**Tic-Tac-Toe Data-Model Sketch**

var player = {

playerLetter: ‘’,

playerName: ‘’,

isTurn: false

}

var gameState = {

gameOver: false,

winnerName: '',

gameName: '',

player1: player,

player2: player,

gameMoves: [ ]

}

**Tic-Tac-Toe AI Pseudo-Code Specification**

1. Win: If the AI has two in a row, it will place a third to get three in a row.
2. Block: If the [opponent] has two in a row, the AI will play the third to block the opponent.
3. Fork: Creation of an opportunity where the AI has two threats to win (two non-blocked lines of 2).
4. Blocking an opponent's fork:
   1. The AI will create two in a row to force the opponent into defending, as long as it doesn't result in them creating a fork. For example, if "X" has a corner, "O" has the center, and "X" has the opposite corner as well, "O" must not play a corner in order to win. (Playing a corner in this scenario creates a fork for "X" to win.)
   2. If there is a configuration where the opponent can fork, the player should block that fork.
5. Center: AI marks the center. (If it is the first move of the game, playing on a corner gives "O" more opportunities to make a mistake and may therefore be the better choice; however, it makes no difference between perfect players.)
6. Opposite corner: If the opponent is in the corner, the AI plays the opposite corner.
7. Empty corner: The AI plays in a corner square.
8. Empty side: The AI plays in a middle square on any of the 4 sides.

**Improving this Test**

The structure of this test does not need much improving at all as it is very involved and drives developers to work on both backend and frontend code. Also, due to the restrictions (most notably prohibiting file system and database store use) it forces developers to use creative yet functional alternatives.

However, scoring based on a linear time is probably not a very good way of evaluating a developer. As in my case, it is very difficult to commit and guarantee a time to focus without being disturbed. Late at nights is fine but working on an exercise like this in the night was not the best idea since at that time I’m near burnt out for the day and not at my best.

Allowing candidates to break up the exercise into byte size times over a day or two day span would probably yield better, more accurate results. An exercise such as this definitely only needs a few hours to complete but breaking up that few hours over a period of a day would work better for developers like me who are almost always on the go.