2010/11

FINAL REPORT

LOCAL HOLD'EM POKER FOR THE GOOGLE ANDROID OS

BACHELOR OF SCIENCE (H) IN SOFTWARE SYSTEMS DEVELOPMENT

V.3

15TH MAY 2011

DECLARATION OF AUTHENTICITY

Except where explicitly stated, this report represents work that I have done myself. I have not submitted the work represented in this report in any other course of study leading to an academic award.

TABLE OF CONTENTS

TABLE OF FIGURES

1. INTRODUCTION

This document will give an account of the development that was undertaken and the results reached for this final year project, Local Hold'em Poker for the Google Android OS, for the course, Bachelor of Science (H) in Software Systems Development. The findings will be described under the following headings: Chosen Platform, Game Type, Main Aim, Target Audience, Software Functionality, Connectivity, Gameplay, Tools and Technologies, Game Design, Methods and Methodologies, Implementation, and Post Development Analysis.

2. CHOSEN PLATFORM

The project was developed for the Google Android OS because the Android is an emerging market and is becoming the main competitor to Apple's iPhone. This holds huge significance for open source development of mobile applications because the Android Market is open to all developers to upload their work and receive feedback from a worldwide audience where as the iPhone App Store is much larger and due to this has become much more difficult to enter by "red-tape" procedures. Developers have waited weeks or even months for their applications to be published in the App Store.

The paid app feature of the Android Market only became available to Irish developers and users on the 30th September 2010. According to Tim Bray (2010), this enables developers to publish and sell their paid apps and games on the Android Market which makes the platform much more lucrative and attractive to develop on.

3. GAME TYPE

Local Hold'em Poker is a completely **free-to-play** multiplayer Texas Hold'em Poker game that can be operated through local connections, (e.g.) Bluetooth and an ad-hoc wireless network, on a Google Android OS compatible phone. Texas Hold'em (also known as hold'em) is a version of the card game poker. After much research of the available Android games and applications, it was found that a project of this type was not available on the market as of 12th November 2010.

Development of this type of game was chosen because Texas Hold'em Poker has exploded in popularity in the last ten years or so and according to Clark (2006), during this time hold'em replaced seven-card stud as the most common game in U.S. casinos. This is a massive step in the right direction in the popularity and recognition stakes for the game. Texas Hold'em is played in homes, pubs, casinos, online, on television and on some mobiles at the moment. This makes hold'em accessible to a huge audience and a vast market which gives this project potential commercial value.

Refer to Figure 16.1 in Appendix C for a Use Case diagram of the project.

4. MAIN AIM

The main aim of this project was to develop a fully functional and completely **free-to-play** local multiplayer Texas Hold'em Poker game that would run on the Android OS. By local, as mentioned in the game title "Local Hold'em Poker", it was meant that the game would be completely playable offline, (i.e.) the user will not need an internet connection. This guaranteed one of the most important features of the game, the free-to-play feature.

5. TARGET AUDIENCE

The target audience for this project was Android compatible phone users who have used or have an interest in using poker apps and games on their phone. It appeals to friends, groups and clubs who wish to play hold'em with their peers locally without having to pay any fee for the service, (e.g.) connection fees, data transfer fees etc. The game is downloadable from the game's website – http://androidlocalholdem.com.

6. SOFTWARE FUNCTIONALITY

6.1. FEATURES

1. The game runs on the Google Android OS.
2. The user can play one-on-one ("heads up") Texas Hold'em Poker with other human players through a Bluetooth connection.
3. Each player can register their details, via their phone or PC, to keep a record of their results, head to heads, and position on the Limit and No Limit league tables. This is stored in the game's website database.
4. The game is free-to-play.
5. The game is available for download from the game's website – http://androidlocalholdem.com .
6. The user interface is simplistic, yet friendly.
7. The user can play Limit and No limit hold'em.
8. The user can play an exhibition or competitive game of both Limit and No limit hold'em.
9. The user can upload the result of a competitive game to their account in the database. Exhibition results are not uploaded to the user's account.
10. The user can choose their avatar, which displays while playing, from the default images provided or an image saved in their phone gallery. This is saved but can be changed in the Settings screen at any time.
11. The user can choose between a Wi-Fi and mobile internet connection to register/login. This is saved but can be changed in the Settings screen at any time.
12. The user can reset a forgotten password through the application and receive a new password to the email address they provided at registration time.
13. The user's details are validated on registration because each username and email address is unique in the database.
14. The user is logged in until explicitly logging out in the Setting screen. This persists through the application's states – On, Off.
15. Sound effects are provided for the most important user actions, (e.g.) call, check

7. CONNECTIVITY

Local connectivity was a vital feature to the successful completion of the project because it enables the multiplayer aspect and ensures the free-to-play mantra of the game. The factors that were considered when choosing a primary connectivity network were:

- the Android OS supported the network
- the range of the network
- the drain on the battery power of the phone
- the transmit speed of the network

WirelessArea Network	Range	Power Drain	Transmit Speed*	Example	Primary Application/ Usage Scenario
Wireless Personal Area Network (WPAN)	10 m	Low	800 Kbps	Bluetooth	Cable replacement between nearby devices
Wireless Local Area Network (WLAN)	100 m (to an access point)	Medium	11 Mbps	Wi-Fi (IEEE 802.11b)	Accessing an existing Ethernet network run on cables

Figure 7.1. Characteristics of different Wireless Area Networks

The table above (Figure 7.1.) presents two types of wireless networks and their characteristics according to Man (2002) in a White Paper commissioned by Socket Communications Inc. Figure 7.1. holds useful information relating to the Local Hold'em project. It shows that a Wireless Personal Area Network (WPAN), (e.g.) Bluetooth, has a range of 10 metres, a low power drain and a transmit speed of 800 Kbps which makes this type of network lightweight and suitable to use with mobile phone applications. In the Android OS, APIs are provided to support Bluetooth settings. These APIs manage discoverability, connection to Bluetooth devices and act as a transport layer for data transfer between devices. Meier (2010) states that the following classes handle Bluetooth devices and connections:

- BluetoothAdapter – The Bluetooth Adapter represents the local Bluetooth device, (i.e.) the device your application is running
- BluetoothDevice – Each remote device with which you wish to communicate is represented as a BluetoothDevice
- BluetoothSocket – Call createRfcommSocketToServiceRecord on a remote Bluetooth Device object to create a Bluetooth Socket that will let you make a connection request to the remote device, and then initiate communications
- BluetoothServerSocket – By creating a Bluetooth Server Socket (using the listenUsingRfcommWithServiceRecord method) on your local Bluetooth Adapter, you can listen for incoming connection requests from Bluetooth Sockets on remote devices

The second network that is detailed in Figure 7.1. is a Wireless Local Area Network (WLAN), (e.g.) Wi-Fi. This network can have a range of 100 metres to an access point, a medium power drain and a transmit speed of 11 Mbps. Wi-Fi network connectivity is implemented in the Android OS but this does not support an ad hoc wireless network. This issue has been raised on the official Android development site on Google Code and is referred to as "Issue 82: wifi - support ad hoc networking" (Google Code 2008). The issue was initially raised in 2008 and was expected to be resolved in the latest version of the Android OS, called Froyo (2.2), but as of 12th November 2010, the issue remained unresolved.

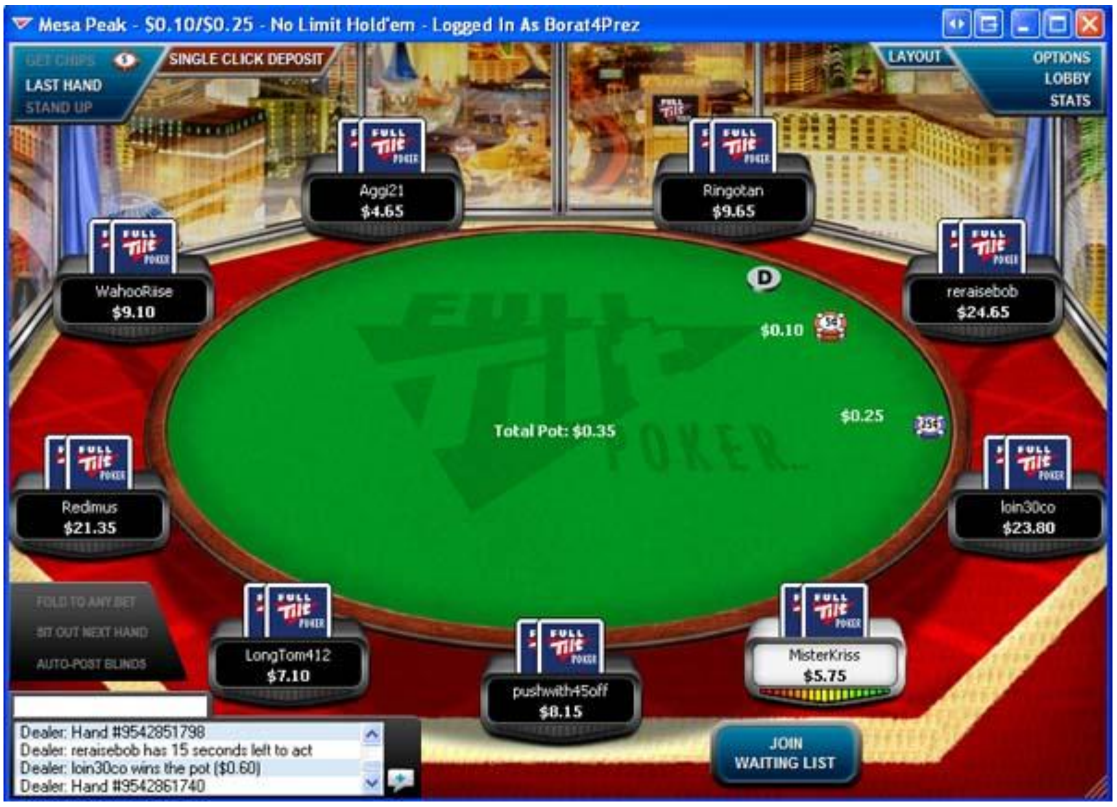
Jradi and Reedtz (2010), two students of the Technical University of Denmark, offer a solution to the ad hoc network issue in their degree thesis. The goal of their project was to "design and implement a suitable distributed routing protocol to manage the communication among many Android devices, running concurrently". This protocol could be developed further and implemented into the Local Hold'em project to give the user a local connection with greater range but due to time constraints this feature has been targeted for a later iteration, outside of the allocated time for this project.

8. GAMEPLAY

Before any card is dealt a player is designated as the “Dealer” for the hand. The player to the Dealer’s left is designated the “small blind” and the next player on his left the “big blind”. The small and big blinds are predefined amounts that have to be bet at the beginning of each hand. The Dealer, small blind and big blind rotate clockwise by one place after each hand has been completed and in some games, the blinds can increase in value after a full round of the table has been completed. This helps the game reach a conclusion by forcing players to bet. When there are two players playing or “heads up play” the Dealer gets the small blind and the other player gets the big blind.

Hold’em is a turned based game where each player has a predefined number of chips (money) at the beginning of the game to bet with. The game begins when the Dealer gives each player (including himself) two cards face down. The small and big blinds are entered into the “Pot” (this is the amount of chips that can be won in the hand) and each player, in turn, gets the choice to call (i.e. enter the same amount of chips as the big blind), fold, or raise. If a player raises, then all other players will have to call or re-raise to continue playing in the hand. In Limit hold’em, there is a “cap” on the number of raises in each round, generally three or four. This first round of betting is known as the “Pre-flop”, shown in Figure 8.1.

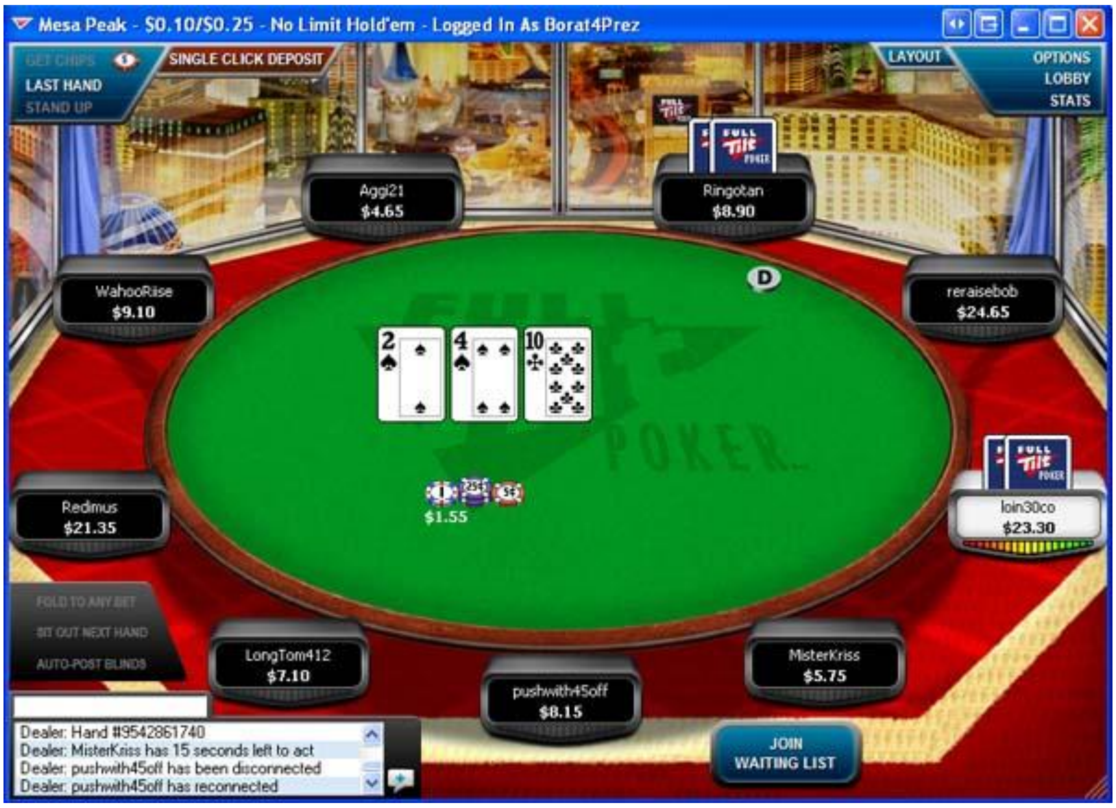
The following five images, Figure 8.1., 8.2., 8.3., 8.4., and 8.5. are taken from FlopTurnRiver (2003).



Dealer: Hand #9542851798
Dealer: reraisebob has 15 seconds left to act
Dealer: loin30co wins the pot ($0.60)
Dealer: Hand #9542861740

Figure 8.1. The Pre-flop

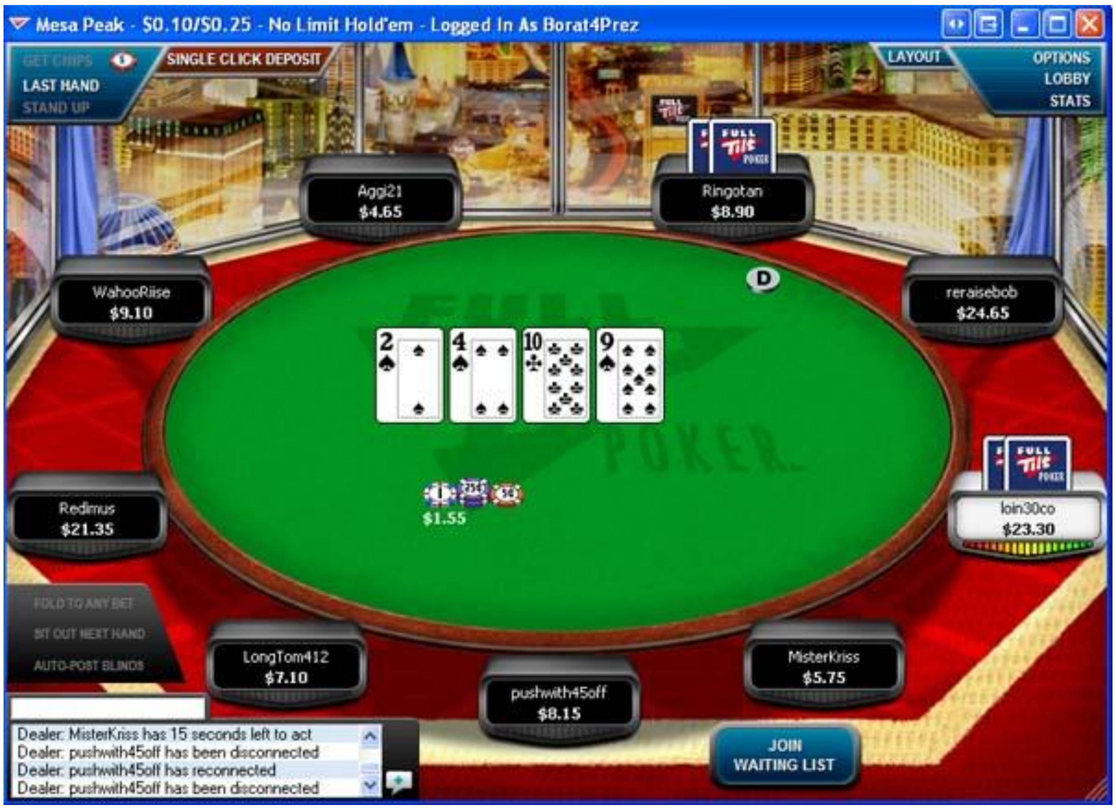
After the first round of betting has been completed, the Dealer places three cards face up on the table. This is called the “Flop”, shown in Figure 8.2. Another round of betting ensues and the remaining players have the choice to check (i.e. see the next card face up without betting if all other players check too), fold or raise.



```
Dealer: Hand #9542861740
Dealer: MisterKriss has 15 seconds left to act
Dealer: pushwith45off has been disconnected
Dealer: pushwith45off has reconnected
```

Figure 8.2. The Flop

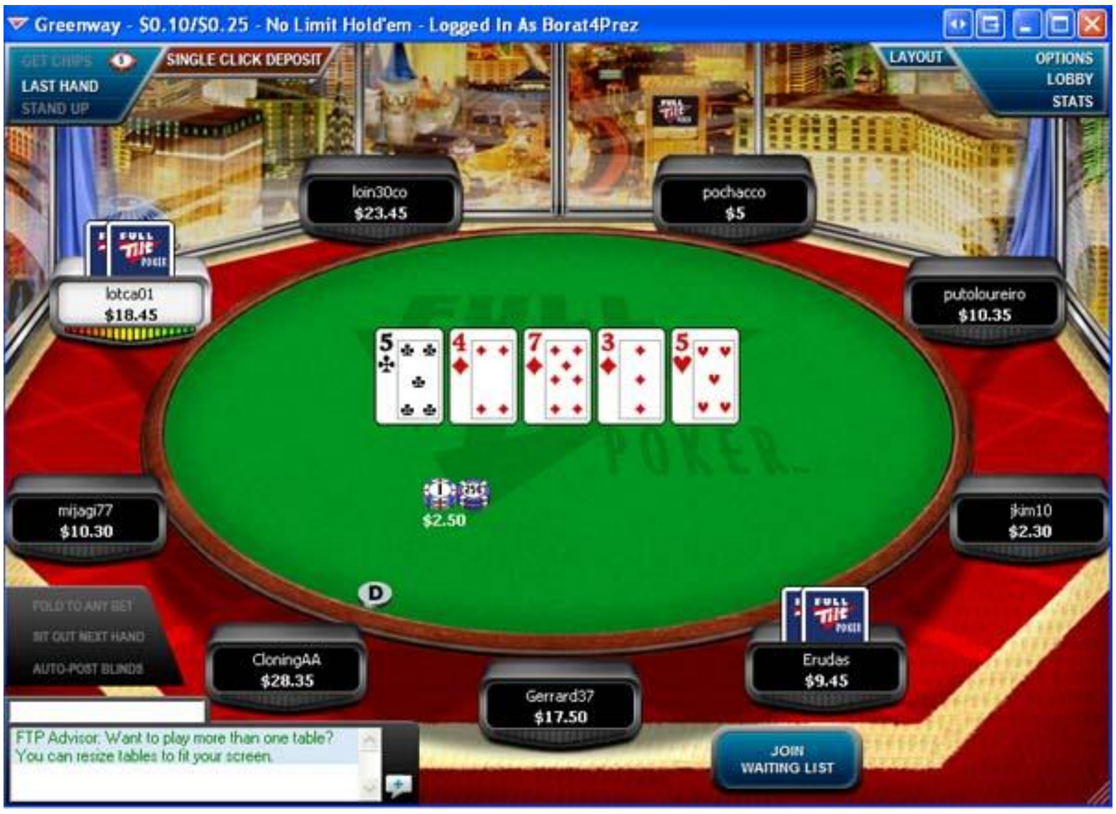
After all betting is complete on the Flop, a fourth card is placed face up on the table by the Dealer and this is called the "Turn", shown in Figure 8.3. A round of betting begins with the first player to the Dealer's left as always.



```
Dealer: MisterKriss has 15 seconds left to act
Dealer: pushwith45off has been disconnected
Dealer: pushwith45off has reconnected
Dealer: pushwith45off has been disconnected
```

Figure 8.3. The Turn

When the betting concludes on the Turn, a fifth and final card is placed face up on the table called the "River", shown in Figure 8.4. The five cards are collectively called the "Community Cards". A final round of betting takes place.



FTP Advisor: Want to play more than one table? You can resize tables to fit your screen.

Figure 8.4. The River

When the betting concludes, the remaining players turn their cards face up and the player with the best hand wins. The Community cards are used by the players, with their own two cards, to create the best possible five card hand. In Figure 8.5., the ranking of poker hands is illustrated.

9. TOOLS AND TECHNOLOGIES

9.1. HARDWARE

- Dell Inspiron 1525: Intel Core2 Duo T8300 @ 2.40 Ghz, 4.00 GB RAM – this is the specification of the laptop that the project was developed with
- 2 x Samsung Galaxy Europa GT-I5500 Google Android OS phones – these phones run Android 2.1. and were used to test the application

9.2. SOFTWARE

- Adobe Photoshop CS4 – Graphics editing software
- Android 2.1. SDK – Android Software Development Kit which includes an emulator, Bluetooth classes, Dalvik Virtual Machine and Daemon tools
- Audacity – Sound editing software
- Eclipse IDE – Java Integrated Development Environment and this also handled the XML requirements
- Java SDK – Java Software Development Kit
- Microsoft Office 2007 – Includes document, spreadsheet and presentation authoring tools

- Windows Vista Home Premium – Operating system

10. GAME DESIGN

10.1. LOOK AND FEEL

- The player has a portrait, top down view of their cards and avatar, the community cards, and their opponent's cards face down and their avatar.
- The player's stack and the pot's value are shown in text and number format.
- There's a user interface at the bottom of the screen for fold, check, call and bet/raise buttons. The touch screen allows the user to select a button. These buttons exit the screen from the bottom when it is the opponent's turn.
- A small dialog box is displayed for a couple of seconds at each turn to tell the players whose turn it is.
- When a player selects the raise or bet button, a dialog box is displayed to facilitate how much they are allowed (their current chip stack) and want to raise by. This also facilitates an All-in raise.
- A dealer button is displayed on screen at all times to indicate the dealer of the current hand.
- At the beginning of a hand, small and big blinds are indicated to the players in a small dialog box that is displayed for a couple of seconds.
- A text field near the player's details will display the amount a player has to call in a particular scenario or if a player checked.
- The action is supplemented by sound effects, (e.g.) the deck shuffle before a hand is dealt

10.2. POKER ENGINE

Due to the vast amount of logic involved in creating a poker game, and the limited time allocated to this project, a number of open source poker games, that have been created in Java and other platforms, were analysed and tested with the view to further develop specific poker methods, (e.g.) hand evaluation, betting algorithms, deck classes etc., that were used in this project. These methods were rewritten and reconfigured for implementation in a Bluetooth network application but they provided a useful starting point.

Oscar Stiger's (2009) Texas Hold'em in Java hosted on Google Code is an example of a project that was very useful. It is the limit version of hold'em and uses Swing GUI's for a console interface which can't be used in the Android environment but his hand evaluator class was successfully implemented with some changes to the code. Stiger's deck class was also reconfigured to perform in the Android environment.

11. METHODS AND METHODOLOGIES

11.1. WORK-BREAKDOWN SCHEDULE

11.1.1. LIST OF WORK PACKAGES AND WORK FLOWS

0. Local Hold'em Poker for the Android OS
1. Requirements Analysis
- **1.1.** Feedback
  - **1.1.1.** Questionnaire
  - **1.1.2.** Unstructured Interviews
  - **1.1.3.** Similar Interest Groups
- **1.2.** Complete the Android tutorial apps
- **1.3.** Unit test Java Poker game engines
- **1.4.** Research the Android Connectivity Library's
  - **1.4.1.** Bluetooth
  - **1.4.2.** Wireless ad hoc network
- **1.5.** Report 1: Feasibility Study
- **1.6.** Report 2
  - **1.6.1.** Analysis
  - **1.6.2.** Design

2. Project Demo
- **2.1.** Code skeleton app for demo purpose
- **2.2.** Presentation

3. Iteration 1
- **3.1.** Implement Screens
  - **3.1.1.** App Welcome Screen
  - **3.1.2.** Game choice screen
  - **3.1.3.** Connection choice screen
- **3.2.** Implement Bluetooth connection

4. Report 3: Project progress update document
5. Iteration 2
- **5.1.** Implement card, table and player images
- **5.2.** Test: Complete a Bluetooth hand
  - **5.2.1.** Two player hand of Hold'em
  - **5.2.2.** Bug fixing
- **5.3.** Implement table GUI
- **5.4.** Hand evaluator

6. Iteration 3
- **6.1.** Test: Complete a Bluetooth game
  - **6.1.1.** Two player game of Limit and No Limit Hold'em
  - **6.1.2.** Bug fixing
- **6.2.** Implement betting function
  - **6.2.1.** Limit betting
  - **6.2.2.** No Limit betting

7. Implement game website
8. Report 4: Final project progress update document
9. Final Project Test Feedback
- **9.1.** Android forum tester's feedback
- **9.2.** Implement feedback

10. Final handover to project supervisor

11.2. PROJECT MILESTONE SUMMARRY

Project Milestone Summary Table	
Task Name	**Date**
Project Start	20/09/2010
End of Analysis and Design Phase	07/12/2010
Iteration 1 Completed	10/02/2011
Iteration 2 Completed	14/03/2011
Iteration 3 Completed	25/04/2011
Project End	16/05/2011

Figure 11.2. Project Milestone Summary Table

12. IMPLEMENTATION

The implementation of the game was the most important part of the project. All the analysis and design work had previously been undertaken and completed in the Analysis and Design document before implementation began. Some of the planned milestones and iteration completion dates were rescheduled due to unforeseen risks that were realised very early on in the implementation phase.

12.1. REALISED RISKS

In the Analysis and Design phase, the risks for the project were documented. These included unrealistic time estimates, real-time performance and personal shortfalls, health problems, developing the wrong user interface and hardware shortfalls.

At the beginning of the implementation phase, hardware problems occurred which was evaluated and documented as a high impact risk. The development laptop, as per the hardware project specification, crashed and was unresponsive to any recovery efforts so it was passed onto an expert in the hardware field. This delayed the implementation phase by three to four weeks which drastically affected the development schedule and led to some revisions in the game's feature set. The revised list of workflows and packages are in Section 11.2 and the new iteration completion dates are illustrated in Section 11.3, Figure 11.2.

12.2. USER INTERFACE

12.2.1. LAYOUT

In the Android environment, the user interface is built using View and ViewGroup objects, (Figure 12.1.). These can be created through the use of xml files or at runtime (in the game's JAVA code). The Android Developer's documentation (2011) states that View objects are the basic units of user interface expression on the Android platform. The View class serves as the base for subclasses called "widgets," which offer fully implemented UI objects, like text fields and buttons. The ViewGroup class serves as the base for subclasses called "layouts," which offer different kinds of layout architecture, like linear, tabular and relative. A View object is a data structure whose properties store the layout parameters and content for a specific rectangular area of the screen.

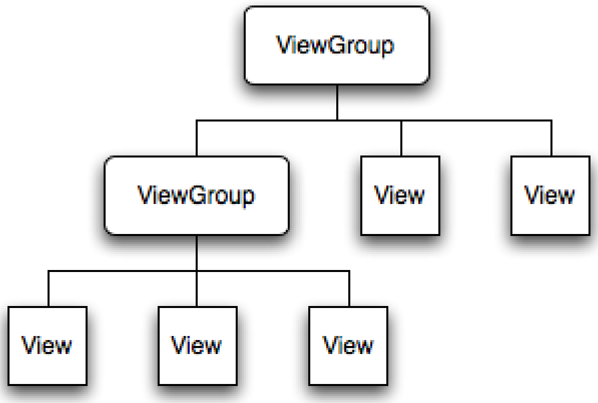


Figure 12.1. Example of Android View Hierarchy

The xml file excerpt in Appendix A, details the layout, including ViewGroups, Views and attributes, pertaining to the yellow highlighted area of Figure 12.3. This small piece of the user interface equates to almost one hundred lines of xml code which shows the detail involved in creating a simple but user friendly interface for a game of this type.

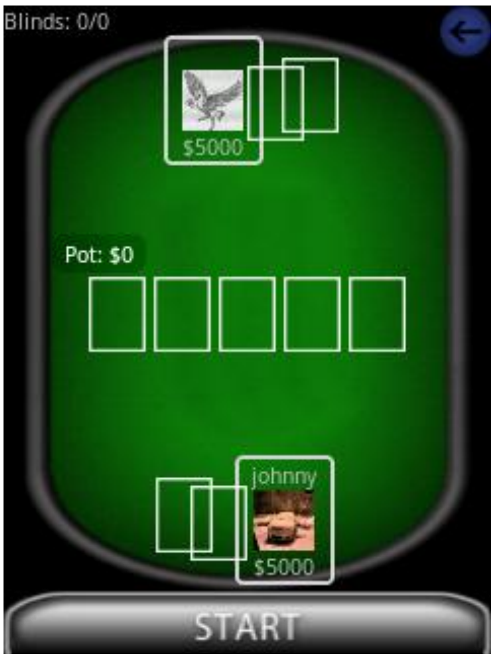


Figure 12.2. Start game

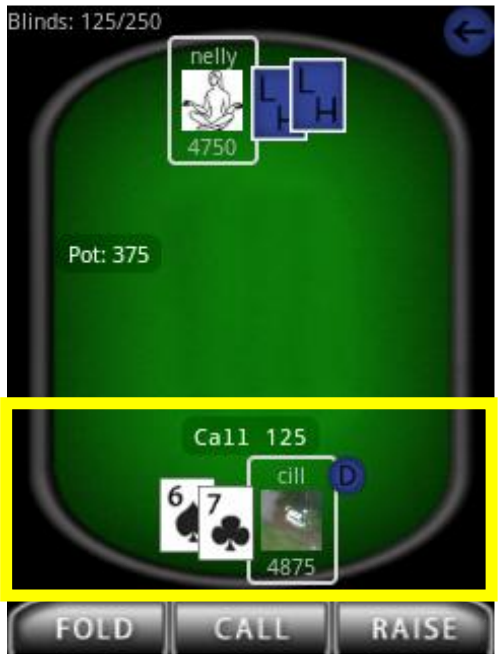


Figure 12.3. Pre-flop options for starter player

Figure 12.4 illustrates the use of the Raise button. A dialog appears above the raise button with a scrollable list of the amounts the player can raise by. The player also has the choice of selecting the All In button or the Cancel button. The Cancel button will remove the Raise dialog and return the player to the original options, shown in Figure 12.3. The raise dialog was implemented with both xml and JAVA code and this is shown in Appendix B.

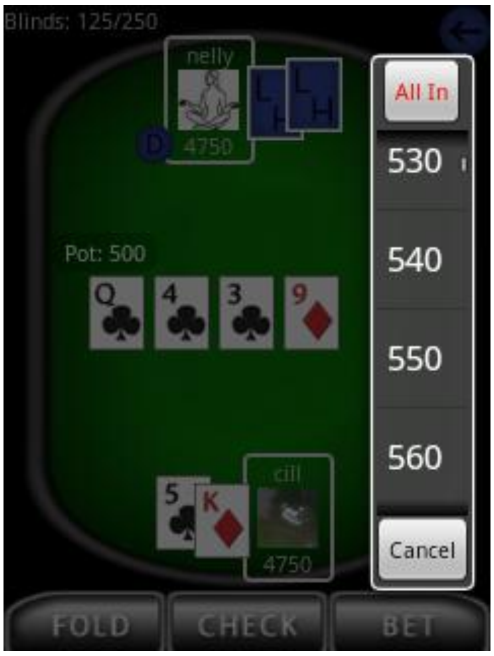


Figure 12.4. Raise dialog

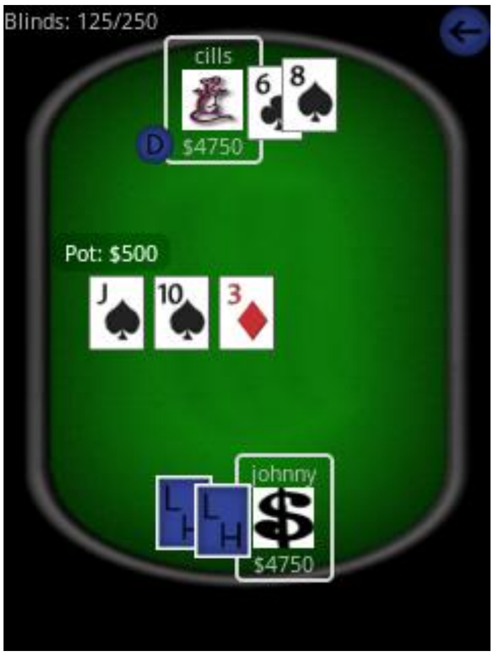
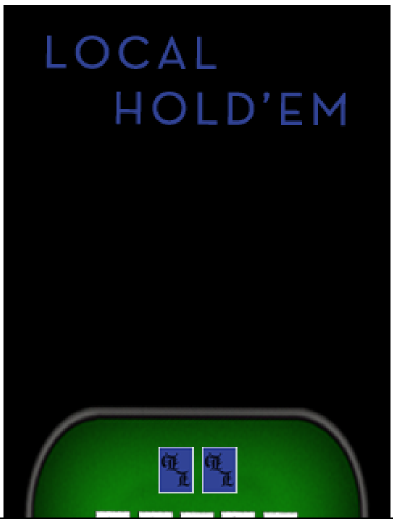


Figure 12.5. The Flop

12.2.2. GRAPHICS

The graphics, for the user interface, were created using Adobe Photoshop CS4 (as per the software project specification). These included the table, twenty eight buttons, fifty four cards (includes the card back and card holder), and eight default avatars. Examples of these are illustrated in the table below, Figure 12.6.

GAME BACKGROUND


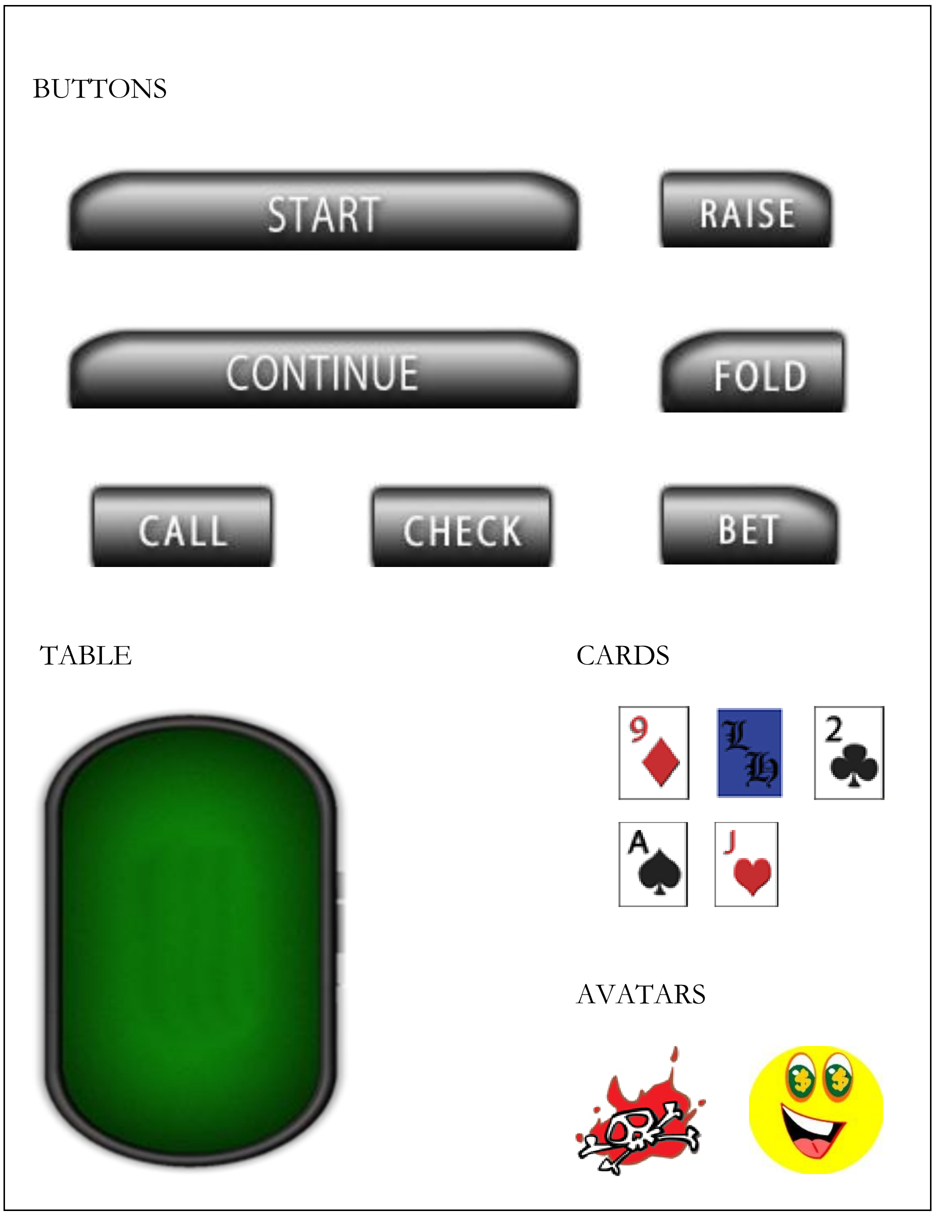


Figure 12.6.Examples of in-game graphics

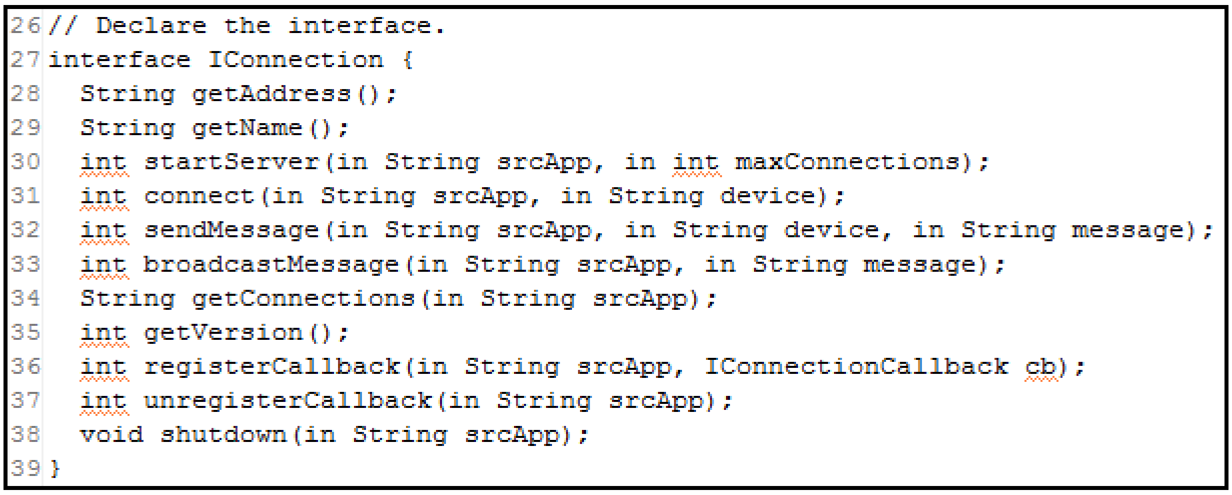
The graphics used in this project are original and took in the region of twenty hours to create. The colours and styles used are maintained throughout the game and give it a uniform look. The Photoshop work included layering, incurring numerous blending options (inner and outer glows, overlays etc.), gradient work on the table and buttons, text styling (kerning), and creating actions for batch work.

12.2.3. SOUND EFFECTS

Sound effects are provided for the most important parts of the game and are included where poker game players would expect them. The sound files were converted to the OGG file format for the best result and compatibility with the Android environment. When the call or check buttons are used, the appropriate voiceover sound effect is emitted to alert both players of the action.

12.3. BLUETOOTH CONNECTIVITY

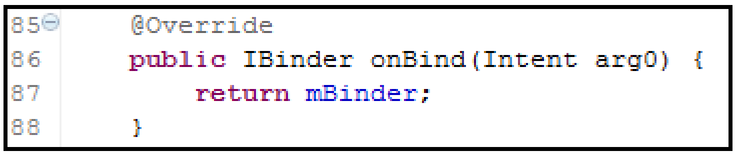
In Section 7. Connectivity, the use of Bluetooth APIs in the Android OS was discussed and this was implemented in the poker.holdem.bt package within the project folder structure. These types of connections can be defined as services in the Android environment. The Local Hold'em game needed a way to send and receive data from the service to its Activities which are using this service. This was done using an IInterface class. These interfaces are quite complex to create, so Google created the "Android Interface Definition Language", – AIDL. This language uses the JAVA syntax and was used in the IConnection.aidl (Figure 12.7) and IConnectionCallback.aidl files in the application. After saving these files in Eclipse, the classes that extend the IInterface, based on the AIDL files, are automatically generated.



```
// Declare the interface.
interface IConnection {
  String getAddress();
  String getName();
  int startServer(in String srcApp, in int maxConnections);
  int connect(in String srcApp, in String device);
  int sendMessage(in String srcApp, in String device, in String message);
  int broadcastMessage(in String srcApp, in String message);
  String getConnections(in String srcApp);
  int getVersion();
  int registerCallback(in String srcApp, IConnectionCallback cb);
  int unregisterCallback(in String srcApp);
  void shutdown(in String srcApp);
}
```

Figure 12.7. IConnection.aidl

Note the arguments in the functions say "in String ....". This signifies the direction of the data, (i.e.) the values are being read in. After the interfaces are defined, they must then be made available to the client classes so they can bind to them. Figure 12.8 shows the implementation of the onBind() method that returns an instance of the ConnectionService class that implements the generated Stub. The Stub class is a subclass of the AIDL class and declares all of its methods. This stub class is called in the method private final IConnection.Stub mBinder = new IConnection.Stub() within the ConnectionService class which also has to extend the android.app.Service class.



```
    @Override
    public IBinder onBind(Intent arg0) {
        return mBinder;
    }
```

Figure 12.8. onBind() implementation

The Connection class simplifies the process of establishing a Bluetooth connection and sending data. It is the link class between the application and the service. In Appendix C, the connect and sendMessage methods in the Connection class are illustrated. These are used for connecting and sending messages between the devices.

All of this backend coding is put into action when the user starts the application. They are immediately presented with a yes/no dialog to enable the Bluetooth adapter on their device if it isn't already switched on by the user. In the Start Game screen, if the user selects the start game button, a yes/no dialog is displayed asking the user to make his device discoverable to other devices and after clicking yes, a progress dialog appears with the message, "Wait for player to connect". This remains on screen until a player connects. The opposing player will select the join game button and will then be presented with a list of the available devices he can connect to. If the devices are connecting for the first time, they will be asked to pair the devices for future use. Both users will choose yes and they can then begin the game. Appendix D displays the OnConnectionServiceReady method which deciphers who started or joined the game and directs them to the correct action.

In the Android framework, security permissions must be explicitly asked for and granted, to allow the use of some of the user's phone features. These features may have an impact on the user's experience so permissions must be acquired. These permissions must be coded in the AndroidManifest.xml file. To grant permission to use the Bluetooth connection and discoverability features on the user's phone, these entries were made in the AndroidManifest.xml file:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

12.4. LOGIN/REGISTER

The first screen in the game that the user is presented with contains three buttons – Login, Register, and Exhibition (one off game that goes directly to ChooseGame screen – no login or registration required). If the user chose's to register, a registration form is displayed with the fields username, password, confirm password, and email. All of these fields are validated on the client side. The username and email provided are validating again on the server side because these two fields have to be unique in the database. The user's information is sent to the Local Hold'em website's (http://androidlocalholdem.com) database, using the Http protocol. A HttpGet call is used to ensure that the user registered successfully. All of this work was done in a worker thread so that the UI thread would not slow down or momentarily freeze. It was coded within a method that extends the AsyncTask class which does this work in a background worker thread.

(e.g.)
```
   private class RegisterUserResponseTask extends AsyncTask<String,
   Void, String> {
      protected String doInBackground(String... urls) {
```

The user login is handled in the same fashion but only requires the username and password. The login screen also offers the user the option to click the forgotten password button where they can enter their unique username and email address which is http'ed to the website database. The username and password are checked against the database as a key/value pair and if they are correct, an email containing a randomly generated password is sent to the provided email address.

Another aspect of the user login/register feature is that the user's preferences are saved. If it is the user's first time using the game, he is asked which internet connection type he wants to use – either Wi-Fi or Mobile, to login/register and this is only enabled for the duration of the Http call. The user is also asked to choose their avatar – a provided game avatar or one from the user's Gallery (personal pictures on device). These two preferences are saved for the user and will not be asked again at start up. The user will also remain logged in until explicitly logging out. In the Settings screen, the user can click the logout button to logout. He can also change the saved connection type and avatar in this screen. These are saved using the SharedPreferences class provided in the Android SDK.

12.5. GAME TESTING

Game testing is a vital part of the implementation process. It was thought during the analysis and design phase that both unit testing and play testing would be used in conjunction with each iteration, but due to time constraints, associated with hardware problems(refer to Section 14.1.), the unit testing of the application had to be excluded. The unit tests would have been very time consuming and would not have tested the Bluetooth features of the game.

Play testing though, has been carried out extensively by a group of four people internally and externally through Android forums and other Android developers online. The game is a free download from the game's website. A direct download link and a QR scan code are provided so everyone has access to it, Figure 12.9.

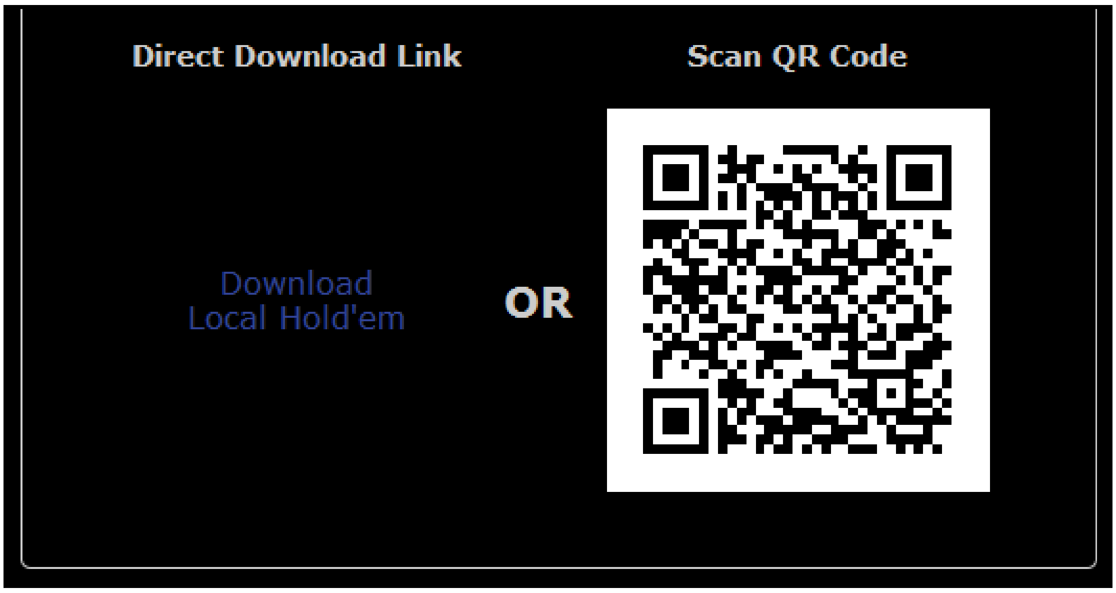


Figure 12.9. Website download options

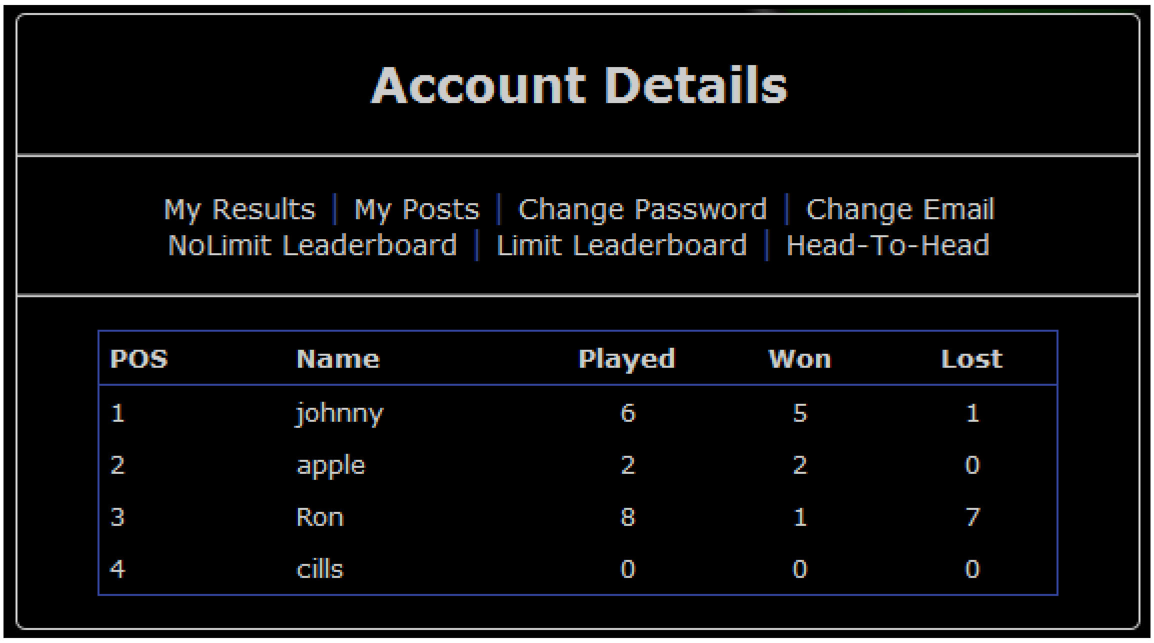
The testing of the application helped to iron out some bugs, (e.g.) the pot's TextView not updating to the correct amount, messages failing to send in the Bluetooth connection, player avatars becoming distorted because they weren't resized properly, All-In button causing opponents stack to become a minus figure etc. In the main, added logic fixed these bugs and the creation of helper classes, such as ResizeImage.java, MyDialog.java, UserDetail.java, MyHttpClient.java to name a few.

12.7. GAME WEBSITE

The game's website is located at http://androidlocalholdem.com. It was developed to:

- allow users access to their own account through a login system, where they can update their details, view leaderboards, and view their results against other users
- give new users a facility to register an account and download the game
- display the limit and no limit leaderboard tables to users in an aesthetically pleasing fashion
- give users and visitors, to the site, an opportunity to post comments, suggestions, and feedback
- advertise the existence of Local Hold'em

The website's most important feature is the persistence of the user's results. At the conclusion of a competitive game on the phone, the user who hosted (server) the game is asked to upload the result to website. This data is saved to the two player's accounts and the correct leaderboard (Figure 12.9) is updated. The users can then login to their account on the website and see the how this data has updated the leaderboard and their own statistics.



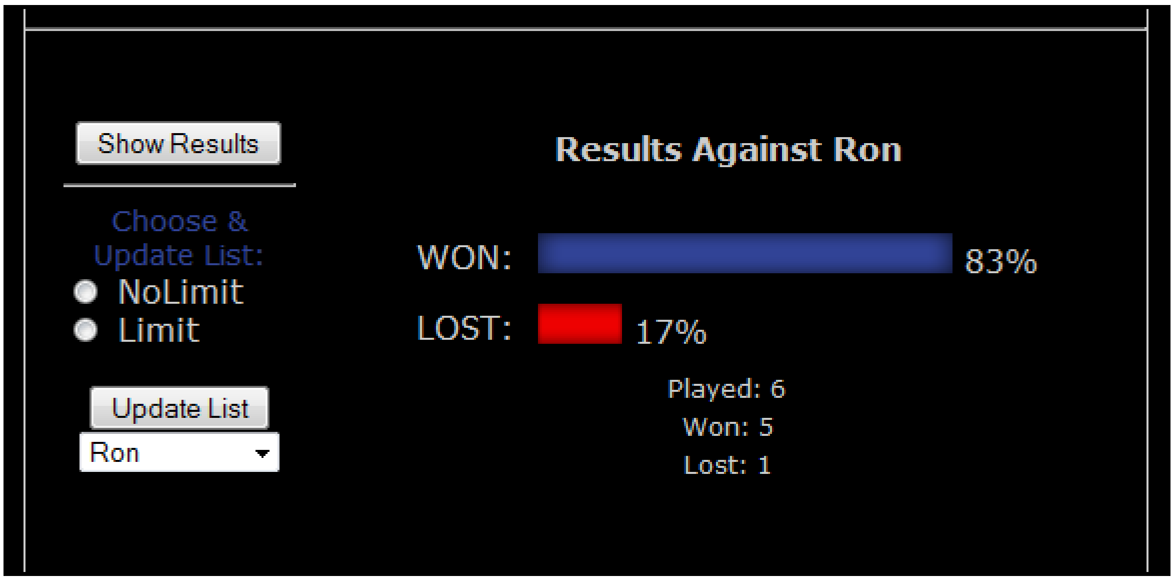
Account Details

My Results | My Posts | Change Password | Change Email
NoLimit Leaderboard | Limit Leaderboard | Head-To-Head

POS	Name	Played	Won	Lost
1	johnny	6	5	1
2	apple	2	2	0
3	Ron	8	1	7
4	cills	0	0	0

Figure 12.10. Website leaderboard and user account options

In the user's account, he can also view his results against other competitors in either Limit or No Limit hold'em. The user can choose one of the radio buttons, Limit or No Limit, and click the Update List. This populates the drop-down box with the users he has played in that format of the game. From there, the user can select one of his competitors from the drop-down box and click the Show Results button which will output the head-to-head results between him and the chosen player. The logged in users won and lost percentages against this competitor are displayed in a horizontal bar graph and the number of games played, won, and lost are displayed underneath. This is illustrated in Figure 12.10.



Played: 6
Won: 5
Lost: 1

Figure 12.11. Head-to-head results example

13. POST DEVELOPMENT ANALYSIS

13.1. CHALLENGES

The development of a project of this nature requires a variety of skills (e.g.) graphics, sound, animation, databases, networking, JAVA, JDBC, JSP, JSP Expression Language, and XML programming, mobile development etc.), and would only be undertaken by a group or team of skilled professionals in the commercial world. The learning curve and workload for a student, with a Multimedia background, was incredibly steep to say the least. Trying to comprehend and implement the JAVA and XML programming languages in the Android environment was daunting at first but with the completed project working successfully and boosting over fourteen thousand (14,000) lines of code (including the website), it has been concluded that these programming skills could be used effectively in future projects.

The networking features of this project were the most difficult to execute. The Android Documentation (2011), Reto Meier's, "Professional Android 2 Application Development" (2010), and Sayed Hashimi et al., "Pro Android 2" (2010), were regularly referenced along with forums and web searches. All of these were vitally important in facilitating the features required for the game. Although Bluetooth connectivity was demonstrated before the first iteration of the project began, this was merely a skeleton version of the final connectivity classes that are now required by the project. The package covering the Bluetooth connectivity within the project folder structure is called poker.holdem.bt and involves almost a thousand lines of code (covered in Section 12.4).

13.2. FUTURE ITERATIONS

Due to the time constraints and some problems during the implementation process, some planned features of the game were pushed to future iterations. If the development of this game continues at a later date, the work packages and workflow has been created for Iteration 4 and 5 below.

13.2.1. ITERATION 4

LIST OF WORK PACKAGES AND WORK FLOWS

1. Iteration 4
- **1.1.** Implement Bluetooth game for more than two players
- **1.2.** Test: Complete a Bluetooth game
  - **1.2.1.** Game of Limit and No Limit Hold'em with more than two players
  - **1.2.2.** Bug fixing
- **1.3.** Implement wireless ad-hoc network connection
- **1.4.** Test: Complete an ad hoc wireless game
  - **1.4.1.** Two player game of Limit and No Limit Hold'em
  - **1.4.2.** Bug fixing

13.2.2. ITERATION 5

LIST OF WORK PACKAGES AND WORK FLOWS

1. Iteration 5
- **1.1.** Implement ad hoc wireless game for more than two players
- **1.2.** Test: Complete an ad hoc wireless game
  - **1.2.1.** Game of Limit and No Limit with more than two players
  - **1.2.2.** Bug fixing

14. CONCLUSION

14.1. EXTERNAL REVIEWS

A version of Local Hold'em was made available for download from the game's website (see Figure 12.10) so that feedback could be acquired. The availability of Local Hold'em was advertised on a new thread and to interested users on various forums, such as androidforum.com, anddev.org, androidcommunity.com, talkandroid.com (thread removed because advertising application), and forums.gentoo.org (banned from forum for advertising game). The most feedback came from talkandroid.com users but this thread was removed for violating one of their rules regarding advertising non-market applications. Even though the thread was removed, some of the users that interacted with the thread downloaded the game and voted on the poll provided at http://androidlocalholdem.com/feedback.jsp. The result, as seen in Figure 14.1, is positive with 80% of users liking Local Hold'em.

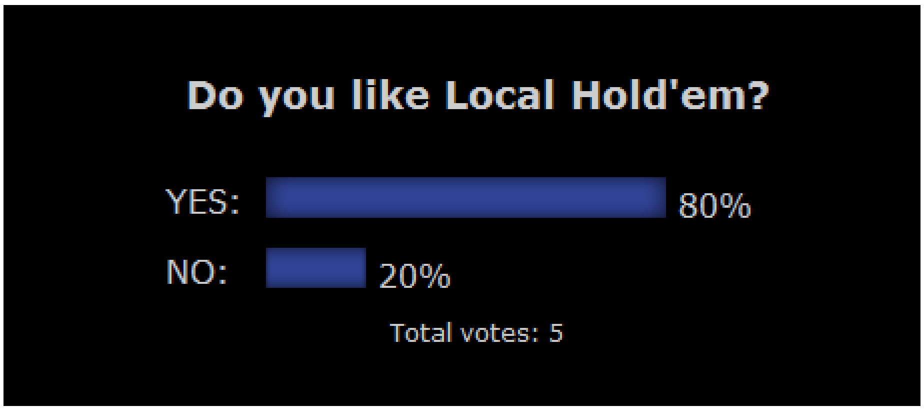
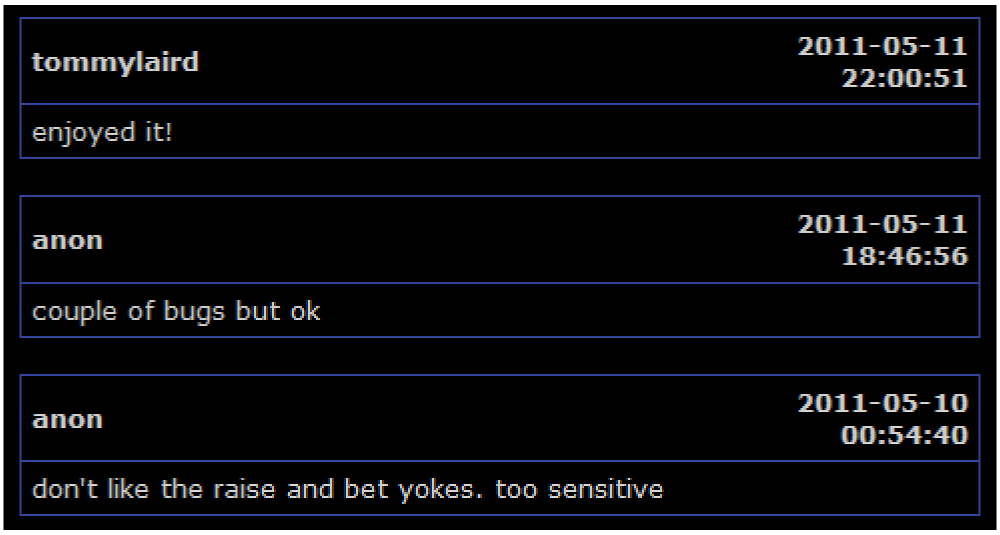


Figure 14.1. Local Hold'em Poll Result

A facility to post feedback was also made available below the poll on the feedback page of the website. Again, the majority of the feedback was positive but also highlighted what some users didn't like about the game. This information was very useful for tweaking some of the applications features. Some of the user feedback is displayed in Figure 14.2.



tommylaird 2011-05-11 22:00:51

enjoyed it!

anon 2011-05-11 18:46:56

couple of bugs but ok

anon 2011-05-10 00:54:40

don't like the raise and bet yokes. too sensitive

Figure 14.2. Sample of user feedback

14.2. SUMMARY

On completion of the project, it was concluded that overall it was a success. This can be confirmed by the number of goals met and personal development that has taken place from the project incarnation to its final conclusion.

As stated in the analysis and design document, the main aim of the project was to develop a fully functional and completely free-to-play local multiplayer Texas Hold'em Poker game that would run on the Android OS. This has been achieved and built upon with added functionality. The final application offers two forms of Texas Hold'em poker – Limit and No Limit which are free-to-play and use a local connection – Bluetooth. Local Hold'em also, saves the users, their details, and offers the user an interactive feature with regard to their result (Head-To-Head option in user's account, Figure 12.11).

The project success can also be qualified by the output of a fully functional game and website for Local Hold'em. The overall project size exceeded fourteen thousand (14,000) lines of code which is truly enormous and an achievement in itself, but to gain a market ready product from this effort is the real triumph. A project of this size and diversity would only be undertaken by a group or team of skilled professionals in the commercial world.

The personal development aspect of the project can also be declared a success. The in depth use of JAVA, XML, SQL, JSP, the JSP Expression Language, Servlets, JDBC, Javascript, and CSS have greatly developed and increased my skill set. The knowledge and experience acquired from developing this project will benefit me to no end in a commercial environment. My confidence to undertake a project of this size again has also been boosted significantly and will stand to me in future projects in the workplace and on a personal basis.

Local Hold'em has achieved its main project aim and expanded upon it with extra functionality and its development has led to the enormous personal growth and development of the personal involved.

Android Developer's Documentation (2011). "User Interface". Android Developer's Documentation. Retrieved on March 21, 2011 from http://developer.android.com/guide/topics/ui/index.html

AndroLib (2010). Retrieved on November 14, 2010 from http://www.androlib.com/

Bray, T. (2010). "More Countries, More buyers, More sellers". Android Developers Blog. Retrieved on October 14, 2010 from http://android-developers.blogspot.com/2010/09/more-countries-more-sellers-more-buyers.html

Clark, B. (2006). "The Dying Days of Las Vegas 1-5 Stud". Two Plus Two Internet Magazine. Retrieved on September 20, 2010 from http://web.archive.org/web/20061123021244/http://www.twoplustwo.com/magazine/issue21/clark0906.html

FlopTurnRiver (2003). Retrieved on November 9, 2010 from http://www.flopturnriver.com/

Google Code (2008). "Issue 82: wifi - support ad hoc networking". Android – An Open Handset Alliance Project. Retrieved on November 12, 2010 from http://code.google.com/p/android/issues/detail?id=82&colspec=ID%20Type%20Status%20Owner%20Summary%20Stars

Hashimi, S., Komatineni, S., and MacLean, D. (2010). Pro Android 2. Apress Inc., New York.

Jradi, R.K. and Reedtz, L.S. (2010). "Ad-hoc network on Android". B.Sc. degree thesis, Technical University of Denmark.

Man, M. (2002). "Bluetooth And Wi-Fi". Socket Communications Inc. White Paper, Socket Communications Inc., Central Court Newark, California.

Meier, R. (2010). Professional Android 2 Application Development. Wiley Publishing, Inc., Indianapolis.

Stigter, O. (2009). Texas Hold'em in Java. Retrieved on November 7, 2010 from http://code.google.com/p/texasholdem-java/

16. APPENDICES

APPENDIX A

XML CODE

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       >

       <TextView
         android:id="@+id/potStr"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="15dip"
            android:text="Pot:0"
            android:layout_alignParentLeft="true"
         />

   <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
       android:orientation="vertical"
       android:layout_width="60dp"
       android:layout_height="wrap_content"
       android:background="@drawable/my_border"
       android:layout_marginLeft="105dp"
       android:id="@+id/clientInfo"
       >
      <TextView
         android:id="@+id/clientName"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:textSize="15dip"
          android:layout_centerHorizontal="true"
          android:maxLines="1"
          android:text=""
       />
       <ImageView
         android:id="@+id/clientImage"
         android:background="@drawable/smoking_baby"
         android:layout_width="40dp"
            android:layout_height="40dp"
            android:layout_centerHorizontal="true"
            android:layout_below="@id/clientName"
      />
       <TextView
         android:id="@+id/clientStack"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="15dip"
            android:layout_below="@id/clientImage"
            android:layout_centerHorizontal="true"
            android:text="1"
         />
         </RelativeLayout>

         <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
          android:orientation="horizontal"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_marginLeft="160dp"
          android:layout_marginTop="15dp"
          android:id="@+id/clientHandView"
          android:visibility="visible"
       >
         <ImageView
            android:id="@+id/clientCard1"
            android:background="@drawable/card_holder"
            android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:layout_marginTop="5dp"
            />
          <ImageView
             android:id="@+id/clientCard2"
             android:background="@drawable/card_holder"
             android:layout_marginLeft="25dp"
             android:layout_width="wrap_content"
               android:layout_height="wrap_content"
            />
      </RelativeLayout>
      <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
          android:orientation="horizontal"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_marginLeft="160dp"
          android:layout_marginTop="15dp"
          android:id="@+id/clientHandViewBack"
          android:visibility="gone"
       >
         <ImageView
            android:id="@+id/clientCardB1"
            android:background="@drawable/card_back"
            android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:layout_marginTop="5dp"
            />
          <ImageView
             android:id="@+id/clientCardB2"
             android:background="@drawable/card_back"
             android:layout_marginLeft="25dp"
             android:layout_width="wrap_content"
               android:layout_height="wrap_content"
            />
      </RelativeLayout>
   </RelativeLayout>
```

APPENDIX B

JAVA CODE

```
private void raiseDialog() {
      raiseDlg = new Dialog(this);
      raiseDlg.requestWindowFeature(Window.FEATURE_NO_TITLE);
      // allow touch outside dialog to cancel
      raiseDlg.setCanceledOnTouchOutside(true);
      raiseDlg.getWindow().getAttributes().x =220;
      LayoutInflater li = (LayoutInflater) Context.getSystemService
                              (Context.LAYOUT_INFLATER_SERVICE);
      // inflate raise dialog box view from xml file
      View view = li.inflate(R.layout.raise_dialog, null, false);
      raiseDlg.setContentView(view);
      listRaise =(ListView)raiseDlg.findViewById(R.id.listViewRaise);
        // list of allowed raise amounts for user
        ArrayList<Integer> listTODO = new ArrayList<Integer>();
         // mType is how the users are differentiated, assigned at
         //beginning of game
         if (mType==0){
            for(int x = raiseListAmt; x <= serverP.getStack(); x++)
               if(x%10 == 0)
                  listTODO.add(x);
         }
         else{
            for(int x = raiseListAmt; x <= clientP.getStack(); x++)
               if(x%10 == 0)
                  listTODO.add(x);
         }

         // convert to array so raise dialog can handle display
         lv_arr = (Integer[]) listTODO.toArray(new Integer[0]);
        listRaise.setAdapter(new ArrayAdapter<Integer>
               (this, android.R.layout.simple_list_item_1, lv_arr));

         // catch the amount selected by the user
        OnItemClickListener listener = new OnItemClickListener (){
         public void onItemClick(AdapterView<?> arg0,
                     android.view.View arg1, int arg2, long arg3) {
            Object choice =  listRaise.getItemAtPosition(arg2);
            raiseAmt = Integer.parseInt(choice.toString());
            if (mType == 0){
               // display player's turn
               Toast.makeText(self, clientP.getName() + " turn",
               0).show();
               // update player stack and display
               serverP.setStack(serverP.getStack() - raiseAmt);
               serverP.setStackTV(serverP.getStack());
            }
            else {
               Toast.makeText(self, serverP.getName() + " turn",
               0).show();
               clientP.setStack(clientP.getStack() - raiseAmt);
               clientP.setStackTV(clientP.getStack());
            }

            // update pot and display
            pot += raiseAmt;
            potStr.setText("Pot:" + pot);
            raiseDlg.dismiss();
            // remove player options on turn end
            vA.vRow(false);
            // send raise amount to opponent's device
            mConnection.sendMessage(rivalDevice, "raise" + raiseAmt);
         }

      };
      listRaise.setOnItemClickListener(listener);
```

```
        // implement All-in button in dialog
        btnAllIn = (Button) raiseDlg.findViewById(R.id.btnAllIn);
        btnAllIn.setOnClickListener(new OnClickListener() {
           public void onClick(View view) {
              if (serverP.getStack() <= clientP.getStack()){
                 raiseAmt = serverP.getStack();
              }
              else {
                 raiseAmt = clientP.getStack();
              }

              if (mType == 0){
                 Toast.makeText(self, clientP.getName() + " turn",
                 0).show();
                 serverP.setStack(serverP.getStack() - raiseAmt);
                 serverP.setStackTV(serverP.getStack());
              }
              else {
                 Toast.makeText(self, serverP.getName() + " turn",
                 0).show();
                 clientP.setStack(clientP.getStack() - raiseAmt);
                 clientP.setStackTV(clientP.getStack());
              }

              pot += raiseAmt;
              potStr.setText("Pot:" + pot);
              raiseDlg.dismiss();
              vA.vRow(false);
              mConnection.sendMessage(rivalDevice, "raise" + raiseAmt);
              }
          });

        // implement Cancel button in dialog
        btnCancel = (Button) raiseDlg.findViewById(R.id.btnCancel);
        btnCancel.setOnClickListener(new OnClickListener() {
           public void onClick(View view) {
              raiseDlg.dismiss();
              }
          });
        raiseDlg.show();
      }
```

XML CODE

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >
   <Button
      android:id="@+id/btnAllIn"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
      android:textColor="#F00F00"
      android:text="All In"
      />
```

```
    <ListView
      android:id="@+id/listViewRaise"
      android:orientation="vertical"
      android:layout_width="60dp"
      android:layout_height="225dp"
      android:layout_gravity="center_horizontal"
      />
    <Button
      android:id="@+id/btnCancel"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
      android:text="Cancel"
      />
</LinearLayout>
```

APPENDIX C

JAVA CODE

```
// method that starts the server device who receives BT connections
// from opponent device
public int startServer(final int maxConnections,
OnIncomingConnectionListener oicListener,
            OnMaxConnectionsReachedListener omcrListener,
OnMessageReceivedListener omrListener,
            OnConnectionLostListener oclListener) {
        if (!mStarted) {
            return Connection.FAILURE;
        }
        if (maxConnections > MAX_SUPPORTED) {
            Log.e(TAG, "The maximum number of allowed connections is
" + MAX_SUPPORTED);
            return Connection.FAILURE;
        }
        mOnIncomingConnectionListener = oicListener;
        mOnMaxConnectionsReachedListener = omcrListener;
        mOnMessageReceivedListener = omrListener;
        mOnConnectionLostListener = oclListener;
        try {
            int result = mIconnection.startServer(mPackageName,
maxConnections);
            mIconnection.registerCallback(mPackageName, mIccb);
            return result;
        } catch (RemoteException e) {
            Log.e(TAG, "RemoteException in startServer", e);
        }
        return Connection.FAILURE;
    }

    // method that connects BT devices
    public int connect(String device, OnMessageReceivedListener
omrListener,
            OnConnectionLostListener oclListener) {
        if (!mStarted) {
            return Connection.FAILURE;
        }
        mOnMessageReceivedListener = omrListener;
        mOnConnectionLostListener = oclListener;
        try {
            int result = mIconnection.connect(mPackageName, device);
            mIconnection.registerCallback(mPackageName, mIccb);
            return result;
        } catch (RemoteException e) {
            Log.e(TAG, "RemoteException in connect", e);
        }
        return Connection.FAILURE;
    }

    // method that sends BT messages
    public int sendMessage(String device, String message) {
        if (!mStarted) {
            return Connection.FAILURE;
        }
        try {
            return mIconnection.sendMessage(mPackageName, device,
message);
        } catch (RemoteException e) {
            Log.e(TAG, "RemoteException in sendMessage", e);
        }
        return Connection.FAILURE;
    }
```

APPENDIX D

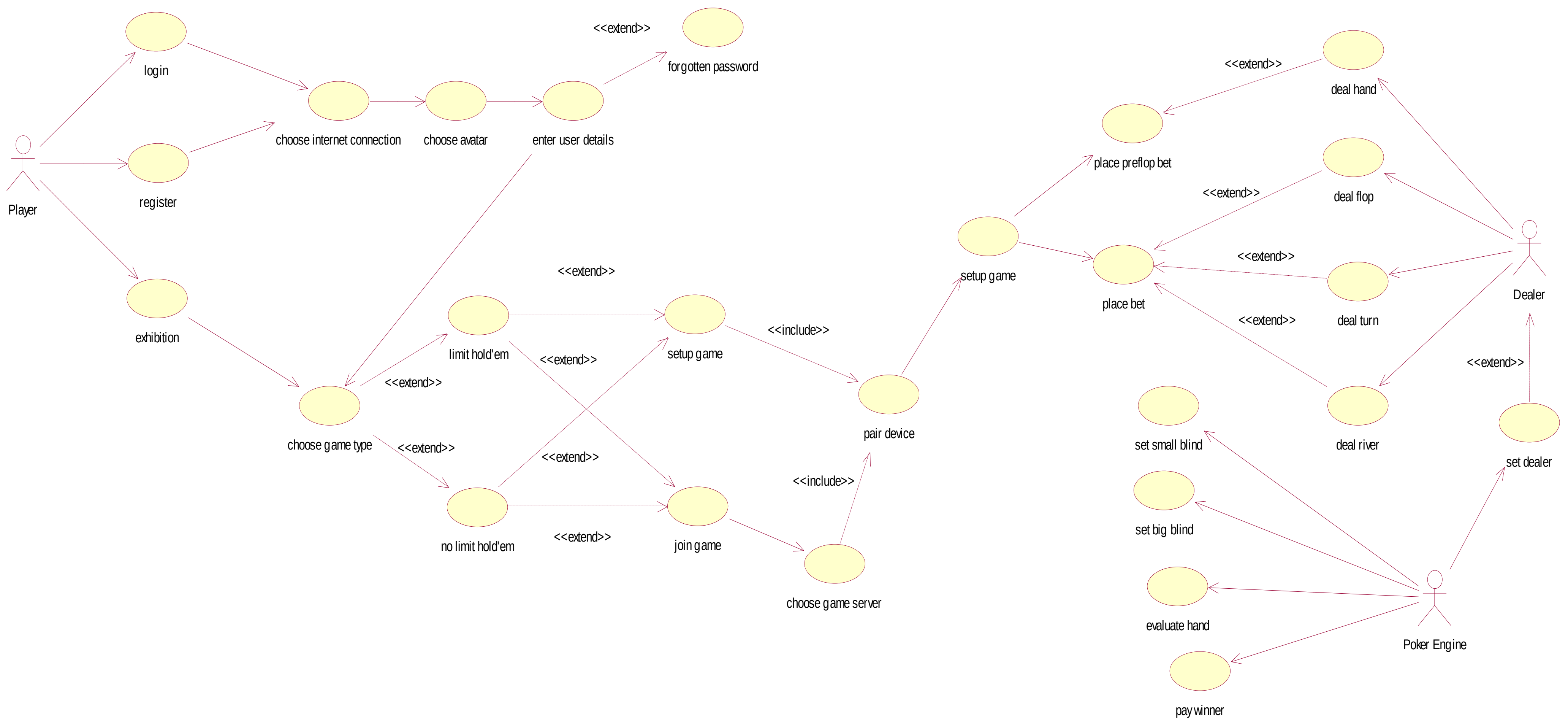
JAVA CODE

```
// method called at start of game to specify server and client user
private OnConnectionServiceReadyListener serviceReadyListener = new
OnConnectionServiceReadyListener() {
        public void OnConnectionServiceReady() {
            // mType == 0 is the server
            // - the user who started the game
            if (mType == 0) {
                mConnection.startServer(1, connectedListener,
maxConnectionsListener,
                        dataReceivedListener, disconnectedListener);
                // test to see if the device was in discoverable mode
                self.setTitle("Local Hold'em: " +
mConnection.getName() + "-" + mConnection.getAddress());
               // progress dialog displayed with message
               // waiting for user to connect
                myDialogMessage = "Wait for player to connect!";
                pd = new ProgressDialog(self);
                pd.setMessage(myDialogMessage);
                pd.show();
            }
            // the user that joined the game is presented
            // with the available devices to connect to
            else {
                Intent serverListIntent = new Intent(self,
ServerListActivity.class);
                startActivityForResult(serverListIntent,
SERVER_LIST_RESULT_CODE);
                // send user name to opponent (server)
                mConnection.sendMessage(rivalDevice,
"cRival"+clientP.getName());
```

```
                }
        }
};
```

APPENDIX E

Figure 16.1. Use Case Diagram for Local Hold'em Poker on the Android OS


Player
login
register
exhibition
choose internet connection
choose avatar
enter user details
<<extend>>
forgotten password
choose game type
<<extend>>
<<extend>>
limit hold'em
no limit hold'em
<<extend>>
<<extend>>
<<extend>>
<<extend>>
setup game
join game
<<include>>
<<include>>
pair device
choose game server
setup game
place preflop bet
place bet
<<extend>>
<<extend>>
<<extend>>
<<extend>>
deal hand
deal flop
deal turn
deal river
Dealer
<<extend>>
set dealer
set small blind
set big blind
evaluate hand
pay winner
Poker Engine