**Problem Statement:** People are always looking for things to do in their spare time. One thing that people like to do is play card games, whether it be with a physical card deck or through the medium of a computer.

**Background Information:** Boredom is a heavy topic to address. We will create a blackjack game with for entertainment purposes to help solve the problem of boredom.

**Environment:** Our environment is the Java Runtime Environment, running on Desktop

**Functional Requirements:**

\*\*\* = time permitting

|  |  |  |
| --- | --- | --- |
| Backlog ID | Functional Requirements | Hours |
| 1 | As a user I would like to be dealt a set of cards for a blackjack game | 2 |
| 2 | As a user I would like to view my own cards | 2 |
| 3 | As a user I would like to be able to “hit” (get another card) | 2 |
| 4 | As a user I would like to be able to view opponent’s face up card(s) | 2 |
| 5 | As a user I would like to be able to “split” my cards if they are the same | 4 |
| 6 | As a user I would like to be able to play against an AI | 8 |
| 7 | As a user I would like to be to win given the win condition of blackjack | 4 |
| 8 | As a user I would like to play against multiple levels of AI\*\*\* | 6 |
| 9 | As a user I would like to be able to play against other human players\*\*\* | 10 |
| 10 | As a user I would like to bet chips against my hand\*\*\* | 2 |
| 11 | As a user I would like to play over an internet connected network\*\*\* | 8 |
| 12 | As a user I would like to be able to save my current amount of chips and return to the game at a later time\*\*\* | 2 |
| 13 | As a user I would like to see card dealing and playing animations\*\*\* | 6 |
| - | Total: | 54 |

**Non-Functional Requirements:**

* Files storing confidential user information, such as number of chips, will be stored securely, not as plain text.
* The application will have a simple and easy to use GUI\*\*\*.

**Use Cases:**

|  |  |
| --- | --- |
| **Case: As a user, I would like to be dealt a set of cards.** | **System Responses** |
| 1. Create 52 instances of a card object | 2. 52 object instances |
| 3. Add all card objects to a list | 4. An arraylist |
| 5. Shuffle the list | 6. A shuffle call |
| 7. Deal two cards to each player | 8. Two object assignments per player |

|  |  |
| --- | --- |
| **Case: As a user I would like to view my own cards** | **System Responses** |
| 1. Retrieve card information from player object | 2. One object call per card per user |
| 3. Print user’s cards | 4. ASCII print method, multiple print calls |

|  |  |
| --- | --- |
| **Case: As a user I would like to “hit” (get another card** | **System Responses** |
| 1. Get the top card of the deck | 2. Get call |
| 3. Add that top card to the player’s hand | 4. Insert the card object into the user’s hand |

|  |  |
| --- | --- |
| **Case: As a user I would like to be able to view opponent’s face up card(s)** | **System Responses** |
| 1. Retrieve opponent user cards | 2. Object calls, one per card per user |
| 3. Display all cards except the first two | 4. ASCII print method, multiple print calls |

|  |  |
| --- | --- |
| **Case: As a user I would like to be able to “split” my cards if they are the same** | **System Responses** |
| 1. Choose option to “split” if cards are the same value | 2. Verifies cards are of the same value |
|  | 3. Creates two separate playing hand entities for the player who split |
|  | 4. For remainder of hand allow player to “hit” for their choice of two hands |

|  |  |
| --- | --- |
| **Case: As a user I would like to be to win given the win condition of blackjack** | **System Responses** |
| 1. Either the Player or the Dealer reaches 21 or above | 2. Player (or dealer) with appropriate cards wins the game |
|  | 3. Winner receives chip pot |

|  |  |
| --- | --- |
| **Case: As a user I would like to play against other human players** | **System Responses** |
| 1. Multiple users connect to localhost | 2. Localhost server is initialized |
| 3. 1 player at a time takes their turn | 4. Hands/chips are updated accordingly |
| 5. Turn shifts to next player | 6. Active player object is updated |

|  |  |
| --- | --- |
| **Case: As a user I would like to play against an AI** | **System Responses** |
| 1. User selects Single-player | 2. Print next menu |
| 3. User selects number of AI | 4. Instance selected number of AI players |

|  |  |
| --- | --- |
| **Case: As a user I would like to play against multiple levels of AI (if time permits)** | **System Responses** |
| 1. Go to main menu/options after game initialization | 2. Present “Choose Difficulty” setting |
| 3. Click on “Choose Difficulty” | 4. Present Easy, Medium, Hard options |
| 5. Click on desired difficulty | 6. Load appropriate AI given difficulty selection |

|  |  |
| --- | --- |
| **Case: As a user I would like to be able to save my current amount of chips and return to the game at a later time\*\*\*** | **System Responses** |
| 1. Do ‘Save Chips’ command/button | 2. Serializes player object, saves file |
| 3. Upon returning, do ‘Load Chips’ command/button | 4. Opens appropriate file and creates user object from that |

|  |  |
| --- | --- |
| **Case: As a user I would like to see card dealing and playing animations** | **System Responses** |
| 1. User interacts with menu and game | 2. Appropriate animations are played |

|  |  |
| --- | --- |
| **Case: As a user I would like to bet chips against my hand** | **System Responses** |
| 1. After receiving dealt cards, click option to bet | 2. Display betting interface |
| 3. Enter amount of chips to bet | 4. Accept bet and add to pool of chips |
|  | 5. Distribute chips to to winner |

|  |  |
| --- | --- |
| **Case: As a user I would like to play over an internet connected network\*\*\*** | **System Responses** |
| 1. Choose ‘Online Game’ | 2. Open connection to server on appropriate port |
|  | 3. Match player up with other players on server |
|  | 4. Proceed as usual with turn rotations |