**Encrypted and concurrent socket Implementation use case of gaming application**

Group -3

**Design Document**

**1. INTRODUCTION:**

Tic-Tac-Toe (or Noughts and crosses, As and Os) is a game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game. Players soon discover that best play from both parties leads to a draw (often referred to as cat or cat's game). Hence, Tic-tac-toe is most often played by Youngers and children. The friendliness of Tic-tac-toe games makes them ideal as a pedagogical tool for teaching the concepts of good sportsmanship and the branch of artificial intelligence that deals with the searching of game trees.

**1.2 PURPOSE:**

The purpose of this project is to get a solid grasp on the fundamentals of the Socket API. Writing such an application in Cpp gives a basic understanding on how the client - server architecture works and overall, on how to use the Socket API to establish communication between client and server applications with the goal of information exchange between the two.

**1.3 OPERATING ENVIRONMENT:**

Operating environment for implementing socket use case of gaming application is:

* Client/server system
* Operating system: Linux
* Platform: Ubuntu

**1.4 FEATURES OF SYSTEM:**

The implementation of the game Tic-Tac-Toe has many features as compared to the traditional way of playing it with paper and pencil. The various features are:

• The game has been made user friendly with proper use of software

• The user can play as many games without any interruption

• The player can win the game, draw the game or will lose the game

• It’s a good brain exercise for all age group people

**2. SOFTWARE REQUIREMENTS:**

S1: Login credentials:

Correct username and password is required to login to have authentication.

S2: Instructions:

The game is played on a grid that's 3 squares by 3 squares.

You are X, your friend (or the computer in this case) is O. Players take turns

Putting their marks in empty squares.

The first player to get 3 of her marks in a row (up, down, across, or diagonally) is

the winner. When all 9 squares are full, the game is over.

S3: Display of the Game Board

S4: Server: player 1 client will send the event to the server. When the server receives it, it will broadcast it to the player 2.Player 2 will then update the grid. Then player 2 will click O.

S5: client: After player 2, Player 1 will then update the grid. Then player 1 will click X, it will send the event to the server.

S6: The first player 1 always places 'X' characters, while the second player 2 always places 'O' characters.

'X' and 'O' characters are always placed into empty squares, never on filled ones.

The game ends when there are three of the same (non-empty) character filling any row, column, or diagonal.

S7: Leader Board: A large board for displaying the ranking of the leaders in a competitive event.

**3. SYSTEM DESIGN:**

SD1: Login credentials:

The design of the login as name, age, gender

SD2: Instructions:

The design of the instructions is the rule which should follow in the are designed here

SD3: Display of the Game Board:

The design of the game is done three X three matrixes which contains of one to nine numbers.

SD4: Socket:

The connection of server side and client side socket id done by binds, list, accept.

SD5: The Game Starts:

The game design is done with connect of both server and client

SD6: Winner:

The design of the winner is done by the server or client should complete the at least three grids with the series of horizontal, vertical, diagonal.

SD7: Play Again:

The design of the play again is just it repeats the game.

SD8: Leader Board:

The design of the leader board is done for the winner of the game.

**4.   UNIT TEST:**

1. Client:

UT1: Provide valid username and password.

UT2: Create a valid socket to initiate the communication

UT3: Client must pass only one argument

UT4: player board initialization

UT5: The game starts from client side and it declares the player wins or loses

UT6: If the connect () function fails then the client must receive an error message of connection failure.

UT7: If read () or write () function fails then the client won’t be able to send or receive the data.

2. Server:

UT1: While defining the structure mention valid elements.

UT2: The port no. in the client side and server side should be same if not connection not establish.

UT3: The bind () and listen () should bind the client and server if fail must return the Binding error while listen () function must listen on particular port number.

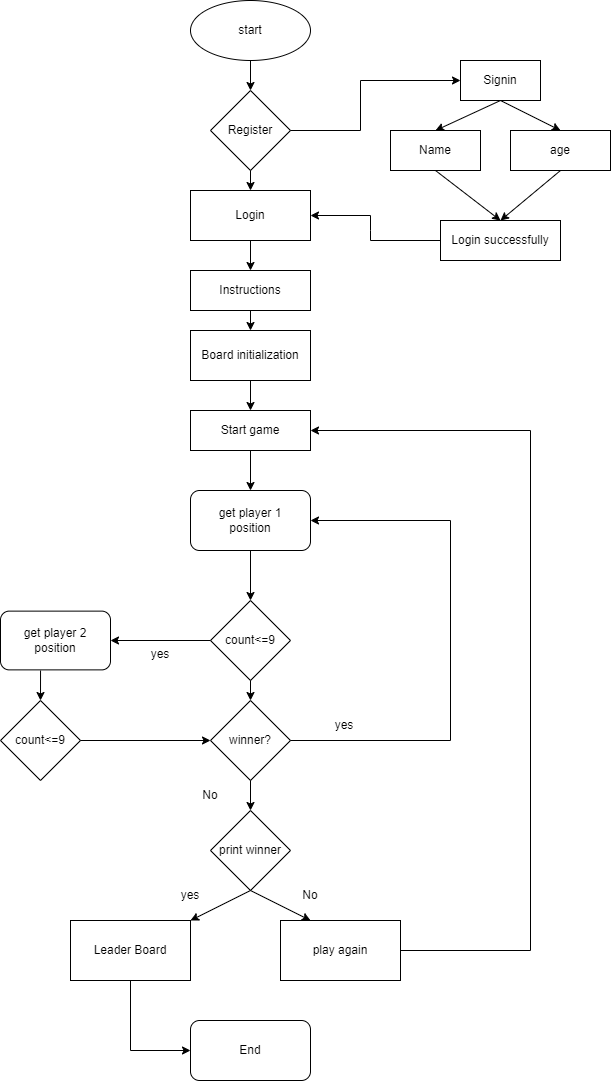
UT4: player board initiation

UT5: The game starts from client side and it declares the player wins or loses

UT6:  Send and receive the request

       Use the read () and write () file descriptor to send and receive the data from server.

**5. FLOWCHART:**

****