Husky Holdem High Level Design Specification - Version 3.0 12/10/08

Aaron Cook, Dan McNamara, David Chanko, Chris Revel

1. Purpose of Document

This document contains the high level design specifications for the Husky Holdem system.

There are four main software sub-systems which make up Husky Holdem. The first is the Account Maintenance interface, where users can edit preferences and view account data. The Statistics Generation component is where a user can search the system's database for information regarding past games and also view statistics generated from saved data. The third sub-system is the Poker Game, which is a complete online poker table where users can play against each other. Lastly, the Player Prediction interface allows a player to view key statistics about their opponent and request advice from the system about how to play a hand or respond to an opponent's action.

A fifth and final component is required to connect these sub-systems together. This system provides user authentication, allowing the user to log in once and access all of our systems. This component is where the login credentials, a username and password, will be verified to provide personal access to all four subsystems.

This document discusses in detail these systems and their underlying components. In addition, the necessary hardware is outlined and some details of the more complicated sub-system components are discussed.

2. Scope of Project

This project is intended to replicate the functionality of two already available services. First, online poker playing website is a service that allows remote players to gamble, the site itself is the trusted escrow for funds. Thus, we will serve an external customer market. Second, we will replicate a poker statistical analysis program such as Hold'em Manager [1], which is a stand alone application that uses hand histories as inputs. This is a "shrinkwrap" application, and serves poker enthusiasts. This group of enthusiasts will be users of our online poker client that wish to examine their poker game in closer detail. The third portion is an original concept, a programmatic assisted client for our online service will provide tips to the player based on what it sees. As an example, our programmatic assisted client might notice the user seems to be playing too many hands, and recommend they fold their more marginal holdings, but in a way that is completely undetected by the other users. Because this functionality will depend heavily on our implementation of the first product, we will be less specific about how they will be interfacing.

3. Sub-Systems

The system consists of four main sub-systems, a basic account management screen, the statistics generation features, the poker game, and the player prediction interface. In addition the fifth integration component ties these systems together. These subsystems are detailed in sections 5.A, 5.B, 5.C, 5.D, and 5.E respectively.

A. Account Management

Purpose

The purpose of the Account Management subsystem is to allow user to manage their account via an online interface.

Functionality

User can create new accounts from the login screen, and this system will handle registration. The user will need an email address, a unique id as a username, and a mailing address.

Existing users can change their password, their mailing address and view their fund balance from this screen.

Component Parts

The first component of this system is to allow the user to edit their account settings. Via this webpage, the user is able to edit their account information and the changes are stored in the mysql database. The user is able to change their password, email address, and location via this screen.

The second component allows the user to delete their account or to clear their accumulated statistics.

B. Statistics Generation

Purpose

The purpose of this sub-system is to provide the user with the statistics generation functionality of the system. The features found here include an interface to upload and download hand histories, a statistics viewing area where users can examine different measures of their past performance, and a graph generation interface to provide the users a visual representation of the data collected.

Functionality

The functionality requirements that are met by this system include all of the statistics generating and viewing functional requirements. This includes the functions described in the User Interface section of the functional specifications under the section Statistics Generator for the input of raw data and the generation and viewing of statistics. The specific functions included are the searching and downloading of hand histories, as well as the upload interface, the general search interface, where users can search for past hands based on a particular opponent or a set of cards etc., and the presentation requirement where this information is displayed to the user either in a list, table, graph, or a combination of these options.

Other Sub-system Interfaces

This sub-system interfaces with the player prediction interface where users can bring up a statistics summary for their opponent to supplement their strategy. The player prediction sub-system is comprised of a subset of the statistics generation component plus some specific functionality to display the information.

Component Parts

The first component of this system is the hand history upload interface. Here a user selects a hand history file from their computer and the system parses the files and adds the data to the statistics database. From here a user can also download the hand history data from the database to their computer in text file form.

A third component allows the user to select a statistic they would like to view, which is then generated for viewing by the statistics viewer component. Here, if appropriate, a graph representing the statistic is generated in addition to a textual summary and explanation of the statistic. Another component allows users to search the database for a history of a particular hand or multiple hands and provides the user with the option of saving the results to a text file.

A user can also search the database for information about opponents and generate statistics that can gauge the opponent's style of play. This data is presented through the same interface used to display the player's own statistics. The statistics presented here are similar to those that are displayed in the poker prediction window.

C. Poker Game

Purpose

This sub-system will allow users to play poker with each other through the Internet. The hand histories that are generated from this system can be imported into the stats application. The user interface of the poker application will be the most developed since this is the one users will be using for the longest time. The user will have different screens to view his account information, view tables currently available, and of course the table when he is actually playing.

Functionality

Our poker application must provide users with the ability to play heads up Texas Hold'em exactly as described by our rules, 100% of the time. Any service that a real-world casino dealer provides will also be provided by our service. This includes shuffling, dealing, alerting players to their turns and the current bet amount, collecting and verifing the size of bets from players, exposing community cards, deciding the winning hand, and delivering winnings to winning players(thus, also handling splitting the pot correctly in the case of a tie). To provide such service, our client will do the following. The user must trust the application to maintain his funds balance accurately. The interface must be simple and appeal to the intuition of the user. The user will be able to look at a single screenshot of the application and know what his cards are if he has not folded, whose turn it is, what bet amount they will be facing, who else is in the hand, what all the community cards are, how many chips each player has, and how many chips are currently in the pot. The user should rely on the computer to shuffle the deck in a random manner, no user should be able to predict the deck order. Further, the shuffle should take less than 5 seconds, so that the user can enjoy more time actually playing. Or equivalently, the system should provide the user with the maximum amount of hands per hour, while still allowing users 30 seconds to act.

Other Sub-system Interfaces

This sub-system will eventually be used by the player prediction sub-system, however since that system will be developed last, this system will not depend on it. We will try to identify useful places for the player prediction to display info, and leave spaces available on the table interface for this purpose.

Component Parts

The poker game subsystem will have the following components- the card shuffler, the table manager, the table itself, and the funds management system.

The card shuffler will handle the pseudo-random shuffles of the decks to be used at the tables. A single shuffle generator will "hand off" single 52 card decks to the tables, so our tables are ignorant of the shuffle method. The time between deck request and deck delivery must be less than 5 seconds.

The table manager provides a list of tables to the users, and the following statistics about the table- blind amounts, players currently playing, average pot size over the past 20 hands, percentage of players seeing the flop per hand over the last 20 hands, and hands per hour speed over the last 20 hands.

The funds management system will be in charge of maintaining the funds of users across tables. So, we need to make sure that the total amount of chips in play is always consistent, and that a user is not using one unit at two different tables. This sub-system will also serve any system that relies on the user's fund balance.

D. Player Prediction

Purpose

The Player Prediction sub-system is designed to provide the user with the in game player prediction functionality of the system. This sub-system is an extension of the statistics generation portion of the system. Accessed through the Poker Game subsystem, a user can generate a set of summary statistics about an opponent to help the player decide how to handle certain game situations.

Functionality

The player information and prediction functionality requirements are covered by this sub-system. This includes the functions described in the User Interface section of the functional specifications under the Player Prediction Interface such as the generation of opponents' statistics and the ability to generate a prediction of what the opponent might do based on this information. The information will be presented in an unobtrusive pop up window which addresses the requirement that this information be available while the user is playing a game without the game experience being disrupted.

Other Sub-system Interfaces

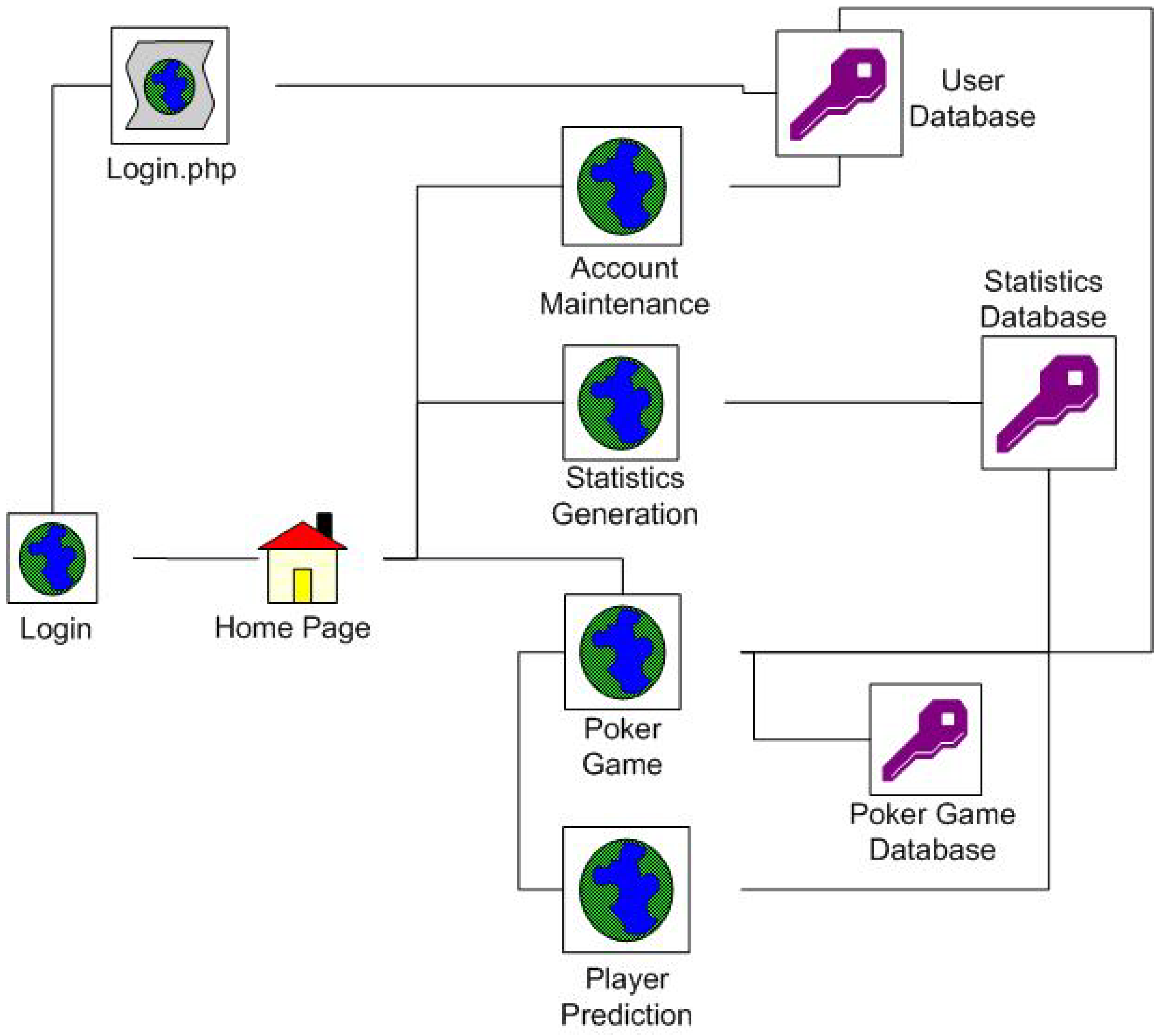
This sub-system is accessed through the Poker Game sub-system and presents data derived from the Statistics Generation sub-system. Although this sub-system will be heavily dependent on the Poker Game and Statistics Generation sub-systems, there is enough separate functionality in the user interface and searching ability to warrant a separate sub-system.

Component Parts

The main component of this sub-system is the display screen. Inside the poker game, a player has the ability to click on an opponent to view the summary of information in a pop-up window. Additionally, a player can search for other information about an opponent through the interface such as their behavior in similar situations in the past. All of this functionality will be put into a small pop-up window in order to disrupt the player's game experience as little as possible.

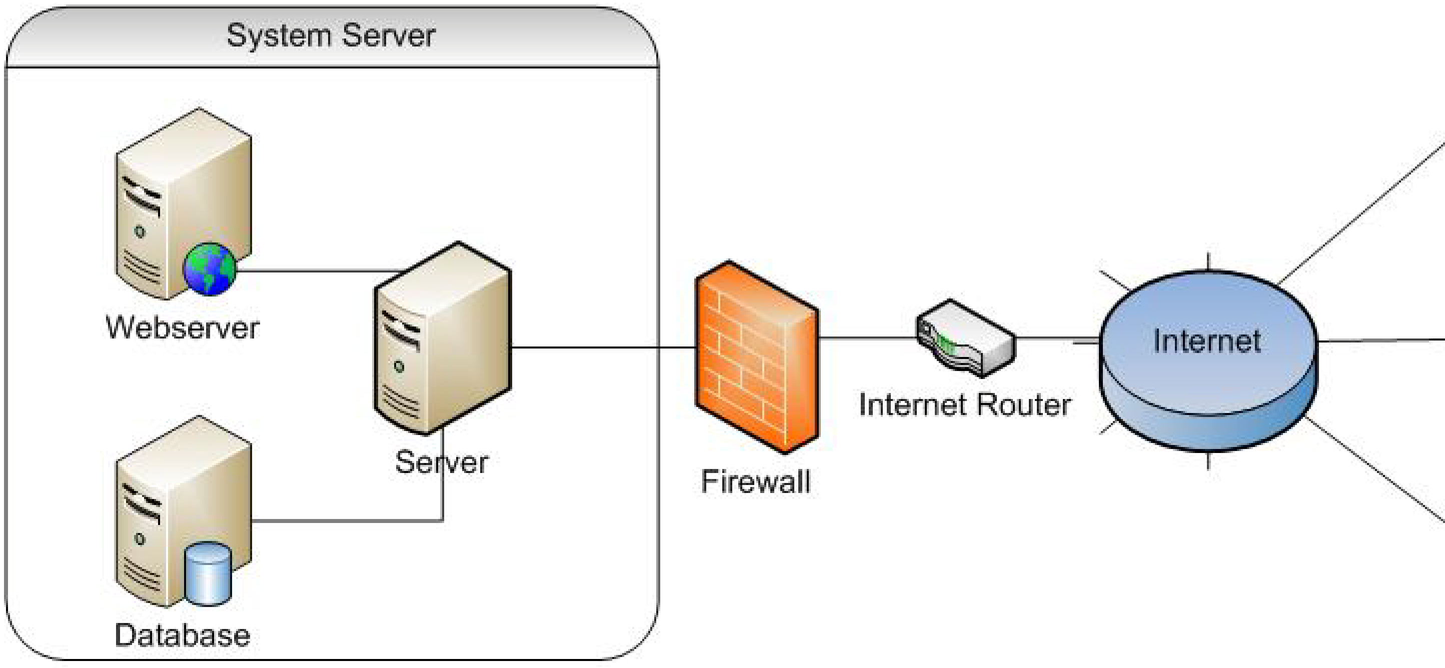
E. Sub-System Integration

These four sub-systems are integrated in the following manner. A general login interface gives a user access to a home page. From here, the account management, statistics generation, and poker game are accessible. These three systems are independent of one another and are kept separate, although in each component a menu bar will allow the user to transition between the systems easily. The player prediction sub-system is accessible through the poker game component and is not available from any of the other sub-systems due to its specific purpose of performing game time analysis.



4. Hardware Components

Hardware components for the system consist of a server with which users' computers interact with. For our purposes, we will have a single server equipped with Apache web server with php and a MySQL database. The server connects to the client computers over the internet and as a result the server requires a network connection. To help protect the system, our server is located behind a firewall.



5. Software Components

A. Account Management Components

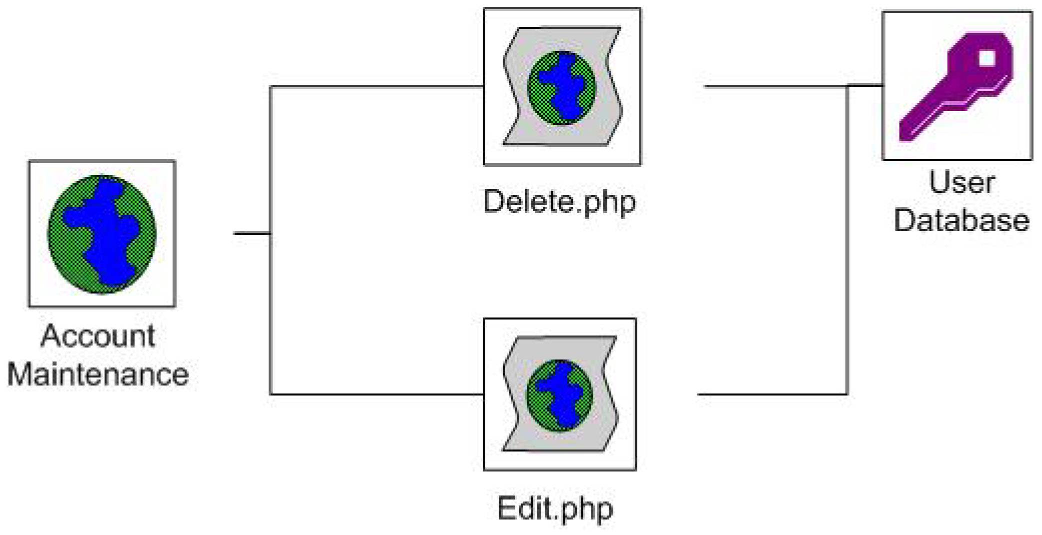
edit.php

The edit.php file obtains account information from the Mysql database and displays it in editable form to the user. If the user makes and submits changes to their account information, edit.php validates the information and then updates the database accordingly. Changes might include- changing the user password, changing the mailing address.

This meets the functional requirements of allowing a user to change their password, view their account value, and change their identification information.

delete.php

The delete.php file allows the user to delete their profile entirely from the database. It also allows the user to clear various stored information, such as statistics. Both options are handled in a way such that confirmation is required, in an attempt to avoid accidental deletions.



B. Statistics Generation Components

Statistics Database

MySQL database containing stored hand histories, game information, and other statistics information. Inside this database is a stored procedure that handles the creation of an action set, which is the system's internal representation of a hand history file. While other database interactions are performed in-line, this is a fairly complicated process and is modularized out for clarity's sake. In addition to the generation of hand histories, the generation of a statistics will be handled by stored procedures as well.

upload.html

Allows the user to select a hand history from their computer to upload. Here a user also specifies the format of their hand history.

upload.php

Handles the actual uploading of hand histories to the database. Upload.html passes a file name and path to this component and after transferring the file to the server, the file is parsed and the information is entered into the statistics database. The parsing functionality is handled by taking the input line by line and based on the format of the line and key words such as a player name or a poker specific verb, the information is assembled into an event, which is the system's representation of a single line of hand history data. This event can have players, cards, value amounts, and a verb attached to it and is stored in the statistics database with a sequence number for retrieval.

Meets the functional requirement of allowing a player to input a hand history.

gethistory.html

If a user would like to view a hand history or a set of histories, they can set a number of parameters and request the information from the database. This page also allows a user to save the data to a text file.

gethistory.php

Handles the retrieval of the hand histories and saves to the user's hard drive if requested. This performs the opposite functionality of upload.php. Here a stored procedure pulls all the events corresponding to a particular poker hand and prints the information to the screen or file if specified. This is handled by iterating through the list of actions and printing lines out to the screen based on the verbs attached to the actions. This can be repeated for any number of hands based on the range of hands selected by the user to print.

Fills the functional requirement of allowing a user to recover a hand history from the database.

statistics.html

Allows the user to select the statistics they would like to view. Allows access to upload.html and gethistory.html. Player can set a number of criteria to perform some set reports on the database as well as request a variety of statistics.

statistics.php

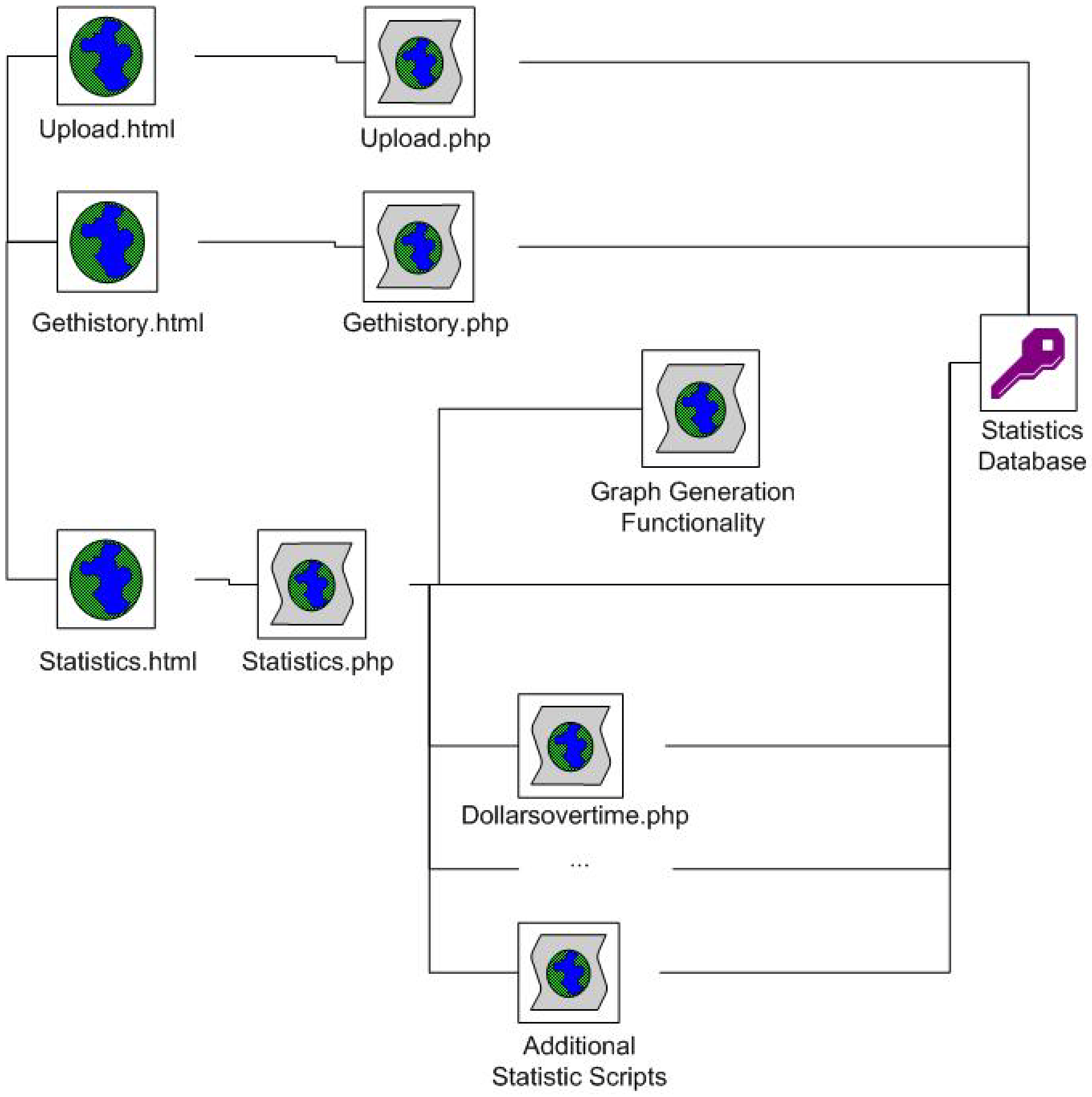
By far the largest and most detailed component of the sub-system, this handles the searches generated in statistics and generates an HTML page with the results. Generates graphs or charts, tables, or other diagrams as well as a text description of what is displayed. Will be split into several sub files for increased modularity and to allow the Player Prediction sub-system access to several functions. The general flow of events for this component involves receiving a query or a statistic to generate from statistics.html, performing a database call, and generating an HTML page to display the results. The code to perform the database interaction and generate a particular statistic is modularized out to allow the Player Prediction sub-system to access the functionality. This will involve the creation of several additional code files with names such as moneyovertime.php to represent the statistic they provide. There are many statistics listed in glossary and for each of these there will be a code file to provide it. As mentioned above, the database interaction will be performed by a stored procedure and the based on the results returned and the user's request, HTML is created with the aid of the graph generation functionality.

Meets the functional requirements of allowing a user to select a number of statistics to generate or to formulate a search query to look up past opponents or past hands. Also meets the requirement of presenting this information of presenting this information to the user.

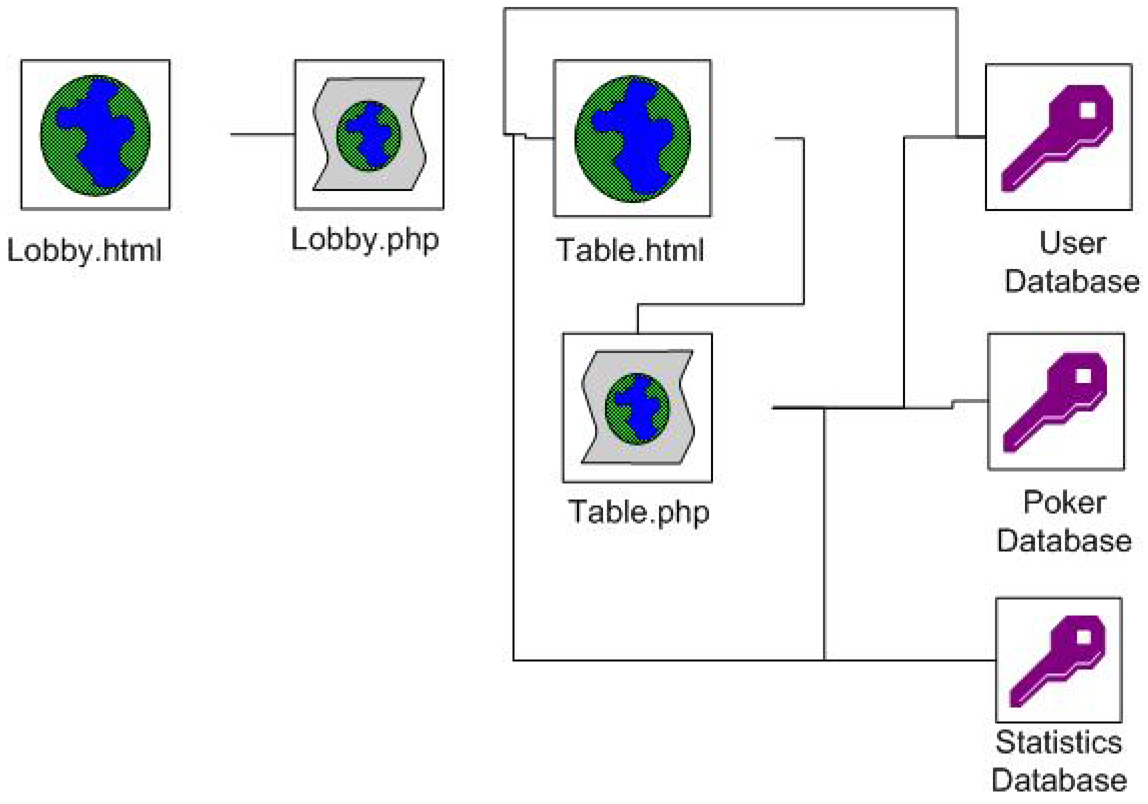
Graph generation functionality

This component will make use of one of several third party graph generation APIs. The package that will be used has not been determined and this decision will more than likely be made during implementation when we see exactly what we need.

Fulfills the graphical component of the functional requirement of displaying the query or statistics results to the user.



C. Poker Game Components



lobby.html

This page will allow the user to see the list of tables, and links to watch the game in progress. So, this page will not show any cards being dealt, but the list of table names and the quick stats described previously.

lobby.php

This is the php to provide the info found in lobby.html.

Facilitates the meeting of the functional requirement of providing the user with a playable poker game by providing a transition from the rest of the system to the poker game interface.

table.html

This is the actual table in play, here we can see the communal cards, players at the table, see part C for more detail. This page will also allow the user to join the table, and use his funds to gamble.

table.php

This code provides the play to the users, including all of the user interaction and card dealing. This component will also need to properly allocate the users funds based on the results of hands, so it will be interfacing with the database component. In the diagram below we see how the table will connect with the user database for funds management, the poker database for game flow rules, and finally the statistics database to ensure that every single hand is recorded in our private section, in case of major system catastrophe or any other forensic need.

Meets the functional requirement of providing the user with a playable poker game.

<AC>

multiplayer.php

The multiplayer.php acts as a gateway to the multiplayer poker game. If the user attempts to access multiplayer.php without being logged in, they are denied If it is not passed a table number, it displays a listing of tables in order. These tables are listed in numeric order, and when clicked, performs a recursive call on multiplayer.php, passing the table number that the user wants to join. When multiplayer.php receives a table number, and calls the getState function, which returns the state of the table that the player wishes to join (see below for the description of the gameState datastructure). From this state, multiplayer.php determines if the game is full. If the game is full, the player is denied access to the table. Then isPlayerAtTable us called, to check if the player is already at the table. If the player is already at the table, then they must have been disconnected, so they are allowed to rejoin the game. Otherwise, the player is added to the table via the addPlayer function. If successful, the multiplayer then calls the printTable() function, which displays the table.

-getState(table) - Given a table number, this function queries the poker database table for the gameState field. It unsearlized this data which is stored in blob format, into a php array. The function gets the username and id of the user calling the function, and appends it to the gameState datastructure and returns it. It returns false, if the table cannot be found.

-setState(table,gameState) - This function serializes the php array gameState, and then inserts it into the gameState field of the poker table, where the ID equals the given table number.

-addplayer(userID)- Calls getState, and adds the user with ID = userID to the gameState's list of players. setState(gameState) is then called, updating the state the game and successfully adding the user to the game.

-isPlayerAtTable(ID, table) - Calls getState(table), and receives the current state of the game at that table. The function then checks if the player with the give ID is at that table. If so, true is returned. Otherwise, the function returns false.

-printTable(table)- Displays the table.html to the user and also initializes header.js javascript code

header.js

The header.js code contains javascript code that runs simultaneously alongside multiplayer.php. This code is inactive when first loaded, and not activated until the printTable function displays the poker table. This code is executed on the client's machine and uses AJAX calls to communicate with the server and it's PHP code. The header javacript code is executed every 3 seconds. It begins by calling ping.php via an ajax call. This php file returns the gameState as a JSON object. The header.js then takes this JSON object and converts it into javacript array. Various values from the gameState are then printed to the page, such as the amount in the pot, the cost to raise and the cost to call. Display of variables releated to a specific player requires attention, because these must be displayed relative to the user's postion. First, the list of players derived from the gameState is looped through, and the position of the player is found at the table. The users hand other other relevant information is then printed at the bottom of the table. Then, from that position each player until the end of the player list is printed at their respective location at the table. This is then repeated seating all of the player clockwise from player. This method preserves the seat positioning for every user playing the game.

The header.js also displays the possible actions a player can perform. If the player chooses an action, such as to raise, setJSON is called, given the JSON object containing the action the user wishes perform, along with a url to get.php.

-getJSON(url) - Performs an ajax call on the given url ("ping.php"). It takes the received JSON object and converts it into an array. This is how the gameState is transfered from PHP to javascript.

-setJSON(url, JSON Object) - Performs an ajax call passing a JSON object to php. By using this function, a player action is sent from the header.js to server via get.php.

ping.php

The ping.php represents the core of the poker game, as it is the bridge between user to server communcations and the game logic. It takes a table number, and retrieves the gameState. From this, it updates the last time the user pinged the server (in the gameState datastructure). It then checks the player list for players if that are late submitting their turns. If any are found, they are forced to fold, and a timeout counter is incremented. If the players timeout counter is greater than 4 (they've missed 4 turns), they are disconnected from the table. The gameState is then passed to the gameLogic fuction, which determines who's turn it is, what stage of the poker game we are at, etc. The gameState is updated, and stored in the database via the setState function. The gameState is then returned as JSON object.

-removePlayer(ID, gameState) - Removes a player with the ID from the gameState datastructure (and thus the game). It then returns gameState.

-gameLogic(gameState) - Passes the gameState to the the game logic function (described below). And returns the updated gameState.

get.php

Is called via javascript when a user selects an action via the setJSON javascript function. This action is passed via a JSON object, that simply contains a number that corresponds to an action.

The get.php gets the gameState via the gameState function. The player calling this file is determined through a PHP Session variable. If it is this players turn, the function then checks the action that the player wants to perform. If the action is 1, then the player wishes to fold. The fold function is then called. If the action is 2, the player chooses to raise. If the action is equal to 3, the call is then called. This function also updates the the calling player's last ping timestamp is updated. The updated gameState is then stored in the database, via the setState function.

-fold(ID, gameState) - Given the ID of player, this function makes that player fold. The cost to fold is deducted from their kibble. The gameState is updated and returned. This function is called when a player chooses to fold and when a player has missed his turn.

-call(ID, gameState) - Given the ID of player, this function makes that player call. The player's cards are updated via the card shuffling function. The gameState is updated, and the gameState is returned.

-raise(ID, gameState,amount) - Given the ID of a player, this function makes that player raise the bet. The gameState is updated and returned.

gameState Datastructure

Players only communicate each other via the gameState datastructure. At any moment during the course of a play, the current state of the game is represented by this datastructure. The game state is stored in the poker database, as a serialized PHP array. This allows it to be represented by just two columns in a database table. This allows the the game state to be quickly and easily accessed. More importantly, during development, it allows us simply to add new key to the PHP array in PHP, and store it into the database using the same setState function; since we don't have to represent the new datastructure by adding new columns in the mysql table. The gameState is initialized via the initGame function. The gameState is retrieved via the getState function and updated via the setState function (described above).

Structure:

-gameState.PlayerList - Contains a list of players at the table. This list is a multidimensional array. Each seat is represented by a key, zero through six. Each element of this array then contians an associative array that represents the player sitting in that seat. This associative arrray stores the players ID, username, hand, winnings, kibble, last ping time, and their last command.

-gameState.Pot - Stores the current value of the pot in kibble.

-gameState.CurrentRoundofBets - Stores the current round of bets

-gameState.toCall- Stores the cost to make a call at this point of the round. This value is 0 if it is not possible call at this point of the round.

-gameState.toRaise - Stores the minimum amount to raise at this point of the round. This value is 0 if it is not possible for the player to raise.

-gameState.Dealer- A value zero through 6, that denotes where the dealer is located at the table.

-gameState.init() - Inititalize the gameState to specific values, when a new round starts.

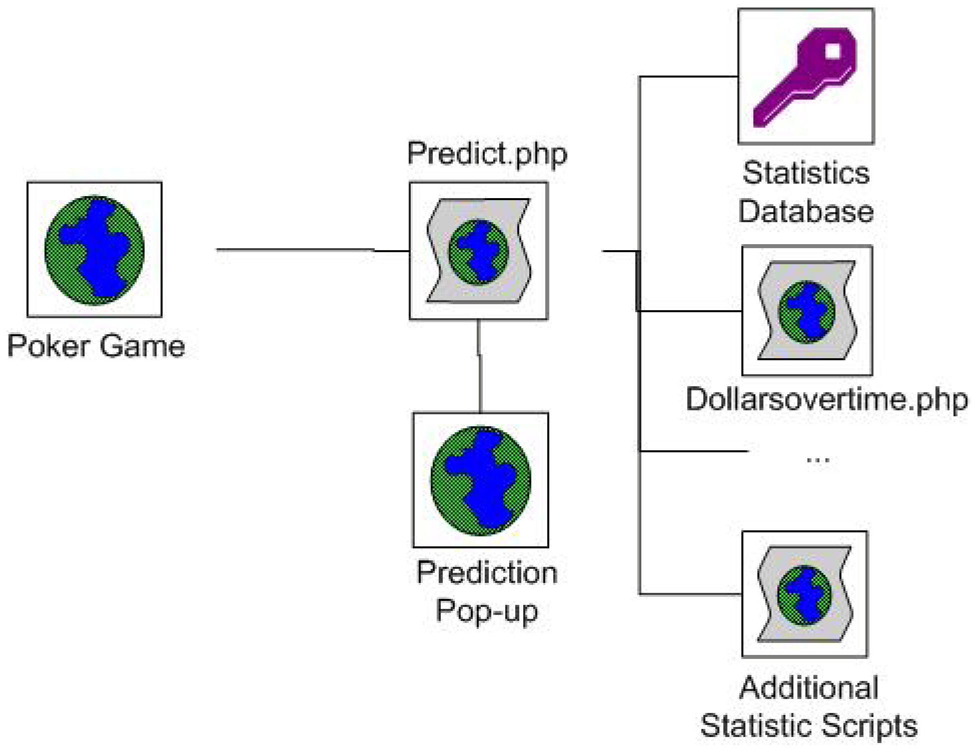
D. Player Prediction Components

This sub-system makes use of several sub-functions of the statistics.php component of the Statistics Generation sub-system.

predict.php

Based on user actions this, component creates an HTML sub-page to supplement the Poker Game interface. Relies on statistics.php functionality, specifically the modularized .php files which provide the basic statistics. When predictive statistics are requested about a player, the appropriate database calls are performed and the data is organized to be displayed. The type of data to be displayed is the same for each player, so the creation of the HTML page involves mainly filling in blanks in a template and presenting the results to the user.

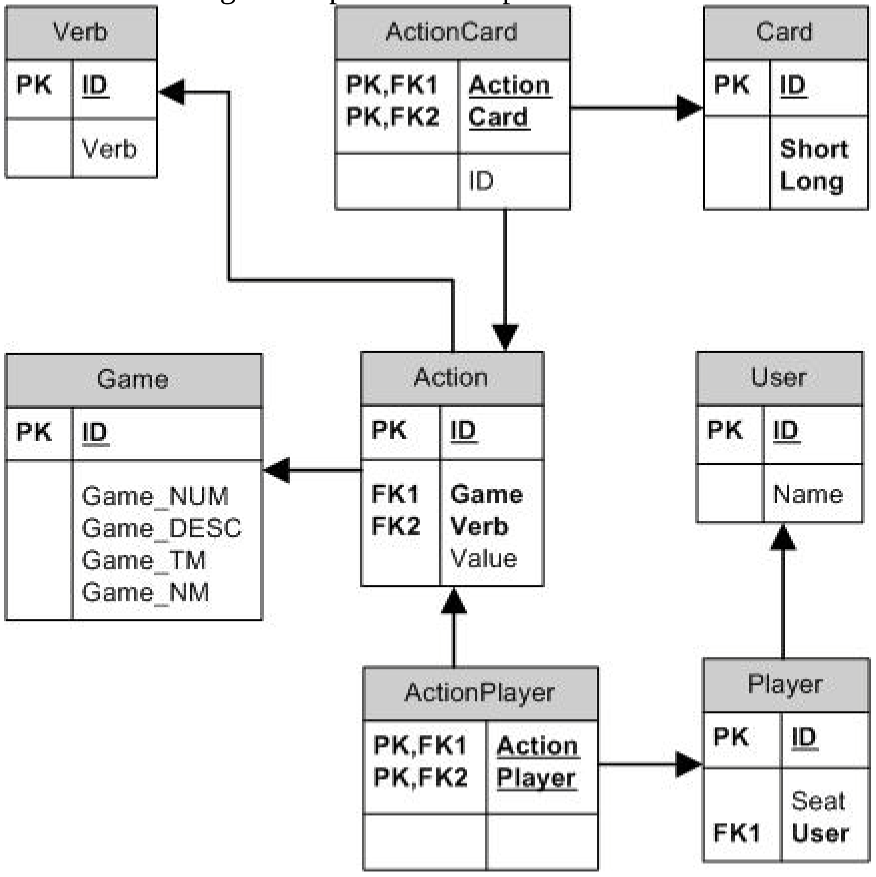
Provides the functional requirement of allowing the user to view statistics about an opponent during a poker game. Also, meets the requirement of providing the user with some indication of how an opponent might behave.



6. Sub-system Requirements

Statistics Generation

One aspect of the Statistics Generation sub-system that has some requirements of its own is the statistics database. This is where all the uploaded hand histories are stored and where the Poker Game actions are logged. As a result, an efficient database design needs to be implemented to ensure the entire system runs smoothly. The current model is presented below, but it is still being developed as our implementation evolves.



This data model allows us to represent a line in a hand history file as an action in our database. A hand, called a game in our database, is made up of a sequence of actions as well as some general identification information. Each action is connected to a game and must have at least a verb. These actions can also contain a value and be connected to players or cards. The sequence number of an action results allows the system to know how to order events when printing hand history files.

Poker Game (Game Logic)

The poker system will be in charge of maintaining the flow of the game, in the form of actions received from the players during their turns. This will involve the correct management of any user disconnects, as well as enforcing time limits on user turns. The game must also decide which hand is the winner at showdown. At the end of a hand, the game will also be responsible for adjusting the funds of all players involved in the hand accordingly.

The following is an initial revision of the algorithm that will determine the best poker hand out of a set of 7 cards:

```
 private card[] cards = new card[7];
private handValue getSevenCardHighValue(card[7] cards)
{
    //determine poker ranking
    handValue handVal = (7 high)
    for(int j = 0; j < 6; j++)
    {
        for(int i = 0; i < 5; i++)
        {
            if(j != i)
            {
                //we have chosen which cards to exclude
                thisHandVal = getHandVal(all cards but i and j)
                if(thisHandVal > handVal)
                {
                    handVal = thisHandVal;
                }
            }
        }
    }
}

private handValue get5CardHighValue(card[5] cards)
{
    bool flush = true;
    bool 3oKind = false;
    int 3oKindVal = null;
    bool 4oKind = false;
    int 4oKindVal = null;
    int high = 0;
    int low = 13;
    int 1stpairVal;
    int 2ndpairVal;
    bool 1stPair = false;
    bool 2ndPair = false;
    bool 2pair;

    suit Suit = null;

    foreach(card theCard in cards)
    {
        //check flush
        if(lastSuit.isNull() || flush)
        {
            Suit = theCard.Suit;
        }
        else if(Suit != theCard.Suit)
        {
            flush = false;
        }

        //check high
        if(high < theCard.Rank)
        {
```

```
    high = theCard.Rank;
}

//check low
if(low > theCard.Rank)
{
    low = theCard.Rank;
}

//check for multiples
int count = 0;
foreach(card otherCard in cards)
{
    if(otherCard != theCard && otherCard.Rank == theCard.Rank)
    {
        count++;
    }
}
switch(count)
{
    case 0:
        //no pairs detected, do nothing
        break;
    case 1:
    {
        //one pair
        if(!1stPairVal.isNull && 1stPairVal = theCard.Rank)
        {
            //we found the pair again, ignore
            break;
        }
        else //we know this is the first card of the pair
        {
            1stPairVal = theCard.Rank;
            1stPair = true;
        }

        //two pair
        if(1stPair && 1stPairRank != theCard.Rank)
        {
            if(!2ndPairVal.isNull && 2ndPairVal = theCard.Rank)
            {
                //found second pair again, ignore
                break;
            }
            else
            {
                2ndPairVal = theCard.Rank;
                2ndPair = true;
            }
        }
    }
    case 2:
    {
        //there can only be one 3 of a kind in 5 cards
        if(!3oKindVal.isNull)
        {
            //found triplet again, ignore
            break;
        }
        else
        {
            3oKindVal = theCard.Rank;
            3oKind = true;
        }
    }
    case 3:
    {
        //there can only be one 4 of a kind in 5 cards
        if(!4oKindVal.isNull)
        {
            //found quads again, ignore
            break;
        }
        else
        {
            4oKindVal = theCard.Rank;
            4oKind = true;
        }
    }
    default:
    throw exception;
}

//now set final value
if(flush && (high == (low + 4)))
{
    //royal flush
}
else if(4ok)
{
    //four of a kind
}
else if(3oKind && 1stpair)
{
    //full house
}
else if(flush)
{
    //flush
}
else if(high == (low + 4) && !3oKind && !1stPair && !2ndPair)
{
    //straight
}
else if(3ok)
{
    //three of a kind
}
else if(1stPair && 2ndPair)
{
    //two pair
```

```
                }
                else if(1stPair)
                {
                    //one pair
                }
                else
                {
                    //high card
                }
            }
        }
    }
}
```

7. Applicable Documents

McNamara, Dan. "Project Proposal" Husky Hold'em 6 October 2008. Version 2.

Chanko, David; Cook, Aaron; McNamara, Dan. "High Level Functional Specifications" Husky Hold'em. 22 September 2008. Version 2.

8. References

1. http://www.holdemmanager.net/

9. Glossary

3Bet% - how often a player raises preflop when facing a raise. thus (re-raising).

$/hour - (dollars/hours) - your effective hourly wage at the table.

Agg - (aggression factor) - (bets + raises)/calls

API - Application programming interface, detailed definition here:
http://en.wikipedia.org/wiki/API

BB/100 - Big blinds won per 100 hands

Bet - To wager money on ones hand, a sum which must be called by an opponent or the current pot and the bet is won by the bettor.

Big Blind or BB - a forced bet, typically twice the size of the small blind. The BB player is usually last to act pre-flop but then will be in early position for the rest of the hand.

Bluff - a bet that is intended to make the opponent fold his hand, which is presumably better than our own. (If we thought our hand was better, we would want to value bet)

Button - In a casino, there is a professional dealer that handles the cards, so a dealer "chip" or "button" is used to indicate who is in the blinds. Thus if you are "on the button" you are last to act in all rounds after pre-flop, and thus the best position at the table.

Call - To place an equal amount of chips into the pot as the bettor, indicating your hand is still valid and you still have a claim to win the pot.

Check - Effectively a bet of 0, The player places no money in the pot, but will have the chance to call if an opponent bets. If all players check, the next card is exposed for "free".

Community cards- cards in poker that are shared by every player in the hand. Thus a players final hand will be 5 cards from his private or hole cards and the community cards.

Continuation bet- A bet made on a later street by a raiser in the previous round of betting.
Quick example-
```
*** HOLE CARDS ***
Dealt to DSM4CK [Qh Ad]
111appi: folds
Lucid_Wolf: folds
charlotti: folds
7FatMiro7: folds
pokerbüro: calls $0.05
DSM4CK: raises $0.20 to $0.30
pokerbüro: calls $0.20
*** FLOP *** [3h Jc 4h]
pokerbüro: checks
DSM4CK: bets $0.30 <-- Continuation bet.
pokerbüro: folds
Uncalled bet ($0.30) returned to DSM4CK
DSM4CK collected $0.60 from pot
DSM4CK: doesn't show hand
```

Cbet% - % that a player will continuation bet. So, A high Cbet% implies the player is betting even when he did not make a stronger hand with the new card(s). Or, a low percentage suggests he is only betting when his hand was helped by the new cards.

EV - (expected value) - In gambling situations, this is the average win in the long run. So, betting on a single side of a six sided dice at 1-1 odds would have an EV of 1/6. Profitable bets have expected values above what the cost of wagering was.

Hand history - The textual record of a single game of poker, that is the play between shuffles.
an example-
```
PokerStars Game #20372428900: Tournament #107954371, $10.00+$0.50 Hold'em No Limit -
Match Round I, Level III (25/50) - 2008/09/13 10:44:52 ET
Table '107954371 1' 2-max Seat #2 is the button
Seat 1: DSM4CK (1215 in chips)
Seat 2: gagnant2 (1785 in chips)
gagnant2: posts small blind 25
```

```
DSM4CK: posts big blind 50
*** HOLE CARDS ***
Dealt to DSM4CK [5d 5s]
gagnant2: calls 25
DSM4CK: checks
*** FLOP *** [Kc Qd 6h]
DSM4CK: checks
gagnant2: bets 50
DSM4CK: calls 50
*** TURN *** [Kc Qd 6h] [7d]
DSM4CK: bets 150
gagnant2: calls 150
*** RIVER *** [Kc Qd 6h 7d] [4c]
DSM4CK: checks
gagnant2: bets 150
DSM4CK: calls 150
*** SHOW DOWN ***
gagnant2: shows [9d Td] (high card King)
DSM4CK: shows [5d 5s] (a pair of Fives)
DSM4CK collected 800 from pot
*** SUMMARY ***
Total pot 800 | Rake 0
Board [Kc Qd 6h 7d 4c]
Seat 1: DSM4CK (big blind) showed [5d 5s] and won (800) with a pair of Fives
Seat 2: gagnant2 (button) (small blind) showed [9d Td] and lost with high card King
```

Funds - the virtual currency which is used on our site, used to acquire chips and play poker. Our site will not provide any real-money play.

FTOP - (fundamental theorum of poker) - "Every time you play a hand differently from the way you would have played it if you could see all your opponents' cards, they gain; and every time you play your hand the same way you would have played it if you could see all their cards, they lose. Conversely, every time opponents play their hands differently from the way they would have if they could see all your cards, you gain; and every time they play their hands the same way they would have played if they could see all your cards, you lose." The Theory of Poker. on page 17-18

Fold - To surrender ones stake in the pot by passing his cards to the dealer, there is no way this player can win the pot, and he will not act for the rest of the hand.

Flop - in texas hold'em, directly after the pre-flop action concludes, the dealler will place 3 cards face up in the center of the table, these cards are considered the flop, and are community cards. The players now have another betting round, known as "flop action".

GUI - graphical user interface, a way to use a program that usually features buttons and menus the user uses a mouse to navigate, opposed to a command line interface that uses the keyboard and provides no hints to the correct commands.

G-Bucks - (Galfond Dollars) - http://sports.espn.go.com/espn/poker/columns/story?columnist=bluff_magazine&id=2817110 I suggest reading the following article, but essentially this is a way to accomodate hand ranges into EV. So if you can assign your opponent a very specific range, you must evaluate your plays based on an average of his possible holding to get the true expected result.

IP - (in position) - In a hand, the player last to act is considered "in position", since he will be able to see the other players actions before he must act, a considerable advantage. Note the player on the button will always be in position if he plays his hand.

No Limit Texas Hold'em - a poker variation in which each player recieves two "hole" cards, and combines them with community cards to make a hand. Complete rules found here.
http://www.pokerstars.com/poker/games/texas-holdem/

OOP - (out of position) - In a hand, the player first to act is considered "out of position", since he will have to act before seeing what anyone else does, a considerable dis-advantage. The player in the small blind will always be OOP if he plays his hand.

Pre-Flop - in texas hold'em, the first round of betting is called "pre-flop action". Each player only has 2 cards, and there are not any community cards available yet.

Poker - "... a game of wagering based on imperfect information that uses cards to construct the situations for wagering." -Harrington on Hold'em, vol 1.

Pot - Prize to be rewarded to the player with the best hand at the showdown, or the last player remaining (in the case of an uncalled bet).

PFR - (Pre-flop raise %) how often a player raises preflop per 100 hands

Raise - To increase the size of the bet required to stay in the hand. In order to raise, someone else must have placed the initial bet.

River - directly after turn betting concludes, the dealer exposes a single, final card, and betting ensues. This is the final round of betting.

Showdown - after all betting has concluded, if more than one player has claim to the pot (has not folded) the players must expose their cards to determine who has the best hand. The player with the best hand will win the entire pot in texas hold'em. If more than one player has equivalent high hands, then the pot is divided between them.

***Sklansky Bucks** - Sklansky bucks/dollars are theoretical money won or lost based on the Fundamental Theorem of Poker. The FTOP is discussed in The Theory of Poker.This concept is similar to EV.*

Small Blind or SB - a forced bet, typically half the size of the big blind. this player is usually 2nd to last acting preflop, but then first to act for the rest of the hand.

Turn - directly after the flop action concludes, the dealer exposes a single card, placed next to the flop. this is a community card, and another round of betting ensues.

Value bet - A bet placed because the player believes his hand is currently best, and we want the opponent with a second best hand to call.

VPIP% - (Voluntarily put in put %)how many hands player put chips in pot per 100 hands

WTSD% - (went to show down %) - if the player has seen the flop, how often does he get to the showdown per 100 hands

W$SD% - (won $ at show down %) how often the player has the best hand at showdown per 100 hands.