Project Planning Worksheet - Unlocked Work (Supplemental)

Below is how GWC Alum Aria Cusenza thought through the case study.

Instructions: Design a plan for how you would approach this take home project.

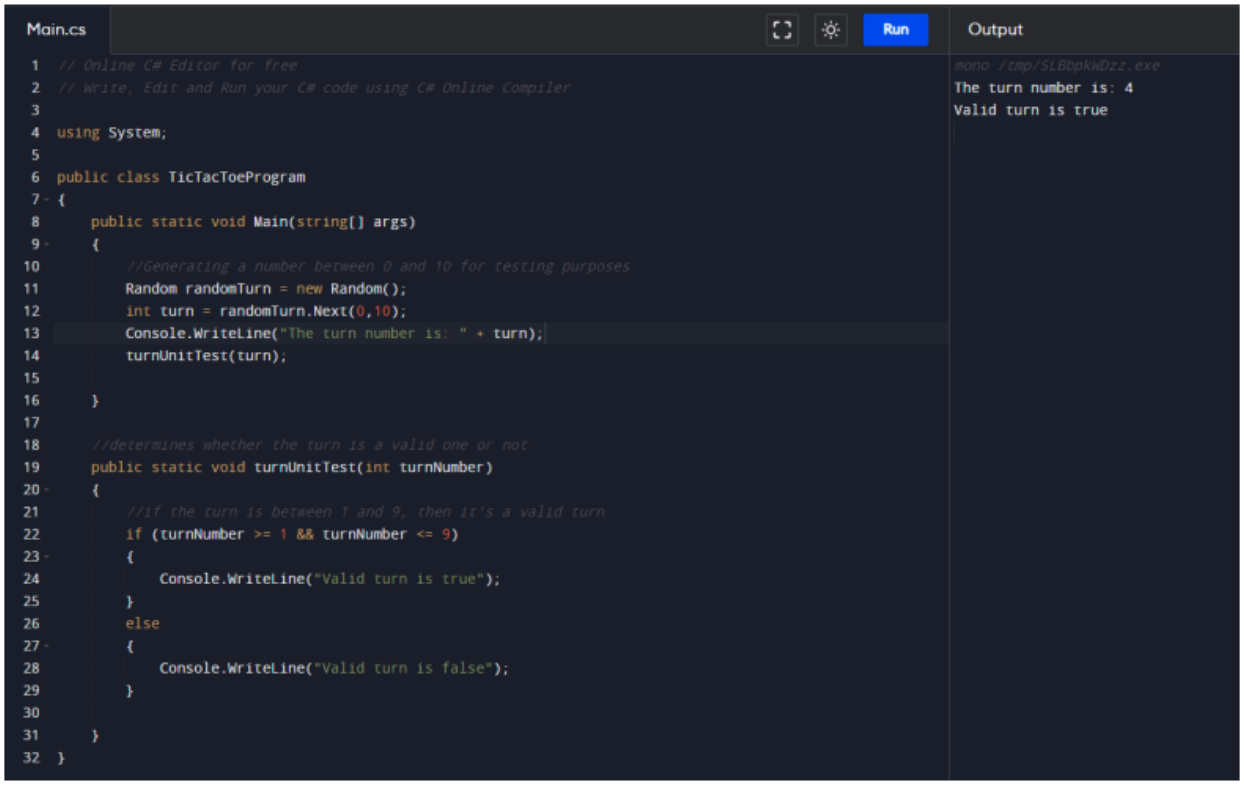
Start by considering the fundamentals of Tic-Tac-Toe:

- ➔ Goal: Be the first to get three in a row
- ➔ Challenge: You don't know where your opponent will place their symbol
- ➔ Core Mechanics: Blocking and writing
- ➔ Components: 9 square grid, players, Xs and Os
- ➔ Rules: There are 2 players. Each player alternates turns placing their symbols until the grid is full or one player gets three in a row, vertically, horizontally, or diagonally.
- ➔ Space: 3x3 grid

Requirements *(What requirements do you have for the game?)*	• The game will need to work with a computer playing against the user. • The user will take their turn, and then they will have to wait for the computer to take their turn. • The game will need to check after every turn if the player who just took their turn has won. • The game will need to account for a tie if neither player has placed three in a row.
Clarifying Questions *(What clarifying questions would you ask the hiring manager before you begin?)*	1. Does the computer that you're playing against need to have varying difficulty levels? 2. Do you care about the level of design for the game display, or is the game functionality more important?
Technical Decisions *1. Consider which programming language you want to use and why?* *2. What user interface will you use?* *3. How will the user provide inputs and see the outputs?*	1. Technical Decision #1: Which programming language to use a. **Rationale**: I chose to write the program in C# for a couple of reasons. The first reason is that it's the programming language that I feel like I have the most experience with, and that I feel most comfortable coding in. The second reason is that there is also the potential to create a GUI form that can be used to display the tic, tac, toe board. 2. Technical Decision #2: What user interface will I use? a. **Rationale:** The user interface that I'm going to use is Visual Studio. I chose Visual Studio because I have a lot of experience with it, so I'm comfortable using it. I also chose Visual Studio because it comes with a nice GUI interface that can be used with C#. 3. Technical Decision #3: How will the user provide inputs and see the outputs? a. **Rationale:** The user will provide inputs on the GUI page that I

	create. The user will also see the outputs on that same GUI page.
Core Technical Components *(Think about how the program will need to work, what are the rules of the game? How will you know if someone won or lost? What edge cases will you consider? What will you ignore?)*	• Each player will take one turn, then let the next player take their turn. This will continue for 9 turns total. • The code will need to check after every turn to see if there are any matches that are three in a row. If there are no matches (and the number of turns has not exceeded 9), then continue the game, otherwise if there is a match then display the winner. • The game will need to make sure that if a winner has not been determined by the time it takes to fill up all the squares, then to stop the game and tell the user they tied. • An edge case that I will need to consider is if the user inputs something other than an X or an O. • Another edge case I will need to consider is whether the user inputs a capital X or O. I can convert the letter in my code to upper to fix this. • Something that I can ignore is if the user takes a long time to play the game, I don't need to add a timer.
Testing Plan *(Consider the scope of your testing and the types of cases or scenarios you would test?)*	Some of the things that I would want to test are the user inputting something other than an X or an O. I will also want to test inputting lowercase and uppercase X's and O's. I will also want to test the game until the end multiple times to make sure it displays the correct winner if the user wins, the computer wins, or if there's a tie.
Task Breakdown *(What steps will you need to take to complete this take home project? Be specific and list in order of priority)*	1. Come up with a plan for the necessary components of my test. 2. Determine the core logic of the program on paper before creating the actual program. Plan out everything in sections that the program will need to accomplish. 3. Decide on what programming language and IDE I will use. Make sure that they're going to be able to accomplish what I need them to. 4. Start coding. 5. Complete coding for the main functionality of the test. 6. Test things out and make sure the code works without errors.

	7. Clean up the code, add any necessary comments, and make sure it's clear and easy to read. 8. See if there's any ways to make my code faster or more efficient. 9. Create a README file for my program
Write a Unit Test *(Write the code for a simple unit test of one of your game components in the your language of choice)*	**What will your unit test check for? Why did you decide to write this unit test?** What I will be checking for in my unit test is if the turn is a valid turn. I decided to write this unit test because it's important to make sure that the game is counting turns correctly (and understanding how many turns there are) so that the game can continue and conclude correctly.
	Paste a link to your unit test code or paste a screenshot of your unit test code below.




```
// Online C# Editor for free
// Write, Edit and Run your C# code using C# Online Compiler

using System;

public class TicTacToeProgram
{
    public static void Main(string[] args)
    {
        //Generating a number between 0 and 10 for testing purposes
        Random randomTurn = new Random();
        int turn = randomTurn.Next(0,10);
        Console.WriteLine("The turn number is: " + turn);
        turnUnitTest(turn);

    }

    //determines whether the turn is a valid one or not
    public static void turnUnitTest(int turnNumber)
    {
        //if the turn is between 1 and 9, then it's a valid turn
        if (turnNumber >= 1 && turnNumber <= 9)
        {
            Console.WriteLine("Valid turn is true");
        }
        else
        {
            Console.WriteLine("Valid turn is false");
        }

    }
}
```