|  |  |  |  |
| --- | --- | --- | --- |
| ID | REQUIREMENT | DESIGN ELEMENT | F/NF |
| REQ001 | When pressed, the reload button (UI-1) shall start a new game | Http server |  |
| REQ002 | At the start of the game, the first player shall display “Waiting for another player” in UI –3, Message Line | Game() |  |
| REQ003 | When the second player joins the game (by loading the html page), UI-3 will display “You are O. Your turn” for player 1 and “You are X. Other players turn.” for player 2. Player 1 has the first turn. | Game.startGame |  |
| REQ004 | A player will request “ownership” of a square by pressing the button. (UI-2) | JS buttonclick() |  |
| REQ005 | Invalid moves are ignored. | Game.update() |  |
| REQ006 | After a valid move, UI-4, will display the ownership of the square by displaying an “X” or a “O” | Game.update() |  |
| REQ007 | After a valid move, UI-3 will display “Other Players Move” and “Your move” (depending on which player you are. | Game.update() |  |
| REQ008 | The system shall check for a winner after every move. If one player is a winner, UI-3 will display “You Win” or “You Lose” | Game.CheckBoard() |  |
| REQ009 | After a winner is declared, no button presses are accepted. | Game.update() |  |
| REQ010 | UI-4 will display ‘?’ when the page is loaded. When the WebSocket connection happens, it will change the ‘?’ to ‘ ‘. Note: this may happen very quickly | Html initialization  JS on\_message |  |
| REQ011 | If the WebSocket connection is terminated unexpectedly, UI-3 will display “Server Offline” | JS on\_close() |  |
| REQ012 | Two different messages, one for each player. UI3 | Game and html |  |
| REQ013 | The server will support many simultaneous games | App.on\_open() |  |
| REQ014 | In the event of a draw, both UI3 will display “Draw” | Game.is\_draw() |  |
| REQ015 |  |  |  |
| REQ016 |  |  |  |
| REQ017 |  |  |  |
| REQ018 |  |  |  |
| REQ019 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

From the gameserver to the webpage

* “You are player #”
* “State of the game board”
  + Includes button state and messages

From the webpage to the gameserver

* “Button Pressed (specific)”

Notional json encoded string. (Note: Expect slight changes when it is turned into source code)

Design

public class Game

public int Players;

public int CurrentTurn;

public int[] Button;

public String[] Msg;

public int GameId;

Game()

StartGame()

CheckBoard(int player)

Update(UserEvent U)

Tick()

public App

Vector<Game> ActiveGames = new Vector<Game>();

int GameId = 1;

OnOpen()

OnMessage

Main()

Http server

WebSocketServer

Index.html

(index.js)

var idx

var gameid

connection.onopen = function (evt)

connection.onclose = function (evt)

connection.onmessage = function (evt)

buttonclick(i)