

Student: Jonathan Hawkins  
Candidate Number: 83384

Evolving AI bots to play Poker

Computer Science and Artificial Intelligence  
Informatics

Supervisor: Paul Gough  
Year: 2006

## **Statement of Originality**

This report is submitted as part requirement for the degree of Computer Science and Artificial Intelligence at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Signed.

Jonathan Hawkins

<b><u>1.0 - INTRODUCTION.....</u></b>	<b><u>6</u></b>
<b><u>1.1 - POKER.....</u></b>	<b><u>6</u></b>
<b><u>1.2 - AI AND GAMES.....</u></b>	<b><u>6</u></b>
<b><u>1.3 - AI AND POKER.....</u></b>	<b><u>7</u></b>
<b><u>1.4 - AIMS.....</u></b>	<b><u>7</u></b>
<b><u>1.5 - STRUCTURE.....</u></b>	<b><u>8</u></b>
<b><u>2.0 - PROFESSIONAL CONSIDERATIONS.....</u></b>	<b><u>9</u></b>
<b><u>3.0 BACKGROUND RESEARCH.....</u></b>	<b><u>9</u></b>
<b><u>3.1 - TURBO TEXAS HOLD'EM (TTH).....</u></b>	<b><u>10</u></b>
<b><u>3.2 - UNIVERSITY OF ALBERTA.....</u></b>	<b><u>10</u></b>
<b><u>3.3 - LIMITED TEXAS HOLD'EM POKER RULES.....</u></b>	<b><u>10</u></b>
<b><u>3.4 - WAYS OF PLAYING POKER.....</u></b>	<b><u>11</u></b>
<b><u>3.4.1 - EVALUATING HANDS.....</u></b>	<b><u>11</u></b>
<b><u>3.4.2 - WORKING THE ODDS.....</u></b>	<b><u>11</u></b>
<b><u>3.4.3 - WORKING THE OTHER PLAYERS.....</u></b>	<b><u>12</u></b>
<b><u>3.4.4 - THE BLUFF.....</u></b>	<b><u>13</u></b>
<b><u>3.5 - RED QUEEN EFFECT.....</u></b>	<b><u>13</u></b>
<b><u>4.0 - REQUIREMENT SPECIFICATIONS.....</u></b>	<b><u>13</u></b>
<b><u>4.1 - WHAT THE PROGRAM WILL ACHIEVE.....</u></b>	<b><u>13</u></b>
<b><u>4.1.1 - WEBSITE GOALS.....</u></b>	<b><u>13</u></b>
<b><u>4.1.2 - APPLET GOALS.....</u></b>	<b><u>14</u></b>
<b><u>4.1.3 - SERVER GOALS.....</u></b>	<b><u>15</u></b>
<b><u>4.2 - CHOICE OF TECHNOLOGIES.....</u></b>	<b><u>15</u></b>
<b><u>5.0 - DESIGN.....</u></b>	<b><u>16</u></b>
<b><u>5.1 - WEBSITE DESIGN.....</u></b>	<b><u>16</u></b>
<b><u>5.2 - DATABASE DESIGN.....</u></b>	<b><u>17</u></b>
<b><u>5.3 - APPLET DESIGN.....</u></b>	<b><u>17</u></b>
<b><u>5.3.1 - APPLET LOBBY DESIGN.....</u></b>	<b><u>18</u></b>
<b><u>5.3.2 - APPLET CREATE GAME ROOM DESIGN.....</u></b>	<b><u>18</u></b>
<b><u>5.3.3 - APPLET GAME ROOM DESIGN.....</u></b>	<b><u>18</u></b>
<b><u>5.4 - SERVER DESIGN.....</u></b>	<b><u>19</u></b>
<b><u>5.4.1 - LOGGING INTO THE SERVER.....</u></b>	<b><u>19</u></b>
<b><u>5.4.2 - CHATTING.....</u></b>	<b><u>20</u></b>
<b><u>5.4.3 - CREATION AND DELETION OF GAME ROOMS.....</u></b>	<b><u>20</u></b>
<b><u>5.4.4 - GAME STRUCTURE AND INTERACTION.....</u></b>	<b><u>20</u></b>
<b><u>5.4.5 - WORKING OUT WHO HAS WON.....</u></b>	<b><u>21</u></b>
<b><u>5.5 - AI PLAYER DESIGN.....</u></b>	<b><u>22</u></b>

<b>6.0 - IMPLEMENTATION</b>	<b>24</b>
<b>6.1 - WEBSITE IMPLEMENTATION</b>	<b>24</b>
<b>6.2 - IMPLEMENTATION OF APPLLET</b>	<b>24</b>
6.2.2 - LOBBY	25
6.2.3 - CREATE GAME ROOM	26
6.2.4 - GAME ROOM	27
<b>6.3 - IMPLEMENTATION OF SERVER</b>	<b>28</b>
6.3.1 - MESSAGING	28
6.3.2 - SERVER	29
6.3.3 - CONNECTION HANDLER	29
6.3.4 - MONITOR	30
6.3.5 - ROOM INTERFACE	31
6.3.5.1 - LOBBY	31
6.3.5.2 - GAME ROOM	32
6.3.6 - HAND CHECKER	34
<b>6.4 - IMPLEMENTATION OF AI PLAYER</b>	<b>35</b>
6.4.1 - AI CONTROLLER	35
6.4.2 - AI PLAYER	35
<b>7.0 - TESTING</b>	<b>36</b>
<b>8.0 - RESULTS</b>	<b>36</b>
<b>8.1 - AI PROGRESSION</b>	<b>37</b>
<b>9.0 - CONCLUSION</b>	<b>37</b>
<b>10.0 - REFERENCES</b>	<b>40</b>
<b>11.0 - APPENDICES</b>	<b>41</b>
<b>11.1 - FULL LIMITED TEXAS HOLD'EM POKER RULES</b>	<b>41</b>
<b>11.2 - LIST OF POKER TERMS USED</b>	<b>43</b>
<b>11.3 - COMPLETE SCORING SYSTEM</b>	<b>44</b>
<b>11.4 - FULL TESTING PROCESS</b>	<b>55</b>
<b>11.4.1 - WEBSITE TESTING</b>	<b>55</b>
<b>11.4.2 - STAND ALONE APPLLET TESTING</b>	<b>55</b>
11.4.2.1 - LOBBY	55
11.4.2.2 - CREATE GAME ROOM	55
11.4.2.3 - GAME ROOM	56
<b>11.4.3 - STAND ALONE SERVER TESTING</b>	<b>56</b>
11.4.3.1 - LOBBY	56
11.4.3.2 - GAME ROOM	57
<b>11.4.4 - COMBINED SERVER AND APPLLET TESTING</b>	<b>59</b>
<b>11.4.5 - USER TESTING</b>	<b>59</b>
<b>11.4.6 - AI TESTING</b>	<b>60</b>

11.5 - PROJECT CODE.....60

## Summary

The aim of this project was two fold:

- To build an online website that allows people to play Limited Texas Hold'em against others for nothing. There will be no money exchanged for playing the game. It is for fun and learning how to play.
- To evolve an AI player to be able to play the game. The ultimate goal would be people being able to play against these AI players and see if the computer can beat a human player.

This is no easy task to build a computer player that could compete fairly against a human in this game. Poker is a game of skill and chance with a huge amount of variation with different hands, and different ways of winning. Solutions that use decision trees to mark out every move are not practical in this environment due to the huge variations, search space and imperfect information.

So a more intelligent approach is needed. The lack of information means decision trees would not work, and many other techniques will struggle. A player does not know all the cards until a game has almost finished, and many hands finish before all the cards are shown, this puts a computer at a huge disadvantage.

For example Chess is a complex game and can take a long time to master. But at the heart of the game it uses calculations and works out your moves ahead of time. The information is all there for you and it is just a case of working out your best move and best strategy.

The Server was the first and largest development. Extensive testing was needed to iron out all bugs and ensure the rooms and games themselves acted correctly. Further more in the event that an error occurred, it was important that corrective action followed promptly.

Both the display and the feel of the client's Applet required a focused approach to achieve a satisfactory result.

The AI was not produced to the desired level, it was found to be a very hard process and the workload was just too great a challenge in the time given. Very basic players could play against one another or against humans but there was insufficient opportunity for evolution.

## 1.0 - Introduction

The motivation for this project was the desire to gain more knowledge about poker, to build an environment to play more often and not lose a lot of money, and also to try and build an AI agent that could beat me.

### 1.1 - Poker

Poker is a game that has been played for hundreds of years in many different versions. Like many other games it has also found its way onto the Internet now, which has only increased its popularity. It is a card game of both skill and chance, usually played for money.

The version of poker I based my work upon is Limited Texas Hold'em. There are four rounds of betting in total. Dealing out two Hole Cards to each player starts a game. These cards are dealt face down, as only that player should know what they have. The first round of betting happens at this point. After the betting has finished the first three *Community Chest* Cards are dealt out into the centre of the table. These cards are dealt face up, as they will be used by all the players in this hand, this is called the *Flop*. Another round of betting happens. Then the fourth *Community Chest* card is dealt out, the *Turn*. The third round of betting happens. The last card to be dealt out is the fifth *Community Chest* card, the *River*. The fourth and last round of betting happens, the winner(s) are found and the *pot* is given out.

This is one hand of poker. A game can be many hands of poker played one after the other. Texas Hold'em is a very common version of poker and is the one used to determine the world champion each year in Las Vegas. I used the Limited version of this game to simplify the betting process for users and the AI players.

Poker has many different strategies, varying from the reserved and cautious player, all the way through to the very aggressive player. If these abilities could be put into a bot and the correct strategy used against an opponent then they could stand a good chance against them.

### 1.2 - AI and Games

There have been various attempts within different game domains to build a computer program that could beat world-class human players. One of the more successful and well-known attempts was Deep Blue [4] a chess game that beat the world champion Garry Kasparov. Although this was a real achievement Deep Blue did not really use AI to beat its opponent.

Chess is a perfect information domain. This means that every possible move, action and reaction can be predicted, which leads to being able to figure out all possible moves to victory. Poker differs from this type of game domain, as it is an imperfect information domain. Each player has his own *hole cards* that only that player knows about. So the technique of looking forward to possible moves does not work well, as you cannot know what everybody has at the start of a game.

### **1.3 - AI And Poker**

There have been many attempts to build poker agents that could beat the masters of the game. Many of these have not been very successful. The University of Alberta Computer Poker Research Group [2] has been working for over a decade at this problem. They have built agents that can play at an intermediate to advanced level but none have reached world-class level. The two best agents they have at this time are Poki and PsOpti[3].

The key features that a strong agent would need to be able to obtain are [3]:

- Hand strength
- Hand potential
- Bluffing
- Unpredictability
- Opponent modelling

One of the main issues with getting an agent to the world-class level is it being able to understand and predict its opponent(s) and the ability to be unpredictable itself. This combination of attributes is very difficult to achieve.

### **1.4 - Aims**

The twin aims of this project were:

- To build an online poker house that people can come and learn to play poker against one another, and ultimately against AI opponents.

- To evolve agents to play on this Server and eventually gain agents that can compete against humans and hold their own. It was believed that the neural network will not be able to do all this work on its own so a number of hand crafted features need to be added to do some of the work for the network, and therefore to guide in the right direction.

There were a number of structures needed to achieve the goals stated:

- Website
- Applet
- Server
- AI agents

The website contains a forum for people to talk about the pros and cons of the poker site. It also acts as a place for registration for new users. The website is the way for users to log into the Server using the Applet.

The Applet is the way in which humans can play poker in the Server. The communication between the Server and Applet uses sockets. It also gives a way of watching the agents play against one another. The Applet is as much as possible a dumb terminal; this makes sure the Server does most of the work. This gives a number of benefits. No information is sent to the user that they should not see and all the information will stay accurate because it is centrally stored.

The Server is the hub of the site. It controls all chatting, games and Game Rooms. It also controls all users, human and AI.

The AI agents are evolved over many games playing against each other.

## **1.5 - Structure**

This project has two main parts, an online poker website and an Artificial Intelligence Agent that learns to play poker.

The website gives the users a way to register and play Limited Texas Hold'em for nothing. The author would like to state clearly that there is no intention to take any money from anyone or allow people to play for money. The site is designed for people to learn to play, and to play for fun with others.

The AI Agents are designed to learn to play the game against one another and if possible to compete against human players also. This will be done through neural networks.

The rest of the report will explain the professional considerations for this project, background research including other attempts at similar projects, rules of the game and considerations to be taking into account when making an AI agent.

Subsequently the requirements and design will be shown, leading to the testing and analysis of results.

Finally the conclusion will pull it all together and explain what happened and possible future improvements.

## **2.0 - Professional Considerations**

There were a number of ethical issues that come with any website which involves gambling. When any data is taken from a user data protection comes into consideration. The website does not take a lot of information from of the user. The user is required to enter a name, which does not have to be their real name; an email address (which needs to be confirmed) and a location that is not a full address and also is optional.

Credit card details are not taken as the users only play with “toy” credits. All this information is saved into a MySQL database on a secure Server so it is not freely available to others. The user would have to log in using a username and encrypted password to view or change any of this information.

A further issue is that gambling should not be available for people under the age of 18. The site has clear warnings that gambling should not be undertaken by anyone under the age of 18. I used a pre build application called PHPBB which is a PHP forum application that checks the user’s age is 13 or over. This was modified to only allow over 18’s to register.

Lastly there is an option to chat in whichever room you are in. So not allowing children to enter the site would eliminate the situation where anyone can talk to them without supervision, which could lead to undesirable results.

## **3.0 Background Research**

A number of applications have been created in relation to this project, some to do with poker environments and some to do with poker agents. Examples are detailed below.

### **3.1 – Turbo Texas Hold'em (TTH)**

Turbo Texas Hold'em is a program designed to improve a person's skill at playing poker with out the need of other people. It has some AI players built into it and has a lot of analytical functions, which enable it to it to tell you the possible moves.

No programming skill is required for this application as it does a lot of the work for you. It allows for a large amount of customization for a lot of different situations that can arise

With this environment I could create and evolve my agents but it does not allow for an online presence, which is part of my goal.

### **3.2 – University Of Alberta**

As previously stated the Canadian University of Alberta [2] has been working for over 10 years to develop a world-class poker agent. Games are often used to develop AI agents because they can often simulate real life situations. In doing so, a game can offer a structure that can show the progress of an agent.

They have a number of agents:

- Vexbot
- Sparbot

Vexbot was their latest agent that uses object modelling which is currently considered the way in which an agent could compete at world class level.

Sparbot was designed to work in heads-up (two player) poker.

Their products have been incorporated into TTH which is available to purchase if required.

### **3.3 – Limited Texas Hold'em Poker Rules**

The basic rules of Limited Texas Hold'em Poker are as follows:

1. The dealer is assigned.
2. Two hole cards are dealt to each player face down
3. Small and Large Blinds are placed on the table
4. The first round of betting happens

5. The first three Community Chest cards are dealt out
6. The second round of betting happens.
7. The fourth Community Chest card is dealt
8. The third round of betting happens
9. The fifth Community Chest card is dealt
10. The fourth and last round of betting happens.

For a fuller explanation of the rules please see the appendix 11.1.

### **3.4 – Ways Of Playing Poker**

There are a number of ways to play poker. Some people have the strategy of being an aggressive player to make people believe they have a better hand than they may have. Some will play the cautious type, making only small bets, if at all, when they feel they have a strong hand.

A lot of the strategy falls into four categories:

- Evaluating the hands
- Working out the odds
- Working out the other players.
- The Bluff.

#### **3.4.1 – Evaluating Hands**

The possible search space for Texas Hold'em is indeed huge. For example, if there were only two players there would be in the region of 1.2 quintillion nodes[6]. Cutting it down with dominated sequence hands (folding to no bet) there are still 1,097,690,218,045,566,601 nodes.

With this in mind the options are very varied for the players. The options open to each player are assessed by reviewing the *hole cards* and the *community cards* they hold. A player needs to think about what their possible hands are, and, what other possible hands could beat their hand. A player may only have a straight but it could be the best possible hand that could be made from all the cards available, so they should bet everything they can. However, that same hand could be beaten by many other hands with only a one or two card difference.

#### **3.4.2 – Working The Odds**

When you understand what hands can be formed from the *hole cards* and the *community chest cards* you can look at the odds. This falls into two main sections :

- The short game
- The longer game

The short game is looking at what you have when the *hole cards* have been dealt. Having a pair of picture cards would be considered to be very valuable where as say a jack and an eight would be worth little. If the player had a strong hand at this point they would bet heavily where as if they had a weaker hand they may very well fold, or at least try and get through to the flop with out betting a large amount.

Whereas some one who looks further ahead may see that a jack and an eight could lead to a straight and wait it out for the flop. Altematively, the player could wait to see if any pairs came out.

The short game is a more cautious strategy that says do not risk too much money if you do not think you can win. Conversely, the longer game says give a bit and see if you can get those extra cards that you need.

For either strategy the player needs to think about the odds of a certain hand coming up. The better the hand type the more the odds drop.

The odds on getting a pair are approximately: 0.422

The odds on getting a flush are approximately: 0.00197

The odds on getting a four of a kind are approximately: 0.00024

The odds on getting the higher cards quickly reduce. This should have a bearing on when and how much a player bets.

### 3.4.3 – Working The Other Players

When playing in the same room as the other players, there are many signs some people can read. The problem with being online is you do not get to see them so you cannot read their faces. Trying to figure out what they are thinking can be quite difficult without the physical presence to give clues.

If you know a certain player just does not bluff and you think they could have a hand that would beat yours, then it maybe a situation in which to fold. The patterns of how people bet can also be a guide to what sort of hand they have. This is where opponent modelling comes into its own. It is taking into account the history of a players actions and outcomes to try and figure out what sort of hand they maybe holding at that moment.

### **3.4.4 - The Bluff**

The bluff can be a great weapon in a player's arsenal. Sometimes a player can win with very poor hand by making everyone fear they have something far stronger than them. This can backfire at times but it also builds in unpredictability. A player does not always play the same way, for example they may not always fold when their hand is weak

### **3.5 - Red Queen Effect**

The Red Queen effect [7] states that a species that is evolving will need to keep moving to just seem like its standing still. For example the predictor prey theory. If the foxes are after the rabbits and the foxes learn how to run faster the rabbits will need to improve their speed and or agility to stay alive. i.e. they need to develop to just seem like they are at the same level of the fox. This is an issue that will need to be considered when evolving the poker agents.

## **4.0 - Requirement Specifications**

The whole project has a number of requirements, which split into different areas again with different technology needs. These are explained below.

### **4.1 - What The Program Will Achieve**

The program breaks down into four main goals:

- A website for the users to:
  - Chat
  - Play poker
  - See their own progress of themselves and that of others
- The Server to control all interaction from the users and games
- The Applet for users to be able to play poker
- Produce an AI poker player.

#### **4.1.1 - Website Goals**

The website is the interface for users to register, communicate with one another and the administrators of the system. The website allows the player to log onto the poker Server and play the game. It also lists all the users and how well they have played so far.

To register the user needs to confirm that they are 18 years of age or older. This is to deal with the gambling issue and to make sure no one is underage. They need to provide a username and real email address. This is to confirm there is a real person behind the login and that password changes are not given out to others. All other details provided are optional.

Communication within the website is done through a message forum. General chat and questions about the site are allowed to be posted. Anyone, registered or not, will be able to post within the public area. However, within the member's only area, only users who have registered and confirmed their email address can post a message.

A script will be used to log in on the Server, which will then redirect them to the Applet which in turn connects to the Server.

#### **4.1.2 - Applet Goals**

The Applet has a number of tasks to perform for the user. It allows users to instant message one another. This will only happen within each room, i.e. someone in the Lobby will not be able to chat with someone in a Game Room. No private messaging is allowed on the site to try to restrict people cheating and telling others what cards they have.

When a user logs into the Applet, the first screen they will see will be the Lobby. This area tells them about all Game Rooms open at that time, the minimum and maximum bets allowed for that room and the number of players sat at the table. It gives the user the ability to join these rooms and to view or play in them. It also gives them the option to create their own Game Room.

The Game Room is the heart of the Applet. It shows the table and all playing on it, the betting process as it happens and what cards are on the table. Only the *community chest cards* will be visible to all players and visitors, the *hole cards* will only be visible to that player.

Instant messaging will be allowed in all rooms apart from when a user is creating a Game Room.

The Applet is, as much as possible, a dumb terminal making the Server do all the work. This will mean different interfaces could be used with the Server with minimal change.

### **4.1.3 – Server Goals**

The Server is the central point. It controls all interactions. It takes all requests from the clients, processes them and informs all clients needed of the updates.

The Server will hold all rooms which in turn hold all users in that room. All chatting will need to be directed to the individual room.

The Lobby needs to know about all Game Rooms being created or destroyed, and, when some one joins or leaves the table.

The Game Room needs to keep track of all people in the room, all people sat at the table and how the game is playing at that time. This is where the body of the work takes place, making sure the game flow always goes correctly. The Game Room also must make sure the correct player makes the correct move in the right order. Within Limited poker there are basically four options open to a player; check, fold, bet or raise.

## **4.2 – Choice Of Technologies**

There are a number of technologies needed to complete this project. For the website HTML, PHP, MySQL, and CSS are the main four forms used. Other forms of Server side scripting are available but PHP is free, readily available on the Servers to be used and there is far more familiarity with this language.

The forum used was a pre build application PHPBB[8]. It uses PHP and MySQL. A few customizations were required to work fully with this project.

To build the Server and Applet it is thought that using the same language would reduce errors and confusion from the interactions of different languages. Java was chosen for the following reasons:

- It is very well known
- It is free to develop
- Both the Server and client models and all users can get it for free.
- Also an Applet could be used to run on the client's browser without any security issues or download of programs being required.

The Server could have been a Servlet or a standalone java application. The decision was to use a standalone application and have it running on the computer at all times due to the form of communication decided upon. The Server was chosen to use socket communications to the Applets rather than

http requests as used with Servlets. As the communication would need to be bidirectional at all times, and it was felt that Servlets did not support this very well.

## **5.0 – Design**

The project breaks down into a number of sections that to a reasonable degree can be tackled alone. They will all need to interact by the end of the project but the initial work does not need a lot of interaction.

### **5.1 – Website Design**

The website is the front end of the project. The presentation of the website therefore is of vital importance. It should do its best to be appealing to all, to be easily accessible for all and to have a clear navigation system.

The site breaks down into five sections:

- Home page
- Rules
- Rankings
- Forum
- Play game

The home page is a general greeting and an explanation of what the site is about.

The Rules page is a full explanation of the Rules of the game and each type of card hand available within the game.

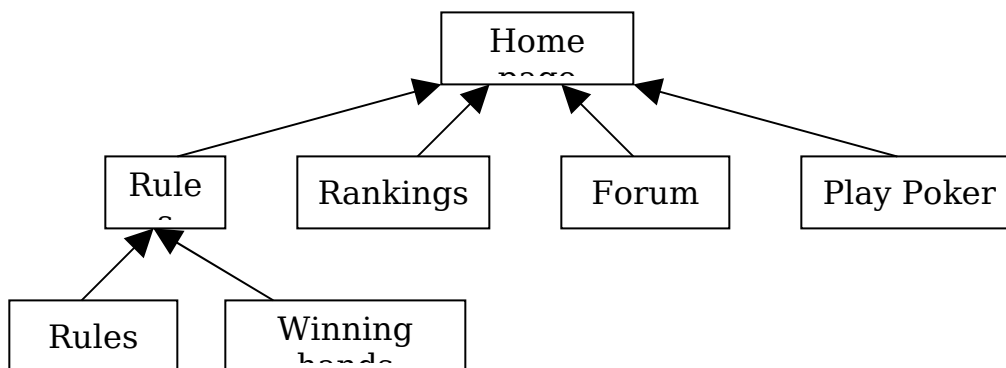
The Forum is a pre build application from PHPBB[8] that required a little customization to make it work within this project. Due to the nature of the site, i.e. gambling, it is felt that all users should be 18 years of age or older in order not to fall foul of the gambling laws that may apply. Real money is not exchanged on this site, as simply toy credits only are used. The forum originally asked for the user to declare if they were under 13 or over 13. Either way they were allowed to register, this had to be changed to only allow over 18's to sign up.

Within the FAQ in the forum helpful comments were added to assist with any queries users may have.

The last alteration was to the database in the background, which is explained below.

To play the game the user needs to log in using their current forum username and password. This is done through a PHP script, which checks the username and password against what has been submitted. If correct, it displays the users personal details and gives a button to enter the Applet.

An overview of the website structure is noted below:



## 5.2 - Database Design

The resulting database design using the PHPBB forum was very simple. The forum creates its own database. This was used with the forum to control nearly all the interaction within the database. The only additions required were:

- Current Credits
- Total Credits
- Games Played
- Hands Played

These were added to the “users” table and updated through a PHP script.

## 5.3 - Applet Design

The Applet needs to deal with a number of areas:

- Log into the Server
- Pass in the clients information
- Enter the user to the Lobby

- Allow chatting between users which ever room they are in
- Allow Game Rooms to be created by a user
- Allow users to enter Game Rooms
- Allow users to play games.

### 5.3.1 – Applet Lobby Design

The Lobby is the first area a user will see when they log into the Applet. It needs to tell the user the current state of the Server:

- All Game Rooms open at that time
  - Their minimum and maximum bets
  - The number of people sat at the table
  - The maximum number of people allowed to sit at the table
- The users current credits
- Any chatting that is taking place since the user joined the Server.

### 5.3.2 – Applet Create Game Room Design

This is a small part that only needs to allow the user to create Game Rooms. It will need to know:

- Name of room
- Minimum bet
- Maximum bet
- Number of players allowed to play

This will need to be sent to the Server as a request to create the room.

### 5.3.3 – Applet Game Room Design

The Game Room is the nuts and bolts of the Applet. It will deal with the game in action, display all moves made and tell the user what they can and cannot do.

A user can join a room without sitting at the table. This means they can watch a game in action, view the chat in that room but not need to deal with cards or betting. They will not be able to see anyone's *hole cards* as these are private to each user. However they will be able to see who is playing, what bets they all make, what the *community chest cards* are, who wins and how much.

The user can request to join the table when a seat is free, to leave the table if already sat at table and leave the room to return to the Lobby.

When a user is in a game the Applet will need to display all moves made by others and the possible moves they can make. The Applet will be, as much as possible a dumb terminal, this will allow central control from the Server and users can use a number of different interfaces.

## **5.4 - Server Design**

The Server is the hub of the application. It will control all aspects of the game, rooms and chatting.

When the Server is created it needs to create the Lobby so that new users can join it when they login.

A monitor is needed to ensure that all the user's connections are not lost and are also dealt with correctly. The monitor is required to check that a client has not logged off, lost its internet connection or closed the browser without performing a logging function. This makes sure all users online are truly there. For example if someone was in the middle of a game and their connection drops, and this was not detected, then the game would hang forever waiting for that user to respond. The monitor should be started with the Server and new users added to when they join. It can also be used to check for inactivity. Taking the previous example, should a user just never respond rather than loose their connection, the game will also hang forever. If a timer is kept for each user then this can also be checked periodically.

The Server needs to limit the number of connections allowed to it at any one time to try and minimize the risk of a Denial Of Service attack (DOS). This does not stop people misusing this service but will aim to limit the potential DOS effect.

### **5.4.1 - Logging Into The Server**

When a new user tries to create a connection to the Server the user needs to send its personal details:

- Name
- Current Credits
- Total Credits

- Games Played
- Hands Played

They need to be added to the monitor. Anyone in the Lobby needs to be informed that the number of people online has changed. Then it is a case of listening on both sides for updates, from the user and from the Server.

### 5.4.2 – Chatting

Chatting is a form of messaging. When a user sends a message to the Server this message needs to be sent to all users in that room.

### 5.4.3 – Creation and Deletion of Game Rooms

When a Game Room is created it should be stored in the Server for later access and control.

The Game Room will act in a similar manner to the Lobby. Chatting will be sent to the users of the room. In addition, there will be the ability to be in the game in this room.

When the last person leaves a room it will be destroyed, all references from the Server will be removed. This keeps everything tidy and will mean the Server should not need to be reset from time to time due to good housekeeping.

On creation and deletion, the users in the Lobby will need be informed of the new information.

### 5.4.4 – Game Structure And Interaction

Games need to hold a lot of information. They need to know the current pot, all players in the game, the minimum and maximum betting options, all the cards given to the players, as well as the *community chest cards*.

A game cannot start, or continue if there are less than two players sat at the table with enough money to play in that hand. There is a need to know which player is the dealer and which player is next in the betting sequence. This is done by seat number, which does not change from hand to hand even if people join or leave the table. As long as that player stays at the table the number will stay the same.

The Game Room needs to deal with people wanting to join and leave the table at any point. A player in the game can request to politely sit out of future hands but continue this hand until the end. They could also just jump straight back to the Lobby leaving a game half way through. This would result in the loss of all the money invested in that hand.

Each round of betting starts left of the dealer and only finishes when either each player has folded or met the largest bet or raises made. This has to happen in the correct order.

If at any point all but one of the players folds, then that player automatically wins that hand and all the money in the pot. If not it continues until all four rounds of betting have concluded. Then it is a case of figuring out who has won that hand.

Once a hand has finished the money needs to be given to the winner(s). All information is reset a new hand can start.

#### **5.4.5 – Working Out Who Has Won**

To work out which hand has won is not completely simple. There are nine different types of hands and each can have from fourteen to one hundred and fifty seven different options in each type. As previously stated this results in an extremely large search space. The different hand types in order of strength from smallest to strongest are:

1. Single card
2. One pair
3. Two pair
4. Three of a kind
5. Straight
6. Flush
7. Full house
8. Four of a kind
9. Royal flush

Each hand needs to be scored, from lowest hand in each type to the highest hand in the best type. For example a single two would score the very lowest where as ace high Royal Flush would score the very best.

This scoring does give a version of probability. The higher the score the less chance there is of getting this hand. This is not perfect mathematical probability but does mean each hand has a unique score, which can then be compared to other hands to work out who has won.

To do this each hand type needs to have an initial score. Then each possible hand within that hand type needs to be ranked.

There are a few situations where there could be a draw between two or more players. In these situations the pot would need to be split between the winners. A draw can be formed if two or more people have the same hand and have used five cards to make the hand [9]. If this happens the pot is split whatever kicker cards the players may have. This is still the case even if all five cards are from the *community chest*.

A kicker card is used to decide if it is a draw or not when two or more players have the same hand but do not use five cards to make their hand. The kicker is the spare card(s) that are not used to form the hand. The player with the higher kicker wins the hand. If the highest kicker is the same then test the second highest kicker, and so on until all kickers are tested. If they are all the same then it is a draw [9].

This has two functions. It will allow the Server to decide who has won a hand so the pot can be distributed and a new game started. The second function is to allow the AI agent to work out their options. It will return an array of all hands actually possible within this deal, this is further explained below.

## **5.5 - AI Player Design**

It was desired to have an AI player that could play against humans or other AI players. To achieve this it was decided to use a combination of a neural net (NN) and some custom functions.

There are two parts to the AI player design. The first is the player itself, the second is the AI controller.

The controller will create the players, create Game Rooms, tell the players which room they should be in and start all games. This will then be able to see who wins and loses each game and be able to rank them to be able to evolve the better bots.

A lot of information could be embedded into the AI player. However it was decided at outset that a basic platform would be used. From that point it could be developed given sufficient time. The initial premise was that the AI player should only play its own game and not consider what the other players are doing or have done or the potential plays they may make.

The information that would be entered into a separate NN input node would be:

- A total of each hand type
- The number of wild cards used to gain those hands

To produce these results the bot class will need to know the *community chest* cards that are present at that time and its own hole cards. To get the answers it will use the Hand Checker class that has been implemented in the Server.

The bot will need to connect directly to the Server and deal with the messages sent from the Server. A lot of these can be ignored. The main ones it will need to deal with are:

- Games starting and finishing
- The players credit level changing
- Community chest card information
- Hole card information
- Betting options given to that player
- Being removed from the table

It will need to make its own decision of when to bet, check / fold, call or raise. To do this the NN will be run with the current information, returning a value for its output. This output has two thresholds applied to it; a betting threshold and a raising threshold. If the value is above the raise then raise. If it is above the bet then bet, or else fold or check depending on the options given.

A possible extension could be to consider what other hands their opponents could have. To do this it could work out what hands everyone else could have excluding this hole cards of that player and pipe these into the NN in the same way as done previously.

There could be a random number generated to see if they bluff. For example if the random number generator is greater than the bluff threshold the player will raise all the time anyway.

The controller will create a population of these bots and set them off playing against each other. The order in which they loose gives a ranking system. Another and possibly more affective system could be giving each bot far more money than they can possibly use in five minutes of playing and see how much each player ends up with. This means all bots can be compared against each other rather than just against the ones they played with

## **6.0 - Implementation**

### **6.1 - Website Implementation**

The website was a small part of the implementation. The forum was a pre built application that required only a little customization. Most pages were static HTML pages and one PHP page to display the rankings of all players.

### **6.2 - Implementation Of Applet**

The Applet has the following functions:

- To read in the user's information from the browser.
- To load in the image of each card for the Game Room.
- To create reflective list of commands Server can call
- To create the GUI
- To set up connection to Server and pass in user details.

To achieve these goals a number of items were needed. The user's information is simply a string for their name and five integers for the other values. The cards are read into vectors for each suit and an image object for the back of a card.

It was decided that using reflection would cut down some of the code for messaging and make it more intuitive. When a message is received from the Server it has a type, for example a chat message has the header "CHAT". This type can be used to reference a method of the same name. To avoid incorrect use of this system and method, the message is prefixed with "Server" when it gets to the client. For example the chat message becomes "ServerCHAT" with a method of the same name. So if someone tried to send a message saying "ServerCHAT," it would be processed by the client as "ServerServerCHAT".

The methods that could be called from the Server are stored in a hash table with the name of the method as the key. This allows the method to be called when a message is received, as long as the message conforms to the method naming protocols. If any information was passed in with the message from the

Server this is put into a substring, cutting off the message header. For example a message received was "CHAT hey there" it would be sent to the chat method with the parameter string of "hey there".

The GUI has a number of components:

- Title
- Main window
- Footer

The title and footer are images loaded in which never change and therefore do not need to be tracked. The main window will change depending on what room the user is in. This window is comprised of a jpanel and a number of other panels inside it. This outer panel needs to be stored so it can be redrawn when a room changes. Each room's structure will be explained below.

Having started the Applet, the connection to the Server completes the process. All communication is done through sockets, which allow communication to go both ways. A Servlet could have been used with HTTP communication but this does not allow for the client new information to be sent out to the client unless it keeps polling the Server for it. This seems inefficient and could over load the Server with an excess of requests.

To be able to both listen and send information to the Server another thread is set up to listen for incoming messages from the Server. Both the reader from the Server and the writer need to be stored as does the separate thread so it can be closed safely later on. An error is thrown up to the user if the connection is refused to the Applet.

### **6.2.2 – Lobby**

The Lobby has the following functions:

- Tell user about the site
- Tell user about their credits
- Tell user about the current rooms available
- Display and allow chatting
- Give access to create Game Room
- Give access to Game Rooms.

The Lobby in essence is just a jpanel with other panels inside it like the main Applet. A label with text tells the user about the site. The credit information for each user is also in label form.

The current room information is more complicated. It has its own abstract table model (ATM) which holds all the information about each room. It has a vector to hold all the rooms, with each room holding its own vector. All the information about the rooms are simply strings that get pulled out and displayed. There are accessor methods to allow addition and removal of rooms. The addition works in a similar way to a hash table. If the name of the room already exists then it is over written instead of a new room added to the vector.

This ATM is used by a jTable, which actually displays the rooms to the screen. This means to update the screen the abstract table model needs to be updated and the jTable just needs to be refreshed on the screen. To ensure that no room updates are missed the ATM is stored in the Applet and a reference is passed into the Lobby.

To display chat messages received from the Server, the chat method would be called on the current room the user is in, and the message would be appended to a JTextArea. There is a quirk with the text area. If the text is appended to the bottom and the scroll-pane is needed, it does not automatically scroll to the bottom. Therefore it needs to be forced by working out where the end is and setting the force there.

To allow a user to chat to others a JTextField is used with a listener on the return key. When this is pressed it takes the sentence entered. If it is not null, it is sent to the Server with a message header of "CHAT" followed by the sentence. This will be received back from the Server as well as everyone else receiving it, who is in the Lobby.

To send the user to another room it needs to request that the Applet replace the JPanel that the Lobby is loaded in and to load the new window.

If it is just loading the create Game Room window then the request is sent and existing JPanel is removed. The new one is created and stored in its place and the screen is refreshed for the user.

If the user is trying to enter a Game Room they must have first selected a game from the jTable. If this has not happened then the button to open that room is left disabled. If a room has been selected the name is retrieved from it and a message is sent to the Server to join that room. If the request is successful the user is told they have now entered a Game Room and the JPanel is replaced with the new game window.

### **6.2.3 – Create Game Room**

The create Game Room is very basic, it just deals with letting the user enter the information for a new room:

1. Name
2. Minimum bet
3. Maximum bet
4. Number of players at table.

The Game Room validates this information, and, if it fits the profile allowed, it will then send the request to the Server to create a new room. The user is then sent back to the Lobby and if the room was created, it will be displayed in the jTable. It makes sure all the options have something entered. The name may not be more than 15 characters long, be called Lobby or have spaces in the name.

The minimum and maximum bet needs to be above zero and the maximum bet needs to be above the minimum. There must be at least two players and a maximum of ten. If any of these conditions fail, an error message is displayed to the user in a JLabel.

As previously stated when this request is sent to the Server the client is sent back to the Lobby view.

#### **6.2.4 – Game Room**

The Game Room has a number of functions:

- It allows users to chat
- It allows users to leave the room and return to the Lobby
- It allows users to request to sit in or out of the table
- It watches games play out
- It facilitates the playing of the game

The chatting is done in exactly the same way as in the Lobby. Messages received from the Server are displayed in a JTextArea and all chat from the user is sent directly to the Server.

A JButton is used to control the way a user can leave the room. A message is sent to the Server to tell it that the user is returning to the Lobby. This request will make sure that the user is cleanly removed from where they presently are. For example if they are in an active game it will make sure they fold and are removed. After this the Lobby's JPanel is reloaded.

To join or leave the table another JButton is used. A request is sent to the Server to join the table. Nothing happens apart from changing the button to display the opposite of before.

When a user does join or leave the table this information is sent to all users. On receipt, the information for that seat is updated on the GUI. If a player has joined the seat a name, their credit value and the back of two cards are displayed.

If a seat becomes free the display is set back to having no name, value or cards displayed.

When a game is in process all actions are sent to all the users in that room. When an action happens a message is displayed just like a chat message is displayed to tell the user what action has happened. When cards are dealt the publicly known cards are displayed and all pot and betting information is updated.

When it is a user's turn to bet they are given their options on buttons to either check / fold, bet or raise according to the value of the game status.

## **6.3 – Implementation Of Server**

The structure of the Server comprises:

- A central Server
- A connection handler for each user
- Rooms
- A monitor
- A hand Checker

When a new user tries to connect to the Server they go to the central Server. The whole communication system is designed to work over sockets using strings as the message.

### **6.3.1 – Messaging**

When a message comes into the Server, or client, it is read into a buffered reader and the string is tokenized. The prefix is added to the first word. The prefix will be "Server" or "client" depending on whether it is Server or client generated. This first word is checked against the list of methods created when the application was started. If there is a corresponding method then that method is invoked. If the tokenizing produced more than one word, it is put back into one string with spaces inserted between the words. This is then passed into the method being invoked.

This technique gives security over method invocation as only the methods intended to be called from outside can be used. As a result the code is neater because it decides it's self which method to call and does not grow with the more functionality added.

All objects are passed as strings. When a room's information is sent it is sent as a string. When someone's credit is sent it is sent as a string and will need to be converted back into an integer at the other end.

### **6.3.2 – Server**

The Server is the central point. It holds a number of areas for all users to interact with:

- The number of people presently online
- All rooms that have been created.
- The monitor.

The Server listens on a socket for new connections. It creates a new connection handler for each user and sets them off in their own thread.

It is where you can change the number of people online and inform the users, add and remove Game Rooms and tell the monitor when a user made a move.

To add a Game Room they are all stored in a hash table with the name of the room as the key. This means a room is easily accessible by using the name. Equally it is just as simple to remove a room using the name only.

The Server also holds the monitor class so when a player needs their timer to be reset, they make the call to the Server to deal with it.

### **6.3.3 – Connection Handler**

The connection handler is what represents the user on the Server. It takes in the requests from the client, processes them and takes the appropriate action. It also dishes out all relevant information to the user.

It is the controller of the user's credits, hand and game totals. When updates are sent to the user, the total is sent to make sure the user is always up to date at all times.

New hands have to be played for the game total to be increased. This stops the user joining a table then leaving to increase their games played.

Just as the Applet sets the up the input from the client and output to the client using sockets, so too does the connection handler. When the connection handler first starts it reads in the user's information, sets the current room to be the Lobby and allows the user join the room.

The connection handler uses reflection to control the messaging, as does the Applet. All messages have "client" prefixed to the command and the hash table is searched for the relevant method. If a command is not recognised a message is sent back to the user explaining the error.

If an exception is thrown, and an error occurs that cannot be dealt with cleanly, then the client is logged off. The logging off process makes sure the person leaves the room safely, this takes into account of a player halfway through a game and removes them safely. The Server is informed that a player has logged off, which in turn lets the other users know the player has left. All connections are closed and makes sure the method closes cleanly so the thread is terminated.

When a chat message is received it is sent to the current room to be displayed.

#### **6.3.4 – Monitor**

The monitor class has one main task, to monitor user's actions, which divides into two sub tasks:

- Monitoring any actions
- Monitoring game actions

To monitor these actions the monitor needs to know who made the action and the current time. Each action / user is stored in a hash table with the user as the key and the time. The time is the current time plus an increment, for any action. Ten minutes in milliseconds is allowed for general action and for game actions it is three minutes in milliseconds. Therefore each time a new action is taken by the user, the time is updated by replacing the old time in the hash table.

Two hash tables are required; one for the general actions and another for game actions.

The class sleeps for thirty seconds then checks through each element in both hash tables to see if the time has expired. If it has, then it is removed from the hash table. Depending on whether it is a game action or a general action, different actions are taken.

If it is any action, the user is sent a message to say they have been logged off completely and their connection is closed.

If it is a game action then the user is sent a message to say they are being removed from the game and sent back to the Lobby. Their credits used so far in that hand are given up.

### **6.3.5 – Room Interface**

There are two rooms involved in the game, the Lobby and all Game Rooms created. They both need to do a number of things:

- Return a string of the room's status
- Send a message to the whole room
- Allow a user to join a room
- Allow a user to leave a room

To make it easier to just have the current room stored, the interface is used to implement both room types.

#### **6.3.5.1 – Lobby**

The Lobby needs to hold a vector of all users in the room and a reference to the Server. To message a room, all users need to be messaged, this is done by iterating through the vector and sending each user the message.

The room status of the Lobby is just its name.

When a player joins the Lobby they need to know about all Game Rooms currently open. This is done by retrieving all rooms from the Server and sending the room status to the user, as well as from the Lobby information. They also need to know the number of people online. This again is achieved by retrieving that information from the Server.

Leaving the Lobby only requires removing that user from the vector.

### 6.3.5.2 – Game Room

The Game Room needs to do a number of tasks:

- Join the room
- Request a seat at the table
- Leave the table
- Leave the room
- Chat
- Betting in the game

The room will need to hold a lot of information:

- The players in the room
- The players sitting at the table
- The players still betting at that time
- A reference to the Server
- The maximum number of people allowed at the table
- Minimum and maximum bets allowed
- The deck of cards
- A list of people who have requested to sit in and leave the table.
- The current status of the hand

The room status of the Game Room is its name, the minimum and maximum bet and the current number of people playing.

When a room is first created a game is not in action; all the variables are initialized; the room waits for people to join it and request a seat.

When a player joins a room they are added to the vector of people in the room and a greeting message is sent to them. They are also sent out the status of any game currently in action and all people sitting at the table.

To join the table a player must request to sit down. This is due to a game being in action at that time. If a game is in progress then they get added to the list which waits until the current hand finishes.

The same is true for wanting to leave the table. A player can simply leave the table at that moment but they may want to finish that hand but not enter the next one so they request to leave. They get added to a vector, which gets processed at the end of each hand to remove anyone currently at the table.

When these two vectors are dealt with the new room's status is issued out to the Lobby. These both are run before each new hand starts. The request out is processed first followed by the request in.

To assign a player a seat the first available seat must be found. Each player holds their own seat number which will never change until they leave the table. All the players get sorted and the lowest number free returned.

To start a game the player must send a request. There must be at least two players to start a game. All players at a start of the game must have enough credits to play. If they request in with out enough credits they will not be let in. In the event that they lost a lot on the last hand, they will be removed from the table until they refresh their credits.

Each time a new game is started a new pack is created and shuffled randomly. A pack is a vector of strings, with the suit then type. For example five of spades is s5 where the ace of hearts is h14.

To initialize a game the players must be advised of:

- The pot size
- The bet size
- The start of a game
- The fact that all players at the table and their seat position
- The fact that the dealer has been set and the small and large blinds are sorted
- The next player has been found and their move options sent to them.

To find the next player, the Game Room simply looks at the mod of the current player and the number of people playing plus one. It then reviews the mod of preceding analysis and the number of people playing again:

```
(currentPlayer % bettingPlayers.size() +1) % bettingPlayers.size()
```

To find out what the players' next moves could be, the Game Room looks at the games status. If the last bet was zero then they can check or lay the minimum or maximum bet. If the last bet was greater than zero, and the total raises have not been met, then they can fold, call or raise, or they could just fold or call.

From this point inwards the system waits for the next player to respond. The monitor makes sure it does not hang here forever. A move is an integer:

- -1 - Fold
- 0 - Check
- >0 - Bet / Raise

If a player folds then they need to be removed from the vector of players in that hand.

If a player checks, there is the need to ensure that this is a valid move, i.e. that everyone else in this round of betting has checked

If it is above zero then they have either called the existing bet, or have raised. If there has been a raise then the counter needs to be incremented to make sure only three raises can be made in a single round of betting.

If the move received is valid, then the users credits need to be amended reflect this. Similarly the pot and the last bet need to be updated. Lastly the next player is found set. The room is told of the bet made, what the pot has now become and the minimum bet.

A check is then performed to see if the current round of betting has finished. It checks all players have bet the same amount and all players have made a bet. This ensures even if people check, everyone has to agree. If the round has finished then the players bets are reset as is the last bet, the round increased and next round is started. If not then tell the next player their moves.

If the game is finished, i.e. if there is only one player left in the game or there have been four rounds of betting, then the hand played for each player is incremented. The winner(s) are notified by the hand checker class. The pot is distributed. Finally all users requesting out and in are acknowledged and the next game starts.

### 6.3.6 – Hand Checker

It is necessary to find out who has won the utility and those hands with seven cards. This utility has been expanded to take into account wild cards so the bots may use it as well. It has been further expanded to take into account which cards to exclude from the search. This can be used to work out all possible hands you have and what hands other can have excluding your hole cards, i.e. two people cannot have four of a kind with the nine of spades unless they are all in the *community chest*.

To do this the cards are split into:

- Number of wild cards entered
- The card values, i.e. the number of fours
- And the number of cards in each suit

Each hand type is unique so a score is attached to it, which gives a ranking system. For a full list and the maths of it please see appendix 11.3. The base values that are used to work out the scores for each hand are worked out when the class is created. Each hand type is worked through with each possibility producing a list of hands available, with the score of each hand and the number of wild cards used. Each of these is put into a hand class to allow them to be sorted and accessed with ease.

There are a large number of hands available so to make things easier there must be at least one card entered otherwise it will just find every hand possible.

This can be used by the AI agents, as well as the Server, to work out the strength of its hands later on.

## **6.4 - Implementation Of AI Player**

The AI section breaks down into two parts, the AI player and the controller.

### **6.4.1 - AI Controller**

The controller divides into a number of sections:

- Creating its reflective method list
- Connecting to the Server
- Creating its bots

Just as the Applet and Server use reflection to build their list of methods that can be called by messages received the controller uses the same technique.

The Controller connects to the Server using sockets as does the bots, sending up the basic information for use within the Lobby. This means the Controller can create any rooms required and ensure things are working correctly.

Lastly the Controller needs to create a vector of bots to put into the newly created room and tell them to join that room and sit at the table.

### **6.4.2 - AI Player**

The AI player has a number of components:

- Its Neural Net ( NN )

- Threshold values:
  - Bet
  - Raise
- Reflective Command List
- Cards known about

The NN is composed of:

- Nine input nodes
- Weights from input to hidden layer
- Eight hidden nodes
- Weights from hidden to output layer
- One output node

The NN is a simple feed forward network so the input nodes are multiplied against the first set of weights then squashed with a sigmoid function and put into the hidden weights. This process is applied from the hidden to output layer to get a fine result from NN.

This value is then checked against the bet and raise threshold to find out what move has been decided upon.

The command list is built in the usual way using reflection to deal with messages from the Server.

The cards are held in simple strings in order that the AI player is able to make decisions.

## 7.0 – Testing

For a complete explanation of the testing that took place to ensure the applications worked as required please see appendix 11.4.

## 8.0 – Results

There have been mixed results within this project. The only aspect that fell below its goal was the AI component.

The website works well, it is a simple part of the project but meets all its goals except in one area. Due to needing root access to the web server that would be running the java server and web pages the author could not use the existing server. Unfortunately due to reliability issues with the new server this could not

be obtained. This meant that the website stayed on the existing server and the poker server and applet could not be connected to them at this time. The server and applet can be run individually and work correctly together, but have not been put online during this project. Therefore all testing was done on a local network.

The Server in most ways worked completely as required. A few bugs have been found that were not able to be fixed within the time allowed. These bugs have been explained in the testing within appendix 11.4.

One extension that was implemented was allowing unlimited poker as well as limited. This meant people could raise as strongly as they wished, or as weakly. This turned out to be a small extension as the code was designed to be able to be abstracted out to an abstract game class and then implement each type of game.

The benefit of making the server the controller is that any clients that connect to the server, act largely as a dumb terminal that only has to react to the options the server feeds to them.

The applet worked as required. The graphics were not always great. The card images were hard to read at times and some objects moved about in certain situations. The oddest bug found was the dealer chip was not always being shown, this bug was not able to be tracked down. This said it did work as required fitting all specifications and was received well by the user test group.

## **8.1 - AI Progression**

The AI was not as successful as the rest of the project. The AI bots were basically random, they did not go through a process of evolution. Whatever random values they were assigned at creation they kept and did not learn from playing the game.

That said they did play against each other and against a human opponent. They did communicate with the server correctly and the author was beaten a few times, but this was more due to luck than anything else.

## **9.0 - Conclusion**

Having brought the project to this point, the outcome has to be reviewed against the benchmark of the two key objectives i.e.

- The creation of a Poker House in which players can both learn to play poker and enjoy playing the game with other players
- Evolve AI agents that can learn to play poker

In both areas there were successes and opportunities for further development and improvement.

The Poker House works as a server was designed to allow people to learn to play the game. It can create an environment where up to ten people can play against each other. In addition a player can play against a primitive AI agent.

From a technical point of view the applet works as required and has a reasonably friendly and useable interface. However it was not possible to put the java server online but full testing was done over a local network and this worked as designed.

As in many projects hindsight can lead to seeing better, clearer and more efficient ways of dealing with a problem. This is also true for this project. Enumerated types could have been used to classify the cards and possibly the hand types as well. This would have meant that some sorting, classifying and dishing out to players would have been easier. As a result the volume of code could have been reduced.

An area that would benefit from further development is that of messaging. The messaging passes the information through the sockets. A check should be put in place to ensure that messages going to and from the server are being correctly delivered. For example, if it is a players go and they get their betting options sent to them but they never get this message, the game has to wait for that user to be kicked off for not reacting. Therefore that player may never be able to bid.

Acks could have been used to prevent this happening. However TCP is assumed to deal with this and make sure the messages arrive at their destination. This situation could have been resolved with another extension. All messages are sent through strings at this point. The string has a header saying what type of message it is then includes any variables if needed. A better way of dealing with this would be have a message object. That has:

- An id, to be used for Acks
- Header
- Variables

All of these can have accessor methods that allow it to be used in a cleaner and more transparent manor.

The project has fallen below its objectives in the AI area. The AI Agents were designed to learn to play the game against one another, and if possible to compete against human players also, using neural networks. A number of hand crafted features need to be added to do some of the work for the network. The key issue was the amount of time taken to remove the vast majority of the bugs which would have inhibited not only the interaction between the players but also the playing of games without errors.

In conclusion the author feels that the overall objective has been achieved in that an enjoyable and challenging game of poker can be enjoyed by players from any corner of the world without the need to be physically in the same room as the other players. Further challenges for the more experienced poker player can be created through the continued development of the Artificial Intelligence player.

## 10.0 - References

- [1] <http://www.msnbc.msn.com/id/6002298/> (URL valid as of 21/3/2006)
- [2] <http://www.cs.ualberta.ca/~games/poker/> (URL valid as of 21/3/2006)
- [3] <http://www.cs.ualberta.ca/~darse/Papers/schauenberg.msc.pdf> (URL valid as of 21/3/2006)
- [4] <http://www.research.ibm.com/deepblue/home/html/b.shtml> (URL valid as of 22/3/2006)
- [5] [http://www.holdemgenius.com/use\\_odds\\_calc\\_g.html?keyword=poker+strategies&g=1&gclid=CPnF3M\\_48oMCFStOEgod9H9cKA](http://www.holdemgenius.com/use_odds_calc_g.html?keyword=poker+strategies&g=1&gclid=CPnF3M_48oMCFStOEgod9H9cKA) (URL valid as of 22/3/2006)
- [6] <http://www.cs.ualberta.ca/~games/poker/FAQ.html#whatwhy> (URL valid as of 22/3/2006)
- [7] <http://pespmc1.vub.ac.be/REDQUEEN.html> (URL valid as of 22/3/2006)
- [8] <http://www.phpbb.com/> (URL valid as of 23/3/2006)
- [9] <http://www.texasholdem-poker.com/forum/phpbb/phpBB2/viewtopic.php?t=3459> (URL valid as of 29/3/2006)

## **11.0 – Appendices**

### **11.1 – Full Limited Texas Hold'em Poker Rules**

1. Before the game starts decide on the Minimum and Maximum bets allowed in the game.
2. Make the small blind and large blind bets.
3. Deal out 2 cards to each player face down.
4. Betting starts left of the big blind.
5. Burn the top card.
6. Deal out 3 community cards face up.
7. Betting starts again left of the dealer.
8. Burn the top card.
9. Deal out the 4th community card face up.
10. Betting starts again left of the dealer.
11. Burn the top card.
12. Deal out the 5th community card face up.
13. Betting starts again left of the dealer.
14. Display cards, starts left of the dealer and goes round in order.
15. Find winner and sort out the pot

#### **Detail of Each Stage**

1. A minimum and Maximum must be determined before the game starts to control how the betting goes for each round. This will control how much money can be lost or won in a single round, for example if there is no limit it could get very large very quickly. The minimum is needed for a couple of reasons, one so people know how much it will cost to play and another so we will know how much the blind bid will cost.
2. The small blind and large blind are done to make sure there is money in the pot for some one to win every time. On the basis that the 2 people left of the dealer will be the blinds, the first person will be the small blind. This means they have to lay half of the minimum bet. The second person is the big blind, which

means they have to lay the full value of the minimum bet. This may seem unfair but the dealer rotates round to the left so the blind will rotate around as well.

3. The Dealer always deals to the left of himself. The 2 cards that are dealt to each player are dealt face down so that no one knows what cards anyone has.

4. The betting happens 4 times through out the hand. The first time it happens it starts to the left of the big blind. There are a number of plays a person can make:

- Fold - you decide that you will not win so you fold. You throw away your cards and do not play anymore in that game. You lose the money that you have already placed as a bet.
- Check - you do not lay any money at that time but you are still in the game. - This can only happen if no one else has laid any money down before in this round.
- Bet - you need to see the bet as it is now. If no one has bet already it is the minimum bet that is put down. If some one has already raised then the bet will to be see their amount.
- Raise - To raise is to see the existing bet and to add the maximum bet to it. This means the new minimum that every one else has to see is the amount that player raised to.

There are some conditions of the betting. It will continue until every player has met the bet, or folded. By this I mean that if anyone raises, then everyone must meet that raise before the betting is finished. There can be only 3 raises in a single round of betting, so a round of betting cannot go on forever. If everyone folds or checks then no money is required to be entered in that round of betting.

5. The Burning of a card happens every time the dealer is going to play a card on the community card pile. The dealer takes the top card off the deck and puts it at the bottom. This is done to stop cheating.

6. There are some cards that are dealt out for everyone to use. These cards are called the "Community Cards" or "Community Chest". They are used to make up the players hands. They are not physically in each player's hand but they can use any of them to form the best 5 card hand.

14. Each player displays all of their cards saying what their best hand is to see if they may have won or not.

15. To find the winner of the game you look at the hand of every player who is still in the game, (i.e. they have not folded). Whoever has the best hand will win. If two or more players have the same hand you need to look at their other card, (the kicker) the higher one wins. If they are both the same then it's a draw between them. If the community cards have the best hand then the pot is divided equally between the people still in the game.

The dealer is then moved round one place to the left and the cards are shuffled and a new game starts.

## **11.2 – List Of Poker Terms Used**

*Community Chest* – the 5 cards placed in the centre of the table, which every player is able to use.

*Minimum bet* – The smallest bet that can be made at that time

*Maximum bet* – The largest raised bet that can be made at that time

*Pot* – The accumulative value of all the betting so far in that round

*Small blind* – half of the minimum bet.

*Large blind* – The minimum bet.

*Hole cards* – The two private cards each player is dealt at the start of each hand

*Flop* - When the first three community cards are dealt

*River* - When the fourth community card is dealt

*Turn* - When the fifth community card is dealt

*Dealer* – The person who is dealing the cards in this round. (This moves to the left after each hand).

*Call* – To equal the existing bet mad by the last player.

*Raise* – To greater the last bet.

### 11.3 – Complete Scoring System

increment	7.5	<b>lowest card</b>	<b>single</b>	<b>one pair</b>	<b>two pair</b>	<b>three of a kind</b>	<b>straight</b>	<b>flush</b>	<b>full house</b>	<b>four of a kind</b>	<b>full flush</b>
card	value		1	7.5	56.25	421.875	3164.0625	23730.469	177978.5	1334839	10011292
2	0.133	<b>2</b>	0.133333 3	1	56.25	421.875	3164.0625	23730.469	177978.5	1334839	10011292
3	0.200	<b>3</b>	0.266666 7	1.5	84.375	632.8125	4746.0938	266967.12	2002258	2669678	2002258
4	0.267	<b>4</b>	0.333333 7	2	112.5	843.75	6328.125	355957.5	3337098	3337098	3337098
5	0.333	<b>5</b>	0.466666 3	2.5	140.625	1054.6875	7910.1563	444946.3	4004517	4004517	4004517
6	0.400	<b>6</b>	0.533333 7	3	168.75	1265.625	9492.1875	533935.5	4671937	4671937	4671937
7	0.467	<b>7</b>	0.666666 7	3.5	196.875	1476.5625	11074.219	622924.8	5339356	5339356	5339356
8	0.533	<b>8</b>	0.733333 3	4	225	1687.5	12656.25	711914.1	6006775	6006775	6006775
9	0.600	<b>9</b>	0.866666 7	4.5	253.125	1898.4375	14238.281	800903.3	6674195	6674195	6674195
10	0.667	<b>10</b>	0.933333 7	5	281.25	2109.375	15820.313	889892.6	7341614	7341614	7341614
11	0.733	<b>11</b>	1.066666 3	5.5	309.375	2320.3125	17402.344	978881.8	8009034	8009034	8009034
12	0.800	<b>12</b>	1.266666 7	6	337.5	2531.25	18984.375	106787.1	8676453	8676453	8676453
13	0.867	<b>13</b>	1.466666 7	6.5	365.625	2742.1875	20566.406	115686.0	9343873	9343873	9343873
14	0.933	<b>14</b>	1.666666 3	7	393.75	2953.125	22148.4	124585.0	9343873	9343873	9343873

			3			5	38		0	2
15			1	0		1	1	0	1	0

<u>79</u>	two pair all			2pair 0.71202	<b>157</b>	full house all			full house
			0.0126582					0.00636	1133.62
3	2	1	28	5	2	3	1	9	1
			0.0253164	1.42405				0.01273	2267.24
4	2	2	56	1	2	4	2	9	2
			0.0379746	2.13607				0.01910	3400.86
4	3	3	84	6	2	5	3	8	3
			0.0506329	2.84810				0.02547	4534.48
5	2	4	11	1	2	6	4	8	4
			0.0632911	3.56012				0.03184	5668.10
5	3	5	39	7	2	7	5	7	6
			0.0759493	4.27215				0.03821	6801.72
5	4	6	67	2	2	8	6	7	7
			0.0886075	4.98417				0.04458	7935.34
6	2	7	95	7	2	9	7	6	8
			0.1012658	5.69620				0.05095	9068.96
6	3	8	23	3	2	10	8	5	9
			0.1139240	6.40822				0.05732	10202.5
6	4	9	51	8	2	11	9	5	9
			0.1265822	7.12025				0.06369	11336.2
6	5	0	78	3	2	12	10	4	1
			0.1392405	7.83227				0.07006	12469.8
7	2	1	06	8	2	13	11	4	3
			0.1518987	8.54430				0.07643	13603.4
7	3	2	34	4	2	14	12	3	5
			0.1645569	9.25632				0.08280	14737.0
7	4	3	62	9	3	2	13	3	7
			0.1772151	9.96835				0.08917	
7	5	4	9	4	3	4	14	2	15870.7
			0.1898734	10.6803				0.09554	17004.3
7	6	5	18	8	3	5	15	1	2

## Jonathan Hawkins

## Final Report

		1	0.2025316	11.3924				0.10191	18137.9
8	2	6	46	1	3	6	16	1	4
		1	0.2151898	12.1044					19271.5
8	3	7	73	3	3	7	17	0.10828	6
		1	0.2278481	12.8164					20405.1
8	4	8	01	6	3	8	18	0.11465	8
		1	0.2405063	13.5284				0.12101	
8	5	9	29	8	3	9	19	9	21538.8
		2	0.2531645	14.2405				0.12738	22672.4
8	6	0	57	1	3	10	20	9	2
		2	0.2658227	14.9525				0.13375	23806.0
8	7	1	85	3	3	11	21	8	4
		2	0.2784810	15.6645				0.14012	24939.6
9	2	2	13	6	3	12	22	7	6
		2	0.2911392	16.3765				0.14649	26073.2
9	3	3	41	8	3	13	23	7	9
		2	0.3037974	17.0886				0.15286	27206.9
9	4	4	68	1	3	14	24	6	1
		2	0.3164556	17.8006				0.15923	28340.5
9	5	5	96	3	4	2	25	6	3
		2	0.3291139	18.5126				0.16560	29474.1
9	6	6	24	6	4	3	26	5	5
		2	0.3417721	19.2246				0.17197	30607.7
9	7	7	52	8	4	5	27	5	7
		2	0.3544303	19.9367				0.17834	31741.3
9	8	8	8	1	4	6	28	4	9
		2	0.3670886	20.6487				0.18471	32875.0
10	2	9	08	3	4	7	29	3	1
		3	0.3797468	21.3607				0.19108	34008.6
10	3	0	35	6	4	8	30	3	3
		3	0.3924050	22.0727				0.19745	35142.2
10	4	1	63	8	4	9	31	2	5
		3	0.4050632	22.7848				0.20382	36275.8
10	5	2	91	1	4	10	32	2	8
		3	0.4177215	23.4968				0.21019	
10	6	3	19	4	4	11	33	1	37409.5
		3	0.4303797	24.2088				0.21656	38543.1
10	7	4	47	6	4	12	34	1	2

		3	0.4430379	24.9208						39676.7
10	8	5	75	9	4	13	35	0.22293	4	
		3	0.4556962	25.6329				0.22929	40810.3	
10	9	6	03	1	4	14	36	9	6	
		3	0.4683544	26.3449				0.23566	41943.9	
11	2	7	3	4	5	2	37	9	8	
		3	0.4810126	27.0569				0.24203		
11	3	8	58	6	5	3	38	8	43077.6	
		3	0.4936708	27.7689				0.24840	44211.2	
11	4	9	86	9	5	4	39	8	2	
		4	0.5063291	28.4810				0.25477	45344.8	
11	5	0	14	1	5	6	40	7	4	
		4	0.5189873	29.1930				0.26114	46478.4	
11	6	1	42	4	5	7	41	6	7	
		4	0.5316455	29.9050				0.26751	47612.0	
11	7	2	7	6	5	8	42	6	9	
		4	0.5443037	30.6170				0.27388	48745.7	
11	8	3	97	9	5	9	43	5	1	
		4	0.5569620	31.3291				0.28025	49879.3	
11	9	4	25	1	5	10	44	5	3	
		4	0.5696202	32.0411				0.28662	51012.9	
11	10	5	53	4	5	11	45	4	5	
		4	0.5822784	32.7531				0.29299	52146.5	
12	2	6	81	6	5	12	46	4	7	
		4	0.5949367	33.4651				0.29936	53280.1	
12	3	7	09	9	5	13	47	3	9	
		4	0.6075949	34.1772				0.30573	54413.8	
12	4	8	37	2	5	14	48	2	1	
		4	0.6202531	34.8892				0.31210	55547.4	
12	5	9	65	4	6	2	49	2	3	
		5	0.6329113	35.6012				0.31847	56681.0	
12	6	0	92	7	6	3	50	1	6	
		5	0.6455696	36.3132				0.32484	57814.6	
12	7	1	2	9	6	4	51	1	8	
		5	0.6582278	37.0253						
12	8	2	48	2	6	5	52	0.33121	58948.3	
		5	0.6708860	37.7373					60081.9	
12	9	3	76	4	6	7	53	0.33758	2	

		5	0.6835443	38.4493				0.34394	61215.5
12	10	4	04	7	6	8	54	9	4
		5	0.6962025	39.1613				0.35031	62349.1
12	11	5	32	9	6	9	55	8	6
		5	0.7088607	39.8734				0.35668	63482.7
13	2	6	59	2	6	10	56	8	8
		5	0.7215189	40.5854				0.36305	
13	3	7	87	4	6	11	57	7	64616.4
		5	0.7341772	41.2974				0.36942	65750.0
13	4	8	15	7	6	12	58	7	2
		5	0.7468354	42.0094				0.37579	66883.6
13	5	9	43	9	6	13	59	6	5
		6	0.7594936	42.7215				0.38216	68017.2
13	6	0	71	2	6	14	60	6	7
		6	0.7721518	43.4335				0.38853	69150.8
13	7	1	99	4	7	2	61	5	9
		6	0.7848101	44.1455				0.39490	70284.5
13	8	2	27	7	7	3	62	4	1
		6	0.7974683	44.8575				0.40127	71418.1
13	9	3	54	9	7	4	63	4	3
		6	0.8101265	45.5696				0.40764	72551.7
13	10	4	82	2	7	5	64	3	5
		6	0.8227848	46.2816				0.41401	73685.3
13	11	5	1	5	7	6	65	3	7
		6	0.8354430	46.9936				0.42038	74818.9
13	12	6	38	7	7	8	66	2	9
		6	0.8481012					0.42675	75952.6
14	2	7	66	47.7057	7	9	67	2	1
		6	0.8607594	48.4177				0.43312	77086.2
14	3	8	94	2	7	10	68	1	4
		6	0.8734177	49.1297					78219.8
14	4	9	22	5	7	11	69	0.43949	6
		7	0.8860759	49.8417					79353.4
14	5	0	49	7	7	12	70	0.44586	8
		7	0.8987341					0.45222	
14	6	1	77	50.5538	7	13	71	9	80487.1
		7	0.9113924	51.2658				0.45859	81620.7
14	7	2	05	2	7	14	72	9	2

		7	0.9240506	51.9778				0.46496	82754.3
14	8	3	33	5	8	2	73	8	4
		7	0.9367088	52.6898				0.47133	83887.9
14	9	4	61	7	8	3	74	8	6
		7	0.9493670					0.47770	85021.5
14	10	5	89	53.4019	8	4	75	7	8
		7	0.9620253	54.1139				0.48407	86155.2
14	11	6	16	2	8	5	76	6	1
		7	0.9746835	54.8259				0.49044	87288.8
14	12	7	44	5	8	6	77	6	3
		7	0.9873417	55.5379				0.49681	88422.4
14	13	8	72	7	8	7	78	5	5
								0.50318	89556.0
					8	9	79	5	7
					8	10	80	4	9
					8	11	81	4	1
					8	12	82	3	3
					8	13	83	2	5
					8	14	84	2	7
					9	2	85	1	96357.8
					9	3	86	1	97491.4
					9	4	87	0.55414	98625.0
					9	5	88	0.56051	99758.6
					9	6	89	0.56687	100892.
					9	7	90	9	102025.
					9	8	91	8	103159.
					9	8	91	8	5

			0.58598	104293.
9	10	92	7	1
			0.59235	105426.
9	11	93	7	8
			0.59872	106560.
9	12	94	6	4
			0.60509	
9	13	95	6	107694
			0.61146	108827.
9	14	96	5	6
			0.61783	109961.
10	2	97	4	2
			0.62420	111094.
10	3	98	4	9
			0.63057	112228.
10	4	99	3	5
			0.63694	113362.
10	5	100	3	1
			0.64331	114495.
10	6	101	2	7
			0.64968	115629.
10	7	102	2	4
			0.65605	
10	8	103	1	116763
				117896.
10	9	104	0.66242	6
				119030.
10	11	105	0.66879	2
			0.67515	120163.
10	12	106	9	8
			0.68152	121297.
10	13	107	9	5
			0.68789	122431.
10	14	108	8	1
			0.69426	123564.
11	2	109	8	7
			0.70063	124698.
11	3	110	7	3

			0.70700	125831.
11	4	111	6	9
			0.71337	126965.
11	5	112	6	6
			0.71974	128099.
11	6	113	5	2
			0.72611	129232.
11	7	114	5	8
			0.73248	130366.
11	8	115	4	4
			0.73885	
11	9	116	4	131500
			0.74522	132633.
11	10	117	3	7
			0.75159	133767.
11	12	118	2	3
			0.75796	134900.
11	13	119	2	9
			0.76433	136034.
11	14	120	1	5
			0.77070	137168.
12	2	121	1	2
				138301.
12	3	122	0.77707	8
			0.78343	139435.
12	4	123	9	4
			0.78980	
12	5	124	9	140569
			0.79617	141702.
12	6	125	8	6
			0.80254	142836.
12	7	126	8	3
			0.80891	143969.
12	8	127	7	9
			0.81528	145103.
12	9	128	7	5
			0.82165	146237.
12	10	129	6	1

			0.82802	147370.
12	11	130	5	7
			0.83439	148504.
12	13	131	5	4
			0.84076	
12	14	132	4	149638
			0.84713	150771.
13	2	133	4	6
			0.85350	151905.
13	3	134	3	2
			0.85987	153038.
13	4	135	3	9
			0.86624	154172.
13	5	136	2	5
			0.87261	155306.
13	6	137	1	1
			0.87898	156439.
13	7	138	1	7
				157573.
13	8	139	0.88535	3
13	9	140	0.89172	158707
			0.89808	159840.
13	10	141	9	6
			0.90445	160974.
13	11	142	9	2
			0.91082	162107.
13	12	143	8	8
			0.91719	163241.
13	14	144	7	4
			0.92356	164375.
13	2	145	7	1
			0.92993	165508.
14	3	146	6	7
			0.93630	166642.
14	4	147	6	3
			0.94267	167775.
14	5	148	5	9
14	6	149	0.94904	168909.

			5	5
			0.95541	170043.
14	7	150	4	2
			0.96178	171176.
14	8	151	3	8
			0.96815	172310.
14	9	152	3	4
			0.97452	
14	10	153	2	173444
			0.98089	174577.
14	11	154	2	7
			0.98726	175711.
14	12	155	1	3
			0.99363	176844.
14	13	156	1	9



## **11.4 – Full Testing Process**

As there are a number of different sections to this project the testing was broken down into various sections in order to verify that each unit worked individually as well as collectively.

### **11.4.1 – Website Testing**

The website itself required minimum testing. The areas tested ensured that that the links worked; the PHP pages loaded and displayed the information correctly; and that the forum was installed and customized correctly.

### **11.4.2 – Stand Alone Applet Testing**

The Applet was built separately to the Server initially. The first level of testing of the Applet was completed on a stand-alone basis. Further in depth consideration was given to how the Applet and the Server would communicate.

#### **11.4.2.1 – Lobby**

The Lobby was tested in a few ways:

- It was tested that the display looked correct.
- When a test button was pushed it added or removed Game Rooms from the jTable as required.
- When a text was typed into the chat box and return was hit it would be appended to the end of the chat window.
- When the refresh credits button was pushed the credits would be increased by the correct amount.

#### **11.4.2.2 – Create Game Room**

This room only required to be tested to make sure that valid rooms only could be created. Some of the requirements were:

- The name “Lobby” was not allowed.
- Room names could be up to twelve characters long
- The number of players must be at least two and a maximum of ten

- The minimum bet must be less than the maximum bet

### **11.4.2.3 – Game Room**

The Game Room required minimum testing to ensure that it displayed all information correctly. The key area of testing was that the display was updated when an update was sent from the Server.

### **11.4.3 – Stand Alone Server Testing**

The Server was tested with a very basic client connection run through dos. This meant all messages received from the Server are shown and all messages sent to the Server have to be written down so there is no chance of missing any information. This was done in conjunction with output prints from the Server if any confusion arose.

An ongoing test within each of the following sections was to see how errors were dealt with. The Server had to be able to rectify incorrect data having been entered, as well as restoring a user's lost connection or a browser being unexpectedly closed.

It was considered vital that the Server tidied up after itself. When a user logged off or was removed then all connections, sockets and information was removed. The same was true for each room. When all users have left a room that room would be removed from the Server and all clients informed.

#### **11.4.3.1 – Lobby**

Lobby needed to test chatting, to make sure whatever was said by one user, was sent out correctly to all users in that room and only that room.

In addition the Lobby was tested to ensure that it could create Game Rooms correctly; that the message sent from the user to the Server was correctly processed and also to see if the Game Room message was correctly sent to the Lobby.

The last test was to make sure that the number of people online was always updated when someone logged on or off from the Server.

### 11.4.3.2 – Game Room

The Game Room required the most testing of all. The author had to be sure that the game made all the correct moves in the correct order at the correct time.

Some errors found during this check were:

- If a player checks then next time makes the minimum bet it still considers it to be a raise.
- It did not change the person's go if it was that person quits out.
- It did not look at a person's kicker cards. It only looked at their main hand. If they had the same hand there would be a draw but a kicker could mean you win or lose.

Some of the checks made during this long process are as follows:

- When a bet was made out of sequence it did not do anything bad, only told that person it was not their go yet
- Check that only three raises can happen in one round of betting, and that this was reset on each round
- Check that once a player has been called they cannot re raise.
- Check that when a player leaves, if it is their go, the next player is told it is now their turn.
- Check when any player leaves they are removed from the betting and table and all users updated.

This testing process took a long time to make sure all bugs were removed before moving this process to the Applet.

A couple of examples of the game testing are as follows:

TEST RAISES - LIMITED 2 THREE AND CHECK THEN MIN BET IS NOT RAISE				
player	move	total		
jack1	hole cards			
jack2	hole cards			
jack3	hole cards			
jack2	small	2	2	
jack3	large	4	4	
jack1	bid -1 4 12	4	4	call
		1		
jack2	bid -1 2 10	0	12	raise
jack3	bid -1 8 16	8	12	call
jack1	bid -1 8 16	8	12	call

	com cards			
jack2	bid 0 4 8	0	0	check
jack3	bid 0 4 12	4	4	call
		1		
jack1	bid -1 4 12	2	12	raise
		1		
jack2	bid -1 12 20	2	12	call
		1		
jack3	bid -1 8 16	6	20	raise
jack1	bid -1 8 16	8	20	call
jack2	bid -1 8 16	8	20	call
	com card			
jack2	bid 0 4 8	8	8	raise
		1		
jack3	bid -1 8 16	6	16	raise
		2		
jack1	bid -1 16 24	4	24	raise
		1		
jack2	bid -1 16	6	24	call
jack3	bid -1 8	8	24	call
	com card			
jack2	bid 0 4 8	8	8	raise
jack3	bid -1 8 16	-1	0	fold
jack1	bid -1 8 16	8	8	call

TEST WHEN CALLED PLAYER DOESN'T  
GET CHANCE TO RAISE WHEN ALL CALLED

player	move		total	
jack1	hole cards			
jack2	hole cards			
jack3	hole cards			
jack2	small	2	2	
jack3	large	4	4	
jack1	bid -1 4 12	4	4	call
jack2	bid -1 2 12	2	4	call
jack3	bid -1 0 8	0	4	check
	com cards			
jack2	bid 0 4 8	0	0	check
jack3	bid 0 4 8	0	0	check
jack1	bid 0 4 8	0	0	check
	com card			
jack2	bid 0 4 8	4	4	call
jack3	bid -1 4 12	4	4	call
jack1	bid -1 4 12	4	4	call
	com card			
jack2	bid 0 4 8	4	4	call
jack3	bid -1 4 12	4	4	call
jack1	bid -1 4 12	4	4	call

TEST 3 raises are only allowed

player	test NO more move	total	
jack1	hole cards		
jack2	hole cards		
jack3	hole cards		
jack2	small	2	2
jack3	large	4	4
jack1	bid -1 4 12	4	4 call
		1	
jack2	bid -1 2 10	0	12 raise
		1	
jack3	bid -1 8 16	6	20 raise
		2	
jack1	bid -1 16 24	4	28 raise
		1	
jack2	bid -1 16	6	28 call

# if high bid is made should throw error

#### **11.4.4 – Combined Server And Applet Testing**

Applet and Server testing was reasonably simple by this point. The author had been assured the processes worked well on their own. It was a case of getting them to communicate with each other and make sure the above processes worked with each other as expected.

#### **11.4.5 – User Testing**

To ensure errors had not been missed by the user, and the author's understanding of the game was not flawed, user testing was carried out.

At this point the author would like to thank them for the time spent working through the testing process and helping to ensure it worked correctly.

Many games were played over a couple of hours to see if any flaws could be found.

A number of flaws were found:

- On some hands the same hole card was dealt to two players. This naturally cannot happen in a real game as only one player can have a card.

- A player dropped out for no reason when it was his go and froze the system, which it was not designed to do. It should have dealt with that situation.
- When the community chest won not all players hole cards are always shown. Sometimes the dealer is not shown.
- The dealer assigned value does not always show up on the Applet.
- When a player folded and did not have enough money to stay in the game the wrong hole cards were shown for him. This issue seems to be related to either seating or people not being removed correctly.

The main comment was it was a good process but some of the cards were not very clear to read. Threes could be misread as eights and hearts as diamonds if they looked quickly. Also a few of the graphics moved about a bit depending on the situation of the game.

#### ***11.4.6 - AI Testing***

Due to the lack of development within the AI side there was not a great deal of testing involved. The main testing was to get the controller to first create one bot and enter it into a room to allow the author to play against it in order to test it worked correctly.

Once this was achieved the next development was create four players and put them in a room and let them play against one another to see if they played against each other correctly.

After a number of alterations these all worked correctly.

#### ***11.5 - Project Code***