Advanced Blackjack Strategy
Software Requirements Specification

5-31-2021 - v2.0
Yushu Chen(yc)
Connor Finch (cmf)
Eric Hsu(eh)
Edison Mielke(enm)

Table of Contents

1. SRS Revision History

Date	Author	Description
5-10-2021	enm	Created Initial Document
5-11-2021	enm	Completed 2.1.
5-11-2021	enm	Completed 2.2.
5-11-2021	enm	Completed 2.3.
5-11-2021	enm	Completed 2.4.
5-11-2021	enm	Completed 2.5.
5-11-2021	enm	Completed 2.6.
5-12-2021	enm	Completed 3.5.
5-12-2021	enm	Completed 3.4.
5-12-2021	enm	Completed 3.3.
5-12-2021	enm	Completed 3.2.
5-12-2021	enm	Completed the First SRS Draft
5-12-2021	eh	Edited 3.1.
5-12-2021	enm	Elaborated on 2.1.
5-29-2021	cmf	Revised 2.1. but citations that cite the odds are still needed.
5-29-2021	cmf	Revised 2.2. but still needs more information before submitting.
5-29-2021	cmf	Deleted 2.3. and rewrote from scratch.
5-29-2021	cmf	Deleted 2.4. and rewrote from scratch but still needs more information added to it.
5-29-2021	cmf	Rewrote 2.5. but still needs work.
5-29-2021	cmf	Rewrote every use case from section 2.6.
5-30-2021	cmf	Rewrote Performance Requirements from section 3.4.
5-30-2021	cmf	Added to Software Attributes from section 3.5.
5-30-2021	cmf	Revised and fixed descriptions/definitions of functions from section 3.2.
5-31-2021	cmf	Ended up deleting section 3.2 and writing it again.
5-31-2021	cmf	Finished added sources to references as well as their in-text citations.
5-31-2021	cmf	Rewrote section 3.1.
5-31-2021	cmf	Fix TOC page numbers.

2. The Concept of Operations (ConOps)

2.1. Current System or Situation

Blackjack is a popular game at casinos where one tries to beat the dealer by producing a hand with a higher total than the dealer's without going over 21. Aces are unique because they can have a value of one or eleven points depending on the player's choosing. The value of each numbered card is equal to their rank while all face cards have a value of ten points each. Even though there is truly no such thing as a fair casino game, blackjack is commonly understood as one of the fairest games that can be played at casinos with a house's edge only being approximately 8% if an induvial is playing naively. However, a person who uses the "…blackjack basic strategy, [is] able to lower this down to as little as 0.2-0.5%, depending on the rules at the table" (mrgreen.com, 2021).

2.2. Justification for a New System

Due to blackjack being a game, there should be a way for an individual to be able to improve at playing it casually with friends or more seriously at casinos. The goal of this new system is to provide an interactive way to teach someone how to count cards and use basic strategy to heighten their odds of winning as much as possible.

2.3. Operational Features of the Proposed System

The Advanced Blackjack Strategy System will provide a simple and practical way for an individual to quickly learn strategies such as card counting and the basic strategy to excel their skills while they play blackjack. The system itself will be suitable for anyone who wants to improve their strategies regardless of the skill level they are at. Features such as the ability to change the deck size and card counting method, asks for hints, and practice game modes will encapsulate a wide range of users.

2.4. User Classes

This system only requires two user classes to properly function. The first class is the player who can control every move they make, whereas the second class is the dealer which is controlled entirely by the system. The main user class of this system is the player because the goal of the system is to help them become better at blackjack.

2.5. Modes of Operation

This system provides the player class with two modes of operation. The first mode is the blackjack game itself where an individual can test their skills but still have the option to receive hints if they forget some of the strategies that they practiced. The second mode is blackjack minigame that specifically aims to target the needs of beginner users who are unfamiliar with the basic strategy.

2.6. Use Cases

Use Case: An individual chooses to play blackjack games.

Brief description: This use case describes how the user can interact with the system to play realistic blackjack games like ones played with friends or in casinos.

Actors: User and Simulations module

Preconditions:

1. The individual has successfully entered "1" into the terminal while at the main menu to select the blackjack game module option which will bring them to the blackjack game interface.

Steps to Complete the Task:

- The user is shown their current balance and asked to wager an integer amount of money before being dealt their cards.
- If their wager was successful, they will be able to see the cards in their hand and one of the two cards in the dealer's hand. From there, the user is prompted to enter command based on what they decide to do next. There are two types of commands where one is for hints and the other are for blackjack moves. The hint commands include basic strategy and card count, and the move commands include hit, stand, double, and split.
    - If the user successfully chooses to hit, they are dealt a new card to their hand and given the same prompt as before if their hand value is still less than 21.
    - If the user successfully chooses to stand, the user's turn ends, and the dealer module begins.
    - If the user successfully chooses to double, their bet is doubled and is dealt one last card to their hand before the dealer goes.
    - If the user successfully chooses to split, their two cards are split into two separate hands. From there, they will play out each hand normally as they did before.
    - If the user successfully chooses the basic strategy hint, the system reveals the move they should play according to the basic strategy.
    - If the user successfully chooses the card counting hint, the system reveals data pertaining to the card count such as the true count, running count, and a simple interpretation of the counts.
- After the user is finished, it is the dealer's turn to add cards to their hand. The dealer will keep adding cards to their hand as long as their hand value is less than 17.

- Once the user and dealer have finished, the values of the user's and dealer's hand are compared and displayed on screen with the outcome of the game.

Postconditions:

- The user is prompted with the choice to play again.
- If they choose to play again, they will be able to see their updated balance based on the result of the round of blackjack they just played.

Use Case: An individual wants to practice their ability to remember the basic strategy.

Brief description: This use case describes how a user can practice the basic strategy by selecting the Basic Strategy Practice module.

Actors: User and Simulations module

Preconditions:

- The individual has successfully entered "2" into the terminal while at the main menu to select the Basic Strategy Practice module option which will bring them to blackjack minigame.
- .

Steps to Complete the Task:

- The user is shown the cards in their hand and one of the two cards in the dealer's hand.
- From the information that is being displayed, the user must choose the correct move based on what they remember from the basic strategy.
- After the user enters one of the four possible moves, the system will either confirm that the user's choice was correct or reveal the correct answer if their choice was wrong.

Postconditions:

- The user is prompted with the option to play again.
- The user's balance will be updated based on the most correct answers they got in a row. This high score will be multiplied by 100 and added to their balance. The reason why this was implemented was to incentivize practice and to give a way for beginners to earn more money with they ran out by making too many mistakes in the blackjack game module.

Use Case: An individual wants to practice counting cards.

Brief description: This use case describes how a user can practice counting cards and compare their calculations to those produced by the system.

Actors: User and Simulations module

Preconditions:

- The user is playing blackjack games in the module described in the first use case.

Steps to Complete the Task:

- The default card counting method this system uses is the Hi-Lo strategy because it "…is probably the most used counting strategy in existence. Most simulations and studies are based on this count" (Wattenberger, 2008).

- In order to use this feature effectively, the user should attempt to count cards are their own for as long as possible. Once they are ready, the user can compare their card count with the count calculated by the system.
- The user can choose to reveal a card count hint (described in the first use case) which will display the running count (calculated directly from the counting strategy), true count (calculated by dividing the running count by the approximate number of decks left in the shoe), and an interpretation of what the two different counts mean.

Postconditions:

- Since the user knows the card count, they can take into consideration the basic strategy and card count to determine an even more accurate move choice than they could before.

3. Specific Requirements

3.1. External Interfaces (Inputs and Outputs)

Main Menu:

Input:

- Main menu.
- The purpose of this interface is to let the user decide how they wish to continue with the system.
- The source of the input will come from the user.
- Valid input occurs when the user types one of the four numbers that correspond to the options shown on screen.
- The units of measure come from the user by utilizing Python's input() function.
- The system will store this input as a string to be compared to the options that were displayed to the user in order have the system continue beyond the main menu.

Output:

- Error messages.
- The purpose of the error messages that are outputted while on the main menu are to help the user understand the options available to them.
- The output is produced by the system and displayed on screen for the user to see.
- Upon invalid input, the system will notify the user and ask them to provide a valid input.
- The units of measure come from Python's print() function.
- The outputs will be displayed as strings onto the terminal.

Blackjack:

Input

- Complete Blackjack Game.
- The purpose of this interface is to provide a way for the user to play blackjack by entering a move of their choosing into the terminal.
- The source of this input comes from the user.

- The valid ranges of user input are the different commands offered by the system which include basic strategy, card count, hit, stand, double, and split.
- The units of measure are the commands that a user can enter which is handled by the Player class in participant.py.
- The data format will be strings that represent the cards and commands available to the user for the current round being played.

Output

- Game Results.
- The reason why the game results are outputted is to let the user see the dealer's hand and how much money they won or loss.
- The output is calculated by the system (depending on the outcome of the game) and is then displayed onto the terminal for the user to see.
- The valid ranges of output include the player winning, losing, or tying with the dealer and amount of money depending on those results.
- The units of measure are the values of the player's and dealer's hand that will be compared to one another. The values themselves are calculated using the rules of blackjack in the getValue() function from the Hand class in hand.py.
- The data format of the outputted information will be Python strings.

Minigame:

Input

- Basic Strategy Minigame.
- The purpose of this interface is to provide the user with a more beginner skill level game that aims to help the user improve their memory of the basic strategy.
- The input is nearly the same as the one needed by the Blackjack Game interface which also comes from the user.
- The valid range of inputs only include blackjack game moves which are hit, stand, double, and split.
- The unit of measure is the blackjack move that the user inputted.
- The data formats used will Python strings.

Output

- Minigame Results.
- The purpose of the output is to inform whether the user selected the correct move according to the basic strategy.
- The source of the output is generated by the system after it checks the user's move choice.
- The valid range of output either confirms the user's move choice was correct, or it informs them of the move they should of chose.
- The units of measure required by the system to properly output the correct result are three constant Python dictionaries which store the proper move according to the basic strategy. Three dictionaries are need because there are three different types of hands that can change meaning of the hand's value. The first dictionary is

used when the user's hand does not contain an ace, the second is used when the user's hand does contain an ace, and the third is used when the user's hand can be split.
- The data form of information outputted to the screen will be Python strings.

3.2. Functions

Main Menu:

- Validity Checks
  - The system checks to see if the user entered one of the four options displayed on screen.
- Sequence of Operations
  - The system waits for a user to input an option.
  - Upon receiving input, the system attempts to validate it to see if the input was one of the four options offered.
- Error Handling
  - When a user enters an invalid input, the system displays an error message that informs the user of possible valid inputs that can be chosen from.
  - After reading the message, the user can try again after pressing any key to continue.
- Input/Output Relationship
  - Upon entering an input that causes an error, an output of the error is display on screen for the user.

Blackjack Game:

- Validity Checks
  - Betting
    - The system checks to see if the user enters a bet amount that is not larger than their balance and is a positive integer.
  - Commands
    - Hit: A user requests for another card to be dealt to their hand. This command is always valid unless the user chose the split command on a pair of aces. In this case, a user can only hit once. The reason why this rule exists is because "…it would drop the house edge by 0.19%" (Blackjack Online, 2021) giving the user the advantage.
    - Stand: A user does not wish to add any more cards to their hand and ends their turn.
    - Double: A user requests to add one more card and one more only to their hand and doubles their initial bet. A user can only double after receiving their opening hand (two cards); however, a user can choose double instead of hitting after splitting a pair of aces.

        - Split: A user requests to split their opening hand into two separate hands; however, the two cards in their hand must be the same rank. In addition, a have that has already been split cannot be split again.
        - Basic Strategy: This command is not a blackjack move but is implemented into the system to provide a way for the user to be told what the best blackjack move is according to the strategy.
        - Card Count: This command is not a blackjack move but is implemented into the system to provide a way for a user to compare the card count that they have been keeping track are their own with the one computed by the system.
- Input Processing Sequence
    - A blackjack starts off by the system requesting the user to provide an amount of money they wish to bet.
    - The system deals out two cards to both the user and dealer which is where it waits for the user to enter a command.
    - The result of the game is displayed on screen, and the system asks the user if they wish to play again.
- Error Handling
    - Betting
        - If the user enters a non-integer bet, the system informs them of the error and asks them to try again.
        - If the user enters a negative integer or an integer that would result in a negative balance, the system informs them of the error and asks them to try again.
    - Commands
        - If the user fails to enter one of the listed commands on screen, the system informs the user that the command that they entered was invalid and to try again.
        - If the user enters a valid command that would break the rules of blackjack, the system informs them of the rule that they would break and asks them to try entering a different command.
- Input/Output Conversion
    - If the user enters an input that results in an error, the system outputs a message based on the error that occurred.
    - After the results of a game have been calculated, they are displayed on screen where a user can see the dealer's hand and if they won, lost, or tied.

Basic Strategy Minigame

- Validity Checks
    - Blackjack Move Commands: These are the same commands described in the Blackjack Game; therefore, they will have the exact same characteristics.
- Input Processing Sequence
    - The system deals out two cards to both the user and dealer and waits for the user to enter a Blackjack Move Command.

        - After the user has entered a move, the system compares it with the basic strategy and informs the user did or did not match. It then gives the option for a user to input if they wish to play again.
    - Error Handling
        - In this minigame, there will be no such thing as an invalid move because the system simply checks to see if the user move matches the move from the basic strategy.
    - Input/Output Conversion
        - If the user enters the correct move, the system informs them by outputting a message to the screen that says they were correct.
        - If the user enters an incorrect move, the system will inform the user of the correct move that they should have typed instead.

3.3. Usability Requirements

This will be organized to have few, very noticeable, descriptive buttons that will limit ambiguity for the user. The user should be able to simply learn how the program works by looking at it once without issue. This is essential as many people just want a simple drop in and drop out program, a simple program without "need" of a “README” file or a manual is ideal for this case.

3.4. Performance Requirements

Static Requirements:

- This system performs calculations on many types of data which include:
    - Move choice.
    - Card and Deck classes.
    - Hand value where a hand is a list of cards that a user or dealer owns.
    - Card count.

Dynamic:

- The computational time it would take to retrieve the correct move according to the basic strategy should be no longer than one millisecond because the basic strategy is constant and store in a dictionary.
- The computational time it would take to retrieve the correct card count should also be no longer than one millisecond because the information is computed when cards are dealt to the dealer and player, so the time the player requests to see the card count it would have already been computed.

3.5. Software System Attributes

The Advanced Blackjack Strategy program is designed with reliability and performance in mind, in order to stay as stable as possible we will build this to be lightweight and have as few moving parts as possible, which will additionally play a hand in performance. Performance is important to this program as a slow program will be unintuitive for users and may even be frustrating. Additionally, this system is extremely portable because every module and import used in this project was design by the developers who worked on this project. This also means that the system will be incredibly easy to maintain since there are no outside libraries or modules needed by the system to function properly.

4. References

Blackjack Odds & Probability Explained: Mr Green Casino. Mr Green. (n.d.). https://www.mrgreen.com/en/blackjack/strategies/blackjack-odds.

Faulk, Stuart. (2013). *Understanding Software Requirements*. https://projects.cecs.pdx.edu/attachments/download/904/Faulk_SoftwareRequirements_v4.pdf

IEEE Std 1362-1998 (R2007). (2007). IEEE Guide for Information Technology–System Definition–Concept of Operations (ConOps) Document. https://ieeexplore.ieee.org/document/761853

IEEE Std 830-1998. (2007). IEEE Recommended Practice for Software Requirements Specifications. https://ieeexplore.ieee.org/document/720574

ISO/IEC/IEEE Intl Std 29148:2011. (2011). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/6146379

ISO/IEC/IEEE Intl Std 29148:2018. (2018). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/8559686

Oracle. (2007). White Paper on “Getting Started With Use Case Modeling”. Available at: https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf

The Rules Behind Splitting Aces. Blackjack Online. (n.d.). https://www.blackjackonline.net/guide/the-rules-behind-splitting-aces/.

van Vliet, Hans. (2008). *Software Engineering: Principles and Practice*, 3nd edition, John Wiley & Sons.

Wattenberger, N. (n.d.). Card Counting. Hi-Low – Card Counting Strategy. https://www.qfit.com/cardcounting/Hi-Lo/.

5. Acknowledgements

This is based off the SRS template made by Anthony Hornof for use in his CIS 422 class.