Mini Project

Tic-Tac-Toe Game

Student Name: Kartikey Gupta	**UID:** 22BCA10176
Branch: UIC BCA	**Section/Group:** 3-B
Semester: 5th	**Date of Performance:** 27/10/24
Subject Name: Web Development	**Subject Code:** 22CAH-301

Aim/Overview of the Practical

Objective:

This project aims to develop an interactive Tic-Tac-Toe game utilizing the ASP.NET MVC framework. Through this game, the project will demonstrate the practical application of the Model-View-Controller (MVC) architecture, showcasing both the backend functionality and the frontend design. The game includes key features such as turn-based play, win-condition verification, and an intuitive interface to enrich the user experience.

Technologies Covered:

Built on the ASP.NET MVC framework, the project uses C# for backend logic, ensuring a well-defined separation of concerns. HTML and CSS are used to create a responsive, user-friendly interface. Additionally, JavaScript adds interactivity by handling user prompts and simple animations, enhancing the overall gameplay experience.

Project Scope:

The project covers the full development lifecycle of a basic web application, from configuring the ASP.NET MVC environment in Visual Studio to building the game's logic and interface. The MVC structure provides a modular and easily maintainable architecture, making the application both scalable and adaptable. Upon completion, the game will be tested locally to ensure smooth functionality across various devices and web browsers.

Task to be Done

Project Setup: The initial step involves installing Visual Studio and setting up a new ASP.NET MVC project. The folder structure is arranged to house models, views, and controllers, following MVC architecture best practices.

Model-View-Controller (MVC) Setup: · Define a TicTacToeModel to represent the game's state, including player symbols, the current player, and the game board. Create views to display the board and results, and develop controllers to manage user interactions and game logic.

Game Logic Implementation: Implement the core game logic, such as validating moves, alternating players, and checking for winning conditions after each turn. This includes horizontal, vertical, and diagonal checks to determine if a player has won.

UI Design: · The interface will prioritize simplicity and ease of use, enabling players to intuitively engage with the game. The game board will be displayed in a grid format using either HTML tables or div elements styled with CSS.

Deployment:The application will be run locally via IIS Express, making it accessible in a web browser. In the future, deployment to a cloud service may be considered for broader accessibility and testing.

Algorithm/Flowchart

Algorithm Overview:

1. **Initialize Project**: Set up the project environment, create a new MVC project in Visual Studio, and configure settings.
2. **Define Models**: Create classes that represent database tables (e.g., User, Product).
3. **Set up Database and Entity Framework**: Configure and connect to a SQL database, apply migrations, and use Entity Framework to manage data.
4. **Controller Actions**: Write methods in controllers to handle CRUD operations.
5. **Build Views**: Design views to display data and interact with users.
6. **Authentication Setup**: Implement user login, logout, and registration.
7. **Testing & Debugging**: Test each component and resolve errors.
8. **Deploy Application**: Host the application on a server or local machine.

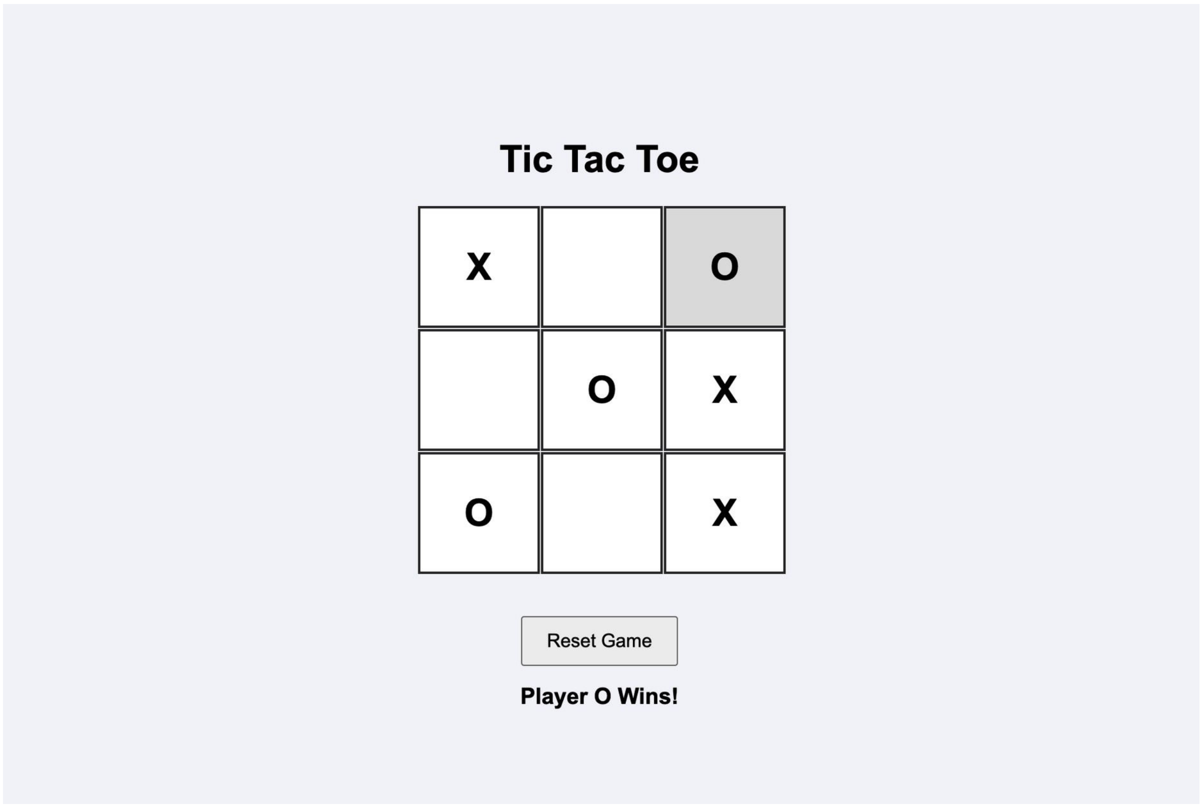
Flowchart:

Project Setup → Model Creation → Database Configuration → CRUD Implementation → UI Design → Authentication → Testing → Deployment

- **Dataset**

  - **Structure:** The dataset for this project is represented as a simple 3x3 grid within the TicTacToeModel. Each cell in the grid can hold one of three states: "X", "O", or null, indicating whether it is occupied and by which player. The game logic maintains the state in memory, simplifying the implementation and allowing for fast updates without needing a database.

  - **Configuration:** The game state is managed directly within the model class. Upon starting a new game, the board is initialized, and players take turns making moves until a winner is declared or the game ends in a draw. This in-memory configuration makes it easier to track the game flow and state changes.



Code for Experiment/Practical

- **Project Setup Code**: Using Visual Studio, create a new ASP.NET Core MVC project.

Model Example:

```
namespace TicTacToe.Models
{
  public class GameModel
  {
    public char[,] Board { get; set; }
    public char CurrentPlayer { get; set; }
    public string Message { get; set; }

    public GameModel()
    {
      Board = new char[3, 3];
      for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
          Board[i, j] = ' ';
      CurrentPlayer = 'X';
      Message = "Player X's turn";
    }

    public bool MakeMove(int row, int col)
    {
      if (Board[row, col] == ' ')
      {
        Board[row, col] = CurrentPlayer;
        if (CheckWin(CurrentPlayer))
        {
          Message = $"Player {CurrentPlayer} wins!";
          return true;
        }
        if (IsBoardFull())
        {
          Message = "It's a draw!";
          return true;
        }
        CurrentPlayer = CurrentPlayer == 'X' ? 'O' : 'X';
        Message = $"Player {CurrentPlayer}'s turn";
        return true;
      }
      return false;
    }

    private bool CheckWin(char player)
```

```
        {
            for (int i = 0; i < 3; i++)
           {
               if (Board[i, 0] == player && Board[i, 1] == player && Board[i, 2] == player)
                   return true;
               if (Board[0, i] == player && Board[1, i] == player && Board[2, i] == player)



                   return true;
            }
            if (Board[0, 0] == player && Board[1, 1] == player && Board[2, 2] == player)
               return true;
            if (Board[0, 2] == player && Board[1, 1] == player && Board[2, 0] == player)
               return true;
            return false;
        }

        private bool IsBoardFull()
        {
            foreach (char cell in Board)
               if (cell == ' ')
                   return false;
            return true;
        }
    }
}
```

Controller Code:

```
using Microsoft.AspNetCore.Mvc;
using TicTacToe.Models;

namespace TicTacToe.Controllers
{
    public class GameController : Controller
    {
        private static GameModel _game = new GameModel();

        public IActionResult Index()
        {
            return View(_game);
        }

        public IActionResult MakeMove(int row, int col)
•         {
•             _game.MakeMove(row, col);



•             return RedirectToAction("Index");
•         }
•         public IActionResult Reset()
•         {
•
•
•
•             _game = new GameModel();
•             return RedirectToAction("Index");
•         }
•     }
• }
```

View Code (Index View for Products):

```
• @model TicTacToe.Models.GameModel

• @{
• ViewData["Title"] = "Tic-Tac-Toe";
• }

• <!DOCTYPE html>
• <html lang="en">
• <head>
• <meta charset="UTF-8">
• <meta name="viewport" content="width=device-width, initial-scale=1.0">
• <title>@ViewData["Title"]</title>
• <style>
• table {
• border-collapse: collapse;
• }

• td {
• width: 60px;
• height: 60px;
• text-align: center;
• font-size: 24px;
• }

• .cell {
• border: 1px solid #000;
• }

• .clickable {
```

- cursor: pointer;
- }
- </style>

- </head>
- <body>
- <h1>@Model.Message</h1>
- <**form** method="post" **asp-action**="MakeMove">
- <table>
- @for (int i = 0; i < 3; i++)
- {
- <tr>
- @for (int j = 0; j < 3; j++)
- {
- <td class="cell">
- @if (Model.Board[i, j] == ' ')
- {
- <button class="clickable" type="submit" name="row" value="@i" formaction="/Game/MakeMove?row=@i&col=@j" style="width: 60px; height: 60px;">@Model.Board[i, j]</button>

- }
- else
- {
- @Model.Board[i, j]
- }
- </td>
- }
- </tr>
- }
- </table>
- </**form**>
- <br />
- <**form** method="get" **asp-action**="Reset">
- <button type="submit">Reset Game</button>
- </**form**>
- </body>
- </html>

Result/Output/Writing Summary

Expected Result:

The expected outcome is a fully functional Tic Tac Toe web application that allows users to engage in a two-player game. Players can alternate turns, and the application will accurately determine the winner or indicate if the game ends in a draw.

User Interaction:

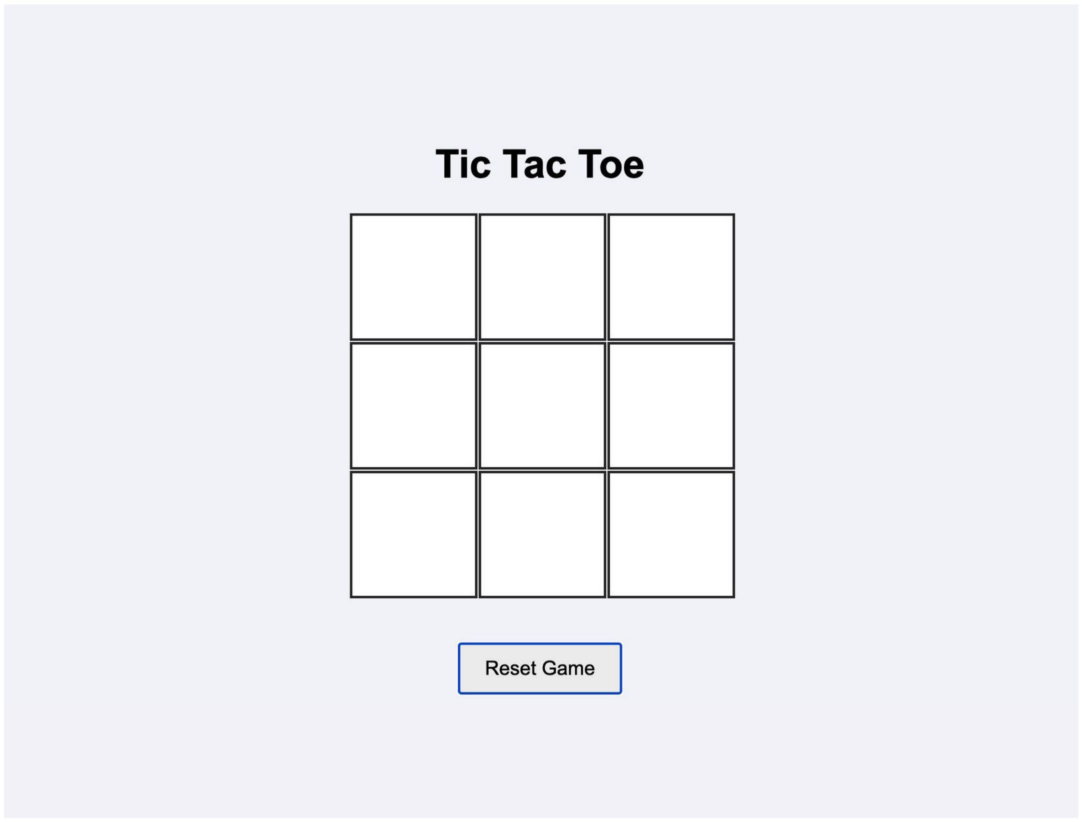
Users can click on the cells of the game grid to make their moves, with visual feedback displayed after each action. The game will update dynamically to show the current player and the state of the board, allowing for an engaging and interactive experience.

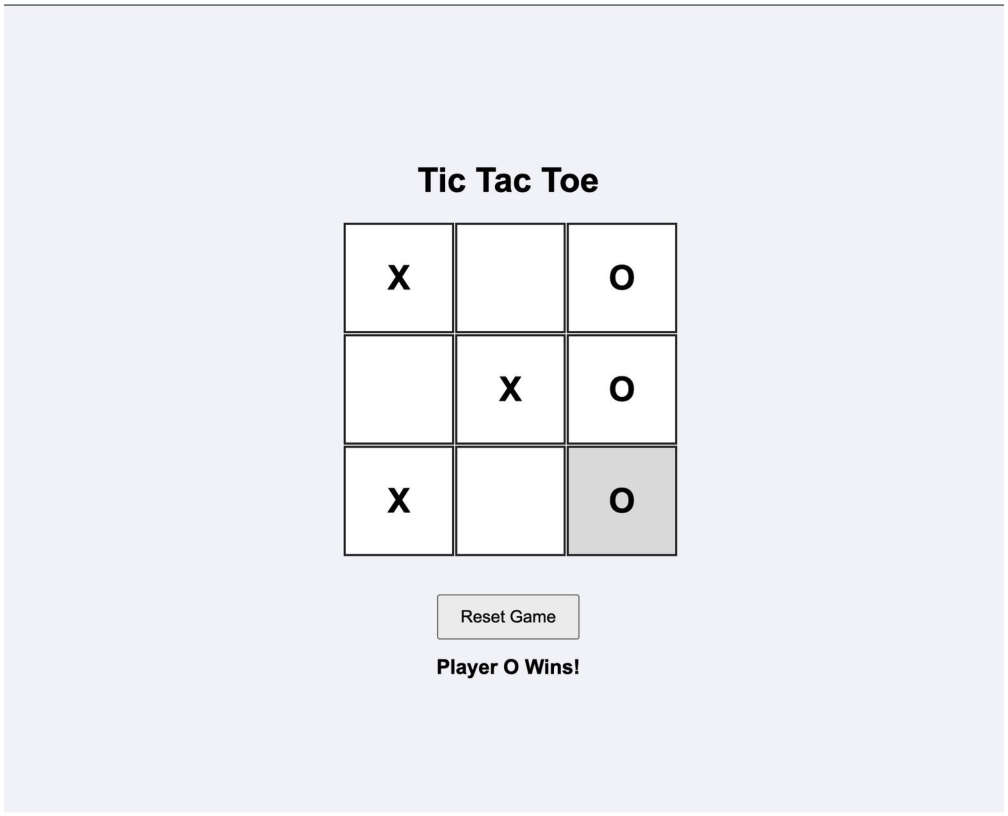
UI/UX:

The interface is designed to be intuitive and user-friendly, featuring a simple layout with a clear game grid and player indicators. CSS is used to enhance visual aesthetics, ensuring that the game is responsive and accessible on various devices, including desktops and mobile phones.

Deployment:

The application is hosted locally using IIS Express during development, allowing for easy access via a web browser. Once tested, the plan is to deploy the application to a cloud service for public access, ensuring that it can be played online by multiple users without additional configuration.





Player O Wins!

Learning Outcomes (What I have learnt):

1. **Project Structuring in ASP.NET Core MVC**: Gained experience in organizing projects using the MVC pattern, focusing on the separation of models, views, and controllers.
2. **Game Logic Implementation**: Learned to design the game logic for Tic Tac Toe, including move validation and win condition checks.
3. **Backend and Frontend Development**: Developed backend controllers to manage game state and created responsive frontend views with Razor, HTML, and CSS.
4. **Dynamic User Interaction**: Implemented features for real-time updates of the game board as players make their moves, enhancing interactivity.
5. **Deployment Processes**: Experienced deploying the application locally and preparing for future cloud deployment to make it accessible to users.
6. This project provided hands-on experience in creating a full-fledged web application, from initial setup through deployment, while focusing on both the gameplay mechanics and user experience.

Evaluation Grid:

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Demonstration and Performance (Pre Lab Quiz)		5
2.	Worksheet		10
3.	Post Lab Quiz		5