**Statement of Work (SOW)**

**Project Title:**

Tic-Tac-Toe Multiplayer Game with Client-Server Architecture

**Project Overview:**

The goal of this project is to create a network-based multiplayer Tic-Tac-Toe game. The system will have a client-server architecture, in which player turns are managed by a central server that also broadcasts the game data to clients. The game will be played in turns by two players who will connect to the server as clients. Getting three marks (X or O) in a row, column, or diagonal is the aim of the game.

**Objectives:**

1. Design a two-player, basic, playable game of Tic Tac Toe.
2. Use Python's socket package to implement client-server communication.
3. Verify that the game is turn-based and that the server is in charge of enforcing rules and turns.
4. Inform both players in real time about the status of the game.
5. Allow game replay option with the option to reset the board.

**Scope of Work:**

**1. Functional Requirements:**

* **Server**:

o Control the current state of the game (3x3 board).   
o Adhere to the game's regulations (acceptable turns, moves, and conditions for wins and draws).   
o After every move, broadcast to both clients the modified game state.   
o Inform clients of the outcome (victory, defeat, or draw).   
· Control player relationships (a maximum of two participants each game).   
o Give players the chance to restart the game after it's over.

* **Clients**:

o Establish a server connection.   
o Present the active game board.   
o Let players enter their movements (column and row).   
o Send the server the move data and watch for a response.   
o Based on the server's answer, update the game board.   
o After the game, display the outcome (win, lose, or draw).

**2. Non-Functional Requirements:**

* **Performance**:
* The game must refresh the board in real time with the least amount of latency.
* Verify that the game can effectively manage simple client-server interactions.
* **Usability**:
  + Easy to use interface for both the server and clients.
  + Simple to Understand prompts for player input and game status updates.
* **Scalability**:
  + Initial version will support one game session with two players per server.
  + Future scope: Ability to extend the server to handle multiple games with different pairs of clients.
* **Reliability**:
  + Handle connection drops gracefully (e.g., if a player disconnects mid-game).
  + Server should prevent clients from making moves out of turn.

**Deliverables:**

1. **Server Application**:
   * Python script that configures a server to oversee a game of Tic Tac Toe between two clients.
   * Capacity to broadcast updates, validate moves, manage game state, and handle connections.
2. **Client Application**:
   * Two players can connect to the server, view the game board, and enter their movements with this Python script.
3. **Game Logic**:

* Complete application of the Tic Tac Toe rules, including detection of draws and victories.

1. **Documentation**:

* Setup and installation guidelines.
* Code documentation that explains the game logic, client, and server.
* A user manual outlining the rules for the game.

1. **Testing**:
   * A series of test cases to confirm that the game works as intended, covering things like player turns, invalid inputs, win/draw scenarios, and reconnections.

**Project Phases & Timeline:**

**Phase 1:**

* + Put Tic-Tac-Toe's basic game logic (moves, game board, and win/draw detection) into practice.
  + Game logic and project setup (one week)
  + Create a rudimentary game command-line interface.

**Phase 2:**

* Create the server to manage game state and player connections.
* Two weeks of Clients-Server Architecture
* Use sockets (TCP/IP) to implement client-server communication.
* Check the fundamental communication between two clients and the server.

**Phase 3:**

* + Uphold player turns, legal moves, and server-side game regulations.
  + Turn management and game rules (one week)
  + After every move, broadcast to both clients the modified game state.
  + Inform clients of the outcome of the match—win, lose, or draw.

**Phase 4:**

* + **Test various situations, including erroneous inputs, lost connections, and game resets.**
  + **One week of testing and debugging**
  + **Enhance server responsiveness and communication latency.**

**Phase 5: Finalization & Documentation (1 Week)**

* Finalize the codebase.
* Prepare documentation and user instructions.
* Deliver the final project and test results.

**Technical Stack:**

* **Programming Language**: Python 3.12
* **Libraries**:
  + socket for networking.
  + Optionally, threading or asyncio for handling multiple clients (in future scalability).
* **Development Environment**:
  + Local Python setup or any cloud-based environment (Google Colab).

**Assumptions:**

* Both clients are expected to be on the same local network or have access to the same server.
* The server will handle only one game session with two players at a time.

**Project Milestones:**

1. Core game logic developed.
2. Server and client communication established.
3. Turn-based logic and game rules enforcement implemented.
4. Full game functionality with real-time updates between clients.
5. Documentation and testing complete.

**Success Criteria:**

* No significant issues or crashes during gameplay; two players may join, play, and finish a game of Tic-Tac-Toe with successful client-server connection.
* An intuitive UI that is simple to use for both players.
* The outcomes of the game are precise and appropriately conveyed to both players.