div> <div>+</div> <div>Add New Player</div> </div> <div> <div></div> <div>Shot Chart</div> </div>					

Figure 9Basketball Stat Keeper Sub Player Screen

Shot Chart

My Team ☒

Opponent Team

Select Player

All Players

Back

Figure 8Basketball Stat Keeper Shot Placement Screen

Chantel Alford

Two Point Made

Two Point Miss

Three Point Made

Three Point Miss

Free Throw Made

Free Throw Miss

Foul

Assist

Offensive Rebound

Defensive Rebound

Block

Turnover

Steal

Jump Ball

Figure 7Basketball Stat Keeper Action Screen

The app can record every stat needed for a full match report, but it lacks in error correction. All the necessary statistics can be recorded. Forcing a player to be chosen before an action did reduce errors in recording unlike with the previous application.

3.1.4.3 Summary of Heuristic's Evaluation

	iTouchStats Basketball	Basketball Stats Keeper
Visibility of system status	3	2
Match between system and the real world	5	8
User control and freedom	7	2
Consistency and standards	6	7
Error prevention	1	3
Minimize the user's memory load	1	4
Customization and shortcuts	6	5
Aesthetic and minimalist design	3	6
Help users recognize, diagnose, and recover from errors	5	5
Help and documentation	4	2
Physical interaction and ergonomics	3	3
Totals	44/110	47/110

Figure 11 Summary of Heuristics Evaluation

After the evaluation of both apps it is clear they are both strong apps functionally. All possible stats can be recorded but with varying degrees of difficulty. These solutions are well established and have been available for seven years in some cases. This project is a much shorter time frame so matching them for all their various levels of functionality will be a challenging task time wise. But by taking what was learned in discussions with the target audience it is possible to improve upon features.

This app will need to reduce the amount of effort that is required to input a single statistic and avoid using multiple screens where possible. This will have the combined effect of lessening distractions and decreasing input time. The reports generated from the data should be easy to understand and follow while also providing more advantages than a simple scoresheet. Limiting the amount of information that is displayed on the input screen will decrease recording confusion. The most important statistic is score and should be visible on this screen, less important statistics can be hidden away as they are not required constantly. A timeline of inputted events should be used for easy deletion as it was the users preferred method.

3.2 Technologies Evaluated and Key Decisions.

3.2.1 Android

Android was chosen as the projects development medium for multiple reasons. The most obvious choice for hosting this program was a mobile device. These statistics are recorded for both home and away matches with certain venues requiring lots of travel. A mobile phone would be the most convenient device to use while traveling as it's small, portable and most users will already be bringing one with them. Other devices like a laptop may also be dependent on plug sockets for lasting the duration of a match, whereas a mobile phone will in most cases not require charging. The most important reason for choosing a mobile device was the need for a touch screen to implement the gesture input system. While some high-end laptops and devices may have this capability, it would not be a common feature unlike with a mobile phone where the majority have this as standard.

The choice then became IOS, android or a hybrid application. While hybrid applications are easier to scale and can work cross platform with one codebase, native was the better choice for this style of application. Native applications are faster and give a more reliable performance which will be essential when communicating with the online server during the stress of a live fast-paced game situation. Implementing custom gestures which are the key feature of this project will also be easier on a native application. Hybrid apps are for quick deployment when the speed at which you get to market is a key factor for the project. That is not a concern for this application and the more reliable app should be coded natively[14]. Lastly having coded in android before and not IOS the author felt starting this project in a language they were comfortable using was essential as other aspects of the project would require learning from scratch such as Django REST API. When researching the market share for smartphones it is overwhelmingly clear that Android is dominating due to their lower price range compared to IOS[15]. By choosing to develop in android a greater percentage of people would be able to avail of the app.

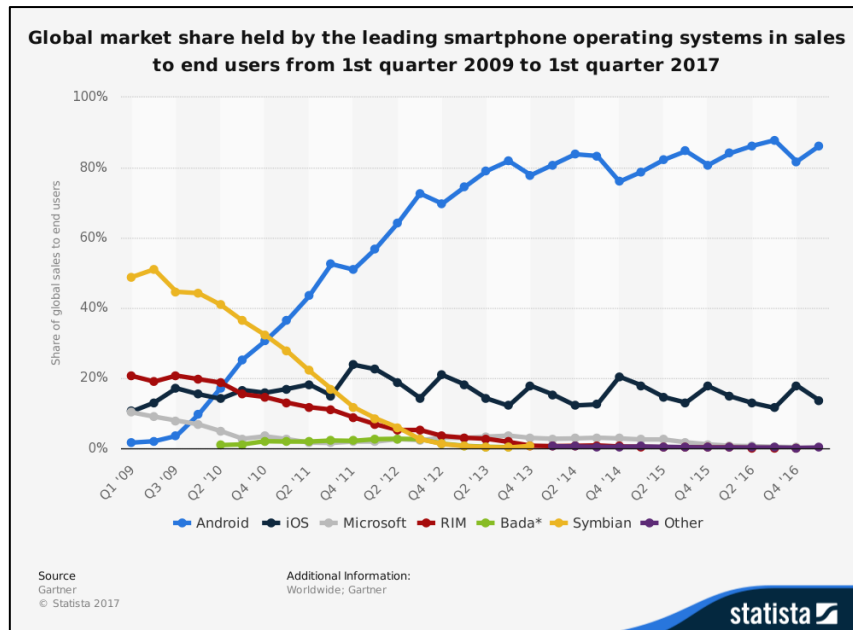


Figure 12 Smartphone Market Share

3.2.2 Django REST Framework

Before beginning this project, the author had very little knowledge of web frameworks. Researching Flask and Django was the first point of call as these are the most popular frameworks that are used for both small projects and large companies alike. Flask is described as a micro-framework meaning it's fast and straightforward to set up but is stripped back to handle the bare minimum in terms of functionality. External libraries are used to add more functionality if it's needed. When researching the differences between these two frameworks Flask was described as perfect for simple web apps. Having little experience and being unsure of the requirements of my application when it came to the server this seemed risky. Taking this into account Django was chosen as it comes with more capabilities in case they were needed later in the development of the project.

Aside from Django being the well-rounded framework it is also coded in python. While having little experience in Python it was understood that it would be a good skill to obtain. Also, with upcoming modules during the year that would involve Python it was a desirable choice to get some experience using it. It is a skill with great benefits due to its rise in popularity over the last few years. "Worldwide, Java is the most popular language, Python grew the most in the last 5 years (10.9%) and PHP lost the most (-5.5%)"[16]. As this language is growing in popularity within the industry jobs associated with this skill are too. Personal development would improve by working in this area. Lastly, the Python community online offered great support in terms of help, documentation and tutorials.

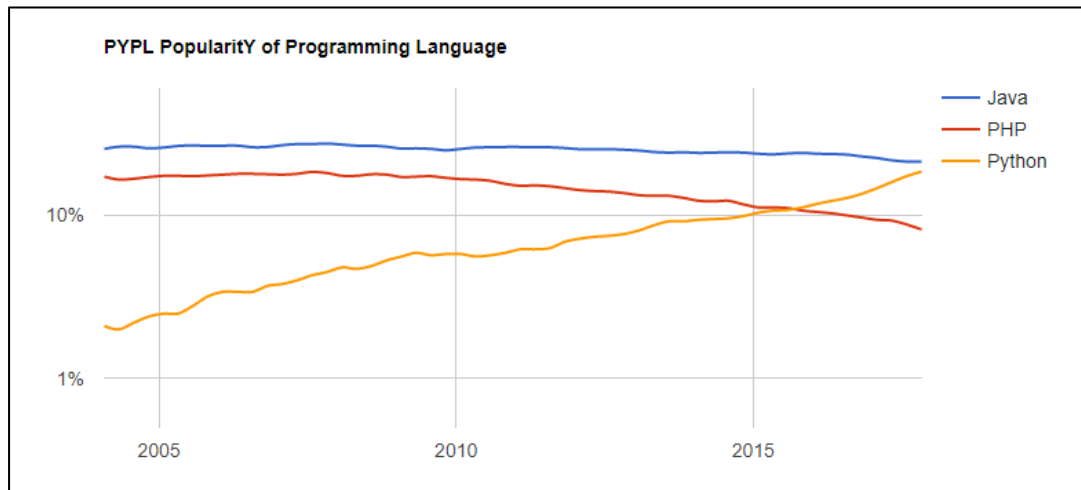


Figure 13 Showcasing Python's growing popularity

3.2.3 DigitalOcean

Avoiding a standalone application and in the effort to create a vertical prototype for the interim an online server was needed to host the database. Using an online provider, the data will always be available if the app has a connection to Wi-Fi and the server is running. These online providers are very reliable unlike a local server on a laptop that might not always be running. Having used Amazon Web Services in a cloud computing module as part of this course it was decided that researching other providers could be of benefit having found it difficult in the past.

After some quick research DigitalOcean was the chosen provider. A student developer pack from GitHub offers credit for DigitalOcean's services. The cheapest tier was more than enough to get started and the credits would pay for this tier until the end of development. Within minutes a droplet was running with an Ubuntu server deployed and running online. This process could not have been more straightforward to understand. Following from that DigitalOcean has tutorials available online that detail many topics including Django, Java and Python which were very useful for this project. The online community on DigitalOcean provide a lot of discussion and documentation which was a key factor when choosing a provider.

3.2.4 MySQL

The database needed to be compatible with DigitalOcean and Django. Researching the options, the most popular were SQLite, MySQL and PostgreSQL. SQLite is the default database that is used by Django when creating applications. While it is fast and efficient it works best with single user applications. If the application that's being designed will have multiple clients all using the one database, it is not the best solution. Given that multiple users will all be recording different matches at the same time during testing continuing with SQLite did not seem like the best option.

PostgreSQL is the more advanced option that the author had not used before and had little knowledge of. PostgreSQL offers a very reliable system that implements ACID properties by using Multiversion Concurrency Control. But this level of concurrency can be overkill for most simple setups such as smaller projects like this one. Speed in terms of read operations makes it less desirable than other options which were a

considerable factor for this project as sending multiple requests to a server at a fast pace was essential. Overall these factors, as well as little knowledge of the platform, ruled out this option.

After researching MySQL was the solution for this project. Its popularity gave this option an edge as there is a wealth of information online making getting started and troubleshooting easier. MySQL databases work with a lot of data and scale very easily which made it the best choice early in the project when requirements were not decided fully. Lastly, MySQL does not compromise on speed, unlike other solutions. After weighing all options and having had previous experience with MySQL, it was the right solution for the project[17].

4 Design

4.1 Design Methodology Choice

As the requirements for this project are well defined at the beginning of this report it was possible to consider simpler methodologies. A more traditional method such as waterfall works in a sequential process. Developers work on each of the steps until they have been perfected and only then will they move onto the next step. If there is a need to go back to the work that was done, they must start from the beginning. The entire project will be designed and functioning before it has been tested by a potential user[18]. This is a very straightforward model to follow that can be very beneficial for simple products that will not encounter a need for change to the original design. As mentioned in the [Project Challenge](#) section earlier the author struggled with redesign throughout the lifecycle of the project. While the initial requirements stayed the same other features that would have made the app a more well-rounded product were abandoned. This was due to setbacks that were met along the way which required changes to be made to the original design.

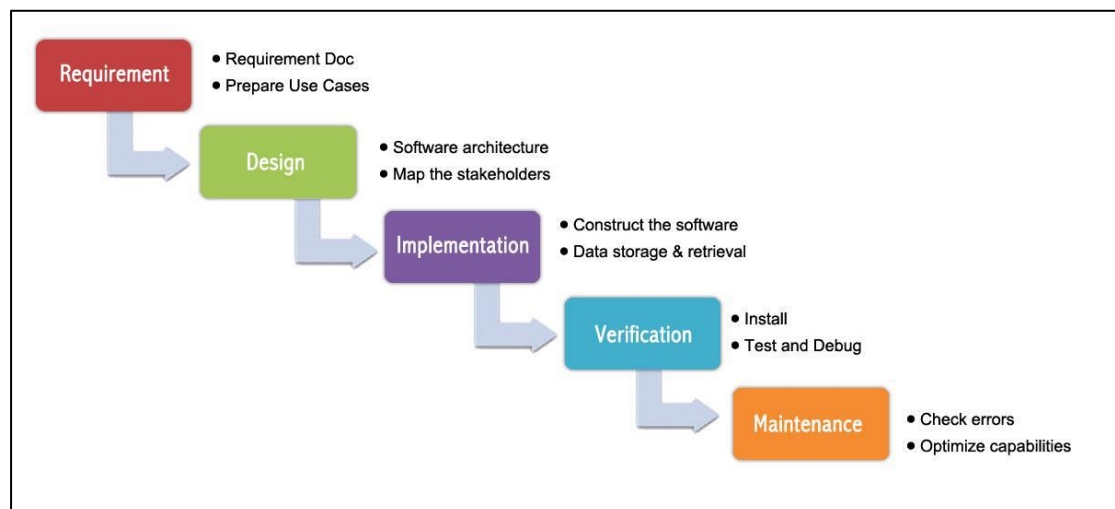


Figure 14 Waterfall Methodology

There will also be future iterations of the project and this allows for endless functionality that could be added. To accommodate for the potential for change an agile methodology was the best approach to stick to. The project can be broken into separate parts to develop, with each new section of the project working independently from the next and changes in requirements can be accounted for. Also, as the author is a member of both a local and college club they can avail of the constant feedback from other players and coaches. Constantly having the ability to test out features in game situations brought issues to the forefront well in advance and without this iterative work cycle errors could have gone unnoticed for too long and become impossible to fix.

There was also an element of prototyping being used throughout the development process. As the gestures are a unique idea that are not seen in many commercial apps nowadays there was no precedent in how to develop them. Multiple versions of inputting were tested rather simply before the best was chosen to be the selected style. This will be explained in greater detail in the [Project Components](#) section. Once a

design was chosen and the requirements were fully defined a feature focused methodology was followed. Creating a individual components of the app that would later come together[19].

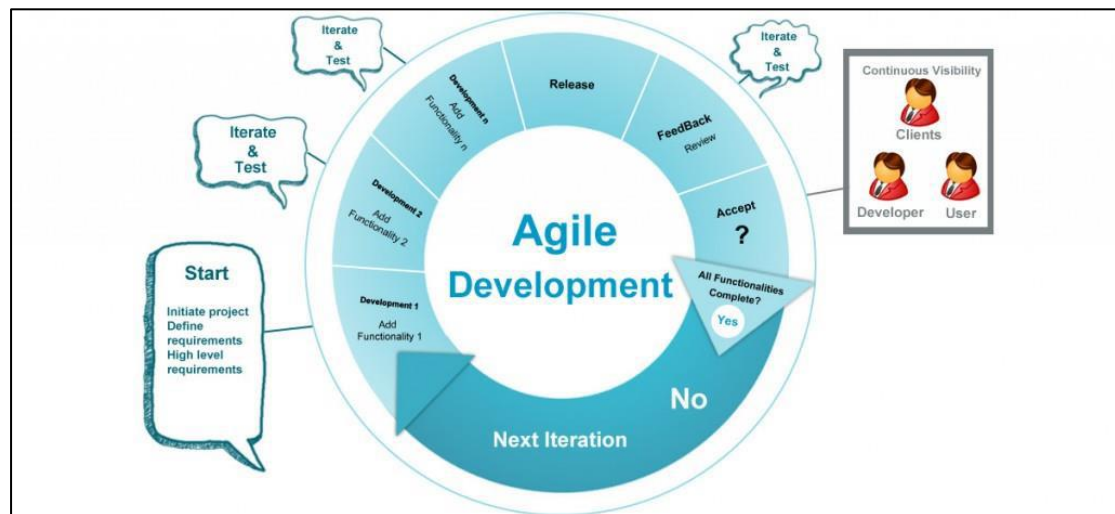


Figure 15 Agile Methodology

The design section details the methodologies maintained during this project. It includes a description of each component in the project both frontend and backend. Choices for the design of the user interface are explained and visuals are provided. This chapter also lists use cases for the project.

4.2 User Interface Design

The below figure is a prototype that was built early in the project stages to create a mock-up of the user interface. The pages are features that would appear in a fully functioning app that is ready to be released. Initially the author was hoping to implement these screens but as they were not essential for recording or reporting they were put off until a future iteration. They are still included in the submission of the app to demonstrate what it will look like when fully functional. They follow the android material design guidelines closely to create a modern and intuitive design. These guidelines are for developers to follow to create high end looking apps. The existing solutions are all quite old looking and by following these guidelines this project is a standout when comparing interfaces. Some testers also noted that the tab design resembles that of Facebook Messenger or WhatsApp and they felt they understood how to navigate through the app quickly as a result. This is like Nielsen's Heuristics as the testers were demonstrating consistency and standards. The project implements a list system to display all inputs that have been made that can be deleted to implement recognition rather than recall. The gestures also work in a consistent manner to benefit from recall, but an image is also provided to if recognition is needed. The colour scheme is also consistent throughout they app.

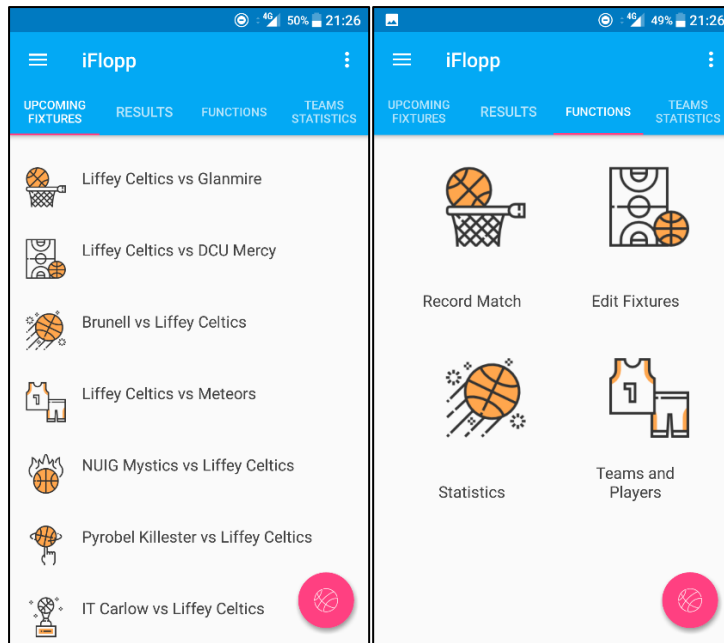


Figure 16 Material Design Prototype

4.3 Use Cases Diagrams

This use case diagram shows how a user can interact with the program. It should be a straightforward process where the user is not confused about the navigation of the application. On initial loading of the program the user can either create a new fixture in the system. They are then brought to a set of input buttons to fill in the info for the server. If they wish to view the result of a game they can choose the result report that is displayed in a list.

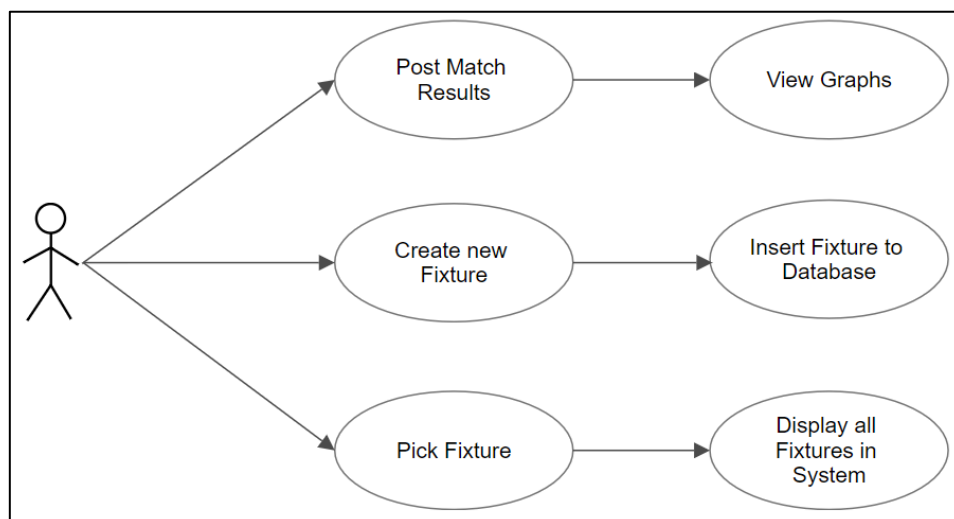


Figure 17 Initial Menu Use Case

The second use case describes how a user can input individual player stats for a

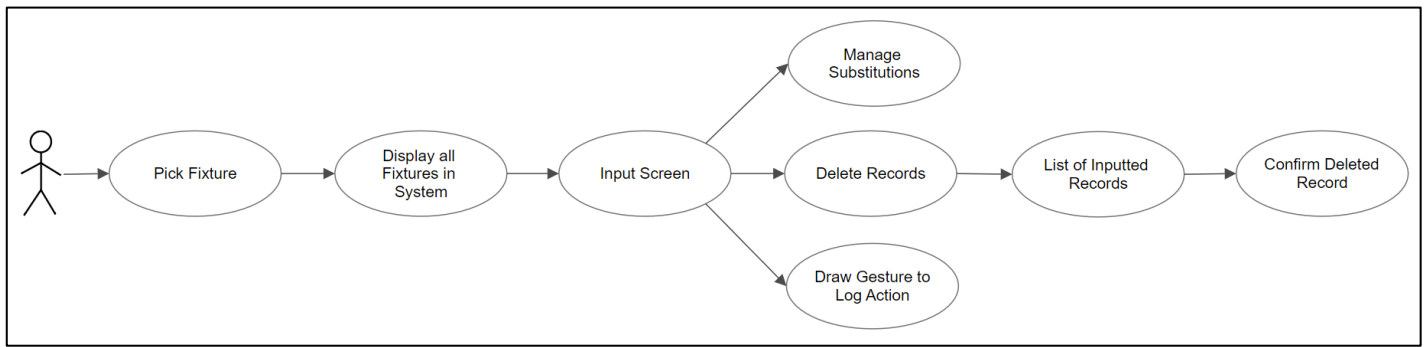


Figure 18 Stat Input Use Case

particular fixture. First they choose the pick fixture button which generates the list of possible fixtures. Once they choose the right fixture the input screen is opened. In here they can draw gestures to send stats to the server. Substitute players in or out depending who is on court at the time. Lastly if a mistake was made during recording they can open a list of all stats they've inputted and can delete any that are incorrect.

5 Architecture & Development

5.1 System Architecture

The front end of the design consists of an android mobile device. This device makes use of a Django REST API to communicate with the database on the online server. The API was created using the Django REST framework and is installed on the online server. The Android device uses the Volley library to send requests to a specific URL named in the app that corresponds to the correct Django model for the request. If the application needs to make a GET request to see all the actions that have been stored it will call a JSON array request to `http://178.62.2.33:8000/api/action/?format=json` to list all the actions stored in the database. If permission is granted, a response code of 200 is delivered. This means permission for the request is granted and the data is returned. The JSON data at this location is retrieved and the Android app manipulates the resulting arrays to determine which actions belong to which players and what fixture they're related to and so on.



Figure 19 System Architecture Diagram

The Django API gathers the information for this JSON request by communicating with the MySQL database also stored on the server. The MySQL database contains all the tables needed to store the information that is entered by the user. When action statistics are being entered by the user during a match the requests are sent to the database through the API very similarly to the GET request. Both the API and database are running on an Ubuntu server that has been deployed on the cloud service provider DigitalOcean.

```

private void getAllActions() {
    String url = "http://178.62.2.33:8000/api/action/?format=json";
    JsonRequest jsonArrayRequest = new JsonRequest // PARSING THE JSON VALUES
        (url, new Response.Listener<JSONArray>() {
            @Override
            public void onResponse(JSONArray response) {
                action_length = 0;
                action_array = new String[(response.length())];
                fixture_array = new String[(response.length())];
                action_id_array = new String[(response.length())];
                action_player_array = new String[(response.length())];
                try{
                    // Loop through the array elements
                    for(int i=0;i<response.length();i++){
                        // Get current json object
                        JSONObject json_object = response.getJSONObject(i);
                        // Get the current team (json object) data
                        String id = json_object.getString( name: "action_id");
                        String type = json_object.getString( name: "action_type");
                        String fixture = json_object.getString( name: "fixture_id");
                        String player = json_object.getString( name: "player_id");
                        int j = i;
                        action_length = action_length +1;
                        //add to the array used for POST
                        action_array[j]= type;
                        fixture_array[j]= fixture;
                        action_id_array[j]= id;
                        action_player_array[j]= player;
                    }
                    //Log.d(TAG, "action_array"+Arrays.toString(action_array));
                    //Log.d(TAG, "fixture_array"+Arrays.toString(fixture_array));
                    //Log.d(TAG, "action_id_array"+Arrays.toString(action_id_array));
                    //Log.d(TAG, "action_player_array"+Arrays.toString(action_player_array));
                    //Log.d(TAG, "action_length"+(action_length));
                    changeUrlsToNames();
                } catch (JSONException e){
                    e.printStackTrace();
                }
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                // TODO Auto-generated method stub
            }
        });
    MySingleton.getInstance(getContext()).addToRequestQueue(jsonArrayRequest);
}

```

Figure 20 Sample GET Request for Actions made by Android Volley

```

private void insertTheStat() {
    String url = "http://178.62.2.33:8000/api/action/?format=json";
    JSONObject json = new JSONObject();
    try {
        json.put( name: "action_type", selectedType);
        json.put( name: "player_id", value: "http://178.62.2.33:8000/api/player/"+selectedPlayer+"/?format=json");
        json.put( name: "fixture_id", value: "http://178.62.2.33:8000/api/fixture/"+fixture_id+"/?format=json");
    } catch (JSONException e) {
        e.printStackTrace();
    }

    final JsonObjectRequest postRequest = new JsonObjectRequest(url, json,
        new Response.Listener<JSONObject>()
        {
            @Override
            public void onResponse(JSONObject response) {
                // response
                Log.d(TAG, String.valueOf(response));
                selectedPlayer = "0";
                selectedType = "0";
                playerSelectedView.setText("Chosen Player: ");
                actionSelectedView.setText("Chosen Stat: ");
            }
        },
        new Response.ErrorListener()
        {
            @Override
            public void onErrorResponse(VolleyError error) {
                // error
                Log.d(TAG, msg: "Error.Response: " + error.getMessage());
            }
        }
    );
    MySingleton.getInstance(StatInput.this).addToRequestQueue(postRequest);
}

```

Figure 21 Sample Insert Request made by Android Volley

5.2 Project Components

5.2.1 Key Implementations

The most important feature of this project is the ability to record statistics. Without this feature the app is useless. The gestures work by using a gesture overlay that encompasses the entire screen.

```

GestureOverlayView gestureOverlay = (GestureOverlayView) findViewById(R.id.gestures);
gestureOverlay.addOnGesturePerformedListener(StatInput.this);

```

```

private void makeDecisions() {
    //selectedPlayer = "0"; move to input
    if(result.equals("left") || result.equals("right")) {
        switch (result) {
            case "left":
                Log.d(TAG, msg: "left");
                teamSelected = away_id;
                teamSelectedView.setText("Away");
                circlePlayerView.setText("Circle: "+circle_player_away_name);
                squarePlayerView.setText("Square: "+square_player_away_name);
                trianglePlayerView.setText("Triangle: "+triangle_player_away_name);
                semicirclePlayerView.setText("Semi-Circle: "+semicircle_player_away_name);
                heartPlayerView.setText("Heart: "+heart_player_away_name);
                break;
            case "right":
                Log.d(TAG, msg: "right");
                teamSelected = home_id;
                teamSelectedView.setText("Home");
                circlePlayerView.setText("Circle: "+circle_player_home_name);
                squarePlayerView.setText("Square: "+square_player_home_name);
                trianglePlayerView.setText("Triangle: "+triangle_player_home_name);
                semicirclePlayerView.setText("Semi-Circle: "+semicircle_player_home_name);
                heartPlayerView.setText("Heart: "+heart_player_home_name);
                break;
        }
    }
}

```

When a user swipes left they select the Away team while right selects the home team. Once a team is selected they now must choose a player by drawing a shape. The following if statement ensured the correct players id was chosen based on the shape drawn and changed a text view to notify the user that the person has been selected.

```
else if(result.equals("circle") || result.equals("square") || result.equals("triangle") || result.equals("semicircle") || result.equals("heart"))
    switch (result) {
        case ("circle"):
            Log.d(TAG, msg: "circle");
            if(teamSelected == away_id){
                selectedPlayer = circle_player_away;
                playerSelectedView.setText("Chosen Player: "+circle_player_away_name);
            }
            if(teamSelected == home_id){
                selectedPlayer = circle_player_home;
                playerSelectedView.setText("Chosen Player: "+circle_player_home_name);
            }
            Log.d(TAG, msg: "selected_player"+(selectedPlayer));
            break;
        case ("square"):
            Log.d(TAG, msg: "square");
            if(teamSelected == away_id){
                selectedPlayer = square_player_away;
                playerSelectedView.setText("Chosen Player: "+square_player_away_name);
            }
            if(teamSelected == home_id){
                selectedPlayer = square_player_home;
                playerSelectedView.setText("Chosen Player: "+square_player_home_name);
            }
            Log.d(TAG, msg: "selected_player"+(selectedPlayer));
            break;
    }
```

Now the action must be selected that will be sent to the server. Another If statement controls this. The if statement will only fire if a player has been selected and

```
if (selectedPlayer != "0") {
```

After this the insert code shown previously will be called. If the user wishes to make a substitution the team id is sent to the substitution activity using an intent and a GET request is made for all players from that team. Then after all players are assigned a shape the player values are sent back to the input activity. A delete works very similarly to a GET. Except instead of requesting the information you are sending the information to be deleted. Alert dialogs were used as a method of confirmation from the user.

```
heart_button = (Button) findViewById(R.id.btnHeart);
heart_button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        AlertDialog.Builder builder = new AlertDialog.Builder( context: SubstitutionPage.this, R.style.AppCompatAlertDialogTheme);
        builder.setTitle("Select Player").setAdapter(dataAdapter, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                // TODO Auto-generated method stub
                selectedHeart = dataAdapter.getItem(which);
                for (int i = 0; i < names_array.length; i++) {
                    if ((selectedHeart.equals(names_array[i]))) {
                        int j = i;
                        heart_value = player_array[j];
                        heart_value_name = selectedHeart;
                        Log.d(TAG, msg: "Heart Player Value" + String.valueOf(heart_value));
                    }
                }
                if(circle_value.equals(heart_value) || square_value.equals(heart_value) || triangle_value.equals(heart_value) || semicircle_value.equals(heart_value)){
                    heart_value = " ";
                    heart_value_name = " ";
                    Log.d(TAG, msg: "Heart Player Value" + String.valueOf(heart_value));
                    Toast.makeText( context: SubstitutionPage.this, text: "Player Chosen Already", Toast.LENGTH_LONG).show();
                }
                else{
                    heart_button.setText(selectedHeart);
                }
            }
        });
        builder.show();
    }
});
```

```

CustomListAdapter listAdapter = new CustomListAdapter( context: this, nameArray);
listView = (ListView) findViewById(R.id.listviewID);
listView.setAdapter(listAdapter);

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        AlertDialog.Builder alert = new AlertDialog.Builder( context: DeleteRecords.this);
        alert.setTitle("Delete entry");
        alert.setMessage("Are you sure you want to delete?");
        alert.setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                // continue with delete
                selected_action_id = action_id_arraylist_no_nulls.get(position);

                Log.d(TAG, msg: "selected_action_id"+ selected_action_id);
                String url = "http://178.62.2.33:8000/api/action/"+selected_action_id+"?format=json";
                StringRequest jsonArrayRequest = new StringRequest(Request.Method.DELETE, url, // PARSING THE JSON VALUES
                    new Response.Listener<String>() {
                        @Override
                        public void onResponse(String response) {
                            nameArray.remove(position);
                            action_id_arraylist_no_nulls.remove(position);
                            listAdapter.notifyDataSetChanged();
                            Log.d(TAG, msg: "DONE");
                        }
                    }, new Response.ErrorListener() {

                        @Override
                        public void onErrorResponse(VolleyError error) {
                            // TODO Auto-generated method stub
                        }
                    }
                ));
                MySingleton.getInstance(DeleteRecords.this).addToRequestQueue(jsonArrayRequest);
            }
        });
        alert.setNegativeButton(android.R.string.no, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                // close dialog
                dialog.cancel();
            }
        });
        alert.show();
    }
});

```

A singleton class was used to create one request queue that lasted the length of the app running for each of these request calls

```

// Singleton class
public class MySingleton {
    private static MySingleton mInstance;
    private RequestQueue mRequestQueue;
    private static Context mContext;

    private MySingleton(Context context) {
        mContext = context;
        mRequestQueue = getRequestQueue();
    }

    public static synchronized MySingleton getInstance(Context context) {
        if (mInstance == null) {
            mInstance = new MySingleton(context);
        }
        return mInstance;
    }

    public RequestQueue getRequestQueue() {
        if (mRequestQueue == null) {
            // getApplicationContext() is key, it keeps you from leaking the
            // Activity or BroadcastReceiver if someone passes one in.
            mRequestQueue = Volley.newRequestQueue(mContext.getApplicationContext());
        }
        return mRequestQueue;
    }

    public <T> void addToRequestQueue(Request<T> req) { getRequestQueue().add(req); }
}

```

The remaining features all work by making a call to the server and displaying the response. They all work the same as the example GET's I included earlier.

5.3 Problems and Challenges Encountered and Resolved

There were big problems with server communications. Python and Django had to be learned from scratch and this became cumbersome at times. It's well documented but none the less took a lot of time to get working. This took away time allocated towards finishing features later at the end of the project.

Several features would be required to make this app fully functional and viable for use in competitive games. The absence of some of these is due to them being of a lower priority in the objectives. They can be added in a later build.

Gestures took a lot of time to get working as the only way to do a custom gesture is to use gesture overlay which is provided by android, but it can be buggy at times. Trying to weed out these bugs was a tiresome process and sucked up a lot of time. Main complaints in early testing was of the gestures being sticky or not being recorded correctly. This could be the result of the overlay or it could be the lack of familiarity with the user. To combat this more gesture patterns could be provided, to ensure that it was the users at fault. An image was added demonstrating how the gestures are to be drawn. After this change feedback regarding sticky gestures reduced.

As gestures were the biggest selling feature of the app it was thought best to neglect more finishing features in the app to get them working correctly. This changed use cases and the finished product.

It was desired to cut down on repeated code by making one or two functions that call the server, but this couldn't be done to solve problems with loading calls. Some functions needed to be run after information had been received from the server but were running too early. To ensure they only ran after the server call they had to be written in the call to the server. This resulted in challenges with both view holders and list adapters.

Time management wasn't perfect and because of these challenges desired features could not be implemented. There was initially a concern about new users getting used to the app, but most testers replied that after a few minutes of recording they found the app very straight forward, albeit a bit daunting when starting off. A YouTube demonstration video could be used to help with this problem. It was also initially planned to use a numbering system when identifying players but the problems with gestures prevented the ability to do this. Instead geometric shapes were used. This solved the gesture problem as now most of the time they are recognised correctly as the shapes are more distinct. It was concerned that this system would frustrate users but again after a few minutes of recording the shapes became easier and easier to associate with a player and most users were satisfied with this method.

5.4 External API's

MPAndroidChart is a third-party library available on GitHub that provides charting capabilities for Android applications. There is no standard method of drawing charts in Android apps but there are many libraries available online that provide this functionality. During the development of the post-match report stage of the project testing was done with many different graphing libraries. Other libraries were far more

complicated than MPAndroidChart and had less functionality in terms of the graphs that could be drawn.

```
dependencies {
    compile 'com.android.volley:volley:1.0.0'
    compile 'com.android.support:design:26.1.0'
    compile 'com.android.support:cardview-v7:26.1.0'
    compile 'com.android.support:recyclerview-v7:26.1.0'
    compile 'com.github.clans:fab:1.6.2'
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.github.PhilJay:MPAndroidChart:v3.0.3'
```

Figure 22 MPAndroidChart inclusion in build gradle

The library can be added as a dependency in the gradle of the project. A chart is defined in the xml and declared by using find by id like any other android element. Points are added to the chart as special ArrayLists for example a BarEntry for a bar chart.

```
BarDataSet set1;
BarDataSet set2;
BarData data;
ArrayList<IBarDataSet> dataSets = new ArrayList<>();
ArrayList<BarEntry> entries1 = new ArrayList<>();
ArrayList<BarEntry> entries2 = new ArrayList<>();
ArrayList<String> labels = new ArrayList<>();
float groupSpace = 0.08f;
```

Figure 23 Declaring Bar Chart

It's as simple as that to create very dynamic and beautiful graphs using this library[20].

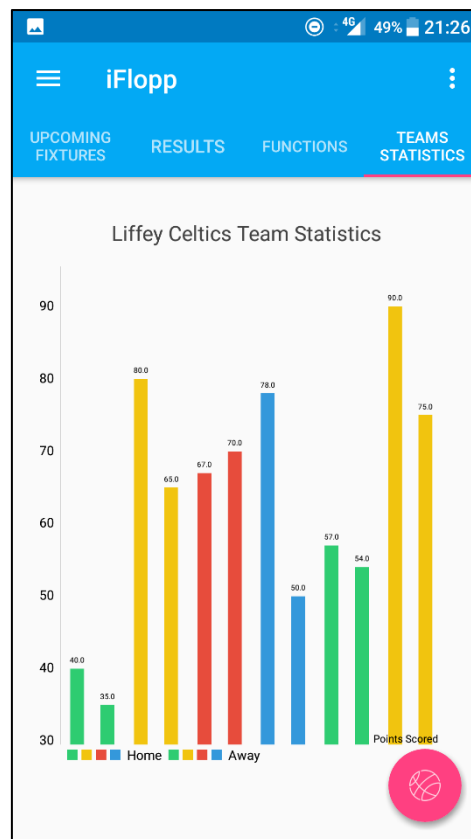


Figure 24 Sample Barchart

6 System Validation

6.1 Testing

The goal of testing was to evaluate if my application had improved upon the traditional input system. Did the gestures aid the user during recording and provide a less distracting and enjoyable method of stat recording? Did users find it easy to use or where they overwhelmed with the learning curve? Did the end report provide an easier to read output that could be used by a coach regardless of level? Where the objectives of this project met?

The first test was a short excerpt from a basketball match of a high standard. The chosen clip contained plenty of action to record. At this stage reporting was not finalised and the test was focused on the viability of the gestures. Each tester was given a run through of the application and asked to try record as many stats as they could. The testers all had varying experience when it came to basketball, for some they had never seen a match and others were of a very high standard. This test was not about accuracy but about the ease of using the gestures. Many of the testers struggled to record any statistics.

The problem was lack of recorded gestures. There were not enough patterns stored in the application to be very accurate. All testers then recorded 10 patterns each for the application. Making sure to draw the pattern in lots of unusual ways. This took a very

long time but when the testers tried to record stats again they had improved immensely. To fix the gestures after this test many more patterns were added to the application. It became apparent that certain shapes were a problem while other shapes were easily recognisable. From here the gestures for identifying players were changed from numbers 1 through 5 to distinct shapes. This improved player selection immensely, but certain types of actions had to be abandoned or drawn in a specific way. Turnover was impossible to record as the gesture overlay will not recognise the pattern at all. This is unusual as the gestures for swiping left and right and the gesture for recording 1 point all worked. To counteract this problem an artist designed an image for the stat input page that can be used as the background for the gesture overlay. It defines the best way to draw each gesture to ensure a good result. Overtime by spending more hours adding in gestures this may not be necessary, but this solution was needed to solve the problem before submission.

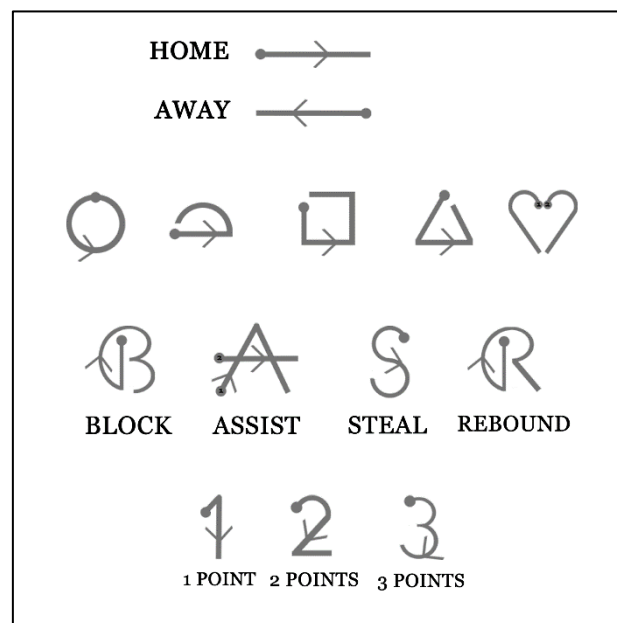


Figure 25 Image Defining the Gestures

After this solution was implemented testers were asked to watch a video of a full-length game. They were under instruction not to rewind or pause but to try record the correct score for the game. The testers this time around were all familiar with basketball in some form but not necessarily stat recording. Most of the testers gave back accurate results with just two unable to get the exact correct score. The testers filled out a questionnaire after the game. While most were happy with the app and found it fun to use and not very distracting, there were still complaints of the gestures being “sticky” or they would not work for a few seconds before working again. This is again a problem fixed by entering more gestures into the application. But for the most part the app was a success.

Lastly, the ultimate test was taking the app to the Basketball Ireland Colleges Intervarsity’s tournament to record live matches. A dedicated stat recorder used the application to record stats for some of DIT’s matchups. The resulting reports were then shown to the coach. He found the reports very insightful and used them throughout the weekend and even at half time to determine what players on the

opposition were playing the best. After the tournament there was a discussion between different members of the target audience and the author. Every member found the application very beneficial in terms of reporting and suggested more graphs they would like to see in the reports in future iterations. While the gestures did prove to be an annoyance most members agreed that the app achieved what it set out to do. After some initial pre-match warm up time they found they fell into a rhythm with the app. It did not distract from the overall experience of watching the game and the reporting although not extensive at this time it proved to be a real asset to a teams performance.

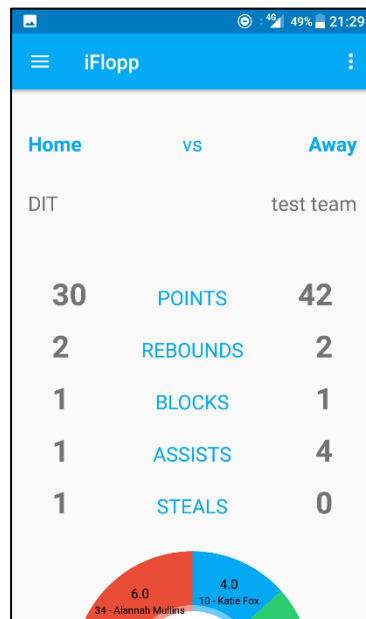


Figure 26 DIT Tournament Match Report

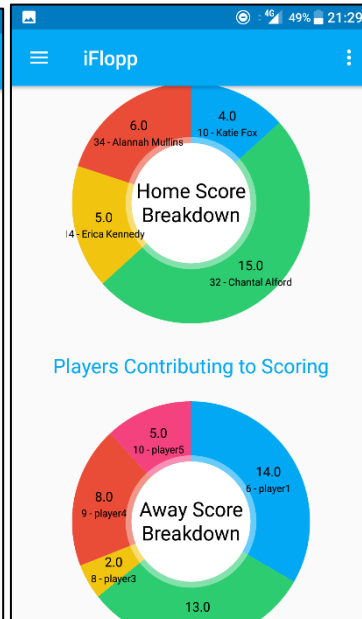


Figure 27 DIT Tournament Match Report

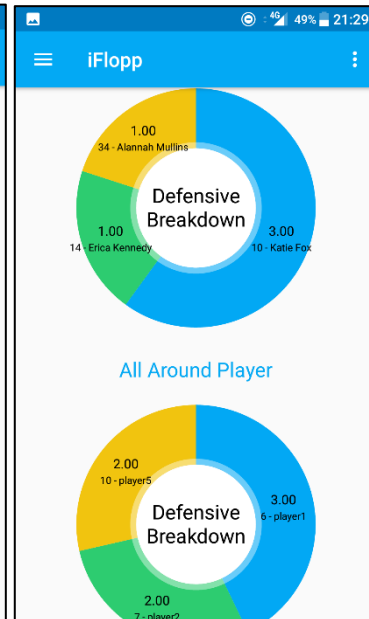


Figure 28 DIT Tournament Match Report

6.2 Demonstration

6.2.1 Creating a New Fixture

A user can start recording a match by first entering details of the fixture choose Create New Fixture from the main page FAB as seen in the next figure. They choose the two participating teams and decided who is home and away. The venue the match is being held in and the date of the fixture. Alert dialog boxes are used with buttons to achieve this effect.

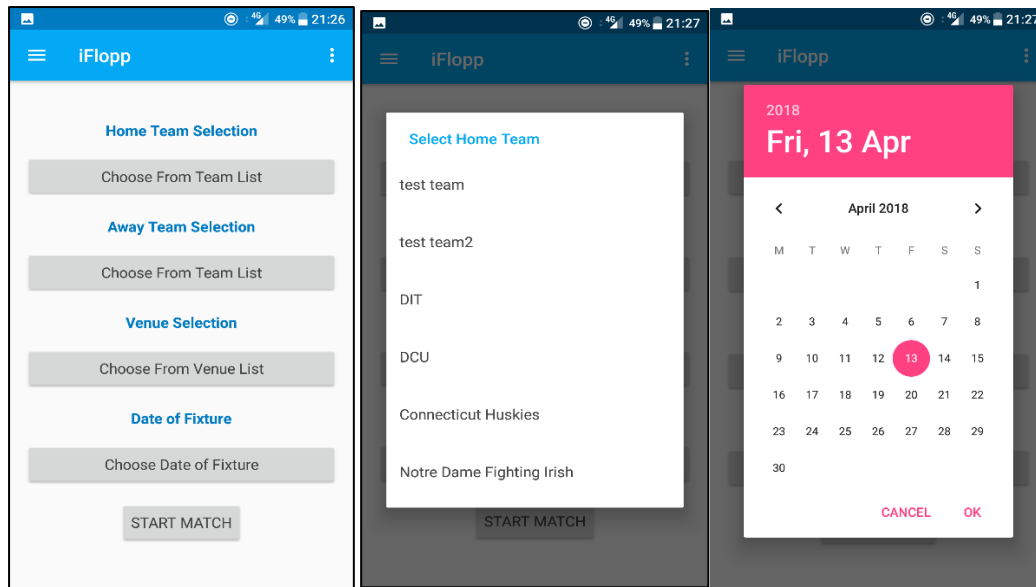


Figure 29 Creating a Fixture

6.2.2 Choosing a Fixture

Next the user can pick the fixture they've just entered into the system to start recording a match. This is the main page of the app. Pressing the FAB button gives the user 2 options to choose from. The second option Pick Recorded Fixture will bring the user to a list as seen on the far right. The user then chooses the game they wish to record stats for.

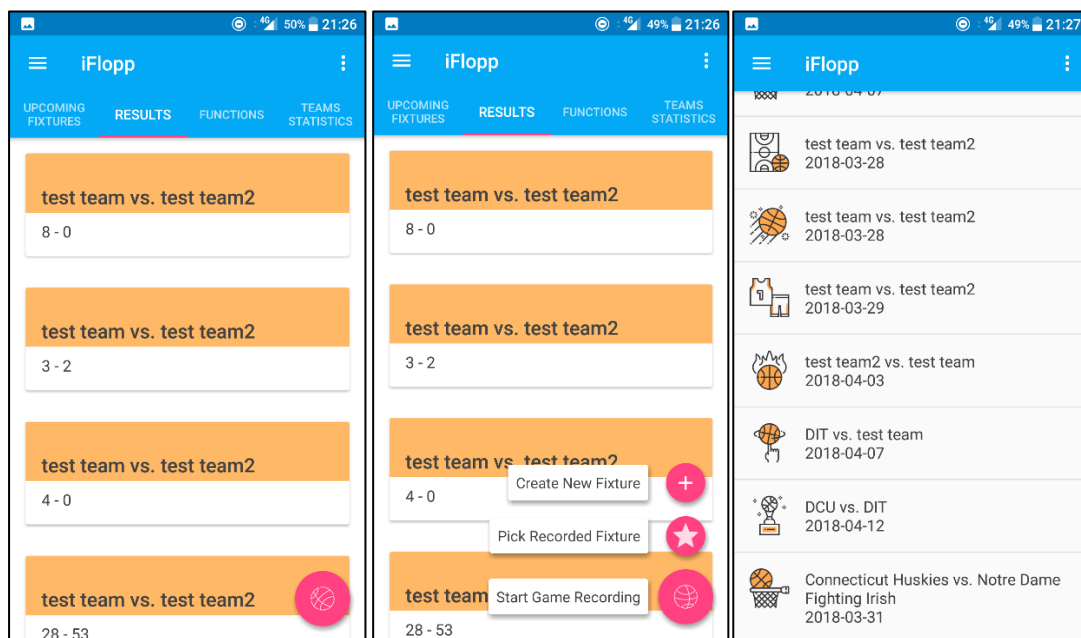
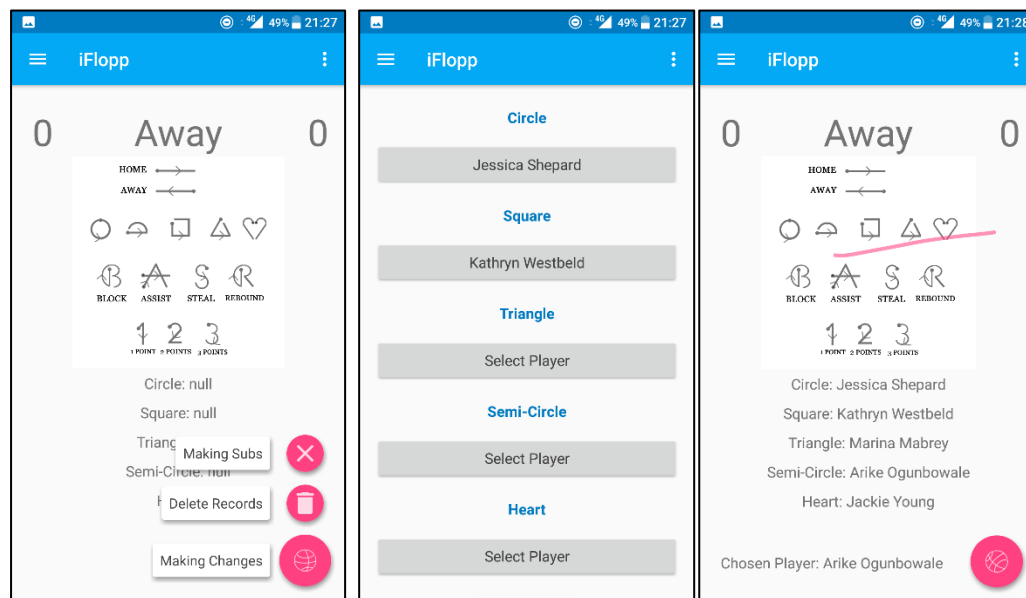


Figure 24 Choosing a Match to Record

6.2.3 Recording Statistics

In the next figure we can see that the stat recording page has been opened. Initially no players have been subbed into the game and gestures are left at null. By choosing

Making Subs in the new stat FAB button the user is given the option to sub in their players for the away team. Once all players have been assigned choose the make subs button and the players names will appear beside the name of the gesture that represents them. Switching players between gestures throughout the game does not affect the recording and subbing is carried out this way throughout the game. To sub in the home team the user must swipe right and repeat the same process. Once all players are subbed in the user can begin to record stats. The gestures can be drawn over the entire screen and are drawn in a pink colour when working correctly. First the user must select a player using the swipe left or right gesture to choose the correct team and then one of the five shapes to select a player. After a player is selected their name appears on the bottom of the screen to indicate they have been chosen. To record a stat for this player the user then draws a gesture that represents one of the types of actions. If something is entered incorrectly the user can open the FAB again and choose delete records. This will generate a list of every action that has been recorded for that game so far. To delete an incorrect record, the user taps on the specific record and a dialog box ensures they want to delete this record. When the game is over use the back button to exit back to the main menu.



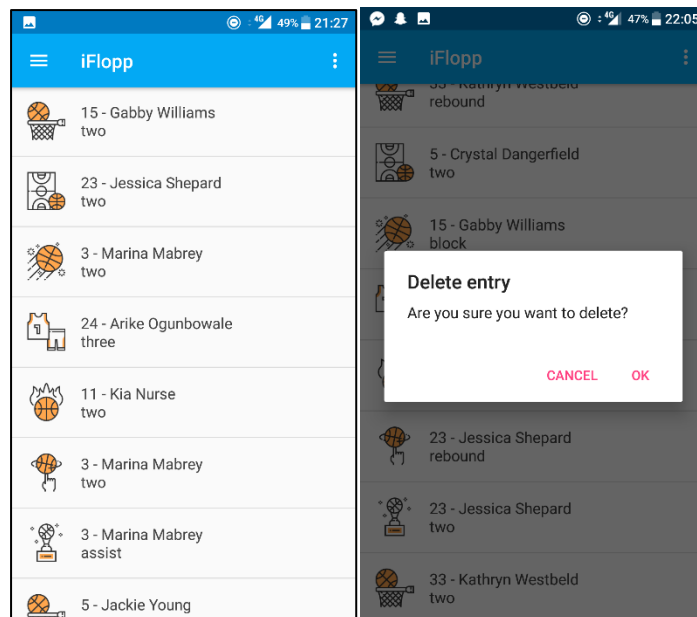


Figure 30 Demonstrating all Inserts

6.2.4 Viewing Match Reports

Lastly the user can view the generated reports from the games that have been recorded. Choose from the main page list, the game you wish to view as seen in the testing section there is a count of all stats recorded in the game for both teams. There are also two graphs for each team detailing what players are scoring the most and what players are doing the most off the ball work.

7 Project Plan

As mentioned before a lot of the struggles of this FYP resulted from over promising initially with my project plan. The biggest changes occurred when setbacks halted my ability to progress with the project. This put a lot of time pressure on an already unreasonable project management scheme. Each setback resulted in losing a piece of functionality that although not crucial to the main objective of the app, downgraded its complexity and the lost features would have helped with this issue.

If I was repeating the project, I would scale down the initial idea so that the time pressure on myself would be lessened. I would then be able to focus on the key features like I had done but not feel guilty that I was under performing. But I intend to continue building on top of this foundation that I have created. The response from the basketball community has been immense and I am excited to continue building this product as I can see the impact it has had. I will add functionality for users to create login systems and save personal matches that aren't available to everyone. I will

consider implementing splitting the workload over two phones but that will be a long-haul goal. Lastly functionality for inserting new teams and players is essential.

8 Conclusion

At the end of the project I am left with an app that accomplishes what it set out to do with some minor setbacks. The gesture input is functional and when it works well does become a viable method of input for recording. But the “sticky” gestures may continue being a problem as the only solution is to keep adding more patterns.

The most important lesson I have taken away from this experience is that over promising early on in a project is a recipe for disaster. I spent the entirety of the project underwhelmed with my progress and stressed from the constraints I had placed on myself. But despite this I enjoyed working on this project as basketball is such a big part of my life. It was a dream working on something and seeing it benefit your team mates and clubs.

9 Bibliography

- [1] S. Silverman, “Why Is the Game of Basketball So Popular?,” *LIVESTRONG.COM*. [Online]. Available: <https://www.livestrong.com/article/364098-why-is-the-game-of-basketball-so-popular/>. [Accessed: 13-Apr-2018].
- [2] “SFIA Sports, Fitness, and Leisure Activities Topline Participation Report Indicates Americans are Changing Their Activity Habits.” [Online]. Available: https://www.sfia.org/press/904_SFIA-Sports%2C-Fitness%2C-and-Leisure-Activities-Topline-Participation-Report-Indicates-Americans-are-Changing-Their-Activity-Habits. [Accessed: 13-Apr-2018].
- [3] D. Holloway and D. Holloway, “TV Ratings: NCAA Men’s Basketball Tournament Up Slightly for CBS,” *Variety*, 16-Mar-2018. .
- [4] O. release, “NBA on ABC viewership up 17 percent for 2017-18 season,” *NBA.com*. [Online]. Available: <http://www.nba.com/article/2018/04/10/nba-abc-viewership-upoffi>. [Accessed: 13-Apr-2018].
- [5] D. Kopf and D. Kopf, “Data analytics have made the NBA unrecognizable,” *Quartz*, 18-Oct-2017. .
- [6] L. Steinberg, “CHANGING THE GAME: The Rise of Sports Analytics,” *Forbes*. [Online]. Available: <https://www.forbes.com/sites/leighsteinberg/2015/08/18/changing-the-game-the-rise-of-sports-analytics/>. [Accessed: 13-Apr-2018].
- [7] Tifo Football, *What Is Moneyball?* .
- [8] “How Sports Technology and Analytics Are Changing Sports,” *Concordia University Texas*, 02-Dec-2015. [Online]. Available: <https://online.concordia.edu/sports-administration/sports-technology-and-analytics/>. [Accessed: 13-Apr-2018].
- [9] “Basketball Analytics | Basketball Player Tracking,” *STATS*. [Online]. Available: <https://www.stats.com/basketball/>. [Accessed: 13-Apr-2018].
- [10] “Basketball Ireland.” [Online]. Available: <http://www.basketballireland.ie/stats-tables/?comp=121822&sort=points>. [Accessed: 13-Apr-2018].
- [11] R. Inostroza, C. Rusu, S. Roncagliolo, C. Jimenez, and V. Rusu, “Usability Heuristics for Touchscreen-based Mobile Devices,” in *2012 Ninth International Conference on Information Technology - New Generations*, 2012, pp. 662–667.
- [12] “iTouchStats Basketball on the App Store,” *App Store*. [Online]. Available: <https://itunes.apple.com/us/app/itouchstats-basketball/id407696691?mt=8>. [Accessed: 22-Nov-2017].
- [13] S. Inc, *Basketball Stats Keeper*. Scoutlit Inc., 2016.
- [14] “Native, Web or Hybrid Apps? What’s The Difference?,” *MobiLoud*, 20-Feb-2018. [Online]. Available: <https://www.mobiloud.com/blog/native-web-or-hybrid-apps/>. [Accessed: 13-Apr-2018].
- [15] “Mobile OS market share 2017,” *Statista*. [Online]. Available: <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>. [Accessed: 25-Nov-2017].
- [16] “PYPL PopularitY of Programming Language index.” [Online]. Available: <http://pypl.github.io/PYPL.html>. [Accessed: 26-Nov-2017].
- [17] “SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems,” *DigitalOcean*. [Online]. Available:

- <https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems>. [Accessed: 27-Nov-2017].
- [18] “Why Agile Is The Best Alternate Methodology To Waterfall? – Blog | Asahi Technologies.” .
- [19] “Understanding the Agile Software Development Lifecycle and Process Workflow,” *Smartsheet*, 18-Aug-2016. [Online]. Available: <https://www.smartsheet.com/understanding-agile-software-development-lifecycle-and-process-workflow>. [Accessed: 23-Nov-2017].
- [20] P. Jahoda, *MPAndroidChart: A powerful & Android chart view / graph view library, supporting line- bar- pie- radar- bubble- and candlestick charts as well as scaling, dragging and animations*. 2018.

10 Appendix

10.1 Usability Heuristics for Touchscreen-based Mobile Devices

- 1) Visibility of system status: The device should keep the user informed about all the processes and state changes through the use of a specific kind of feedback, in a reasonable time.
- 2) Match between system and the real world: The device should speak the users' language with words, phrases and concepts familiar to the user, instead of system-oriented concepts and/or technicalities. The device should follow the real world conventions and physical laws, displaying the information in a logical and natural order.
- 3) User control and freedom: The device should allow the user to undo and redo his actions, and it should provide “emergency exits” to leave the unwanted state. These options should be clearly pointed, preferably through a physical button or similar; the user shouldn't be forced to pass through an extended dialogue.
- 4) Consistency and standards: The device should follow the established conventions, on condition that the user should be able to do things in a familiar, standard and consistent way.
- 5) Error prevention: The device should have a careful graphic user interface and physical user interface design, in order to prevent errors. The non-available functionalities should be hidden or disabled and the user should be able to get additional information about all available functionality. Users should be warned when errors are likely to occur.
- 6) Minimize the user's memory load: The device should minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.
- 7) Customization and shortcuts: The device should provide basic configuration options and should give expert users access to advanced configuration options. The device should provide shortcuts to the most frequent tasks and should allow their customization and/or definition.
- 8) Aesthetic and minimalist design: The device should avoid displaying irrelevant or rarely needed information. Each extra information unit reduces the system performance.
- 9) Help users recognize, diagnose, and recover from errors: Error messages in the device should be expressed in plain language (no codes), precisely indicating the problem, and constructively suggesting a solution.
- 10) Help and documentation: The device should provide easy-to-find documentation and help, centred on the user's current task. A list of concrete (and not too large) steps to carry out should be provided.

11) Physical interaction and ergonomics: The device should provide physical buttons or similar user interface elements for main functionalities. Elements should be placed in a recognizable position. The device dimensions, shape, and user interface elements in general, should fit the natural posture of the hand.[11]