Assignment: Portfolio Project

Introduction

In this assignment, you'll apply the Python programming skills and DSA concepts you've learned throughout this course to build a complete, playable implementation of the classic card game of your choosing. Select from https://bicyclecards.com/. This project brings together multiple important concepts—from implementing shuffle algorithms and search techniques to designing object-oriented class structures and managing game state. You'll create a command-line application where players can compete against a computer opponent. By completing this project, you'll demonstrate your ability to translate theoretical computer science concepts into practical, working software.

Learning Objectives

- Implement a card shuffling algorithm (Fisher-Yates shuffle)
- Implement search algorithms for card matching
- Practice working with data structures
- Build a complete interactive program with game logic

Project Requirements

1. Card and Deck Implementation

- Create a `Card` class with suit and rank attributes
- Implement a `Deck` class that contains a standard 52-card deck
- Create a Fisher-Yates shuffle algorythm method for shuffling the deck

2. Player Implementation

- Design a `Player` class (for both human and computer players)
- Implement methods to:
    - Add cards to hand
    - Remove cards from hand
    - Search for matching cards in hand
    - Form and count "books" (sets of 4 matching cards)

3. Game Logic

- Implement the rules of Card Game:
    - Deal required number of cards to each player initially (2 players)
    - Players take turns
    - If opponent has requested cards, they must give them all

    - If not, player draws from the deck
    - Game Ends when player wins, loses, or quit
    - Win Condition
    - Lose Condition

4. Algorithm Requirements

- Implement a sorting algorithm of your choice to sort the hunan and ai player's hands
- Implement a search algorithm to search for cards in sorted hands
- Implement the Fisher-Yates shuffle algorithm

5. Computer AI

- The computer opponent should choose a random card from it's hand

6. User Interface

- Create a text-based interface in the command line
- Display clear game state information
- Show the player's hand, number of books, and remaining cards in deck
- Provide win/loss notification at the end

7. Provide documentation

- README.md includes:
    - About Game
    - Summary Rules and Logic
    - Project Structure
    - How to Run Game
    - Feature implemented
    - Future Features you would implement
    - Libraries Used
    - Pictures of Examples Outputs
    - Other relative

Examples Output of Go Fish

Player's turn

```
Player 1's turn!
Player 1's score: 0
Your hand:
2 of Diamonds
3 of Clubs
8 of Clubs
9 of Spades
A of Clubs
J of Clubs
Q of Spades
Ask for a rank: Q
Player 1 asks Computer for Qs.
Computer gives 1 Q(s) to Player 1.
```

Computer's turn

```
Computer's turn!
Computer's score: 0
Computer's hand is hidden.
Computer is thinking...
Computer asks Player 1 for 8s.
Player 1 says 'Go Fish!'
```

Game Over

```
Game over! Player 1 wins with 7 books!
Player 1: 7 books
Computer: 6 books
```

Example project structure

```
go_fish/
├── README.md
├── main.py
├── card.py
├── deck.py
├── player.py
└── game.py
```