

# Tic tac toe

Av Olle Lindkvist, student ID: olli5947

## Sammanfattning

Den här webbsidan utgör ett nätverksbaserat spel, tre i rad. Webbsidan är kopplad till en databas och genom den så kan användaren skapa spelarprofiler och olika spel instanser. För att spela måste man vara två personer, varje spelare väljer sin sparade användar profil. Sedan är det ett vanligt tre i rad spel, där ena spelaren placerar ut X och ena spelaren placerar ut O. Varje spel blir även sparat i databasen och användarna kan kolla på tidigare färdig-spelade omgångar i en "scoreboard".

# Framtagande

## Idé samling och start av projekt

När jag tänkte på vad jag har lärt mig i denna kurs så tänkte jag direkt på databaser. Jag har såklart lärt mig om det i tidigare kurser, men med dessa uppgifter, med hjälp av PHP så lärde jag mig hur det fungerar mer i praktiken. Därav ville jag göra en webbsida som fungerar med hjälp av en databas.

Jag tänkte på vad jag har gjort i tidigare kurser gällande projektarbeten och kom fram till att det roligaste och mest givande för mig har varit att skapa spel (Ex MPROG).

Då började jag tänka vad för spel som skulle kunna använda sig av funktionaliteter som jag har lärt mig av i denna kurs. Det landade i att ett "tre i rad" spel kan uppfylla kraven.

För att påbörja projektet ville jag skapa lite generella punkter som är tänkt att vara med i spelet. Det slutade med något i denna stil:

- Tydligt och responsivt tre i rad spel
- Skapa egna användare
- Möta varandra online
- Spara varje spelad match i en databas
- Kunna visa upp en scoreboard av tidigare spelade matcher

## Skapandet av databasen och första funktionaliteten

För att börja skapa detta projekt tänkte jag att det är bäst att ta det lugnt i början, inte direkt gå in på hela spelmekaniken. Så jag började med att få till funktionaliteten för användaren att kunna skapa "användarprofiler". Tanken var att man skulle kunna skapa en profil, exempelvis ditt namn och sedan visa vems tur det är med ditt namn osv.

För att implementera detta skapade jag en databas, med hjälp av programmet XAMPP som jag tidigare använde mig av kursen och sedan phpMyAdmin. När jag skapade databasen använde jag mig av det engelska namnet för tre i rad, namnet på databasen blev `tic_tac_toe_db`. Där skapade jag en tabell som heter `players`, där kolumnerna `player_id` och `player_name` ska spara skapade användare i sin databas. För att sedan koppla databasen till min webbsida och kunna använda mig av den framöver så skapade jag en PHP - fil som heter `db_connection.php`, där jag helt enkelt hämtar databasen med `mysql`, `host` och `dbname`.

För att implementera funktionaliteten med att skapa användare behövde jag skapa en HTML sida som innehåller ett formulär, detta formulär skickar sedan den skapade profilen med hjälp av Javascript och PHP till min databas. När jag skickar data med hjälp av Javascript och PHP använder jag mig av `fetch` och sedan skickar jag data i JSON format.

När detta var skapat implementerade jag funktionaliteten för att kunna hämta de skapade användarprofilerna. Det var relativt enkelt och i detta skede skapade jag även en liten meny, som ska fungera som en första sida. Så när användaren väljer att klicka på `startgame` ska

den komma till en annan HTML - sida där användaren ska välja en skapad användarprofil. Här använde jag mig av Javascript och PHP igen för att kunna hämta data från databasen och visa i två listor, där ena listan ska representera spelaren som ska vara X och den andra listan ska visa spelaren O.

## Spelet

Efter jag hade skapat min tänkta databas, påbörjat lite funktionalitet för min webbsida blev det tid att skapa själva spel funktionaliteten. Eftersom att jag senare i projektet vill kunna visa en poängtavla tänkte jag att spelet ska skapas i en databas. Därav skapade jag en ny tabell som jag döpte till games, i den finns det game\_id, player\_x, player\_o, board\_state, current\_player och winner. Alla dessa kolumner har sin funktionalitet och hjälper till för att skapa detta spel. För att gå igenom kolumnerna lite snabbt kan jag börja med game\_id, som används som primärnyckel för att skapa en unik id till varje nytt spel. Kolumnerna player\_x och player\_o blir tilldelade 1 eller 2, de är till för att hålla koll på exempelvis vilken spelares tur det är.

Nästa kolumn heter board\_state och håller hela tiden koll på vilken användare som klickar vart på spelbrädet, till en början är board\_state tom, sen fylls det på tills någon användaren har tre i rad eller att det blir lika.

Current\_player kolumnen håller koll på vilken spelare som börjar och default value är X vilket betyder att spelare X alltid börjar.

Winner får sitt värde om någon spelare vinner eller det blir lika, den startar med värdet NULL.

För att få allt detta att uppdateras korrekt och fungera så har jag skapat tre olika php filer; create\_game.php, load\_game.php och play\_move.php. För att få dem att fungera måste jag även ha Javascript filer, i detta fall har jag en game\_logic.js och select\_players.js.

Create game anropas när användaren har valt två spelare, alltså den som ska vara X och O. Create game gör så att det skapas ett nytt spel i tabellen, games, som jag nämnde tidigare. När den är skapad så skickas det game\_id med i url:n till en ny HTML sida som heter game.html. Med hjälp av det nyskapade game\_id:et går det att hämta det nya spelet, med load\_game.php.

När sidan är skapad och spelbrädet har laddats in kommer varje "drag" av spelaren att uppdateras med hjälp av play\_move.php. Den uppdaterar vilken spelares tur det är, var användaren klickar, kollar om någon vinner, ser till så att det inte går att klicka i en ruta som redan är ifylld etc.

## Scoreboard

När spellogiken fungerade som jag ville så var det dags för nästa steg, att skapa en scoreboard/ poängtavla. För att göra det behövde jag skapa en Javascript fil med namnet scoreboard.js och en php fil med namnet get\_scoreboard.php. Jag skapade en separat Javascript fil för detta på grund av att jag vill kunna återanvända den koden, jag vill kunna hämta den på två olika ställen, direkt i menyn och efter ett avslutat spel (det implementerar jag senare).

För att göra det skapade jag en global funktion i javascript, openScoreboard(), som jag kan använda i både menu.js och game\_logic.js. I den globala funktionen hämtar jag den data som jag vill visa från min data med hjälp av fetch(get\_scoreboard.php) och sedan med hjälp av innerHTML skapar jag en poängtavla i form av en tabell.

## Sista ändringar och insikt

I detta skede hade jag ett fungerande spel, med en fungerande databas och ytterligare lite funktionalitet. Jag var relativt nöjd, men det hade tagit en lång tid att komma hit, längre än vad jag trodde att det skulle ta. Därför kände jag att försöka implementera någon slags "online spel" funktionalitet inte fanns på kartan, det skulle ta för lång tid. Så istället fixade jag lite små ändringar, såsom att gå till webbsidan med lite CSS. Jag implementerade även en ruta som kommer upp i slutet av ett spel, där användaren kan välja att spela igen, gå till menyn eller titta på poängtavlan.

I slutändan är jag relativt nöjd med mitt arbete, det finns definitivt fler funktioner som jag hade velat implementera, såsom online spel eller spela emot en dator. Men med den tiden jag hade tillgänglig för detta projekt kände jag att det inte gick att implementera mycket mer. Denna funktionaliteten blev relativt invecklad för ett simpelt tre i rad spel, men jag är ändå nöjd med min prestation.

## Form

Den färdiga webbsidan består av 3 CSS, 4 HTML, 5 Javascript och 7 PHP filer. Det låter mycket men vissa av dem är relativt små och alla uppfyller sin funktionalitet, exempelvis är en php fil, db\_connection.php bara till för att kunna använda databasen i de andra php-filerna, med hjälp av require 'db\_connection.php'.

För att förklara centrala kodavsnitt i mitt spel kan jag gå igenom hur det fungerar när en användare spelar. Vi hoppar till funktionaliteten av att välja en färdig skapad användarprofil. När användare är på denna sida blir de mött av två select element, där det finns två listor med skapade spelaprofiler. Listan uppdateras med hjälp av select\_players.js och get\_players.php:

```
fetch('../PHP/get_players.php')
  .then(res => res.json())
  .then(data => {
    data.players.forEach(player => {
      const optionX = document.createElement("option");
      optionX.value = player.player_id;
      optionX.textContent = player.player_name;
      selectX.appendChild(optionX);

      const option0 = document.createElement("option");
      option0.value = player.player_id;
      option0.textContent = player.player_name;
      select0.appendChild(option0);
    });
  })
  .catch(err => console.error("Error fetching players:", err));
```

```
<?php
require '../PHP/db_connection.php';

$stmt = $db->prepare("SELECT player_id, player_name FROM players");
$stmt->execute();
$players = $stmt->fetchAll(PDO::FETCH_ASSOC);

echo json_encode(['players' => $players]);
```

Här hämtas player\_id och player\_name från databasen och sedan visas det i en lista, det finns två olika select element med samma data, vad som händer närmast är detta:

```

btnStart.addEventListener("click", () => {
  const xId = selectX.value;
  const oId = selectO.value;
  if (xId === oId) {
    alert("You cannot choose the same player for both X and O!");
    return;
  }
  fetch("../PHP/create_game.php", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
      playerX: xId,
      playerO: oId
    })
  })
  .then(res => res.json())
  .then(data => {
    if (data.success) {
      window.location.href = `../HTML/game.html?game_id=${data.game_id}`;
    } else {
      alert("Failed to create game.");
    }
  });
});

```

Koden kollar först så att det är två unika valda spelarprofiler, sedan skapas spelet i databasen och skickas som url, med det skapade game\_id. Användaren kommer då till HTML-sidan game.html, där spelet laddas med hjälp av load\_game.php i game\_logic.js:

```

fetch("../PHP/load_game.php?game_id=${gameId}")
  .then((res) => res.json())
  .then((data) => {
    updateBoard(data);
    statusElem.textContent = data.status;
    checkGameOver(data.status);
  })
  .catch((err) => console.error("Error loading game:", err));

```

```

$gameId = $_GET['game_id'];
if (!$gameId) {
  echo json_encode([
    'board' => array_fill(0, 9, ""),
    'status' => "No game_id provided"
  ]);
  exit;
}

$sql = "SELECT
  g.*,
  px.player_name AS player_x_name,
  po.player_name AS player_o_name
FROM games g
JOIN players px ON g.player_x = px.player_id
JOIN players po ON g.player_o = po.player_id
WHERE g.game_id = :gid";

$stmt = $db->prepare($sql);
$stmt->execute(['gid' => $gameId]);
$game = $stmt->fetch(PDO::FETCH_ASSOC);

```

I load\_game finns även funktionaliteten att visa vilken användares tur det är i frontenden, med hjälp av att använda JOIN på player\_x och player\_o till players tabellens player\_id. När spelet har laddats så lyssnar en EventListener på när en användare klickar på spelplanen och då uppdateras databasen vid varje klick:

```

board.addEventListener("click", (event) => {
  if (event.target.classList.contains("cell") && !event.target.classList.contains("taken")) {
    const cellIndex = event.target.dataset.index;

    // Skicka drag till servern
    fetch("../PHP/play_move.php", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        cell: cellIndex,
        gameId: gameId
      })
    })
    .then((res) => res.json())
    .then((data) => {
      updateBoard(data);
      statusElem.textContent = data.status;
      checkGameOver(data.status);
    })
    .catch((err) => console.error("Error playing move:", err));
  }
});

```

Play\_move.php börjar med att säkerställa att ingen data saknas och liknande:

```

$data = json_decode(file_get_contents("php://input"), true);
$cellIndex = $data['cell'] ?? null;
$gameId = $data['gameId'] ?? null;

if ($cellIndex === null || !$gameId) {
  echo json_encode([
    'board' => [
      "", "", "", "", "", "", "", "", ""
    ],
    'status' => "Missing data"
  ]);
  exit;
}

```

Sedan har den hand om funktionalitet som att kolla om en ruta är ledig, växla spelare och uppdatera databasen vid varje klick:

```

// Kolla om rutan är ledig
if ($board[$cellIndex] === "") {
    $board[$cellIndex] = $currentPlayer;
    $winner = checkWinner($board);

    // Om ingen vinnare, växla spelare. Om vi har en vinnare behåll current_player och winner
    if (!$winner) {
        $nextPlayer = ($currentPlayer === 'X') ? 'O' : 'X';
    } else {
        $nextPlayer = $currentPlayer;
    }

    // Uppdatera databasen: bräde, current_player och winner (om någon vann)
    $updateQuery = $db->prepare("
        UPDATE games
        SET board_state = :board_state,
            current_player = :next_player,
            winner = :winner
        WHERE game_id = :game_id
    ");
    $updateQuery->execute([
        ':board_state' => json_encode($board),
        ':next_player' => $winner ? $currentPlayer : $nextPlayer,
        ':winner' => $winner,
        ':game_id' => $gameId
    ]);
}

```

Det är det stora hela, sen finns det såklart mer funktionalitet runt om på webbsidan.

## Pseudokod

Denna symbol: // symboliserar en generell kommentar

Game\_logic.js:

//Vid laddning av dokumentet (DOM fully loaded)

If document is loaded then:

//Lokala variabler: board, statusElem, btnBack, gameOverModal etc.

//Läser in gameId från URL-parameter ?game\_id=...

urlParams = Ny URL sökparameter från windows.location.search

gameId = hämta game\_id från parametern

response = Hämta speldatan från load\_game.php + gameId

If response.ok then

data = parseJSON(response)

Call function updateBoard(data)

statusElem.text = data.status (status kommer från JSON-data)

Call function checkGameOver(data.status)

Else

log("Error loading game")

//Ett klick på spelbrädet

Board\_onclick = (event) ->

If event.target innehåller class("cell") och inte innehåller class("taken")

cellIndex = event.target.dataset(från databasen)("index")

//Skicka POST-förfrågan till play\_move.php med fetch

//Det ser i princip exakt ut som "response" från ovan

FUNKTION updateBoard(data)

    Töm spelbrädet

    LOOP index och cellValue i data.board

        Skapa nytt div element

        Lägg till div i classlist, för att styla med CSS

        Ge varje cell ett index

        Sätt cellens text innehåll till ett Värde ("X", "O" eller tom sträng)

    If cell upptagen

        CSS ändrar utseende av musen så att det inte går att klicka

FUNKTION checkGameOver(status)

    If status innehåller "Winner: " eller status innehåller "draw"

        Öppnar dialogruta med status

FUNKTION openGameOverModal(statusMessage)

    Text.title = "Game Over!"

    Text = statusMessage

    Visa gameoverModal

Play\_move.php:

data = Inputs kommer via POST från game\_logic.js

cellIndex = data['cell']

gameId = data[gameId]

If cellIndex = NULL eller gameId = NULL

    RETURNERA JSON(Tomma fält på brädet, status: "Missing data")

sql = Hämta spelet från games + JOIN players

Result\_game = exekveraSQL(med sql, gameId)

If NOT result\_game

    RETURNERA JSON(Tomma fält på brädet, status: "No game found!")

Board = hämta board\_state

currentPlayer = hämta current\_player

winner = hämta winner

//Detta körs när spelet är slut

If winner != NULL

    If winner = 'X'

        displayStatus = Winner + player\_x\_name + 'X'

    else if winner = 'O'

        displayStatus = Winner + player\_o\_name + 'O'

    Else



```
    displayStatus = It's a draw  
    RETURNERA JSON(board, status)
```

```
If board(cellIndex) = ""  
    Lägg current_players symbol(X eller O)  
    newWinner = Kollar om winner  
  
    If newWinner != NULL  
        Om ingen vinnare växla spelare  
    Else  
        Om någon vinner behåll current_player och winner
```

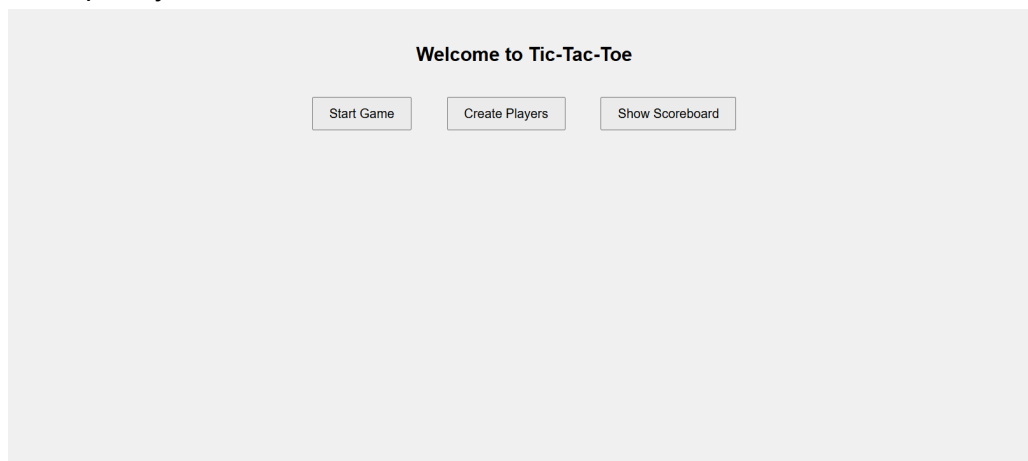
```
//Uppdatera i databasen
```

```
//Detta körs varje drag  
If winner != NULL  
    If winner = 'X'  
        displayStatus = Winner + player_x_name + 'X'  
    else if winner = 'O'  
        displayStatus = Winner + player_o_name + 'O'  
    Else  
        displayStatus = It's a draw  
Else  
    //Om ingen vinnare nextPlayer har turen  
    RETURNERA JSON(board, status)
```

```
FUNKTION checkWinner(board)  
    winningCombos = olika sätt att vinna på i form av rad, kolumn eller diagonalt i en array  
    LOOP combo i winningCombo  
        //Loopar igenom alla vinnarkombinationer  
        //om spelbrädet blir fullt returnera "D" för draw
```

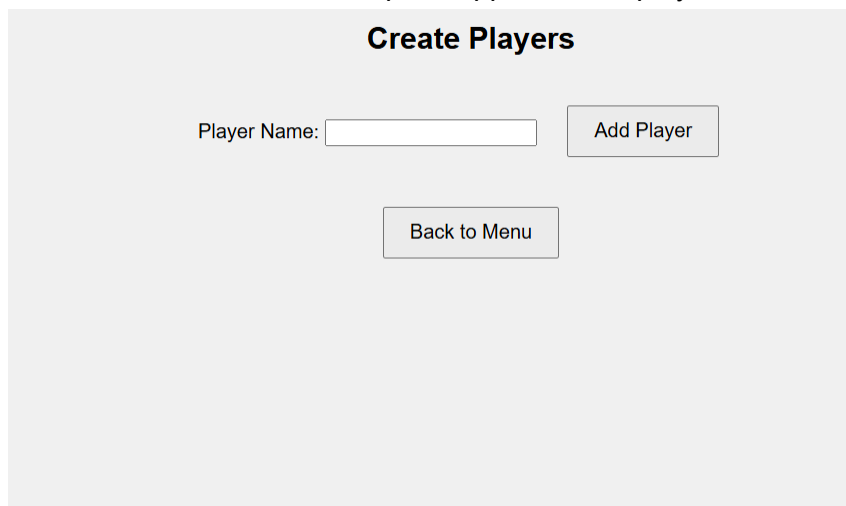
# Funktion

När användaren laddar upp min webbsida möts den av en meny med tre olika alternativ som ser ut på följande sätt:



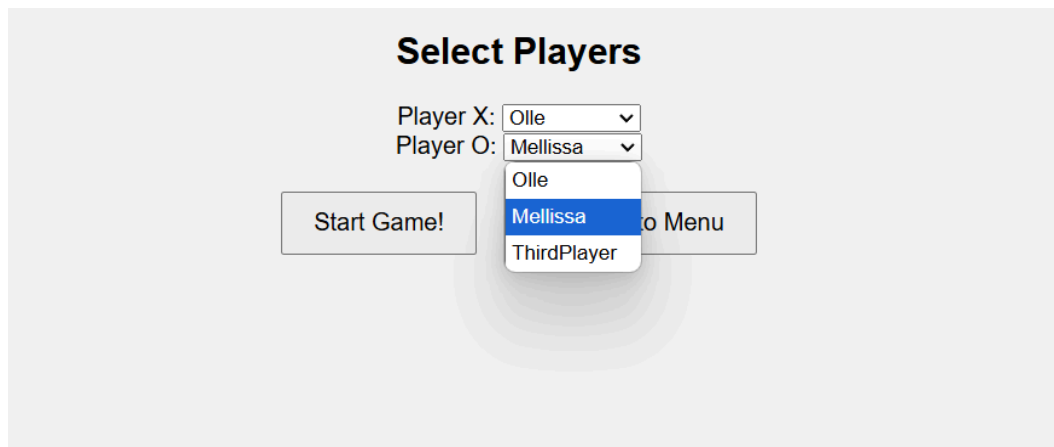
The screenshot shows a light gray rectangular area representing a web page. At the top center, the text "Welcome to Tic-Tac-Toe" is displayed in a bold, black font. Below this text, there are three rectangular buttons arranged horizontally. Each button has a thin black border and contains text in a small, black font. From left to right, the buttons are labeled "Start Game", "Create Players", and "Show Scoreboard".

För att spela detta spel måste användaren först skapa minst två spelarprofiler, för att göra det måste användaren klicka på knappen create player och då ser det ut såhär:

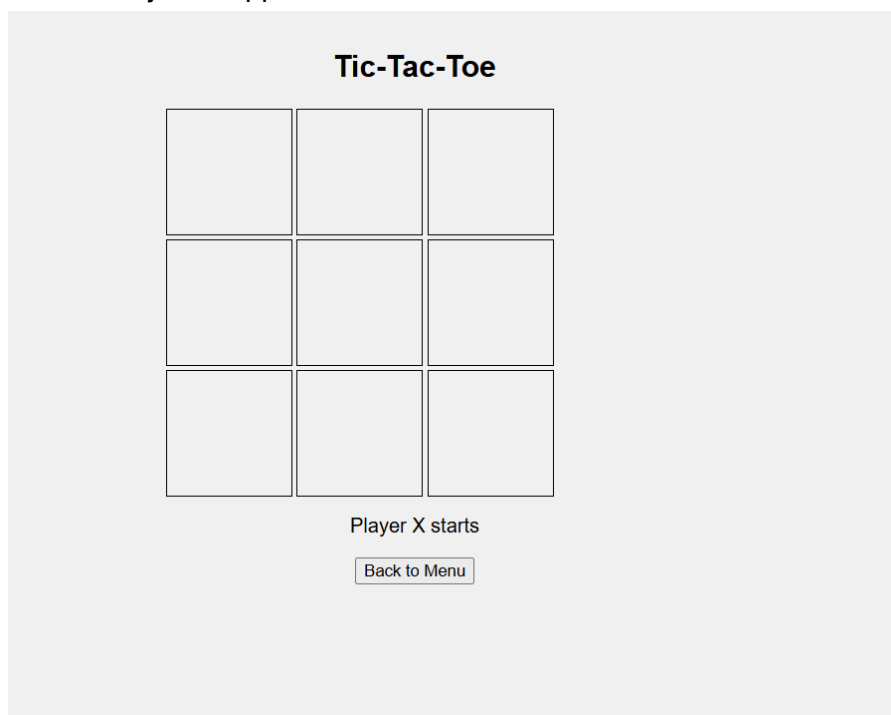


The screenshot shows a light gray rectangular area representing a web page. At the top center, the text "Create Players" is displayed in a bold, black font. Below this text, there is a form consisting of a label "Player Name:" followed by a text input field. To the right of the input field is a rectangular button with a thin black border labeled "Add Player". Below the "Add Player" button, centered horizontally, is another rectangular button with a thin black border labeled "Back to Menu".

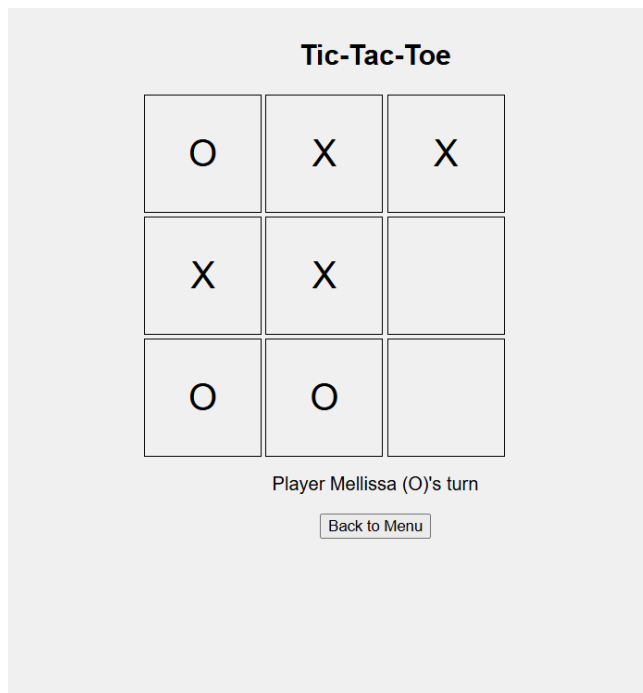
När användaren skriver ett namn och klickar på add player så skapas det en ny spelare i tabellen players i databasen. Nu kan användaren gå tillbaka till menyn och klicka på start game, då kommer följande sida upp där användaren ska välja två spelarprofiler som ska möta varandra:



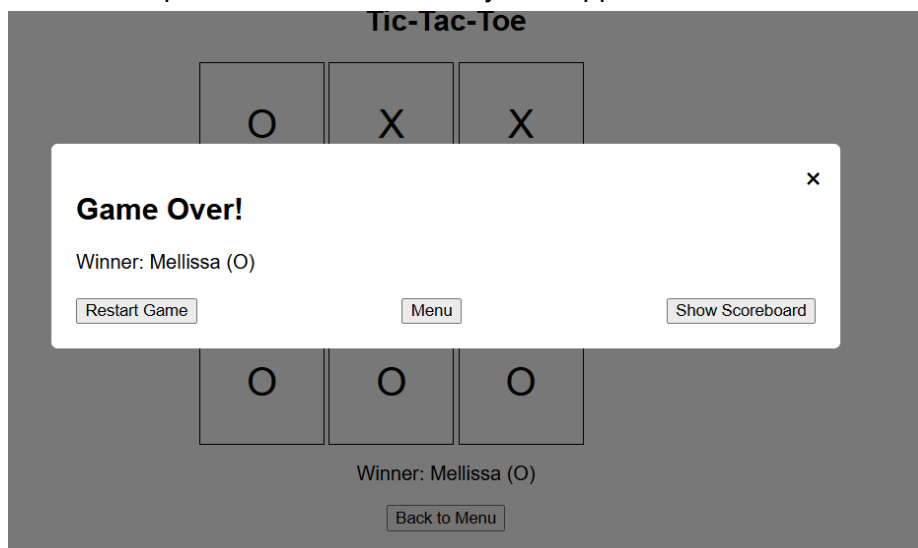
När två olika spelarprofiler är valda kan användaren klicka på start game knappen, då kommer följande upp:



Sedan är det bara att börja spela, som det står så börjar den spelaren som valde "X", alltså i detta exemplets fall spelare Olle. Sedan kan det se ut så här under spelets gång:



Och när en spelare vinner kommer följande upp:



Här kan man alltså antingen klicka på restart game, menu eller show scoreboard. Om man klickar på restart game kommer man till select players sidan igen. Klickar man på meny knappen kommer man till menyn och klickar man på show scoreboard kommer något som liknar detta upp (beroende på hur mycket man har spelat tidigare, tabellen fylls på dynamiskt):

## Scoreboard

x

Game ID	Player X	Player O	Winner
32	Olle	Mellissa	Mellissa (O)
31	Olle	Mellissa	Olle (X)
30	Olle	ThirdPlayer	Olle (X)
29	ThirdPlayer	Olle	ThirdPlayer (X)
27	Olle	Mellissa	Draw
25	Olle	Mellissa	Mellissa (O)
13	Olle	Mellissa	Olle (X)
12	Mellissa	Olle	Mellissa (X)
11	Mellissa	Olle	Mellissa (X)
10	Mellissa	Olle	Mellissa (X)
9	Olle	Mellissa	Olle (X)
8	Olle	Mellissa	Olle (X)
7	Mellissa	Olle	Mellissa (X)
6	Olle	Mellissa	Draw
5	Olle	Mellissa	Olle (X)
3	Olle	Mellissa	Olle (X)
2	Olle	Mellissa	Olle (X)
1	Olle	Mellissa	Olle (X)

Denna ruta kommer även upp i om du klickar på samma knapp i menyn.