***GROUP: DISNEY PRINCESS Simple Tic-tac-toe***

***Members:***

***Jasmine Casem***

***Beatrice Pedroche***

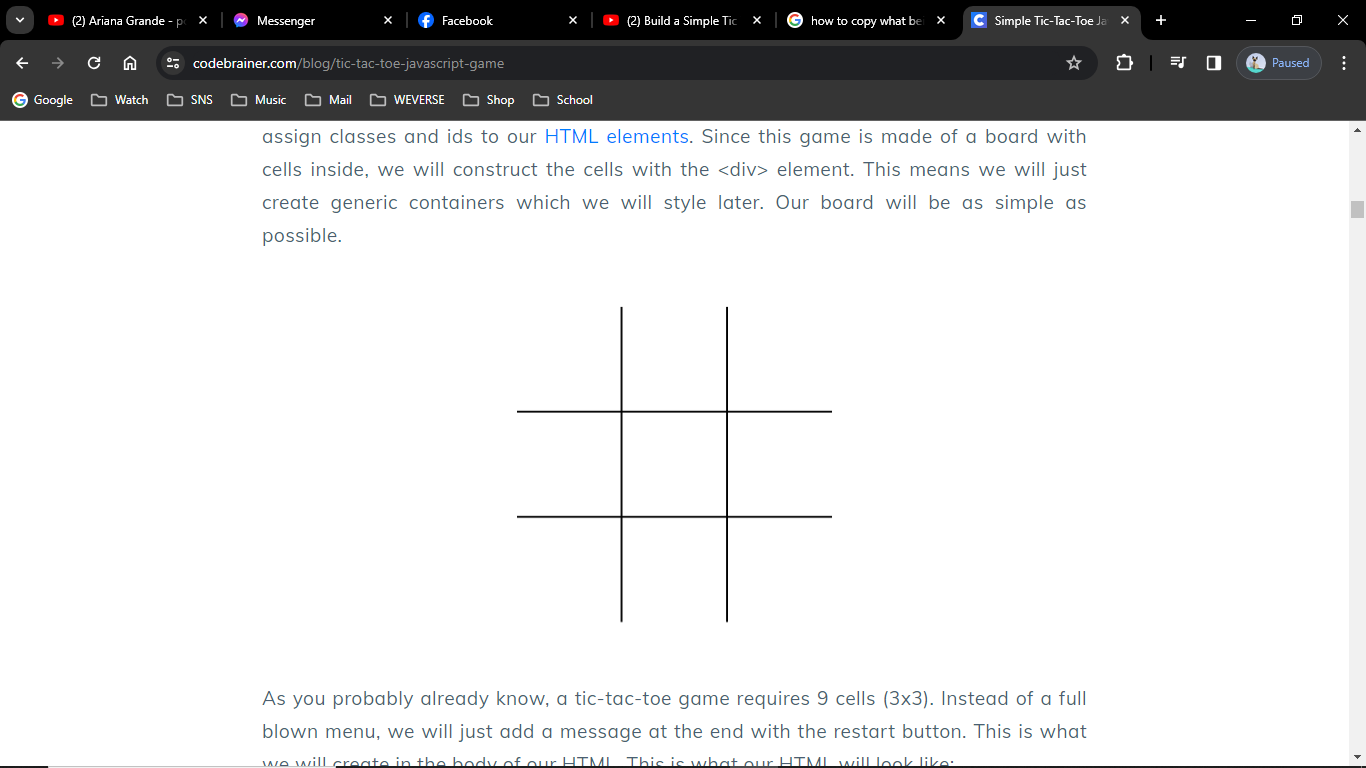
***Jean Ivory H. Flores***

Our group built a simple Tic Tac Toe game with only HTML, JavaScript and CSS using Visual Studio Code and GitHub. We will be using validation checks to implement the game features. The UI is designed using CSS so it is easy to create. This game is built on the front-end using simple logic and validation checks only. The ***‘’index.html’’*** contains the structure of the project***. ‘’style.css’’*** contains the styling of the application and ***‘’script.js’’*** contains the logic of the game.

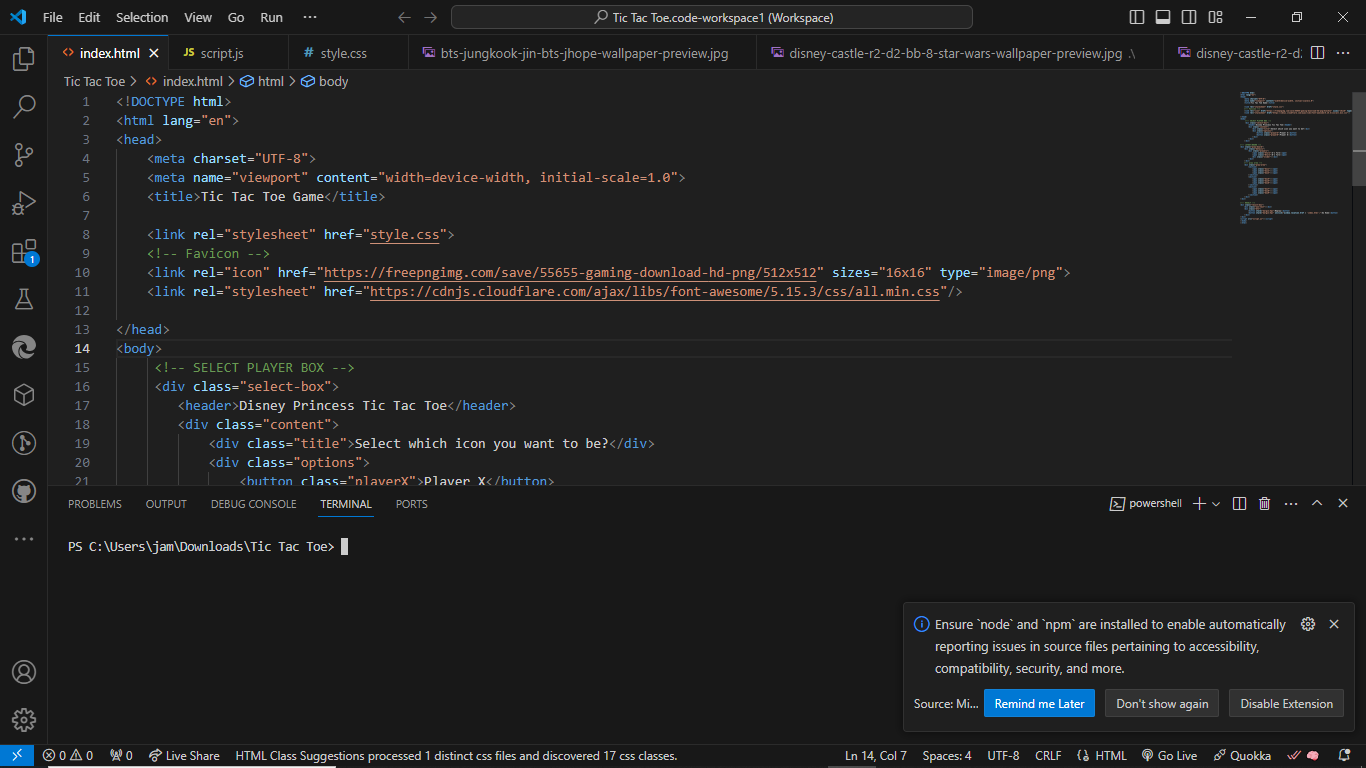
The tic-tac-toe game that we built will first allow users to select which player they want to be. Either they can be ‘’Player X’’ OR ‘’Player 0’’. Now, whenever it switches to another player which is in this case the computer, the tab will change into a nice-looking animation that fills the tabs for that player and allowing the computer to pick whatever spaces they can. The player who makes a straight 3-block chain wins the game. After knowing whether the user or the bot won, the user can go ahead and click the ‘’Go Home’’ button and it will reset itself.

To stay organized and optimized when making games, we separated different parts of code into different files. For our Tic-Tac-Toe JavaScript game, we used only three different files since it is a simple game. In the index.html we will assign classes to all separate constructors of our game. We will style our game in the style.css. Last but not least we will write our script in the script.js.

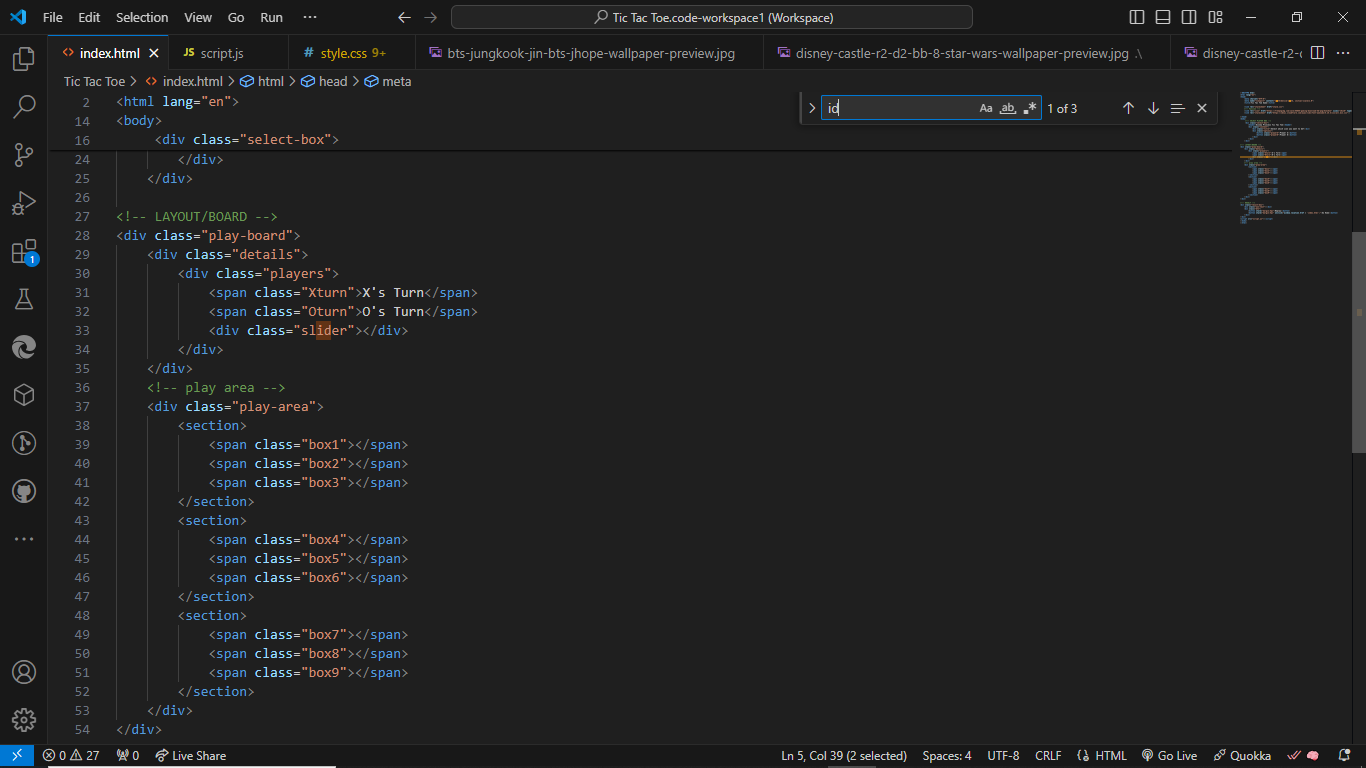
Starting off with HTML. This is the most simple and short part of our code. Here we assigned classes and ids to our HTML elements. Since this game is made of a board with cells inside, we will construct the cells with the <div> element. This means we just created a generic container which we will style later. Our board will be as simple as possible.



As you probably already know, a tic-tac-toe game requires 9 cells (3x3). Instead of a full-blown menu, we just added a message at the end with the replay button. This is what we will create in the body of our HTML. This is what our HTML will look like:



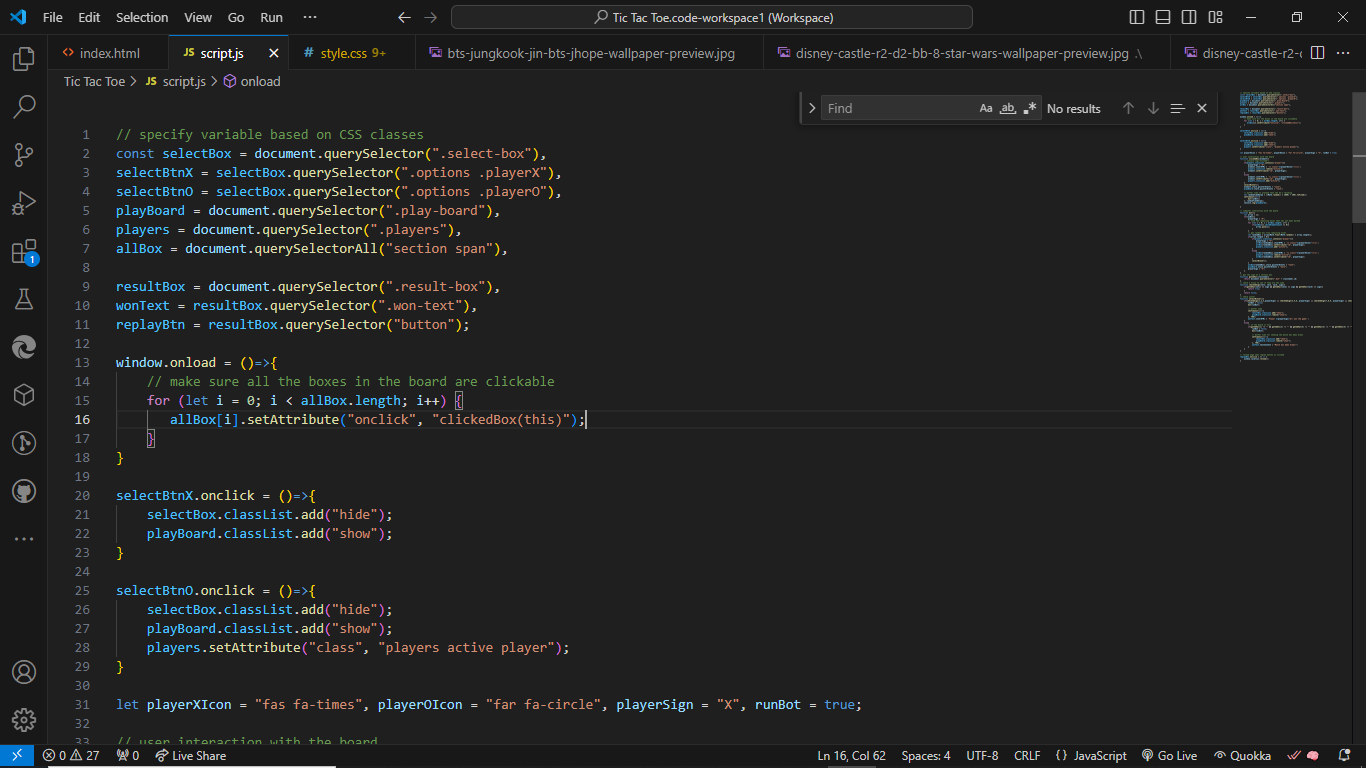
The head is kept simple. Here we just titled our web page and connected the index.html to the stylesheet and script.



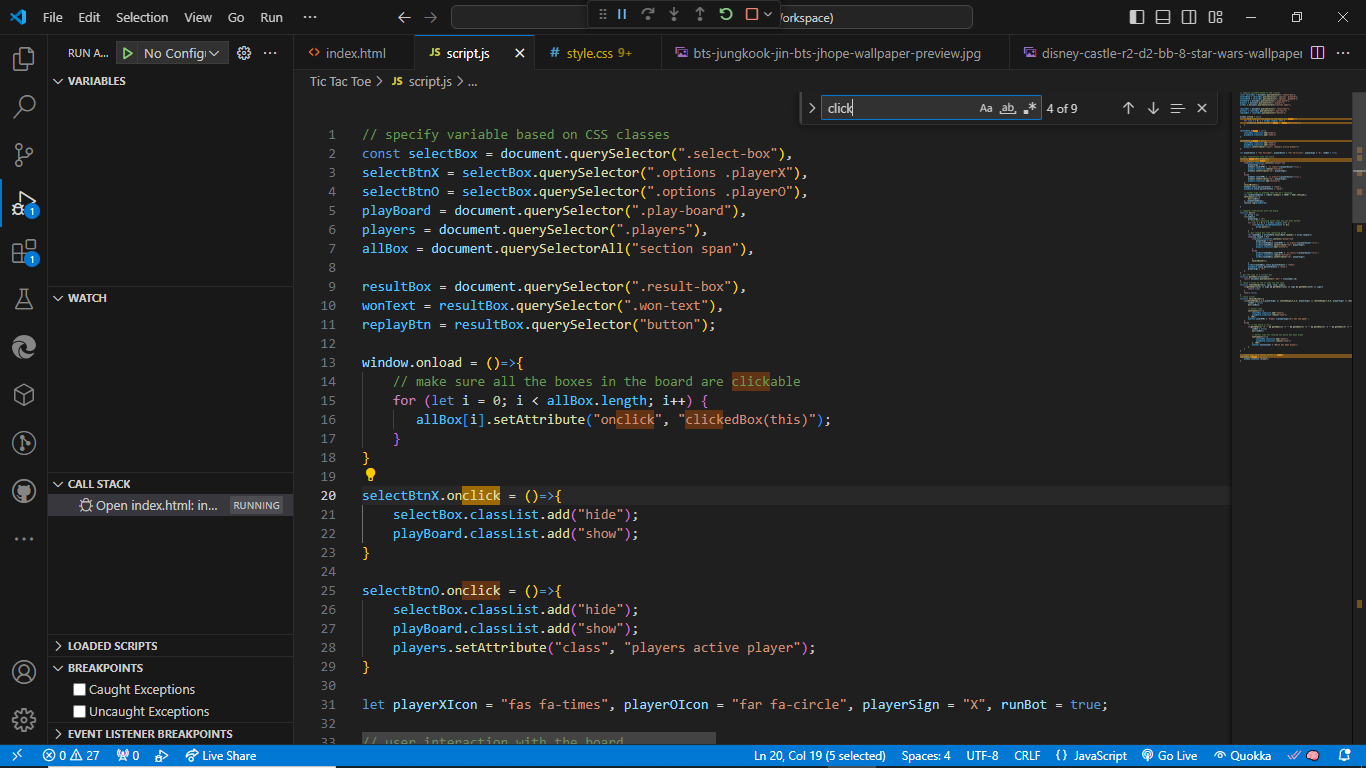
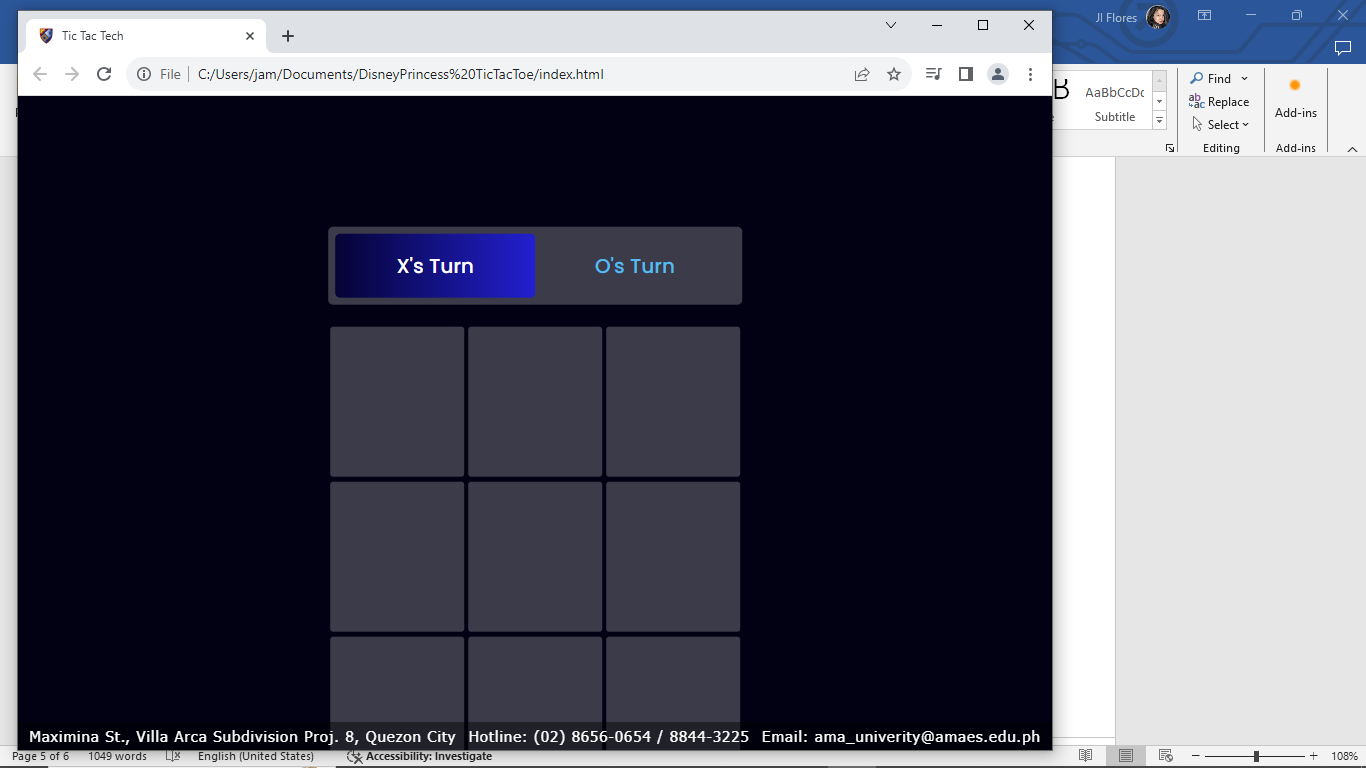
***In the body we created our board and named its class.***

Now we create code for the Tic-Tac-Toe JavaScript game. This part is what makes the game playful. In the first few lines of our script, we created a constant variable for our x and o characters. These combinations will help us determine if the game is over or not, by checking if any of the combinations match the current gameplay.

Here we used the id tags we assigned in the index.html to save the values of all the board elements, winning message and the restart button. For this we used the JavaScript method ***querySelector()*** which returns the first element within the document that matches the specified selector. We also used the squared brackets ([]) to target the attribute.

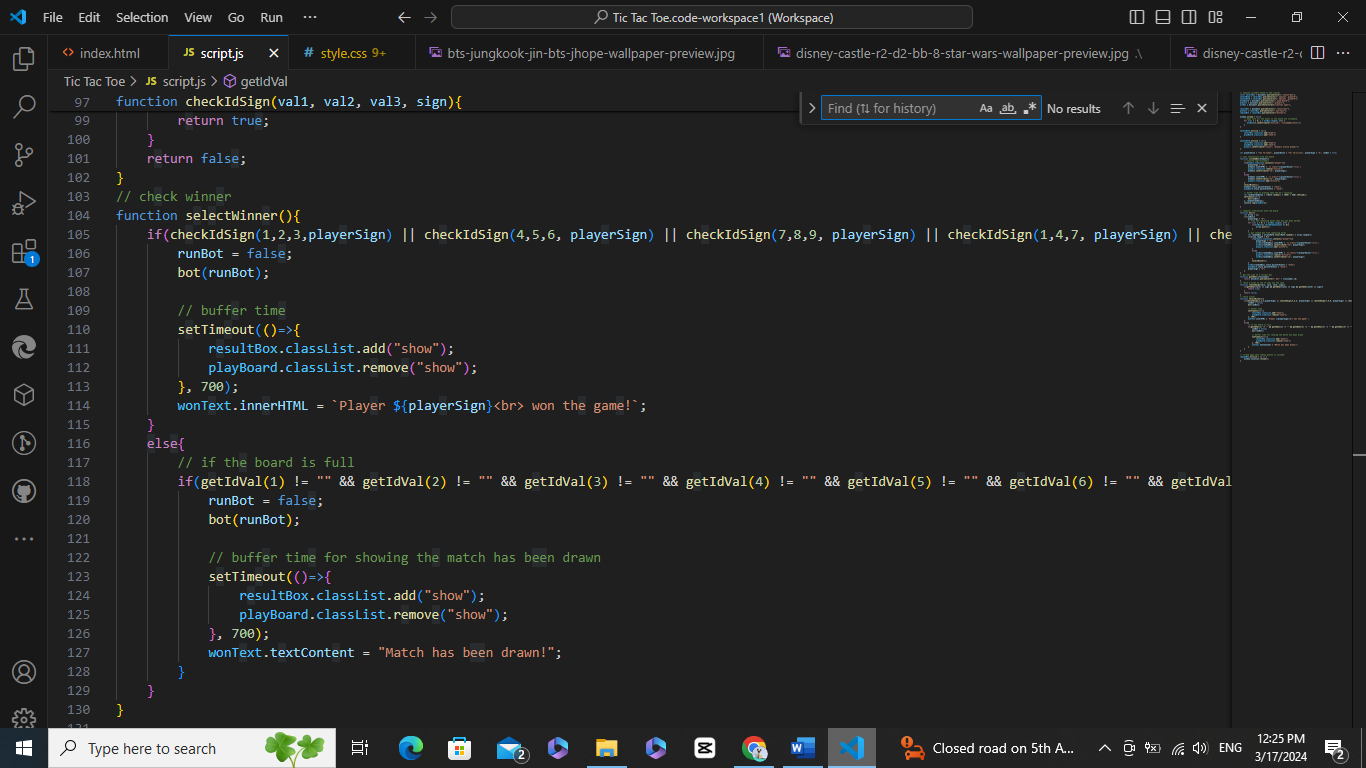


In the function ***selectBox*** we handle the mouse click events for the cells in the board. In short, the variable saves the character (X or O), whose turn it is at the moment. Another function is used in the if statement to check if someone has already won by comparing the winning combinations to the gameplay. This way it determines whether there is a draw or not.

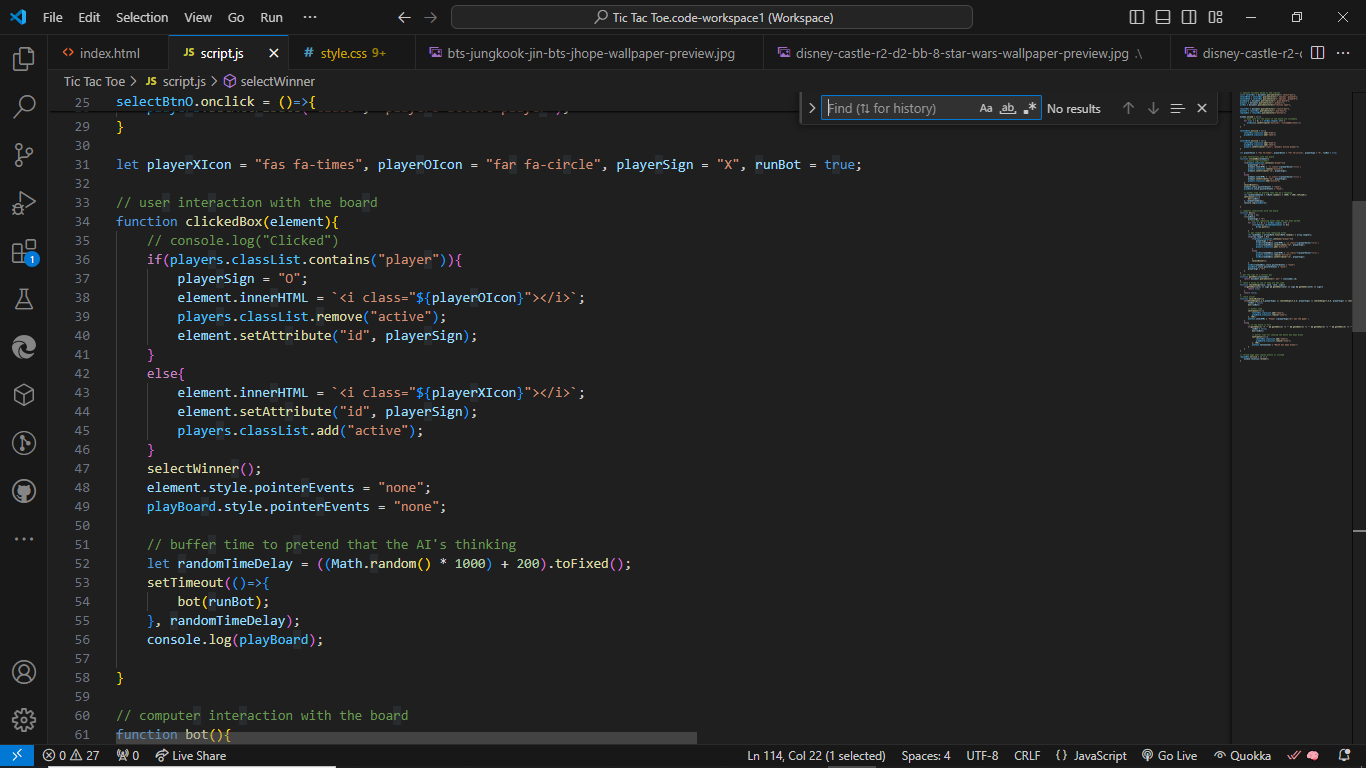
***When does the game of Tic-Tac-Toe end?***

The selectWinner() function is the function that ends the game. The function can either display a winner message which specifies which character won or a message that states there is no winner – it is a draw, depending on the outcome of the if statement.



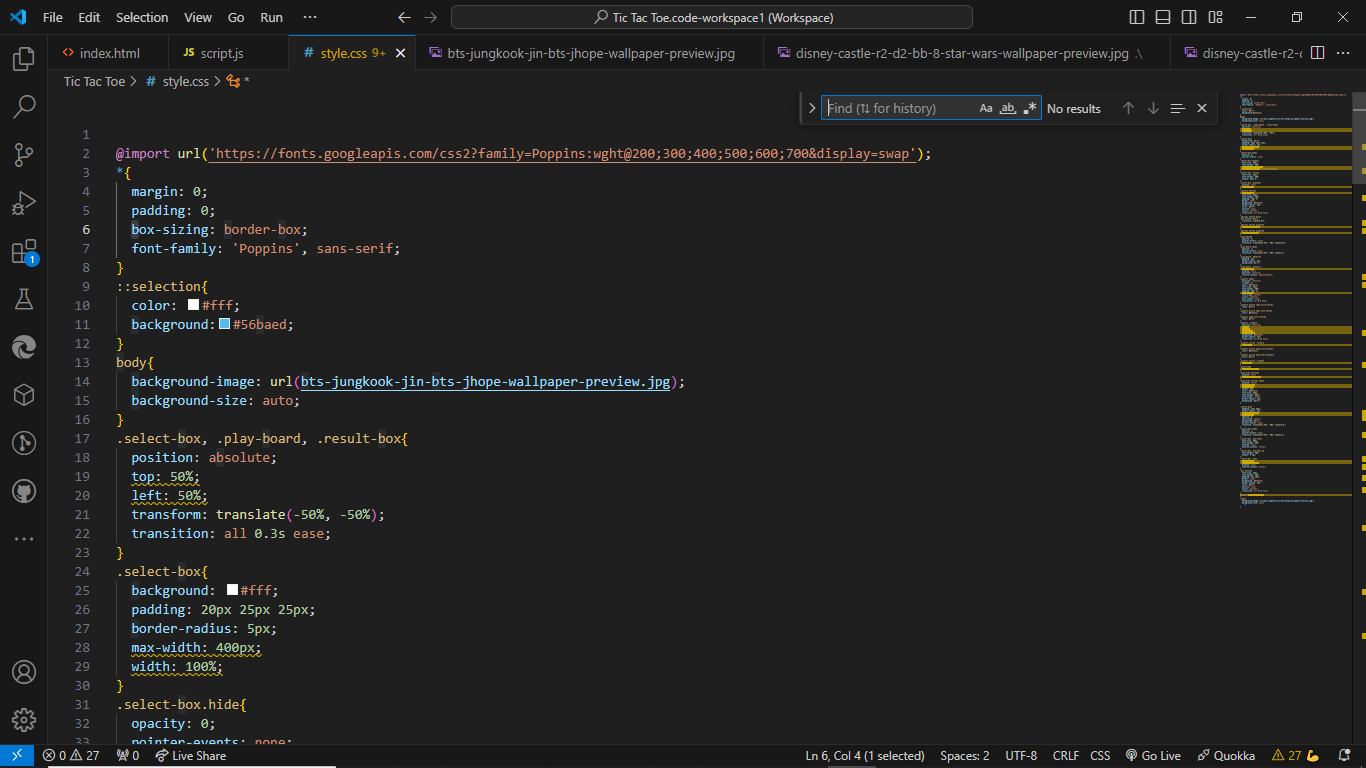
***Making Tic-Tac-Toe JavaScript game more interactive***

In the upcoming part of our JavaScript code, we will set the cursor hovering effect onto the board. This will make it easier for the player to aim at the cells. It also makes our game a bit more responsive.



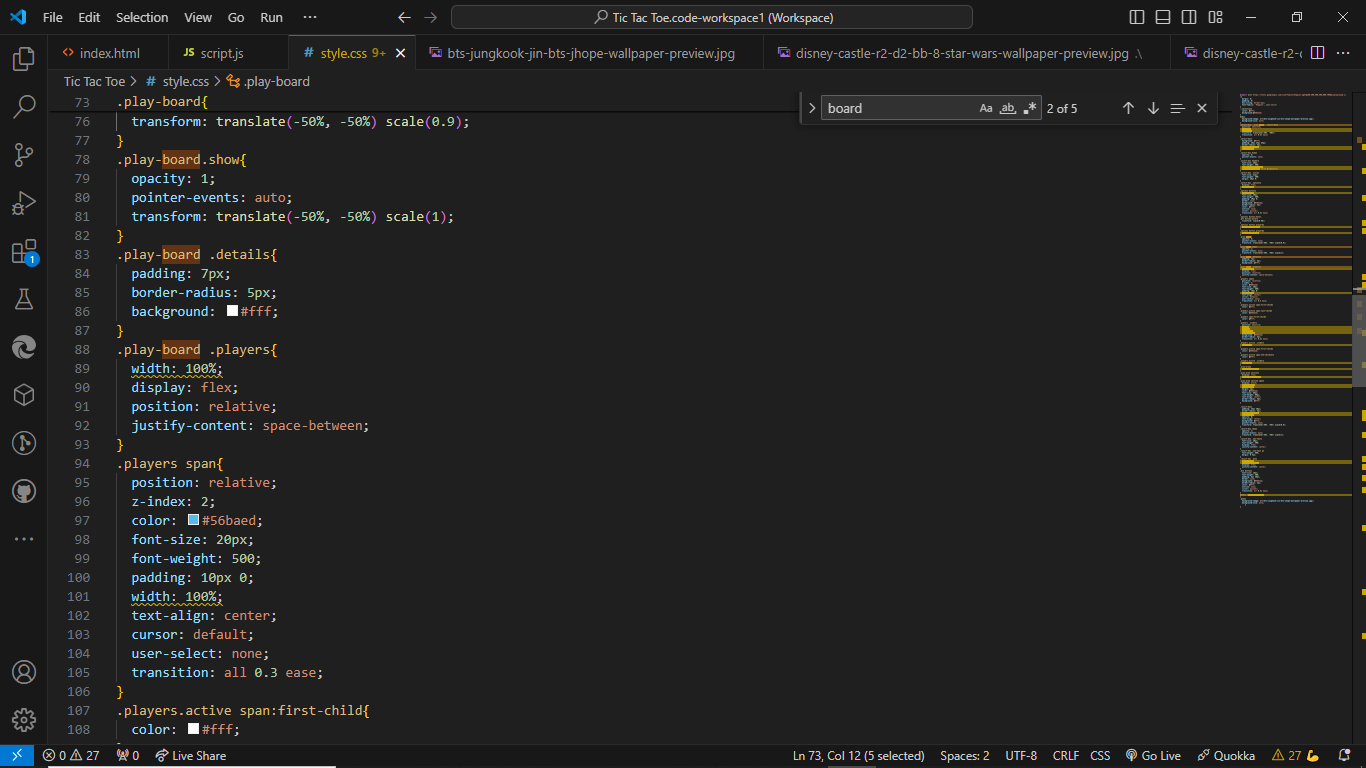
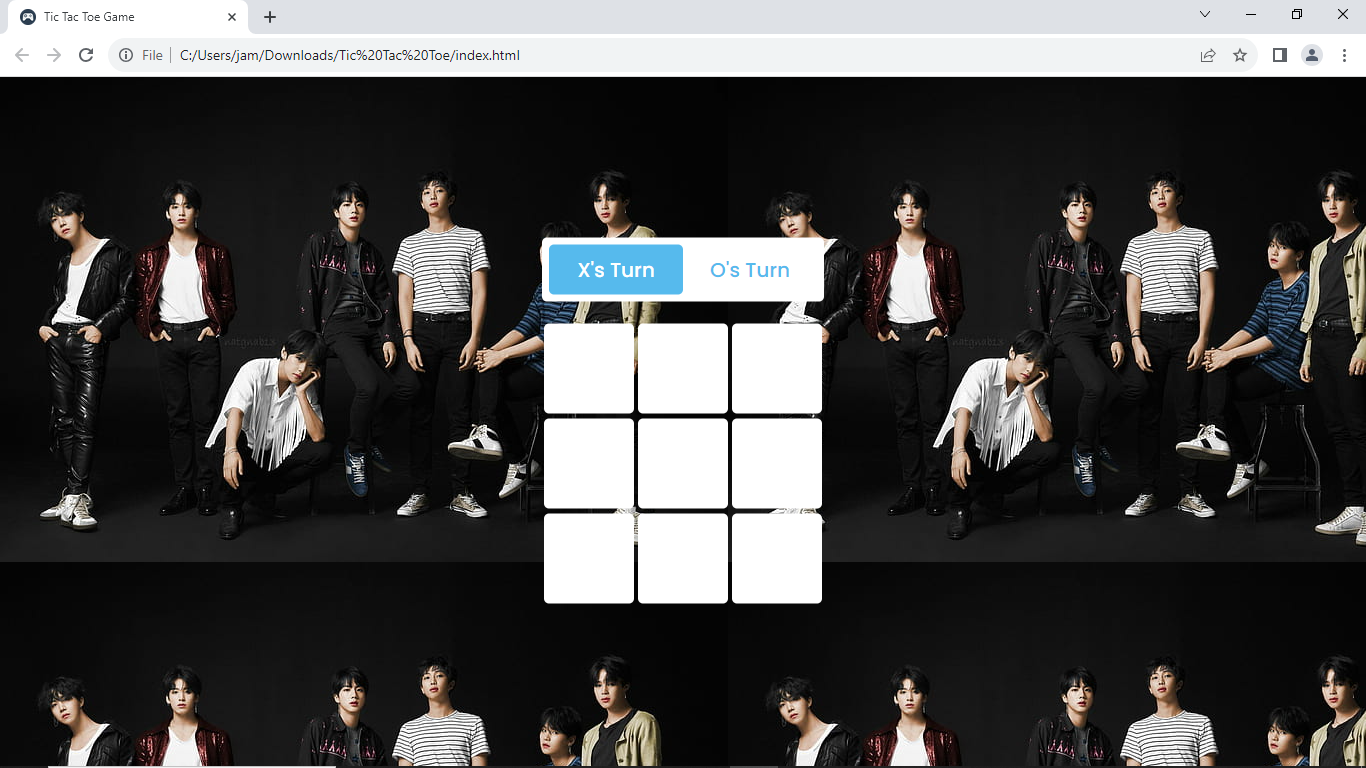
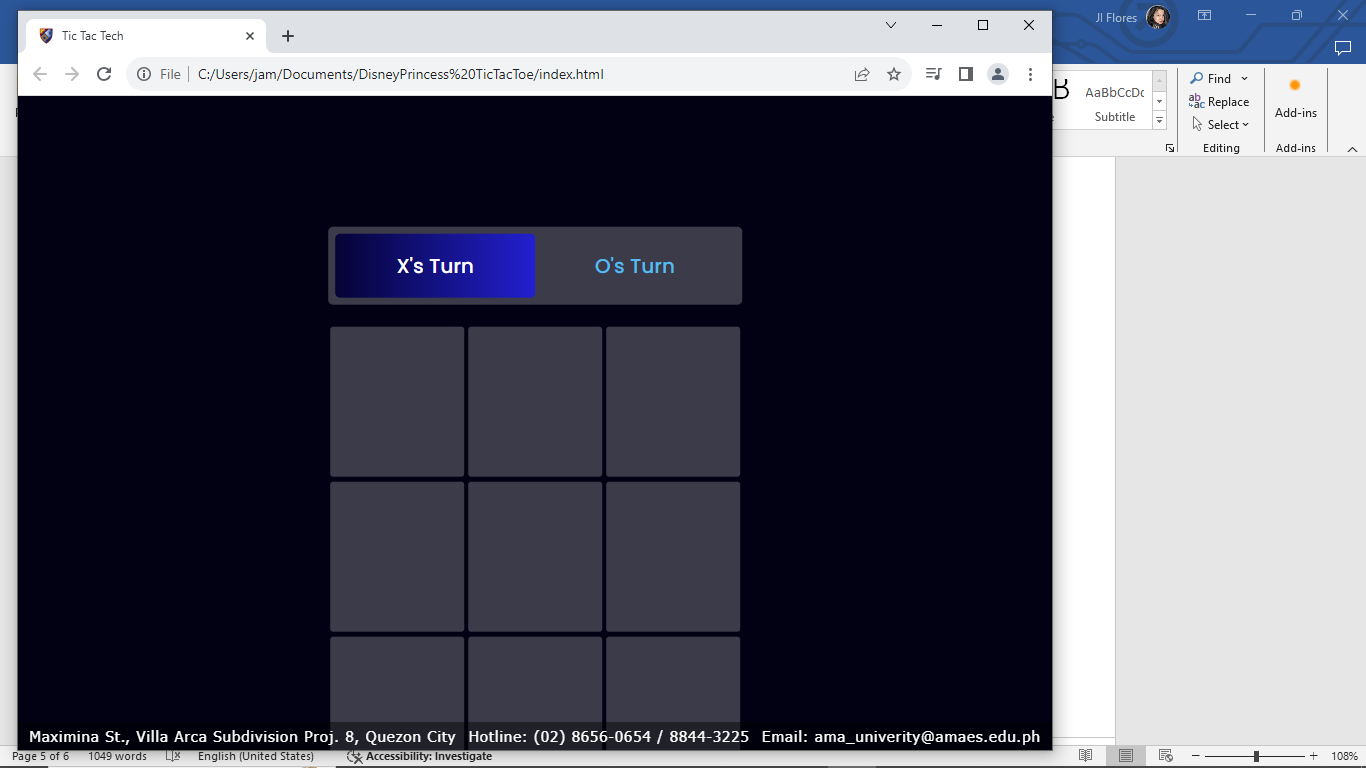
***CSS - styling our game***

The upcoming part of our code is where you can let your imagination and creativity loose. In the stylesheet we used our id tags and classes to personalize the visuals of our game. All the way from borders and line width to colors and text size. Since this part of the code can be completely personalized you can either write it from scratch, suitable for your own preferences or use our example. You can also just change the colors, size, fonts etc. if you will be using our example the color of X and O, and width of the X sign. See personalization below to see how you can make Tic-Tac-Toe more to your liking.

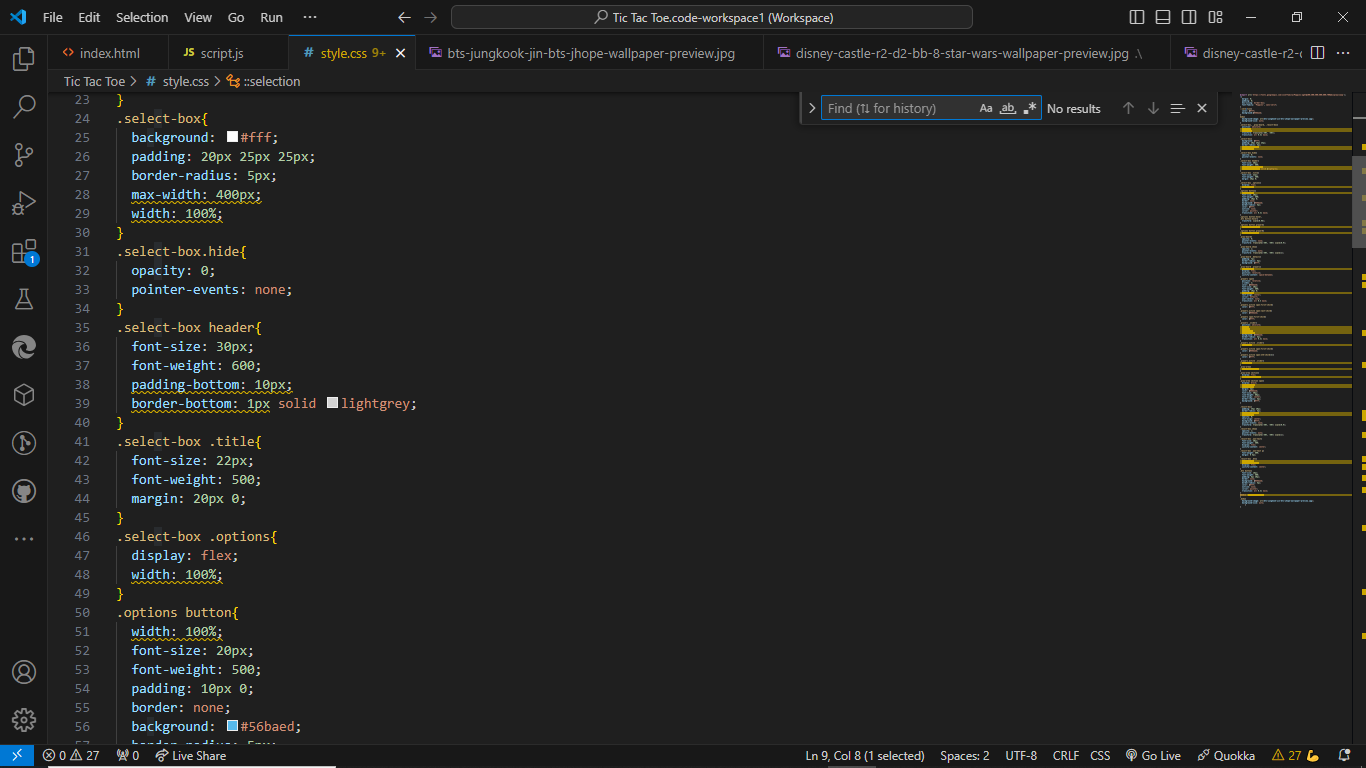
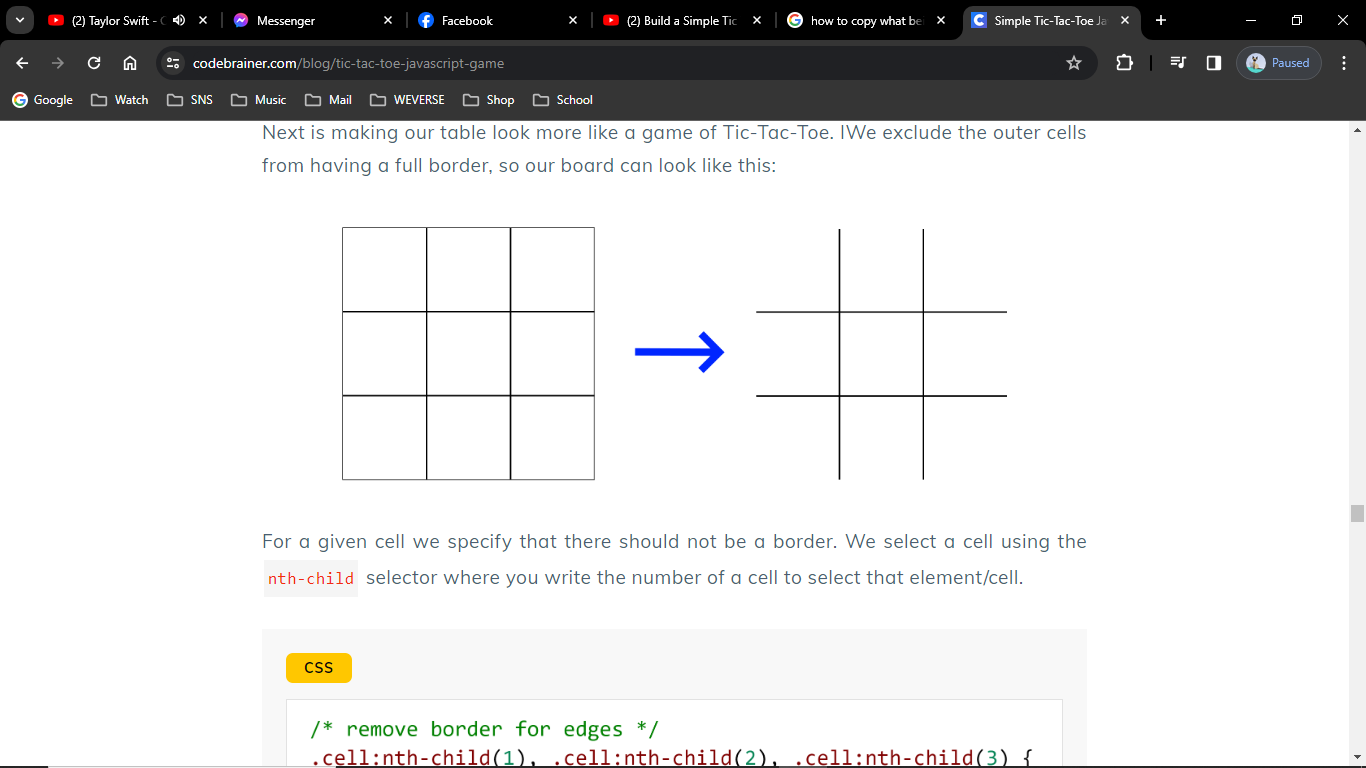


***CSS for board for Tic-Tac-Toe***

Starting with visualizing our HTML, the margin creates spaces between elements with defined borders, here it is used to make zero borders for the whole screen. After that we describe the element with our board class name, specifying its width, height, how and where it is displayed. grid-template-columns is a less known property so to keep it simple, it specifies the number and width/s of columns in a grid layout. This way we make our 9 cells for playing.

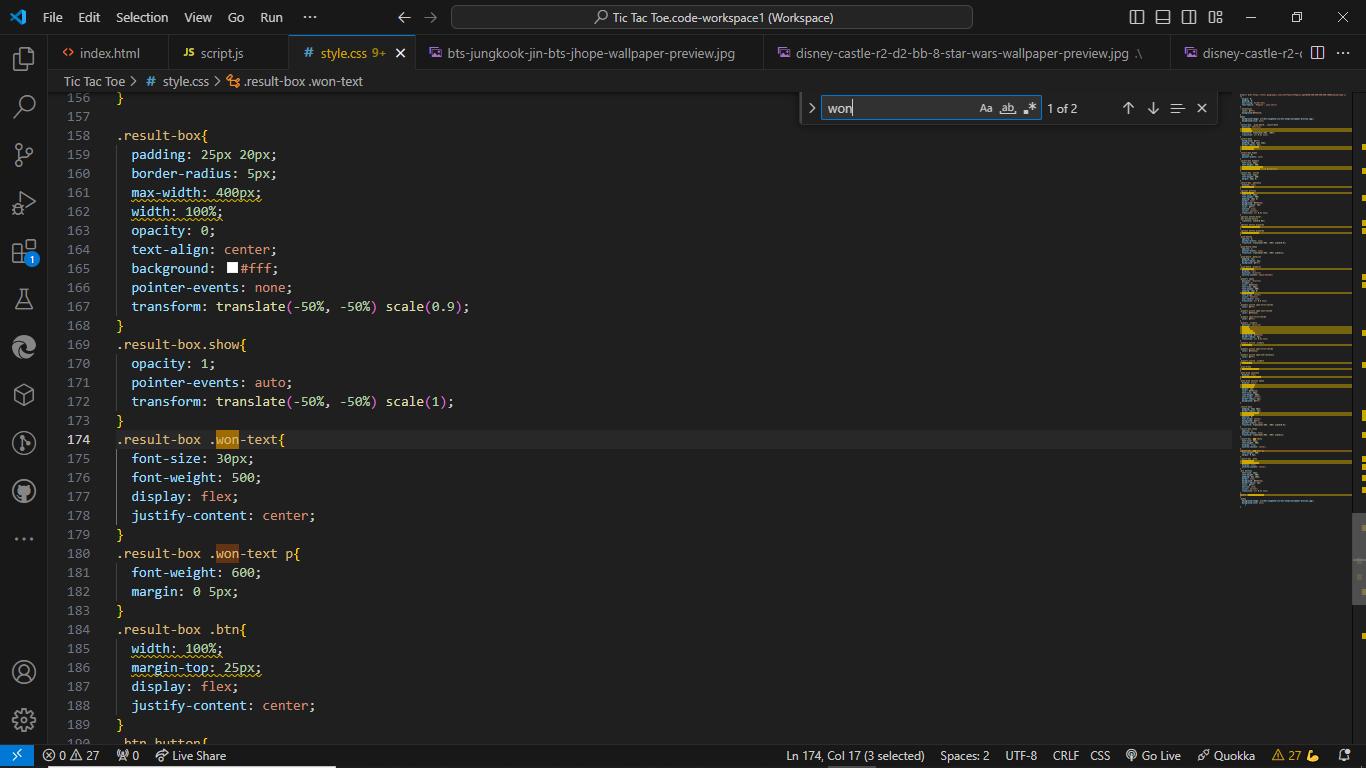
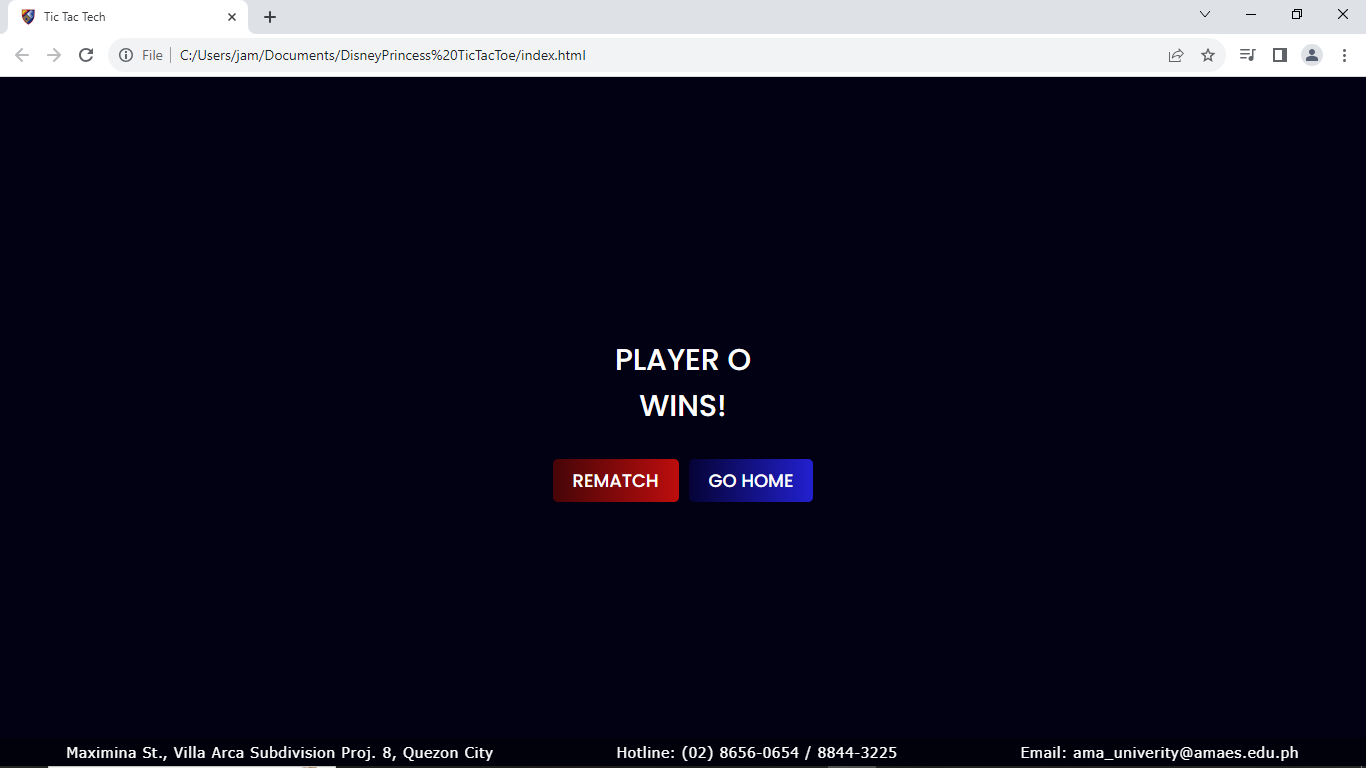
 

Next is making our table look more like a game of Tic-Tac-Toe. We exclude the outer cells from having a full border, so our board can look like this:



***Styling a winning message***

Styling for a winning message is more straightforward. Here we set fonts, and color for the text. Make the button a bit rounded. And add a hover effect at the end. Getting close to the finish line, we now play with our winning or draw message. Here we centered it on the screen and set the color to white.

***Personalizing the game by changing the CSS***

After following instructions on how to make a Tic-Tac-Toe JavaScript game, you can now personally choose colors of the game elements and make it yours. You can change color using variables --color and --color-set. If you want to change the color of the border, change the border color of the cell.