

Encrypted and concurrent socket implementation using gaming application (TIC-TAC-TOE GAME-USING-CLIENT-SERVER-SOCKET)

**High Level Design & Low Level Design**

The purpose of this document is to provide with a template for documenting both HLD & LLD.

**Document Control :**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Revision History** | | | | | | | | |
|  |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| 15.11.2022 |  |  |  | | | |  | |
|  |  |  |  | | | |  | |
|  |  |  |  | | | |  | |

[**1. Introduction**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3znysh7)  **5**

[1.1. Key project Objective](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2et92p0)  5

[1.2. ABBREVITIONS](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.tyjcwt)  5

[1.3. FEATURES](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3dy6vkm)  5

[1. 4. PURPOSE](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1t3h5sf)  5

[1.5. Project Scope 5](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.4d34og8)

[1.6. Functional Overview](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3rdcrjn)  6

[1.7. Assumptions, Dependencies & Constraints](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.26in1rg)  7

[1.8. Risks](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.lnxbz9) 7

[**2. Design Overview**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.35nkun2)  **8**

[2.1. Design Objectives](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1ksv4uv)  10

[*2.1.1. Recommended Architecture*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.44sinio) 10

[2.2. Architectural Strategies](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2jxsxqh)  10

[*2.2.1. Design Alternative*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.z337ya) 10

[*2.2.2. Reuse of Existing Common Services/Utilities*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3j2qqm3) 10

[*2.2.3. Creation of New Common Services/Utilities*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1y810tw)  11

[*2.2.4. User Interface Paradigms*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.4i7ojhp) 11

[*2.2.5. System Interface Paradigms*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2xcytpi) 11

[*2.2.6. Error Detection / Exceptional Handling*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1ci93xb) 11

[*2.2.7. Memory Management*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3whwml4) 11

[*2.2.8. Performance*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2bn6wsx) 11

[*2.2.9. Security*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.qsh70q) 11

[*2.2.10. Concurrency and Synchronization*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3as4poj) 11

[*2.2.11. Housekeeping and Maintenance*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1pxezwc) 11

[**3. System Architecture**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.49x2ik5)  **12**

[3.1. System Architecture Diagram. (Not Necessary)](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2p2csry)  12

[3.2. System Use-Cases](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.147n2zr)  13

[3.3. Subsystem Architecture](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3o7alnk)  14

[3.4. System Interfaces](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.23ckvvd)  14

[*3.4.1. Internal Interfaces*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.ihv636)  *14*

[*3.4.2. External Interfaces*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.32hioqz) *14*

[**4. Detailed System Design**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1hmsyys)  **14**

[4.1. Key Entities](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.41mghml)  14

[4.2. Detailed-Level Database Design](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2grqrue)  14

[*4.2.1. Data Mapping Information*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.vx1227)  14

[*4.2.2. Data Conversion*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3fwokq0)  14

[4.3. Archival and retention requirements](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1v1yuxt)  14

[4.4. Disaster and Failure Recovery](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.4f1mdlm)  15

[4.5. Business Process workflow](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2u6wntf)  15

[4.6. Business Process Modeling and Management (as applicable)](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.19c6y18)  15

[4.7. Business Logic](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3tbugp1)  15

[4.8. Variables](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.28h4qwu)  15

[4.9. Activity / Class Diagrams (as applicable)](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.nmf14n)  15

[4.10. Data Migration](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.37m2jsg)  15

[*4.10.1. Architectural Representation*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1mrcu09)  15

[*4.10.2. Architectural Goals and Constraints*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.46r0co2)  16

[*4.10.3. Logical View*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2lwamvv)  16

[*4.10.4. Architecturally Significant Design Packages*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.111kx3o)  16

[*4.10.5. Data model*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3l18frh)  16

[*4.10.6. Deployment View*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1egqt2p)  16

[**5. Environment Description**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3ygebqi)   **16**

[5.1. Time Zone Support](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2dlolyb)  16

[5.2. Language Support](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.sqyw64)  16

[5.3. User Desktop Requirements](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3cqmetx)  16

[5.4. Server-Side Requirements](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1rvwp1q)  16

[*5.4.1. Deployment Considerations*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.4bvk7pj)  17

[*5.4.2. Application Server Disk Space*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2r0uhxc)  17

[*5.4.3. Database Server Disk Space*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1664s55)  17

[*5.4.4. Integration Requirements*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3q5sasy)  17

[*5.4.5. Jobs*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.25b2l0r)  17

[*5.4.6. Network*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.kgcv8k)  17

[*5.4.7. Others*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.34g0dwd)  17

[5.5. Configuration](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1jlao46)  17

[*5.5.1. Operating System*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.43ky6rz)  17

[*5.5.2. Database*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.2iq8gzs)  17

[*5.5.3. Network*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.xvir7l)  17

[*5.5.4. Desktop*](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.3hv69ve)  17

[**6. References**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.1x0gk37)   **17**

[**7. Appendix**](https://docs.google.com/document/d/12Zkn-7XhoUFHzHxoF7Trfcw706LFqpnO/edit#heading=h.4h042r0)  **18**

**1 Introduction**

Tic-tac-toe is a straightforward two-player game that, if both players play their best, will always end in a tie. The game is also known as ‘X’ s and ‘O’ s or zeros and crosses.

A computer or other device can be used to play the game of tic tac toe, which is typically played by drawing on paper. This timeless game provides the foundation for other ones, like Connect 4.

Around the first century B.C., a primitive version of the game was played in the Roman Empire. Three pebbles at a time is what the name "terni lapilli" denotes. Roman ruins have been discovered to be covered with chalk grid patterns from the game. Ruins in ancient Egypt have also yielded evidence of the game.

The game's British moniker, "noughts and crosses," saw its first print appearance in 1864. The term "tick-tack-toe" first appeared in literature in 1884, although it referred to a children's game played on a slate.

The crucial Rules of the Game :

* The game must be played by two players (in this program between HUMAN and COMPUTER).
* Both players mark their cells with the letters "O" and "X".
* The game ends when one of the players fills an entire row, column or diagonal with either the character ('O' or 'X') of that player.
* If no one wins, the match is considered a draw.

**1.1 Key project Objective**

Implementing an Online TIC-TAC-TOE Game

**1.2 ABBREVITIONS**

|  |  |
| --- | --- |
| OTTTG | Online TIC-TAC-TOE Game |
| OG | Online Game |

**1.3 FEATURES**

* 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.

**1.4 PURPOSE**

• Experience with socket programming, experience with implementing a network application.

**1.5 PROJECT SCOPE**

Online Voting System is a system which enables all citizens to cast their vote online.

The purpose is to increase the voting percentage across the country, as in the

present system people have to visit the booth to cast their vote and those people

who live out of their home town are not able to cast vote during the elections. So

due to this the voting percentage across the country is very less. Through this

software those people who live out of their home town will also be able to cast their

votes as this system is online.

And this software is on a block

Online Voting System is a system which enables all citizens to cast their vote online.

The purpose is to increase the voting percentage across the country, as in the

present system people have to visit the booth to cast their vote and those people

who live out of their home town are not able to cast vote during the elections. So

due to this the voting percentage across the country is very less. Through this

software those people who live out of their home town will also be able to cast their

votes as this system is online.

And this software is on a block

The game play will be simple :

1. There will be a simple square game board divided into nine tiles or grid spaces. When the player clicks on one of the grid spaces, it will be assigned either an "X" or an "O". The game is over when one player claims 3 grid spaces in a row or there are no moves left. The game will have a small amount of polish to make it complete.
2. At the start of the game, the board will not be active until the first player has chosen whether they are to play "X" or "O". A panel will indicate whose turn it is. When the game is over, a banner will display the winner or announce a draw if no one wins. A restart button will be displayed when the game is over, returning the game to the starting state when clicked.

**The game will need a few basic elements.**

* A background providing a backdrop for the the entire game.
* An element that will be our game board.
* An element, or set of elements, that breaks the game board up into nine areas in an even grid.
* Nine tiles that can be assigned either an "X" or and "O", but once assigned these values will persist and not be changeable by the players - either the current player or the opponent.
* Logic to change sides when a player takes their turn.
* Logic to check for a "Win" condition, allowing for draws where no one wins.
* A panel that displays who is the winner when the game is over.

**For polish, towards the end of the project, we could have some other, helpful elements**.

* A way to choose the starting player's side, "X" or "O".
* An indicator of whose turn it is.
* A restart button.
* Very basic instructions.

Lastly, as part of this exercize, we are going to be using nothing more than the elements provided by Unity’s built in UI toolset.

**1.6 FUNCTIONAL OVERVIEW**

[Tic-Tac-Toe](https://en.wikipedia.org/wiki/Tic-tac-toe) is a simple game for two players that we enjoyed playing as kids (especially in boring classrooms). The game involves 2 players placing their respective symbols in a 3x3 grid. The player who manages to place three of their symbols in horizontal/vertical/diagonal row wins the game. If either player fails to do so the game ends in a draw. If both the people always play their optimal strategies the game always ends in a draw.

1. Login with admin / user
2. Instructions
3. Broad Initialization
4. Start game
5. Get player 1 position
6. Get player 2 position
7. Print winner
8. Leader Board
9. Play again
10. End

**1.7  Assumptions, Dependencies & Constraints**

OPERATING SYSTEMS:

Operating environment for implementing DNS are:

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

**1.8  Risks**

            No Risk(As it is for Gamming purpose)

# **2 Design Overview**

1. Login Credentials
   1. In this module the credentials entered by the server/client are then validated by the system. If the server/client enters valid credentials then it will move to the further step else the system will prompt the server/client with an error message.
2. Start
   1. This is the start block which indicates the start of the program.  
      which will accept the client and server credentials (like the username and password). On validation of these credentials the system will allow the client and server to further communicate else if the credentials are wrong it will display an error message indicating “Invalid Credentials”.
3. Server Login
   1. This is the module used for the server login where the server has to enter the credentials (username and password).
4. Client Login
   1. This is the module used for the client login where the client, if new, has to register with an username and password and then login by entering the same credentials (username and password). If already registered the client can login with the credentials.
5. Instruction
6. The game is played on a grid that's 3 squares by 3 squares.
7. You are X, your friend (or the computer in this case) is O. Players take turns putting their marks in empty squares.
8. The first player to get 3 of her marks in a row (up, down, across, or diagonally) is the winner.
9. When all 9 squares are full, the game is over. If no player has 3 marks in a row, the game ends in a tie.

1. Board initialization

In this game, two players will be played and you have one print board on the screen where from 1 to 9 number will be displayed or you can say it box number. Now, you have to choose X or O for the specific box number. For example, if you have to select any number

then for X or O will be shown on the print board, and turn for next will be there. The task is to create a C++ program to implement a 3×3 Tic-Tac-Toe game for two players.

1. Start game

It’s starts the game for the two players.

1. Players ’1’ positions

The first player go first. Though traditionally, the first player goes with "X", you can allow the first player to decide whether to go with "X"s or to go with "O"s.These symbols will be

placed on the table, in an attempt to have three of them in a row. If you're going first,then the best move you can make is to move into the center. This will maximize your chances of winning since you'll be able to create a row of three "X"s or "O"s in more combinations (4) this way than if you chose a different square.

1. Players ’2’ positions

The second player go second. After the first player goes, then the second player should put down his symbol, which will be different from the symbol of the first player. The second player can either try to block the first player from creating a row of three or focus on creating their row of three. Ideally, the player can do both

1. Count(moves of the players)

Keep alternating moves until one of the players has drawn a row of three symbols or until no one can win.The first player to draw three of their symbols in a row, whether it is horizontal, vertical,or diagonal, has won tic-tac-toe. However, if both players are playing with optimal strategy, then there's a good chance that no one will win because you will have blocked all of each other's opportunities to create a row of three.

1. Winner

When you're the first one up, there is a simple strategy on how to win tic tac toe: put your 'X' in any corner. This move will pretty much send you to the winner's circle every time, so long as your opponent doesn't put their first 'O' in the center box.

1. Leader board

A large board for displaying the ranking of the leaders in a competitive event (such as a golf tournament)

1. Play again

Played at the beginning of a tournament or just prior to the tournament depending on how the tournament is defined. In a play-in, the lowest qualifiers and/or participants who have earned conditional qualification compete for qualification to the main portion of the tournament. This gives an added advantage to the higher or direct qualifiers, allowing them to rest and/or play non-elimination games, while the lower teams extend themselves by playing in elimination games.

1. End

This ensures that the program has terminated.

## 2.1 Design Objectives

## Create login credential page for both server and clients.

Its show the instruction and Board initialization

And stats the games. players gets position.

After completing the game its shows winner.

And then its goes for play again or leader board(your choice)

End.

### 2.1.1 Recommended Architecture

Generic

## 2.2 Architectural Strategies

* Header files
* Structures

### 2.2.1  Design Alternative

NA

### 2.2.2 Reuse of Existing Common Services/Utilities

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include<unistd.h>

#include<string.h>

### 2.2.3 Creation of New Common Services/Utilities

           NA

2.2.4 User Interface Paradigms

          Command Line Interface: Terminal

### 2.2.5 System Interface Paradigms

           Command Line Interface: Terminal

2.2.6 Error Detection / Exceptional Handling

           Error detection :

1. Invalid Domain Name
2. IP address does not exist
3. Errors will be handled by perror

2.2.7 Memory Management

NA

2.2.8 Performance

NA

### 2.2.9 Security

                  For security purposes the system asks for login credentials from server and client.

2.2.10 Concurrency and Synchronization

      NA

### 2.2.11 Housekeeping and Maintenance

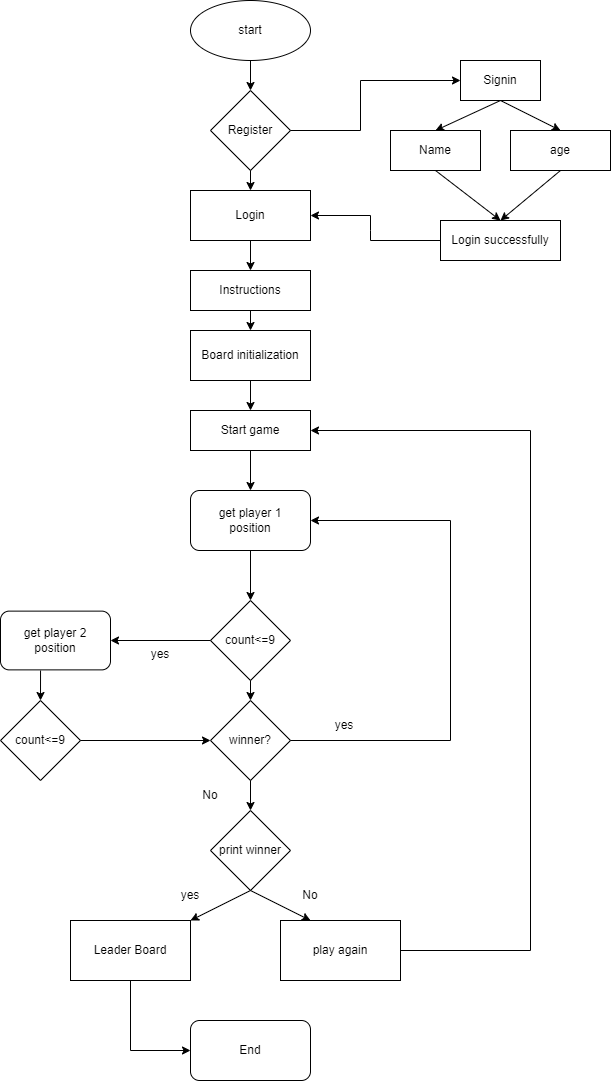
        NA

# **3 System Architecture**

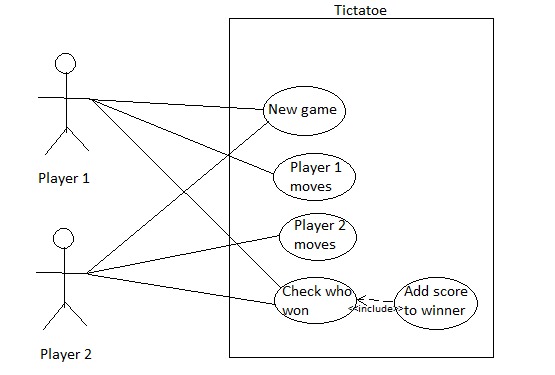
**LEVEL 0 DFD:**

**LEVEL 1 DFD:**

## 3.1 System Architecture Diagram



## 3.2  System Use-Cases



## 3.3  Subsystem Architecture

    NA

## 3.4  System Interfaces

      NA

### 3.4.1 Internal Interfaces

     NA

### 3.4.2 External Interfaces

     NA

# 

# **4 Detailed System Design**

The code starts by declaring the struct sockaddr\_in and hostent. After that client socket will be created. Using connect() the client establishes connection with the server. After a successful connection client must enter the domain name the validation of this domain name will be done by IP Address.The message will be sent to the server.

The server will first do binding after that it will be listening on a particular port which will be the same for server and client.Accept() will be called which will accept the connection from the client. The received domain name will be accepted by the server and will search for the equivalent domain .after fetching the IP address server will send the IP address to the client.

## 4.1  Key Entities

* Valid domain/host name
* IP Address

## 4.2  Detailed-Level Database Design

    NA

### 4.2.1  Data Mapping Information

Mapping the IP address from server side is done by gethostbyname()

### 4.2.2  Data Conversion

Converting the IP address(IPv4 format) from binary to standard text format using   inet\_ntop()

## 4.3  Archival and retention requirements

NA

## 4.4  Disaster and Failure Recovery

* We don’t have any control over the system. In case of failure, source code is safe.
* Use of Git.

## 4.5  Business Process workflow

NA

## 4.6  Business Process Modeling and Management (as applicable)

                  NA

## 4.7  Business Logic

NA

## 4.8  Variables

NA

## 4.9  Activity / Class Diagrams (as applicable)

       Pseudocode for Server Side:

       Create a server socket

       Bind socket to specific port where client will connect with the server

Game start from server side

Start marking

       Display the winner

       Quit or play again

End

       Pseudocode for Client Side:

       Create a Client socket

       Bind socket to specific port where server will connect with the client

Game start from Client side

Start marking

       Display the winner

       Quit or play again

End

## 4.10 Data Migration

NA

### 4.10.1 Architectural Representation

NA

### 4.10.2 Architectural Goals and Constraints

The project is just for educational purposes.

### 4.10.3 Logical View

NA

### 4.10.4 Architecturally Significant Design Packages

NA

### 4.10.5 Data model

NA

### 4.10.6 Deployment View

NA

# **5 Environment Description**

GCC: In Linux, the GCC stands for GNU Compiler Collection. It is a compiler system for the various programming languages. It is mainly used to compile the C and C++ programs.

Socket Programming: Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

UBUNTU: Ubuntu is an open-source operating system (OS) based on the Debian GNU/Linux distribution. Ubuntu incorporates all the features of a Unix OS with an added customizable GUI, which makes it popular in universities and research organizations. Ubuntu is primarily designed to be used on personal computers, although a server edition does also exist.

GITHUB: GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

## 5.1 Time Zone Support

             NA

## 5.2 Language Support

             NA

## 5.3 User Desktop Requirements

Linux, Ubuntu

## 5.4 Server-Side Requirements

Linux, Ubuntu

### 5.4.1 Deployment Considerations

          NA

### 5.4.2 Application Server Disk Space

          NA

### 5.4.3 Database Server Disk Space

          NA

### 5.4.4 Integration Requirements

            NA

### 5.4.5 Jobs

                              NA

### 5.4.6 Network

           NA

### 5.4.7 Others

            NA

## 5.5  Configuration

                    NA

### 5.5.1 Operating System

Linux desktop editions with 8 GB RAM- A GUI-based LINUX system must be   used

### 5.5.2 Database

      NA

### 5.5.3  Network

*[Describe the Network configuration requirements here. Details of all the Network Components etc.]*

### 5.5.4  Desktop

* CPU : Intel i3/i5/i7 generation 3 and later
* RAM: 4GB or greater - For optimal performance, 6GB or 8GB are recommended if you will be running multiple browser tabs and/or multiple applications at the same time
* Internal memory:476 GB SSD/HDD.

**6. References**

<https://man7.org/linux/man-pages/index.html>

[Introduction to Sockets Programming in C using TCP/IP](https://www.csd.uoc.gr/~hy556/material/tutorials/cs556-3rd-tutorial.pdf)

https://secure.img1-fg.wfcdn.com/docresources/640720000/311/3116108.pdf

# **7 Appendix**

**Change Log**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **QMS Template Version Control (Maintained by QA)** | | | | | |
|  |  |  |  |  |  |
| **Date** | **Version** | **Author** | | **Description** | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |
|  |  |  | |  | |