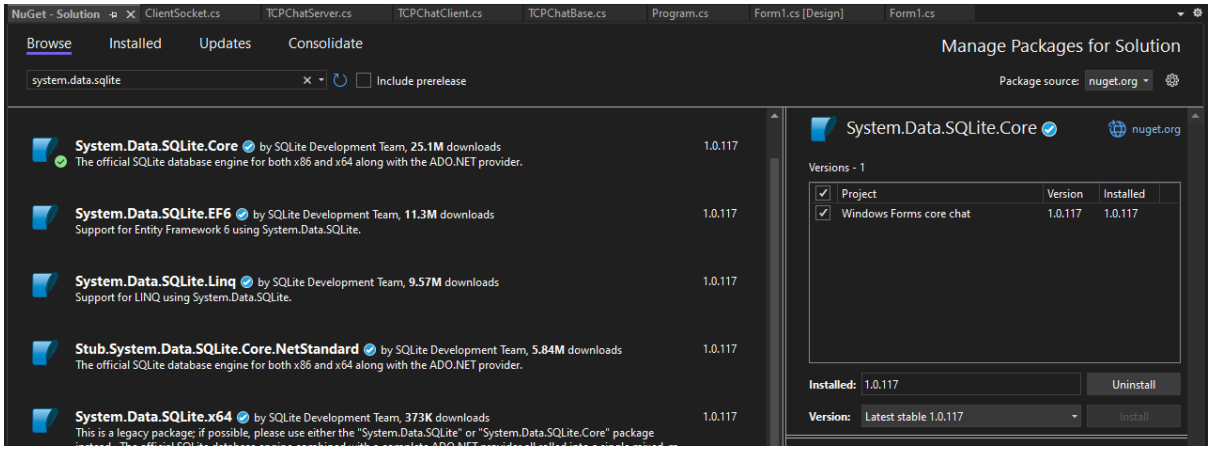
TORRENS UNIVERSITY OF AUSTRALIA

NDS203 NETWORKING AND DATABASE SYSTEMS

ASSESSMENT 3

Step 1



Step 2

2.1 Open database connection

```
private static SQLiteConnection connection;

// connect to the database file
1 reference
private static SQLiteConnection GetDB()
{
    try
    {
        connection = new SQLiteConnection("Data Source=databasefile.db");
        return connection;
    }
    catch (Exception)
    {

    }

    return null;
}
```

2.2 Create the user table

```
// create the user table
1 reference
public static string CreateUser()
{
    SQLiteConnection con = GetDB();
    if (con != null)
    {
        con.Open();
        // sql query as a string
        string sql = "CREATE TABLE IF NOT EXISTS user (id INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "username text, password text, wins INTEGER, losses INTEGER, draws INTEGER);";
        SQLiteCommand query = new SQLiteCommand(sql, con);
        // execute the sql query
        query.ExecuteNonQuery();
        con.Close();

        // display message if user table created
        return "User table succefully created.";
    }
    else
        return "Cannot connect to the database!";
}
```

```
    chatTextBox.Text += "Server setup complete" + Environment.NewLine;

    // Assignment 3 - Step 2 - Create the User table after server setup process complete
    chatTextBox.Text += DatabaseServer.CreateUser();
```

Step 3

3.1 Add state variable

```
// created an enum to store the states of the client
5 references
public enum ClientState
{
    login, chatting, playing
}

// created an enum to store the player numbers
5 references
public enum PlayerNumber
{
    nonplayer, player1, player2
}
```

```
    // create a new variable as 'state' to track each client state as for Assignment 3 - Step 3
    public ClientState state = ClientState.login;

    public Double level = 1; // custom variable, used to track the client current level

    // store what player they are
    public PlayerNumber player = PlayerNumber.nonplayer;
```

3.2 Client can log in to the server

3.2.1 When the client login to the server

```
// begins Assignment 3 - Step 3
Form2 form2 = new Form2();
form2.ShowDialog();
ChatTextBox.Text +=  form2.data + Environment.NewLine;
client.SendString("!username " + form2.username);
```

3.2.2 Client can either register a new username and password or log in with credentials

```
// begins Assignment 3 - Step 3
Form2 form2 = new Form2();
form2.ShowDialog();
ChatTextBox.Text +=  form2.data + Environment.NewLine;
client.SendString("!username " + form2.username);
```

```
    username = textBox1.Text;
    data = DatabaseServer.LoginUser(textBox1.Text, textBox2.Text);

    if (data != null)
    {
        // close the second form
        this.Close();
    }
    else
    {
        textBox2.Text = null;
    }
```

```
public static string LoginUser(string username, string password)
{
    SQLiteConnection con = GetDB();

    // temporary variable as a count
    int count = 0;

    if (con != null)
    {
        con.Open();
        // sql query as a string
        string sql = "SELECT username, password FROM user;";
        SQLiteCommand query = new(sql, con);

        // execute the sql query check if user already exists
        SQLiteDataReader reader = query.ExecuteReader();

        while (reader.Read())
        {
            if (username == reader["username"].ToString())
            {
                if (password == reader["password"].ToString())
                    count++;
                else
                    return null;
            }
        }

        if (count == 0)
        {
            sql = "INSERT INTO user (username, password, wins, losses, draws) VALUES (@p1, @p2, 5, 0, 0);";
            query = new SQLiteCommand(sql, con);
            query.Parameters.Add(new SQLiteParameter("@p1", username));
            query.Parameters.Add(new SQLiteParameter("@p2", password));
            query.ExecuteNonQuery();

            con.Close();

            return "You are successfully logged In!";
        }
        else
        {
            con.Close();
            return "Welcome back!";
        }
    }
    else
        return "User login unsuccessfull";
}
```

3.2.3 Client is progressed to the chatting phase

```
currentClientSocket.state = ClientState.chatting;
byte[] data = Encoding.ASCII.GetBytes("Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.");
currentClientSocket.socket.Send(data);
```

3.3 Players can join the TicTacToe game

3.3.1 Created another class to act as the board

```
public class TicTacToeTeam
{
    private ClientSocket player1;
    private ClientSocket player2;

    2 references
    public TicTacToeTeam(ClientSocket p1, ClientSocket p2)
    {
        player1 = p1;
        player2 = p2;
    }

    // check current/new game has both two players, and if not add the player to the game
    1 reference
    public bool CheckSpaceAvailable(ClientSocket player)
    {
        if (player1 == null)
            player1 = player;
        else if (player2 == null)
            player2 = player;
        else
            return false; // if players cannot add to current game, then return false

        return true; // if process success, return true
    }

    // return player 1, can access by the outside
    7 references
    public ClientSocket GetPlayer1()
    {
        return player1;
    }

    // return player  2, can access by the outside
    6 references
    public ClientSocket GetPlayer2()
    {
        return player2;
    }

    // uses to get second player/ other player from the game
    2 references
    public ClientSocket GetOtherPlayer(ClientSocket player)
    {
        if(player1 == player)
            return player2;
        else
            return player1;
    }

    // uses to check is game already has two players according to the current client
    1 reference
    public bool IsTwoPlayersAvailable()
    {
        return IsPlayerNotNull(player1) && IsPlayerNotNull(player2);
    }

    // uses to check is the player null or not
    2 references
    public bool IsPlayerNotNull(ClientSocket player)
    {
        return player != null;
    }

    // uses to check is the player equal or not

    1 reference
    public bool IsPlayerEqual(ClientSocket player)
    {
        if (player == player1)
            return true;
        else if (player == player2)
            return true;
        return false;
    }
}
```

3.3.2 Check and send a message back to the user

```
else if (text.ToLower() == "!join")
{
    // if the current games have a space then the player will be added to it
    if (board.CheckSpaceAvailable(currentClientSocket))
    {
        currentClientSocket.state = ClientState.playing;
        SendToClient("Your state changed to <playing>. Now you can play the TicTacToe game.", currentClientSocket);

        if (board.GetPlayer1() == currentClientSocket)
        {
            currentClientSocket.player = PlayerNumber.player1;
            SendToClient("You are the Player 1", currentClientSocket);
        }
        else
        {
            currentClientSocket.player = PlayerNumber.player2;
            SendToClient("You are the Player 2 ", currentClientSocket);
        }
    }
    else
    {
        SendToClient("No space available on the TicTacToe game. Please wait!", currentClientSocket);
    }
}
```

```
else if (text.ToLower().Contains("player 1"))
{
    this.game.myTurn = true;
    this.game.playerTileType = TileType.cross;
    AddToChat(text);
    AddToChat("Now it is your turn to make a move.");
}
else if (text.ToLower().Contains("player 2"))
{
    this.game.myTurn = false;
    this.game.playerTileType = TileType.naught;
    AddToChat(text);
}
```

Step 4

4.1 When it is the client's turn, they will be allowed to use the game board

```
private void AttemptMove(int i)
{
    if (ticTacToe.myTurn)
    {
        bool validMove = ticTacToe.SetTile(i, ticTacToe.playerTileType);
        if (validMove)
        {
            //tell server about it
            if (client != null)
                client.SendString("mv@p1p2%" + ticTacToe.GridToString());

            ticTacToe.myTurn = false;//call this too when ready with server
        }
        //example, do something similar from server
        GameState gs = ticTacToe.GetGameState();
        if (gs == GameState.crossWins)
        {
            client.SendString("p1@p1p2%");
        }
        if (gs == GameState.naughtWins)
        {
            client.SendString("p2@p1p2%");
        }
        if (gs == GameState.draw)
        {
            client.SendString("dd@p1p2%");
        }
    }
}
```

```
public bool SetTile(int index, TileType tileType)
{
    if (grid[index] == TileType.blank)
    {
        grid[index] = tileType;
        if (buttons.Count >= 9)
        {
            buttons[index].Invoke((Action)delegate
            {
                buttons[index].Text = TileTypeToString(tileType);
            });
        }

        return true;
    }
    else
        return false;
}
```

4.2 When they make a move, that information needs to send to the server

```
else if (text.ToLower().Contains("mv@p1p2%"))
{
    // used mv@p1p2% code to send movement information

    // temporary variable
    int count = 0;

    // check game as already two players, if not can't move
    if (board.IsPlayerEqual(currentClientSocket) && board.IsTwoPlayersAvailable())
    {
        // update the server's board uisng move information
        UpdateGame(text.Remove(0, 8));

        // send move information to current client
        SendToClient(text, currentClientSocket);

        // send move information to other player
        SendToClient(text, board.GetOtherPlayer(currentClientSocket));
        SendToClient("ct@p1p2%", board.GetOtherPlayer(currentClientSocket));
        count++;
    }

    if (count == 0)
    {
        // only one player in the game
        SendToClient("cm@p1p2%", currentClientSocket);
    }
}
```

4.2.1 If only one player in the game

```
else if (text.ToLower() == "cm@p1p2%")
{
    this.game.myTurn = true;
    this.game.ResetBoard();
    AddToChat("Other player unavailable!");
}
```

4.2.2 Inform second-player about the move

```
// send move information to other player
SendToClient(text, board.GetOtherPlayer(currentClientSocket));
SendToClient("ct@p1p2%", board.GetOtherPlayer(currentClientSocket));
```

```
if (text.ToLower().Contains("mv@p1p2%") && !text.Remove(0, 8).ToLower().Contains("mv@p1p2%"))
{
    // update the client's board uisng move information
    UpdateGame(text.Remove(0, 8));
}
```

4.2.3 Enable second-player movement

```
else if (text.ToLower() == "ct@p1p2%")
{
    // inform the player that it it their turn to make a move
    this.game.myTurn = true;
    AddToChat("Now it is your turn to make a move.");
}
```

Step 5

5.1 When the game is over, the server should update the player's scores

```
//example, do something similar from server
GameState gs = ticTacToe.GetGameState();
if (gs == GameState.crossWins)
{
    client.SendString("p1@p1p2%");
}
if (gs == GameState.naughtWins)
{
    client.SendString("p2@p1p2%");
}
if (gs == GameState.draw)
{
    client.SendString("dd@p1p2%");
}
```

```
else if (text.ToLower() == "p1@p1p2%" || text.ToLower() == "p2@p1p2%" || text.ToLower() == "dd@p1p2%")
{
    if (text.ToLower() == "p1@p1p2%")
    {
        SendToAll("Player: {" + board.GetPlayer1().username + "} wins!", null);
        // add each players scores into the database - updates wins,losses column
        DatabaseServer.AddScoreToDB(board.GetPlayer1().username, "wins");
        DatabaseServer.AddScoreToDB(board.GetPlayer2().username, "losses");
    }
    else if (text.ToLower() == "p2@p1p2%")
    {
        SendToAll("Player: {" + board.GetPlayer2().username + "} wins!", null);
        // add each players scores into the database - updates wins,losses column
        DatabaseServer.AddScoreToDB(board.GetPlayer2().username, "wins");
        DatabaseServer.AddScoreToDB(board.GetPlayer1().username, "losses");
    }
    else
    {
        SendToAll("Match between {" + board.GetPlayer1().username + "} and {" + board.GetPlayer2().username + "} draws!", null);
        // add each players scores into the database - updates draws column
        DatabaseServer.AddScoreToDB(board.GetPlayer1().username, "draws");
        DatabaseServer.AddScoreToDB(board.GetPlayer2().username, "draws");
    }
```

```
// update user's details according to the data
6 references
public static void AddScoreToDB(string username, string type)
{
    SQLiteConnection con = GetDB();

    con.Open();

    // sql query as a string
    string sql = "UPDATE user SET ";
    sql += type;
    sql += " = @p3 WHERE username = @p2";
    SQLiteCommand query = new(sql, con);
    //query.Parameters.Add(new SQLiteParameter("@p1", type));
    query.Parameters.Add(new SQLiteParameter("@p2", username));
    query.Parameters.Add(new SQLiteParameter("@p3", GetWinLossDrawTotal(username, type) + 1));
    query.ExecuteNonQuery();

    con.Close();
```

```
public static int GetWinLossDrawTotal(string username, string type)
{
    // temporary variable to store total
    int total = 0;

    SQLiteConnection con = GetDB();

    con.Open();

    // sql query as a string, get the previous count of wins/losses/draws
    string sql = "SELECT wins, losses, draws FROM user WHERE username = @p2;";
    SQLiteCommand query = new(sql, con);
    query.Parameters.Add(new SQLiteParameter("@p1", type));
    query.Parameters.Add(new SQLiteParameter("@p2", username));

    // execute the sql
    SQLiteDataReader reader = query.ExecuteReader();
    while (reader.Read())
    {
        total = Convert.ToInt32(reader[type].ToString());
    }

    con.Close();

    return total;
}
```

5.2 And server lets players know

```
// reset server game board
ResetBoard();
SendToClient("rb@p1p2%", board.GetPlayer1());
SendToClient("rb@p1p2%", board.GetPlayer2());

// end the current game and start a new game with zero players
board.NewGame();
```

```
else if (text.ToLower() == "rb@p1p2%")
{
    this.game.myTurn = false;
    this.game.playerTileType = TileType.blank;
    ResetBoard();

    currentClientSocket.state = ClientState.chatting;
    currentClientSocket.player = PlayerNumber.nonplayer;
    AddToChat("Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.");
}
```

Step 6

6.1 Outputs all scores in the database

```
else if (text.ToLower() == "!scores")
{
    List<string> data = DatabaseServer.GetScores();
    SendToClient(Environment.NewLine + "Scoreboard ---------------------------------", currentClientSocket);
    SendToClient("Wins \tLosses \tDraws \tUsername " + Environment.NewLine, currentClientSocket);
    // data[0] means username, data[1] means wins, data[2] means losses,
    // data[3] means draws, data[4] again username, like wise
    for (int i = 0; i < data.Count(); i+=4)
    {
        string rowData = data[i + 1] + "\t" + data[i + 2] + "\t" + data[i + 3] + "\t" + data[i] + Environment.NewLine;
        SendToClient(rowData, currentClientSocket);
    }
}
```

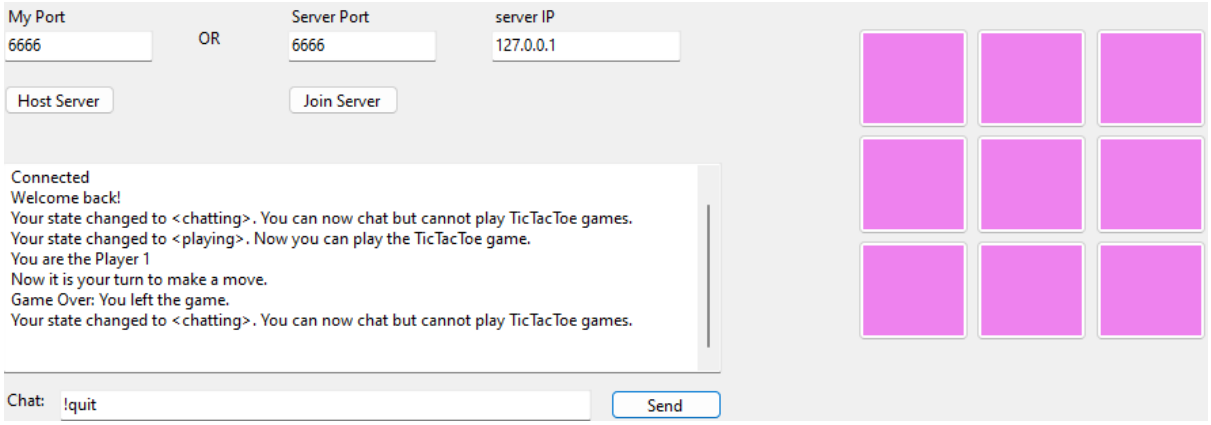
```
//  outputs all scores in the database
1 reference
public static List<string> GetScores()
{
    // temporary string list to store table data
    List<string> data = new();

    SQLiteConnection con = GetDB();

    con.Open();

    // sql query as a string
    string sql = "SELECT username, wins, losses, draws FROM user ORDER BY wins desc";
    SQLiteCommand query = new(sql, con);

    // execute the sql query check if user already exists
    SQLiteDataReader reader = query.ExecuteReader();

    while (reader.Read())
    {
        data.Add(reader["username"].ToString());
        data.Add(reader["wins"].ToString());
        data.Add(reader["losses"].ToString());
        data.Add(reader["draws"].ToString());
    }

    con.Close();

    return data;
}
```

Step 7(Extra)

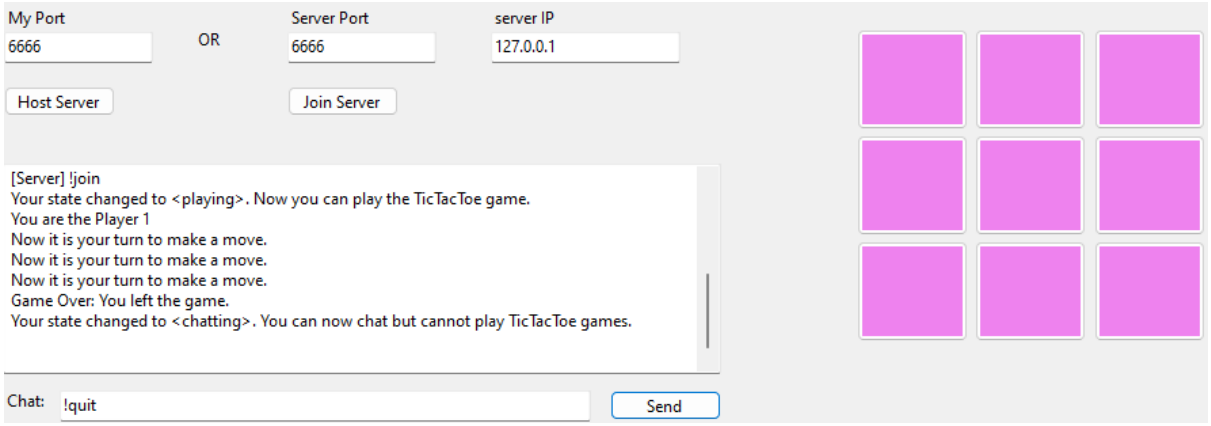
Players can leave the game whenever they wanted to

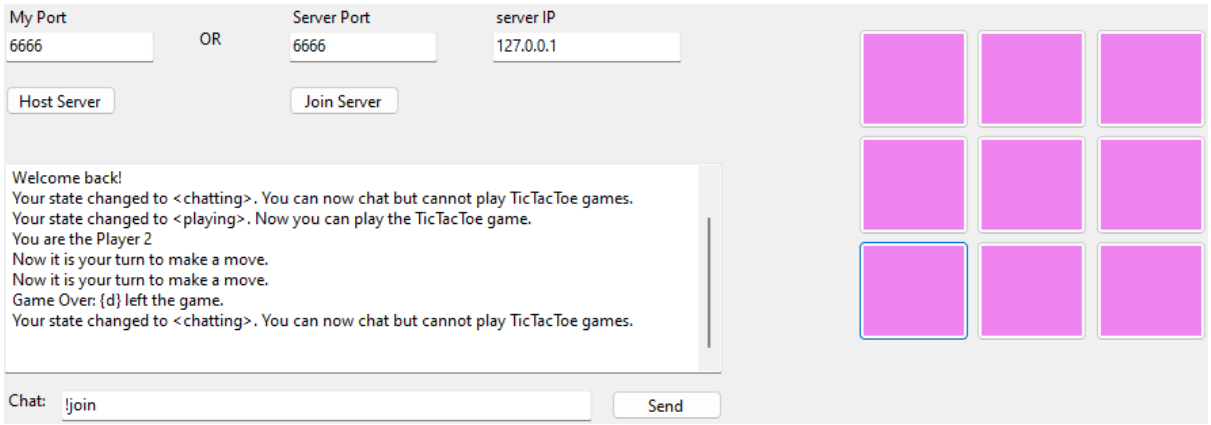
```
// extra feature - players can leave the game at anytime
else if (text.ToLower() == "!quit")
{
    if (board.IsPlayerEqual(currentClientSocket))
    {
        // reset server game board
        ResetBoard();
        SendToClient("Game Over: You left the game.", currentClientSocket);
        // reset the client board
        SendToClient("rb@p1p2%", currentClientSocket);

        // check is second player available
        if (board.IsPlayerNotNull(board.GetOtherPlayer(currentClientSocket)))
        {
            SendToClient("Game Over: {" + currentClientSocket.username + "} left the game.", board.GetOtherPlayer(currentClientSocket));
            // reset the client board
            SendToClient("rb@p1p2%", board.GetOtherPlayer(currentClientSocket));
        }

        // end the current game and start a new game with zero players
        board.NewGame();
    }
}
```

7.1 Only one player in the game



7.2 If both players in the game

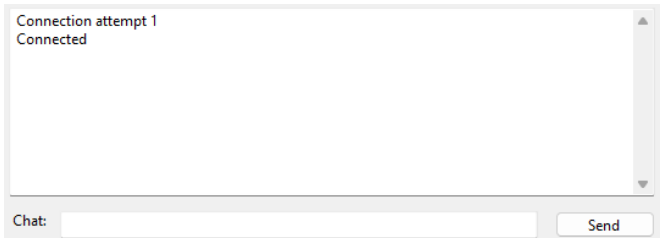


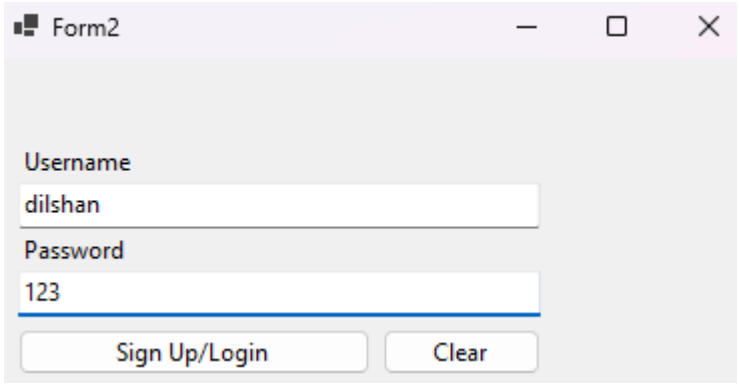


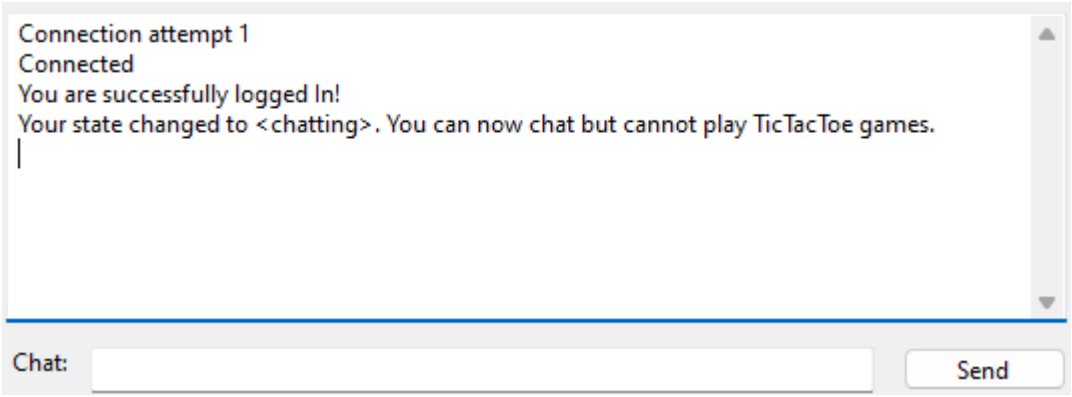
Visual Outputs

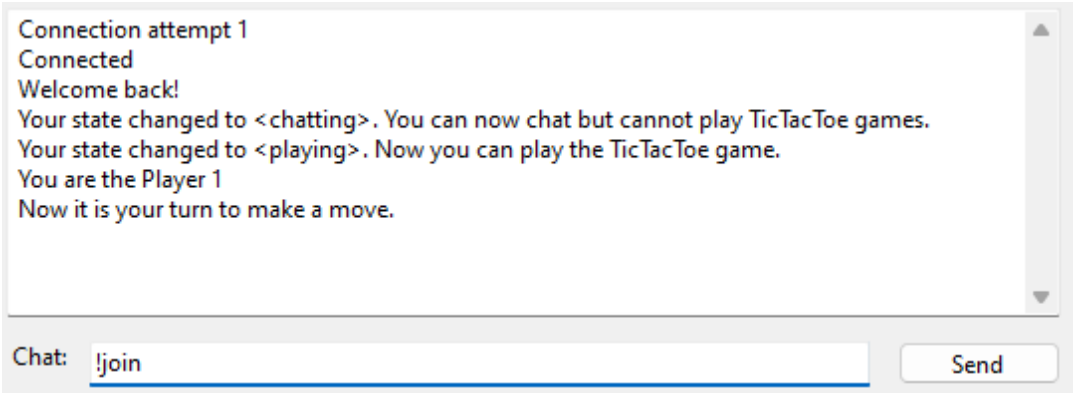
Step 2

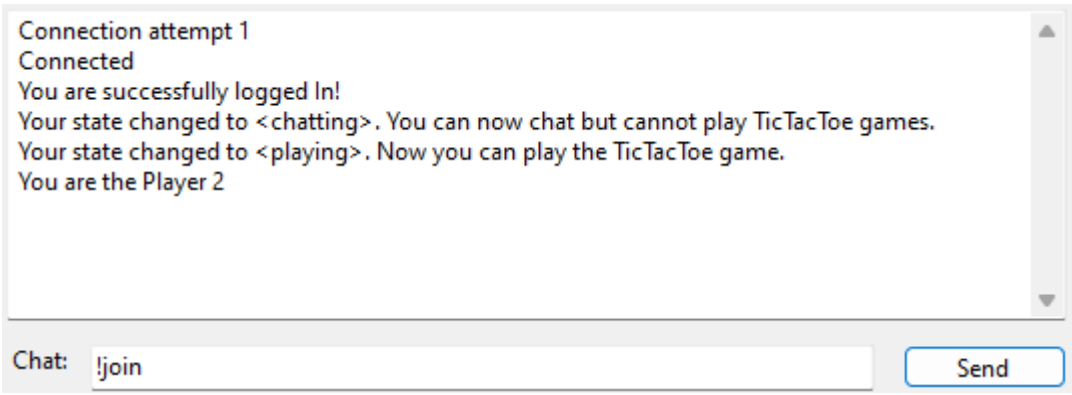
Step 3

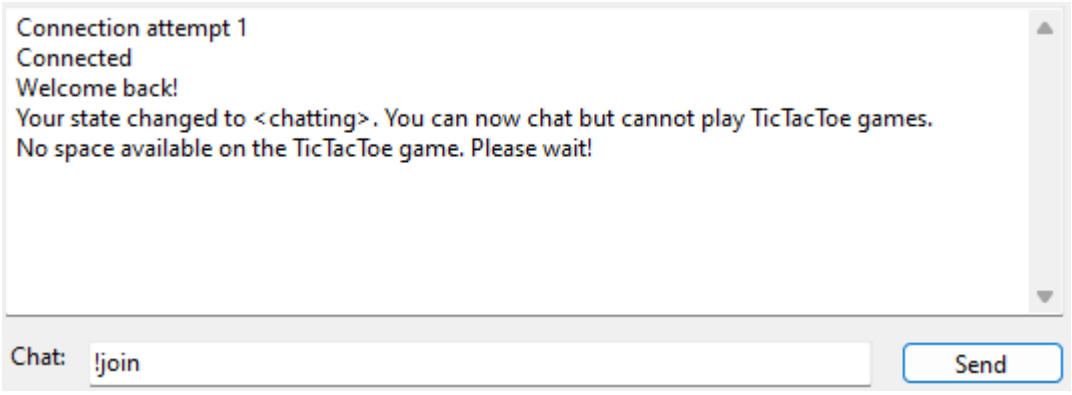




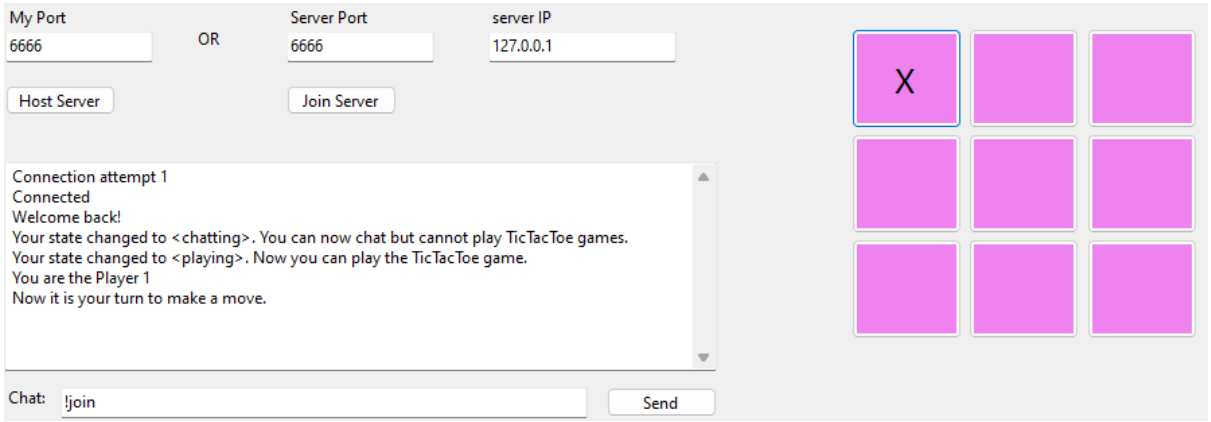

Connection attempt 1
Connected
You are successfully logged In!
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
Chat:
Send

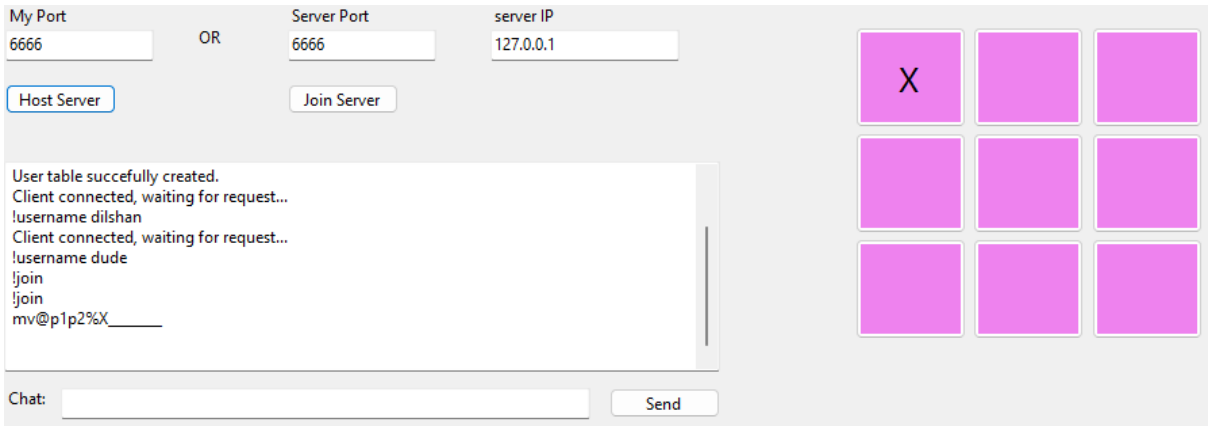

Connection attempt 1
Connected
Welcome back!
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
Your state changed to <playing>. Now you can play the TicTacToe game.
You are the Player 1
Now it is your turn to make a move.
Chat:
!join
Send

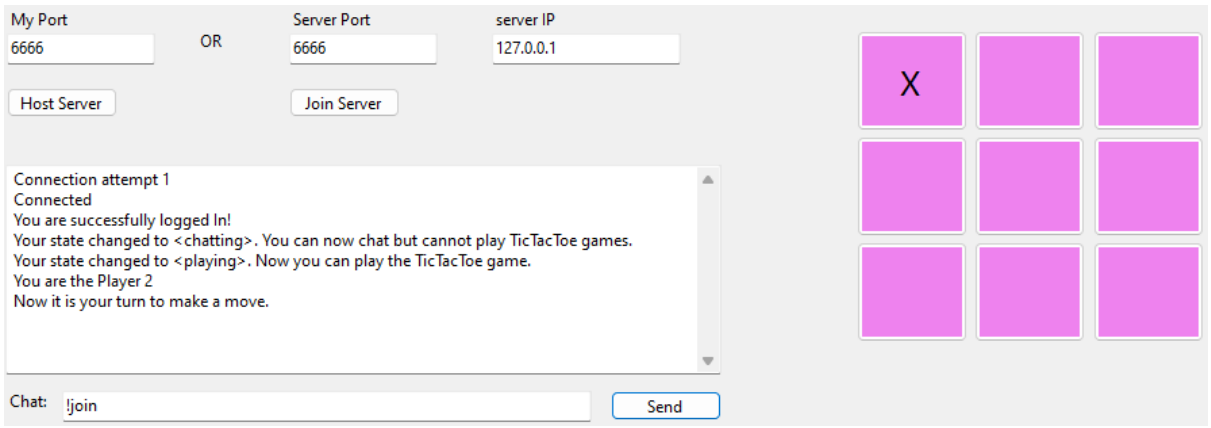

Connection attempt 1
Connected
You are successfully logged In!
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
Your state changed to <playing>. Now you can play the TicTacToe game.
You are the Player 2
Chat:
!join
Send

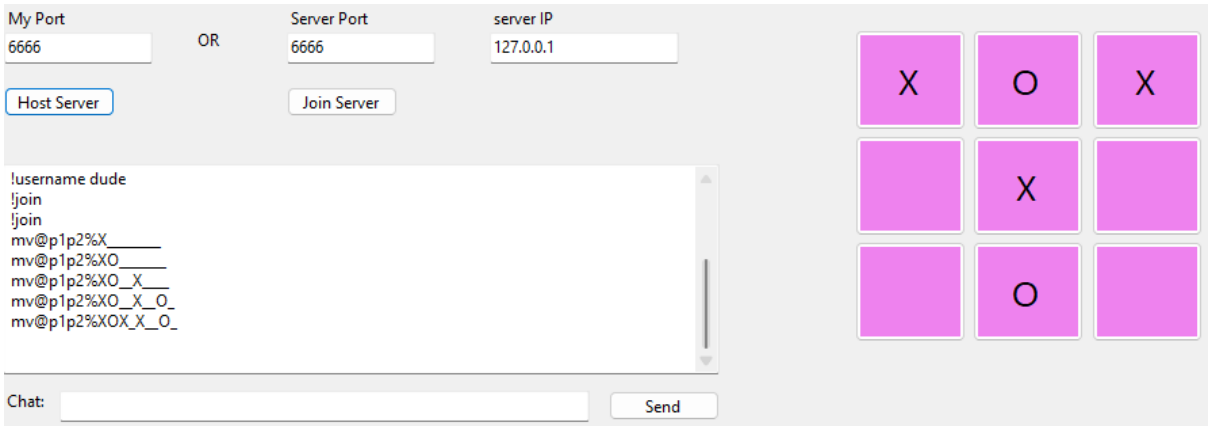

Connection attempt 1
Connected
Welcome back!
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
No space available on the TicTacToe game. Please wait!
Chat:
!join
Send

Step 4

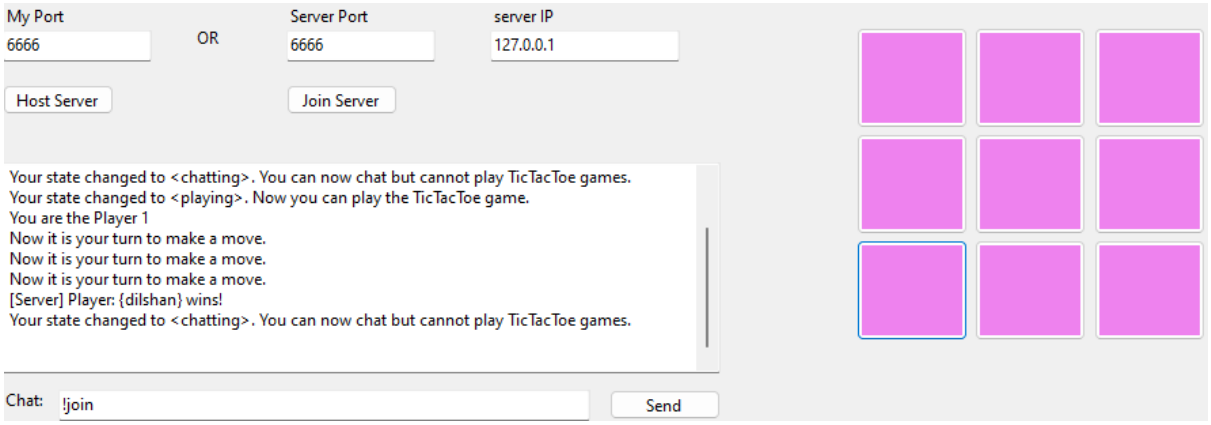

My Port
6666
OR
Server Port
6666
server IP
127.0.0.1
Host Server
Join Server
Connection attempt 1
Connected
Welcome back!
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
Your state changed to <playing>. Now you can play the TicTacToe game.
You are the Player 1
Now it is your turn to make a move.
Chat:
!join
Send
X

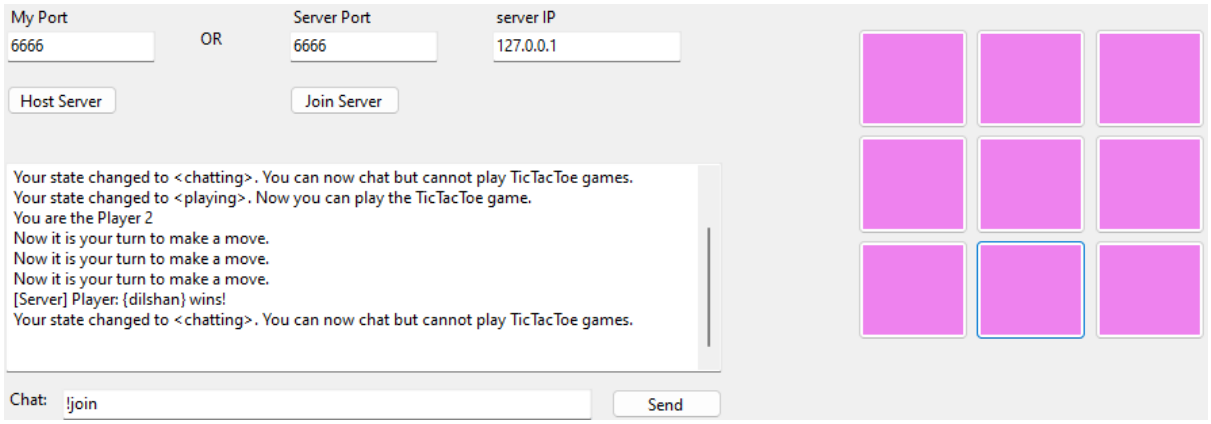

My Port
6666
OR
Server Port
6666
server IP
127.0.0.1
Host Server
Join Server
User table succefully created.
Client connected, waiting for request...
!username dilshan
Client connected, waiting for request...
!username dude
!join
!join
mv@p1p2%X_____
Chat:
Send
X


My Port
6666
OR
Server Port
6666
server IP
127.0.0.1
Host Server
Join Server
Connection attempt 1
Connected
You are successfully logged In!
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
Your state changed to <playing>. Now you can play the TicTacToe game.
You are the Player 2
Now it is your turn to make a move.
Chat:
!join
Send
X

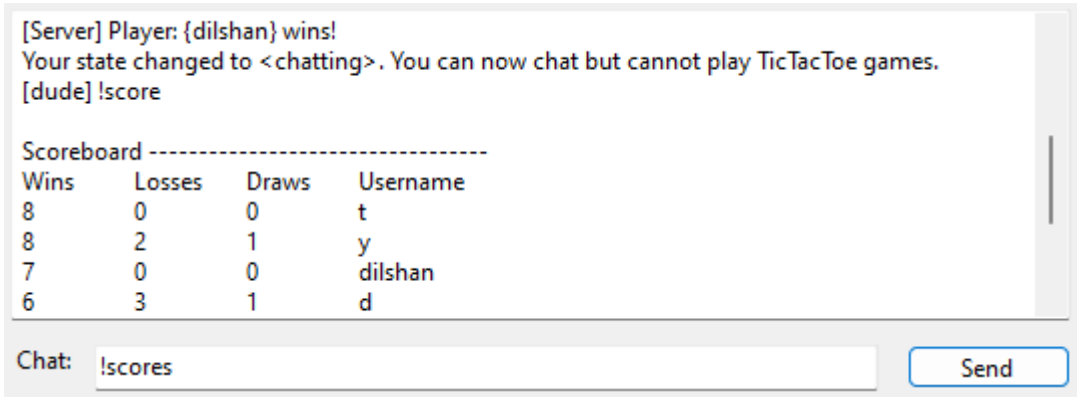

My Port
6666
OR
Server Port
6666
server IP
127.0.0.1
Host Server
Join Server
!username dude
!join
!join
mv@p1p2%X_____
mv@p1p2%XO____
mv@p1p2%XO__X___
mv@p1p2%XO__X__O_
mv@p1p2%XOX_X__O_
Chat:
Send
X
O
X
X
O

Step 5


My Port
6666
OR
Server Port
6666
server IP
127.0.0.1
Host Server
Join Server
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
Your state changed to <playing>. Now you can play the TicTacToe game.
You are the Player 1
Now it is your turn to make a move.
Now it is your turn to make a move.
Now it is your turn to make a move.
[Server] Player: {dilshan} wins!
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
Chat:
!join
Send


My Port
6666
OR
Server Port
6666
server IP
127.0.0.1
Host Server
Join Server
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
Your state changed to <playing>. Now you can play the TicTacToe game.
You are the Player 2
Now it is your turn to make a move.
Now it is your turn to make a move.
Now it is your turn to make a move.
[Server] Player: {dilshan} wins!
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
Chat:
!join
Send

Step 6


[Server] Player: {dilshan} wins!
Your state changed to <chatting>. You can now chat but cannot play TicTacToe games.
[dude] !score
Scoreboard ----------
Wins Losses Draws Username
8 0 0 t
8 2 1 y
7 0 0 dilshan
6 3 1 d
Chat:
!scores
Send