

# **Tic Tac Toe Web Game**

Front-end: HTML, CSS, JavaScript

Back-end: C, Python (Flask)

Shahd El-Refai

3 September 2023

## Project Overview

The Tic Tac Toe web game project consists of both front-end components using HTML, CSS, and JavaScript and back-end components using C and Python (Flask).

### Front-End Components:

1. HTML: Provides the structure of the web page and defines the game board and user interface elements.
2. CSS: Styles the web page, making it visually appealing and user-friendly.
3. JavaScript: Manages the game's functionality on the client-side, including user input and rendering game updates.

### Back-End Components:

The back-end of the game is responsible for handling web requests and managing the game's logic. It is divided into two parts:

1. Python (Flask):
  - Purpose: Acts as the intermediary between the front-end and the game's logic.
  - Functionality: Handles incoming web requests from the client (JavaScript) and forwards game inputs to the C server.
  - Framework: Utilizes the Flask framework to streamline web request handling.
2. C (Server):
  - Purpose: Manages the core game logic and maintains the game state.
  - Functionality: Receives game inputs from the Python server, processes them to update the game, and sends game updates back to the Python server.
  - Communication: Establishes and maintains a TCP connection (Socket) with the Python server, enabling continuous communication throughout the game.

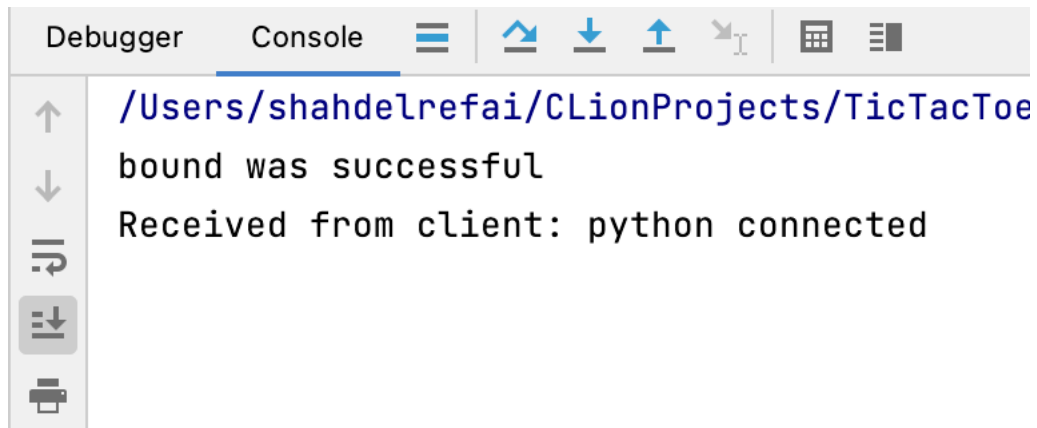
**Communication Flow:**

- The front-end (HTML, CSS, JavaScript) handles user interactions and sends game-related requests to the Python server.
- The Python server (Flask) receives these requests, and forwards them to the C server via the established TCP connection.
- The C server processes the game inputs, updates the game state, and sends game updates back to the Python server.
- The Python server relays these updates to the front-end, ensuring that players see the current state of the game.

This architecture allows for a seamless flow of game data and interactions between the front-end, Python server, and C server, resulting in a functioning Tic Tac Toe web game.

## Server-Client Connection

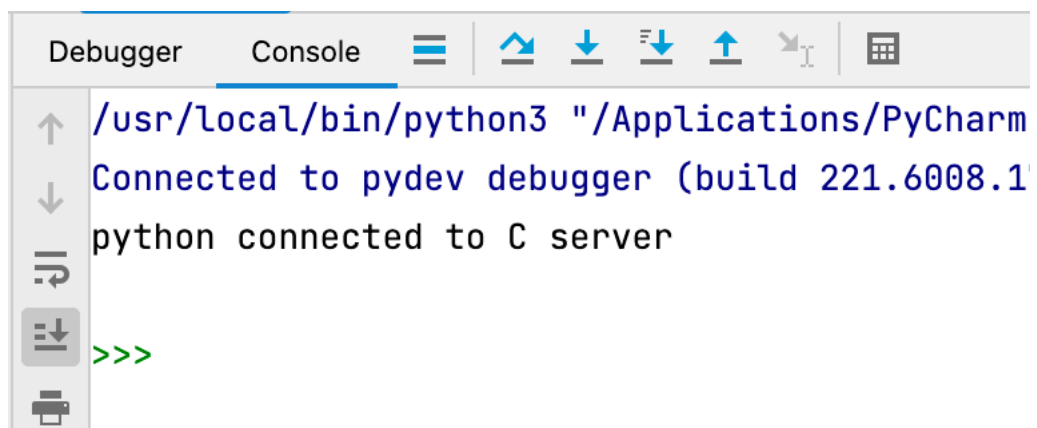
C Server:

A screenshot of a software development environment's console window. The window has a title bar with 'Debugger' and 'Console' tabs, and a toolbar with various icons. The console output shows the following text: 

```
/Users/shahdelrefai/CLionProjects/TicTacToe  
bound was successful  
Received from client: python connected
```

```
↑  
↓  
↶  
↷  
⌵  
⌶
```

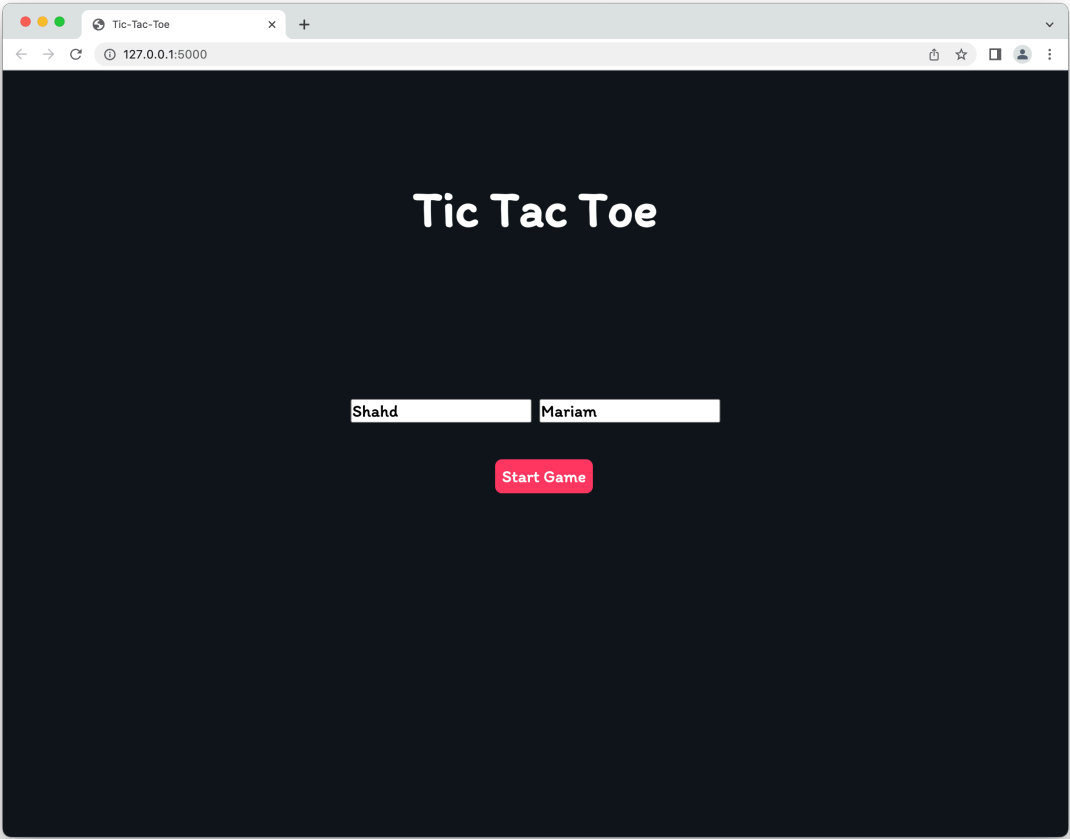
Python Client:

A screenshot of a software development environment's console window. The window has a title bar with 'Debugger' and 'Console' tabs, and a toolbar with various icons. The console output shows the following text: 

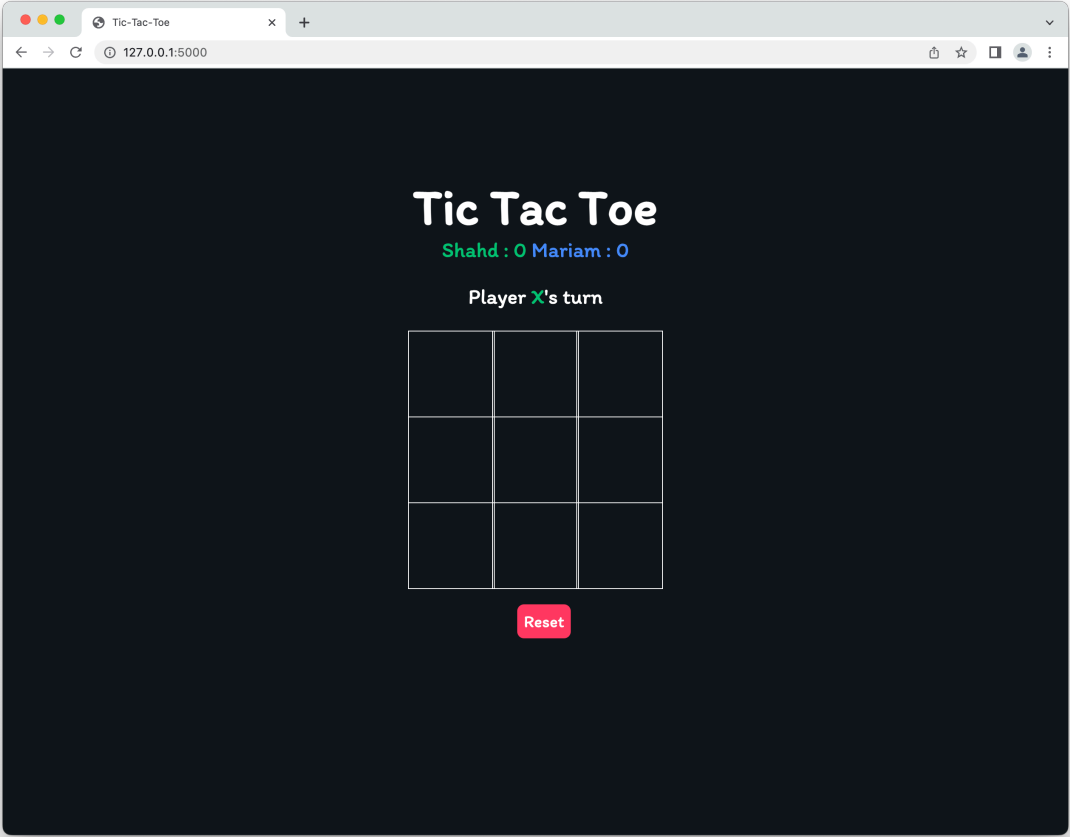
```
/usr/local/bin/python3 "/Applications/PyCharm  
Connected to pydev debugger (build 221.6008.1  
python connected to C server  
  
>>>
```

```
↑  
↓  
↶  
↷  
⌵  
⌶
```

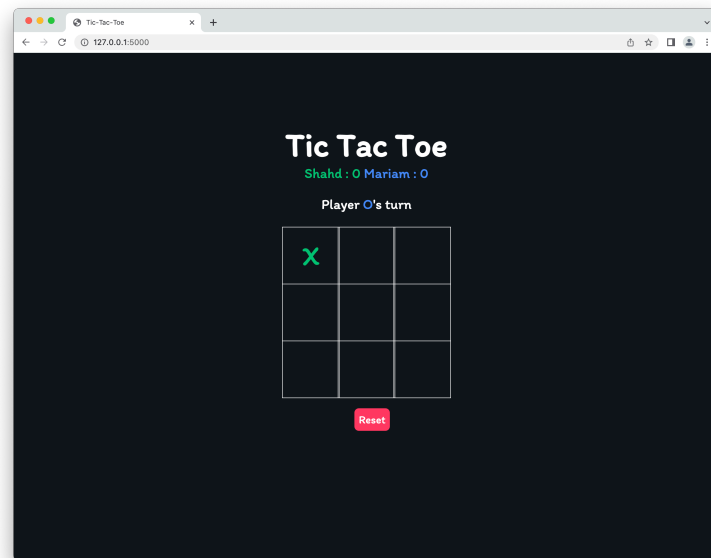
Input Player Names Page



Starting Game Page



## Playing a Valid Move



C Server:

Receiving index 0 which was a valid move, therefore responded to client with “vc1” where v corresponds to valid, allowing the move to be completed on the front-end.

```

New Round
Move received from client : 0
Response : vc1

```

Trying to play move of index 0 again:

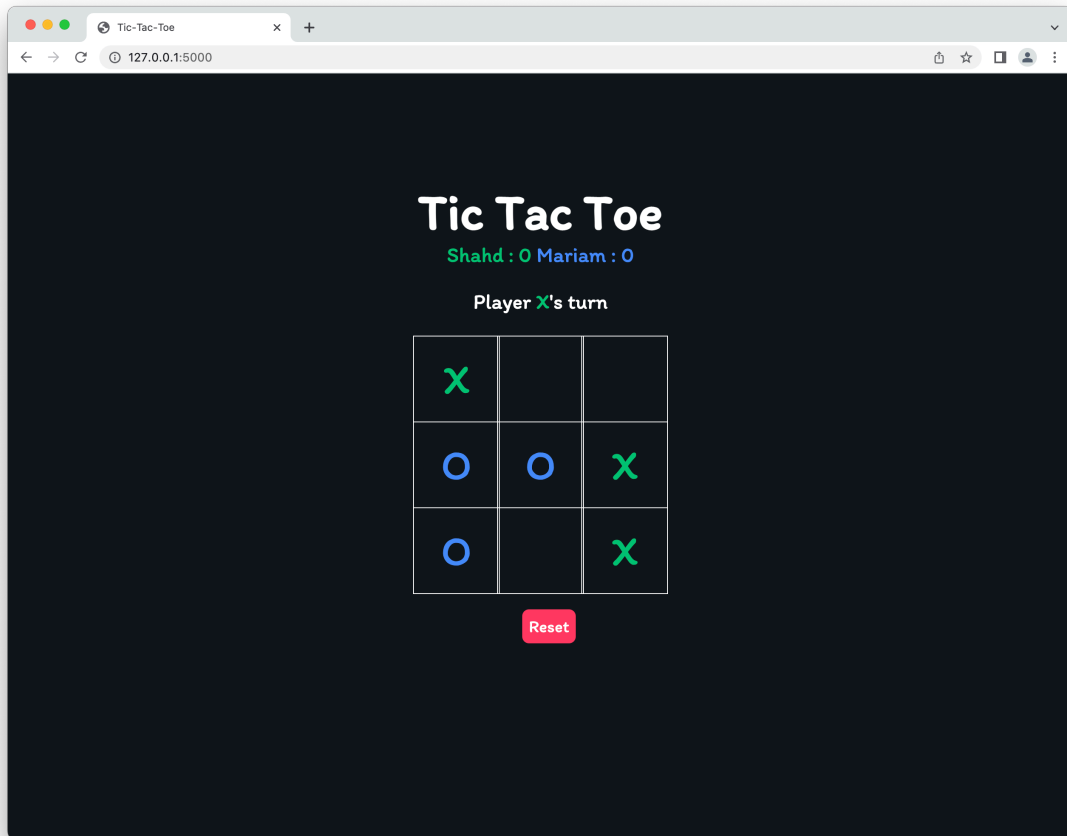
```

New Round
Move received from client : 0
Response : vc1
Move received from client : 0
Response : nnn

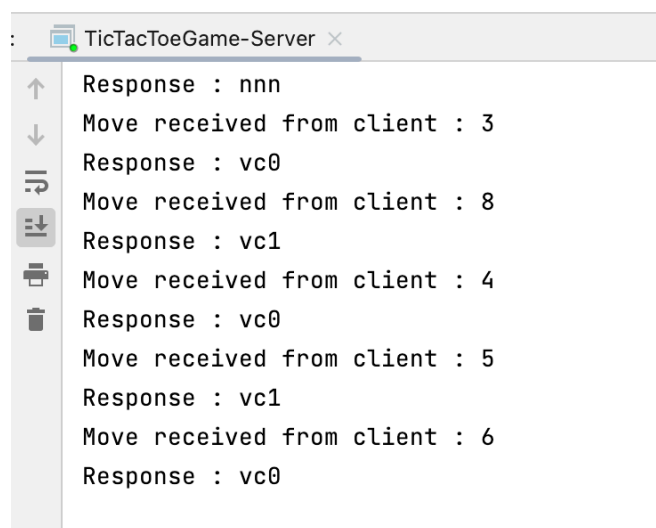
```

Responding with “nnn” corresponding with “not a valid move”, preventing the move to be completed on the front-end.

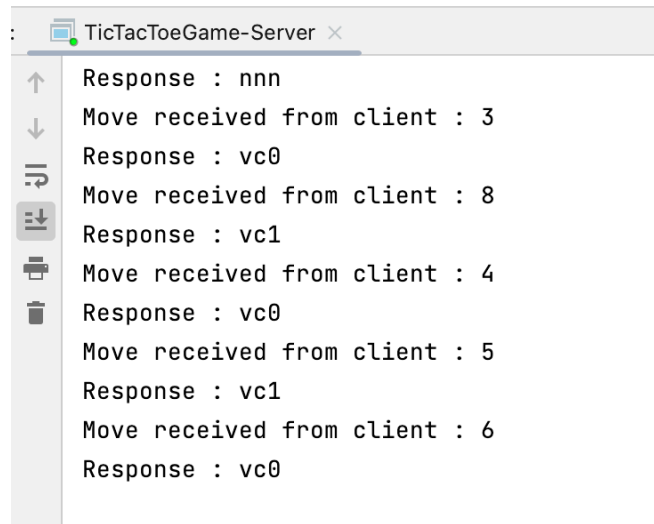
## Checking for a Win or Tie



C server checks for each move for a win or tie, here there was neither, therefore responding with “vc0” or “vc1”, where c corresponds to “continue game”.



## Changing Players' Turns

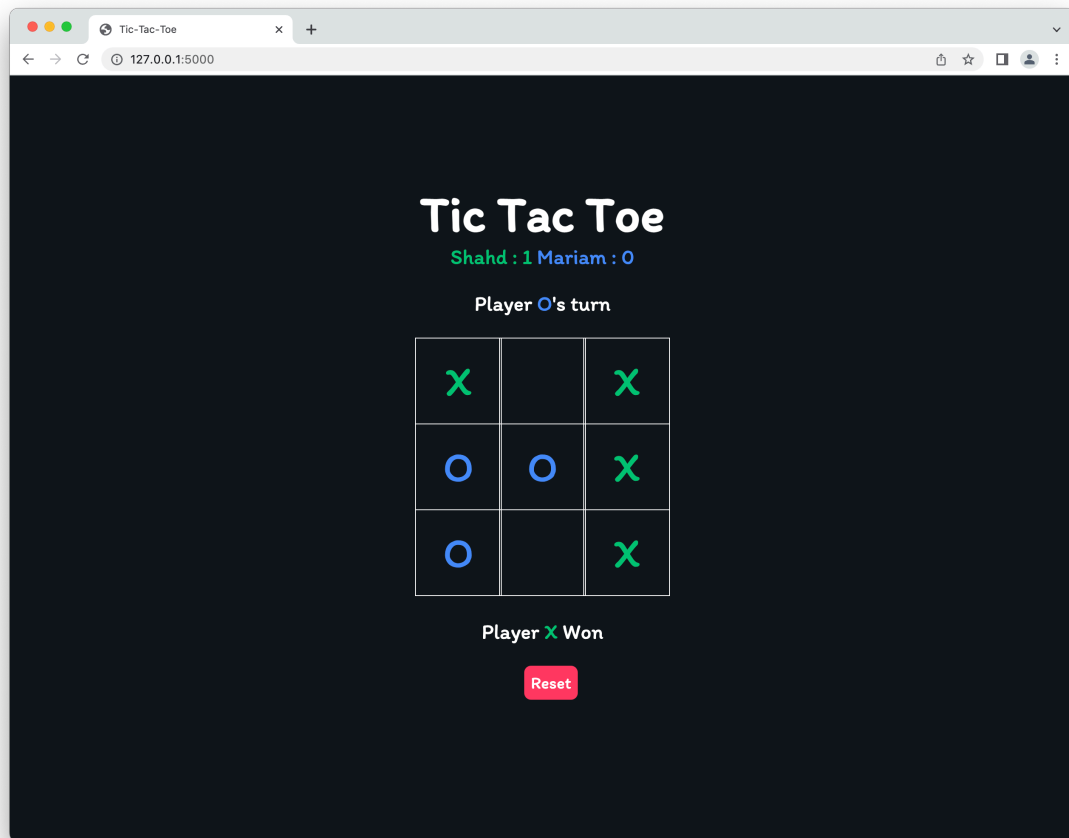


```
TicTacToeGame-Server x
↑ Response : nnn
↓ Move received from client : 3
! Response : vc0
? Move received from client : 8
→ Response : vc1
← Move received from client : 4
⇄ Response : vc0
⇅ Move received from client : 5
⇆ Response : vc1
⇇ Move received from client : 6
⇈ Response : vc0
```

For each valid move, if the next turn is for player 1, the server responds with “vc**0**”, where 0 corresponds to player 1, similarly for player 2, the server responds with “vc**1**”, where 1 corresponds to player 2.



## Happens a Win

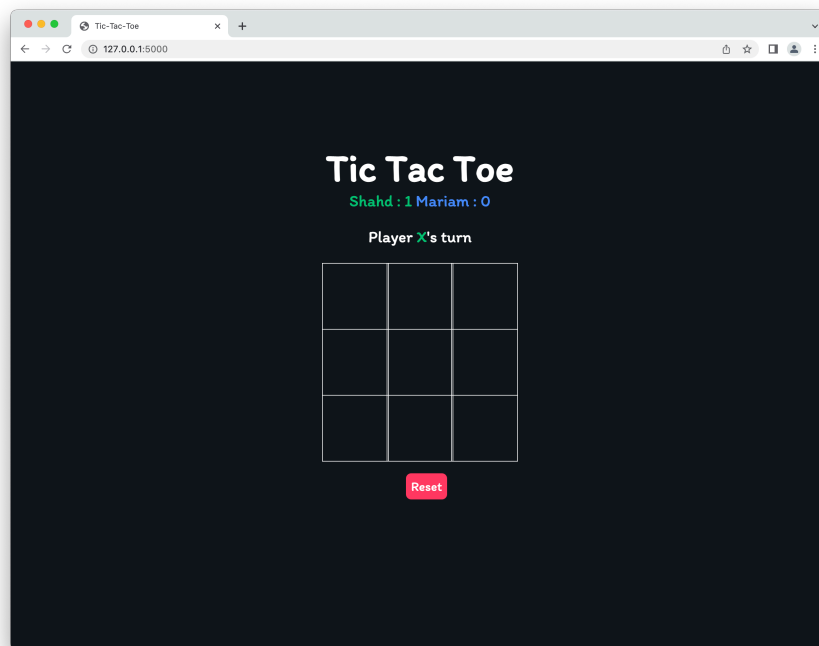


Checking for a win and it is indeed a win, the server responds with “v01”, where 0 corresponds to “player 1 won”, allowing the front-end to stop the round, announces a win for player 1 and updates the score.

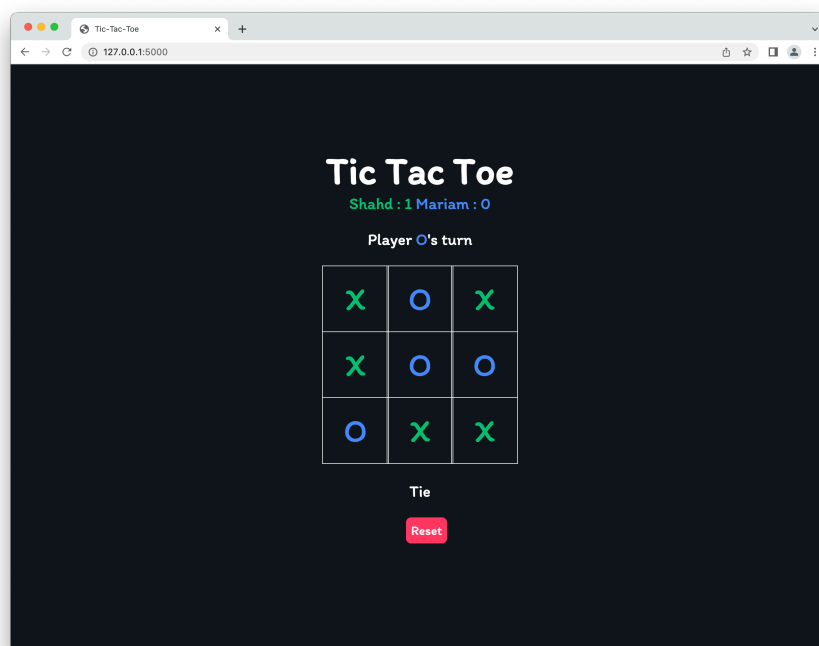
```
Move received from client : 2
Response : v01
New Round
```

The server automatically starts a new round.

## Using the Reset Button



## Happens a Tie



It is indeed a tie, the server responds with “vt1”, where t corresponds to tie.

```
Move received from client : 7
Response : vt1
New Round
```

## **User Manual**

- Run the C script to create C server.
- Run Python script that will connect to C server and run web application.
- Visit the link that is serving the web application.
- Two players enter their names and press Start Game.
- Play Tic Tac Toe!