Shih Chi Han
CS5010
P1

Texas Hold'em Poker Game(OOP)

Project Overview

This project aims to develop a Texas Hold'em Poker Game App where players can compete against both human and computer opponents while managing their accounts and tracking their game history. The focus of this project is not just on game mechanics but primarily on demonstrating Object-Oriented Programming (OOP) pillars, principles, and design patterns in a structured and scalable manner.

Business Requirements

Core Functionality

- Player Management
  - Users should be able to create a player account using email, password, and birthday, and log in to save progress.
  - Players should have a starting chip balance that updates after each game.
  - Players should be able to view their win/loss history with past game results.
  - The system should dynamically create players (human or AI) based on user choices.
- Game Setup & Customization
  - Users should be able to choose the number of computer opponents before starting the game.
  - Each computer player should have a distinct difficulty level or betting strategy(aggressive, cautious, balanced) assigned by the system using a strategy-based decision system.
  - The game should randomly select a dealer at the beginning of each round.
- Turn-Based Gameplay
  - Players take turns performing actions such as betting, checking, folding, or calling.
  - The system should handle betting rounds including pre-flop, flop, turn, and river.
  - The game should automatically proceed to the next phase after all players have acted.
- Hand Evaluation & Winner Declaration
  - The system should automatically evaluate hands and determine the winner.

- - Players should be able to view all final hands and understand the winning hand.
    - The winning player should receive the pot while others lose their bets.
- Account & History Tracking
    - Account should have a game history log showing previous matches.
    - The system should notify the account manager whenever a player wins or loses.
    - The system should maintain a leaderboard ranking that ranks top players by total chips.

Target Audience
- Players aged 21+ who are interested in Texas Hold'em Poker Game.
- Player Types:
    - Beginner: New to Texas Hold'em, prefers easy AI opponents.
    - Casual: Plays for fun, prefers a balanced AI experience.
    - Professional: Skilled players who want a challenging AI.

Nouns and Verbs Extraction
Nouns:
- Users
- Players (Beginner, Casual, Professional)
- Player Accounts(email, password, and birthday)
- Chip Balance
- Game
- Progress
- Win/Loss History
- Past Game Results
- Computer player
- Difficulty Level
- Betting Strategy (Aggressive, Cautious, Balanced)
- Dealer
- Round
- Turn
- Actions (betting, checking, folding, calling)
- Betting Rounds
- Pre-flop, Flop, Turn, River
- Next Phase
- Hands
- Final Hands
- Winning Hand

- Pot
- Bets
- Game History Log
- Previous Matches
- Account Manager
- Leaderboard
- Ranking
- Total Chips

- Create
- Log in
- Save
- Update
- View
- Choose
- Start
- Select
- Assign
- Take turns
- Perform
- Bet
- Check
- Fold
- Call
- Handle
- Proceed
- Evaluate
- Determine
- Receive
- Lose
- Show
- Notify
- Maintain
- Rank

Challenge Questions
- Can players customize the number of computer opponents and difficulty?
- Can players change the strategy of the computer opponents?
- Can players view their game history to know the amount they won or lost in previous games?

Summary of Classes, Attributes and Associations

- Player
    - Attributes: name, chipBalance, hand, isDealer
- HumanPlayer
    - Methods: makeDecision()
- ComputerPlayer
    - Attributes: difficultyLevel, bettingStrategy
- BettingStrategy (Strategy Pattern)
    - AggressiveBetting, CautiousBetting, BalancedBetting
- PlayerFactory (Factory Pattern)
- Game
    - Attributes: players, deck, communityCards, pot, currentDealer, gamePhase
- GameState
    - Attributes: currentGame, roundNumber, activePlayers
- Card
    - Attributes: suit, rank
- Deck
    - Attributes: cards
- BettingRound
    - Attributes: currentBet, playersInRound
- GameHistory
    - Attributes: gameID, dateTime, numPlayers, finalWinLoss
- PlayerAccount
    - Attributes: name, email, password, birthday, history
- AccountManager
    - Attributes: accounts

- Leaderboard
    - Attributes: playerRanks

Relationships

- Association:
    - GameRoom - Game: One GameRoom creates and configures 1 Game per session.
    - ComputerPlayer - BettingStrategy: Each ComputerPlayer has 1 BettingStrategy(one of the types).
- Composition:
    - Game - Deck: One Game composes 1 Deck.

- ○ Game - Player: One Game contains 2..* Player(HumanPlayer and ComputerPlayer).
  - ○ Game - BettingRound: One Game contains multiple betting rounds.
  - ○ PlayerAccount - GameHistory: One PlayerAccount contains 0..* GameHistory records.
  - ○ Deck - Card: One Deck is composed of exactly 52 Card objects.
- ● Aggregation
  - ○ HumanPlayer - PlayerAccount: Each HumanPlayer is connected with 1 PlayerAccount
  - ○ AccountManager - PlayerAccount: One AccountManager manages 0..* PlayerAccount objects.
  - ○ Leaderboard - PlayerAccount: The Leaderboard aggregates 0..* PlayerAccount objects to compute rankings
- ● Inheritance
  - ○ HumanPlayer and ComputerPlayer extend Player
  - ○ AggressiveBetting, CautiousBetting, and BalancedBetting extend BettingStrategy

Dimensions
- ● Skill Level(Beginner, Casual, Professional): Determines the difficulty of the AI opponent.
- ● Engagement Level: How often the user plays and their purpose

User Personas
- ● Emily Carter (Beginner, Plays Occasionally, Learning the game)
  - ○ Age: 23
  - ○ Occupation: College Student
  - ○ Background:
    - ■ Recently started playing Texas Hold'em after watching a tournament on TV.
    - ■ Has played poker casually with friends but lacks confidence in real games.
    - ■ Prefers a low-pressure learning environment where he can improve at his own pace.
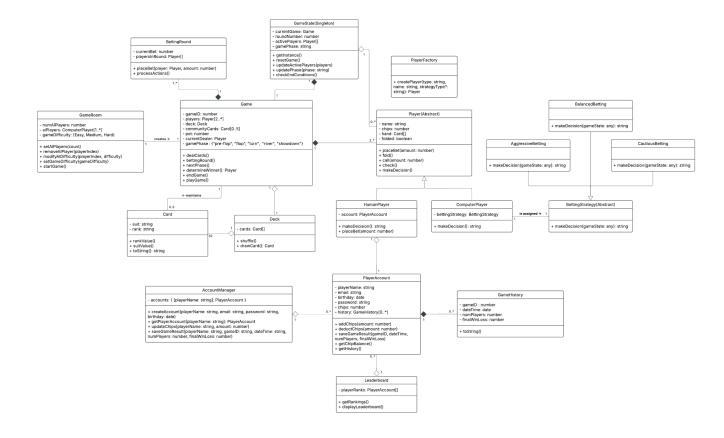 - ○ Needs:
    - ■ A beginner-friendly AI that plays predictably and allows him to practice decision-making.
    - ■ A simple game history to track wins and losses.
    - ■ A way to gradually increase the challenge by adjusting AI difficulty.

- James Thompson (Professional, Plays frequently, Seeks high-level competition)
  - Age: 35
  - Occupation: Financial Analyst
  - Background:
    - Plays poker as a serious hobby and enjoys competition.
    - Competes in online poker tournaments and aspires to play in live events.
    - Has a strong understanding of probabilities, bluffing, and game theory.
  - Needs:
    - An AI opponent that mimics real players using different betting strategies.
    - A ranking system to see how his rank is.
    - A player account that can save his chip balance and game history

User Stories

1. Emily Carter (Beginner, Plays Occasionally, Learning the game)
   1. As a beginner player, I want to choose an easy AI opponent so that I can practice without feeling overwhelmed.
   2. As a player, I want to check my past game results so that I can keep track of the amount I won or lost in previous games
   3. As a player, I want to adjust the AI difficulty so that I can gradually improve my skills and face stronger opponents.
2. James Thompson (Professional, Plays frequently, Seeks high-level competition)
   1. As an advanced player, I want to play against AI players that mimics real players, with different skill levels and strategies, so that I can improve my decision-making under realistic conditions.
   2. As a competitive player, I want to see my overall ranking in all players so that I can compare my performance with other players.
   3. As a player who plays frequently, I want to have a player account that saves my progress so that I can keep my chip balance and game history.

UML Class Diagram

BettingRound
- currentBet: number
- playersInRound: Player[]
+ placeBet(player: Player, amount: number)
+ processActions()

GameState (Singleton)
- currentGame: Game
- roundNumber: number
- activePlayers: Player[]
- gamePhase: string
+ getInstance()
+ resetGame()
+ updateActivePlayers(players)
+ updatePhase(phase: string)
+ checkEndConditions()

PlayerFactory
+ createPlayer(type: string, name: string, strategyType?: string): Player

GameRoom
- numAIPlayers: number
- aiPlayers: ComputerPlayer[1..*]
- gameDifficulty: {Easy, Medium, Hard}
+ setAIPlayers(count)
+ removeAIPlayer(playerIndex)
+ modifyAIDifficulty(playerIndex, difficulty)
+ setGameDifficulty(gameDifficulty)
+ startGame()

Game
- gameID: number
- players: Player[2..*]
- deck: Deck
- communityCards: Card[0..5]
- pot: number
- currentDealer: Player
- gamePhase : {"pre-flop", "flop", "turn", "river", "showdown"}
+ dealCards()
+ bettingRound()
+ nextPhase()
+ determineWinner(): Player
+ endGame()
+ playGame()

Player (Abstract)
- name: string
- chips: number
- hand: Card[]
- folded: boolean
+ placeBet(amount: number)
+ fold()
+ call(amount: number)
+ check()
+ makeDecision()

BalancedBetting
+ makeDecision(gameState: any): string

AggressiveBetting
+ makeDecision(gameState: any): string

CautiousBetting
+ makeDecision(gameState: any): string

Card
- suit: string
- rank: string
+ rankValue()
+ suitValue()
+ toString(): string

Deck
- cards: Card[]
+ shuffle()
+ drawCard(): Card

HumanPlayer
- account: PlayerAccount
+ makeDecision(): string
+ placeBet(amount: number)

ComputerPlayer
- bettingStrategy: BettingStrategy
+ makeDecision(): string

BettingStrategy (Abstract)
+ makeDecision(gameState: any): string

is assigned →

AccountManager
- accounts: { [playerName: string]: PlayerAccount }
+ createAccount(playerName: string, email: string, password: string, birthday: date)
+ getPlayerAccount(playerName: string): PlayerAccount
+ updateChips(playerName: string, amount: number)
+ saveGameResult(playerName: string, gameID: string, dateTime: string, numPlayers: number, finalWinLoss: number)

PlayerAccount
- playerName: string
- email: string
- birthday: date
- password: string
- chips: number
- history: GameHistory[0..*]
+ addChips(amount: number)
+ deductChips(amount: number)
+ saveGameResult(gameID, dateTime, numPlayers, finalWinLoss)
+ getChipBalance()
+ getHistory()

GameHistory
- gameID : number
- dateTime: date
- numPlayers: number
- finalWinLoss: number
+ toString()

Leaderboard
- playerRanks: PlayerAccount[]
+ getRankings()
+ displayLeaderboard()

← maintains

creates →

Interface Low Level Mockups

AI Usage
- ChatGPT (4o)
  - Used for project overview and core functionality
    - Prompt:
      For the project, I'd like to build a Texas Hold'em Poker Game where players can compete against both human and computer opponents. The game follows standard Texas Hold'em rules. Players can choose how many computer opponents to include, and the system simulates their actions. Additionally, the users have their account allowing them to save their chip balance and track their win/loss history across multiple sessions.
      Features:
      - The user can decide how many computer players to include before starting the game.
      - Players have accounts where their chip balance and game history are saved.

- Players can keep track of and view their win or loss amount history on each game.
- Players take turns betting, checking, folding, or calling.
- The game reveals the flop, turn, and river cards according to Texas Hold'em rules.
- The system automatically evaluates hands and determines the winner at showdown.

Above is my project idea, create an overview and business requirements as bullet points.

- ○ Used for user dimensions
  - ■ Prompt: What are some dimensions that classify the users
- ○ Used for personas and user stories
  - ■ Prompt: Create 2 User personas and 3 user stories per persona