

ScoreCard Application

Milestone 2 – Project Specifications and Early Implementation



December 13, 2013

TEAM 6: Cheung Chen, EdWIN OLIVO, RYAN PACIFICI

GREG REINHOLD, JOESPH SULLIVAN, YU ZHANG

1. **Introduction**
   1. Background and Motivation

In professional sports, score keeping is important because it quantifies a team’s success and an individual’s achievements. Although this may hold true for non-professional athletes, professional athletes dedicate their lives to achieving team and personal success in sports. Some scorecards have historical significance; like those for a perfect game (baseball) or 500-plus yards passing (football) or over-60 point performance (basketball). These written records will remain after the moment is long over and fans can reminisce about these moments through the scorecard (including every out, every completion, or every shot taken).

The project to be developed is a scoreboard application that can track statistics at sporting events. There are existing products similar to this proposed application, but they either only provide the ability to track a single sport, or are simply too complicated to use. Thus, the goal of our project is to make a user-friendly application on Android platform that can keep track of game statistics for basketball, baseball, football, and soccer.

* 1. Document Conventions

During the documentation process for this project, the team will attempt to write in a professional manner. Each document produced will be divided into content oriented sections. There will be titles for each section and sub-section (if any), each denoted by bold fonts, italicized fonts, and/or underlined phrases.

Working in a technical field, there may be terms that readers may not understand. So there is a need for a glossary at the end of the document that will define some of these terminologies. Furthermore, because this project deals with sporting events, there may also be a need for a glossary of sports terms that are used throughout the documentation process. It cannot be assumed that everyone who reads the documents is knowledgeable about sports and its terms.

Throughout the course of the design, implemention, and testing phases of the project, the team may decide to make changes to the documents. To recognize that changes have been made, there will be an additional section stating what was previously written and the changes that were made. The date and the reason(s) for the change will be detailed in this portion of the document as well.

Developers may add notes into documents regarding their thoughts about a particular feature or process that they feel is questionable. These comments will be in a more informal tone. These notes are intended as reminders or ideas for future endeavors. In the documents, such comments will be written in red colored fonts so that they are distinguishable from other text.

* 1. Team Members and Task Breakdown

Team 6 consists of six members, all Computer Science and Engineering majors at the University of Connecticut, but each has a different skill set and attribute to contribute. Joseph Sullivan is working on a minor in Digital Arts and Cheung Chen is quite talented at drawing and designing. They are valuable assets in designing the interfaces for the project. Ryan Pacifici spent much of the past summer (of 2013) working with databases, so his insight on data storage and management is tremendous for the team. Greg Reinhold is versatile and he is proficient at all levels of software engineering. He is also an avid sports fan. His knowledge of sports and his versatility makes him an intricate part of the team.

The developers are each focusing on a unique part of the application. The outline for the task breakdown is as follows:

* Cheung is primarily working on the user interface that the user will interact with when scoring a game. This includes the appearance the screen will take on during the score keeping as well as the flow of events when certain buttons are pushed. He also is designing the interactive field of play, such as the shot chart for basketball.
* Ryan is heading the development of the database of statistics. His main job is to create a central database that will be updated as the user is keeping score and will be able to update the statistics that are being viewed by people on other devices.
* Greg is implementing the viewing of the statistics on the devices. This will involve deciding which stats are relevant to each sport as well as obtaining the information from the databases and displaying them in a box score format that is simple and easy to understand for the user.
* Joe will be the primary graphic designer to ensure the final product is visually appealing. He is also in charge of creating the mockups for our design so that we have our plan well documented before we are too deep in implementation. Finally, he will be assisting Ryan in database setup and maintenance.
* Yu will be focusing on the real time sharing between the scorekeeper and the game’s viewers. His main goal is to complete the live-updating capability so that a user can watch the live play-by-play of a game.
* Edwin will be assisting Cheung with the development of the user interface. He will create some java and xml classes to facilitate the UI implementation. Since he likes soccer is currently implementing the soccer interface.
  1. References

<http://developer.android.com/index.html> Android development manual

<http://developer.android.com/sdk/installing/studio.html> Getting Started with Android Studio

1. **Overall Description**

2.1 Product Function and Perspective

The main functionality of this application will be game data recording. A user will be able to set up different games, players, and other data related to the sport that is typically stored for statistical purposes. The user can then view various calculated statistics filtered by game, player, and other relevant information. Another function would be to use added networking capability to allow for a user to broadcast in real-time the game that they are tracking. This would allow for other users to connect and view that game on their own mobile device and see the updates that the broadcasting user makes.

2.2 User Classes and Characteristics

There are three planned user types for this application. The typical user is someone who is keeping track of game information for personal use. This user would not need networking capability to retrieve information that other users have saved. This user would only have access to the information that they have recorded into the application themselves. With the added feature of being able to broadcast a user’s game recordings, two new user types are added. One of these would be a person who is broadcasting their game recording. This user would have the same options as the classical user would to score and record statistics of a game. The other user type would be a viewer user. This user would input the credentials needed to access a game that is being broadcasted. Once this user has connected to a broadcasted game, their screen will mimic that of the broadcaster’s screen, however they will not be able to add or change any of the information that is present.

2.3 Developing Environment

The application may be developed using one of two possible development environments. Firstly, the team could use the Eclipse development environment to program in Java. The developing team would then use an Android emulator plugin or an Android phone itself with Eclipse to test out application during builds. Eclipse is a good option because the developers are very familiar with the environment. This would cut down on the overhead time of learning to work with a new software development environment.

However, recently, Google has developed Android Studio, a development environment designed specifically for Android application. Both Android Studio and the Eclipse environment have a variety of useful aspects in developing Android applications. More specifically, the ADK (Android Development Kit) offers template-based wizards to create common Android designs and components, a rich layout editor that allows developers to drag-and-drop UI components, previewing layout on multiple screen configurations, and more.

The use of Android Studio has given the team some troubles in the early stages of implementation, leading to an increased use of the Eclipse plug-in. Since Android Studio is still in the early stages of development, there are some bugs that have affected a few team members, making it simpler for the team to rely on the Eclipse environment.

We are currently using both developed environments. Cheung is using Eclipse with android plug-in, and Edwin is using Android Studio. The java and xml classes work the same in both environments. For testing, we are using Android 4.3 and 2.3 emulators from the SDK manager package, these emulators are usually slow but they are very useful

2.4 User Environment

This application is targeted for Android phone with the potential to expand to iOS devices and tablets (iOS and Android), in the future.

2.5 Design and/or Implementation Constraints

Hardware is the largest constraint since the application will be designed for Android phones. Also, since this application is intended for mobile device use, a major constraint is speed/performance. There are many devices currently available on the market so the team must make a conscious effort to create efficient code that will react fast on a wide range of devices. Along with speed is network connectivity. Since there will be features that use the network to communicate with other devices running this application, there is a need to design the application in compliance with standard network protocols. Personal constraints include learning how to program for the Android platform as well as network and database use.

2.6 Assumptions and Dependencies

Interfaces will be created with multiple platforms in mind. This will be done to create a unified product, should users use the application on different devices. Since Android has a variety of screen sizes, it is important that the layout isn’t hard coded so that it can fit on the screen of each device.

The design will also be modular so features can be added and removed based on quality/necessity, as this type of application will constantly need more advanced features to remain relevant. It is also assumed that the device will have access to the internet (at least periodically).

1. **External Interface Requirements**

3.1 User Interfaces

By using this software, the user will mostly interact with the data collecting interfaces. Therefore, it is of great importance to make these interfaces user-friendly and appealing. The completed project will consist of three or more such user interfaces, one for each sport that is implemented into the application.

3.1.1 *Basketball Interface*

The basic layout of the basketball interface is as follows: the top portion of the screen will be used for keeping track of the more important game statistics (current scores, team fouls, timeouts remaining, and possibly the time remaining in the game), and the lower section will be used for various control buttons. The majority of the screen will be an interactive image of the basketball court for the user to use as an input mechanism.

The most important aspect of the user interface for basketball is the interactive court. The user will tap on a particular spot on the basketball court to indicate that a particular shot is taken. A pop-up bar will appear to allow the user to choose which team and player took the shot. After the user makes his or her selection, the application will then ask whether the shot was made or missed and whether an assist was accredited to another player.

Other than taking shots, there are other statistics involved in the game of basketball. To record these stats, the buttons on the lower section are designed for the user to keep track of these plays. Offensive and defensive rebounds, steals, blocks, turnovers, and personal and technical fouls are just few examples of these auxiliary basketball stats. When these buttons are pressed, they will pop-up a menu that asks for more detail about the play that occurred. For rebounds, turnovers, and technical fouls, it will ask for the perpetrator. For steals and personal fouls, it will need the victim as well.

The last part of the interface will keep track of score and timeouts. In this section of the interface, the user can tap the timeouts in order to record the use of a timeout. Other than this, this portion of the interface will mostly be controlled by inputs from the user to other parts of the interface.

3.1.2 *Baseball Interface*

The top of the baseball interface will keep track of the more important statistical data in a baseball game: runs, hits, errors, the inning (top or bottom) and the balls and strikes. Below the scoreboard will be a baseball field. At times, this image will just depict the base-path conditions of the game. At other times, it will become an interactive image to allow the user to input information about the game (base-running). At the bottom will be a toolbar consisting of icons representing the most common plays in baseball.

The most important aspect of this interface is the icons on the bottom of the interface. These icons will allow the user to record what is happening in the ballgame. In this toolbar will be the most common plays in baseball: hit, out, foul, strike, and ball. The strike icon will divide into two icons (strikeout swinging and strikeout looking) when the batter has two strikes. There will be one additional button that will extend the toolbar, opening a list of lesser common plays in baseball: bunting, wild pitch, hit by pitch, etc.

Depending on which play occurs, the application may ask for more information. For example, if there was an out, then the program may question for what type of out it was (fly-out, ground-out, etc.). Another would be hits; it could be a single, double, triple, or a home-run. After asking for that information, the interactive field will appear to ask for where the ball landed or where it was caught. The interactive baseball field will stay and the user will be asked to make any base-running moves that happen.

Very unusual plays may happen in baseball games, so this is just a simple preliminary design of the baseball interface. It may not be possible to list all features and details until the implementation of the basis of this interface is complete.

3.1.3 *American Football Interface*

Like the previous two interfaces, the football interface will have the same graphic features: scoreboard, an image of the field, and buttons that help record information. In this interface, the buttons will once again serve as the most vital aspect. At first, there will be three icons (pass, rush, and penalty) for the three different ways to move a football. There will then be a series of queries asking for additional information about the play (yardage gained, direction of the play, players involved, and whether a fumble occurred).

Due to the volume of players in a football game, it should be necessary for the user to input information about the players during the game. In football, there could be three wide receivers. After recording a few plays, if only the generic name of WR1, WR2, and WR3 are allowed, then it could get confusing who each really is. So in the football interface, the goal is to allow the user to add name (or uniform number) on the go to lower the chances of possible confusions.

The scoreboard interface for football should mimic that of basketball with slight modification of the graphical style. The statistics to be listed on the scoreboard include timeouts remaining, the time remaining, and the current score. As for the football field image that will cover most of the interface, it will serve as a tracker for the current drive and statistic. Unlike the other two interfaces, the football field will probably not be involved in the inputting of game statistics.

3.1.4 *Soccer Interface*

The score of the soccer match will be kept at the top of the screen along with the time remaining in the score game. Because the time is kept at this location, the developers believe it is also a good location for the button to add additional time to the game (which is unique from the other sports).

Like the other interfaces, soccer will have its own unique set of buttons for the different plays that occur in a soccer match. Most important statistics to keep in a soccer game is goals and assists. But other statistics such as fouls, yellow and red cards, free kicks, and corner kicks are valuable as well. Of course, the goalkeeper’s hard work should not be forgotten, so there is a need to record saves and save opportunities for the goalkeeper. Lesser important stats that may or may not be recorded include tackles and passing because they occur too frequently and too fast in most occasions.

In this implementation, there will be an image of the soccer field that will allow the user to pin-point the location in which a recorded play occurs. Because the soccer field is quite large and the mobile display is quite small, it should allow the user to zoom-in or out of a particular site on the field, so that the marking of a location is as exact as possible.

3.2 Hardware Interfaces

The primary mobile platform that this application will be compatible with is the Android operating system. Linux-based, this operating system is mainly designed for touchscreen mobile devices and tablet computers. Android phones have screens that range from the smaller 3.2 inch screens (earlier phones) to 6 inch displays (the latest Android-powered phones). For tablets, the screen size ranges from 7 inches to slightly over 10 inches.

As for the CPU, older smartphones had one CPU core. Nowadays, dual-core and quad-core phones are not uncommon. Most mobile devices boast a clock rate of over 1 GHz. As for space, the phone may have a smaller 16GB internal space, but with the improvement in SD cards, the space in phones can be extend to an impressive 128GB.

3.3 Software Interfaces

In a mobile application, much of the interface involves user interactions. In this section, the interfaces that will be detailed are those that the user will not have active interactions with. These are interfaces that show results of calculations to the user or that the user can completely ignore. To see a few mockups concerning the software interfaces, as well as a further description about what functions each one will complete, please refer to **Appendix B**.

3.3.1 *Start-Up Interfaces*

At startup, the program will allow the user to select what type of sporting event he or she is recording. Afterwards, the user can input the names of the team and player, or the uniform numbers of the players. Here, the application will not force the user to input information. If the user does not input anything, there will be a generic name given to the players. For example, in basketball, it could be PG (point guard), SG (shooting guard), C (center), PF (power forward), SF (small forward) and BN1 through *N* (bench players 1 to some number *N*).

3.3.2 *Data Collection Interfaces*

The individual features of the scorekeeping interface are detailed in section 3.1. This sub-section will explain some common features that will be implemented in all the different data collecting interfaces.

On all the interfaces, a button to undo the most recent entries to the database (removing the latest input) will be implemented. Scoring games can be very difficult so in the design of a scorecard program, a margin for error must be set for the user. Another feature would be to allow the editing of a previously inputted record while the game is still being recorded. It would not be a good idea to allow only editing of statistics at the end of the game because the user may forget the change he or she desires to make.

One thing to note on the data collection interfaces for basketball and football is that game time is recorded. There have been debates among the developers about this feature because any time-keeping features may be very difficult for the user to use. So the team will add in this time component, but will allow the user to choose whether to enable or disable this feature. In addition, if the user makes a mistake in time-keeping, there will be a method to correct the timing mistakes.

3.3.3 *Analysis Interface*

After the game is played, the user can look over the game statistics that have been recorded. Like the box-score features found on sports websites, it will give an overview of the game. Here, there will be additional options to edit the statistics, just in case that there was a mistake that was not realized during the game. In addition to the box-score, there may be an option to view a written recap of the game (unconfirmed feature). Also in this post-game interface will be buttons for the options to export the game or send the game as a file to another person (so there may be icons of Facebook, Gmail, Twitter, and/or other social media sites).

3.4 Communication Protocols and Interfaces

A goal of the project is to allow users to share the scorecard they recorded with other people. In today’s society this can be achieved through social media sites. In this application, a share feature will be implemented. The program will convert the scorecard into a data sheet and allow the user to share it through the method he or she prefers. Currently, the expected methods that are to be implemented are email (especially Google and its technologies) and Facebook. In the application’s interface, there should be a share button that will initiate this feature. The team also desires to be able to send live updates to other devices, which would require a central database to communicate between the users.

3.4.1 *Database*

For saving and accessing statistics, the team has decided to use a database. This decision will allow for easier customization for searches. With this scorecard, the user will be able to view statistics grouped in a number of ways, such as by an individual player, by team, by season, and other criteria. By using a database and SQL queries, having searches be flexible will allow for different types of searches to be implemented relatively easily. In addition, adding new search criteria at a later point will be easier to implement with a database, as a new SQL query can be created to search upon this new criteria, giving the application greater flexibility.

MySQL will be used as the database platform. One advantage of using MySQL is that there is no cost associated with using or setting up the database with MySQL, unlike a number of other popular systems such as Oracle. In addition to this, the team has some experience in using MySQL, which will help cut down on a part of the learning time associated with this project. The database will be accessed through specific java functions that will be written for this application. A JDBC driver will allow the java code to interact with the database to perform the required functions.

The database will be hosted on a server, in which all users will access. To ensure each user’s data is secure and private, each user will have a dedicated schema that by default only they have access to. This setup will prevent users from modifying another user’s data. By setting the database up in this manner, it also becomes possible to make information sharing simple, by choosing to give another user read and/or write privileges to their schema.

Inside each schema, there will be a number of tables that will be used to store the various data related to tracking a game, team, and player. Each game, team, and player that is stored will be given a unique identifier that will be used for queries. There will be several tables that will map each of these identifiers to their respective hierarchy. For example, there will be a table that will map what players are on a specific team, and another table that will map which teams were involved in a game, and which sport that game belonged to. These mapping tables will allow for quick navigation through the database to retrieve needed information. When storing information into the database, only the critical game information will be saved. Many statistics are generated through calculations involving the critical data from a game or a player. These values will be calculated as needed instead of being saved into the database. This method will cut down on the amount of data that is stored and will remove the need to constantly update these values every time the critical data changes.

*3.4.2 Sharing Information:*

One benefit and feature of the application will be the ability to share results and statistics with other users. With the default settings, only the user will be able to read or modify their statistic information. This ensures that the data cannot be tampered with by other users. A user will be able to give permissions to other people on a user-by-user case. This feature will allow for sharing all of a user’s recorded statistics with only the users that they choose. Write privileges can also be given to other users. This feature will be implemented at the database level, using schemas and user privileges. For more information on this setup, please see the database section of this document.

1. **System Features**

4.1 Description and Priority

The key component of the project is the interface that users of this software will interact with when keeping scores at a sporting event. Ideally, the design of the user interface will be graphically appealing and user-friendly. Once the interfaces for inputting statistics are completed, the next step is to develop the methods to present the information after the game.

With a completed interface, the next important feature to include is a method to store and send the recorded statistics. Numerical data can easily be stored in and loaded from a small file which can be saved onto the mobile device (initially to the Android OS). It is also desirable to have a central database that will store the data that is entered by every user.

Having an interface and a method of data storage, it should be required that the different sport scorecards are integrated into one application. One of the objectives of the application is to provide the ability to keep score for multiple types of sporting events with one mobile app. So integration of the different scorecards is mandatory.

Also desired is the ability to send the scores of a game to another person. The team can utilize the various social media tools prevalent today to achieve this goal. People often keep scores so that they can inform others of what happened at a sporting event, so this sharing feature will make possible.

For more advanced features, the team hopes to add an analysis tool that will take the game data collected and calculate advanced sports metrics. (Sabermetrics and football efficiency ratings, etc.) To even further this feature, the developers could attempt to develop a method of taking the recorded stats and generate a written review of the game.

Perhaps this software can be extended to keeping the statistics for a particular team. So instead of keeping score for only one game, game statistics for one team can be accumulated. Then, rather than analyzing stats for only a single game, the program can be used to see a player or a team’s progress over an entire season. If one person cannot attend every game, then this can be a feature to allow different users to synchronize their scorecards to complete collection of all games played by a team.

The developers could then try to implement the same software on the iOS or making the interface compatible with Android tablets. Having multiple mobile platforms with the ability to use this software will increase the popularity and the usefulness of this application. Because the backbone of the application is already designed and implemented, this transferring process should be feasible.

This project does not have to end here. To extend beyond the already listed features, the team can develop a server that keeps all the information recorded. If this is implemented, then it opens other possibilities to the project. These may include: changing methods in which the data is stored (from in mobile device to in a server) and instant updates of scores to others who are keeping track of the game through this app (live-feed). Nonetheless, these are very advanced features that will be implemented only after perfecting all the other higher priority feature.

4.2 Action Plan

The development process will be divided into four milestones. The team has set goals to be completed at the end of each milestone. The following is the four milestones, their dates, and what the developers should have accomplished by that milestone:

* *Milestone 1:* October 15

By this first milestone, the team must have an understanding of the scope of the project and have a design document detailing this project. Furthermore, the developers should have a good understanding of the platform that they will be using in the development process.

* *Milestone 2:* December 1

At this second milestone, the team should have solidified most if not all the attributes of the project (resulting in a finalized design document). In addition, members of the team must decide whether they would like continue on with this project. If they will, the team would have already started the implementation process.

* *Milestone 3:* January 15

This third milestone would be roughly the start of the spring semester. If the members are willing to work over the winter break, then most (preferably all) of the interface features should be completed. And integration of the interfaces should be starting, as well as the development of saving features and communication (sharing features).

* *Milestone 4:* March 1

In this juncture of the development process, the application should somewhat resemble the finished product. The interface, saving and loading features, and the communication features should be completed. The team should be working on testing and perfecting existing features. If that is completed, then members should be working on implementing advanced features.

* *Milestone 5:* April 15

By this final stage, all desired features are implemented, tested, and perfected. Any additional advanced features that are developed should be tested as well. The team should also be prepared for a presentation on their project.

4.3 Functional Requirement

Upon completion, the product should (at minimum) be able to:

* Allow users to easily input sport statistics and names (players and teams)
* View the statistic record (during and after games)
* Store the statistics into a file and load them at later times (to view)
* Share the scorecard with other people through email or other social media options

If time permits, this application may also be able to:

* Analyze the data recorded and report something useful to the user
* Player or team tracking system (collection of all games played by one team in a season)
* Creating a server and implementing a live-scoring system
* Be supported by multiple mobile platforms (phones to tablets, Android to iOS)

1. **Other Nonfunctional Requirements**

5.1 Performance Requirements

As there are many different performance requirements that must accounted for, they have broken down into the list below:

5.1.1 *Response Time*

As an app that is going to be used to keep track of live sporting events, the user should not be able to notice any lag when performing routine actions within the app. Since most of the in-game actions that will be performed aren’t computationally intensive (i.e. record a rebound in basketball, a penalty in football, etc.), it shouldn’t be difficult to avoid lag.

However, a seemingly-instantaneous response time isn’t required for all of the features of the app. For example, sharing the game updates across the network can have a few seconds or more of lag and not diminish the experience of those following along remotely. The goal is to limit this lag as much as possible and aim to allow the live-feed to update in a seemingly real-time manner.

5.1.2 *Workload*

Since the app is only used by one person at a time on any given device, workload isn’t a factor on an individual basis. However, once again it is the real-time updating that is the developer’s main concern for this performance requirement. Luckily (or unluckily, depending on how one looks at it) it is not expected that the app being used by professional teams with large fan-bases, meaning large scale support will not be needed.

The plan will be to have support for about 15-20 people to view the live feed of the game without considerable delay. This will allow a small local team to give support of live stats to their fan-base. For example, the parents of a team of Little Leaguers or Peewee football players will be able watch their children even if they couldn’t make it to the game.

5.1.3 *Reliability*

As mentioned before, the app must be flawless when the user is in the process of keeping track of the stats during a live game. The app cannot crash during the middle of the game or else the user will miss their chance at recording a large portion of stats and risk losing any unsaved data. The fact that the scorekeeping process isn’t very computationally intensive should help limit crashing and maintain the reliability that is desired.

However, yet again the biggest issue in regards to this performance requirement will be a result of the live feed of the games. Not only will the app have the potential to crash for the people who are downloading the live updates, but it also could cause issues with the scorekeeper himself. It is possible that the constant upload of information will be too much for the app, causing a slow response time or even a crash. This will need to be monitored as development continues.

5.2 Security Requirements

As of now, there are no foreseeable reasons for requiring any sensitive or confidential information in order for the app to run effectively. The app may require some sort of username and password associated with an email in order to sync accounts, but this plan will be further developed as the networking implementation is drawn up. In order to accommodate a username and password, basic encryption and verification from a central server would be needed. However, if it is decided it would be more effective to store information directly on the user device instead of having a central server, the app would not need to store emails and passwords and the security requirements would be greatly diminished. Since the statistics of a Little League baseball game aren’t remotely confidential, there isn’t a risk of the information being intercepted as it is sent from the device to the server.

5.3 Software Quality Attributes

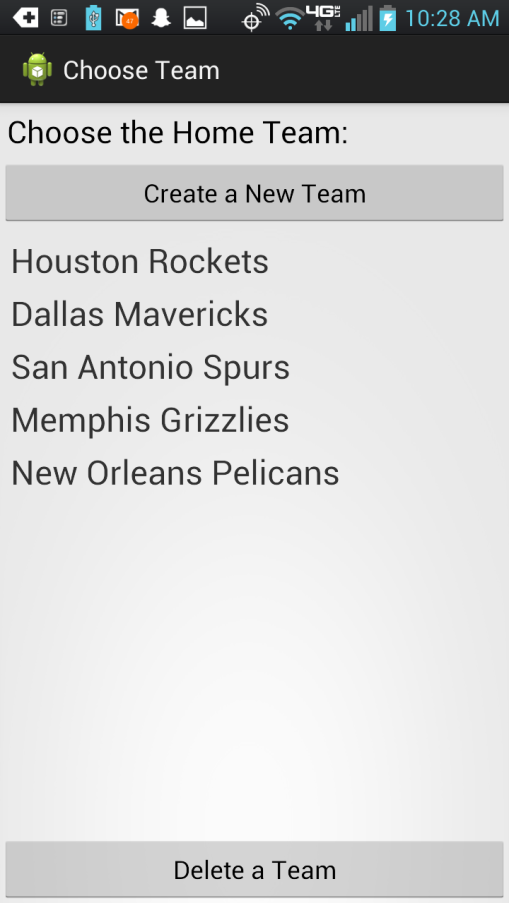
One of the main concerns with this app is to create a single suite of scorecards that are simple and intuitive for the users. In order to accomplish this, the team wants to have consistency from one sport to the next, such that the interfaces and basic flow is similar regardless of the sport. While there will be many differences that are necessitated by the fact that each game is unique, it should be intuitive to the point that the user will feel comfortable with all three sports even if they only did a tutorial for one. Part of this consistency has to do with the reusability of the code. By using similar menus and game flow, each scorecard will be uniform throughout the app.

5.4 Project Documentation – Will be completed once the project is further along in the development process.

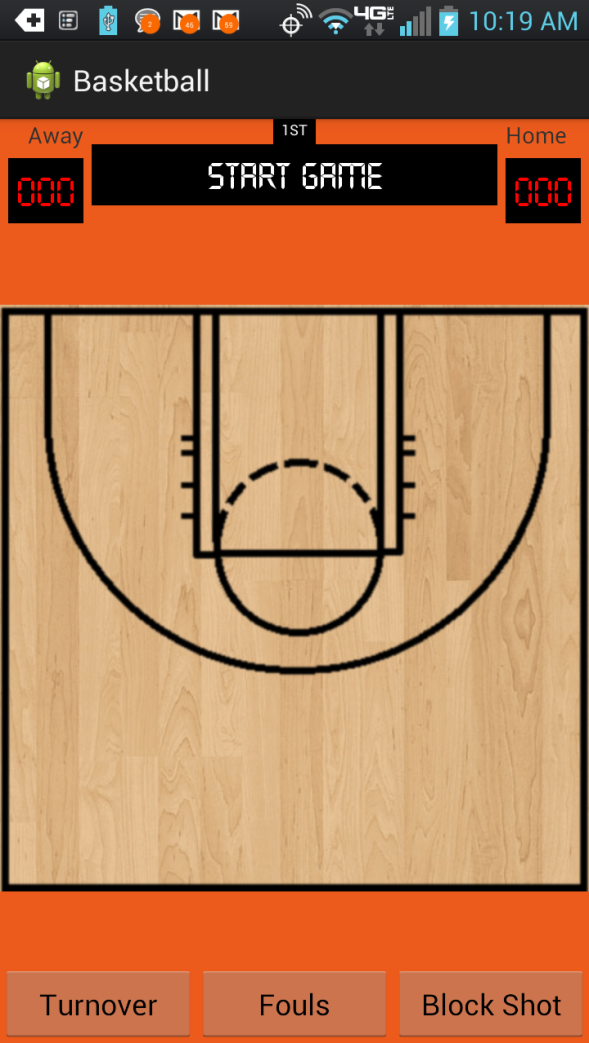
5.5 User Documentation – The following is a simple description of how a user will be using the TUSK application through the use of screenshots pulled from preliminary implementation of the design. In order to get a more in-depth description of the functions of each screen, please refer to the mockup descriptions in **Appendix B**.



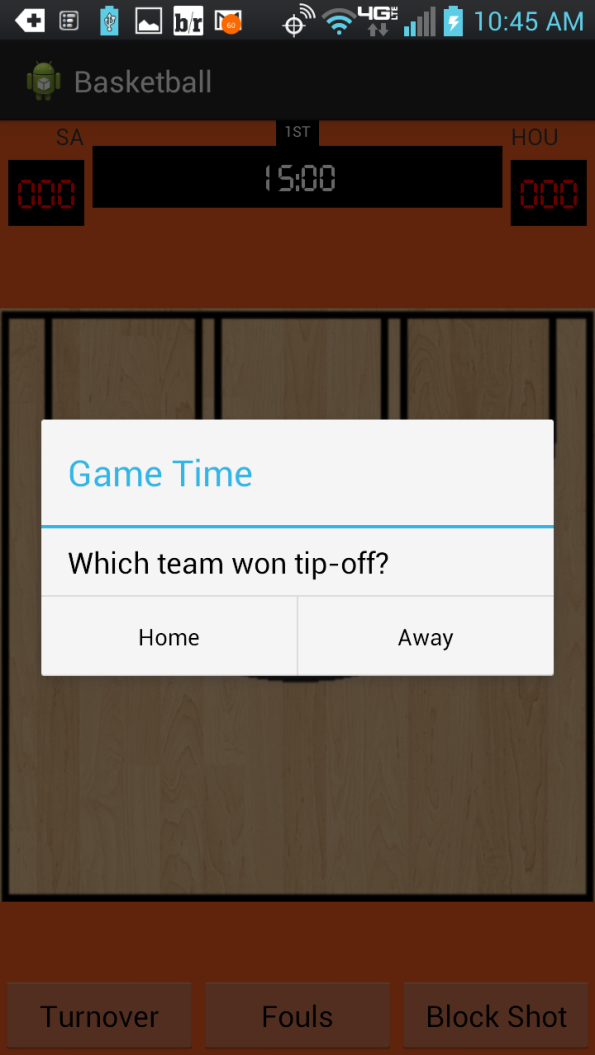
The home screen has all four icons, each of which will lead to a new game for their respective sport. It also has a ‘View Statistics’, ‘Options’, and ‘Live Tracker’ button.



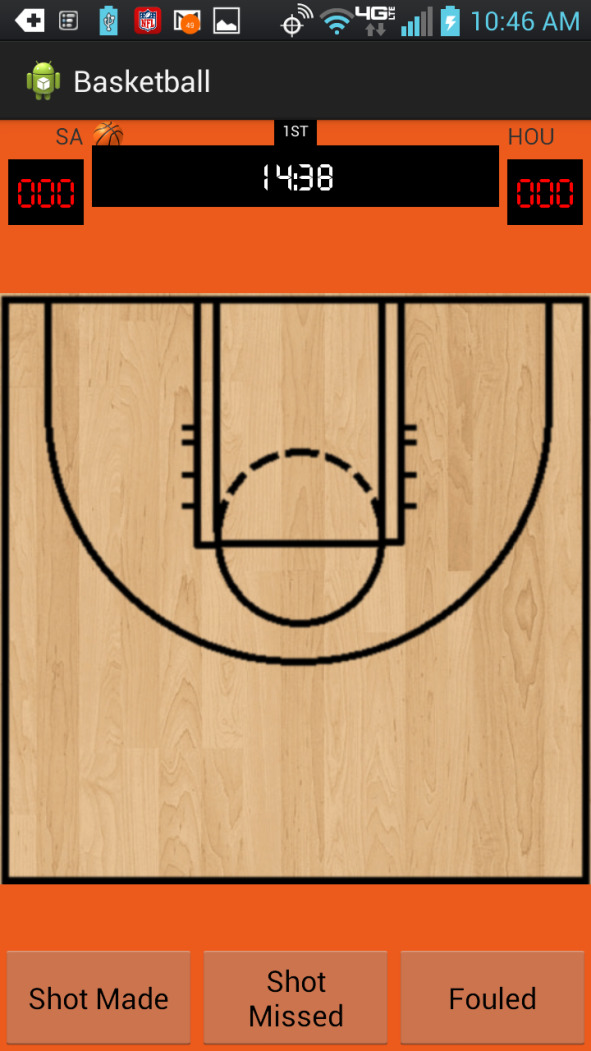
The Choose Team menu allows the user to create a new team or select pre-existing teams to keep score of.



The basketball interface has the score of the two teams in the upper corners as well as the clock centered between them. The court in the center is interactive, while the buttons below perform the scorekeeping functions that their text implies.



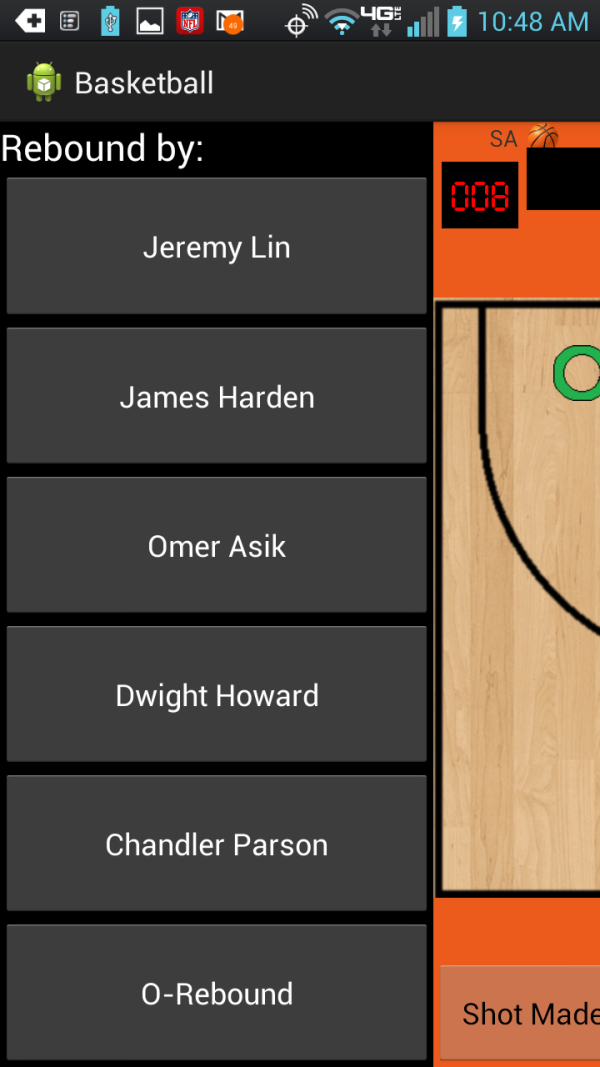
To start a game, click on “START GAME”. This will create a pop up that asks the user which team won the tip off to begin the game.



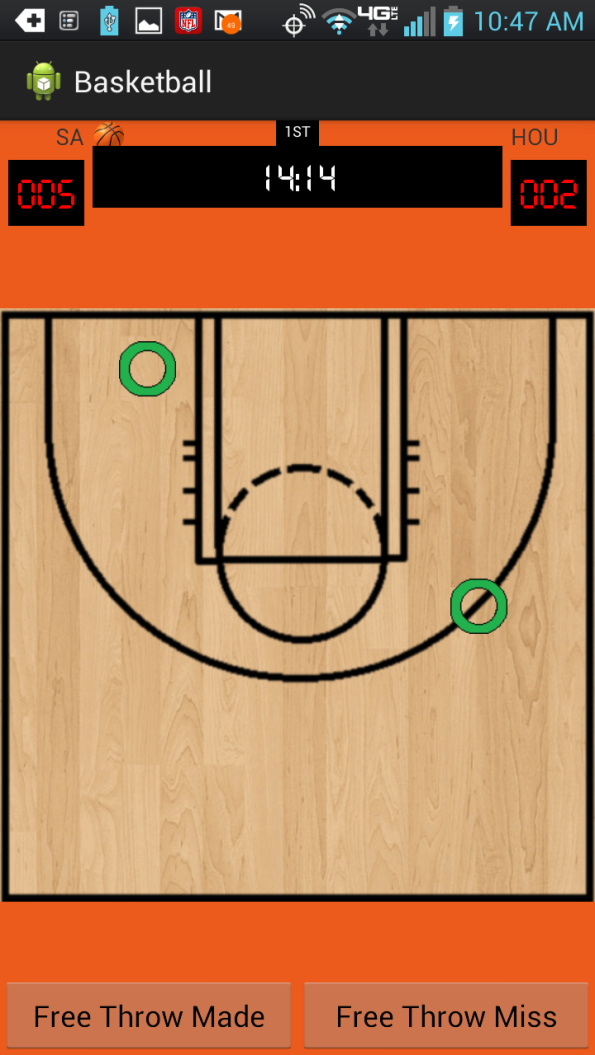
Once possession has been established, the timer begins. Touching anywhere on the court results in this screen, where the user is prompted to select either a shot made, missed, or foul.



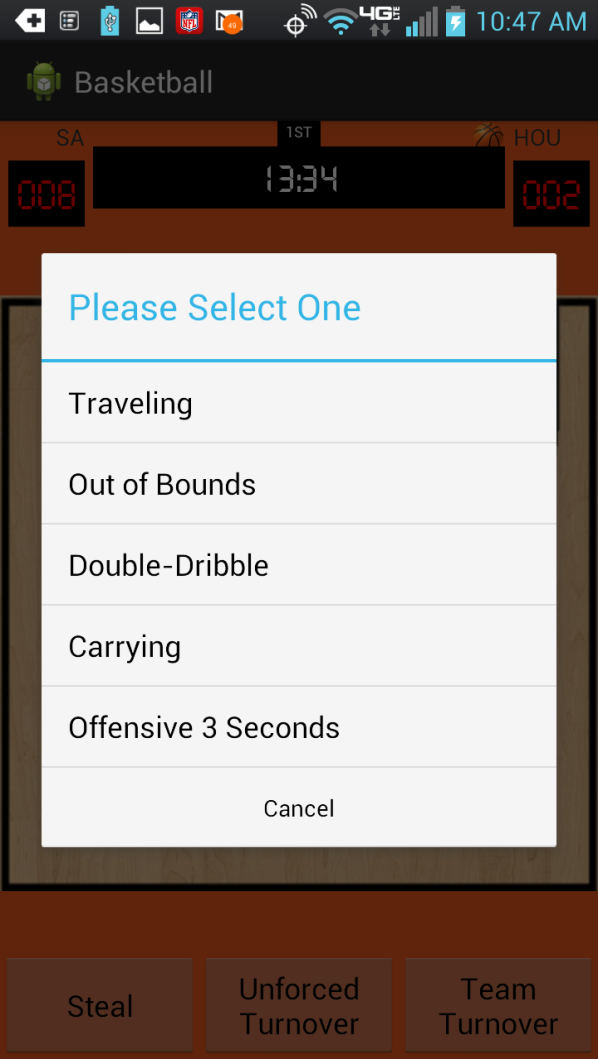
Once the shot result has been selected, a fly out menu will appear with the names of the players currently on the court. Selecting one will assign the shot to that player. If the shot was made, it will then prompt for an assist. If it was missed, it will prompt for a rebound (see below).



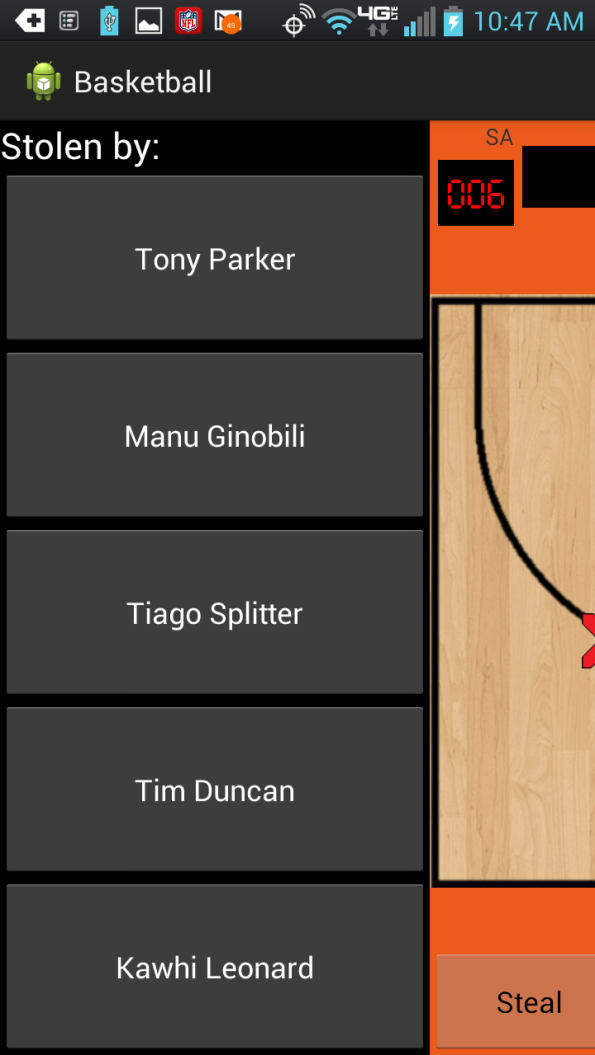
A rebound allows the user to select a player to get a defensive rebound as well as the option to switch to the other team to assign an offensive rebound.



The court will automatically populate with the current team’s shot chart. Green circles = made shots, red X’s = missed shots. This screen also shows the buttons after a shooting foul is called, allowing the scorer to track free throw attempts.



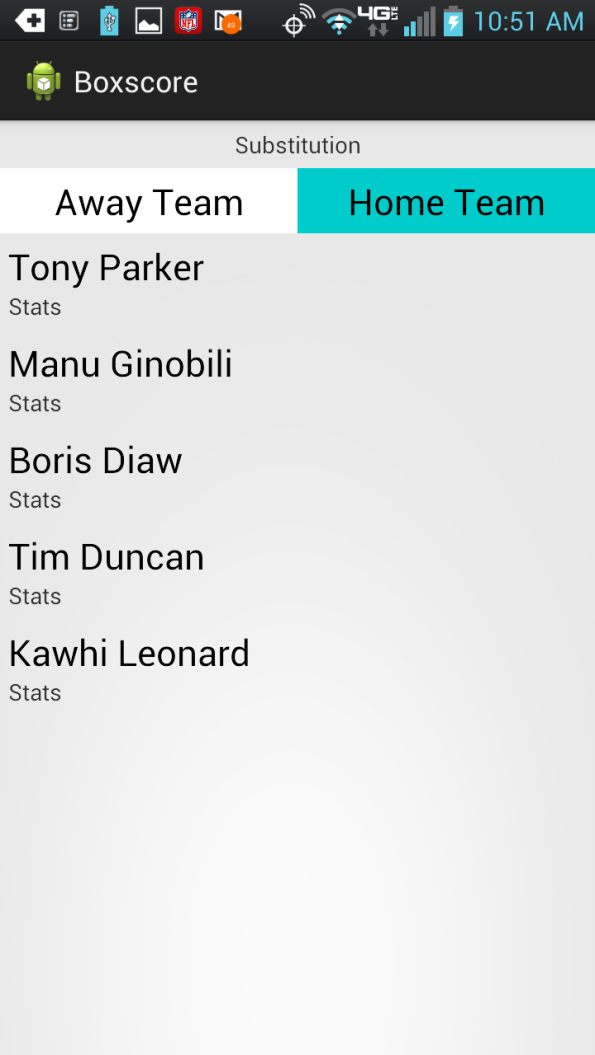
A list of unforced turnovers pops up when the “unforced turnover” button is selected, prompting the user to select which type of turnover as well as which player committed it.



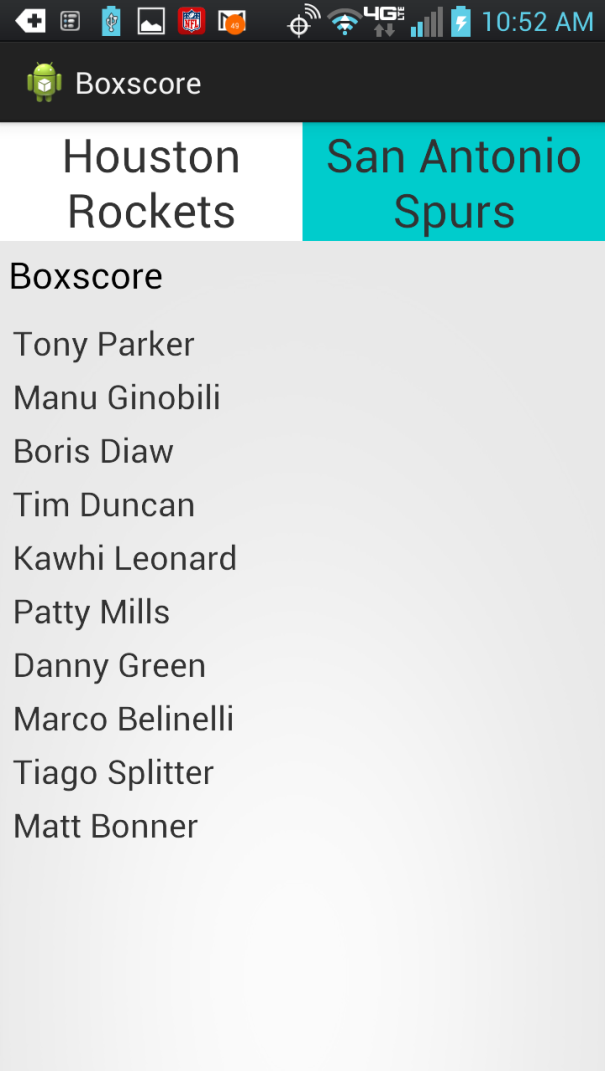
When selecting the “Steal” option, the user is prompted to choose who stole the ball as well as who the ball was stolen from.



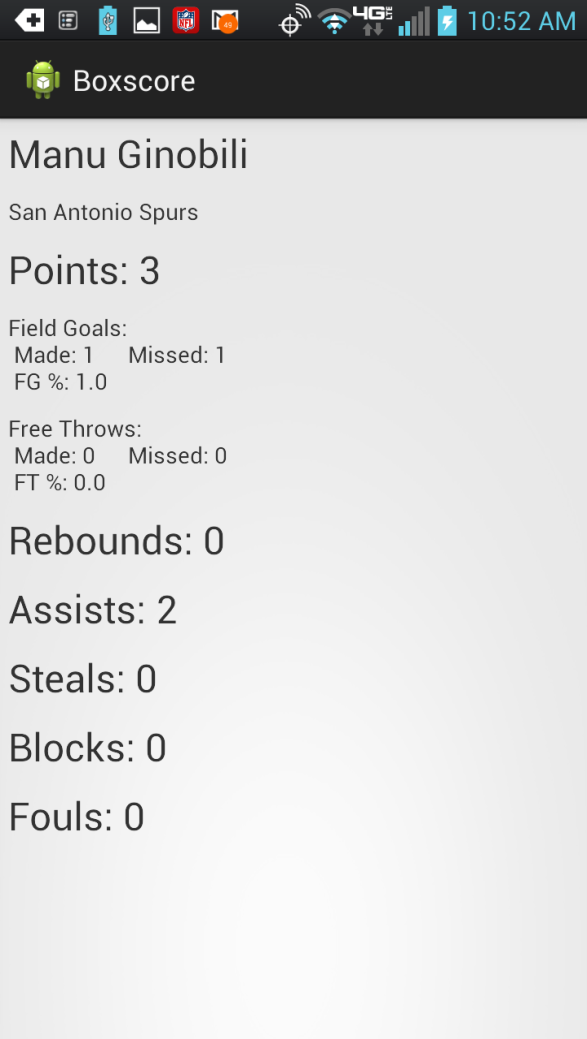
By clicking the menu button on your smartphone, this menu is displayed, allowing the user to select one of many options (see screenshots below for more details).



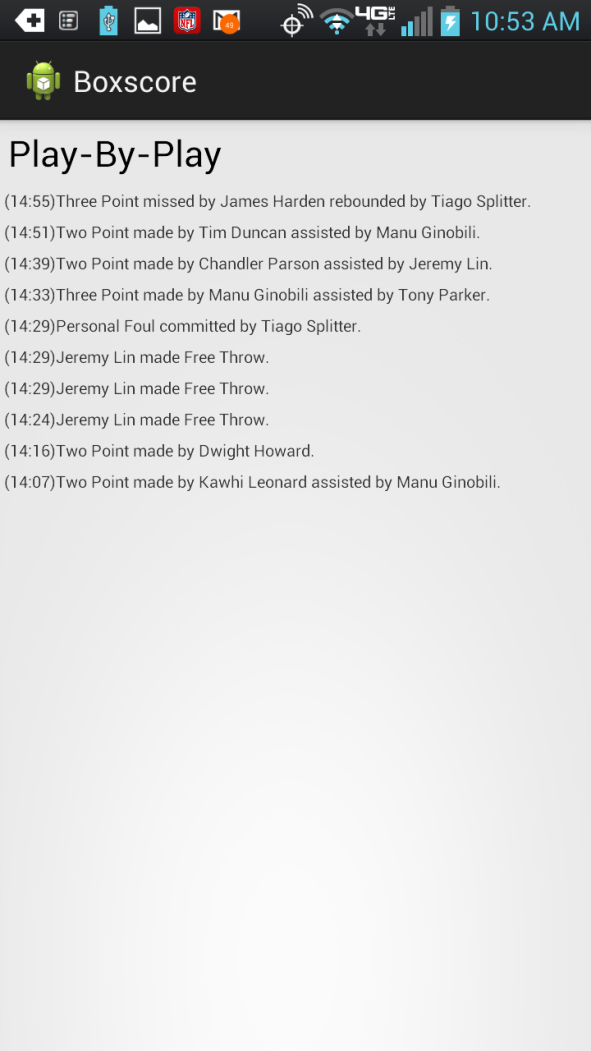
The substitution screen allows you to toggle between the two teams. Selecting one player leads to a screen where you select a bench player to swap them out for.



The box score also lets you toggle between both teams. By clicking on a player’s name, a screen containing their stats will be displayed (see below).



Here is an example of a player’s box score early on in a game. It displays the basic stats, but will be expanded on in later implementation.



The play-by-play screen shows a scrollable list of the plays that have occurred so far in the game as well as the time at which they happened.

5.6 Other Requirements

It is important to make the application visually appealing to the potential. At first glance, the style (or the looks) of a software may be more important than its substance. Keeping this thought in mind, the developers must balance the usability of this application and its visual grandeur. Although style gets the user to user this software, substance is what causes the user to continually use this product.

1. **Appendix**

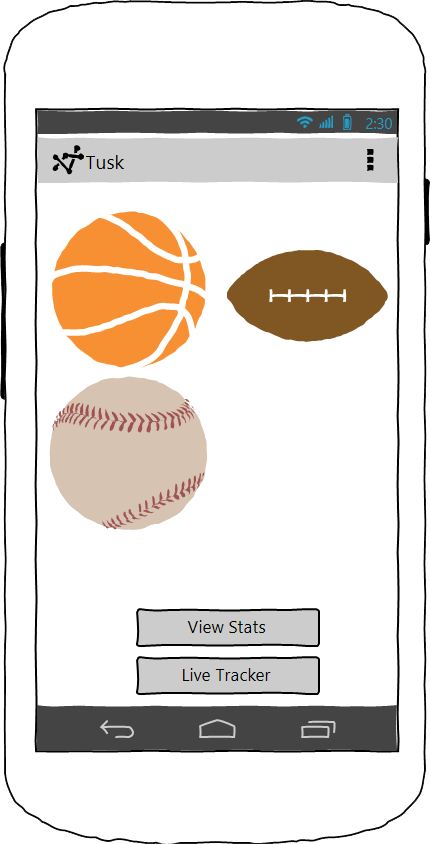
Appendix A: Terminology

Appendix B: User Interface Mockups

Appendix C: Statistics

**Appendix B: User Interface Mockups:**

1. **Home Screen**



The home screen will have an icon for each of the four sports we are creating a scorekeeper for: basketball, football, baseball, and soccer. From here, you can go to three different types of screens:

* By clicking one of the ball icons, you will be taken to the team selection screen. This is the beginning process of keeping track of a game. This is shown in part b.
* The “View Stats” button allows a user to view statistics from a previous game that they have access rights to see. These stats vary from sport to sport, but generally will consist of a box score, play-by-play, and some sort of interactive field of play.
* The “Live Tracker” button will allow the user to follow the progress of a live game that is currently being tracked by another user.

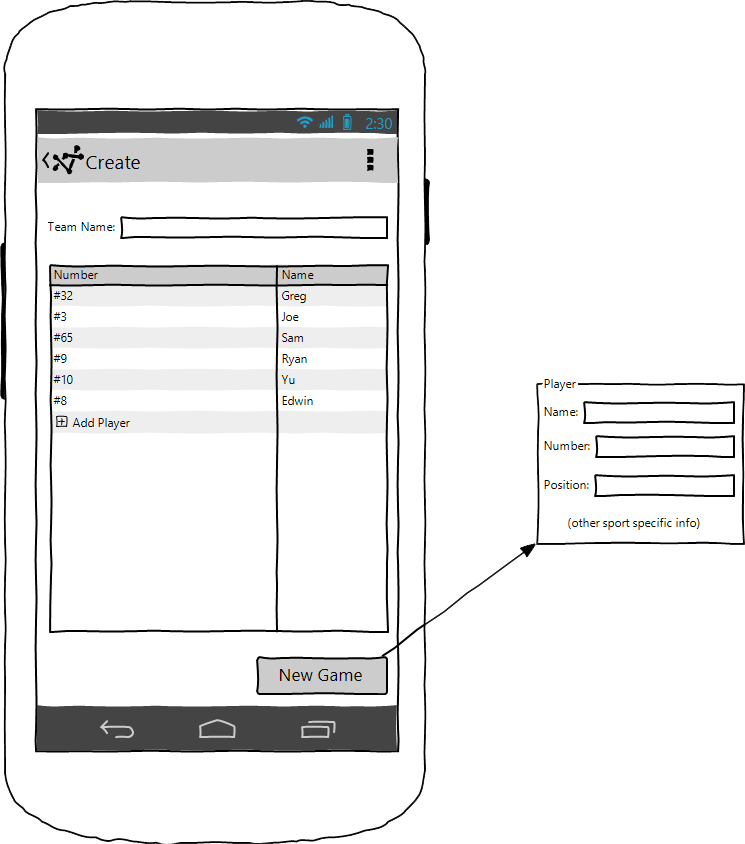
1. **Team Selection**



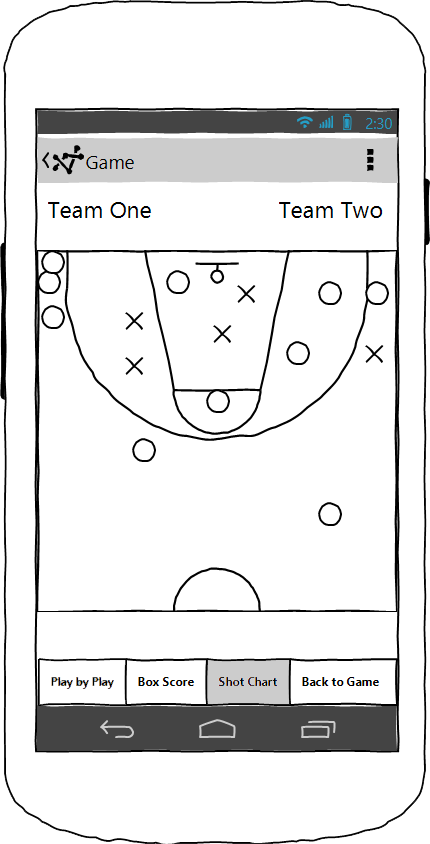
The team selection page will be the page that is displayed after the user selects one of the ball icons from the main screen. This page is customized based on which sport was selected. Not only will the correct sport name be at the top of the screen, but it will also display previously-created teams for this sport. For example, if the user creates a team called the Patriots for football, the Patriots will be displayed when the user selects the football icon, but not when they select one of the other three.

The user will also be able to create a new team from this screen. The player creation screen will be displayed (part c), allowing a user to enter the name, number, and position of the players on their new team.

Once both teams have been entered and selected, the “New Game” button will begin a new game for scorekeeping.

1. **Player Creation**

The player creation screen allows the user to input information on the players that are on their team. After clicking “Add Player”, the user will be prompted to enter the player’s information, such as their name, number, position, and other sport-specific details (i.e. dominant throwing arm/batting style for baseball). This information will be stored with that team name, allowing the user to immediately access the players that they have already entered.

1. **In-game Statistics Display – Basketball example**

The statistics display for each sport will have a multitude of screens that will graphically and numerically show the user the layout of the game. In this particular example of what the basketball users will see, there are tabs on the bottom that will shift from one view to another. The far left is the ‘Play by Play’, which gives a list of all of the plays from the game. The second tab is the ‘Box Score’. In this tab, there will be a list of all of the players that, when clicked on, will give an in depth look at their statistics, also in the form of a list. Finally, the ‘Shot Chart’ shows the spots on the floor where the team made or missed their shots. This can be toggled between the two teams, allowing you to see their individual shot charts. This tab will be sport-specific since each sport has its own version of a playing field that will need to be displayed.

1. **Post-game Statistics Display - Filtering**



The idea behind the post-game statistics display is to allow the user to filter the database of statistics by team or individual players as well as by the dates of the games played. If the user selects to view an individual player’s statistics, they will get a game log during the time period specified, as shown in the above table. Meanwhile, if they choose to view an entire team’s statistics, they will get the averages of each of those players during the timespan, as the second table indicates.

**Appendix C: Statistics:**

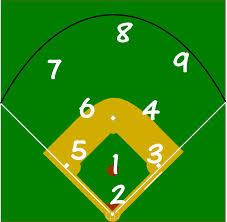
**1. Basketball Stats:**

* **FG** (field goals): in the form of FGM-FGA (ex. 4-6 means the player made 4 of 6 shots). Includes ALL 2 and 3 point shot attempts, but does NOT include free throws.
  + FGM = field goals made, FGA = field goals attempted
  + FG%
* **3FG** (3 point field goals): same as above, except only for 3 point shot attempts
  + 3FGM = 3 point field goals made, 3FGA = 3 point field goals attempted
  + 3FG%
* **FT** (free throws): same form as above, except only for free throws. Worth 1 point for each make.
  + FTM = Free throws made, FTA = free throws attempted
  + FT%
* **PTS** (points): Total number of points scored.
  + PTS = 1\*(FTM) + 2\*(2 point shots made) + 3\*(3FGM)
  + Using the stats that are being kept track of above, the alternative way to write it is:
  + PTS = 1\*(FTM) + 2\*(FGM-3FGM) + 3\*(3FGM)
* **OREB** (offensive rebound): When a player gets a rebound on the offensive end
* **DREB** (defensive rebound): When a player gets a rebound on the defensive end)
* **REB** (total rebounds) = OREB + DREB
* **AST** (assists): Awarded when a player made a pass that directly led to a made basket by one of their teammates.
* **STL** (steal): a player steals the ball from their opponent
* **BLK** (block): number of times a player blocks an opponent’s shot
* **TO** (turnover): number of turnovers
* **PF** (personal fouls): Number of fouls committed by a player
* **A/TO** (assist to turnover ratio) = AST/TO
* **+/-** (???): keeps track of point differential when a player is in the game. This may be tough to do because users may not keep track of substitutions correctly, leading to inaccurate stats.
* **EFF** (Efficiency from NBA.com): = ((Points + Rebounds + Assists + Steals + Blocks) - ((Field Goals Att. - Field Goals Made) + (Free Throws Att. - Free Throws Made) + Turnovers))
* **PER** (Player Efficiency Rating): This is extremely complex and requires minutes played to be kept track of, which we may or may not be implementing. The full formula is at <http://www.basketball-reference.com/about/per.html>.
* **AFG%** (adjusted field goal percentage): adjusts field goal percentage based on the fact that 3 point shots are worth more than 2 point shots.
  + Formula = [FGM + (3FGM/2)]/FGA or ](PTS – FTM)/FGA]/2

**2. Baseball Stats:**

Each at-bat will need the sequence of pitches (ball/strike/ball in play) to be recorded individually. However, it is usually the final result that affects the hitter/pitchers overall statistics.

1. **Hitting:**
   * **AB** (at bats): how many times a player goes to the plate (not including walks, sacrifice flies, or HBP [hit by pitch])
   * **HBP** (hit by pitch): whenever a player gets hit by a pitch
   * **SAC** (sacrifice fly/bunt): does not count as an AB for a hitter. Simply used to designate that they have sacrificed themselves to advance a teammate.
   * **H** (hit): when the batter gets a hit
     + **1B** = single
     + **2B** = double
     + **3B** = triple
     + **HR** = home run
   * **R** (run): when the player scores a run
   * **RBI** (runs batted in): When a player gets a hit, walk, or sacrifice that scores a teammate
   * **BB** (base on balls OR walk): when a batter gets walked (draws 4 balls)
   * **SO** (strike out): when a player strikes out (gets 3 strikes)
   * **#P** (number of pitches): the total number of pitches that a hitter faced.
   * **AVG** (batting average): H/AB
   * **OBP** (on base percentage): % of at bats where a hitter gets on base. Does not include errors or fielder choices.
     + **OBP** = (H + BB + HBP) / (AB + BB + HBP + SAC)
   * **SLG** (slugging percentage): total number of bases per at bat
     + **SLG =** (1B + 2B\*2 + 3B\*3 + HR\*4) / AB
   * **OPS** (on base plus slugging): = OBP + SLG
   * Other hitting scenarios that aren’t reflected in the box score:
     + FC (fielder’s choice): when a hitter gets on base because the fielder chose to get a different baserunner out instead of the hitter. This counts as an at-bat but not a hit for the batter (as a result hurting their batting average).
     + E (error): the batter gets on base due to an error by a fielder. This also counts as an at bat but not a hit for the batter
2. **Baserunning**:
   * **SB** (stolen base)
   * **CS** (caught stealing): when a baserunner attempts to steal a base but is tagged out
   * **SB%** (stolen base percentage) = SB / (SB+CS)
   * **PO** (pick off): when a baserunner is thrown out after taking too big of a lead.
3. **Pitching:**
   * **IP** (innings pitched): number of innings pitched by a pitcher. The values that can be inserted into this are and number with either a 0, 1, or 2 in the tenths place. For example, a pitcher can pitch 3.2 innings, which means they successfully made it through 3 complete innings as well as got 2 additional outs (making for a total of 11 outs).
   * **H** (hits): number of hits allowed
   * **R** (runs): number of runs allowed
   * **ER** (earned runs): number of earned runs allowed. A run is unearned when an error allows a run to score that wouldn’t have been scored if the error hadn’t occurred.
   * **BB** (walks): number of walks allowed
   * **SO** (strike outs): number of batters the pitcher strikes out
   * **HR** (home runs): number of home runs allowed
   * **PC** (pitch count): number of pitches thrown
   * **ST** (strikes): number of strikes thrown
   * **ERA** (earned run average): number of earned runs allowed per 9 innings
     + **ERA** = ER/9
   * **WHIP** (Walks and hits per innings pitched) = (BB + H) / IP
     + One thing to note is that IP in this formula is NOT in the form mentioned above. Instead of a pitcher pitching 3.1 innings, this formula requires it to be 3.33 (3 and 1/3) innings pitched.
4. **Fielding:**
   * **E** (error): an error is charged to a fielder if they fail to make a routine play.
   * When each ball is hit in play, a sequence of numbers indicates where the ball went in order to record the out. See the picture below to know which position is assigned which number:



For example, if a ground ball is hit to the shortstop, who throws a runner out at second base to the second baseman who then proceeds to turn a double play to the first baseman, then the play would be ruled a 6-4-3 double play.

**3. Football Stats:**

1. Passing:
   * **Cmp** (completions): number of completed passes
   * **Att** (attempts): number of attempted passes
   * **Yds** (yards): number of passing yards
   * **TD** (touchdowns): number of touchdowns thrown
   * **Int** (Interceptions): number of interceptions thrown
   * **2PC** (2 point conversion): number of successful 2 point conversions thrown
   * **QB rating:**
2. Rushing:
   * **Att** (attempts): number of rushing attempts
   * **Yds** (yards): number of rushing yards
   * **TD** (touchdowns): number of rushing touchdowns
   * **2PC** (2 point conversion): number of successful 2 point conversions run for
3. Receiving:
   * **Tgt** (targets): number of times the QB threw a pass in the direction of a receiver
     + We may or may not implement this depending on if we require the user to input the intended receiver on incompletions. For simplicity we may want them to just say whether it was complete or not.
   * **Rec** (receptions): number of receptions
   * **Yds** (yards): number of receiving yards
   * **TD** (touchdowns): number of receiving touchdowns
   * **2PC** (2 point conversion): number of successful 2 point conversions caught
4. Ballcarrier:
   * **Fmb** (fumbles): number of fumbles by a player
   * **FL** (fumbles lost): number of times a player’s fumble is recovered by a member of the opposing team
5. Defense (this isn’t 100% necessary since they can be tough to keep track of at times):
   * **T** (tackles): number of tackles
   * **TFL** (tackle for loss: number of times a defensive player records a tackle behind the line of scrimmage. A sack counts as a TFL
   * **Sk** (sack): tackling the QB behind the line of scrimmage. A sack can be shared, allowing a player to have a half of a point (ex. They can have 2.5 sacks instead of being forced to have an integer value for sacks).
   * **FF** (forced fumble): awarded to a player who forces an offensive player to fumble
   * **FR** (fumble recovery): when a player recovers a fumble
   * **Int** (interception): awarded to a defensive player who catches a pass from the QB.
   * **DFTD** (defensive touchdown): when a player on the defensive side of the ball returns either a fumble or an interception for a touchdown.
6. Kicking:
   * **XPM** (extra points made)
   * **XPA** (extra points attempted)
   * **FGM** (field goals made
   * **FGA** (field goals attempted)
7. Punting:
   * **Pnt** (number of punts)
   * **Yds** (yards): total yards punted
   * **Y/P** (yards per punt) = Yds/Pnt
   * **In20** (inside 20): number of punts downed within the 20 yard line
8. Special Teams:
   * Kick Returns:
     + **Rt** (returns): number of returns
     + **Yds** (yards): total number of return yards
     + **Y/Rt** (yards per return) = Yds/Rt
     + **TD** (touchdowns): return touchdown
   * Punt Returns:
     + **Rt** (returns): number of returns
     + **Yds** (yards): total number of return yards
     + **Y/Rt** (yards per return) = Yds/Rt
     + **TD** (touchdowns): return touchdown
9. Note: Almost all of these can have a “Long” stat that indicates the longest value for that stat recorded by that individual player

**4. Soccer Stats:**

* **Shots**: Number of shots attempted
* **Shots on Goal (SOG)**: Number of shots attempted that would have or did enter the goal
* **Goals (G)**: Number of goals scored
* **Assist** **(A)**: A pass that directly leads to the scoring of a goal
* **Shot Accuracy (%)** = Shots on Goal / Shots \* 100
* **Fouls**: Awarded when the referee blows the whistle for a player performing an illegal action
* **Corner Kicks**: Number of times a team attempts a corner kick
  + **Successful Corner Kicks**: Number of goals scored directly off a corner kick
  + **Corner Kick %** = (Successful Corner Kicks)/(Corner Kicks) \* 100
* **Penalty Shots**: Number of times a team is awarded a penalty shot
  + **Successful Penalty Shots**: Number of goals scored on a Penalty Shot
  + **Penalty Shot % =** (Successful Penalty Shots) / (Penalty Shots) \* 100
* **Offside**: Number of times a team is called offside
* **Yellow Cards**: Total number of yellow cards earned by members of the team
* **Red Cards**: Total number of red cards earned by members of the team
* **Saves**: Number of times a goalie stops a shot that would have otherwise been a goal.