**ULSTER UNIVERSITY**

**SCHOOL OF COMPUTING AND INFORMATION ENGINEERING**

**MSc Professional Software Development 2015/16**

# COM814: Project

# Dissertation

**Student Name: Christopher Patton**

**Supervisor: Janet Allison**
**Second Marker: Dr. Adrian Moore**

**Submission Date: 01 September, 2016**

## Plagiarism Statement

"*I declare that this is all my own work and that any material I have referred to has been accurately referenced. I have read the University's policy on plagiarism and understand the definition of plagiarism. If it is shown that material has been plagiarised, or I have otherwise attempted to obtain an unfair advantage for myself or others, I understand that I may face sanctions in accordance with the policies and procedures of the University. A mark of zero may be awarded and the reason for that mark will be recorded on my file*."

Signed: Christopher Patton


## Acknowledgements

I would like to extend my gratitude to everyone who has contributed to the successful completion of this project. In particular I am thankful for the support of Dr. Moira McAlister and Dr. Zhonglin He, whose guidance throughout the project's initiation and subsequent *Analysis and Design* phase were invaluable, as was the feedback and support received from my supervisor, Janet Allison, and second marker, Dr. Adrian Moore, throughout the remainder of the project. Furthermore, I would like to thank my focus group of football referees for contributing with their ideas during the project's *Analysis and Design* phase and for participating in the testing of my software. And last, but by no means least, I would like to show my appreciation for my family for their continual support and encouragement over the past year. Thank you all.

# CONTENTS

LIST OF FIGURES

LIST OF TABLES

# Abstract

A technological revolution has resulted in ubiquitous computing throughout 21st century societies. The widespread adoption of mobile computing devices such as tablets and smartphones has recently paved the way for unprecedented developments in the field of wearable technology. While these modern technologies have permeated professional sporting industries, as evidenced by goal line technology utilising smartwatches with the aim of improving the decision-making of professional association football referees, there remains a distinct lack of cost-effective wearable game management solutions available to football referees operating at the grassroots level.

The following dissertation chronicles the process of developing the smartwatch application *RefWatch: Soccer Edition*, which provides a smart game management solution for football referees by facilitating accurate timekeeping, recording game events such as goals scored, disciplinary cards issued, and player substitutions, and generating accurate post-match reports. The application's design and functions were thoroughly researched and planned, partially through dialogue with a focus group comprising the application's target demographic - namely football referees officiating at the grassroots level - in order to produce the most intuitive and efficient system possible. The system's expedient adoption was facilitated, furthermore, by limiting the requisite learning timeframe for its users, while care was taken to avoid any possibility of interference with a user's performance of their officiating responsibilities.

The dissertation presents an examination of a grassroots association football referee's role and an investigation of the current market for referee products to provide the reader with a clear understanding of the project's problem area and, moreover, the solution provided by the *RefWatch: Soccer Edition* smartwatch application. The author's comprehensive analysis in conjunction with detailed discourse with stakeholders has revealed the market's salients needs and, furthermore, facilitated the creation of a design concept incorporating the key functional requirements which were subsequently implemented in the application's prototype. The analysis, design, implementation and testing processes are detailed herein, alongside recommendations for future developments of the application.



**Keywords**: Smartwatch, Wearable Technology, Mobile Application, Android, Football, Soccer, Referee.

# 1: Introduction

## 1.1: Football and Technology

The sport of association football, or 'soccer' as it is sometimes known internationally, enjoys global levels of support and participation which are unsurpassed in the sporting arena. In its most recent *Big Count* study FIFA, football's global governing body, revealed a marked increase in active player involvement in the sport from 242 million to 265 million between the years 2000 and 2006 (FIFA.com, 2007). The continual rise of the game's global appeal in recent times was further highlighted by the 2015 FIFA Women's World Cup Final, which set new record-high television audiences for football in the United States and Japan (FIFA.com, 2015).

The growth in football's global fan base has been underpinned by a technological revolution in telecommunications and information availability. This has precipitated hitherto imperceptible levels of access to the game for the sport's fans, with everything from live television broadcasts of matches from across the globe to up-to-the-minute news reports on the inner machinations of their favourite professional football clubs, managerial sackings or the latest debate surrounding erroneous refereeing decisions driving the sport's spiraling popularity. Furthermore, as the professional game has gradually been transformed into a modern industry in its own right, technological innovations such as the development of modern stadia offering free wi-fi connections to spectators, for example, highlights technology's continuing influence on the sport's evolution on and off the field of play.

Football's law-makers have been somewhat reluctant, however, to emulate the levels of technological adoption witnessed in other popular sports such as rugby union; in 2001 the use of a television match official (TMO) was introduced during televised top-level club and international games to aid the referee's decision-making through video reviews of in-game incidents (FindRugbyNow.com, 2012). The slow pace of technological adoption in football can largely be attributed to widely-held concerns that technology, such as video officiating, has the potential to interfere with the traditional flow and entertain levels of the game. Nevertheless, technological innovations affecting how the game is officiated on the field have gradually been implemented at football's highest professional levels. The 2014 FIFA World Cup, for instance, provided the venue for football's first use of goal line technology

by incorporating a system which sends an alert to the referee via a custom-built smartwatch, notifying them that a goal has been scored when the ball has crossed the goal line as determined by a camera-based system (Saunders, 2014).

Technology's place in football has become increasingly important in assisting the sport's officials. While developments such as goal line technology may be beneficial to referees operating at football's highest levels, however, referees officiating at the grassroots level are often faced with very different problems to those encountered by their professional counterparts and a lack of effective technological solutions.

## 1.2: Problem Statement

Despite unprecedented innovations in the field of wearable technology and increasing technological implementation throughout various aspects of the professional football industry, there is an absence of cost-effective wearable game management solutions available to football referees operating at the grassroots level.

## 1.3: Aim

The aim of this project is to develop a smartwatch application as a solution to the problems experienced by grassroots football referees, which are discussed in detail in the subsequent chapter. The application will facilitate the key responsibilities of referees such as effective timekeeping, recording of match events such as goals scored, disciplinary cards issued and player substitutions, and efficient post-match administration. The app will be designed, moreover, with minimal disruption to the referee's workflow in mind, while presenting an effective and efficient problem-solving solution.

## 1.4: Project Objectives

A number of objectives were defined which, when fulfilled, would contribute to the achievement of the project's overarching aim.

- Investigate the role of a football referee and advancements in wearable technology, and establish an understanding of how a smartwatch application can be incorporated seamlessly into the referee's workflow while accurately fulfilling their responsibilities.

- Identify existing products and software applications available to football referees, analyze their strengths and weaknesses and identify features which could be enhanced and incorporated in a smartwatch application.

- Create a questionnaire and distribute it to a focus group of football referees operating at the grassroots level of the sport. Feedback received from the questionnaire, in conjunction with the analysis of existing products available to referees, will facilitate the crystallization of functional and non-functional requirements of the application's key stakeholders.

- Identify an operating system which will serve as the platform upon which to design, develop and implement the smartwatch application's required software components.

- Develop an attractive user interface and an intuitive architecture for the application, which facilitates all of the identified system requirements in an efficient and robust manner.

- Present a working application prototype to a focus group for testing and evaluation purposes.

- Evaluate the software prior to release to ensure that the overarching project aim has been achieved and, furthermore, to provide recommendations for future developments to further enhance the product.

## 1.5: Dissertation Outline

The following outline provides a brief overview of each of the dissertation's chapters:

- **Chapter 2: Background Review**: Presented in chapter 2 are descriptions of the role and responsibilities of, and the equipment typically utilised by, grassroots football referees. The project's key stakeholders are identified alongside a brief history of wearable technology and examples of key products released to the market throughout the development curve of the smartwatch. The concept of integrating a smartwatch application with a referee's workflow is illustrated in rich picture diagram format, while existing products available to referees such as dedicated 'referee watches' and mobile applications are described and analyzed.

- **Chapter 3: Requirements Analysis**: The functional and non-functional requirements of the software application's end-users, which were identified through discourse with the project's key stakeholders, background research, and problem analysis, are presented in chapter 3. The proposed application's key functionality is detailed, furthermore, and user requirements presented in the format of 'user stories'.

- **Chapter 4: Professional Issues**: The fourth chapter discusses a number of professional issues associated with the project. A business case for the proposed application is presented, alongside a number of health and safety and technical risks and the preventive measures which were employed to ensure the project's successful completion. Finally, the project's categorization by Ulster University's Ethics Committee is detailed and discussed.

- **Chapter 5 - Design**: The design of the application's system architecture, database and Graphical User Interface are outlined alongside a discussion of key design considerations such as the unique challenges presented by the smartwatch form factor. The merits of the software development methodology adopted for the duration of the project is discussed, while an architectural overview of the application's chosen platform is presented.

- **Chapter 6 - Implementation**: A review of the implementation process, chapter 6 provides an insight to the challenges associated with the identification of software components and their successful deployment. The key tasks involved during each cycle of the iterative implementation process is detailed, with an emphasis placed upon the creation of the solution application's core functionality.

- **Chapter 7 - Testing & Evaluation**: Software testing, a vital element of software development, is discussed in conjunction with the in-house and focus group testing methodologies employed in advance of the solution application's deployment. The testing results and technical specifications of the testing device are included in associated Appendices.

- **Chapter 8 - Conclusion and Recommendations**: The final chapter provides an overview of the entire project, discussing the requirements which have been successfully implemented and presenting suggestions for their future enhancement and, furthermore, the development of those which have been left unfulfilled.

# 2: Background Review

## 2.1: Role and Responsibilities of the Referee

At its core the main role of the football referee, regardless of the level at which they officiate, is to administer the 'Laws of the Game'. The responsibilities of a football match's lead official extend far beyond this, however, as the referee is placed in charge of the entire event including a number of duties to be fulfilled before, during and after the game.

### 2.1.1: Pre-game Responsibilities

The referee will normally arrive at least fifteen minutes before a football match's scheduled kick-off time in order to perform a number of appointed duties. They must inspect the playing surface, for example, and confirm that the pitch is free of any hazardous materials such as broken glass or rocks to meet the required standards and ensure player safety. Furthermore, the referee must confirm that the players are appropriately attired and have been divested of any jewelry or similarly dangerous articles; the referee has the authority to delay the game's kick-off until all players conform with these rules. The number of players per team must be approved and any necessary discussion with players and coaches completed before a coin toss determines which team will begin the first period of play in possession of the ball. In the event of hazardous weather conditions such as lightning storms, frozen or waterlogged pitches or extreme heat the referee may choose to abandon the game before kick-off or during game play. (Football-Bible.com, 2015).

### 2.1.2: Intra-game Responsibilities

The referee's main concern during game play is administering FIFA's 'Laws of the Game'. There are seventeen laws in total concerning everything from the dimensions of the field of play to penalising player indiscipline during a match. The referee upholds the rules and punishes any infringements by sanctioning yellow or red cards to the offending players. Any disciplinary action is recorded by the referee and subsequently forms the basis of investigations into any misconduct. As the game's timekeeper the referee also ensures that the game lasts for the pre-determined duration, while any events which result in a stoppage in play, such as player substitutions or injuries, are recorded by the referee and additional time - known as injury- or stoppage-time - is added to the corresponding half's duration. When a player is injured, moreover, the referee suspends play and ensures that the player receives adequate medical treatment before play is resumed. The referee must

ensure that, while the rules are fairly enforced, the competitiveness and flow of the match are unaffected and that, where possible, it is completed with minimal interruptions. Furthermore, the validity of goals scored during the game and the final outcome are arbitrated by the referee, with the aid of their assistants (where available) when adjudicating controversial incidents (Football-Bible.com, 2015).

### 2.1.3: Post-game Responsibilities

Following a game's conclusion the referee will meet with their assistants to verify the game's scoreline, confirm which players received yellow or red cards, and determine any additional information to be included in the referee's post-match report. Often this report is used not only to confirm the result of the match, but also as the basis for paying the officials so the accuracy of the record is important. The submission process for game reports varies, with some managing associations requiring a paper copy to be mailed while others have dedicated online reporting systems. The referee may be required to follow-up on intra-game incidents such as player injuries. In youth football leagues in particular, where determining the extent of a player's injury may initially prove difficult, the referee's duty of care dictates that they ensure the managing association follows the correct protocol by tracking the player's condition (Williams, 2016).

## 2.2: Problems at the Grassroots Level

While football's top officials may have to perform under the scrutiny of stadiums packed with spectators and in the glare of media cameras, grassroots referees face wide-ranging pressures. Whereas professional referees administer the game's rules in largely uniform scenarios, modifications to the application of the 'Laws of the Game', such as the duration of the periods of play or the number of permitted substitutions per team, for example, are authorised by FIFA when referees are officiating in the under-16 and over-35 age groups. This means that the matches officiated by grassroots referees can vary greatly in nature (FIFA.com, 2015). Some of the variables encountered by referees operating at football's lower levels include:

- Age group of the participants, ranging from small children to older adults.
- The number of participants, ranging from 5-a-side to 11 players per team.
- Level of competition, ranging from recreational to competitive.
- Playing surface and venue, ranging from grass to artificial turf to indoor gyms.
- Duration of the game, corresponding to age group and level of competition.

Unlike their professional counterparts, moreover, grassroots referees typically perform their role in addition to fulfilling the obligations of their full-time occupation. Consequently many referees officiating at the grassroots level are faced with time constraints which are only exacerbated, for instance, by the often inefficient process of completing their post-match administrative duties.

## 2.3: Referee Equipment

At the top levels of football such as the English Premier League, for example, the officials are highly experienced and perform their duties in a professional capacity. In accordance with the high-profile status of the games they officiate and expectation of correspondingly high performance levels, professional referees typically carry equipment which is well designed and subjected to rigorous testing before use. Such equipment is typically inaccessible to their grassroots counterparts, however, due to prohibitive price tags or the lack of public distribution.

The equipment carried by English Premier League referees includes:

- Radio receiver and headset: facilitates communication between the referee and their assistants.
- Flags receiver: a system which automatically notifies the referee to an assistant's offside call through vibration or audible alerts.
- Primary timekeeping watch: facilitates the referee's role as principal timekeeper.
- Whistle and accompanying lanyard: used by the referee to signal the stoppage and resumption of gameplay.
- Goal decision watch: sends a signal to the referee when the ball has crossed the goal line.
- Pen or pencil and disciplinary cards: issued to players by the referee when they infringe upon the game's rules, with the names of offenders and nature of the infringement noted and reported by the referee after the game (In The Opinion Of The Referee, 2014).

In contrast, the equipment utilised by officials at the grassroots level varies from referee to referee although the principal responsibilities of timekeeping and scorekeeping necessitate that all officials use equipment such as a watch and a medium to record the match's scoreline, such as a score card and pen or pencil. Although their access to state-of-the-art referee technology is somewhat restricted, however, grassroots referees have shown a

desire to adopt accessible technologies such as GPS-enabled fitness trackers (See *Appendix Two*).

## 2.4: Project Stakeholders

Any person, group, or organization that is actively involved in a software development project, is affected by the software system's outcome, or can influence its outcome, can be referred to as a 'project stakeholder'. In this instance the project's main stakeholders were identified as:

- amateur referees
- FIFA world governing body
- local football associations
- football clubs
- football players
- football fans and spectators
- football officials
- referee mobile app developers
- referee watch manufacturers

The stakeholders, their concerns, and developer considerations are presented in a rich picture diagram (See *Appendix One*).

## 2.5: 'Wearables' and the "Technology Revolution"

In recent decades human society has experienced what Antón, Silberglitt and Schneider have labelled a multidisciplinary "technology revolution". Increasing human connectivity with information technology in the 21st century has, for instance, accelerated an ongoing process of globalisation, enabled widespread dissemination of knowledge and created opportunities for economic prosperity and improved quality of life (Antón, Silberglitt and Schneider, 2001). The establishment of a networked world has fundamentally changed the way that humans connect with one another and their surrounding environments; the number of devices connected to the internet has risen from 500 million to approximately 14 billion in the past decade (Cisco, 2011; Cisco, 2014). Computing devices have become more affordable and accessible and, consequently, users have derived greater value from them while developing higher expectations of what these products should offer them in an

increasingly cloud-based ecosystem. The aforementioned confluence of factors has precipitated a mass consumer market for mobile and, subsequently, wearable devices.

As technology devices such as the smartphone have been adopted in ever greater numbers they become more integral to the daily life of their users. The use of personal computing devices such as smartphones, tablets and laptops can become inconvenient or intrusive, however, as they distract the user from their real-life activities in their immediate environment once they have become immersed in the virtual user experience. In the context of a football referee's role and responsibilities such distractions would be detrimental to performance levels. Whereas more complex computational tasks or interactions with electronic devices, such as word processing or watching movies, may require a larger device such as a laptop, tablet or smartphone, in environments where the user's tasks require that user interaction time must be kept to a minimum using a wearable device can be advantageous.

## 2.6: The Smartwatch: Emergence & Landmarks

Examples of sophisticated wearable technology with processors and small amounts of on-board data storage, such as the Pulsar *NL C01* digital watch, were first released to the consumer marketplace in the early 1980s. Sectors such as medicine, the military, space exploration, and academia, furthermore, have found numerous applications for wearable computers. Yet the smartwatch as a mass-produced consumer product with powerful on-board processing and storage capabilities is a more recent development. The Samsung *S9110 Watch Phone*, released in 2009, introduced many of the features made popular by smartphones to a wearable device, including Bluetooth LE, a touch screen, improved battery life, and the ability to connect to other internet-connected devices. Further developments in the wearables field included the first use of a modified version of Google's *Android* operating system in a wrist watch-sized device and the introduction in 2012 of small, wearable products which combined fitness analytics and music playback.

The development of fitness wearables has proven to be a lucrative sector of the wearable technology market as sporting companies such as Nike compete with tech giants such as Microsoft to produce GPS trackers and exercise monitors; global sales in 2015 were reported at 25 million units, an increase of 11.5 million units over 2014's sales figures. In comparison, global smartwatch sales stood at 4 million in 2014 while 26 million sales were

forecast for 2015 (Statista, 2016). Many of the largest technology manufacturers have developed and released their own variations of the smartwatch concept since 2013 as they compete to exploit the largely untapped market potential, a development predicted by Acer's S.T. Liew who envisaged an intensified focus on the wearable market in a 2013 interview, "I think that every consumer company should be looking at wearable. Wearable isn't new... it just hasn't exploded in the way that it should. But the opportunity's for billions of dollars' worth of industry" (Hall, 2013).

The list of high-profile companies engaged in smartwatch development includes Acer, Apple, BlackBerry, Google, LG, Microsoft, Samsung, Sony and Toshiba, resulting in several important smartwatch iterations hitting the market since 2013 (Mims, 2013). The *Galaxy Gear*, for instance, was the first of six smartwatch products released by Samsung between 2013 and the final quarter of 2015, while the crowd-funded *Pebble Smartwatch* created by Pebble Technology Corporation became the first *Android*- and *iOS*-compatible smartwatch. Originally running Google's *Android Wear* operating system (OS) on their wearables, Samsung has recently released smartwatch models which run their own *Tizen* OS with the capability to process cellular data such as phone calls independently from a smartphone through the use of an onboard Sim card slot. Google, meanwhile, has focused on collaboration with other manufacturers who agreed to adopt the former's smartwatch OS on their devices. This has resulted in the release of flagship *Android Wear* products such as the Motorola *Moto 360*, Asus *ZenWatch*, LG *G Watch* and Sony *SmartWatch 3*.

The most recent smartwatch models possess similar technical specifications, with the current standards being 512MB RAM with 4GB of internal storage, dual- or quad-core processors, LED touch screens and Bluetooth LE, Wi-Fi and 3G connectivity. The Sony *SmartWatch 3* was the first smartwatch to provide a built-in GPS tracking unit popularising its use as a sports watch and fitness tracking device without the requirement for a paired smartphone; this feature has since been matched by the Samsung *Gear S,* released in November 2015. The recent forays of traditional watch makers such as Tag Heuer into smartwatch development highlights a gradual diversification of the smartwatch market into two broad categories: fashion and fitness. While fashion-oriented smartwatch models are being developed with a round display intended to resemble traditional timepieces, and have adopted customisable features such as interchangeable straps and longer battery life as the core focus of their design, other models are marketed as fitness devices which utilise larger rectangular screens, numerous sensors and GPS. The arrival of Apple's

*Apple Watch* in early-2015 attracted mass public attention to the wearable technology market, moreover, which seems certain to continually evolve and expand alongside the form and functionality of the devices themselves (Marshall, 2015).

## 2.7: Referee Workflow & Smartwatch Integration

A smart watch can support user input through simple touch menus, voice commands and gesture control which, as Mishra asserts, should reduce the "distractive and constraining effect on the user" (Mishra, 2015). As a lightweight and minimally intrusive device, therefore, a smartwatch has the potential to host an application with the tools required by a football referee while allowing the user to maintain their focus on their immediate physical environment and activity.

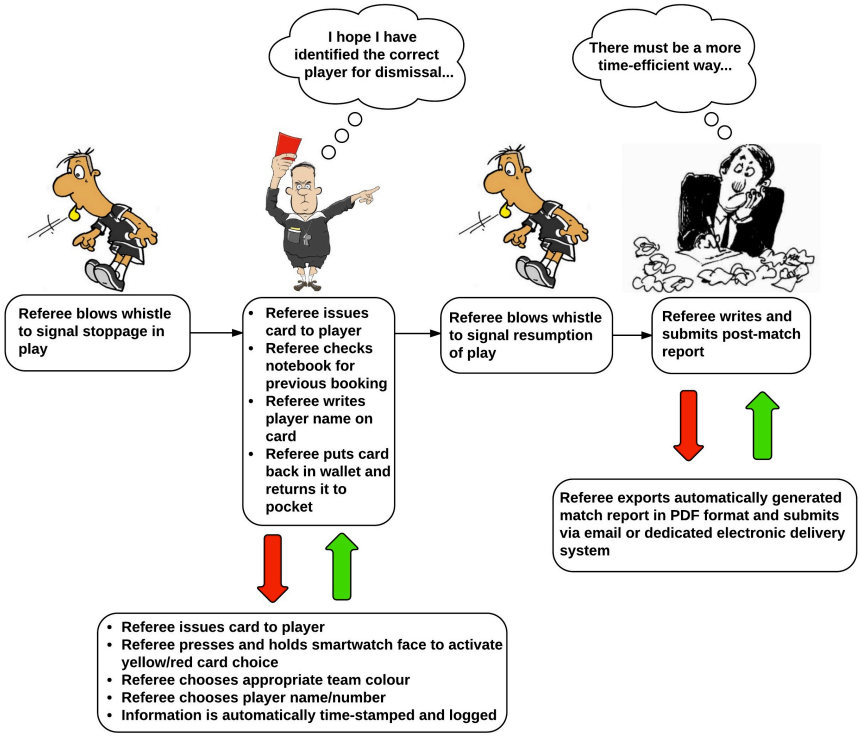



Figure 1: Rich picture depicting the role of a minimally invasive smartwatch application solution in the referee's traditional workflow.

The functionality provided by the modern smartwatch has the potential to facilitate not only a dependable timekeeping alternative for football referees, but a comprehensive game management system capable of tracking intra-game events and processing match reports to improve the accuracy of information and reduce the referee's administrative workload. Furthermore, the smartwatch has the potential to integrate seamlessly with the referee's traditional workflow while eliminating errors, as illustrated by the rich picture diagram in Figure 1.

## 2.8: Comparative Analysis of Existing Products

### 2.8.1: Referee Watches

**SPINTSO PDA**

**Device Features**

The *PDA* (Personal Data Assistant) from Swedish company Spintso (Sport Intelligent Solutions) provides multi-functional timekeeping, local sound recording for up to 30 seconds, and simplified administration through exportable match reports while Bluetooth can facilitate communication between multiple Spintso units. The device's features can be navigated using a keypad positioned above the screen whose large size accommodates timers and a scoreboard indicating the number of goals scored by each team. The stopwatch timer changes colour when the pre-selected half duration has expired. A vibration alert reminds the referee when the timer has been paused due to stoppages in gameplay.

**Device Drawbacks**

The device's price of over £200 is prohibitive. It's size and shape can make the device cumbersome to accommodate and requires the use of an accompanying wrist strap to attach it to the inner forearm. The ability to discretely glance at the screen is curtailed as a result.

**SPINTSO Referee Watch Pro**

**Watch Features**

Whereas the Spintso *PDA* attempts to provide an all-encompassing game management solution, the *Referee Watch Pro* focuses on the referee's timekeeping duties. With the option to select the number of periods in the match (0-9), number of minutes per period (1-99) and the number of minutes per break between periods (0-99), the watch is easily customizable to the referee's specifications. The watch has four timers which run simultaneously during a game: one counting up (always rolling), one counting down (stops when the ref stops the watch), one to count additional time (starts when the ref stops the watch) and one to measure time during the interval between periods. A vibration alarm provides an alert every ten seconds when the referee forgets to restart the main timer following a stoppage in play, or when regulation match time or the interval break is coming to an end.

### Watch Drawbacks

The watch is relatively costly with a RRP of $150, and according to online reviews the build quality undermines the well-designed timer features. Once the timing of the final period has elapsed, furthermore, all of the timers on the watch stop which will not facilitate the additional time which is often required at the end of a game after the pre-calculated stoppage time has been fulfilled.

## Casio RFT100

### Watch Features

The Casio *RFT100* offers six modes of operation: time, stopwatch, timer, alternative time zone, world time, and alarm. A pre-set countdown timer is available in 45, 40, 35, 25, 20, 12, and 10 minute increments allowing the referee to select the desired duration of each half of the match. A vibration alarm, which lasts for ten successive 1-second vibrations, can be used to signal the end of each half.

### Watch Drawbacks

Although the timer can be stopped and restarted to account for stoppages in play, the watch does not offer the capability to calculate the stoppage time to be added at the end of the half and a backup watch would be required to fulfill this role.


Figure 2: Referee Watches (left to right): Spintso PDA, Spintso Referee Watch Pro and Casio RFT100.

### 2.8.2: Referee Mobile Applications

**Ref Wallet**


**App Features**

Available for Apple's *iOS* platform, *Ref Wallet* offers a free option and an option with additional functionality for the price of $4.99. The app provides scorecard and scheduling features which enable the referee to record goals scored, yellow and red cards issued, and add their past and upcoming games to the schedule. Additional features include a dual-capability timer to track both regulation and injury time, and the option to export match reports in PDF format in the paid version.

**Design Drawbacks**

The app has many worthwhile features related to a referee's administrative responsibilities such as game scheduling, payments, and the ability to export detailed match reports. The user interface does not feel intuitive for game management purposes, however, with too many small buttons cluttering the screen which can be pressed erroneously in the fast-paced environment of a football game. The app is exclusively available for Apple's iPhone mobile phone devices, which has undermined its adoption as a popular solution for referees.

**Soccer Referee - Shingo 2.5**

**App Features**

Exclusively available for *Android* mobile phones, this app enables referees or coaches with the ability to edit team names, team colours, player names and additional information such as birth dates and player positions. During a game, the app can record game events such as goals scored, penalty kicks, substitutions, offsides, fouls, and yellow and red cards

issued. A stopwatch and game log, which can be exported as a match report via email, are also included.

**Design Drawbacks**

The app is marketed to appeal to coaches and spectators in addition to match officials. Consequently it overlooks many elements of the referee's role, such as an inability to stop and look at a screen for a prolonged period of time during gameplay. With so many features included navigation can be cumbersome and time consuming. The app is exclusively available for download on *Android* mobile phones, which would prove impractical for referees and limits the app's market potential. Many users have reported various bugs in their app reviews on the Google *Play Store*.

# 3: Requirements Analysis

## 3.1: Introduction

The application's requirements have been identified and defined through discourse with key stakeholders through the medium of a study group questionnaire and further informed through research and problem analysis. By identifying the functional and non-functional requirements of the system in this manner, the needs of the user will be reflected in its final design and implementation (Sommerville, 2011).

## 3.2: Focus Group Analysis

Based on the findings of the comparative analysis of existing solutions currently available in the marketplace (see *Chapter 2.8*), an online research questionnaire containing a total of 12 questions was created and distributed to the project's key stakeholders - football referees officiating at the grassroots levels of the sport (See *Appendix Two*). The questionnaire focus group comprised participants who had been sourced in two popular online discussion boards used exclusively by football referees. The study's participants volunteered personal data related to their football officiating activities and presented their views and preferences on a number of topics relevant to understanding the project's problem and formulating a solution including:

- The duration and level of the participants' refereeing experience.
- The equipment currently utilised by the participants in their role as a referee.
- The number of participants currently in ownership of a smartwatch.
- The willingness of participants to adopt a smartwatch application for refereeing, and the features they would like to see implemented in such an application.
- The operating system and smartwatch screen format favoured by participants.

The experience level of the focus group's participants ranged from less than 1 year to 20 years, although at least half of the referees had 6 or more years of experience. A majority of the referees had officiated at the lowest grassroots level, overseeing youth and adult recreational games, while a sizable sample of participants had refereed games at more competitive levels such as university or adult amateur competitions; 1 respondent had officiated professionally. The questionnaire respondents therefore comprised the demographic targeted by the solution application, and the data received in this study was vital in informing the software's requirements, design and implementation. The list of

functional requirements presented to the focus group, while not exhaustive, comprised the most applicable features found in existing solutions in addition to others determined through research. Furthermore, the respondents identified additional functional and non-functional requirements which are discussed further in this paper (See *System Requirements*).

An analysis of the focus group's key findings, including visual representations of the collected data, is presented in Appendix Two. Further engagement with the focus group, which took the form of group testing of a software prototype, was helpful in identifying and minimizing design, functionality and stability issues ahead of the application's deployment. The project's testing and evaluation processes are documented in Chapter 7.

## 3.3: System Requirements

### 3.3.1: Functional Requirements

1. **Timekeeping**: The application will require a dual timer, with the ability to track elapsed time in each period of gameplay in a football match and calculate and add the additional time accrued due to stoppages in play.

2. **Scorekeeping**: The application will require a built-in scoreboard which will allow a referee to add goals for either team when scored.

3. **Disciplinary card allocation**: The application will require a system of allocation for disciplinary cards to be assigned to players on either team when they infringe upon the 'Laws of the Game', and the manner of their indiscretion.

4. **Player substitution processing**: The application will require a method of selecting a player as they leave the field of play and their replacement as they enter the field of play.

5. **Recording of player injuries**: The application will require the functionality to record when a player has been injured during the game.

6. **Time stamp registered game events**: The application will require a method of recording and saving the time of any of the aforementioned events as they occur and are processed by the referee during the game.

7. **Export game reports**: The application will require a method of saving the data processed during a game, retrieving this data and exporting it in a format which can be used by the referee as the basis of their post-match report.

8. **Referee activity tracking**: The application will require the use of a smartwatch's hardware such as sensors and GPS to track data associated with the referee's activities such as heart rate and distance covered during a game.

9. **Audio recording**: The application will require an audio recording feature to allow the referee to store audio notes on the game's events, such as acts of player indiscipline or the use of inappropriate language.

10. **'Laws of the Game' reference guide**: The application will require an accessible guide to the 'Laws of the Game' which the referee can reference in an efficient and intuitive manner.

### 3.3.2: Non-functional Requirements

11. **Reliability**: The application must be available to the user at all times. The rate of failure must effectively be zero before deployment due to the nature of its intended use. Constraints such as limited smartwatch battery life must be taken into consideration.

12. **Robustness**: In the event of system failure the time to restart must be minimal. The probability of data corruption must be minimized before deployment.

13. **Speed**: User/event response time, i.e. the time taken by the system to process the users' transactions, must be minimal in addition to a providing the user with a quick screen refresh time.

14. **Ease of use**: Training time for the application's users should be kept to a minimum. An easy to understand user guide will be required to instruct users. Tolerance of input errors on the part of the user should be considered, with the intended end goal achievable through minimal correction by the user.

15. **Portability**: The application will be developed for compatibility with a number of target systems to increase accessibility to a greater number of users.

### 3.3.3: User Stories

| User Requirement | Criteria |
|---|---|
| As a user I want an intuitive user interface requiring minimal interaction. | • Situational testing utilised as new features are developed and implemented to ensure system facilitates quick interactions which do not distract from the referee's real-world environment. |
| As a user I want to navigate through the system using simple commands. | • Include simple touch screens or need for gestures to control user navigation.<br>• Consistent navigation to ensure user familiarity with system. |
| As a user I want a stable system. | • Rigorous testing to ensure that the application can withstand use during a game without fear of system failure or data loss. |
| As a user I want a stand-alone solution. | • Use of built-in smartwatch features such as GPS can negate requirement for accompanying mobile device. |
| As an older user I want a screen which is easily readable. | • Testing of UI colour scheme and font type and size in various weather and lighting conditions to determine optimal readability. |
| As a user I want the option to modify any data that I have entered. | • Use of buttons to remove a goal, for example, after it has been added or, alternatively, a toast to prompt user if they are sure they want to proceed with the data they have entered in order to reduce amount of incorrectly inputted data. |
| As a user I want to be notified when the time in each half of the match has elapsed. | • Inclusion of audible or vibrating alerts, customisable to the users' specifications. |
| As a user I want to retrieve my data after a game. | • Exportable game reports in PDF format, for instance, to facilitate retrieval and analysis of data after the conclusion of a game.<br>• The storage of game data in a database. |
| As a user I want to know the elapsed time of the match at a glance. | • Use of *Android Wear*'s 'always-on' feature to facilitate access to game timer without need for screen touch or gesture. |

Table 1: Table outlining user requirements for the application.

In addition to the aforementioned functional and non-functional system requirements defined in collaboration with the project's stakeholders, further suggestions for feature implementation received from the focus group included the ability to connect a smartwatch running the application with other smartwatches. Such a development could effectively

eradicate the need for the use of beeper flags by assistant referees as, in theory, when the referee and their assistant referees are using the application in conjunction it could enable each team member to send an alert to another through a simple tap of their watch's screen. While such suggestions provide great insight to the needs of referees and their assistants, the scope of this project is relatively limited in its focus upon the most salient needs of the grassroots referee and more advanced features are unlikely to be achievable at this juncture. Further feature implementation and refinement will be consigned to future iterations of the application.

## 3.4: Summary

The preceding chapter has presented the requirements of the software project's end users, as determined by the key findings gleaned from market research in conjunction with analysis of the results of a focus group questionnaire. The system requirements informed both the design and implementation processes, which are documented in the subsequent chapters.

# 4: Professional Issues

The following presents an analysis of the professional issues associated with the project.

## 4.1: Business Case

*RefWatch: Soccer Edition* is a smartwatch application proposed for use on a smartwatch device running Google's *Android Wear* operating system. The application aims to provide football referees with an alternative to using pen and paper to record football match events while in turn offering a cost-effective alternative to dedicated referee watches.

While existing referee watches such as the Spintso *Referee Watch Pro* primarily facilitate a referee's timekeeping responsibilities, *RefWatch: Soccer Edition* will reduce the cost to the consumer through the inclusion of a number of additional features within the application:

- Dual timer
- Score keeping
- Yellow/Red Card allocation
- GPS tracking
- 'Laws of the Game'
- Exportable match reports

The added functionality afforded to referees will, consequently, eradicate the need to carry additional resources such as reference guides to 'Laws of the Game', dedicated fitness tracking devices, and additional watches for timekeeping purpose, thereby reducing the costs associated with refereeing while exportable match reports will reduce the time requirement placed upon referees when writing and submitting their reports to the relevant footballing authorities.

Due to the distinct lack of smartwatch applications currently available to referees, *RefWatch: Soccer Edition* has undeniable market potential. The referee focus group provided an overwhelming indication that referees are ready to adopt a well-designed smartwatch referee app should it become available (see *Appendix Two*). While Spintso's *PDA*, for instance, is a dedicated multi-functional device (see *2.3: Comparative Analysis of Existing Products*) it's price is prohibitive when compared to a smartwatch such as the Sony *SmartWatch3*. While Spintso's products are used exclusively for refereeing,

furthermore, a smartwatch can be utilised in a multitude of contexts by the user and therefore offers greater value for money. While *RefWatch: Soccer Edition* may be a niche product, furthermore, the number of FIFA-registered referees was listed at 840,000 in the world governing body's 2006 edition of its *Big Count* report; this marked an increase of 7% on the 2000 report. This number does not take into account a larger demographic of unregistered referees operating at the grassroots level, and will only have grown in the interim in accordance with football's growing global appeal (FIFA.com, 2007). There is therefore a significant number of potential users for the *RefWatch: Soccer Edition* app. By choosing the *Android Wear* platform for the application's development, moreover, it will be accessible to a greater share of these potential users since devices running Google's smartwatch OS are not locked to a single app store (Developer.xamarin.com, 2016).

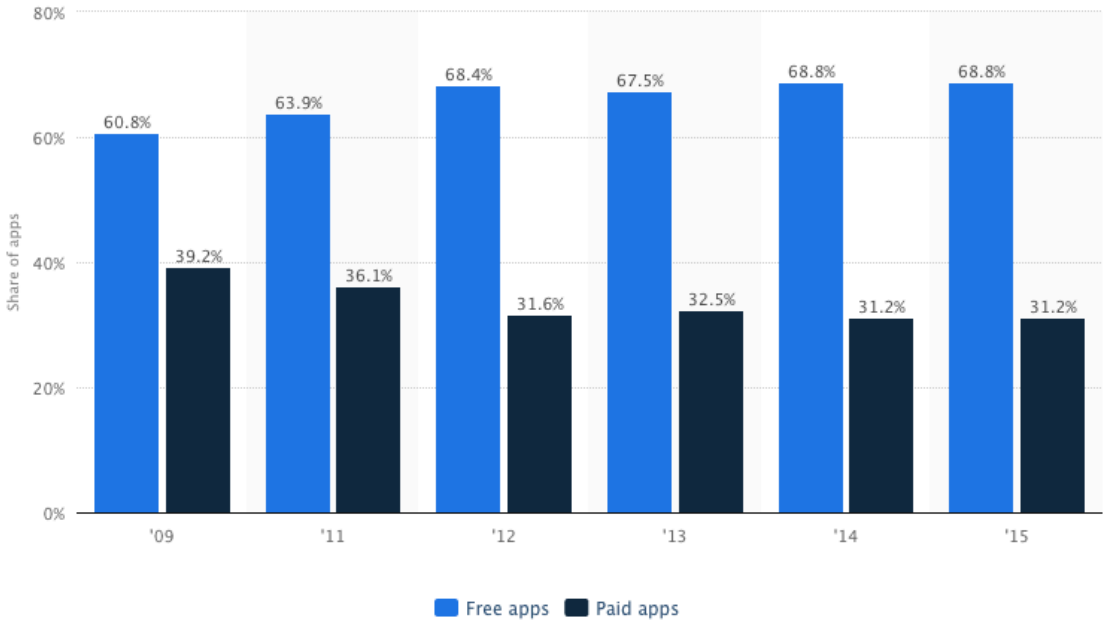



Figure 3. Chart showing the distribution of free and paid Android apps on the Google Play app store from 2009 to 2015.

The trend in mobile application pricing in recent years has seen a move towards a 'freemium' model, which offers the user a free-to-download product with additional costs incurred to access premium in-app content. Research suggests, furthermore, that *iOS* users are slightly more inclined to pay for applications than their *Android* counterparts (Sourcebits, 2013). As Figure 3 illustrates, from 2009 to 2015 the proportion of free to download applications hosted on the Google *Play Store* increased from 60.8% to 68.8% (Statista, 2015). Similarly, over 90% of the Apple *App Store*'s contents fell within the $0-$4.99 price range in 2013. Although free applications are increasingly popular and have proven to be a sound monetization strategy for developers, however, users are more likely

to purchase an app if it offers a niche functionality which targets a specific demographic of the market (Sourcebits, 2013).

When considering a suitable price tag for *RefWatch: Soccer Edition*, the cost of existing mobile applications was taken account:

- *Soccer Referee Shingo 2.5* is a free app which produces revenue for its developer by incorporating in-app advertising. The app currently has almost 100,000 downloads in the Google *Play Store* (Spinkeysoft.com, 2016).

- The basic version of the *Referee Wallet iOS* app is a free-to-download iteration containing the core functionality of timekeeping and recording match events, while a paid version priced at $4.99 includes the additional functionality of exportable match reports in PDF format (Epstein, 2013).

With a price tag of $4.99/£3.50, *RefWatch: Soccer Edition* should prove to be equally accessible to the consumer and lucrative for the developer. The application's development costs are currently projected to be minimal.

## 4.2: Project Risks

The following risk management assessment presents an analysis of the identified health and safety, technical and project management risks associated with this project alongside the appropriate countermeasures to be used in order to minimize risk and, moreover, ensure the project's optimal outcome.

### 4.2.1: Health and Safety Risks

It has been determined that there are no identifiable health and safety risks to the developer or end user associated with this project.

### 4.2.2: Project Management Risks

| Risk factor | Probability* | Threat* | Risk (Probability × Threat) | Preventive measures |
|---|---|---|---|---|
| Project overrun | 2 | 3 | 6 | Effective time management procedures such as weekly timetable planning and incremental development will ensure that deadlines are met. |
| Insufficient research | 1 | 2 | 2 | A repository of research topics and keywords will ensure that all relevant topics are investigated in sufficient depth. |
| Unforeseen circumstance or developer oversight | 2 | 3 | 6 | Agile methodology to produce an early working prototype ensuring minimum disruption to delivery schedule. |

Table 2: Table illustrating the project's management risks and associated preventive measures.

*The level of probability and threat is identified using a scale of 1 to 5, with 1 representing the lowest and 5 the highest.

### 4.2.3: Technical Risks

| Risk factor | Probability* | Threat* | Risk (Probability × Threat) | Preventive measures |
|---|---|---|---|---|
| Android Studio data corruption or loss | 2 | 4 | 8 | Create backup save of code at each step in the development process. This will facilitate use of previous versions to pinpoint any errors with frequent software testing. |
| Dissertation data corruption or loss | 1 | 4 | 4 | Daily backup of work on external storage to ensure minimal risk of corruption/loss. |
| Hardware malfunction | 2 | 3 | 6 | Installation of latest stable OS updates, with access to alternative systems as a failsafe. |
| Internet Service Provider failure | 1 | 1 | 1 | Readily-available alternative sources of internet access will ensure isolated loss of service poses minimal risk to the overall project. |
| Design or architecture is infeasible | 2 | 4 | 8 | Extensively researched compatibility analysis, and incremental design and implementation process will ensure maximum flexibility to meet UI or architecture redesign requirements. |

| Risk factor | Probability* | Threat* | Risk (Probability × Threat) | Preventive measures |
|---|---|---|---|---|
| Software lacks stability | 2 | 3 | 6 | Frequent testing of software to ensure identification and eradication of components that crash the system. |

Table 3: Table illustrating the project's technical risks and their associated preventive measures.

*The level of probability and threat is identified by using a scale of 1 to 5, with 1 representing the lowest and 5 the highest.

## 4.3: Ethical Considerations

This project meets the definition for research in Category Z as determined by Ulster University's Ethics Committee and therefore does not meet the requirement for a full Ethical review. All risks and ethical procedural implications have been considered. The project will be conducted at all times in compliance with the research description/protocol and in accordance with the University's requirements on recording and reporting. No invasive research will be conducted throughout the project's duration. The project will not collect or use personal data. Any data used will be simulated solely for software testing purposes. Furthermore, the project's study group participants do not constitute a vulnerable demographic.

# 5: Design

The design phase of the software development lifecycle is fundamentally important to the achievement of an optimal end-user experience. It was necessary to attain a thorough understanding of the users' requirements in order to define the various elements of the final software system to be implemented and to determine how the application will look and function. The design process for the development of the *RefWatch: Soccer Edition* smartwatch application incorporated three core aspects:

1) Architectural design.
2) Database design.
3) Graphical User Interface (GUI) design.

The following chapter discusses the necessary integration of User Experience Design (UXD) concepts in conjunction with a keen understanding of the identified user requirements (see *3: Requirements Analysis*) when developing the system architecture, an appropriate database and a Graphical User Interface (GUI) for implementation in the application's prototype. A number of key design considerations, including discussions of the smartwatch form factor and the relationship between the application's interface and its external environment, are presented. In addition, a brief insight is provided to the architecture of the application's chosen platform, namely *Android Wear* and the related *Android* operating system, while the design of the application's architectural layout and a suitable database for data storage, categorization and retrieval are also recounted.

## 5.1: Software Development Methodology

A Scrum approach to the Agile software development methodology was utilised throughout this project, from its inception during analysis and requirements verification through to the design, implementation, testing and deployment phases. While specification-based software development methodologies such as a Waterfall approach typically identify the requirements before designing, building and testing a system, Somerville argues that they "are not geared to rapid software development." Conversely, Agile is a responsive and flexible methodology which enables developers to react to changing requirements or problems as they are discovered. Furthermore, Agile encourages the rapid fulfillment of

software projects through the development of a series of versions. This incremental delivery allows each version to be evaluated in collaboration with stakeholders and then refined as requirements change or evolve (Somerville, 2011). The Agile approach to development is illustrated diagrammatically by Figure 4.

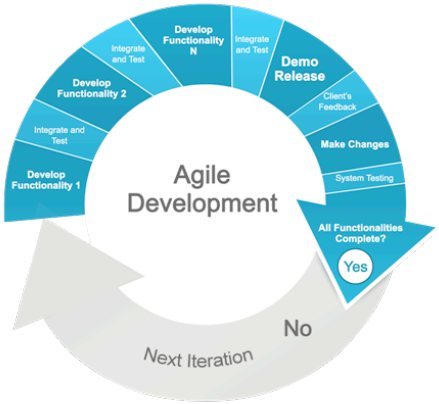



Figure 4: Illustration of the Agile Methodology (StrategyBeach, 2013).

The Scrum management process, which is a version of Agile, aligned closely with the planned structure of the overall project including its initialisation in the *Analysis and Design* phase. The aforementioned initial phase was followed by a series of incremental 'sprint' cycles of 2-4 weeks in duration which facilitated rapid development of the application's core functionality and, finally, concluded with a project closure phase which in this instance represented a written dissertation. An illustration of the Scrum approach to management is displayed in Figure 5 (Schwaber, 2004; Schwaber and Beedle, 2001). Since the Scrum approach typically utilises daily or weekly collaborative meetings between a software project's team of developers, where priorities can be assigned to a product backlog containing the project's development requirements, the chosen methodology required some alteration to accommodate the presence of a single developer and project manager in this instance. Nevertheless, the 'sprints' approach ensured timely fulfillment of the application's objectives by utilising a rapid development and review process.

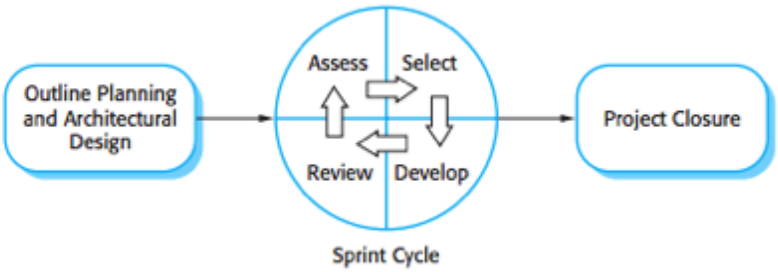



Figure 5: Diagram illustrating the Scrum management process (Somerville, 2011).

## 5.2: Development Platform

The application was developed using *Google*'s *Android Studio* platform, the official Integrated Development Environment (IDE) for *Android Wear* application development, based on the IntelliJ IDEA (Developer.android.com, 2016 c). The *Android Studio* IDE offers a number of resources and tools which were utilised throughout the development of the *RefWatch: Soccer Edition* smartwatch application, including:

- A code editor where the programmer can write and edit text-based files using the Java and XML programming languages. The code editor expedites the process of writing source code by detecting and highlighting programming errors in real-time and, furthermore, by providing suggestions for code completion to reduce the amount of typing required by the programmer.

- Testing tools and frameworks such as an emulator which enables the developer to preview their application's layout and functionality in real-time by configuring an Android Virtual Device (AVD).

- An SQLite database platform built into the library layer of the *Android Studio* architecture, facilitating the application's data storage (Techotopia.com, n.d.).

## 5.3: *Android Wear* Architecture

*Android Wear* works through wireless communication between its wearable host device and the handheld device, such as a smartphone, to which it is 'paired'. The handheld device will typically run version 4.3 or higher of the *Android* OS as it requires Bluetooth LE, a feature which was not supported by earlier versions. When a connection has been established between devices, notification messages and other data can be exchanged or synced between the wearable device and the handheld device. This data can subsequently trigger appropriate actions on either device. The diagram in Figure 6 illustrates the architecture of a wearable device and how it is interconnected with that of its handheld counterpart.

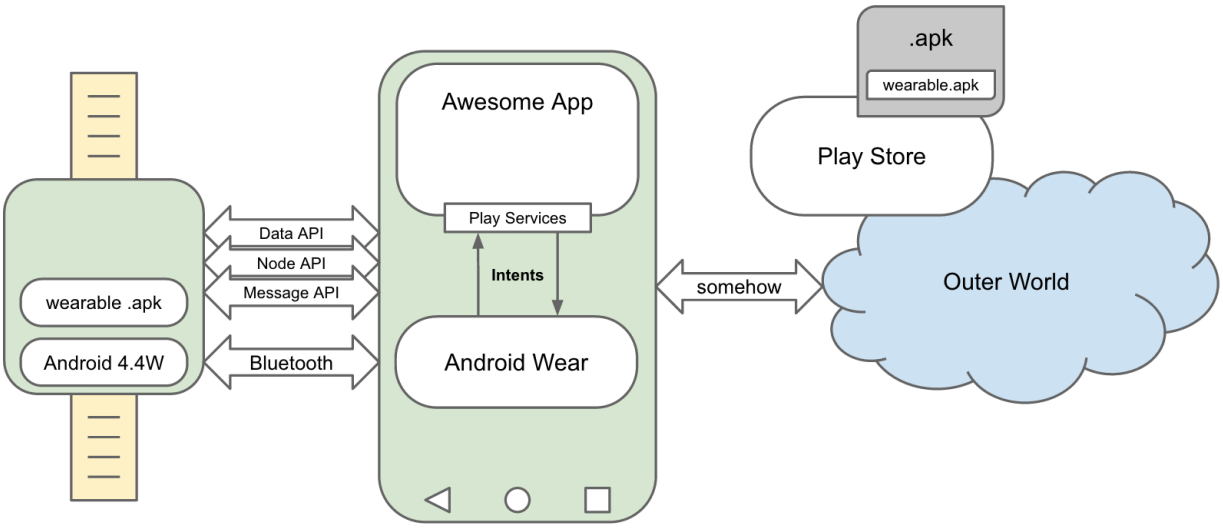



Figure 6: Diagram showing the typical Android Wear architecture and its connection with a handheld Android device (Daniel, 2015).

A number of Application Programming Interfaces (APIs) are provided as a facet of Google's *Play* services (version 5.0) which allow a developer's application to access the operating system's features. When a connection is established between a wearable and handheld device, for instance, a node can be assigned to handle specific functions such as tracking a user's GPS co-ordinates; the Data API class is responsible for the data being synched between the aforementioned devices and provides a syncing mechanism at both ends of the connection (Daniel, 2015).

## 5.4: Prototype Design

Following the introduction of *Android Wear* as a platform for wearable devices, Google has developed a set of design guidelines called *Material Design* which provides a framework for developers working across the various *Android* platforms. Before work on the development of an application prototype could commence it was therefore necessary to thoughtfully appraise Google's design principles, while also re-assessing the conceptual designs devised for the system's architectural structure and user interface layouts in the *Analysis and Design* phase of the project. By envisaging specific end-use cases based upon the system requirements and user stories outlined in the preceding chapter it was possible to crystalise how the final application would function. Earlier architectural and user interface designs could consequently be improved upon and the software components capable of fulfilling the functionality required by the system's end users could be identified through further research.

To aid the design process it was necessary to categorise the application's functionality and software component development tasks. The resulting table contains four categories - 'Graphical User Interface', 'Core Functionality', 'Customisable Options' and 'Peripheral Functionality' - and can be viewed below (see *Table 4*). In several instances the system's requirements where grouped together into subsystems - the required timer functions, for example, where grouped together under the umbrella of a 'timekeeping' subsystem - which not only informed the research process when identifying software solutions, but also influenced the design of a coherent system architecture and intuitive Graphical User Interface (GUI) layouts as documented in the subsequent sub-chapters.

<table>
<tr><th>Graphical User Interface</th><th>Core Functionality</th><th>Customisable Options</th><th>Peripheral Functionality</th></tr>
<tr><td>Application logo</td><td>Timekeeping:<br>- Countdown timer<br>- Elapsed gameplay timer<br>- Stoppage timer</td><td>Period duration</td><td>GPS activity tracking</td></tr>
<tr><td>Application icons</td><td>Score Card</td><td>Number of periods</td><td>Audio recording</td></tr>
<tr><td>Application colour scheme</td><td>Period Indicator</td><td>Interval duration</td><td rowspan="2">Exportable match reports</td></tr>
<tr><td rowspan="2">Screen layouts</td><td rowspan="2">Recording match events:<br>- Goals<br>- Penalty Kicks<br>- Substitutions<br>- Disciplinary cards</td><td>Audible or tactile alerts</td></tr>
<tr><td>Language selection</td><td>'Laws of the Game' reference guide</td></tr>
</table>

Table 4: Table categorising the system's requirements and software component development tasks.

## 5.5: Architectural Design

A software system's architecture has been described by Fowler as the "highest-level breakdown of a system into its parts..." (Fowler, Martin. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002.). The process of designing the software system's architecture therefore encompassed the selection of the structural elements and interfaces by which the system is composed while ensuring that all of the fundamentally important user requirements were satisfied. It was important, furthermore, that the architectural design ensures that the application's reliability and processing speed match the users' expectations while remaining flexible and amenable to future developments.

An incremental approach to the architectural design process was utilised. Such an approach, which incorporated iterative testing, facilitated the refinement of the design to a baseline architectural model and ensured, moreover, that all of the necessary requirements were accounted for and a thorough understanding of how the application's various sub-system's and functions would interconnect. A visualisation of the application's architectural design is outlined in a flow diagram (see *Figure 7)*.

The application initiates with a 'splash' page which displays a logo and through which, with a click of the watch's screen, the user gains access to the *Main Menu* screen. The *Main Menu* presents the user with four 'clickable' options though which they can access the system's main functions:

- **Match**: The *Match* menu tab grants the user access to the application's core functionality, such as timekeeping facilities and the ability to create and store a record of important match events.

- **Reports**: A potential future development, the *Reports* menu tab will present the user with the ability to export saved match reports from the application's SQLite database in PDF format.

- **Options**: The *Options* menu tab gives the user access to customisable elements of the application, such as a choice between audio and tactile alerts, and language selection.

- **Quit**: By clicking the *Quit* menu tab the application terminates.

The development of the core user pathway initiated by selecting the *Match* option, which contains the application's core functionality, was the main focus of the design and implementation phase of the project. By clicking on the *Match* menu tab the application navigates to the *Period Duration Selection* screen, which presents the user with the ability to select a period duration ranging between one and forty-five minutes in length. After selecting the period duration, the user is presented with the option of choosing between one and four periods of play on the *Number of Periods Selection* screen. If the number of periods chosen is greater than one, the option to set the duration of the interval between periods is subsequently presented on the *Period Duration* screen. After customising the match duration settings to their specifications the user is presented with the *Main Match*

*Screen*, which forms a central hub for Human-Computer Interaction (HCI) during a match. The *Main Match Screen* accommodates the application's timekeeping and scorekeeping functionality. It provides a pathway, moreover, to the selection of match events through the adjacent *Match Event Selection Screen* where the user can choose from a list of options which provide access to the various functions which update the referee's running record of important match events.

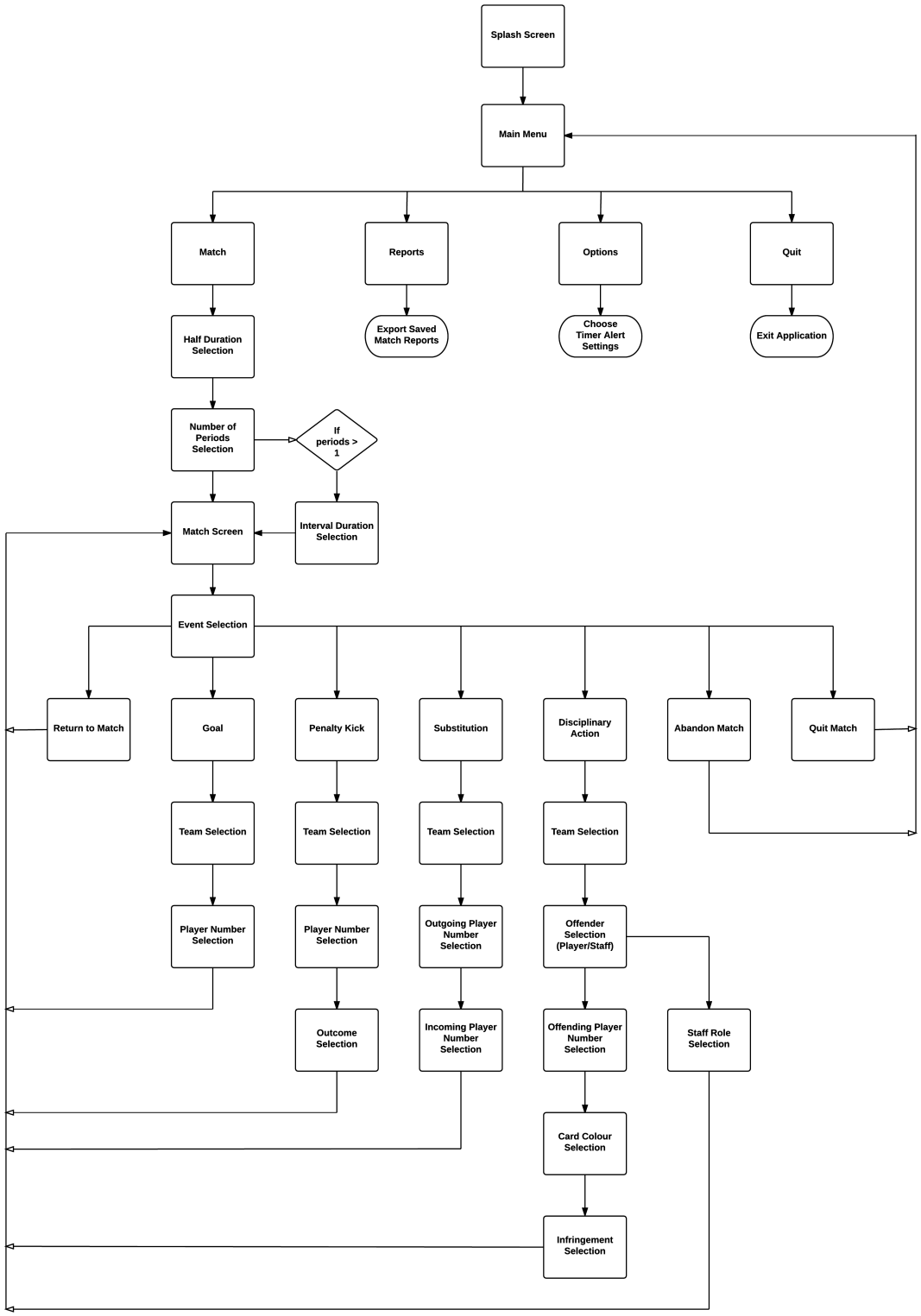



Figure 7: Flow diagram depicting the application's architectural design.

## 5.6: Database Design

The *RefWatch: Soccer Edition* application incorporates a database in which to store, organise and retrieve the software system's data. An Entity Relationship data model was implemented in the *Android Studio* IDE through the use of an SQLite library, which is incorporated in the *Android* and *Android Wear* architectures and was used to create and manage the relational database within the application. The design of the database is directly related to the *RefWatch* application's architectural design and the intuitive path a user takes through the application's various screens and functions. The information received from a referee as they customise a match's settings, such as the duration and number of periods, and record events such as goals scored, for instance, are stored in the database and can be retrieved to form the basis of exportable game reports. The database employs a relational database methodology which includes a number of relations, or tables, each of which represents an entity in the database.

The database's design contains five relations or tables: *match*, *goal*, *penalty_kick*, *substitution*, and *disciplinary_action*. Each relation and the data to be contained within it references a football match or a specific aspect of a match, and each entity is easily identifiable by its assigned title. The *match* table, for example, stores a match 'ID' which uniquely identifies any match that has been processed by the application. Additionally, each table in the database contains various attributes, or columns, which relate to a characteristic of the parent entity. For instance, the *match* relation contains attributes which store data relating to a match's date and time of kick-off, period duration, number of periods and, where applicable, interval duration as selected by the user in the appropriate application screen. Additional relations are designated to store the data related to important match events such as goals scored, penalty kicks scored or missed, player substitutions, and any disciplinary cards allocated to the participating teams. Each of the match event entities represented in the database - *goal*, *penalty-kick* and *pk_scored*, *substitution* and *disciplinary_action* - are designated various attributes in the same manner as the *match* relation, which enables the effective categorization of match data and ensures its efficient storage and retrieval when exporting the users' post-match reports.

Each table contains a primary key, an attribute/column (or combination of columns) which uniquely identifies any tuple/row in a table. The primary key contains a unique value for each row of data in the table; it cannot contain null values, and therefore enforces the

database's entity integrity. An associated foreign key is utilised to establish relationships between tables within the database (Techopedia.com, n.d.). A relationship is established, for instance, between the *match* relation and various match event entities through the inclusion of the *match* table's primary key, *match_id*, as a foreign key in each of the event entity tables. Each match event utilises a primary key comprising a unique ID related to the specific type of event - *goal_id*, *pk_id*, *subs_id* or *discipline_id* - thereby differentiating between the numerous occurrences of each event. Since it is possible for many instances of an event such as a substitution to occur in a single match, however, the cardinality of the relationship between the match table and the four event tables is one-to-many (1:N); each occurrence of a match, represented by a unique match_ID, can be associated with a number of occurrences of a goal_id, pk_id, subs_id or discipline_id.

Each of the database's tables are listed in Appendix Three, while their attributes and relationships to one another are illustrated in an Entity Relationship diagram (see *Figure 8*).

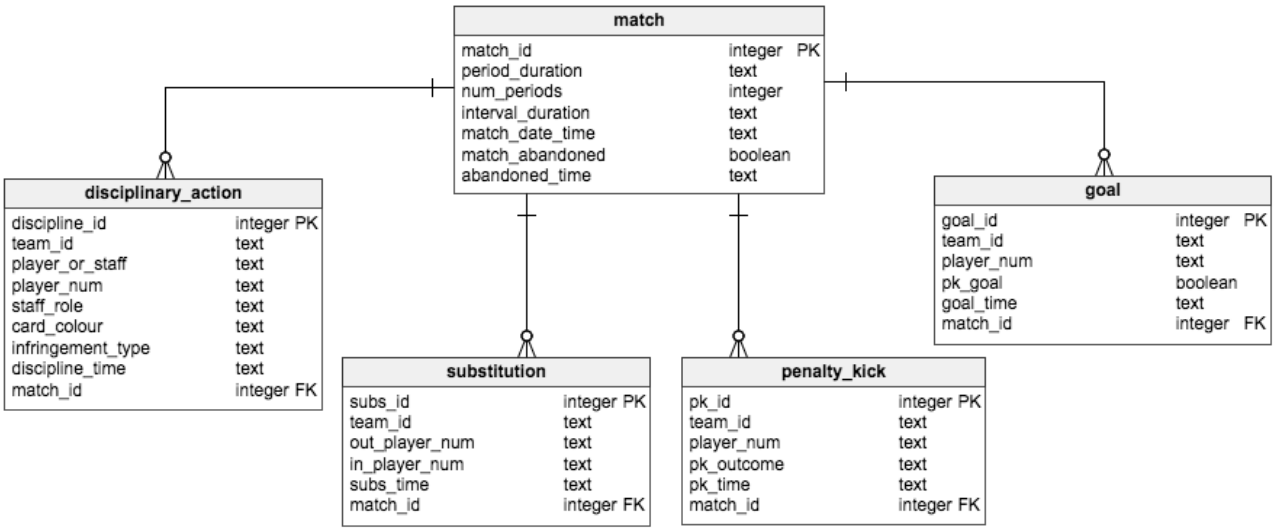



Figure 8: ER diagram illustrating the application's database design.

## 5.7: GUI Design

The design of a software application's user interface (UI) is fundamentally important to the users' experience of the product. The application's UI acts as the medium for interaction between the human user and the system's hardware with the purpose of ensuring not only the user's effective operation of the system, but also the hardware's feedback of information to inform and aid the user's decision making process (Johnson, 2014). Guidelines for intuitive software user interface design have been promoted since the

1970s and repeatedly revised. Any software design process must therefore take into consideration Human-Computer-Interaction (HCI) paradigms and software engineering issues to ensure the end product's accuracy and usability match the users' expectations.

The following sub-chapter identifies the necessary integration of User Experience Design (UXD) concepts alongside the identified user requirements which culminated in the development of a Graphical User Interface (GUI) prototype for the *RefWatch: Soccer Edition* application. Furthermore, a discussion of key design considerations, including the smartwatch's form factor and the relationship between the application's interface and its external environment, are presented.

### 5.7.1: Form Factor

The form factor, which takes into account the size, configuration and physical arrangement of the device hosting the software application, must be considered before an object-oriented design can be formulated. Devices such as tablets or smartphones have much larger displays than a smartwatch and can therefore accommodate more information in a single screen. A smartwatch application must therefore be designed specifically to be effective on a small screen. The following represent some of the most important elements of the smartwatch's form factor affecting the application's design:

- The decreased form factor of the smartwatch device presents a number of potential challenges and constraints. The amount of available memory and processing power is typically reduced in smaller devices, for example, which could have had an adverse effect on the application's performance if overlooked.

- In order to conserve a smartwatch's limited battery life, earlier generations of applications could not function in an ever-present manner when the user was no longer interacting with the watch's screen. This could have proven problematic for the development of a referee application with timekeeping functionality, as the referee must be able to glance at their watch to determine the time which has elapsed at any given moment throughout the duration of the football match they are officiating. With the release of *Android Wear* 5.1., however, Google introduced a new 'always-on' feature which allows the user to see their app at all times - albeit in a black-and-white themed screen which continues to conserve their device's battery life - and thereby effectively eliminating this concern (Droid Life, 2015).

- Smartwatches were partially designed with a view to displaying notifications from mobile applications hosted on the smartphone to which the watch is 'paired' or connected. In the context of a referee's role, however, receiving such notifications would be distracting and unwelcome. By placing the smartphone in 'Airplane Mode' this problem can be removed, as notifications from other applications running on the smartphone will be blocked until the user reverts the device to its default setting.

- Whereas a platform such as *WatchOS* currently runs exclusively on the Apple Watch, *Android Wear* is an operating system used by a myriad of different devices with varied specifications and capabilities. It is therefore important to take into consideration that any function of the proposed application which requires specific hardware features, such as GPS for activity tracking, may not work as anticipated on certain devices. Furthermore, when designing the UI the screen resolution of a device must be taken into account (Developer.xamarin.com, 2016). Testing on multiple devices will therefore be necessary to validate the application's performance and to ensure its optimal accessibility and usability.

- While wearable technology such as the smartwatch may present some design challengers for the developer, however, they have also provided opportunities for users to interact with their device in new and different ways. One commentator has predicted, moreover, that "Gesture recognition, tapping patterns, health data and vibrational communication are all some of the capabilities that will enable the future of screenless interactions" (Fjordnet.com, 2015).

### 5.7.2: Human-Computer-Interaction

Developing an understanding of the system's interface and the context or environment in which it will be used was fundamental to the User Experience Design (UXD) process. The role and responsibilities of a football referee are conducted in a fluid, fast-paced environment and therefore any interaction between the human user and system must be simple and easy to perform quickly yet accurately. Furthermore, football matches are played in diverse weather and lighting conditions, so the user interface's colour scheme and layout display must be readable in any situation and will rigorously tested.

### 5.7.3: Template Design

Following the introduction of *Android Wear* as a platform for wearable devices, Google developed a set of design guidelines called *Material Design* which provides a framework for developers working across the various *Android* platforms. Taking into account Google's guidelines, the process of designing screen layouts was initiated by creating a number of wireframe images (see Figure 9) which served as a foundation upon which the application's final GUI design was based. By providing access to a selection of development resources commonly utilised by Android developers within the *Android Studio* development environment, including an extensive selection of widgets ranging from buttons to progress bars, the online prototyping tool *Proto.io* facilitated the tools necessary to create a number of conceptual screen layouts (Proto.io, 2016). The wireframe images were implemented and, in many instances, improved upon during the project's implementation phase which is documented in *Chapter 6: Implementation*.

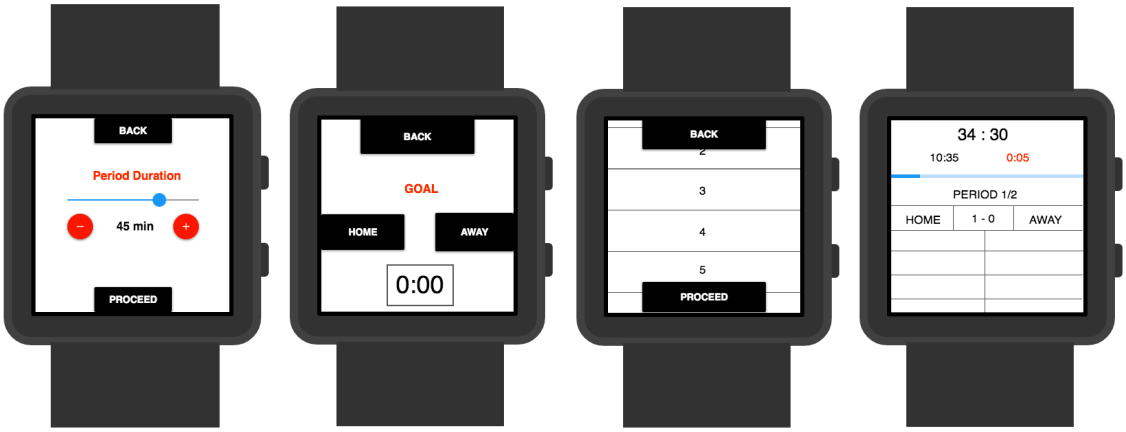



Figure 9: Wireframe images illustrating conceptual designs for the application.

### 5.7.4: Colour Scheme

The application's chosen colour scheme was chosen to satisfy Google's *Material Design* guidelines on colour. Taking its inspiration from "contemporary architecture, road signs, pavement marking tape, and athletic courts", the guidelines recommend the use of primary and accent colours while encouraging consistency regarding application styling (Material design guidelines, n.d.).

In line with Google's recommendations the *RefWatch* application makes use of a white background (colour code: #ffffff) alongside the accent colours of red (#ff0000) and black

(#000000) for most of its text and icons, a theme which was chosen for most of the application's screens to ensure consistency. The chosen colour scheme establishes a vibrant and aesthetically pleasing interface with adequate contrast levels which perform well under a variety of lighting conditions. A colour swatch illustrating the application's primary colour scheme is shown in Figure 10; the colour codes are included in the annotation.

Figure 10: Application colour scheme: red (#ff0000), white (#ffffff) and black (#000000).

### 5.7.5: Navigation

While it is possible to utilise 'screenless' interactions by integrating gesture control - the ability to perform actions, such as navigation from one screen to another, through wrist movements - in the design of a smartwatch app, it was determined that in a fast-paced, mobile environment such as a football match this would be detrimental to the application's usability. Through the use of a simple, consistent and intuitive touchscreen design, moreover, it was possible to fulfill user requirements such as the need for minimal interaction with the app using simple commands which minimise distractions from the referees' immediate tasks and environment.

A number of clickable buttons and icons were incorporated in the application's layouts to facilitate actions and navigation from one screen to another. Some of the buttons, such as those used consistently to navigate back and forward between screens, were designed using Adobe's *Photoshop Elements* software while for other functions clickable textviews and listviews were selected. A selection of the tools utilised to facilitate the application's navigation, along with descriptions of their functionality and screen locations, are listed in Table 5.

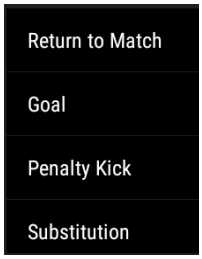
| Icon | Name | Functionality | Position |
|---|---|---|---|
|  | Custom Imageview Back Button | Facilitates user access to the preceding screen on their current pathway through the application. | Atop many of the application's screens. |
|  | Custom Imageview Proceed Button | Facilitates user access to the subsequent screen on their current pathway through the application. | At the bottom of many of the application's screens. |
|  | Clickable Textview | Facilitates user access to the application's main pathways on the 'Main Menu' screen, and to the intra-game 'Match Event Selection' menu through the clickable timers. | Main menu and the timer atop the main 'Match Screen'. |
|  | Selection Button | Facilitates actions such as Team Selection for the purposes of accurate recording of match events. | On the left and right of the lower-centre portion of screens. |
|  | Listview Menu | Facilitates user access to the application's match event recording functionality - goals scored, penalty kicks awarded, player substitutions, and disciplinary action - during the course of a match. | Full screen. |

Table 5: Table containing information on the common methods of navigation throughout the application.

### 5.7.6: Layout Development

The design of the application's user interface has been largely informed by a number of sources identified during the *Analysis & Design* phase of the project including Google's *Material Design* guidelines and existing products such as Spintso's *Referee Watch.* Further research has revealed a variety of useful literary and web resources which have been drawn upon including Ben Schneiderman's *The User Interface*, a seminal text on the subject of Human-Computer Interaction (HCI). Schneiderman's "Eight Golden Rules of Interface Design" (see *Appendix Five*), which advocate "consistency" and "universal usability" among its core principles for HCI interaction, were incorporated or adapted where appropriate during the interface design process (Schneiderman, 1987).

The central aim throughout the GUI design process was to produce a user interface which would facilitate a football referee's primary match responsibilities - namely timekeeping and recording match events such as goals scored, disciplinary actions, and player substitutions - in a simple, intuitive, and time efficient manner. Traditionally a soccer

referee utilises a wrist watch for time keeping in addition to a report card upon which they can record match events through the medium of a pen or pencil. In order to create a truly innovative application it was determined that the UI design should incorporate a facility to display key details of the core functions - timekeeping and the recording of match events - in a single screen. The user interface design of the *Main Match Screen*, which acts as the application's central hub for referee activity, is the result of this innovative vision and is depicted in Figure 11. It's features include:

- A countdown timer textview. Displaying 'START' when the user first navigates to the match screen, the countdown timer appears and begins to count down to zero upon a click of the screen. The amount of time displayed by the countdown timer is determined by the user on the Period Duration Selection screen (see *Figure 12)*.
- A timer textview which displays the total elapsed playing time, including the time accumulated during stoppages in play. which counts up from 00:00.
- A timer textview which displays the time accumulated during stoppages in play.
- A progress bar which traverses the screen from left-to-right and displays the countdown timer's current progress towards 00:00.
- A period indicator textview which displays the current period of play and the total number of periods to be played. Any number of periods between one and four can be selected by the user on the *Number of Periods Selection Screen*.
- Textviews to indicate the current scoreline and designate the number of goals scored by each of the participating teams, both home and away.
- Scrollable listviews for both the home and away team, which will display and organise icons depicting important recorded match events such as yellow or red cards.

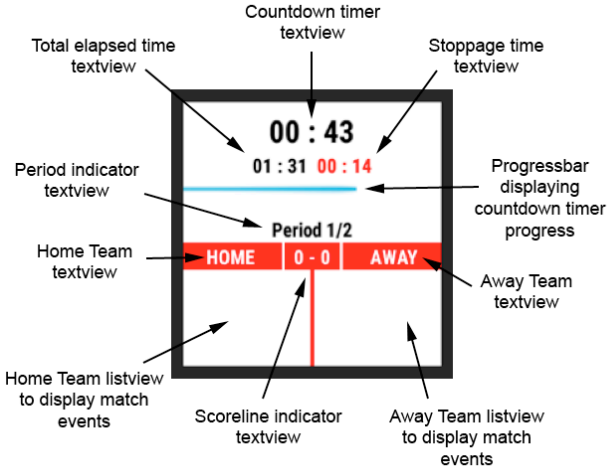



Figure 11: *Main Match Screen* interface design features.

After starting the countdown timer, a subsequent click of the timer navigates the user to the *Match Event Selection Screen* where they can select from a number of options presented in a scrollable listview format. The options are as follows:

- Return to Match
- Goal
- Penalty Kick
- Substitution
- Disciplinary Action
- Abandon Match
- Quit

The *Goal* option, for example, allows the user to increment the number of goals scored by either team. A depiction of the *Goal* option's screen layout is included in an *Application User Story Board* diagram (see *Appendix Four*). The story board illustrates a user's prospective pathway through the application, and includes pseudocode which was subsequently used as a foundation when coding the app's functionality in *Android Studio* (see *Chapter 6: Implementation*). By clicking on the *Return to Match* option the user can return to the *Main Match Screen*, giving the referee the flexibility to accumulate stoppage time during any stoppages in play regardless of whether they have a match event to record using one the Match Event Selection options e.g. when the referee stops play to award a free kick which does not require any supplementary discipline.

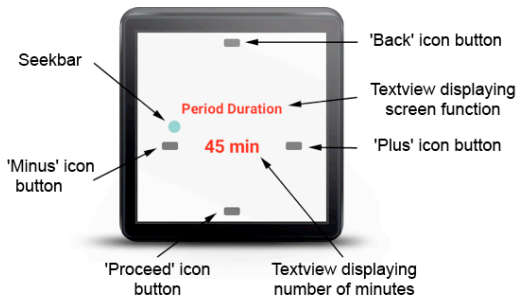



Figure 12: *Period Duration Selection* interface design features.

Consistency is a theme which is evident throughout the UI design which facilitates intuitive navigation throughout the application and promotes the expedient learning of its features and use. Such an approach will help the user to feel in control of their experience within

the application's environment and should ensure its widespread appeal and adoption among the grassroots football referee community. A duplicate layout design has been chosen for sections of the application which can be naturally grouped together due to similarities in their function such as, for example, the *Period Duration*, *Number of Periods* and *Interval Duration* selection screens. The widgets included in the aforementioned layouts, and their functionality, are illustrated in Figure 12 while the final design following the inclusion of custom designed image buttons is shown in Figure 13.


Figure 13: *Period Duration Selection* interface final design including custom image buttons.

# 6: Implementation

## 6.1: Introduction

The implementation phase of the software development project involved taking the user requirements and conceptual design work from the earlier analysis and design phases, and identifying, programming and configuring the software components capable of producing a solution system. The following chapter documents the implementation of an *Android Wear* software application with layouts and functionality derived from the templates and subsequent prototype developed during the project's preceding design phase and intended to fulfill the core requirements outlined in Chapter 3. The work concluded during each implementation cycle is documented herein, while the development environment and tools utilised are highlighted and discussed. The fundamental components of Android application development which underpin the system's functionality are outlined in addition to the presentation of an implementation overview of the system's core functionality. Finally, the challenges encountered during the implementation process are revealed alongside the solutions which were devised and implemented in order to overcome them.

## 6.2: Implementation Cycles

The project's associated time constraints required that the expected scope for implementation would be confined to the GUI and core functionality, the latter of which had been pre-defined during the design phase (see *5.5: Prototype Design*); peripheral functionality and customisable options were to be consigned to a backlog of future developments. It was determined, however, that the implementation of a fully-functional stand-alone solution to the problems encountered by grassroots soccer referees and, moreover, an application capable of fulfilling their most important responsibilities could be still be achieved by striving to fulfill the additional implementation of exportable match reports and customisable alerts to signal important timekeeping landmarks. The Agile software development methodology was adopted throughout the implementation process. The responsive nature of the Agile methodology promoted a flexible approach to implementation, and enabled backlogged tasks to be prioritised or de-prioritised and expectations for deliverables to be revised when appropriate. Due to the timeline for

delivery, however, superior time management and the continual evaluation of targets remained essential to the success of the endeavour.

Before the implementation process was initialised a plan was drawn up which divided the components identified during prototype development and each component's associated implementation tasks into cyclical units. The implementation cycles - which are listed in Table 6 along with information on their corresponding tasks, programming language, and implementation environment - each lasted between one and two weeks in duration. Thorough research was conducted throughout the implementation process in order to attain a fundamental awareness of the components incorporated within the Android framework and its available APIs and an understanding of how they should be implemented in source code. A number of literary and internet sources, including video tutorials encompassing numerous aspects of Android application development, were utilised throughout the implementation cycles; a full list of project references is included on page 77.

| Cycle Number | Implementation Tasks | Code | Environment |
|---|---|---|---|
| 1 | *Splash Screen* functionality<br><br>*Main Menu Screen* functionality including:<br>- Match option navigation to next screen<br>- Quit option terminates application<br><br>*Main Match Screen* functionality including:<br>- Countdown timer counting down from 45 minutes to zero<br>- Elapsed gameplay timer counting up continuously<br>- Stoppage timer counting up during stoppages in play | Java | Android Studio |
| 2 | *Period Duration Screen* functionality including:<br>- Increment or decrement number of minutes using Seekbar and/or 'plus' and 'minus' buttons<br><br>*Number of Periods Selection Screen* functionality including:<br>- Increment or decrement number of periods using Seekbar and/or 'plus' and 'minus' buttons<br><br>Interval Duration Selection Screen functionality including:<br>- Increment or decrement number of minutes using Seekbar and/or 'plus' and 'minus' buttons<br><br>*Main Match Screen* functionality including:<br>- Countdown timer displays number of minutes selected at the *Period Duration Selection Screen* | Java | Android Studio |

| Cycle Number | Implementation Tasks | Code | Environment |
|---|---|---|---|
| 3 | *Match Event Selection Screen* functionality including:<br>- 'Goal' option increments scorecard textview<br>- 'Return to Match' option navigates to *Main Match Screen* | Java | Android Studio |
| 4 | Designing custom button graphics for GUI displays<br><br>*Match Event Selection Screen* pathways including:<br>- Connecting all screens for each Match Event option | N/A<br><br>Java | Photoshop<br><br>Android Studio |
| 5 | *Main Match Screen* functionality including:<br>- Countdown timer displaying appropriate duration when progressing between periods of play and intervals<br>- User control of period progression by clicking the elapsed gameplay timer<br>- Resetting the stoppage timer to zero at the beginning of each new period | Java | Android Studio |
| 6 | Adding custom buttons to layouts and adjusting layouts for optimal performance on various devices<br><br>Creating a record of match events by inserting icons and associated data such as player numbers for each event type in the *Main Match Screen*'s listviews | XML<br><br>Java | Android Studio |
| 7 | SQLite database implementation | SQL<br><br>Java/SQL | DB Browser for SQLite<br><br>Android Studio |
| 8 | *Options Screen* layout designed to accommodate Alert Settings through the Options menu tab<br><br>Reports option added to *Main Menu Screen* to facilitate exportable match reports<br><br>Audio or vibration alerts, as selected by the user, added to signal countdown timer completion in each period<br><br>Exportable match reports created by drawing data from SQLite database | XML<br><br>Java | Android Studio |

Table 6: Outline of the project's implementation cycles including their corresponding tasks, programming language and implementation environment.

## 6.3: *Android Studio* Project Development

The development of an Android Studio project involves a myriad of components including java source code and a number of separate resources, some of which are automatically generated upon the project's initialisation.

### 6.3.1: Project Initialisation

The application's implementation was initialised by creating a new project in *Android Studio*. When developing an *Android Wear* application it is necessary to create both a 'Mobile' module and a 'Wear' module. The 'Mobile' module is used to develop an application for mobile phones and tablets, while the 'Wear' module is used to develop an application for smartwatches. Both of the modules are packaged together for distribution through *Google*'s 'Play Store', and subsequently downloaded by the user on to their *Android* mobile phone; the 'Wear' application is automatically synchronised from the phone to the user's smartwatch (Developer.android.com, n.d. k). When creating the *Android Studio* project the minimum SDK for a 'Phone & Tablet' application was set to API 18: *Android* 4.3 (*Jelly Bean*), the earliest version of *Android* that can connect to *Android Wear* devices. The wearable application has a minimum SDK of API 20: *Android* 4.4 (*KitKat Wear*), allowing the app to target the broadest possible selection of wearable devices that are currently available in the marketplace (Developer.android.com., n.d. e).

### 6.3.2: Java Classes

Android applications are largely written in the Java programming language. While new APIs are continually developed and released by Google to increase the possibilities for feature development and to enhance performance levels of Android applications and the devices running the Android OS, a number of components remain fundamental to Android Studio project development.

**Activities**

The *RefWatch: Soccer Edition* application contains twenty-eight java classes, each of which extends the *Activity* class imported from the Android SDK's *android.app* package. Each activity is a component which represents a screen in the application and references a corresponding XML file containing the screen's intended layout design. For example, the *GoalTeamSelectionActivity* java class contains the source code which controls the application's behaviour when a user navigates to the *Team Selection Screen* in the *Goal* match event pathway, while the *goal_team_selection.xml* file contains the views and properties which comprise that particular screen's UI layout. The activity's UI layout is defined in the source code by calling the *setContentView()* method and passing the resource *R.layout.goal_team_selection* to the method as a parameter (see *Figure 14*). One activity is designated as the 'main' activity, which will be the first screen the user

accesses when the application is initially launched. For example, the *WearMainActivity* class in the 'Wear' module of the *RefWatch* Android Studio project is assigned the ability of main activity and category of launcher in the application 'Manifest' file (see *Figure 15*) (Developer.android.com. (n.d. c). The corresponding XML layout for the launcher screen is assigned by invoking the *setContentView()* method and passing the resource *R.layout.activity_wear_main* as a parameter.

```
@Override
public void onCreate(Bundle b){
    super.onCreate(b);
    setContentView(R.layout.goal_team_selection);//used to set the layout for the class
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);//used to maintain screen visibility
    findViewById(R.id.backButton).setOnClickListener((v) → {
            finish();//close the activity
    });
```

Figure 14: setContentView() and findViewById() methods.

```
<application
    android:allowBackup="true"
    android:icon="@drawable/icon"
    android:label="RefWatch: Soccer Edition"
    android:supportsRtl="true"
    android:theme="@style/AppTheme" >
    <activity
        android:name=".WearMainActivity"
        android:label="RefWatch: Soccer Edition" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".ExportActivity" android:screenOrientation="portrait" >

    </activity>
    <activity
        android:name="com.nononsenseapps.filepicker.FilePickerActivity"
        android:label="RefWatch: Soccer Edition"
        android:theme="@style/FilePickerTheme">
```

Terminal | 0: Messages | 6: Android Monitor | 3: Find | 4: Run | TODO

Session 'wear': running (today 08:51)

Figure 15: Manifest file showing .MAIN and .LAUNCHER designations.

An activity can be started, stopped, resumed and destroyed during its lifecycle. Managing the transition between the various activity states is essential to developing a functional application and fluid user experience. When a new activity is started, it is "pushed onto the back stack and takes user focus." When the user is finished interacting with the current activity, such as when they press a 'Back' button to navigate to a previous activity, for instance, it is removed from the stack and destroyed at which juncture the previous activity would resume. The activity lifecycle and the transitions between its various states are illustrated in Figure 16 (Developer.android.com., n.d. b).

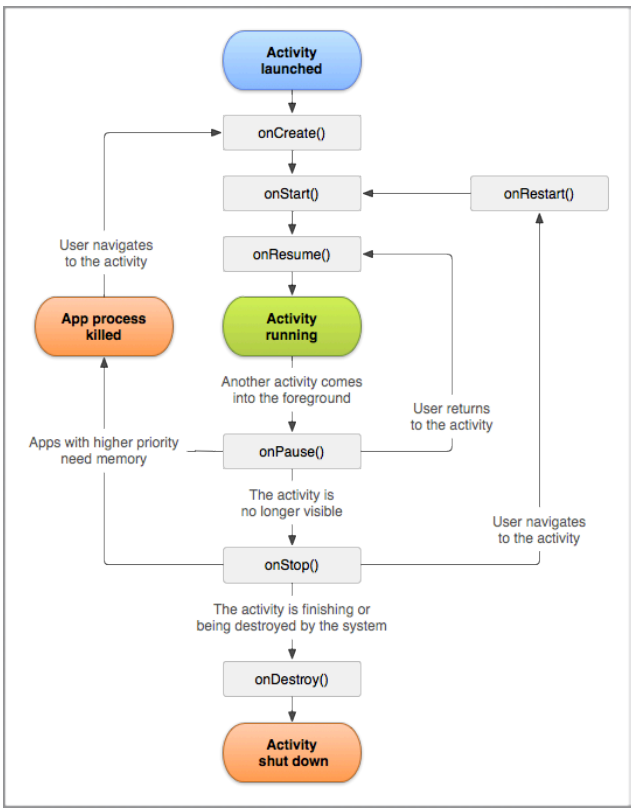



Figure 16: Diagram illustrating the Android activity lifecycle including transitions between activity states (Developer.android.com., n.d. b).

An activity can receive a variety of lifecycle 'callback' methods when there is a change in its state such as when it is starting, pausing, resuming or being destroyed. The callback methods, which allow work to be undertaken related to each change in activity state when implemented, include the following:

- **onCreate()**: This method is called by the system when the activity is created, and any components which are required by the activity are declared within the *onCreate() method*. The activity's corresponding UI layout XML file is set, moreover, by calling the *setContentView()* method within the *onCreate()* method.

- **onStart()**: This method is called when the activity is becoming visible to the user. It is followed by the *onResume()* method if the activity comes to the foreground and the *onStop()* method if it becomes hidden.

- **onPause()**: This method is called when the user leaves an activity and it moves into the background, although since they may return to the screen it does not necessarily indicate that the activity should be destroyed.

- **onResume()**: When an activity resumes upon the invocation of the *onResume()* method it can reacquire any necessary resources and resume any actions that were interrupted when paused.

- **onDestroy()**: This method is called before the activity is destroyed (Stackoverflow.com, 2014; Developer.android.com., n.d. b).

In the instance when an activity is paused or stopped, the state of the activity along with information about its member variables is retained in the activity object which is stored in the system's memory. Any interactions or changes in the activity are therefore kept in memory until the activity becomes visible when it returns to the foreground. If an activity object is vulnerable to destruction by the system when it needs to recover memory, however, it is necessary to call the onSavedInstanceState() method which saves information about the state of the activity (Developer.android.com., n.d. d).

**Intents**

Each activity is independent from the others, yet they work in conjunction with one another to form a cohesive user experience. When the user moves from one screen to another the corresponding activities are activated by *intents*, which are "asynchronous message(s) [used to] bind individual components to each other at runtime" like "messengers that request an action from other components". Intents are created by instantiating Intent objects which "define a message to activate either a specific component or a specific type of component" (Developer.android.com., n.d. d). An activity can be started by passing an Intent to the startActivity() method, for instance, as evidenced in Figure 17 depicting the 'moveToGoalPlayerNumberScreen()' method which controls user navigation from the *Goal Team Selection* screen to the *Goal Player Selection* screen. The Intent object contains information passed to it as a parameter which in the example snippet of source code is the name of the *GoalPlayerNumberActivity.class* activity which should be activated, while the *putExtra()* method is called in order to send an extra notifier designating which team's player list should be accessed when selecting the shirt number of the goalscorer. Finally, the *startActivity()* method is invoked with the Intent object 'i' passed as its parameter, which will result in the system starting the designated *GoalPlayerNumberActivity.class* activity (Developer.android.com., n.d. h).

```
//Method which navigates to the GoalPlayerNumberActivity and sends an indicator for which team
private void moveToGoalPlayerNumberScreen(String team){
    Intent i = new Intent(this, GoalPlayerNumberActivity.class);
    i.putExtra(TEAM, team);
    startActivity(i);
}//moveToGoalPlayerNumberScreen
```

Figure 17: moveToGoalPlayerNumberScreen() method.

**Bundles**

While an Intent is used to describe or signal the activity to be executed, passing data between the application's screens is facilitated by a component called a *Bundle*. An intent can contain a Bundle which enables extra data, referred to as *Extras*, to be sent along with the Intent object. When the user selects the period duration in the RefWatch application, for example, the onCreate() method in the CountdownTimerActivity class contains code implementation of the Bundle 'bn', which passes the users' period duration, number of periods, and interval duration selections from the preceding screens to the Main Match Screen as illustrated in Figure 18.

```
Bundle bn = getIntent().getExtras();//used to get the values sent with the intent that started the class.
if(bn != null){ //check whether the values sent with the intent is empty or not
    mDurMin = bn.getInt(MATCH_DUR);//get the period duration time that the user has chosen
    if(bn.containsKey(PERIODS)) {//check if the number of periods sent with the intent is empty or not
        tPeriod = bn.getInt(PERIODS);//get the number of periods that the user has chosen
    }//if
    if(bn.containsKey(HALF_TIME_INTERVAL)) {//check if the interval time sent with the intent is empty or not
        hPeriod = bn.getInt(HALF_TIME_INTERVAL);//get the interval time that the user has chosen
    }//if
}//if
db.newMatchItem(getHMS(mDurMin * MIN), tPeriod, getHMS(hPeriod * MIN), DateFormat.getDateTimeInstance().format(new Date()));//add new match record to the database
setCListeners();//call the method that make the textviews clickable and sort functionality for every text view
completed = false;//making this boolean false means that the match is not completed
}//onCreate
```

Figure 18: Example of a Bundle contained within an Intent object.

### 6.3.3: Additional Resources

Within the 'Wear' module of the Android Studio project a number of subfolders contain additional resources some of which are, as previously alluded to, automatically generated when the project is created.

**AndroidManifest.xml**

Stored at the root of the Android Studio project, the AndroidManifest.xml file is a useful tool which facilitates the identification and management of the various components that are required by the system to build and run the application. All of the application's components or activities are declared in the manifest file along with any permissions that the application

will require, such as use of a device's hardware functionality or the ability to read and write data to the system's memory. Furthermore, the manifest file contains fundamental information such as the package name, the application's name or label and version number, the activity which is accessed when the application is launched by the user, and a pathway to the application's dock icons. The manifest file is continually updated during development to account for all of the components which are required at runtime; failure to do so would result in a compilation error at runtime. The *AndroidManifest.xml* file for the *RefWatch: Soccer Edition* project contains all of the aforementioned information, including the necessary permissions for use of the smartwatch's vibration functionality and to read and write data to its file directory when the user chooses the destination at which to save exported match reports; a screenshot of the file is provided in Figure 19.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.christopherpatton.refwearprosocceredition" >

    <uses-feature android:name="android.hardware.type.watch" android:required="true"/>

    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
    <uses-permission android:name="com.google.android.c2dm.intent.REGISTRATION"/>
    <uses-permission android:name="android.permission.WAKE_LOCK" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/icon"
        android:label="RefWatch: Soccer Edition"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".WearMainActivity"
            android:label="RefWatch: Soccer Edition" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ExportActivity" android:screenOrientation="portrait" >

        </activity>
        <activity
```

Figure 19: Android Manifest file contents.

## User Interface Layouts

The application's visual presentation was defined in Android Studio using XML files. The user interface is essentially comprised of 'views', which are objects derived from the Android SDK's *View* class. A view, which controls an assigned space which has been defined within an activity's window, responds to user interactions such as button clicks or swipe gestures. The organisation of the user interface layouts, colour scheme, menus and styling were largely implemented by utilising a selection of ready-made views known as 'widgets' which are provided by Android Studio. A number of views deployed in the *RefWatch* application's layouts, including image buttons, progress bars, radio groups,

textviews and listviews, provide visual and interactive elements to the screens. Each component added to an XML layout is represented by a declared java variable in the layout file's corresponding activity. In Android Studio's layout 'Design' tab, widgets can be dragged from the Palette and dropped directly on to an empty view configured to represent the screen dimensions of one of a range of selectable virtual devices. The hierarchical relationship between parent, child and sibling views is visible in the 'Component Tree' while the properties of each view can be altered, moreover, in the 'Properties' panel which enables quick customisation of everything from view IDs to background colour. The layout editor's 'Design' mode window is depicted in Figure 20. The layout editor also provides a 'Text' mode which enables view object implementation using XML code. When a custom border was required to surround the timer textviews inhabiting many of the match event pathway screens, for instance, the solution was written in an XML layout file entitled *countdowntimer_textview_border.xml* and stored in the drawable folder (see Figure 21).

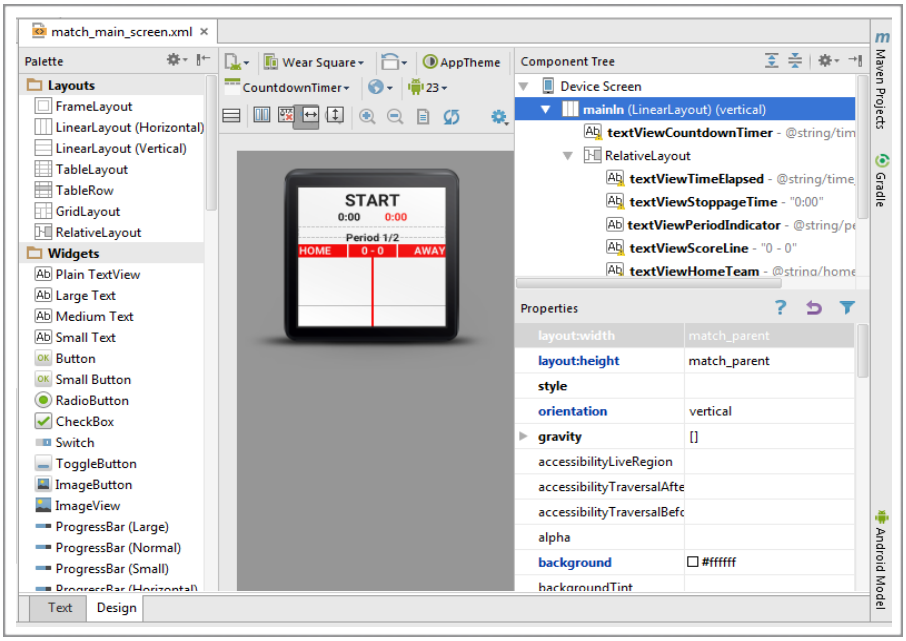


Figure 20: Android Studio layout editor in 'Design' mode.

match_main_screen.xml × countdowntimer_textview_border.xml ×

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
android:shape="rectangle" >

    <!-- view background colour -->
    <solid
        android:color="#ffffff" >
    </solid>

    <!-- view border colour and width -->
    <stroke
        android:width="1dp"
        android:color="#000000" >
    </stroke>

    <!-- The radius makes the corners rounded -->
    <corners
        android:radius="2dp" >
    </corners>

</shape>
```

Figure 21: XML code which implements a custom textview border.

The layout XML files are stored in the *layout* folder, which in turn is a sub-folder of the *res* folder. The res folder also contains the *values* folder, the aforementioned *drawable* folder and the *raw* folder, which store the *strings* and *style* XML files, image files, and audio files respectively, each of which were utilised during the application's implementation. The *drawable* folder, for example, contains all of the images used within the application including the launcher icon, splash logo, image button backgrounds and match event icons, while the *raw* folder contains an mp3 audio file containing the air horn sound effect which can be selected by the user to signal the end of a match period's regulation duration (Zapsplat.com, n.d.). To reference these resources in the application's source code the Android SDK tools generate a unique resource ID. For instance, the image file named *goal_icon.png* is referenced using the resource ID *R.drawable.goal_icon* (Developer.android.com., n.d. d). Storing the resources in the *res* folder apart from the java source code which governs the application's behaviour allows the layout designs to be maintained separately, and provides the flexibility to create alternative resources and layout designs when an application will be deployed on numerous devices with disparate screen dimensions or orientations. Defining strings in XML, for example, allows any text used in UI layouts to be translated to different languages. When a corresponding string XML file for each language is saved in the *values* folder, the system can apply the appropriate language strings to the UI according to user's chosen settings (Developer.android.com., n.d. d).

**Gradle Scripts**

Within the *Gradle Scripts* subfolder exist the gradle build files for both the Mobile and Wear modules of the project. These files contain information on the project's configuration such as the minimum and target SDK version, project version name and code, and a list of dependencies, all of which is required by the gradle build system. Before packaging and publishing the *RefWatch* Android Wear application, for instance, it was necessary to ensure that the *wearApp project (':wear')* dependency was declared in the *build.gradle (Module: mobile)* file (Developer.android.com., n.d. k). The gradle build system converts the Android Studio project's source files into their necessary formats and packages them as an *APK* file which is used to install the application on the target Android-powered device (Stackoverflow.com, 2015).

## 6.4: Core Functionality

Due to the expansive nature of the development project it is the implementation of the functionality required to fulfill key requirements such as timekeeping, selecting and recording match events, storing time stamped event data in a database, and retrieving and exporting match reports in the medium of a text document, which is the chief focus of the processes recounted in the following subchapters.

### 6.4.1: Multiple Timers

Although the contributors to the requirements gathering focus group requested a duel timer capable of tracking the period duration and accumulated stoppage time of a soccer match, it was determined that the application's timer function would draw on and enhance the design utilised by the *Spintso Referee Watch* which employs three timers. A video demonstrating the watch's functionality was studied in order to gain an understanding of how it could be replicated and then improved upon (YouTube, 2015 b). The timers implemented in the solution smartwatch application, and their functionality, are as follows:

- **Countdown timer**: The countdown timer displays the number of minutes selected by the user on the *Period Duration Selection* screen, counting down until it reaches zero. The countdown timer is paused when the referee navigates away from the *Main Match Screen* to record match events or a stoppage in play.

- **Stoppages timer**: The stoppages timer displays the stoppage time accumulated when the user is recording match events. It is paused when the user returns to the *Main Match Screen* and the countdown timer resumes.

- **Total elapsed duration timer**: The total elapsed duration timer runs in synchronicity with the countdown timer, counting up from zero and running continuously. It continues to run when the user accesses the *Match Event Selection Menu* screen and its subsequent pathways. It therefore displays the sum of the time which has elapsed in the countdown timer and the time which has accumulated in the stoppage timer.

The application's timers and other *Main Match Screen* functionality are underpinned by the source code stored in the *CountdownTimerActivity.java* class of the Android Studio project. The period duration, number of periods and, where applicable, interval duration are passed in to the activity via a Bundle and converted to the required timer textviews' format of minutes and seconds (mm:ss) by the *getHMS()* method as shown in Figure 22. Separate timers and their functions were instantiated to represent the necessary countdown, total elapsed duration and stoppages timers.

```
//======================
//Method to convert the time passed in as long to minutes and seconds and format as 00:00
public static String getHMS(long diff) {
    long left = diff;//used to store the value passed to the method as a parameter

    int mins = (int)(left/MIN);//format minutes
    left = left - mins * MIN;//format minutes

    int sec = (int)(left/SEC);//format seconds

    return (mins<=9?"0"+mins:mins)+" : "+(sec<=9?"0"+sec:sec);//return the formatted time
}//getHMS
```

Figure 22: getHMS() method.

Upon appraisal it was determined that the *Spintso Referee Watch* design has a shortcoming: the progression from one period of play to the interval between periods occurrs automatically when the countdown timer reaches zero. Such a progression did not account for the nature of a soccer game, which often requires more than the accumulated stoppage time at the end of a half or period of play. It was decided, therefore, that the aforementioned progression should be controlled by the user. When the countdown timer reaches zero, the total elapsed duration timer continues to accumulate additional time until the user clicks on its textview to signal their desire to progress to the interval or end the game if the final period has reached its conclusion. This was accomplished by instantiating

an additional ‘unlimited timer’ through the *AfterHalfTimer()* method as shown in Figure 23. To alert the user that the countdown timer has reached zero and create awareness that they can prompt progression to the subsequent interval an audio or vibration alert is activated, while the background colour of the *Main Match Screen* was altered programmatically to be displayed as green during additional time and orange during the interval between periods; the *setRedBackground()* method which is responsible for controlling background colour changes is shown in Figure 24.

```
//====================================
//Unlimited timer method(elapsed timer)
public void AfterHalfTimer(){
    if(resume) {//if halftime/interval
        setCListeners();//make the text view clickable
    }
    if(completed) {//if match completed
        setCListeners();//make the text view clickable
    }
    countDownTimerTextView.setText(getHMS(00));//used to set the value of the countdown timer to 00:00
    cnTimer = new Timer();//create the unlimited timer for elapsed text view
    cnTimer.scheduleAtFixedRate(() -> {
            time2++;//increase the textview indicator of elapsed timer
            runOnUiThread(() -> {
                    elapsed.setText(getHMS(time2*SEC));//set elapsed textview value to time2 indicator formatted by getHMS method(elapsed timer)

                });

        },0, SEC);

}//AfterHalfTimer
```

Figure 23: AfterHalfTimer() method.

```
//
//This method manages the background colour during the match
public void setRedBackground(int color){

    LinearLayout mlm = (LinearLayout)findViewById(R.id.mainln);//used to find the layout of both team's lists
    ListView list1 = (ListView)findViewById(R.id.listViewHomeTeam);//used to find the list view of home team
    ListView list2 = (ListView)findViewById(R.id.listViewAwayTeam);//used to find the list view of away team

    int colorOrange = Color.parseColor("#F3FD8A");//used to read the orange colour code in java
    int colorWhite = Color.parseColor("#ffffff");//used to read the white colour code in java
    int colorGreen = Color.parseColor("#82F4C0");//used to read the green colour code in java

    if(color == 1){//check if the code sent with method is 1
        list1.setBackgroundColor(colorWhite);//used to set the background colour of the home team list view to white
        list2.setBackgroundColor(colorWhite);//used to set the background colour of the away team list view to white
        mlm.setBackgroundColor(colorWhite);//used to set the background colour of the layout that holds the list views to white
    }else if(color == 2){//check if the code sent with method is 2
        list1.setBackgroundColor(colorOrange);//used to set the background colour of the home team list view to orange
        list2.setBackgroundColor(colorOrange);//used to set the background colour of the away team list view to orange
        mlm.setBackgroundColor(colorOrange);//used to set the background colour of the layout that holds the list views to orange
    }else{//check if the code sent with method is 3 or anything else
        list1.setBackgroundColor(colorGreen);//used to set the background colour of the home team list view to green
        list2.setBackgroundColor(colorGreen);//used to set the background colour of the away team list view to green
        mlm.setBackgroundColor(colorGreen);//used to set the background colour of the layout that holds the list views to green
    }//if/else/else
}//setRedBackground
```

Figure 24: setRedBackground() method.

### 6.4.2: Recording Match Events

The user can record match events by accessing the *Match Event Selection Menu* screen after starting the countdown timer in any given period of play. The menu was implemented as a scrollable listview which was populated with an array of Strings containing the following menu option names:

- Goal
- Penalty Kick
- Substitution
- Disciplinary Action
- Abandon Match
- Quit

Access to the *Match Event Selection Menu* is controlled by the *setCListeners()* method in the *CountdownTimer.java* class. When the user accesses the *Main Match Screen* for the first time the countdown timer textview displays the text ‘START’ which, when clicked, starts the countdown timer and elapsed timer. An additional click of the timer textview provides access to the *Match Event Selection Menu*. Controlling user access to the menu in such a manner was accomplished by declaring a boolean variable, *firstTimeClicked*, which was initialised to ‘true’ and included in an ‘if’ statement to determine if the textview had been clicked for the first time. On the subsequent click the boolean’s value is set to ‘false’. The method is shown in Figure 25. A ‘switch’ statement was utilised in the *MatchEventSelectionActivity* class to determine which activity should be started when the user makes their selection from the list of available options as shown in Figure 26. If they choose either the ‘Abandon Match’ or ‘Quit’ options a dialog is displayed which prompts the user to make a ‘Yes’ or ‘No’ choice. This feature was implemented to prevent unnecessary abandonments or the erroneous loss of match data.

```
//this method manages the text view clicks map
public void setCListeners() {
    countDownTimerTextView.setOnClickListener(new View.OnClickListener() {//countdown timer click functionality
        @Override
        public void onClick(View v) {
            if(firstTimeClicked) {//check if it's been clicked before
                //== If its first click, start all timers
                setElapsedMatchDurTime();//call method that starts the main match countdown timer
                firstTimeClicked = false;//set as clicked
             }else {//== if countdown timer has been clicked before

                    startActivity(new Intent(getApplicationContext(), MatchEventSelectionActivity.class));//go to MatchEventSelection Activity

            }//if/else
        }//onClick
    });
```

Figure 25: setCListeners() method.

```
switch (pos){//check the item clicked
    case 0: finish();; return;
    case 1: startActivity(new Intent(getApplicationContext(), GoalTeamSelectionActivity.class)); return;//go to GoalTeamSelectionActivity
    case 2: startActivity(new Intent(getApplicationContext(), PenaltyKickActivity.class)); return;// go to PenaltyKickActivity
    case 3: startActivity(new Intent(getApplicationContext(), SubstitutionTeamSelectionActivity.class)); return;//go to SubstitutionTeamSelectionActivity
    case 4: startActivity(new Intent(getApplicationContext(), DisciplinaryTeamSelection.class)); return;// go to DisciplinaryTeamSelection
    case 5:
        new AlertDialog.Builder(MatchEventSelectionActivity.this).setMessage("Confirm Abandon Match?")//make choice confirmation dialog
                .setPositiveButton("Yes", (dialog, which) -> {

                        db.updateAbandoned(CountdownTimerActivity.elapsed.getText().toString());//calling this method to update abandoned values for the match
                        Intent intent = new Intent(getApplicationContext(), MenuActivity.class);//go to MenuActivity
                        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);//close all other activities
                        startActivity(intent);// start MenuActivity
                        finish();//close this activity
                    }).setNegativeButton("No", (dialog, which) -> {

                dialog.dismiss();//close the dialog
            }).create().show();//show the dialog
        return;
    case 6:
        new AlertDialog.Builder(MatchEventSelectionActivity.this).setMessage("Confirm Quit Match?")//make choice confirmation dialog
                .setPositiveButton("Yes", (dialog, which) -> {
```

Figure 26: ‘switch’ statement which facilitates navigation in the Match Event Selection menu.

Each of the match event options provided in the listview grants the user access to a different pathway of screens. Each pathway begins with a *Team Selection* screen which enables the user to select the team - “HOME” or “AWAY” - to which the event corresponds. When the user clicks a button on any of the application’s screens the corresponding activity utilises the *setOnClickListener()* method in conjunction with the *findViewById()* method, which assigns an instance of onClickListener to an appropriate button object, for instance, which has been declared and initialised in the java class and assigned to a corresponding view in a layout file. For example, in the *GoalTeamSelectionActivity.java* class the user can click on the “Back” button to navigate to the previous screen, or the “HOME” or “AWAY” buttons to select a team and move to the *Goal Player Number Selection* screen. The *onClick()* method is invoked for each button within the *onCreate()* method (see Figure 27). Since one listview populated with number options is used for both teams in the subsequent screen, moreover, it was necessary to implement a method -

*moveToGoalPlayerNumberScreen()* - which accepts an indication of the chosen team as an argument. The subsequent *Goal Player Number Selection* screen uses an ArrayList of String values to populate a listview with player number options. It was decided that the inclusion of the options 'N/A' and 'O.G.' would account for games played at the grassroots level with no discernible player shirt numbers and the event of an own goal being scored respectively.

```
public void onCreate(Bundle b){
    super.onCreate(b);
    setContentView(R.layout.goal_team_selection);//used to set the layout for the class
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);//used to maintain screen visibility
    findViewById(R.id.backButton).setOnClickListener(new View.OnClickListener() {//used to set clicking functionality for Back button
        @Override
        public void onClick(View v) {
            finish();//close the activity
        }
    });
    findViewById(R.id.coachButton).setOnClickListener(new View.OnClickListener() {//used to set clicking functionality for Home button
        @Override
        public void onClick(View v) {
            moveToGoalPlayerNumberScreen(HOME);//navigate to player number selection screen for home team
        }//onClick
    });
    findViewById(R.id.awayButton).setOnClickListener(new View.OnClickListener() {//used to set clicking functionality for Away button
        @Override
        public void onClick(View v) {
            moveToGoalPlayerNumberScreen(AWAY);//navigate to player number selection screen for away team
        }//onClick
```

Figure 27: Setting buttons to 'clickable' in the onCreate() method.

When the user has progressed through the screens of their chosen match event pathway they are redirected to the *Main Match Screen* and the countdown timer recommences. Their chosen match event is also displayed in a listview designated to either the home or away team, with an icon representing the type of event and the player number(s) associated with the event; a list of icons and their associated events is listed in the *RefWatch: Soccer Edition User Guide* (see *Appendix 9: User Guide*). A distinct *TeamListAdapter* class was implemented to manage the actions associated with populating the listviews containing each team's match event record on the *Main Match Screen* and, furthermore, formatting how the information is displayed for ease of accessibility during a match.

Data errors were minimised in the *Substitution* match event feature by preventing a user from choosing the same player number for both 'Outgoing' and 'Incoming' player selections; if the user selects number two in the *Outgoing Player Selection* screen, for example, then the number two option is removed from the subsequent *Incoming Player Selection* screen. It was also necessary to remove a player's number from the *Player Number Selection* screen listviews for each associated match event when that player had been removed from the match as the result of a substitution or due to a dismissal after

receiving a red card. This was achieved by declaring ArrayLists in the *CountdownTimerActivity* class to store players who had received yellow cards, red cards, or had been selected as an outgoing substitute (see Figure 28).

```
//Array lists for both teams used to store players removed by substitution in order to remove the player from next substitution selection
public static ArrayList<String> outPlayerhome,outPlayeraway;

//yellowCardPlayerhome,yellowCardPlayeraway array lists for both teams used to store players that receive a yellow card in order to give them a red card with second yellow card
//redCardPlayerhome,redCardPlayeraway array lists for both teams used to store players that receive red cards to remove them from the match completely in any match event's player
public static ArrayList<String> yellowCardPlayerhome,yellowCardPlayeraway,redCardPlayerhome,redCardPlayeraway;

findViewById(R.id.rcInfringementProceedButton).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {//used to set clicking functionality for Proceed button
        if(DisciplinaryTeamSelection.TEAM.equals("home")){//check if team is home
            CountdownTimerActivity.redCardPlayerhome.add(String.valueOf(DisciplinaryPlayerNumberSelection.chosen));//add value to red card list to avoid using players again
```

Figure 28: Code implemented to prevent duplicate player number selection.

### 6.4.3: Database

The creation and implementation of an SQLite database was necessary to locally store the match data which forms the basis of the exportable match report function. Due to the required complexity of the database it was decided to create a database file using the open source software ‘DB Browser for SQLite’ (see Figure 29), and subsequently import the file as an asset to Android Studio. As with other aspects of the application’s implementation much of the database code which was adapted and the guidance to implement it was sourced from stackoverflow.com and a variety of video tutorial series hosted on youtube.com and udemy.com. The *DataBaseHandler.java* class creates the database, its tables and associated data types from the *matches.db* file stored in the *assets* folder - which is itself a sub-folder of the *res* folder - and contains methods, moreover, which insert new match event data items in the database’s tables for future retrieval.

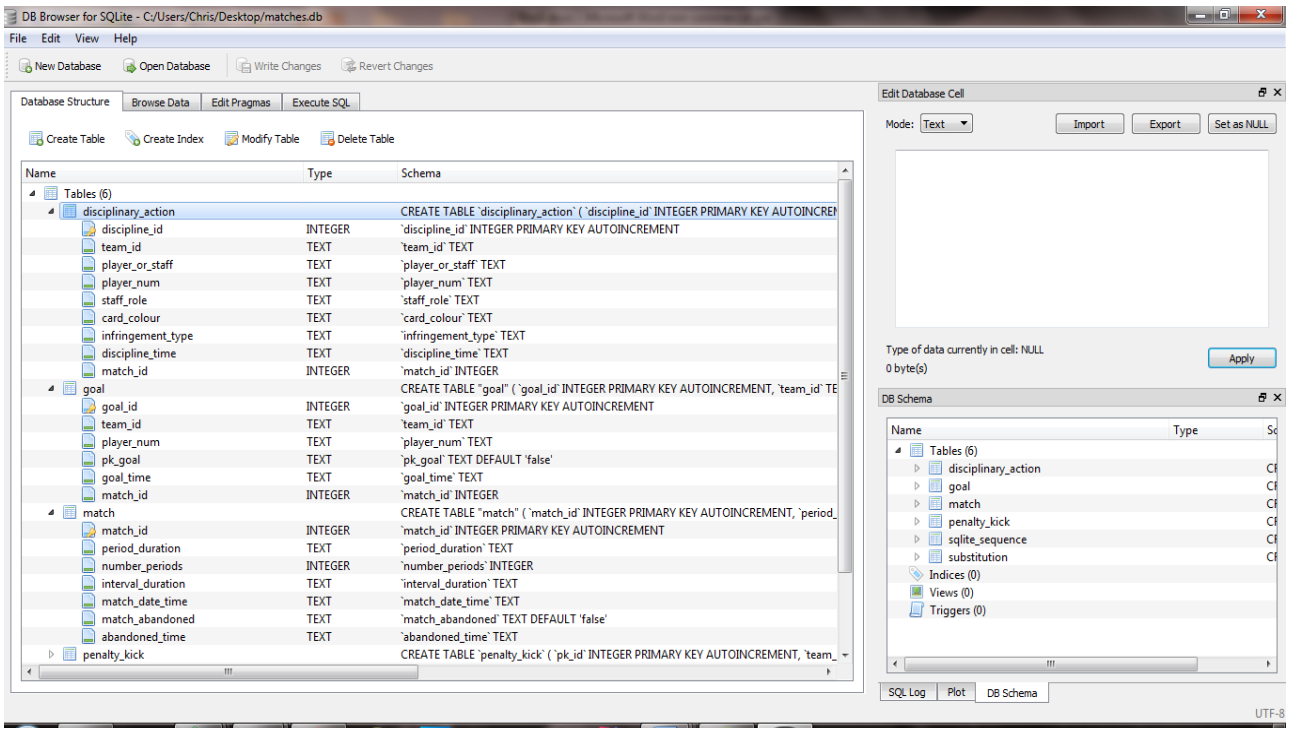



Figure 29: Creating database file using DB Browser for SQLite software.

The *copyDataBaseFromAsset()* method shown in Figure 30 copies the database from the *matches.db* file contained in the *assets* sub-folder and the *openDataBase()* method subsequently opens it. The *onCreate()* method, which accepts an SQLiteDatabase as a parameter, then creates the tables. The *onUpgrade()* method replaces older versions of the database with available newer versions, while the *newMatchItem()*, *newGoalItem()*, *newPenaltyItem()*, *newSubstitutionItem()* and *newDisciplinaryItem()* methods add new data items for each match event type to the database's corresponding tables; Figure 31 shows the *newGoalItem()* method. When a user selects the 'Quit' option in the *Match Event Selection Menu* all saved match event data must be deleted from the relevant match's record in the database. This action is handled by the *deleteMatch()* method, shown in Figure 32. When a match is abandoned, however, the match event data up until the time of abandonment is retained in the database, an action which is handled by the *updateAbandoned()* method. An instance of *DataBaseHandler* was declared in each activity which processes match events to facilitate storage of the new data items in the database, while the match report file containing all of the data items for a given match are created by the *generateReports()* method which also formats the report and generates a dialog to display it in the *Reports* screen accessible from the application's *Main Menu* (Schoolsteps.in, 2015; Sheraz and Dichone, 2015; Stackoverflow.com, 2013; Stackoverflow.com, 2012; Yoda Learning, 2015; YouTube, 2015 a).

```
//This method copies the database from assets
 public void CopyDataBaseFromAsset() throws IOException {

     InputStream myInput = myContext.getAssets().open(DATABASE_NAME);

     // Path to the just created empty db
     String outFileName = getDatabasePath();

     // If the path doesn't exist first, create it
     File f = new File(myContext.getApplicationInfo().dataDir
             + DB_PATH_SUFFIX);
     if (!f.exists())
         f.mkdir();

     // Open the empty db as the output stream
     OutputStream myOutput = new FileOutputStream(outFileName);

     // Transfer bytes from the inputfile to the outputfile
     byte[] buffer = new byte[1024];
     int length;
     while ((length = myInput.read(buffer)) > 0) {
         myOutput.write(buffer, 0, length);
     }//while

     // Close the streams
     myOutput.flush();
     myOutput.close();
     myInput.close();

 }//CopyDataBaseFromAsset
```

Figure 30: copyDataBaseFromAsset() method.

```
//Method to add a new Goal record into the database
public void newGoalItem(String team_id,String player_num,String goal_time) {
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = null;
    try {
        cursor = db
                .rawQuery(
                        "SELECT * FROM "+TABLE_Match, null);
        cursor.moveToLast();
        int match_id = cursor.getInt(0);
        cursor.close();
        String ROW1 = "INSERT INTO goal (team_id, player_num, goal_time, match_id) Values ('"+team_id+"', '"+player_num+"', '"+goal_time+"', '"+match_id+"')";
        db.execSQL(ROW1);


    }catch(Exception e){
        e.printStackTrace();
    }finally {

    }//try-catch-finally

}//newGoalItem
```

Figure 31: newGoalItem() method.

```
//Method which deletes all details of a match
public void deleteMatch(int match_id) {
    SQLiteDatabase db = this.getReadableDatabase();

    try {

        String ROW1 = "DELETE FROM "+TABLE_Match+" WHERE match_id= '"+match_id+"'";
        String ROW2 = "DELETE FROM "+TABLE_Goal+" WHERE match_id= '"+match_id+"'";
        String ROW3 = "DELETE FROM "+TABLE_Penalty+" WHERE match_id= '"+match_id+"'";
        String ROW4 = "DELETE FROM "+TABLE_Sub+" WHERE match_id= '"+match_id+"'";
        String ROW5 = "DELETE FROM "+TABLE_Disc+" WHERE match_id= '"+match_id+"'";
        db.execSQL(ROW1);
        db.execSQL(ROW2);
        db.execSQL(ROW3);
        db.execSQL(ROW4);
        db.execSQL(ROW5);

    }catch(Exception e){
        e.printStackTrace();
    }finally {


    }//try-catch-finally

}//deleteMatch
```

Figure 32: deleteMatch() method.

### 6.4.4: Exportable Match Reports

The match data saved in the database can be exported to a text file once the user has successfully completed a match or chosen to save the data from an abandoned match. When they return to the application's *Main Menu* screen the user can select the *Reports* menu option to access a listview containing a list of completed match data distinguishable by the kickoff date and time. By selecting a match from the available list and clicking the 'Export' button the user can access another screen from which they can choose the folder in their wearable device's file directory where they wish to save the text file containing their match report. The *Reports* screen layout required a corresponding *ExportActivity* class in Android Studio, while the *DataBaseHandler* class discussed in the previous 'Database' sub-chapter handled the retrieval of match data from the SQLite database and subsequent creation of the text file to contain the information. The source code and guidance used to implement the file picker library created and managed by the *ExportActivity* class was sourced from the website Github (GitHub, 2016).

## 6.5: Challenges Encountered

Although a number of challenges were encountered during the implementation process, the iterative approach taken to the application's development during each implementation cycle enabled frequent testing of deployed solutions using Android Studio's inbuilt emulator. When the emulated application's behaviour did not meet expectations, implemented source code could be refined or revised as necessary and then tested. Testing during the implementation phase therefore largely took the form of regression testing, which required that the functionality successfully implemented had to be checked to ensure performance had not been adversely affected by subsequent implementations or revisions. After the application's deployment a dedicated testing phase ensued which is documented in the subsequent chapter.

After packaging the application and installing it on a Sony *SmartWatch 3* device, a number of challenges revealed themselves which had not been hitherto considered and which required further research and code implementation to overcome. For instance, by default *Android Wear* applications disappear into the background after a brief period of user inactivity. In the context of a soccer referee officiating a match, however, the user requires that the screen is always visible and 'glanceable'. It was therefore necessary to add a

‘Wake Lock’ permission to the Manifest File to ensure that the application is alway visible on the watch’s screen and a subsequent line of code in each applicable activity to activate the feature (see Figure 33) (Developer.android.com., n.d. i). Furthermore, by default any applications residing on the watch’s connected smartphone device can send event notifications to the watch. For example, the phone’s email application will send a notification that a new email has been delivered to the user’s inbox. The notification appears on the watch’s face in the form of a card, which the user can dismiss by swiping the card from left to right until it disappears from the screen. Receiving such notifications would be distracting to a referee while officiating a soccer match, however, and it was therefore necessary to find a solution which would prevent these interruptions. While a programmatic solution could not be uncovered among the existing Android APIs, it was possible to access the Android Wear settings on the connected phone and add applications to a ‘Block Notifications’ list which negated the threat of interruptions while running the *RefWatch: Soccer Edition* application.

```
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);//used to maintain screen visibility
```

Figure 33: Code implemented in Activities which maintains screen visibility.

## 6.6: Summary

The application’s implementation process was extremely challenging and time consuming. It required strict project and time management skills, abundant research and dedication and perseverance to overcome the challenge of creating an Android Wear application with what was initially little more than a rudimentary understanding of development on the Android platform. Due to time constraints, however, the scope of the implementation phase had to be narrowed to encompass what were considered to be the most important functional requirements while peripheral functionality was consigned to the status of ‘future developments’. The expansive nature of the project and associated time constraints required, moreover, that the discussion of the application’s software component implementation is fairly restrictive in nature. The application’s source code, details of which are included in the Appendices (see *Appendix Ten*), contains additional comments that further explain the functionality of the implemented components.

# 7: Testing and Evaluation

## 7.1: Introduction

Software testing, the process of evaluating a software system or the components of which it is comprised, is necessary in order to establish if the system's requirements have been fulfilled (Tutorialspoint.com., 2016 a). The process of software testing, which involves the execution of a program or application with the aim of identifying bugs in the code which could lead to errors and sub-optimal performance levels or unfulfilled requirements, is outlined and detailed in the ensuing chapter. The results of in-house functional testing are detailed and examined, while the responses to a testing questionnaire distributed to a focus group of end users are also presented and discussed.

## 7.2: Testing

Software testing assumes various guises and can be conducted during every phase of the Software Development Life Cycle. In the analysis phase, for example, the verification of the project's requirements can be considered as a form of testing. Similarly, reviewing and refining the application's design during the design phase is considered to be a form of testing (Tutorialspoint.com, 2016 b). Throughout the development of the *RefWatch: Soccer Edition* application testing has been conducted in an incremental and iterative manner in line with the project's adopted Agile software development methodology. Each development cycle, or sprint, has involved the development and implementation of application features and functionality followed by a review or testing process. Following the implementation of a new section of Java code, for example, the expected functionality could be tested immediately by using an *Android Virtual Device* (AVD) in the *Android Studio* emulator, thereby facilitating responsive troubleshooting procedures which involved the identification, isolation and correction of bugs and ensured rapid fulfillment of requirements. Such an approach to testing enabled the deployment of an Alpha version of the software in advance of a dedicated testing and application validation phase.

The testing and validation of the deployed Alpha version of *RefWatch: Soccer Edition* assumed a twofold approach: in-house functionality testing, and the release of the software to a dedicated testing focus group. The testing of the application was predominantly conducted using a Sony *SmartWatch 3* device running on the *Android Wear* operating system; technical specifications of the chosen testing device are listed in

Appendix Six. The Sony *SmartWatch 3* was chosen for testing due to it's large square screen form factor, which was deemed optimal for soccer referees in comparison to a round screen form factor, in addition to a number of features such as built-in GPS which are not currently available with other devices in the marketplace. Due to the unavailability of a round-faced smartwatch device, however, an *Android Virtual Device* (AVD) was configured in *Android Studio* to conduct necessary testing of the application's layouts and functionality when using an 'Android Wear Round' format smartwatch.

### 7.2.1: In-house View and Functionality Testing

The aforementioned iterative software development model adopted for this project has reduced the dependency of testing on the deployed Alpha version of the software. It remained necessary, however, to determine if the implemented design and functionality had met the project's requirements. The in-house view testing process involved executing the software and systematically loading each of the application's screens before determining if the layouts were arranged as expected on both square and round format devices and met all requirements. The results of view testing are listed in Appendix Seven. The in-house functionality testing assumed the form of cataloguing each of the application's functions and developments, and an extensive combination of possible user commands and pathways through the system, before systematically examining if the application functioned as required. The testing procedures involved manually executing the software and its numerous functions by simulating a number of scenarios that could be commonly encountered by a referee during the course of a soccer match. A test plan was formulated which involved a number of steps:

- Determining the desired functionality that the application should perform in each test.
- Recording the testing output data.
- Comparing the desired outcome with the actual results of the test cases.

The results of the in-house testing phase are illustrated in Appendix Seven. The conducted functionality tests did not reveal any bugs or errors. While the testing procedures have been thorough and extensive, however, it is possible that there may be a number of paths throughout the application, scenarios or actions which were not executed or that have been overlooked. Ongoing maintenance of the application will therefore be essential in order to rectify any bugs or errors which may be identified following the application's release.

### 7.2.2: Focus Group Testing

Since only six respondents from the online focus group used during the requirements gathering phase of the project were currently in possession of a smartwatch device and, moreover, due to time constraints associated with the project's fulfillment deadline, it was determined that the formation of an additional focus group with direct access to the Sony *SmartWatch 3* testing device would be necessary. The testing focus group was comprised of five soccer enthusiasts with extensive experience of playing the sport at both the youth and adult amateur level; one of the respondents had experience of officiating at the grassroots level.

A questionnaire, which contained a number of questions and defined scenarios for the testers to attempt, was distributed to the testing focus group. The results of the questionnaire were used to ascertain if the system's core requirements were met by the deployed version of the *RefWatch: Soccer Edition* application. Each tester was supplied with the watch and a user guide (see Appendix Nine) which contained a combination of written and visual guidance to complete the following assigned tasks:

1. Enter the *Options* menu and select the vibration alert setting.
2. Select the *Match* option from the main menu, and proceed using the following match duration parameters:

• Period Duration: 5 minutes
• Number of Periods: 2
• Interval Duration: 1 minute

3. Upon accessing the *Main Match Screen*, the users were tasked with starting the match timers and completing the following match event registrations:

- Register a goal.
- Register a penalty kick.
- Register a substitution.
- Register a yellow card infringement to a player.
- Register a red card infringement to a coach.

4. Upon the conclusion of the match the user was requested to exit from the match, return to the *Main Menu* and export a report of their saved match data from the *Reports* menu option.

5. The users were asked to rate the difficulty of completing each task on a scale ranging from one to five, according to the following criteria:

- 1: Very difficult
- 2: Somewhat difficult
- 3: Neither difficult or easy
- 4: Somewhat easy
- 5: Very easy

The average user difficulty rating for each task, and a corresponding overall average user rating for the application, can be viewed in Table 7. User feedback was overwhelmingly positive, as evidenced by the aggregate average user rating of 4.9, and is testament to a strict level of adherence to established User Experience and User Interface design principles throughout the application's development.

| Task No. | Task Description | Average User Rating |
|---|---|---|
| 1 | Enter *Options* menu and select vibration alerts setting | 5 |
| 2 | Enter *Match* and select match duration settings | 5 |
| 3 | Register a goal | 5 |
| 4 | Register a penalty kick | 4.6 |
| 5 | Register a player substitution | 5 |
| 6 | Register a yellow card infringement to a player | 4.6 |
| 7 | Register a red card infringement to a coach | 5 |
| 8 | Enter *Reports* menu and export match report | 5 |
| **Aggregate Average User Rating** | | **4.9** |

Table 7: Testing focus group average user ratings for application 'ease of use' tasks.

In addition to completing scenario-based functionality tests, each user was asked to provide a rating which corresponded with their level of agreement with a number of written statements regarding the application's design and functionality; the statements were derived from the user stories collected during the project's problem analysis and

requirements gathering phase (see *3.3.3: User Stories*). The users were also invited to provide additional comments on each statement, and recommendations for improvement to the deployed version of the application. Each user's individual questionnaire responses, which are included in Appendix Eight, were analyzed and taken into consideration for inclusion in the *Recommendations* listed in the subsequent chapter.

# 8: Conclusion and Recommendations

## 8.1: Introduction

The following chapter provides an overall appraisal of the project and details where objectives and requirements have been realised or left unfulfilled. The merits of the project and its accomplishments are examined and discussed, alongside a critical evaluation of each stage of the project's research and development processes. Finally, a number of recommendations for future feature development and enhancement are presented.

## 8.2: Project Overview

The project has encompassed each stage of the Software Development Life Cycle - research and analysis, design, implementation, testing and evaluation - and has provided an invaluable insight to the intricacies and challenges associated with a significant software development undertaking. The project has been formulated around the conceptual development of a wearable game management solution for grassroots soccer referees in an effort to replicate the technological innovation which is evident at the highest echelons of the sport. Following substantial background research into the role of a soccer referee and the capabilities of wearable technology, the project's overarching aim was crystalised and the development of a smartwatch game management application which would enable grassroots soccer referees to track time, record match events and, furthermore, export post-match game reports with the goal of streamlining the processes involved in an official's post-match administrative duties, was presented as the solution to the problems commonly experienced by lower-level referees (see *2.2: Problems at the Grassroots Level*). A number of objectives were defined to guide and inform the project's overall progress towards the successful fulfillment of the project's aim (see *1.4: Project Objectives* and *1.3: Project Aim*); a discussion of the project's objectives and requirements follows in the subsequent sub-chapter.

The average timeframe for development of a mobile application is 18 weeks (Kinvey.com, 2013). Following the analysis and preliminary conceptual design phase, which lasted approximately 4 weeks and involved liaising with key stakeholders to determine the application's functional requirements, evaluating the project's potential risks and ethical considerations, and producing a business case for the proposed application, the duration for the remainder of the project can be approximated as 14 weeks. Analyzing, designing,

implementing, testing and deploying an *Android Wear* app in such a restricted timeframe represented a challenging undertaking which demanded significant effort, particularly for a developer possessing a rudimentary knowledge in many aspects of *Android* application development. The number of hours committed to the project in its entirety, while difficult to calculate accurately, is estimated at between 800 and 1000 hours.

## 8.3: Results Assessment

Despite the restricted development timeframe it was determined that the release of a fully functional version of the application, capable of fulfilling the key responsibilities of a referee, was achievable. To ensure this goal was met, however, it was necessary to divide the functional requirements, identified through discourse with a focus group consisting of soccer referees and additional comparative analysis of existing products during the project's 'analysis and design' phase, into two categories: core functionality and peripheral functionality i.e. further developments. The core functionality encompasses timekeeping, scorekeeping, disciplinary card allocation, player substitutions and exportable game reports comprising data accumulated from time stamped game events during the course of an officiated match. Upon completion of the project each of the core functional requirements has been successfully implemented in the deployed version of the *RefWatch: Soccer Edition* application. A categorised list of functional requirements, both successfully implemented and those awaiting future development, is contained in Table 8.

| Functional Requirements | |
|---|---|
| **Successfully Implemented** | **Future Developments** |
| Timekeeping | Record of player injuries |
| Scorekeeping | Referee activity tracking (GPS) |
| Disciplinary card allocation | Audio recording |
| Player substitution processing | 'Laws of the Game' reference guide |
| Time stamp registered game events | For additional suggestions of future developments see *7.5: Recommendations* |
| Exportable game reports | |

Table 8: Project functional requirements and associated implementation results.

## 8.4: Objectives and Achievements

A number of project objectives (see *1.4: Project Objectives*) were defined which contributed to the achievement of the project's overarching aim. In summary, the objectives included:

- Investigating the role of a football referee and advancements in wearable technology.
- Identifying and comparing existing products and software applications available to football referees.
- Creating a questionnaire and distribute it to a focus group of football referees operating at the grassroots level of the sport.
- Identifying an operating system to serve as the platform upon which to design, develop and implement the smartwatch application's required software components.
- Developing an attractive user interface and an intuitive architecture for the application.
- Presenting a working application prototype to a focus group for testing and evaluation purposes.

All of the aforementioned objectives have been successfully fulfilled. A number of software development objectives, which are closely aligned with the requirements of the application's end users and which were defined as user stories (see *3.3.3: User Stories*), have been considered throughout the application's design and implementation stages. In addition to the stated non-functional requirements (see *3.3.2: Non-functional Requirements*) - reliability, robustness, speed, ease of use, and portability - the user requirements have been implemented and fulfilled where possible as highlighted by the overwhelmingly positive response received in the testing questionnaire (see *Appendix Eight: Testing Focus Group Questionnaire Results*). In specific cases, however, the inherent limitations of the smartwatch form factor has undermined the satisfactory achievement of a requirement. For example, smartwatch battery constraints will mean that in situations where referees are required to officiate multiple matches with significant durations in quick succession, a common occurrence at youth soccer tournaments for instance, the device's battery life may not be sufficient. The Sony *SmartWatch 3*, which was used as the application's host device for testing purposes, is prone to overheating

after prolonged periods of use, clearly a shortcoming of the device's chosen hardware components and overall design. As smartwatch designs are enhanced in future iterations, however, such hardware issues should be minimised or even eradicated.

## 8.5: Recommendations

While the *RefWatch: Soccer Edition* application's core functionality requirements were successfully implemented and the project's aim fulfilled, a number of recommendations for improvements to the current iteration of the application have been formed through critical evaluation and in response to the feedback received from the focus group during the project's testing phase:

- **Increased customisation of period/interval duration**: *RefWatch* v.1.0 provides the user with the ability to set a period or interval duration in their preferred number of minutes. In future versions it would be beneficial to offer improved customisation of period or interval duration in the format of minutes and seconds (mm:ss).

- **Option for extra time and penalties**: The application does not currently facilitate soccer matches which require extra time and penalties to decide their outcome. Including such a feature would be a beneficial inclusion which could accommodate the officiating of cup matches which must often be decided through the addition of extra time periods and/or penalty kicks when they finish in a draw at the end of regulation time.

- **Separate screen for tracking recorded match events**: The application's current design maintains a record of match events by displaying a graphic in the appropriate team's designated listview located on the *Main Match Screen*. While the current design provides users with the ability to track elapsed time, the match's scoreline and recorded events all on a single screen, the smartwatch's reduced form factor requires that each function assume a relatively small portion of the device's display. By removing the listviews and creating a separate screen on which to keep a running log of recorded match events including the time of each event, the *Main Match Screen* could accommodate a redesign which increases the size of the textviews which facilitate the application's triple timer and scoreboard functions and thereby improve overall visibility during gameplay.

- **Alternative navigation methodology**: The application's current navigation system relies upon clickable buttons and textviews through which to navigate from one screen to another. An alternative method of navigation, which could reduce the number of interactions required to record a match event, is the 'swipe view'. Swipe views provide *Android* users with the ability to navigate laterally from one screen to another through horizontal finger gestures. The application's current design requires that the user select the appropriate team, 'home' or 'away', in each of the match event selection pathways. Implementing swipe views, however, could enable *RefWatch* users to access the *Match Event Selection Screen*, for instance, by swiping left for the 'home' team or right for the 'away' team, thereby removing the need for separate team selection screens entirely and consequently reducing user interaction times which should ideally be kept to a minimum (Developer.android.com, 2016 b).

- **Accompanying mobile app**: The development of an accompanying mobile application from which the user could create an account, log in or out, and save or manage their personal details or match settings would add another level of customisation to the application's user experience. The mobile device's larger form factor and facilitation of data input through an onboard keyboard could simplify the scheduling of forthcoming matches including their dates, times and venues, while enabling the user to record specific details of match participants including the names of the teams, players, coaching staff and officials, and each team's colours. With the implementation of peripheral functionality such as GPS tracking, moreover, the user's mobile device could be utilised as a central hub for data analysis with access to aggregated statistics from across all of their scheduled matches including total and average cards issued per game, the distance the referee runs in a game and a GPS map of their movements. Access to a 'Laws of the Game' reference guide, another future development, would be more effective from a mobile device due to its larger form factor which would enhance the users' ability to view and interact with such a text-based feature.

- **Additional match screen information**: The addition of the actual date and time at the user's location on the *Main Match Screen* could be beneficial if officiating in tournaments consisting of multiple soccer matches with strictly scheduled kick-off and conclusion times. Another useful addition could be the minute at which match events are recorded, although with available layout space at a premium under the current configuration this

may require a separate screen for recording events as referenced in the aforementioned ‘Alternative navigation methodology’ suggestion.

- **Match report deletion**: The Reports screen currently stores the details of all match reports saved in the database in a list format from which the user can choose to export a text version to their device. It would be beneficial to add the functionality to delete a report from the aforementioned list when the report or saved data is no longer required.

## 8.6: Final Note

The smartwatch application’s core functionality, as determined through discourse with the project’s key stakeholders, was successfully implemented and, moreover, received favourable feedback from the testing focus group. The deployed version of the *RefWatch: Soccer Edition* application facilitates the key responsibilities of a soccer referee - timekeeping, scorekeeping, and recording important game events including player substitutions and the allocation of disciplinary cards - and reduces their post-match administrative workload by exporting match reports consisting of saved data retrieved from an SQLite database directly to their device. While the current iteration of the application fulfills the project’s goals, a number of potential improvements and additional developments for future versions have been identified and discussed. The current iteration of the application will therefore serve as a foundation upon which a more comprehensive solution to game management and administrative duties for soccer referees officiating at the grassroots level of the sport can be formulated and developed. Furthermore, the application’s main features could conceivably be replicated and adjusted to aid officials in other popular team sports such as rugby or hockey. The project, which has grown into an extensive body of work, has provided invaluable opportunities for experiential learning in the field of software development, and particularly the practices associated with the Agile methodology and *Android* application development. Such experience and skill acquisition will be informative to the development challenges which lie ahead.

# References


Antón, P., Silberglitt, R. and Schneider, J. (2001). *The global technology revolution*. Santa Monica, CA: RAND.

Buchalka, T. (2014). *Learn Android Lollipop Development: Build an Android App Now*. Available at: https://www.udemy.com/android-marshmallow-java-app-development-course/learn/v4/overview [Accessed 31 May 2016].

Calvo, A. (2015). *Beginning Android Wearables*. New York, NY: Springer.

Cisco (2011). *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*. Available at: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf [Accessed 19 Jan. 2016]

Cisco (2014). *How Many Internet Connections are in the World? Right. Now*. [online] Available at http://blogs.cisco.com/news/cisco-connections-counter/ [Accessed 19 Jan. 2016]

Daniel, S. (2015). *Android Wearable Programming*. 1st ed. [ebook] Birmingham, UK: Packt Publishing, pp.3-6. Available at: https://www.safaribooksonline.com/libarary/view/android-wearable-programming/9781785280153/ch01s02.html [Accessed 21 Jan. 2016].

Developer.android.com, (n.d. a). *Android Developers*. [online] Available at: http://developer.android.com [Accessed 20 Jan. 2016].

Developer.android.com. (n.d. b). *Activities*. Available at: https://developer.android.com/guide/components/activities.html [Accessed 1 Jun. 2016].

Developer.android.com. (n.d. c). *App Manifest*. Available at: https://developer.android.com/guide/topics/manifest/manifest-intro.html [Accessed 1 Jun. 2016].

Developer.android.com. (n.d. d). *Application Fundamentals*. Available at: https://developer.android.com/guide/components/fundamentals.html [Accessed 3 Jun. 2016].

Developer.android.com. (n.d. e). *Creating and Running a Wearable App*. Available at: https://developer.android.com/training/wearables/apps/creating.html [Accessed 27 May 2016].

Developer.android.com. (n.d. f). *Creating Swipe Views with Tabs.* Available at: https://developer.android.com/training/implementing-navigation/lateral.html [Accessed 16 Aug. 2016].

Developer.android.com. (n.d. g). *Exiting Full-Screen Activities*. Available at: https://developer.android.com/training/wearables/ui/exit.html [Accessed 13 Aug. 2016].

Developer.android.com. (n.d. h). *Intents and Intent Filters*. Available at: https://developer.android.com/guide/components/intents-filters.html [Accessed 6 Jun. 2016].

Developer.android.com. (n.d. i). *Keeping Your App Visible*. Available at: https://developer.android.com/training/wearables/apps/always-on.html [Accessed 13 Aug. 2016].

Developer.android.com. (n.d. j). *Meet Android Studio*. Available at: https://developer.android.com/studio/intro/index.html [Accessed 23 Jul. 2016].

Developer.android.com. (n.d. k). *Packaging Wearable Apps*. Available at: https://developer.android.com/training/wearables/apps/packaging.html [Accessed 3 Jun. 2016].).

Developer.xamarin.com, (2016). *Introduction to the Mobile Software Development Lifecycle* [online] Available at: https://developer.xamarin.com/guides/cross-platform/getting_started/introduction_to_mobile_sdlc/ [Accessed 20 Jan. 2016].

Droid Life, (2015). *How to: Use Always-On Apps With Android Wear*. [online] Available at: http://www.droid-life.com/2015/06/26/how-to-use-always-on-apps-with-android-wear/ [Accessed 20 Jan. 2016].

Epstein, M. (2013). *Referee Wallet on the App Store*. [online] App Store. Available at: https://itunes.apple.com/us/app/referee-wallet/id593790946?mt=8 [Accessed 12 Jan. 2016].

FIFA.com, (2007). *Big Count*. [online] Available at: http://www.fifa.com/worldfootball/bigcount/index.html [Accessed 30 Dec. 2015].

FIFA.com, (2015 a). *Laws of the Game 2015/16*. [online] Available at: http://www.fifa.com/mm/Document/FootballDevelopment/Refereeing/02/36/01/11/LawsofthegamewebEN_Neutral.pdf [Accessed 7 Jan. 2016].

FIFA.com, (2015 b). World Cup Final smashes TV records in US, Japan. [online] Available at: http://www.fifa.com/womensworldcup/news/y=2015/m=7/news=world-cup-final-smashes-tv-records-in-us-japan-2661775.html [Accessed 6 Jan. 2016].

FindRugbyNow.com, (2012). *A Look at the Television Match Official*. [online] Available at: http://www.findrugbynow.com/2012/10/television-match-official/ [Accessed 6 Jan. 2016].

Football-Bible.com. (2015). Soccer Referee Responsibilities. [online] Available at: http://www.football-bible.com/soccer-info/soccer-referee-responsibilities.html [Accessed 16 Jan. 2016].

Fjordnet.com, (2015) *A Designer's Guide to Wearables*. [online] Available at: https://wearablesguide.fjordnet.com/ [Accessed 18 Dec. 2015].

GitHub. (2016). *spacecowboy/NoNonsense-FilePicker*. Available at: https://github.com/spacecowboy/NoNonsense-FilePicker [Accessed 10 Aug. 2016].

Guide to Procedures: Referees, Assistant Referees and Fourth Officials. (2003). 1st ed. [ebook] Chicago, Illinois: U.S. Soccer Federation, pp.10-24. Available at: http://www.wcusc.org/docs/Referees/US_Guide_to_Procedures.pdf [Accessed 7 Jan. 2016].

Hall, C. (2013). *Acer: We are looking at wearable, coming in 2014*. [online] Pocket-lint.com. Available at: http://www.pocket-lint.com/news/121756-acer-we-are-looking-at-wearable-coming-in-2014 [Accessed 18 Jan. 2016].

IDC.com (2016). *IDC: Smartphone OS Market Share*. [online] Available at: http://www.idc.com/prodserv/smartphone-os-market-share.jsp [Accessed 20 Jan. 2016].

In The Opinion Of The Referee, (2013 a). *Gear Review: Casio RFT100 Referee Watch*. [online] Available at: http://www.intheopinionofthereferee.com/gear-reviews/gear-review-casio-rft100-referee-watch [Accessed 12 Jan. 2016].

In The Opinion Of The Referee, (2013 b). *Gear Review: SPINTSO Referee Watch*. [online] Available at: http://www.intheopinionofthereferee.com/gear-reviews/gear-review-spintso-referee-watch [Accessed 12 Jan. 2016].

In The Opinion Of The Referee, (2014). *Premier League Referee Gear Revealed*. [online] Available at: http://www.intheopinionofthereferee.com/gear-reviews/premier-league-referee-gear-revealed [Accessed 16 Jan. 2016].

Johnson, J. (2014). *Designing with the mind in mind*. Waltham, MA: Morgan Kaufmann.

Kinvey.com, (2013). *How Long Does it Take to Build a Mobile App?* [online] Available at: http://www.kinvey.com/blog/2086/how-long-does-it-take-to-build-a-mobile-app [Accessed 20 Jan. 2016].

Lucidchart, (2016). *Lucidchart*. [online] Available at: http://www.lucidchart.com [Accessed 10 Jan. 2016].

Marshall, G. (2015). *Before Apple Watch: the timely history of the smartwatch*. [online] TechRadar. Available at: http://www.techradar.com/news/wearables/before-iwatch-the-timely-history-of-the-smartwatch-1176685 [Accessed 18 Jan. 2016].

Material design guidelines. (n.d.). *Color - Style - Material design guidelines*. Available at: https://material.google.com/style/color.html# [Accessed 13 Jun. 2016].

Mims, C. (2013). *Almost every major consumer electronics manufacturer is now working on a smart watch*. [online] Quartz. Available at: http://qz.com/101058/smart-watch-explosion/ [Accessed 18 Jan. 2016].

Mishra, S. (2015). *Wearable Android: Android Wear and Google Fit App Development*. Hoboken, NJ: Wiley.

Monk, A. and Howard, S. (1998). *The Rich Picture: A Tool for Reasoning About Work Context*. 1st ed. [ebook] New York, NY, USA: Association for Computing Machinery, pp. 21-30. Available at: http://www.moodle2.tfe.umu.se/pluginfile.php/32063/mod_resource/content/1/Rich%20pictures%20-monk.pdf [Accessed 10 Jan. 2016].

Online Charts and Graphs, (2016). *ONLINE CHARTS | create and design your own charts and diagrams online*. [online] Available at: http://www.onlinecharttool.com [Accessed 16 Jan. 2016].

Play.google.com, (2016). *Soccer Referee - Shingo 2.5*. [online] Available at: https://play.google.com/store/apps/details?id=com.spinkeysoft.shingo&hl=en [Accessed 12 Jan. 2016].

Proto.io, (2016). *Proto.io - Prototypes that feel real*. [online] Available at: http://www.proto.io [Accessed 9 Jan. 2016].

Saunders, S. (2014). *Advanced technology in the football industry*. [online] The Positive. Available at: http://www.thepostive.com/advanced-technology-in-the-football-industry/ [Accessed 6 Jan. 2016].

Schoolsteps.in, (2015). *Learn Android Wear Programming*. Available at: https://www.udemy.com/learn-android-wear-programming/learn/v4/overview [Accessed 31 May 2016].

Sheraz, F. and Dichone, P. (2015). *The Complete Android & Java Developer Course*. Available at: https://www.udemy.com/complete-android-developer-course/learn/v4/overview [Accessed 30 May 2016].

SoccerRefereeUSA.com, (n.d.). *Information for New Referees*. [online] Available at: http://soccerrefereeusa.com/Information%20for%20New%20Referees.pdf [Accessed 7 Jan. 2016].

SoccerRefereeUSA.com, (2011). *Laws of the Game Made Easy*. [online] Available at: http://soccerrefereeusa.com/images/RefereeCenter/laws%20of%20the%20game%20made%20easy.pdf [Accessed 7 Jan. 2016].

SoccerRefereeUSA.com, (2014). *US Soccer Advice To Referees 2013/14*. [online] Available at: http://soccerrefereeusa.com/images/RefereeCenter/atr14.pdf [Accessed 7 Jan. 2016].

Sommerville, I. (2011). *Software engineering*. Boston: Pearson.

Sourcebits, (2013). *Paid vs. Free Apps in the App Store vs. Google Play*. [online] Available at: http://sourcebits.com/app-development-design-blog/paid-vs-free-apps-app-store-vs-google-play/ [Accessed 13 Jan. 2016].

Spinkeysoft.com, (2016). *Soccer Referee Shingo 2.5*. [online] Available at: http://www.spinkeysoft.com/soccer-referee-shingo-2-5/ [Accessed 12 Jan. 2016].

Spintso.se, (2016). *English | Spintso*. [online] Available at: http://www.spintso.se/en/eng [Accessed 12 Jan. 2016].

Stackoverflow.com. (2014,). *Android activity life cycle - what are all these methods for?* Available at: http://stackoverflow.com/questions/8515936/android-activity-life-cycle-what-are-all-these-methods-for [Accessed 3 Jun. 2016].

Stackoverflow.com. (2013). *Copy SQLite database from assets folder*. Available at: http://stackoverflow.com/questions/16354154/copy-sqlite-database-from-assets-folder [Accessed 8 Aug. 2016].

Stackoverflow.com. (2012). *How to use an existing database with an Android application*. Available at: http://stackoverflow.com/questions/9109438/how-to-use-an-existing-database-with-an-android-application [Accessed 13 Aug. 2016].

Stackoverflow.com. (2015). *What is Gradle in Android Studio?* Available at: http://stackoverflow.com/questions/16754643/what-is-gradle-in-android-studio [Accessed 16 Aug. 2016].

Statista, (2015). *Distribution of free and paid Android apps 2015*. [online] Available at: http://www.statista.com/statistics/266211/distribution-of-free-and-paid-android-apps/ [Accessed 13 Jan. 2016].

Statista, (2016). *Health & fitness tracker unit sales regions worldwide 2014-2015*. [online] Available at: http://www.statista.com/statistics/413265/health-and-fitness-tracker-worldwide-unit-sales-region/ [Accessed 18 Jan. 2016].

StrategyBeach, (2013). *Agile Development Methodology.* [online] Available at: http://www.strategybeach.com/our-agile-development-methodology/ [Accessed 20 Jan. 2016].

Techotopia.com. (n.d.). *The Basics of the Android Studio Code Editor*. Available at: http://www.techotopia.com/index.php/The_Basics_of_the_Android_Studio_Code_Editor [Accessed 23 Jul. 2016].

The Economist, (2011). *Ranking sports' popularity.* [online] Available at: http://www.economist.com/blogs/gametheory/2011/09/ranking-sports'-popularity [Accessed 5 Jan. 2016].

www.tutorialspoint.com. (2016 a). *Software Testing - Overview*. Available at: http://www.tutorialspoint.com/software_testing/software_testing_overview.htm [Accessed 13 Aug. 2016].

www.tutorialspoint.com. (2016 b). *Software Testing Tutorial*. Available at: http://www.tutorialspoint.com/software_testing/ [Accessed 13 Aug. 2016].

Wallmüller, E., (2002). *Risk management for IT and software projects*. In Business continuity (pp. 165-178). Springer Berlin Heidelberg.

Williams, C. (2016). *Roles of a Soccer Referee*. [online] Houston Chronicle. Available at: http://www.work.chron.com/roles-soccer-referee-13468.html [Accessed 7 Jan. 2016].

Yoda Learning, (2015). *Development for Wearables: Android Wear*. Available at: https://www.udemy.com/android-wear/learn/v4/overview [Accessed 30 May 2016].

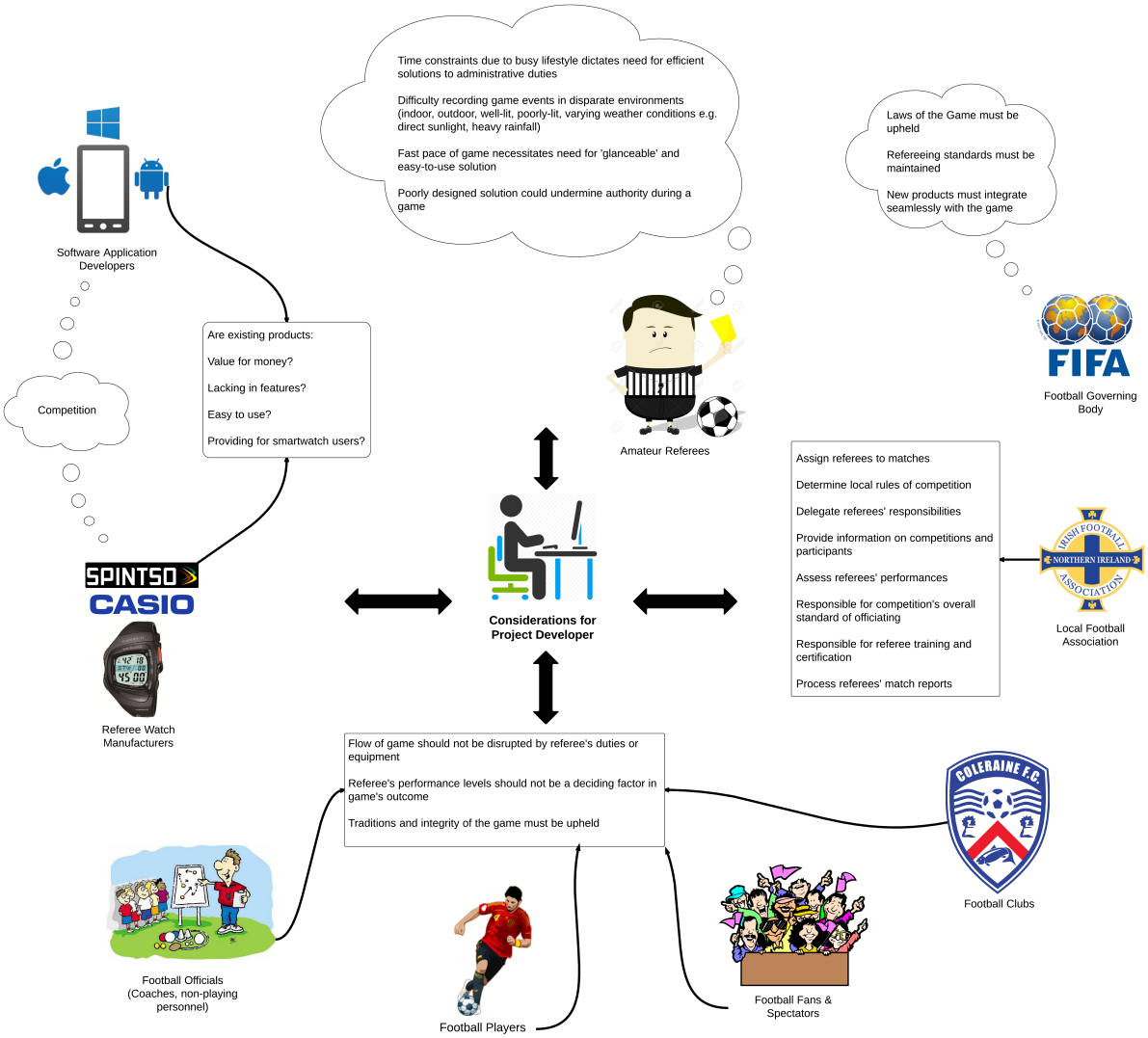
YouTube. (2015 a). *Android studio: Use attach SQLite database file in assets folder with customize adapter*. Available at: https://www.youtube.com/watch?v=aLnVhgYdVd8 [Accessed 13 Aug. 2016].

YouTube. (2015 b). *SPINTSO Referee Watch Demo*. Available at: https://www.youtube.com/watch?v=3WG2Sv0SI_w&index=11&list=PLiPMJ5jfqePTpgQCW1SGCmF7BLM2ZOa-- [Accessed 14 Jun. 2016].

Zapsplat.com. (n.d.). *Air Horn – ZapSplat – Download free sound effects*. Available at: http://www.zapsplat.com/sound-effect-category/air-horn/ [Accessed 2 Aug. 2016].

# Appendices

## Appendix One
### Rich Picture Outlining Stakeholders and Developer Considerations



# Appendix Two
## Requirements Gathering Questionnaire

### 2.1: Focus Group Questionnaire and Response Data

**Question 1: How long have you been a football referee?**
See *2.2: Questionnaire Response Data Charts*.

**Question 2: What level of games do you officiate on a regular basis? Please choose all that apply.**

| | |
|---|---|
| A. Youth | 23/77% |
| B. High School | 15/50% |
| C. College/University | 11/37% |
| D. Adult Recreational | 19/63% |
| E. Adult Amateur (e.g. Inter-County, PDL, NPSL) | 11/37% |
| F. Professional (e.g. MLS, NWSL, NASL, USL) | 1/3% |

**Question 3: What equipment do you typically use when officiating a game? Please choose all that apply.**

| | |
|---|---|
| A. Score Card with Pen/Pencil | 30/100% |
| B. Whistle | 29/97% |
| C. Watch | 29/97% |
| D. Dedicated Fitness Tracker (e.g. Fitbit, Garmin, Polar) | 10/33% |
| E. Smartwatch (e.g. Apple Watch, Sony SmartWatch 3) | 2/7% |
| F. Headset | 1/3% |

**Question 4: What make and model of watch do you currently use for refereeing?**

| | |
|---|---|
| A. Spintso PDA | 1/3% |
| B. Spintso Referee Watch (Pro, 2S, 2X) | 2/7% |
| C. Casio RFT100 (or similar Casio) | 13/37% |
| D. Smartwatch | 2/7% |
| E. Other (Please specify) | 14/47% |

**Question 5: Do you currently own a smartwatch?**

| | |
|---|---|
| A. Yes | 6/20% |
| B. No | 24/80% |

**Question 5.1: If you own a smartwatch, do you use it for refereeing?**

| | |
|---|---|
| A. Yes | 2/33% |
| B. No | 4/66% |

**Question 5.2: If you use your smartwatch for refereeing, which apps do you use?**
- “Referee” for Tizen OS.

**Question 6: If a well-designed smartwatch referee app became available, how likely are you to use it?**

| | |
|---|---|
| A. Extremely likely | 13/43% |
| B. Fairly likely | 13/43% |
| C. Undecided | 3/10% |
| D. Fairly unlikely | 0 |
| E. Extremely unlikely | 1/3% |

**Question 7: What is your preferred platform for a referee app?**

| | |
|---|---|
| A. Apple watchOS | 12/40% |
| B. Android Wear | 15/50% |
| C. Tizen OS | 1/3% |
| D. Pebble OS | 2/7% |
| E. Other | 0 |

**Question 8: What features would you like to see in a smartwatch app for referees?**

| | |
|---|---|
| A. Dual timer | 29/97% |
| B. Score keeping | 24/80% |
| C. Yellow/Red Card allocation | 22/73% |
| D. Record player substitutions | 13/43% |
| E. Record player injuries | 11/37% |
| F. Time stamp registered events | 15/50% |
| G. GPS tracking | 22/73% |
| H. Laws of the Game reference guide | 10/33% |
| I. Exportable game reports | 19/63% |
| J. Other (please specify) | 3/10% |

**Question 9: Which shape of smartwatch screen would be your preference for a referee app?**

| | |
|---|---|
| A. Rectangular | 24/80% |
| B. Round | 6/20% |

**Question 10: If the smartwatch app required the use of a mobile phone, would you be willing to carry your phone while officiating a game?**

| | |
|---|---|
| A. Yes | 5/19% |
| B. No | 25/81% |

**Question 11: Do you have any reservations about the use of smartwatches for refereeing?**

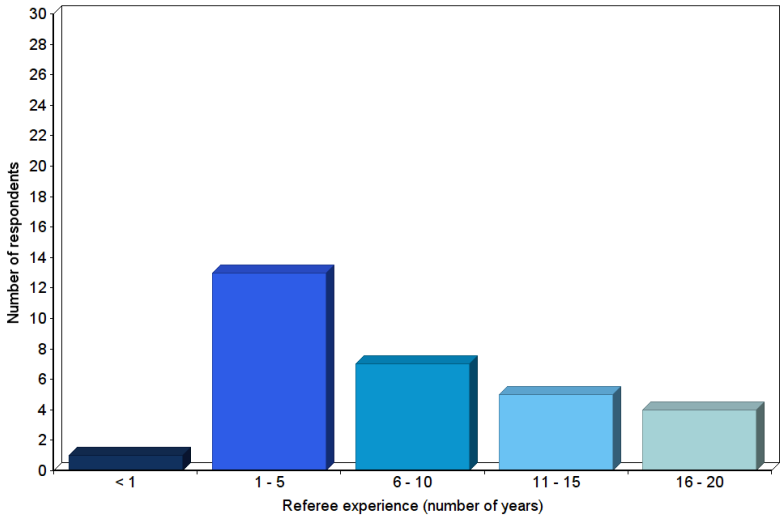
- "...I would not want to carry my phone while officiating."
- "To put everything into one technology system on your wrist could take longer to use and is risky if the battery dies or the watch breaks on the field."
- "No, I think they are great to track fitness."
- "Some features would seem finicky to enter on a smartwatch. Battery life is another issue."
- "For older users my main concern is the ability to read the screen, plus the amount of time it would take to enter score updates or cards."
- "I would need to be 100% sure that it would not lose my information or crash while refereeing."
- "Concerned about an app crashing during a game."
- "It can't fail. I carry 2 watches, 2 pencils, 2 sets of cards (each with paper) in a wallet, and 2 whistles. For nearly every game, I only use 1 of each - but, I have lost a pencil, whistle, a watch has failed, the paper has ripped, etc. So, 2 of everything. With the smart watch, I would still have another watch and the doubles of everything but it would be nice to be more accurate in event times and in knowing how much I have exercised."
- "Smartwatches are too expensive."
- "Fitness tracking would be a bit overkill since most watches automatically do this. Keep in mind that we can't be carrying phones on our person so the app needs to be able to stand alone."

**Question 12: Please provide any additional comments on smartwatches or features you would like to see implemented in a smartwatch app designed for referees.**

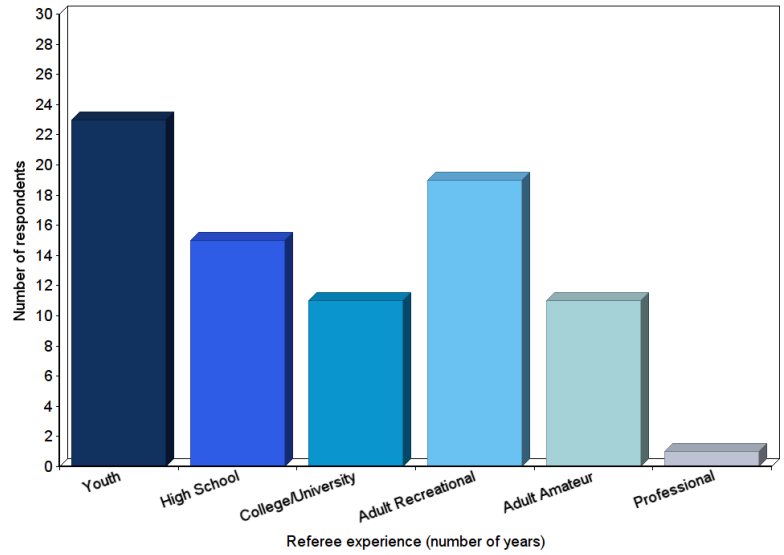
- “Should be dirt- and water-proof. Screen should be really bright to beat the sun rays. Just one screen for the game, refs don’t have time to slide diff screens during the game.”
- “It has to be simple and quick to use.”
- “The user experience must be trivial for each on-field action - 2 touches (perhaps 1 touch to select the action and 1 touch to effect the action). And don’t forget that we make mistakes - I know it’s hard to believe but during the 2nd half I often award the goal to the wrong team - perhaps with a color display where the color of the team surrounds the score would help me not make that mistake. And it would be nice if the Time Elapsed indicator could be a tactile buzz as opposed to an audible noise.”
- “Can you have a voice recorder built in for when players and officials are talking, so if a player is abusive it can record that.”
- “Ability to pair with other smartwatches, like the assistant referee’s - when they tap theirs it vibrates on my wrist as well.”
- “It should be easy to use. For example, I shouldn’t have to fiddle around trying to find the scorer’s section or something similar. I was also thinking, if possible, voice recognition so you could just say the name of a person who got cautioned or scored and then it either recognises and notes it, or you could manually listen to it later for your notes/match report.”
- “It needs to be simple. Keep time and record an event by typing in the number. Yellow and red cards should be able to designate reason (e.g. Unsporting behaviour, reckless challenge, etc.).”
- “Ideally, you want to keep the glances to the watch the same as they are now. Playing around with the watch to record a player number is quite impractical. But a button or gesture of some kind that presents a full screen reference number could be handy in recording events on the fly. Post game you flip through each reference.”
- “If a whole officiating team had the app, instead of using beeper flags assistant referees could have a large button on the watch which would vibrate and buzz the referee’s watch. You could also wirelessly connect the watches and use headsets for a cost-effective communications system.”

## 2.2: Questionnaire Response Data Charts

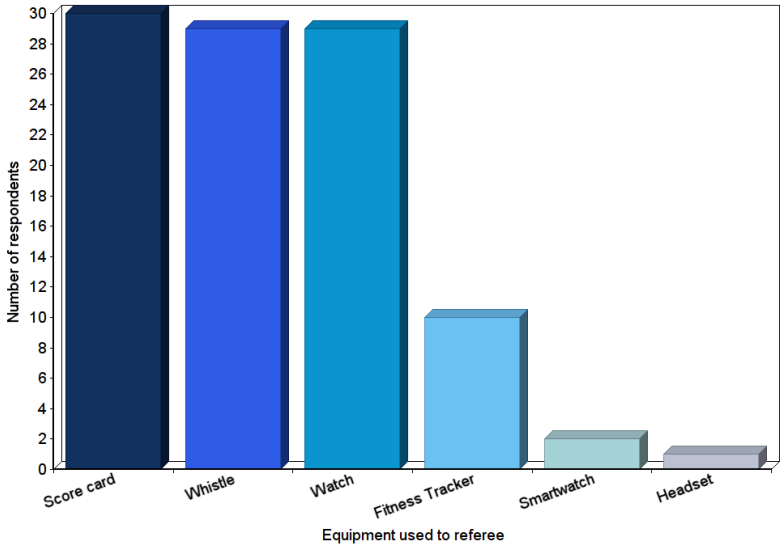
### Question 1: How long have you been a football referee?



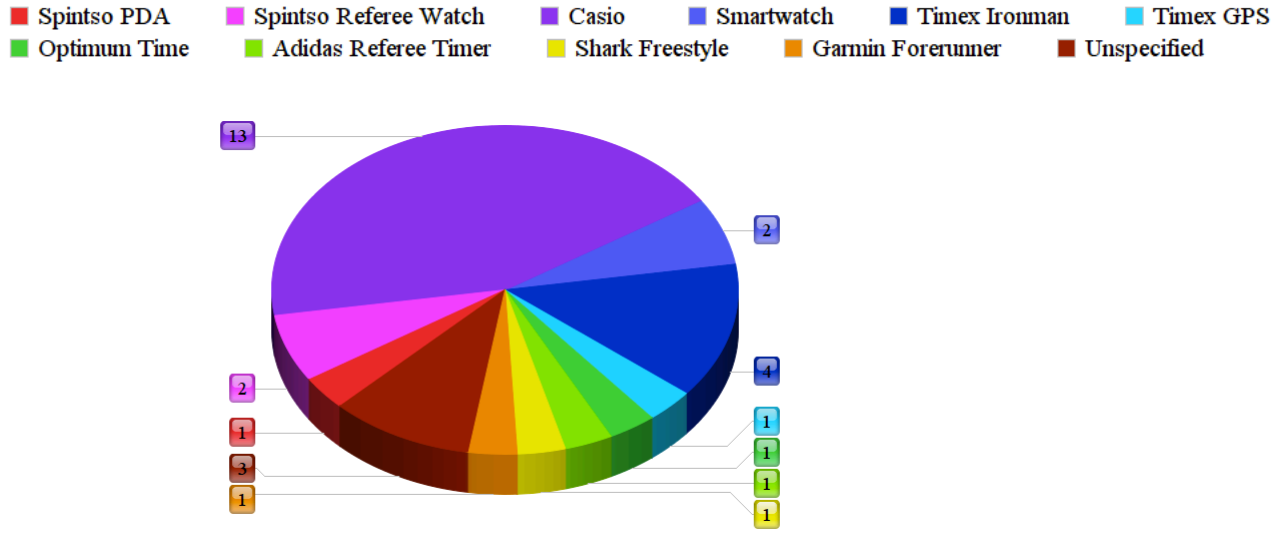

### Question 2: What level of games do you officiate on a regular basis?



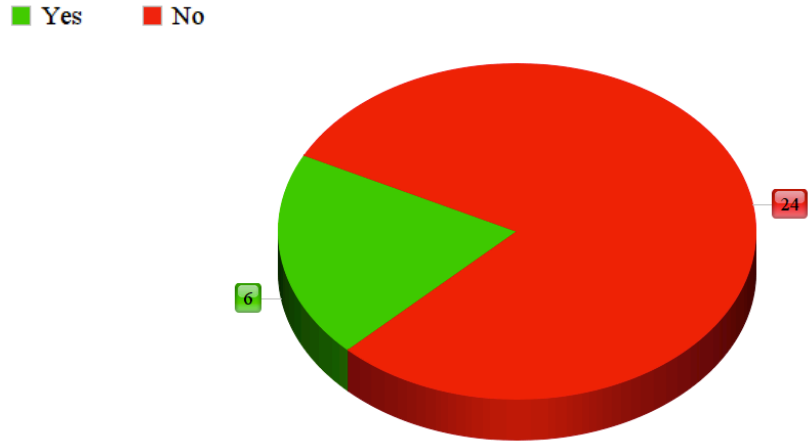

### Question 3: What equipment do you typically use when officiating a game?



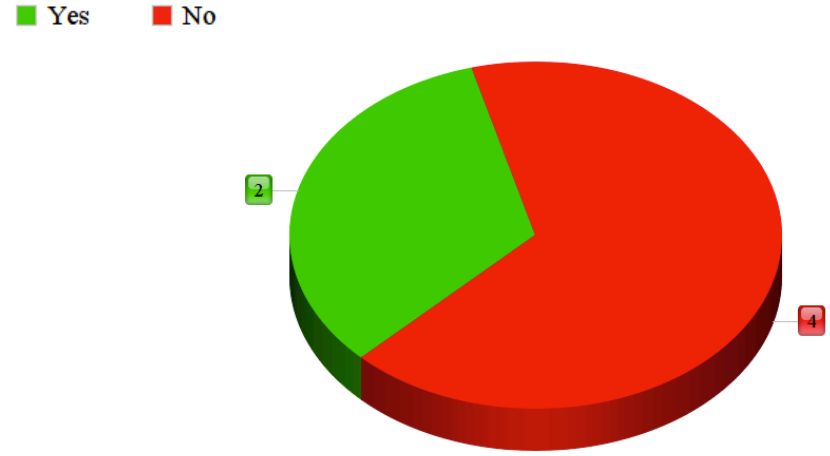
## Question 4: What make and model of watch do you currently use for refereeing?



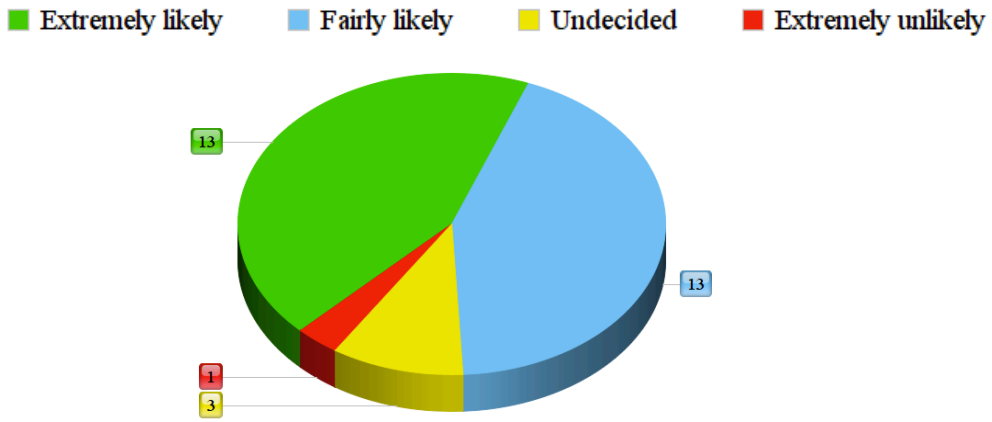

## Question 5: Do you currently own a smartwatch?



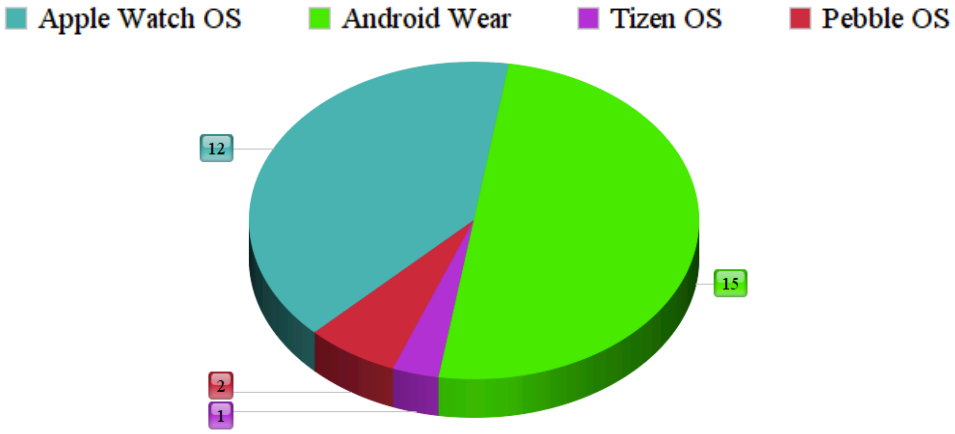

## Question 5.1: If you own a smartwatch, do you use it for refereeing?



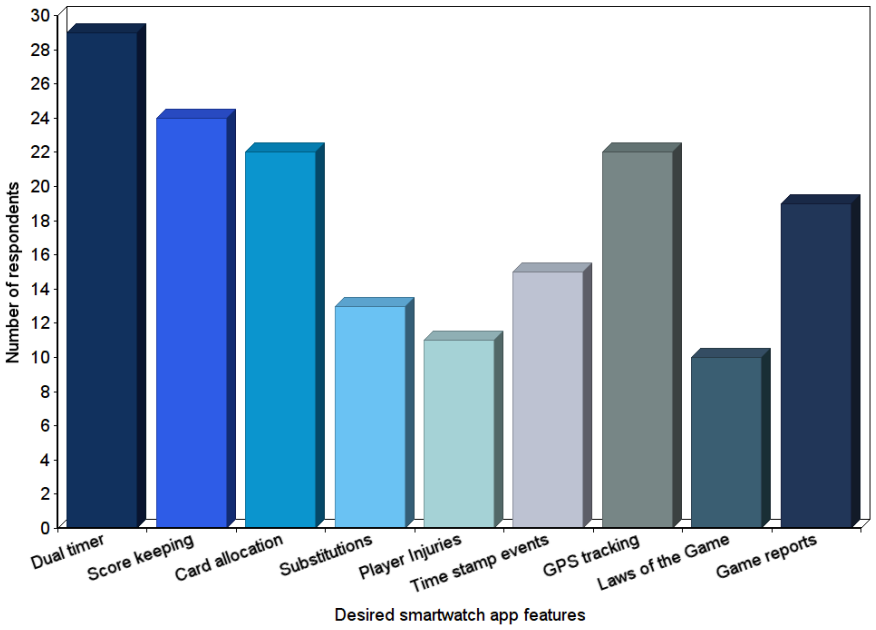
**Question 6: If a well-designed smartwatch referee app became available, how likely are you to use it?**



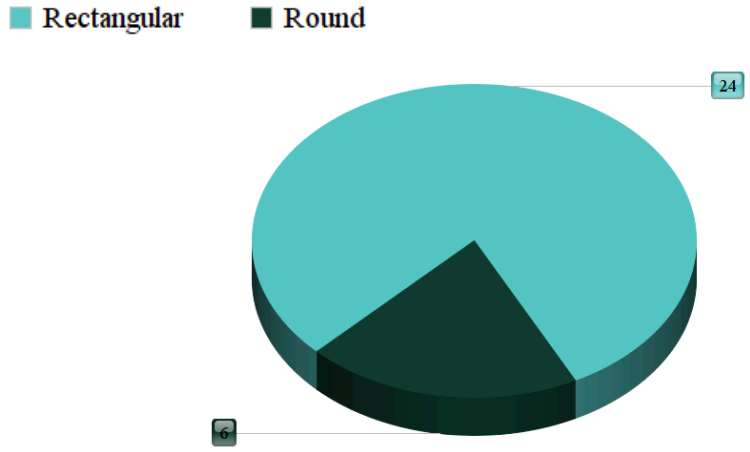

**Question 7: What is your preferred platform for a referee app?**



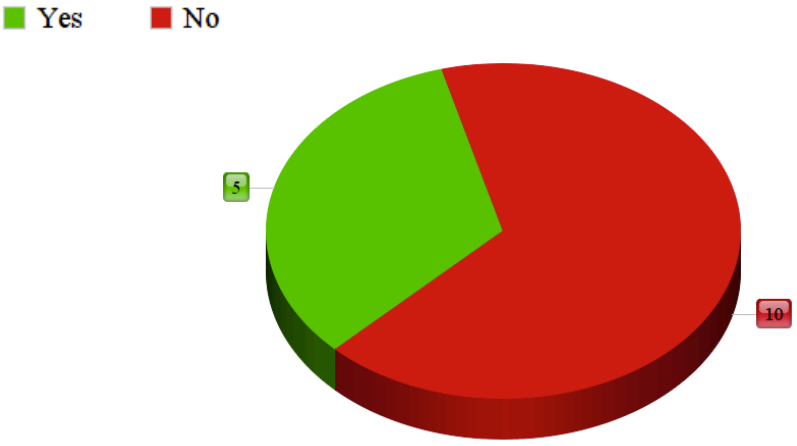

**Question 8: What features would you like to see in a smartwatch app for referees?**



**Question 9: Which shape of smartwatch screen would be your preference for a referee app?**




**Question 10: If the smartwatch app required the use of a mobile phone, would you be willing to carry your phone while officiating a game?**



# Appendix Three
## Database Tables

| match | | | |
|---|---|---|---|
| Name | Type | Keys | Value |
| match_id | integer | Primary Key | Auto-increment |
| period_duration | text | | Not null |
| number_periods | integer | | Not null |
| interval_duration | text | | Default(null) |
| match_date_time | text | | Not null |
| match_abandoned | boolean | | Default(false) |
| abandoned_time | text | | Default(null) |

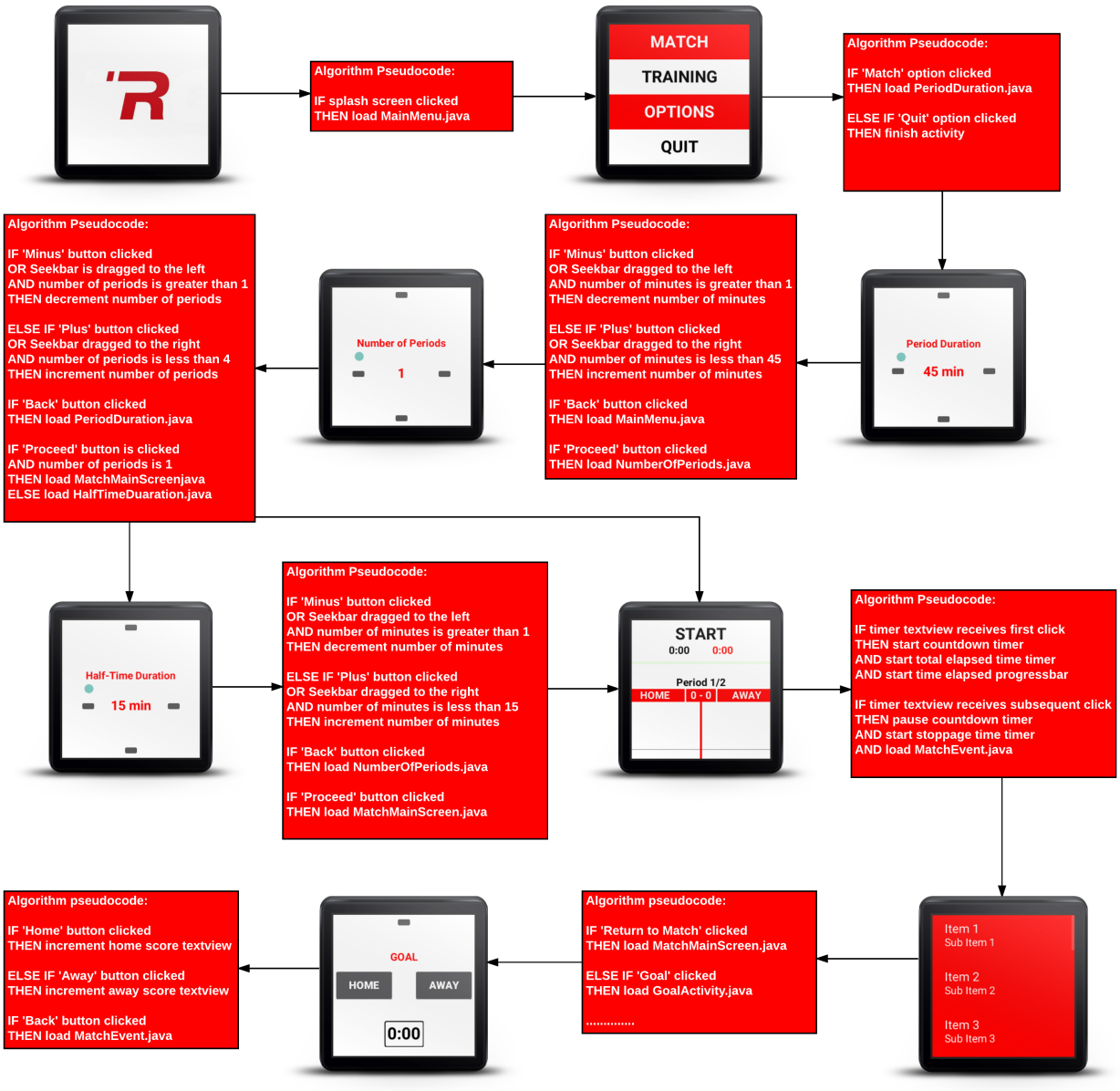
| goal | | | |
|---|---|---|---|
| Name | Type | Description | Value |
| goal_id | integer | Primary Key | Auto_increment |
| team_id | text | | Not null |
| player_num | text | | Not null |
| pk_goal | boolean | | Default (false) |
| goal_time | text | | Not null |
| match_id | integer | Foreign Key | Not null |

| penalty_kick | | | |
|---|---|---|---|
| Name | Type | Description | Value |
| pk_id | integer | Primary Key | Auto-increment |
| team_id | text | | Not null |
| player_num | text | | Not null |
| pk_outcome | text | | Not null |
| pk_time | text | | Not null |
| match_id | integer | Foreign Key | Not null |

| substitution | | | |
|---|---|---|---|
| Name | Type | Description | Value |
| subs_id | integer | Primary Key | Auto-increment |
| team_id | text | | Not null |
| out_player_num | text | | Not null |
| in_player_num | text | | Not null |
| subs_time | text | | Not null |
| match_id | integer | Foreign Key | Not null |

| disciplinary_action | | | |
|---|---|---|---|
| Name | Type | Description | Value |
| discipline_id | integer | Primary Key | Auto-increment |
| team_id | text | | Not null |
| player_or_staff | text | | Not null |
| player_num | text | | Default(null) |
| staff_role | text | | Default(null) |
| card_colour | text | | Default(null) |
| infringement_type | text | | Default(null) |
| discipline_time | text | | Default(null) |
| match_id | integer | Foreign Key | Not null |

# Appendix Four
## Application User Storyboard



## Appendix Five
### Schneiderman's "Eight Golden Rules of Interface Design"

1. **Strive for consistency**.

Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent color, layout, capitalization, fonts, and so on should be employed throughout. Exceptions, such as required confirmation of the delete command or no echoing of passwords, should be comprehensible and limited in number.

**2. Cater to universal usability.**

Recognize the needs of diverse users and design for plasticity, facilitating transformation of content. Novice to expert differences, age ranges, disabilities, and technological diversity each enrich the spectrum of requirements that guides design. Adding features for novices, such as explanations, and features for experts, such as shortcuts and faster pacing, can enrich the interface design and improve perceived system quality.

**3. Offer informative feedback.**

For every user action, there should be system feedback. For frequent and minor actions, the response can be modest, whereas for infrequent and major actions, the response should be more substantial. Visual presentation of the objects of interest provides a convenient environment for showing changes explicitly.

**4. Design dialogs to yield closure.**

Sequences of actions should be organized into groups with a beginning, middle, and end. Informative feedback at the completion of a group of actions gives operators the satisfaction of accomplishment, a sense of relief, a signal to drop contingency plans from their minds, and an indicator to prepare for the next group of actions. For example, e-commerce web sites move users from selecting products to the checkout, ending with a clear confirmation page that completes the transaction.

**5. Prevent errors.**

As much as possible, design the system such that users cannot make serious errors; for example, gray out menu items that are not appropriate and do not allow alphabetic characters in numeric entry fields. If a user makes an error, the interface should detect the

error and offer simple, constructive, and specific instructions for recovery. For example, users should not have to retype an entire name-address form if they enter an invalid zip code, but rather should be guided to repair only the faulty part. Erroneous actions should leave the system state unchanged, or the interface should give instructions about restoring the state.

**6. Permit easy reversal of actions.**

As much as possible, actions should be reversible. This feature relieves anxiety, since the user knows that errors can be undone, and encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data-entry task, or a complete group of actions, such as entry of a name-address block.

**7. Support internal locus of control.**

Experienced users strongly desire the sense that they are in charge of the interface and that the interface responds to their actions. They don't want surprises or changes in familiar behavior, and they are annoyed by tedious data-entry sequences, difficulty in obtaining necessary information, and inability to produce their desired result.

**8. Reduce short-term memory load.**

Humans' limited capacity for information processing in short-term memory (the rule of thumb is that we can remember "seven plus or minus two chunks" of information) requires that designers avoid interfaces in which users must remember information from one screen and then use that information on another screen. It means that cell phones should not require re-entry of phone numbers, web-site locations should remain visible, multiple-page displays should be consolidated, and sufficient training time should be allotted for complex sequences of actions.

## Appendix Six
### Sony *SmartWatch 3* Technical Specifications (Sony Mobile, 2016)

**Requirements**
- Android 4.3 and onwards
- Android™ Wear

**Water protected**
- IP68 rated

**Performance**
- Quad ARM A7, 1.2 Ghz
- 512 MB RAM, 4 GB eMMC

**Controls**
- Voice, touch and gesture input
- Microphone
- On/off/wake up key

**Battery**
- 420mA (up to 2 days normal use)

**Sensors**
- Ambient light sensors
- Accelerometer
- Compass
- Gyro
- GPS

**Connectors**
- Bluetooth® 4.0
- NFC
- Micro USB
- Wi-Fi


Figure 1: The Sony *SmartWatch 3*.

# Appendix Seven
## In-house Testing Results

### 7.1: Layout View Testing

| Application screen | Square configuration | Round configuration |
|---|---|---|
| Splash screen | ✔ | ✔ |
| Main Menu | ✔ | ✔ |
| Period Duration Selection | ✔ | ✔ |
| Number of Periods Selection | ✔ | ✔ |
| Interval Duration Selection | ✔ | ✔ |
| Main Match Screen | ✔ | ✔ |
| Match Event Selection | ✔ | ✔ |
| Goal Team Selection | ✔ | ✔ |
| Goal Player Selection | ✔ | ✔ |
| Penalty Kick Team Selection | ✔ | ✔ |
| Penalty Kick Player Selection | ✔ | ✔ |
| Penalty Kick Outcome Selection | ✔ | ✔ |
| Substitution Team Selection | ✔ | ✔ |
| Substitution Outgoing Player Selection | ✔ | ✔ |
| Substitution Incoming Player Selection | ✔ | ✔ |
| Disciplinary Action Team Selection | ✔ | ✔ |
| Disciplinary Action Player/Staff Selection | ✔ | ✔ |
| Disciplinary Action Player Selection | ✔ | ✔ |
| Disciplinary Action Card Colour Selection | ✔ | ✔ |
| Disciplinary Action Yellow Card Infringement | ✔ | ✔ |
| Disciplinary Action Red Card Infringement | ✔ | ✔ |
| Disciplinary Action Staff Role Selection | ✔ | ✔ |
| Exportable Match Reports Menu | ✔ | ✔ |
| Options Menu | ✔ | ✔ |

Table 1: Results of layout view testing on Square and Round devices.

## 7.2: Software Component Functionality Testing

### 7.2.1: Main Menu Screen Testing

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Main Menu | *Match* menu option | Navigate to *Period Duration Selection* screen | Successful |
| Main Menu | *Reports* menu option | Navigate to *Exportable Reports* screen | Successful |
| Main Menu | Options menu option | Navigate to *Options* screen | Successful |
| Main Menu | Quit menu option | Quit the application | Successful |

### 7.2.2: Match Duration Settings Testing

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Match Duration Selection | Back button | Navigate to *Main Menu* screen | Successful |
| Match Duration Selection | Minus button | Decrement number of minutes if textview displays between 2 and 45 | Successful |
| Match Duration Selection | Plus button | Increment the number of minutes if textview displays between 1 and 44 | Successful |
| Match Duration Selection | Progress bar | Increment/decrement the number of minutes according to progress bar position | Successful |
| Match Duration Selection | Proceed button | Navigate to *Number of Periods Selection* screen | Successful |
| Number of Periods Selection | Back button | Navigate to *Period Duration Selection* screen | Successful |
| Number of Periods Selection | Minus button | Decrement number of periods if textview displays between 2 and 4 | Successful |
| Number of Periods Selection | Plus button | Increment the number of periods if textview displays between 1 and 3 | Successful |
| Number of Periods Selection | Progress bar | Increment/decrement the number of periods according to progress bar position | Successful |
| Number of Periods Selection | Proceed button | Navigate to *Interval Duration Selection* screen if number of periods is greater than 1 | Successful |
| Interval Duration Selection | Back button | Navigate to *Number of Periods Selection* screen | Successful |
| Interval Duration Selection | Minus button | Decrement number of minutes if textview displays between 2 and 15 | Successful |

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Interval Duration Selection | Plus button | Increment the number of minutes if textview displays between 1 and 14 | Successful |
| Interval Duration Selection | Progress bar | Increment/decrement the number of minutes according to progress bar position | Successful |
| Interval Duration Selection | Proceed button | Navigate to *Main Match Screen* | Successful |

### 7.2.3: Main Match Screen Testing

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Main Match Screen | START textview | Start Countdown and Elapsed Time timers | Successful |
| Main Match Screen | Countdown timer | Navigate to *Match Event Selection* screen when clicked during game and pauses until the user returns | Successful |
| Main Match Screen | Stoppage timer | Running when user navigates to Match Event Selection screen | Successful |
| Main Match Screen | Elapsed Time timer | Runs continuously, displaying the total elapsed time during a game and controls progression between periods when clicked by the user during additional time | Successful |

### 7.2.4: Match Event Selection Testing - Return to Match

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Match Event Selection | Return to Match option | Navigates to the Main Match screen, restarts the Countdown timer and pauses the Stoppage timer | Successful |

### 7.2.5: Match Event Selection Testing - Goal

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Match Event Selection | Goal option | Navigates to the Goal Team Selection screen | Successful |
| Goal Team Selection | Back button | Navigates to the Match Event Selection screen | Successful |
| Goal Team Selection | Home and Away buttons | Navigates to Goal Player Selection screen | Successful |
| Goal Team Selection | Elapsed Timer textview | Displays the current elapsed game time | Successful |

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Goal Player Selection | Back button | Navigates to the Goal Team Selection screen | Successful |
| Goal Player Selection | Proceed button | Navigates to Main Match screen and stores goal image and player number in the designated team's listview | Successful |

### 7.2.6: Match Event Selection Testing - Penalty Kick

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Match Event Selection | Penalty Kick option | Navigates to the PK Team Selection screen | Successful |
| PK Team Selection | Back button | Navigates to the Match Event Selection screen | Successful |
| PK Team Selection | Home and Away buttons | Navigates to PK Player Selection screen | Successful |
| PK Team Selection | Elapsed Timer textview | Displays the current elapsed game time | Successful |
| PK Player Selection | Back button | Navigates to the PK Team Selection screen | Successful |
| PK Player Selection | Proceed button | Navigates to PK Outcome Selection screen | Successful |
| PK Outcome Selection | Back button | Navigates to PK Player Selection screen | Successful |
| PK Outcome Selection | Scored and Missed buttons | Navigates to Main Match screen and stores scored or missed pk image and player number in the designated team's listview | Successful |
| PK Outcome Selection | Elapsed Timer textview | Displays the current elapsed game time | Successful |

### 7.2.7: Match Event Selection Testing - Substitution

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Match Event Selection | Substitution option | Navigates to the Substitution Team Selection screen | Successful |
| Substitution Team Selection | Back button | Navigates to the Match Event Selection screen | Successful |
| Substitution Team Selection | Home and Away buttons | Navigates to the Outgoing Player Selection screen | Successful |
| Substitution Team Selection | Elapsed Timer textview | Displays the current elapsed game time | Successful |

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Substitution Outgoing Player Selection | Back button | Navigates to the Substitution Team Selection screen | Successful |
| Substitution Outgoing Player Selection | Outgoing button | Navigates to the Incoming Player Selection screen | Successful |
| Substitution Incoming Player Selection | Back button | Navigates to the Outgoing Player Selection screen | Successful |
| Substitution Incoming Player Selection | Incoming button | Navigates to Main Match screen and stores substitution image and player numbers in the designated team's listview | Successful |

### 7.2.8: Match Event Selection Testing - Disciplinary Action

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Match Event Selection | Disciplinary Action option | Navigates to the Disciplinary Action Team Selection screen | Successful |
| Disciplinary Action Team Selection | Back button | Navigates to the Match Event Selection screen | Successful |
| Disciplinary Action Team Selection | Home and Away buttons | Navigates to the Player or Staff Selection screen | Successful |
| Disciplinary Action Team Selection | Elapsed Timer textview | Displays the current elapsed game time | Successful |
| Disciplinary Action Player or Staff Selection | Back button | Navigates to the Disciplinary Action Team Selection screen | Successful |
| Disciplinary Action Player or Staff Selection | Player button | Navigates to the Disciplinary Action Player Number Selection screen | Successful |
| Disciplinary Action Player or Staff Selection | Staff button | Navigates to the Disciplinary Action Staff Role Selection screen | Successful |
| Disciplinary Action Player or Staff Selection | Elapsed Timer textview | Displays the current elapsed game time | Successful |
| Disciplinary Action Player Number Selection | Back button | Navigates to the Player or Staff Selection screen | Successful |
| Disciplinary Action Player Number Selection | Proceed button | Navigates to the Disciplinary Action Card Colour Selection screen | Successful |
| Disciplinary Action Card Colour Selection | Back button | Navigates to the Disciplinary Action Player Number Selection screen | Successful |
| Disciplinary Action Card Colour Selection | Yellow Card button | Navigates to the Yellow Card Infringement Type Selection screen | Successful |
| Disciplinary Action Card Colour Selection | Red Card button | Navigates to the Red Card Infringement Type Selection screen | Successful |

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Disciplinary Action Card Colour Selection | Elapsed Timer textview | Displays the current elapsed game time | Successful |
| Yellow Card Infringement Type Selection | Back button | Navigates to the Disciplinary Action Card Colour Selection screen | Successful |
| Yellow Card Infringement Type Selection | Proceed button | Navigates to Main Match screen and stores yellow card image and player number in the designated team's listview | Successful |
| Red Card Infringement Type Selection | Back button | Navigates to the Disciplinary Action Card Colour Selection screen | Successful |
| Red Card Infringement Type Selection | Proceed button | Navigates to Main Match screen and stores red card image and player number in the designated team's listview | Successful |
| Disciplinary Action Staff Role Selection | Back button | Navigates to the Player or Staff Selection screen | Successful |
| Disciplinary Action Staff Role Selection | Coach button | Navigates to Main Match screen and stores red card image and staff role in the designated team's listview | Successful |
| Disciplinary Action Staff Role Selection | Other button | Navigates to Main Match screen and stores red card image and staff role in the designated team's listview | Successful |
| Disciplinary Action Staff Role Selection | Elapsed Timer textview | Displays the current elapsed game time | Successful |

### 7.2.9: Match Event Selection Testing - Abandon Match

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Match Event Selection | Abandon Match option | Navigates to the Main Menu screen and saves match data up until the recorded match time of abandonment | Successful |

### 7.2.10: Match Event Selection Testing - Quit

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Match Event Selection | Quit option | Navigates to the Main Menu screen without saving match data | Successful |

### 7.2.11: Options Screen Testing

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Options screen | Radio buttons | Selects Audible or Vibration alerts when Countdown timer reaches zero | Successful |
| Options screen | Back button | Navigates to the Main Menu screen without saving changes to alert settings | Successful |
| Options screen | Confirm button | Navigates to the Main Menu screen and saves changes to alert settings | Successful |

### 7.2.12: Reports Screen Testing

| Application Screen | Component Tested | Expected Functionality | Outcome |
|---|---|---|---|
| Reports screen | Export button | Selected match data will be exported to the user's mobile device in document format | Successful |
| Reports screen | Back button | Navigates to the Main Menu screen | Successful |

# Appendix Eight
## Testing Focus Group Questionnaire Results

### 8.1: User One

| Questionnaire Statement | User Rating | | | | | Additional Comments |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| Navigation throughout the system is consistent and uses simple, easy-to-learn commands. | | | | | ✔ | N/A |
| I can use the system while officiating a game without fear of system failure or data loss. | | | | | ✔ | N/A |
| The system provides a stand-alone solution to a referee's core responsibilities. | | | | | ✔ | N/A |
| The Graphical User Interface is easily readable. | | | | | ✔ | N/A |
| The system's design will minimize data input errors. | | | | | ✔ | N/A |
| The system provides adequate notification of elapsed regulation time at the end of each period. | | | | | ✔ | N/A |
| The system's game data storage facilities adequately fulfill match report requirements. | | | | | ✔ | N/A |
| The elapsed time of the match is always visible at a glance. | | | | | ✔ | N/A |
| The system's design allows the user to record match events in a time efficient manner. | | | | | ✔ | N/A |
| The system's design minimizes the need for user interaction during a match. | | | | | ✔ | N/A |
| **Comments or Recommendations** | Use full infringement title rather than abbreviations for disciplinary actions. | | | | | |

## 8.2: User Two

| Questionnaire Statement | User Rating | | | | | Additional Comments |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| Navigation throughout the system is consistent and uses simple, easy-to-learn commands. | | | | | ✔ | After a few tries the app's navigation was very easy. |
| I can use the system while officiating a game without fear of system failure or data loss. | | | | ✔ | | While no errors occurred during testing, the device's battery life might be a shortcoming. |
| The system provides a stand-alone solution to a referee's core responsibilities. | | | | | ✔ | N/A |
| The Graphical User Interface is easily readable. | | | | | ✔ | N/A |
| The system's design will minimize data input errors. | | | | | ✔ | Nice use of proceed and back buttons to cancel or confirm selections. |
| The system provides adequate notification of elapsed regulation time at the end of each period. | | | | | ✔ | N/A |
| The system's game data storage facilities adequately fulfill match report requirements. | | | | ✔ | | Adding player and team names would be a nice addition. |
| The elapsed time of the match is always visible at a glance. | | | | | ✔ | The inclusion of an elapsed timer in match event selection screens is an excellent feature. |
| The system's design allows the user to record match events in a time efficient manner. | | | | | ✔ | Duration of interactions is roughly equal to the traditional 'pen and paper' method. |
| The system's design minimizes the need for user interaction during a match. | | | | | ✔ | Timers are always visible so interaction is restricted to stoppages in play or recording events. |
| **Comments or Recommendations** | Prompt the user to issue a red card when a player has two yellow cards - currently the 2CT (second caution) option is selected in the red card section to register a second yellow card, which doesn't record the infringement type for the second yellow card. | | | | | |

### 8.3: User Three

| Questionnaire Statement | User Rating | | | | | Additional Comments |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| Navigation throughout the system is consistent and uses simple, easy-to-learn commands. | | | | | ✔ | While it is easy to learn, the User Guide was also helpful. |
| I can use the system while officiating a game without fear of system failure or data loss. | | | | ✔ | | I would feel comfortable using the app, but there is a chance of hardware failure with any technology so a standard watch as a backup would be sensible. |
| The system provides a stand-alone solution to a referee's core responsibilities. | | | | | ✔ | N/A |
| The Graphical User Interface is easily readable. | | | | | ✔ | N/A |
| The system's design will minimize data input errors. | | | | | ✔ | N/A |
| The system provides adequate notification of elapsed regulation time at the end of each period. | | | | | ✔ | N/A |
| The system's game data storage facilities adequately fulfill match report requirements. | | | | | ✔ | N/A |
| The elapsed time of the match is always visible at a glance. | | | | | ✔ | N/A |
| The system's design allows the user to record match events in a time efficient manner. | | | | | ✔ | N/A |
| The system's design minimizes the need for user interaction during a match. | | | | | ✔ | N/A |
| **Comments or Recommendations** | When a PK is missed but a goal scored from follow-up, the user has to re-enter the event selection menu to register the goal separately. Is there a way to streamline this process? | | | | | |

## 8.4: User Four

| Questionnaire Statement | User Rating | | | | | Additional Comments |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| Navigation throughout the system is consistent and uses simple, easy-to-learn commands. | | | | | ✔ | I really liked the use of red and green backgrounds for the outgoing and incoming player substitutions, as it subliminally helps the user to avoid mixing up the order of player numbers. |
| I can use the system while officiating a game without fear of system failure or data loss. | | | | | ✔ | N/A |
| The system provides a stand-alone solution to a referee's core responsibilities. | | | | | ✔ | N/A |
| The Graphical User Interface is easily readable. | | | ✔ | | | Nice bright colour scheme, but the timers and scoreboard could be bigger with the record of match events stored on a separate screen. |
| The system's design will minimize data input errors. | | | | | ✔ | N/A |
| The system provides adequate notification of elapsed regulation time at the end of each period. | | | | | ✔ | N/A |
| The system's game data storage facilities adequately fulfill match report requirements. | | | | | ✔ | N/A |
| The elapsed time of the match is always visible at a glance. | | | | | ✔ | N/A |
| The system's design allows the user to record match events in a time efficient manner. | | | | | ✔ | N/A |
| The system's design minimizes the need for user interaction during a match. | | | | | ✔ | N/A |
| **Comments or Recommendations** | N/A | | | | | |

## 8.5: User Five

| Questionnaire Statement | User Rating | | | | | Additional Comments |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | |
| Navigation throughout the system is consistent and uses simple, easy-to-learn commands. | | | | | ✔ | N/A |
| I can use the system while officiating a game without fear of system failure or data loss. | | | | | ✔ | N/A |
| The system provides a stand-alone solution to a referee's core responsibilities. | | | | | ✔ | N/A |
| The Graphical User Interface is easily readable. | | | | | ✔ | N/A |
| The system's design will minimize data input errors. | | | | | ✔ | N/A |
| The system provides adequate notification of elapsed regulation time at the end of each period. | | | | | ✔ | N/A |
| The system's game data storage facilities adequately fulfill match report requirements. | | | | | ✔ | N/A |
| The elapsed time of the match is always visible at a glance. | | | | | ✔ | N/A |
| The system's design allows the user to record match events in a time efficient manner. | | | | | ✔ | N/A |
| The system's design minimizes the need for user interaction during a match. | | | | | ✔ | N/A |
| **Comments or Recommendations** | N/A | | | | | |

## Appendix Nine
### *RefWatch: Soccer Edition* User Guide

Thank you for downloading *RefWatch: Soccer Edition*, the stand-alone game management solution for soccer referees.

## Step-By-Step Operational Instructions

### Splash Screen

When the user clicks on the *RefWatch: Soccer Edition* application icon located in the dock on their compatible *Android Wear* device (see *Figure 1*), they will be presented with a splash screen containing the application's logo (see *Figure 2*). To enter the application and navigate to the Main Menu simply click on the smartwatch screen.


Figure 1: *RefWatch: Soccer Edition* dock icon.


Figure 2: *RefWatch: Soccer Edition* splash screen.

### Main Menu

The Main Menu screen (see *Figure 3*) provides the user with a central hub from which they can access the application's key features. The menu contains four clickable options whose features and operations are explained in the subsequent sections of the guide:

## • Match

The Match section of the application contains all of the tools required by a soccer referee to fulfill their main responsibilities during a game. Before beginning a game, however, it is necessary for the user to select their preferred game duration settings.


*Figure 3: Main Menu Screen.*

### Period Duration Selection

The Period Duration Selection screen (see *Figure 4*) provides the user with the option to customise the duration of each period of play in the game. By clicking on the ‘plus’ or ‘minus’ buttons, or dragging the pointer on the horizontal progress bar, the user can set the period duration in minutes to last between 1 and 45 minutes. To confirm their selection the user must click on the ‘Proceed’ button. By clicking on the ‘Back’ button the user can return to the previous screen.

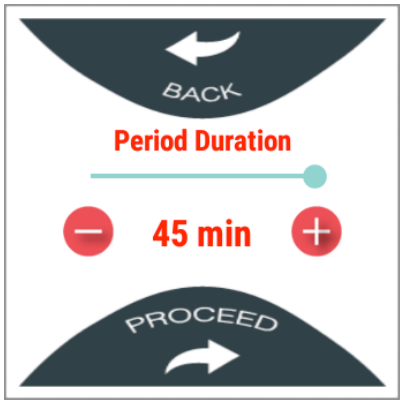



Figure 4: Period Duration Selection screen.

### Number of Periods Selection

The Number of Periods Selection screen (see *Figure 5*) provides the user with the option to customise the number of periods of play in the game. By clicking on the ‘plus’ or ‘minus’ buttons, or dragging the pointer on the horizontal progress bar, the user can select

between 1 and 4 periods of gameplay. To confirm their selection the user must click on the ‘Proceed’ button. By clicking on the ‘Back’ button the user can return to the previous screen.

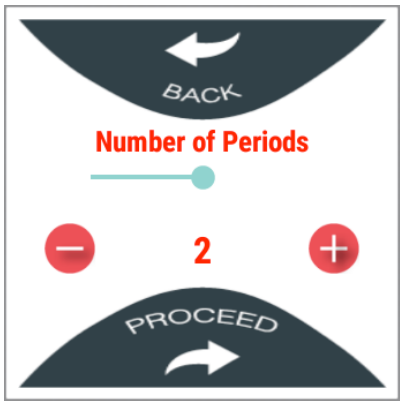



Figure 5: Number of Periods Selection screen.

## Interval Duration Selection

If the user selects more than one period of gameplay, the Interval Duration Selection screen (see *Figure 6*) provides them with the additional option to customise the duration of each interval between periods of play in the game. By clicking on the ‘plus’ or ‘minus’ buttons, or dragging the pointer on the horizontal progress bar, the user can set the period duration in minutes to last between 1 and 15 minutes. To confirm their selection the user must click on the ‘Proceed’ button. By clicking on the ‘Back’ button the user can return to the previous screen.

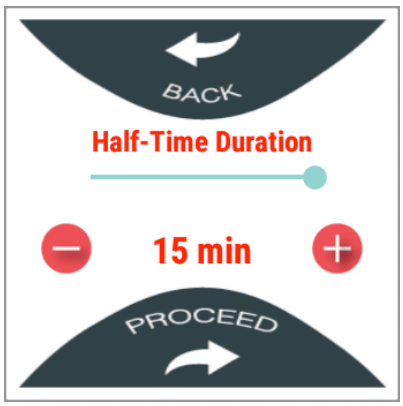



Figure 6: Half-Time/Interval Duration Selection screen.

## Main Match Screen

After customising their preferred duration settings the user will access the Main Match Screen (see Figure 7) from where they can begin tracking the game's time and record match events which will form the basis of their post-game report. The Main Match Screen contains a number of visual aids including:

- Match duration timers.
- Scoreboard displaying the number of goals scored by each team.
- Notepad which displays a record of match events for each team.


Figure 7: Main Match Screen.

### Starting a Match

To begin the game's timer function, the user must click on the 'START' button. The large black countdown timer will begin to count down the period's duration, as selected by the user in the Period Duration Selection screen, until it reaches zero. The smaller black elapsed game duration timer, which runs continuously throughout the game until the user chooses to end the period of play, will count up from zero and displays the total elapsed game time.

### Recording Match Events

The user can record match events by accessing the Match Event Selection Menu (see Figure 8). After starting the match, the menu can be accessed with a subsequent click of the countdown timer.


Figure 8: Match Event Selection Menu.

## Match Event Selection Menu

The *Match Event Selection Menu* provides the user with a number of useful options, including the ability to record the important match events in line FIFA's 'Laws of the Game'. The menu's options include:

- **Goal**: Record any goals scored during the match. The user can select the scoring team ('Home' or 'Away') (see Figure 9) and the shirt number of the goalscorer (see Figure 10); matches containing players with no shirt number are accounted for with the 'N/A' option, while own goals can be recorded by selecting the 'O.G.' option. A record of the goal is stored in the form of a soccer ball icon (see Figure 11) on the Main Match Screen in the designated team's 'match event card'.


Figure 9: Goal Team Selection screen.

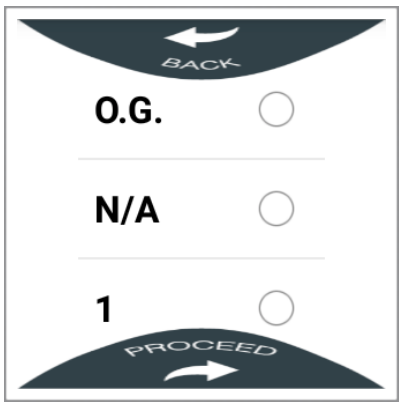



Figure 10: Goal Player Number Selection Screen.

Figure 11: Goal icon.

- **Penalty Kick**: Record any penalty kicks awarded during the match. The user can select the team allocated the penalty kick (see Figure 12), the number of the player designated to take the kick (see Figure 13), and the outcome ('Scored' or 'Missed') (see Figure 14). A record of the penalty kick is stored in the form of one of two icons - one representing a scored kick, another a missed kick (see Figure 15) - on the Main Match Screen in the designated team's 'match event card'.


Figure 12: PK Team Selection screen.

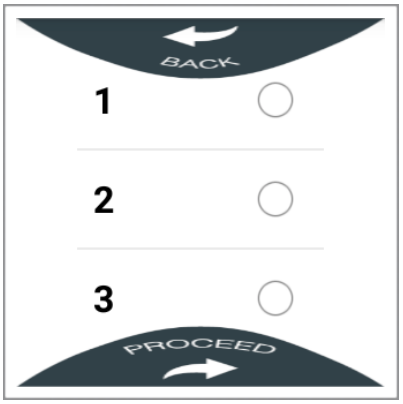



Figure 13: PK Player Number Selection Screen.

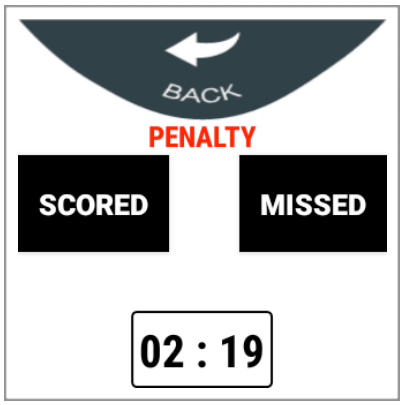



Figure 14: PK Outcome Selection screen.

Figure 15: 'Scored' penalty kick (left) and 'Missed' penalty kick (right) icons.

- **Substitution**: Record any player substitutions processed during a game. The user can select the team changing their personnel (see Figure 16), the number of the outgoing player (see Figure 17) and the number of the incoming player (see Figure 18). A record

of the substitution is stored in the form of an icon containing a red arrow and a green arrows (see Figure 19) on the Main Match Screen in the designated team's 'match event card'.


Figure 16: Substitution Team Selection screen.

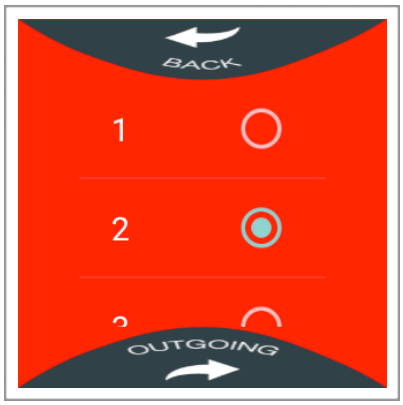



Figure 17: Substitution Outgoing Player Selection screen.


Figure 18: Substitution Incoming Player Selection screen.

Figure 19: Substitution icon.

- **Disciplinary Action**: Record any yellow or red cards issued to players or staff during a game. The user can select the team to which they wish to allocate the disciplinary card (see Figure 20), choose between players or staff members ('Coach' or 'Other') (see Figures 21 and 22), and the type of infringement committed by players (see Figures 23 and 24). A record of the disciplinary action is stored in the form of an icon representing a yellow card, red card, or yellow and red card for two yellow card offences resulting in the issuing of a red card (see Figure 25). The icon is displayed on the Main Match Screen in the designated team's 'match event card'.


Figure 20: Disciplinary Action Team Selection screen.


Figure 21: Disciplinary Action Player/Staff Selection screen.


Figure 22: Disciplinary Action Staff Role Selection screen.

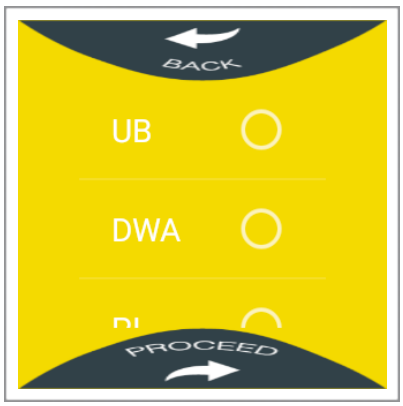



Figure 23: Disciplinary Action Yellow Card Infringement Selection screen.

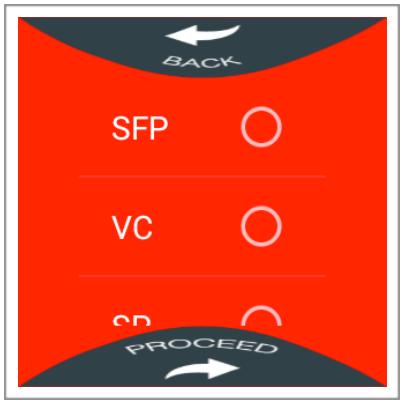



Figure 24: Disciplinary Action Red Card Infringement Selection screen.

Figure 25: Yellow card (left), red card as the result of two yellow cards (middle) and red card (right) icons.

The selectable infringement types are provided in an abbreviated format in the application. A full description of each infringement type is provided in Table 1.

| Yellow Card Infringements | | Red Card Infringements | |
|---|---|---|---|
| **Abbreviation** | **Meaning** | **Abbreviation** | **Meaning** |
| UB | Unsporting behaviour | SFP | Serious foul play |
| DWA | Dissent by word or action | VC | Violent conduct |
| PI | Persistent infringement | SP | Spitting |
| DR | Delaying the restart | DGH | Denying a goalscoring opportunity via handling |
| FRD | Failing to respect the distance (10 yards) | | |
| EWP | Entering without permission | DGFK | Denying a goalscoring opportunity via a free kick offense |
| LWP | Leaving without permission | | |
| PDM | Playing in a dangerous manner | AL | Abusive language |

Table 1: Yellow and Red Card infringement abbreviations and associated meanings.

- **Abandon Match**: The user can abandon the current match while retaining all saved match data. They will be prompted to confirm their selection (see Figure 26), with the 'Yes' option navigating them to the Main Menu screen and the 'No' option returning them to the Match Event Selection Menu.


Figure 26: Abandon Match confirmation prompt screen.

- **Quit**: The user can quit the current match and discard all saved match data. They will be prompted to confirm their selection (see Figure 27), with the 'Yes' option navigating them to the Main Menu screen and the 'No' option returning them to the Match Event Selection Menu.


Figure 27: Quit confirmation prompt screen.

Match events are stored in the Main Match Screen in icon form providing an easily accessible running record of the match's key events (see Figure 28).

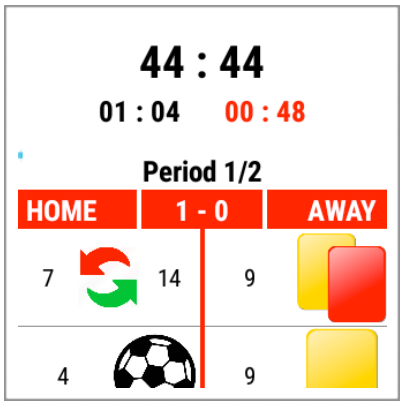



Figure 28: Main Match Screen displaying a record of match events.

## Progressing Between Periods

If the user selects more than one period of play, it will be necessary to signal to the system when you are ready to progress from each period to the interval and on to the subsequent period. When the countdown timer reaches zero, the user's chosen alert - audible or vibration - will signal the end of regulation time plus time added on for stoppages in play; the match is therefore in additional time. Additional time is also highlighted by a change in the screen's background colour from white to green (see Figure 29). During the additional time the Match Event Selection Menu can still be accessed by clicking on the large black countdown timer. By clicking the smaller black elapsed game duration timer, however, the user will commence to the interval. During the interval the screen's background colour changes to yellow, and the Period Indicator changes to 'Interval' (see Figure 30). The Match Event Selection menu is no longer accessible during the interval, which can be manually ended by the user when the large black countdown timer reaches zero by

clicking the small black elapsed game duration timer. The subsequent period of play will then begin and the elapsed game duration timer will be reset to display the total time from previous periods minus any additional time.


Figure 29: Main Match Screen during additional time.


Figure 30: Main Match Screen during an Interval between periods of play.

---

- **Reports**

The ‘Reports’ menu presents the user with the ability to export a post-match report containing saved match data to their connected *Android* mobile device. The ‘Reports’ interface (see Figure 31) displays a list of saved match data from which the user can choose before confirming their selection by clicking the ‘Export’ button and selecting a storage destination in their wearable device’s file directory (see Figure 32). By clicking on the ‘Back’ button the user can return to the main menu.

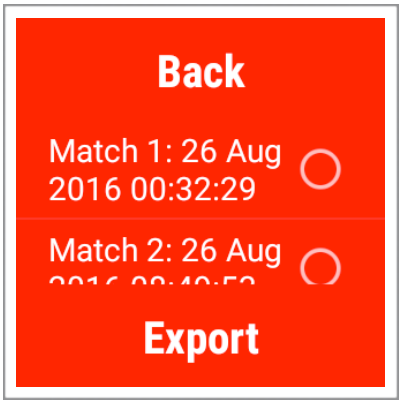



Figure 31: Reports interface.


Figure 32: Selecting file directory location for saved file.

---

• **Options**

The 'Options' menu presents the user with the ability to customise the application's alert settings (see Figure 33). The two available options are:

- Audio: The audio alert setting will play an airhorn sound to alert the user when their chosen period duration has expired. Bluetooth headphones are required for devices with no in-built loudspeaker.
- Vibration: The vibration alert setting will produce a tactile 'vibration' to alert the user when their chosen period duration has expired.


Figure 33: Options interface.

---

• **Quit**

The user can quit the application by clicking the 'Quit' main menu option.

---

# Appendix Ten
## Android Studio Project

The Android Studio project files containing the application's source code is located in an attached CD-ROM. The project files can be opened in Android Studio and compiled in order to test the applications layouts and functionality using the emulator.