Kuwait University
College of Computing Sciences and Engineering
Department of Information Science

COMP 2139
Web Application Development with C# .NET

Lab 9
The Tic-Tac-Toe Application

Table of Contents

Laboratory #9 – The Tic-Tac-Toe Application

1. Laboratory Objective

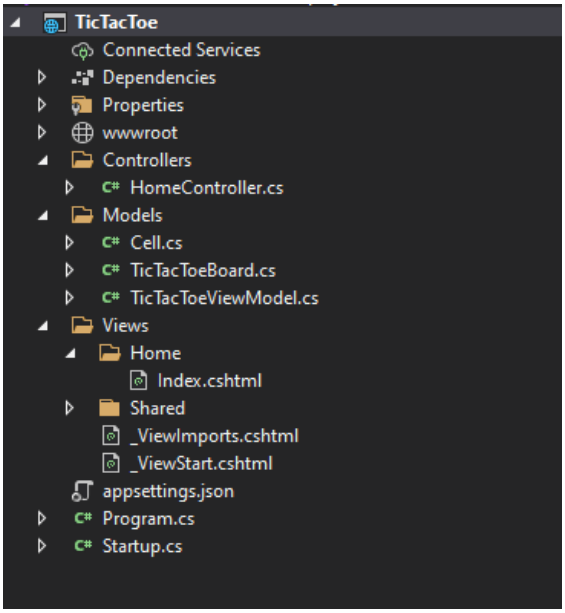
The goal of this lab is to build a single page application (SPA) that uses model binding to pass data between controller and views.

2. **Laboratory Learning Outcomes:** After conducting this laboratory students will be able to:
    a. Create and maintain a Git repository project
    b. Create the Tic-tac-Toe Application using C#.NET
    c. Build a single-page, using model-binding to pass data between controller and views.

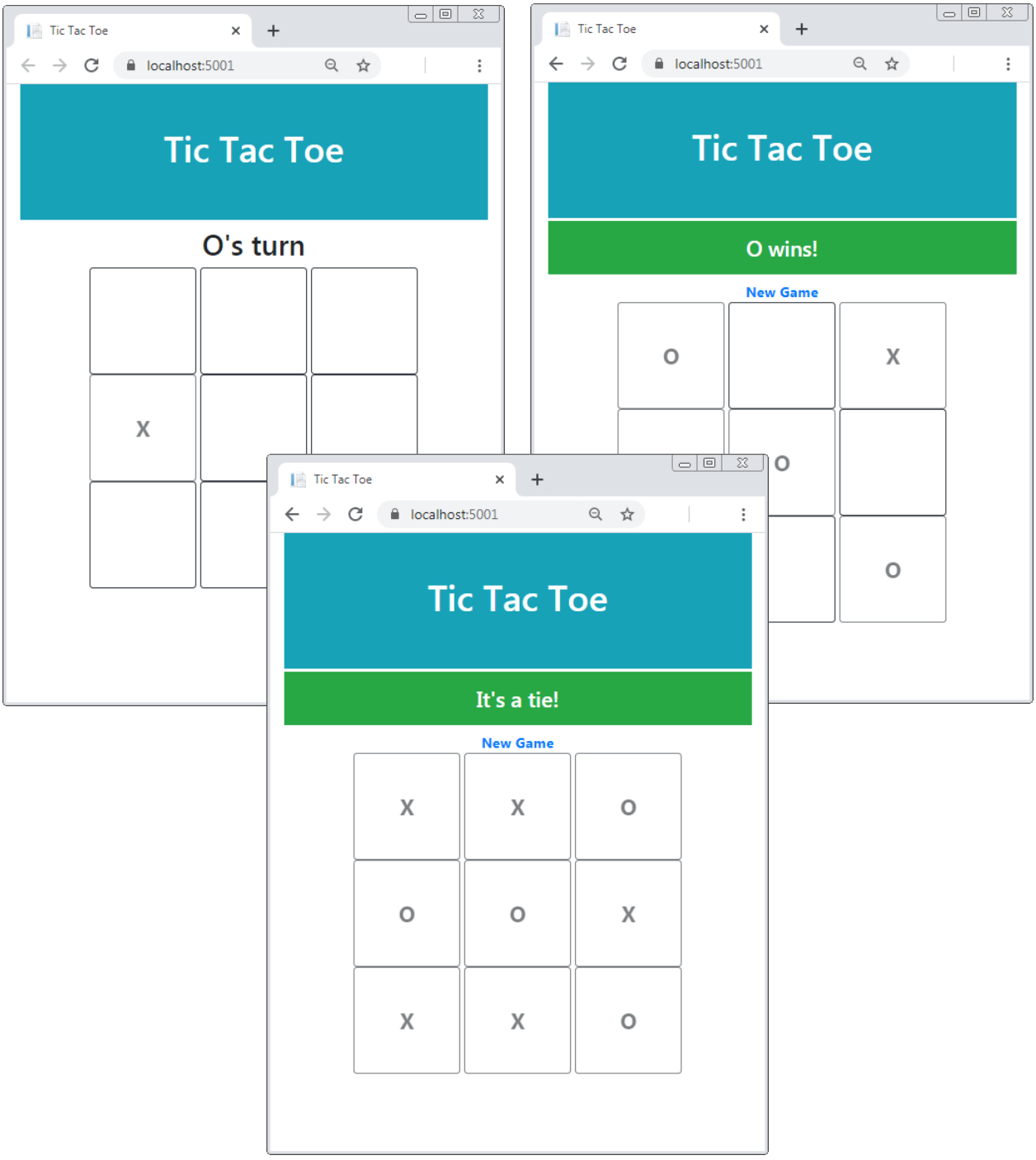
3. Laboratory Instructions

For this lab, we will build a single-page, web application that will have a structure similar to the one presented below:

The Tic-Tac-Toe Game Application



The Tic-Tac-Toe Application



Specifications

- When the application starts, it should display nine <button> elements, styled as illustrated in the screenshot provided, to appear as a 3-x-3 grid.
- Above the grid, an <h2> element should say that it's X's turn.
- When the user clicks a button, the page should post to the server, where the X or O for that button is saved.
- When the page re-loads, the clicked <button> element should be marked X or O and should be disabled. In addition, clicking the browser's Refresh button should redisplay the page, not post the previous click again.
- After each turn, the <h2> element above the grid should say whose turn it is.
- When there are three Xs or three Os in a line (vertically, horizontally, or diagonally), the game is over and the player with the three marks in a line wins. When this happens, the application should say who won and replace the <h2> element with a New Game link.
- When there are no more blank cells but there's no winner, then the game ends in a tie. When this happens, the app should notify the user and replace the <h2> element with a New Game link.
- When the user clicks the New Game link, a new grid of <button> elements should display. Above the grid, an <h2> element should say that it's X's turn.

Tic-Tac-Toe Model Representation in Code (List of Cells with Ids)

TopLeft	**TopMiddle**	**TopRight**
MiddleLeft	**MiddleMiddle**	**MiddleRight**
BottomLeft	**BottomMiddle**	**BottomRight**

Creating Git Repository (Optional / Review)

1. Refer to past lab instructions on creating new repositories and clone them into your local workspace.

Creating the Web Application (Review)

1. Within Visual Studio select **File→New→Project**, select **ASP.NET Core Web Application**.
    a. Fill in your desired project details for Project Name, and Location, then select create
    b. From the resultant page, select **Web-Application (Model-View-Controller),** then **Create**
    c. You should then be presented with your workspace to start creating your project.
    d. Clean-up (delete/remove) all auto-generated **controllers**, **views**, **models** from the project

Create the Tic-Tac-Toe Model

1. Based on the screenshot provided for the Tic-Tac-Toe game, create the application model classes, based on the following design:

Cell

Attribute	**Description**
Id	String primary key
Mark	String used to identify if the position has been marked
IsBlank	Used to return Boolean state value of Mark
IsEndCell	Used to determine if the cell is on the end (“Right”)

TicTacBoard

Attribute	**Description**
Cells	List<Cell> represents the list of cells
HasWinner	Boolean representing if a winner has been determined
WinningMark	String used to mark 1 of the 3 winning cells (3 cells for a win)
HasAllCellsSelected	Boolean used to track if all cells have been marked
Methods	**Description**
TicTacToeBoard	Constructor, initialize list of cells in the constructor Include one cell object for each cell on the tic tac toe grid
GetCells	Returns Cells back to the caller
CheckForWinner	Determines if a winner exists given the current status of the board.
IsWinner	Accepts three marks and validate if all marks match, and not null.

TicTacToeViewModel

Attribute	**Description**
Cells	List<Cell> for the view to query against.
Selected	Single Cell for view.
IsGameOver	Boolean that is used to determine if the game is won.

Create the Context

1. No database for this lab, so this is not required.

Update appsettings.json

1. No database for this lab, so no changes required.

Update Startup.cs

1. Default routing is sufficient

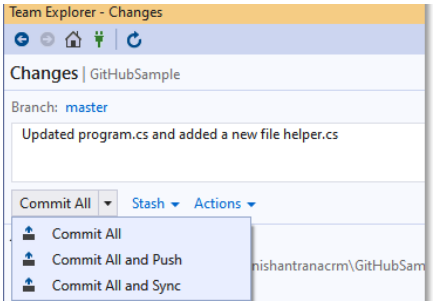
Create the HomeController

1. The HomeController for the Tic-Tac-Toe application requires the following action methods:
    i. IActionResult **Index()** – [*HttpGet*]
    ii. RedirectToActionResult **Index(TicTacToeModelView)** – [*HttpPost*]

Run and test the Application, then Commit

1. Run the application and test to make sure it functions as expected (correct any errors).

2. If all is well, commit the code to your code GitHub repository. To commit the code, navigate to the git changes tab, and select **commit all and push**



Home Work

1. Based on the wireframe/screenshots provided, create the following views for the project namely:
    - Index.cshtml
    - Layout.cshtml