Live Scoring Cricket: A mobile application for real time cricket scorekeeping

Patrick Mallam
B.A.(Mod.) Business and Computing
Final Year Project April 2013
Supervisor: Mary Sharp

School of Computer Science and Statistics
O’Reilly Institute, Trinity College, Dublin 2, Ireland
DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university

_________________________________  __________________________
Name                                      Date
Table of Contents

Abstract .......................................................................................................................... 4
Introduction ..................................................................................................................... 5
    TEAMtalk Media: ........................................................................................................ 5
    Mobile in Africa: ........................................................................................................ 6
    The Laws of Cricket: ................................................................................................. 7
    Requirements: ........................................................................................................... 11
Database Structure ........................................................................................................ 13
    Primary Keys: ........................................................................................................... 14
    Table Descriptions: ................................................................................................. 15
The Android Application ............................................................................................... 17
    JSONParser.java ....................................................................................................... 18
    Android Activities and Intents .................................................................................. 18
    Async Task executions: ......................................................................................... 23
Web Interface .................................................................................................................. 27
    Matches ...................................................................................................................... 27
    Players ....................................................................................................................... 28
    Teams ......................................................................................................................... 28
Future Work .................................................................................................................... 29
Conclusion ...................................................................................................................... 32
Acknowledgements: ..................................................................................................... 32
References: ................................................................................................................... 33
Appendix: ...................................................................................................................... 34
Abstract

The following project focuses on an issue raised in discussion with a representative of “TEAMtalk Media”, a company specializing in the reporting of sports news and statistics. A portion of their live sports reporting operation is carried out in the Republic of South Africa, where cricket is an immensely popular sport.

Reporting for local and regional cricket matches in newer African markets has proven difficult for a number of reasons. These include the remote location of matches, the fact that some matches are not televised, and subsequently, the difficulty associated with hiring a reporter in a different country and trusting them with equipment to effectively relay match information for publication. With that in mind it became evident there was a prerequisite for an application that could be developed for a mobile device and used as a digital scorecard for remote match reporting. This project focuses on the steps taken to develop such an application for the Android platform of mobile devices, as well as a structured MySQL database and web interface for displaying the data obtained from scoring these matches.
Introduction

TEAMtalk Media:

The basis for this project was formulated after a discussion with a representative of TEAMtalk Media, a sports content publishing company, always keen to expand to new markets within the sports industry. Some of the brands included under the TEAMtalk umbrella include: “SkySports.com”, “Planet-F1.com”, “PlanetRugby.com”, “Cricket365.com”, “Football411.com”, and “LiveScore.com” (See full table in Appendix, Figure 1).

“TEAMtalk media specializes in the creation and commercialisation of sports news, content and data across multiple platforms. We distribute a broad range of ready sports content directly to sports fans and business partners, including football clubs, sports federations, broadcasters, bookmakers, publishers and mobile network operators”

(TEAMtalkMedia.com, April 2013)

From a computer science perspective, the most interesting part of the TEAMtalk business model is the data collection service they make use of in order to deliver usable and reliable information to broadcasters, publishers and bookmakers. One of their primary aims is to enable informed decision making by providing deep and credible data, in as close to real time as possible.

TEAMtalk Media covers a comprehensive range of sporting events spanning the global calendar, with a focus on the major UK and African football leagues. With an uptake in mobile applications and a surge of new users being able to access information from their mobile devices there has been an uptake in the demand for real-time match coverage across a diverse range of sports. TEAMtalk’s coverage of global and regional sporting events produces an average of 1,000 points of match data per event. TEAMtalk Media also offers in-house bookmaking and betting expertise and provides live content, data and technology to various betting houses in South Africa and the UK.
TEAMtalk Media currently has two main methodologies for taking down data for the matches they cover. The first being the employment of several people tasked with watching matches that are televised, and transcribing the data they have gathered directly into the systems used by the company in order to store the data. The second method related to non-televised matches whereby a reporter would have to be sent to the match in question along with a laptop loaded up with the software required, as well as a data connection in order to relay the information as it occurred.

The specific problem facing TEAMtalk Media which prompted this project was that when it came to reporting live scores of a recent rights acquisition for cricket from certain remote locations outside of South Africa but within the African continent there became an issue of trust with regards to hired sports reporters and the equipment they were given in order to report on the match. It seemed therefore that an adequate solution to this problem was to provide a mobile application that could be run on a device owned by the reporter, which would enable the reporter to create a match scorecard from the grounds of the live game, without the need for any special equipment.

**Mobile in Africa:**

The African continent has one of the highest mobile penetration rates in the world, and on top of that, most of the people on the continent interact with the web solely through their mobile devices (State of the Mobile Web, March 2012). Mobile web usage is so high in Africa for a number of reasons, chiefly among them being the fact that it is the cheapest way to gain access to the internet. Secondly, Africa’s wired networks are nowhere near as well developed as the rest of the world, and in many cases it would not be cost effective for line operators to invest in heavily upgrading their networks just to cater for high-speed internet access. As a result, the development of mobile data technology is extremely high across the African continent.

It therefore seemed a logical step to design a mobile application that allows reporters to simply, and accurately relay information from these matches via their mobile device. Seeing as penetration of mobile data usage is relatively high in Africa (Appendix, Figure 2) it would also be highly possible to achieve this goal. With respect to data transfer, as the information being transferred is largely text based, there would also be a
low impact on data usage, meaning that if a game were covered in a region where high
speed data access wasn’t possible, it would still be a feasible solution. The main problem
facing the project was how to adequately construct a database and application to score
what is the very complex sport of cricket.

The Laws of Cricket:

Unlike some other sports, cricket is governed by a series of Codes of Law rather
than a set of rules. The Laws are exclusively governed by the Marylebone Cricket Club
(MCC) of London, and have been since it’s foundation in 1787. In total there are forty-
two complex and in-depth laws covering the players and officials, the equipment and
layout of the pitch, structure of the game, scoring and winning, mechanics of dismissal,
methods of dismissal, laws for the fielding side, and fair and unfair play. Although not all
of the Laws apply to the application in question, it is useful to briefly outline the aim of
the game, as well as highlight the Laws which have greatest impact on the design of the
application for scoring cricket matches.

Players & Officials:

Law 1: Cricket is played between two teams consisting of eleven players per side,
including a designated captain.

Law 3: Each match is governed by at least two umpires whose job it is to ensure the
game abides by the Laws set out in the MCC document. The umpires are the decision
makers on the pitch.

Equipment & Pitch:

Law 5: The ball must be a standard circumference of between 22.4cm and 22.9cm.

Law 6: The bat must be of a standard size, no more than 97cm in length, and no more
than 10.8cm wide, made of wood – most commonly Willow.
Law 7: The pitch is a standard size of 22m long and 3m wide. The pitch is different from the ground as you can see in Diagram 1 below; the pitch sits within the grounds, usually somewhere near the centre and is where batting and bowling take place during the match.

![Diagram 1: The cricket pitch](image)

Law 8: The wicket consists of three wooden stumps 71cm in height placed equal distance apart on the batting crease. They are positioned so as to be 23cm wide, and resting between the two gaps created by the stumps are the bails. The primary aim of the batsman during any game of cricket is to protect his wicket. Should the bails be knocked from the stumps, the batsman is deemed dismissed – however this is only one of nine ways to dismiss a batsman. See Diagram 2 below.

![Diagram 2: The Wicket](image)
Scoring:

**Law 12:** An Innings refers to a period of time where one team is batting and the other is bowling. The captain of the team who wins the coin toss (known as “The Toss”) prior to commencement of the game decides whether to bat or bowl first. An innings is closed once all the batsmen are dismissed, no further batsmen are fit to play, the innings is declared forfeited by the batting team’s captain, or the over limit has been reached.

**Law 18:** Runs are scored when the two batsmen run to each other’s end of the pitch. More than one run can be scored from one ball bowled.

**Law 19:** A boundary is marked around the edge of the field of play. If a ball crosses this boundary without making contact with the ground six runs are given to the batting team, and the batsmen can return to their position prior to the bowling of the ball. If the ball crosses the boundary after making contact with the ground, four runs are awarded to the batting team and the batsmen return to their positions prior to the bowling of the ball.

**Law 21:** The side with the most runs at the end of the match is the winner. In the case of a tie, the result is a draw - although in some game variations there is an extra over per side. If a team is unable to complete an innings within the specified time limit, the match is considered drawn.

**Law 22:** An over consists of six balls bowled to the batsman, excluding wides and no balls. Common game types such as Twenty/Twenty and One Day Internationals will be limited to a number of overs per innings.

**Law 24:** A ball can be deemed “no ball” for several reasons: if the bowler bowls from the wrong place; or if he straightens his elbow during the delivery; or if the bowling is dangerous; or if the ball bounces more than twice or rolls along the ground before reaching the batsman; or if the fielders are standing in illegal places. A no ball adds one run to the batting team’s score, in addition to any other runs which are scored off it, and the batsman can’t be dismissed off a no ball except by being run out, or by handling the ball, hitting the ball twice, or obstructing the field. The score from a no ball is marked as an extra on the scorecard, and an “NB” on the bowler’s statistics.

**Law 25:** A ball can be deemed “wide” by the umpire if in their opinion the batsman did not have a fair opportunity to score off the ball – in most cases this applies to stray
bowling. Similar to a no ball a wide adds one run to the batting team’s score, in addition to any other runs which are scored off it. Also, the batsman can't be dismissed off a wide except by being run out or stumped, or by handling the ball, hitting his wicket, or obstructing the field.

**Law 26:** Byes are scored when the batsman fails to make contact with the ball with the bat, however the wicket keeper is unable to stop the ball, and as a result the batsman can score runs off the ball. This is usually marked down against the wicket keeper. Byes are credited to the team total, but not to the batsman’s total.

**Methods of Dismissal**

**Law 30:** Bowled: A batsman is out if his wicket is put down by a ball delivered by the bowler. It is irrelevant whether the ball has touched the bat, glove, or any part of the batsman before going on to put down the wicket, though it may not touch another player or an umpire before doing so.

**Law 31:** Timed Out: An incoming batsman must be ready to face a ball (or be at the crease with his partner ready to face a ball) within 3 minutes of the outgoing batsman being dismissed, otherwise the incoming batsman will be out. This rarely happens.

**Law 32:** Caught: If the ball hits the bat or the hand holding the bat and is then caught by a fielder from the opposing side before the ball makes contact with the ground, the batsman is considered out.

**Law 33:** Handled the ball: If a batsman willfully handles the ball with a hand that is not touching the bat, the batsman is deemed out.

**Law 34:** Hit the ball twice: If the batsman hits the ball twice, other than for the purpose of defending his wicket (sometimes the ball will bounce back to the bat after initial contact) or without consent from the opposition, the batsman is dismissed.

**Law 35:** Hit wicket: If the batsman makes any contact with the wicket causing the bails to be disturbed while the ball is in play, the batsman is considered out.

**Law 36:** Leg before wicket (LBW): If the ball hits the batsman without first hitting the bat, but would have hit the wicket if the batsman was not there, and the ball does not pitch on the leg side of the wicket, the batsman will be out. However, if the ball strikes
the batsman outside the line of the off-stump, and the batsman was attempting to play a
stroke, he is not out.

**Law 37:** Obstructing the field: If a batsman willfully obstructs the opposition by word or
action, he is out.

**Law 38:** Run out: A batsman is out if at any time while the ball is in play no part of his
bat or person is grounded behind the popping crease and his wicket is fairly put down by
the opposing side.

**Law 39:** Stumped: A batsman is out when the wicket-keeper (see Law 40) puts down the
wicket, while the batsman is out of his crease and not attempting a run.

**Requirements:**

The aim of this project is to design a simple android based interface that a reporter
with a basic understanding of the rules of cricket could easily use to score a match of
 cricket. The data could then be passed on via the android device’s internet connection to a
MySQL database and used to display the match statistics in as close to real time as
possible via a web interface. The web interface would output similar scorecards to those
used commonly by various sports reporting agencies, such as the one in Diagram 3. As
well as this, the website would serve as a portal for browsing statistics of various players,
teams and matches stored in the database.

In order for the application to be successful the android application would have to
abide by many of the Laws listed above, mainly in the scoring and dismissal columns.
Some keys to success are listed below:

- **Game Setup Definition:** First of all selecting the two teams competing
  against each other.
- **Ground Information:** Data collected on the ground and umpires to be
  adjudicating the game
- **Game Type Definition:** Defining the type of rules to be applied, limited
  overs (i.e. 20 overs per innings, 50 overs per innings) or a five day test match.
  Also defining for record keeping the subtype of the match
- **Scoring**: Handling the various run scoring and dismissal Laws in cricket in the program, so the user doesn’t have to do so manually
- **Normalised Database**: Simple Database structure to accommodate information received in MySQL
- **Structured Statistics Display**: Reliable structured display of database information in as many web browsers as possible.

![New Zealand XI Innings 171 for 7 (20.0 overs)](image)

(Diagram 3: BBC Sports, 2013)

The following sections layout the steps taken to achieve the requirements set out above. The sections are broken down into: The design and structure of the MySQL database, the design of the android app, and finally the design of the web interface for viewing the statistics generated by the android application.
Database Structure

It became clear early on in development that the most important part of the whole application was the structure of the MySQL database. This was because the application relied on this data to be structured correctly in order for the android app to feed the data into the database and for the web interface to display the data in a meaningful way.

There were three distinct phases in the development of the structure of the database. The first (See Diagram 4) resulted from quick look at the most important pieces of information displayed on a cricket scorecard, namely the match information, and the players scoring information. This resulted in a basic understanding of the need for distinct batting and bowling statistics for each player, rather than a single scoring record for each player within each match.

(Diagram 4: First Draft of MySQL Schema)

The second phase incorporated the changes to the scoring tables as well as including tables for the Players, the Teams, the Umpires and the Grounds. The second phase improved the structure of the Match table to include type, subtype, ground_id, result, win_margin, win_type, win_with_balls_left, and the umpire_id’s. It was later decided that some of this information was too detailed, and beyond the scope of the simple input available to the android application. It would involve too much typing in the android application, something to steer clear of when trying to reduce the possibility of...
human error. The second phase was a big step forward to a more normalised and relational database structure.

The issue arose of how to ensure that a player’s participation in a single match reflected on both that match’s statistics, as well as a general historical record of that player’s performance. As a result the second iteration of the database schema included a pairing of both player_id and match_id in the Batting and Bowling tables to ensure that if a player had multiple records in the Batting and Bowling tables they could be uniquely identified by the combination of the primary keys player_id & match_id in order to assess performance historically vs. performance in a single match.

The final phase took into mind the issues with the second phase and settled on the structure outlined in the Entity Relationship Diagram described in Diagram 5.

![Entity Relationship Diagram (ERD) Final DB](Diagram 5: Final ERD for MySQL Database)

**Primary Keys:**
• **umpire_id**: Refers to the umpire in the Umpires table; foreign key in the Matches table - auto incremental when a new umpire is inserted into the database.

• **ground_id**: Refers to the ground in the Grounds table; foreign key in the Matches table – auto incremental when a new ground is inserted into the database.

• **match_id**: Refers to the unique match listed in the Matches table; foreign key in the Bowling and Batting tables – auto incremental when a new match is inserted into the database.

• **team_id**: Refers to the unique team listed in the Teams table; foreign key in the Matches, Batting and Bowling tables – auto incremental when a new team is inserted into the database.

• **player_id**: Refers to the unique player listed in the Players table; foreign key in the Batting and Bowling tables – auto incremental when a new player is inserted into the database.

**Table Descriptions:**

- **Umpires**: Stores the names of the various umpires.
- **Grounds**: Stores the names, cities, countries of the various grounds.
- **Teams**: Stores the name, and an image link for displaying a team logo on the web interface.
- **Players**: Stores a shortened name attribute for the scorecard, a fullname attribute for the player profile, as well as an image link for the players profile on the web interface.
- **Matches**: Stores various match attributes including information relating to the home and away teams, the ground id, runs scored and wickets for each team, the type and sub type of each match and the umpires adjudicating. There is also an attribute for storing the team who won the coin toss prior to winning the match.
- **Batting**: As well as including the player_id and match_id this table includes all the information relating to a players scoring while at bat. arrival_score,
arrival_wickets, arrival_overs record the current score, wickets, and overs when the batsman enters play. Similarly departure_score, departure_wickets, and departure_overs record the relevant score, wickets, and over when/if the batsman is dismissed. Scoring is broken into runs, fours and sixes. Dismissal is handled by dismissal_type and relate back to the 9 methods of dismissal described between Laws 30-39. In the event of a dismissal the dismissal_bowler_id is updated to include the bowler’s id, and in the event the method of dismissal is a catch, the caught_by_id is updated to include the player_id of the catcher.

- **Bowling:** The bowling table is similar to the batting table, however it only includes the details relevant to a bowlers scoring. In this case it counts the overs_bowled number of balls_bowled, number of maidens (which refers to a complete over bowled without conceding a run), number of runs_conceded, number of wickets_taken, and the number of no balls and wides bowled.

A complete MySQL schema for the database is displayed in Appendix Figure 5.

Having finalised the structure of the database and ensured that it was as close to the 3rd Normal Form as possible, development on the android application could begin. The following section describes the processes of design and implementation of the Live Score Cricket application.
The Android Application

The android application portion of this project aimed to reflect some of the desktop based cricket scoring applications available today (See Diagram 7), in a more condensed manner. Every effort was made to make the input of data as easy as possible for the user. From the start there was a clear vision as to what the android application should look and behave like (See Appendix Figure 3 and 4). Paper prototyping allowed easy experimentation with ideas. The current version of the application differs greatly from the original prototypes and although it is functional there are many small changes that could be made to improve functionality.

(Diagram 6: Structure of Android Application and Web Interface)

(Diagram 7: Cricket Statz Setup Screen 2012)
The following sections describe the structure of each of the classes in the android application as well as some of the aspects of a typical android application. This particular application used the android API level 17, and was optimised for use on devices from version 4.2 back to version 3.0 of the android operating system.

**JSONParser.java**

Java Script Object Notation is a standard used for transferring structured data over a network connection. In the case of the live score cricket application this standard was used to link the MySQL database with the android application itself. In order for our application to work the JSONParser class would have to lookup various URLs (the PHP files with the MySQL queries on them) and parse the data into usable bits of information. Likewise the JSONParser also allows the application to pass data to the server, and on to the database for insertion and updates of information.

**Android Activities and Intents**

Android activities act as single focused tasks that a user can interact with. In the case of the cricket scoring application there are seven total activities the user can interact with while scoring a cricket match. Each activity is linked to an XML document which dictates the manner in which the User Interface (UI) displays information.

Android intents are abstract descriptions of activities to be performed. An Intent’s “most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.” (Android Developers, 2013)
MainSplashActivity.java linked to splash_activity.xml

![Diagram](image.png)

This is the first screen the user sees when they open the Live Score Cricket app on their android device. It consists of two main buttons that are fairly self-explanatory. A user can choose to either create a new game to start scoring by tapping the “Start New Game” button, or they can choose to resume scoring a game that has already been setup by selecting the “Resume Game” button.

Each button, like all the buttons used in this application, has what is known as an “onClickListener” associated with an action to be performed. In the case of the “Start New Game” button, the onClickListener invokes a new Intent to open the TeamSetup.java activity, and display the associated team_setup.xml UI. This is the Java/Android equivalent of navigating between web pages. Similarly the “Resume Game” button invokes a new Intent to open GamesList.java and the list_games.xml, and all_games.xml UIs.

There is no database communication going on within the MainSplashActivity.java activity. The “main screen test” button on the page is purely for debugging purposes and invokes the opening of the main scoring activity, MainActivity.java.
After selecting to start a new game, the user is prompted to select the teams competing against each other in this particular match. These teams are displayed on “Spinners” – the android development term for a drop down menu. It was decided that instead of allowing the user to enter the team names in manually using the device keyboard, the teams would be inserted into the database prior to the match. This would ensure that no errors would be caused during manual data insertion. This method also solved the issue of having a user manually enter in the names of upwards of at least 22 players.

In the TeamSetup.java activity a concurrent network operation (known as an Async Task) called LoadAllTeams() is performed as the TeamSetup activity opens. This is to populate the spinners with the names of the teams currently stored in the MySQL database. LoadAllTeams() is executed by passing a JSON HTTP “GET” request to a PHP file resting on the server where the database is stored. The PHP file “list_teams.php” is run, querying the database for all the team names within the Teams table. The response is pushed into a PHP array, then encoded into a json_response. The JSON response is then pushed into a JSON object in the android application. The response is then looped through based on the length of the array, and the team name is added to an ArrayList which will populate both spinners with the names of the teams in the Teams table.

When the user has selected the teams competing against each other, and has selected “Submit” the Async Task CreateNewTeams() is invoked. Here the names of the
two selected teams are posted by passing a JSON HTTP “POST” request to the PHP file “add_teams.PHP”. The “add_teams.PHP” file then performs two queries finding the team_id’s of the home and away team, and then inserts a new record into the Matches table with the home_id and away_id as well as the current date. The new match record has now been successfully created. On successful completion of the Async Task CreateNewTeams() the intent to open GroundInfo.java is invoked.

**GroundInfo.java linked to ground_info.xml**

![GroundInfo](image)

**(Diagram 10: GroundInfo)**

After inputting the two teams competing the user is then prompted to give some information regarding the ground at which the match is being played. As with the team setup page, GroundInfo.java uses a JSON HTTP “POST” request to the “add_ground.PHP” file resting on the server.

Upon receiving the post parameters of name, city, country, ump1 and ump2, the following MySQL queries and inserts take place:

1. Inserts ground information into the Grounds table
2. Fetches the id for the newly created ground by selecting the maximum
   \textit{ground\_id} in the table
3. Inserts \textit{ump1} name into Umpires table
4. Fetches \textit{ump1} id by selecting the maximum id in the table
5. Inserts \textit{ump2} name into Umpires table
6. Fetches \textit{ump2} id by selecting the maximum id in the table
7. Fetches the \textit{match\_id} of the most recently created match in the Matches table
8. Updates the values of the \textit{ground\_id, umpireA\_id,} and \textit{umpireB\_id} in the
   Matches table for the most recently created match

\textit{FinalSetup.java linked to final\_setup.xml}

\textit{FinalSetup.java} allows the user to input the match type, and match subtype in exactly the same way as \textit{GroundInfro.java} allows the user to type in data. As with \textit{FinalSetup.java} a JSON HTTP “POST” request is submitted and calls upon the “add\_game\_type.PHP” file resting on the server.

Upon receiving the post parameters \textit{type}, and \textit{subtype} the following MySQL queries take place:

1. Fetches the \textit{match\_id} of the most recently created match in the Matches table
2. Updates the values of the \textit{type}, and \textit{subtype} in the Matches table for the most recently created match

The \textit{type} of match determines how many overs are to be played in the game, and is important to the main scoring phase of the application. At the moment the scoring is set up only to deal with twenty and fifty overs per innings.

\textit{GameList.java linked to all\_games.xml and list\_games.xml}

\textit{GameList.java} is the penultimate screen before the user can begin scoring a match. It uses a LoadAllMatches() Async Task to list out the match id’s onto the screen. Again this task makes use of a JSON HTTP “GET” request from the “list\_games.PHP” file, and the response must again be parsed in order to create the list. The user then clicks
on the relevant match to them – in the case of a newly created match this will be the match id of the highest value.

Upon clicking the relevant match the user invokes a new intent to start `MainActivity.java`. As well as this, the relevant `match_id` is attached to the intent as an extra piece of information. This extra piece of information is carried over to allow `MainActivity.java` to begin loading match information on the next screen for the relevant match. It is the first piece of data used by `MainActivity.java`.

**`MainActivity.java` linked to `activity_main.xml`**

(Diagram 11, `MainActivity`, initial and final layout)

**Async Task executions:**

A number of Async Tasks are executed prior to the user being able to score or continue to score a match. These tasks are outlined below.
LoadMatchInfo():

As MainActivity.java opens, the very first instruction it receives is to load the intent extra passed to it by the previous activity (GameList.java). This extra contains the match_id of the match about to be scored. LoadMatchInfo() calls upon the “populate_scoresheet.php” file on the server and uses a JSON HTTP “GET” request to get basic information about the match about to be scored.

This includes the various scores for each team, as well as the team_id’s for the home and away team. These will be used by other Async Tasks when posting values for run scoring. When LoadMatchInfo() has completed its execution it triggers the start of the LoadAllPlayers() Async Task.

LoadAllPlayers():

LoadAllPlayers() performs an almost identical task to that of the LoadAllTeams() Async Task described in the TeamSetup.java class. The only difference being that rather than loading all the teams to the relevant spinners LoadAllPlayers() loads the relevant players for the match being scored onto the spinners: bowlerSpinner (the spinner at the top of the screen), batASpinner (the middle spinner), and batBSpinner (the bottom spinner).

These spinners are crucial as they allow the user to describe the bowler and two batsmen involved in all scoring activities. Each spinner has an onClickListener which is invoked anytime the user clicks a scoring button. When LoadAllPlayers() has completed its execution it fires the LoadPlayerAScores() Async Task.

LoadPlayerAScores() & LoadPlayerBScores():

LoadPlayerAScores() takes in the string value of the batASpinner and the match_id in order to load the current score of the batter on that spinner from the Batting table. It does this by passing a JSON HTTP “POST” request to “player_scores.php” which sits on the server. First the players table is queried for the player_id of the player whose name is listed on batASpinner. Then, using that id along with the posted match_id the Batting table is queried and the resulting information is passed into an array and
pushed into the JSON response. The JSON response is decoded in the application and the player’s score is set beside the spinner along with a calculation of their strike rate.

Upon completion of this task, the LoadPlayerBScores() is fired. This task does exactly the same thing as LoadPlayerAScores() but for the second batsman. Once LoadPlayerBScores() has executed the game is finally ready to be scored.

A major issue in this application was ensuring that these network Async Tasks were completed in sequence of each other rather than in parallel to avoid a common error known as a “leaked window”. This error occurs when two tasks are competing for resources on the same thread of an application. It causes a full crash of the application.

AddRunsA() & AddRunsB():

As with the Async Tasks above AddRunsA() passes a number of POST parameters to “add_runs.PHP” resting on the server. All scoring logic is handled by this PHP file. These parameters are defined in each of the scoring buttons on the screen, and are activated by their respective onClickListener(). The onClickListener() then fires the AddRunsA() Async Task and makes note of the following parameters for insertion to the score sheet: wd, nb, wkt, bye, action, player_name, bowler_name, and match_id. Below follows a list of the various actions performed on the database when the AddRunsA() Async Task is executed. For each of the scenarios if a batsman or bowler is not yet listed in the Batting or Bowling table under the match_id for the match being scored, they are inserted into the match prior to any updates on their scoring record are completed.

Dot:

**Parameters passed to PHP file:**

*action = “0”*;

**MySQL update:**

Batsman: balls_faced = balls_faced + 1;
Bowler: balls = balls + 1;
Wicket:

**Parameters passed to PHP file:**

```php
action = "0";
wicket = "1";
```

**MySQL update:**

Batsman: `balls_faced = balls_faced + 1, dismissal_bowler_id = bowler id`
Bowler: `balls = balls + 1, wickets = wickets + 1`;
Match: batting team wickets = batting team wickets + 1;

No Ball:

**Parameters passed to PHP file:**

```php
action = "0";
nb = "1";
```

**MySQL update:**

Batsman: `balls_faced = balls_faced + 1`;
Bowler: `num_nb = num_nb + 1`;
Match: batting team runs = batting team runs + 1;

Wide:

**Parameters passed to PHP file:**

```php
action = "0";
wd = "1";
```

**MySQL update:**

Batsman: `balls_faced = balls_faced + 1`;
Bowler: `num_wd = num_wd + 1`;
Match: batting team runs = batting team runs + 1;

4 Bye:

**Parameters passed to PHP file:**

```php
action = "0";
bye = "4";
```

**MySQL update:**

Batsman: `balls_faced = balls_faced + 1`;
Bowler: `balls = balls + 1`;
Match: batting team runs = batting team runs + 4;
Bye:

**Parameters passed to PHP file:**
- `action = “0”`;
- `bye = “1”`;

**MySQL update:**
- Batsman: `balls_faced = balls_faced + 1`;
- Bowler: `balls = balls + 1`;
- Match: batting team runs = batting team runs + 1;

1,2,3,4,5,6 Runs:

**Parameters passed to PHP file:**
- `action = “1-6”`;

**MySQL update:**
- Batsman: `balls_faced = balls_faced + 1; runs = runs + action`;
- If (`action = 4`) `fours = fours + 1;` if (`action = 6`) `{sixes = sixes +1;}`
- Bowler: `balls = balls + 1; runs_conceded = runs_conceded + action`;
- Match: batting team runs = batting team runs + `action`;

Other:

Not yet functional

AddRunsB() functions exactly like AddRunsB() except it passes the parameters of the second batsman, and listens for the scores from the second batsman’s scoring buttons.

**Web Interface**

The web interface was designed using a combination of PHP, CSS, HTML, and Javascript. The Javascript is mainly in combination with the CSS to make the navigation bars and hover over effects of the tables look more visually appealing. The full web interface for viewing match, team, and player statistics can be viewed at:

http://mallamp.com/print_match_table.php

Matches

A list of matches existing in the database is displayed in a table. The match table includes information on the home and away teams, the match types, the match sub types
and the date on which the match occurred. Each table row has a link to the relevant scorecard for that match.

Following the link sends a html form action “post” method containing the match_id of the match on that row to the “view_match.php” page. The view match page is created using queries based on the match_id. The many queries required to display this page are not going to be listed here as they can be viewed in the PHP files included on the disc accompanying this paper. Match statistics are displayed in a format similar to other cricket statistics reporting websites. Every effort has been made in the css files (adapted from twitter’s bootstrap) to make the statistics display consistently in a number of different browsers. So far Google Chrome, Mozilla Firefox, and Internet Explorer are displaying information correctly.

Each player listed in the scorecard is linked to their unique statistics page. On this page there is a table listing their batting and bowling statistics for all games in the database. Their totals, and averages for the statistics shown are also included. On the right hand side of the page is a fixed records sheet with a profile picture of the player, their name, a link to their team page and their own records table. If the contents of the batting and bowling statistics were to extend the height of the screen, when the user scrolls down, the right hand side profile information and records attributed to the player would remain fixed in position, appearing to float beside the information.

Players

The players page lists out all the players in the database in a grid formation of profile pictures. Each name is a link to that person’s individual statistics. To the right is the beginning of a player-wide record assessment. At the moment this only includes the player with the most runs in a single game, the player with the most wickets in a single game, and the player with the most maidens in a single game. In the future this could be expanded to include far more in depth queries into the data base. These records also stay fixed in position as the user scrolls up and down through the data.

Teams

Similar to the players page, the teams page displays a grid of team logos and names representing the teams stored in the MySQL database. From this page a user can
access individual team statistics. Each team page includes a table documenting the matches they have competed/are competing in and also a full grid listing of players on their squad. As before these player icons will lead the user to that player’s specific profile.

In summation the web interface adequately displays the current information in the database, and can quickly display updated information as it arrives into the database from the android application. In the next section future developments and improvements for the web interface and android application are discussed.

**Future Work**

Structured evaluation:

So far the android application has been tested to abide by the Laws set out in the introduction of this report. It has not been thoroughly tested. A recent trial scoring of an Indian Premier League T20 match between Kings XI and Pune Warriors was tested by the developer for 4 complete overs. In those four overs the scoring rules and dismissal rules were behaving as they should be, however further testing would be beneficial to the project. The relevant scorecard can be viewed on the web interface.

Although the android application has been completed in such a way that it is possible to score a cricket match, it has yet to be tested fully. Ideally a structured evaluation would include 5-10 participants with a basic knowledge of the Laws of cricket attempting to score a live cricket match. This could be done in one of two ways, either at a cricket ground where all five participants attend the match and attempt to score the game, or through a simulation using video of a recently scored match. Both scenarios would test the accuracy of the application, and would seek to highlight issues so far unseen.

Rule improvement:
Future rule improvement to the application would incorporate the test match style of play as well as the limited overs style of play. At the moment it is possible to score a test match, but the onus is on the user to keep their wits about them when identifying the bowling and batting players.

In terms of the methods of dismissal, a major part of the application, at the moment a player can only be declared out, rather than declared out and have their method of dismissal marked on the scorecard. The next step towards improving this involves a pop-up message prompting the user to select the method of dismissal, and if applicable any other players involved in the dismissal.

Android UI improvement:

The current layout of the android UI uses two scoring inputs, making the screen look cluttered. This was largely to reduce the complexity of having to program a method to determine who was at the batsman’s crease at any given time. The testing device used for this application had a 4.3” screen, one of the larger in the android device pool. Having not been tested on a smaller device it is safe to say the cluttered input may display drastically differently. Further UI improvement would focus on reducing clutter as well as incorporating feedback from user evaluation.

Web improvement:

First among the changes would be to set up a simple user login function. This would simply mean creating a new table in the MySQL database and creating a schema allowing for tiered user access. For example a regular user may only be able to browse while an administrator would be able to edit data from within the web interface. Following that a new page to add teams and players to the database from within the web interface would be beneficial. Currently any new teams and players must be added via the PHPMyAdmin interface with the MySQL database. A search page could also be set up to query the data by date, ground, players, score or any other number of combinations of the variables stored in the database.
An update to include an events “ticker” in the style of text commentary could be included at the top of each scorecard as a way for the reporter to relay commentary via the android application. In order for this to be set up a new table would have to be set up in the MySQL database with parameters such as `event_id`, `match_id`, and `commentary`. In this manner each match would have a record kept of all commentary throughout.

An update to store ball by ball information for over-by-over score graphing would also be encouraged. This data would require a separate table to be created storing the `match_id` and over record for each match.

In summary the current structure of the database and web interface allows for multiple simple enhancements. This highly adaptable platform has been regarded as a success in the project.
Conclusion

As far as completing an android application and web interface for viewing the information created by that application are concerned, the Live Score Cricket project was a success. The android application could become commercially viable for a company such as TEAMtalk Media, and certainly improves on the current procedures followed. However, it is important to note that this project is still in beta, and that a complete package reliable enough to produce statistics and data for the likes of bookmakers is still a little ways off. As proof of concept and an academic exercise this project fulfilled its requirements.

Acknowledgements:

I’d like to acknowledge the guidance provided by Mary Sharp throughout this project. It’s always great to meet a fellow sports enthusiast.
References:


Appendix:

Figure 1:

TEAMtalk Brands:

Cricket365.com  planetsport.com
GOLF365.com  PLANET-F1.com
EXTREME365.com  planetrugby.com
OFFonF1.com  FOOTBALL411.com
S4Rugby.com

TEAMtalk Networks/Publishers:

Supersport  Live  Media24  Vodacom  Sky Sports
MTN  vodacom  Infomedia
Yahoo!  ESPN  Praekelt Foundation  Phumelela

TEAMtalk Official Data Partners:

(Source: TEAMtalkMedia.com, April 2013)
Figure 2 – Mobile Penetration in Africa

(Opera, State of the Mobile Web, July 2012)
Figure 3 – Initial Match Setup Prototyping:

…Continued on next page…
Figure 4 – Initial Scoring Page Prototyping:

- Each ‘button’ adjusts the score accordingly and advances play accordingly.
- At the end of each over the user is prompted to select the new bowler.
- At each wicket the user is prompted to select the new batsman and method by which wicket caused:
  - bowled
  - caught
  - run out
- maybe not necessary?
### Batting

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Default</th>
<th>Links to</th>
</tr>
</thead>
<tbody>
<tr>
<td>player_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Players -&gt; player_id</td>
</tr>
<tr>
<td>match_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Matches -&gt; match_id</td>
</tr>
<tr>
<td>arrival_score</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>arrival_wickets</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>arrival_overs</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>departure_score</td>
<td>int(11)</td>
<td>Yes</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>departure_wickets</td>
<td>int(11)</td>
<td>Yes</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>departure_overs</td>
<td>int(11)</td>
<td>Yes</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>runs</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>fours</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>sixes</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>dismissal_type</td>
<td>varchar(45)</td>
<td>Yes</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>dismissal_bowler_id</td>
<td>int(11)</td>
<td>Yes</td>
<td>0</td>
<td>Players -&gt; player_id</td>
</tr>
<tr>
<td>caughtby_id</td>
<td>int(11)</td>
<td>Yes</td>
<td>0</td>
<td>Players -&gt; player_id</td>
</tr>
<tr>
<td>team_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Teams -&gt; team_id</td>
</tr>
</tbody>
</table>

### Bowling

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Default</th>
<th>Links to</th>
</tr>
</thead>
<tbody>
<tr>
<td>player_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Players -&gt; player_id</td>
</tr>
<tr>
<td>match_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Matches -&gt; match_id</td>
</tr>
<tr>
<td>overs_bowled</td>
<td>float</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>balls_bowled</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>maidens</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>runs_conceded</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>wickets_taken</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>num_nb</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>num_wd</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>team_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Teams -&gt; team_id</td>
</tr>
</tbody>
</table>

### Matches

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Default</th>
<th>Links to</th>
</tr>
</thead>
<tbody>
<tr>
<td>match_id</td>
<td>int(11)</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>varchar(15)</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>subtype</td>
<td>varchar(15)</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ground_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Grounds -&gt; ground_id</td>
</tr>
<tr>
<td>date</td>
<td>date</td>
<td>No</td>
<td>00000-00-00</td>
<td></td>
</tr>
<tr>
<td>balls_per_over</td>
<td>int(1)</td>
<td>No</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>home_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Teams -&gt; team_id</td>
</tr>
<tr>
<td>away_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Teams -&gt; team_id</td>
</tr>
<tr>
<td>toss_win_team_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Teams -&gt; team_id</td>
</tr>
<tr>
<td>home_runs</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>home_wickets</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>away_runs</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>away_wickets</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>umpireA_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Umpires -&gt; umpire_id</td>
</tr>
<tr>
<td>umpireB_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Umpires -&gt; umpire_id</td>
</tr>
</tbody>
</table>
## Players

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Default</th>
<th>Links to</th>
</tr>
</thead>
<tbody>
<tr>
<td>player_id</td>
<td>int(11)</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>team_id</td>
<td>int(11)</td>
<td>No</td>
<td>0</td>
<td>Teams -&gt; team_id</td>
</tr>
<tr>
<td>name</td>
<td>varchar(45)</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fullname</td>
<td>varchar(45)</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>image_link</td>
<td>varchar(150)</td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Teams

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>team_id</td>
<td>int(3)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>varchar(30)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>image_link</td>
<td>varchar(150)</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

## Umpires

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>umpire_id</td>
<td>int(3)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>varchar(30)</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

## Grounds

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ground_id</td>
<td>int(3)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>varchar(45)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>city</td>
<td>varchar(15)</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>country</td>
<td>varchar(20)</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>