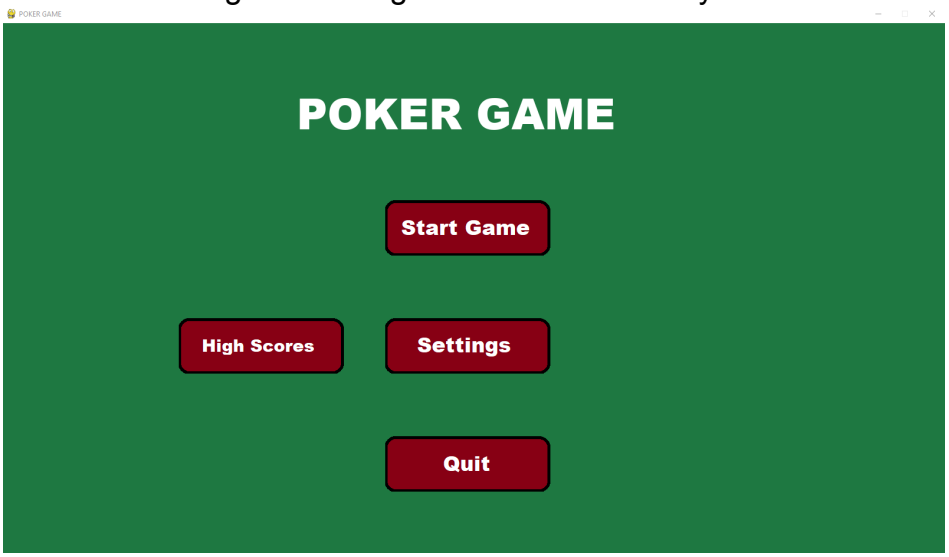
CS 4398
Software Engineering Project

TEXAS HOLD'EM POKER GAME

I. Test Cases

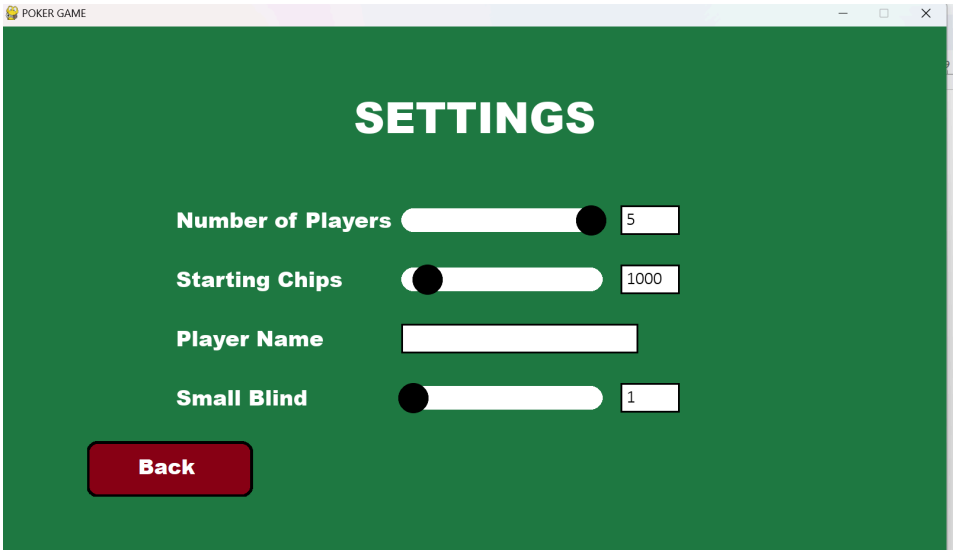
A. Game Initialization:

- Verify that the game initializes without errors.
- Check that the main menu is displayed upon initialization.
- Ensure that the game settings are loaded correctly.



B. Menu Interaction:

- Click on the "Start" button and verify that the game transitions to the in-game state.
- Click on the "Settings" button and verify that the game transitions to the settings state.
- Click on the "High Scores" button and verify that the game transitions to the high scores state.
- Click on the "Quit" button and verify that the game exits without errors.



C. Game Initialization with Settings:
Start a game from the main menu and verify that it initializes with the correct settings (number of players, starting chips, small blind).

D. In-Game Actions:

- During gameplay, click on the "Check" button and verify that the player's action is processed correctly.
- Click on the "Call" button and verify that the player's action is processed correctly.
- Enter a bet amount in the textbox and click on the "Bet" button, then verify that the player's bet is processed correctly.
- Click on the "Fold" button and verify that the player's action is

processed correctly.

E. Game Progression:

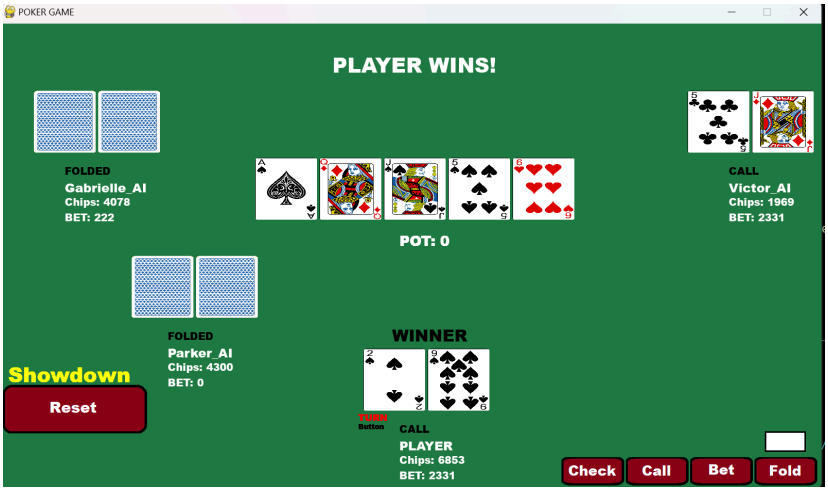
- Test the progression of the game through different stages (pre-flop, flop, turn, river) and verify that it advances correctly.
- Verify that the game updates the pot size and player actions accordingly during each stage.

At this point in the game, it has progressed through the pre-flop, flop, and is now at the *deal turn* cards stage.

F. Winning Conditions:

- Test scenarios where a player wins the round (by having the best hand or all other players folding).
- Verify that the game correctly identifies the winner(s) and

distributes the pot accordingly.



G. Game Termination:

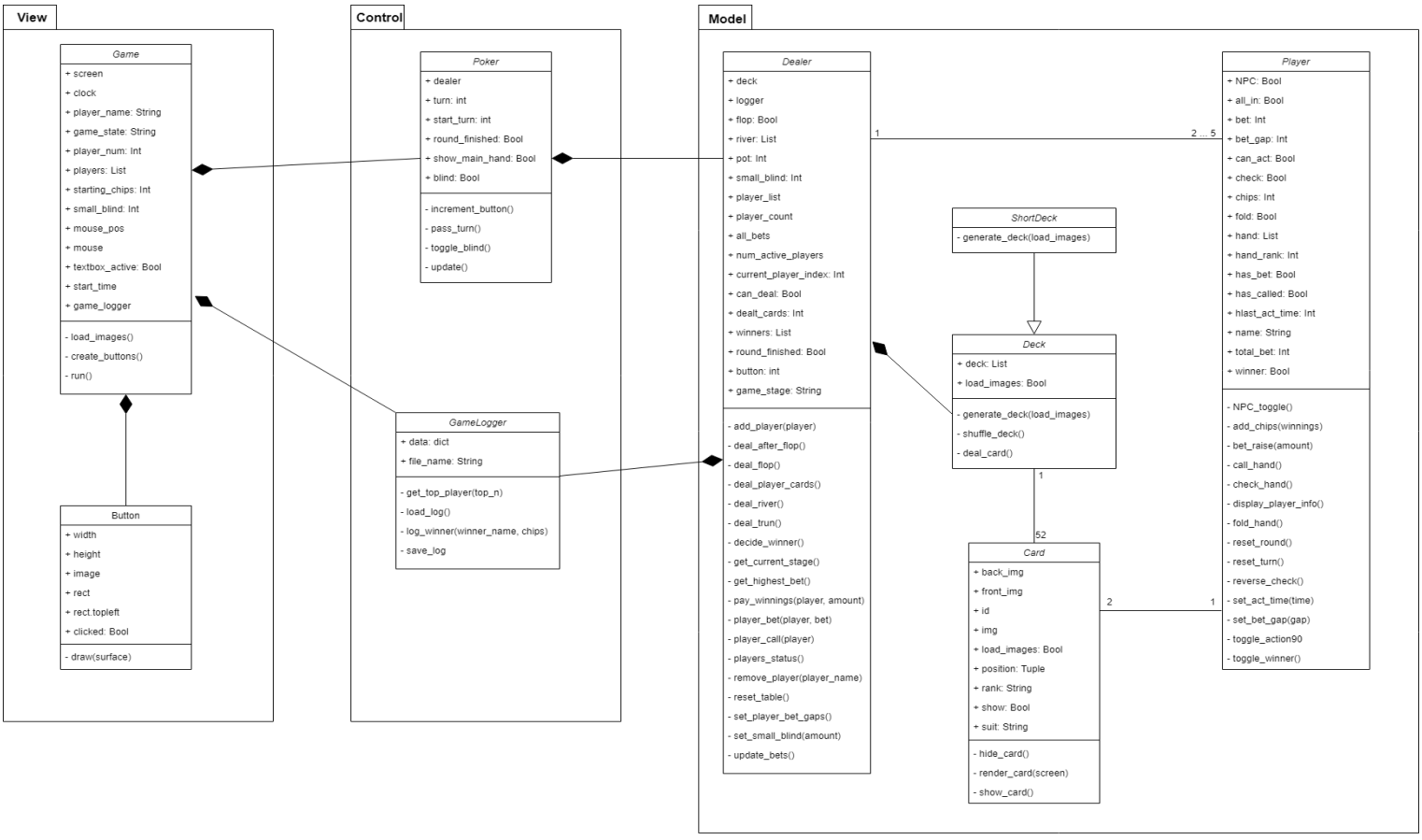
- Test quitting the game from the main menu and in-game.
- Verify that the game terminates without errors.

  The game exits as expected from the main screen without errors

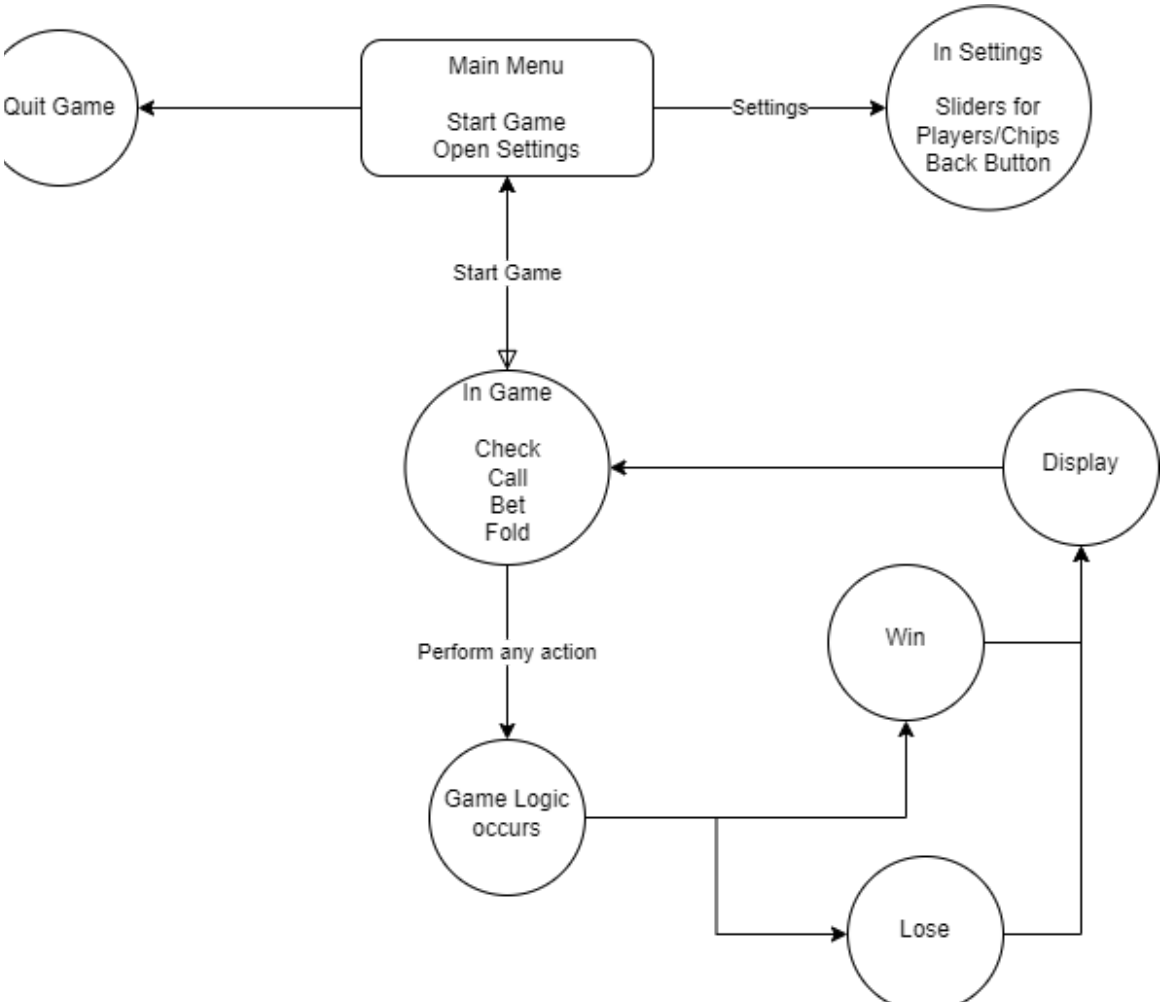
H. Performance:

  - Test the game's performance under various conditions (e.g., different screen resolutions, number of players).
  - Ensure that the game maintains a stable frame rate and responsiveness.

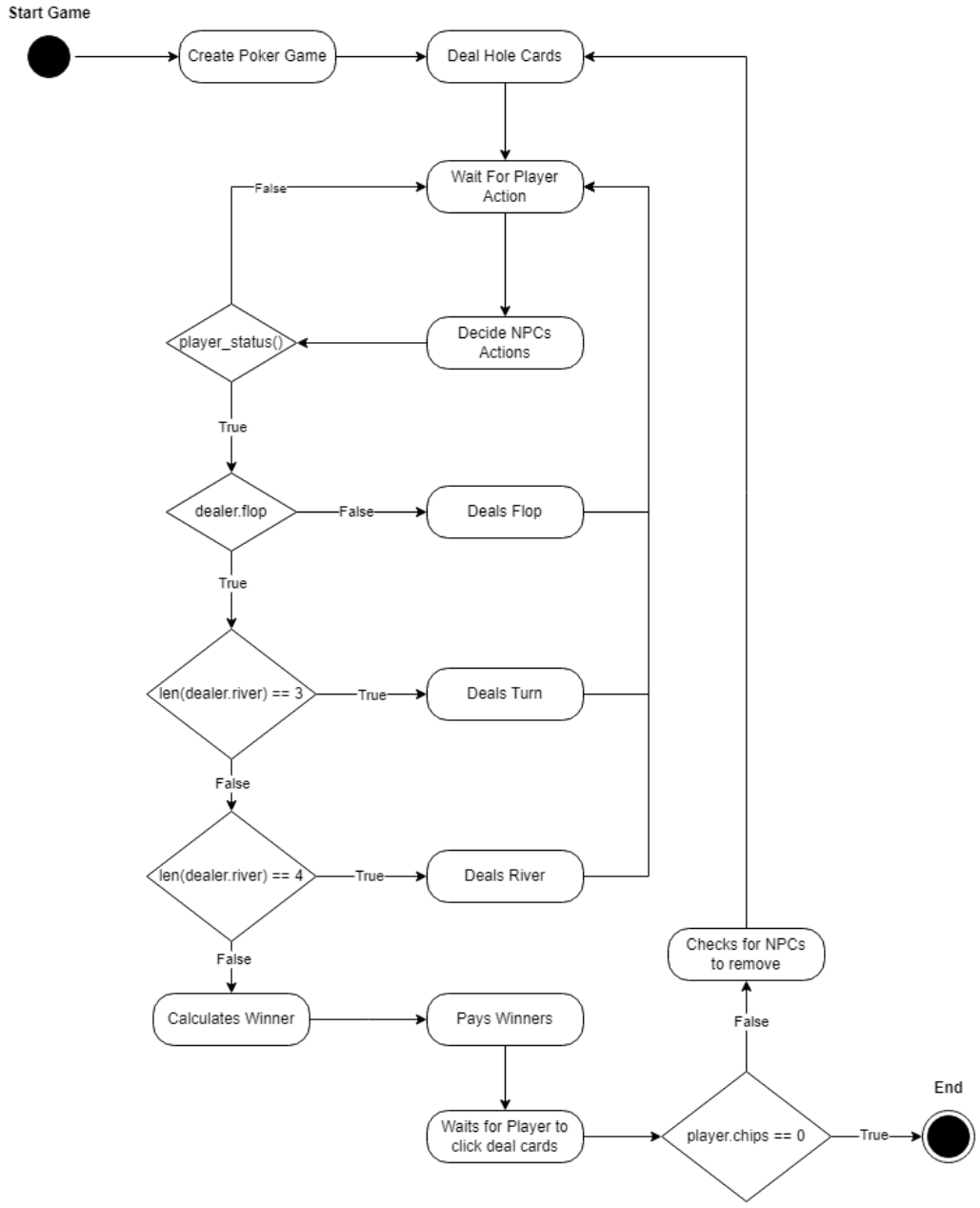
II. UML Class Diagram



III. System State Charts



CS 4398
Software Engineering Project



IV. System Structure and Algorithms Implemented

Artificial Intelligence Integration

Clustering Process

The *Texas Hold'em* game employs a clustering process used in the implementation of a poker AI. The primary components are focused on clustering different stages of poker hands to optimize decision-making processes.

Modules

The **CardInfoLutBuilder** module constructs and manages a lookup table for card information and centroids for each betting round (pre-flop, flop, turn, and river). It facilitates the abstraction of poker hands into different clusters based on their statistical strength and potential.

Algorithms Used:

K-Means Clustering: Used to categorize similar hands into clusters based on their features, primarily using the *K-Means* algorithm from the *scikit-learn* library.

Wasserstein Distance: Measures the distance between hand strength distributions, helping to identify the centroid that a particular hand strength distribution is closest to.

The **CardCombos** module manages the generation of card combinations (or hand histories) for each street (pre-flop, flop, turn, river) which is crucial for simulating various game scenarios and calculating expected hand strengths.

Algorithms Used:

Combinatorial Analysis: Generates all possible combinations of cards for given scenarios using Python's *itertools.combinations*.

The **Preflop** module provides functions for creating a lossless abstraction of starting hands, specifically tailored for short-deck games. This abstraction reduces the complexity of the decision-making process in the pre-flop stage by categorizing starting hands based on their ranks and suits.

Algorithms Used:

PHEvaluator: ranks hands based on predefined set of rules and categorizes them accordingly.

The **GameUtility** module serves as a bridge between the clustering process and the PyGame logic.

Machine Learning and AI

Model Training

The lookup tables and centroids created by the clustering process provide a way to quickly access complex strategic information without recomputing it. This information is crucial for the AI to learn effective strategies through techniques like counterfactual regret minimization (CFR).

Algorithm used:
Counterfactual Regret Minimization (CFR)

Strategy Development

By referring to the precomputed clusters, the AI can learn to recognize which actions lead to better outcomes in similar situations, refining its strategies over time.

Real-time Decisions

Once the training is complete, the AI can utilize the learned strategies during actual gameplay. The strategies developed during training are stored in strategy profile data structure optimized for quick retrieval.

Optimization

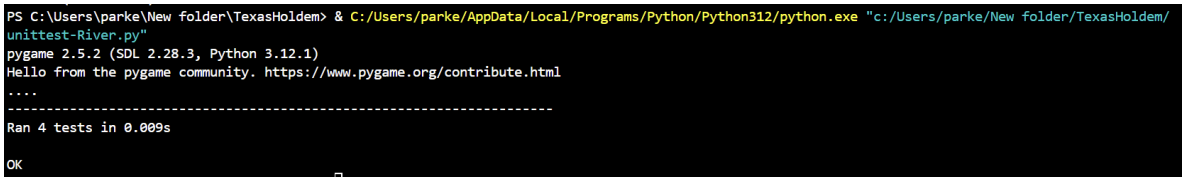
The precomputed data helps to reduce the computation needed during gameplay, allowing the AI to make fast decisions based on complex probabilistic calculations done beforehand during training. It relies on the strategies it has already learned.

V. Source Code

https://github.com/aewhitfield90/TexasHoldem/tree/main

VI. Unit Test Cases

The code for all of the unit test cases can be found at:
https://github.com/aewhitfield90/TexasHoldem/tree/main



VII. PowerPoint Presentation

Uploaded as a separate document.