BLACKJACK CUI GAME

PFDS PROJECT REPORT

Submitted in partial fulfillment of the requirement of the degree of

BACHELORS OF TECHNOLOGY

to

The NorthCap University

by

Piyush Gambhir – 21CSU349

Harsh Pratap – 21CSU311

Under the supervision of

Dr. Srishti Sharma

Department of Computer Science and Engineering

Department of Computer Science and Engineering

School of Engineering and Technology

The NorthCap University, Gurugram- 122001, India

Session 2022-2023

INDEX

Motivation

The main motivation behind this project was a personal interest in the game of Blackjack and other casino card games. A further motivation was the ability to develop a software tool which can be used to help others learn how to play in a way which was not based around simply reading rules and then guesswork.

Problem Statement

Create a Blackjack CUI game for the player, keeping all the rules in mind. The dealer would be a computer and the choice of deck could be one, two or maximum three. If you have good probability, you may apply that.

Introduction

We have created a game called “Blackjack” by using character user interface (CUI) with some specific rules. Here, by default the dealer would be PC and the choice of deck given to the user would be one, two or maximum three. We have imported “Random, OS and Time library and have used various functions for smooth running of the game.

Technology Stack Used

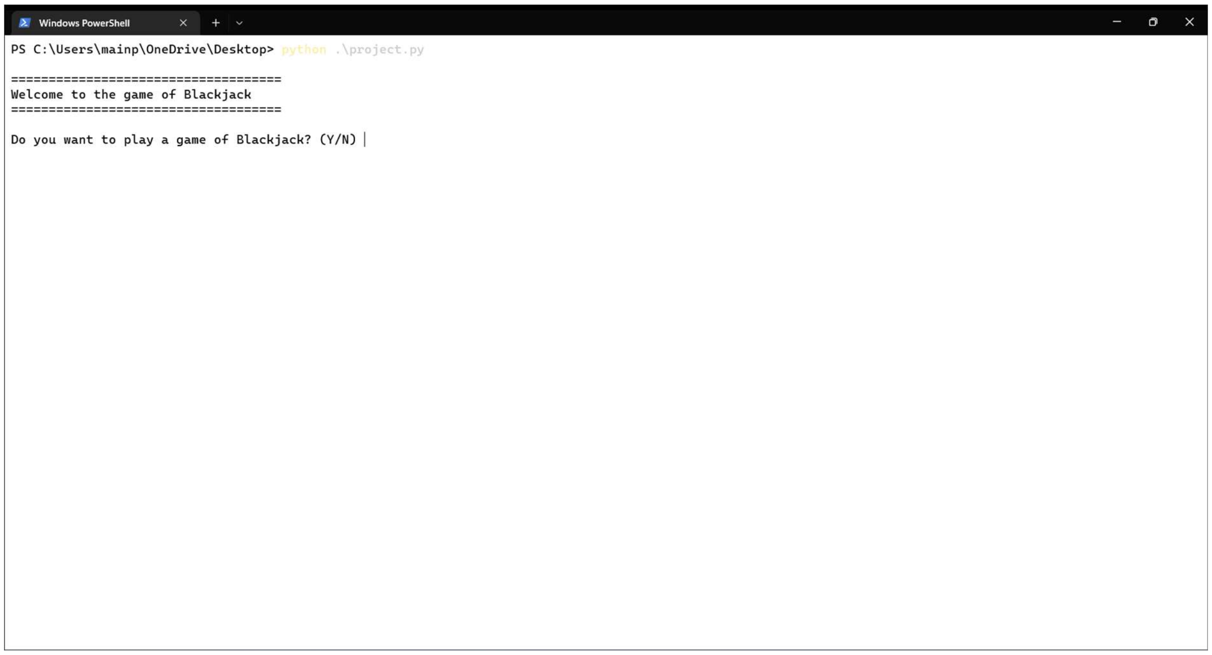
1. Python 3.10
2. OS Module
Random Module

CUI – Character User Interface

- CUI is a way for users to interact with computers and it uses only alphanumeric characters and pseudo graphics for input-output and presentation of information.
- Its biggest advantage is that enables the users to use the system efficiently and quickly.

Game Flow

You will get a player card first after executing the code.

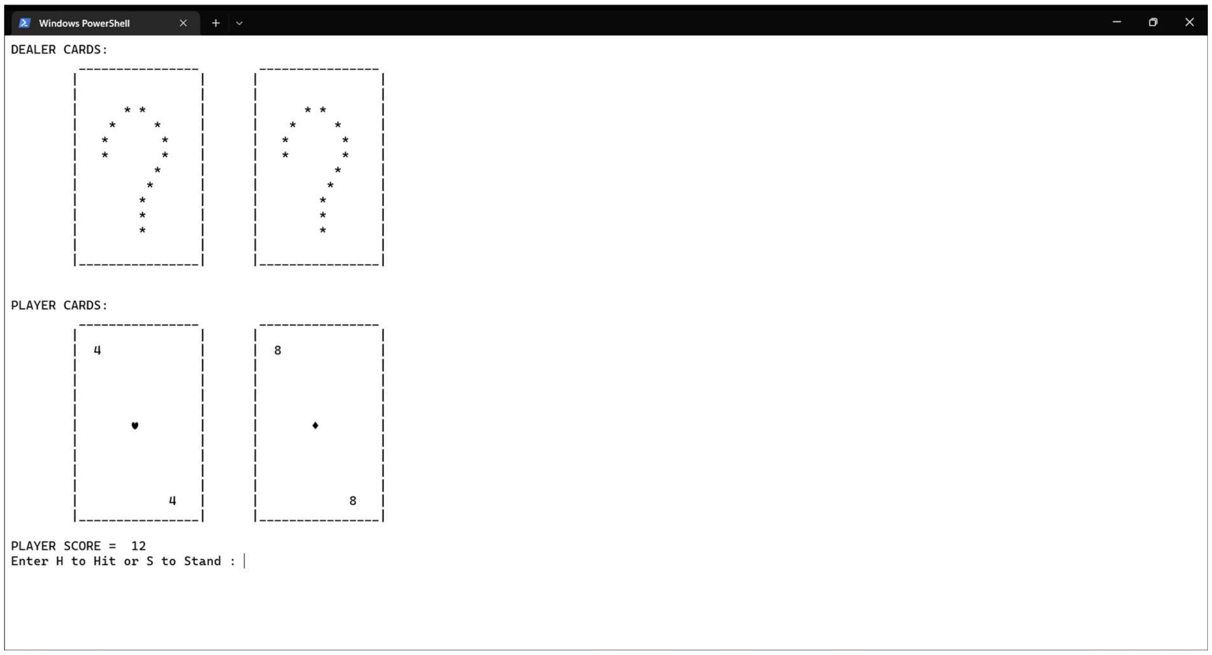


```
PS C:\Users\mainp\OneDrive\Desktop> python .\project.py

====================================
Welcome to the game of Blackjack
====================================

Do you want to play a game of Blackjack? (Y/N)
```

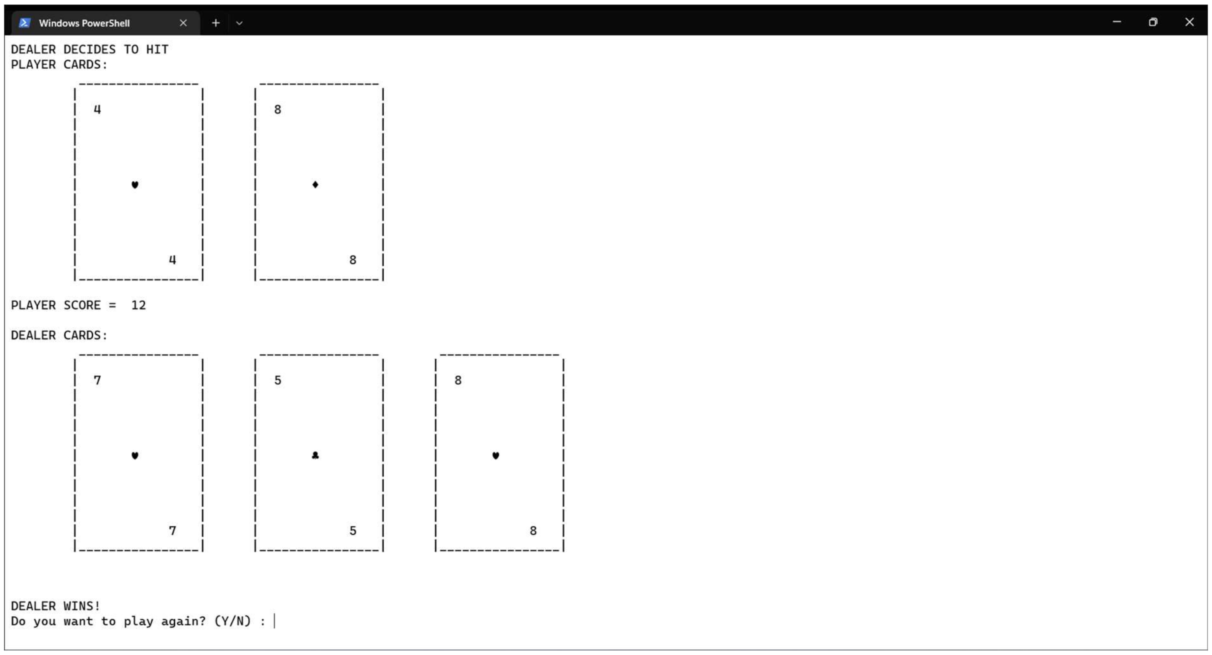
As per the game blackjack, player gets an optionto Hit(H) or Stand(S) with the available cards



```
DEALER CARDS:
         ----------------       ----------------
        |                |     |                |
        |                |     |                |
        |      * *       |     |      * *       |
        |    *     *     |     |    *     *     |
        |   *       *    |     |   *       *    |
        |   *       *    |     |   *       *    |
        |          *     |     |          *     |
        |         *      |     |         *      |
        |        *       |     |        *       |
        |        *       |     |        *       |
        |        *       |     |        *       |
        |                |     |                |
        |                |     |                |
        |________________|     |________________|


PLAYER CARDS:
         ----------------       ----------------
        |                |     |                |
        |  4             |     |  8             |
        |                |     |                |
        |                |     |                |
        |                |     |                |
        |       ♥        |     |       ♦        |
        |                |     |                |
        |                |     |                |
        |                |     |                |
        |                |     |                |
        |            4   |     |            8   |
        |________________|     |________________|

PLAYER SCORE =  12
Enter H to Hit or S to Stand :
```



```
DEALER DECIDES TO HIT
PLAYER CARDS:
         ________________       ________________
        |                |     |                |
        |  4             |     |  8             |
        |                |     |                |
        |                |     |                |
        |                |     |                |
        |       ♥        |     |       ♦        |
        |                |     |                |
        |                |     |                |
        |                |     |                |
        |            4   |     |            8   |
        |________________|     |________________|

PLAYER SCORE =  12

DEALER CARDS:
         ________________       ________________       ________________
        |                |     |                |     |                |
        |  7             |     |  5             |     |  8             |
        |                |     |                |     |                |
        |                |     |                |     |                |
        |                |     |                |     |                |
        |       ♥        |     |       ♣        |     |       ♥        |
        |                |     |                |     |                |
        |                |     |                |     |                |
        |                |     |                |     |                |
        |            7   |     |            5   |     |            8   |
        |________________|     |________________|     |________________|


DEALER WINS!
Do you want to play again? (Y/N) :
```

After both the player and Dealer cards are revealed, the game gets over**.**

Conclusion and Future Work

The game “Blackjack” is widely played, but in offline mode. Our program provides the users to play and enjoy it in hybrid mode also and is user friendly. Anyone with some basic knowledge of the game can easily understand and enjoy the game.

This project can be further developed with GUI using tkinter and also, we can implement machine learning for gameplay from the dealer’s side.

References

1. Github.com
2. Stackoverflow.com

Appendix A: Code

```
"""
Project Problem Statement

Create a blackjack CUI game for the player, keeping all the rules in mind.
The dealer here would be a computer.
The choice of deck could be one, two or maximum three. If you are good at
probability, you may apply that.
"""


# Importing Required Modules
import os
import time
import random


"""


BlackJack Game

"""


# Function to clear the terimanl
def clear_terminal():
    os.system('cls' if os.name == 'nt' else 'clear')

# Card class to create card objects


class Card():
    def __init__(self, suit, card, card_score_value):

        # Suit of the Card like Spades, Clubs, Diamonds, Hearts
        self.__suit = suit

        # Representing the card like
2,3,4,5,6,7,8,9,10,Jack,Queen,King,Ace
        self.__card = card

        # Score value of the card
```

```
            self.__card_score_value = card_score_value

        def get_suit(self):
            return self.__suit

        def get_card(self):
            return self.__card

        def get_card_score_value(self):
            return self.__card_score_value

        def set_card_score_value(self, card_score_value):
            self.__card_score_value = card_score_value

    # Function to create deck of cards
    def deck():
        suits = ['Clubs', 'Diamonds', 'Hearts', 'Spades']

        cards = ['2', '3', '4', '5', '6', '7', '8',
                 '9', '10', 'Jack', 'Queen', 'King', 'Ace']

        card_values = {'2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8,
                       '9': 9, '10': 10, 'Jack': 10, 'Queen': 10, 'King': 10,
    'Ace': 11}
        deck = []
        for suit in suits:
            for card in cards:
                deck.append(Card(suit, card, card_values[card]))
        return deck

    # Function to print the cards
    def print_cards(cards, hidden):

        suit_symbols = {"Spades": "♠", "Hearts": "♥",
                        "Clubs": "♣", "Diamonds": "♦"}

        card_symbols = {"2": "2", "3": "3", "4": "4", "5": "5", "6": "6", "7":
    "7", "8": "8", "9": "9", "10": "10",
                        "Jack": "J", "Queen": "Q", "King": "K", "Ace": "A"}

        s = ""
        for card in cards:
            if (hidden):
```

```
        s += "\t ________________"
    else:
        s += "\t ________________"
print(s)

s = ""
for card in cards:
    if (hidden):
        s += "\t|                |"
    else:
        s += "\t|                |"
print(s)

s = ""
for card in cards:
    if (hidden):
        s += "\t|                |"
    else:
        if (card.get_card()) == '10':
            s = s + \
                "\t|  {}            |".format(
                    card_symbols[card.get_card()])
        else:
            s = s + \
                "\t|  {}             |".format(
                    card_symbols[card.get_card()])
print(s)

s = ""
for card in cards:
    if (hidden):
        s += "\t|      * *       |"
    else:
        s += "\t|                |"
print(s)

s = ""
for card in cards:
    if (hidden):
        s += "\t|    *     *     |"
    else:
        s += "\t|                |"
print(s)
```

```
    s = ""
    for card in cards:
        if (hidden):
            s += "\t|   *       *    |"
        else:
            s += "\t|                |"
    print(s)

    s = ""
    for card in cards:
        if (hidden):
            s += "\t|   *       *    |"
        else:
            s += "\t|                |"
    print(s)

    s = ""
    for card in cards:
        if (hidden):
            s += "\t|          *     |"
        else:
            s += 
"\t|       {}        |".format(suit_symbols[card.get_suit()])
    print(s)

    s = ""
    for card in cards:
        if (hidden):
            s += "\t|         *      |"
        else:
            s += "\t|                |"
    print(s)

    s = ""
    for card in cards:
        if (hidden):
            s += "\t|        *       |"
        else:
            s += "\t|                |"
    print(s)

    s = ""
```

```python
    for card in cards:
        if (hidden):
            s += "\t|        *       |"
        else:
            s += "\t|                |"
    print(s)

    s = ""
    for card in cards:
        if (hidden):
            s += "\t|        *       |"
        else:
            s += "\t|                |"
    print(s)

    s = ""
    for card in cards:
        if (hidden):
            s += "\t|                |"
        else:
            if (card.get_card() == '10'):
                s += "\t|            {}  |".format(
                    card_symbols[card.get_card()])
            else:
                s += "\t|            {}   |".format(
                    card_symbols[card.get_card()])
    print(s)

    s = ""
    for card in cards:
        if (hidden):
            s += "\t|________________|"
        else:
            s += "\t|________________|"
    print(s)
    print()


def play_game(deck):
 # Cards for both dealer and player
    player_cards = []
    dealer_cards = []
    # Scores for both dealer and player
    player_score = 0
    dealer_score = 0

    clear_terminal()

    # Initial dealing for player and dealer
    while len(player_cards) < 2:

        # Randomly dealing a card to player
        player_card = random.choice(deck)
        player_cards.append(player_card)
        deck.remove(player_card)

        # Updating the player score
        player_score += player_card.get_card_score_value()

        # In case both the cards are Ace, make the first ace value as 1
        if len(player_cards) == 2:
            if player_cards[0].get_card_score_value() == 11 and
player_cards[1].get_card_score_value() == 11:
                player_cards[0].set_card_score_value(1)
                player_score -= 10

        # Randomly dealing a card to dealer
        dealer_card = random.choice(deck)
        dealer_cards.append(dealer_card)
        deck.remove(dealer_card)

        # Updating the dealer score
        dealer_score += dealer_card.get_card_score_value()

        # In case both the cards are Ace, make the second ace value as 1
        if len(dealer_cards) == 2:
            if dealer_cards[0].get_card_score_value() == 11 and
dealer_cards[1].get_card_score_value() == 11:
                dealer_cards[1].set_card_score_value(1)
                dealer_score -= 10

    check_blackjack(player_cards, player_score, dealer_cards,
dealer_score)

    # Managing the player moves
```

```
        if (player_score < 21):
            clear_terminal()

            print("DEALER CARDS: ")
            print_cards(dealer_cards, True)

            print()

            print_player_cards_and_score(player_cards, player_score, False)

            choice = input("Enter H to Hit or S to Stand : ")

            # Checking for invalid input
            if len(choice) != 1 or (choice.upper() != 'H' and choice.upper()
    != 'S'):
                clear_terminal()
                print("Invalid Choice! Try Again")

            # If player decides to HIT
            if choice.upper() == 'H':

                # Dealing a new card
                player_card = random.choice(deck)
                player_cards.append(player_card)
                deck.remove(player_card)

                # Updating player score
                player_score += player_card.get_card_score_value()

                # Updating player score in case player's card have ace in them
                c = 0
                while player_score > 21 and c < len(player_cards):
                    if player_cards[c].get_card_score_value() == 11:
                        player_cards[c].set_card_score_value(1)
                        player_score -= 10
                        c += 1
                    else:
                        c += 1

                # Checking for blackjack
                check_blackjack(player_cards, player_score,
                                dealer_cards, dealer_score)
            # If player decides to Stand
            if choice.upper() == 'S':
                print("Player Stands")
                print("Dealer's Turn")

        # Managing the dealer moves
        while dealer_score < 17:
            clear_terminal()

            print("DEALER DECIDES TO HIT")
            # Dealing card for dealer
            dealer_card = random.choice(deck)
            dealer_cards.append(dealer_card)
            deck.remove(dealer_card)

            # Updating the dealer's score
            dealer_score += dealer_card.get_card_score_value()

            # Updating player score in case player's card have ace in them
            c = 0
            while dealer_score > 21 and c < len(dealer_cards):
                if dealer_cards[c].get_card_score_value() == 11:
                    dealer_cards[c].set_card_score_value(1)
                    dealer_score -= 10
                    c += 1
                else:
                    c += 1

        # Checking for blackjack
        check_blackjack(player_cards, player_score, dealer_cards,
    dealer_score)

        # print player and dealer cards
        print_player_cards_and_score(player_cards, player_score, False)
        print()
        print("DEALER CARDS: ")
        print_cards(dealer_cards, False)

        # Checking for the winner
        check_winner(player_score, dealer_score)


    # Print dealer cards and score
```

```python
def print_dealer_cards_and_score(dealer_cards, dealer_score, hidden):
    print("DEALER CARDS: ")
    print_cards(dealer_cards[:-1], hidden)
    print("DEALER SCORE = ", dealer_score -
          dealer_cards[-1].get_card_score_value())


# Print player cards and score
def print_player_cards_and_score(player_cards, player_score, hidden):
    print("PLAYER CARDS: ")
    print_cards(player_cards, hidden)
    print("PLAYER SCORE = ", player_score)

# Check if player or dealer has blackjack
def check_blackjack(player_cards, player_score, dealer_cards,
dealer_score):
    if player_score == 21:
        clear_terminal()
        print_player_cards_and_score(player_cards, player_score, False)
        print_dealer_cards_and_score(dealer_cards, dealer_score, False)
        print("BLACKJACK! YOU WIN!")
        play_again()
    # Dealer gets a blackjack
    if dealer_score == 21:
        clear_terminal()
        print_player_cards_and_score(player_cards, player_score, False)
        print_dealer_cards_and_score(dealer_cards, dealer_score, False)
        print("DEALER HAS A BLACKJACK! PLAYER LOSES!")
        play_again()


def check_winner(player_score, dealer_score):

    # Checking for dealer's win
    if player_score > 21:
        print("PLAYER BUSTED! GAME OVER!")
        play_again()

    # Checking for player's win
    if dealer_score > 21:
        print("DEALER BUSTED! YOU WIN!")
        play_again()
    # Checking for tie
    if dealer_score == player_score:
        print("\n")
        print("TIE GAME!")
        play_again()

    # Player Wins
    elif player_score > dealer_score:
        print("\n")
        print("PLAYER WINS!")
        play_again()

    # Dealer Wins
    else:
        print("\n")
        print("DEALER WINS!")
        play_again()


def play_again():
    choice = input("Do you want to play again? (Y/N) : ")
    if choice.upper() == 'Y':
        main()
    else:
        quit()


def main():

    print("""
====================================
Welcome to the game of Blackjack
====================================
    """)
    if (input("Do you want to play a game of Blackjack? (Y/N) ").upper()
== 'Y'):
        clear_terminal()
        # Creating a deck of cards
        card_deck = deck()

        # Starting the game
        play_game(card_deck)
    else:
        print("Thanks for playing")
        quit()
main()
```