2018

A Blackjack Game and Simulation

A2 - COMPUTER SCIENCE COURSEWORK

Contents

Analysis

Background and Problem Definition

is in and has approximately 1300 students and 140 staff members. The college primarily teaches students aged 16-18, studying A-level education however it does offer adult education courses. a teacher at the college, runs a board games club on Friday lunchtimes. One of the most popular games played here is non-monetary version of blackjack, but recently the number of students attending has declined due to missing cards in the deck, and loss of chips. has asked me if it is possible to create a digital version of blackjack which saves the chip count of the player and allows them to resume the following week(s) with the same number of chips. He hopes this new system will bring back lost players to the club and then attract new members.

Since the game is non-monetary, players will have a total balance, given in say coins (¢) which will then be converted into chips. These chips will be what's bet, but the player can clearly see their overall balance without having to count the chips. Because of this, no money is involved as the game is purely for fun, therefore making it legal. Many students who attend the board games club study a variation of Mathematics. Therefore, they may be interested to see the probabilities of winning their hand as most people know; the odds are stacked against the player and in favour of the house (dealer). Therefore, having a "hints" function which allows the player to see the chance of them going bust would be helpful for new players to understand what their chances of winning their hand is.

has asked me to also make an auto run feature which will allow the program to run using Artificial Intelligence (AI) to make the turn decisions which will speed up the game. Players can view the AI as if it was their own player, and he also asked me to show the probability of winning/losing to the user, to help new players pick up the game.

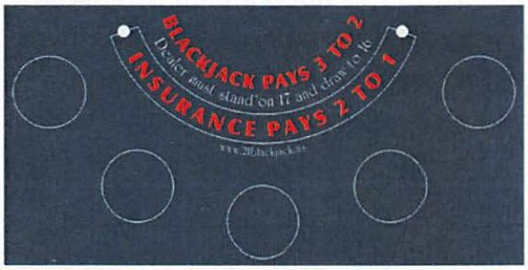
My project is aimed to create a way for the board games club members to play blackjack each week, maintaining their chips each time, and allowing new players to see the odds of them winning or losing. The project will also have the auto run feature to simulate hands played. I believe this program will appeal to existing players and attract new players due to the hand simulation and probability features allowing them to pick up the rules of the game quickly and the save feature allowing them to pick up where they left off the previous week.

I will be interviewing an active member of the club called to find more information about what my solution should include, and I will be using this interview to gather a collection of key objectives which I should aim to meet to make my new solution a success.

Description of the Current System

The current board games club is held on Friday lunchtimes starting at 12:40 and finishing at 13:40. Club members arrive and can select a game from the front of the room. They can then take their game to an available table and set it up. But, if there are not any free tables, they either; must wait for a game to become available or spectate an ongoing game. Typically, the students begin by setting up their game and then play the game until either someone wins, or the games club has ended. If they're playing blackjack, then one student is assigned to be the dealer for that game. The dealer is rotated on a weekly basis allowing all players to either play the game or play as the dealer. If the games club has ended, they must pack up the game and place it back at the front of the classroom. Since blackjack is a short game, games are never left mid-play when board games club

ends. The chip count of each player is recorded and written down on a piece of paper allowing players to maintain the same amount of chips throughout the club and the year.
The rules of blackjack are as follows; each player is dealt two cards, they are then given the choice between hitting another card, sticking with the cards they have, splitting their hand and finally doubling down. Hitting a card means that you are dealt another card. Sticking with your cards means that you are not dealt anymore cards and your turn is over. Splitting your hand means that your hand of two cards becomes two hands, with one card in each. You can then play these two hands separately, for example your first hand could go bust, but you can still play with your second hand. Splitting will also double your bet, so it is seen as risky. Finally, doubling down means that you double your initial bet, but you are only dealt one more card, essentially meaning you are dealt one card and then automatically you must stick.
The aim of the game is to either get a blackjack (21 card value after 2 cards being dealt) or to reach a 21-total card value. You can also win by having a higher card value than the dealer, or if the dealer goes bust. A card value is basically the sum of all your cards. The numerical cards have a card value of their numerical value, aces can equal 11 or 1, and the royal cards are equal to 10. Players can decide to stick if they are happy with their card value, and then risk their luck against the dealer. If the dealer gets a higher card value than the player, whilst remaining under 21, then the player will lose. The dealer however must **always** hit another card until their hand value is equal to or higher than 17. If a player has a card value greater than 21, they are considered as being "bust" which is just a word assigned to a player who has a card value higher than 21 and henceforth they are no longer in the current round.



Here is an example of a blackjack table, as you can see, there is a dedicated area for the dealer (Top of the table), and then 5 areas for up to 5 players to take part in the game.

Analysis of Similar Systems

Considering the problems which has mentioned, there are multiple blackjack systems online which would be solutions, but, these versions either require accounts, use monetary currency or lack features which were given to me in the problem definition. I'm going to look at these systems despite this to help me create a list of objectives for my solution.

The first example I have looked at is a website called **WizardOfOdds**, this site allows anyone to play blackjack against a dealer starting with 1000 currency.

This blackjack simulator allows the player to see hints, see their bet and their balance. This simulator is very basic as it doesn't allow you to save your balance for another time, and it also doesn't allow multiple players to play at the same time, it is limited to one player. This would present an issue in solving the current problem as Mike has indicated (In the background and product definition) that a multiplayer ability is crucial and the ability for users to save their balances over multiple days of playing was an important feature. This simulator does have a very easy to use UI with simple buttons located at the bottom and your chips in the bottom left.

Summary of System 1 (**WizardOfOdds**)

Good	**Bad**
• User-friendly system which anyone can understand • Hints system to allow new players to learn the game	• Only single player available • No saving of balances for later games
What the proposed solution will need: • The ability for multiple players to be at the table playing at the same time • The saving of balances between hands to allow players to pick up their balance each week • A hints system to allow new players to learn the game	

The next system which I've looked at is another site called **casinotop10**, this site allows you to play blackjack for free against an AI dealer similarly to **WizardOfOdds**.

Summary of System 2 (**WizardOfOdds**)

The UI for this system if much more complicated than the UI in the previous system. Something which stood out to me immediately was the constant advertisement of real-world gambling. This is featured when you first load the game up and you are greeted by a huge sidebar with links to 10 real gambling sites. Also, similarly to WizardOfOdds, casinotop10 only allows one player to play at once.

Free Blackjack

With its great payouts, easy-to implement strategy and simple rules, Blackjack has become one of the most popular casino games the world over. In fact, the classic card game has even been immortalized in many movies, such as '21' featuring Kevin Spacey and none other than the Golden Globe winner 'Rain Man'. If you've never played the game before and would like to experience all the glitz and glamour associated with playing the table game, you can experience the thrill of playing CasinoTop10's Free Blackjack game, on desktop or mobile, by clicking on the 'Try it for Free' button above.

ABOUT OUR FREE BLACKJACK GAME

CasinoTop10's Free Blackjack game serves to virtually transport you to one of the world's most notorious gambling destinations- Sin City. Yet, the sole difference between playing at a casino in the likes of one of the world's gambling meccas and playing the game at an online casino is that it won't cost you an arm

This is an issue as the promotion of gambling is illegal for minors. Then, once the game is loaded, there is a permanent header at the top of the game which advertises another site which features real world gambling. Other than the over-the-top advertisement of gambling, this is a very good simulation. The only thing missing from this simulation is a hints button to help new players. Another issue with this system is that you cannot save your balance for future games.

Good	Bad
• Detailed UI with multiple options	• Prominent advertisement of gambling sites • No multiplayer capabilities • Lack of a hints button • No saving features
What the proposed solution will need: • The ability for multiple users to play at the same time • A balance saving feature • No real-world gambling advertisements	

After completing these analysis, I have discovered more questions which I can ask and about their requirements for the new system which will outline in the interview section.

Problems with the Current System

Problems with the current system include the lack of decks which reduces the number of games that can be played. Since the college does not feel the board games club is important enough to receive funding, the games club cannot expand its member base by purchasing new decks of playing cards. Similarly to the lack of decks, some decks also miss certain cards, for example one deck is missing the Jack of Clubs. This reduces the odds-on players obtaining a blackjack as the Jack of Clubs could be required.

Secondly, since blackjack requires a dealer, one member of the board games club cannot play the game and must play as the dealer. As I previously mentioned, the dealer is rotated weekly, but this still means one player cannot partake in the actual game. There has also previously been issues with players "forgetting" to attend board games club when it's their turn to be the dealer. This causes a disruption because another player must step up and play as the dealer.

Thirdly, another issue with the current system is that there are a limited number of chips which limits how much money you can have and how many players can play. More chips could be ordered but chip sets tend to be very expensive. Both this issue, and my first issue link in with cost and that the board games club isn't a priority for the college hence why the club is under resourced.

Another issue with the current system is that if you want to play any of the games outside of board games club, you would need to find , and if he is teaching you cannot access the cards due to them being stored in his classroom. Therefore, students cannot play games outside of games club and say during their free periods. This also links with how students require free desk space to play the game. Students used to play in the library but recently they have been asked to leave as they're not studying.

Finally, blackjack is a very popular game at games club, this means it takes a while to complete a hand since you need to wait for everyone to either go bust or stick with their current hand. Similarly to my first point, more decks could be bought to allow the students to play more hands, but the board games club is at the bottom of the priority list for the college and relies on student bought decks.

Identification of the Users, Their Needs, and Acceptable Limitations.

Identification of the Prospective Users

Students will be the primary users of my program, there will not be any login as such, but students can load their profiles, but regardless of their profile they will have full access to all the features inside the program. Access to view the code will not be available as students may try to change code such as the bet pay-outs or change the dealer AI to bend the odds in their favour. However, the code will be available for read-only access on certain nights such as open days to allow people to get a grasp of what they will be doing during the Computer Science A-Level.

Identification of User Needs and Acceptable Limitations.

To allow me to understand my end user's requirements, I conducted an interview with an active member of the board games club called I asked questions about the current system as I feel he would be an accurate representation of the needs of all the members of the club, being the most active and involved member. The interview transcript is below:

Q1: *What is the current system for blackjack games?*

A1: Players will arrive during lunch on Fridays, they will be dealt cards from decks and are given whatever chips we have, this limits the number of games we can play due to lack of cards and chips.

Analysis of response: has indicated that there is difficulty with the current system when it comes to card and chip numbers.

Q2: *What problems are there with the current system?*

A2: There are three main issues I can think of. Firstly, time is wasted setting up the games at the start of the club, this time could be better spent playing more games. Secondly, many decks are missing cards, this means some games have limited outcomes, for example one deck is missing the Jack of Clubs, and meaning blackjack are less likely. Finally, there are a limited number of decks, therefore sometimes people cannot play and must spectate. Also, since one player needs to be the dealer, one person always misses out on playing the game.

Analysis of response: has indicated that there is time wasted at the start of the clubs setting up the games, there is also issues with cards missing and lack of cards altogether. He also said that one player always needs to play the dealer, I will ask more about this.

Q3: *You mentioned one person always needs to be the dealer, how do you think this could be tackled in the new solution.*

A3: Well, I believe if there was an AI which would play as the dealer every game, it would allow all students to play the game and hopefully meaning more students enjoy themselves at the club.

Analysis of response: told me that an AI dealer would improve the quality of the club.

- Key Objective 1: The dealer's duties will be performed by an AI dealer.

Q4: *Are there any essential features you would like to see in the finished system?*

A4: I'd like to see a visualization of how many chips you current have included with the exact value of those chips. This would add a realistic feel to the game, despite the fact that it is played digitally. The ability to save your chips for next week would be a nice feature allowing us to build up currency. I feel that having an "auto save" feature which would overwrite a user's save file immediately after a hand ends, hopefully stopping people "rage quitting" if they lose their bet. Having a simplistic GUI would be essential, allowing students to easily access the program and play it.

Analysis of response: has told me that a visual indicator of you balance in chip form would be an essential addition to the new system, I feel this would make a good objective. He has also stated that the ability to save your balances each week would be a necessary addition, I also feel this would make a good key objective. And finally, the addition of a simplistic UI, this is 100% key to making a good solution.

- Key Objective 2: Auto stacking of chips relevant to players balances.
- Key Objective 3: Auto save of player balances.
- Key Objective 4: Simplistic UI.

Q5: *Do you believe that a digital version of blackjack on a computer could be a solution to the issues you're facing with the current system?*

A5: Yes! I think that all the issues, notably the issues with a lack of chips and cards can be solved easily with a digital program.

Analysis of response: has clarified the importance of a program and has indicated that a program would solve the two biggest issues with the current system.

Q6: *How would you expect the current system to be improved by using a computerised system?*

A6: Firstly, the computer would take care of the role of the dealer, meaning that one extra player per game can take part. Secondly, the computer would handle the dealing, making sure that it is 100% fair to all players. Also, a computerised solution would allow a hints function to be included which would allow new players to learn the game as they play. A leaderboard function would also allow players to compete for the top spot, and this could all be kept by the computer.

Analysis of response: has detailed that a hints feature would be key to this program, I will ask for more details about this next. He has also stated that a leaderboard would be a good addition to add a competitive side to the club.

- Key Objective 5: A leaderboard showing the top 5 player balances.

Q7: *You mentioned about a hint solution to be added into the program, can you go into more detail about this?*

A7: Yes, I had the idea that a button inside the program could allow users to see information such as their hand total, their probability of winning, and the probability of losing. This addition would allow new players who maybe haven't played blackjack before to simply pick up the game in a matter of minutes!

Analysis of response: has indicated the importance of a hints feature here, I think this is an essential key objective.

- Key Objective 6: The ability to see your hand total.
- Key Objective 7: Users can see their probability of winning/losing and are guided on whether to stick or hit.

Q8: *Do you think there will be any limitations on this project?*

A8: Yes, I think that resources may be a limitation, I need this program done as soon as possible as the club loses members every week due to the issues I have discussed. This also is a large project, and I understand that this is your first big project, so you may also be a limitation to the speed of this project.

Analysis of response: answer clarifies the importance of this projects completion.

Q9: *How do you feel new players will feel about the system?*

A9: I think that new players may have some difficulty picking up the program as people tend to pick up the game quicker when seeing it played in front of them. But, I do feel that a simulate hands feature showing the AI playing the game and showing statistics for that hand would allow new players to understand the game.

Analysis of response: has indicated that a simulation feature may be a good addition, but, I feel that due to limitations, this may not necessarily be a key objective, as the game working is more important.

Q10: *Do you have any ideas about how the UI should look and how the simulation should be visualized?*

A10: I think a simplistic UI is essential to this program as I want the play time to be maximized and not have time wasted navigating through menus. I think focus should be on making the solution work, and not on providing a fancy UI. In terms of visualising the simulation, I'd like to see a visual animation of the cards being dealt. This would add a lot of depth to the program rather than having stationary dealing and cards teleporting themselves across the board.

Analysis of response: has indicated the focus should be on the creation of the solution and not on the UI of it. He has also mentioned the importance of visual dealing, I believe this to be a key objective.

- Key Objective 8: Cards are dealt visually to players.

Q11: *Do you feel the new system will bring back lost players?*

A11: Yes! The plan was to invest into buying more decks, but this new system is much cheaper and significantly better than what we had planned. I do feel that the guarantee of students playing a game with no issues such as missing cards and not enough decks will attract old and new members to the club.

Analysis of response: Once again here, (_ has indicated the importance of this new solution to bringing back lost players to the club.

To conclude, these are the key requirements for the new solution gathered from my interview with

- Key Objective 1: The dealer's duties will be performed by an AI dealer.
- Key Objective 2: Auto stacking of chips relevant to players balances.
- Key Objective 3: Auto save of player balances.
- Key Objective 4: Simplistic UI.
- Key Objective 5: A leaderboard showing the top 5 player balances.
- Key Objective 6: The ability to see your hand total.
- Key Objective 7: Users can see their probability of winning/losing and are guided on whether to stick or hit.
- Key Objective 8: Cards are dealt visually to players.

Acceptable Limitations

There are various limitations that I may face during the development and execution of the new system. These include;

Time Limitations:

has asked that I get the project completed before Christmas to allow time for the board games club to play the new system. But I do feel that this deadline is too soon therefore i may aim to finish the project in Febuary. However, if I finish the program in May, and college finishes in June, I would class the project as a failure as the new system will only be available to the board games club for less than a month.

My Own Programming Skill:

This program has some difficult aspects such as the AI, and the dealing animations. I have not programmed either of these therefore they may prove to be a challenge when I come to programming them. This may lead to very simple dealing animations or not very complex AI.

Lack of Computers with VB Installed:

There's only currently a select few computers at the college which have Visual Studio installed. This may mean that in my free time I need to either find an available computer, or I simply cannot work on my project if none are free. This linked with my first limitation (Time) may mean the project gets delayed depending on the availability of computers with Visual Studio installed inside of college.

Proposed Solution (With Chosen Language and Justification of Language)

After analysing the possible solutions already (WizzardOfOdds and casinotop10) and their advantages and disadvantages I have chosen to create my own solution in VB.NET. I have decided that creating a bespoke solution will benefit George and the rest of the blackjack club as I can create something which is tailored to their needs.

Use of VB.NET

To create my blackjack game, I have decided to write the program using the language VB.NET and to use the Integrated Development Environment (IDE) Visual Studio 2015. I have chosen to use this programming language as I have been programming using VB.NET and using Visual Studio 2015 for 2 years now, therefore time will not be wasted learning another language such as Java and learning to navigate a new IDE. VB.NET is an object oriented, event driven language which means that I can create classes which represent my players and AI, then also create functions which are executed upon button click events inside the program.

The Visual Studio 2015 IDE is a good IDE to use as it has built in indentation and colour coding. This makes formatting and structuring your code a lot easier to do, which will save valuable time. Also, VB.NET and Visual Studio have a large online presence for support, with multiple official pages from Microsoft and then multiple community driven forums available for support if needed.

The IDE has an easy to use interface which allows the simple addition of breakpoints throughout the code which will assist me with troubleshooting and debugging my code. These breakpoints allow me to step through my code line-by-line and check what all variable values equal, and closely debug my code for errors.

Programming using forms is very easy inside Visual Studio as it allows the simple creation of forms without lengthy coding and supports a drag and drop feature for elements such as buttons and picture boxes. This will help me create a simplistic UI like George requested.

I could create my solution through the Visual Studio Professional package IDE, but despite this package including more features compared to the Community package, it is very costly at approximately £500. This is an impractical price tag for software for my use case, therefore I will be using the Community edition of Visual Studio 2015 to keep the costs down.

Advantages of using VB	Disadvantages of using VB
Free IDE allows me to keep costs low.	Hard to transfer to other operating systems.
Supports object-oriented code.	Pro edition with more features is very expensive (£500).
Code is formatted for easy reading and editing thus saving me time.	Reliant on many version specific runtime DLLs which may change with new versions.
Large quantities of resources available online for support.	
Ability to use break points in the code, this allows me to find and fix errors simply.	
Easy creation of forms using drag and drop	
Event driven which allows me to create click events.	

Objectives

1. The form must load with a main menu, showing buttons to start a new game, load a game, simulate a game and view leaderboards.
2. If the "start new game" option is selected the game must ask the user for their name and store that name for later use.
3. If an invalid name is entered, you will ask for re-entry.
4. The board then loads a card table in a new form.
5. The chip count of the player is loaded in if they selected to load a game.
6. If the player didn't select to load a game, 1000¢ is loaded into the balance and a profile is created.
7. Players set a bet by selecting how much ¢ they want to bet.
8. Cards are dealt, 2 to the dealer, and 2 to each player.
9. If the player has a blackjack, they instantly receive triple their bet, and are out of the current hand.
10. One of the dealer's cards are shown to the players whilst one remains hidden.
11. Players are then asked whether they want to hit, stick, double or split, starting to the right of the dealer and ending to the left of the dealer.
12. There are hints which show the hand total of each player.
13. If a player goes bust, a visual indicator is shown.
14. If a player decides to stick, or goes bust, the game moves on to the next player.
15. If a player selects to hit, a card is dealt to them.
16. Once all players have had their turn, going bust or sticking, the dealer's final card is revealed therefore deciding if each player has lost or won.
17. The relevant bets are then paid out to each player, depending on whether they just won, or got a blackjack. (2x the bet for blackjack, 1.5x the bet for winning)
18. Player ¢ counts are then updated accordingly.
19. The game then automatically saves the updated chip counts for each player.
20. If the players click the "Yes" option in the "Return to Main Menu" choice box, they are redirected to the MainMenu form.
21. If the players click the "No" option in the "Return to Main Menu" choice box, they remain on the PlayingBoard form.
22. If the "Simulate Hands" button is clicked from the MainMenu, the user is asked how many simulated players they wish to have.
23. The PlayingBoard is loaded and then the number of players the user asked for are loaded.
24. Cards are dealt similarly to that of a normal game, except they are not playing, it's an AI.
25. The AI will look at the cards, if their cards have a low value and their bet it minimal, the AI will either hit another card, or double down, depending on the total of their hand.
26. If the AI player has a large hand value, then the AI will stick with their cards as they do not want to go bust.
27. If the AI has two of the same cards, it will always split its hand.
28. Whatever the decision for the AI, the player will be presented with justification as to why the AI made the decision it did.
29. There is a button allowing the user to decide when to stop simulating hands.
30. If the user selects "View leaderboards", all available user game saves are read, and compiled into ascending order.

31. The top 5 users are then presented to the user with their corresponding chip counts.

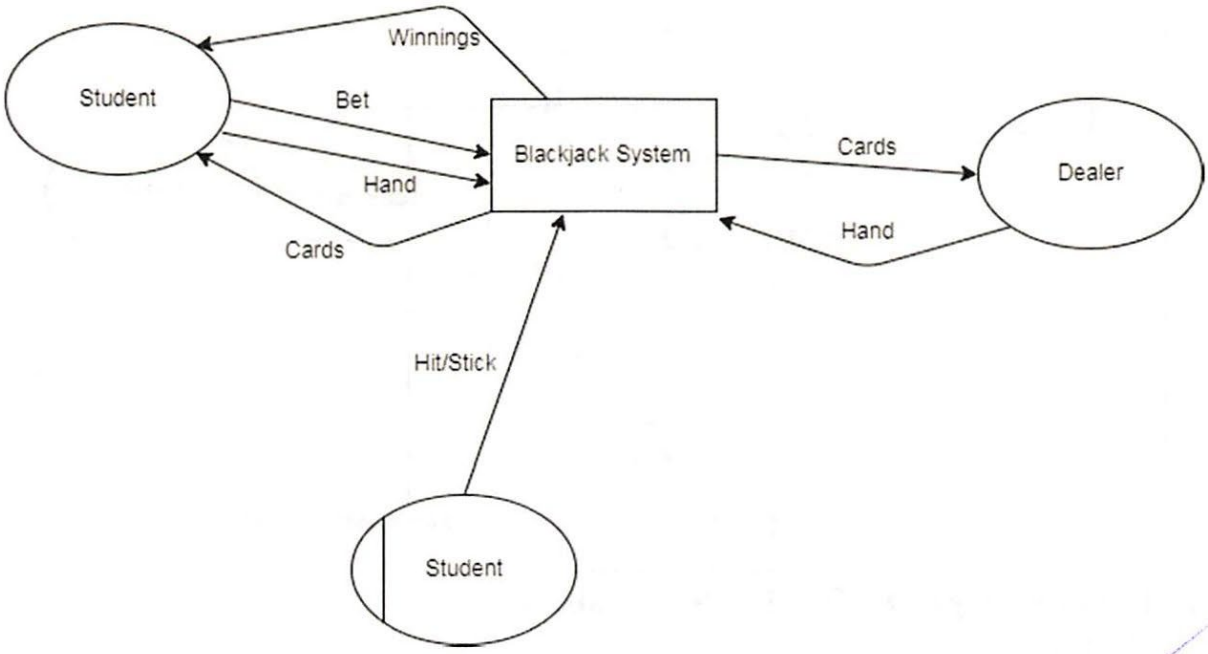
Data Dictionary

In the current system, there are small amounts of data stored. These include;

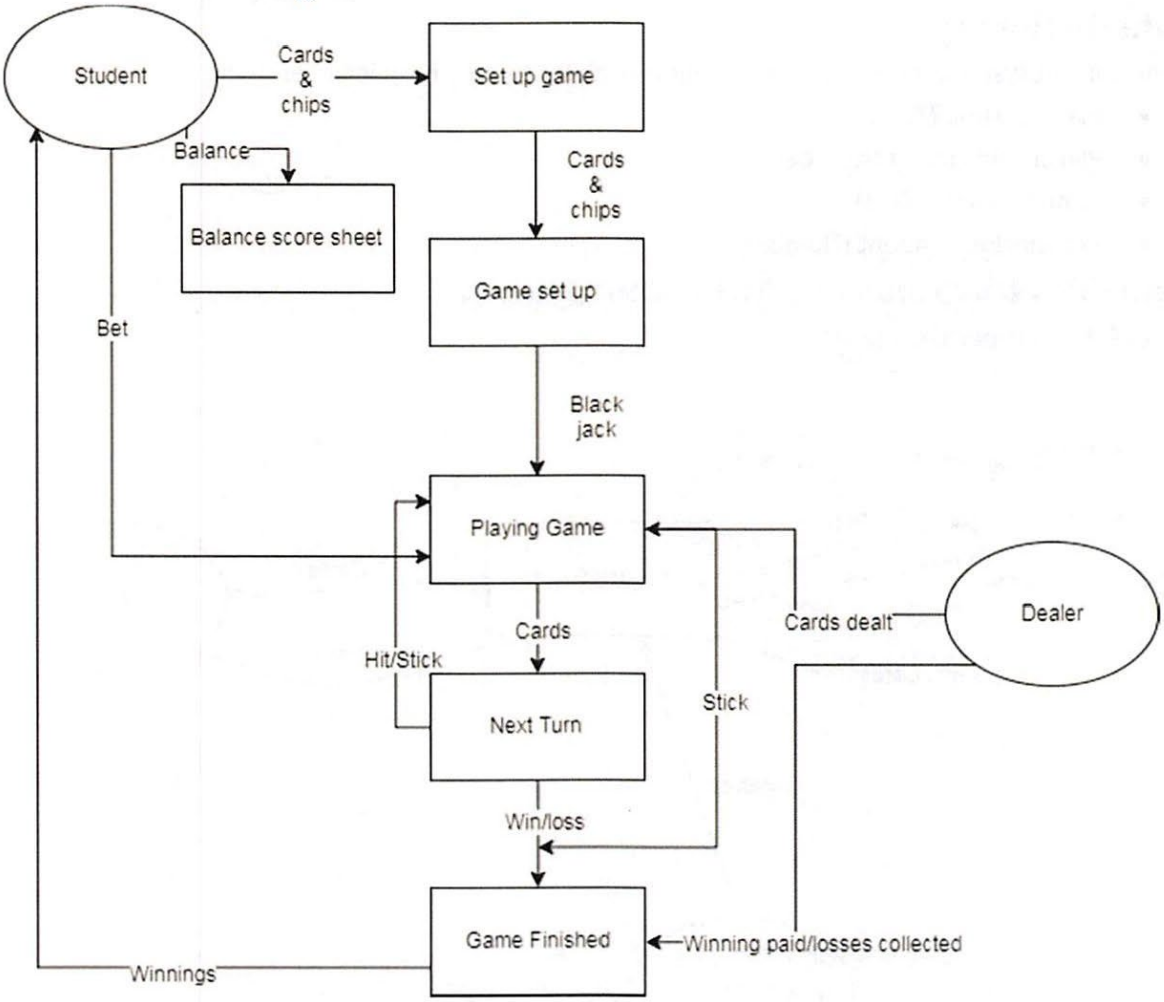
- Players name (Text)
- Players chip count (Number)
- Winners name (Text)
- Bet win/loss amount (Number)

Data Flow Diagrams for the Current System

Level 0 Data Flow Diagram:

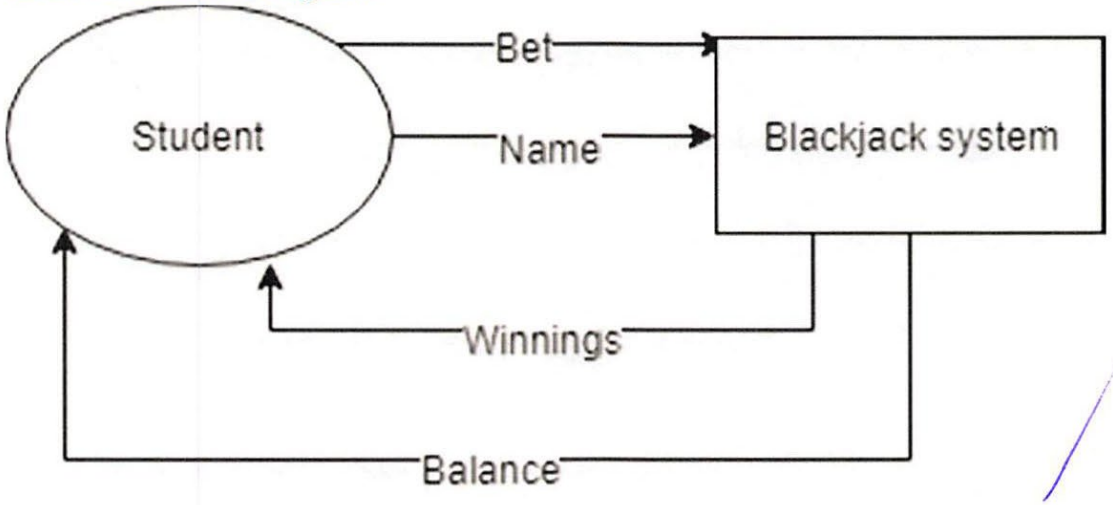


Level 1 Data Flow Diagram:

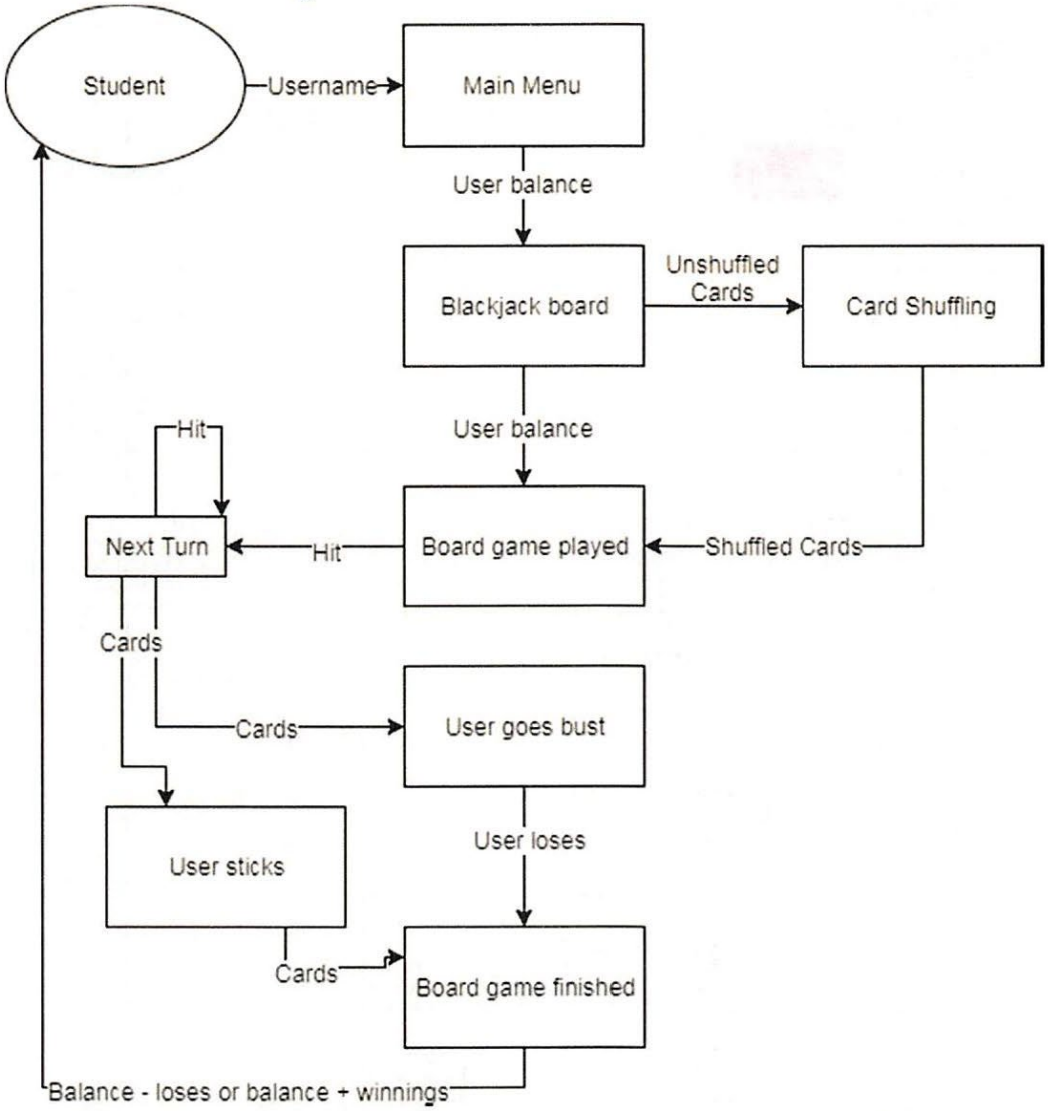


Data Flow Diagrams for the New System

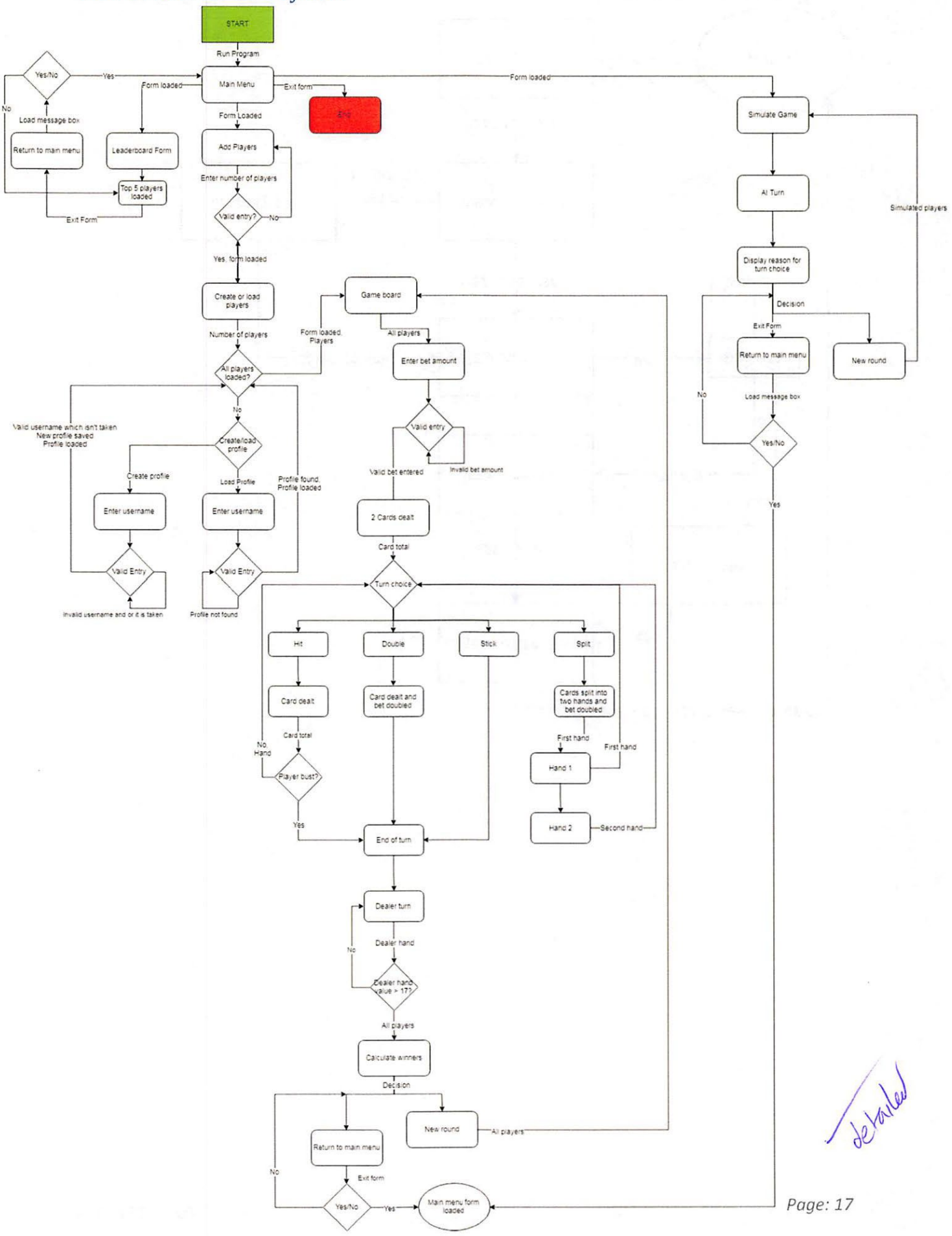
Level 0 Data Flow Diagram:



Level 1 Data Flow Diagram:



Flow of Data for New System



detailed

I used the circle labelled "Main menu form loaded" to indicate that the Yes output would direct the user to the box labelled Main Menu at the top of my flowchart and they would be able to decide which form to enter after (Leaderboard, Add Player, Simulate game ect). My program can only end when the user is viewing the Main Menu form and clicks the End button. If the user is inside any other form, they need to return to the main menu to exit.

Data Sources and Destinations

For the current system to function, certain data is required. This data includes;

- Player choice, whether they want to hit, stick, split or double after seeing their hand.
- Player bet, how much they want to bet on the current next hand.
- Player chip count, the amount of chips each player has currently (Their balance).

The system also outputs certain data such as;

- Winnings from bets, how much the player has won from their bet, and if they haven't won then winnings are 0.
- Winning players, which players have won.
- Updated chip counts after bets.

Data Volumes

Space Complexity

My system will store a large amount of data. I will try my best to keep this amount to a minimum, but certain aspects such as decks and hands are necessary.

Stored Data	Max Characters	Bytes Required
Player name (Max 4 players)	15 per player, 15 * 4 = 60	60 bytes
Deck (Array of size 52)	2 per card, 2 * 52 = 104	104 bytes
Bet amount	5 per player, 5 * 4 = 20	20 bytes
Winnings	15 per player, 5 * 15 = 60	60 bytes
Player hand (array size 2)	2 per card, 4 * 5 (extra for dealer) = 20	20 bytes

The total amount of bytes stored by my program is ~264 bytes. This value may be less as it has been calculated to include if all 4 players are playing the game at once, yet you can play with less. Most of this value is taken up by the deck of cards, but, I cannot try and reduce this value.

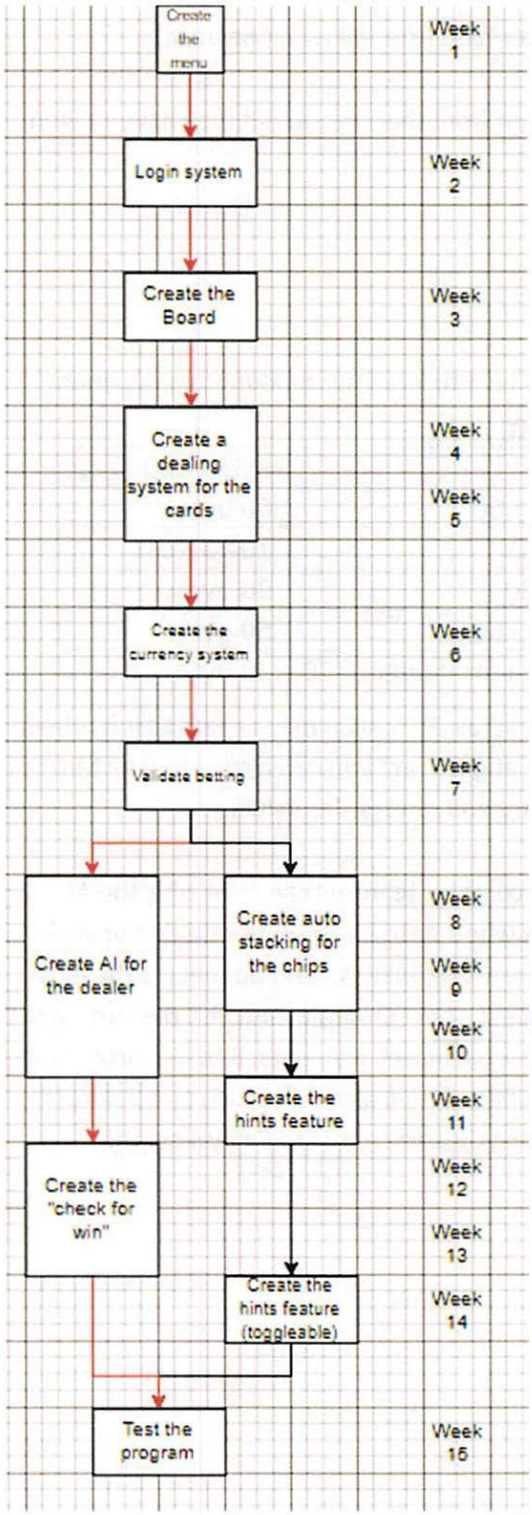
Time Complexity

My game doesn't have much time complexity. The most complex feature I can think of is the AI simulation. I think this because the AI will have to decide whether to hit or split or double or stick. This shouldn't take very long to calculate as there are only 4 possibilities, but I do feel it's the most complex part of my program. The only other complex aspect of my program could be the sorting of player balances and names to create a leaderboard. Player balanaces would need to be sorted whilst retaining the name associated with each balance. A modified sorting algorithm such as quick sort qould be required. This modified algorithm may take a long time to create and may slow the creation of the solution.

Design

Critical Path Diagram

This critical path shows that my project should be completed in approximately 15 weeks. If I follow the path shown, the project should be completed in 15 weeks, however some tasks may take less, or some take more time then I predicted, therefore this figure may be slightly higher or lower. I could decrease the amount of time needed to complete this project by using abstraction enabling me to re-use subroutines and functions for different things. For example, the check for win in blackjack should be able to be reused for each scenario.



My critical path is 15 weeks, but it misses out the auto stacking of the chips and the hits feature. This is because the hints feature and auto stacking chips are not necessities and the program will work without these features, therefore they are not critical.

System Design

Before creating my blackjack game, I will need to plan how details of the program will look, and how it will link to each form. This will mean I can quickly create my forms inside of Visual Studio without needing to plan it out as I go along.

The Main Menu has the option to either, add players to the game, start the game, exit the game, toggle hints, simulate a game, or view the leaderboards. If the AddPlayers button is selected, they are taken to the Add Players form. If they select the NewGame button, they are taken to a new form called the Playing Board. This handles all the gameplay of blackjack. If they select the exit game, the Main Menu screen will close therefore exiting the game. If they select the toggle hints button, hints will either be enabled/disabled depending on the previous state of the button. And finally, if they click on the ViewLeaderboard button, they will be taken to a separate sub which will show them the top 5 players names and balances.

The Add Players form will allow players to load their profile or create a new one. This will have a simple input box to enter their existing progile name or create a new one. There will be validation for this to make sure people can't create a new profile with an existing profiles name. If someone does try to do this, they will receive an error. Once this has been done, their information will either be loaded from the system, or a profile will be created. Creating a profile will involve creating a text document with their name and balance which defaults to 1000.

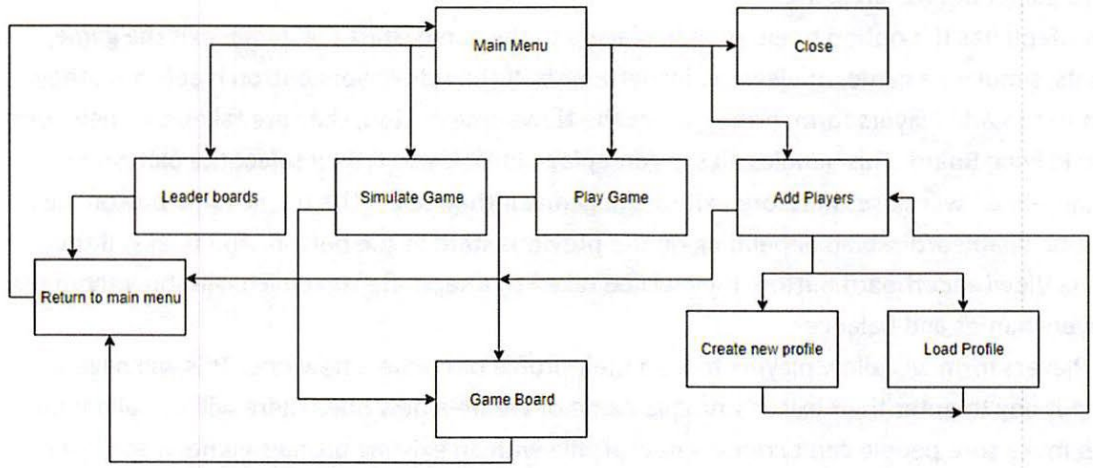
The Playing Board form will feature a background of a blackjack board, and the names of all the player which have been loaded in along with their corresponding balances. Then, cards will be dealt to each player going clockwise starting with player 1 and finishing with the dealer. After this, an input box will appear which will give the player to option of choosing their bet based on their cards. This input box will have validation to make sure that the player does not bet more than they have in their balance. Then, starting from player 1, each player will be greeted with 4 buttons, one for hit, one for stick, and then the split and double buttons when appropriate. They can click either of these buttons, for example hit, and they will be dealt another card to their hand. If they decide to click stick, their hand will remain the same and the game will move on and highlight the next player to the left of the previous player. If this becomes the dealer, the dealer's cards are then shown to the players and the winners are calculated. This form will also feature a visual indicator of the player's balances in the form of visual chip count. This count will be made up of picture boxes containing images of poker chips. These images will dynamically adjust themselves depending on the balances of the player at the current time. Once the winners have been selected, a button allowing a new hand to be played is shown, this button, if clicked, will restart the game and play another hand. If they do not want to play a new hand however, then an exit game button will be in the bottom right which will take them back to the Main Menu.

The Leaderboard form will be very basic. It will feature 5 lines of text; these lines feature the names and balances of the top 5 players in the game. This information will be loaded from a text document that stores the top 5 players. This is a very basic form, and the only button inside this form will be an exit button which will return the player to the Main Menu. The purpose of this form is to allow the players to see the top 5 players and compete to be featured on this leaderboard.

The Simulate Game form will be very similar to the playing board form, except there will be no input boxes and buttons. This form will deal cards to however many AI players the user selects and will chose a bet amount for each AI based on their cards and their balances. This bet will be placed, and then the AI will make hit/stick/double/split decisions based on their cards and the cards that the dealer has. This will all be done one-by-one for each AI starting from the far right and ending on the

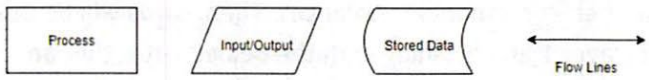
far left. Once each round has been completed, the player can choose whether to simulate another hand, or to exit. If they decide to exit, they will be taken to the MainMenu.

Modular System Design

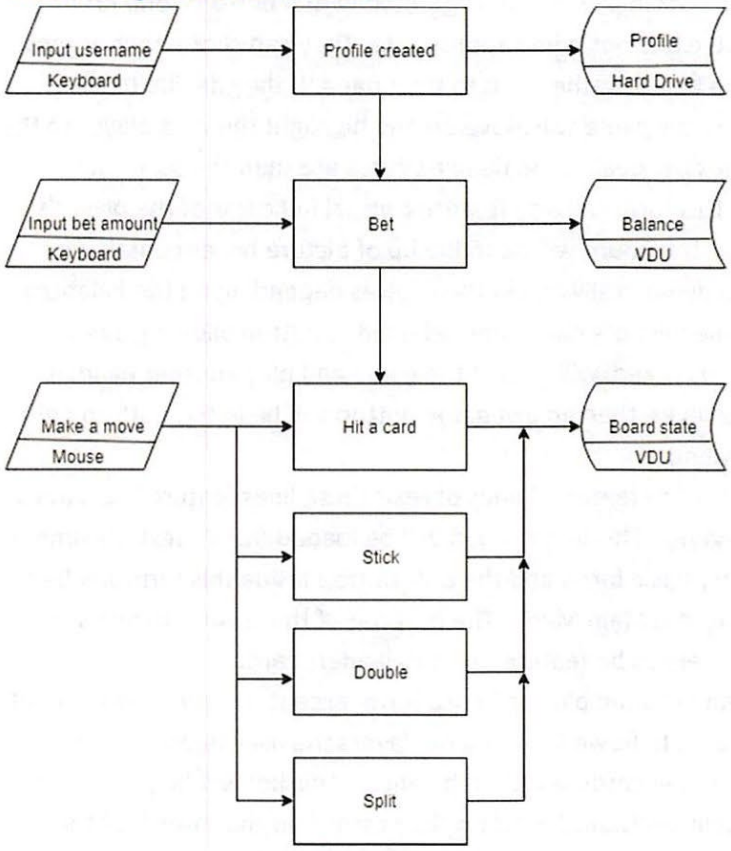


System Flow Charts

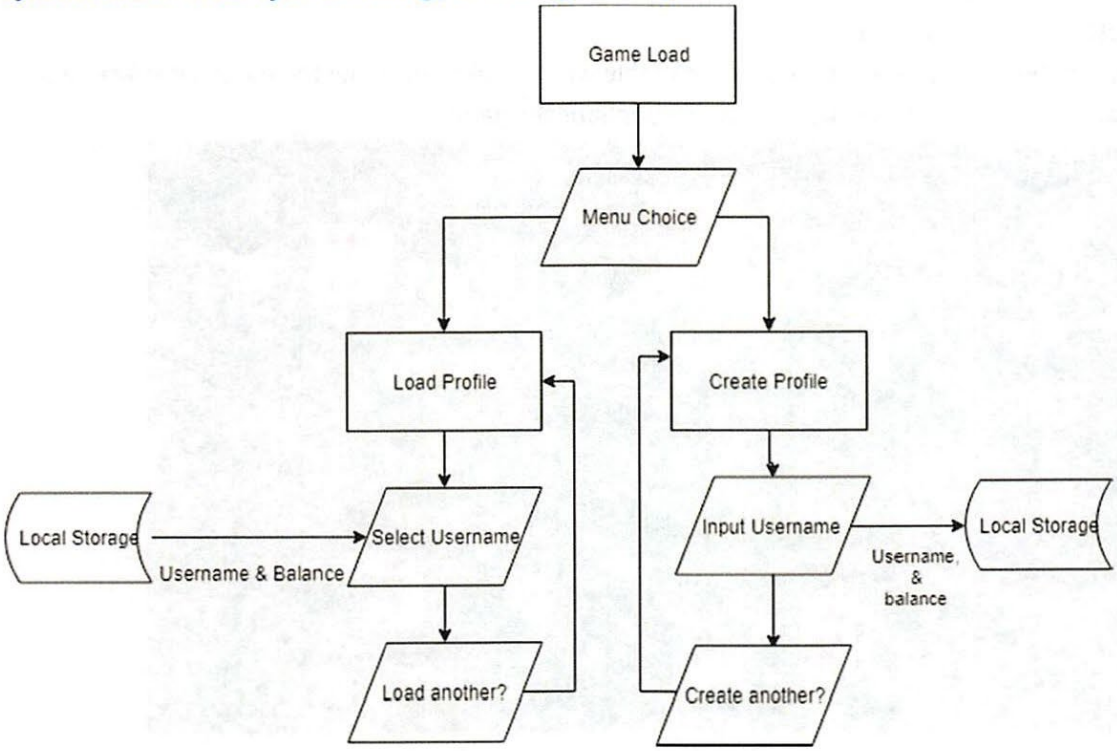
Key



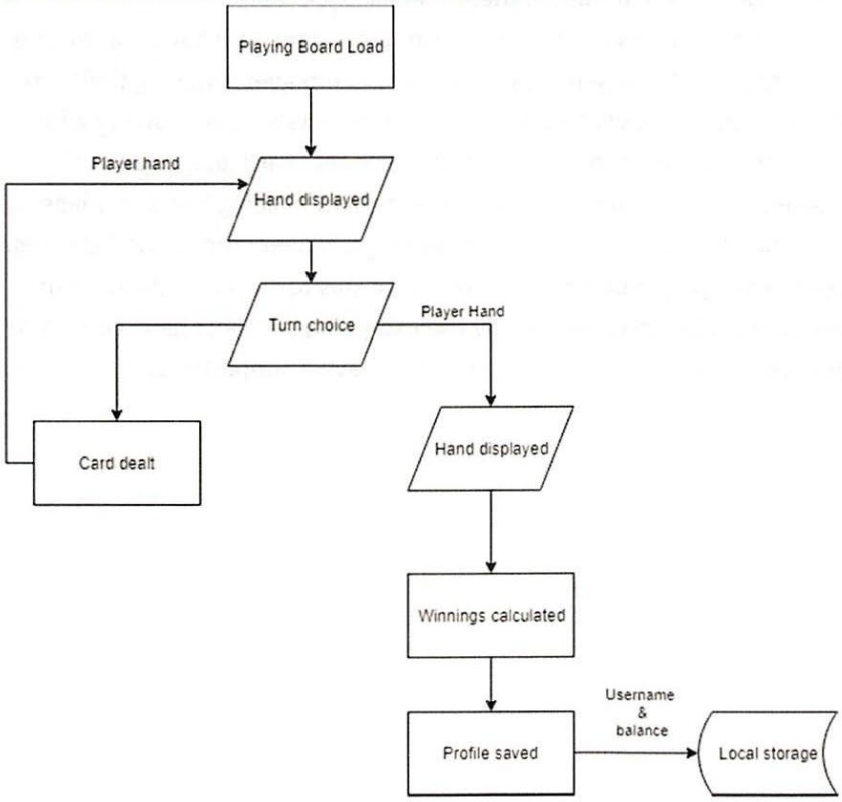
General System Flow Chart



System Flow Chart for Loading/Creating A Profile



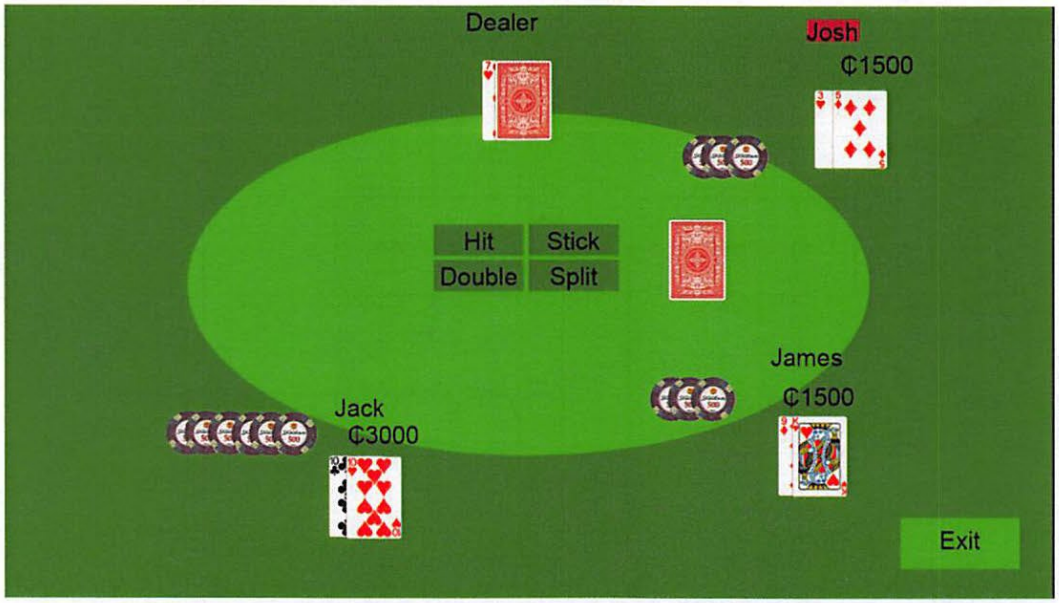
System Flow Chart for the Playing Board



Interface Design and Rationale

User Interface Design (HCI)

The diagram below shows what my blackjack table will look like, including the menu interfaces and some cards and chips which will be shown throughout the game.



This form features a predominantly green colour to match that of a real-life blackjack table. All my forms will follow this green/red colour scheme to match that of a blackjack table.

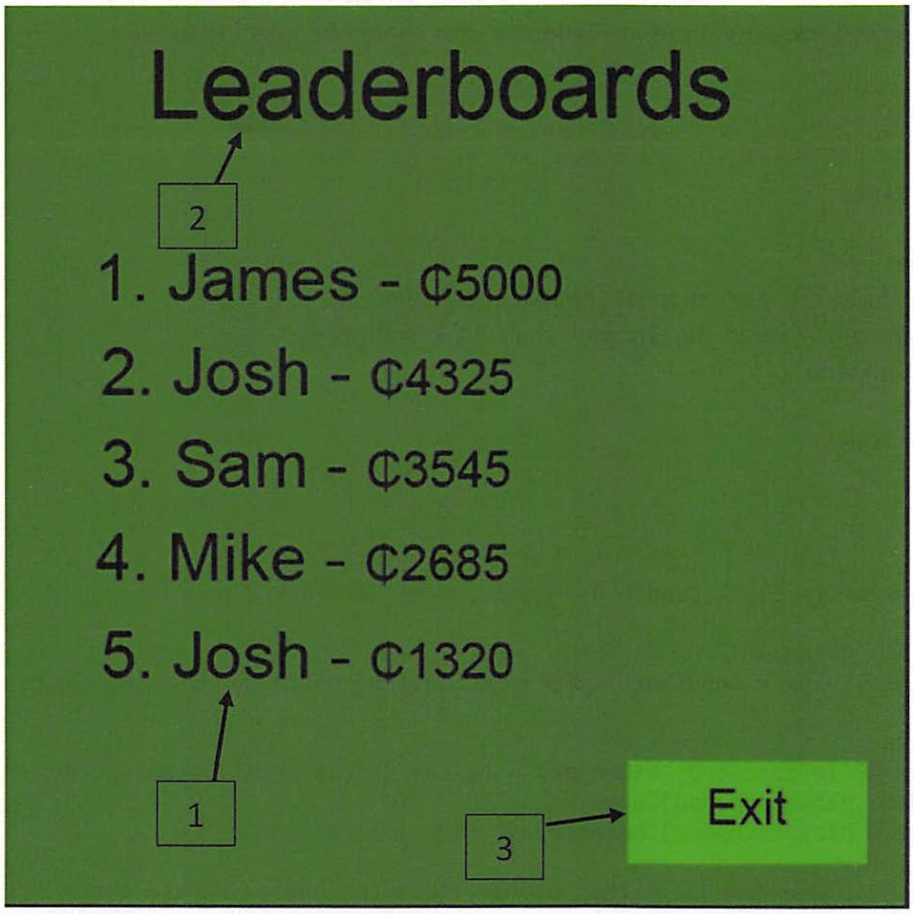
I have used a red highlighted label to represent which player's turn it currently is. I have chosen this as green/red are contrasting colours therefore making it very clear which player is having their turn.

I have decided to the Microsoft Sans Serif font, with a size of 14 to due to its easy readability which should maintain my aim of an easy to use and read UI. I decided to use real card images and chip images inside this form to add an element of realism to the game, instead of using cartoon images

When a game is started, there are no cards on the board, just the chips, player names and balances. The cards are then visually dealt to each player one-by-one. The UI of this form is very basic, with only 5 buttons, the hit/stick/double/split buttons, and finally the exit game button. I have chosen to do this because one of my objectives I gathered from George was to have a simplistic UI.

Output Design

The outputs for the program will be shown in either a label, a picture box or a text document. The labels will be in a contrasting colour to the background, mostly black. My form which will have the most outputs will be the leaderboard form.



1. The players name is in a larger font size than their balance allowing their name to stand out. Their name is also in a contrasting colour compared to the background for the same reason.
2. The title is in a significantly larger font size compared to the rest of the form. I have done this to allow the user to easily read what form it is they are looking at.
3. This is the exit button; this button will take the user back to the main menu. The background of this button contrasts any other colours featured in this form to make it stand out and allow the user to easily see how they can exit this form and return to the main menu.

Algorithms

Player Name Validation:

This algorithm will handle the validation for username input. It will do this by checking to see if a profile already exists, if the username is too long (max of 12 characters), and if the username is empty to begin with. If any of these conditions are True, the user should be asked to enter a username again.

Pseudocode Algorithm

```
Username as String
Valid as Boolean = False
INPUT Username
DO UNTIL Valid = True
      IF Length(Username) longer than 15 characters THEN
            SAY "Username cannot be longer than 12 characters"
            INPUT Username
      ELSE
            Valid = True
      END
LOOP
```

Coded Algorithm

```
    Private Sub btnNewProfile_Click(sender As Object, e As EventArgs) Handles
btnNewProfile.Click
        Dim strError As String = ""
        Dim username As String = InputBox("Enter your desired username here (No more
than 12 characters):")
        'Create profile
1       If (createNewProfile(password, username) = False) Or (saveProfile(password,
MainMenu.player(MainMenu.playersLoaded).name,
MainMenu.player(MainMenu.playersLoaded).balance, strError) = False) Or
(username.Length > 12) Then
1a          MsgBox("ERROR! Username is either already taken or empty. Please enter
again: " & strError)
        Else
            saveProfile(password, MainMenu.player(MainMenu.playersLoaded).name,
MainMenu.player(MainMenu.playersLoaded).balance, strError)
            displayProfile(password, MainMenu.player(MainMenu.playersLoaded).name,
MainMenu.player(MainMenu.playersLoaded).balance, strError)
        End If
    End Sub

Public Function createNewProfile(ByVal password As String, ByVal username As
String, Optional ByRef strError As String = "") As Boolean

        Dim validUsername As Boolean = False
        Try
2               If (username.Contains(" ")) Or (username.Length = 0) Or
(verifyUsername(username) = False) Then
                    Return False
                Else
                    validUsername = True
                    MainMenu.player(MainMenu.playersLoaded).name = username
                    MainMenu.player(MainMenu.playersLoaded).balance = 1000
                End If
            Return True
        Catch ex As Exception
            strError = ex.Message
```

```
            Return False
        End Try
    End Function

Function verifyUsername(ByVal username As String, Optional ByRef strError As
String = "") As Boolean

        Try
3           Dim tempHandle As IO.StreamReader = New IO.StreamReader("gamesave_" &
username & ".txt")
            tempHandle.Close()
            Return False
        Catch ex As Exception
            strError = ex.Message
            Return True
        End Try

    End Function
```

Description of the Algorithm

1 – This will go to the function createNewProfile and if it returns False then the user is presented with an error (1a), or if there is an error saving the profile, it will also return Fasle.

2 – This will check if the username contains any spaces, is empty, is longer than 12 characters or if the verifyUsername function returns a False value.

3 - This will try to create a StreamReader equal to a gamesave file with the username the user entered, if this gamesave exists then the verifyUsername function returns False (Meaning the username is taken as a gamesave already exists), and if the gamesave doesn't exist, the Try statement will reach the Exception and return True (Meaning the username does not exist as the StreamReader could not find a gamesave file with the username entered).

Test Data

1 – If you entered the username "Josh2", the createNewProfile function would be called with the password 123, and the username "Josh2".

2 – The IF statement will not be opened, and the vreifyUsername function will be called with username "Josh2".

3 – This will create a StreamReader for a file "gamesave_Josh2.txt". NOTE: This file doesn't exist, so the Try statement will break, causing it to return True, and set the strError parameter equal to the exception message. After this, the player name property will be assigned "Josh2", and the balance property will be assigned the default value of 1000.

Dealer Turn:

This algorithm will handle the AI for the dealer. If the card total of the dealer is less than 17, the dealer must hit for another card. As soon as the card total is higher than or equal to 17, the dealer must stick. However, the dealer does not need to take their turn if all the players are bust.

Pseudocode Algorithm

```
IF allPlayers are not Bust
      CardTotal as Integer
      CardTotal = Dealer card value
      IF CardTotal is less than 17 THEN
            DealCard(Dealer)
      ELSE
            Stick(Dealer)
      END
```

```
END
```

Coded Algorithm

```
    Sub dealerTurn()

        Dim allBust As Boolean = False
1       showDealerCard()

2       For i As Integer = 1 To MainMenu.numberOfPlayers
            If MainMenu.player(i).handValue < 21 Then
                allBust = False
            End If
        Next

3       If allBust = True Then
            If MainMenu.simulatedGame = False Then
                showDealerStatHints()
            End If
        Else
            Do
4               MainMenu.dealer.handValue = getDealerHandValue()
5               If MainMenu.dealer.handValue < 21 And MainMenu.dealer.handValue < 17
         Then
6                   dealCardToDealer()
                    If MainMenu.simulatedGame = False Then
7                       showDealerStatHints()
                    End If
8                   System.Threading.Thread.Sleep(100)
                End If

9           Loop Until MainMenu.dealer.handValue > 16
        End If

    End Sub
```

Description of the Algorithm

1 – This line will show the dealers second card because it is hidden from view of the players.

2 – This will loop through all the players, and if any player is not bust, that means all the players cannot be bust. This is done because if all players are bust, the dealer does not need to take a turn.

3 – If the allBust Boolean is True, then the dealer will not take their turn and the turn code will not be executed.

4 - This will use the getDealerHandValue function to calculate the dealer's hand value and set the handValue property to this value.

5 - If the dealer's handValue property is less than 21 but also less than 17, the IF statement is opened.

6 - The dealCardToDealer sub is then called which will deal a card to the dealer.

7 - The showDealerStatHints sub is then called which updates the hint labels for the dealer.

8 - The program will then wait 0.1 second to give the illusion of the dealer deciding their turn.

9 - Lines 1-6 will then repeat until the dealers handValue is above 16 (The dealer must hit until their hand value is 17 or higher.

Test Data

1 - For this example, I will pretend the dealer's hand consists of an 8 of spades and a 7 of diamonds. This line would cause the picturebox for the 7 of diamonds to be made visable again due to it being hidden.

2 – This will now loop for each player, for this example I am going to pretend that no players are bust.

3 – Due to allBust being equal to False, the If statement will be entered.
4 – The getDealerHandValue will return the value of 15, and then make the handValue property equal to 15.
5 – Because the dealer's handValue (15) is less than 17 but also less than 21, the If statement will be opened causing the dealCardToDealer subroutine to be called.
6 – This line would deal a card to the dealer both visually and in the dealer cards property.
7 – This line will cause the dealers stat labels to be updated again. These will show 15 + the card value of the card they were dealt.
8 – The dealer will then pause for 100 milliseconds to give the illusion of them making a choice.
9 – This will then loop until the dealer's hand value is greater than 16, because of this, the loop would only initiate if the dealer was dealt an ace card because that has a value of 1 and would total 16, and any other card would cause the loop to exit.

Shuffle Cards:

This algorithm will shuffle the deck of cards. It does this by picking a random card, and then placing that into a new array and then doing it again until all 52 cards have been placed into a new array.

Pseudocode Algorithm

```
RandomNumber as Integer
Cards(52,3) as String
ShuffledCards(52,3) as String
RemainingCards as Integer = 52
FOR a as Integer = 1 to 52
      RandomNumber = random number from RemainingCards
            FOR b as Integer = 1 to 3
                  ShuffledCards(a,b) = Cards(RandomNum,b)
            END
      DELETE RandomNum from RemainingCards
END
```

Coded Algorithm

```
Function shuffleCards(ByVal unshuffledCards As List(Of String)) As List(Of String)

1       Dim shuffledDeck As New List(Of String)
        Dim rand As Integer
        Dim count As Integer = 51
2       Randomize()
3       For i As Integer = 1 To 52
4           rand = Int(Rnd() * count)
5           shuffledDeck.Add(unshuffledCards(rand))
6           unshuffledCards.Remove(unshuffledCards(rand))
7           count -= 1
8           System.Threading.Thread.Sleep(50)
        Next
        Return shuffledDeck

    End Function
```

Description of the Algorithm

1 - This creates a new List of String called shuffledDeck
2 - This calls the built in Randomize function which ensures that the random number I create later on is entirely random
3 - This For loop will loop from 1 to 52, this is done so the shuffledDeck contains all 52 cards.

4 - This creates a rand variable equal to a random number from 0 to 51 (This number is used to choose which card from the unshuffled deck should be added to the shuffled deck)
5 - This line will add the card in the unshuffledCards list at position rand to the shuffledDeck
6 - This line will then remove the card which has just been added to the shuffledDeck from the unshuffledDeck
7 - This line will then reduce the count by 1 so the next random number is adjusted for the length of the unshuffledDeck list. This ensures that there isn't an "index was out of the bounds of the array" error by trying to add a card in a position that does not exist.
8 - This Sleep line is used to ensure that the random number generated remains random due to the number being generated based on the system clock.

Test Data

1 – This line will create an empty List of String called shuffledDeck
2 – The Randomize function will be called.
3 – This will loop from 1 to 52
4 – On this line, the variable rand will be made equal to the Int of rnd * count. For this example, I am going to pretend that count is equal to 30 and rnd is equal to 0.3. This would make rand equal to 9.
5 – This will then add the card in position 9 from unshuffledCards to the next available position inside the shuffledDeck list. For this example, let's pretend that the 9 of spades was there. The 9 of spades would then be placed into position 21 of the shuffledCards list.
6 – This line will then remove the 9 of spades from the unshuffledCards list to not duplicate the card in the next shuffle or in future shuffles.
7 – This will reduce the count by 1, so it will go from 30 to 29.
8 – This line will then sleep for 50 milliseconds to allow the next Rnd() to be different.

Auto Chip Stacking:

This algorithm will automatically sort the players balance into chips. For example, if the user had 51$, they would have 2 25$ chips and a 1$ chip. This would need to sort out how many of each chip the user has, display them an image of either 1$ chips, 5$ chips, 25$ chips, 100$ chips and 500$ chips.

Pseudocode Algorithm

```
IF balance / 500chip > 1 THEN
	500Count = floor.(balance/500chip)
	Remaining = balance - 500Count
END
100Count = chipStack(balance,remaining,100Chip)
25Count = chipStack(balance,remaining,25Chip)
5Count = chipStack(balance,remaining,5Chip)
1Count = chipStack(balance,remaining,1Chip)

FUNCTION chipStack(ByRef Balance, ByVal remaining, ByVal chipValue,ByVal)
chipCount as Integer
	If remaining /chipValue > 1 THEN
		chipCount = floor.(remaining/chipVlaue)
		Remaining = remaining - chipValue
	END
RETURN chipCount
END FUNCTION

displayChips(allChips)
```

Coded Algorithm

```
Sub updateChips()

        Dim balance As Integer
        Dim remainingBalance As Integer
        Dim chip500Count As Integer
        Dim chip100Count As Integer
        Dim chip25Count As Integer
        Dim chip10Count As Integer
        Dim chip5Count As Integer
        Dim player As player

1       Me.hideChipImages()

2           chip500Count = Math.Floor(balance / 500)
3            remainingBalance = balance Mod 500
4            If remainingBalance <> 0 Then
                 chip100Count = Math.Floor(remainingBalance / 100)
                 remainingBalance = remainingBalance Mod 100
                 If remainingBalance <> 0 Then
                     chip25Count = Math.Floor(remainingBalance / 25)
                     remainingBalance = remainingBalance Mod 25
                     If remainingBalance <> 0 Then
                         chip10Count = Math.Floor(remainingBalance / 10)
                         remainingBalance = remainingBalance Mod 10
                         If remainingBalance <> 0 Then
                             chip5Count = Math.Floor(remainingBalance / 5)
                             remainingBalance = remainingBalance Mod 5
                         End If
                     End If
                 End If
            End If

        Me.displayChipbox(500, chip500Count)
        Me.displayChipbox(100, chip100Count)
        Me.displayChipbox(25, chip25Count)
        Me.displayChipbox(10, chip10Count)
        Me.displayChipbox(5, chip5Count)

        If Settings.sounds = True Then
            My.Computer.Audio.Play(soundEffectPath & "Chips_Effect.wav")
        End If

    8   displayPlayerStats(Me)

    End Sub

    Sub displayChipbox(ByVal chipType As Integer, ByVal chipCount As Integer)

    5   If chipCount > 0 Then
    6       For i As Integer = 1 To chipCount
    7           returnChipBox(Me.number, chipType, i).Show()
            Next
        End If

    End Sub

    Sub displayPlayerStats(ByVal player As base_player)
```

```
        Dim nameLabel As Label = returnLabelName(player.number, "Name")
        Dim balanceLabel As Label = returnLabelName(player.number, "Balance")
        nameLabel.Text = player.name
        balanceLabel.Text = "¢" & player.balance

    End Sub

    Function returnChipBox(ByVal chipPlayer As Integer, ByVal chipType As Integer,
ByVal chipNumber As Integer) As PictureBox

        For Each item In PlayingBoard.Controls
            If item.name = "pbP" & chipPlayer & "_" & chipType & "_" & chipNumber Then
                returnChipBox = item
            End If
        Next

    End Function
```

Description of the Algorithm

1 -This sub will hide all the chip images,
2 - This will make the 500Count (Used to store the number of 500 chips the current player has) by dividing the player balance by 500. The Math.Floor function is performed to always round this number down.
3 - This line will make the remainingBalance variable equal to the player's balance MOD the chip number. Eg if the player had 501 balance, this will make remainingBalance = 1
4 - This If statement will check if the ChipCount is greater than 0, meaning the player has at least 1 of that chip.
5 - If the if Statement goes through, the FOR loop will then loop for each ChipCount which means the player has at least 1 of that chip.
6 - This will then return the chip box (A picture box) for the player by using their number, the value of the chip, and the number of chips already made visible.
7 – This line will loop from 1 to however many of that chip the player has, in order to display them all.
8 - This Sub will update the visual labels for the player balances.

Test Data

1 – This will call the hideCardImages subroutine from the base_player class.
2 – For this example, I am going to pretend that player 2 has a balance of 1050. This will make chip500Count equal to the Floor of 1050 / 500 which is 2.
3 – Remaining balance will be made equal to 1050 MOD 500 which is 50.
NOTE – Steps 2 and 3 will repeat for 100, 25, 10 and 5. This will create int25Count equal to 2 to make up the final 50 value.
4 – This line will then enter the If statement as chip500Count is greater than 0 (it is 2).
5 – This will loop from 1 to 2 as int500Count is equal to 2.
6 – This will return the chip box for player 2's first 500 chip. (Its player 2 because k is equal to 2). This PictureBox is then made visible.
7 – The FOR statement will loop 2 times for the 500 chip due to the chipCount being 2.
8 – This final line will update the player balance labels.

Get Player Card Value:

This algorithm will get the value of the player's hand. E.g. if the player had a 10 of diamonds and a 3 of spades, the algorithm will return an integer of value 13.

Pseudocode Algorithm

```
cardOneValue  as Integer
cardTwoValue as Integer
cardOneValue = integer(playerOne.cardOne)
cardTwoValue = integer(playerOne.cardTwo)
OUTPUT cardOneValue + cardTwoValue
```

Coded Algorithm

```
Function getPlayerHandValue(ByRef player As player) As Integer

        Dim cardType As Integer
        Dim currentCard As String

        player.valuedCount = 1
1       If player.valuedCount <> player.dealtCards + 1 Then
2           For i As Integer = player.valuedCount To (player.cards.Count - 1)
3               currentCard = player.cards(i)
4               If currentCard.Length = 2 Then
5                   If IsNumeric(currentCard.Substring(0, 1)) = False Then
                        Select Case currentCard.Substring(0, 1)
                            Case "A"
6                               cardType = 11
                                player.hasAces = True
                            Case "J", "Q", "K"
7                               cardType = 10
                        End Select
                    Else
                        cardType = currentCard.Substring(0, 1)
                    End If
                Else
                    cardType = currentCard.Substring(0, 2)
                End If
                If IsNumeric(cardType) = False Then
                    Select Case cardType
                        Case "A"
                            cardType = 11
                        Case "J", "Q", "K"
                            cardType = 10
                    End Select
                End If
                player.valuedCount += 1
8a              getPlayerHandValue += cardType
8b          Next
        End If

9       Return getPlayerHandValue

    End Function
```

Description of the Algorithm

1 - This IF statement will check that the player's valuedCount property is not equal to the number of cards they have been dealt + 1, this would mean that the players hand has already been valued, so the function does not need to be executed again.

2 - This will loop the function for however many cards haven't been valued. This is done to stop the Function valuing the same card every time it is called.

3 - This will make the currentCard variable equal to the players first card that hasn't been valued. Identified by i.
4 - This IF statement will check if the length of the currentCard variable = 2, if it does, this means it is any card apart from a 10. This is because a standard card is 2 characters long, eg 6C = 6 of clubs. Whereas a 10 is 2 characters long, e.g. 10C = 10 of clubs.
5 - This will check if the first character of the currentCard is numeric. If it is, that means the card is any card from 1-9. The only cards that it cannot be are any of the royal cards, or an ace. IT will then make cardType equal to the first character of the card name.
6/7 - If the card is not numerical, the code moves onto this line. It will then check if the card begins with either an A, in that case cardType is made equal to 11, due to aces being 11 (NOTE: There is a separate subroutine for making an ace equal to 1), it will then see if the first letter equals either "J", "Q", or "K". In this case, cardType is made equal to 10.
8a – This line will make the Function equal to the cardType value.
8b - It will then repeat steps 1/7 for anymore card that have not been added to the players handValue.
9 - The Function will then return the integer value which it calculated.

Test Data

For this example, I will pretend that player 2 has been passed into this Function. Player 2's cards List consists of (,8C,JS), meaning the 8 of Clubs and Jack of Spades (NOTE: The first value of a players cards list is empty intentionally). Player 2's valuedCount is equal to 1. Player 2 has a dealtCards value of 2.

1 – This will check if Player 2 has already had all their cards valued. Player 2 has a valuedCount of 1, and a dealtCards value of 2. 2+1=3, and 1<> 3. Because of this, the If statement will be opened.
2 – This line will loop from 1 to 2 because the Player 2 has a valuedCount of 1, and Player 2's cards list has a count of 3 (empty space,8C,JS). 3 – 1 = 2.
3 – This will set the currentCard variable equal to Player 2's card in position i which is equal to 1. This card is 8C. This will make currentCard equal to 8C.
4 – This will check if the length of currentCard is equal to 2, the length of "8C" is 2 therefore this if statement will be opened.
5 – This will check if the first character of currentCard ("8") is numeric, meaning it is a number. This is true as 8 is a number, therefore the If statement will not be opened, and the else case will be opened. This will make cardType equal to the first character of currentCard, which is 8 in this case.
8a – This will make the Function equal to itself + cardType. In this case, the Function will currently be made equal to 8.
8b – The For loop will then loop for i equal to 2 this time. This will be its last loop as it loops from 1 to 2.
3 - currentCard will be made equal to Player 2's second card this time ("JC").
4 – currentCard is of length 2, therefore this if statement will be opened.
5 – The first character of currentCard ("J") is not numeric, therefore this If statement will be opened.
6 – This select case statement will open the case of the first character of currentCard("J"), this will open the Case "J", "Q", "K".
7 – This line will make cardType equal to 10, because a royal card has a card value of 10.
8a – This will now add 10 to the Function value, this will make the function equal to 18 now.
8b – The For loop will no longer loop due to it being of value 2, which was its end value.
9 – This will then return the integer value of the Function, which was 18.

Creating A Split

This algorithm will see if the player has two cards of the same value, e.g. the 6 of clubs and the 6 of hearts. If this is True, a built-in subroutine inside the Player class will create two new hands for the player and create all the appropriate variables required to deal to the two new Lists separately. Then, a subroutine will be called which will visually separate the hands, moving the players' second card bellow their first, giving the visual effecct that there are two separate hands.

Pseudocode Algorithm

```
If Cards(1) = Cards(2) THEN
       NEW handOne = Cards(1)
       NEW handTwo = Cards(2)
       Player.split = TRUE
       CLEAR Cards
       Player.balance - player.bet
       Player.bet *= 2
END IF
handTwo(1).LocationY = handOne(1).LocationY-10
```

Coded Algorithm

```
Sub checkForSplit(ByVal player As player)

1       Select Case player.cards(1).Length
2           Case 2
3               If player.cards(1).Substring(0, 1) = player.cards(2).Substring(0, 1)
And (player.balance - player.betAmount > 0) Then
                    player.hasSplit = True
                End If
4           Case 3
                Select Case player.cards(2).Length
                    Case 3
                        If player.cards(2).Length = 3 Then
                            player.hasSplit = True
                        End If
                End Select
        End Select

    End Sub
```

Once the game loads, the player will be asked if they want to split as part of the startGame sub, this uses the following code:

```
If currentPlayer.hasSplit = True Then
5                   Select Case MsgBox(currentPlayer.name & " would you like to split
your hand?", MsgBoxStyle.YesNo, "Split")
6                       Case MsgBoxResult.Yes
7                           currentPlayer.doneSplit = True
                            currentPlayer.createSplit()
                            PlayingBoard.btnHit.Show()
                            PlayingBoard.btnStick.Show()
                            'FIRST SPLIT HAND
8                           Do Until currentPlayer.finishedTurnArrayOne = True
                                Application.DoEvents()
                            Loop
                            If currentPlayer.hasAces = True And
getSplitHandValueWithAces(currentPlayer, 1) > 21 Then
                                currentPlayer.finishedTurnArrayOne = False
                                Do Until currentPlayer.finishedTurnArrayOne = True
                                    Application.DoEvents()
                                Loop
                            End If
```

```
9                                   currentPlayer.changeSplit()
                                    'SECOND SPLIT HAND
                                    Do Until currentPlayer.finishedTurnArrayTwo = True
                                        Application.DoEvents()
                                    Loop
                                    If currentPlayer.hasAces = True And
getSplitHandValueWithAces(currentPlayer, 2) > 21 Then
                                        currentPlayer.finishedTurnArrayTwo = False
                                        Do Until currentPlayer.finishedTurnArrayTwo = True
                                            Application.DoEvents()
                                        Loop
                                    End If
                                    PlayingBoard.btnDouble.Hide()
                                    Case Else
                        End Select

Public Sub createSplit()

10a         splitSetOne = New List(Of String)
10b         splitSetTwo = New List(Of String)
11a         Me.splitCardsOne.Add("")
11b         Me.splitCardsTwo.Add("")
12          Me.arrayToDealTo = 1

13a         Me.splitCardsOne.Add(Me.cards(1))
13b         Me.splitCardsTwo.Add(Me.cards(2))
            Me.dealtCardsArrayOne = 2
            Me.dealtCardsArrayTwo = 2
14          Me.cards.Clear()

15          Me.balance -= Me.betAmount
16          Me.betAmount *= 2
17          updateChips()

18          splitHandsVisually(Me)

        End Sub
Public Sub splitHandsVisually(ByVal player As player)

19          For i As Integer = 1 To 6
20              moveCardY(returnCardBox(player.splitCardsTwo(1)), 5)
                Application.DoEvents()
21              System.Threading.Thread.Sleep(10)
            Next
22          For i As Integer = 1 To 4
23              moveCardX(returnCardBox(player.splitCardsTwo(1)), -3.125)
                Application.DoEvents()
                System.Threading.Thread.Sleep(10)
            Next

        End Sub
```

Description of the Algorithm

1 - This will perform a Select Case on the length of the players' first card.
2 – If the length of the players first card is two, this means that the player as any card, other than a 10.
3 – This line will check if the first character of the players' first card is the same as the first characters in the players' second card. If this is the case, then the players' hasSplit property will be set to True.
4 – If the players' first card is of length 3, this means that it is a 10, therefore all we need to check is that the players' second card is a 10, as that is the only possible option, we do not need to compare the substring of the first and second cards like for Case 2.
5 – This line will perform a Select Case on the output of the message box asking the player if they want to split. This message box has the YesNo style applied, therefore there will be 2 buttons, Yes and No presented to the user.
6 – If the user clicks the Yes button to split their hand, the following code will be executed.
7a – This line will change the doneSplit property to True, this is used when dealing or checking if the player has Split their hand later in the code.
7b – This line will call the createSplit subroutine which creates the two separate hands (More detail on that sub further down).
8 – This Loop will repeat until the player has finished their turn for their first-hand list.
9 – This will call the changeSplit subroutine which will simply flip the arrayToDealTo property which is used when dealing to the separate split hands. This is done because the arrayToDealTo property is used when dealing to indicate which split array should be dealt to.
10a – This line will create a new list called splitSetOne which will hold the cards for their first split hand.
10b – This will create a new list called splitSetTwo which will hold the cards for their second split hand.
11a/b – This will add an empty card (""), which just performs as a filler, so that I can easily address each card as n, instead of n-1 later in the code due to lists starting from 0 and not 1.
12 – This line will set the arrayToDealTo property to 1, is used in dealing to the hand lists and is set to 1 because the first-hand list is the first to be dealt to.
13a/b – This line(s) will add the players' first card to their splitCardsOne list, and then add their second card to the splitCardsTwo list. Their hand has now been fully split into two separate hands.
14 – This line will clear the contents of their cards array, there is no specific reason to do this however.
15 – This line will reduce the players' balance by their bet, this is because when you split your hands, you also double your bet, and therefore the player must have their bet taken away from their balance once more.
16 – Now, the players' betAmount property will be doubled due to them splitting their hand and increasing their bet by two.
17 – Due to the player increasing their bet by double, the players chip counts and visual bet labels must be updated accordingly.
18 – This line will then call the splitHandsVisually subroutine which will visually separate the players' cards into two new hands. This is because up to this point, the splitting of the players' cards has only been code-based, nothing has happened visually.
19 – This will loop 6 times, only for the reason because I want to move the second card 30 units down and moving the card 5 units down 6 times seems to look smooth.
20 – This will call the moveCardY subroutine which will move the card given along the y-axis however many specified units, in this case 5.
21 – The system will then pause for 10 milliseconds to slow the animation of the card moving.
22 – This will now loop 4 times to move the card on the x-axis. This value isn't as random as the y-axis value because the second card is offset by default to allow the player to see their first card

despite their second being dealt. Therefore, I will need to reduce the x value by this offset, which is 12.5.
23 – Similarly to line 23, but, this will call the moveCardX subroutine which will move the card on the x-axis specified units.

Test Data

For this example, I am going to pretend that Player 1 has been passed into this subroutine. Player 1 has a balance of 1500, and they bet a value of 150. Player 1's card list consists of (6C,6D), meaning their cards are the 6 of Clubs and the 6 of Diamonds. Player 1's name is Gavin.
1 – This line will select case the length of Gavin's first card (6C), in this case, this will be 2.
2 – This will open the case 2 line due to "6C" being of length 2.
3 – This line will check if the first character of Gavin's first card ("6") is equal to the first character of his second card ("6"), and then that his balance (1500) – his bet (150), is greater than 0. In this case, 6 = 6 and 1500 – 150 = 1350 which is > than 0. This would cause the IF statement to open. This would make Gavin's hasSplit property equal to True.
4 – N/A
5 – Because Gavin's hasSplit property is equal to True, he is asked if he wants to Split his hand.
6/7 – If Gavin choses the "Yes" button, his doneSplit property will be made equal to True, and his createSplit subroutine will be executed.
10a-b – This will create two new Lists to store Gavin's new cards (one for each hand).
11a-b – This will add an empty value to each of Gavin's new lists to make referring to his cards easier.
12 – This will set Gavin's arrayToDealTo propert equal to 1, indicating that his first array is his current "active" one, and is the one which the game should deal to and value.
13a-b – These lines will add Gavin's first card to his first split array, in this case the 6 of Clubs (6C), and then his second card to his second split array, in this case the 6 of Diamonds (6D).
14 – This line will clear Gavin's cards array.
15 – This will then reduce Gavin's balance (1500) by his bet (150) resulting in a new balance of 1350.
16 – Gavin's betAmount property is then doubled due to splitting causing a double in your bet.
17 – Due to Gavin's betAmount and balance properties changing, his chips need updating therefore the updateChips subroutine is called.
18 – This line will call the splitHandsVisually subroutine with Gavin's player object as the parameter.
19 – This will loop with i equal to 1-6.
20 – This line will move the result of the returnCardBox Function with Gavin's second split card lists first card. This PictureBox is then moved 5 units down on the y-axis.
21 – The system will then pause for 10 milliseconds to slow the movement.
22 – This will loop with i equal to 1-4.
23 – Similarly line 20, this will move the PictureBox of Gavin's second split cards first card, but this time it will move it -3.125 units on the x-axis. These movements will result in Gavin's second card being moved directly below his first.
8 – This will loop until Gavin's finishedTurnArrayOne property is equal to True, meaning he has finished his turn.
9 – Once Gavin has had his turn for his first split hand, meaning he has either stuck or gone bust, the changeSplit subroutine is called which simply alternated the arrayToDealTo property, in this case it will be made equal to 2 instead of 1. This will then deal cards to Gavin's second split list instead of his first. Therefore allowing Gavin to have a "second" turn because of his 2 hands.

Class Definitions

The *base_player* class is a base class of which all my classes inherit. This base class contains all the properties which every player requires. These include properties such as a name and a balance. (I've listed these properties bellow). The class also contains subroutines which all classes require such as the ability to display your chips and reset the player for the next round.

The *player* class is a class assigned to any real players (Actual humans playing the game). This class inherits the *base_player* class but has the addition of properties to allow the splitting of cards. The *player* class also has the addition of subroutines to handle the splitting of cards, and a New subroutine which is called every time a player object is created, which is used to "setup" the class before the game starts.

The *simulated_Player* class is assigned to any simulated players created within the simulate game feature of the program. This class is almost the same as the *base_player class*, but it uses a modified New subroutine which similarly to the *player* class, is used to "setupt" the object when it is created.

The *AI_Player* class is assigned to the dealer, and only the dealer. Similarly to the *simulated_Player* class, the *AI_Player* class is very similar to the *base_player* class, but only has the addition of the New subroutine and an overritten reset soubroutine. The New subroutine sets the valuedCount property equal to 1, which is not done in the base class. The reset subroutine is overwritten from the *base_player's* reset subroutine due to the addition of setting the dealtCards property equal to 1, instead of the 0 inside the *base_player* subroutine. This is done due to the differences with the dealing to the dealer and to the player.

Properties

Class Name	Properties	Function
base_player	name	Stores a String value which represents the players' name.
	number	Stores an integer value which indicates the players' number. Eg the first player would have a number of 1.
	handValue	Stores an Integer value of the value of the players' hand.
	balance	Stores an Integer value which indicates the current players' balance.
	valuedCount	Will store the number of cards the player or dealer has had valued.
	dealtCards	Will store the number of cards the player or dealer has been dealt.
	cards()	An array which stores the cards of the player or dealer.
	hasBlackjack	Stores a Boolean value indicating if a player has a blackjack.
	betAmount	Stores the amount the player has decided to bet
	finishedTurn	Stores a Boolean value indicating when the player has finished their turn.
player	arrayToDealTo	Used to store which cards array to deal to if the player has split their cards.
	dealtCardsArrayOne	An integer which stores how many cards have been dealt to the first split hand.

	dealtCardsArrayTwo	An integer which stores how many cards have been dealt to the second split hand.
	handValueArrayOne	An integer which stores the hand value for the players' first split hand.
	handValueArrayTwo	An integer which stores the hand value for the players' second split hand.
	valuedCountArrayOne	An integer which stores how many cards have been valued from the players' first split hand.
	valuedCountArrayTwo	An integer which stores how many cards have been valued from the players' second split hand.
	finishedTurnArrayOne	A Boolean value which indicates whether the player has finished their turn for their first split hand.
	finishedTurnArrayTwo	A Boolean value which indicates whether the player has finished their turn for their second split hand.
	doneSplit	A Boolean value which indicates whether the player has chosen to split or not.
	hasSplit	A Boolean value which indicates whether the player has a split-able hand.
	hasAce	Stores a Boolean value which indicated whether the player has any ace cards in their hand
	hasSplit	Stores a Boolean value indicating if the player can split
	splitCardsOne	Stores the cards for their first split hand
	splitCardsTwo	Stores the cards for their second split hand

Subroutines and Functions

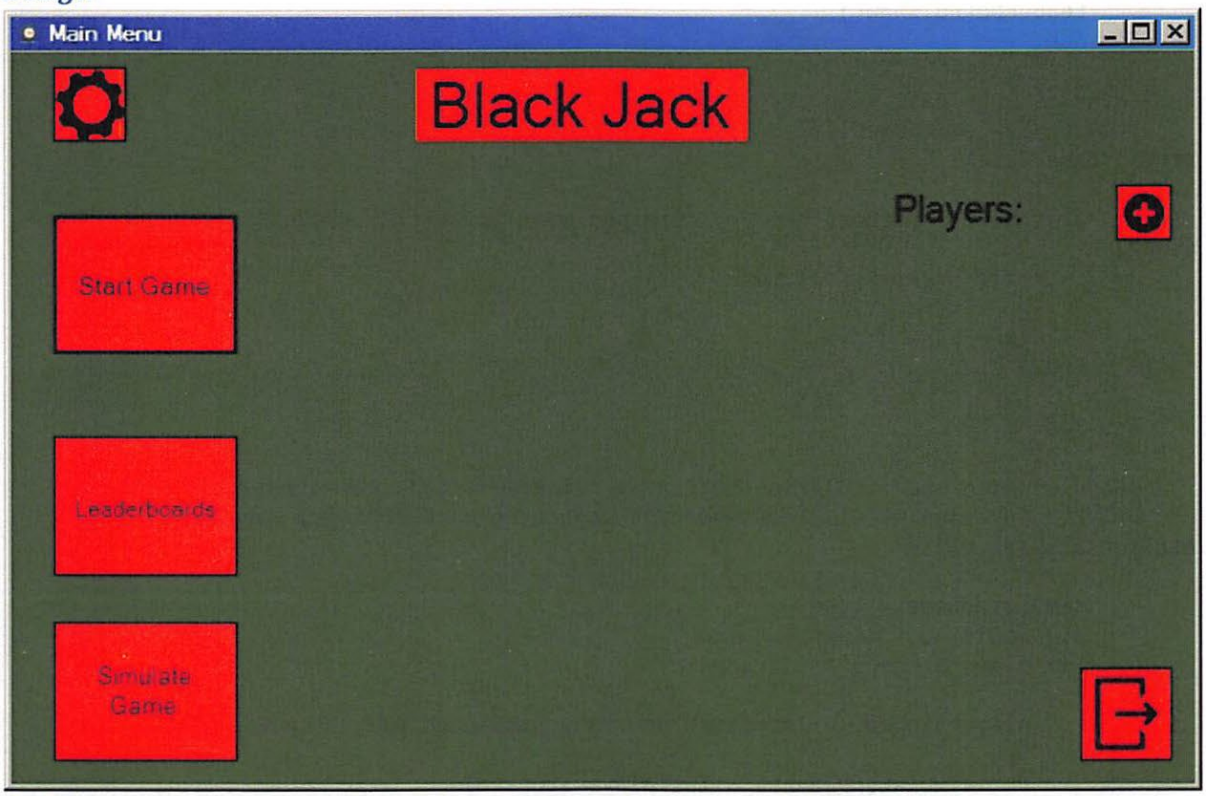
Class Name	Subroutine	Function
base_player	reset	Used when resetting the board. Will reset the card list, will reset the valuedCount and reset the dealtCards.
	updateChips	Will look at the players balance and calculate how many of each chip they would have, and then display these chips accordingly.
	displayChipBox	Will display a chip picturebox for a player when given a chipType (eg 500 chip), and the chip number (eg the third 500 chip) as parameters.
	hideChipImages	This subroutine will hide all the chip pictureboxes for whatever player calls the subroutine.
player	New	Creates the card array and fills the card at position 0 with "". Sets finishedTurn variable to false. Used inside the turn system. Sets hasSplit to False, used when checking for a split.
	createSplit	Creates two new card arrays and splits their cards into those arrays. Sets the hasSplit value to True and sets the arrayToDealTo to 1, which is used when dealing to the split hands.
	changeSplit	Changes the arrayToDealTo variable to the other array. Used when dealing to the split cards.

AI_player	(overrides) reset	Has the addition of setting the dealer's dealtCards equal to 1.
	New	Creates the card array and fills the card at position 0 with "". Sets the valuedCount to 1. Used when valuing the dealer's cards.
simulated_Player	New	Creates the card array and fills the card at position 0 with "". Sets finishedTurn variable to false. Used inside the turn system.

Implementation

Main Menu Form (MainMenu.vb)

Design



Code

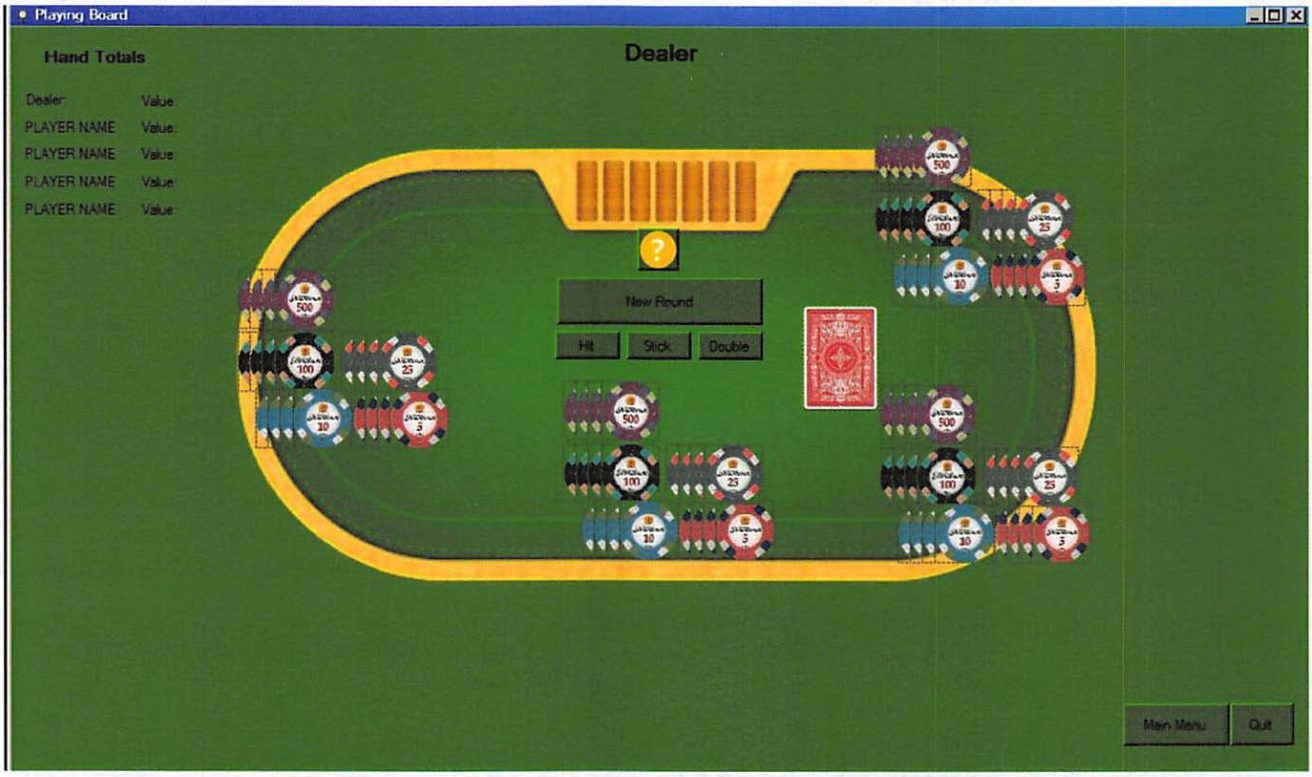
```
Public Class MainMenu

    Public Shared player(4) As player
    Public Shared simPlayer(4) As simulated_Player
    Public Shared dealer As AI_Player
    Public Shared numberOfPlayers As Integer = 0
    Public Shared playersLoaded As Integer = 0
    Public Shared simulatedPlayers As Integer
    Public Shared simulatedGame As Boolean = False

    'This subroutine is called when the mainmenu is loaded and it simply sets up the
settings option
    Private Sub MainMenu_Load(sender As Object, e As EventArgs) Handles Me.Load
        Settings.cbHints.Checked = True
        Settings.cbSounds.Checked = False
    End Sub

    'Handles when the start game button is clicked. (Moves to the playing board)
    'Verifies that all players have been loaded.
```

```
    Private Sub startGameClicked(sender As Object, e As EventArgs) Handles
btnStartGame.Click
        If playersLoaded = numberOfPlayers And playersLoaded <> 0 Then
            PlayingBoard.Show()
            Me.Hide()
            GameMethods.startGame()
        Else
            MsgBox("ERROR! Add all players or create a new game.")
        End If
    End Sub

    'Handles when the leaderboards button is clicked. (Moves to the leaderboards)
    Private Sub leaderboardsClicked(sender As Object, e As EventArgs) Handles
btnLeaderboards.Click
        Me.Hide()
        Leaderboard.Show()
    End Sub

    'Handles when exit game button is clicked (closes the game)
    Private Sub exitGameClicked(sender As Object, e As EventArgs) Handles
btnExitGame.Click

        Select Case MsgBox("Are you sure you want to quit?", MsgBoxStyle.YesNo, "Exit
Game")
            Case MsgBoxResult.Yes
                End
            Case Else
        End Select

    End Sub

    'Handles when the simulation button is clicked. (Loads the simulation form)
    Private Sub simulateClicked(sender As Object, e As EventArgs) Handles
btnSimulateGame.Click

        simulatedGame = True
        Dim valid As Boolean = False
        Dim playerStorage As String
        Try
            playerStorage = InputBox("How many simulated players would you like?")
            Try
                simulatedPlayers = Int(playerStorage)
            Catch ex As Exception
                MsgBox("You must enter a numeric value between 1 and 4!")
            End Try
            If playerStorage > 4 Or playerStorage < 1 Then
                MsgBox("You cannot have more than 4 simulated players!")
            Else
                valid = True
            End If
        Catch ex As Exception
            MsgBox("You cannot have 0 simulated players!")
        End Try
        If valid = True Then
#Disable Warning BC42104 ' Variable is used before it has been assigned a value
            simulatedPlayers = playerStorage
#Enable Warning BC42104 ' Variable is used before it has been assigned a value

            For i As Integer = 1 To simulatedPlayers
                simPlayer(i) = New simulated_Player
                simPlayer(i).number = i
            Next
```

```
            dealer = New AI_Player
            Me.Hide()
            PlayingBoard.Show()
            GameMethods.startSimulation()
        End If


    End Sub

    'Handles when the add player button is clicked. (Goes to the add player sub)
    'Also checks that all the players havent already been added.
    Private Sub addPlayerClicked(sender As Object, e As EventArgs) Handles
btnAddPlayer.Click

        If numberOfPlayers = 0 Then
            Dim numberStorage As String
            Try
                numberStorage = InputBox("How many players are there? (1-4)")
                If numberStorage < 1 Or numberStorage > 4 Then
                    MsgBox("Number of players cannot be less than 0 or greater than
4!")
                    Exit Sub
                End If
                numberOfPlayers = numberStorage
                lblPlayers.Text = "Players " & playersLoaded & "/" & numberOfPlayers
                For i As Integer = 1 To numberOfPlayers
                    player(i) = New player
                Next
                dealer = New AI_Player
            Catch ex As Exception
                MsgBox("Number of players cannot be 0!")
            End Try
        End If
        If playersLoaded <> numberOfPlayers Then
            playersLoaded += 1
            lblPlayers.Text = "Players " & playersLoaded & "/" & numberOfPlayers
            AddPlayer.Show()
            Me.Hide()
        ElseIf numberOfPlayers = 0 Then
            MsgBox("Start a new game before adding players! Click the New Game
button!")
        Else
            MsgBox("Max players reached!")
        End If
    End Sub

    'Handles when the player clicks the settings icon.
    'Will load the settings form and hide the mainmenu
    Private Sub btnSettings_Click(sender As Object, e As EventArgs) Handles
btnSettings.Click
        Settings.Show()
        Me.Hide()
  End Sub
```

Playing Board Form (PlayingBoard.vb)

Design



Code

```
Public Class PlayingBoard

    'Handles the main menu button click event.
    'Checks that every player has had their turn first.
    Private Sub btnMainMenu_Click(sender As Object, e As EventArgs) Handles
btnMainMenu.Click

        If MainMenu.simulatedGame = False Then
            If roundFinished = True Then
                MainMenu.Show()
                Me.Close()
            Else
                MsgBox("You cannot return to the main menu until everyone has finished
their turn.")
            End If
        Else
            MainMenu.Show()
            Me.Close()
        End If

    End Sub

    'If the quit button is clicked, the player will be asked if they are sure they
want to quit, instead of the program being closed instantly.
    Private Sub btnQuit_Click(sender As Object, e As EventArgs) Handles btnQuit.Click

        Select Case MsgBox("Are you sure you want to quit?", MsgBoxStyle.YesNo, "Exit
Game")
            Case MsgBoxResult.Yes
                End
            Case Else
        End Select
```

```
    End Sub

    'This code is executed when the form is loaded, it checks if the user has selected
to play a simulated game, if tey have it will assign names and balances to the AI
players.
    'If the user has not decided to play a simulated game, the game will assign player
numbers, and then check the status of the settings form.
    Private Sub PlayingBoard_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        If MainMenu.simulatedGame = False Then
            For i As Integer = 1 To MainMenu.numberOfPlayers
                MainMenu.player(i).number = i
            Next
            btnNewRound.Hide()
        Else
            Select Case MainMenu.simulatedPlayers
                Case 1
                    MainMenu.simPlayer(1).name = "Josh"
                    MainMenu.simPlayer(1).balance = "575"
                Case 2
                    MainMenu.simPlayer(1).name = "Josh"
                    MainMenu.simPlayer(1).balance = "575"
                    MainMenu.simPlayer(2).name = "James"
                    MainMenu.simPlayer(2).balance = "355"
                Case 3
                    MainMenu.simPlayer(1).name = "Josh"
                    MainMenu.simPlayer(1).balance = "575"
                    MainMenu.simPlayer(2).name = "James"
                    MainMenu.simPlayer(2).balance = "355"
                    MainMenu.simPlayer(3).name = "Sam"
                    MainMenu.simPlayer(3).balance = "585"
                Case 4
                    MainMenu.simPlayer(1).name = "Josh"
                    MainMenu.simPlayer(1).balance = "575"
                    MainMenu.simPlayer(2).name = "James"
                    MainMenu.simPlayer(2).balance = "355"
                    MainMenu.simPlayer(3).name = "Sam"
                    MainMenu.simPlayer(3).balance = "585"
                    MainMenu.simPlayer(4).name = "Mike"
                    MainMenu.simPlayer(4).balance = "1005"
            End Select
        End If

        btnShowHint.Hide()
        If Settings.cbHints.Checked = True Then
            Settings.hints = True
        Else
            lblHandTotalTitle.Hide()
        End If
        If Settings.cbSounds.Checked = True Then
            Settings.sounds = True
        End If

    End Sub

    'If the player clicks the new round button and the game is a simulated game, then
the game will reset accoring to the code inside the simulation reset sub routines,
    'However if it is a regular game, the code for the regular reset sub routines will
be executed.
```

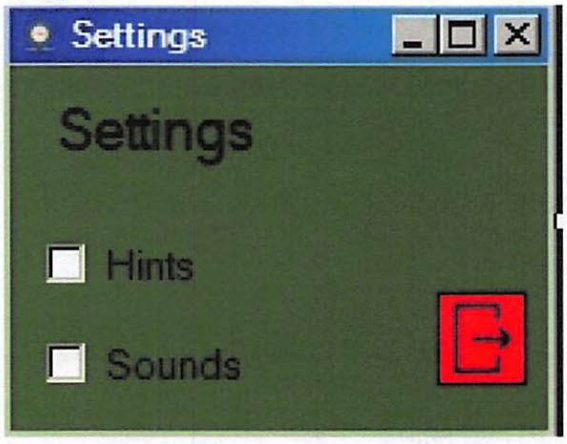
```
    Private Sub btnNewRound_Click(sender As Object, e As EventArgs) Handles
btnNewRound.Click
        If MainMenu.simulatedGame = False Then
            If roundFinished = True Then
                btnNewRound.Hide()
                resetBoard()
                startGame()
            End If
        Else
            btnNewRound.Hide()
            resetSimulationBoard()
            startSimulation()
        End If

    End Sub

'If the user clicks the hit button, it first checks that all the cards have been dealt
to all players
    'Then, it will see if the player has split their deck, this is done because if
they have, the dealing is done differently.
    'If they have split, they will be dealt a card based on their current "split"
hand,
    'Whereas if they havent split, they will be dealt a card to their hand and have
their hing labels updated
    'Regardless of whether that have split or not the game will then check to see if
they are bust or not, if they are, their turn is ended with the finishedTurn property.
    Private Sub btnHit_Click(sender As Object, e As EventArgs) Handles btnHit.Click

        If finishedDealing = True Then
            If currentPlayer.doneSplit = True Then
                If currentPlayer.finishedTurnArrayOne = False Then
                    dealCardToSplitArray(currentPlayer)
                    If getSplitHandValueWithAces(currentPlayer, 1) < 22 And
currentPlayer.hasAces = True Then
                        currentPlayer.finishedTurnArrayOne = False
                    ElseIf getSplitHandValue(currentPlayer, 1) > 21 And
currentPlayer.hasAces = False Then
                        currentPlayer.finishedTurnArrayOne = True
                        MsgBox("Unlucky! " & currentPlayer.name & " Your first hand is
now bust!")
                    ElseIf currentPlayer.hasAces = True And
getSplitHandValueWithAces(currentPlayer, 1) > 21 And getSplitHandValue(currentPlayer,
1) > 21 Then
                        currentPlayer.finishedTurnArrayOne = True
                        MsgBox("Unlucky! " & currentPlayer.name & " Your first hand is
now bust!")
                    ElseIf getSplitHandValue(currentPlayer, 1) = 21 Then
                        currentPlayer.finishedTurnArrayOne = True
                    End If
                Else
                    dealCardToSplitArray(currentPlayer)
                    If getSplitHandValueWithAces(currentPlayer, 2) < 22 And
currentPlayer.hasAces = True Then
                        currentPlayer.finishedTurnArrayTwo = False
                    ElseIf getSplitHandValue(currentPlayer, 2) > 21 And
currentPlayer.hasAces = False Then
                        currentPlayer.finishedTurnArrayTwo = True
                        MsgBox("Unlucky! " & currentPlayer.name & " Your second hand
is now bust!")
                    ElseIf currentPlayer.hasAces = True And
getSplitHandValueWithAces(currentPlayer, 2) > 21 And getSplitHandValue(currentPlayer,
2) > 21 Then
```

```
                        currentPlayer.finishedTurnArrayTwo = True
                        MsgBox("Unlucky! " & currentPlayer.name & " Your second hand
is now bust!")
                    ElseIf getSplitHandValue(currentPlayer, 2) = 21 Then
                        currentPlayer.finishedTurnArrayTwo = True
                    End If
                End If
            Else
                dealCardToPlayer(currentPlayer)
                showPlayerStatHints()
                If getPlayerHandValueWithAces(currentPlayer) < 22 And
currentPlayer.hasAces = True Then
                    currentPlayer.finishedTurn = False
                ElseIf getPlayerHandValue(currentPlayer) > 21 And
currentPlayer.hasAces = False Then
                    currentPlayer.finishedTurn = True
                    MsgBox("Unlucky! " & currentPlayer.name & " You are now bust!")
                ElseIf currentPlayer.hasAces = True And
getPlayerHandValueWithAces(currentPlayer) > 21 And getPlayerHandValue(currentPlayer) >
21 Then
                    currentPlayer.finishedTurn = True
                    MsgBox("Unlucky! " & currentPlayer.name & " You are now bust!")
                ElseIf getPlayerHandValue(currentPlayer) = 21 Then
                    currentPlayer.finishedTurn = True
                End If
            End If
        End If

    End Sub

    'If the user clicks the stick button, it will check if they have split hteir hand
as if they have, they will have another turn if they have stuck with their first hand
    'If the user hasnt split, their finishedTurn property is assigned a True value,
this is then used to move the turn system onto the next player
    Private Sub btnStick_Click(sender As Object, e As EventArgs) Handles
btnStick.Click

        If currentPlayer.doneSplit = True Then
            If currentPlayer.finishedTurnArrayOne = False Then
                If currentPlayer.hasAces = True Then
                    currentPlayer.handValueArrayOne =
getSplitHandValueWithAces(currentPlayer, 1)
                Else
                    currentPlayer.handValueArrayOne = getSplitHandValue(currentPlayer,
1)
                End If
                currentPlayer.finishedTurnArrayOne = True
            Else
                If currentPlayer.hasAces = True Then
                    currentPlayer.handValueArrayTwo =
getSplitHandValueWithAces(currentPlayer, 2)
                Else
                    currentPlayer.handValueArrayTwo = getSplitHandValue(currentPlayer,
2)
                End If
                currentPlayer.finishedTurnArrayTwo = True
            End If
        Else
            If currentPlayer.hasAces = True Then
                currentPlayer.handValue = getPlayerHandValueWithAces(currentPlayer)
            Else
                currentPlayer.handValue = getPlayerHandValue(currentPlayer)
```

```
            End If
            currentPlayer.finishedTurn = True
        End If

    End Sub

    'If the user clicks the double button, their bet is doubled (new bet subtracted
from balance)
    'The user is then dealt one more card, and then their finishedTurn property is set
to True, and their chip count is updated.
    Private Sub btnDouble_Click(sender As Object, e As EventArgs) Handles
btnDouble.Click
        currentPlayer.balance -= (currentPlayer.betAmount)
        currentPlayer.betAmount *= 2
        dealCardToPlayer(currentPlayer)
        currentPlayer.finishedTurn = True
        currentPlayer.updateChips()
    End Sub

    'If the user selects the hint button, the giveHint function is used with the
current player as a parameter to give a hint
    'This hint is then displayed to the user in the form of a message box.
    Private Sub btnShowHint_Click(sender As Object, e As EventArgs) Handles
btnShowHint.Click
        MsgBox(giveHint(currentPlayer))
    End Sub

End Class
```

Settings Form (Settings.vb)

Design



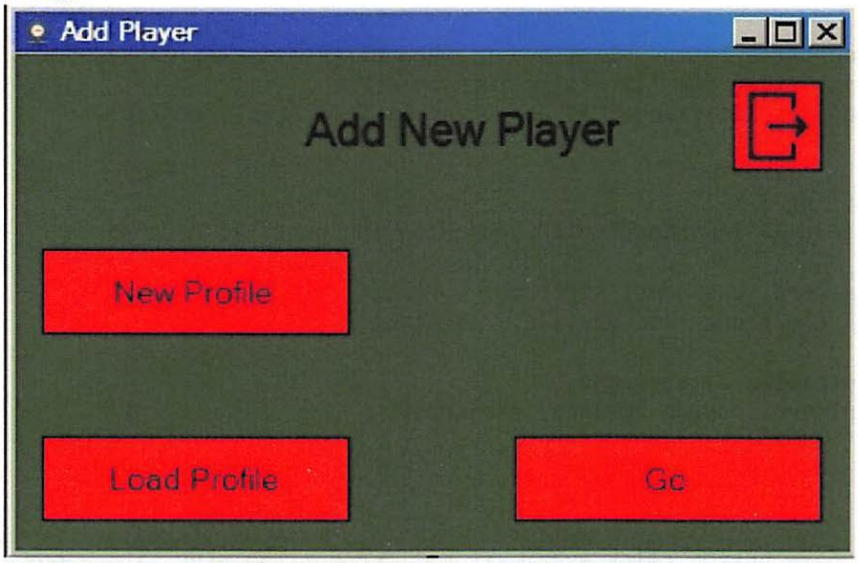
Code

```
Public Class Settings

    Public Shared hints As Boolean
    Public Shared sounds As Boolean

    'Handles the ExitGame button click event, will prompt the user if they are sure
that they want to exit, if they click Yes, they are returned to the mainmenu.
    Private Sub btnExitGame_Click(sender As Object, e As EventArgs) Handles
btnExitGame.Click
        Select Case MsgBox("Are you sure you want return to the main menu?",
MsgBoxStyle.YesNo, "Return to main menu")
            Case MsgBoxResult.Yes
                MainMenu.Show()
                Me.Hide()
```

```
            Case Else
        End Select
    End Sub

End Class
```

Add Player Form (AddPlayer.vb)

Design



Code

```
Public Class AddPlayer

    Const password As String = "abc"
    Public enteredName As Boolean

    'Handles the go button being clicked
    'Leads back to the main menu.
    Private Sub btnGo_Clicked(sender As Object, e As EventArgs) Handles btnGo.Click
        If enteredName = True Then
            Me.Close()
            MainMenu.Show()
        Else
            MsgBox("ERROR! Load a profile or create one first")
        End If
    End Sub

    'Loaded when the add player form is loaded.
    'Sets enteredName to False.
    Private Sub AddPlayer_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        enteredName = False
    End Sub

    Private Sub btnNewProfile_Click(sender As Object, e As EventArgs) Handles
btnNewProfile.Click
        Dim strError As String = ""
        Dim username As String = InputBox("Enter your desired username here (No more
than 12 characters):")
        'Create profile
        If (createNewProfile(password, username) = False) Or (saveProfile(password,
MainMenu.player(MainMenu.playersLoaded).name,
MainMenu.player(MainMenu.playersLoaded).balance, strError) = False) Or
(username.Length > 12) Then
```

```
            MsgBox("ERROR! Username is either already taken or empty. Please enter
again: " & strError)
        Else
            saveProfile(password, MainMenu.player(MainMenu.playersLoaded).name,
MainMenu.player(MainMenu.playersLoaded).balance, strError)
            displayProfile(password, MainMenu.player(MainMenu.playersLoaded).name,
MainMenu.player(MainMenu.playersLoaded).balance, strError)
        End If
    End Sub

    'This handles the LoadProfile button click event, when clicked it will attempt to
load a profile, if it doesnt exist then it will return an error.
    Private Sub btnLoadProfile_Click(sender As Object, e As EventArgs) Handles
btnLoadProfile.Click

        Dim strError As String = ""
        Dim username As String = InputBox("Enter your username here:")

        If loadProfile(password, username, strError) = False Then
            MsgBox("ERROR! Could not find profile: " & strError)
        Else
            displayProfile(password, username,
MainMenu.player(MainMenu.playersLoaded).balance, strError)
        End If

    End Sub

    'This handles the ExitGame button click event, it will prompt the user with a box
asking if they are sure, if they click Yes, they are returned to the MainMenu.
    Private Sub btnExitGame_Click(sender As Object, e As EventArgs) Handles
btnExitGame.Click
        Select Case MsgBox("Are you sure you want to return to the main menu?",
MsgBoxStyle.YesNo, "Return To Main Menu")
            Case MsgBoxResult.Yes
                MainMenu.playersLoaded = 0
                MainMenu.Show()
                Me.Hide()
            Case Else
        End Select
    End Sub
End Class
```

Game Methods Module (GameMethods.vb)

Code

```
Module GameMethods

    'GAME VARIABLES

    Public finishedDealing As Boolean = False
    Public roundFinished As Boolean = False
    Public cards As New List(Of String)
    Public currentPlayer As player

    Private playerBets(MainMenu.numberOfPlayers) As Integer
    Private currentCardToMove As PictureBox
    Private playerToRecieveCard As Integer
    Private currentDealtPlayer As player

    Private xDone As Boolean = False
    Private yDone As Boolean = False

    Private numberOfSimulatedPlayers As Integer
```

```
    Private currentSimulationPlayer As simulated_Player
    Private currentDealtSimulationPlayer As simulated_Player

    Public Const dealerCardPath As String = "X:\College\A-Level\Computer
Science\Coursework\BlackJack - Josh Dawson\BlackJack - Josh Dawson\bin\Debug\Card
Images\"
    Public Const soundEffectPath As String = "X:\College\A-Level\Computer
Science\Coursework\BlackJack - Josh Dawson\BlackJack - Josh Dawson\bin\Debug\Sound
Effects\"


    'Hides all the chip images.
    'Displays the player stats for each player in the game (username and balance).
    'Generates a deck of cards and shuffles the deck
    'Deals a card to each player in the lobby, then deals again.
    'Gets the chip count for each player's balance
    Sub startGame()

        Dim currentPlayerTurn As Integer = 1

        hideButtons()
        hideAllChipImages()
        cards = shuffleCards(generateCards())
        dealCardsToPlayerObjects(cards)
        setDealtCards()
        For i As Integer = 1 To MainMenu.numberOfPlayers
            MainMenu.player(i).updateChips()
        Next
        getPlayerBets()
        For i As Integer = 1 To MainMenu.numberOfPlayers
            MainMenu.player(i).updateChips()
        Next
        dealFirstCardsToPlayersVisually()

        'GAME BEGINS
        showPlayerStatHints()
        'Blackjack check
        'Sets the hasBlackjack property to true, used when calculating the bet amounts
        For i As Integer = 1 To MainMenu.numberOfPlayers
            If hasBlackjack(MainMenu.player(i)) Then
                MainMenu.player(i).hasBlackjack = True
                MsgBox("Congratulations! " & MainMenu.player(i).name & " you have a
Blackjack!")
                MainMenu.player(i).finishedTurn = True
            End If
            checkForSplit(MainMenu.player(i))
        Next
        If getDealerHandValue() = 21 Then
            MainMenu.dealer.hasBlackjack = True
        End If

        If Settings.hints = True Then
            PlayingBoard.btnShowHint.Show()
        End If
        'Turn system
        For i As Integer = 1 To MainMenu.numberOfPlayers
            currentPlayer = MainMenu.player(i)
            returnLabelName(currentPlayer.number, "Name").BackColor = Color.Red
            If currentPlayer.finishedTurn = False And
getPlayerHandValue(currentPlayer) < 21 Then
                If currentPlayer.hasSplit = True Then
```

```
                    Select Case MsgBox(currentPlayer.name & " would you like to split
your hand?", MsgBoxStyle.YesNo, "Split")
                        Case MsgBoxResult.Yes
                            currentPlayer.doneSplit = True
                            currentPlayer.createSplit()
                            PlayingBoard.btnHit.Show()
                            PlayingBoard.btnStick.Show()
                            'FIRST SPLIT HAND
                            Do Until currentPlayer.finishedTurnArrayOne = True
                                    Application.DoEvents()
                                Loop
                                If currentPlayer.hasAces = True And
getSplitHandValueWithAces(currentPlayer, 1) > 21 Then
                                    currentPlayer.finishedTurnArrayOne = False
                                    Do Until currentPlayer.finishedTurnArrayOne = True
                                        Application.DoEvents()
                                    Loop
                                End If
                                currentPlayer.changeSplit()
                                'SECOND SPLIT HAND
                                Do Until currentPlayer.finishedTurnArrayTwo = True
                                    Application.DoEvents()
                                Loop
                                If currentPlayer.hasAces = True And
getSplitHandValueWithAces(currentPlayer, 2) > 21 Then
                                    currentPlayer.finishedTurnArrayTwo = False
                                    Do Until currentPlayer.finishedTurnArrayTwo = True
                                        Application.DoEvents()
                                    Loop
                                End If
                                PlayingBoard.btnDouble.Hide()
                                Case Else
                    End Select
                Else
                    PlayingBoard.btnHit.Show()
                    PlayingBoard.btnStick.Show()
                    If currentPlayer.balance - currentPlayer.betAmount > 0 Then
                        PlayingBoard.btnDouble.Show()
                    End If
                    Do Until currentPlayer.finishedTurn = True
                        Application.DoEvents()
                    Loop
                    If currentPlayer.hasAces = True And
getPlayerHandValueWithAces(currentPlayer) > 21 Then
                        currentPlayer.finishedTurn = False
                        Do Until currentPlayer.finishedTurn = True
                            Application.DoEvents()
                        Loop
                    End If
                    PlayingBoard.btnDouble.Hide()
                End If
            End If
            showPlayerStatHints()
            returnLabelName(currentPlayer.number, "Name").BackColor =
Color.Transparent
        Next
        'Turns are done
        hideButtons()
        'If there is a player who isnt bust, the dealer will recieve a card till they
have a higher handVlaue than 17 or go bust.
        dealerTurn()
        showDealerStatHints()
```

```
        calculateWinners()
        For i As Integer = 1 To MainMenu.numberOfPlayers
            MainMenu.player(i).updateChips()
        Next
        savePlayerProfilesAndUpdateLeaderboards()
        roundFinished = True
        PlayingBoard.btnNewRound.Show()

    End Sub

'Moves the players second card bellow the first card, showing they are two different
hands
    Public Sub splitHandsVisually(ByVal player As player)

        For i As Integer = 1 To 6
            moveCardY(returnCardBox(player.splitCardsTwo(1)), 5)
            Application.DoEvents()
            System.Threading.Thread.Sleep(10)
        Next
        For i As Integer = 1 To 4
            moveCardX(returnCardBox(player.splitCardsTwo(1)), -3.125)
            Application.DoEvents()
            System.Threading.Thread.Sleep(10)
        Next

    End Sub

    'Will detect if the players handValue is equal to 21, if it is, they have a
blackjack and this function is made equal to True
    Private Function hasBlackjack(ByVal player As player) As Boolean

        If getPlayerHandValue(player) = 21 Then
            player.hasBlackjack = True
        End If

    End Function

    'When a player is passed as a parameter, it will look at their hand and give a
hint based on it
    Function giveHint(ByVal player As player) As String

        player.handValue = getPlayerHandValue(player)
        If player.handValue > 17 Then
            giveHint = player.name & " i do not reccomend hitting a card as your hand
value is " & player.handValue & ". In my opinion you should STICK."
        ElseIf player.handValue < 15 And ((player.balance - player.betAmount > 0) And
(player.balance / 2 - player.betAmount > 0)) Then
            giveHint = player.name & " i reccomend doubling down as your bet is quite
small compared to your balance! In my opinion you should DOUBLE."
        ElseIf player.handValue < 14 Then
            giveHint = player.name & " i reccomend hitting a card as your hand value
is " & player.handValue & ". In my opinion you should HIT."
        ElseIf getDealerFirstCardValue() > 10 Then
            giveHint = player.name & " i reccomend sticking as the dealers first card
is high! In my opinion you should STICK."
        ElseIf player.handValue > 1 Then
            giveHint = player.name & " i reccomend hitting as the dealers first card
isnt very high, and your hand value (" & player.handValue & ") is quite low. In my
opinion you should HIT."
        Else
            giveHint = "ERROR"
        End If
```

```
    End Function

    'Will get the handValue of the dealers first card
    'Is used when displaying the dealer's handValue in the hint section, but will keep
their second card a mystery
    Function getDealerFirstCardValue() As Integer

        Dim cardType As Integer

        If MainMenu.dealer.cards(1).Length = 2 Then
            If IsNumeric(MainMenu.dealer.cards(1).Substring(0, 1)) = False Then
                Select Case MainMenu.dealer.cards(1).Substring(0, 1)
                    Case "A"
                        cardType = 11
                    Case "J", "Q", "K"
                        cardType = 10
                End Select
            Else
                cardType = MainMenu.dealer.cards(1).Substring(0, 1)
            End If
        Else
            cardType = MainMenu.dealer.cards(1).Substring(0, 2)
        End If
        If IsNumeric(cardType) = False Then
            Select Case cardType
                Case "A"
                    cardType = 11
                Case "J", "Q", "K"
                    cardType = 10
            End Select
        End If
        getDealerFirstCardValue = cardType

    End Function

    'Will look at all the players and find which player has a winning hand
    Sub calculateWinners()

        For i As Integer = 1 To MainMenu.numberOfPlayers
            If MainMenu.player(i).doneSplit = True Then
                If (MainMenu.player(i).hasAces = True) And
(getSplitHandValue(MainMenu.player(i), 1) > 21 And
(getSplitHandValueWithAces(MainMenu.player(i), 1))) Then
                    MainMenu.player(i).handValueArrayOne =
getSplitHandValueWithAces(MainMenu.player(i), 1)
                Else
                    MainMenu.player(i).handValueArrayOne =
getSplitHandValue(MainMenu.player(i), 1)
                End If
                'Checks if they have an equal handValue
                If MainMenu.player(i).handValueArrayOne = MainMenu.dealer.handValue
Then
                    MsgBox("Its a tie! " & MainMenu.player(i).name & " you have won
back your bet of ¢" & calculateBetReturn(MainMenu.player(i), 0.5) & " for your first
hand!")
                    'Checks if the player has a handTotal higher than the dealer but
its less than 22, or that it is less than 22 and the dealer went bust.
                ElseIf (MainMenu.player(i).handValueArrayOne >
MainMenu.dealer.handValue And MainMenu.player(i).handValueArrayOne < 22) Or
(MainMenu.player(i).handValueArrayOne < 22 And MainMenu.dealer.handValue > 21) Then
```

```
                    MsgBox(MainMenu.player(i).name & " you have won ¢" &
calculateBetReturn(MainMenu.player(i), 0.75) & " from your first hand!")
                End If
                If (MainMenu.player(i).hasAces = True) And
(getSplitHandValue(MainMenu.player(i), 2) > 21 And
(getSplitHandValueWithAces(MainMenu.player(i), 2))) Then
                    MainMenu.player(i).handValueArrayTwo =
getSplitHandValueWithAces(MainMenu.player(i), 2)
                Else
                    MainMenu.player(i).handValueArrayTwo =
getSplitHandValue(MainMenu.player(i), 2)
                End If

                If MainMenu.player(i).handValueArrayTwo = MainMenu.dealer.handValue
Then
                    MsgBox("Its a tie! " & MainMenu.player(i).name & " you have won
back your bet of ¢" & calculateBetReturn(MainMenu.player(i), 0.5) & " for your second
hand!")
                    'Checks if the player has a handTotal higher than the dealer but
its less than 22, or that it is less than 22 and the dealer went bust.
                ElseIf (MainMenu.player(i).handValueArrayTwo >
MainMenu.dealer.handValue And MainMenu.player(i).handValueArrayOne < 22) Or
(MainMenu.player(i).handValueArrayTwo < 22 And MainMenu.dealer.handValue > 21) Then
                    MsgBox(MainMenu.player(i).name & " you have won ¢" &
calculateBetReturn(MainMenu.player(i), 0.75) & " from your second hand!")
                End If
            Else
                If (MainMenu.player(i).hasAces = True) And
(getPlayerHandValue(MainMenu.player(i))) > 21 And
(getPlayerHandValueWithAces(MainMenu.player(i)) < 22) Then
                    MainMenu.player(i).handValue =
getPlayerHandValueWithAces(MainMenu.player(i))
                Else
                    MainMenu.player(i).handValue =
getPlayerHandValue(MainMenu.player(i))
                End If
                'Checks if the player has a blackjack
                If MainMenu.player(i).hasBlackjack = True Then
                    MsgBox(MainMenu.player(i).name & " you got a blackjack and won ¢"
& calculateBetReturn(MainMenu.player(i), 2))
                    'Checks if they have an equal handValue
                ElseIf MainMenu.player(i).handValue = MainMenu.dealer.handValue Then
                    MsgBox("Its a tie! " & MainMenu.player(i).name & " you have won
back your bet of ¢" & calculateBetReturn(MainMenu.player(i), 1))
                    'Checks if the player has a handTotal higher than the dealer but
its less than 22, or that it is less than 22 and the dealer went bust.
                ElseIf (MainMenu.player(i).handValue > MainMenu.dealer.handValue And
MainMenu.player(i).handValue < 22) Or (MainMenu.player(i).handValue < 22 And
MainMenu.dealer.handValue > 21) Then
                    MsgBox(MainMenu.player(i).name & " you have won ¢" &
calculateBetReturn(MainMenu.player(i), 1.5))
                End If
            End If
            MainMenu.player(i).updateChips()
        Next

    End Sub

    'Shows the player card values.
    Sub showPlayerStatHints()

        If Settings.hints = True Then
```

```
            Dim nameLabel As Label
            Dim valueLabel As Label

            For i As Integer = 1 To MainMenu.numberOfPlayers

                MainMenu.player(i).handValue = getPlayerHandValue(MainMenu.player(i))

                nameLabel = returnStatLabelName("Player" & i, "Name")
                valueLabel = returnStatLabelName("Player" & i, "Value")

                nameLabel.Show()
                nameLabel.Text = MainMenu.player(i).name

                valueLabel.Show()

                If MainMenu.player(i).handValue > 21 Then
                    If MainMenu.player(i).hasAces = True And
getPlayerHandValueWithAces(MainMenu.player(i)) > 21 Then
                        valueLabel.Text = "BUST"
                        valueLabel.BackColor = Color.Red()
                    ElseIf MainMenu.player(i).hasAces = True And
getPlayerHandValueWithAces(MainMenu.player(i)) < 22 Then
                        valueLabel.Text = "Value: " &
getPlayerHandValueWithAces(MainMenu.player(i))
                    Else
                        valueLabel.Text = "BUST"
                        valueLabel.BackColor = Color.Red()
                    End If
                Else
                    valueLabel.Text = "Value: " & MainMenu.player(i).handValue
                End If
            Next
        End If

    End Sub

    'Returns the label name of a statLabel.
    Function returnStatLabelName(ByVal name As String, ByVal type As String) As Label

        If type = "Name" Or type = "Value" Then
            For Each label In PlayingBoard.Controls
                If label.name = "lbl" & name & "Stats" & type Then
                    Return label
                End If
            Next
        End If

    End Function

    'Shows the dealers stat hint labels with card values.
    Sub showDealerStatHints()

        If Settings.hints = True Then
            Dim nameLabel As Label
            Dim valueLabel As Label

            MainMenu.dealer.handValue = getDealerHandValue()

            valueLabel = returnStatLabelName("Dealer", "Value")
            nameLabel = returnStatLabelName("Dealer", "Name")

            valueLabel.Show()
```

```
            nameLabel.Show()

            If MainMenu.dealer.handValue > 21 Or MainMenu.dealer.handValue = 100 Then
                valueLabel.Text = "BUST"
                valueLabel.BackColor = Color.Red()
            Else
                valueLabel.Text = "Value: " & MainMenu.dealer.handValue
            End If
        End If

    End Sub

    'Resets the hint labels
    Sub resetHintLabels()

        Dim valueLabel As Label

        For i As Integer = 1 To MainMenu.numberOfPlayers
            valueLabel = returnStatLabelName("Player" & i, "Value")
            valueLabel.BackColor = Color.Transparent
            valueLabel.Text = "Value:"
        Next

        returnStatLabelName("Dealer", "Value").Text = "Value:"
        returnStatLabelName("Dealer", "Value").BackColor = Color.Transparent

    End Sub

    'Handles the dealer AI.
    Sub dealerTurn()

        Dim allBust As Boolean = False
        showDealerCard()

        For i As Integer = 1 To MainMenu.numberOfPlayers
            If MainMenu.player(i).handValue < 21 Then
                allBust = False
            End If
        Next

        If allBust = True Then
            If MainMenu.simulatedGame = False Then
                showDealerStatHints()
            End If
        Else
            Do
                MainMenu.dealer.handValue = getDealerHandValue()
                If MainMenu.dealer.handValue < 21 And MainMenu.dealer.handValue < 17
Then
                    dealCardToDealer()
                    If MainMenu.simulatedGame = False Then
                        showDealerStatHints()
                    End If
                    System.Threading.Thread.Sleep(100)
                End If

            Loop Until MainMenu.dealer.handValue > 16
        End If

    End Sub

    'This sub will handle a simulated dealer turn
```

```
    Sub simulatedDealerTurn()

        Dim allBust As Boolean = False
        showDealerCard()

        For i As Integer = 1 To MainMenu.simulatedPlayers
            If MainMenu.simPlayer(i).handValue < 21 Then
                allBust = False
            End If
        Next

        If allBust <> True Then
            Do
                MainMenu.dealer.handValue = getDealerHandValue()
                If MainMenu.dealer.handValue < 21 And MainMenu.dealer.handValue < 17
Then
                    dealCardToDealer()
                    System.Threading.Thread.Sleep(100)
                End If
            Loop Until MainMenu.dealer.handValue > 16
        End If

    End Sub


    'Handles the new profile button click event.
    'Asks for a username until validUsername = true
    'Once true, the player object is assigned the username and a balance of 1000.
    'The players profile is then saved using saveProfile and then displayed using
displayProfile.
    Public Function createNewProfile(ByVal password As String, ByVal username As
String, Optional ByRef strError As String = "") As Boolean

        Dim validUsername As Boolean = False
        Try
            If (username.Contains(" ")) Or (username.Length = 0) Or
(verifyUsername(username) = False) Then
                Return False
            Else
                validUsername = True
                MainMenu.player(MainMenu.playersLoaded).name = username
                MainMenu.player(MainMenu.playersLoaded).balance = 1000
            End If
            Return True
        Catch ex As Exception
            strError = ex.Message
            Return False
        End Try

    End Function

    'Deals a card to a player when given a player as a parameter
    Sub dealCardToPlayer(ByVal player As player)

        player.cards.Add(cards.ElementAt(0))
        cards.RemoveAt(0)

        yDone = False
        xDone = False
        playerToRecieveCard = player.number
        currentDealtPlayer = player
        Dim playerCard = currentDealtPlayer.cards(currentDealtPlayer.dealtCards)
```

```
        currentCardToMove = returnCardBox(playerCard)
        moveCard()
        currentDealtPlayer.dealtCards += 1

    End Sub

    'A modified version of the dealCardToPlayer sub, but uses the arrayToDealTo
property to deal a card to the correct array if the player has split their hand
    Sub dealCardToSplitArray(ByVal player As player)

        If player.arrayToDealTo = 1 Then
            player.splitCardsOne.Add(cards.ElementAt(0))
            cards.RemoveAt(0)
            yDone = False
            xDone = False
            playerToRecieveCard = player.number
            currentDealtPlayer = player
            Dim playerCard =
currentDealtPlayer.splitCardsOne(currentDealtPlayer.dealtCardsArrayOne)
            currentCardToMove = returnCardBox(playerCard)
            moveCard()
            currentDealtPlayer.dealtCardsArrayOne += 1
        Else
            player.splitCardsTwo.Add(cards.ElementAt(0))
            cards.RemoveAt(0)
            yDone = False
            xDone = False
            playerToRecieveCard = player.number
            currentDealtPlayer = player
            Dim playerCard =
currentDealtPlayer.splitCardsTwo(currentDealtPlayer.dealtCardsArrayTwo)
            currentCardToMove = returnCardBox(playerCard)
            moveCard()
            currentDealtPlayer.dealtCardsArrayTwo += 1
        End If

    End Sub

    'Deals a card to the dealer
    Sub dealCardToDealer()

        Dim dealer As AI_Player = MainMenu.dealer

        dealer.cards.Add(cards.ElementAt(0))
        cards.RemoveAt(0)

        yDone = False
        xDone = False
        playerToRecieveCard = 0
        Dim dealerCard = dealer.cards(dealer.dealtCards)
        currentCardToMove = returnCardBox(dealerCard)
        moveCard()
        dealer.dealtCards += 1

    End Sub

    'Checks to see if a player has a split, if so it will iniciate the createSplit
sub.
    Sub checkForSplit(ByVal player As player)

        Select Case player.cards(1).Length
            Case 2
```

```
                If player.cards(1).Substring(0, 1) = player.cards(2).Substring(0, 1)
And (player.balance - player.betAmount > 0) Then
                    player.hasSplit = True
                End If
            Case 3
                Select Case player.cards(2).Length
                    Case 3
                        If player.cards(2).Length = 3 Then
                            player.hasSplit = True
                        End If
                End Select
        End Select

    End Sub

    'Used when saving the users profile.
    'Saves a txt document in the debug folder called "gamesave_*username*" with the
appropriate balance.
    Public Function saveProfile(ByVal password As String, ByVal username As String,
ByVal balance As Integer, Optional ByRef strError As String = "") As Boolean

        Try
            Dim fileHandle As IO.StreamWriter = New IO.StreamWriter("gamesave_" &
username & ".txt")
            fileHandle.WriteLine(encrypt(balance, password))
            fileHandle.Close()
            Return True
        Catch ex As Exception
            strError = ex.Message
            Return False
        End Try

    End Function

    'Handels btnLoadProfile.
    'Searches the debug folder for the username given, loads the profile if it exists,
if not, flags an error.
    Public Function loadProfile(ByVal password As String, ByVal username As String,
Optional ByRef strError As String = "") As Boolean

        Try
            If username = "" Then
                strError = "Username is blank!"
                Return False
            End If
            Dim fileHandle As IO.StreamReader = New IO.StreamReader("gamesave_" &
username & ".txt")
            MainMenu.player(MainMenu.playersLoaded).balance =
decrypt(fileHandle.ReadLine(), password)
            fileHandle.Close()
            MainMenu.player(MainMenu.playersLoaded).name = username
            Return True
        Catch ex As Exception
            strError = ex.Message
            Return False
        End Try

    End Function

    'Verifies that a username given doesnt already exist.
    'Returns False if a username exists and True if it doesnt.
```

```
    Function verifyUsername(ByVal username As String, Optional ByRef strError As
String = "") As Boolean

        Try
            Dim tempHandle As IO.StreamReader = New IO.StreamReader("gamesave_" &
username & ".txt")
            tempHandle.Close()
            Return False
        Catch ex As Exception
            strError = ex.Message
            Return True
        End Try

    End Function

    'Changes the labels in the lobby system with the users username and balance.
    Public Function displayProfile(ByVal password As String, ByVal username As String,
ByVal balance As Integer, Optional ByRef strError As String = "") As Boolean

        Try
            saveProfile(password, MainMenu.player(MainMenu.playersLoaded).name,
MainMenu.player(MainMenu.playersLoaded).balance)
            AddPlayer.lblUsername.Text = "Player: " &
MainMenu.player(MainMenu.playersLoaded).name
            AddPlayer.lblBalance.Text = "Balance: ¢" &
MainMenu.player(MainMenu.playersLoaded).balance
            AddPlayer.enteredName = True
            Select Case MainMenu.playersLoaded
                Case 1
                    MainMenu.lblPlayer1.Text = MainMenu.player(1).name & " ¢" &
MainMenu.player(1).balance
                Case 2
                    MainMenu.lblPlayer2.Text = MainMenu.player(2).name & " ¢" &
MainMenu.player(2).balance
                Case 3
                    MainMenu.lblPlayer3.Text = MainMenu.player(3).name & " ¢" &
MainMenu.player(3).balance
                Case 4
                    MainMenu.lblPlayer4.Text = MainMenu.player(4).name & " ¢" &
MainMenu.player(4).balance
            End Select
            Return True
        Catch ex As Exception
            strError = ex.Message
            Return False
        End Try

    End Function

    'Hides stick, hit and double buttons.
    Sub hideButtons()

        PlayingBoard.btnHit.Hide()
        PlayingBoard.btnStick.Hide()
        PlayingBoard.btnDouble.Hide()
        PlayingBoard.btnShowHint.Hide()

    End Sub

    'Will load the 5 players inside the current leaderboard txt document
    'If any of these players played in the previous game, their balance is updated,
otherwise they are added to the savedPlayers array
```

```
    'The array is then sorted using the "quickSortLeaderboards" subroutine which will
sort them using the quicksort algorithm based on their balance, despite the fact that
a string array is passed in with their balance and name appended
    'The top 5 balances from this sorted array are then saved to the leaderboard txt
document
    Sub savePlayerProfilesAndUpdateLeaderboards()

        Dim readPlayers() As String = readLeaderboards()
        Dim savedPlayers(8) As String
        Dim savedPlayersCount As Integer = 5
        Dim replaced As Boolean = False
        Dim savedPlayersLength As Integer = 0

        For i As Integer = 0 To 4
            savedPlayers(i) = readPlayers(i)
        Next

        For i As Integer = 1 To MainMenu.numberOfPlayers
            replaced = False
            For j As Integer = 0 To 4
                If savedPlayers(j).Contains(MainMenu.player(i).name) Then
                    savedPlayers(j) = MainMenu.player(i).balance & "." &
MainMenu.player(i).name
                    replaced = True
                End If
                If j = 4 And replaced = False Then
                    savedPlayers(savedPlayersCount) = MainMenu.player(i).balance & "."
& MainMenu.player(i).name
                    savedPlayersCount += 1
                End If
            Next
            replaced = False
        Next

        For i As Integer = savedPlayersCount To 8
            savedPlayers(i) = "EMPTY"
        Next

        For i As Integer = 0 To 8
            If savedPlayers(i).Contains(".") Then
                savedPlayersLength += 1
            End If
        Next

        Dim unsortedPlayers(savedPlayersLength - 1) As String
        For i As Integer = 0 To (savedPlayersLength - 1)
            unsortedPlayers(i) = savedPlayers(i)
        Next

        quickSortLeaderboard(unsortedPlayers, 0, (savedPlayersLength - 1))
        Dim sortedPlayers() As String = unsortedPlayers
        Dim saveStart As Integer = savedPlayers.Count - 5
        Dim fileHandle As IO.StreamWriter = New IO.StreamWriter("Leaderboard.txt")
        For i As Integer = 1 To 5
            If i <> 5 Then
                fileHandle.Write(sortedPlayers(saveStart) & ":")
            Else
                fileHandle.Write(sortedPlayers(saveStart))
            End If
            saveStart -= 1
        Next
        fileHandle.Close()
```

```
    End Sub

    'Resets the playing board for a new round
    Sub resetBoard()

        resetHintLabels()
        PlayingBoard.pbTopOfDeck.BringToFront()
        showDealerCard()
        hideAllChipImages()
        roundFinished = False

        'Resets all the cards to the correct position
        For Each card In PlayingBoard.Controls
            If card.width = 64 Then
                card.location = New Point(663, 230)
            End If
        Next

        PlayingBoard.pbTopOfDeck.BringToFront()
        For i As Integer = 1 To MainMenu.numberOfPlayers
            MainMenu.player(i).reset()
        Next
        MainMenu.dealer.reset()
        PlayingBoard.pbTopOfDeck.BringToFront()

    End Sub

    'Generates bet winnings when given a player and the correct bet multiplyer.
    Function calculateBetReturn(ByVal player As player, ByVal multiplyer As Double) As
Integer

        Dim overlap As Integer = 0

        calculateBetReturn = (player.betAmount * multiplyer)
        If calculateBetReturn Mod 5 <> 0 Then
            overlap = calculateBetReturn Mod 5
            calculateBetReturn -= overlap
        End If
        player.balance += calculateBetReturn


    End Function

    'Gets the file name for a card when given the name of the card.
    Function getCardFileName(ByVal card As String) As String

        Dim name As String = card
        Dim type As String
        Dim suit As String

        Select Case name.Substring(0, 1)
            Case 2, 3, 4, 5, 6, 7, 8, 9
                type = name.Substring(0, 1)
            Case 1
                type = name.Substring(0, 2)
                Select Case name.Substring(2, 1)
                    Case "C"
                        suit = "clubs"
                    Case "D"
                        suit = "diamonds"
                    Case "H"
```

```
                    suit = "hearts"
                Case "S"
                    suit = "spades"
            End Select
        Case "J"
            type = "jack"
        Case "K"
            type = "king"
        Case "Q"
            type = "queen"
        Case "A"
            type = "ace"
    End Select
    If name.Length = 2 Then
        Select Case name.Substring(1, 1)
            Case "C"
                suit = "clubs"
            Case "D"
                suit = "diamonds"
            Case "H"
                suit = "hearts"
            Case "S"
                suit = "spades"
        End Select
    End If

    getCardFileName = type & "_of_" & suit

End Function

'Will reveal the dealers second card.
Sub showDealerCard()

    Dim fileName As String

    fileName = dealerCardPath & getCardFileName(MainMenu.dealer.cards(2)) & ".png"
    returnCardBox(MainMenu.dealer.cards(2)).Image = Image.FromFile(fileName)

End Sub

'Hides the dealers last card from view.
Sub hideDealerCard()

    returnCardBox(MainMenu.dealer.cards((MainMenu.dealer.cards.Count) - 1)).Image
= Image.FromFile("back_of_card.png")
    returnCardBox(MainMenu.dealer.cards((MainMenu.dealer.cards.Count) -
1)).SizeMode = PictureBoxSizeMode.StretchImage

End Sub

'This subroutine will hide the picturebox of every chip on the Playing Board
Sub hideAllChipImages()

    Dim chipToHide As Integer
    Dim chipType() As Integer = {500, 100, 25, 10, 5}

    For j As Integer = 0 To 4
        chipToHide = chipType(j)
        For i As Integer = 1 To 4
            For k As Integer = 1 To 4
                returnChipBox(i, chipToHide, k).Hide()
            Next
```

```
            Next
        Next

    End Sub

    'Points to the chip PictureBox when given the player, chip value and chip number.
    Function returnChipBox(ByVal chipPlayer As Integer, ByVal chipType As Integer,
ByVal chipNumber As Integer) As PictureBox

        For Each pictureBox In PlayingBoard.Controls
            If pictureBox.name = "pbP" & chipPlayer & "_" & chipType & "_" &
chipNumber Then
                returnChipBox = pictureBox
            End If
        Next

    End Function

    'Sets each players dealt cards to 1. Used in dealing cards visually.
    Sub setDealtCards()

        If MainMenu.simulatedGame = False Then
            For i As Integer = 1 To MainMenu.numberOfPlayers
                MainMenu.player(i).dealtCards = 1
            Next
            MainMenu.dealer.dealtCards = 1
        Else
            For i As Integer = 1 To MainMenu.simulatedPlayers
                MainMenu.simPlayer(i).dealtCards = 1
            Next
            MainMenu.dealer.dealtCards = 1
        End If

    End Sub

    'Gets the bets for each player
    Sub getPlayerBets()

        Dim playerBet As String
        Dim intPlayerBet As Integer
        Dim validBet As Boolean = False
        Try
            For i As Integer = 1 To MainMenu.numberOfPlayers
                Do Until validBet = True
                    playerBet = InputBox("How much would you like to bet " &
playerName(i) & "?")
                    Try
                        intPlayerBet = Int(playerBet)
                        If playerBet <= playerBalance(i) And playerBet Mod 5 = 0 Then
                            validBet = True
                            MainMenu.player(i).betAmount = playerBet
                            MainMenu.player(i).balance -= MainMenu.player(i).betAmount
                        Else
                            MsgBox("ERROR! Either your bet is too high or your bet is
not a multiple of 5!")
                        End If
                    Catch ex As Exception
                        MsgBox("ERROR! You must enter a numerical value!")
                    End Try
                Loop
                validBet = False
            Next
```

```
        Catch ex As Exception
        End Try

    End Sub

    'Returns the name of the player number given as a parameter.
    Function playerName(ByVal playerNumber As Integer) As String

        playerName = MainMenu.player(playerNumber).name

    End Function

    'Returns the balance of the player number given as a parameter.
    Function playerBalance(ByVal playerNumber As Integer) As Integer

        playerBalance = MainMenu.player(playerNumber).balance

    End Function

    'Sets the card to deal as the next card the player needs to be dealt and starts
the deal timer.
    Sub dealFirstCardsToPlayersVisually()

        For k As Integer = 1 To 2
            For i As Integer = 1 To MainMenu.numberOfPlayers
                yDone = False
                xDone = False
                playerToRecieveCard = i
                currentDealtPlayer = MainMenu.player(i)
                Dim playerCard =
currentDealtPlayer.cards(currentDealtPlayer.dealtCards)
                currentCardToMove = returnCardBox(playerCard)
                moveCard()
                currentDealtPlayer.dealtCards += 1
            Next

            'Deals the dealer their 2 cards
            yDone = False
            xDone = False
            playerToRecieveCard = 0
            Dim dealerCard = MainMenu.dealer.cards(MainMenu.dealer.dealtCards)
            currentCardToMove = returnCardBox(dealerCard)
            If k = 2 Then
                hideDealerCard()
            End If
            moveCard()
            MainMenu.dealer.dealtCards += 1
        Next

    End Sub

    'Deals a card to the playerToRecieveCard
    Private Sub moveCard()

        Dim xAddition As Integer
        Dim yAddition As Integer
        Dim dealerXAddition As Integer
        Dim dealerCardPos = New Integer() {430, 40}
        Dim player1CardPos = New Integer() {910, 118}
        Dim player2CardPos = New Integer() {910, 360}
        Dim player3CardPos = New Integer() {422, 510}
        Dim player4CardPos = New Integer() {10, 310}
```

```
        currentCardToMove.BringToFront()
        finishedDealing = False

        If Settings.sounds = True Then
            My.Computer.Audio.Play(soundEffectPath & "Dealing_Effect.wav")
        End If

        If MainMenu.simulatedGame = False Then
            If currentDealtPlayer.doneSplit = True Then
                If currentDealtPlayer.arrayToDealTo = 1 Then
                    Select Case currentDealtPlayer.dealtCardsArrayOne
                        Case 2
                            xAddition = 6.25 * (currentDealtPlayer.dealtCardsArrayOne)
                        Case Else
                            xAddition = 12.5 * (currentDealtPlayer.dealtCardsArrayOne
- 1)
                    End Select
                Else
                    yAddition = 30
                    Select Case currentDealtPlayer.dealtCardsArrayTwo
                        Case 2
                            xAddition = 6.25 * (currentDealtPlayer.dealtCardsArrayTwo)
                        Case Else
                            xAddition = 12.5 * (currentDealtPlayer.dealtCardsArrayOne
- 1)
                    End Select
                End If
            Else
                If playerToRecieveCard <> 0 Then
                    If currentDealtPlayer.dealtCards > 1 Then
                        Select Case currentDealtPlayer.dealtCards
                            Case 2
                                xAddition = 6.25 * (currentDealtPlayer.dealtCards)
                            Case Else
                                xAddition = 12.5 * (currentDealtPlayer.dealtCards - 1)
                        End Select
                    End If
                End If
            End If
        End If

        Select Case playerToRecieveCard
            Case 0
                If MainMenu.dealer.dealtCards > 1 Then
                    Select Case MainMenu.dealer.dealtCards
                        Case 2
                            dealerXAddition = 6.25 * (MainMenu.dealer.dealtCards)
                        Case Else
                            dealerXAddition = 12.5 * (MainMenu.dealer.dealtCards - 1)
                    End Select
                End If
                initiateCardMove(dealerCardPos, dealerXAddition, ">", ">", -5, -5)
            Case 1
                initiateCardMove(player1CardPos, xAddition, "<", ">", 5, -5)
            Case 2
                initiateCardMove(player2CardPos, xAddition, "<", "<", 5, 5)
            Case 3
                initiateCardMove(player3CardPos, xAddition, ">", "<", -5, 5)
            Case 4
                initiateCardMove(player4CardPos, xAddition, ">", "<", -5, 5)
        End Select
```

```
        finishedDealing = True

    End Sub

    'Displays the players stats on the board (username and balance)
    Sub displayPlayerStats(ByVal player As base_player)

        Dim nameLabel As Label = returnLabelName(player.number, "Name")
        Dim balanceLabel As Label = returnLabelName(player.number, "Balance")
        nameLabel.Text = player.name
        balanceLabel.Text = "¢" & player.balance

    End Sub

    'Returns the label name for each player using the number.
    Function returnLabelName(ByVal playerNumber As Integer, ByVal nameOrBalance As
String) As Label

        For Each label In PlayingBoard.Controls
            If label.name = "lblPlayer" & playerNumber & nameOrBalance Then
                Return label
            End If
        Next
    End Function

    'Deals the cards on the deck to each player object.
    Sub dealCardsToPlayerObjects(ByVal cards As List(Of String))

        For i As Integer = 1 To 2
            For k As Integer = 1 To MainMenu.numberOfPlayers
                MainMenu.player(k).cards.Add(cards.ElementAt(i))
                cards.RemoveAt(i)
            Next
        Next

        For i As Integer = 1 To 2
            MainMenu.dealer.cards.Add(cards.ElementAt(0))
            cards.RemoveAt(0)
        Next

    End Sub

    'Generates a deck of cards in order
    Function generateCards() As List(Of String)

        Dim suit As String
        Dim cards As New List(Of String)
        For cardNum As Integer = 1 To 13
            For suitNum As Integer = 1 To 4
                If suitNum = 1 Then
                    suit = "H"
                ElseIf suitNum = 2 Then
                    suit = "D"
                ElseIf suitNum = 3 Then
                    suit = "C"
                Else
                    suit = "S"
                End If
                Select Case cardNum
                    Case 1
                        cards.Add("A" & suit)
```

```
                    Case 2, 3, 4, 5, 6, 7, 8, 9, 10
                        cards.Add(cardNum & suit)
                    Case 11
                        cards.Add("J" & suit)
                    Case 12
                        cards.Add("Q" & suit)
                    Case 13
                        cards.Add("K" & suit)
                End Select
            Next
        Next
        Return cards

    End Function

    'Deals the ordered cards randomly.
    Function shuffleCards(ByVal unshuffledCards As List(Of String)) As List(Of String)

        Dim shuffledDeck As New List(Of String)
        Dim rand As Integer
        Dim count As Integer = 51
        Randomize()
        For i As Integer = 1 To 52
            rand = Int(Rnd() * count)
            shuffledDeck.Add(unshuffledCards(rand))
            unshuffledCards.Remove(unshuffledCards(rand))
            count -= 1
            System.Threading.Thread.Sleep(50)
        Next

        If Settings.sounds = True Then
            My.Computer.Audio.Play(soundEffectPath & "Shuffling_Effect.wav",
        AudioPlayMode.WaitToComplete)
        End If

        Return shuffledDeck

    End Function

    'Gets the dealer card value
    Function getDealerHandValue() As Integer

        Dim cardType As Integer
        Dim currentCard As String
        MainMenu.dealer.valuedCount = 1
        If MainMenu.dealer.valuedCount <> MainMenu.dealer.dealtCards + 1 Then
            For i As Integer = MainMenu.dealer.valuedCount To
(MainMenu.dealer.cards.Count - 1)
                currentCard = MainMenu.dealer.cards(i)
                If currentCard.Length = 2 Then
                    If IsNumeric(currentCard.Substring(0, 1)) = False Then
                        Select Case currentCard.Substring(0, 1)
                            Case "A"
                                cardType = 11
                            Case "J", "Q", "K"
                                cardType = 10
                        End Select
                    Else
                        cardType = currentCard.Substring(0, 1)
                    End If
                Else
                    cardType = currentCard.Substring(0, 2)
```

```
            End If
            If IsNumeric(cardType) = False Then
                Select Case cardType
                    Case "A"
                        cardType = 11
                    Case "J", "Q", "K"
                        cardType = 10
                End Select
            End If
            MainMenu.dealer.valuedCount += 1
            getDealerHandValue += cardType
        Next
    End If

    Return getDealerHandValue

End Function

'Gets the cardValue of the player given as a parameter.
Function getPlayerHandValue(ByRef player As player) As Integer

    Dim cardType As Integer
    Dim currentCard As String
    Dim aceAddition As Integer = 0
    Dim numberOfAces As Integer = 0

    player.valuedCount = 1
    If player.valuedCount <> player.dealtCards + 1 Then
        For i As Integer = player.valuedCount To (player.cards.Count - 1)
            currentCard = player.cards(i)
            If currentCard.Length = 2 Then
                If IsNumeric(currentCard.Substring(0, 1)) = False Then
                    Select Case currentCard.Substring(0, 1)
                        Case "A"
                            cardType = 11
                            player.hasAces = True
                        Case "J", "Q", "K"
                            cardType = 10
                    End Select
                Else
                    cardType = currentCard.Substring(0, 1)
                End If
            Else
                cardType = currentCard.Substring(0, 2)
            End If
            If IsNumeric(cardType) = False Then
                Select Case cardType
                    Case "A"
                        cardType = 11
                        player.hasAces = True
                    Case "J", "Q", "K"
                        cardType = 10
                End Select
            End If
            player.valuedCount += 1
            getPlayerHandValue += cardType
        Next
    End If

    Return getPlayerHandValue

End Function
```

```
    'Gets the player's hand value, but sets Ace = 1 instead of 11
    Function getPlayerHandValueWithAces(ByVal player As player) As Integer

        Dim cardType As Integer
        Dim currentCard As String

        player.valuedCount = 1
        If player.valuedCount <> player.dealtCards + 1 Then
            For i As Integer = player.valuedCount To (player.cards.Count - 1)
                currentCard = player.cards(i)
                If currentCard.Length = 2 Then
                    If IsNumeric(currentCard.Substring(0, 1)) = False Then
                        Select Case currentCard.Substring(0, 1)
                            Case "A"
                                cardType = 1
                            Case "J", "Q", "K"
                                cardType = 10
                        End Select
                    Else
                        cardType = currentCard.Substring(0, 1)
                    End If
                Else
                    cardType = currentCard.Substring(0, 2)
                End If
                If IsNumeric(cardType) = False Then
                    Select Case cardType
                        Case "A"
                            cardType = 1
                        Case "J", "Q", "K"
                            cardType = 10
                    End Select
                End If
                player.valuedCount += 1
                getPlayerHandValueWithAces += cardType
            Next
        End If

        Return getPlayerHandValueWithAces

    End Function

    'A modified version of the getPlayerHandValue function, but you pass a hand
variable as a parameter which indicated which of the player's two split hands to value
    Function getSplitHandValue(ByVal player As player, ByVal hand As Integer) As
Integer

        Dim cardType As Integer
        Dim currentCard As String

        If hand = 1 Then
            player.valuedCountArrayOne = 1
            If player.valuedCountArrayOne <> player.dealtCardsArrayOne + 1 Then
                For i As Integer = player.valuedCountArrayOne To
(player.splitCardsOne.Count - 1)
                    currentCard = player.splitCardsOne(i)
                    If currentCard.Length = 2 Then
                        If IsNumeric(currentCard.Substring(0, 1)) = False Then
                            Select Case currentCard.Substring(0, 1)
                                Case "A"
                                    cardType = 11
                                Case "J", "Q", "K"
                                    cardType = 10
```

```
                        End Select
                    Else
                        cardType = currentCard.Substring(0, 1)
                    End If
                Else
                    cardType = currentCard.Substring(0, 2)
                End If
                If IsNumeric(cardType) = False Then
                    Select Case cardType
                        Case "A"
                            cardType = 11
                        Case "J", "Q", "K"
                            cardType = 10
                    End Select
                End If
                player.valuedCountArrayOne += 1
                getSplitHandValue += cardType
            Next
        End If
    Else
        player.valuedCountArrayTwo = 1
        If player.valuedCountArrayTwo <> player.dealtCardsArrayTwo + 1 Then
            For i As Integer = player.valuedCountArrayTwo To
(player.splitCardsTwo.Count - 1)
                currentCard = player.splitCardsTwo(i)
                If currentCard.Length = 2 Then
                    If IsNumeric(currentCard.Substring(0, 1)) = False Then
                        Select Case currentCard.Substring(0, 1)
                            Case "A"
                                cardType = 11
                            Case "J", "Q", "K"
                                cardType = 10
                        End Select
                    Else
                        cardType = currentCard.Substring(0, 1)
                    End If
                Else
                    cardType = currentCard.Substring(0, 2)
                End If
                If IsNumeric(cardType) = False Then
                    Select Case cardType
                        Case "A"
                            cardType = 11
                        Case "J", "Q", "K"
                            cardType = 10
                    End Select
                End If
                player.valuedCountArrayOne += 1
                getSplitHandValue += cardType
            Next
        End If
    End If

    Return getSplitHandValue

End Function

    'A modified version of the getPlayerHandValue function, but you pass a hand
variable as a parameter which indicated which of the player's two split hands to
value. But, Ace cards are equal to 1 not 11
    Function getSplitHandValueWithAces(ByVal player As player, ByVal hand As Integer)
As Integer
```

```
        Dim cardType As Integer
        Dim currentCard As String

        If hand = 1 Then
            player.valuedCountArrayOne = 1
            If player.valuedCountArrayOne <> player.dealtCardsArrayOne + 1 Then
                For i As Integer = player.valuedCountArrayOne To
(player.splitCardsOne.Count - 1)
                    currentCard = player.splitCardsOne(i)
                    If currentCard.Length = 2 Then
                        If IsNumeric(currentCard.Substring(0, 1)) = False Then
                            Select Case currentCard.Substring(0, 1)
                                Case "A"
                                    cardType = 1
                                Case "J", "Q", "K"
                                    cardType = 10
                            End Select
                        Else
                            cardType = currentCard.Substring(0, 1)
                        End If
                    Else
                        cardType = currentCard.Substring(0, 2)
                    End If
                    If IsNumeric(cardType) = False Then
                        Select Case cardType
                            Case "A"
                                cardType = 1
                            Case "J", "Q", "K"
                                cardType = 10
                        End Select
                    End If
                    player.valuedCountArrayOne += 1
                    getSplitHandValueWithAces += cardType
                Next
            End If
        Else
            player.valuedCountArrayTwo = 1
            If player.valuedCountArrayTwo <> player.dealtCardsArrayTwo + 1 Then
                For i As Integer = player.valuedCountArrayTwo To
(player.splitCardsTwo.Count - 1)
                    currentCard = player.splitCardsTwo(i)
                    If currentCard.Length = 2 Then
                        If IsNumeric(currentCard.Substring(0, 1)) = False Then
                            Select Case currentCard.Substring(0, 1)
                                Case "A"
                                    cardType = 11
                                Case "J", "Q", "K"
                                    cardType = 10
                            End Select
                        Else
                            cardType = currentCard.Substring(0, 1)
                        End If
                    Else
                        cardType = currentCard.Substring(0, 2)
                    End If
                    If IsNumeric(cardType) = False Then
                        Select Case cardType
                            Case "A"
                                cardType = 11
                            Case "J", "Q", "K"
                                cardType = 10
```

```
                    End Select
                End If
                player.valuedCountArrayOne += 1
                getSplitHandValueWithAces += cardType
            Next
        End If
    End If

    Return getSplitHandValueWithAces

End Function

'Returns the picturebox of a card given as a string parameter.
Function returnCardBox(ByVal card As String) As PictureBox

    Dim cardType As String
    Dim cardSuit As String
    If card.Length = 2 Then
        cardType = card.Substring(0, 1)
        cardSuit = card.Substring(1, 1)
    Else
        cardType = card.Substring(0, 2)
        cardSuit = card.Substring(2, 1)
    End If

    For Each pictureBox In PlayingBoard.Controls
        If pictureBox.name = "pb" & cardType & cardSuit Then
            Return pictureBox
        End If
    Next
    Return Nothing

End Function

'Will move a card on the x axis using a value given as a parameter.
Sub moveCardX(ByVal card As PictureBox, ByVal x As Integer)

    card.Location = New Point(card.Location.X + x, card.Location.Y)

End Sub

'Will move a card on the y axis using a value given as a parameter.
Sub moveCardY(ByVal card As PictureBox, ByVal y As Integer)

    card.Location = New Point(card.Location.X, card.Location.Y + y)

End Sub

'Starts the simulation code
'Hides all the buttons as they are not used when simulating hands
Sub startSimulation()

    resetHintLabels()
    numberOfSimulatedPlayers = MainMenu.simulatedPlayers

    Dim currentPlayerTurn As Integer = 1
    Dim pauseTime As Integer = 1000

    PlayingBoard.btnNewRound.Hide()
    hideButtons()
    hideHintLabels()
    hideAllChipImages()
```

```
        cards = shuffleCards(generateCards())
        dealCardsToSimulatedPlayerObjects(cards)
        setDealtCards()
        For i As Integer = 1 To MainMenu.simulatedPlayers
            MainMenu.simPlayer(i).updateChips()
        Next
        Application.DoEvents()
        System.Threading.Thread.Sleep(pauseTime)
        calculateSimulationBets()
        For i As Integer = 1 To MainMenu.simulatedPlayers
            MainMenu.simPlayer(i).updateChips()
        Next
        dealFirstCardsToSimulatedPlayersVisually()
        System.Threading.Thread.Sleep(pauseTime)

        'GAME BEGINS
        'Blackjack check
        'Sets the simulatedPlayer property called hasBlackjack to True, used when
allocating winnings
        For i As Integer = 1 To MainMenu.simulatedPlayers
            If getSimulatedPlayerHandValue(MainMenu.simPlayer(i)) = 21 Then
                MainMenu.simPlayer(i).hasBlackjack = True
                MsgBox(MainMenu.simPlayer(i).name & " has a blackjack!")
                MainMenu.simPlayer(i).finishedTurn = True
            End If
        Next
        If getDealerHandValue() = 21 Then
            MainMenu.dealer.hasBlackJack = True
        End If

        'Turn system
        For i As Integer = 1 To MainMenu.simulatedPlayers
            currentSimulationPlayer = MainMenu.simPlayer(i)
            returnLabelName(currentSimulationPlayer.number, "Name").BackColor =
Color.Red
            If currentSimulationPlayer.finishedTurn = False Then
                Do Until getSimulatedPlayerHandValue(currentSimulationPlayer) > 16
                    MsgBox(currentSimulationPlayer.name & " has decided to hit because
his hand value is only " & getSimulatedPlayerHandValue(currentSimulationPlayer) & ".")
                    dealCardToSimulationPlayer(currentSimulationPlayer)
                Loop
                If getSimulatedPlayerHandValue(currentSimulationPlayer) > 21 Then
                    MsgBox(currentSimulationPlayer.name & " is now bust! His hand
value is " & getSimulatedPlayerHandValue(currentSimulationPlayer) & " which is greater
than 21!")
                Else
                    MsgBox(currentSimulationPlayer.name & " has decided to stick
becuase his hand value is " & getSimulatedPlayerHandValue(currentSimulationPlayer) &
".")
                End If
                currentSimulationPlayer.finishedTurn = True
                System.Threading.Thread.Sleep(500)
            End If
            showPlayerStatHints()
            returnLabelName(currentSimulationPlayer.number, "Name").BackColor =
Color.Transparent
        Next
        'Turns are done
        dealerTurn()
        calculateSimulationWinners()
        For i As Integer = 1 To MainMenu.simulatedPlayers
            MainMenu.simPlayer(i).updateChips()
```

```
        Next
        roundFinished = True
        PlayingBoard.btnNewRound.Show()

    End Sub

    'Calculates which simulated players have won and explains how they have won
    Private Sub calculateSimulationWinners()

        For i As Integer = 1 To MainMenu.simulatedPlayers
            MainMenu.simPlayer(i).handValue =
getSimulatedPlayerHandValue(MainMenu.simPlayer(i))
            'Checks if the player has a blackjack
            If MainMenu.simPlayer(i).hasBlackjack = True Then
                calculateSimulationBetReturn(MainMenu.simPlayer(i), 2)
                MsgBox(MainMenu.simPlayer(i).name & " got a blackjack and won ¢" &
(Math.Floor(MainMenu.simPlayer(i).betAmount * 2)) & ".")
                'Checks if they have an equal handValue
            ElseIf (MainMenu.simPlayer(i).handValue = MainMenu.dealer.handValue) And
(MainMenu.simPlayer(i).handValue < 21) Then
                MsgBox("Its a tie! " & MainMenu.simPlayer(i).name & " won back his bet
of ¢" & MainMenu.simPlayer(i).betAmount & " because he has the same hand value as the
dealer! (" & MainMenu.simPlayer(i).handValue & ")")
                calculateSimulationBetReturn(MainMenu.simPlayer(i), 1)
                'Checks if the player has a handTotal higher than the dealer but its
less than 22, or that it is less than 22 and the dealer went bust.
            ElseIf (MainMenu.simPlayer(i).handValue > MainMenu.dealer.handValue And
MainMenu.simPlayer(i).handValue < 22) Then
                MsgBox(MainMenu.simPlayer(i).name & " has won ¢" &
(Math.Floor(MainMenu.simPlayer(i).betAmount * 1.5)) & " becasue his hand value is
higher than the dealer!" & MainMenu.simPlayer(i).name & " has a value of " &
MainMenu.simPlayer(i).handValue & " and the dealer has a value of " &
MainMenu.dealer.handValue & "!")
                calculateSimulationBetReturn(MainMenu.simPlayer(i), 1.5)
            ElseIf (MainMenu.simPlayer(i).handValue < 22 And MainMenu.dealer.handValue
> 21) Then
                MsgBox(MainMenu.simPlayer(i).name & " has won ¢" &
(Math.Floor(MainMenu.simPlayer(i).betAmount * 1.5)) & " because the dealer went
bust!")
                calculateSimulationBetReturn(MainMenu.simPlayer(i), 1.5)
            ElseIf (MainMenu.simPlayer(i).handValue < MainMenu.dealer.handValue) And
MainMenu.dealer.handValue < 22 Then
                MsgBox(MainMenu.simPlayer(i).name & " has not won anything because the
dealer has a higher handvalue!" & MainMenu.simPlayer(i).name & " has a value of " &
MainMenu.simPlayer(i).handValue & " and the dealer has a value of " &
MainMenu.dealer.handValue & "!")
            Else
                MsgBox(MainMenu.simPlayer(i).name & " has not won anything because he
went bust!")
            End If
            MainMenu.simPlayer(i).updateChips()
        Next

    End Sub

    'Calculates the bet returns for each simulated player based on their bets
    Function calculateSimulationBetReturn(ByVal player As simulated_Player, ByVal
multiplyer As Integer) As Integer

        calculateSimulationBetReturn = (player.betAmount * multiplyer)
        player.balance += calculateSimulationBetReturn
```

```
    End Function

    'Hids the hint labels as they are not needed during a simulation
    Private Sub hideHintLabels()
        PlayingBoard.lblHandTotalTitle.Hide()
    End Sub

    'Deals a card to the simulation player.
    'Used when hitting another card
    Sub dealCardToSimulationPlayer(ByVal player As simulated_Player)

        player.cards.Add(cards.ElementAt(0))
        cards.RemoveAt(0)

        yDone = False
        xDone = False
        playerToRecieveCard = player.number
        currentDealtSimulationPlayer = player
        Dim playerCard =
currentDealtSimulationPlayer.cards(currentDealtSimulationPlayer.dealtCards)
        currentCardToMove = returnCardBox(playerCard)
        moveSimulatedPlayerCard()
        currentDealtSimulationPlayer.dealtCards += 1

    End Sub

    'Calculates the bets for each player by randomly generating a number
    Private Sub calculateSimulationBets()

        Dim currentPlayer As simulated_Player
        Dim randomInt As Integer
        Randomize()

        For i As Integer = 1 To MainMenu.simulatedPlayers

            currentPlayer = MainMenu.simPlayer(i)
            randomInt = Int(Rnd() * 4)
            Select Case randomInt
                Case 1
                    currentPlayer.betAmount = 100
                Case 2
                    currentPlayer.betAmount = 50
                Case 3
                    currentPlayer.betAmount = 25
                Case Else
                    currentPlayer.betAmount = 10
            End Select
            If currentPlayer.balance - currentPlayer.betAmount < 0 Then
                currentPlayer.betAmount = 0
            End If
            MsgBox(currentPlayer.name & " has bet ¢" & currentPlayer.betAmount & "!")
            currentPlayer.balance -= currentPlayer.betAmount
            System.Threading.Thread.Sleep(100)
        Next

        updateSimulatedPlayerStats()

    End Sub

    'Deals cards to the objects of each simulated player
    Private Sub dealCardsToSimulatedPlayerObjects(cards As List(Of String))
```

```
    For i As Integer = 1 To 2
        For k As Integer = 1 To MainMenu.simulatedPlayers
            MainMenu.simPlayer(k).cards.Add(cards.ElementAt(i))
            cards.RemoveAt(i)
        Next
    Next

    For i As Integer = 1 To 2
        MainMenu.dealer.cards.Add(cards.ElementAt(0))
        cards.RemoveAt(0)
    Next

End Sub

'Gets the handValue for a simulated player
Function getSimulatedPlayerHandValue(ByRef player As simulated_Player) As Integer

    Dim cardType As Integer
    Dim currentCard As String
    player.valuedCount = 1
    If player.valuedCount <> player.dealtCards + 1 Then
        For i As Integer = player.valuedCount To (player.cards.Count - 1)
            currentCard = player.cards(i)
            If currentCard.Length = 2 Then
                If IsNumeric(currentCard.Substring(0, 1)) = False Then
                    Select Case currentCard.Substring(0, 1)
                        Case "A"
                            cardType = 11
                        Case "J", "Q", "K"
                            cardType = 10
                    End Select
                Else
                    cardType = currentCard.Substring(0, 1)
                End If
            Else
                cardType = currentCard.Substring(0, 2)
            End If
            If IsNumeric(cardType) = False Then
                Select Case cardType
                    Case "A"
                        cardType = 11
                    Case "J", "Q", "K"
                        cardType = 10
                End Select
            End If
            player.valuedCount += 1
            getSimulatedPlayerHandValue += cardType
        Next
    End If

    Return getSimulatedPlayerHandValue

End Function

'Updates the simulated player stats
Sub updateSimulatedPlayerStats()

    For i As Integer = 1 To MainMenu.simulatedPlayers
        displaySimulatedPlayerStats(MainMenu.simPlayer(i), i)
    Next

End Sub
```

```
    'Displays the simulated players stats on the board (username and balance)
    Sub displaySimulatedPlayerStats(ByVal player As simulated_Player, ByVal
playerNumber As Integer)

        Dim nameLabel As Label = returnLabelName(playerNumber, "Name")
        Dim balanceLabel As Label = returnLabelName(playerNumber, "Balance")
        nameLabel.Text = player.name
        balanceLabel.Text = "¢" & player.balance

    End Sub

    'Deals cards to the simulated players using the pictureboxes
    Sub dealFirstCardsToSimulatedPlayersVisually()

        For k As Integer = 1 To 2
            For i As Integer = 1 To MainMenu.simulatedPlayers
                yDone = False
                xDone = False
                playerToRecieveCard = i
                currentDealtSimulationPlayer = MainMenu.simPlayer(i)
                Dim playerCard =
currentDealtSimulationPlayer.cards(currentDealtSimulationPlayer.dealtCards)
                currentCardToMove = returnCardBox(playerCard)
                moveSimulatedPlayerCard()
                currentDealtSimulationPlayer.dealtCards += 1
            Next

            'Deals the dealer their 2 cards
            yDone = False
            xDone = False
            playerToRecieveCard = 0
            Dim dealerCard = MainMenu.dealer.cards(MainMenu.dealer.dealtCards)
            currentCardToMove = returnCardBox(dealerCard)
            If k = 2 Then
                hideDealerCard()
            End If
            moveSimulatedPlayerCard()
            MainMenu.dealer.dealtCards += 1
        Next

    End Sub

    'Deals a card to the playerToRecieveCard
    Private Sub moveSimulatedPlayerCard()

        Dim xAddition As Integer
        Dim dealerXAddition As Integer
        Dim dealerCardPos = New Integer() {430, 40}
        Dim player1CardPos = New Integer() {910, 118}
        Dim player2CardPos = New Integer() {910, 360}
        Dim player3CardPos = New Integer() {422, 510}
        Dim player4CardPos = New Integer() {10, 310}

        currentCardToMove.BringToFront()
        finishedDealing = False

        If currentDealtSimulationPlayer.dealtCards > 1 Then
            Select Case currentDealtSimulationPlayer.dealtCards
                Case 2
                    xAddition = 6.25 * (currentDealtSimulationPlayer.dealtCards)
                Case Else
```

```
                xAddition = 12.5 * (currentDealtSimulationPlayer.dealtCards - 1)
            End Select
        End If

        Select Case playerToRecieveCard
            Case 0
                If MainMenu.dealer.dealtCards > 1 Then
                    Select Case MainMenu.dealer.dealtCards
                        Case 2
                            dealerXAddition = 6.25 * (MainMenu.dealer.dealtCards)
                        Case Else
                            dealerXAddition = 12.5 * (MainMenu.dealer.dealtCards - 1)
                    End Select
                End If
                initiateCardMove(dealerCardPos, dealerXAddition, ">", ">", -5, -5)
            Case 1
                initiateCardMove(player1CardPos, xAddition, "<", ">", 5, -5)
            Case 2
                initiateCardMove(player2CardPos, xAddition, "<", "<", 5, 5)
            Case 3
                initiateCardMove(player3CardPos, xAddition, ">", "<", -5, 5)
            Case 4
                initiateCardMove(player4CardPos, xAddition, ">", "<", -5, 5)
        End Select

        finishedDealing = True

    End Sub

    'Will form the algorithm to move a card based upon the cards target position, the
xAddition for the card, and then the sign for x and y which relates to whether it is
moving it on the x/y axis until its location is < or > the target location.
    'yMove and xMove relate to which direction to move the card, eg whether it needs
to be increased/decreased in the x/y co-ordinates
    Sub initiateCardMove(ByVal cardPos As Integer(), ByVal xAddition As Integer, ByVal
xSign As Char, ByVal ySign As Char, ByVal xMove As Integer, ByVal yMove As Integer)


        Dim baseX As Integer = cardPos(0)
        Dim baseY As Integer = cardPos(1)

        Do While (xDone = False Or yDone = False)
            If xSign = "<" Then
                initiateCardMoveXLessThan(baseX, xAddition, xMove)
            Else
                initiateCardMoveXMoreThan(baseX, xAddition, xMove)
            End If
            If ySign = "<" Then
                initiateCardMoveYLessThan(baseY, yMove)
            Else
                initiateCardMoveYMoreThan(baseY, yMove)
            End If
        Loop
        currentCardToMove.Location = New Point(baseX + xAddition, baseY)

    End Sub

    'This will move a card if it needs to be reduced on the x axis, until it is below
the baseX + xAddition, xMove number of units.
    Sub initiateCardMoveXLessThan(ByVal baseX As Integer, ByVal xAddition As Integer,
ByVal xMove As Integer)
```

```
        If currentCardToMove.Location.X < baseX + xAddition Then
            moveCardX(currentCardToMove, xMove)
            Application.DoEvents()
            System.Threading.Thread.Sleep(10)
        Else
            xDone = True
        End If

    End Sub

    'This will move a card if it needs to be reduced on the x axis, until it is above
the baseX + xAddition, xMove number of units.
    Sub initiateCardMoveXMoreThan(ByVal baseX As Integer, ByVal xAddition As Integer,
ByVal xMove As Integer)

        If currentCardToMove.Location.X > baseX + xAddition Then
            moveCardX(currentCardToMove, xMove)
            Application.DoEvents()
            System.Threading.Thread.Sleep(10)
        Else
            xDone = True
        End If

    End Sub

    'This will move a card if it needs to be reduced on the y axis, until it is below
the baseY, yMove number of units.
    Sub initiateCardMoveYLessThan(ByVal baseY As Integer, ByVal yMove As Integer)

        If currentCardToMove.Location.Y < baseY Then
            moveCardY(currentCardToMove, yMove)
            Application.DoEvents()
            System.Threading.Thread.Sleep(10)
        Else
            yDone = True
        End If

    End Sub

    'This will move a card if it needs to be increased on the y axis, until it is
below the baseY, yMove number of units.
    Sub initiateCardMoveYMoreThan(ByVal baseY As Integer, ByVal yMove As Integer)

        If currentCardToMove.Location.Y > baseY Then
            moveCardY(currentCardToMove, yMove)
            Application.DoEvents()
            System.Threading.Thread.Sleep(10)
        Else
            yDone = True
        End If

    End Sub

    'Resets the board for another simulated round
    Sub resetSimulationBoard()

        PlayingBoard.pbTopOfDeck.BringToFront()
        showDealerCard()
        hideAllChipImages()
        roundFinished = False
        'Resets all the cards to the correct position
        For Each card In PlayingBoard.Controls
```

```
            If card.width = 64 Then
                card.location = New Point(663, 230)
            End If
        Next
        PlayingBoard.pbTopOfDeck.BringToFront()
        For i As Integer = 1 To MainMenu.simulatedPlayers
            MainMenu.simPlayer(i).reset()
        Next
        MainMenu.dealer.reset()
        PlayingBoard.pbTopOfDeck.BringToFront()

    End Sub

    'This function will return an array containing the 5 players inside the
leaderboard text document
    Function readLeaderboards() As String()

        Dim fileHandle As IO.StreamReader = New IO.StreamReader("Leaderboard.txt")
        Dim leaderboard As String = fileHandle.ReadLine()
        fileHandle.Close()
        readLeaderboards = leaderboard.Split(":")

    End Function

    'This subroutine uses the quicksort algorithm on a string array
    'This quicksort will compare the substring which contains an integer value
representing the players balance, and then replace those according around the pivot.
    Sub quickSortLeaderboard(ByRef array As String(), ByVal low As Integer, ByVal high
As Integer)

        Dim tempStorage As String
        Dim lowerIndex As Integer = low
        Dim upperIndex As Integer = high
        Dim pivot As String = array((lowerIndex + upperIndex) \ 2)

        Dim pivotSubstring As Integer = getPivotSubstringCount(pivot)
        Dim lowerSubstring As Integer = getArraySubstringCount(array, lowerIndex)
        Dim upperSubstring As Integer = getArraySubstringCount(array, upperIndex)

        Do Until lowerIndex > upperIndex
            Do While (CInt(array(lowerIndex).Substring(0, lowerSubstring))) <
(CInt(pivot.Substring(0, pivotSubstring)))
                lowerIndex += 1
                lowerSubstring = getArraySubstringCount(array, lowerIndex)
            Loop

            Do While (CInt(array(upperIndex).Substring(0, upperSubstring))) >
CInt((pivot.Substring(0, pivotSubstring)))
                upperIndex -= 1
                upperSubstring = getArraySubstringCount(array, upperIndex)
            Loop

            If lowerIndex <= upperIndex Then
                tempStorage = array(lowerIndex)
                array(lowerIndex) = array(upperIndex)
                array(upperIndex) = tempStorage
                lowerIndex += 1
                upperIndex -= 1
            End If
        Loop

        If (lowerIndex < high) Then
```

```
            quickSortLeaderboard(array, lowerIndex, high)
        End If

        If (upperIndex > low) Then
            quickSortLeaderboard(array, low, upperIndex)
        End If

    End Sub

    'This funtion will get the substring required to read the integer value inside the
array positin index to refer to the  balance inside the string.
    'This is done because the array stores a string, with a balance followed by a .
followed by a name eg "500.Josh", because i want to quickSort the balances, i need to
only compare the balances on not the whole string
    Function getArraySubstringCount(ByVal array As String(), ByVal index As Integer)
As Integer

        Dim substringCount As Integer = 0
        Do Until array(index).Substring((substringCount), 1) = "."
            substringCount += 1
        Loop
        getArraySubstringCount = substringCount

    End Function

    'This function will get the substring required to read the integer value inside
the pivot string
    'This is done because the array stores a string, with a balance followed by a .
followed by a name eg "500.Josh", because i want to quickSort the balances, i need to
only compare the balances on not the whole string
    Function getPivotSubstringCount(ByVal pivot As String) As Integer

        Dim substringCount As Integer = 0
        Do Until pivot.Substring(substringCount, 1) = "."
            substringCount += 1
        Loop
        getPivotSubstringCount = substringCount

    End Function

    'This subroutine will load the 5 players inside the leaderboards txt document into
the leaderboardPlayers array.
    'A for loop will then loop through from 0 - 5 displaying the correct balances and
names from the strings inside the leaderboardPlayers array
    Sub displayLeaderboards()

        Dim leaderboardPlayers() As String = readLeaderboards()

        For i As Integer = 0 To 4
            Dim count As Integer = 0
            While leaderboardPlayers(i).Substring(count, 1) <> "."
                count += 1
            End While
            Select Case i
                Case 0
                    Leaderboard.lblFirstBalance.Text = "¢" &
leaderboardPlayers(i).Substring(0, (count))
                    Leaderboard.lblFirstName.Text =
leaderboardPlayers(i).Substring(count + 1, CInt(leaderboardPlayers(i).Length) - (count
+ 1)) & " -"
                Case 1
```

```
                    Leaderboard.lblSecondBalance.Text = "¢" &
leaderboardPlayers(i).Substring(0, (count))
                    Leaderboard.lblSecondName.Text =
leaderboardPlayers(i).Substring(count + 1, CInt(leaderboardPlayers(i).Length) - (count
+ 1)) & " -"
                Case 2
                    Leaderboard.lblThirdBalance.Text = "¢" &
leaderboardPlayers(i).Substring(0, (count))
                    Leaderboard.lblThirdName.Text =
leaderboardPlayers(i).Substring(count + 1, CInt(leaderboardPlayers(i).Length) - (count
+ 1)) & " -"
                Case 3
                    Leaderboard.lblFourthBalance.Text = "¢" &
leaderboardPlayers(i).Substring(0, (count))
                    Leaderboard.lblFourthName.Text =
leaderboardPlayers(i).Substring(count + 1, CInt(leaderboardPlayers(i).Length) - (count
+ 1)) & " -"
                Case 4
                    Leaderboard.lblFifthBalance.Text = "¢" &
leaderboardPlayers(i).Substring(0, (count))
                    Leaderboard.lblFifthName.Text =
leaderboardPlayers(i).Substring(count + 1, CInt(leaderboardPlayers(i).Length) - (count
+ 1)) & " -"
            End Select
        Next

    End Sub

End Module
```

Utility Methods Module (UtilityMethods.vb)

Code

```
Module UtilityMethods

    'Encrypts an input given with the pass given
    'Used in encrypting the users balance
    '@ Origional author Shane Gowland from Stack Overflow.
    Public Function encrypt(ByVal input As String, ByVal pass As String) As String
        Dim AES As New System.Security.Cryptography.RijndaelManaged
        Dim Hash_AES As New System.Security.Cryptography.MD5CryptoServiceProvider
        Dim encrypted As String = ""
        Try
            Dim hash(31) As Byte
            Dim temp As Byte() =
Hash_AES.ComputeHash(System.Text.ASCIIEncoding.ASCII.GetBytes(pass))
            Array.Copy(temp, 0, hash, 0, 16)
            Array.Copy(temp, 0, hash, 15, 16)
            AES.Key = hash
            AES.Mode = Security.Cryptography.CipherMode.ECB
            Dim DESEncrypter As System.Security.Cryptography.ICryptoTransform =
AES.CreateEncryptor
            Dim Buffer As Byte() = System.Text.ASCIIEncoding.ASCII.GetBytes(input)
            encrypted =
Convert.ToBase64String(DESEncrypter.TransformFinalBlock(Buffer, 0, Buffer.Length))
            Return encrypted
        Catch ex As Exception
        End Try
    End Function

    'Decrypts an input using the pass given.
    'Used when decrypting the users balance for display.
```

```
    '@ Origional author Shane Gowland from Stack Overflow.
    Public Function decrypt(ByVal input As String, ByVal pass As String) As String
        Dim AES As New System.Security.Cryptography.RijndaelManaged
        Dim Hash_AES As New System.Security.Cryptography.MD5CryptoServiceProvider
        Dim decrypted As String = ""
        Try
            Dim hash(31) As Byte
            Dim temp As Byte() =
Hash_AES.ComputeHash(System.Text.ASCIIEncoding.ASCII.GetBytes(pass))
            Array.Copy(temp, 0, hash, 0, 16)
            Array.Copy(temp, 0, hash, 15, 16)
            AES.Key = hash
            AES.Mode = Security.Cryptography.CipherMode.ECB
            Dim DESDecrypter As System.Security.Cryptography.ICryptoTransform =
AES.CreateDecryptor
            Dim Buffer As Byte() = Convert.FromBase64String(input)
            decrypted =
System.Text.ASCIIEncoding.ASCII.GetString(DESDecrypter.TransformFinalBlock(Buffer, 0,
Buffer.Length))
            Return decrypted
        Catch ex As Exception
        End Try

    End Function

End Module
```

Base Player Class (base_player.vb)

Code

```
Public Class base_player

    Private deck As New List(Of String)
    Private numberOfDealtCards As Integer
    Private cardsValue As Integer
    Private numberOfCardsAlreadyValued As Integer
    Private blackJack As Boolean
    Private money As Integer
    Private pNum As Integer
    Private username As String
    Private finishedGo As Boolean
    Private moneyBet As Integer

    'This subroutine will hide the player's chip images and then display the chip
images which correspond to the players balance.
    Public Sub updateChips()

        Dim balance As Integer = Me.balance
        Dim remainingBalance As Integer
        Dim chip500Count As Integer = 0
        Dim chip100Count As Integer = 0
        Dim chip25Count As Integer = 0
        Dim chip10Count As Integer = 0
        Dim chip5Count As Integer = 0

        Me.hideChipImages()

        chip500Count = Math.Floor(balance / 500)
        remainingBalance = balance Mod 500
        If remainingBalance <> 0 Then
            chip100Count = Math.Floor(remainingBalance / 100)
            remainingBalance = remainingBalance Mod 100
```

```
            If remainingBalance <> 0 Then
                chip25Count = Math.Floor(remainingBalance / 25)
                remainingBalance = remainingBalance Mod 25
                If remainingBalance <> 0 Then
                    chip10Count = Math.Floor(remainingBalance / 10)
                    remainingBalance = remainingBalance Mod 10
                    If remainingBalance <> 0 Then
                        chip5Count = Math.Floor(remainingBalance / 5)
                        remainingBalance = remainingBalance Mod 5
                    End If
                End If
            End If
        End If

        Me.displayChipbox(500, chip500Count)
        Me.displayChipbox(100, chip100Count)
        Me.displayChipbox(25, chip25Count)
        Me.displayChipbox(10, chip10Count)
        Me.displayChipbox(5, chip5Count)

        If Settings.sounds = True Then
            My.Computer.Audio.Play(soundEffectPath & "Chips_Effect.wav")
        End If

        displayPlayerStats(Me)

    End Sub

    'This subroutine will display a chip when given the chip type (eg 500 chip) and
the chip number (eg the 5th chip)
    Sub displayChipbox(ByVal chipType As Integer, ByVal chipCount As Integer)

        If chipCount > 0 Then
            For i As Integer = 1 To chipCount
                returnChipBox(Me.number, chipType, i).Show()
            Next
        End If

    End Sub

    'This subroutine will hide all the chip images of a certain player
    Sub hideChipImages()

        Dim chipToHide As Integer
        Dim chipType() As Integer = {500, 100, 25, 10, 5}

        For j As Integer = 0 To 4
            chipToHide = chipType(j)
            For k As Integer = 1 To 4
                returnChipBox(Me.number, chipToHide, k).Hide()
            Next
        Next

    End Sub

    'This subroutine is called at the end of every round and handles the reset of the
player object to prepare for the next round.
    Public Overridable Sub reset()

        Me.cards.Clear()
        Me.valuedCount = 1
        Me.cards.Add("")
```

```
        Me.dealtCards = 0
        Me.finishedTurn = False

    End Sub

    Public Property name As String
        Get
            Return username
        End Get
        Set(value As String)
            username = value
        End Set
    End Property

    Public Property hasBlackjack As Boolean
        Get
            Return blackJack
        End Get
        Set(value As Boolean)
            blackJack = value
        End Set
    End Property

    Public Property number As Integer
        Get
            Return pNum
        End Get
        Set(value As Integer)
            pNum = value
        End Set
    End Property

    Public Property handValue As Integer
        Get
            Return cardsValue
        End Get
        Set(value As Integer)
            cardsValue = value
        End Set
    End Property

    Public Property balance As Integer
        Get
            Return money
        End Get
        Set(value As Integer)
            money = value
        End Set
    End Property

    Public Property valuedCount As Integer
        Get
            Return numberOfCardsAlreadyValued
        End Get
        Set(value As Integer)
            numberOfCardsAlreadyValued = value
        End Set
    End Property

    Public Property dealtCards As Integer
        Get
            Return numberOfDealtCards
```

```
        End Get
        Set(value As Integer)
            numberOfDealtCards = value
        End Set
    End Property

    Public Property cards() As List(Of String)
        Get
            Return deck
        End Get
        Set(value As List(Of String))
            deck = value
        End Set
    End Property

    Public Property finishedTurn As Integer
        Get
            Return finishedGo
        End Get
        Set(value As Integer)
            finishedGo = value
        End Set
    End Property

    Public Property betAmount As Integer
        Get
            Return moneyBet
        End Get
        Set(value As Integer)
            moneyBet = value
        End Set
    End Property

End Class
```

Player Class (player.vb)

Code

```
Public Class player
    Inherits base_player

    Private cardNumber As Integer
    Private numberOfDealtCards As Integer
    Private winner As Boolean
    Private splitSetOne As List(Of String)
    Private splitSetTwo As List(Of String)
    Private haveSplit As Boolean
    Private dSplit As Boolean
    Private nextCardArray As Integer
    Private Aces As Boolean
    Private doneArrayOne As Boolean
    Private doneArrayTwo As Boolean
    Private cardNumberArrayOne As Integer
    Private cardNumberArrayTwo As Integer
    Private valueCardArrayOne As Integer
    Private valueCardArrayTwo As Integer
    Private valueOne As Integer
    Private valueTwo As Integer

    'This sub is called everytime a player is created, it sets the player class up for
the start of the game
    Public Sub New()
```

```
        cards.Add("")
        valuedCount = 1
        finishedTurn = False
        hasSplit = False

    End Sub

    'This subroutine is called everytime a player splits their hand and it creates two
new lists to represent their hands and handles anything else which happens when you
split your hand
    Public Sub createSplit()

        splitSetOne = New List(Of String)
        splitSetTwo = New List(Of String)
        Me.splitCardsOne.Add("")
        Me.splitCardsTwo.Add("")
        Me.arrayToDealTo = 1

        Me.splitCardsOne.Add(Me.cards(1))
        Me.splitCardsTwo.Add(Me.cards(2))
        Me.dealtCardsArrayOne = 2
        Me.dealtCardsArrayTwo = 2
        Me.cards.Clear()

        Me.balance -= Me.betAmount
        Me.betAmount *= 2
        updateChips()

        splitHandsVisually(Me)

    End Sub

    'This subrotuine is used to change the arrayToDealTo property to the opposite,
this is done when dealing and addressing the current hand the dealer is dealing to
when a player is split
    Public Sub changeSplit()

        If Me.arrayToDealTo = 1 Then
            Me.arrayToDealTo = 2
        ElseIf Me.arrayToDealTo = 2 Then
            Me.arrayToDealTo = 1
        End If

    End Sub

    Public Property dealtCardsArrayOne As Integer
        Get
            Return cardNumberArrayOne
        End Get
        Set(value As Integer)
            cardNumberArrayOne = value
        End Set
    End Property

    Public Property handValueArrayOne As Integer
        Get
            Return valueOne
        End Get
        Set(value As Integer)
            valueOne = value
        End Set
```

```
End Property

Public Property handValueArrayTwo As Integer
    Get
        Return valueTwo
    End Get
    Set(value As Integer)
        valueTwo = value
    End Set
End Property

Public Property dealtCardsArrayTwo As Integer
    Get
        Return cardNumberArrayTwo
    End Get
    Set(value As Integer)
        cardNumberArrayTwo = value
    End Set
End Property

Public Property valuedCountArrayOne As Integer
    Get
        Return valueCardArrayOne
    End Get
    Set(value As Integer)
        valueCardArrayOne = value
    End Set
End Property

Public Property valuedCountArrayTwo As Integer
    Get
        Return valueCardArrayTwo
    End Get
    Set(value As Integer)
        valueCardArrayTwo = value
    End Set
End Property

Public Property finishedTurnArrayOne As Boolean
    Get
        Return doneArrayOne
    End Get
    Set(value As Boolean)
        doneArrayOne = value
    End Set
End Property

Public Property finishedTurnArrayTwo As Boolean
    Get
        Return doneArrayTwo
    End Get
    Set(value As Boolean)
        doneArrayTwo = value
    End Set
End Property

Public Property doneSplit As Boolean
    Get
        Return dSplit
    End Get
    Set(value As Boolean)
        dSplit = value
```

```
        End Set
    End Property

    Public Property hasAces As Boolean
        Get
            Return Aces
        End Get
        Set(value As Boolean)
            Aces = value
        End Set
    End Property

    Public Property arrayToDealTo As Integer
        Get
            Return nextCardArray
        End Get
        Set(value As Integer)
            nextCardArray = value
        End Set
    End Property

    Public Property hasSplit As Boolean
        Set(value As Boolean)
            haveSplit = value
        End Set
        Get
            Return haveSplit
        End Get
    End Property

    Public Property splitCardsOne() As List(Of String)
        Get
            Return splitSetOne
        End Get
        Set(value As List(Of String))
            splitSetOne = value
        End Set
    End Property

    Public Property splitCardsTwo() As List(Of String)
        Get
            Return splitSetTwo
        End Get
        Set(value As List(Of String))
            splitSetTwo = value
        End Set
    End Property

End Class
```

Simulation Player Class (simulated_Player.vb)

Code

```
Public Class simulated_Player

    Inherits base_player

    'This subroutine is called every time a simulate_Player is created and it sets up
the simulated player for the first simulated round
    Public Sub New()

        cards.Add("")
```

```
        valuedCount = 1
        finishedTurn = False

    End Sub

End Class
```

AI Player Class (AI_Player.vb)

Code

```
Public Class AI_Player
    Inherits base_player

    'This sub is called every time an AI_Player is created, and is used to setup the
object for the game
    Public Sub New()

        cards.Add("")
        valuedCount = 1

    End Sub

    'This reset sub is modified to set the dealtCards property equal to 1 due to
differences with the dealer and player dealing.
    Public Overrides Sub reset()

        MyBase.reset()
        Me.dealtCards = 1

    End Sub

End Class
```

How to Run the Program

To run the program, you will need to load it inside the Visual Studio IDE due to the editing of file paths required. At the top of the GameMethods.vb file, there are two constants which list the file paths for the card images, and the sound effects. Simply change the drive letters to match wherever you are running the program from. After this, simply compile the program and it should run perfectly. There are no text documents you need to create or delete.

Testing

In this section I will be performing tests which allow me to ensure my created solution works as intended. Wherever there is the option to input data, I will be testing this for boundary data, typical data (normal data) and finally erroneous data. Boundary data refers to data which is expected to be entered by the user, but is on the edge of valid and invalid, for example if I expect a number between 5 and 10, the number 10 would be boundary data because it is on the edge of the bound. Typical data is data which the program expects to be entered. Using the 5 to 10 example from before, the number 7 would be typical data because it is between 5 and 10, and a number. Finally, erroneous data is data which is unexpected by the solution, using the same example, a string, would be classed as erroneous data because the program is expecting a number which is between 5 and 10.

Due to minimal implementation of user input in my program, testing for boundary and erroneous data will be limited. However, I will be testing all areas of my program such as button click events, and data presentation.

Tests

Form: MainMenu					
Test Number	Test Description	Test Data (Input)	Expected Results (Output)	Actual Results (Output)	Test Pass
1	When the program loads, does the MainMenu form open?	**Normal Data:** Click run on the program	The MainMenu form will load.	The MainMenu form loads.	Yes.
2	When you click the "New Game" button, you are asked how many players you want to play?	**Normal Data:** Clicking the "New Game" button.	User is asked how many players they want to add.	An input box loads asking the user "How many players are there? (1-4)"	Yes.
3a	Typing the number of players after clicking the "New Game" button using *normal data* and clicking OK will be accepted.	**Normal data:** "2" is entered in the box.	The game should accept the value and return to the MainMenu form with 0/2 players loaded in the lobby.	No error was given, and the lobby label now says, "Players 0/2".	Yes.
3b	Typing the number of players after clicking the "New Game" button using *boundary data* and clicking OK will be accepted.	**Boundary data:** 4.1 is entered in the input box.	A message box appears saying the number of players cannot be less than 0 or greater than 4.	The value of 4.1 was not accepted and a message box was presented stating that the number of players cannot be less than 0 or greater	Yes.

				than 4. The user was then redirected to the MainMenu form without any players loaded.	
3c	Typing the number of players after clicking the "New Game" button using *erroneous data* and clicking OK will be rejected.	**Erroneous data:** No input is entered to the input box.	A message box should appear which states that "The number of players cannot be 0"!	An empty entry is not accepted, and the user is told that the number of players cannot be 0. They are then returned to the MainMenu form and no players are loaded into the game.	Yes
4	Clicking the "Add Player" button before creating a new game will not load the AddPlayer form.	**Normal Data:** Clicking the "Add Player" button before creating a game.	A message box appears not allowing you to add players before you create a game.	A message box saying "Start a new game before adding players! Click the New Game button!" appeared, once you click OK, you are redirected to the MainMenu form.	Yes
5	Clicking the "Add Player" button when all the players are loaded will not allow any new players to be added.	**Normal Data:** Create a new game with 2 players, load those 2 players and then try to add another player.	The game gives an error because the max players are already loaded.	A message box appears saying "Max players reached!" and the player is taken back to the MainMenu form.	Yes
6	If the max players are not already loaded and you have created a new game, when you click the "Add Player" button, you are taken to the	**Normal Data:** Creating a game with four players, loading two and then trying to add a third.	The AddPlayer form is loaded and the main menu is temporarily hidden.	Upon clicking the "Add Player" button, the player is taken to the AddPlayer	Yes

	AddPlayer form.			form whilst the MainMenu form is hidden.	
7	The "Settings button" is clicked and leads the player to the Settings form.	**Normal Data:** Clicking the settings button.	The Settings form Is opened and the MainMenu form is temporarily hidden from view.	Upon clicking the "Settings" button, the Settings form is loaded correctly and the MainMenu form is hidden.	Yes
8	Clicking "Leaderboard" button will load Leaderboard form.	**Normal Data:** Clicking the "Leaderboard" button.	The Leaderboard form loads and the MainMenu form is hidden.	The Leaderboard form is loaded correctly and the MainMenu form is hidden from view.	Yes
9	Clicking the "End Game" button should terminate the program.	**Normal Data:** Clicking the "End Game" button.	The MainMenu form is closed therefore stopping the program.	The program gets closed.	Yes
10a	Typing the number of players after clicking the simulation button using *normal data* and clicking OK.	**Normal Data:** The number 2 is entered in the simulated player box.	The PlayingBoard form is loaded with 2 AI players and the dealer.	The value is accepted and the PlayingBoard form is loaded with 2 simulated players loaded and the dealer.	Yes
10b	Typing the number of players after clicking the simulation button using *boundary data* and clicking OK.	**Boundary Data:** 4 simulated players are entered in the input box.	The PlayingBoard form is loaded with 4 Ai players and the dealer.	The value is accepted and the PlayingBoard form is loaded with 4 simulated players and the dealer.	Yes
10c	Typing the number of players after clicking the simulation button using *erroneous data* and clicking OK.	**Erroneous Data:** Entering "five" into the input box.	The player is presented with a message box stating that a number is required.	The value is rejected, and a message box appears saying "You must enter a numeric value!"	Yes

11	When a player is loaded, their name and balance is displayed in the lobby.	**Normal Data:** Loading a player called "Andrew" with a balance of ¢1000.	The lobby is updated displaying the name of the loaded player along with their balance.	The lobby changed to display "Andrew ¢1000"	Yes
Form: AddPlayer					
Test Number	Test Description	Test Data (Input)	Expected Results (Output)	Actual Results (Output)	Test Pass
12	If the user clicks the "New Profile" button, they are asked to enter their desired username.	**Normal Data:** Clicking the "New Profile" button.	An input box will open asking the user to enter their desired username.	An input box appeared asking the user to enter the username they wish to use.	Yes
13	If you enter a username into the input box, value is accepted without error and the new profile is displayed using *normal data*.	**Normal Data:** Entering "James" into the username box.	The value is accepted, and a profile is created with the name James and balance ¢1000.	No errors are presented and the username "James" is loaded with a balance of ¢1000.	Yes
14a	If you enter a username into the input box, the value is accepted without error and the new profile is displayed using *boundary data*.	**Boundary Data:** A username of length 12 is entered into the box to create a new profile, "qqqqqqqqqqqq"	This username is accepted as a username must be up to 12 characters.	No error given, and a profile is created with the username of "qqqqqqqqqqqq" and a balance of ¢1000.	Yes
14b	If you enter a username into the input box, the value is rejected with an error using *erroneous data*.	**Erroneous Data** A 15-letter username is entered into the input box for a new profile. "aaaaaaaaaaaaaaa"	The username should not be accepted, and an appropriate error message presented to the user.	The username was not accepted, and a message box was presented saying "You cannot have a username greater than 12 characters!"	Yes
14c	If the user selects the "Load Profile" button, an input box appears asking them what their username is.	**Normal Data:** Clicking the "Load Profile" button.	An input box appears which asks the player what their username is.	An input box appeared on the screen and asked the player to "Enter your	Yes

				username here"	
15a	Entering an existing username in the username box after clicking the "Load Profile" button using *normal data* will load that profile without errors.	**Normal Data:** Entering "Josh" into the username box. (This profile already exists)	The profile of Josh should be loaded correctly, and his balance should be displayed to the right of the form.	Josh's profile is loaded correctly, and his name and balance are displayed to the right.	Yes
15b	Entering an existing username in the username box after clicking the "Load Profile" button using *erroneous data* will reject this username.	**Erroneous Data:** Enter a username which does not exist into the username box. "Holly". (This username does not exist)	The user should be presented with an error stating it could not find the profile "Holly".	The player is presented with a message box stating it could not find the profile.	Yes
15c	If you select the return to main menu button, you will be asked if you want to return to the MainMenu form.	**Normal Data:** Clicking the "Return to main menu" button.	An input box with two choices "Yes" and "No" should be opened.	The input box is loaded correctly and asks me if I am sure I want to return to the main menu.	Yes
16	If you select "Yes" after clicking the "Return to main menu" button, you are taken to the MainMenu form and the AddPlayer form is closed.	**Normal Data:** Clicking the "Yes" option inside the choice box.	The AddPlayer form should be hidden and the MainMenu form is displayed to the user.	The AddPlayer form is hidden correctly and the MainMenu form is loaded correctly.	Yes
17	If you select "No" after clicking the return to main menu button, you are not taken to the MainMenu form and you remain on the AddPlayer form.	**Normal Data:** Clicking the "No" option inside the choice box.	The AddPlayer form should remain as it was before you clicked the return to main menu button.	The MainMenu form is not loaded and the AddPlayer form remains visible.	Yes
18	Clicking the "Go" button without loading a player should not return to the MainMenu form and present the user with an error.	**Normal Data:** Click the "Go" button without creating or loading a profile.	The user should be presented with an error message box stating they need to create a profile or load one before adding a player.	The user is presented with a message box which states "ERROR! Load a profile or create one first!" and is	Yes

				then kept on the AddPlayer form.	
19	Clicking the "Go" button with a player loaded correctly should load the MainMenu form and close the AddPlayer form.	**Normal Data:** Clicking the "Go" button with a loaded player. Player name "Josh".	Josh should be displayed in the lobby section of the MainMenu form without error.	The AddPlayer form is closed and the MainMenu form is made visible with Josh's name and balance displayed.	Yes
Form: Leaderboard					
Test Number	Test Description	Test Data (Input)	Expected Results (Output)	Actual Results (Output)	Test Pass
20	Does the form display the top 5 players names and balances correctly?	**Normal Data:** Open the Leaderboard form with 5 players' profiles saved.	The form should show the top 5 players' names and balances correctly.	The 5 players inside the leaderboard text document were displayed correctly in balance order.	Yes
21	Clicking the "Return to main menu" button will ask the user if they want to return to the main menu.	**Normal Data:** Clicking the "Return to main menu" button.	An input box with two options should appear asking if they want to return to the main menu.	An input box appeared which asked the user 'Are you sure you want to return to the main menu?"	Yes
22a	Clicking 'Yes" after clicking the "Return to main menu" button will load the MainMenu form correctly and hide the Leaderboard form.	**Normal Data:** Clicking the "Yes" option in the input box.	The MainMenu form should be shown and the Leaderboard form is hidden.	The MainMenu form was shown correctly whilst the Leaderboard form was hidden from view.	Yes
22b	Clicking the "No" option after clicking the "Return to main menu: button will not load the MainMenu form and will keep the Leaderboard form	**Normal Data:** Clicking the "No" option in the input box.	The MainMenu form should not be loaded and the Leaderboard form should remain visible.	After clicking the button, the MainMenu form was not loaded and the Leaderboard form remained visible.	Yes

	shown.				
Form: Settings					
Test Number	Test Description	Test Data (Input)	Expected Results (Output)	Actual Results (Output)	Test Pass
23	Upon loading the Settings form, the "Hints" option will be enabled by default and the "Sounds" option will be disabled by default.	**Normal Data:** Opening the Settings form and not changing any values.	The checkbox for hints will be checked and the sounds checkbox will be disabled.	Upon loading the form, I noticed that the hints checkbox was checked whilst the sounds checkbox was not checked.	Yes
24	Clicking the "Back" button will hide the Settings form and show the MainMenu form.	**Normal Data:** Clicking the "Back" button.	The Settings form should be hidden whilst the MainMenu form is made visible again.	The Settings form was hidden from view whilst the MainMenu form was made visible again.	Yes
25	If you uncheck the "Hints" checkbox then the hints button will no longer be visible whilst in the PlayingBoard form.	**Normal Data:** Unchecking the "Hints" checkbox.	If you load a game, the PlayingBoard form will not feature the Hints button because the setting is toggled off.	The Hints button is not shown to the players as the checkbox in the Settings form is not checked.	Yes
26	If the "Hints: checkbox is checked, then the "Hints" button is shown the players' in the PlayingBoard form.	**Normal Data:** Checking the "Hints" checkbox.	The "Hints" button being checked will result in the "Hints" button in the PlayingBoard form being visible and clickable.	After starting a game with the Hints checkbox checked, the "Hints" button was made visible and clickable.	Yes
27a	If you check the "Sounds" checkbox, sounds will be played throughout gameplay in the PlayingBoard form.	**Normal Data:** Checking the "Sounds" checkbox.	Sounds will be played throughout gameplay because the 'Sounds" checkbox was toggled on.	Sounds were played correctly upon toggling the "Sounds" checkbox.	Yes

27b	If you leave the "Sounds" checkbox unchecked, sound will be disabled throughout gameplay.	**Normal Data:** Unchecking the "Sounds" checkbox.	Sounds should not be played throughout gameplay due to the Sounds" checkbox being toggled off.	Sound was not played during gameplay.	Yes
Form: PlayingBoard (Player vs Dealer)					
Test Number	Test Description	Test Data (Input)	Expected Results (Output)	Actual Results (Output)	Test Pass
28	When the PlayingBoard form loads it will fill in the correct players' names and balances and then assign the correct chips to each player.	**Normal Data:** Loading the PlayingBoard form with 4 players loaded.	All 4 players have their balances and names displayed, with the correct chip counts for each player.	The PlayingBoard form loaded correctly, displaying each players' name and balance and then displayed their current chip amount visually.	Yes
29	Once the chip counts and names have been displayed, the players will be asked how much they went to be, starting from the player to the right of the dealer and ending on the player to the left.	**Normal Data:** Loading the PlayingBoard form.	The player is asked how much they wish to bet, starting from right to left of the dealer.	An input box appeared correctly and asked the rightmost player how much they wish to bet and ended with the leftmost player.	Yes
30a	Entering a bet amount using *normal data*.	**Normal data:** 50 is entered in the input box.	The bet amount is accepted and the game moves on the next player.	The bet was accepted, and the game then asked the next player for their bet.	Yes
30b	Entering a bet amount using *boundary data*.	**Boundary data:** Entering the entire players balance into the input box.	The bet amount should be accepted, and the game moves on to the next player.	The game accepted the bet correctly, and then moved on to asking the next player how much they wish to	Yes

				bet.	
30c	Entering a bet amount using *erroneous data*.	**Erroneous data:** Entering "Ten" into the input box.	A message box should appear stating the user must enter a numerical value and then the user should be asked for their bet again.	A message box appeared stating "ERROR! You must enter a numerical value!" and the bet was not accepted. The user was then asked for their bet again.	Yes
31	Once all the players have entered their bets, their corresponding chip counts will visually update to show their new balances. (Old balance – bet)	**Normal Data:** Having a balance of ¢500 and betting ¢100.	The visual chip amount will change from 1 ¢500 chip to 4 100 chips.	The bet was accepted and then once all players had bet, the chip count was changed from 1 ¢500 chip to 4 ¢100 chips.	Yes
32	Once all the players have entered their bets, cards will be dealt one-by-one starting from the rightmost player and ending up at the leftmost player.	**Normal Data:** Entering the bets for all players.	Cards should be dealt one at a time to each player, starting from the right of the dealer and ending up at the left.	The cards are dealt correctly starting with the right player and ending with the leftmost player, one-by-one.	Yes
33	Once all the cards have been dealt, the rightmost player will be highlighted indicating that it is their turn.	**Normal Data:** Entering bets for all players.	The first player (to the right of the dealer) will have their name highlighted. Indicating it Is their turn.	Upon clicking the "Add Player" button, the player is taken to the AddPlayer form whilst the MainMenu form is hidden.	Yes
34	Once all the cards have been dealt, if the "Hints" setting, is enabled, the player Hand Total labels will be updated to the left of the screen. These will display the total number of each	**Normal Data:** Dealing cards to all players, testing player "Josh" who has two 6's.	Josh's hand total label should display the number 12.	Once all the cards were dealt, Josh's hand total label displayed the number 12 correctly.	Yes

	players' hand.				
35	4 Buttons appear in the centre of the table after the player is selected for their turn. These buttons are "Hit" "Stick" "Double" (If they can double) and "Split", if they have a split available.	**Normal Data:** Placing bets for all players.	The 4 buttons representing the choices the players can choose should appear in the centre of the board.	The buttons appeared correctly, the split button did not appear due to not having a "split-able" hand.	Yes
36a	If the player has a "split-able" hand, they will be asked if they want to split their hand.	**Normal Data:** Having a hand which can be split into two new hands.	A choice box with two choices (Yes and no) should be presented to the user.	The box was displayed correctly.	Yes
36b	If the user selects "Yes" on the split option, their single hand will visual separate into two new hands made up of their existing cards.	**Normal Data:** Clicking the "Yes" button inside the split message box.	The users hand should split into two hands (Their second card moves bellow their first card).	The hand split into two correctly and the second card moved bellow their first.	Yes
36c	If the user hits a card to a split hand, it will deal to the correct hand, whether that is their first hand or their second hand.	**Normal Data:** Hitting a card with a split hand, to the first hand.	The user should be dealt a card to their first (higher up) hand, and it should be offset correctly.	The card was dealt to the first user hand and offset the correct amount.	Yes
37	If the user clicks the "Hit" button, they are dealt another card.	**Normal Data:** Clicking the "Hit" button.	The next card in the deck should be dealt to the player slightly offset so they can see their cards.	The next card in the list was dealt to the player correctly.	Yes
38	One a player has been dealt a card, their corresponding Hand Total label is updated with their new hand total.	**Normal Data:** Clicking the "Hit" button as Josh who has 2 6's and is dealt a 10.	Josh's hand total label should be updated to display his new hand total which is 16.	Josh's Hand total label was updated correctly to display 16.	Yes
39	If a player goes bust after hitting, their hand total label is updated to display "BUST" in a red	**Normal Data:** Hitting until a player goes bust.	The hand total label should display "BUST" with a red background and	The hand total label updated correctly and a message box stating	Yes

	background, and they are presented with a message box telling them they are bust.		a message box should appear which states that the player is bust.	"Unlucky! Josh, you are now bust!"	
40	If the user goes bust, the turn system moves on to the next player?	**Normal Data:** Hitting till bust as player 1.	The player to the left of the previous player should be highlighted and presented with the correct choice buttons.	The player to the left of the previous player was highlighted and allowed to take their turn.	Yes
41	If a player sticks, will the turn system move onto the next player?	**Normal Data:** Sticking as player 1.	Once player 1 sticks, the turn system should move on to the next available player.	When player 1 stuck, the turn system then moved onto player 2.	Yes
42	If the user selects the "Double" button, is their bet doubled and are they dealt one final card?	**Normal Data:** Clicking the "Double" button.	The player should be dealt another card and have their bet doubled in size.	The player was dealt one final card, and their bet was doubled. The game then moved onto the next players turn.	Yes
43a	If you try to click the "Main Menu" button, you will not be able to back out until everyone has finished their turn.	**Normal Data:** Clicking the "Main Menu" button mid-game.	A message box should appear saying you can't return to the main menu until everyone has finished their turn.	The MainMenu form was not loaded, and a message box appeared stating "You cannot return to the main menu until everyone has finished their turn".	Yes
43b	If you click the "Main Menu" button when the round is over, you will be redirected to the MainMenu form and the PlayingBoard form will be hidden.	**Normal Data:** Clicking the "Main Menu" button when the current round is finished.	The MainMenu form should be displayed and the PlayingBoard form hidden from view.	The MainMenu form was displayed correctly whilst the PlayingBoard form was closed.	Yes

44	Clicking the "Hints" button will display a hint unique to the player clicking, and will look at the players cards, bet, and the dealer's cards to give a hint.	**Normal Data:** Clicking the "Hint" button.	A hint is displayed to the user based on their cards and bet along with the dealer's cards.	The player was presented with a message box which contained a hint unique to them.	Yes
45	Once all players have had their turn, the dealer's second card is revealed.	**Normal Data:** Finishing all turns for all players.	The dealer's card should be turned over revealing what card the dealer had.	The dealer's card was revealed correctly and turned from the back of a card to the front.	Yes
46	Once the dealer' card has been displayed, the dealer's Hand Total label is updated with the new Hand Total.	**Normal Data:** Turning over the dealer's second card. Example: the dealer's first card was a 6 and his second was a 10.	The dealer's hand value should be updated from 6, to 16 once his second card is shown.	The dealer's Hand Total label was updated as it should be from 6 to 16.	Yes
47a	The dealer will hit until their hand total is greater than or equal to 17, or they go bust. Using *normal data.*	**Normal Data:** Forcing the dealer to be dealt a hand total of 20 after 3 cards (6,4,10).	The dealer should stick after their third card is dealt.	The dealer stuck after his third card was dealt because his hand value was greater than 17.	Yes
47b	The dealer will hit until their hand total is greater than or equal to 17, or they go bust. Using *boundary data.*	**Boundary Data:** Forcing the dealer to be dealt a hand total of 17 after 3 cards (6,5,6).	The dealer should stick because their hand is 17 after their third card is dealt.	The dealer decided to stick because his third card brought his hand total to 17.	Yes
47c	The dealer will hit until their hand total is greater than or equal to 17, or they go bust. Using *erroneous data.*	**Erroneous Data:** The dealer is dealt a hand value of 20 with his first 2 cards. (10, Queen).	The dealer should stick without being dealt a third card.	The dealer decided to stick because his hand value was greater than 17 with his first 2 cards.	Yes
48a	Once the dealer has finished their turn, winnings are calculated and displayed to each	**Normal Data:** Complete all turns from all players, with a winning hand from Josh	Bet winnings are presented to Josh with a multiplier of 1.5 because the	A message box appeared stating "Josh you have won ¢75".	Yes

	player in the form of a message box if they won. (Dealer goes bust scenario).	who beat the dealer because the dealer went bust.	dealer went bust. Josh bet ¢50 so his winnings should be ¢75.		
48b	Once the dealer has finished their turn, winnings are calculated and displayed to each player in the form of a message box if they won. (Player has a blackjack scenario).	**Normal Data:** Complete all turns from all players, with a winning hand from Josh who beat the dealer because he has a blackjack.	Bet winnings are presented to Josh with a multiplier of 2 because he got a blackjack. Josh bet ¢50, so his winnings should be ¢100.	A message box appeared stating "Josh you have won ¢100".	Yes
48c	Once the dealer has finished their turn, winnings are calculated and displayed to each player in the form of a message box if they won. (Player and dealer have the same hand value scenario).	**Normal Data:** Complete all turns from all players, with a tie game because Josh and the dealer have the same hand value.	Bet winnings are presented to Josh with a multiplier of 1 because Josh and the dealer have the same hand value. Josh bet ¢50 so his return should be ¢50.	A message box appeared stating "Josh, it's a tie! You have won back your bet of ¢50".	Yes
49a	Once the game is finished, does a "New Round" button appear in the centre of the table?	**Normal Data:** Finishing a round.	A button appears in the centre of the table titled "New Round".	A "New Round" button appeared as planned in the centre of the table.	Yes
49b	Once the game is finished, each player's gamesave file is updated with their new balance.	**Normal Data:** Finishing a round.	The gamesave file for the player should be updated with their new balance.	The gamesave for all the players was updated correctly with their new balances.	Yes
49c	Once the game is finished, the leaderboard.txt document players are loaded, compared with the current players, and any balances are updated or added. Then, the list is quicksorted to find the new top 5	**Normal Data:** Finishing a round and having players who are not on the leaderboard and some who are.	The leaderboard.txt document should be updated with any new players and overwrite any existing players with new balances.	The leaderboard.txt document was updated accordingly, and the new top 5 players and balances were saved for future use.	Yes

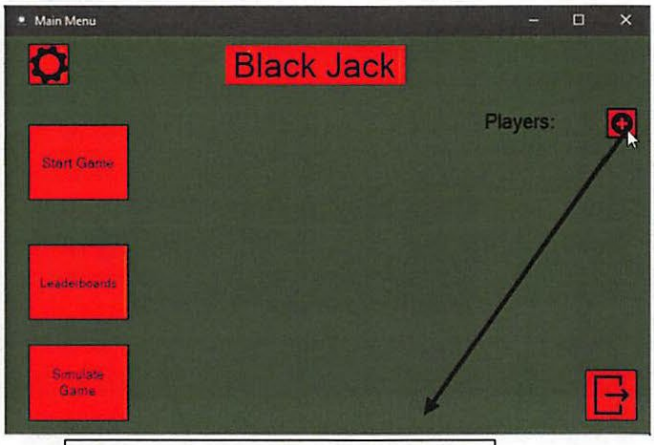
	players, and then re-saved to the leaderboards.txt document.				
50	If you click the "New Round" button, the table is reset (Cards are returned to the deck) and the turn system restarts itself beginning with asking for bets.	**Normal Data:** Clicking the "New Round" button.	All cards are returned to the deck on the table and players are then asked for their bets.	All cards that had been dealt in the previous round were returned the deck in the centre of the screen and players were asked for their bets.	Yes
51	Clicking the "Quit" button will open a confirmation box with two options, "Yes", and "No".	**Normal Data:** Clicking the "Quit" button.	A confirmation box should appear which contains only two buttons, one for "Yes" and one for "No".	A confirmation box was loaded correctly which displayed the expected buttons.	Yes
52a	If you click the "Yes" option inside the confirmation box, the PlayingBoard form is closed and the program is terminated.	**Normal Data:** Clicking the "Yes" option inside the confirmation box.	The PlayingBoard form should be closed thus terminating the program.	The PlayingBoard for was closed as expected and the program was terminated.	Yes
52b	If you click the "No" option inside the confirmation box, the PlayingBoard form is not closed and the program remains active.	**Normal Data:** Clicking the "No" option inside the confirmation box.	The PlayingBoard form should remain visible and the confirmation box should disappear.	The PlayingBoard form was not closed and the program remained active.	Yes
Form: PlayingBoard (AI vs Dealer)					
Test Number	Test Description	Test Data (Input)	Expected Results (Output)	Actual Results (Output)	Test Pass
52	Upon loading the PlayingBoard form for Simulation, each simulated player decides their bet.	**Normal Data:** Opening the PlayingBoard form after selecting a simulated game.	Simulated player bets are calculated and displayed in the form of message boxes.	Upon loaded, each simulated player displayed their bet in the form of a message box.	Yes

53	Once each bet has been displayed, the bet amount is deducted from the simulated player's balance and chip counts are updated accordingly.	**Normal Data:** Placing a bet	Simulated players balances are updated after their bets and their chip counts are updated showing their new balances in chip form.	The balance labels were updated correctly, and each simulated player's chip amount were recalculated for their new balance.	Yes
54	Once each player has decided their bet, they decide on their move, this move is then displayed to the user with justification as to why they chose to do so.	**Normal Data:** Simulated player Josh has a 2 and 6.	A message box should appear showing the simulated player's chosen turn, and why they chose it.	A message box appeared stating that "Josh has decided to hit because his hand value is only 8".	Yes
55	If the simulated player decided to hit, a new card is dealt to them and they will then repeat *Test 54*.	**Normal Data:** The simulated player decided to hit.	A new should be dealt to the simulated player.	A new card was dealt to the simulated player.	Yes
56	If the simulated player decides to stick, the game will then move onto the next simulated player if there is one available.	**Normal Data:** The simulated player decided to stick.	The turn system will then move onto the next simulated player.	The turn system moved onto the next player as expected.	Yes
57	One all simulated players have finished their turn, the dealer will have their turn and reveal their second card Then displaying why they decided to hit/stick similarly to the simulated players.	**Normal Data:** All simulated players finish their turn.	The dealer's second card should be revealed, and the dealer will have their turn, whilst explaining what they decided to do and display this, similarly to the simulated players.	The second card was revealed correctly, and the dealer made their turn whilst displaying the reasoning.	Yes
58	Once all simulated players and the dealer have had their turn, the winnings are then calculated and displayed in the	**Normal Data:** Simulated players and the dealer finish their turns.	Player winnings should be calculated, and their winnings should be displayed	Player winnings are displayed correctly with justification as to why they	Yes

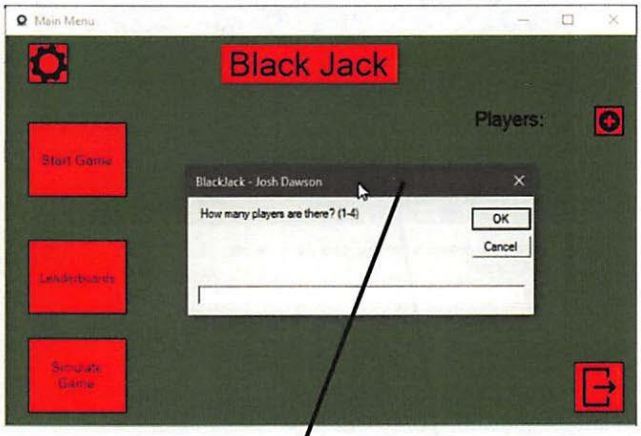
	same was as in *Test 48a,48b,48c.*		correctly.	won what they did.	
59	A "New Round" button will appear inside the centre of the table which if clicked will reset the board similarly to inside Test 50.	**Normal Data:** Clicking the "New Round" button.	The table should be reset, and all cards are returned to their position in the deck, ready for the next round.	The cards were reset correctly, and the game restarted to *Test 52.*	Yes
60	Clicking the "MainMenu" button will return you to the MainMenu form.	**Normal Data:** Clicking the "MainMenu" button.	The PlayingBoard form should be closed and the MainMenu form should be displayed.	The PlayingBoard form was closed as expected and the MainMenu form was made visable.	Yes
61	Clicking the "Quit" button will open a confirmation box with two options, "Yes", and "No".	**Normal Data:** Clicking the "Quit" button.	A confirmation box should appear which contains only two buttons, one for "Yes" and one for "No".	A confirmation box was loaded correctly which displayed the expected buttons.	Yes
62a	If you click the "Yes" option inside the confirmation box, the PlayingBoard form is closed and the program is terminated.	**Normal Data:** Clicking the "Yes" option inside the confirmation box.	The PlayingBoard form should be closed thus terminating the program.	The PlayingBoard for was closed as expected and the program was terminated.	Yes
62b	If you click the "No" option inside the confirmation box, the PlayingBoard form is not closed and the program remains active.	**Normal Data:** Clicking the "No" option inside the confirmation box.	The PlayingBoard form should remain visible and the confirmation box should disappear.	The PlayingBoard form was not closed and the program remained active.	Yes

Evidence

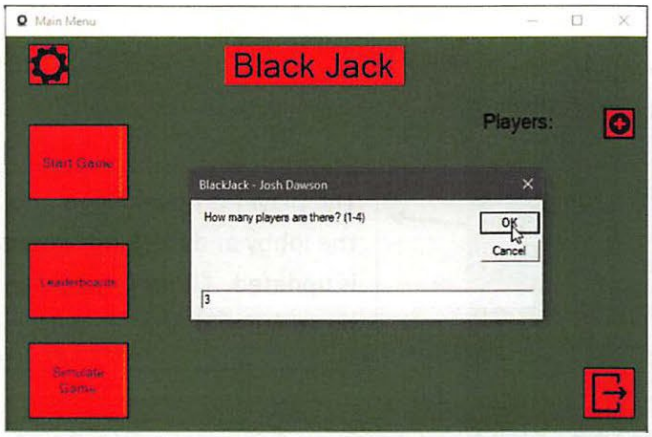
Evidence 1 – Test 2

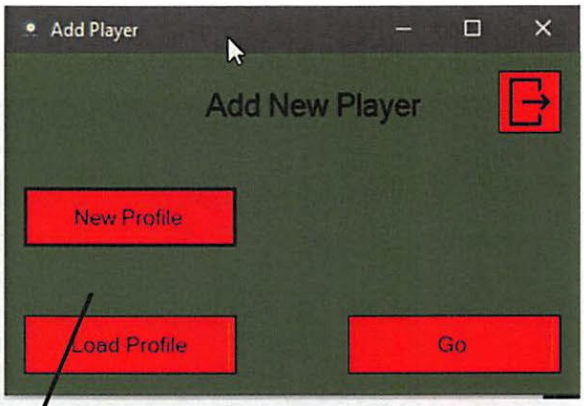


Clicking the "New Game" button.

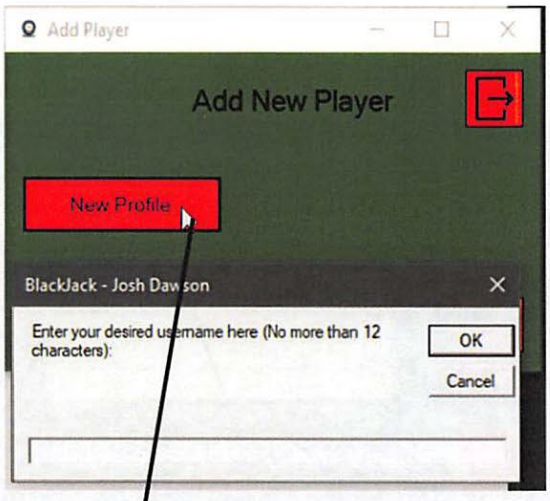


A message box appears which will ask you how many players you want to add.



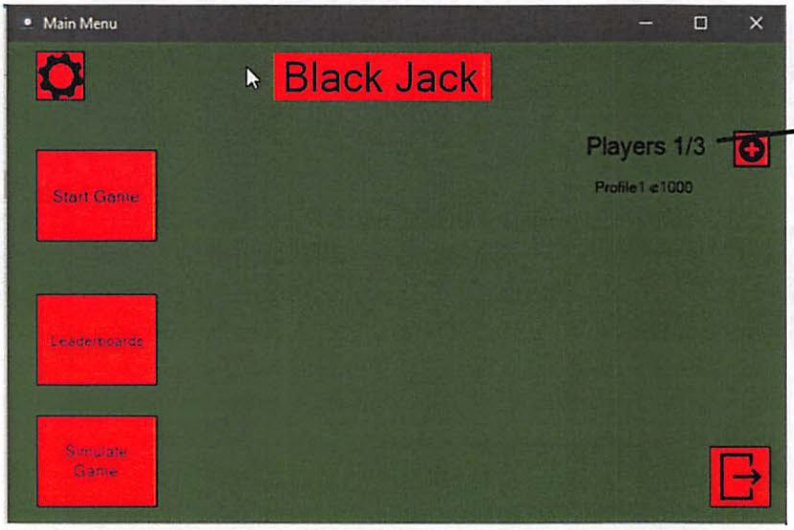


The AddPlayer form is displayed which allows you to create a new profile or load one.



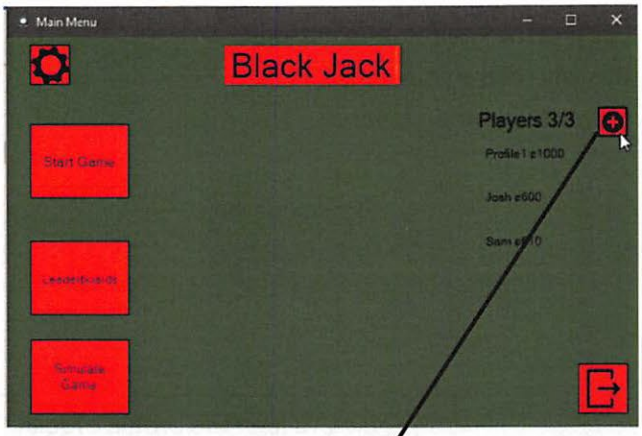
Clicking the New Profile button will ask you to choose a name.

The name entered is loaded with a balance of ¢1000.

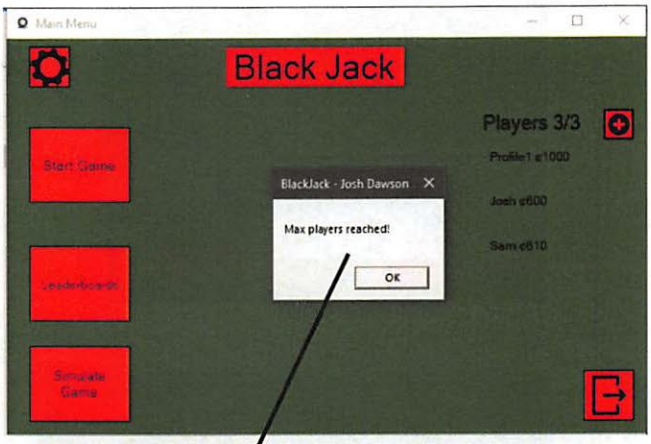


The player is now loaded into the lobby and the lobby counter is updated. Other players can be added with the "+" button.

Evidence 2 – Test 5

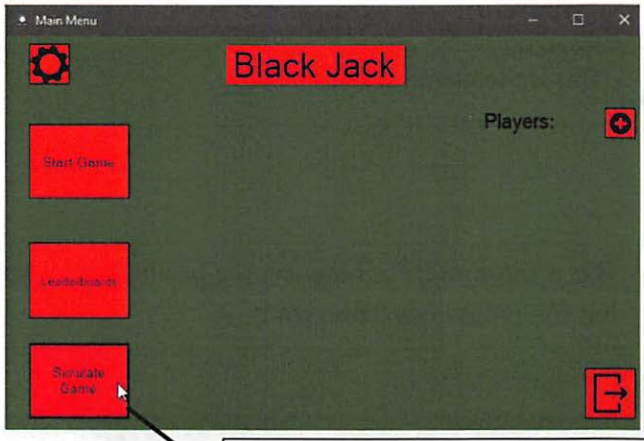


Clicking the "Add Player" button despite all 3 players being loaded.

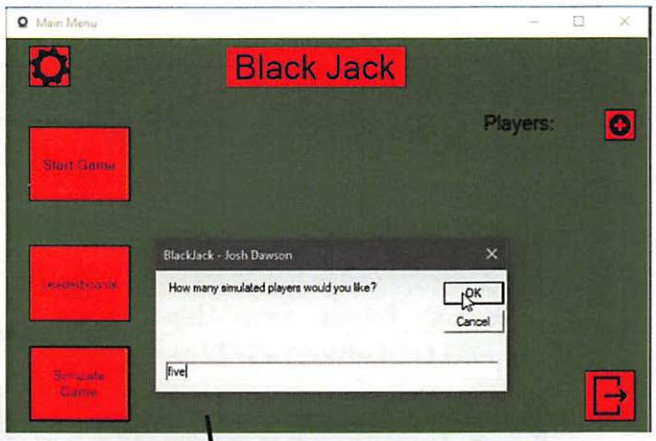


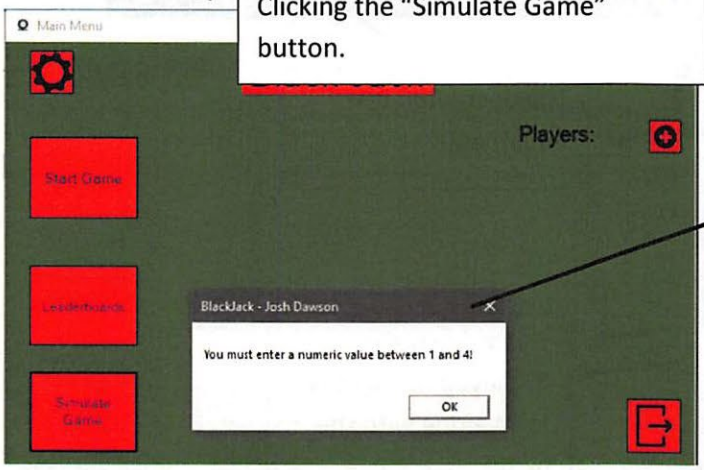
A message box appears with an appropriate error.

Evidence 3 – Test 10b



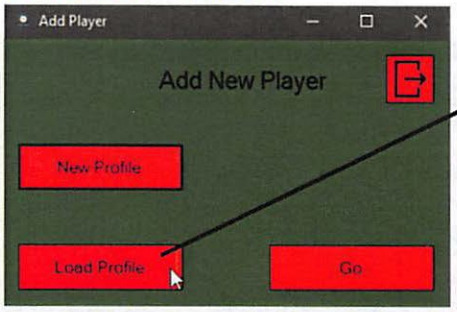
Clicking the "Simulate Game" button.





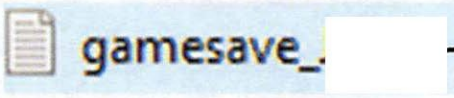
The value is not accepted, and an appropriate error message is presented.

Evidence 4 – Test 15a



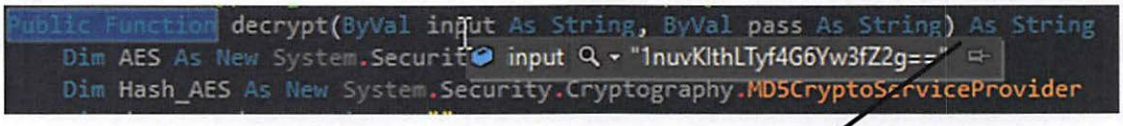
Clicking the Load Profile button, this will read a gamesave file for a player to get their balance.

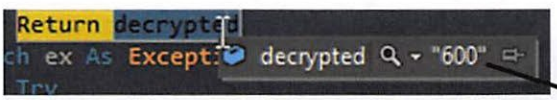
I'm going to load the profile of .



This is the gamesave file for .

This is the contents of the file; the balance is encrypted. But the program will decrypt it



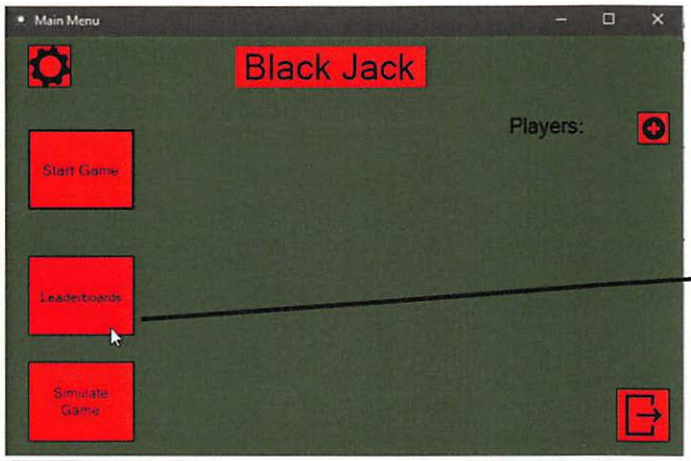


The decrypted text equals 600.

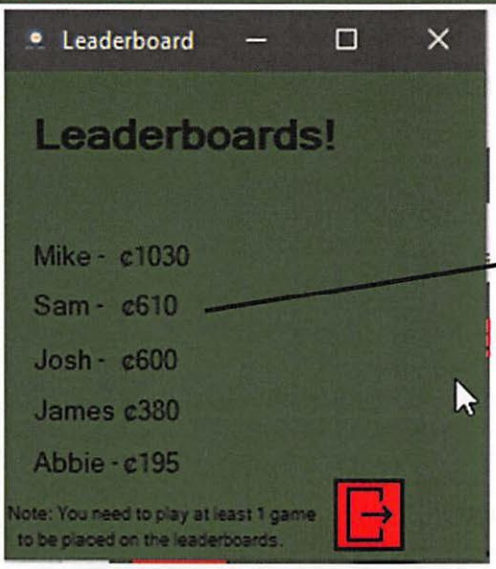
This value is then displayed in plaintext form along with the name.

Evidence 5 – Test 20

This is the contents of the leaderboards.txt document. The names are already in order because they are sorted before being saved. (Evidence 23 shows this process).

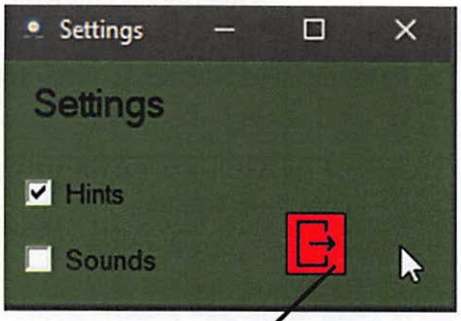


Clicking the leaderboard button shoud load the contents of the leaderboard.txt document and display the names and balances accordingly.

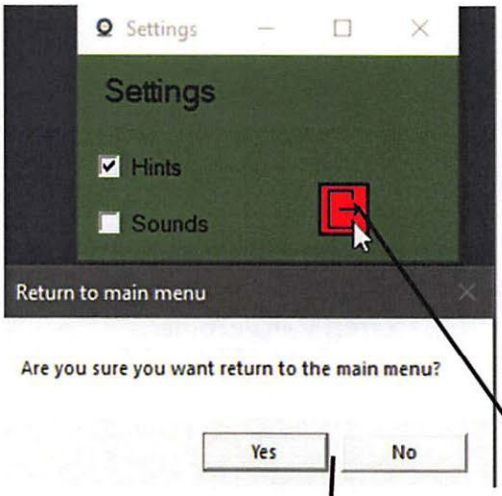


Here you can see all 5 names and balances were displayed correctly and in order.

Evidence 6 – Test 24



Inside the settings menu.

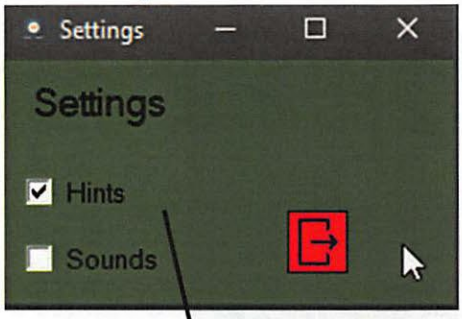


Clicked the "Return" button.

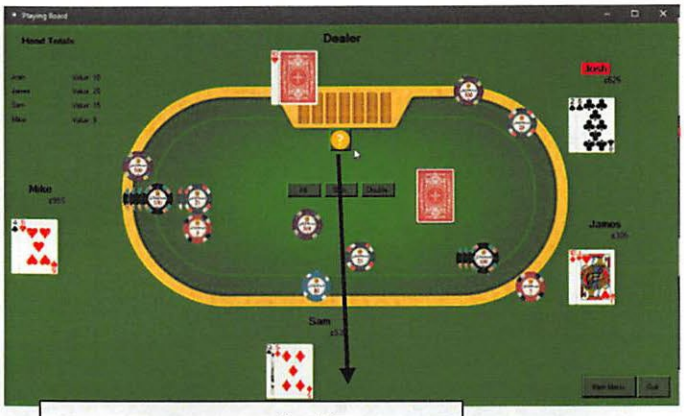
Message box appears asking if I am sure.

Clicking yes returns me to the main menu.

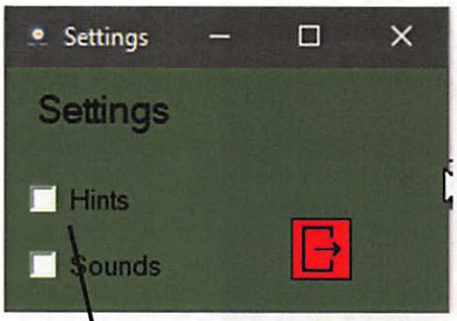
Evidence 7/8 – Test 25 and 26



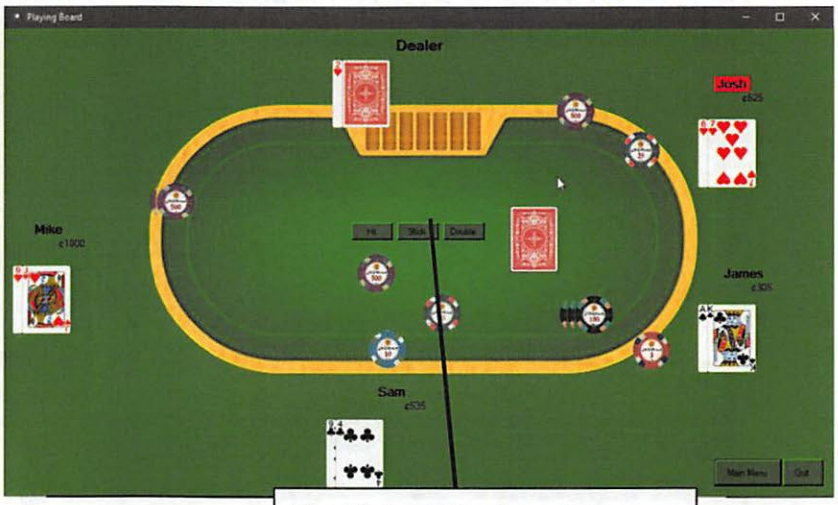
The hints checkbox is checked, so the hints button should appear on the playing board.



Here you can see the hints button is visible.

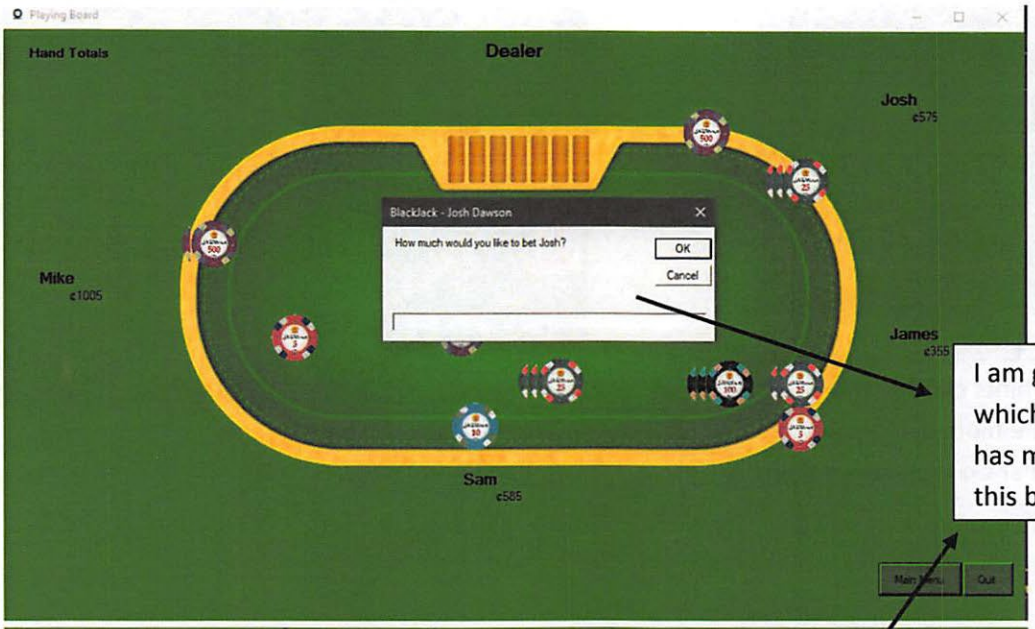


Now that the hints checkbox is unchecked, the hints button shouldn't appear on the playing board.

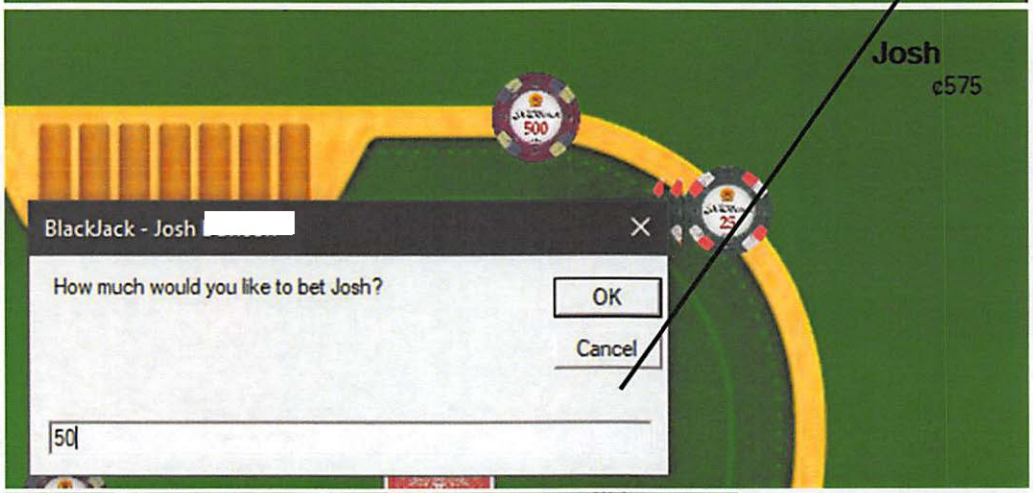


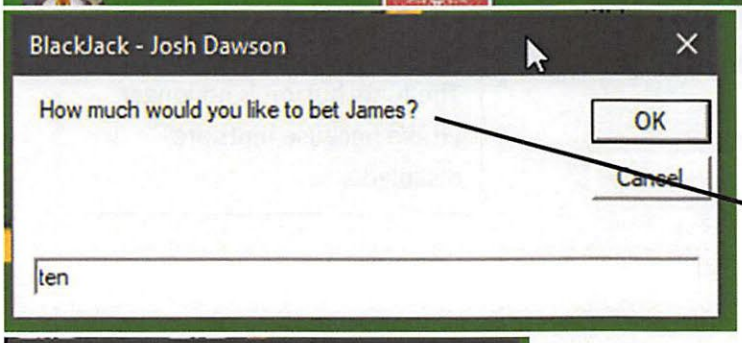
The hints button is no longer visible because hints are disabled.

Evidence 9/10/11 – Test 30a/b/c

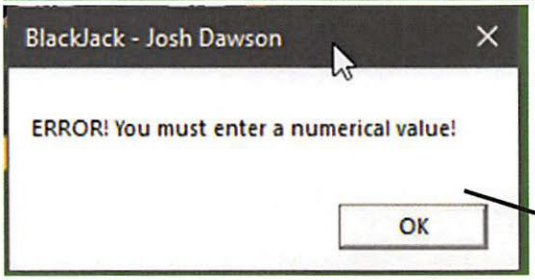


I am going to bet typical data, which will be 50. Because Josh has more than 50 balance, this bet should be accepted.

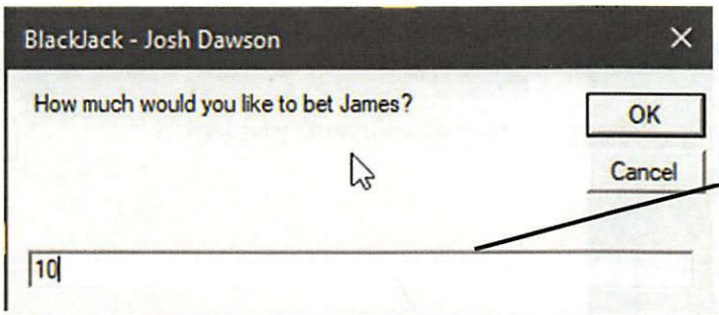




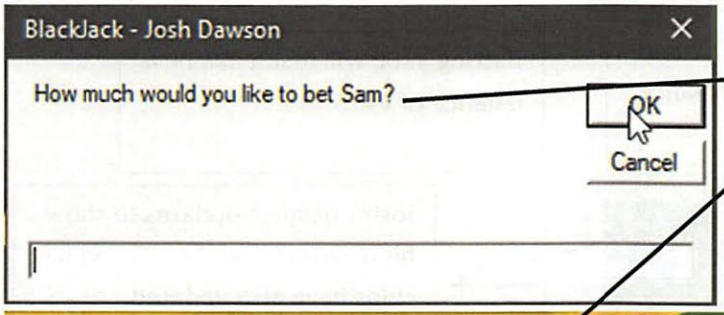
The game moved onto the next player, indicating Josh's bet was valid. I am going to enter the string "ten" into the box, this shouldn't be accepted because it is not an integer.



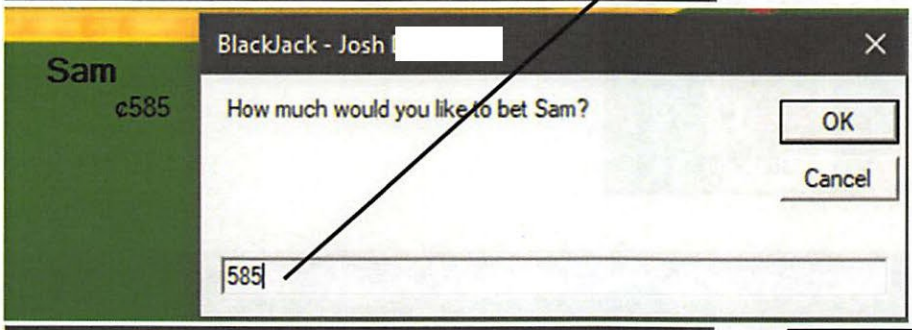
Here you can see that the input of "ten" was not accepted and an error was flagged.

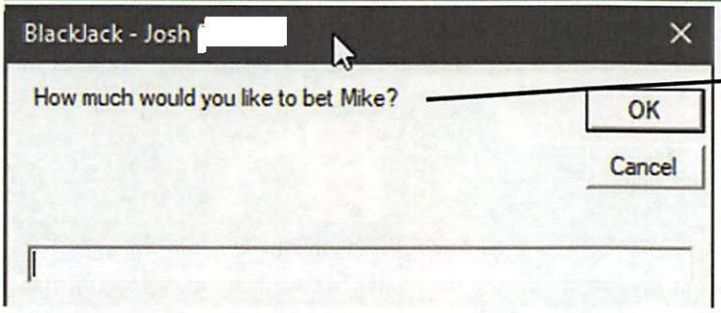


It is still James' turn because his bet was invalid, so this time I will enter a valid bet of 10, the game should then move onto Sam as he is the next player.



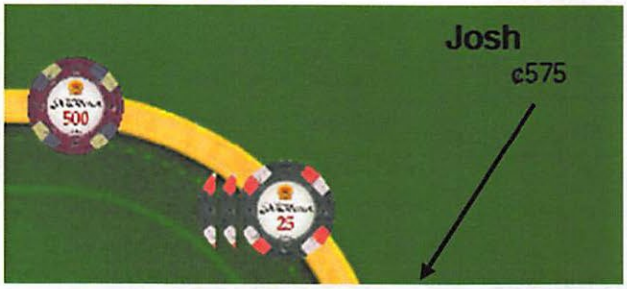
The game moved onto Sam, and now I will test boundary data. I will enter Sam's full balance of 585 as a bet. This should be accepted.



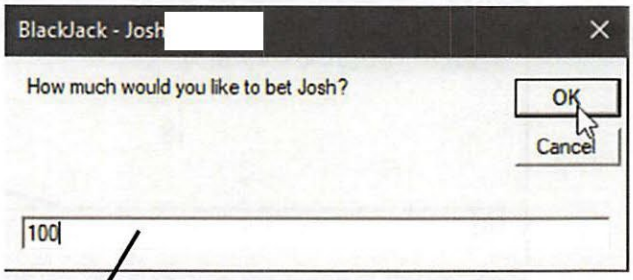


The game has moved onto the final player, Mike. This also indicates that Sam's bet of 585 was accepted.

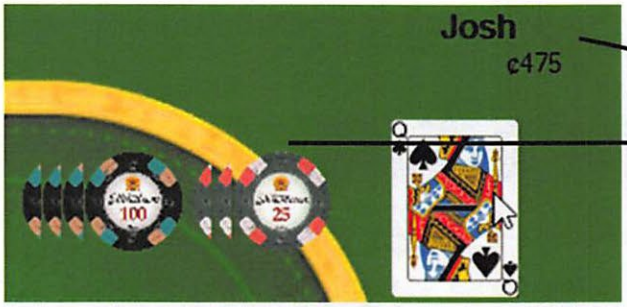
Evidence 12 – Test 31



Josh has a balance of ¢575, and his chips represent this balance in chip-form.

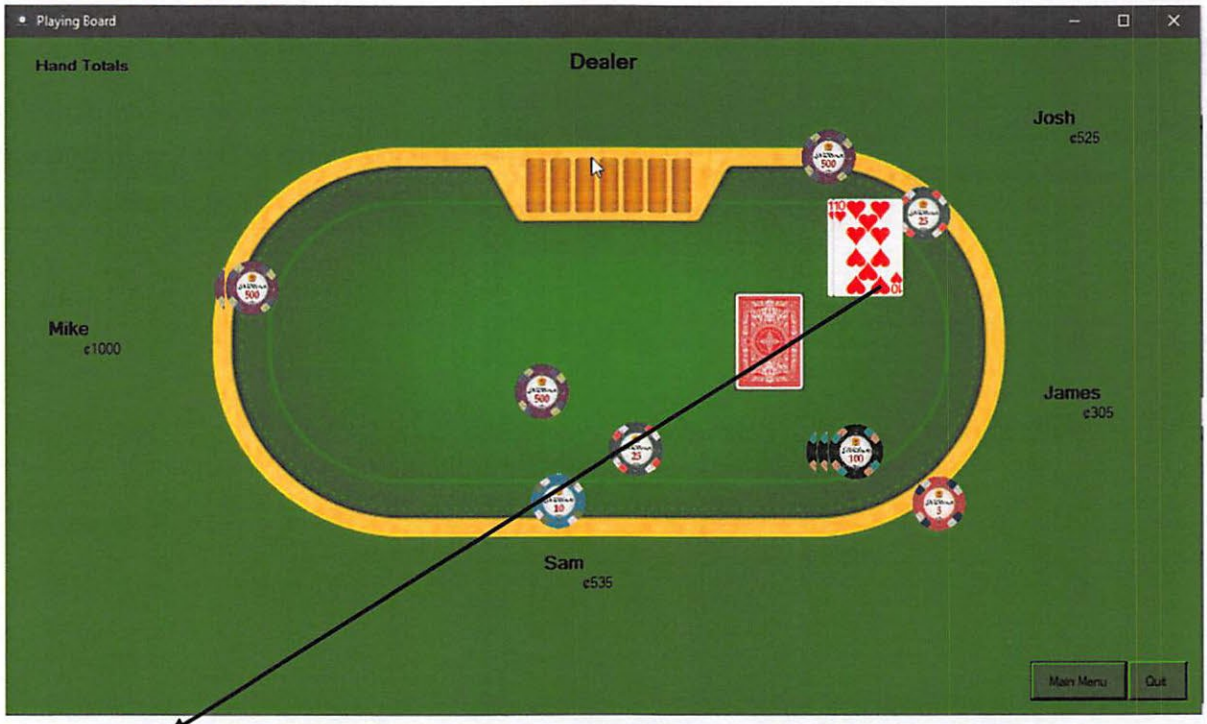


Betting ¢100 will result in a new balance of ¢475.

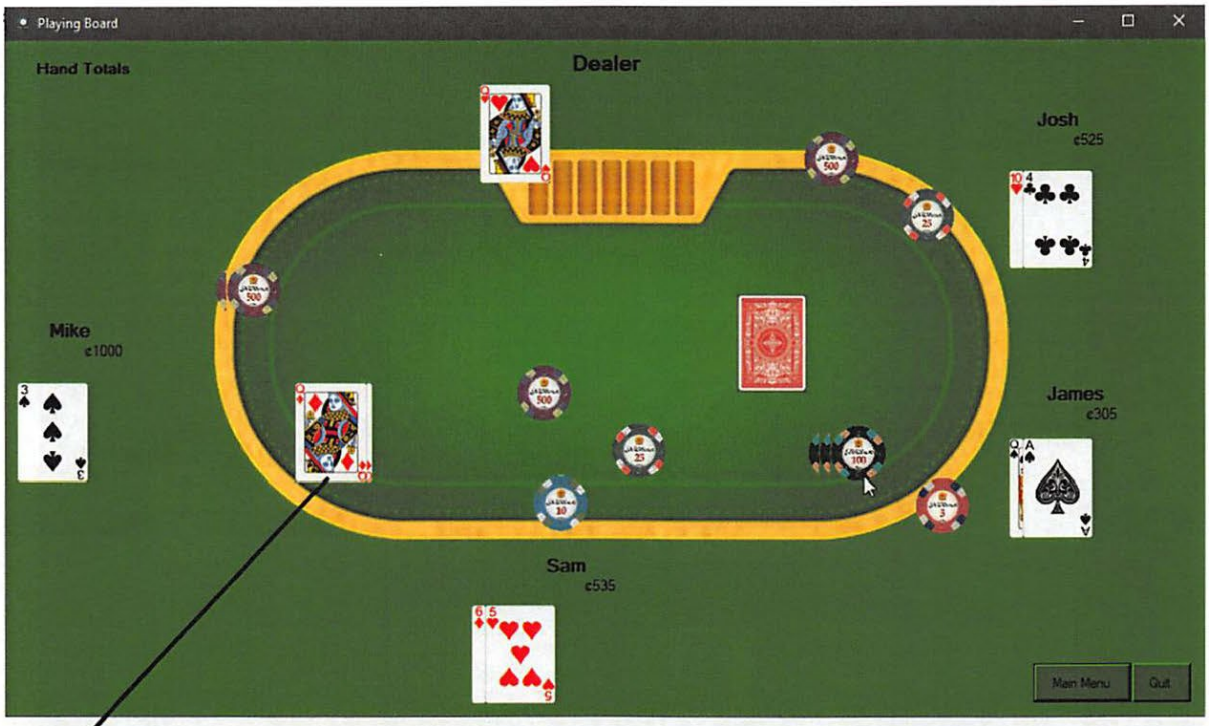


Josh's balance updates to show his new balance of ¢475 and his chips have also updated accordingly.

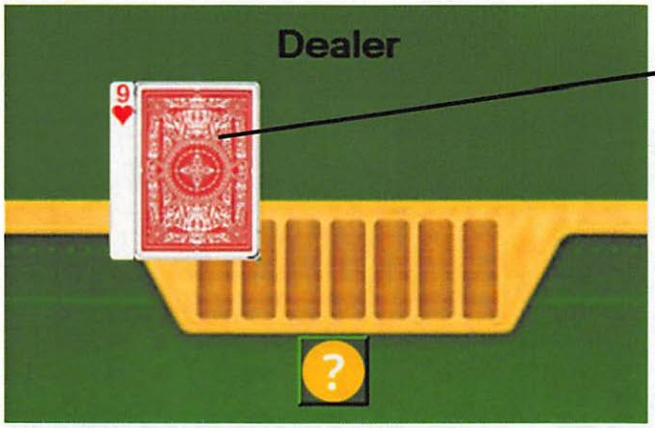
Evidence 13 – Test 32



Cards are dealt one-by-one starting from the player to the right of the dealer and ending with the player to the left.

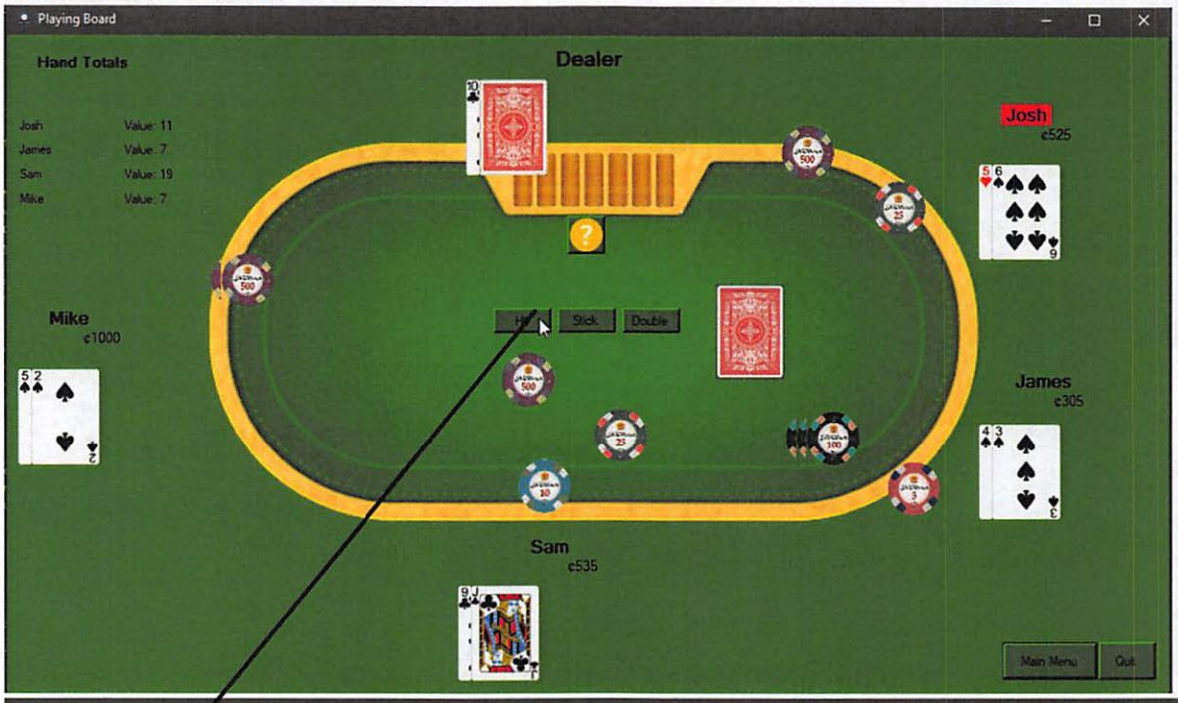


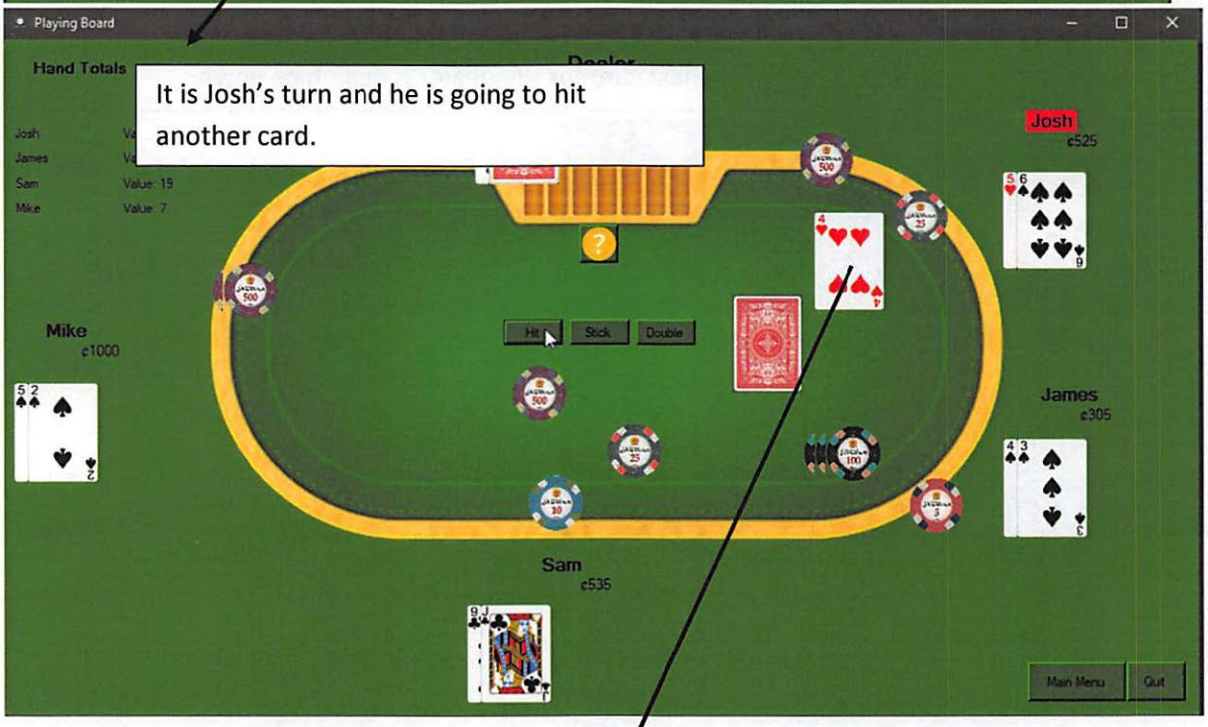
Once the final card is dealt to a player, the last card for the dealer is dealt face down.



Dealer's final card is dealt face down to hide it from the players.

Evidence 14 – Test 37

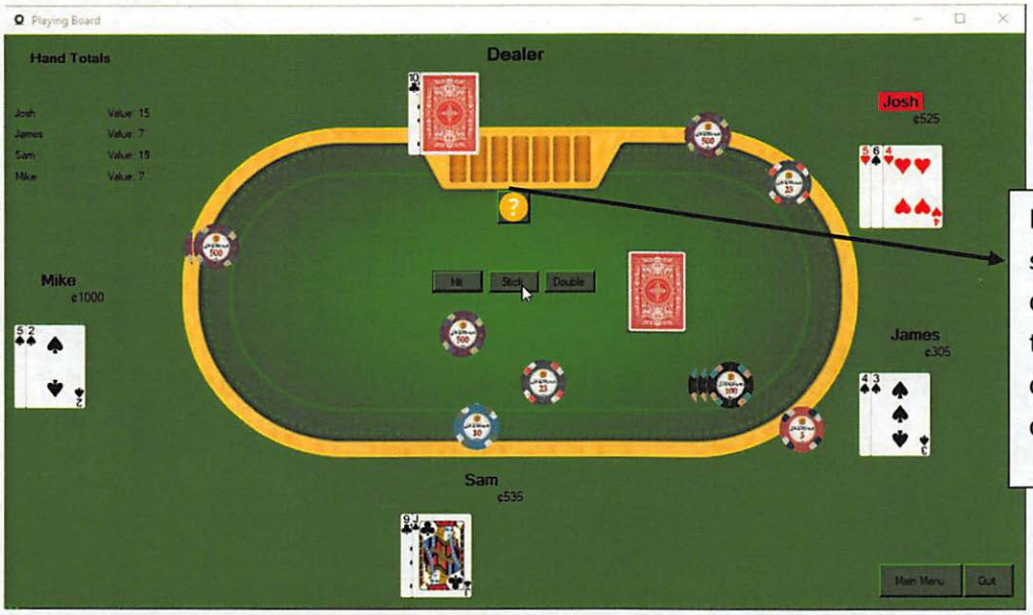




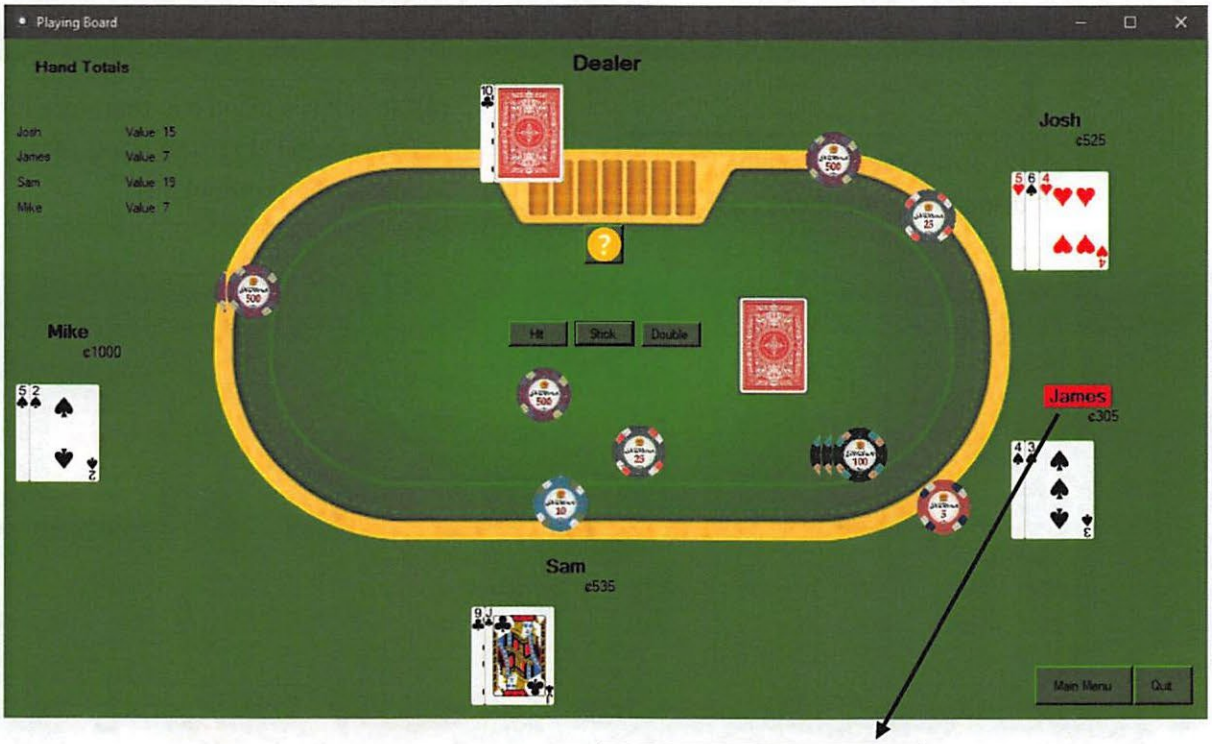
After hitting, another card is dealt to Josh's hand.

This card is automatically offset to allow Josh to still see his previous cards.

Evidence 15 – Test 41

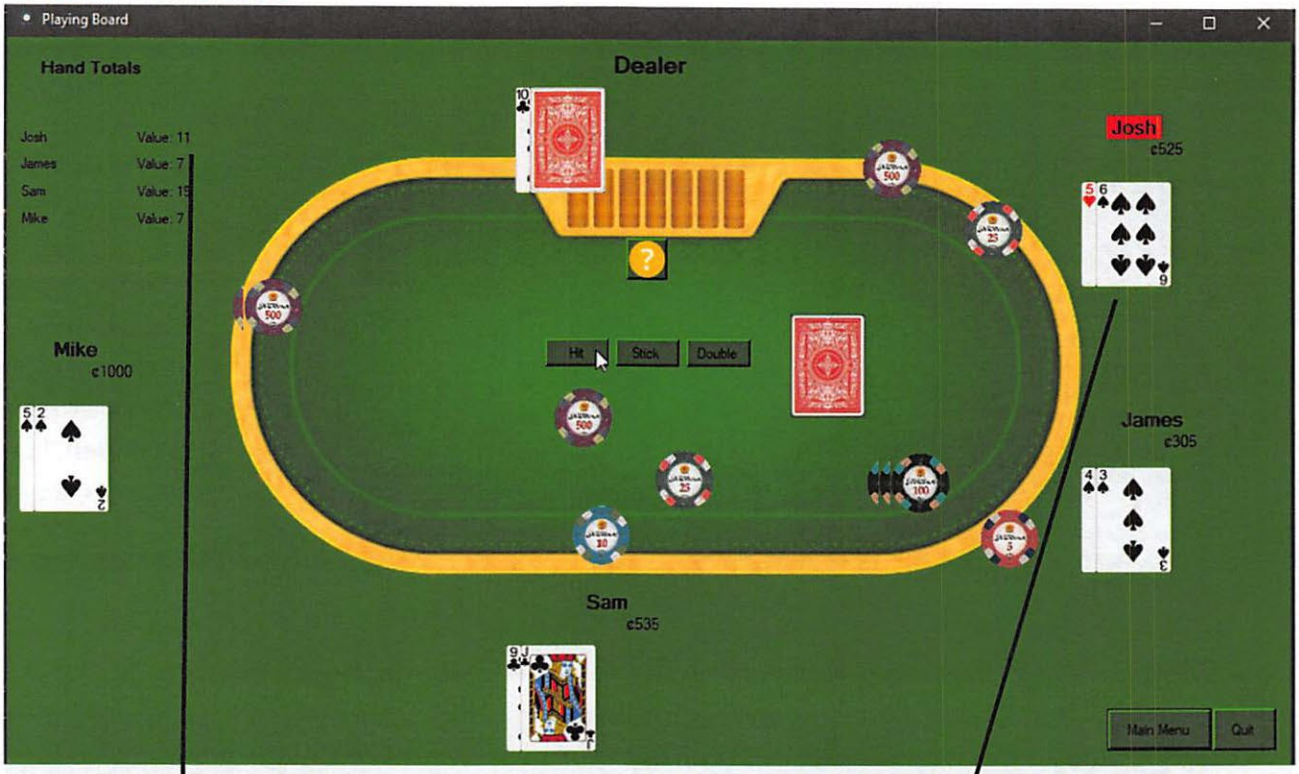


I am now going to stick, this should then cause the turn system to move to the player on my left, in this case it is James.



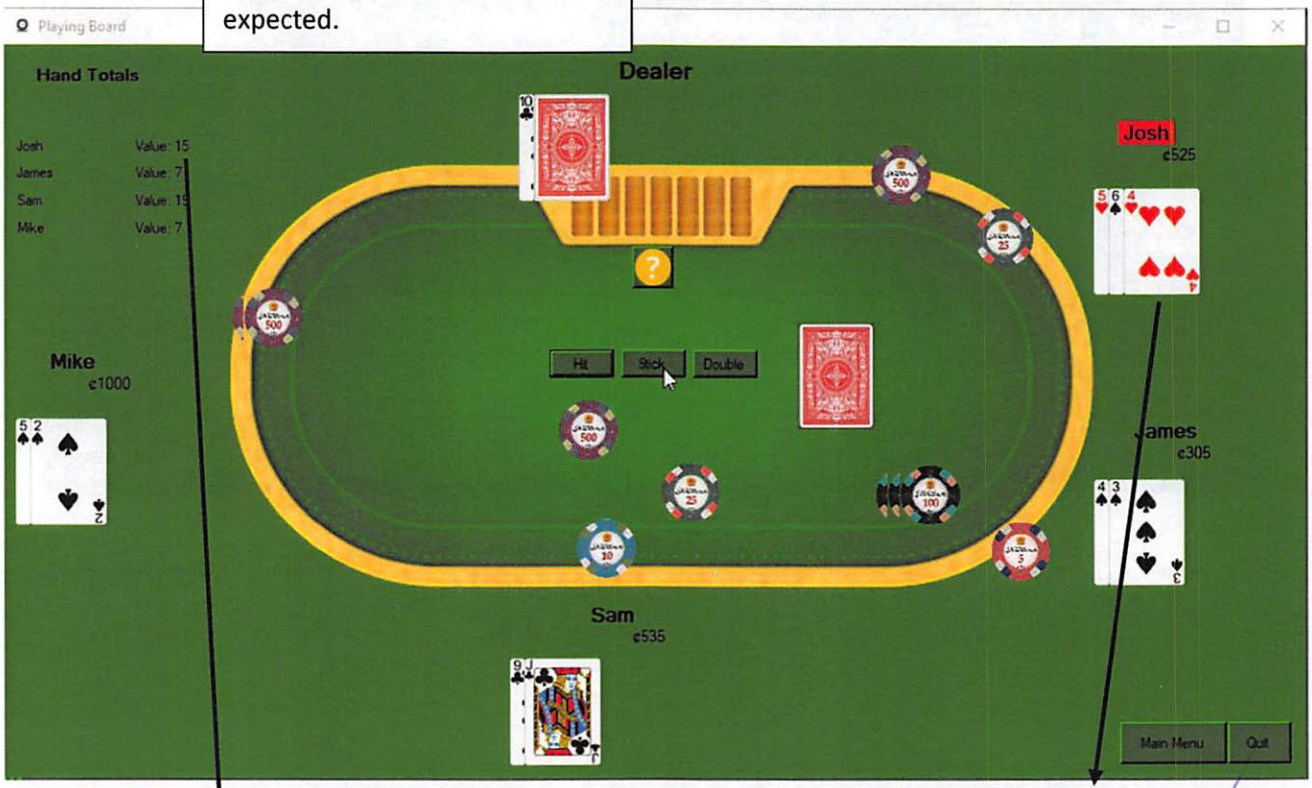
James' name is now highlighted indicating it is his turn.

Evidence 16 – Test 34



(1) Josh has a 5 and a 6, therefore his hand total is 11. The Hand Total label should show this.

(2) Josh's hand total label displays the number 11 as expected.

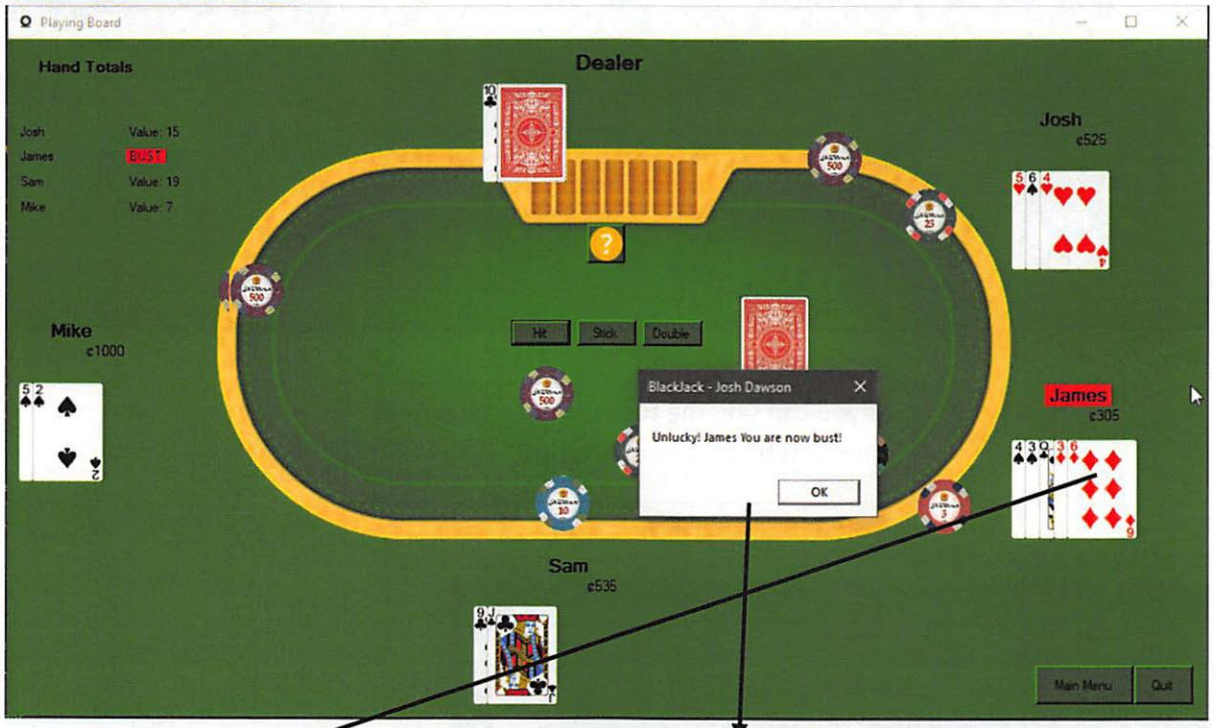


1) Josh was dealt a 4 of hearts, so his new hand total is 15.

2) As you can see, his hand total label updated to 15 accordingly.

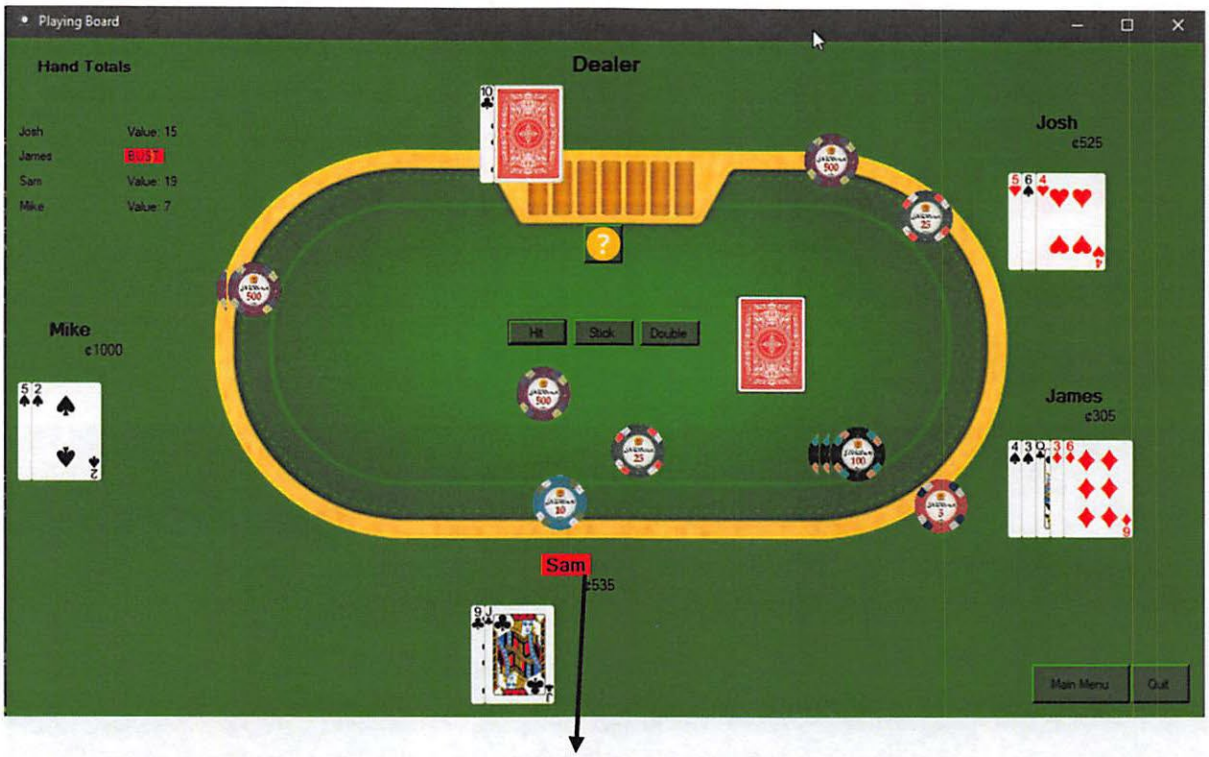
Evidence 17 – Test 39

James has a hand value of 20, therefore only an ace would keep him in the game, but I am going to hit until he is bust.



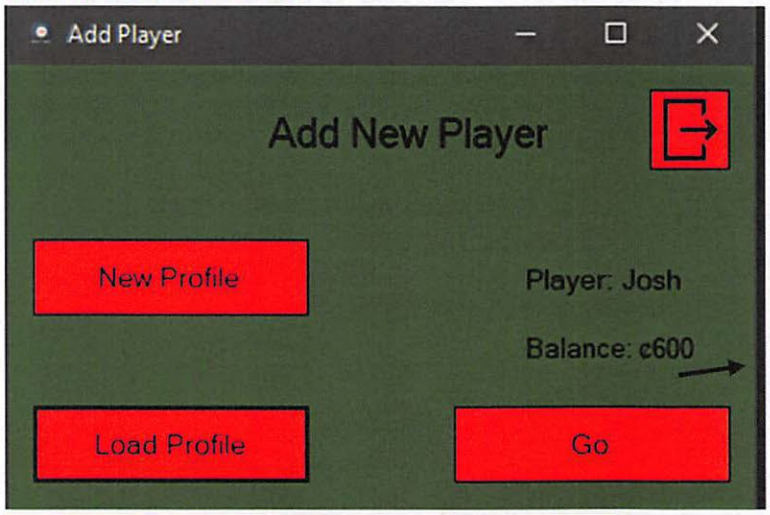
(1) James was dealt a 6 meaning his hand value was 25, therefore making him bust.

(2) A message box appeared telling James that he is now bust.



As you can see, the turn system now moved onto Sam because James is bust.

Evidence 18 – Test 49b



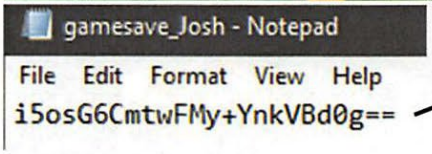
To test if balances get overwritten, I am going to load Josh, who has a balance of ¢600.

When loaded in, Josh has a balance of ¢600.

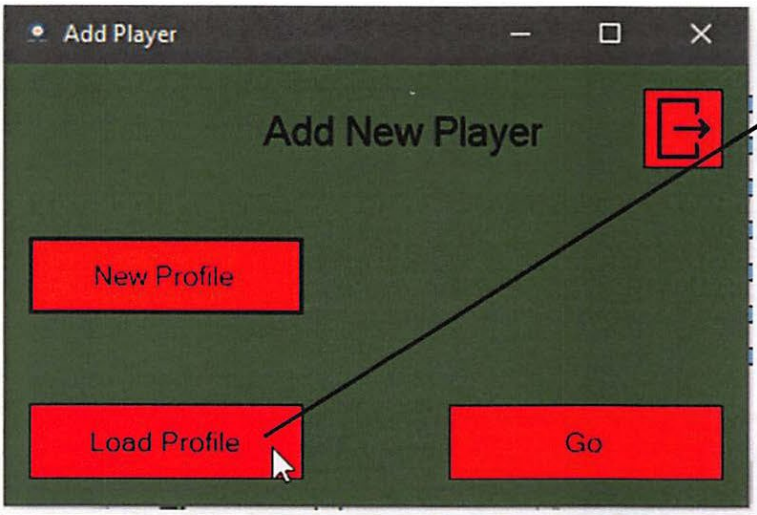
I am going to bet ¢150, but purposefully lose.

My balance has updated to display ¢450, because I bet ¢150.

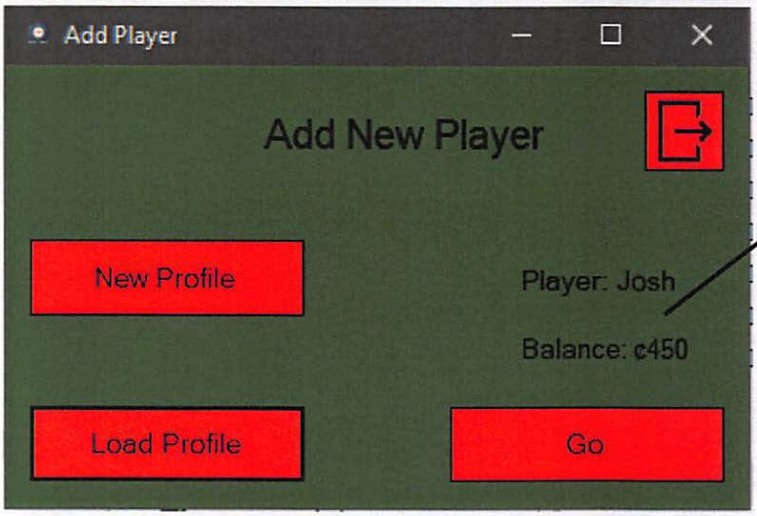
I went bust, this means I lost my ¢150 bet, and my new balance remains as ¢450.



This is the new string held inside the gamesave file.

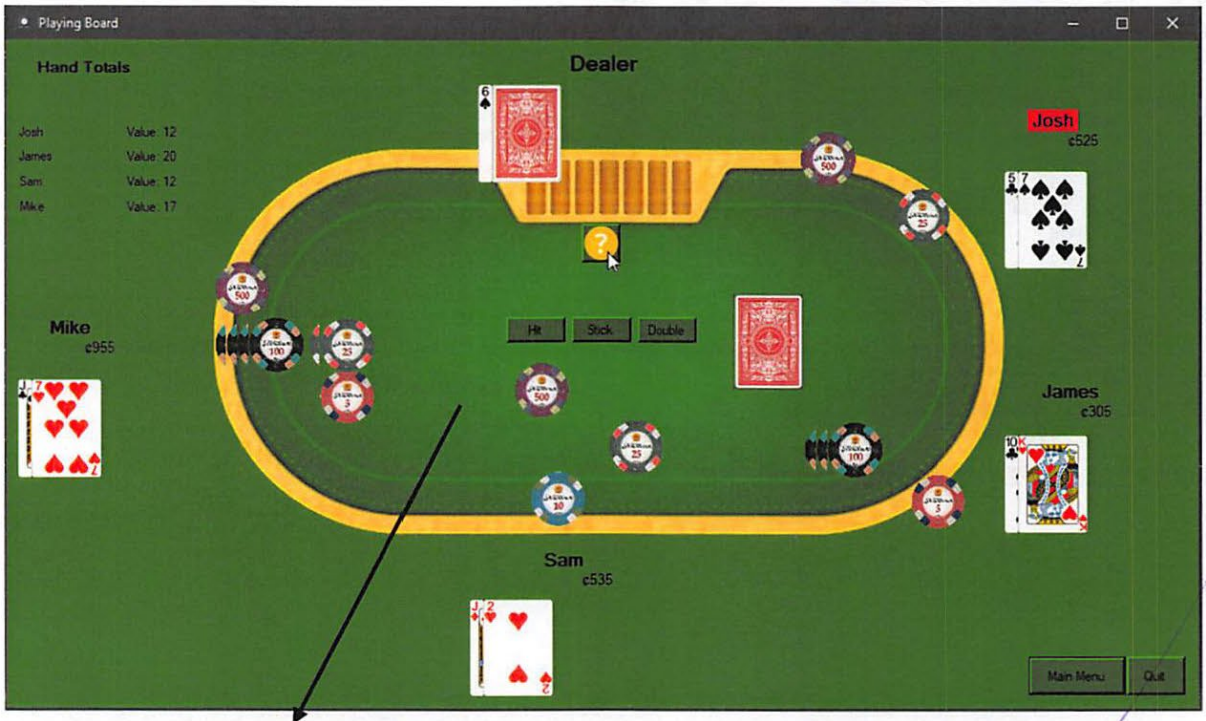


I am going to load Josh's profile again and see if his balance is ¢450, not ¢600.

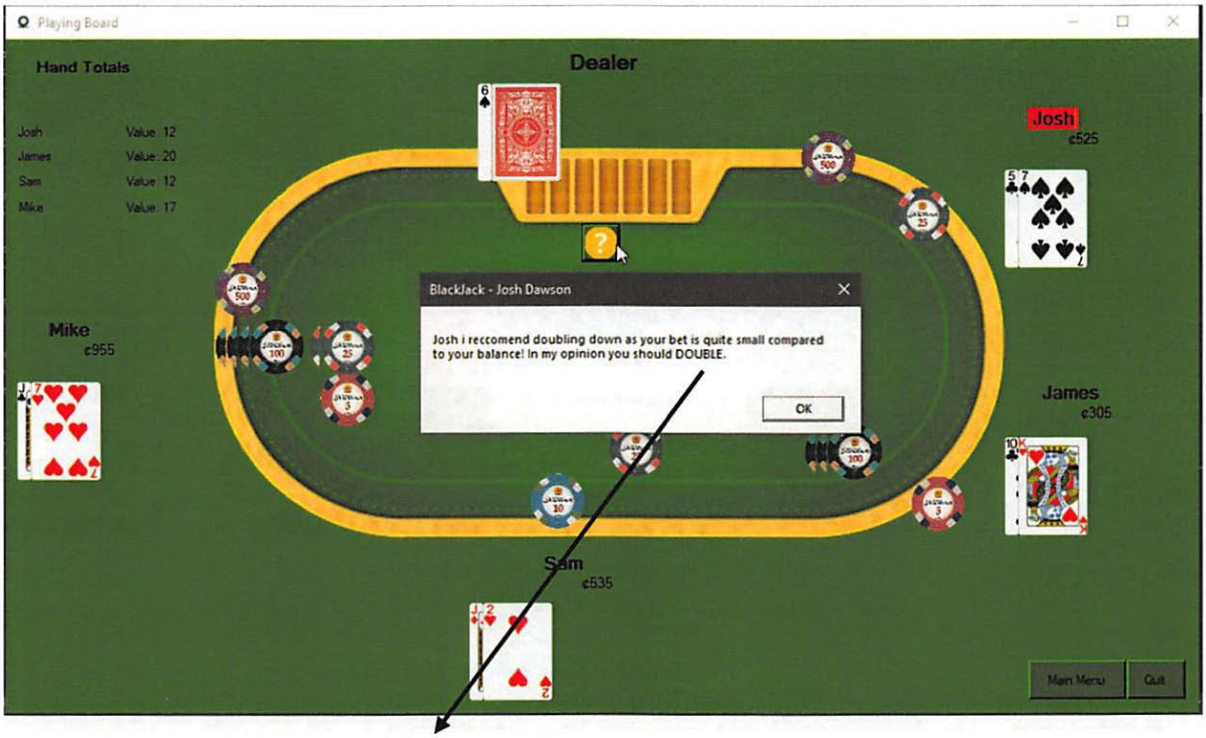


As you can see, Josh's profile loaded in correctly, but his balance is now ¢450, not ¢600 from last game.

Evidence 19 – Test 44

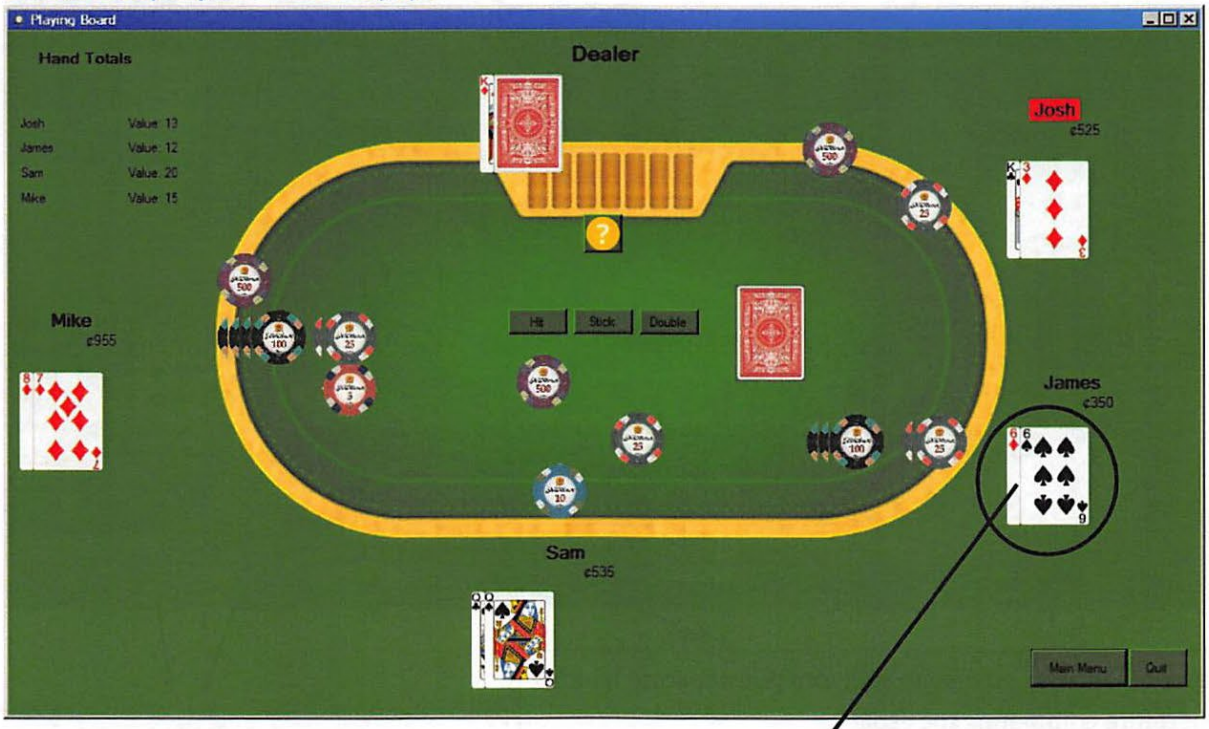


Here I am going to click the hint button. A message box should appear showing a hint for Josh.



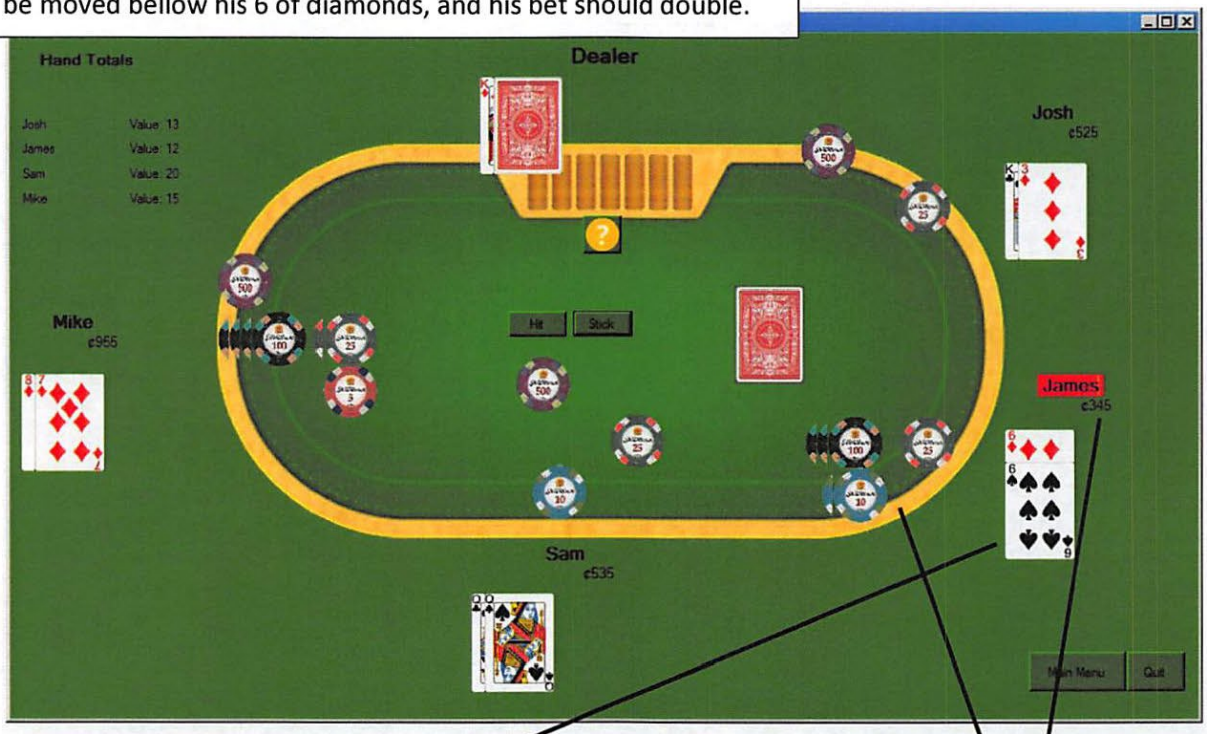
A hint appeared in a message box which gave a hint unique to Josh and his hand.

Evidence20/21/22- Test 36a/b/c



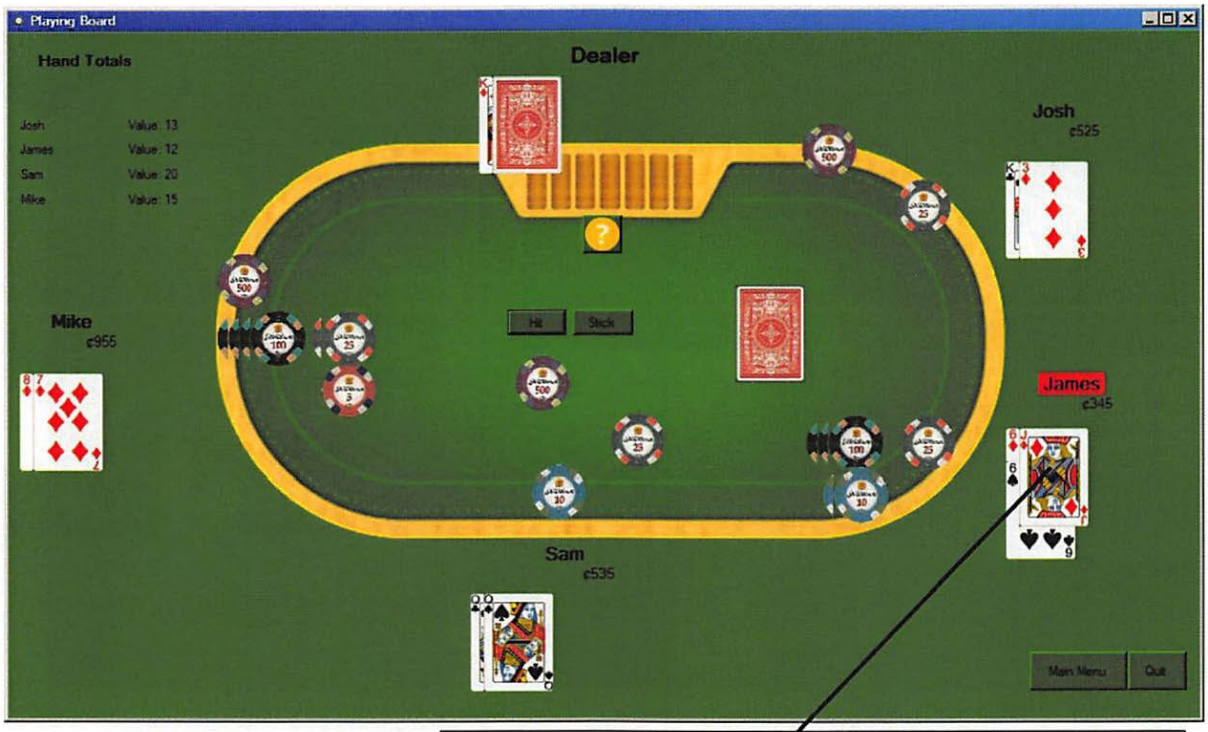
Here you can see James has 2 sixes. Because of this, he should be asked if he wants to split his hand when it is his turn.

James has now been asked if he wants to split his hand, I am going to select the "Yes" option, and James' 6 of spades should be moved bellow his 6 of diamonds, and his bet should double.

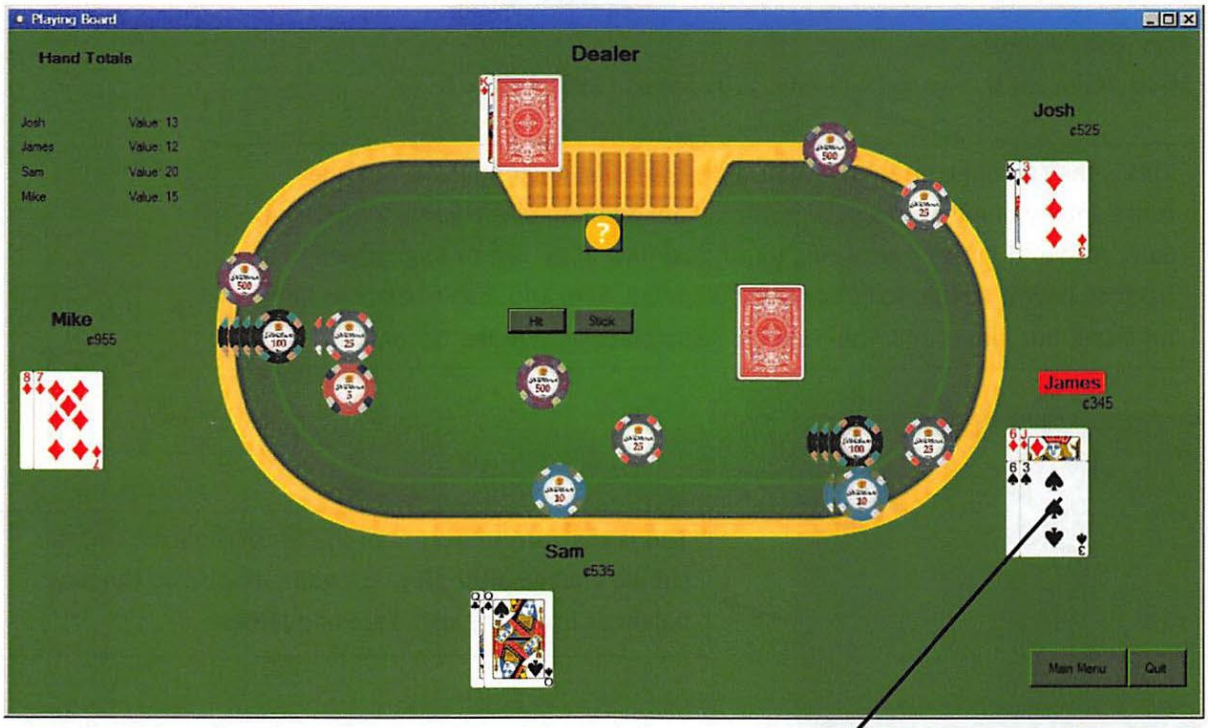


James' hand has now been split into two separate hands, therefore completing the "split".

Due to splitting your hand increasing your bet by double, James' initial bet of ¢5 has been doubled, and his chip counts have also been updated accordingly.



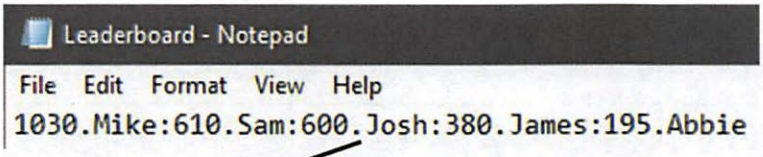
After hitting a card, the Jack of diamonds was dealt to James' first hand, with the correct offset, whilst his second hand (6 of spades) is still visible. If James sticks, it should still be his turn, but his cards will be dealt to his second hand.



After clicking the Stick button, James' name was still highlighted which indicates that it was still his turn. Then, after clicking the Hit button once more, James was dealt a card, but this time to his second hand (6 of spades), and as you can see, the card dealt (3 of spades) was dealt to his second hand, and not dealt after the Jack of diamonds. If I stick as James now, the turn system should move over to Sam.

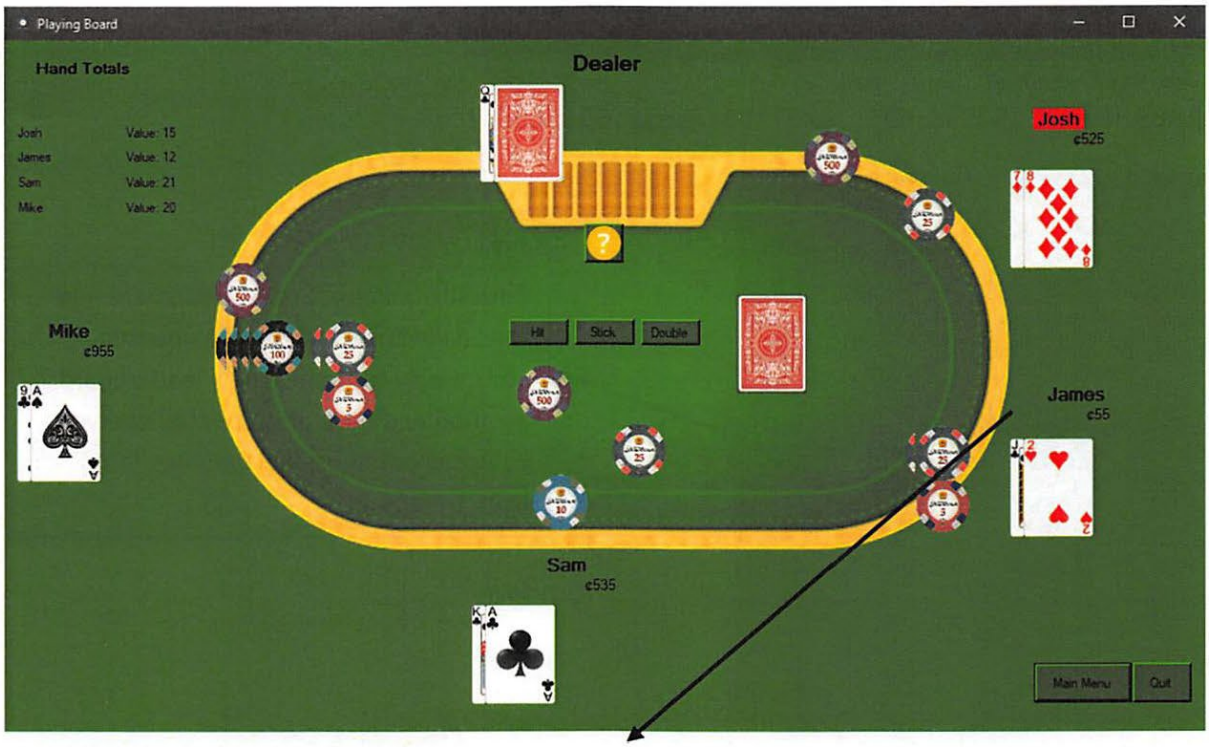
Due to Sam's name being highlighted, you can see that the Turn system moved over to the next player correctly after James stuck with his second split hand.

Evidence 23 – Test 49c

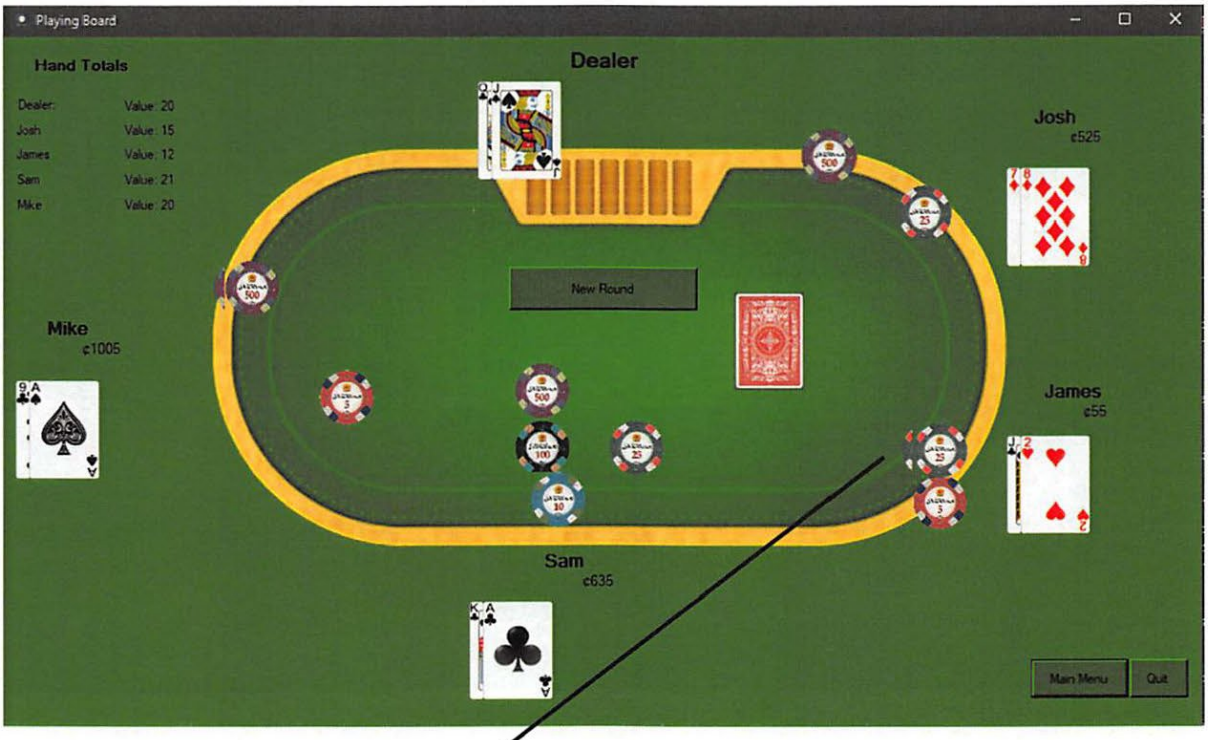


This is the current contents of the Leaderboard.txt document. To test the functionality of my quicksort algorithm, I am going to play a game with Mike, Sam, Josh and James. I am going to purposefully allow James to go down to a balance bellow Abbie, so not only should Mike, Sam and Josh's balances be updated, but James and Abbie should swap places inside the document.

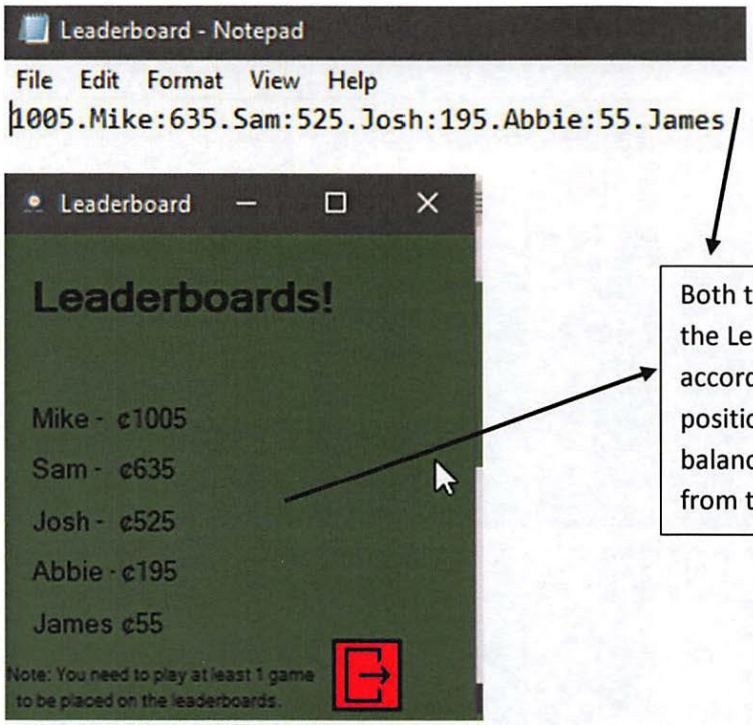
This is the current output of the Leaderboard form, this should change after playing a game to display the new balances for Mike, Josh, Sam and James.



I placed bets for each player but made sure to bet a high amount for James to allow him to drop in the leaderboard.

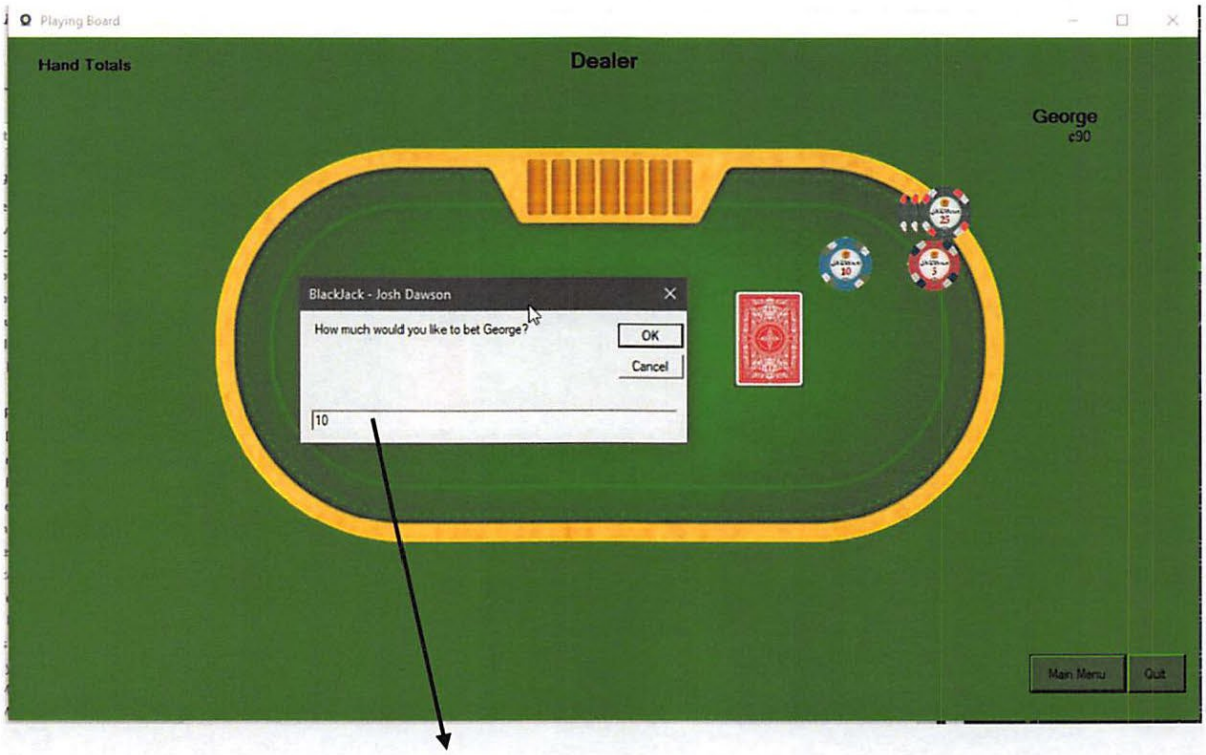


Now that the game has finished, you can see everyones new balances based on who won hands. James did not win; therefore he now has a lower balance than Abbie, and should drop in the leaderboard accordingly.

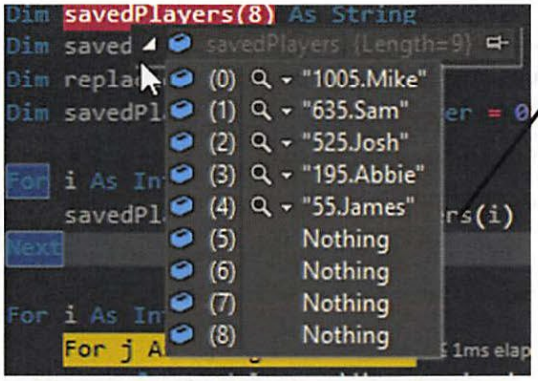


Both the leaderboard.txt document and the Leaderboard form have updated accordingly for James' new leaderboard position and have also updated everyone balances based on their winings/loses from the round.

I am now going to play a game with one new player, George. I am doing this to show that a new player can be added to the leaderboard and the lowest player (James) will be removed.



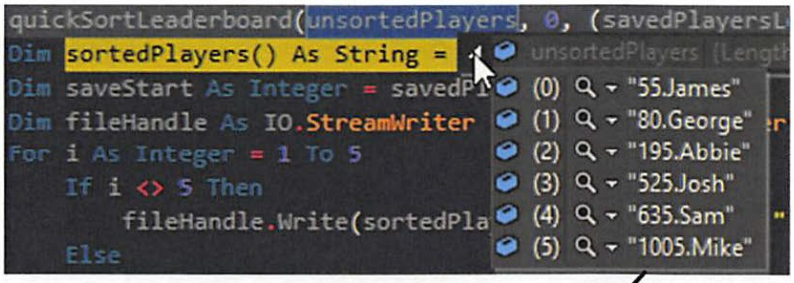
I am going to bet ¢10 of George's ¢90 balance. Whatever the outcome of this game, he will place above James in the leaderboard.



The game is over, inside the savePlayerProfilesAndLeadaerboards subroutine, all the current players in the leaderboard.txt document are loaded into the savedPlayers array shown above.

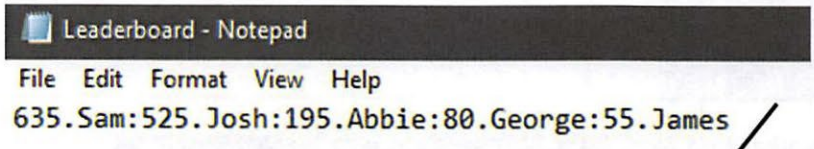
The savedPlayers array is now copied to a new array called unsortedPlayers, and any active players who do not already have a leaderboard place are added (In this case George is added to the end).

This unsortedPlayers array is now passed into my quickSortLeaderboard subroutine with a low value of 0 (the start of the array) and a high value of 5 (The end of the array). This subroutine will perform the quicksort algorithm on the balance of the player, but when swapping values, it will swap the balance and the name, therefore allowing both the balance and name of a player to be quicksorted just based on the balance of a player.



This is the result of the quickSortLeaderboard subroutine. As you can see, it is an array of the same size, but this time the balances are sorted.

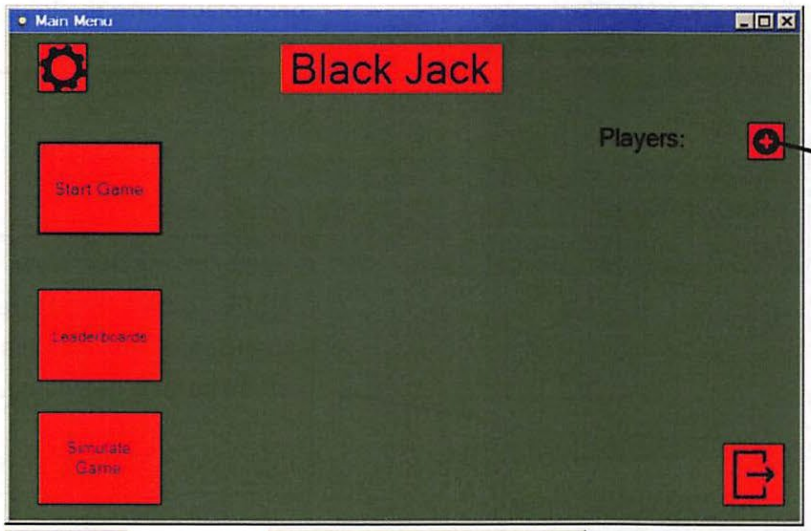
The lowest values are then removed from the array until there are only 5 values remaining. And as you can see, the Leaderboard has updated with George.



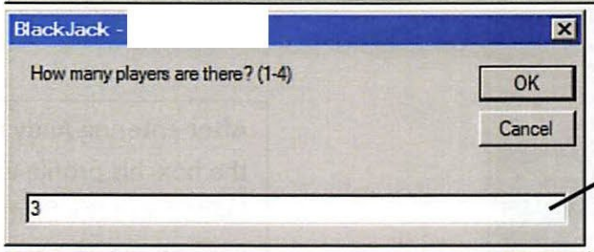
The leaderboard.txt document has also updated accordingly to George's balance.

System Test 1

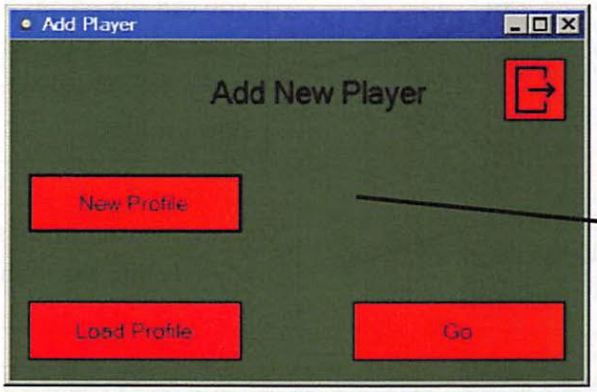
In ths system test i will be testing the ability to play a game with three players, two of which are loaded with the username "Josh" and "George". The final player will be created with the username "Andy". I will be testing if the game can be played from start to finish with 4 rounds, and then that Josh and George's balances are updated accordingly, and that Andy has a profile created with his balance inside.



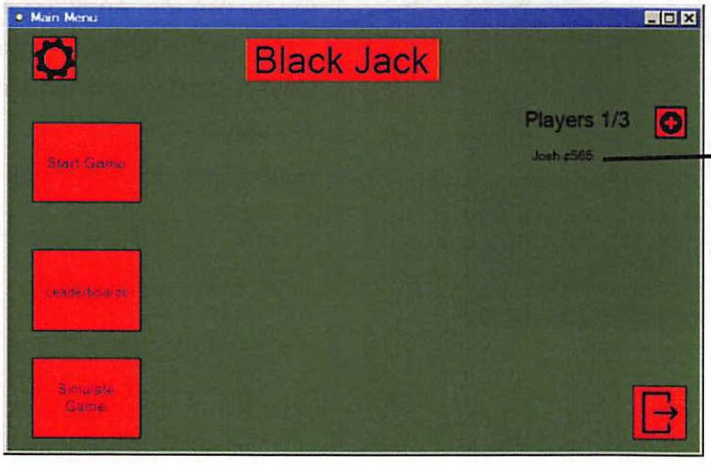
I am going to click the "Add Players" button which will ask me how many players I want to add.



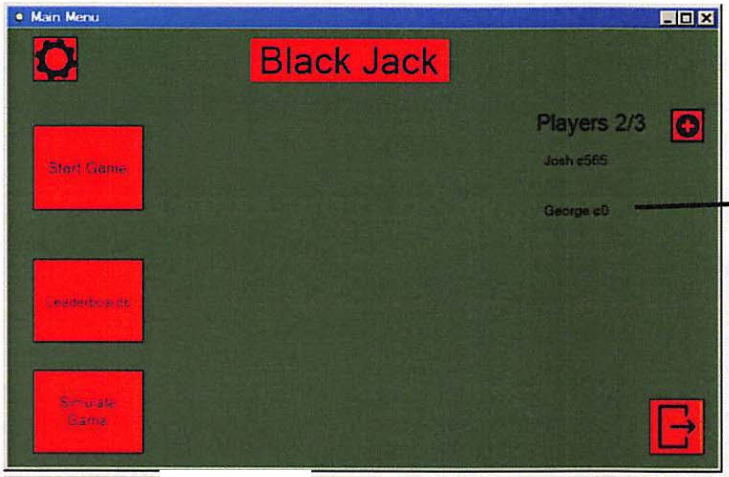
As I mentioned before, I am going to be testing with 3 players.



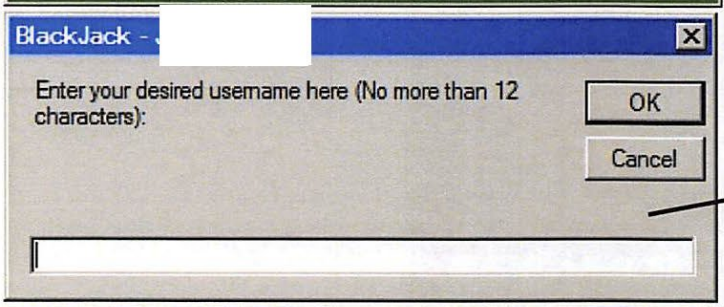
I am brought to the "Add New Player" form, I am going to load two players, Josh and George, and then create a new profile for Andy.



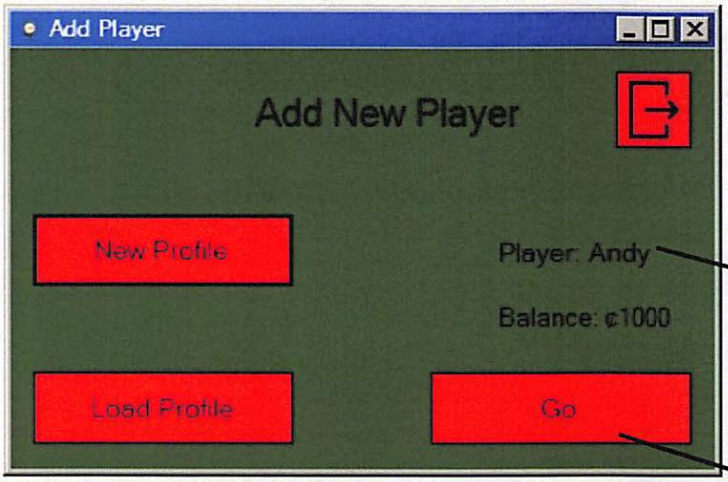
Josh was successfully loaded as his balance is not the default value of ¢1000, as you can see he has been loaded into the lobby.



I repeated the process for George, and as you can see he has been loaded in with a balance of ¢0, and then has been places into the lobby.

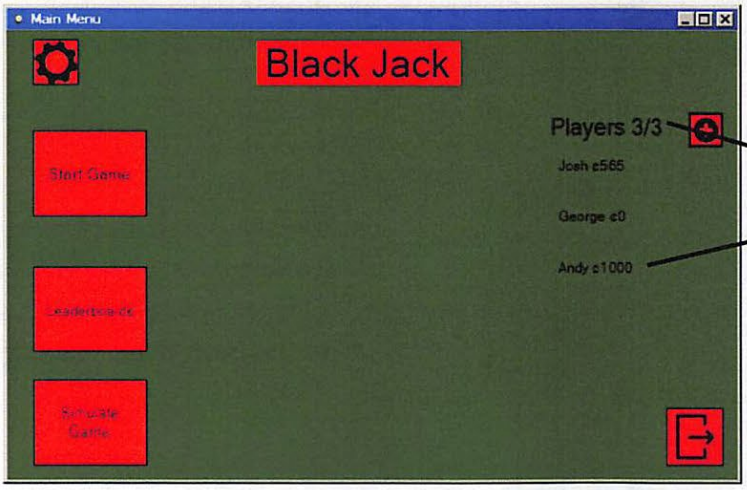


Upon clicking the "New Profile" button, I was greeted to this input box which asked me to enter a username.

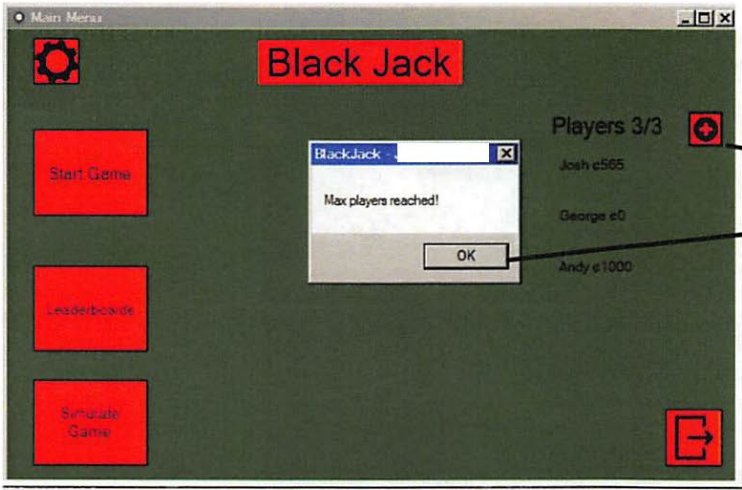


After entering Andy into the box, his profile was loaded into the form. As you can see he has a balance of ¢1000, which is the default value for new profiles.

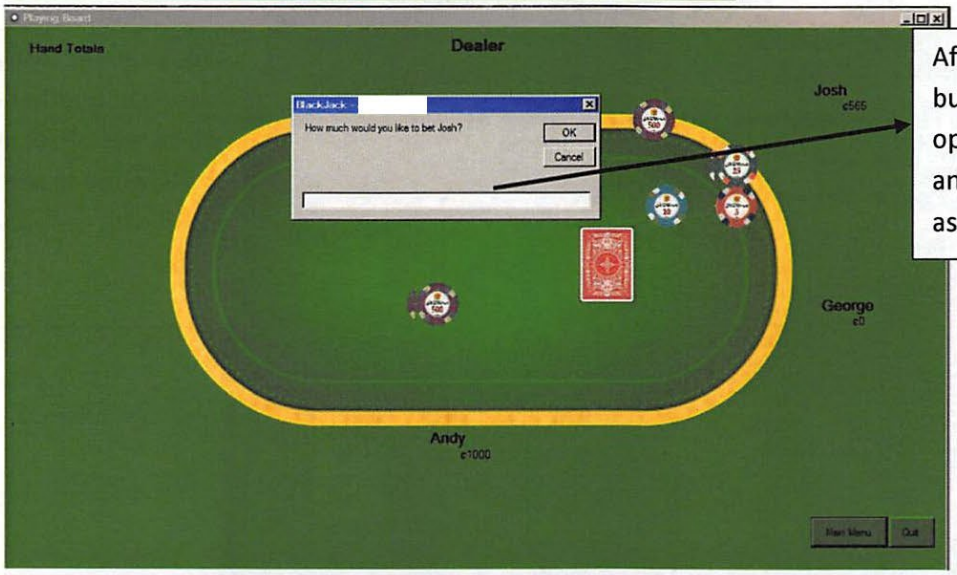
Clicking the Go button should load Andy into the lobby.



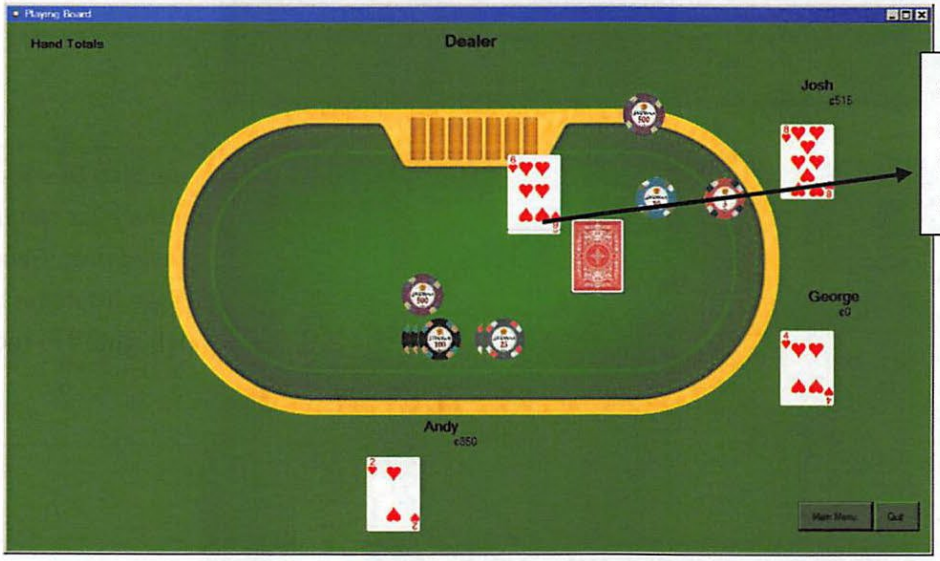
As you can see, Andy has been added to the lobby, and the players label is now showing 3/3 players loaded.



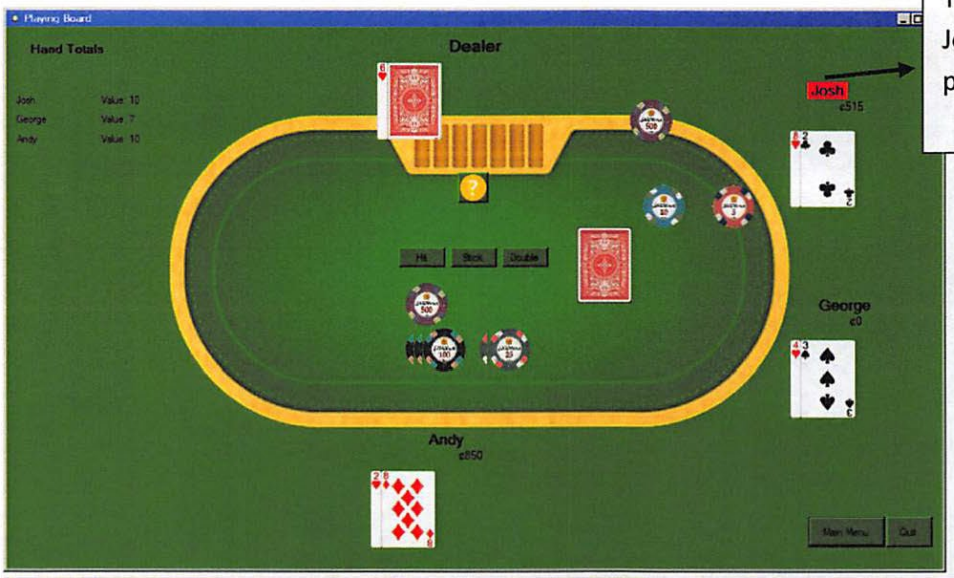
Upon clicking the "Add players" button again, I was greeted with an error.



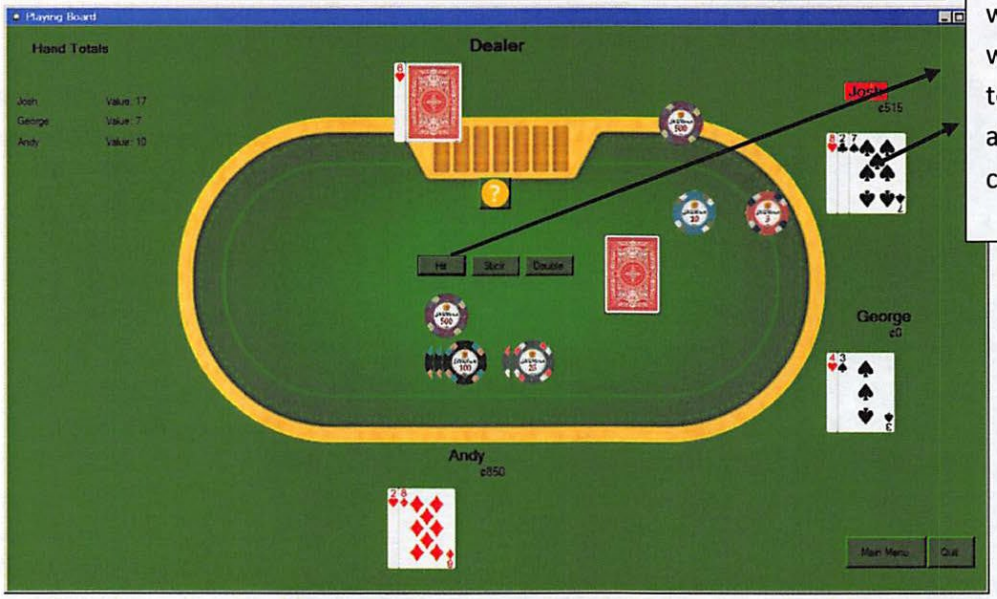
After clicking the "Start Game" button, the Playing Board was opened with all three players and their balances. I was then asked how much I wish to bet.



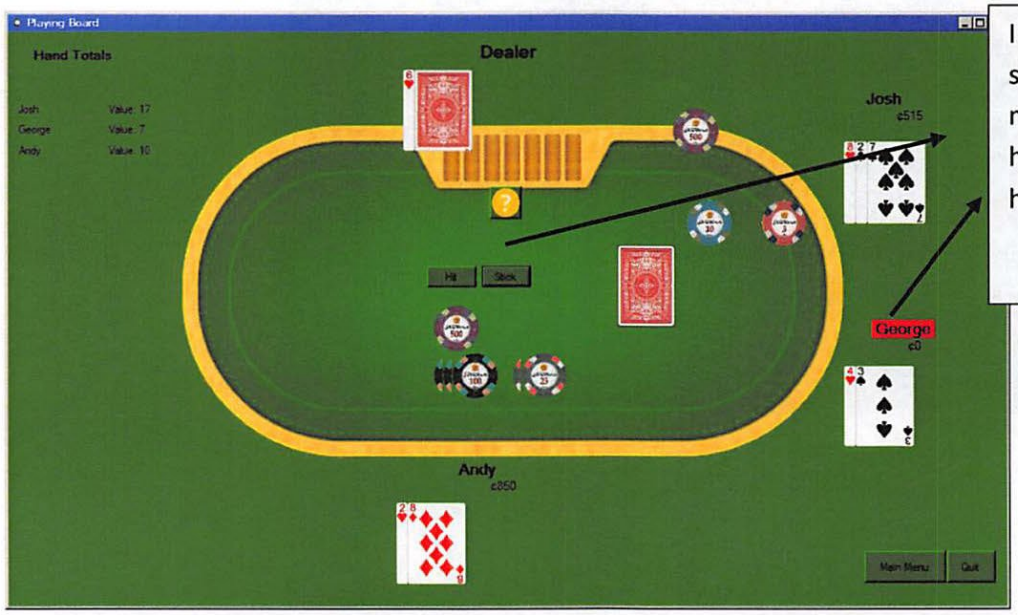
After all players have bet, the game dealt two cards, one at a time, to each player and the dealer.



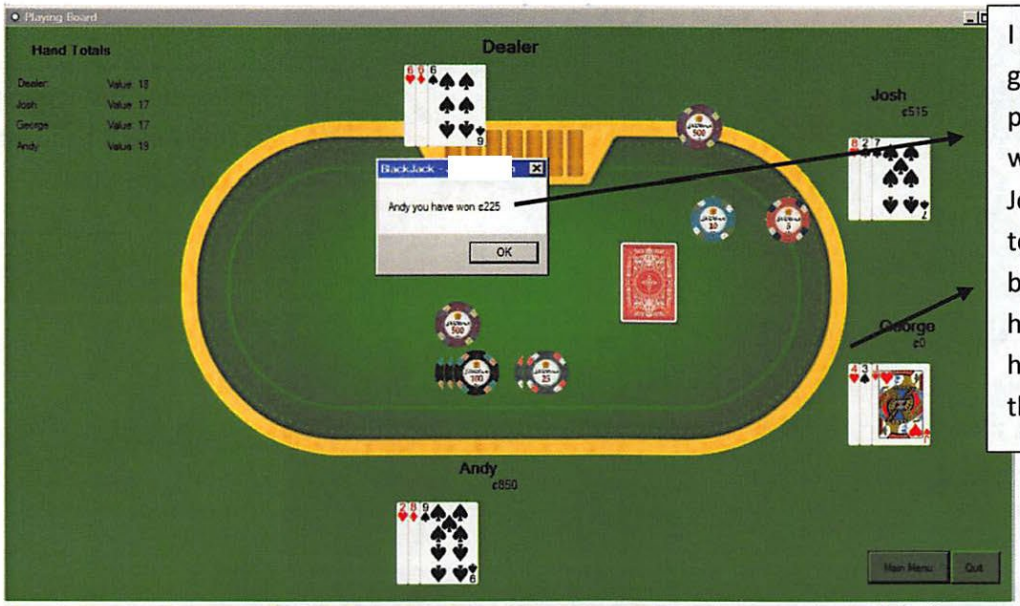
The turn system started with Josh as he is the rightmost player.



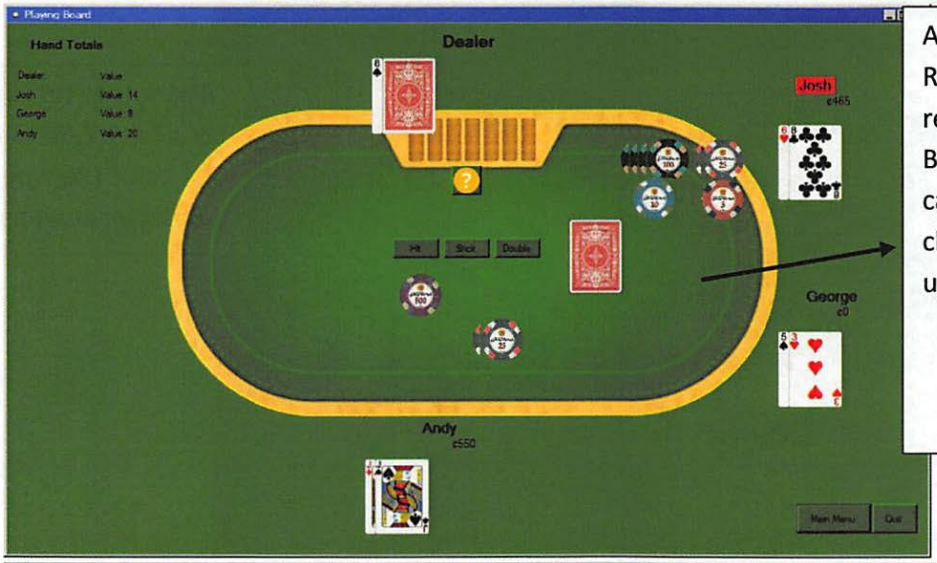
I hit a card as Josh and was dealt another card which was offset in order to allow me to see it along with my previous cards.



I decided to stick as Josh, so the turn system has moved onto George, hence his name being highlighted in red.

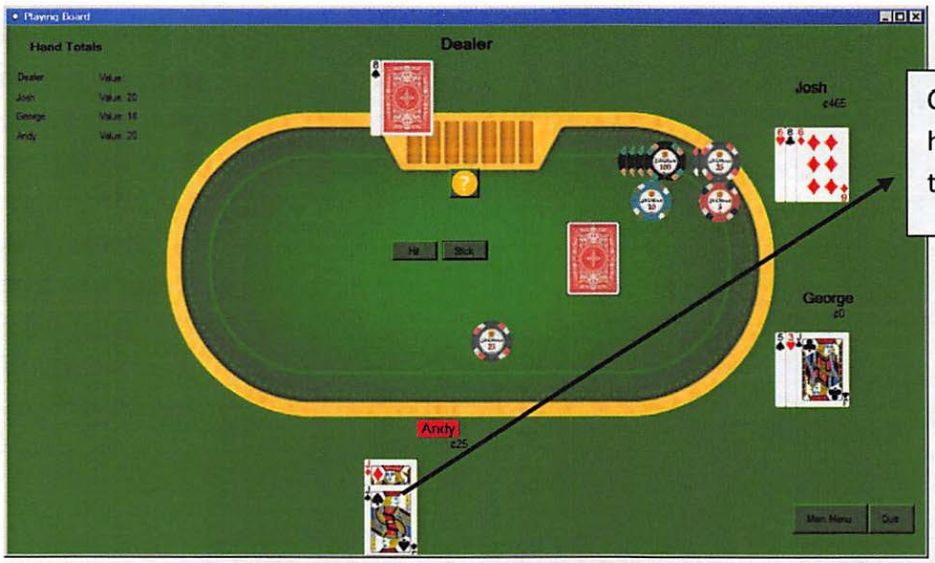


I took George and Andy's goes, and now the game presented the bet winnings for each player. Josh and George lost due to their hand values both being 17, but the dealer had a hand value of 19, hence which Andy was the only person who won.

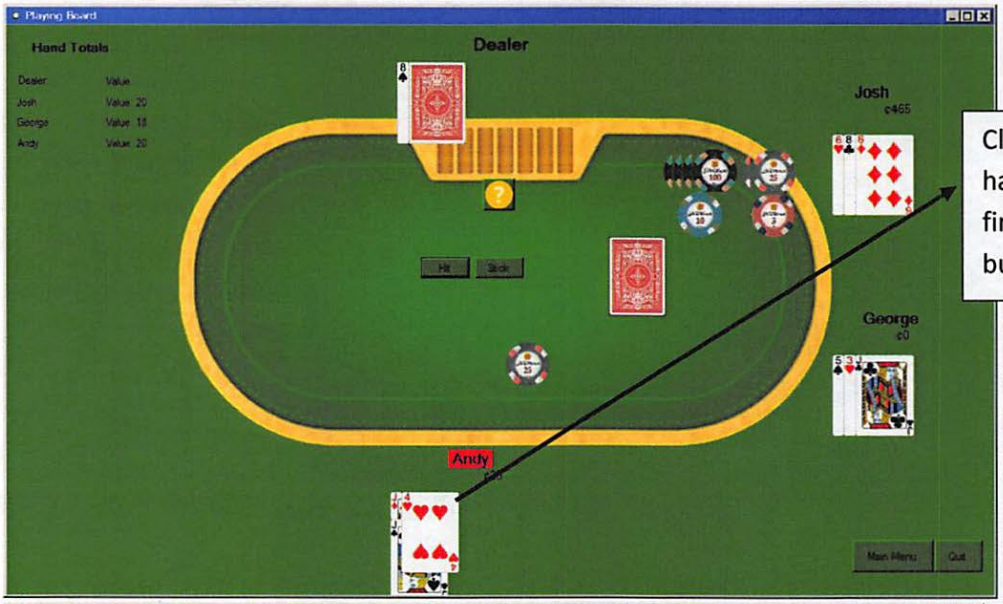


After clicking the "New Round" button, the game reloaded the Playing Board and dealt new cards. As you can see, the chip amounts were also updated.

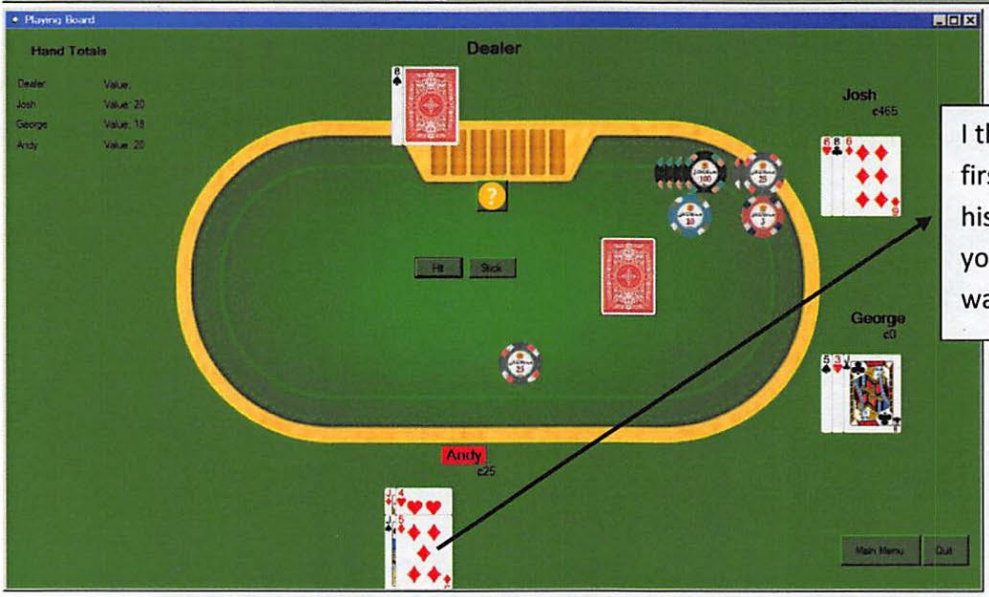
Andy has a "split-able" hand (Two of the same card), so he was greeted to this choice box.



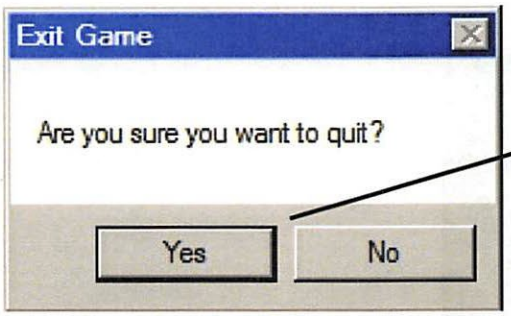
Clicking the "Yes" option has split Andy's hand into two separate hands.



Clicking the "Hit" button, has dealt a card to Andy's first hand. (He would be bust if he hadn't have split)



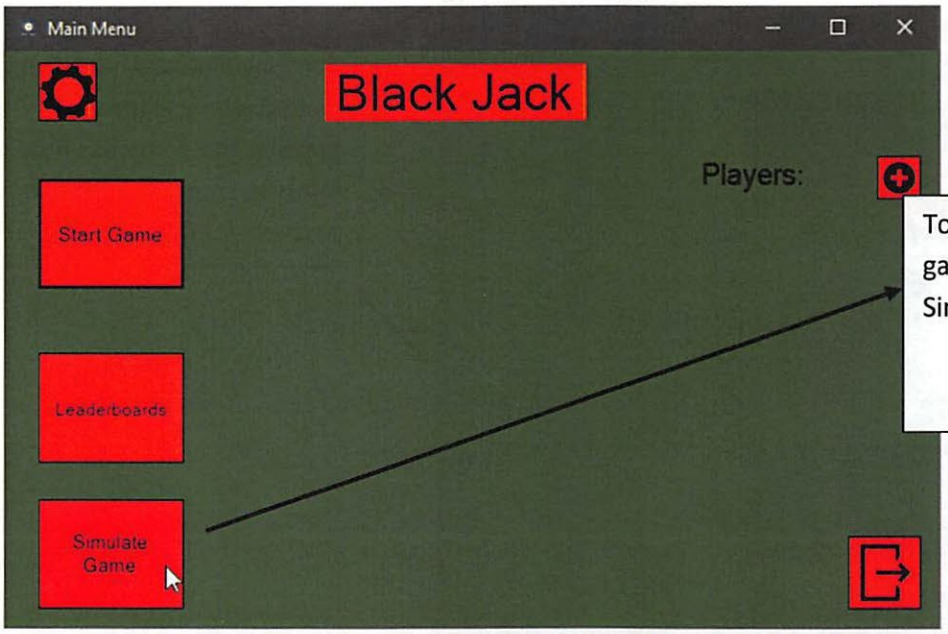
I then stuck with Andy's first hand, and then hit for his second hand. And as you can see, a second card was dealt to his second



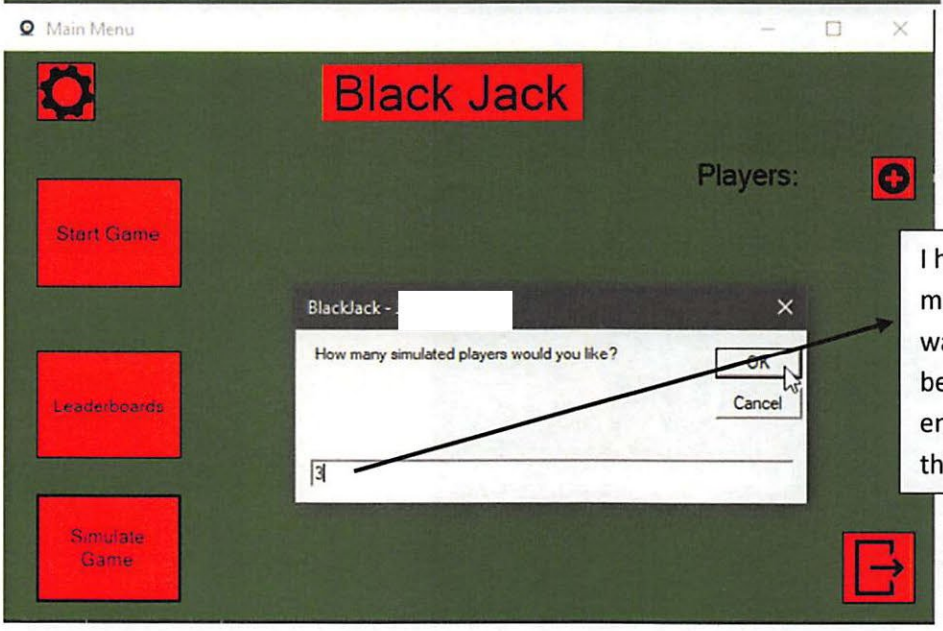
After clicking the "Quit" button, I was prompted with this choice box, and clicking "Yes" closed the program.

System Test 2

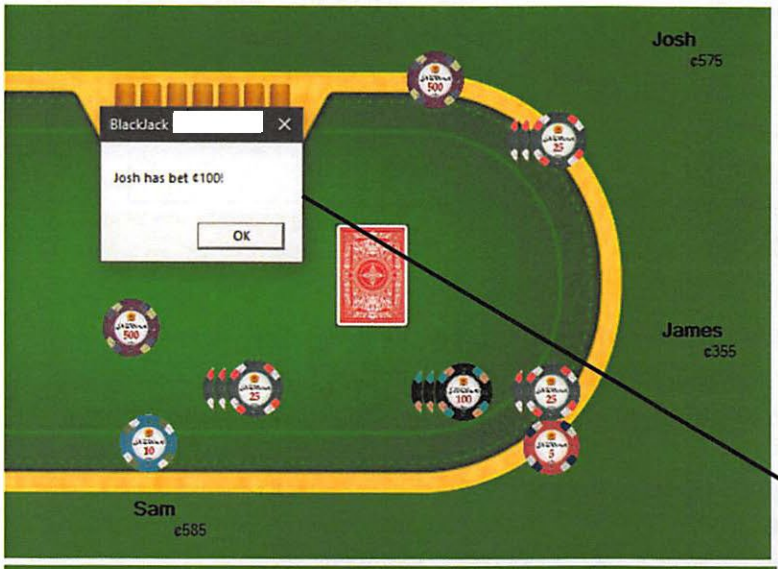
This system test will be testing a simulated game of 3 players consisting of 3 rounds. I will be checking that each simulated player makes different moves based on their hands and they bet different amounts each time and receive different cards every round, and finally that their moves make sense based on their hands.

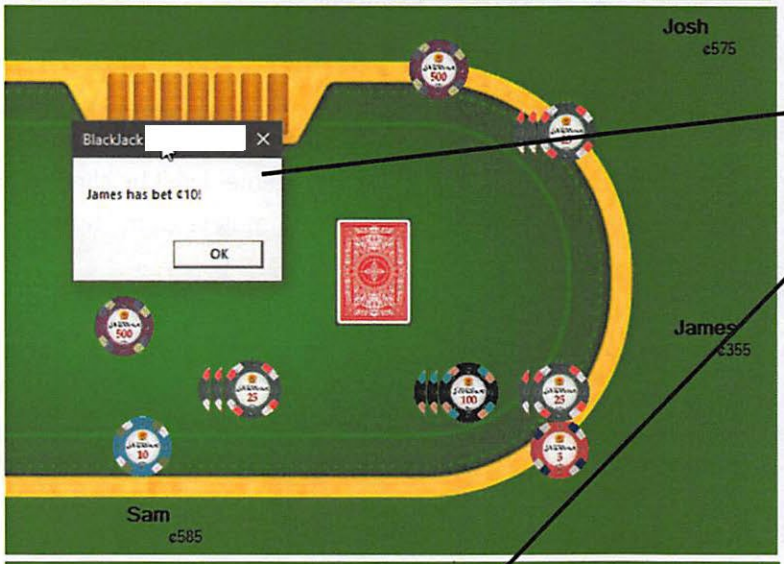


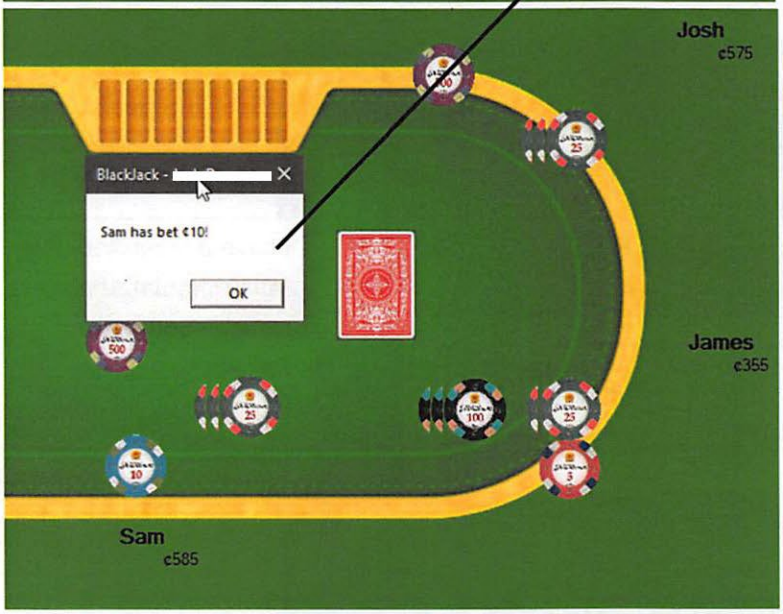
To start the simulated game, I will be clicking the Simulate Game button.



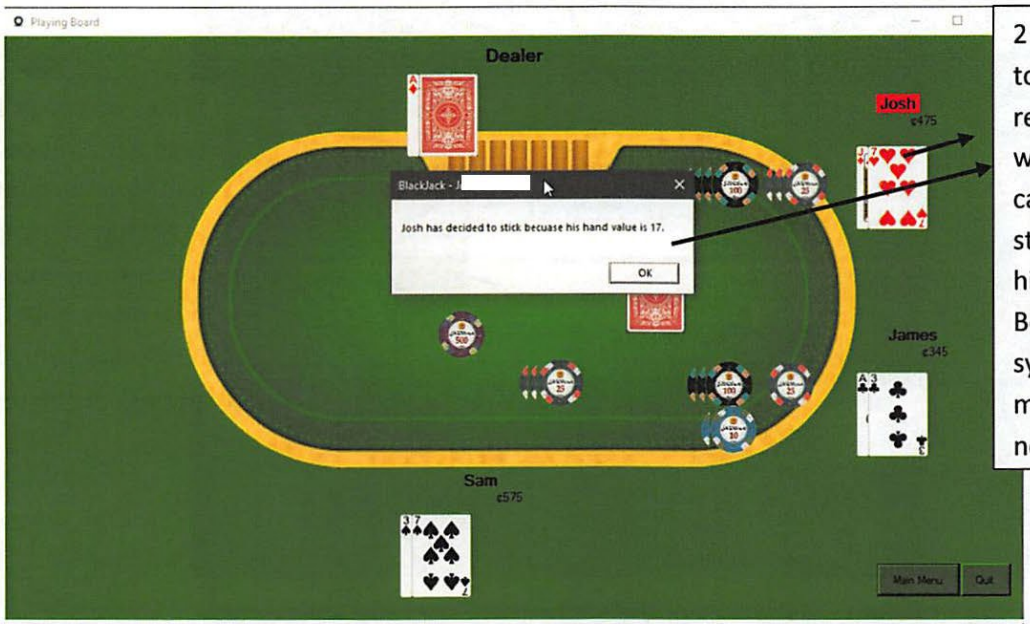
I have been asked how many simulated players I want to add, because I will be testing 3 players, I entered the number 3 into this box.





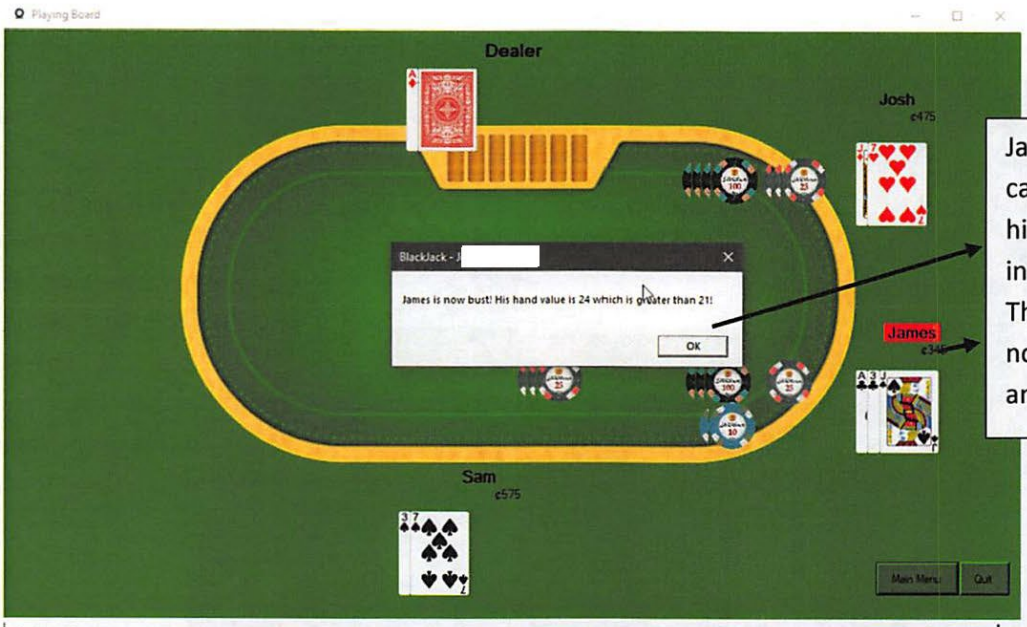


The PlayingBoard has been loaded with 3 simulated players. The game has now informed me of how much each player has bet.

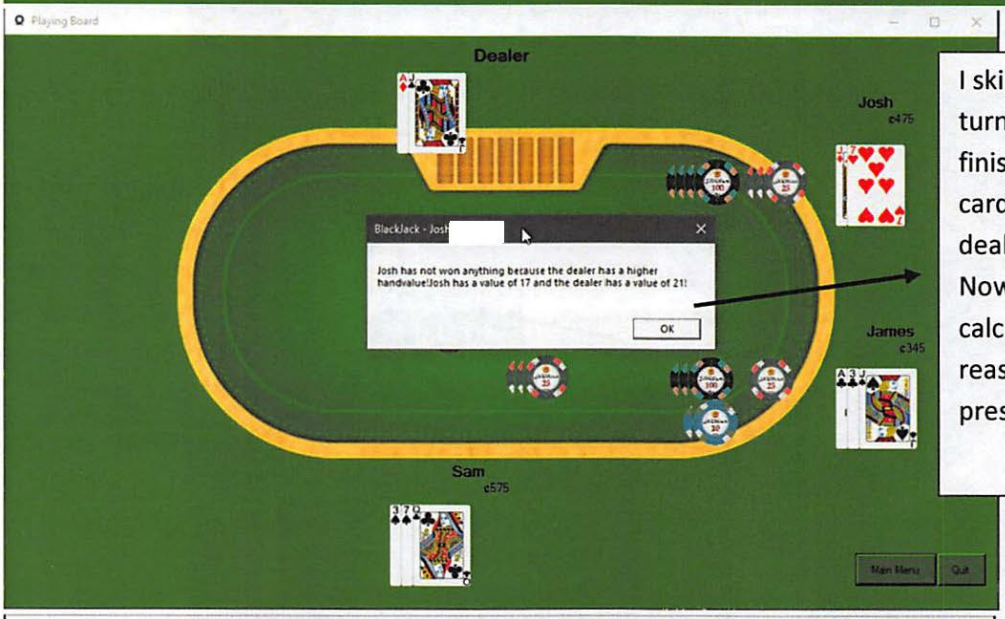


2 Cards were then dealt to each player, and then reasoning for Josh's turn was displayed. Here you can see Josh decided to stick because he had a high handvalue (17). Bcasue of this, the turn system should now move onto James (The next player).

Reasoning for James' turn was given next. James decided to hit a card, because of this, he should be dealt one more card.



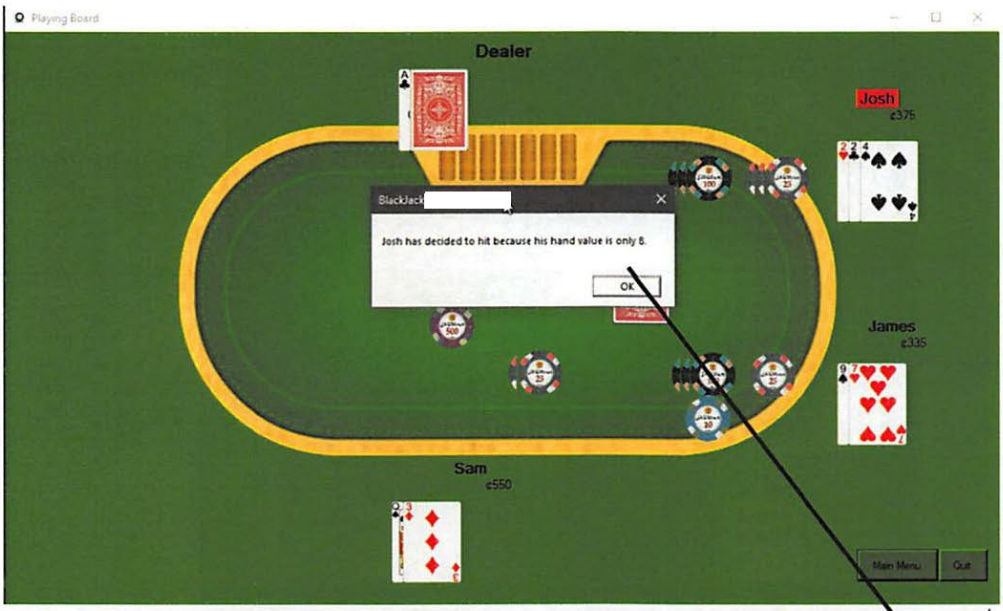
James was dealt another card, but, this caused him to go bust, as shown in his turn reasoning. The turn system should now move onto the third and final player, Sam.

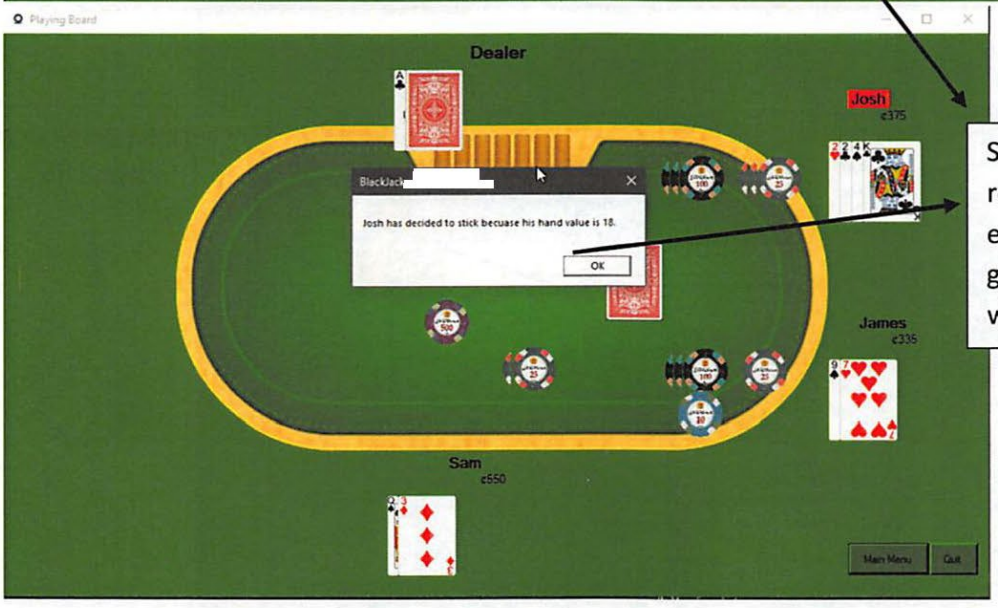


I skipped through Sam's turn. After Sam's turn finished, the dealer's card was shown, and the dealer had their turn. Now, the winners are calculated, and the reason for each win is presented to the user.

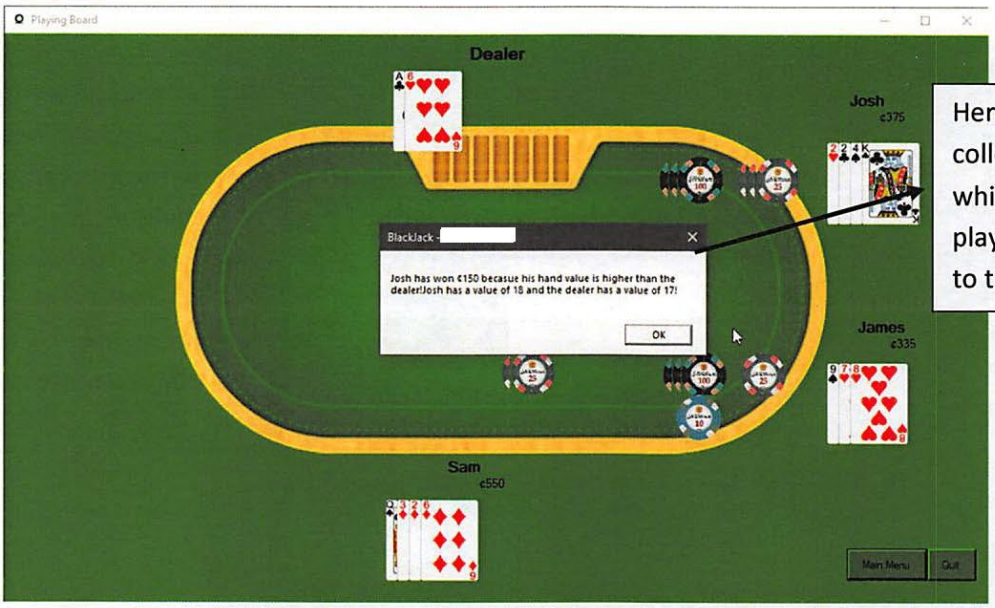
Clicking the New Round button will reset the board and allow each player to bet again.

As you can see, all the players have been dealt new cards, but retain their balances from last round. I skipped through the betting section, and now reasoning for each players' move is given.

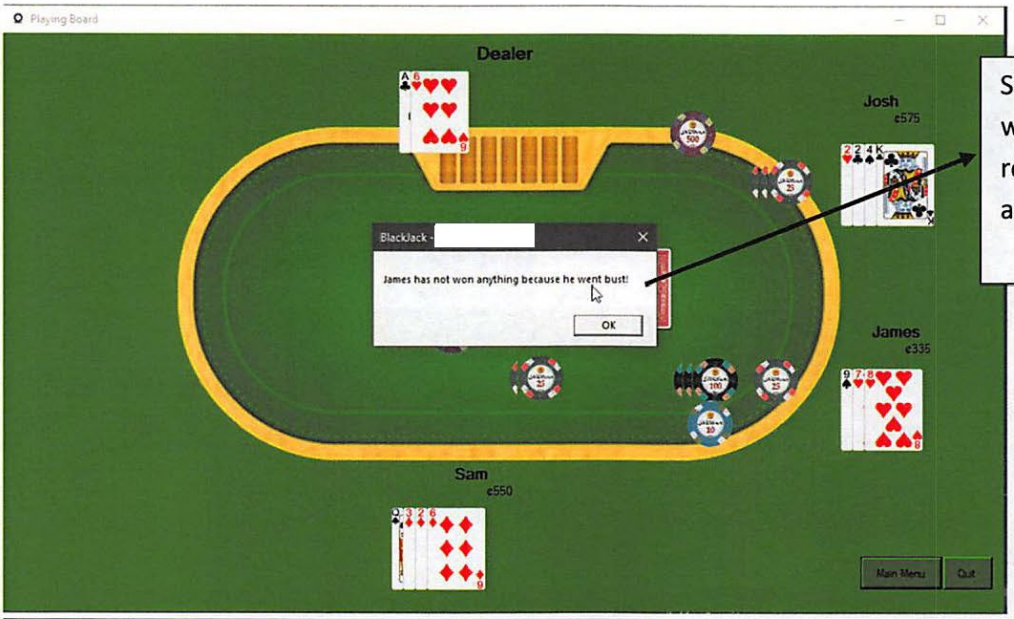




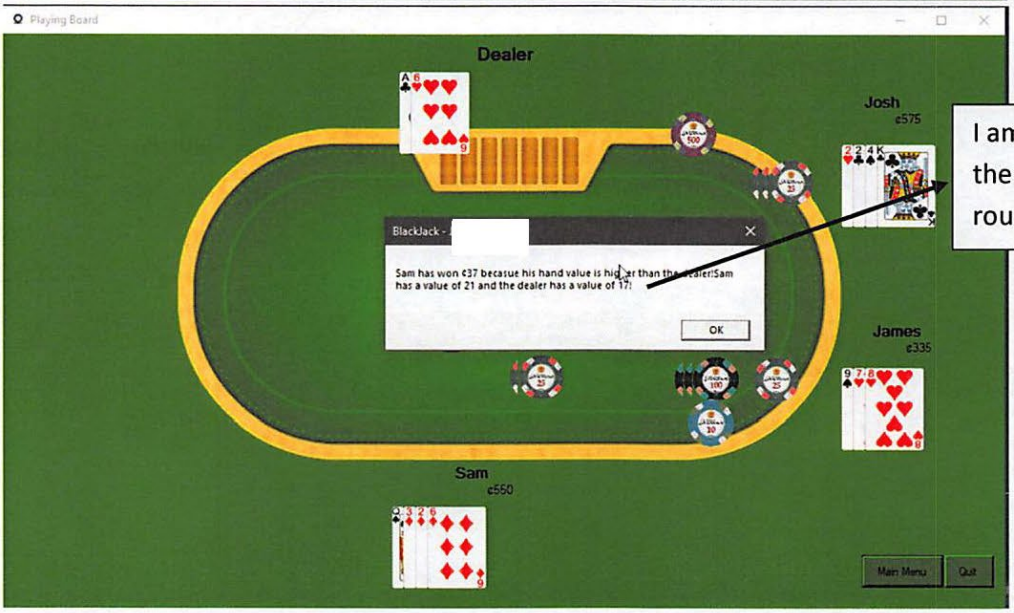
Similarly to last round, reasoning is given for each turn. I am now going to skip to the winnings again.



Here the players are collecting their winnings whilst reasoning for each players win is presented to the user.

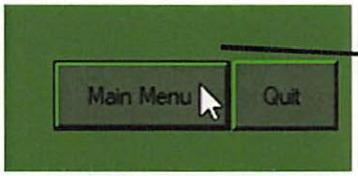


Similarly to a player winning, if a player loses, reason for that lose is also given.

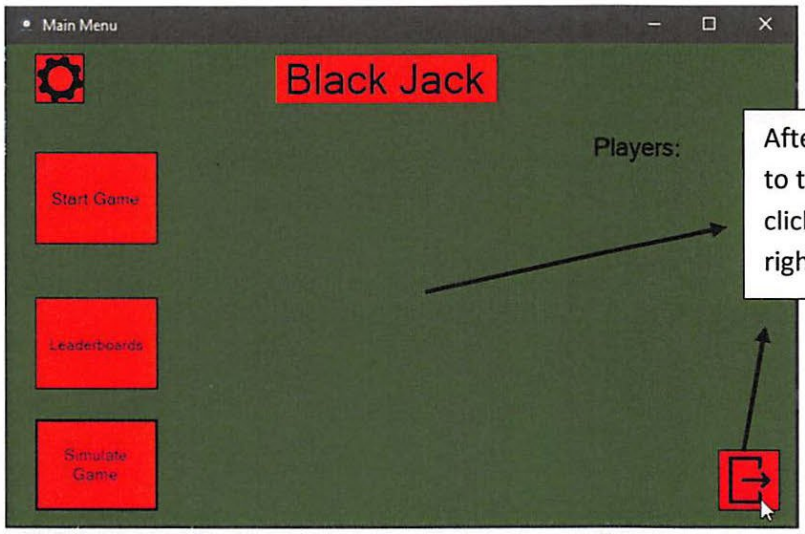


I am now going to start the third and final round.

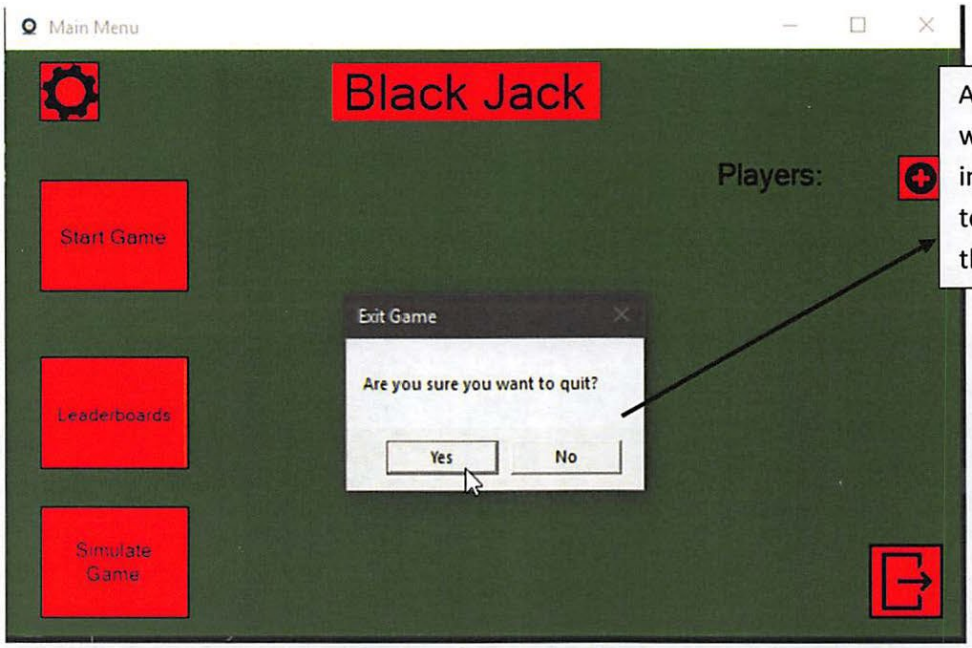
I started and watched the third and final simulated round. As you can see, all the players' balances have progressed and they have received different cards once again.



I am now going to return to the MainMenu by clicking the Main Menu button as I have played the target 3 rounds.



After clicking the button, I was returned to the MainMenu. Now, to exit I must click the Exit Game button in the bottom right.



After clicking the button, I was prompted with this input box. Because I want to quit, I will click "Yes", and the program will close.

Evaluation

General Appraisal

Overall, I believe my project has gone very well. I feel that I have gained experience dealing with large projects and planning them. My analysis section helped me gain understanding from my end users and allowed me to tailor the program to an end user's needs. My design section helped me greatly when creating a robust system which removed issues with existing systems. The complexity of my program required me to use new coding techniques such as Object-Oriented programming, and Event Driven programming. Creating the split-aspect of blackjack was one the most difficult and challenging part of my program and caused delays and issues, but, I feel my final solution meets my end user's needs, therefore I believe the project was a success. A second challenging and difficult aspect of my solution to create was the sorting for the leadboards. Due to players being saved into the leaderboard text document with their name appended to their balance, I had to modify my quicksort algorithm to only look at their balance whilst applying the quicksort algorithm, but still retain the appended username. This caused significant delays but luckily, I managed to complete it in time. I feel that the dealing went very well as I expected it to be very challenging, but I managed to complete it quickly. Using static x and y co-ordinates which represent each player "base" card location, and then adding to these for each card proved to be an excellent and effective method of dealing.

Meeting Objectives

#	Objective	How it was met?	How well it was met?	How could it be improved?
1	The form must load with a main menu, having buttons to start a new game, load a game, simulate a game and view leaderboards.	I created buttons inside the IDE which lead to new forms when clicked.	Perfectly	No improvements.
2	If the "start new game" option is selected the game must ask the user for their name and store that name for later use.	I used input boxes with validation and assigning their input to a variable inside the player class called "name".	Perfectly	No improvements.
3	If an invalid name is entered, you will ask for re-entry.	I created DO loop checking their entry is valid (It is not longer than 12 characters and contains no numbers).	Perfectly	No improvements.
4	The board then loads a card table in a new form.	I created a new form called "PlayingBoard" which has the appearance of a blackjack table.	Perfectly	No improvements.
5	The chip count of the player is loaded in if they	I created a Sub where the chip count is loaded	Perfectly	Store all players balances and names

	selected to load a game.	from a gamesave txt file with the name of the corresponding player.		in one document reducing the number of files created.
6	If the player didn't select to load a game, 1000¢ is loaded into the balance and chips and a profile is created.	I created a Sub where a gamesave txt document is created with the corresponding player name and a balance of 1000¢.	Perfectly	No improvements.
7	Players set a bet by selecting how much ¢ they want to bet.	I created a message box asking the players how much they want to bet, this value is then assigned to the property "betAmount".	Perfectly	Players could click each chip they want to bet instead of entering an integer value.
8	Cards are dealt, 2 to the dealer, and 2 to each player.	I used DO loops to move cards at small increments until they reach a "base" (X, Y) co-ordinate.	Somewhat perfectly, the cards do not move in a smooth manner.	Use a smaller value to move the cards each second, creating a smoother motion but slowing the dealing speed as a result.
9	If the player has a blackjack, they instantly receive triple their bet, and are out of the current hand.	I passed the player object through a function which calculates their handValue, if it is equal to 21 after 2 cards, they must have a blackjack.	Perfectly	No improvements.
10	One of the dealer's cards are shown to the players whilst one remains hidden.	I used a Sub where the dealer's second card's picture box is made equal to the back of a card before being dealt, therefore when it is dealt, it looks turned over.	Perfectly	No improvements.
11	Players are then asked whether they want to hit, stick, double or split, starting to the right of the dealer and ending to the left of the dealer	I created a FOR loop which loops through the players and allows them to take their move by clicking the buttons displayed in the centre of the board.	Perfectly	No improvements.

12	There are hints which show the hand total of each player	The same function mentioned for objective #9 is used to calculate the players handValue, this value is then displayed inside a label on the PlayingBoard.	Perfectly	No improvements.
13	If a player goes bust, a visual indicator is shown	If the players' handValue totals above 21, they have gone bust and their handValue label (Objective #12) is given a red background and reads "BUST".	Perfectly	No improvements.
14	If a player decides to stick, or goes bust, the game moves on to the next player.	Once the player has had their turn, the FOR loop will move onto the next available player.	Perfectly	No improvements.
15	If a player selects to hit, a card is dealt to them.	I used the same Sub I created to move a card (Objective #8) after the card in position 0 from the deck array is assigned to the players' hand list. The card is dealt with an offset however to allow the player to see all their cards.	Perfectly.	No improvements.
16	Once all players have had their turn, going bust or sticking, the dealer's final card is revealed therefore deciding if each player has lost or won.	I created a Sub which will reverse Objective #10, by changing the Dealer's card Picture Box back to the correct image (The top of the card).	Perfectly	No improvements.
17	The relevant bets are then paid out to each player, depending on whether they just won, or got a blackjack. (2x the bet for blackjack, 1.5x the bet for winning)	I used an IF statement which will check conditions such as the dealer's handValue and the players' handValue to decide their correct bet multiplier. This is	Perfectly	No improvements.

		then passed into an algorithm which awards the correct winnings based on the multiplier.		
18	Player ¢ counts are then updated accordingly.	The balance labels are then updated with the new player balances. This is done using a simple FOR loop.	Perfectly	No improvements.
19	The game then automatically saves the updated chip counts for each player.	I created an IF statement which calculates their balance MOD every chip number and calculated the remaining balance, and then MODs this value with the next lowest chip and so on... to calculate the number of each chip the player has.	Somewhat perfectly, chips are not calculated dynamically which means there is a maximum amount of chips a player can have before the program will crash.	Create the chips dynamically based on the player's balance, allowing the game to not crash above a balance of "2560".
20	If the players click the "Yes" option in the "Return to Main Menu" choice box, they are redirected to the MainMenu form.	The MainMenu form is loaded whilst the PlayingBoard form is hidden.	Perfectly	No improvements.
21	If the players click the "No" option in the "Return to Main Menu" choice box, they remain on the PlayingBoard form.	The MainMenu form is not loaded and the PlayingBoard form remains in view.	Perfectly	
22	If the "Simulate Hands" button is clicked from the MainMenu, the user is asked how many simulated players they wish to have.	A message box appears allowing the user to enter any number of players between 1 and 4. This is validated for any number outside this range, and for a non-numerical input.	Perfectly	No improvements.
23	The PlayingBoard is loaded and then the number of players the	I used a FOR loop to create the number of player classes the user	Perfectly	No improvements.

	user asked for are loaded.	asked for, and then loaded them into the game using pre-set names and balances.		
24	Cards are dealt similarly to that of a normal game, except they are not playing, it's an AI.	I used a modified version of the subroutine mentioned in Objective #8.	Perfectly	No improvements.
25	The AI will look at the cards, if their cards have a low value and their bet it minimal, the AI will either hit another card, or double down, depending on the total of their hand.	A subroutine looks at the AI players' cards and decides whether to hit or double depending on the players' hand value and bet.	Perfectly	The subroutine could be improved by looking at other players' cards as well as the dealers and their own to make a more accurate decision.
26	If the AI player has a large hand value, then the AI will stick with their cards as they do not want to go bust.	The same subroutine from Objective #25 is used to look at the AI players' cards, and if their hand value is large then the player will stick.	Perfectly	Similarly to Objective #25, the subroutine could be improved by looking at other players' cards to make a more accurate decision.
27	If the AI has two of the same cards, it will always split its hand.	The same subroutine is used from Objective #25 which will look at the AI players' cards, if the AI has two of the same cards, the subroutine will cause their cards to split.	Perfectly	Similarly to Objective #25, the subroutine could be improved by looking at other players' cards to make a more accurate decision.
28	Whatever the decision for the AI, the player will be presented with justification as to why the AI made the decision it did.	Depending on the decision the AI player made, a message with customised values such as bet values and hand values will be generated and displayed to the user through a message box.	Perfectly	Tailor the responses more to accommodate more different responses.
29	There is a button allowing the user to decide when to stop simulating hands.	Creating buttons which hide the Playing Board and show the MainMenu upon	Perfectly	No improvements.

		clicking.		
30	If the user selects "View leaderboards", all available user game saves are read, and compiled into ascending order.	I created a new form which contains 5 name labels and 5 balance labels for the top 5 players.	Perfectly	No improvements.
31	The top 5 users are then presented to the user with their corresponding chip counts.	A subroutine loads the player names and balances from the leaderboards txt document and performs merge sort to sort the top 5 players, and then show these players in the labels.	Perfectly	No improvements.

End-User Feedback

End-User Feedback

I showed my finished program to my end user (, and he tested my program. He then gave me feedback on my program, and bellow is the feedback he gave:

- The program meets all the requirements I gave you,
- The main menu UI is very easy to navigate,
- The saving/loading of user profiles works very well, but I think the UI for creating or loading needs to be reworked as it is complicated for new users,
- The actual gameplay takes longer per round than I would have liked, but I understand that it would be difficult to allow all players to take their move at the same time,
- I like the addition of sound, but, I feel that focus should have been on improving aspects such as the UI before adding unnecessary features like sound,
- The buttons for taking your moves are easy to interact with,
- The simulation feature is an excellent way for new players to learn the game despite the AI being very basic, an improved AI would be a nice addition in the future,
- The hints feature is a good addition along with showing the hand total for each player,
- I think being able to bet by clicking your chips instead of typing a numerical value would b e a nice addition thus reducing the opportunity for invalid entry.

Analysis of Feedback

My end user seems very pleased with the system which I created and commended the ability for new players to learn the game very quickly using the simulation feature. The only issues that were raised were about the complexity of loading from an existing game save, and the length of gameplay.

When designing my main menu, I considered various techniques to lead the user to the different features of my program. I decided to stick to a colour palette of only two colours. Those colours being red and green. I am glad that my end user has recognised the simplicity of the main menu despite critiquing the UI for loading a gamesave and creating a profile.

I agree with my end user that the actual gameplay element of my solution is very slow paced. However, my coding ability limits me to be able to create a solution which allows all players to take their turns simultaneously. Also, issues with the speed of cards being dealt were raised, but this cannot be fixed as allowing cards to be dealt quicker would result in a less-smooth dealing motion.
I also agree with my end user that the simulated game feature is excellent for new players but the AI for the simulated players is simple. I believe that time and my coding ability caused the AI not to be as complex as my end user may have hoped. I also had to bear in mind that the AI can only look at their cards, and the other players' cards, this still leaves a large probability that the AI may chose the wrong move due to there being a maximum of 43 unknown cards at once. I agree that an improved AI would be a good feature to add in the future.
Finally, I also agree that the ability to click chips and select your bet that way is a much better system than my message box system. But, that system was what was planned, but due to restrictions in my coding ability, I could not get this system to work and had to compromise with the message box system.

Suggested Improvements

The inclusion of simultaneous turns would be a nice feature that should be added in hopes of speeding up gameplay. The main issue that I encountered when try to implement this feature was that all players use one computer to play, this means having 4 players make 4 different moves would require 4 mice and 4 instances of the playing board to be open which is impractical. This could be solved by allowing players to access the game from different computers whilst playing against each other, however because of my lack of programming ability, and time constraints, this solution would not be viable.
Another addition which would have complimented the program is the ability to bet by selecting which chips you wish to bet. The issue with this was addressing each chip and updating their location and storing the amount they have bet before the player confirms this value. I believe this feature could be implemented in an updated version of the solution further down the line however. This would remove one point of error and in turn speed up the program because you can play the game without the keyboard once you have loaded or created your profile.
A third improvement which I could implement is the dynamic addition and deletion of cards and chips based on which cards and chips are in play. Doing this would reduce the amount of RAM which the program takes up due to the program having to load all 52 cards and 125 chips. This improvement may allow for the game to run on older machines with less RAM. Or has the possibility of speeding up the game.
A stronger AI inside the "Simulate Game" portion of my program would be good improvement to my game. The AI currently is basic and does not analyse other players' cards to make a loose prediction about what card may be drawn next therefore improving the guess, even just a small amount. Unlike some of my previous suggested improvements, this improvement may be complex to implement due to programming difficulty, and this improvement was not implemented in my program to begin with due to time constraints. This improvement could also be complimented with improved and more accurate AI reasoning for their move. Currently there is a small variation of reasoning the AI can give for making their move which may lead to newer players making incorrect moves based on the reasoning they read during a simulated game.

Comment 2
Comment 3
Comment 4
Comment 5
Comment 6