DOC 334 Computer Programming ICW Report

Module Code	: DOC334
Module Title	: Computer Programming
Module Leader	: Mr. Nishan Saliya
Assessment Type	: Individual Coursework
Issued Date	: 14th of March 2025
Submission Date	: 04th of April 2025
Submission Time	: 23:59 P. M.
Weight	: 50%

Student ID: 20240690
Student Name: P.H.N.Y. Mendis
Centre: Galle

I. Abstract

This report is based on the standard deck of cards-based card game and normal cards and joker cards according to the ICW specifications. This single user playing system target is to create a fully functional, user-friendly card game system following the rules of the game. The game system is developed using functions, variables, loops to handle the system functions. Main complex program in in the system is game play with shuffling, dealing and scoring processes. This report covers the background of the game system including architecture, design decisions and the system success proof. Game logic and error handling can be seen as the most significant features in the system. This project demonstrates the successful implementation of fundamental programming concepts I the development of an interactive card playing game.

II. Acknowledgements

I would like to show my sincere gratitude toward all who contributed to the success of this project. First, I want to thank my lecturers for their continuous guidance, friendly support and feedback throughout the course.

I also want to thank course instructors for providing the resources relevant to the ICW which helped me to understand the process clearly and finish it. A special thanks to my badge mates for sharing their thoughts and experiences which helped me to understand the subject material. Lastly, I am deeply grateful to my family and friends for their continuous love, affection and support which brought me to here.

III. Table of Contents

Table of Contents

1. Introduction

The overall objective of this independent coursework (ICW) is to learn and implement a suggested solution in detail. The task entails the use of ideas and concepts gained from the course and implementing them in designing, developing, and documenting a working existing solution to the requirements given. The report is a record of the entire process, ranging from preparation and study to the implementation and test process at the end.

In the following sections, we have given an overview of the key points of the project, problem statement, methodology, tools employed, and results. The aim is to give an overview of the scope, challenges, and achievements of the project.

Practical sessions for demonstrating the practice application of theory to actual life problems, such as problem-solving, analysis, and creative thinking, are also a part of this project. This is a project report in the format to provide an overall overview of the project development, methodologies employed, and achievements realized.

How important effective planning, implementation, and communication at different stages in every project are, is highlighted, and the expectation of gaining overall proficiency on the topic in real-life situations.

2. Game Overview

The game designed in this project is a card-based game which works according to the rules of the game and under the instructions of the ICW specifications. However, the game is engaging and challenging which has potential for both PC and player to win as this is a single player game model. This draws inspiration from traditional card games but with more interactive enhancements which make this game more enjoyable to the users.

This consists of maximum 5 and minimum 1 round which is 1 round on default. The card deck is a collection of cards, the same as in traditional method which has 2-10, J, Q, K, A and Joker which is categorized under the 3 icons which are called diamonds, clubs, spades not only in here also in the game too. The suits are not applicable to the Joker cards because those two cards are either black or colored.

The deck has 54 cards and distributes between the human and PC. Players should navigate rounds, managing their hand of cards while predicting and reacting to opponents' moves. As mentioned previously, the game divides into rounds which each player takes turns to draw, play or discard cards eventually.

The game's victory is under not only skills and intelligence but also under luck as the game competitiveness can be increased suddenly and turn to a war too. Tie is also a situation which can be happened due to the competitiveness of the game system which make the game more engaging to players.

Through this game, this project aims to analyze how traditional games can be enhanced by modern design principles, to create more engaging experience. The project focuses on the backend system, clear rules, and seamless gameplay which those players enjoy a smooth and immersive experience.

3. Algorithm Description

This system simulated a basic and interactive card game between player and PC. The game follows the steps of a traditional card game where players draw cards, compare values and collect winning cards. This game supports about 5 rounds and 1 round by default. The players can see their game results later because game result saves in both html and txt files with date time and other information.

1. Initialization of game setup with assets need

    - Create the deck of cards which includes 54 cards.
    - Shuffle the deck randomly
    - Create 2 players
    - Distribute cards among members
    - Shuffle each player's cards before starting.

2. Gameplay Loop

    The number of rounds playing should be specified before starting, otherwise the game will be played just one round by default as mentioned earlier.

    - Step 1: Players draw one card: The human and the PC play their top card simultaneously.

    - Step 2: Compare Cards
      The card values are compared using a predefined ranking system.
      The player with the higher-ranked card wins and takes both cards.
      If the cards are equal, a "War" will happen.

    - Step 3: War Handling (If Tie Occurs)
      Each player draws three more cards (if available).
      A fourth card is drawn and compared to determine the winner.
      If the fourth card is also a tie, the war continues.

    - Step 4: Repeat Until Decks Are Empty
      The round continues until one player runs out of cards.

3. End of Round Handling

    - Players merge their won cards back into their deck.
    - Decks shuffle before starting the next round.
    - Round history will save.

4. Game Finish

    - When all rounds are played, the game checks:

      Total cards in each player's possession.
      Number of wars fought.
      Determine Human, PC, or Tie as the winner.

5. Data Store & Output

    - Text File Output: Date, Time, Game history, Rounds played, Winner.
    - HTML Output: Date, Time, Game history, Rounds played, Winner.

The above algorithm about the project game system will help to understand how the process in backend works throughout the game playing from the beginning to end. The game is under the rules, so the game algorithm is also written under the rules of the game and the instructions of the coursework specifications.

4. Code Explanation

1) Card Functions
   Card creation and comparison:
   - create_card(value, suit): Returns card dictionary with value and suit
   - card_to_string(card): Prints card as string
   - compare_cards(card1, card2): Compares two cards by rank (returns 1 if card1 > card2, -1 if card1 < card2, 0 if equal)

2) Deck Functions
   The deck of cards:
   - build_deck(): Builds a standard 54 card deck
   - shuffle_deck(deck): Randomly shuffles the deck
   - deal_card(deck): Deals and returns top card of deck

3) Player Functions
   Player actions:
   - create_player(name): Creates a player dictionary with name, deck, and won cards
   - add_card_to_player(): Adds a card to player's deck
   - shuffle_player_deck(): Shuffles player's cards
   - play_card_from_player(): Plays top card from player's deck
   - win_cards_for_player(): Adds won cards to player's collection
   - total_player_cards(): Sum all cards a player owns

4) Game Functions
   Game functioning method:
   - initialize_game(rounds): Initializes game with players, shuffles and deals cards
   - play_round(game):
     Play round of the game
   - resolve_war():
     Special case implementation cards equal:
     Both players put down 3 face-down cards
     Compares 4th card to decide war winner
     Addresses repeat wars recursively
   - play_game():
     Coordinates the big picture game flow for each round
   - determine_winner():
     Compares total cards to determine winner

5) File Output Functions
   Write game result

- save_to_file():
  Saves game results to a text file
- save_to_html():
  Prints a HTML version of the results

6) Main Function
   Processes command prompt options (number of rounds)

   Initializes and plays the game
   Prints results into files

   Main Game Mechanisms:
   Card Rank: 2 (lowest) to A (high), Jokers (highest)

   Game Play:
   Players both flip top cards simultaneously
   Higher card wins both cards
   Tied cards lead to war

   War:
   Both players lay down 3 cards face down
   4th card declares the war winner
   Winner gets all cards in the war

   Winning:
   Player with most cards after all rounds wins

   Data Structures Used:
   Cards:
   Dictionaries with 'value' and 'suit' as keys

   Players:
   Dictionaries with name, deck, and won cards

   Game State:
   Dictionary maintaining players, rounds, and history

5. Pseudocode

```
#Initialize modules
import random
import sys
import os
from datetime import datetime

#Card functions
def create_card(value, suit=None): #to create a card
    return {'value': value, 'suit': suit}

def card_to_string(card): #convert card set to a string
    if card['suit']:
        return f"{card['value']}{card['suit']}"
    return f"{card['value']}"

#function to compare cards
def compare_cards(card1, card2):
    values_order = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A', 'Joker']
    idx1 = values_order.index(card1['value'])
    idx2 = values_order.index(card2['value'])
    if idx1 > idx2:
        return 1
    elif idx1 < idx2:
        return -1
    else:
        return 0

# Deck functions
def build_deck():
    suits = ['♥', '♦', '♣', '♠'] #hearts, diamonds, clubs, spades
    values = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']
    cards = []

    for suit in suits: #create 52 cards
        for value in values:
            cards.append(create_card(value, suit))

    # Add 2 Joker cards
    cards.append(create_card('Joker'))
    cards.append(create_card('Joker'))
    return cards

#shuffling all 54 cards together
def shuffle_deck(deck):
    random.shuffle(deck)

def deal_card(deck):
    if len(deck) > 0: #check if the deck has cards
        return deck.pop() #ensure cards not reused
```

```
    return None

# Player functions
def create_player(name): #create a player with empty deck and a pile called won cards
    return {
        'name': name,
        'deck': [],
        'won_cards': []
    }

def add_card_to_player(player, card): #add cards to a player's deck
    player['deck'].append(card)

def shuffle_player_deck(player): #shuffle a plyer's deck
    random.shuffle(player['deck'])

def play_card_from_player(player): #remove a card from player's deck function
    if len(player['deck']) > 0:
        return player['deck'].pop(0)
    return None #no cards left

def win_cards_for_player(player, cards): #add won cards to a player's pile
    player['won_cards'].extend(cards)

def total_player_cards(player): #count total cards one player has
    return len(player['deck']) + len(player['won_cards'])

#functions in game
def initialize_game(rounds):
    human = create_player("Human")
    pc = create_player("PC")
    war_count = 0 #war round counter
    game_history = [] #storing history of round
    full_history = [] #to store history of entire game

    #Build and shuffle deck
    deck = build_deck()
    shuffle_deck(deck)

    #Distribute cards evenly
    while len(deck) > 0:
        add_card_to_player(human, deal_card(deck))
        add_card_to_player(pc, deal_card(deck))

    #shuffling individual decks
    shuffle_player_deck(human)
    shuffle_player_deck(pc)

    return {
        'rounds': rounds,
```

```
        'human': human,
        'pc': pc,
        'war_count': war_count,
        'game_history': game_history,
        'full_history': full_history
    }

#function to play one round
def play_round(game):
    human_card = play_card_from_player(game['human'])
    pc_card = play_card_from_player(game['pc'])

    if human_card is None or pc_card is None:
        return False  #game over

    result = f"{card_to_string(human_card)} vs {card_to_string(pc_card)} - "
    comparison = compare_cards(human_card, pc_card)

    if comparison > 0:
        win_cards_for_player(game['human'], [human_card, pc_card])
        result += "H" #player win
    elif comparison < 0:
        win_cards_for_player(game['pc'], [human_card, pc_card])
        result += "P" #pc win
    else:
        #war!
        war_cards = [human_card, pc_card]
        result += "WAR"
        game['war_count'] += 1
        war_winner = resolve_war(game, war_cards)
        if war_winner:
            win_cards_for_player(war_winner, war_cards)

    game['game_history'].append(result)
    return True

#resolve the war
def resolve_war(game, war_cards):
    human = game['human']
    pc = game['pc']

    for _ in range(3): #drawing 3 additional cards
        h_card = play_card_from_player(human)
        p_card = play_card_from_player(pc)

        if h_card is None or p_card is None:
            return None #no cards to continue war then

        war_cards.extend([h_card, p_card])
```

```
    #compare the 4th card
    h_fourth = play_card_from_player(human)
    p_fourth = play_card_from_player(pc)

    if h_fourth is None or p_fourth is None:
        return None

    war_cards.extend([h_fourth, p_fourth])
    comparison = compare_cards(h_fourth, p_fourth)

    if comparison > 0:
        return human #player win the war
    elif comparison < 0:
        return pc #pc win the war
    else:
        #another war!!! (if tie)
        return resolve_war(game, war_cards)

def play_game(game):
    now = datetime.now() #get current date and time
    print(f"Date: {now.strftime('%Y-%m-%d')}")
    print(f"Time: {now.strftime('%H:%M')}")
    print(f"\nTotal Rounds: {game['rounds']}\n")

    for round_num in range(1, game['rounds'] + 1):
        print(f"\nRound {round_num} results")
        print("-------------")
        print("No : Hum vs PC - Winner")

        #continue playing rounds till one player's deck is empty
        while game['human']['deck'] and game['pc']['deck']:
            if not play_round(game):
                break

        #display result of a round
        for i, result in enumerate(game['game_history'], 1):
            print(f"{i}: {result}")

        #store full history with round number
        game['full_history'].append((round_num, game['game_history'].copy()))

        #reset for next round if want
        if round_num < game['rounds']:
            #combineing won cards again to deck
            game['human']['deck'].extend(game['human']['won_cards'])
            game['pc']['deck'].extend(game['pc']['won_cards'])
            game['human']['won_cards'] = []
            game['pc']['won_cards'] = []
            shuffle_player_deck(game['human'])
            shuffle_player_deck(game['pc'])
```

```
        game['game_history'] = []
        game['war_count'] = 0

#determining winner by comparing all the cards of both players.
def determine_winner(game):
    human_total = total_player_cards(game['human']) #get total cards from player
    pc_total = total_player_cards(game['pc']) #get total cards for pc

    #display final result counts
    print(f"\nPC card count {pc_total}")
    print(f"Human card count {human_total}")
    print(f"War count {game['war_count']}")

    #determine winner and display
    if human_total > pc_total:
        print("\nHuman won the game!")
        return "Human"
    elif pc_total > human_total:
        print("\nPC won the game!")
        return "PC"
    else:
        print("\nThe game is a tie!")
        return "Tie"
#save result to txt file
def save_to_file(game):
    now = datetime.now()
    random_num = random.randint(1000, 9999)
    filename = f"{now.strftime('%Y%m%d_%H-%M')}_{random_num}.txt"

    with open(filename, 'w', encoding='utf-8') as f:
        #writing date,time, total round amount
        f.write(f"Date: {now.strftime('%Y-%m-%d')}\n")
        f.write(f"Time: {now.strftime('%H:%M')}\n\n")
        f.write(f"Total Rounds: {game['rounds']}\n\n")

        #write result for every round
        for round_num, round_history in game['full_history']:
            f.write(f"Round {round_num} results\n")
            f.write("-------------\n")
            f.write("No : Hum vs PC - Winner\n")
            for i, result in enumerate(round_history, 1):
                f.write(f"{i}: {result}\n")
            f.write("\n")

        human_total = total_player_cards(game['human'])
        pc_total = total_player_cards(game['pc'])

#write results
        f.write(f"PC card count {pc_total}\n")
        f.write(f"Human card count {human_total}\n")
```

```
        f.write(f"War count {game['war_count']}\n\n")

    #declare winner
        if human_total > pc_total:
            f.write("Human won the game!\n")
        elif pc_total > human_total:
            f.write("PC won the game!\n")
        else:
            f.write("The game is a tie!\n")

        #now generate HTML version
        html_file = save_to_html(game, filename)
        return filename, html_file

#saving result in html page
def save_to_html(game, txt_filename):
    """Generate an HTML version of the game results"""
    html_filename = os.path.splitext(txt_filename)[0] + ".html"

    with open(html_filename, 'w', encoding='utf-8') as f: #html structure basic one
        f.write(f"""<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Game Results</title>
    <style>
        body {{ font-family: Arial, sans-serif; }}
        .round {{ margin-top: 20px; }}
        .round-header {{ font-weight: bold; }}
        .results {{ margin-left: 20px; }}
    </style>
</head>
<body>
    <p>Date: {datetime.now().strftime('%Y-%m-%d')}</p>
    <p>Time: {datetime.now().strftime('%H:%M')}</p>
    <p><br><br>Total Rounds: {game['rounds']}<br><br></p>
""")

        #add round results
        for round_num, round_history in game['full_history']:
            f.write(f"""
    <div class="round">
        <p class="round-header">Round {round_num} Results</p>
        <pre class="results">
""")
            for i, result in enumerate(round_history, 1):
                f.write(f"{i}: {result}\n")
            f.write("""
        </pre>
```

```
    </div>
""")

        #add summary
        human_total = total_player_cards(game['human'])
        pc_total = total_player_cards(game['pc'])

        f.write(f"""
    <pre>
PC card count: {pc_total}
Human card count: {human_total}
War count: {game['war_count']}
""")

        #add winner
        if human_total > pc_total:
            f.write("\nHuman won the game!\n")
        elif pc_total > human_total:
            f.write("\nPC won the game!\n")
        else:
            f.write("\nThe game is a tie!\n")

        f.write("""</pre>
</body>
</html>
""")

    return html_filename

def main():
    #command line arguments
    rounds = 1  #default 1
    if len(sys.argv) > 1:
        try:
            rounds = int(sys.argv[1])
            if rounds < 1 or rounds > 5:
                print("Invalid number of rounds. Must be between 1 and 5. Using default (1 round).")
                rounds = 1
        except ValueError:
            print("Invalid argument. Using default (1 round).")

    #functions instialization
    game = initialize_game(rounds)
    play_game(game)
    winner = determine_winner(game)
    save_to_file(game)

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

6. Test Cases

i. Screenshots

```
C:\Users\novan\Desktop>python war.py
Date: 2025-04-03
Time: 12:37

Total Rounds: 1


Round 1 results
-------------
No : Hum vs PC - Winner
1: A♦ vs 8♥ - H
2: 10♣ vs 3♣ - H
3: A♣ vs 3♥ - H
4: 9♣ vs 2♦ - H
5: 8♠ vs Q♦ - P
6: 10♦ vs 7♥ - H
7: 6♦ vs 8♣ - P
8: Joker vs 3♦ - H
9: 4♦ vs J♣ - P
10: 5♠ vs 9♥ - P
11: 4♣ vs 7♠ - P
12: 6♠ vs A♥ - P
13: 7♦ vs J♦ - P
14: 2♥ vs Q♣ - P
15: K♠ vs 4♥ - H
16: 5♥ vs 8♦ - P
17: Q♥ vs J♠ - H
18: 7♣ vs 4♠ - H
19: Joker vs 9♦ - H
20: K♦ vs 6♥ - H
21: J♥ vs 2♣ - H
22: 10♠ vs 5♣ - H
23: 9♠ vs 5♦ - H
24: 6♣ vs 3♠ - H
25: A♠ vs K♣ - H
26: Q♠ vs K♥ - P
27: 2♠ vs 10♥ - P

PC card count 22
Human card count 32
War count 0

Human won the game!

C:\Users\novan\Desktop>
```

```
C:\Users\novan\Desktop>python war.py 3
Date: 2025-04-03
Time: 12:45

Total Rounds: 3


Round 1 results
-------------
No : Hum vs PC - Winner
1: 2♥ vs 2♣ - WAR
2: 10♦ vs 2♦ - H
3: 8♥ vs 9♠ - P
4: K♦ vs K♥ - WAR
5: K♣ vs 3♠ - H
6: J♥ vs 9♦ - H
7: J♣ vs 6♦ - H
8: 3♦ vs A♣ - P
9: Joker vs 10♠ - H
10: 3♣ vs 9♣ - P
11: A♥ vs 10♥ - H
12: 7♠ vs 6♥ - H
13: 5♦ vs 9♥ - P
14: 8♠ vs 4♦ - H
15: 2♠ vs 5♣ - P
16: 3♥ vs Q♠ - P
17: 10♣ vs 4♥ - H
18: A♦ vs 4♣ - H
19: 8♣ vs 8♦ - WAR
```

```
Round 2 results
-------------
No : Hum vs PC - Winner
1: 9♦ vs 9♠ - WAR
2: J♣ vs 8♥ - H
3: 4♥ vs K♥ - P
4: 4♦ vs A♣ - P
5: J♥ vs 5♣ - H
6: 10♣ vs Q♥ - P
7: 2♦ vs 2♣ - WAR
8: A♥ vs 3♦ - H
9: 6♥ vs 5♥ - H
10: 10♥ vs 3♣ - H
11: 10♠ vs J♦ - P
12: 3♠ vs Q♠ - P
```

```
Round 3 results
-------------
No : Hum vs PC - Winner
1: 3♦ vs A♣ - P
2: 10♦ vs 6♣ - H
3: 7♣ vs 9♠ - P
4: 5♣ vs 2♥ - H
5: Joker vs 6♠ - H
6: J♥ vs 5♠ - H
7: 2♣ vs 3♠ - P
8: 2♦ vs 4♦ - P
9: A♠ vs K♥ - H
10: 6♥ vs 6♦ - WAR
11: 7♠ vs Q♣ - P
12: A♥ vs 2♠ - H
13: 8♥ vs Q♠ - P
14: 10♥ vs 7♥ - H
15: 4♠ vs K♣ - P
16: 3♣ vs Q♦ - P

PC card count 38
Human card count 14
War count 1

PC won the game!

C:\Users\novan\Desktop>
```

```
C:\Users\novan\Desktop>python war.py 9
Invalid number of rounds. Must be between 1 and 5. Using default (1 round).
Date: 2025-04-03
Time: 13:01
```

20250403_13-01_8709	4/3/2025 1:01 PM	Microsoft Edge HT...	2 KB
20250403_13-01_8709	4/3/2025 1:01 PM	Text Document	1 KB

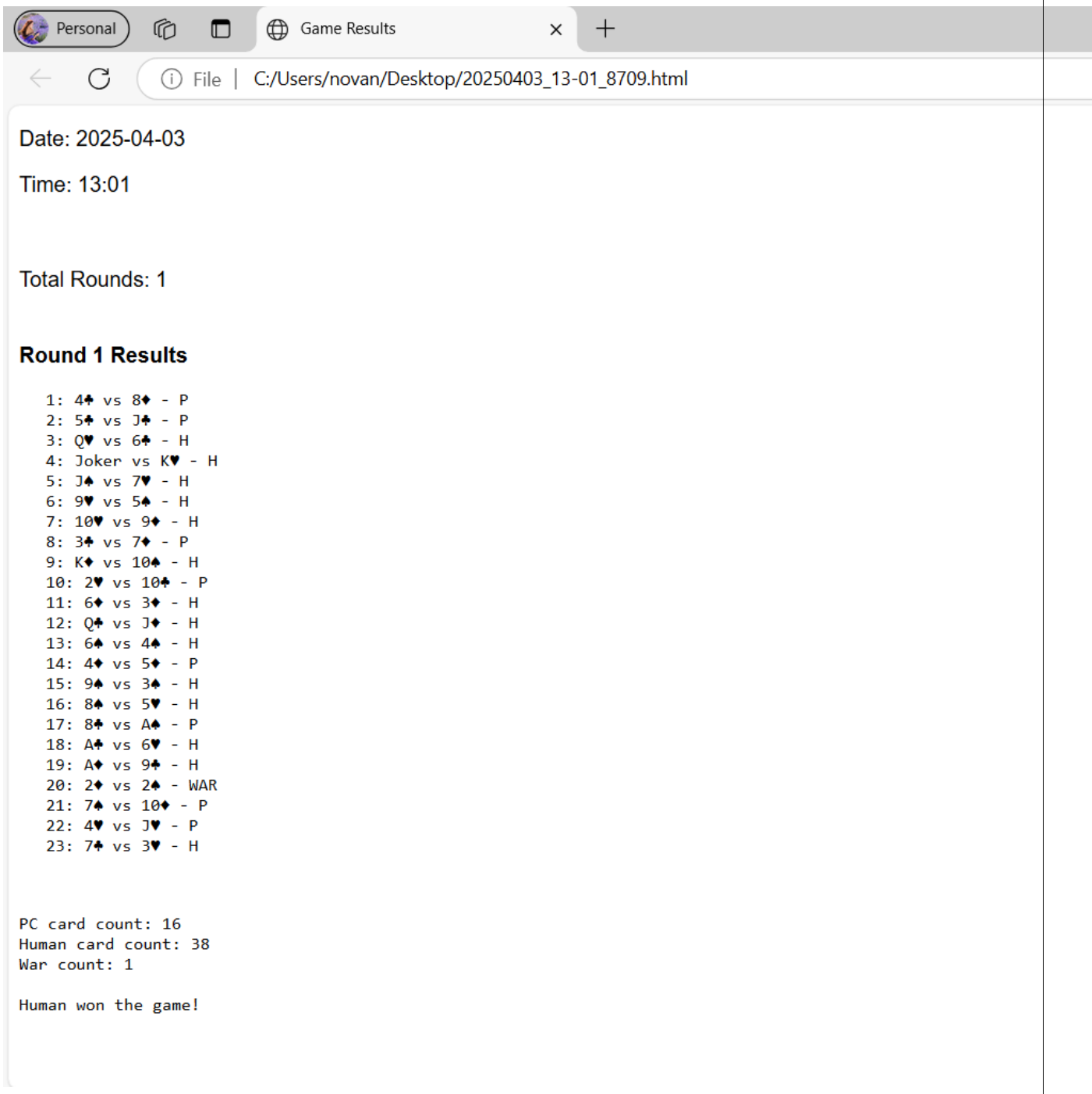
```
Date: 2025-04-03
Time: 13:01

Total Rounds: 1

Round 1 results
-------------
No : Hum vs PC - Winner
1: 4♠ vs 8♦ - P
2: 5♠ vs J♠ - P
3: Q♥ vs 6♠ - H
4: Joker vs K♥ - H
5: J♠ vs 7♥ - H
6: 9♥ vs 5♠ - H
7: 10♥ vs 9♦ - H
8: 3♠ vs 7♦ - P
9: K♦ vs 10♠ - H
10: 2♥ vs 10♠ - P
11: 6♦ vs 3♦ - H
12: Q♠ vs J♦ - H
13: 6♠ vs 4♠ - H
14: 4♦ vs 5♦ - P
15: 9♠ vs 3♠ - H
16: 8♠ vs 5♥ - H
17: 8♠ vs A♠ - P
18: A♠ vs 6♥ - H
19: A♦ vs 9♠ - H
20: 2♦ vs 2♠ - WAR
21: 7♠ vs 10♦ - P
22: 4♥ vs J♥ - P
23: 7♠ vs 3♥ - H

PC card count 16
Human card count 38
War count 1

Human won the game!
```


Personal
Game Results
File | C:/Users/novan/Desktop/20250403_13-01_8709.html
Date: 2025-04-03
Time: 13:01
Total Rounds: 1
Round 1 Results
1: 4♣ vs 8♦ - P
2: 5♣ vs J♣ - P
3: Q♥ vs 6♣ - H
4: Joker vs K♥ - H
5: J♠ vs 7♥ - H
6: 9♥ vs 5♠ - H
7: 10♥ vs 9♦ - H
8: 3♣ vs 7♦ - P
9: K♦ vs 10♠ - H
10: 2♥ vs 10♣ - P
11: 6♦ vs 3♦ - H
12: Q♣ vs J♦ - H
13: 6♠ vs 4♠ - H
14: 4♦ vs 5♦ - P
15: 9♠ vs 3♠ - H
16: 8♠ vs 5♥ - H
17: 8♣ vs A♠ - P
18: A♣ vs 6♥ - H
19: A♦ vs 9♣ - H
20: 2♦ vs 2♠ - WAR
21: 7♠ vs 10♦ - P
22: 4♥ vs J♥ - P
23: 7♣ vs 3♥ - H
PC card count: 16
Human card count: 38
War count: 1
Human won the game!

ii. Tables

Test Case Number	Input	Expected Output	Details of the Output	Result
01	python war.py	Code runs 1 time and result display	Code ran 1 time and result displayed successfully	Pass
02	python war.py 3	Code run 3 times and result display	Code run 3 times and display result successfully	Pass
03	python war.py 3	Code run 3 times and result display	Code run 3 times and display result successfully	Pass
04	python war.py 3	Code run 3 times and result display	Code run 3 times and display result successfully	Pass
05	python war.py 9	Code display error massage	Code displayed error message as expected	Pass
06	File naming	File name with date, time and 4 random numbers.	File name with date, time and 4 random numbers.	Pass
07	Text file result display	Result display in txt file as in specifications	Result displayed in txt file as in specifications	Pass
08	Html file creation	Result displayed in html file as in specifications	Result displayed in html file as in specifications	pass

7. Limits and Future Developments

i. Limitations:

The facts which limit the project's success

- Scalability constraints: the current system only have option to single user play.
- Limited features: as the game developed under the ICW specifications, user interaction is strictly limited.
- Hardware/software dependencies: the system may require specific hardware or software configurations which limit its adaptability to different environments.
- User-friendly interfaces: as the game is console-based game, Gui usage is very low which lowers the game excitement.

ii. Future Enhancements:

some identical features can address these limitations and improve quality within the game,

- Optimizing large scale development: enhancing system performance to accept single user play as well as multiuser play also. to handle large-scale datasets too.
- Additional features: implementing advanced functions like ai powered automation, security developments, analytics and sound system.
- Cross platform suitability: ensuring the system is fully compatible with different kinds of devices and systems.
- User interface enhancements: improving and designing graphical user interface for the game which makes the game more interactive and engaging.

By including those enhancements to the system, the game can revolve into a user-friendly, scalable and comfortable solution in future.

8. Conclusion

This course work project provides an opportunity for students to apply their theoretical knowledge into their practical life to ensure that they will adapt to the new technological environment and their future industry processes. This project covers all the specific stages in system development such as requirements analysis, system design, implementation, testing, evaluation and maintenance in a structured way. The project report covers all the important features of the system such as algorithm, code explanation, game overview, testing and results. Also, this project ensures to reveal limitations of the project as well as the future enhancements to make sure the game is open for developers. testing confirms that system functionality, performance and reliability is at a high level. From this project, the technological skills, problem solving skills and project management skills improvement is without any doubt. The experience gains from this project reinforced the importance of effective planning, implementation and testing in any software development project. Finally, it can be considered that this project course work has been enriching experience and equipping foundation to future research and development in real world applications. The knowledge gained project this system development will be a great support to the degree program challenges and also after that.

9. References

W3Schools.com (no date). https://www.w3schools.com/python/.

Shuffling a list of objects (no date). https://stackoverflow.com/questions/976882/shuffling-a-list-of-objects.

W3Schools.com (no date b). https://www.w3schools.com/python/python_file_handling.asp.

Python File Handling Archives (2022). https://pynative.com/python/file-handling/.

Python File Handling Archives (2022). https://pynative.com/python/file-handling/.