Web Security

Prof. Yinzhi Cao

Fall 2025

Homework 1: JavaScript Game: Five-in-a-Row

Project due on September 16th at Noon in Canvas

In this homework, we will explore a web application that allows two players to participate in a famous game, Five-in-a-Row. Here are the game descriptions and rules copied from Wikipedia:

"**Five in a Row** … is traditionally played with GO pieces (black and white stones) on a go board with 19x19 intersections"

"Black plays first if white did not just win, and players alternate in placing a stone of their color on an empty intersection. The winner is the first player to get an unbroken row of five stones horizontally, vertically, or diagonally."

To design the game, we are going to start from a three-stage procedure:

(1) a JavaScript client,
(2) a server application coordinating two JavaScript clients, and
(3) [Bonus] a server-side AI program that can play against a single player.

JavaScript Client [50 points]: Please first design and implement a JavaScript client embedded inside an HTML5 webpage that allows two players to play five-in-a-row locally. You are free to use any HTML5 enabled features, such as Canvas, localStorage and sessionStorage.

Note that your application needs to be compatible with all the major web browsers, such as Chrome, Microsoft Edge, Safari, and Firefox. In addition, we should not expect any errors to show up in the JavaScript console.

Server Application [50 points]: Please design and implement a server application in Node.js. The server application needs to authenticate the JavaScript client, store all the states, return the updates from the other player, and judge which player wins in the end.

Note that the server application needs to verify whether a client-side move is legitimate, because a malicious client may, for example, pace a stone on an occupied intersection. Such verification needs to be done at both the client side through JavaScript and the server side. The reason is that a malicious client may modify the client-side JavaScript to bypass the check.

In order to enable the client and server communication, we also need a client-side receiver that accepts packages from the server, and then guides the JavaScript application written in our first step. Such client-side receiver may utilize AJAX or WebSocket for the communication.

When this step is finished, we expect that two players can open browser windows, connect to the web server, and play together. The first-joined player will start first, and then the server is able to notify both players whether he/she wins or loses.

For demonstration purpose, the server may be deployed at the same host (i.e., localhost) as the client-side JavaScript application.

[Bonus] An AI program [10 points]: Please design an AI program at the server side that can play with a single player. The AI program will be communicating with the server application the same as the JavaScript client, and instructing the server application to place stones. The human player will go first, and the AI program will go after the human player.

Note that your AI program needs to at least beat a player that randomly places stones on the board. In particular the AI program needs to have the following abilities:

1. Block the human player and prevent him or her from winning. For example, when the AI program sees the following pattern:

   XXX

   It may put a stone adjacent to the pattern to prevent a four-in-a-row, like the following:

   *XXX
2. Try to form a five-in-a-row, when blocking the human player.

The AI program can be written in any language that you prefer. However, preferably, it will be written in the same language as your server-side application.

Notes:

- For all three tasks, submit your code **through Canvas**.
- If you like, you may compete your AI program with others, and see which one is more intelligent.
- If you are interested, you may migrate your application to a mobile platform through WebView.