Network Programming

Final Project Report

Network Programming - COMP-2100-01A

Professor Salem Othman

Ghasif Syed, Tong Trinh, Yehua Chen

April 2, 2021

Introduction to the problem

For this final project, we are going to create a card game called Blackjack. The reason why we choose this topic is that this is a good opportunity for us to apply socket programming (the communication between client and server) in a real application. As you know, Blackjack is a card game that pits player versus dealer, which is a great opportunity to have the server act as the dealer and the player as a client. A traditional 52-cards deck is used and each card has a point value attached to it. All the face cards will count as 10 and all other cards will count the value that is printed on them. But there is a special card that allows the value count to be either an 11 or 1, this card is the Ace. So this will be a challenge for us to consider all possible results when you have multiple Aces in your hand. In this game, the client is considered as the player, and the server is considered as the dealer. We are making this game to allow multiple players to play (allow multiple clients to connect with one server), but each of the players is going to play their own game, they are not playing the same game together.

Relevant work

There have been other approaches to this problem as Blackjack is a widely popular game and it is quite universal. It is originally played as a card game and there have been many attempts and approaches towards creating the game of Blackjack through coding. During our research process we looked at many of these approaches to gauge how our project might look like, as well as to take inspiration from. We saw a few different approaches to coding blackjack, one that used Pygame to use a GUI, one that had cool designs to play the game using just the CLI, and one where there was a Blackjack game simulator which would replay thousands of times to get

probabilities of wins/losses/ties. However, we still hadn't found the implementation of Blackjack through socket programming enabling multiple clients to play while connecting to a server.

That is where we researched more about threading, despite learning a bit about it during our lab sessions we still had to look into it further as we were going to attempt creating a game where it would let you play multiple games. Using threading was not easy, since we did not have much experience coding with it, however despite that the approaches to the problem we looked at before still came in handy as it gave us a base-line of how to format and organize our code. Looking at the approach with GUI, we got really discouraged with the idea due to the time constraints we had since that would have been a lot for all three of us to research and create the game with a GUI. Though when we looked at one of the CLI examples, that was very interesting as the game still looked good and played well. Using this CLI approach we went forward and that helped us a lot as we knew we could make the game in CLI and have it play well.

After looking at these approaches we did not want to copy the same thing, so we tried not to refer to their code. First we went forward with our own code and tried utilizing a real deck system with 52 cards, but we resorted to using a random assortment of cards & suits. There were times where we got stuck with making it so the client could play multiple games, but we couldn't find anything online as none of the approaches we looked at were coding Blackjack in respect to socket programming. We had one week to resolve that problem in order to stay on track with our timeline, so we just experimented with our code and eventually using while loops got it so that each client could play multiple games.

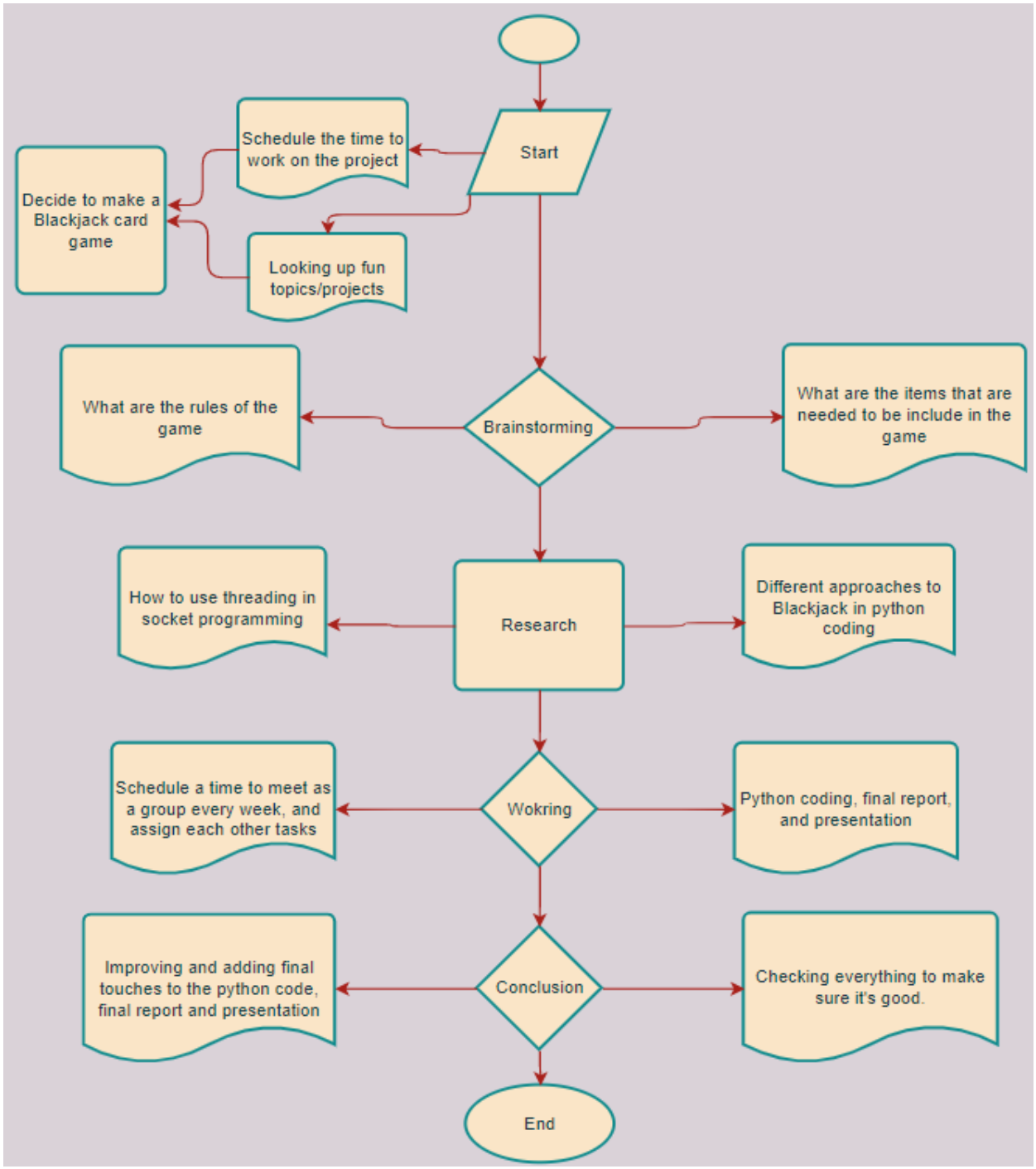
Description of the project

The implementation process for our code first started with us creating a basic server/client network for the program, in order to make sure that they can communicate with each other. After we established that there was a connection with the server, we started creating the bulk of our server code, starting off with implementing threading. On the server, we created suits, ranks, and assigned the correct values for each rank in a python dictionary. Also, we assign the server as a dealer and the client as a player creating a hand for each in the form of an array. We utilized lots of functions in our code in order to streamline the process of our program, to make it organized and easy to follow. One of the functions we used is utilized to start a new game, as it resets the hands and adds two cards to both the dealer and player. More functions that are important to the code are the add card and show card functions which add a card to the player/dealer hand and print out a specific card from an index of the player/dealer hand respectively.

To aid the player on the representation of their cards we added a function that adds up the value of the dealer/player hand, this helps in determining the points that the player/dealer have to assist the player in deciding whether to hit or stand. A special case that we found while coding this function is that there is a special case in this game where the Ace card counts as a value of a 1 or an 11. We decided to take care of this by making an algorithm that counts the number of aces in the player/dealer hand. Counting these aces is important as we can never count having 2 aces as both being values of 11, so to get around that we have to initially count aces as 1 and if the hand contains an ace and the total value of the hand is less than or equal to 11, we can count that ace as an 11 instead of 1. Few other functions which help in card representation is the print deck function which prints the full deck of the player/dealer, it does so by taking all the card

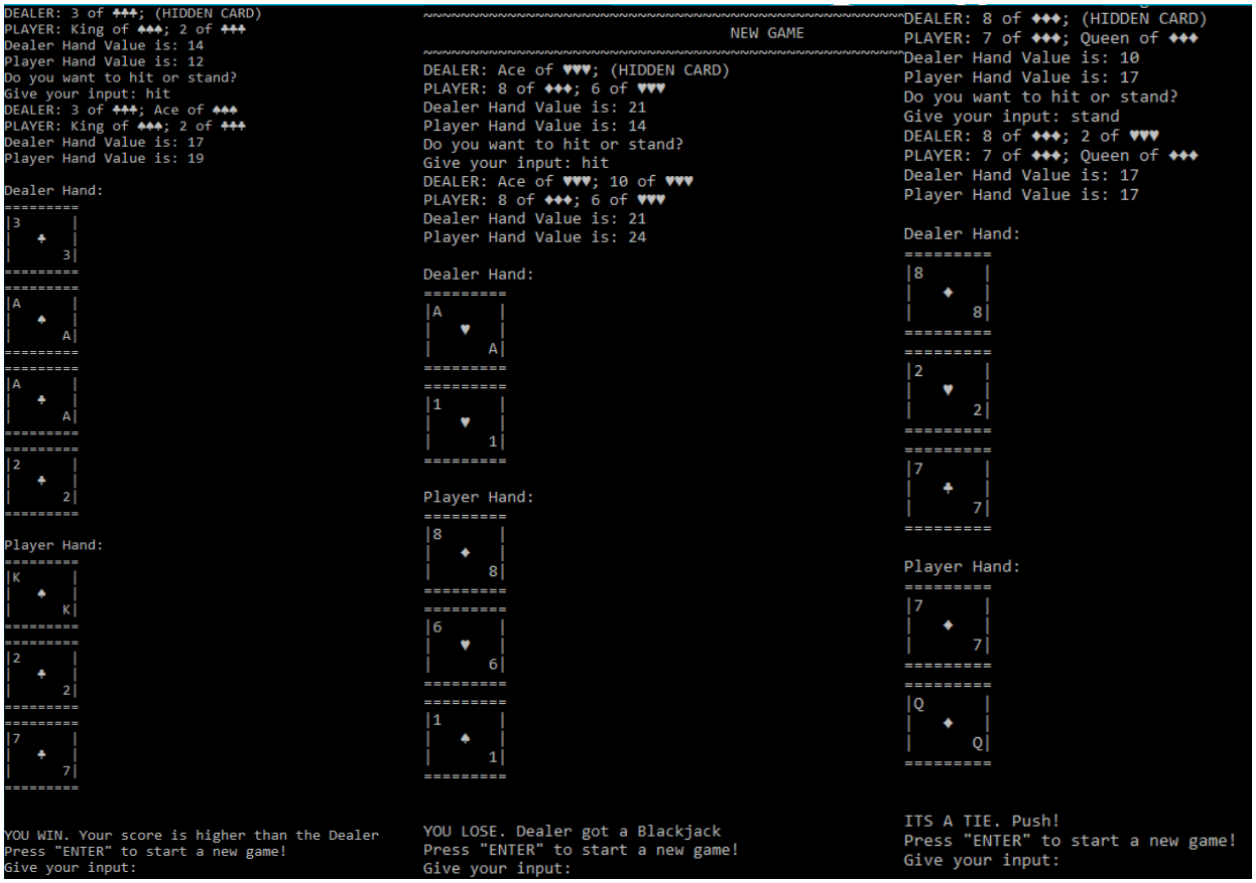
information from the hand and creating ASCII art to represent the card and returns a string to later use in the CLI. We also use functions to represent and format the current game and new game to make it easier for the player to distinguish the difference between the current game they are playing and if they want to start a new game.

The logic for the actual communication between server and client comes with the while loop we used in order for the client to be able to play multiple games by starting a new game. This code receives input from the client on whether or not they want to hit or stand, and using that it evaluates their choice. If the game ends with it either being a win/loss/tie a new game will be started and the player will be asked if they want to start the new game or not. We coded the game in a way where it is streamlined and easy to understand and we also commented the code to make it easier for anyone to follow along our code and understand all the functions and logic we used throughout. The following diagram underneath displays the workflow that we had during the duration of this project from the start to end.


Start
Schedule the time to work on the project
Decide to make a Blackjack card game
Looking up fun topics/projects
Brainstorming
What are the rules of the game
What are the items that are needed to be include in the game
Research
How to use threading in socket programming
Different approaches to Blackjack in python coding
Wokring
Schedule a time to meet as a group every week, and assign each other tasks
Python coding, final report, and presentation
Conclusion
Improving and adding final touches to the python code, final report and presentation
Checking everything to make sure it's good.
End

Results and Evaluation

Here are the results from playing the game. Either you win, lose, or tie.



The goal of Blackjack games is to get your card values as close to 21 as possible without busting (going over 21 points). A player needs to have a greater value than the dealer to beat them. An example of a winning situation in blackjack is that if a player's hand has a greater total point value than the dealers without going over 21 points, the player wins. Otherwise, the dealer wins. Another situation that may arise during gameplay is, if both the dealer and player have the same hand or points the round is a tie; in the game of blackjack this is called a 'push'.

We also make the cards visible for the players. Usually without GUI, people will just show the information to the users by using words. But pictures and images are very helpful for

people to understand things easier and faster. So we decided to use symbols and numbers to show the information out for players, and they don't have to take much time to read all the words that are showing out after they run the program. An example of this is we utilized ASCII to represent the suits in the game, as well as creating ASCII art to make mock cards to include in the game, in order to make the game more interesting to look at in the CLI (Command Line Interface). Compared with what we said in the proposal, we did a lot to improve our code along the way and added a lot of things in the program to make it more interesting and better.

Conclusion and Future work

Creating blackjack in python taught us a lot, not just the coding and socket programming part, but the whole process of it. We got to learn and experience ourselves from the ground up starting a project, brainstorming a topic/game, getting everyone on the same page, creating mockups and timelines, assigning each other tasks, and most importantly learn how it is to code as a team. Despite us successfully reaching our goal of creating blackjack through socket programming in python, we still ended up facing setbacks during the way. One of these includes logical issues with the aforementioned idea of an ace counting as either a point value of 1 or 11. Another that took us a long time to figure out and work towards was to add replayability in our code, this was possibly the biggest setback we had and at one point we were contemplating on keeping it so the client only plays one game and exits. However, we decided that it was possible and together we thought of a solution utilizing while loops inside of our thread to keep it looping through the program in order to add replayability.

If we were put in a situation where we had to start over again there would be a complete overhaul to our code. This is because we would most definitely research how to implement a

GUI in our program through socket programming, potentially though using Pygame. Creating blackjack through a GUI would require us to look into classes and different libraries, though our code would not go to waste as lots of our code can be salvaged as we used functions for a lot of the program, and functions are great as they aren't tied specifically to a string of code. With that being said the teamwork would not need to change as we were keeping on track with the project timeline we set for each other, and we met each other multiple times in a week to check in and add on to the project accordingly.

If we were to have more time on this project, and could spend more time on it there are a couple of things that we would add. One of these additions that we were planning on doing was the use of betting coins. The coins are a form of currency used to bet between the player and the dealer (Casino), there would be adept methods and solutions for calculating the betting between the Server & Client; very similar to the Banking simulation we did in one of the labs. A betting system would give more depth to our game and make it more interesting and enjoyable and that is definitely a future addition that we would add. Another addition that would serve our game of blackjack well is the incorporation of a GUI (Graphical User Interface). A GUI can be used to display objects and convey information more accurately and clearly. Currently we are using CLI (Command Line Interface) to present our game and play it, however in the future if a GUI was used it would be able to show pictures and in this case cards. Overall making the game more enjoyable as well as pleasing to look at as it wouldn't be played in the command line via text, but with the use of buttons and images on the screen.

References

- https://bicyclecards.com/how-to-play/blackjack/
- https://www.askpython.com/python/examples/blackjack-game-using-python
- https://www.youtube.com/watch?v=IPMcV_IXtX4
- https://games.washingtonpost.com/games/blackjack
- https://www.geeksforgeeks.org/socket-programming-multi-threading-python/
- https://docs.python.org/3/library/threading.html