Ambar Vishnoi

TExas Hold’Em Poker



Contents

[1.0 Analysis of the problem 2](#_Toc85301916)

[1.0.1 Summary of the project 2](#_Toc85301917)

[1.1 Problem Identification 2](#_Toc85301918)

[1.1.1 Features that make the problem solvable by computational methods 2](#_Toc85301919)

[1.1.2 Why the problem is amenable to a computational approach 3](#_Toc85301920)

[1.2 Stakeholders 3](#_Toc85301921)

[1.2.1 Identifying the end user 3](#_Toc85301922)

[1.2.2 Why the solution is appropriate for the end user 3](#_Toc85301923)

[1.3 Research 4](#_Toc85301924)

[1.3.1 Similar problems and solutions 4](#_Toc85301925)

[1.3.2 Essential features of the proposed solution 9](#_Toc85301926)

[1.3.3 Limitations of the proposed solution 10](#_Toc85301927)

[1.4 Specifying the proposed solution 11](#_Toc85301928)

[1.4.1 Solution’s requirements 11](#_Toc85301929)

[1.4.2 Success criteria 12](#_Toc85301930)

[2.0 Design of the solution 14](#_Toc85301931)

[2.1 Decomposition of the Problem 14](#_Toc85301932)

[2.1.1 Breaking Down the problem 14](#_Toc85301933)

[2.1.2 Justification 14](#_Toc85301934)

[2.2 Describing the Solution 15](#_Toc85301935)

[2.2.1 Explaining the Structure 15](#_Toc85301936)

[2.2.2 Describing the Algorithms 18](#_Toc85301937)

[2.2.3 Usability Features 23](#_Toc85301938)

[2.2.4 Identifying key aspects 23](#_Toc85301939)

[2.3 Approaching the Testing 23](#_Toc85301940)

[2.3.1 The test Data 23](#_Toc85301941)

# Analysis of the problem

### Summary of the project

Poker (Texas Hold’em Poker) is an extremely popular card game. Many casinos feature multiple tables and there are regular large tournaments played with professional players and large prize pools. The game is famous for several reasons: it is one of the few games where players will play and bet against each other, rather than playing against the casino (such as Blackjack). It is also famous because the game is not purely based on luck, nor is it solely based on skill, instead, in order to be good at the game, one must have a good ability to turn their luck (or lack of it) into results.

The game consists of 5 cards which all players ‘share’ (known as ‘community cards’), with each player having 2 cards to themselves (known as ‘hole cards’). The players will then bet against each other as the community cards get revealed, to see who has the best hand, which should be a total of 5 cards with at least one card from your hole cards.

However, I always have thought that learning the game proves extremely difficult, and what is even harder, is getting good at it.

My aim is to create a Texas Hold’em Poker game that will allow inexperienced players to develop their skills and learn the game, while also allowing experienced players to have a challenging opponent. I will do this by creating AI players at various difficulty levels so that stakeholders can choose the experience they desire. I will also be implementing a multiplayer feature that will allow stakeholders to play on multiple machines against each other. Furthermore, a key part of the program will be the Poker Tutor – a feature which can be turned on or off and will guide novice players into making the right decisions by displaying odds and giving tips. Lastly, a tutorial feature will teach the basics of the game to any new player.

## 1.1 Problem Identification

### 1.1.1 Features that make the problem solvable by computational methods

One key feature of the program is the Poker Tutor, which must calculate the probabilities of winning the hand at each turn and display them in a user-friendly way. The Tutor must also use these probabilities to display suggested moves and tips that match any scenario.

In order to achieve this, the program must use all the available data from the current hand and carry out multiple complex calculations. For future moves, the computer has to be almost 3 steps ahead of the player; therefore, the solution requires computational power since these methods and algorithms would be time consuming and labour intensive without the aid of a computer.

Furthermore, the fact that players are able to play on different machines, means that a computational method is required - there is a need to communicate between remote devices. Players are able to play in the same game yet not be in the same place geographically which would be impossible to achieve without using a computational method.

Additionally, tasks in my solution can be decomposed and abstracted. For instance, when looking at the Poker Tutor. This task can be broken into further subtasks: calculating probabilities, suggesting moves, giving appropriate tips; which can then be broken further. Furthermore, the subtasks that have been decomposed can be abstracted to ensure that they will work in any scenario not just a specific one, and so the problem requires computational thinking.

### 1.1.2 Why the problem is amenable to a computational approach

The game will require sequential processing, for instance, at the start of each hand, cards will be dealt, blinds will be placed, and then cards will be turned over in the centre. Iterative processing will also be required, during each betting stage players take turn deciding how much to bet, the process repeats itself until everyone has the same bet. Moreover, during the card dealing, cards will be given to each player repetitively, with each player receiving 2 cards from a randomised deck, alternating who the cards are dealt to. Because of the nature of these processes (require iteration and sequential processing), they lend themselves to a computational approach.

When dealing, cards will be animated from the deck into each players’ hands. Likewise, players betting and folding will need to be correctly displayed and animated. Furthermore, hole cards; chips; pot size and the community cards will all need to be stored throughout the game, appropriately. A need for animation, graphics and data storage makes the problem amenable to a computational approach since these would not be achievable without using a computer.

## 1.2 Stakeholders

### 1.2.1 Identifying the end user

The end-user can be any person who has an interest in poker, wants to learn poker, or who is already experienced at poker. The end user can also be an experienced gamer, a casual gamer, or a new gamer. End users who are new to poker should find that the solution is not too challenging, which might turn them away from the game. Whereas users who have more knowledge on poker should find that the solution is not too basic, to ensure that they do not get disinterested.

All end-users require the solution to be enjoyable, and the game should be relevant to their skill level. The solution should be simple to follow and efficient so that playing does not feel tedious, boring, or repetitive.

New gamers should be able to learn the controls and rules so that they feel comfortable while playing, on the other hand, experienced gamers should be able to bypass tutorial features so they do not get irritated while learning concepts they are already familiar with.

### 1.2.2 Why the solution is appropriate for the end user

The solution is able to satisfy end users of all levels of expertise in poker since there are various features that the end user can alter, so that the solution can be suited to them. For example, someone who is new to the game can learn the rules via the tutorial feature, whereas someone who is experienced is able to play against ‘expert’ level AI. There are also other options to make the solution easier or harder, such as choosing the amount of chips all players start with, or the blind sizes.

The end-user is able to play and compete against virtual AI which gives them the appropriate usage out of the solution since not all stakeholders will have other people to actively play against. Similarly, the solution provides end users with the ability to play poker even without the physical equipment required to play, such as the cards or betting chips. This appeals to the casual gamers, who might not want to invest in the physical equipment or want to spend time completing the tedious task of distributing chips and shuffling cards, but instead can play poker in an instant, using my solution. In addition, end-users can play against other end-users who are on different machines. This is important to the end users since they do not have to be physically together in order to play, instead can play at any place with other end users in separate locations.

The user interface will also be user-friendly and clear so that end users are able to get the most pleasurable and satisfying experience, which is a required specification of the end user. Additionally, stakeholders will be able to play the game with just a few clicks, making the solution efficient; this ensures that the end-user does not find the solution exhausting and laborious to play.

## 1.3 Research

### 1.3.1 Similar problems and solutions

#### 1.3.1.1 Solution 1 – Zynga Poker

Figure

Zynga Poker is one of the biggest free online poker games. Each account has their own set of virtual chips that they can use to enter tournaments, win jackpots, and challenge other players. The game boasts the fact that they offer both casual Texas Hold’em as well as competitive tournaments.

In Figure 1, a player has won with a flush, and pop up text has appeared on the display to show the user what hand has won. In addition the outline of the players profile has changed to indicate they have won. I would like to implement these features into my solution as this makes the solution more accessible to novice players since the display is clear and concise. Each player has their own profile picture, to make each player distinct from each other. This is a feature I would not like to implement since this is outside the scope of my solution, and it is much easier to give each player individual names, and unique pre-selected colours. Furthermore, the avatar on the left in Figure 1 is only included in the advertising, and is not actually included in the game-play.

The game offers a VIP program, a feature which I will not be implementing since it is not within the scope of the solution. The VIP program offers in-game benefits such as ‘exclusive chip package offerings and special poker game modes’. I will not be implementing this feature into my solution since I will only be creating one poker game mode – Texas Hold’em - and players will not have their own accounts with their own chip count. Instead, each game will be treated separately, and any winnings or losses will be kept local to each game. Also, ‘in-game benefits’ is a feature I do not want in my solution since I want to keep the solution fair for all stakeholders and not a ‘pay-to-win’ system.

Furthermore, a key feature of the game is variety – you can play a faster 5 player table or a slower but larger 9 player table. The user can also play competitive tournament style poker with ‘high stakes’ and can compete against millions of other players in large scale leagues. Or the user can opt to have a more casual ‘classic experience’. Zynga Poker also promises ‘games for all experiences and skill levels’[[1]](#footnote-1).

An experience with a large variety will be a key feature of my solution – allowing stakeholders to fully customise their experience – for instance, choosing how many players they would like to play against, the level of AI, the amount of starting chips, the blind sizes, or the ability to play against real players. However, I will not be implementing tournament style games or large-scale leagues as this is outside the scope of my solution. The stakeholders require variety in my solution, to guarantee that the experience is relevant for their skill level and gaming experience.

Lastly, they offer fair play – the experience should feel like you are getting ‘the true Vegas-style game’. This is also a feature I would like to have in my solution as I feel that the user should have an experience as close to possible as a real table experience – so all probabilities and rules must be the same as in real life. Zynga Poker also runs adverts on their game, however, I will not be implementing adverts into my solution since this is outside the scope of my solution, and adverts are aggravating and bothersome to the stakeholders which makes the solution less appealing.

Figure

The actual table is shown in the Figure 2. On each players turn; a timer appears above the players’ profile. If the player does not make a move within this time, they are forced to fold or check depending on whether or not they have matched other players’ bets. This is a feature I would like to implement in my game to ensure the game goes smoothly and quickly. Moreover it makes it evident to the end-user how much time they have to make a move, and does not keep other players waiting too long, which guarantees that stakeholders do not lose their interest.

Figure

In addition, there are icons next to each player, signalling if the player has checked, called, raised, or folded, along with their bet amount. In Figure 3, an icon shows that a player has called (shown by the yellow ‘B’) $11K. These icons are an effective way to display a lot of information in a minimalist and straightforward way. Therefore, I will implement similar icons that will have the same purpose, in order to satisfy the stakeholders need for the solution to be as simple as possible. Novice players may find these icons difficult; however, a tutorial feature will explain these icons and their meanings, to certify that the solution is appropriate for all end-users.

#### 1.3.1.2 Solution 2 – WSOP Poker

Figure

In Figure 3, WSOP Poker’s main menu is shown. Although, I will not be including the assorted styles of tables as shown above, the overall aesthetic and layout of the menu is something I will be implementing into my main menu. This is because the menu is straightforward and obvious, so any new gamer will find it easy navigating around the menus. The menu includes large buttons such as ‘PLAY NOW!’ and the buttons are inside larger blocks which include imagery of the type of table the user will be playing. My menu will also feature blocks of imagery with large buttons for the various actions. WSOP Poker’s menu features a ‘Rules’ section whereas my menu will feature a tutorial section that will include all the basic rules of the game since I would like the solution to be accessible to players of all skill levels.

WSOP Poker’s table is shown in Figure 5. The table shares a lot of features with Zynga Poker’s table: the timers, the placement of hole cards, community cards, dealer, and bet icons. However, there are some distinctive features such as the ‘Fold’ and ‘All-In’ buttons which are located to the left and right if the user’s hole cards. These buttons are large and easy for the user to click and the large space between them creates a clear distinction between the two actions. For these reasons, the buttons in my solution will also have a wide separation and have a generous size, however, my solution should include all actions, for example, check, call or raise.

Another feature of this table is the gift feature – where users can gift any other player a certain amount of chips. This is a feature that I will not be implementing into my solution since this is not a requirement for the end user and is not realistic. The end user should have the most realistic experience while playing, so a gifting feature is not necessary as it does not fulfil this requirement.

WSOP Poker has a player ranking system, a competitive tournament style mode where players can ‘rise through the ranks’. In the Figure 5, players have an icon with a single letter next to their chip count – this is the player rank, a measure of how good a players’ poker skills are. Player ranks are outside the scope of my solution, so I will not be including them. Users will not have their own account and since each game is local (everything that happens within the game, stays within the game) ranks are a feature that would not fit with the overall solution.

In addition, when a player folds their profile picture becomes opaque to signal that that player is no longer playing this hand. This feature is useful and explicitly shows who is currently playing the current hand. Accordingly, I will be implementing this feature into my solution as this fulfils the stakeholders' requirements of having the game be easy to follow, however, since I am not implementing profile pictures, player names will become grey instead of profile pictures.

Figure

#### 1.3.1.3 Solution 3 – Poker heat

Figure

Poker Heat’s table is shown in Figure 5 and shares lots of similarities with the aforementioned solutions. Poker Heat also features a chat system which can be activated using the button in the top right corner. Although this does add some realism (allows users to speak with one another), it is not a necessity to the end user and therefore I will not be implementing it. It is likely that friends who will play my solution will be in the same geographical area and so do not need a chat system.

Poker Heat features buttons in the bottom right for the various actions even when it is not your turn. These are ‘pre-select moves’ – players can select a move before their turn, and when it eventually comes to their turn, the action is done instantaneously. This is a feature I would like to have in the overall solution, as it speeds up the tempo of the game and reduces the amount of work the user must do in order to play the game. This gives the user an overall better experience since they are not waiting around for their turn to come.

Figure

Poker Heat has a bar next to the user’s profile, which provides a quick way for the user to see the likelihood of winning. Located next to this bar, is the user’s best possible hand. These features aid unpractised players in making better decisions, and therefore will be a part of the Poker Tutor. This means experienced player can play without these assists, and novice players can choose to have them on in order to have a more pleasurable experience. The bar is a much simpler way of providing the complex probabilities to a new player, however the tutor should have an option to enable the user to look at the probabilities in detail.

Figure

As shown in Figure 7, Poker Heat offers a collection of ‘rings’ which are rewards for playing the game. In order to earn these rings, the user must complete various missions and tasks given to them each day. Players can then choose to wear these rings on the table to show other players their achievements and poker skill. I have chosen not to implement a rewards system and daily missions since this is outside the scope of my solution.

Lastly, Poker Heat offers daily rewards and free chips that the player can use in their poker games by means of a daily free spin shown in the figure above. In my solution, the player does not have a chip count and each game they are given a fresh set of chips, therefore, the free chip system does not fit with my overall solution and so I have decided not to implement it. Furthermore, the chips you receive are scaled based on your ‘league’ and since I have decided also not to implement a ranking system, a daily rewards system does not suit my solution and is not a requirement for the stakeholders.

Lastly, common complaints about Poker Heat are that the odds are not realistic and there seems to be rigged hands[[2]](#footnote-2). My solution will ensure that all end users feel that the game is completely fair and that all probabilities are true-to-life and realistic. This makes the solution more appealing to stakeholders.

### 1.3.2 Essential features of the proposed solution

A key feature of the solution is the ability to customize the game experience: being able to activate the poker tutor, change AI difficulty, change number of players, starting chip amount and blind sizes. This variety of choices allows the end user to tailor the experience to each individual requirement that they might have – for example make the game more challenging or make the game shorter.

The solution should have an aesthetically pleasing menu that has all the appropriate actions such as create new game, settings, quit and tutorial. Each action should fulfil its purpose and be functional. The menu is the first thing the end user will see and so a good-looking menu will make the overall experience for the stakeholders better. A functional menu ensures the end user does not get frustrated trying to navigate the menu and fulfils the stakeholders’ requirement for my solution to be accessible and a satisfying experience.

The Poker Tutor should display probabilities in a novice friendly way, should suggest appropriate moves based on several variables: player chip amount, other players’ chip amounts, probability of winning, probabilities of getting each hand, and other players’ actions. The Poker Tutor should also give various tips throughout the game. The Poker Tutor is essential to providing a new gamer or a novice poker player a more pleasant encounter with my solution. Inexperienced players are able to learn Poker while also having the opportunity to win, which they might not be able to do, whilst learning poker against skilled opponents with no assists or tips.

Another essential feature of my program is the AI. Stakeholders require my solution to be played without other real players and AI are a perfect solution to this problem. Stakeholders can also customize the difficulty of the AI to suit their own skill level, which ensures that the solution provides the end user with a challenge whilst not being impossible to win.

Each player should have a fixed time limit per turn to ensure that the game goes quickly and there is not a lot of downtime between turns. A timer should be displayed for each player in an obvious way so that the end user knows exactly how much time they have to make a move. Furthermore, being able to pre-select moves is a key feature of the solution as this also minimises the amount of time between each turn where the user is doing nothing, and so therefore fulfils the stakeholder’s requirements of the solution being efficient.

Additionally, stakeholders require the ability to be able to play against other stakeholders. This allows for a more competitive style of poker while also creating a more realistic experience which the end user requires. It also allows stakeholders who may not have the physical requirements to play poker, to virtually play poker in a way that is as close to the physical experience as possible. Furthermore, stakeholders may want to play with less than the minimum number of players required to start a game, so the solution should allow the end user to fill empty spaces with AI of their chosen difficulty through a menu option, as this will make the table appear to be full.

The solution should make Poker easy and accessible to play while also making the end-user always feel occupied. Features such as small icons to signal each players’ move and large pre-select buttons and move select buttons will all fulfil these criteria. Moreover, player names will become opaque once they have folded and once a player has won, the solution should make it clear to end users, who has won; what they have won with (for example Flush or Straight) and the amount they have won. These features mean that novice gamers and inexperienced poker players will have minimal issues when playing my solution.

### 1.3.3 Limitations of the proposed solution

Each game is kept local to each other. This means that any amount of winnings or losses in a game will not be transferred over to the next game. Players also do not have individual accounts where they would be able to have their own chip amount. Because of these reasons, having features such as daily bonuses, free chips, missions, and tasks for the user to complete are all impossible in my solution. However, the Poker Tutor could offer tasks within each game that would help newer players to make better choices, and they could be rewarded within each game. On the other hand, receiving daily free chips, cannot be implemented since each player does not have their own wallet that carries over through each game, and rewards such as rings, that require multiple games to earn are also unable to be implemented.

In addition, the solution will only include one type of poker. This means that the solution has a reduced variety, since stakeholders might want to play a different variation of poker, however the solution only includes Texas Hold’em. Because of this, stakeholders will only have the option to customize their Texas Hold’em experience and not have to chance to experiment with other forms of Poker. The solution will only include one type of Poker since multiple variations is outside the scope of my solution.

Moreover, it is impossible to implement a league or ranking system because no information from previous games will be stored, so the solution will not have any idea on how skilled the end user is. Since the solution will not store any data on each player, a global leader-board is not possible, and so the solution will not be able to compare the skill level of players with other players and so would not be able to fit players into leagues or ranks. The solution could give a rough skill rating at the end of each game; however, this would not carry over to later games.

Furthermore, players are unable to interact with each other within my solution through features such as chatting systems or gifting systems. However, these features will not be a necessity to stakeholders as the majority of end users will be in the same location while playing my solution.

## 1.4 Specifying the proposed solution

### 1.4.1 Solution’s requirements

The solution will require basic hardware inputs: a mouse and a keyboard. This project will only be on PC, so these inputs are necessary in order to play the game since the solution requires the user’s inputs, which will be done by clicking buttons using the mouse, or entering in a username, or numbers (such as betting amount) using the keyboard. The solution will predominantly use the mouse however, most desktops already have these peripherals so I do not foresee any issues with the end-users.

Furthermore, the end user needs a monitor since the solution will display the poker game. In order to play Poker on my solution, the end user will have to see, the cards, bets, where to enter inputs and the menu screen, so the solution requires some type of display. A display is required to display the graphical user interface, which will aid novice users. Furthermore, the solution will be in 1920x1080 as this is the resolution most monitors will be in and is the largest and clearest display that the majority of stakeholders will have, however, the game should scale down if the display is not large enough.

The solution requires the operating system Windows 10, since this is the operating system, I will be developing my solution in and so will not work on other operating systems or previous versions. Windows 10 64-bit requires a 1 GHz processor, 2GB of RAM and 20GB Hard disk space, which are all basic specifications for most desktops, so end users should be able to meet them.

Moreover, Python 3.8.6 will be required as this is the version that I will be developing my solution in. Python requires a 64-bit CPU, with 4 GB RAM and 5 GB of secondary storage, so these are all additional requirements for the end users’ desktop since they will need Python. Furthermore the libraries that the solution will require are Tkinter, Time and Random.

Additionally, more secondary storage will be required for assets that my solution requires and for the solution itself, therefore the solution requires 25.05 GB secondary storage space in total, leaving 50 MB for assets and code. So the total requirements are a 64-bit 1 GHz CPU; 6 GB RAM; 25.05 GB secondary storage; a 1920x1080 display; a mouse and a keyboard.

### 1.4.2 Success criteria

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Criteria** | **Justification** | **How will it be measured** | **Criteria met Y/N** |
| 1 | The solution must have a graphical user interface. | A GUI (Graphical User Interface) is more appealing to the end user than other types of user interfaces such as using command lines. Also a GUI will make the game easy to play for inexperienced users or novice gamers. | Check if there is a GUI or not. |  |
| 2 | The game should allow the player to fold at any time | The program should follow the same rules as real life Poker. | User testing and feedback from experienced gamers. |  |
| 3 | Players should be able to All-In at any point | The program should follow the same rules as real life Poker. | User testing and feedback from experienced gamers. |  |
| 4 | Players should only be allowed to play valid moves – invalid moves should be greyed out. | The program should follow the same rules as real-life Poker – for example, the player should not be able to check if they have not matched the pervious bets. | User testing and feedback from experienced gamers. |  |
| 5 | Blinds should be dealt automatically | Making these processes automatic means that the game feels smoother to the end user and means that they have to do less repetitive tasks so the solution does not feel tedious. | Play a game and check that the blinds are dealt at the start of each hand by the correct players. |  |
| 6 | The dealer button and blinds should rotate after each hand. | The program should follow the same rules as real life Poker. | Play a game and check that the blinds and dealer buttons are rotating as they would in real life. |  |
| 7 | If a player runs out of money they should be out of the game. | The program should follow the same rules as real life Poker. | Play a game and ensure that once a player runs out of money they go out of the game. This can also be checked through user testing and feedback. |  |
| 8 | If every other player has no money, then a winner should be declared and the game ended | The program should follow the same rules as real life Poker. | User testing and feedback. |  |
| 9 | The Poker Tutor should suggest helpful tips and moves. | Novice gamers should find that solution accommodates to their skill level. | User testing – novice gamers’ feedback |  |
| 10 | End users should be able to play against other end users. | End users may need to use my solution to play with others if they do not have the physical means to play poker. | Run the program on multiple machines. Check if they can connect to the same game. |  |
| 11 | End users should have the option to play against AI that matches their skill level. | End users require the solution to be appropriate to their own skill. Multiple AI difficulties fulfils this requirement. | User testing – feedback from players of various skill levels. |  |
| 12 | The menu and game must be aesthetically pleasing. | Experienced gamers will find that if the solution is aesthetically pleasing, they are more likely to keep playing. | User testing and feedback. |  |
| 13 | The Poker Tutor must display odds of winning the hand by using a percentage bar. | Novice gamers would appreciate the complex probabilities to be displayed in a simple and novice-friendly way | User testing and feedback. |  |
| 14 | End users should have the option to turn any assists on or off. | Experienced users will find that assists may take away from the realism and lower the skill required. | Play game multiple times, changing the assists each time. |  |
| 15 | All odds must be realistic and true-to-life. | End users require that the game should not feel fake, which may make the user irritated | Compare odds from real life to odds in the game by playing the game several times and collecting data. |  |
| 16 | Buttons should be large and unambiguous. | Inexperienced users may find it difficult understanding the game, so the solution should be clear to ensure that novice gamers have a pleasurable learning experience. | User testing and feedback. |  |
| 17 | The tutorial must introduce the solution to inexperienced users in a user-friendly way. | End users should be able to learn the controls, rules, and any helpful tips, in one place. | User testing and feedback from novice users. |  |
| 18 | End users should be able to play with different numbers of players or AI. | Customization and variety are key requirements for the end user as they allow the game to feel fresh and new whenever the end user plays. | Testing with AI and users on multiple machines with different starting conditions and variables. |  |
| 19 | End users must have the option to add AI to any multiplayer game | End users may not have enough people to play a multiplayer game, but still want a realistic experience. Options to fill multiplayer games with AI will therefore make the multiplayer feature accessible to more end users. | User testing and feedback. |  |
| 20 | Each player should have a fixed time limit on their turn. | Ensures that if a player goes AFK or simply wants to waste time, it does not ruin other player’s experience. | Whilst playing, do not play a move in your time, and check that the timer runs out. |  |
| 21 | Pre-select buttons should allow end users to select their move before their turn arrives. | End users should feel like they are always playing and not waiting around. Pre-select buttons ensure that the game feels smooth with minimal waiting time | Use pre-select buttons and check if they are functional and fulfil their intended purpose. |  |
| 22 | The end user should be able to dictate how much money they start each game with. | End users can choose how long they would like the game to be by choosing starting money, which is vital for end users who may have little time. | Start multiple games with different variables. Test if these changes fulfil their intended purpose. |  |
| 23 | Players’ names will become greyed out once they have folded. | This feature will make it explicitly clear to all players, who is in the current hand, and therefore is extremely useful for inexperienced players | While in a game, click the fold button and check if the name becomes grey. Check if this happens to other players. |  |
| 24 | If a player does not complete a move within their time, they should be forced a move | If the user runs out of time the program must move onto the next player in order to ensure a smooth experience. Therefore the player who has run out of time must be forced a move by the program. | Whilst playing, do not play a move in your time and see if the game plays a move for you. |  |
| 25 | The Poker Tutor should provide end users with the option to view the complex probabilities. | Slightly more experienced users may want to see more than just a probability bar. The poker tutor should allow these users to see the in-depth probabilities and odds | User testing and feedback. |  |
| 26 | End users should be able to skip the tutorial | Experienced gamers and players who have played the game previously, should not be forced to do a tutorial | While in the tutorial, click the skip button and check if the tutorial is skipped. |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Design of the solution

## 2.1 Decomposition of the Problem

### 2.1.1 Breaking Down the problem

### 2.1.2 Justification

## 2.2 Describing the Solution

### 2.2.1 Explaining the Structure

#### 2.2.1.1 Classes

I will be using Object Orientated Programming because...

##### Player

INSERT IMG

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Purpose** | **Data Type** |
| hand | Stores the players hand in the current round. | Array |
| name | Stores the players name | String |
| balance | Stores the players available chip count that the player can use. | Integer |
| position |  | String |
| ready | Determines whether or not the player is ready to move onto the next hand (they have matched the current bet or have raised it) | Boolean |

|  |  |  |
| --- | --- | --- |
| **Method** | **Purpose** | **Justification** |
| check() |  |  |
| raise(amount) |  |  |
| call() |  |  |
| fold() |  |  |
| bestRank() | Calculates the best possible hand that the player has with the current community cards. |  |

##### AI

The AI class will inherit from the Player class so will share all of its methods and attributes. The class will have additional methods and attributes to define the player as an AI.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Purpose** | **Data type** |
| difficulty |  | Integer |

|  |  |  |
| --- | --- | --- |
| **Method** | **Purpose** | **Justification** |
| check() |  |  |
| raise(amount) |  |  |
| call() |  |  |
| Fold() |  |  |
| bestRank() |  |  |

##### Card

|  |  |  |
| --- | --- | --- |
| **Atrribute** | **Purpose** | **Data Type** |
| suit | Stores the suit of the card | String |
| value | Stores the number value of the card – King becomes 14 | Integer |
| image | Stores the physical image of the card | Image |
| name | Stores the filename of the image of the card. | String |

|  |  |  |
| --- | --- | --- |
| **Method** | **Purpose** | **Justification** |
| flip() | Changes the image to the opposing image. If the back of the card is displayed, the method will change it to the front. |  |

##### Deck

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Purpose** | **Data Type** |
| cards | Stores all the card objects left in the deck that can be dealt. | Array |
| used | Stores all the cards that have already been dealt. These cards will go back into ‘cards’ at the start of each round | Array |

|  |  |  |
| --- | --- | --- |
| **Method** | **Purpose** | **Justification** |
| deal(location,amount) | Deals an amount of cards to the location. Takes in amount (number of cards to be dealt) and location (the array in which the cards should be dealt to) as parameters. |  |
| reset() | Resets the cards to have the original 52 cards. Pops all the cards in ‘used’ and appends them to ‘cards’ |  |

##### Round

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Purpose** | **Data Type** |
| community | Stores all the community cards. | Array |
| pot | Stores the current pot(s). The pot needs to be an array since there can be multiple pots if a player goes ‘All-In’ | Array |
| stage | Stores the current stage the round is at. | String |
| currBet | Stores the highest bet in the current stage. Players must either meet this bet or raise it by at least double. | Integer |
| currPlayer | Stores the index of the current player who is next to play their move in the allPlayers array. | Integer |
| bigBlind | Stores the big blind of the current round. | Integer |

|  |  |  |
| --- | --- | --- |
| **Method** | **Purpose** | **Justification** |
| nextStage() | Checks if everyone is ready to move onto the next stage (have the same bet or everyone has checked). If yes then reset currBet and currPlayer, deal the next card(s) and increment stage. |  |
| calcWinner() | If the stage is at river and nextStage() is true then a winner must be calculate. Calculate each player’s best hand and compare with each other to find the winner. |  |
| newRound() | Once the winner has been decided. A new round needs to be started. All attributes need to be reset including, pot, cards, community and each players’ hand. New cards must also be dealt to every player. |  |

##### Tutor

|  |  |  |
| --- | --- | --- |
| **Method** | **Purpose** | **Data Type** |
| level | Determines how much the tutor should display or calculate. Also increasing difficulty of AI will use higher levels of Tutor in order to determine their move. | Integer |

|  |  |  |
| --- | --- | --- |
| **Method** | **Purpose** | **Justification** |
| calcEV() | Calculates the expected profit the action will make over the longrun. |  |
| calcEquity() | Calculates the percentage of improving the hand and therefore gives an approximate win percentage. |  |
| calcEachRank() | Calculates the percentage that the player will end up with each rank. |  |
| calcOuts() | Calculates the number of cards that make up the specified rank. |  |

#### 2.2.1.2 Files

#### 2.2.1.3 UI DESIGN

##### Title screen

In Figure ,the first iteration of the title screen is shown, the use of a GUI satisfies success criteria 1. All buttons are ordered in larger hexagonal blocks in order to make navigating the menu straightforward and simple, which fulfils success criteria 16. The “PLAY NOW” button is bigger than all other buttons and is centred in the screen to make it stand out to other buttons. The rest of the buttons are split into their respective sections which are titled in order to aid inexperienced gamers with navigation.

After user feedback, I decided to make another iteration. Users suggested that I make a change to the colour scheme and add more imagery – there was nothing on the screen that represented the game that was being played, apart from the word “poker” written once on a button. The colours were too vibrant and were not aesthetically pleasing; instead a calmer colour scheme would be more suited. Moreover the lack of imagery further emphasised the fact that the screen did not represent poker and so would not be suitable for inexperienced users. Furthermore the quit button is ambiguous to new gamers – it is much clearer to have the words quit instead of a red X symbol that could confuse novice gamers.

In Figure , the second iteration is shown. The screen has a background image of poker chips which is a significant improvement to the first iteration since the screen clearly signifies a gambling game. In addition, including the title of the game “Pocket Aces” and the label “Texas Hold’em Poker” next to the play button, make it explicitly clear that the game that the user is playing is poker.

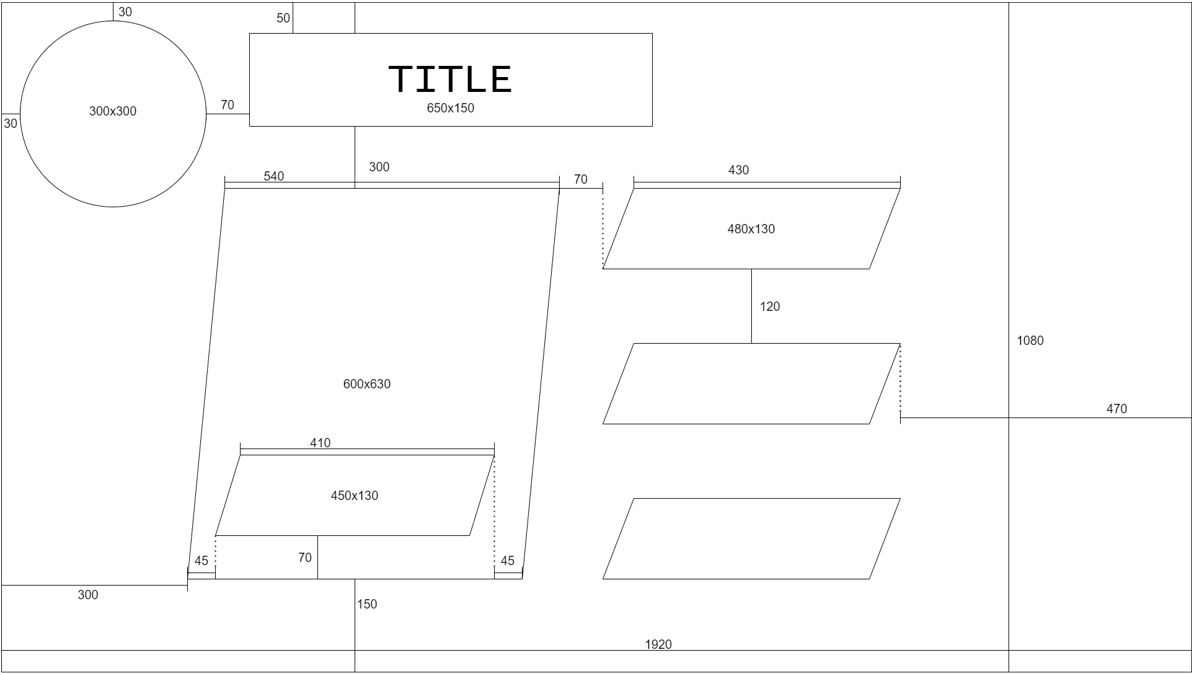
The quit symbol has been replaced with the word “QUIT” to make it clear for inexperienced users, and the colour scheme has been changed to reflect the background image. Also the number of buttons has been reduced for simplicity for the end user and will also make it easier to implement since each button will lead to individual screen.

Users complimented the choice of background image and title – commenting on how the screen made it much clearer that the game was about poker and how combining two sets of buttons into one set made navigation significantly easier. Feedback also suggested that the change of quit button was beneficial and that explicitly putting “Texas Hold’em” eliminated any chance of confusion about which type of poker that users would be able to play. However feedback also suggested that the colour scheme was unappealing and the colours of the buttons and blocks did not match with each other or with the background image. Users appreciated the font choice but did not like the different text colours, commenting on how the text did not stand out against the colours they were on, so did not satisfy success criteria 12. Therefore, I decided upon user feedback that another iteration would be needed that satisfied all appropriate success criteria.

The final iteration is shown in Figure . The background image and title have been kept the same due to positive user feedback. However all the buttons and blocks have been improved to parallelogram shapes and the colour scheme has been changed to contrast the green background which allows the title screen to be significantly more aesthetically enjoyable, which therefore does fulfil success criteria 12.

The “PLAY NOW” button has its own separate block to make it stand out to other buttons and is also in a different colour to all other buttons in order to draw the user’s attention to it. Inside the block is an image of chips and the label “No limit hold’em” which is a much shorter label than previously, so emphasises this iteration’s simplicity and ease of use for novice gamers. I chose to use chips as the image inside of the block, and not a set of cards because of the fact that the logo will contain two ace cards, so having two similar images will look too repetitive.

**Wireframe:**

****

Title relates to poker directly and is also large and central to enforce open communication with the user

Large logo will be located in to top left of the screen - users are likely to immediately read from the top left of the screen.



Image of chips is unambiguous and inexperienced users will be able to pick up

Label allows clear communication with the user,

Separate block for “PLAY NOW” creates visual separation from other buttons.

Darker tone of colour for block and using a white button creates clear distinction

Background represents felt of a poker table with stacks of chips on it so immediately familiarises the user with the game.



Buttons have consistent colour and shape which improves learn-ability for the user.

Teal colour for buttons contrasts the green background well and does not blend into the background image.

Darker colour for block and using a white button creates clear distinction

##### Usability features

### 2.2.2 Describing the Algorithms

#### 2.2.2. Dealing Cards

Pseudocode algorithm:

PUBLIC PROCEDURE deal(amount, location)  
 FOR i FROM 0 TO amount  
 index = random.randint (0, len(self.cards))  
 location.append(self.cards[index])  
 self.used.append(self.cards.pop(index))  
 END FOR  
END PROCEDURE

#### 2.2.2. Moving to the nExt round

Pseudocode algorithm:

PUBLIC FUNCTION nextRound()  
 SET next TO True  
 FOR player IN allPlayers  
 IF NOT player.ready THEN  
 SET next TO False  
 END IF  
 END FOR  
 RETURN next  
END PROCEDURE

#### 2.2.2. BEst Rank

One solution to calculating a hand’s best rank is to check for each rank in descending order of rank strength until one of your checks is true. However, although this method appears to be simpler, it in fact is less efficient than other methods, since the majority of hands will have the best rank of either: pair, two pair or high card (the more common the hand is, the lower down it is in the rankings[[3]](#footnote-3)). This algorithm would have to go through at least 6 checks just to get to two pair.

An alternate solution is to check the ranks in descending order of probability. However this solution would be extremely inefficient since each time the algorithm would run, a check would have to be done on all 9 ranks to be sure that the best rank has been found.

A more efficient algorithm is demonstrated in a flowchart shown in Figure . This algorithm is more efficient since it takes into account that some ranks are better versions of others – for example a four of a kind is just a three of a kind with an extra card of the same value. Therefore instead of having to check each individual rank every time, the algorithm can check a select few ranks and get to the best rank, on average, quicker. With this algorithm, there are only 3 checks required to get to pair, (an extremely common hand: occurs just under 50% of the time), so the majority of the time the new algorithm will be more efficient, since it will undergo less checks.

PUBLIC FUNCTION hasFlush(community):  
 suitCounter = [0,0,0,0]  
 answer = False  
 totalCards = community + self.hand  
 FOR card IN totalCards  
 IF card.suit == “C”  
 suitCounter[0] += 1  
 ELIF card.suit == “D”  
 suitCounter[1] += 1  
 ELIF card.suit == “H”  
 suitCounter[2] += 1  
 ELIF card.suit == “S”  
 suitCounter[3] += 1  
 END IF  
 END FOR  
 FOR suit IN suitCounter  
 IF suit >= 5  
 answer = True  
 ENDIF  
 END FOR  
 RETURN answer  
END FUNCTION

PUBLIC FUNCTION hasSameCard(community, amount)  
 answer = False  
 totalCards = community + self.hand  
 cardsValue = []  
 FOR card in totalCards #creating an array of just values  
 cardsValue.append(card.value)  
 ENDFOR  
 FOR card in cardsValue  
 count = cardsValue.count(card)  
 IF count == amount  
 answer = True  
 ENDIF  
 ENDFOR  
 RETURN answer  
END FUNCTION

PUBLIC FUNCTION hasSraight(community)  
 totalCards = community + self.hand  
 cardsValue = []  
 FOR card in totalCards #creating an array of just values  
 cardsValue.append(card.value)  
 ENDFOR  
 IF 14 in cardsValue  
 cardsValue.append(1)  
 ENDIF  
 answer = False  
 cardsValue.sort()  
 FOR start FROM 0 TO (len(cardsValue)-4) #there are (length -4) combinations of 5 cards  
 straight = True #defines if these 5 cards make a straight  
 FOR index FROM start TO start+4 #repeating for the 4 out of 5 cards  
 IF cardsValue [index+1] -1 <> cardsValue [index]  
 straight = False  
 BREAK  
 ENDIF  
 ENDFOR  
 IF straight #if these cards made a straight  
 answer = True  
 ENDIF  
 END FOR  
 RETURN answer  
END FUNCTION

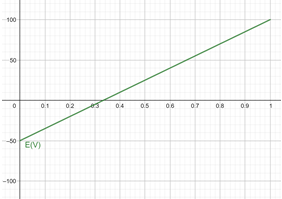
PUBLIC FUNCTION hasFullHouse(community)  
 answer = False  
 IF hasSameCard(community,3) AND hasSameCard(community,2)  
 answer = True  
 ENDIF  
 RETURN answer  
END FUNCTION

PUBLIC FUNCTION hasTwoPair(community)  
 answer = False  
 totalCards = community + self.hand  
 cardsValue = []  
 FOR card in totalCards #creating an array of just values  
 cardsValue.append(card.value)  
 ENDFOR  
 FOR card in cardsValue  
 count = cardsValue.count(card)  
 IF count == 2  
 amount += 1  
 ENDIF  
 ENDFOR  
 IF amount >= 2:  
 answer = True  
 RETURN answer  
END FUNCTION

#### 2.2.2. EXPECTED VALUE

Expected value is the profit you expect to get in the long run for a certain action (for example call or raise). To calculate the expected value you must use the formula:

**C:\Users\avishnoi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\49E1658F.tmp**Where P is the pot size, W is the equity and M is the amount to call. The reciprocal of the x-intercept is the pot odds.

****

Figure

In Figure 9, the graph of expected value (EV) is shown with the pot size of 100 and the amount to call is 50. The graph shows a linear relationship between EV and equity. For low equity, the EV is negative and therefore you are expected to make a loss if you call in the long run. For higher values of equity the EV is positive and so you are expected to make a profit. The x-intercept shows the minimum equity needed for the action to be profitable and the reciprocal of this equity is known as the ‘pot odds’ which is a ratio of the amount to call to the pot size.

EV will be a key statistic that the Poker Tutor will display. A significant part of Poker is consistently making +EV plays. If the Poker Tutor displays the EV, it will allow inexperienced players to get better at the game since they are making better decisions. Also the Poker Tutor should be able to suggest tips (SC ) and the EV will be a statistic that the Poker Tutor can use to suggest an appropriate move based on whether the play is +EV or –EV.

**Pseudocode algorithm:**

PUBLIC FUNCTION calcEV(potSize, equity, amCall)  
 EV = ((potSize+amCall)\*equity) – amCall

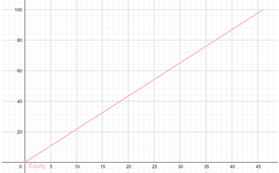
potOdds = potSize/amCall   
 return EV, potOdds  
END FUNCTION

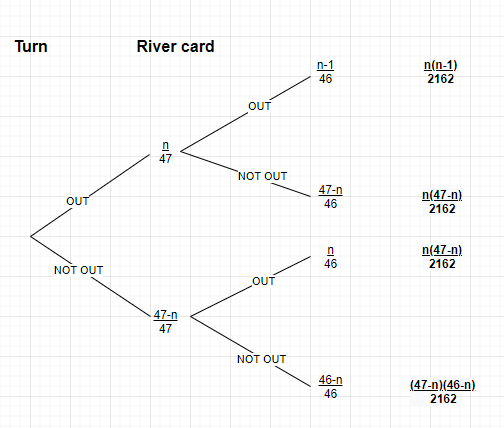
#### 2.2.2. Equity

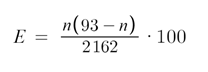
 Equity is the approximate win percentage at a given stage and is based on the number of ‘outs’ you have. ‘Outs’ refers to any card that significantly improves your hand or completes a draw. For example needing a King to complete a straight draw would give you 4 outs since there are 4 kings in the deck.

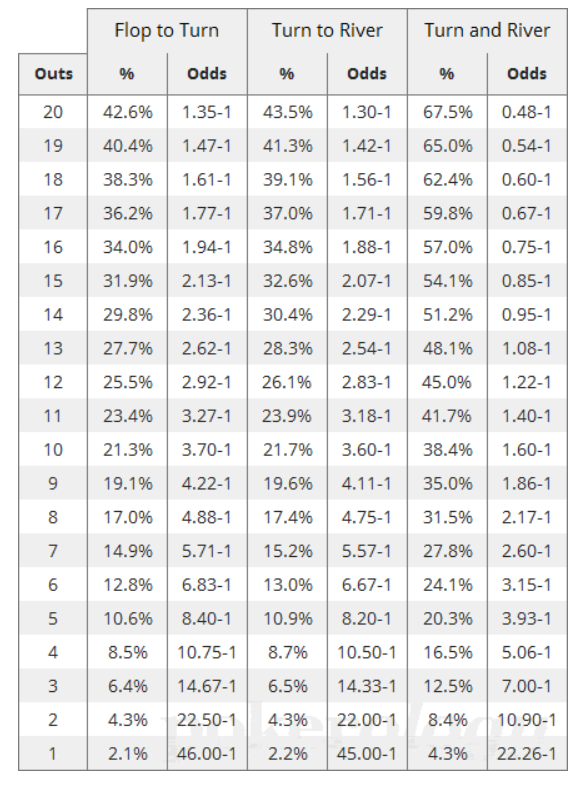
On turn to river , the Equity is just the number of outs divided by 46 since you can assume there are 46 cards left in the deck (4 community cards, and 2 hole cards). When plotted on a graph the number of outs is proportional to the equity as shown in Figure.

C:\Users\avishnoi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\9805C4B.tmp

Where E is the equity and n is the number of outs.

However on flop to turn, the equity is more complicated since there are two more cards to come. A helpful demonstration of this problem is a probability tree (shown in Figure ), that shows the probability of getting n, amount of outs on the turn card and river card.

In Figure , these probabilities are shown in a table. The column Flop to Turn shows the probabilities of getting n amount of outs on the turn which is calculated by n/47. Similarly Turn to River is calculated by dividing the number of outs by 46. Turn and River is calculated by finding the probability of getting no outs on the river and no outs on the turn, then subtracting this probability from 1 to get the probability that you do get an out. Simplifying this, gives the formula for achieving at least one out on the turn or river:

Where E is the equity and n is the number of outs

#### 2.2.2 Each rank

The Poker tutor must be able to display the probabilities of achieving each rank. The

|  |  |  |
| --- | --- | --- |
| **Formula** | **Graph** | **Purpose** |
| **C:\Users\avishnoi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\3E09C0C7.tmp**      E – Equity (y-axis)  n – Number of outs (x-axis) | **C:\Users\avishnoi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\82592D6D.tmp** | Calculates the equity for when outs are required on both the turn and the river. For example, needing 2 hearts to complete a flush draw. |
| **C:\Users\avishnoi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\D4238F03.tmp**  c    E – Equity (y-axis)  n – Number of outs (x-axis) | **C:\Users\avishnoi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\609FE309.tmp** | Calculates the equity for when at least one out is required. For example needing at least on club on a flush draw. |
| **C:\Users\avishnoi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B83582FF.tmp**        E – Equity (y-axis)  n – Number of outs (x-axis) | **C:\Users\avishnoi\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\67488465.tmp** | Exactly one out |
|  |  |  |

### 2.2.4 Identifying key aspects

#### 2.2.4.1 Key variables

## 2.3 Approaching the Testing

### 2.3.1 The test Data

1. Source: https://play.google.com/store/apps/details?id=com.zynga.livepoker [↑](#footnote-ref-1)
2. Source: <https://play.google.com/store/apps/details?id=com.bigblueparrot.pokerfriends> – Reviews section [↑](#footnote-ref-2)
3. Source: https://en.wikipedia.org/wiki/Poker\_probability [↑](#footnote-ref-3)