HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

School of Information and communications technology

Network Programming

❧ • ❧

Group 10 report

Topic : FUNGAME

Group's members:

Name	MSSV
1. Đặng Việt Anh	20176686
2. Nguyễn Minh Tú	20154203

Supervisor: Dr. Trần Nguyên Ngọc

Hanoi, May, 2021

Table of contents

Prologue

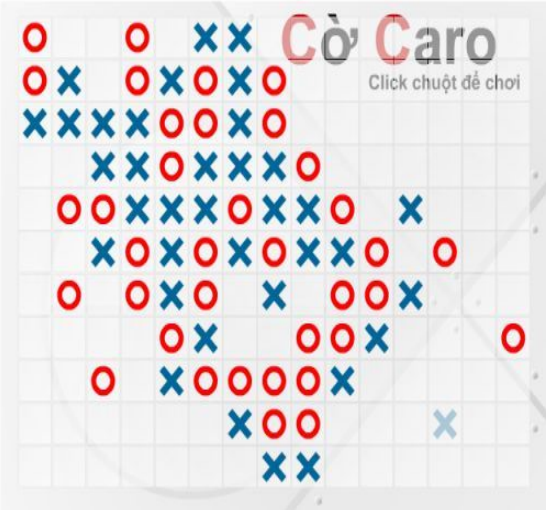
First of all, we would like to sincerely thank the enthusiastic help and teaching of teacher Tran Nguyen Ngoc. Thanks to the knowledge in the practical network programming subject and comments in progress presentations, we have completed the topic Fun Game within the framework of the course report.

In progress of working, shortcomings in the process of implementation and synthesis are inevitable. We hope to receive your comments and guidance for better performance.

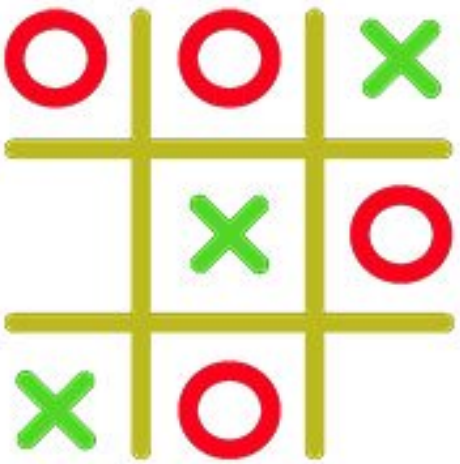
1. Introduction

Our group tend to build traditional games: Tic Tac Toe game and Caro game. This games are both for entertainment and improving thinking ability of players.

This games is built network programming with basic functions like play games, view ranking, view log, etc. In technique, we use Minimax algorithm for cpu player, hiding password by termios.h library and timeout value.



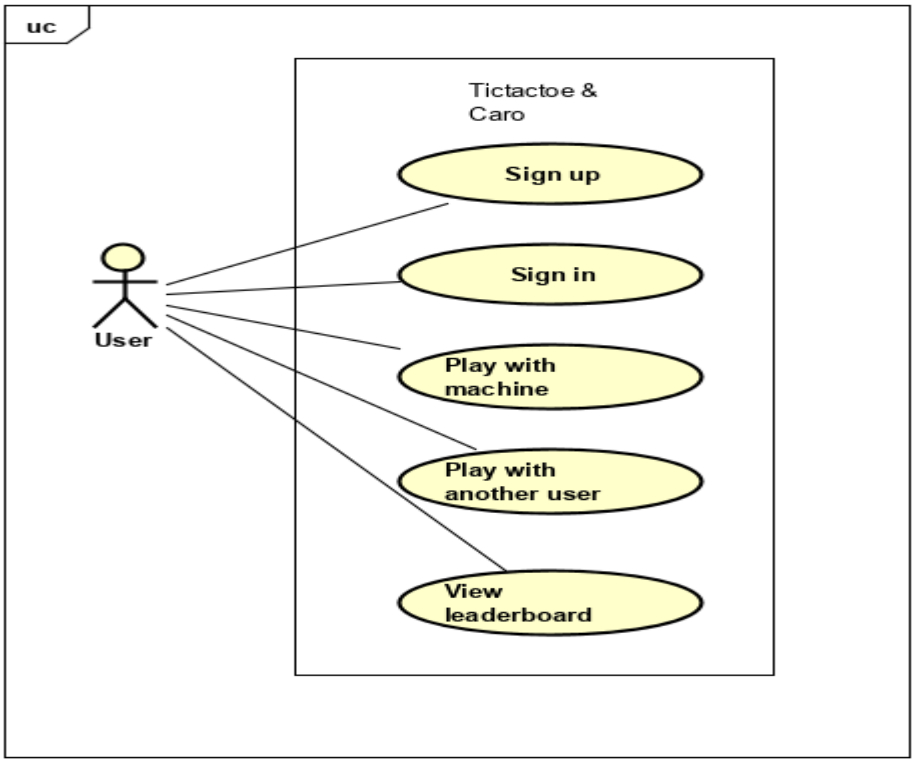
Caro Game



Tic Tac Toe Game

2. Design analysis

✓ Use case diagram



✓ Server-Client architecture

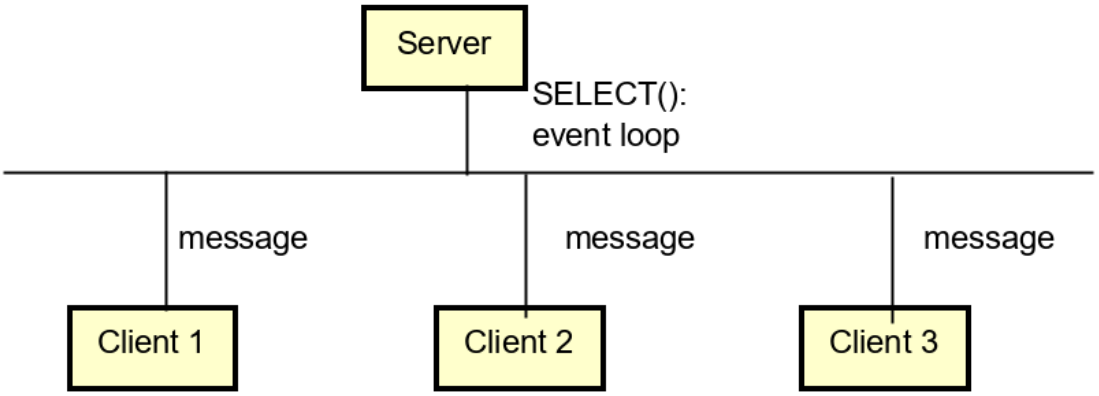
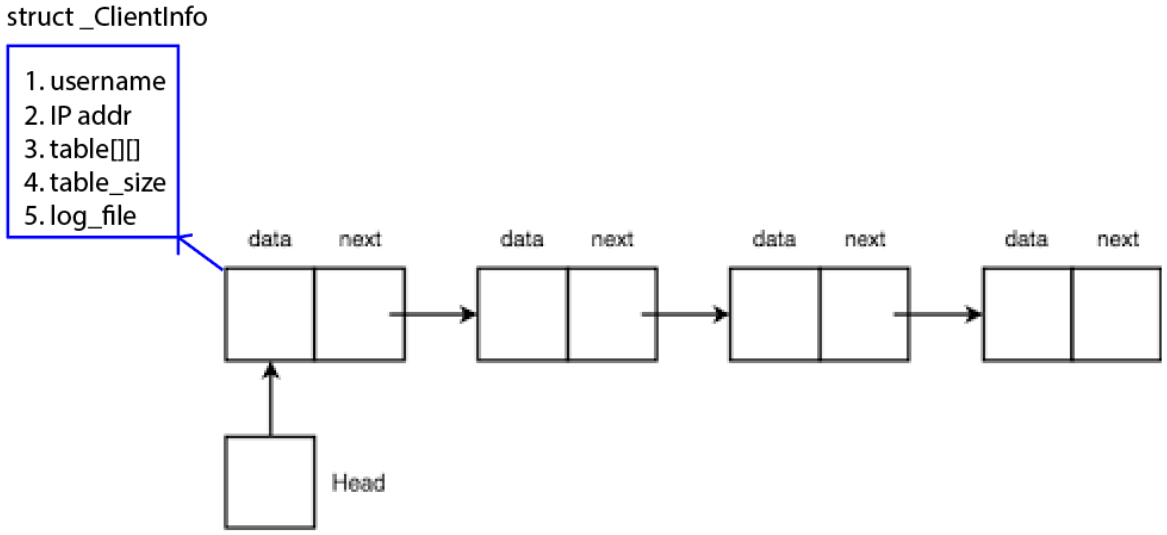


Fig: Client-Server architecture

- Event- driven socket program
- Server sits on an event loop
- Clients' messages are handled upon arrival

✓ Server-Client model: Many clients connect to one server

✓ Singly linked list:



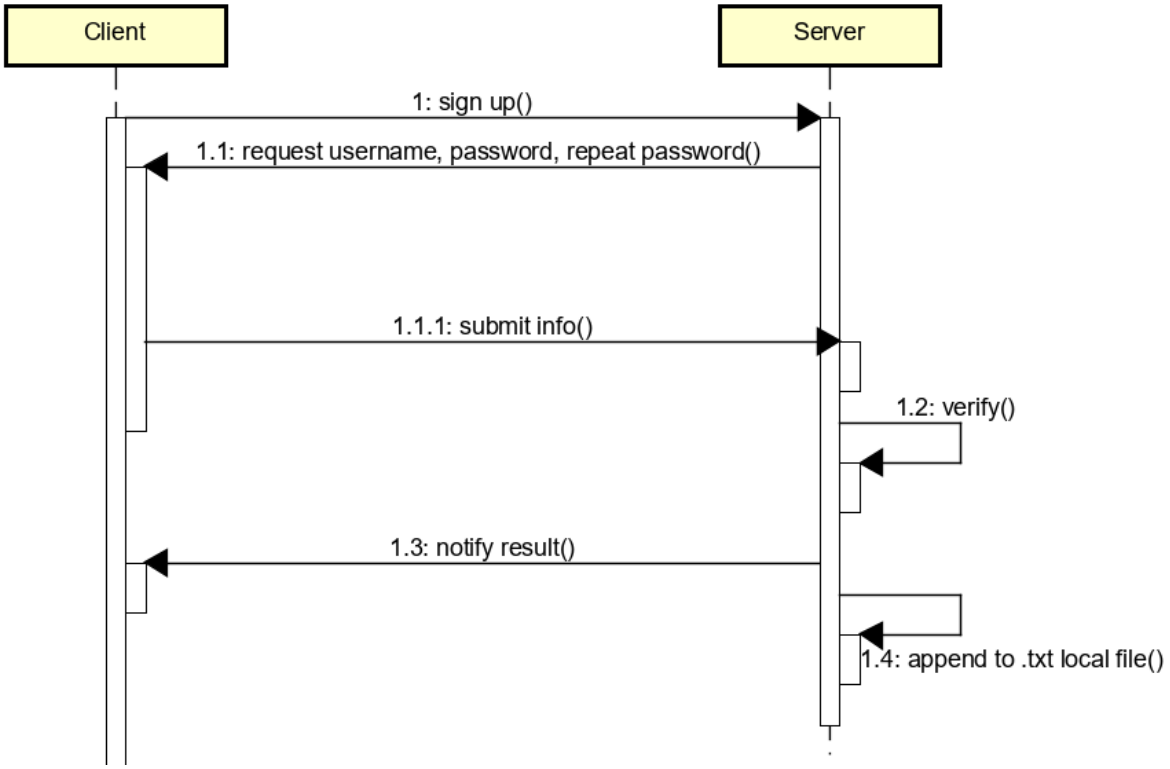
- Server uses this list to store current games of different clients
- Operations: new, get, add, remove

✓ Communication Protocol:

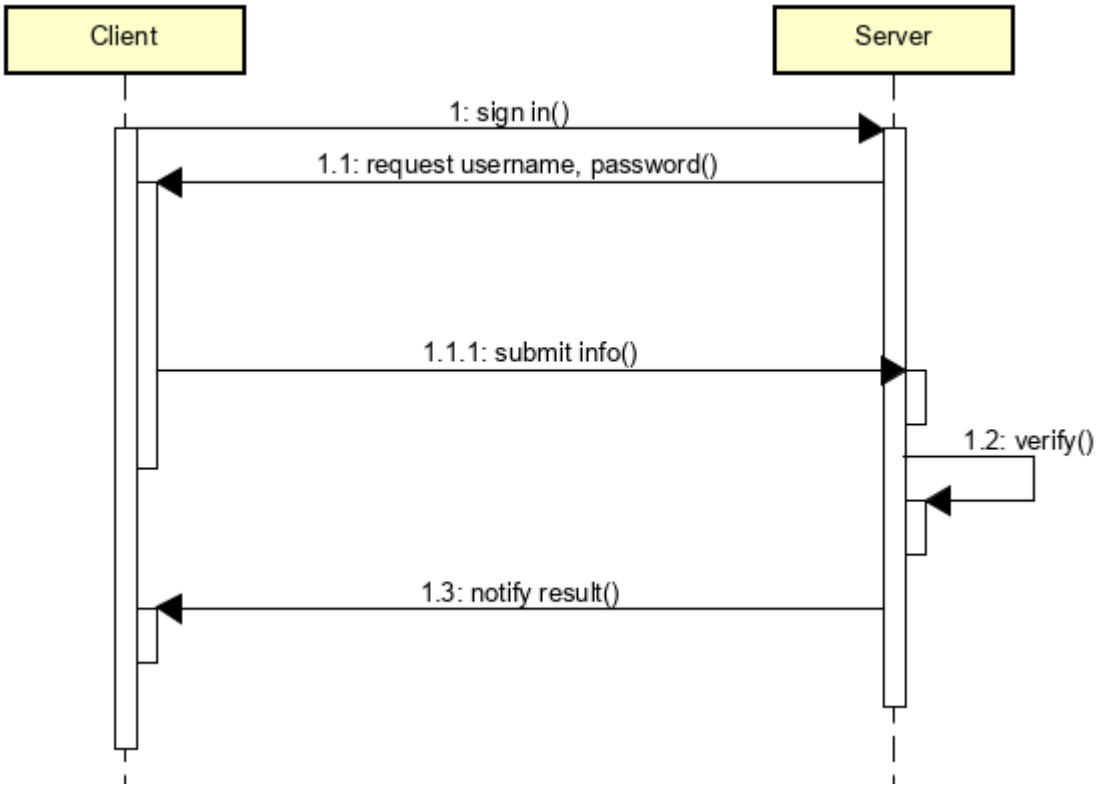
Messages between clients and server are in form: SIGNAL#PARAM1#PARAM2…

Signal	Param	Explaination
SIGNAL_CARO_NEWGAME	None	Client requests a new caro game
SIGNAL_CREATEUSER	Username#Password	Client wants to sign up with given name & password
SIGNAL_CARO_WIN	None	Server notifies client that he/she has won
...		

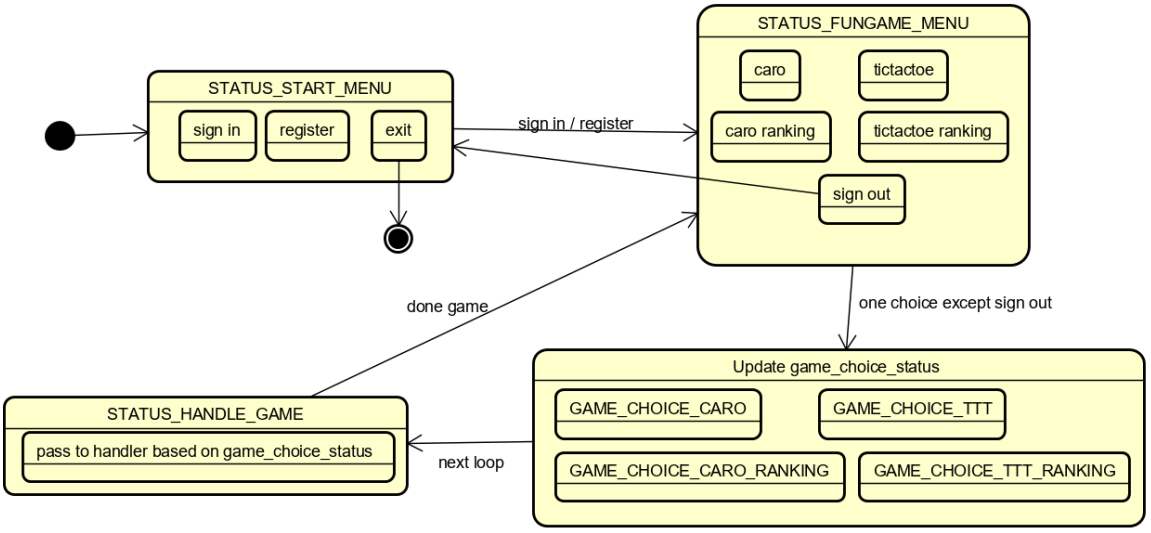
✓ Sign up



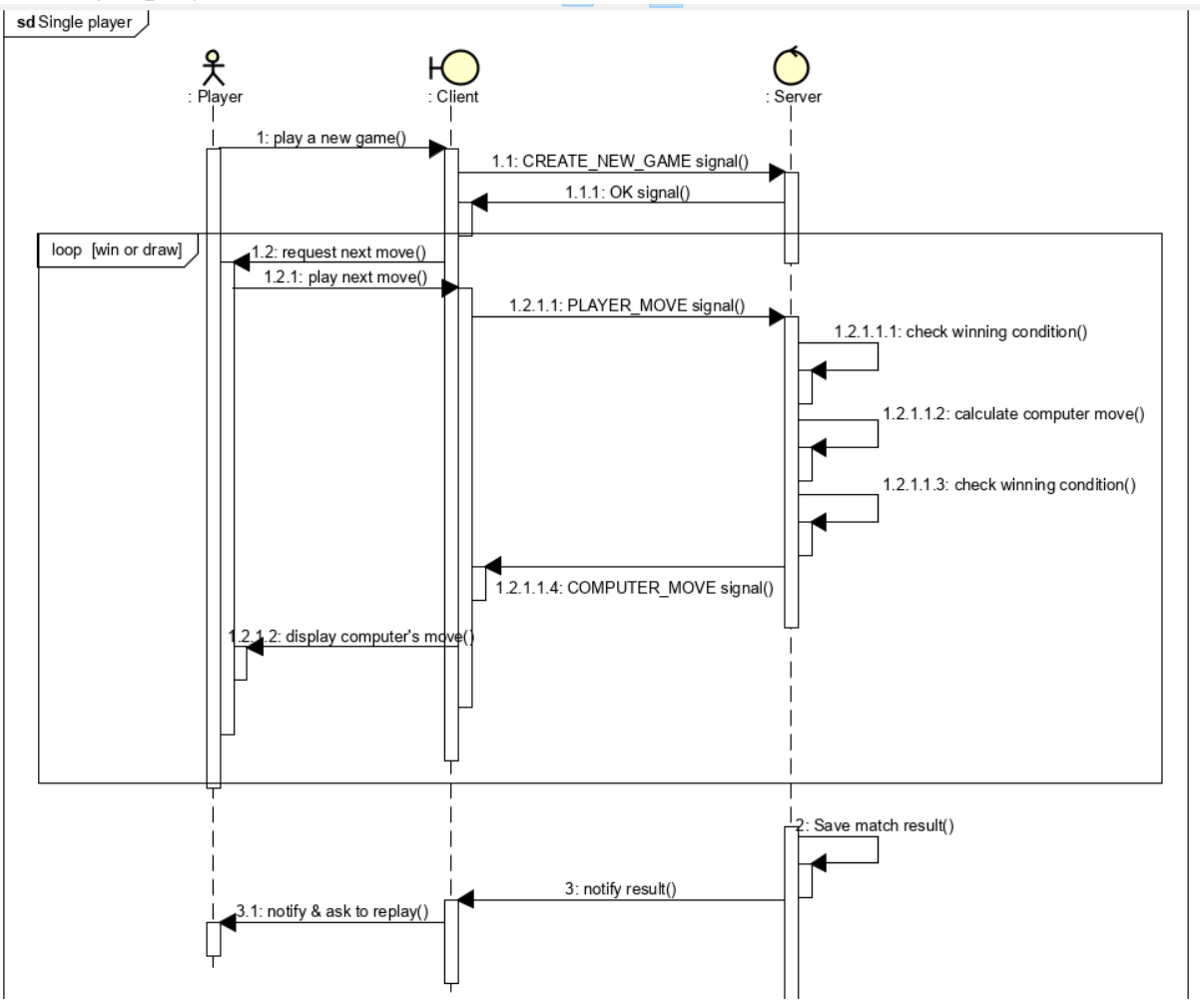
✓ Sign in



✓ Menu Navigation: Finite State Machine

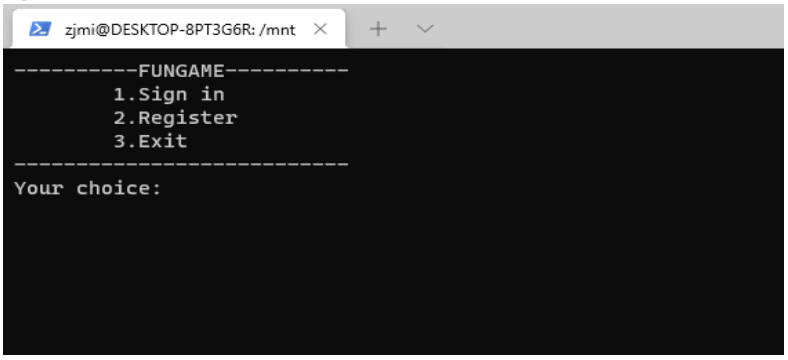


✓ Single player:



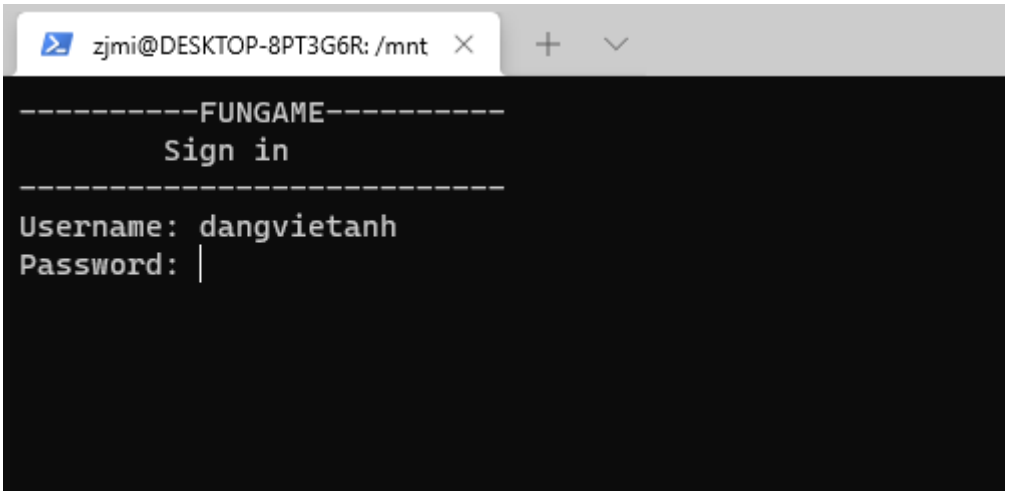
3. Function

3.1: Register new user



After clients connected to server, STATUS_START_MENU appear with 3 options, clients choose option 2 for register, after that they input username and hiding password.

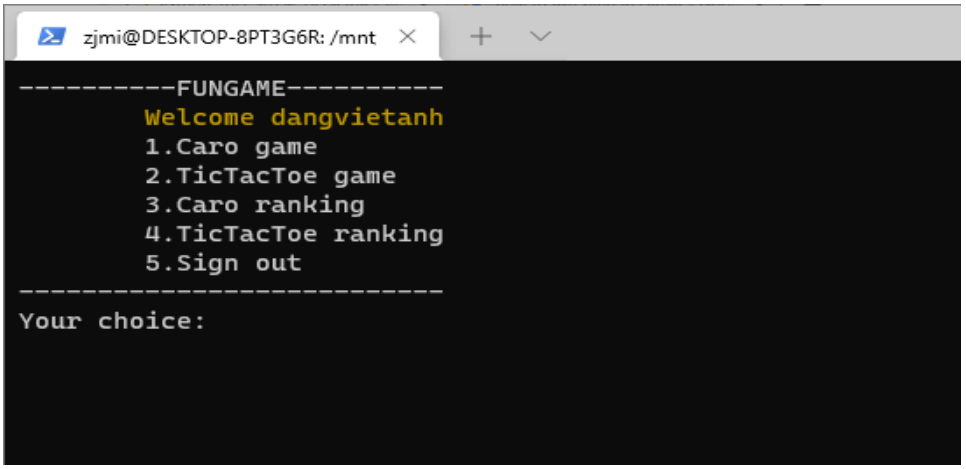
3.2: Sign in



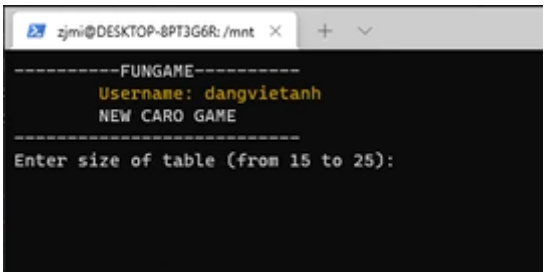
At STATUS_START_MENU, choosing option 1-> input username and password.

3.3: Play game

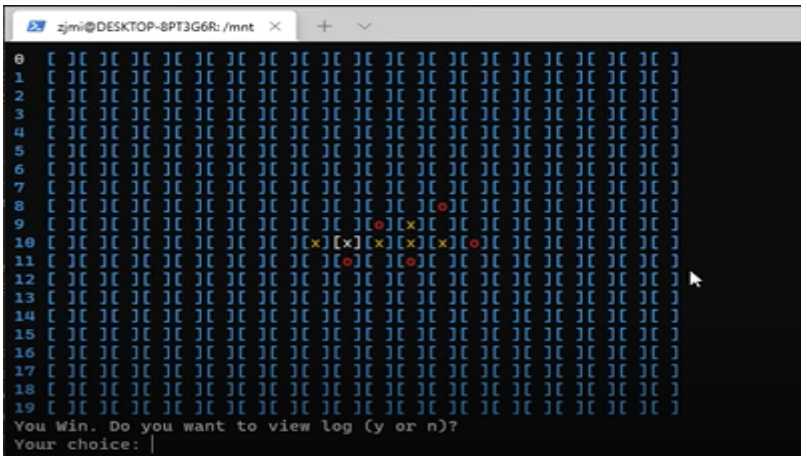
After sign in, STATUS_FUNGAME_MENU will appear.



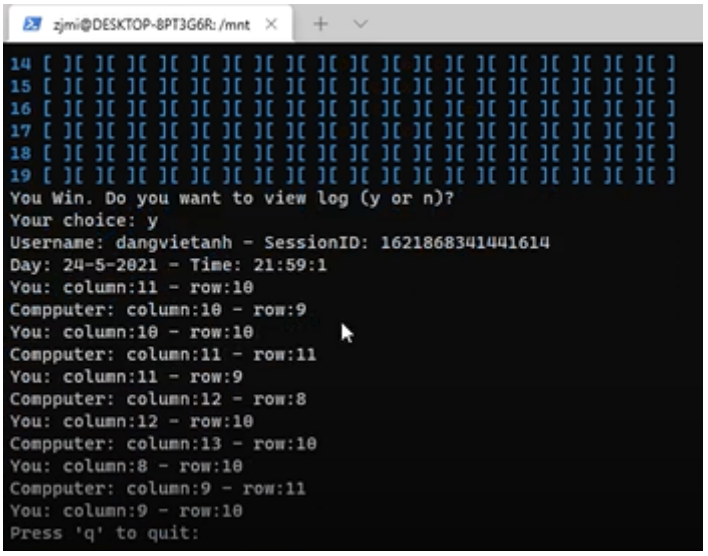
With option 1, we will play Caro game:



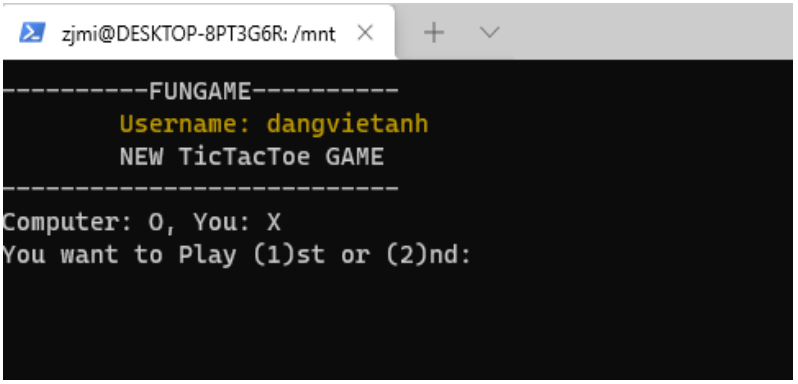
Next, we choose size of table Caro(from 15 to 25 pixels), then this table will appear:



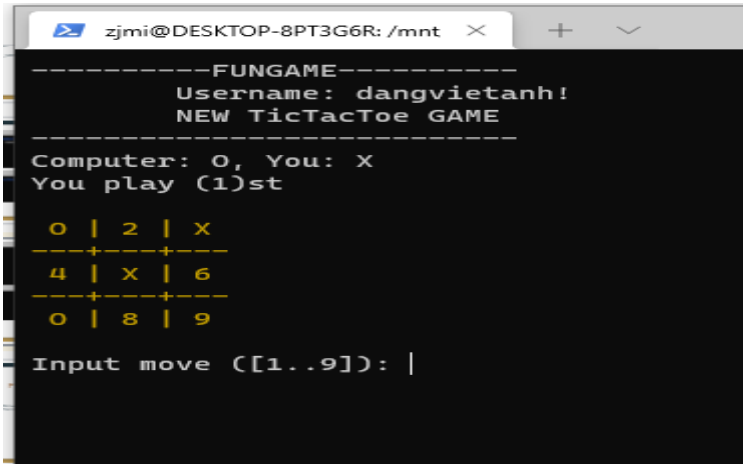
After a time of play and finish, board will show up result and ask for viewing log:



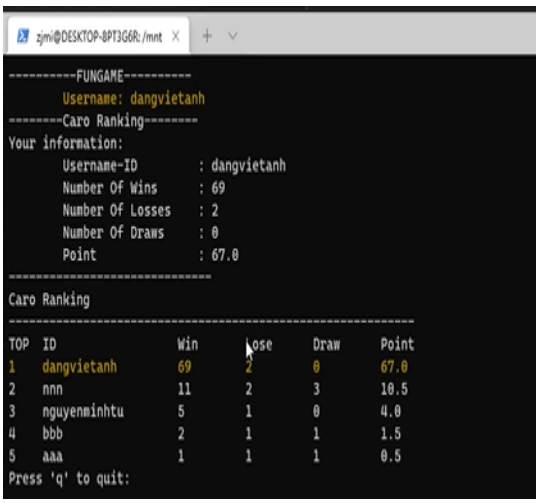
With option 2, we will play Tic Tac Toe game:

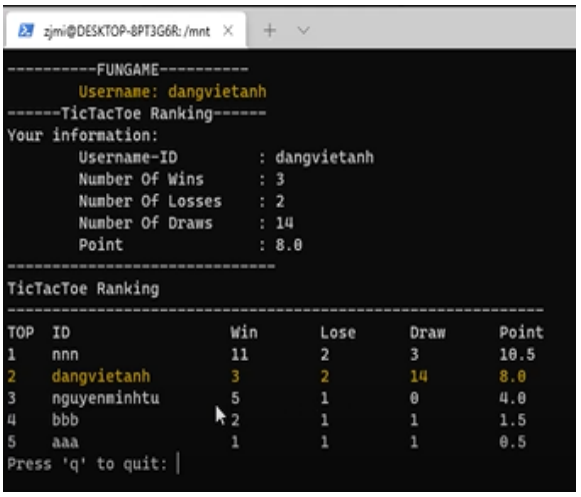


Next, we choose order to play:



With option 3 and 4, we see ranking of games:





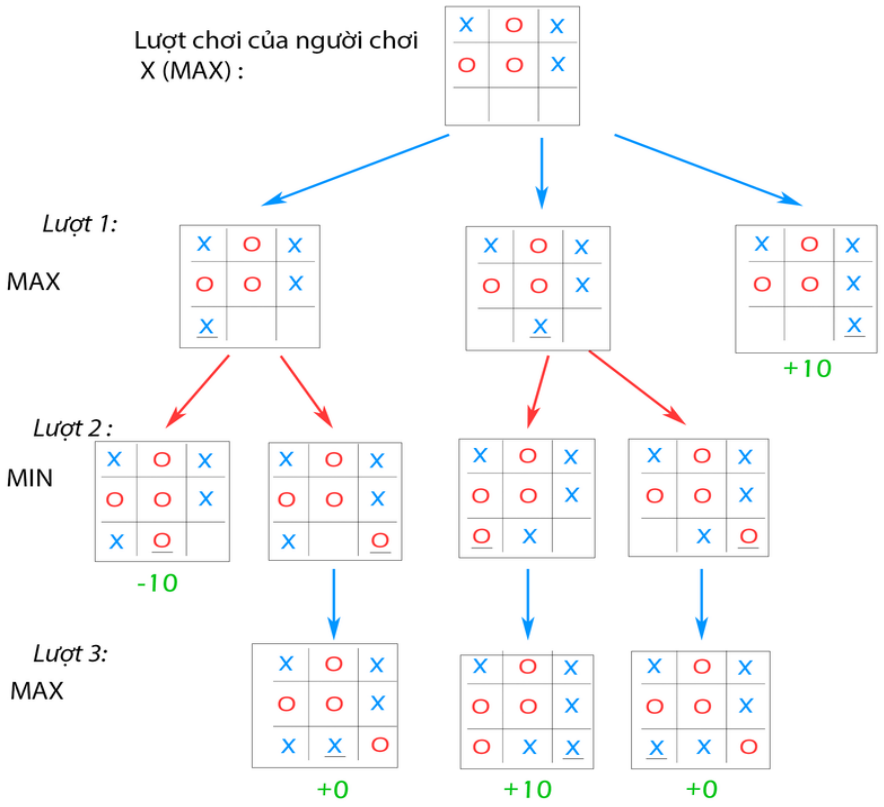
4. Technique

4.1 Minimax Algorithm

- Concepts:

- Game tree is the model tree contain each state, case of game follow turn play
- Each node depicts one state of current game in game tree
- Node is called leaf which is at end of game(state is win, lose or tie)

- Minimax algorithm: Two player are represented by MAX and MIN. MAX always try to win and optimize play, meanwhile MIN try to make MAX get the smallest score. This algorithm perform by how to valuate each node in game tree: Node belonging to MAX is assigned max value and node belonging to MIN is assigned min value. From there, players have base to play next properly.



Follow the graph: First, we see that the current state of game is X's turn(MAX). Our convention is when X win MAX = +10, X lose MIN = -10 and tie = 0. In turn 1, MAX has 3 ways. If MAX go to case 3, X win, if not we go next to next turns.

4.2 TCP connection client side:

```
//Step 1: Construct socket
  if((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1){
    strcpy(error,"Error Socket!!!");
    return -1;
  }

  //Step 2: Specify server address
  server_addr.sin_family = AF_INET;
  server_addr.sin_port = htons(PORT);
  server_addr.sin_addr.s_addr = inet_addr(serverAddress);

  // set timeout
  struct timeval timeout;
  timeout.tv_sec = 20; // after 20 seconds connect will timeout
  timeout.tv_usec = 0;
  if (setsockopt (sock, SOL_SOCKET, SO_RCVTIMEO, (char *)&timeout,
 sizeof(timeout)) < 0){
    return -1;
  }
  else if (setsockopt (sock, SOL_SOCKET, SO_SNDTIMEO, (char
```

```
*)&timeout, sizeof(timeout)) < 0){
    return -1;
  }

  //Step 3: Request to connect server
  if( connect(sock, (struct sockaddr*)&server_addr, sizeof(struct
sockaddr)) == -1){
    errorConnect = errno;
    sprintf(error,"Error! Can not connect to server!
%s",strerror(errorConnect));
    return -1;
  }
  //Step 4: Communicate with server
  send(sock, send_msg, strlen(send_msg), 0);
  recieved = recv(sock, recv_msg, BUFF_SIZE, 0);
  recv_msg[recieved] = '\0';
  strcpy(send_msg, SIGNAL_CLOSE);
  send(sock, send_msg, strlen(send_msg), 0);
  close(sock);
  if(recieved == -1){
    printf("\nError: Timeout!!!\n");
    return -1;
  }
  isCommunicating = 0; // ngat ket noi
  return 0;
```

4.3 TCP connection server side

```
// Step 1: Construct a TCP socket to listen connection request
  if((sock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("Socket error\n");
    exit(-1);
  }

     if(setsockopt(sock,SOL_SOCKET,SO_REUSEADDR,&true,sizeof(i
     nt)) == -1) {
    perror("Setsockopt error\n");
    exit(-2);
  }

  //Step 2: Bind address to socket
  server_addr.sin_family = AF_INET;
  server_addr.sin_port = htons(PORT);
  server_addr.sin_addr.s_addr = INADDR_ANY;
  bzero(&(server_addr.sin_zero),8);

  if (bind(sock, (struct sockaddr *)&server_addr,
     sizeof(struct sockaddr)) == -1) {
    perror("Unable to bind\n");
    exit(-3);
  }
```

```
//Step 3: Listen request from client
if (listen(sock, 5) == -1) {
  perror("Listen error\n");
  exit(-4);
}
printf("FUNGAME waiting for client on port %d\n", PORT);
fflush(stdout);

FD_SET(sock, &master);
fdmax = sock;

// Set timeout
struct timeval timeout;
timeout.tv_sec  = 1000;  // after 1000 seconds will timeout
timeout.tv_usec = 0;
//Step 4: Communicate with clients
while(1){
  read_fds = master;
  rc = select(fdmax + 1, &read_fds, NULL, NULL, &timeout);
  if( rc == -1){
    perror("select() error!\n");
    exit(-6);
  }
  else if (rc == 0){
   printf("  select() timed out. End program.\n");
   exit(-5);
  }
  for(i = 0; i <= fdmax; i++){
    if(FD_ISSET(i, &read_fds)){
      if(i == sock){
        sin_size = sizeof(struct sockaddr_in);
        connected = accept(sock, (struct
   sockaddr*)&client_addr, &sin_size);
        if(connected == -1){
          perror("accept error!\n");
          exit(-7);
        }
        else{
          FD_SET(connected, &master);
          if(connected > fdmax)
            fdmax = connected;
          printf("Got a connection from (%s , %d) with fd =
   %d\n",
   inet_ntoa(client_addr.sin_addr),ntohs(client_addr.sin_por
   t), connected);
          handleDataFromClient(connected);
        }
      }
      else{
```

```
            handleDataFromClient(i);
          }
        }
      }
    }
    close(sock);
    return 0;
```

4.4 Other technique

- Use library **termios.h** to hide password.
- Use timeout value for both server and clients.

5. Evaluate

- Program have complete functions: Sign in, register, play game, view log, view ranking.
- Functions activate normal, stable.
- Solve timeout error: Set timeout for both clients and server.

6. Work Contribution

Đặng Việt Anh (50%)	Nguyễn Minh Tú (50%)
● Game logic ● Finite state machine menu ● Login-sign up logic	● Game interface ● Design server-client communication

7. Conclusion

Above is all contents of our project, which are showed up from design analysis to complete program.

Video demo: https://youtu.be/9_IYJvR2kPI

Source code: https://github.com/GoodGuy69/caroPj.git

8. References

1. Slides of teacher Tran Nguyen Ngoc
2. "Unix-network-programming" ebook - W. Richard Stevens, Bill Fenner, Andrew M. Rudoff
3. Reference for termios.h library: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.bpxbd00/rttcga.htm
4. Minimax algorithm: https://viblo.asia/p/thuat-toan-minimax-ai-trong-game-APqzeaVVzVe