INFORMATICS
INSTITUTE OF
TECHNOLOGY

Foundation Certificate in Higher Education

Module Code : DOC334

Module Title : Computer Programming

Module Leader : Nishan Saliya Harankahawa

Assessment Type : Individual

Issued Date : 14th March 2025

Hand-in Date : 04th April 2025

Weight : 50%

Student Group	C
Student Name	D.N.K Athukorala
Student ID	20241109
Centre	Colombo

I. Abstract

This Report documents the development of a war card game that runs on command line interface that made through python. This implementation uses object orientated programming to model cards, decks and game logic. Key features of this Game is card distribution, won card management, war scenario management and won/loss management. The Game stats automatically save into a text file and a html file. The code was validated through 10 cases including input validation and scenario validation. All tests passed confirming the functionality of the program. Assumptions Made to Modify the game to better functionality and mentioned in the report.

II. Acknowledgement

I sincerely thank Mr. Nisal Saliya for his valuable lecturing and tutoring, which greatly helped me in completing this Course work. His guidance, support and insightful feedback were really helpful in enhancing my understanding and improving the quality of this project.

Best regards,

D.N.K Athukorala

Table of Contents

1 Contents

III. List Of figures

IV. List of Tables

1 Game Overview

1. All cards in the deck of cards are being used where the 2 Joker cards have the highest value. So, the order would be,

- 2,3,4,5,6,7,8,9,J,Q,K,A and Joker
- Each card number will accompany with 4 suits as hearts (♥), diamonds (♦), clubs (♣), and spades (♠). However, the suits are not applicable to the 2 Joker cards. Those two cards are either black only or coloured.

2. The full deck of cards must be shuffled (all 54 cards)

3. Then the cards are distributed evenly among the human and PC (27 for each)

4. The human and the PC must shuffle their own deck of cards before starting the game

5. The first round starts when both keep drawing one card at a time and compare.

- If the human card value is higher than the PC card, the human wins and winner gets to keep both cards. If not the PC wins. For example, if human played 5♠ and the PC played 3♥, the human wins and get to keep both cards
- The suits are not taken into consideration.
- If both draws the same card (value is same but the suit is different), then it's a war! In a war, both parties will draw 3 more cards which the value is not shown and another card (4th card) which the value is shown. The 4th card decides who is the winner. For example, if human played J♠ and the PC played J♥, both of them will draw 3 cards where the value is not shown. Then if the fourth set of cards come as 6♦ for human and 2♣ for PC, the human wins.
    - If the 4th card is also the same in value, this process will go on till a higher 4th card is drawn by either party.

6. The default game play will run the game for one round and count all the cards belongs to the human and the PC at the end. The one with the highest number of cards will win. If the number of cards is equal, the game is a tie.

- If the player selected 2 to 5 game rounds (from console), the game will be run that many number of times and then start counting cards of each player to decide the winner.

7. The game simulation should happen automatically (start to end) once the player starts the war game. No user intervein is needed in between. Just display the status of the game in the console.

- For example, there is no need to ask the human player's name or a question like "Do you like to shuffle the cards?", "Do you want to draw the cards?", "Do you want to replay the game?", etc.

8. The game status must be saved to a text file automatically at the end of the game play. The text file name must be in below format.

- [date(yy-mm-dd)]_[time(hh-mm)]_[4 digit random number].txt for example, 20250325_8-55_4587.txt or 20250402_16-24_6359.txt, etc.

2 Algorithm

1. Initialization phase
   1.1. Card Value Initialization
      1.1.1. Adding card values (2-10)
      1.1.2. Adding face cards (Jack, Queen, King, Ace)
      1.1.3. Adding special cards (Joker)
   1.2. Card Suits Initialization
      1.2.1. Adding the Suit values (Hearts, Diamonds, Clubs, Spades)

2. Deck Creation and Distribution
   2.1. Generating the Deck by Adding the cards
   2.2. Shuffling the cards
   2.3. Distributing the cards equally to the user and the computer

3. Game Mechanism (Each Round)
   3.1. If both players card count is more than one, Draw a card from each player
   3.2. Compare each card's values (without caring about the suits)
   3.3. Checking if it is a win/loss or a Draw
   3.4. According to the results, the cards go to the winner

4. War Handling
   4.1. If both cards are equal, it is a war
   4.2. Checking if both players have enough cards to play the war
   4.3. If yes, then each player puts 3 face down cards else the player who has the most cards wins
   4.4. If not, both players get the remaining cards from their winning decks.
   4.5. After 3 cards each player puts a faceup card, winner gets all the cards
   4.6. If that faceup card is equal, it is another war

5. Game Mechanism (Full Game)
   5.1. The game mechanism for each round continues until the player with the most cards at the beginning runs out of cards
   5.2. After a round, won cards and the remaining cards of each user getting combined and shuffles their individual decks
   5.3. If the user wanted to play more than one game, the game would play one more time

6. Saving Game History
   6.1. Text File
      6.1.1. Naming the file as (played date. played time. 4-digit random number)
      6.1.2. Saving the game details round by round
      6.1.3. Saving the war details
   6.2. HTML file
      6.2.1. Naming the file by replacing the text(txt) to html(html) extensions.
      6.2.2. HTML Formatted Game Visualization like the text file

7. Game Conclusion
    7.1. Distribute cards to each player
    7.2. Count the number of wars that occurred
    7.3. Determine the final winner
    7.4. Save the Game details

8. Control structure
    8.1. Ask for the input of the user (1 to 5 Rounds)
    8.2. Play the game according to the given rounds
    8.3. Handle wars
    8.4. Decide the winner
    8.5. Output the winner, Loser and the War count

3 Python Program

```
import random
import sys
from datetime import datetime
import os



#To represent each card


class Card:
  def __init__(self,value,suit=None,is_joker=False):
    self.value=value
    self.suit=suit
    self.is_joker=is_joker

  def __str__(self):
    if self.is_joker:
      return "Joker"

    value_names= { 2:'2' , 3:'3' , 4:'4' , 5:'5' , 6:'6' , 7:'7' , 8:'8' , 9:'9' , 10:'10' , 11:'J' , 12:'Q' , 13:'K' ,
14:'A' }

    symbols= { "hearts": "♥" , "diamonds": "♦" , "clubs": "♣" , "spades": "♠" }

    return f'{value_names[self.value]}{symbols[self.suit]}'

  def __lt__(self,other):
```

```
        return self.value < other.value


#To manage a collection of cards

class PlayingDeck:
    def __init__(self, cards=None):
        self.cards=cards if cards else []

    def shuffle(self):        #function to shuffle the cards
        random.shuffle(self.cards)

    def drawing_card(self):  #function to draw a card
        if len(self.cards) > 0:
            return self.cards.pop(0)
        else:
            return None

    def store_card(self,card):   #store cards
        self.cards.append(card)

    def store_cards(self,cards):  #store cards
        self.cards.extend(cards)

    def count(self):      #to count the num of cards
        return len(self.cards)


#Deck Creation Functions

class WarGame:
```

```
def __init__(self,rounds=1):
    self.rounds = min(max(rounds,1),5) #to limit the rounds between 1 to 5
    self.players_cards = PlayingDeck()
    self.computer_cards = PlayingDeck()
    self.player_won_pile = PlayingDeck()
    self.computer_won_pile = PlayingDeck()
    self.game_history = []
    self.turn_records = []
    self.war_count = 0
    self.round_results = []
    self.all_round_turn_records = []

def game_setup(self):
    self.players_cards = PlayingDeck()    #creating seprate decks for (win/in play)
    self.computer_cards = PlayingDeck()
    self.player_won_pile = PlayingDeck()
    self.computer_won_pile = PlayingDeck()

    full_deck = PlayingDeck()
    suits = ['hearts','diamonds','clubs','spades']

    for suit in suits:
        for value in range(2,15):
            full_deck.store_card(Card(value, suit))

    full_deck.store_card(Card(15, is_joker=True))
    full_deck.store_card(Card(15, is_joker=True))

    total_cards = full_deck.count()
    print(f"Total cards in Deck: {total_cards}".center(80))
```

```
    full_deck.shuffle() #shuffling the full deck


    give_to_player = True
    while full_deck.count() > 0:
      if give_to_player:
        self.players_cards.store_card(full_deck.drawing_card())
        give_to_player = False
      else:
        self.computer_cards.store_card(full_deck.drawing_card())
        give_to_player = True


    print("Cards shuffled and Distributed Among The two players\n".center(80))
    print(f"Human cards: {self.players_cards.count() }")
    print(f"Pc cards: {self.computer_cards.count() }\n\n")



  def handle_war(self,cards_in_play):     #war handling Function
    self.turn_records.append(" Additional war cards ")
    self.war_count += 1


    available_human_cards = self.players_cards.count()
    available_pc_cards = self.computer_cards.count()


    print(f"WAR Has Started!!!".center(80))
    message = f"\nHuman has {available_human_cards} cards and PC has {available_pc_cards}
cards"
    print(message.center(80))


    if available_human_cards < 4 and self.player_won_pile.count() > 0:  #To get cards from the
winning deck
      cards_needed = 4 - available_human_cards
      new_card_add = min(cards_needed,self.player_won_pile.count())
```

```
        if new_card_add > 0:
            print(f"Human Getting {new_card_add} cards from his won pile for this WAR")
            self.player_won_pile.shuffle()
            for _ in range(new_card_add):
                self.players_cards.store_card(self.player_won_pile.drawing_card())
            available_human_cards = self.players_cards.count()

    if available_pc_cards < 4 and self.computer_won_pile.count() > 0:    #To get cards from the winning deck
        cards_needed = 4 - available_pc_cards
        new_card_add = min(cards_needed,self.computer_won_pile.count())
        if new_card_add > 0:
            print(f"PC Getting {new_card_add} cards from the won pile for the WAR")
            self.computer_won_pile.shuffle()
            for _ in range(new_card_add):
                self.computer_cards.store_card(self.computer_won_pile.drawing_card())
            available_pc_cards = self.computer_cards.count()

    available_human_cards = self.players_cards.count()
    available_pc_cards = self.computer_cards.count()



    if available_human_cards == 0:            #if one player is only ran out of cards
        print("Human has no more cards - computer wins the war")
        return "pc"
    if available_pc_cards == 0:
        print("Computer has no more cards - Human wins the war")
        return "human"

    if available_human_cards < 4 and available_pc_cards < 4:     #if both players dont have enough cards for war
        print("Both players Don't have enough cards for a full war even after adding new cards")
```

```
    for _ in range(available_human_cards - 1):
        cards_in_play.append(self.players_cards.drawing_card())

    for _ in range(available_pc_cards - 1):
        cards_in_play.append(self.computer_cards.drawing_card())



    human_up_card = self.players_cards.drawing_card()
    pc_up_card = self.computer_cards.drawing_card()
    cards_in_play.extend([human_up_card,pc_up_card])

    print(f"Human final war card: {human_up_card}")
    print(f"PC final war card: {pc_up_card}")

    if human_up_card.value > pc_up_card.value:
        return "human"
    else:
        return "pc"

human_face_down = min(3,available_human_cards - 1)
pc_face_down = min(3,available_pc_cards -1)

for _ in range(human_face_down):
    cards_in_play.append(self.players_cards.drawing_card())

for _ in range(pc_face_down):
    cards_in_play.append(self.computer_cards.drawing_card())

human_up_card = self.players_cards.drawing_card()
pc_up_card = self.computer_cards.drawing_card()
```

```
    cards_in_play.extend([human_up_card,pc_up_card])
    print(f"Human war card: {human_up_card}")
    print(f"PC war card: {pc_up_card}")

    if human_up_card.value == pc_up_card.value:    #if there is another war occured
      print("It's Another WAR!!!")
      return self.handle_war(cards_in_play)
    elif human_up_card.value > pc_up_card.value:
      return "human"
    else:
      return "pc"



  def play_game(self):
    for round_num in range(1, self.rounds + 1):
      self.turn_records = []
      self.war_count = 0
      print(f"\n --- Round {round_num} ---")

      #To complete the assumpiton of After the second round The game Has to continue until
player with the most cards runs out of cards


      if round_num == 1:
        self.game_setup()
      else:
        print("Starting new round")
        player_with_most_cards = "human" if human_started_with >= pc_started_with else "pc"
        print(f"{player_with_most_cards.capitalize()} started with the most cards")



        self.players_cards.store_cards(self.player_won_pile.cards)
```

```
        self.computer_cards.store_cards(self.computer_won_pile.cards)

        self.player_won_pile = PlayingDeck()
        self.computer_won_pile = PlayingDeck()

        self.players_cards.shuffle()
        self.computer_cards.shuffle()

        print(f"Human starts with {self.players_cards.count()} cards")
        print(f"Computer starts with {self.computer_cards.count()} cards")
        print(f"Total cards in play: {self.players_cards.count() + self.computer_cards.count()}")


    human_started_with = self.players_cards.count()
    pc_started_with = self.computer_cards.count()

    player_with_most_cards = None if round_num == 1 else ("human"if human_started_with
>= pc_started_with else "pc")


    card_difference = abs(human_started_with - pc_started_with)

    turn = 1

    while True:
        if round_num == 1:
            if self.players_cards.count() == 0 or self.computer_cards.count() == 0:
                print(f"\nBoth Players Ran Out of cards. Round {round_num} complete.".center(80))
                break
        else:
            if player_with_most_cards == "human" and self.players_cards.count() == 0:
```

```
            print(f"\nHuman (who started with most cards) is out of cards. Round {round_num} complete.".center(80))
            break

        if player_with_most_cards == "pc" and self.computer_cards.count() == 0:
            print(f"\nComputer (who started with most cards) is out of cards. Round {round_num} complete.".center(80))
            break

        if human_started_with < pc_started_with and self.players_cards.count() == 0:
            if self.player_won_pile.count() > 0:
                cards_needed = min(card_difference,self.player_won_pile.count())
                print(f"Human is getting {cards_needed} cards from won pile")
                self.player_won_pile.shuffle()
                for _ in range(cards_needed):
                    if self.player_won_pile.count() > 0:
                        self.players_cards.store_card(self.player_won_pile.drawing_card())
            else:
                print("Human has no more cards to play")
                break


        if pc_started_with < human_started_with and self.computer_cards.count() == 0:
            if self.computer_won_pile.count() > 0:
                cards_needed = min(card_difference,self.computer_won_pile.count())
                print(f"Computer is getting {cards_needed} cards from won pile")
                for _ in range(cards_needed):
                    if self.computer_won_pile.count() > 0:
                        self.computer_cards.store_card(self.computer_won_pile.drawing_card())

            else:
                print("Computer has no more cards to play. Ending Round.")
```

```
        break

    if self.players_cards.count() == 0 or self.computer_cards.count() == 0:
        print("One Player has no more cards to play. Ending Round.")
        break

    #Console Ouput
    print(f"\nTurn {turn}:")



    human_card = self.players_cards.drawing_card()
    pc_card = self.computer_cards.drawing_card()

    cards_in_play = [human_card,pc_card]

    print(f"Player Puts: {human_card}")
    print(f"Computer Puts: {pc_card}")

    if human_card.value == pc_card.value:
        self.turn_records.append(f"{turn} : {human_card} vs {pc_card} - WAR")

        result = self.handle_war(cards_in_play)

        if result == "human":
            self.player_won_pile.store_cards(cards_in_play)
            print(f"Player wins the war and gets {len(cards_in_play)} cards")
        else:
            self.computer_won_pile.store_cards(cards_in_play)
            print(f"Computer wins the war and gets {len(cards_in_play)} cards")

    elif human_card.value > pc_card.value:
```

```
            self.player_won_pile.store_cards(cards_in_play)
            print(f"Player wins and gets 2 cards")
            self.turn_records.append(f"{turn} : {human_card} vs {pc_card} - H")

        else:
            self.computer_won_pile.store_cards(cards_in_play)
            print(f"Computer wins and gets 2 cards")
            self.turn_records.append(f"{turn} : {human_card} vs {pc_card} -P")



        turn += 1

    human_card_count = self.players_cards.count() + self.player_won_pile.count()
    pc_card_count = self.computer_cards.count() + self.computer_won_pile.count()
    total_cards = human_card_count + pc_card_count

    print(f"\nRound {round_num} Complete!")
    print(f"Human cards: {self.players_cards.count()} in hand, {self.player_won_pile.count()} in won pile")
    print(f"Computer cards: {self.computer_cards.count()} in hand, {self.computer_won_pile.count()} in won pile")
    print(f"Total cards in play: {total_cards}")



    self.all_round_turn_records.append(self.turn_records.copy())



    round_winner = "PC" if pc_card_count > human_card_count else "Human" if human_card_count > pc_card_count else "Tie"
    self.round_results.append({"round": round_num,
                               "human_cards": human_card_count,
                               "pc_cards": pc_card_count,
```

```
                    "war_count": self.war_count,
                    "winner": round_winner})

    print("\n----- ROUND SUMMARY -----")
    print(f"Round {round_num} results")
    print("NO : Hum vs PC - Winner")

    turn_number = 1

    for turn_record in self.turn_records:
        if ":" in turn_record:
            turn_content = turn_record.split(":",1)[1].strip()
            print(f"{turn_number} : {turn_content}")
            turn_number += 1
        else:
            print(turn_record)

    print("\nPC card count",pc_card_count)
    print("Human card count",human_card_count)
    print("War count",self.war_count)

    if pc_card_count > human_card_count:
        print("\nPC won the round!")
    elif human_card_count > pc_card_count:
        print("\nHuman won the round!")
    else:
        print("\nTie")


pc_wins = sum(1 for r in self.round_results if r["winner"] == "PC")
human_wins = sum(1 for r in self.round_results if r["winner"] == "Human")
```

```
    print("\n---YOUR GAME HAS FINISHED---")
    print(f"Rounds played: {self.rounds}")
    print(f"PC won {pc_wins} rounds")
    print(f"Human won {human_wins} rounds")

    if pc_wins > human_wins:          #To Find the Overall winner of all the rounds
        print("\nPC won the Game")
        overall_winner = "PC"
    elif human_wins > pc_wins:
        print("\nHuman won the Game")
        overall_winner = "Human"
    else:
        print("\nThe Game is a Tie")
        overall_winner = "Tie"

    return overall_winner

def save_game_log(self):     #save to text file
    time = datetime.now()
    random_num = random.randint(1000,9999)
    date_str = time.strftime("%Y%m%d")
    time_str = time.strftime("%H-%M")

    filename = f"{date_str}_{time_str}_{random_num}.txt"

    with open(filename, 'w' , encoding = 'utf-8' ) as f:
        f.write("      WAR GAME        ")
        time = datetime.now()
        f.write(f"Date : {time.strftime('%Y-%m-%d')}\n")
        f.write(f"Time : {time.strftime('%H:%M')}\n\n")
```

```
f.write(f"Total Rounds : {self.rounds}\n\n")

for round_num in range(1,self.rounds + 1):
    round_data = self.round_results[round_num - 1]
    round_turns = self.all_round_turn_records[round_num - 1]

    f.write("Round Results\n")
    f.write("NO : PC VS H - Winner\n")

    turn_number = 1

    for turn in round_turns:
        if ":" in turn:
            turn_content = turn.split(":",1)[1].strip()
            f.write(f"{turn_number} : {turn_content}\n")
            turn_number += 1
        else:
            f.write(f"{turn}\n")



    f.write(f"{turn}\n")
    f.write(f"PC card count {round_data['pc_cards']}\n")
    f.write(f"Human card count {round_data['human_cards']}\n")
    f.write(f"War count {round_data['war_count']}\n\n")

    if round_data['winner'] == "PC":
        f.write("PC Won The Round\n\n")
    elif round_data['winner'] == "Human":
        f.write("Human Won The Round\n\n")
    else:
        f.write("It's a Tie\n\n")
```

```
            f.write("------------------------\n\n")

        return filename



    def save_html_log(self,txt_filename):      #save to html file
        html_filename = txt_filename.replace('.txt','.html')
        time = datetime.now()


        html_content = f"""<!DOCTYPE html>
    <html>
    <head>
        <title> War Card Game - {time.strftime('%Y-%m-%d')} </title>
        <style>
            body {{ font-family: Consolas, monospace; line-height: 1.5; margin: 20px; }}
            h1 {{ color: #333; }}
            -header {{ background - color: #333; color:white; padding: 2px 10px; }}
            -menu {{ color: #666; margin-bottom: 15px; }}
            -game-history {{ font-family: consolas, monospace; }}
            -war {{ color:red; font-weight:bold; }}
            -winner {{ color: green; font-weight: bold; }}
            table {{ border-collapse: collapse; width: 100%;margin:15px 0; }}
            td, th {{ padding: 5px; text-allign: left; }}
            .round-summary {{ background-color: #f0f0f0; padding: 10px; margin-bottom: 20px; }}
            .round-header {{ background-color: #333; color: white; padding: 5px 10px; margin-top:
20px; }}
            .round-separator {{ border-top: 2px solid #333; margin: 20px 0; }}
        </style>
    </head>
    <body>
```

```
    <div class="header"> WAR GAME </div>

    <div class="game-history">
      <p>Date : {time.strftime('%Y-%m-%d') } </p>
      <p>Time : {time.strftime('%H:%M') } </p>
      <p>Total Rounds : {self.rounds}</p>

"""
    for round_num in range(1,self.rounds + 1):
      round_data = self.round_results[round_num - 1]
      round_turns = self.all_round_turn_records[round_num - 1]



      html_content += f"""<div class="round-header">Round {round_num} results</div>
                  <div class="round-summary">
                    <p>No : Hum vs PC - Winner</p>"""
      turn_number = 1

      for turn in round_turns:
        if ":" in turn:
          turn_content = turn.split(":",1)[1].strip()

          if "WAR" in turn_content:
            html_content += f'<p class="war">{turn_number} : {turn_content}</p>\n'
          else:
            html_content += f'<p>{turn_number} : {turn_content}</p>\n'

          turn_number += 1

        else:
          html_content += f'<p>{turn}</p>\n'
```

```
    html_content += f"""
    <p>PC card count {round_data['pc_cards']}</p>
    <p>Human card count {round_data['human_cards']}</p>
    <p>War Count {round_data['war_count']}</p>

    <p class="winner">"""



    if round_data['winner'] == "PC":
        html_content += "PC won the round!"
    elif round_data['winner'] == "Human":
        html_content += "Human won the round!"
    else:
        html_content += "Round ended in a tie!"

    html_content += """</p>
    </div>
    <div class="round-separator"></div> """

pc_wins = sum(1 for r in self.round_results if r["winner"] == "PC")
human_wins = sum(1 for r in self.round_results if r["winner"] == "Human")

html_content += f"""
    <div class="round-header">Game Summary</div>
    <div class="winner round-summary">
        <p>PC won {pc_wins} rounds</p>
        <p>Human won {human_wins} rounds</p>
"""
```

```
        if pc_wins > human_wins:
            html_content += "<p><strong>PC Won The Game</strong></p>"
        elif human_wins > pc_wins:
            html_content += "<p><strong>Human Won The Game</strong></p>"
        else:
            html_content += "<p><strong>It's a Tie</strong></p>"

        html_content += """
            </div>
            <hr>
        </div>
    </body>
    </html>
    """

        with open(html_filename, 'w' , encoding='utf-8') as f:
            f.write(html_content)

        return html_filename

def main():        #main function to run the game
    if len(sys.argv) > 1:
        try:
            rounds = int(sys.argv[1])
            if not (1<= rounds <=5):
                print("Please a Value between 1 to 5.(ex:war 3)")
                return
            game = WarGame(rounds)
        except ValueError:
            print("Please a Value between 1 to 5.(ex:war 3)")
            return
```

```
    else:
        game = WarGame()



    game.play_game()

    txt_filename = game.save_game_log()

    try:
        html_filename = game.save_html_log(txt_filename)
    except Exception as e:
        pass

if __name__ == "__main__":
    main()
```

4 Assumptions

1. Game Only supports 1-5 Rounds (default 1)

2. The Game continues only until the player who started with the most cards runs out of cards (after round 1)

   Test case (5.2.4) explains, if Human starts with 28 cards and pc with 26, the game ends when human finishes their 28 cards

3. If Both players Don't have Enough cards to play the war, they both can get the needed cards from their won pile to continue the war

   Test case (5.2.2) explains, if Human and pc needed more cards for war, they get that by their won pile.

4. After all the rounds the player who wins the most rounds wins the overall game

5. A round winner is determined by the total cards collected in the end of the round, (Not just the won cards)

6. If both players won the same number of rounds the game is a tie.

5 Test Cases

5.1 Input Validation Test Cases

5.1.1 Test case 1

Table 1:Test case 1

Test case	Input	Expected Output	Actual Output	Result
1	war	Game Runs for one round -save the results to the text file -save the results to the html file	Game ran for exactly one Round -Game results saved to the text and html file	pass

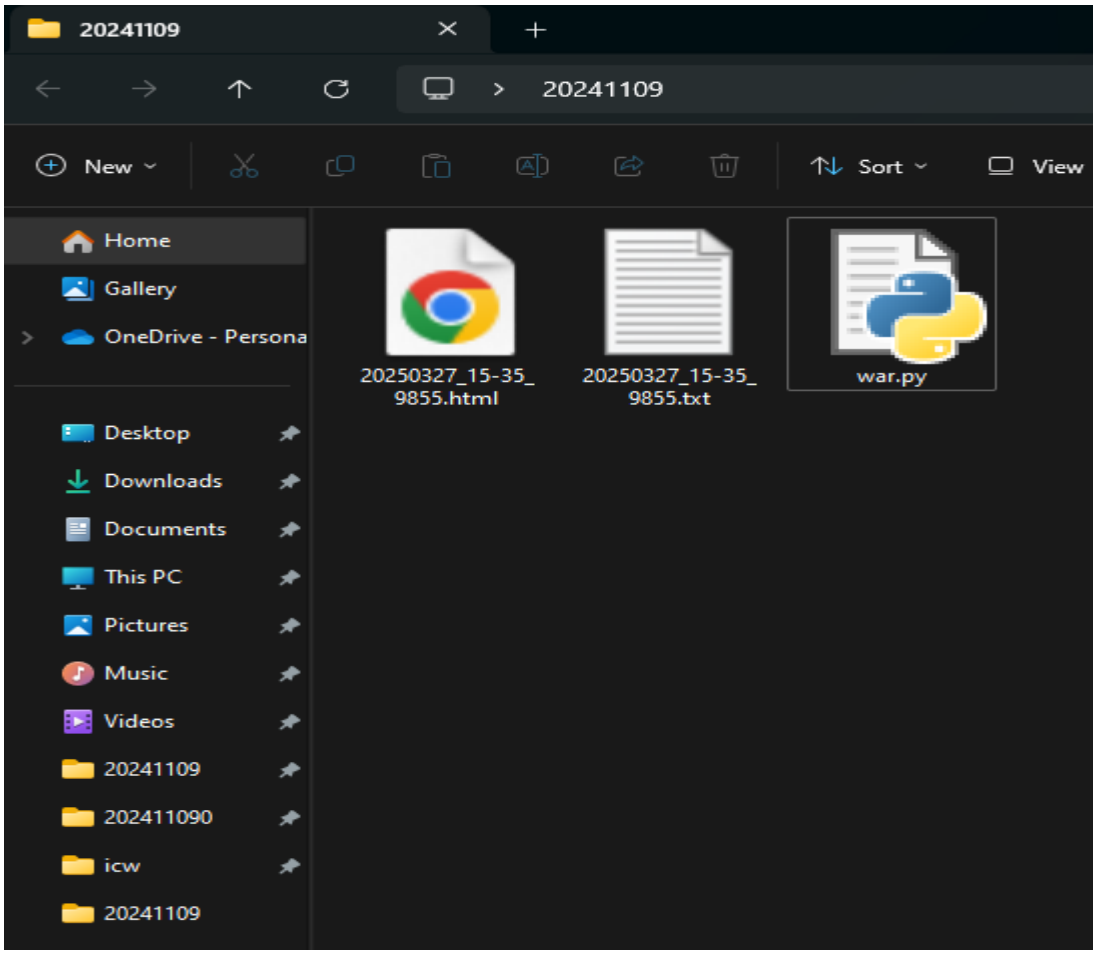


Figure 1 : when user inputs "war" saves a TXT file and HTML file

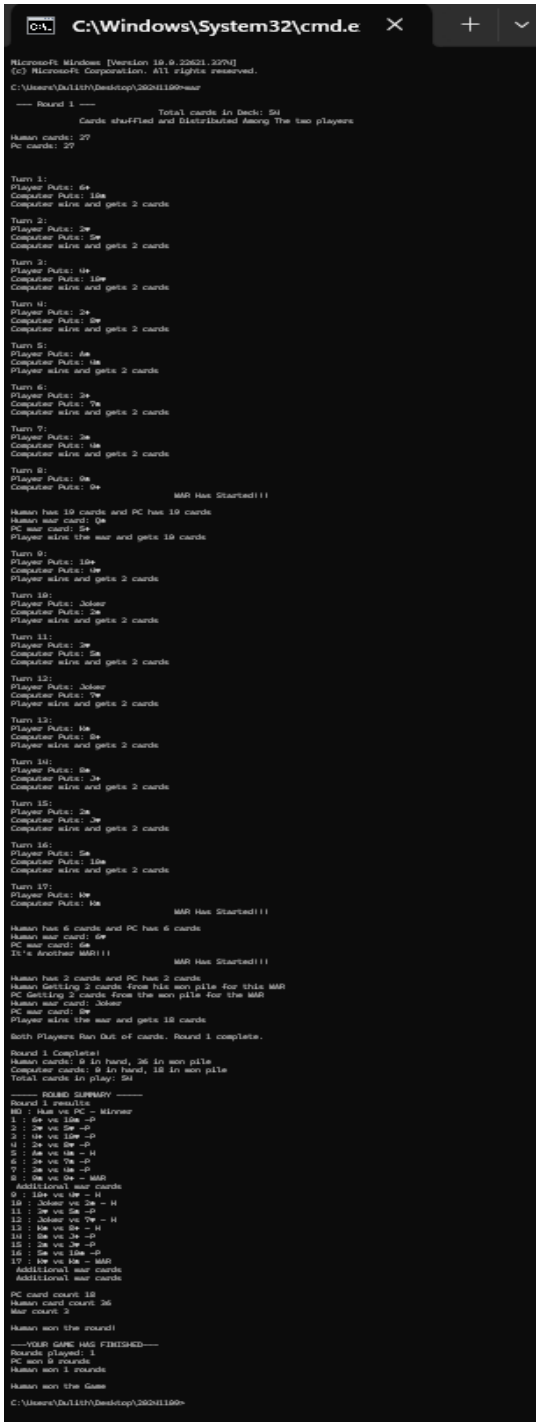


Figure 2: When the user inputs "war" the game runs on the console

```
        WAR GAME          Date : 2025-04-04
Time : 18:26

Total Rounds : 1

Round Results
NO : PC VS H - Winner
1 : 6♦ vs 10♣ -P
2 : 2♥ vs 5♥ -P
3 : 4♦ vs 10♥ -P
4 : 2♦ vs 8♥ -P
5 : A♠ vs 4♣ - H
6 : 3♦ vs 7♣ -P
7 : 3♠ vs 4♠ -P
8 : 9♣ vs 9♦ - WAR
 Additional war cards
9 : 10♦ vs 4♥ - H
10 : Joker vs 2♠ - H
11 : 3♥ vs 5♣ -P
12 : Joker vs 7♥ - H
13 : K♠ vs 8♦ - H
14 : 8♠ vs J♦ -P
15 : 2♣ vs J♥ -P
16 : 5♠ vs 10♠ -P
17 : K♥ vs K♣ - WAR
 Additional war cards
 Additional war cards
 Additional war cards
PC card count 18
Human card count 36
War count 3

Human Won The Round

------------------------
```

Figure 3:Game History Saved into the text file

WAR GAME

Date : 2025-04-04

Time : 18:26

Total Rounds : 1

Round 1 results

No : Hum vs PC - Winner

1 : 6♦ vs 10♣ -P

2 : 2♥ vs 5♥ -P

3 : 4♦ vs 10♥ -P

4 : 2♦ vs 8♥ -P

5 : A♠ vs 4♣ - H

6 : 3♦ vs 7♣ -P

7 : 3♠ vs 4♠ -P

8 : 9♣ vs 9♦ - WAR

Additional war cards

9 : 10♦ vs 4♥ - H

10 : Joker vs 2♠ - H

11 : 3♥ vs 5♣ -P

12 : Joker vs 7♥ - H

13 : K♠ vs 8♦ - H

14 : 8♠ vs J♦ -P

15 : 2♣ vs J♥ -P

16 : 5♠ vs 10♠ -P

17 : K♥ vs K♣ - WAR

Additional war cards

Additional war cards

PC card count 18

Human card count 36

War Count 3

Human won the round!

Game Summary

PC won 0 rounds

Human won 1 rounds

Human Won The Game

Figure 4:Game History Saved into the HTML file

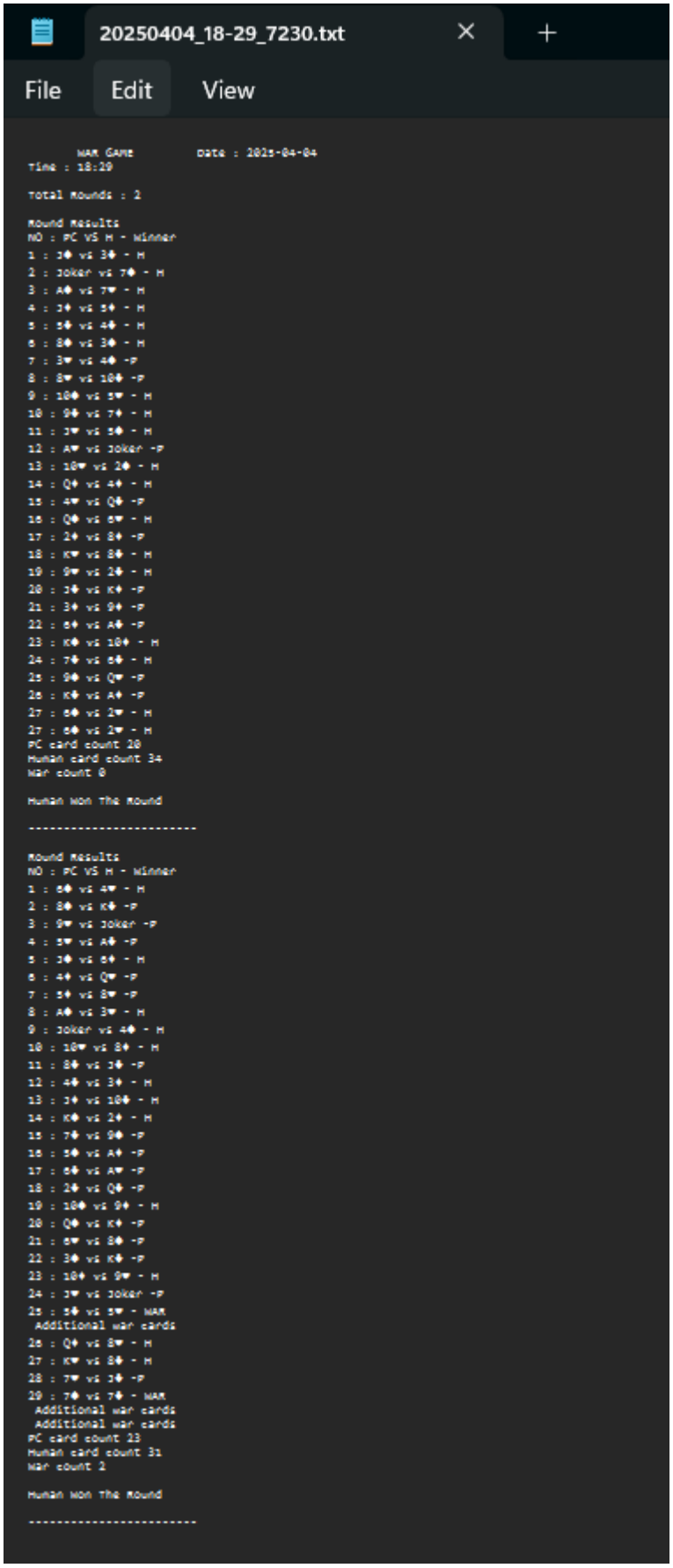
5.1.2 Test case 2

Table 2:Test case 2

Test Case	Input	Expected Output	Actual Output	Result
2	war 2	Game plays exactly for 2 rounds and saves to text file and html file	Game plays exactly for 2 rounds and saves to text file and html file	pass

Figure 5: Round 1 result

Figure 6: Round 2 Results



20250404_18-29_7230.txt

File Edit View

```
        WAR GAME          Date : 2025-04-04
Time : 18:29

Total Rounds : 2

Round Results
NO : PC VS H - Winner
1 : 3♦ vs 3♦ - H
2 : Joker vs 7♦ - H
3 : A♦ vs 7♥ - H
4 : J♦ vs 5♦ - H
5 : 5♦ vs 4♦ - H
6 : 8♦ vs 3♦ - H
7 : 3♥ vs 4♦ -P
8 : 8♥ vs 10♦ -P
9 : 10♦ vs 5♥ - H
10 : 9♦ vs 7♦ - H
11 : J♥ vs 5♦ - H
12 : A♥ vs Joker -P
13 : 10♥ vs 2♦ - H
14 : Q♦ vs 4♦ - H
15 : 4♥ vs Q♦ -P
16 : Q♦ vs 6♥ - H
17 : 2♦ vs 8♦ -P
18 : K♥ vs 8♦ - H
19 : 9♥ vs 2♦ - H
20 : J♦ vs K♦ -P
21 : 3♦ vs 9♦ -P
22 : 6♦ vs A♦ -P
23 : K♦ vs 10♦ - H
24 : 7♦ vs 6♦ - H
25 : 9♦ vs Q♥ -P
26 : K♦ vs A♦ -P
27 : 6♦ vs 2♥ - H
27 : 6♦ vs 2♥ - H
PC card count 20
Human card count 34
War count 0

Human Won The Round

------------------------

Round Results
NO : PC VS H - Winner
1 : 5♦ vs 4♥ - H
2 : 8♦ vs K♦ -P
3 : 9♥ vs Joker -P
4 : 5♥ vs A♦ -P
5 : J♦ vs 6♦ - H
6 : 4♦ vs Q♥ -P
7 : 5♦ vs 8♥ -P
8 : A♦ vs 3♥ - H
9 : Joker vs 4♦ - H
10 : 10♥ vs 8♦ - H
11 : 8♦ vs J♦ -P
12 : 4♦ vs 3♦ - H
13 : J♦ vs 10♦ - H
14 : K♦ vs 2♦ - H
15 : 7♦ vs 9♦ -P
16 : 5♦ vs A♦ -P
17 : 6♦ vs A♥ -P
18 : 2♦ vs Q♦ -P
19 : 10♦ vs 9♦ - H
20 : Q♦ vs K♦ -P
21 : 6♥ vs 8♦ -P
22 : 3♦ vs K♦ -P
23 : 10♦ vs 9♥ - H
24 : J♥ vs Joker -P
25 : 5♦ vs 5♥ - WAR
 Additional war cards
26 : Q♦ vs 8♥ - H
27 : K♥ vs 8♦ - H
28 : 7♥ vs J♦ -P
29 : 7♦ vs 7♦ - WAR
 Additional war cards
 Additional war cards
PC card count 23
Human card count 31
War count 2

Human Won The Round

------------------------
```

Figure 7 : war 2 round text file

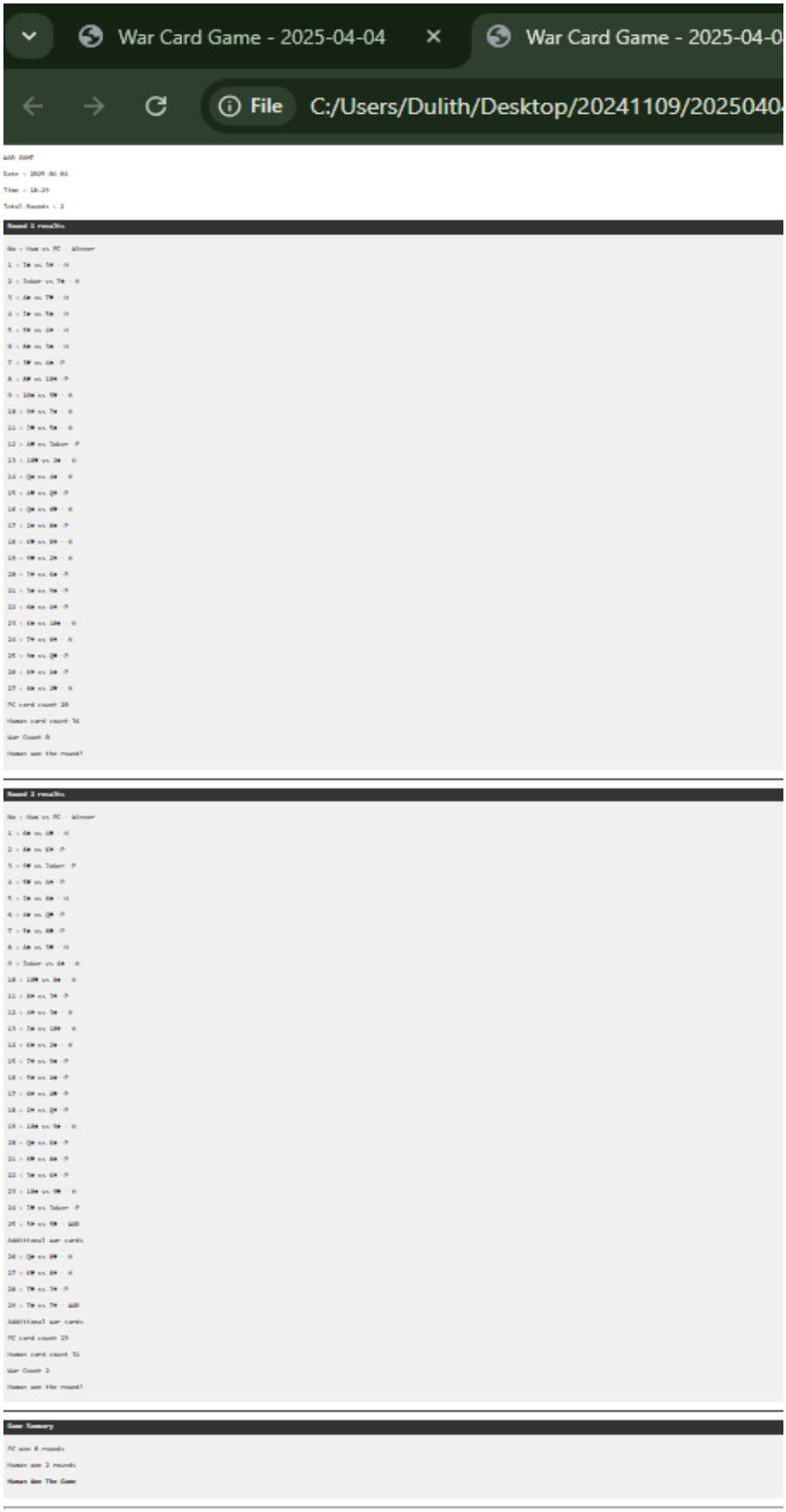


Figure 8 : war 2 round html file

5.1.3 Test case 3

Table 3:Test case 3

Test Case	Input	Expected Output	Actual Output	Result
3	War 0	Game doesn't play show an error as to enter a value between 1 to 5 after entering 'war'	Game doesn't play show an error as to enter a value between 1 to 5 after entering 'war'	pass

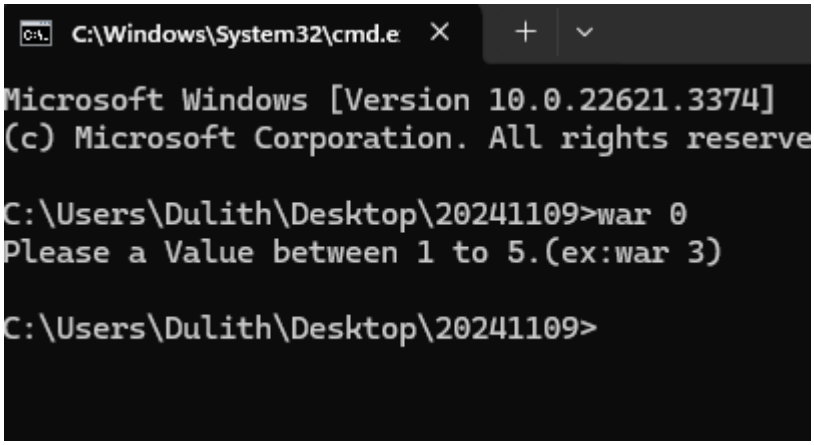


Figure 9 : 'war 0' error handling

5.1.4 Test case 4

Table 4:Test case 4

Test Case	Input	Expected Output	Actual Output	Result
4	War 6	Game doesn't play show an error as to enter a value between 1 to 5 after entering 'war'	Game dosen't play show an error as to enter a value between 1 to 5 after entering 'war'	pass

```
C:\Users\Dulith\Desktop\20241109>war 6
Please a Value between 1 to 5.(ex:war 3)

C:\Users\Dulith\Desktop\20241109>
```

Figure 10: 'war 6' error handling

5.1.5 Test case 5

Table 5:Test case 5

Test Case	Input	Expected Output	Actual Output	Result
5	War abc	Game Doesn't Play and Give the output as play “Insert a Number Between 1 to 5. (“ex: war 4”)”	Game Doesn't Play and Give the output as play “Insert a Number Between 1 to 5. (“ex: war 4”)”	pass

```
C:\Users\Dulith\Desktop\20241109>war abc
Please a Value between 1 to 5.(ex:war 3)

C:\Users\Dulith\Desktop\20241109>
```

Figure 11: ‘war abc’ error handling

5.2 Scenario Based Validation

5.2.1 Test case 6

Table 6:Test case 6

Test Case	Scenario tested	Expected Output	Actual Output	Result
6	War happens correctly	When the two players cards get equal the war function needs to happen	When the two players cards get equal the war function happened	pass

```
Turn 6:
Player Puts: Q♦
Computer Puts: Q♠
                                WAR Has Started!!!

Human has 21 cards and PC has 21 cards
Human war card: 6♣
PC war card: 10♠
Computer wins the war and gets 10 cards
```

Figure 12: war scenario

5.2.2 Test Case 7

Table 7:Test case 7

Test Case	Scenario tested	Expected Output	Actual Output	Result
7	When both players don't have enough cards for war	Both players should get the needed cards from their winning deck and continue the war.	Both players should get the needed cards from their winning deck and continue the war.	pass

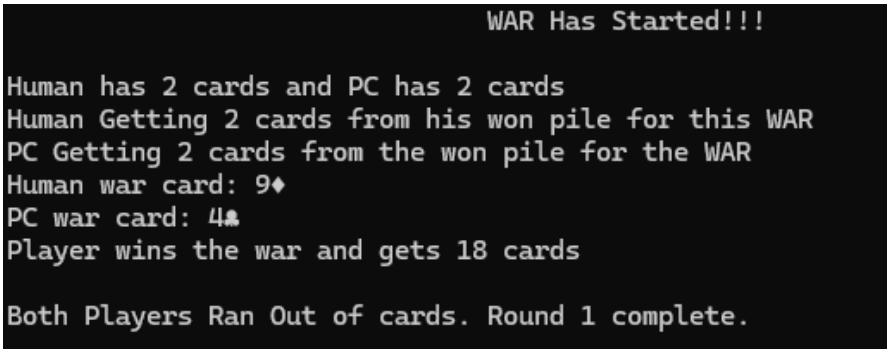


Figure 13 : Getting Cards from the winning deck to play the war

5.2.3 Test Case 8

Table 8:Test case 8

Test Case	Scenario tested	Expected Output	Actual Output	Result
8	More than one war scenario	In a war scenario if the faceup card also matches there has to be another war	In a war scenario, the faceup card also matched and another war started	pass

```
Turn 21:
Player Puts: 5♠
Computer Puts: 5♦
                                WAR Has Started!!!

Human has 6 cards and PC has 6 cards
Human war card: 2♦
PC war card: 2♠
It's Another WAR!!!
                                WAR Has Started!!!

Human has 2 cards and PC has 2 cards
Human Getting 2 cards from his won pile for this WAR
PC Getting 2 cards from the won pile for the WAR
Human war card: 9♦
PC war card: 4♣
Player wins the war and gets 18 cards
```

Figure 14:WAR after WAR scenario

5.2.4 Test Case 9

Table 9:Test case 9

Test Case	Scenario tested	Expected Output	Actual Output	Result
9	Game Draws Decider	The Game played only until the player with the most cards in the beginning ran out of cards	The Game played only until the player with the most cards in the beginning ran out of cards	pass

```
 --- Round 4 ---
Starting a new round
Pc started with the most cards
Human starts with 26 cards
Computer starts with 28 cards
Total cards in play: 54
```

Figure 15:Beginning of the round

```
Turn 28:
Player Puts: A♥
Computer Puts: J♣
Player wins and gets 2 cards

Computer (who started with most cards) is out of cards. Round 4 complete.

Round 4 Complete!
Human cards: 0 in hand, 34 in won pile
Computer cards: 0 in hand, 20 in won pile
Total cards in play: 54
```

Figure 16:how the round stopped

5.2.5 Test case 10

Table 10:Test case 10

Test Case	Scenario tested	Expected Output	Actual Output	Result
10	Decide The Winner	After the All the rounds that user wanted to play the player won the most rounds wins the whole game	After the All the rounds that user wanted to play the player won the most rounds wins the whole game	pass

```
---YOUR GAME HAS FINISHED---
Rounds played: 5
PC won 3 rounds
Human won 2 rounds

PC won the Game
```

Figure 17:Overall winner

6 References

W3 Schools (2019). *Python Functions*. [online] W3schools.com. Available at: https://www.w3schools.com/python/python_functions.asp

GeeksforGeeks (2016). *Python Modules*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/python-modules/

Python.org Available at: https://www.python.org/

Real Python Available at: https://realpython.com/