Team 7
Matthew Thompson, Conner Simmering, and Slaton Spangler
Texas Hold ‘Em Game

CSCI 4448 Final Report

1. List the features that were implemented (table with ID and title).

USR05	As a player, I want to be able view my cards.
USR07	As a player, I want to be able to leave the game at any time.
USR09	As a player, I want to be able to join a game.
USR04	As a player, I want to be able to receive my two cards at the start of the hand.
USR06	As a player, I want to be able to view the cards on the board, including the flop, the turn, and the river.
USR01	As a player, I want to be able to fold, raise, or call before the flop.
USR03	As a player, I want to be able to know if I won the hand.
FNC03	Each user’s chip stack should be displayed to the rest of the table.
FNC04	Before each hand, each player should be dealt two cards which are hidden from the other players.
FNC14	If a user folds they should be removed from the hand and any bet they have made should be added to the pot.
FNC22	The program should be published online.
NF01	The system should be compatible with Windows, macOS, and Linux.
NF04	The game should be free to play.
NF05	The game should comply with all applicable laws and regulations.

USR02	As a player, I want to be able to check, call, raise, or fold after the flop, turn, and river.
USR06	As a player, I want to be able to view the cards on the board, including the flop, the turn, and the river.
FNC07	After the flop is displayed, a betting round should occur.
FNC06	After the player's hands are dealt and betting is completed, three cards (the flop) should be displayed to all players.
FNC08	After the flop betting round is complete, one card should be displayed to all players (the turn).
FNC09	After the turn card is displayed, a betting round should occur.
FNC10	After the turn betting round is complete, one card should be displayed to all players (the river).
FNC11	After the river card is displayed, a betting round should occur.
NF03	The interface of the game should be considered intuitive by the users.
FNC18	The user with the highest value hand at the end of the last betting round has the value of the pot added to their chip stack

2. List the features were not implemented from Part 2 (table with ID and title).

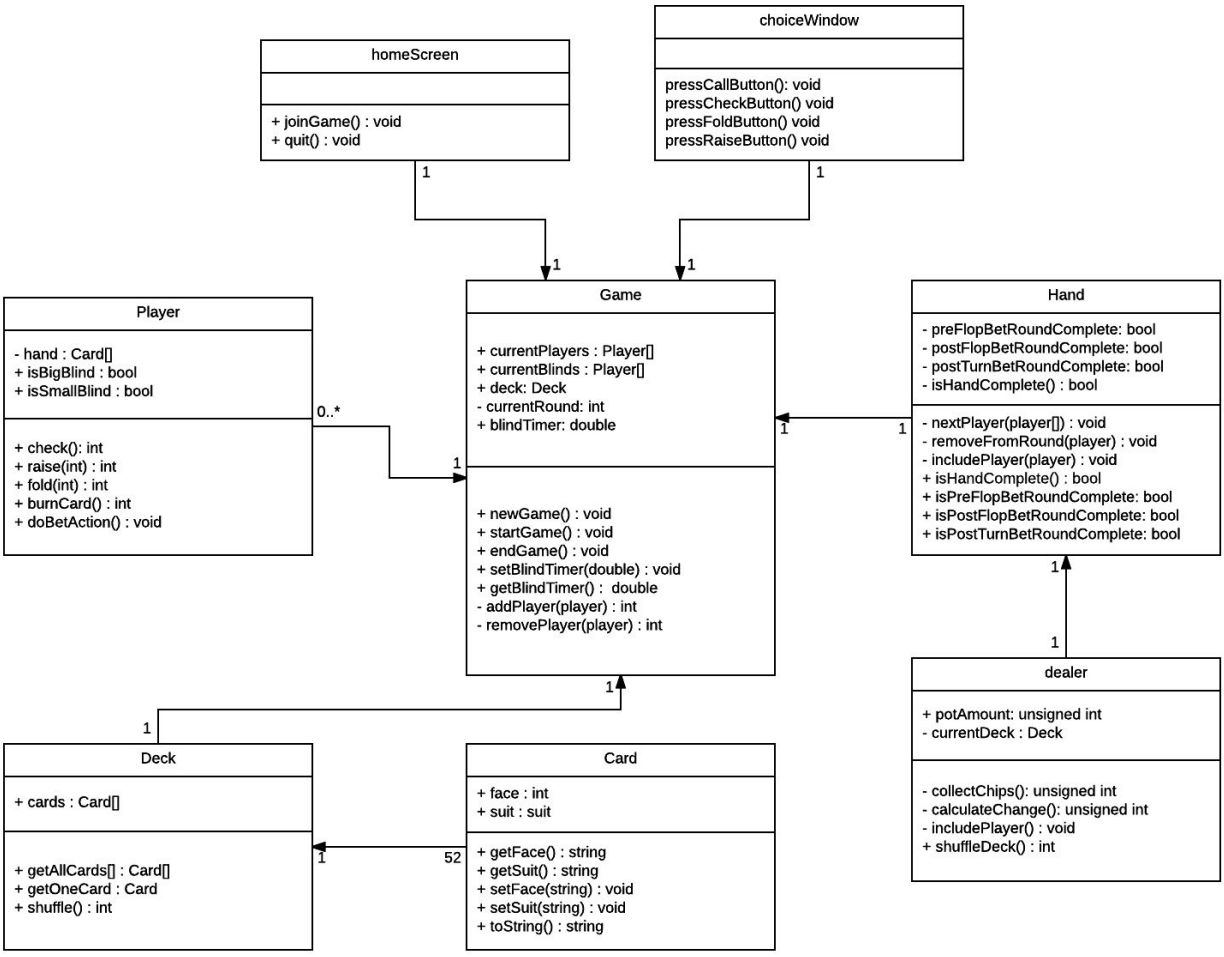
USR08	As a player, I want to be able to post blinds if it’s my turn to do so.
FNC01	A blind timer should be displayed to all players.
FNC05	Before each hand, the players who are in the blinds positions should have their blinds automatically posted.

FNC12	After the river betting round is complete, each player who remains in the hand should show their hand to the other players, beginning with the closest player to the small blind. Each player following should then have their hand shown to the other players if it beats the previously shown hands, or should be able to choose to show or fold their hand if it does not beat the previously shown hands. The player with the best hand should have the value of the pot added to their chip stack.
FNC13	In each betting round, betting should start with the player closest to the small blind, and continue clockwise around the table until all players have either folded or called the highest bet.
FNC15	The dealer button and blind positions should move one player to the left after every hand.
NF02	The game should offer clear instructions on how to use the system and clear rules of the game.
FNC16	The blinds should double after every expiration of the blind timer.
FNC19	If all other players fold, the last player in the hand should have the value of the pot added to their chip stack.
FNC20	Before the first hand, each player should be dealt one card, and the player with the highest card will be the first dealer.
FNC21	The small blind should be posted by the player to the left of the dealer, and the big blind should be posted by the player two positions to the left of the dealer.
FNC17	Every betting round besides the pre-flop betting round begins with the small blind.
FNC23	Users should join a game by entering a username that is unique relative to the other players at the table.

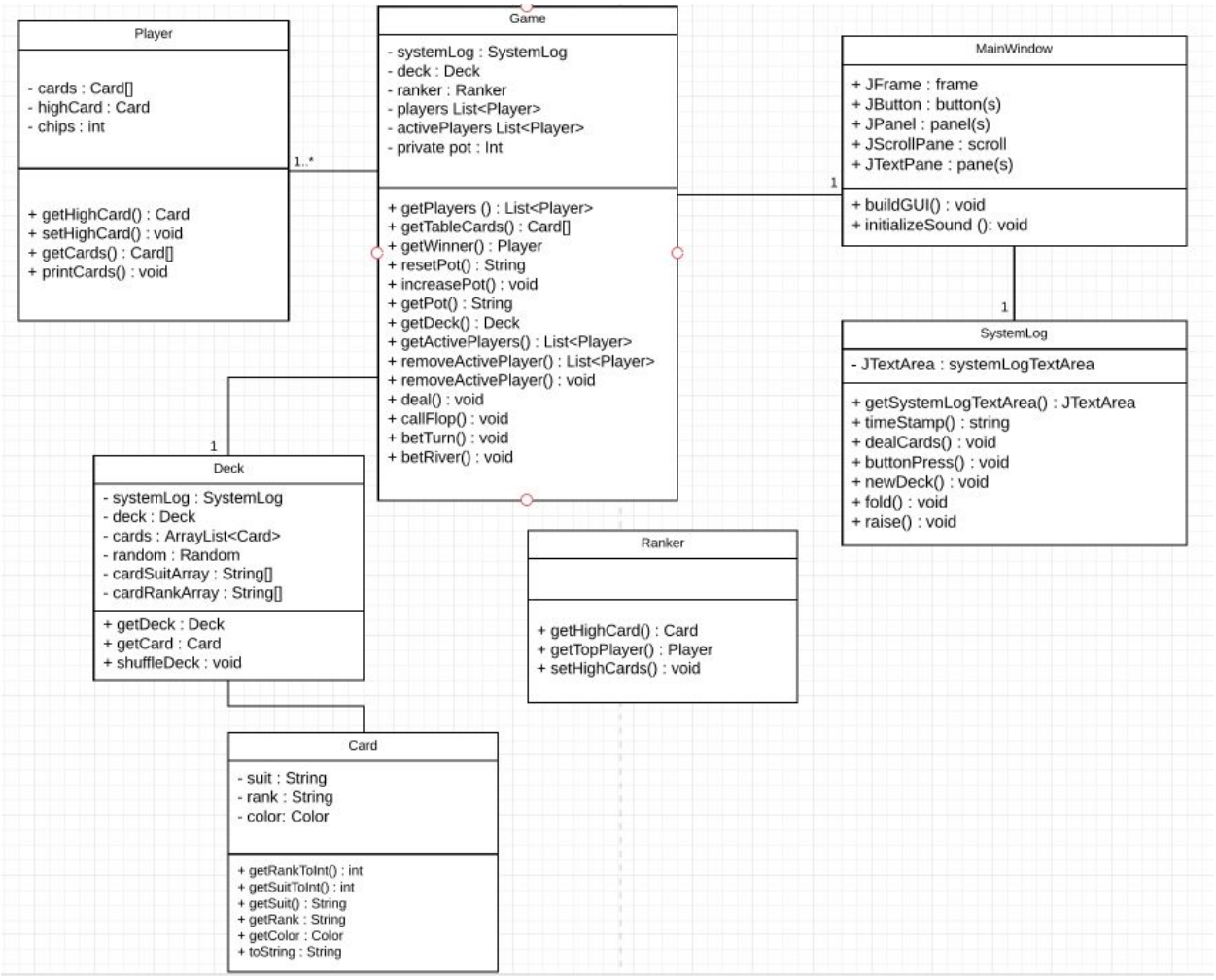
FNC24	Raises must be at least double the previously highest bet.

3. Show your Part 2 class diagram and your final class diagram. What changed? Why? If it did not change much, then discuss how doing the design up front helped in the development.

Part 2 Class Diagram:

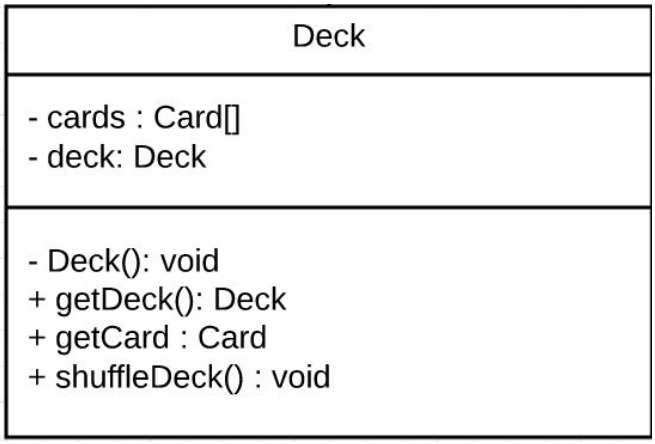


Final Class Diagram:



Much of the system's design changed during the course of the development of the prototype. This can be somewhat attributed to the fact that we were not familiar with the set of design patterns we learned before we completed Part 2. After learning the patterns and beginning the implementation of the system, we recognized several places where the design of the system could be improved. This resulted in the final class diagram seen above.

4. Did you make use of any design patterns in the implementation of your final prototype? If so, how? Show the classes from your class diagram that implement each design pattern (each design pattern as a separate image in the .PDF). If not, where could you make use of design patterns in your system? Show a class diagram of how you could implement each design pattern and compare how it would change from your current class diagram.



We used the Singleton design pattern for our deck.
We made the Deck constructor private and provide method to return (singleton) Deck instance and provided methods to return and reset (shuffle) the deck.

Potential design patterns:

- Strategy for implementing different sets of rules

5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

By stepping through the process of creating, designing, and implementing an Object-Oriented system, we learned several things. Chief among them is the importance of designing a system before implementing it. As developers, we often intuitively want to begin coding as soon as we have even the rough outlines of a concept. This invariably leads to non-optimal code, as doing the implementation of a system before the design does not allow the developer to recognize flaws in the original (conceptual) design. Furthermore, designing a system before implementing it allows the developer to recognize places where best practices, such as design patterns, can be used, and where bad practices, such as anti-patterns, can be avoided.

Another crucial concept we learned from this process was the idea of using previously identified best practices from other developers when creating our own systems. The set of design patterns we learned was invaluable in creating efficient, well-encapsulated, extensible code while avoiding the pitfalls of attempting to invent design patterns from scratch. By using the theories and practices developed by others, we were able to produce much more sustainable code. Similarly, learning different kinds of identified anti-patterns allowed us to recognize when we were implementing our system in a way that was known to be suboptimal. By building our system on the ideas of developers that came before us, we were able to produce a much better system.

Finally, we learned that standard procedures must be used in order for collaborative development to be effective. For instance, UML was instrumental in allowing us to express our conceived design of the system to each other, and converge on a final design that took the best parts of each group member's ideas. UML provided a standard language that we were able to use to effectively and efficiently communicate our ideas about the design and implementation of the system to each other. We also learned the importance of a strong development workflow, using systems like Git and practices like branching, pull requests, and descriptive commit comments to implement the system in the best way possible.

Between the lessons learned from designing a system before implementing it, using best practices in the implementation, and employing standard procedures during development, we gained a breadth and depth of knowledge about the development lifecycle that will be fundamental to the skills we continue to gain as developers - especially those relevant to the world of Object-Oriented Analysis and Design.