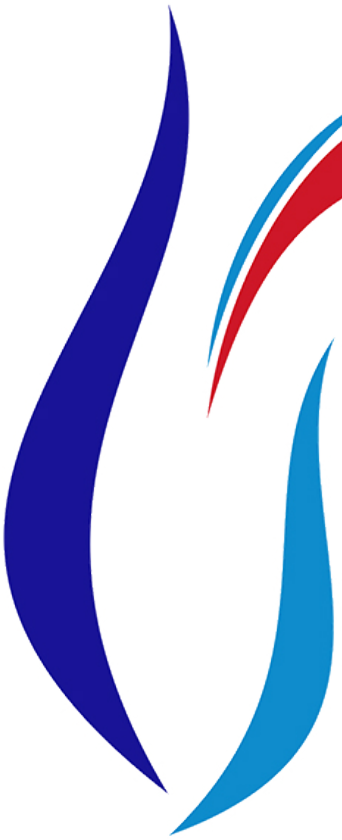


MANİSA
CELAL BAYAR
ÜNİVERSİTESİ

HASAN FERDI TURGUTLU
TEKNOLOJI FAKÜLTESI

HASAN FERDİ TURGUTLU TEKNOLOJİ FAKÜLTESİ

YAZILIM SINAMA ÖDEV #1

YUSUF DEDE 152802053
FAHRETTİN AKSU 162804013

TİC TAC TOE OYUNU | DOKÜMANTASYONU

1.Amaç

Basit bir xox oyunu olan Tic Tac Toe projesini 3 modu olan bir oyun olarak tasarladık. Bunlar “Aynı Bilgisayar Üzerinde”, “Bilgisayara Karşı” ve “Ağ Üzerinden Oyun” modlarıdır. Oyun, web teknolojileri kullanılarak geliştirildi. Asıl amaç oyunu kazanmaktır :)

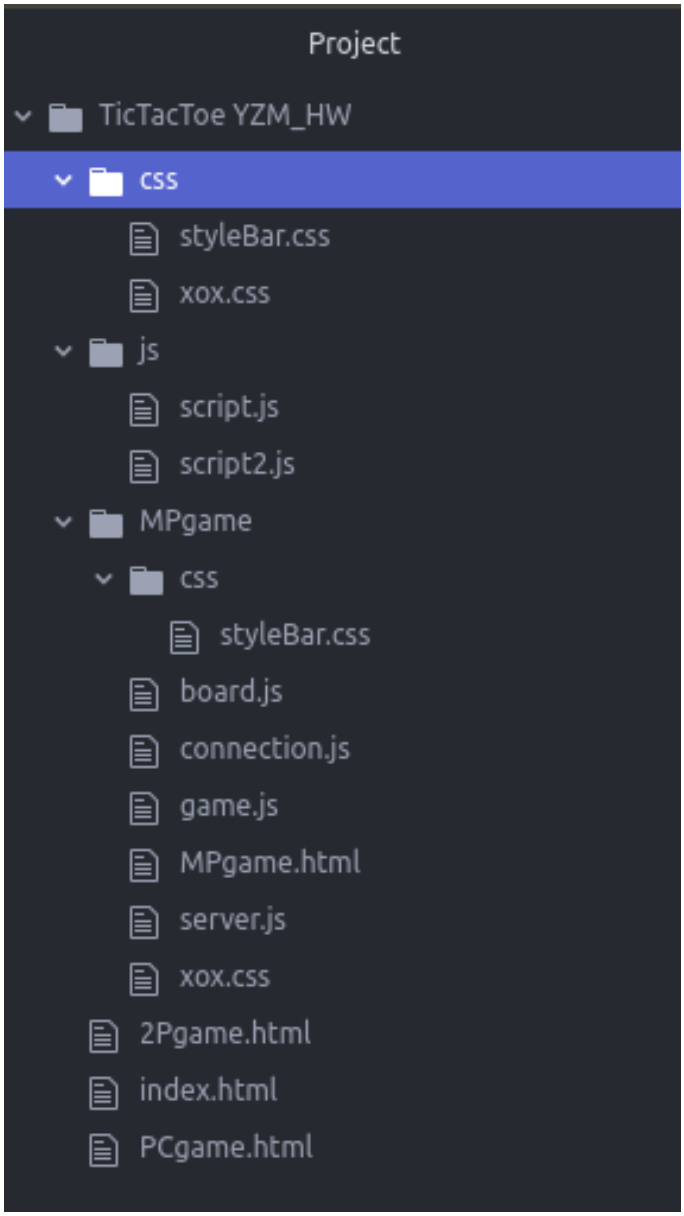
2.Girdiler

Oyun web üzerinden mouse click olayı ile çalışmaktadır. Oyun içerisindeki tabloda bulunan karelere click olayı ile oyunda girdilerin aktarılması sağlanmaktadır.

3.Mantığı

Oyun içindeki “X” ve “O” sembollerini çapraz veya düz olmak üzere üç tane ardı ardına getirmektir. Oyundaki kural ise beraberlik olması durumunda süre avantajı ile birinin galip gelmesi.

4.Program Kodu



Yazmış olduğumuz oyun web üzerinde çalışmaktadır. Programın dosya bağımlıklıkları yandaki şekilde gösterilmektedir.

Farklı olan bir nokta ise “Ağ Üzerinde Oyun(MPgame)” modunun ayrı olarak oyuna implemente edilmekte olduğudur. Çünkü kodlar belli kısımları “GitHub” üzerinden çekilmiştir.

Yazılmış olan oyunun kaynakları

https://codepen.io/Rudchyk/pen/qNOEGj?editors=0010#0
https://codepen.io/ElaMoscicka/pen/aEpBwV
https://github.com/Darhazer/nodejs-tic-tac-toe

script.js | “Bilgisayar Üzerinden Oyun”

```
var time=0;
var TimerTurn=true;
var TimerBarStop=true;
var origBoard; // her karede ne olduğunu izleyen bir dizi: X, O veya hiçbir şey
const OPlayer = 'O';
const aiPlayer = 'X';

const winCombos = [  //dizi kazanan kombinasyonları gösterecek
    [0, 1, 2],
    [3, 4, 5],
    [6, 7, 8],
    [0, 3, 6],
    [1, 4, 7],
    [2, 5, 8],
    [0, 4, 8],
    [6, 4, 2]
];

var turn_count = 0;
const cells = document.querySelectorAll('.cell'); // hücre değişkeni, bir 'cell' sınıfına sahip olan her
                                                  //öğeye bir başvuru depolayacak.

startGame(); // oyuna başlamak için arama fonksiyonu



//Oyunu başlatmak için işlevi tanımlamak ("Replay" düğmesine tıkladığınızda da çalışır)
function startGame() {
  TimerBarStop=true;
  progress(5,10, $('#progressBar'));
  turn_count=0;
  document.querySelector(".endgame").style.display = "none";
  origBoard = Array.from(Array(9).keys()) //diziyi 0-9 arası her sayıya dönüştür
  for (var i=0; i< cells.length; i++) {
  cells[i].innerText = ''; //hücrede hiçbir şey olmayacak
  cells[i].style.removeProperty('background-color'); // arka plan rengini kaldırarak
  cells[i].addEventListener('click',turnClick, false); //turnClick işlevini çağırmak

    }
}



// turnClick işlevini tanımlama
function turnClick (square) {
  console.log(TimerTurn);
    if (typeof origBoard[square.target.id] === 'number') { //sadece tıklanan id bir sayıysa, bu kimse
bu noktada oynamadığı anlamına gelir
        turn(square.target.id, OPlayer);
        if(!checkTie()) turn(bestSpot(), aiPlayer); //bir dönüş alarak insan oyuncu

      }
}
```

script.js | “Bilgisayar Üzerinden Oyun” devam...

```
// dönüş fonksiyonunu tanımlama
function turn(squareId, player) {
    origBoard[squareId] = player;
    document.getElementById(squareId).innerText = player;  // ekranın güncellenmesiyle
oynatıcının nereye tıklandığını görebiliriz
    let gameWon = checkWin(origBoard, player)
    if (gameWon) gameOver(gameWon)  // Sıra alındığında oyunun kazanılıp kazanılmadığını
                                                // kontrol edeceğiz

}


// checkWin işlevini tanımlama
function checkWin(board, player) {
  let plays = board.reduce((a, e, i) =>
  (e === player) ? a.concat(i) : a, []); //Oynatıcının oynadığı her dizini bulmak
  let gameWon = null;
  for (let [index, win] of winCombos.entries()) { //Oyunun her winCombos üzerinden döngü
                                                          //yaparak kazanılıp kazanılmadığını kontrol etme
    if (win.every(elem => plays.indexOf(elem) > -1)) { //Bu kazananın kazananı olarak sayılan her
                                                                  //yerde oyuncu oynandı
      gameWon = {index: index, player: player};  //hangi oyuncuyu kazanırsa kazanır ve hangi
                                                          // oyuncu kazanırdı
      break;
    }
}
return gameWon;
}


// oyun tanımlama işlevi
function gameOver(gameWon) {
  for (let index of winCombos[gameWon.index]) { // WinCombos'un her dizininden geçiyor
    document.getElementById(index).style.backgroundColor = gameWon.player ===
     OPlayer ? "#4da6ff" : "#ff0000";  // AI arka plan rengini kırmızıya kazandıysa, arka plan
                                                 // rengini kırmızıya kazandıysa
}
  for (var i= 0; i < cells.length; i++ ) { // artık hücreleri tıklayamadığımızdan emin olmak
    cells[i].removeEventListener('click', turnClick,false);

}
          //İnsan Oyuncu kazanırsa "Sen kazandın!" gösterse, aksi takdirde "Kayboldun".
  declareWinner(gameWon.player === OPlayer ? "Sen kazandın!" : "Bilgisayar Kazandı!");
  TimerBarStop=false;
}
```

script.js | “Bilgisayar Üzerinden Oyun” devam...

```
// declareWinner işlevini tanımlama
function declareWinner(who) {
    document.querySelector(".endgame").style.display = "block";
    document.querySelector(".endgame .text").innerText = who;
}

// emptySuares işlevinin tanımlanması
function emptySquares() {

    return origBoard.filter(s => typeof s === 'number'); // eleman türünün sayıya eşit olup
                                                         // olmadığını görmek için origBoard'daki
                                                         // tüm öğeleri filtreleyin.Eğer evet ise, onu
                                                         // iade edeceğiz (sayılar boş olan tüm
                                                         // kareler, X ve O ile kareler boş değil)
}
// bestSpot işlevini tanımlama
function bestSpot() {
    return emptySquares()[0];
}
// checkTie işlevinin tanımlanması
function checkTie() {
  if (emptySquares().length === 0) { // her kare doluysa ve kimse kazanmadıysa o zaman bir kravat
    for (var i = 0; i < cells.length; i++) {
      cells[i].style.backgroundColor = "#66ff66"; // arka plan rengini yeşil olarak ayarlama
      cells[i].removeEventListener('click', turnClick, false); // makindeclareWinner
                                                               // (gameWon.player === Oplayer? "Sen
                                                               // kazandın!": "Kaybettin.")
                                                               // emin kullanıcı oyun bitti her yerde
                                                               // tıklayamaz


      }

    if(time>6){
        declareWinner("Süre Farkı O kazandı!");
    }else{
        declareWinner("Süre Farkı AI kazandı!");
    }
    TimerBarStop=false;

    return true; // bir kravat olarak doğru dönüyor
}
return false;
}
```

script.js | “Bilgisayar Üzerinden Oyun” devam...

```
function progress(timeleft, timetotal, $element) {
    var progressBarWidth = timeleft * $element.width() / timetotal;
    $element.find('div').animate({ width: progressBarWidth }, 600).html(Math.floor());
    if(timeleft > 0 && timeleft<10) {
        setTimeout(function() {
            if(TimerTurn==false){
                if(TimerBarStop){progress(timeleft +1, timetotal, $element);}

            }
            else{
                if(TimerBarStop){progress(timeleft -1, timetotal, $element);}

            }


        }, 1000);

    }

    if(timeleft==0){
        for (var i= 0; i < cells.length; i++ ) { // artık hücreleri tıklayamadığımızdan emin olmak
            cells[i].removeEventListener('click', turnClick,false);

            }
        declareWinner("X Kazandı!");
    }
    if(timeleft==10){
        for (var i= 0; i < cells.length; i++ ) { // artık hücreleri tıklayamadığımızdan emin olmak
            cells[i].removeEventListener('click', turnClick,false);
            }
        declareWinner("O Kazandı!");
    }
    time=timeleft;
};
```

script2.js | “2 Kişilk Oyun”

```
//defining turnClick function
function turnClick (square) {
  console.log(TimerTurn);
    if (typeof origBoard[square.target.id] === 'number') { //if id that was just clicked is a number,
that means no one played in this spot
      if (turn_count % 2 == 0)
      {
        turn(square.target.id, OPlayer); //human player taking a turn
        TimerTurn=false;
        checkTie();
      }
      else {
        turn(square.target.id, XPlayer); //human player taking a turn
        TimerTurn=true;
        checkTie();
      }
      turn_count++;
    }
}
```

script2.js üzerindeki kodlar script.js teki kodlarla aynı sadece değişiklik olarak yukarıdaki bölüm eklenmiştir. Burada iki oyuncu arasındaki sıranın aktarılması fonksiyonları yer almaktadır.

board.js | “Ağ Üzerinden Oyun”

```
Board = function(table){
        var self = this;
        this.enabled = false; // the game object controls when the board is enabled
        this.html_cells = []; // the html representation of the board
        this.board = []; // the board matrix, used to represent the game state
        this.symbol = null; // the symbol player plays with

        // cache fileld/total cells so you don't have to loop on each move
        this.filled_cells = 0;
        this.total_cells = 9;

        // initialize
        (function(){
        var rows = Array.prototype.slice.call(table.getElementsByTagName('tr'));
        for (var i in rows) {
                self.html_cells[i] =Array.prototype.slice.call(rows[i].getElementsByTagName('td'));
                if (self.html_cells[i].length != rows.length) {
                throw new Error('The table should have the same number of rows and columns');
                }
                self.board[i] = [];
                for (var j in self.html_cells[i]) {
                        self.board[i][j] = null;
                        self.html_cells[i][j].setAttribute('data-row', i);
                        self.html_cells[i][j].setAttribute('data-cell', j);
                }
        }
        })();
```

board.js | “Ağ Üzerinden Oyun” devam...

```
table.onclick = function(event){
        if (!self.enabled) {
                return false;
        }
        var row = event.target.getAttribute('data-row'),
                cell = event.target.getAttribute('data-cell');

        self.set(row, cell, self.symbol);
        }

        var fire_event = function(event_name, data) {
        var event = document.createEvent('CustomEvent');
        event.initCustomEvent(event_name, true, true, data);
        document.dispatchEvent(event);
        }

        this.reset = function(){
        for (i in this.board) {
                for (j in this.board[i]) {
                        this.board[i][j] = null;
                }
        }
        this.filled_cells = 0;
        fire_event('board:reset');
        }

        /**
         * @param int row
         * @param int cell
         * @param mixed value
         * @return true if the given cell was empty; if you try to set already filled cell returns false
         */
        this.set = function(row, cell, value) {
        if (typeof this.board[row] == 'undefined') {
        throw new Error('invalid value ' + row + ' for row, must be in [0..' +this.board.length + ']');
        }
        if (typeof this.board[row][cell] == 'undefined') {
    throw new Error('invalid value ' + cell + ' for cell, must be in [0..' + this.board[row].length + ']')
        }
        if (this.board[row][cell] !== null) {
                return false;
        }
        this.board[row][cell] = value;
        this.html_cells[row][cell].textContent = value;
        this.filled_cells++;

        fire_event('board:set', {row: row, cell: cell, symbol: value});
        var is_winning = this.check_is_winning(row, cell);
        if (!is_winning) {
                this.check_is_full();
        }
        return true;
        }
```

board.js | “Ağ Üzerinden Oyun” devam...

```
	this.check_is_winning = function(row, cell) {
	console.log(row,cell)
	var h = v = d1 = d2 = 0,
		board_size = this.board.length,
		value = this.board[row][cell];
	for (i = 0; i < board_size; i++) {
		// check if all of the values are same...
		// ... at the row
		if (this.board[row][i] === value) h++;
		// ... at the column
		if (this.board[i][cell] === value) v++;
		// at the main diagonal
		if (this.board[i][i] === value) d1++;
		// at the secondary diagonal
		if (this.board[i][(board_size-1)-i] === value) d2++;
	}

	if (h == board_size || v == board_size || d1 == board_size || d2 == board_size) {
		fire_event('board:winning');
		return true;
	}
	return false;
	}
	this.check_is_full = function(){
	var is_full = (this.filled_cells == this.total_cells);
	if (is_full){
			fire_event('board:full');
	}
	return is_full;
	}
};
```

game.js | “Ağ Üzerinden Oyun”

```
var Game = function(table){
	var board = new Board(table);
	var connection = new Connection;

	var init_event_listeners = function(player){
	if (player) {

		document.addEventListener('board:set', function(event){

			is_your_move = (event.detail.symbol == board.symbol);
			if (is_your_move) {
				connection.send('set', event.detail, on_other_player_move);
				Game.message('Waiting for other player to move');
				board.enabled = false;
			} else {
				Game.message('Your turn');
				board.enabled = true;
			}
		});
```

game.js | “Ağ Üzerinden Oyun” devam...

```
	} else {
		document.addEventListener('board:set', wait);

	}
	document.addEventListener('board:winning', function(event){
		msg = 'Game Over';
		if (player) {
			msg = is_your_move ? 'You win' : 'You lose';
			toggle_new_game();
		}
		Game.message(msg);
	});
	document.addEventListener('board:full', function(event){
		Game.message('Game over');
		if (player) {
			toggle_new_game();
		}
	});
	}

	var on_other_player_move = function(data){
	board.set(data.value.row, data.value.cell, data.value.symbol);
	}

	var toggle_new_game = function(){
	document.getElementById('reset').style.display = 'block';
	}

	var wait = function(){
	connection.send('wait', null, on_other_player_move);
	}

	var game_start = function(data){
	if (data.player == 1) {
		board.symbol = 'X';
		Game.message('Your turn');
		board.enabled = true;

		init_event_listeners(true);
	} else if (data.player == 2) {
		board.symbol = 'O';
		Game.message('Waiting for other player to move');
		init_event_listeners(true);
		wait();
	} else {
		Game.message('You are watching a game');
		init_event_listeners(false);
		wait();
	}
}
```

game.js | “Ağ Üzerinden Oyun” devam...

```
	this.run = function(){
		Game.message('Waiting for other player');
		connection.send('init', null, game_start)
		progress(5,10, $('#progressBar'));
		}
	}
	Game.message = function(msg){
	document.getElementById('message').textContent = msg;
}
```

server.js | “Ağ Üzerinden Oyun”

```
var connections = {};
var games = {};
var url = require('http');

var send = function(res, data){
	headers = {
	'Content-type': 'application/json',
	'Access-Control-Allow-Headers': 'Content-type',
	'Access-Control-Allow-Origin': '*'
	}
	message = JSON.stringify(data);
	headers['Contnent-length'] = message.length;

	res.writeHead(200, headers);
	res.end(message);
}

var on_request = function(request,response){
	var body = '';
	if (request.method != 'POST') {
	send(response, null);
	return;
	}
	request.on('data', function (data) {
	body += data;
	});
	request.on('end', function(){
	var params = JSON.parse(body);

	var room = params.room;
	connections[room] = connections[room] || [];
	games[room] = games[room] || false;
	if (params.command == 'init') {
		if (connections[room].length == 1) {
			games[room] = true;
			send(connections[room][0], {player: 1});
			send(response, {player: 2});
			return;
		}
		if (games[room]) {
			send(response, {player: -1});
		}
	}
```

server.js | “Ağ Üzerinden Oyun” devam...

```
		// received command other than init, so game must be started
		games[room] = true;
		if (params.command == 'set') {
			for (var i = connections[room].length -1; i >= 0; i--) {
				res = connections[room][i];
				send(res, params);
				connections[room].splice(i,1);
			}
		}
		connections[room].push(response);
		});
}

var http = require('http').createServer(on_request).listen(8080);
```

connection.js | “Ağ Üzerinden Oyun”

```
Connection = function(){
		var xhr = new XMLHttpRequest();
		var url = 'http://localhost:8080';
		var game_callback = function(){};
		var timeout = 2;
		var last_command = null;
		var last_value = null;
		var self = this;

		xhr.onreadystatechange = function(){
		if (xhr.readyState == 4 && xhr.status == 200) {
			game_callback(JSON.parse(xhr.responseText));
			// reset the timeout to default
			timeout = 1;
			}
		}
		xhr.onerror = function()
		{
		Game.message('There is a problem with the connection. Reconnect in ' + timeout + '
seconds');
		window.setTimeout(function(){
			Game.message('Reconnecting...');
			self.send(last_command, last_value, game_callback);
		},timeout * 1000);
		if (timeout < 10) {
			timeout += 1;
		}
}
```

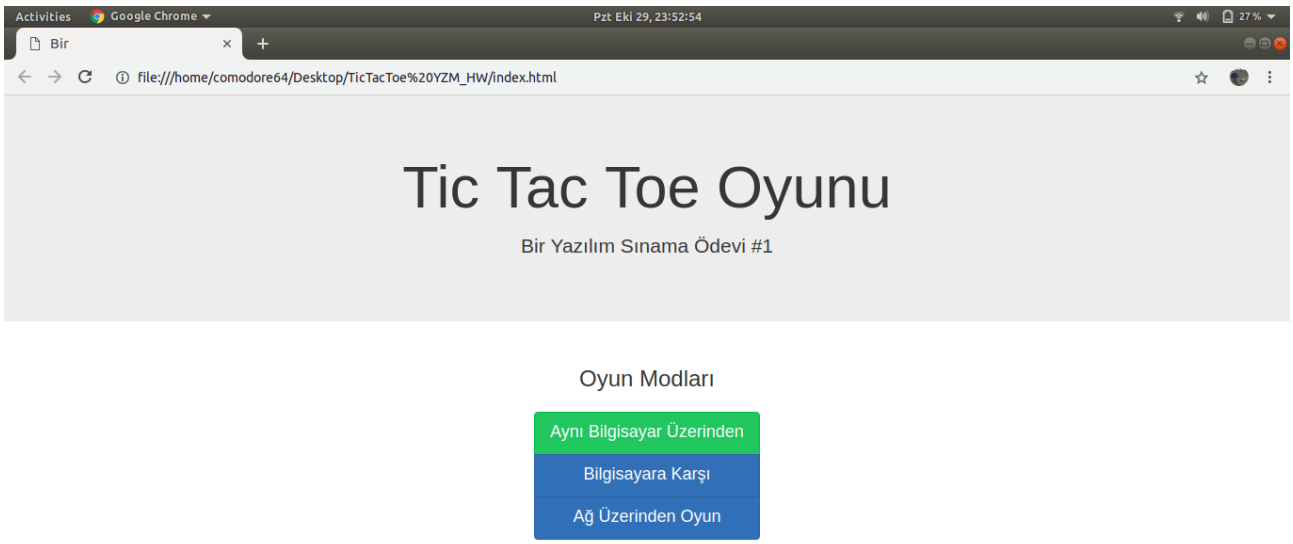
connection.js | “Ağ Üzerinden Oyun”

```
        this.send = function(command, value, callback) {
        console.log(command, value, callback);
        last_command = command;
        last_value = value;
        game_callback = callback;
        data = {
                room: 'test',
                command: command,
                value: value
        }
        xhr.open('POST', url);
        xhr.send(JSON.stringify(data));
        }

        this.stop = function(){
        xhr.abort();
        }
}
```

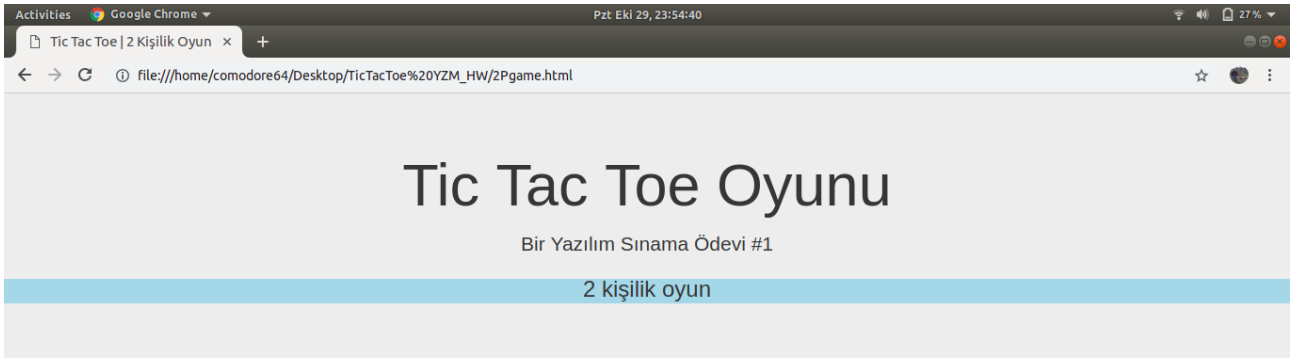
Geriye kalan JavaScript kodlarının tanımlı olduğu html dosyaları göndermiş olduğumuz klasördedir.

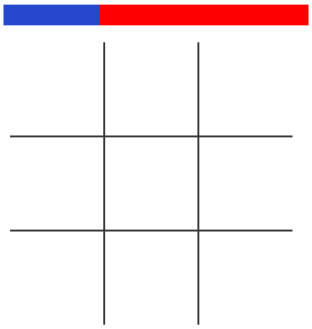
5. Ekran Çıktıları

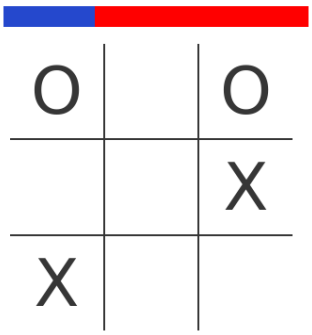


Karşımıza oyun modları olan bir arayüz gelmekterdir.

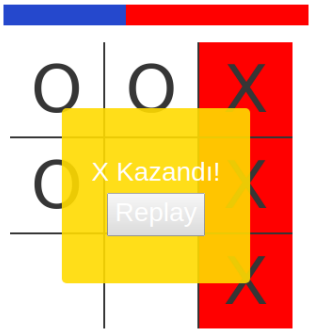
5. Ekran Çıktıları devam...

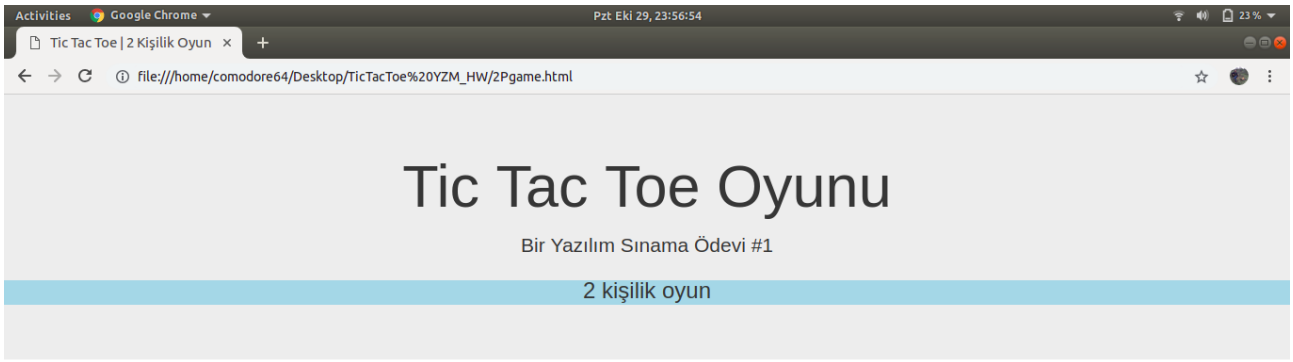




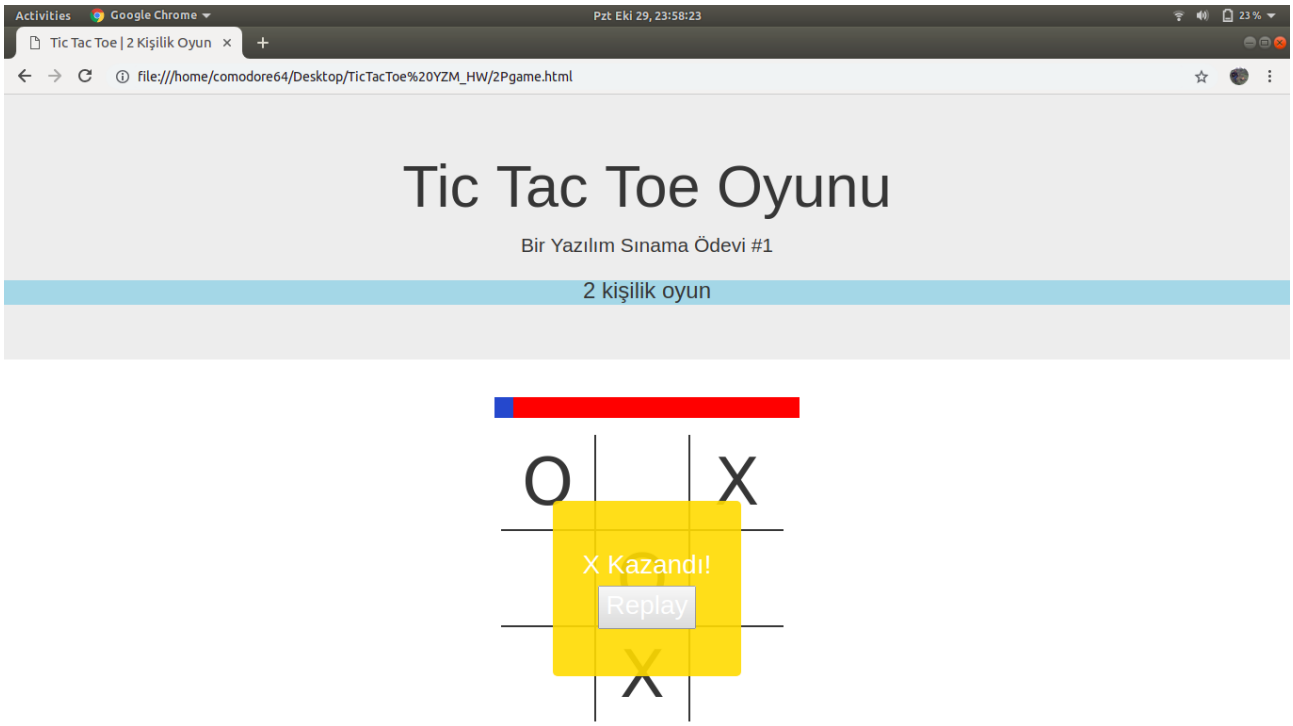


5. Ekran Çıktıları devam...

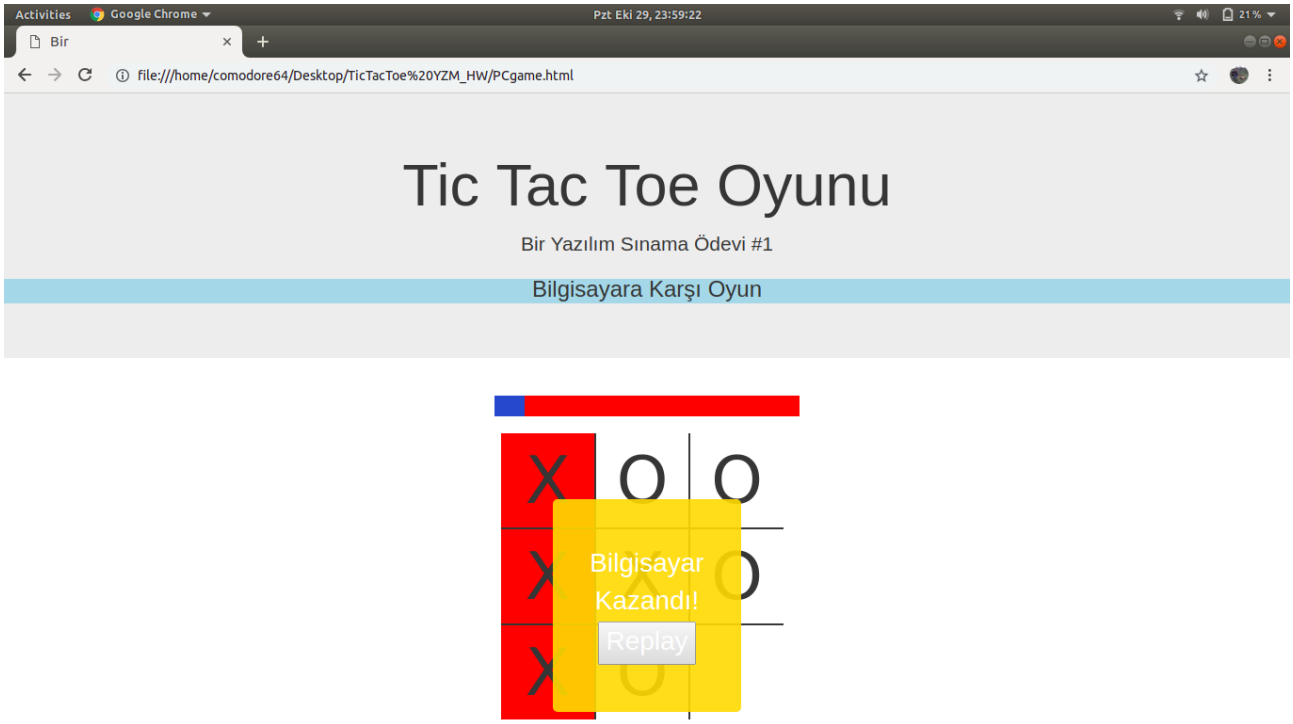


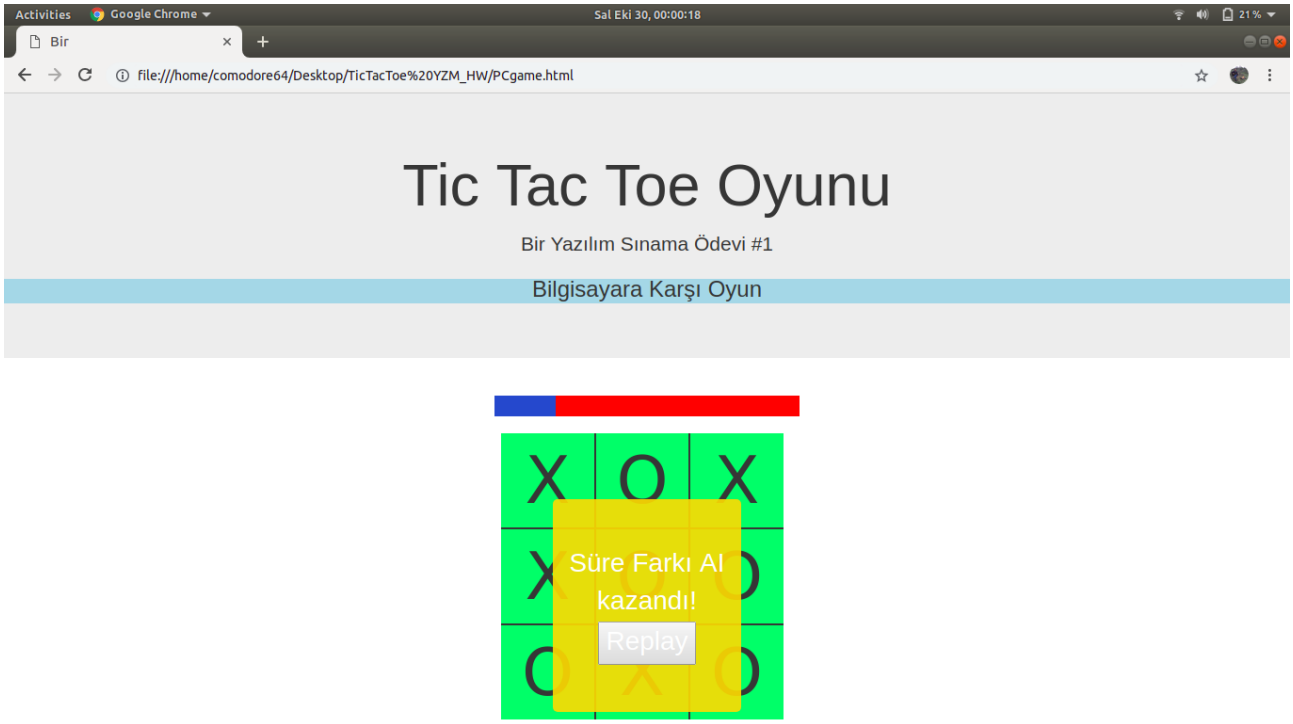


5. Ekran Çıktıları devam...



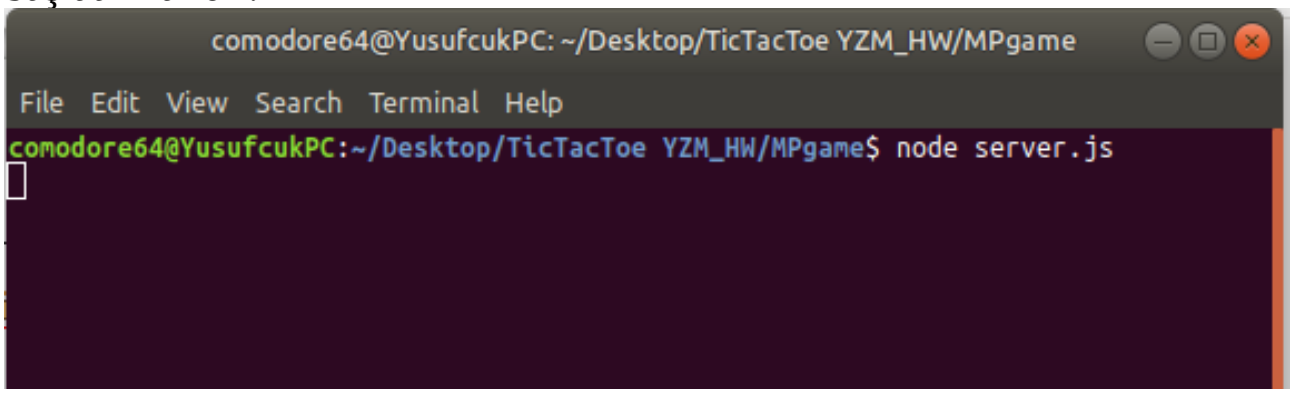
5. Ekran Çıktıları devam...



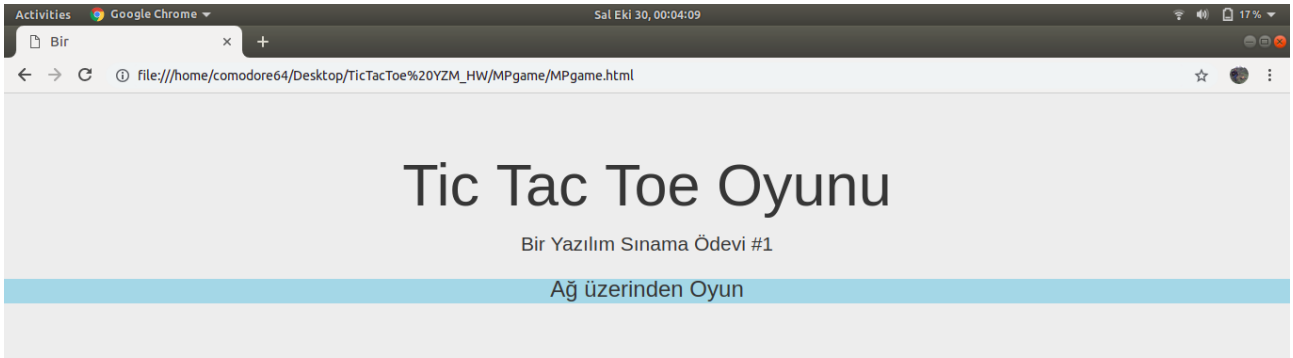


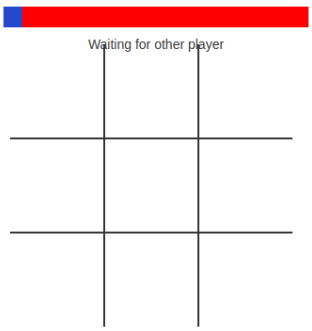
5. Ekran Çıktıları devam...

Ağ üzerinden oyun modu için kullanmış olduğumuz server ilk önce başlatılmalıdır.

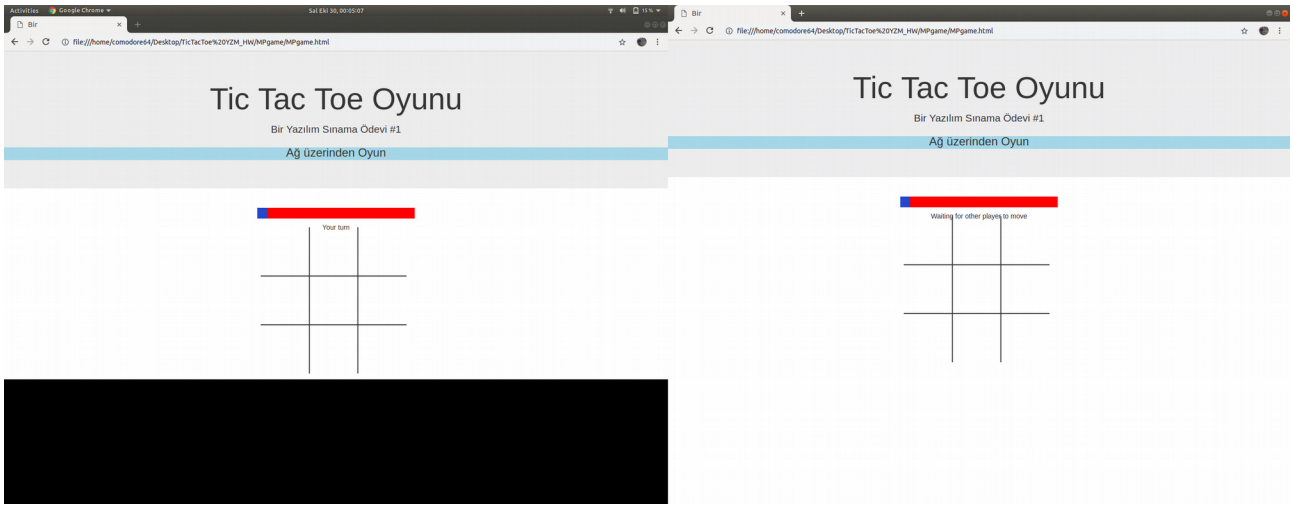


İkinci oyuncu beklemede...

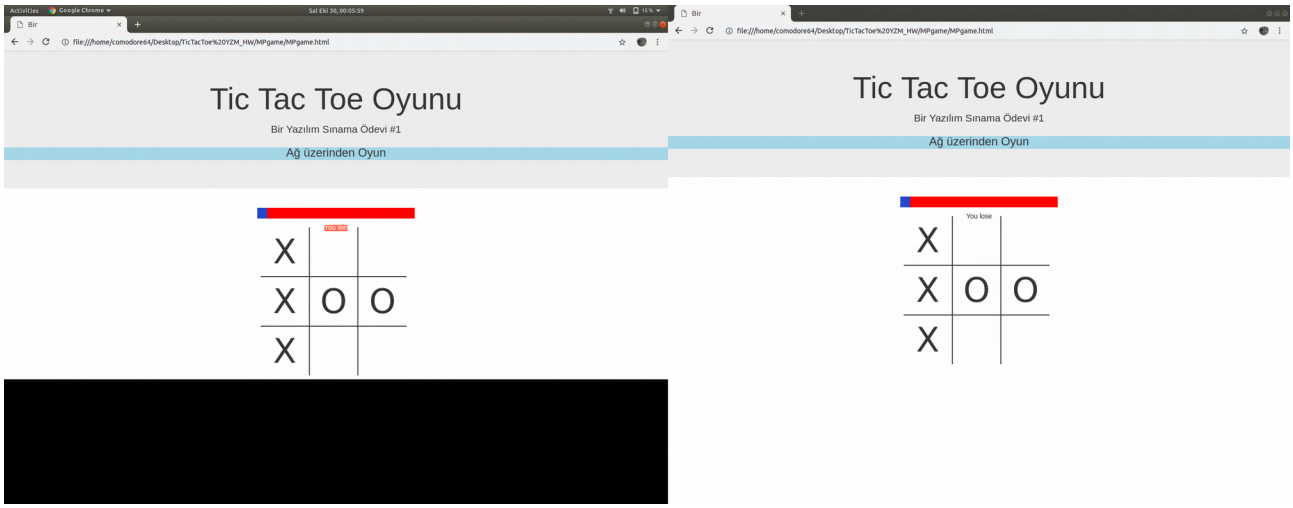




5. Ekran Çıktıları devam...



oyun şuan başladı….



Bu arada site responsive bir yapıya sahip mobil üzerinde tasarım bozulmadan çalışmaktadır.

TEST CASE | TİC TAC TOE OYUNU

Oyunun mouse ile çalıştırılması | Test Case #1

Risk Level	Yüksek
Purpose	Bu test durumunda oyun içerisinde tablo olarak tasarlanan 9 kareye mouse click ile seçilerek test edilmektedir.
Inputs	Oyun tablosu seçilmesi için mouse right click.
Expected Outputs	Yapılan click olayı ile birlikte tablo üzerinde X ve O nun ekranda oluşması.
Pass Criterias	Oyun kuralı gereği timerBar süresine bağlı olarak başlamaktadır. Oyunda 3 sembolü bir araya getiren kazanır.
Fail Criterias	Oyun oynanırken birkaç hızlı oyundan sonra timerBar doğru şekilde çalışmamaktadır.
Test Procedure	Test kullanıcısı, yazılımı gerekli ortamda çalıştırarak test etmektedir ve oyun içerisinde oluşacak bugları tespit etmelidir. Test işlemi tamamlandıktan sonra Pass/Fail olarak belirterek nedenleri ile birlikte raporlamalıdır.

Oyunun ağ üzerinde çalıştırılması | Test Case #2

Risk Level	Yüksek
Purpose	Bu test durumunda oyun iki client(istemci) ile çalıştırılarak tablo olarak tasarlanan 9 kareye mouse click ile seçilerek test edilmektedir.
Inputs	2 Client & oyun tablosu seçilmesi için mouse right click
Expected Outputs	Oyun içersinde birinci client ve ikinci client oyuncuların girmesi ile oyun başlar ve tablo üzerinde X ve O nun ekranda oluşması.
Pass Criterias	Oyun içerisinde eş zamanlı bir şekilde iki client oyuncu arasında oyunun çalışması ve reset veya gameover durumlarında oyunun yeninden başlatılması.
Fail Criterias	Oyun içersinde iki client oyuncu arasında gameover moduna düştükten sonra reset ve yeniden başlatma durumlarının oluşmaması
Test Procedure	Test Kullanıcısı, yazılımı gerekli ortamda iki kişi kullanarak test etmektedir . Oyun içerisinde oluşacak aksaklıkları ve hataları tespit etmelidir. Test işlemi tamamlandıktan sonra Pass/Fail olarak belirterek nedenleri ile birlikte raporlamalıdır.

Oyunun beraberlik durumunun testi | Test Case #3

Risk Level	Yüsek
Purpose	Bu test durumunda oyunun timerBar kuralına göre başlatılarak beraberlik durumunun test edilmesi.
Inputs	Yapılan click olayı ile birlikte tablo üzerinde X ve O nun ekranda oluşması.
Expected Outputs	Oyun kuralı gereği timerBar süresine bağlı olarak başlamaktadır. Oyunda 3 sembolü bir araya getiren kazanır. Eğer beraberlik durumunda süresi az olan veya çok olan kişinin kazananması.
Pass Criterias	Oyun içerisinde 3 sembolün bir araya gelmediği durumlarda timerBar süresine bakılarak kazananın belirlenmesi.
Fail Criterias	Oyunda diğer oyuncunu son hakkı ile X veya O nun bir araya getirdiği zaman bereaberlik sonucunu ulaşması ve timerBar süresine göre kazananın belirlenmesi.
Test Procedure	Test kullanıcısı, yazılımı gerekli ortamda çalıştırarak test etmektedir ve oyun içerisinde oluşacak bugları tespit etmelidir. Test işlemi tamamlandıktan sonra Pass/Fail olarak belirterek nedenleri ile birlikte raporlamalıdır.