TIC-TAC-TOE USING SOCKET SERVER

A PROJECT REPORT

Submitted by

C.KARTHIK REDDY(RA2111030010081)
K.HEMANTH REDDY(RA2111030010087)
M.E.V.S AKHIL VARMA(RA2111030010099)
K.LOKESH REDDY(RA2111030010105)
G.PRANAY(RA2111030010115)

Under the guidance of

D.Saveetha
Assistant Professor, Department of Networking and Communications

In partial satisfaction of the requirements for the course of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

with Specialization in CYBER SECURITY

DEPARTMENT OF NETWORKING AND COMMUNICATIONS
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603203

November 2023

TIC-TAC-TOE USING SOCKET SERVER

A PROJECT REPORT

Submitted by

C.KARTHIK REDDY(RA2111030010081)

K.HEMANTH REDDY(RA2111030010087)

M.E.V.S AKHIL VARMA(RA2111030010099)

K.LOKESH REDDY(RA2111030010105)

G.PRANAY(RA2111030010115)

Under the guidance of

D.Saveetha

Assistant Professor, Department of Networking and Communications

In partial satisfaction of the requirements for the course of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

with Specialization in CYBER SECURITY

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

November 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this course project of Computer Networks (18CSC302J) report titled **"Tic-Tac-Toe using Socket Server "** is the Bonafide work of **KARTHIK(RA2111030010081),HEMANTH(RA2111030010087),AKHIL(RA2111030010099),LOKESH(RA2111030010105),PRANAY(RA2111030010115)** who carried out the project work under my supervision.

Certified further, that to the best of my knowledge the work reported here in does not form part of any other course project report on the basis of which a degree was conferred on an earlier occasion for this or any other candidate.

Dept. of Networking and Communications * SRM IST * Kattankulathur 603 203

D. Saveetha

Ms.D.Saveetha
Assistant Professor
Department of Networking and Communications
SRM Institute of Science and Technology,
Kattankulathur

Dr. Annapurani Panaiyappan K
Professor and Head of the Department
Department of Networking and Communications
SRM Institute of Science and Technology,
Kattankulathur

Department of Networking and Communications

SRM Institute of Science and Technology

Own Work Declaration Form

Course: B.Tech in Computer Science and Engineering with Specialization in Cyber Security

Student Names: Karthik , Hemanth, Akhil ,Lokesh, Pranay

Registration Number: RA2111030010081, RA2111030010087,

RA2111030010099, RA2111030010105, RA2111030010115

Title of Work: TIC-TAC-TOE USING SOCKET SERVER

We, hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not my own.
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)

I understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:

I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my / our own work, except where indicated by referring, and that I have followed the good academic practices noted above.

RA2111030010081
RA2111030010087
RA2111030010099
RA2111030010105
RA2111030010115

If you are working in a group, please write your registration numbers and sign with the date for every in your group.

C. KARTHIK Reddy
K. Hemanth Reddy
M. ENOS AKSHAY VARMA
K. LOKESH Reddy
G. Pranay

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Dr. C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology for the facilities extended for the project work and his continuous support.

We express our profound gratitude to our **Dr. T. V. Gopal,** Dean-CET, SRM Institute of Science and Technology for bringing out novelty in all executions.

We would like to express my heartfelt thanks to **Dr. Revathi Venkataraman,** Professor & Chairperson, School of Computing, SRM Institute of Science and Technology for imparting confidence to complete my course project.

We wish to express my sincere thanks to **Dr. Annapurani Panaiyappan K,** Professor and Head, Department of Networking and Communications, SRM Institute of Science and Technology for their constant encouragement and support.

We are highly thankful to my Course project Faculty **D.Saveetha,** Assistant Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for her assistance, timely suggestion and guidance throughout the duration of this course project.

TABLE OF CONTENTS

ABSTRACT

The "TIC-TAC-TOE Multiplayer Game" is a client-server-based project designed to bring the classic game of Tic-Tac-Toe to a modern, connected world. The system consists of two main components: a server that manages game sessions and a client application that allows users to connect and play against each other. Players can compete with friends or strangers over the internet, making it a fun and engaging way to test their strategic skills. The server handles game logic, player matching, and communication, ensuring a seamless multiplayer experience. The client application offers an intuitive interface for users to make moves and interact with the game. This project incorporates networking, threading, and socket programming to establish communication between the server and clients. It also includes features like visual representation of the game board. The TIC-TAC-TOE Multiplayer Game is an excellent example of client-server programming, highlighting the use of sockets for data exchange, enabling players to engage in a classic game in a contemporary and connected setting. Whether for educational purposes or as a basis for more complex multiplayer games, this project showcases the fundamentals of client-server architecture and offers an enjoyable gaming experience for the users.

INTRODUCTION

One of the most universally played childhood games is TIC TAC TOE. An interactive TIC TAC TOE game is developed where two players will be able to play against each other in a suitable GUI by using proper mouse movements. This game will start by showing a simple display, prompt the user for a move and then prints the new board. The board looks like a large hash symbol (#) with nine slots that can each contain an X, an O, or a blank. There are two players, the X player and the O player. By default, player1 (the O player) takes the initiative. The game will end in two situations: a win, when one player gets three in a row, horizontally, vertically or diagonally. A draw, when there are no remaining places to choose and neither of them has won. Here are some rules for the game:

- The player with symbol 'O' goes first
- Players alternate placing X s and O s on the board until either
- one player has three in a row, horizontally, vertically or diagonally; or
- all nine squares are filled.
- The player who can draw three X s or three O s in a row wins.
- If all nine squares are filled and none of the players have three in a row, the game is a draw.

Problem Statement

Develop a TIC-TAC-TOE multiplayer game that utilizes client-server programming principles to enable players to compete in real-time across a networked environment. The goal is to create a seamless and engaging multiplayer gaming experience while showcasing proficiency in socket programming, multi-threading, and game logic implementation.

LITERATURE SURVEY

S. No	Paper Title	Summary	Methodology/Algorithm Used
1	The Socket Programming and Software Design for Communication Based on Client/Server **Year of Published:** 2009 **Published By:** IEEE	This paper introduces the application of the client/server(C/S) mode, the concept and the programming principle of the socket based on C/S.	Connection Oriented Service Program
2	A design and implementation of active network socket programming **Year of Published:** 2002 **Published By:** IEEE	The concept of programmable nodes and active networks introduces programmability into communication networks.	Active Network Socket Programing (ANSP)
3	Design and Implementation of Client-Server Based Application Using Socket Programming in a Distributed Computing Environment **Year of Published:** 2017 **Published By:** IEEE	This research study discusses the detail overview in developing a client server-based application using socket programming in a distributed computing environment.	Principles and concepts behind socket programming as well as the libraries available in Java.
4	IP to IP Calling Through Socket Programming **Year of Published:** 2021 **Published By:** IEEE	It is used to connect the people in same Wi-Fi network to exchange their communication without having any cost.	Datagram Socket Programing
5	Analysis of Socket Communication Technology Algorithms Under TCP/IP Protocol **Year of Published:** 2019 **Published By:** IEEE	It is used to connect the people in same Wi-Fi network to exchange their communication without having any cost.	The branching algorithm mainly involves enhancing data learning.

REQUIREMENTS

HARDWARE REQUIREMENT [minimum requirement]:

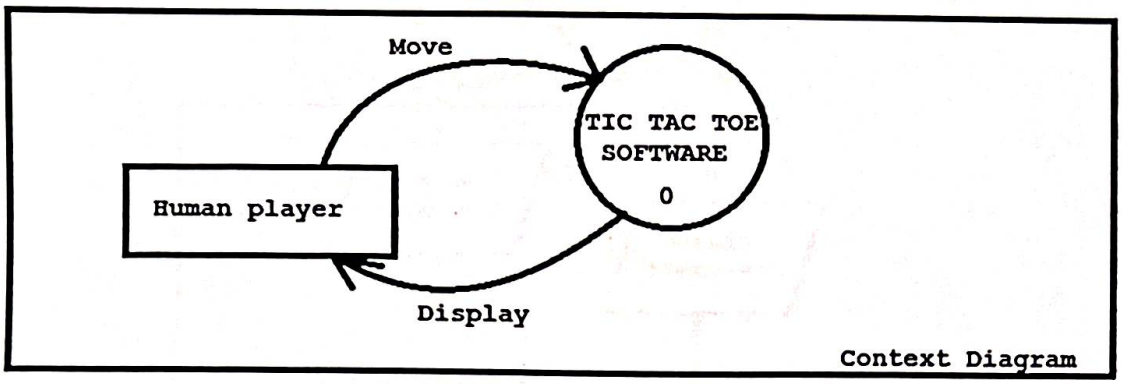
- Minimum RAM: - 1GB
- Minimum Hard Disk: - 128GB
- Processor: - Intel Pentium 4(1.50 GHz) or above

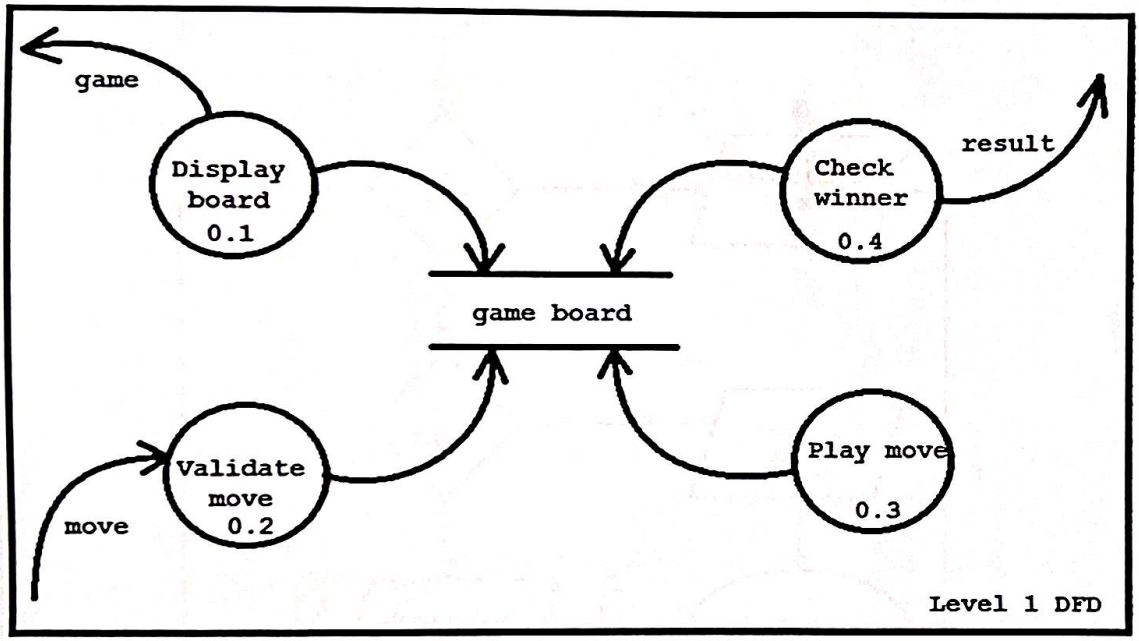
SOFTWARE REQUIREMENT [minimum requirement]:

- Operating System: - Support for both LINUX and WINDOWS users
- Back End: - Python 3.6.0 Interpreter
- Front End Language: - Python3
- Front Design: - Tkinterface

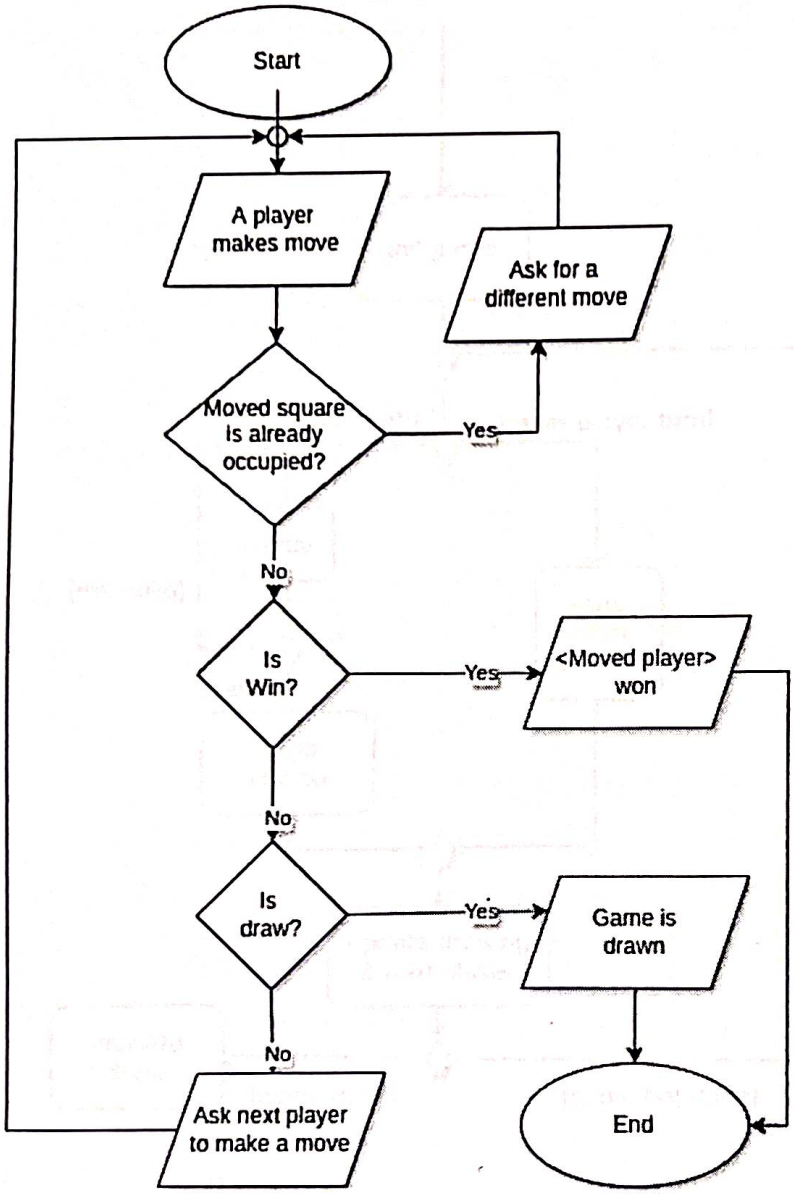
ARCHITECTURE AND DESIGN

1.1. Data Flow Diagram

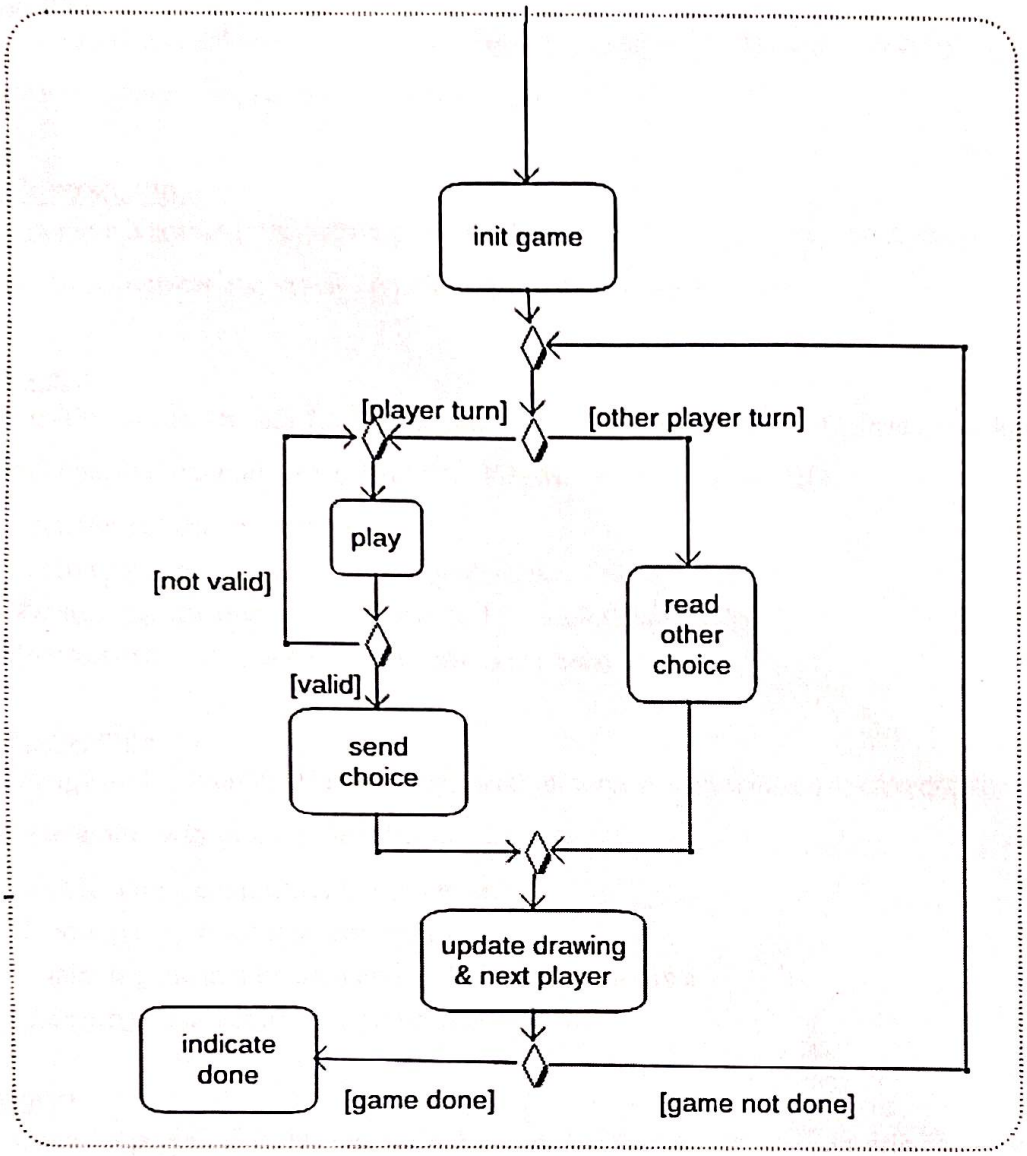




1.2. Block Diagram:



1.3. Architecture Diagram



IMPLEMENTATION

Server Setup:

Set up a dedicated server to host the game. This server will be responsible for managing game sessions, player connections, and game state.

Socket Programming:

Implement socket programming on both the server and client sides. Sockets are used for communication between the server and clients over the network.

Server Logic:

Develop the server-side logic to handle incoming connections from clients, manage player sessions, and maintain the game state. Key server functions include:

- Accepting client connections.
- Creating game sessions and pairing players.
- Managing game moves and checking for win/loss conditions.
- Broadcasting game updates to all relevant clients.

Client Application:

Design and implement the client application, which users will use to connect to the server and play the game. Key client-side functions include:

- Establishing a connection to the server.
- Handling user input to make moves.
- Displaying the game board and updating it in real-time.
- Managing chat features for player interaction.

Deployment:

Once the project is stable and fully functional, deploy the server on a reliable hosting platform that can handle multiple concurrent connections.

APPENDIX

Server.py:

```
import socket # for networking
import pickle # for sending/receiving objects

# import the game
from tic_tac_toe import TicTacToe

HOST = '127.0.0.1' # this address is the "local host"
PORT = 12783       # port to listen on for clients

# set up the server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(5)

# accept a connection from the client
client_socket, client_address = s.accept()
print(f"\nConnnected to {client_address}!")

# set up the game
player_x = TicTacToe("X")

# allow the player to suggest playing again
rematch = True

while rematch == True:
    # a header for an intense tic-tac-toe match!
    print(f"\n\n T I C - T A C - T O E ")

    # the rest is in a loop; if either player has won, it exits
    while player_x.did_win("X") == False and player_x.did_win("O") == False and
player_x.is_draw() == False:

        # draw grid, ask for coordinate
        print(f"\n       Your turn!")
        player_x.draw_grid()
        player_coord = input(f"Enter coordinate: ")
        player_x.edit_square(player_coord)

        # draw the grid again
        player_x.draw_grid()
```

```
    # pickle the symbol list and send it
    x_symbol_list = pickle.dumps(player_x.symbol_list)
    client_socket.send(x_symbol_list)

    # if the player won with the last move or it's a draw, exit the loop
    if player_x.did_win("X") == True or player_x.is_draw() == True:
        break

    # wait to receive the symbol list and update it
    print(f"\nWaiting for other player...")
    o_symbol_list = client_socket.recv(1024)
    o_symbol_list = pickle.loads(o_symbol_list)
    player_x.update_symbol_list(o_symbol_list)

# end game messages
if player_x.did_win("X") == True:
    print(f"Congrats, you won!")
elif player_x.is_draw() == True:
    print(f"It's a draw!")
else:
    print(f"Sorry, the client won.")

# ask for a rematch
host_response = input(f"\nRematch? (Y/N): ")
host_response = host_response.capitalize()
temp_host_resp = host_response
client_response = ""

# pickle response and send it to the client
host_response = pickle.dumps(host_response)
client_socket.send(host_response)

# if the host doesn't want a rematch, we're done here
if temp_host_resp == "N":
    rematch = False

# if the host does want a rematch, we ask the client for their opinion
else:
    # receive client's response
    print(f"Waiting for client response...")
    client_response = client_socket.recv(1024)
    client_response = pickle.loads(client_response)

    # if the client doesn't want a rematch, exit the loop
    if client_response == "N":
        print(f"\nThe client does not want a rematch.")
```

```
            rematch = False

        # if both the host and client want a rematch, restart the game
        else:
            player_x.restart()

spacer = input(f"\nThank you for playing!\nPress enter to quit...\n")

client_socket.close()
```

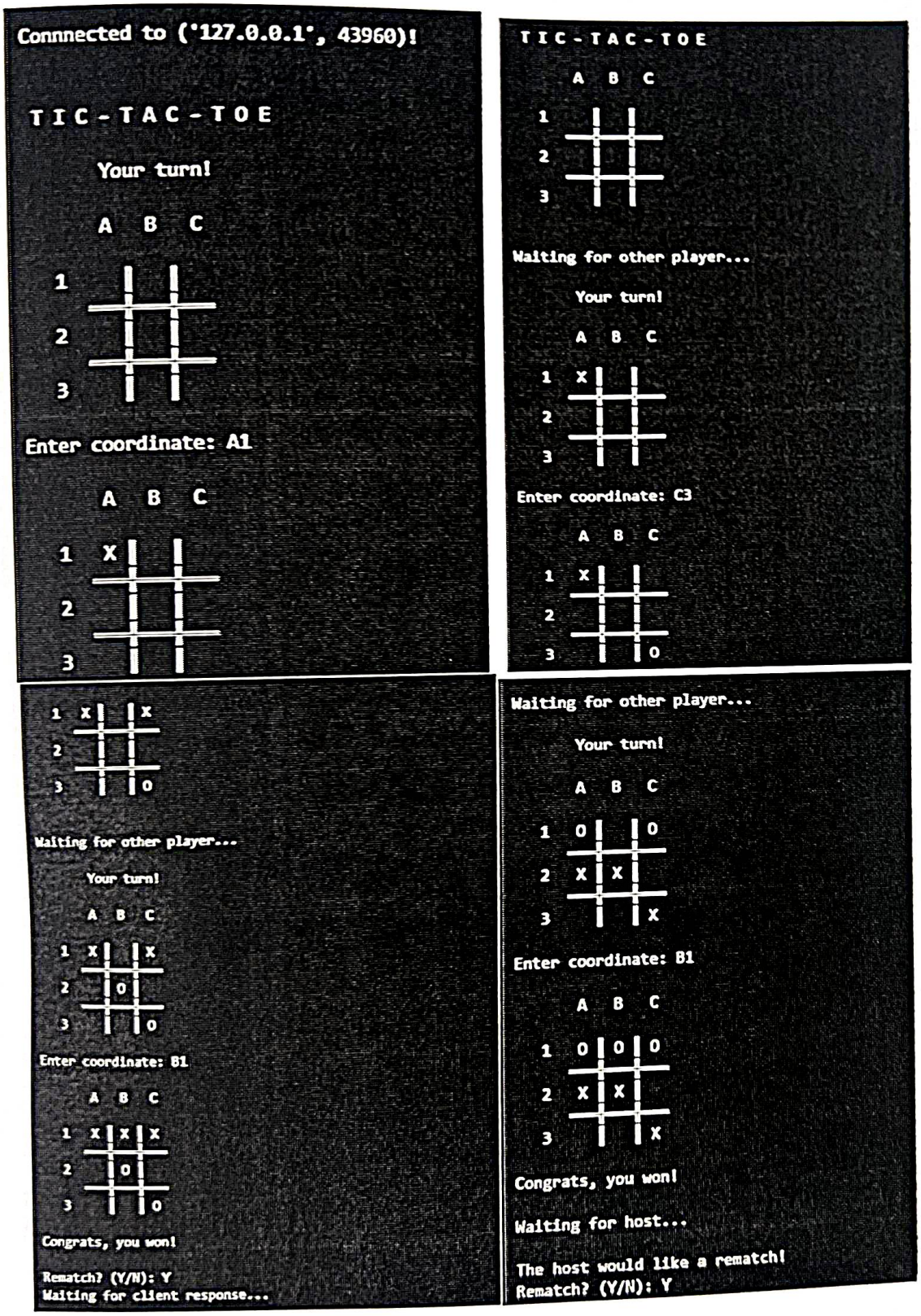
Client.py:

```
import socket # for networking
import pickle # for sending/receiving objects

# import the game
from tic_tac_toe import TicTacToe

HOST = '127.0.0.1' # the server's IP address
PORT = 12783       # the port we're connecting to

# connect to the host
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
print(f"\nConnected to {s.getsockname()}!")

# set up the game
player_o = TicTacToe("O")

# allow the player to suggest playing again
rematch = True

while rematch == True:
    # a header for an intense tic-tac-toe match!
    print(f"\n\n T I C - T A C - T O E ")

    # draw the grid
    player_o.draw_grid()

    # host goes first, client receives first
    print(f"\nWaiting for other player...")
    x_symbol_list = s.recv(1024)
    x_symbol_list = pickle.loads(x_symbol_list)
    player_o.update_symbol_list(x_symbol_list)
```

```
    # the rest is in a loop; if either player has won, it exits

    while player_o.did_win("O") == False and player_o.did_win("X") == False and
player_o.is_draw() == False:
        # draw grid, ask for coordinate
        print(f"\n       Your turn!")
        player_o.draw_grid()
        player_coord = input(f"Enter coordinate: ")
        player_o.edit_square(player_coord)

        # draw grid again
        player_o.draw_grid()

        # pickle the symbol list and send it
        o_symbol_list = pickle.dumps(player_o.symbol_list)
        s.send(o_symbol_list)

        # if the player won with the last move or it's a draw, exit the loop
        if player_o.did_win("O") == True or player_o.is_draw() == True:
            break

        # wait to receive the symbol list and update it
        print(f"\nWaiting for other player...")
        x_symbol_list = s.recv(1024)
        x_symbol_list = pickle.loads(x_symbol_list)
        player_o.update_symbol_list(x_symbol_list)

    # end game messages
    if player_o.did_win("O") == True:
        print(f"Congrats, you won!")
    elif player_o.is_draw() == True:
        print(f"It's a draw!")
    else:
        print(f"Sorry, the host won.")

    # host is being asked for a rematch, awaiting response
    print(f"\nWaiting for host...")
    host_response = s.recv(1024)
    host_response = pickle.loads(host_response)
    client_response = "N"

    # if the host wants a rematch, then the client is asked
    if host_response == "Y":
        print(f"\nThe host would like a rematch!")
        client_response = input("Rematch? (Y/N): ")
        client_response = client_response.capitalize()
```

```
        temp_client_resp = client_response

        # let the host know what the client decided
        client_response = pickle.dumps(client_response)
        s.send(client_response)

        # if the client wants a rematch, restart the game
        if temp_client_resp == "Y":
            player_o.restart()

        # if the client said no, then no rematch
        else:
            rematch = False

    # if the host said no, then no rematch
    else:
        print(f"\nThe host does not want a rematch.")
        rematch = False

spacer = input(f"\nThank you for playing!\nPress enter to quit...\n")

s.close()
```

OUTPUT


Connnected to ('127.0.0.1', 43960)!
TIC-TAC-TOE
Your turn!
A B C
1
2
3
Enter coordinate: A1
A B C
1 X
2
3
TIC-TAC-TOE
A B C
1
2
3
Waiting for other player...
Your turn!
A B C
1 X
2
3
Enter coordinate: C3
A B C
1 X
2
3 O
1 X X
2
3 O
Waiting for other player...
Your turn!
A B C
1 X X
2 O
3 O
Enter coordinate: B1
A B C
1 X X X
2 O
3 O
Congrats, you won!
Rematch? (Y/N): Y
Waiting for client response...
Waiting for other player...
Your turn!
A B C
1 O O
2 X X
3 X
Enter coordinate: B1
A B C
1 O O O
2 X X
3 X
Congrats, you won!
Waiting for host...
The host would like a rematch!
Rematch? (Y/N): Y

```
Waiting for host...

The host would like a rematch!
Rematch? (Y/N): Y


 T I C - T A C - T O E

     A  B  C
  1     |  |
     ---+--+---
  2     |  |
     ---+--+---
  3     |  |


Waiting for other player...

      Your turn!

     A  B  C
  1     |  |
     ---+--+---
  2  X  |  |
     ---+--+---
  3     |  |


Enter coordinate: C1
```

```
Waiting for other player...
Sorry, the client won.

Rematch? (Y/N): Y
Waiting for client response...


 T I C - T A C - T O E

      Your turn!

     A  B  C
  1     |  |
     ---+--+---
  2     |  |
     ---+--+---
  3     |  |


Enter coordinate: B2

     A  B  C
  1     |  |
     ---+--+---
  2     |X |
     ---+--+---
  3     |  |


Waiting for other player...
```

```
     A  B  C
  1  O  |  |
     ---+--+---
  2  X  |X |  O
     ---+--+---
  3  X  |O |  X


Enter coordinate: C1

     A  B  C
  1  O  |  |  O
     ---+--+---
  2  X  |X |  O
     ---+--+---
  3  X  |O |  X


Waiting for other player...
It's a draw!

Waiting for host...

The host does not want a rematch.

Thank you for playing!
Press enter to quit...


Process exited with code: 0
```

CONCLUSION

The "TIC-TAC-TOE Multiplayer Game" project successfully demonstrates the practical application of client-server programming principles, showcasing a fully functional, real-time multiplayer gaming experience. Through the implementation of socket communication, multi-threading, and game logic, this project not only offers an engaging platform for players to compete in the classic game but also serves as an educational resource for understanding the complexities of networked game development. With a user-friendly interface and real-time updates, it achieves the project's objectives of creating a seamless and enjoyable multiplayer game, providing a solid foundation for more advanced multiplayer game development and reinforcing the importance of network programming in the digital gaming landscape. Overall, this endeavor has demonstrated the ability to bring a classic game into the digital age, providing a valuable learning resource for those interested in network programming and serving as a testament to the dynamic and interconnected nature of modern gaming.

REFERENCES

1. The Socket Programming and Software Design for Communication Based on Client/Server (Year of Published: 2009 Published By: IEEE)
2. A design and implementation of active network socket programming (Year of Published: 2002 Published By: IEEE)
3. Design and Implementation of Client-Server Based Application Using Socket Programming in a Distributed Computing Environment (Year of Published: 2017 Published By: IEEE)
4. IP to IP Calling Through Socket Programming (Year of Published: 2021 Published By: IEEE)
5. Analysis of Socket Communication Technology Algorithms Under TCP/IP Protocol (Year of Published: 2019 Published By: IEEE)