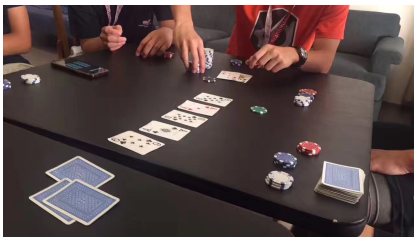
Online Texas Hold'em Poker Platform



I have been playing Texas Hold'em with my friends since the beginning of my senior year. Nothing will be more pleasing for a group of competitors than winning chips from their rivals. However, what was bothersome, if there was any, was that we had to come a long way to gather and play. We often had to arrange well in advance, so that each one of us could be free in the same period of time. Trying to improve this inconvenience, I decided to set up an online poker platform, which would allow us to play anywhere at anytime. A web application would be the best choice, since it could run on any operating system, PC, Mac, and mobile phones.

However, my past web developing experience was mainly associated with server and database. I was in charge of the backend development in the final project of web technology course at Stanford Pre-Collegiate Studies. I was most proud of the advanced search function I implemented to filter colleges according to names, majors, locations, and so on.

Generally, I was not familiar with front-end development things, especially the complex logic behind Texas Hold'em Poker, but I challenged myself to make progress, since there were countless online tutorials and resources to learn from. I paid special attention to a series of videos on bilibili of an engineer teaching front-end developing in javascript. Likewise, I chose to use javascript to simulate a card pile. The idea of object oriented programing I learned from AP Computer Science helped a lot. I constructed the card object at first; a card pile object has an array of card objects and methods including shuffle, flop, turn, river, etc. To implement an impartial and stable shuffle method, I tested my "random sorting" algorithm. It's never ineffective to run a random program repeatedly, meanwhile counting and checking the statistics to observe the general distribution, whether there was a specific pattern or it was even in a sample large enough. Sadly, it was flawed, because it had an uneven distribution. Therefore I had to search online and modify my algorithm. Finally, this scrambling process can finish in an instant; therefore cards could be shuffled quickly automatically instead of by hand.

Next to simulate is money, so that players can bet on their cards. To represent current bet and asset more clearly, I decided to use numbers rather than chips. Moreover, numbers are more comprehensible for players to analyze the situation and make action (whether to check, raise, call, or fold). I also implemented a function to find out winners by comparing and sorting. It first determines the players' card level (double, triple, straight, flush, full house or so), and then compares their high cards if tied. In this way, exactly following the rules, the winner will be found and the money will be redistributed. Thus time-consuming arguing among players could be avoided.

Considering players' experience, I decided to establish instant dual connection between the server and the clients, so that they would play with more joy since no refreshing was needed. Github is where I seek help. In such a great coding community, I found socket.io, which fits my nodeJS server well, and detailed instructions about this tool. Thus I was able to implement it successfully afterwards. I think this is the best part about coding and programing, because I can refer to numerous online resources and join communities for help. Those communities all have an active atmosphere of sharing and helping, where knowledge is easy to access.