Simple Calculator Project Instructions

1. index.html

Create the basic HTML structure for a simple calculator using the following specifications:
- The page must include the following HTML elements: <html>, <head>, <title>, <link>, <body>, , <button>, <script>.
- Link the external CSS file: style.css.
- Link the external JavaScript file: script.js.
- Use a div with class "calculator" to contain all calculator elements.
- Inside it, create an <input> element with id="display" and disabled attribute.
- Below the input, create a grid of buttons inside a div with class "buttons". The layout should be:
  [7] [8] [9] [+]
  [4] [5] [6] [-]
  [1] [2] [3] [*]
  [0] [C] [=] [/]
- Use onclick handlers to call JS functions like appendNumber(), operator(), clearDisplay(), and calculateResult().

2. style.css

Add styles for the calculator layout:
- Reset all margins and paddings using universal selector (*).
- Center the calculator using flexbox on the body element.
- Style the calculator div with background, rounded corners, padding, and shadow.
- Style the input to have full width, right-aligned text, padding, and a subtle background.
- Use grid layout for the ".buttons" container with 4 equal columns and gaps.
- Style the buttons with uniform size, rounded corners, and hover/active effects.

3. script.js

Implement the calculator logic in JavaScript using the following functions:
- Variables: currentInput, currentOperator, firstOperand, and a reference to the display element.
- appendNumber(number): Append the digit to the input and update display.
- operator(op): If firstOperand is null, assign currentInput to it and store the operator.

Otherwise, call calculateResult() and update operator.
- calculateResult(): Perform the arithmetic operation based on the currentOperator. Show result on display.
- clearDisplay(): Reset all variables and clear the display.
- Use document.getElementById('display') to access the input element.
- No need to export functions or attach them to window – define them globally so they work with inline onclick handlers.

Detailed CSS Styling Guide for Calculator

This section describes the CSS styling that should be implemented in the "style.css" file for this project.
Each section explains the purpose of the styles and how they contribute to the overall user interface.

1. Universal Selector (*)

Applies basic resets to all elements:
- Removes default margin and padding from all elements.
- Sets box-sizing to border-box to ensure padding and border are included in width/height.

2. Body Styling

Centers the calculator in the viewport:
- Uses Flexbox to center content both vertically and horizontally.
- Sets height to 100vh to make it occupy the full viewport height.
- Sets a light background color as “#f4f4f4” and font as “Arial, sans-serif”.

3. Calculator Container (.calculator)

Styles the main calculator box:
- Fixed width of 320px for consistent size.
- White background as “#fff”, rounded corners as “10px”, and padding as “20px” for clean layout.
- Subtle box shadow as “0 4px 10px rgba(0, 0, 0, 0.1)” for a lifted appearance.

4. Input Field Styling (input)

Styles the display area of the calculator:
- Full width with fixed height as “50px”.
- Right-aligned text and large font size as “24px” for readability and margin bottom as “20px”.
- Padding as “10px”, border as “1px solid #ccc” and border-radius as “5px” for improved UX.
- Light background as “#f9f9f9” to differentiate it from buttons.

5. Buttons Grid (.buttons)

Defines the grid layout for the calculator buttons:
- Uses CSS Grid with 4 equal columns.
- Adds consistent as “10px” spacing between buttons using gap.

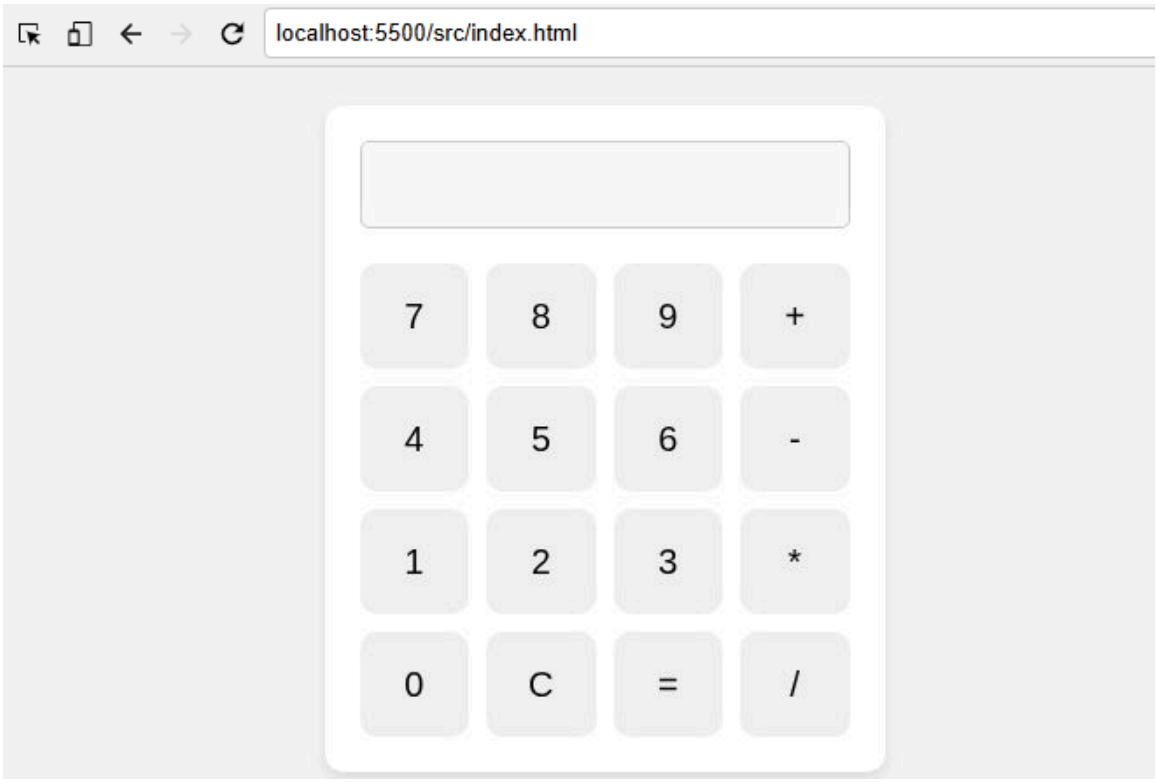
6. Button Styling (button)

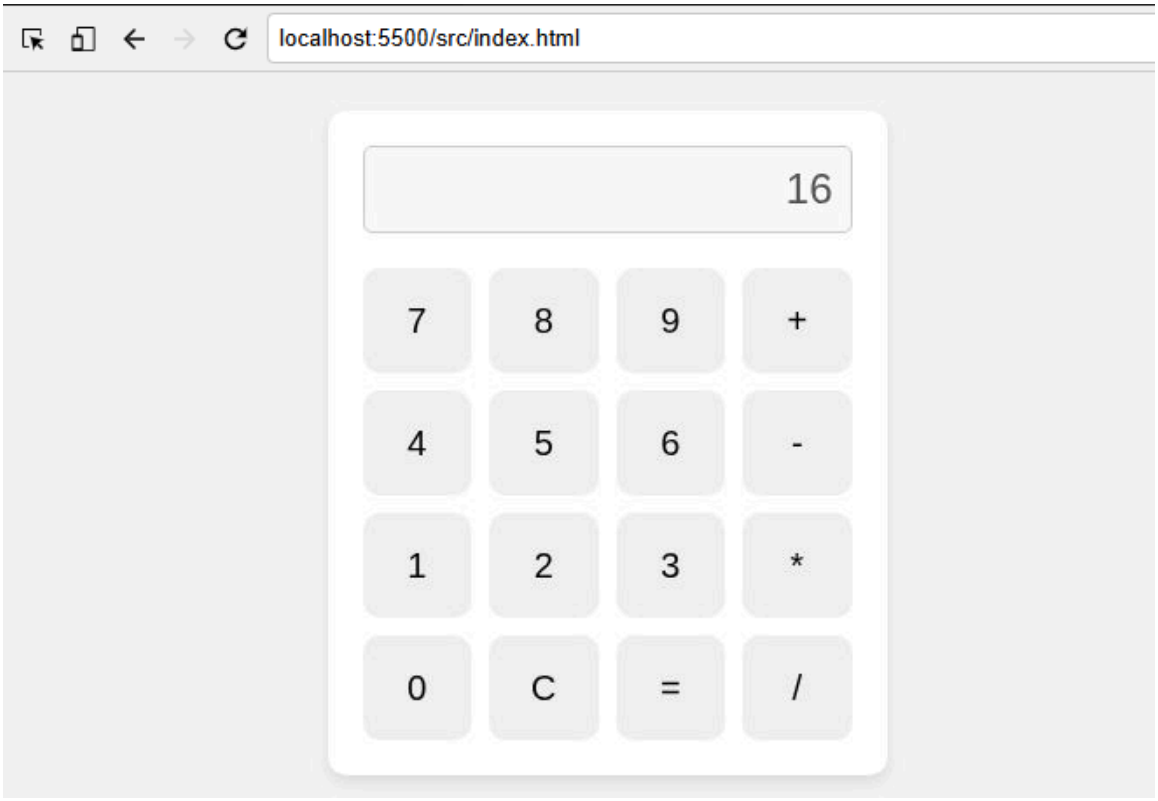
Styles all calculator buttons:
- Uniform height as “60px” and font size as “20px”.
- Light background as “#f0f0f0” and no borders.
- Rounded corners as “10px” and pointer cursor.
- Smooth transition as “background-color 0.3s” effect for hover state.

7. Button Hover and Active States

Enhances interactivity with visual feedback:
- Hover: slightly darker background as “#ddd”.
- Active (pressed): even darker background as “#ccc”.

ScreenShots:




localhost:5500/src/index.html
16
7
8
9
+
4
5
6
-
1
2
3
*
0
C
=
/

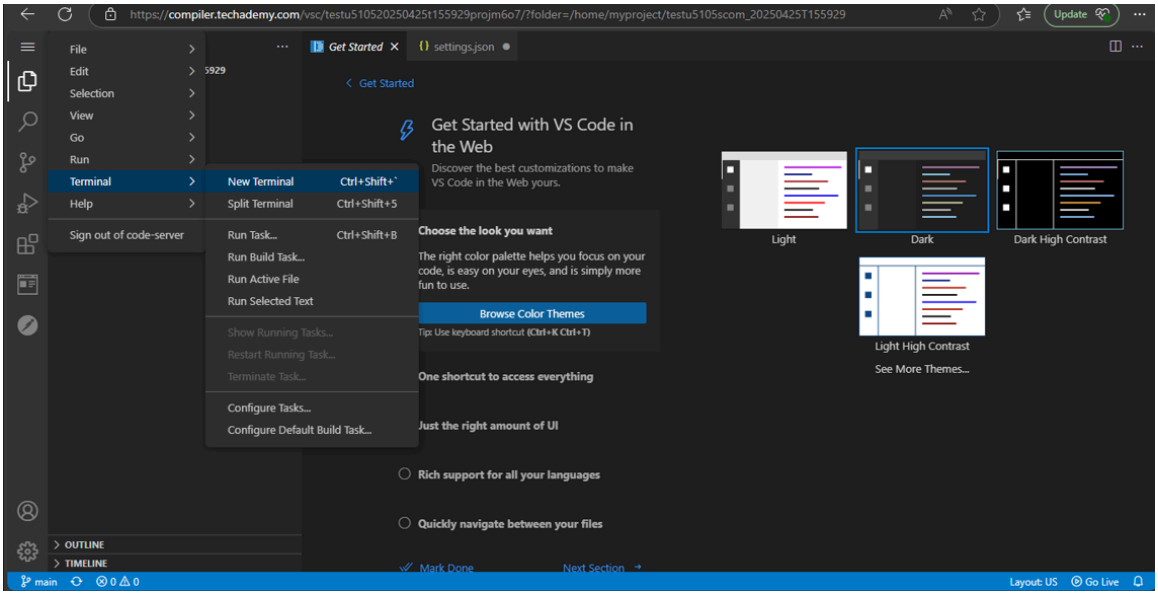
Assessment Guidelines

Step 1:

- Once the VS Code interface loads in the browser, wait until you see the workspace and left sidebar.
- To open the command terminal the test takers, need to go to

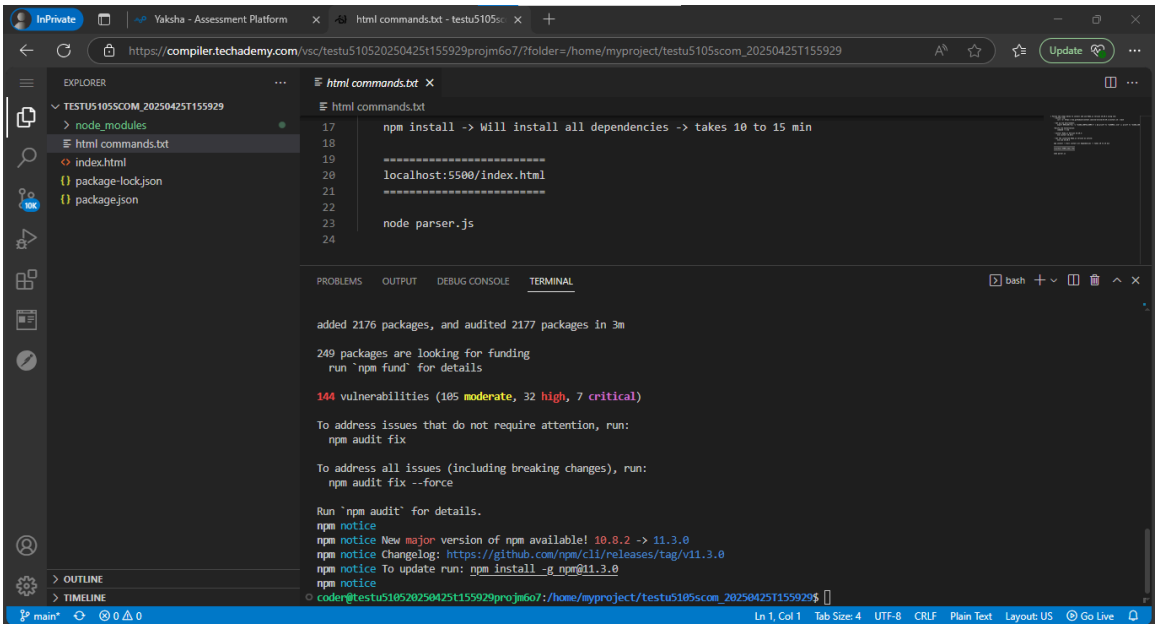
  Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.

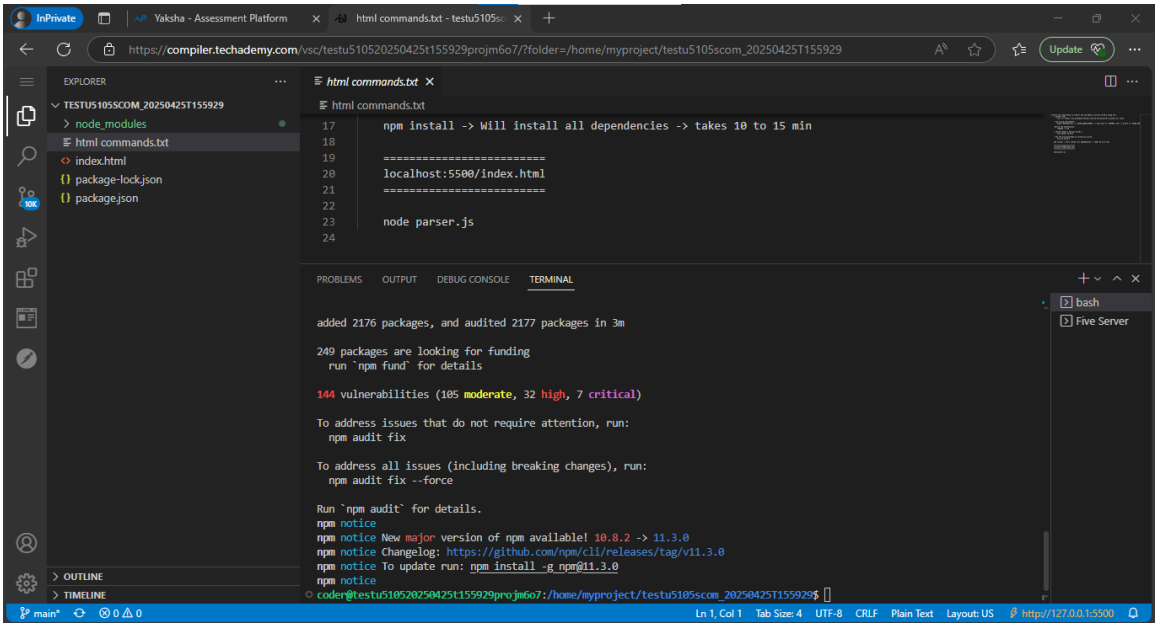
Now in the terminal you need to install all dependencies using the **"npm install"** command.



Step 2:

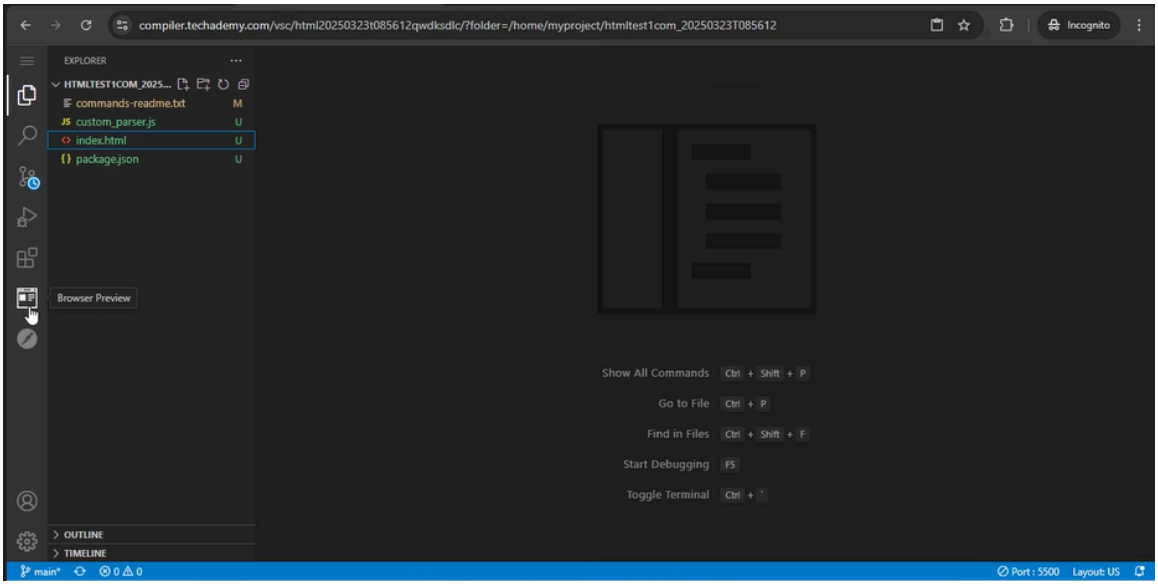
- Once installation completes, go to the **bottom right corner** of the VS Code screen.
- Click the **"Go Live"** button – This will start a **live server**, The server will run at port `5500` (e.g., `http://localhost:5500/`)





Step 3: Preview Output in Browser

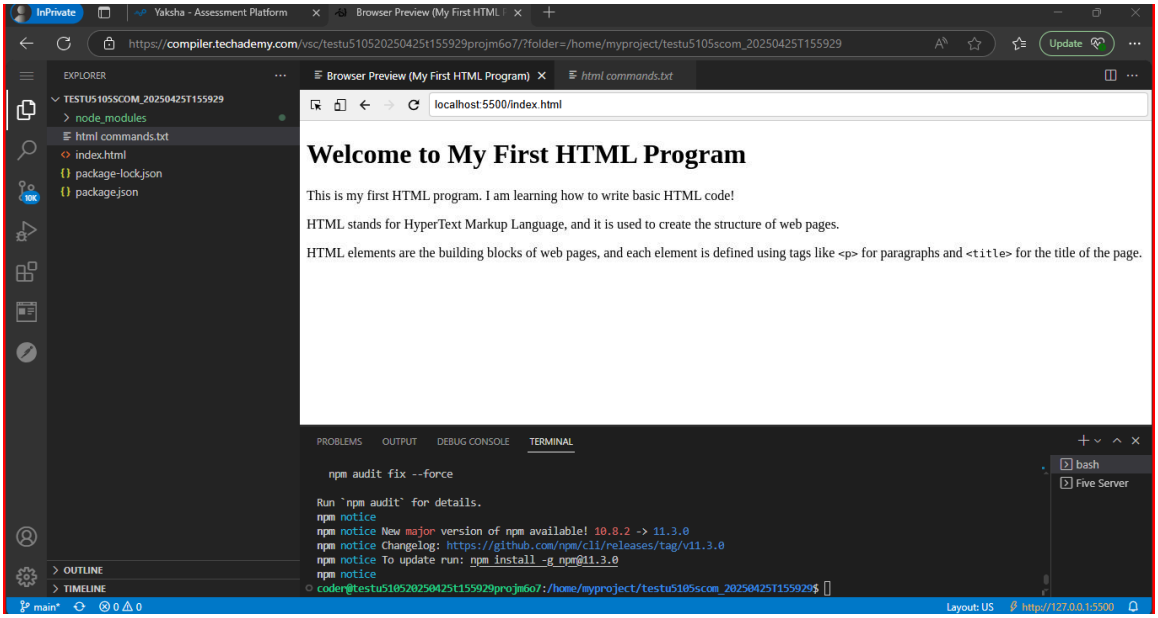
- This is a **web-based application**, so to view it in a browser, use the **internal browser inside the workspace**.
- Click on the **second last icon on the left panel** (the one labeled **"Browser Preview"**). This will open a tab within VS Code where you can **launch and view your application**.
- Note: The application **will not open in your system's local browser** — it must be viewed using the internal browser.



In the **Browser Preview tab**, type the following URL in the address bar and press **Enter**:

Your file is being served on: `localhost:5500/src/index.html`

This will load your HTML file and display the output of your web page **inside the internal browser**.



Step 4:

- Go back to the **terminal** and type the following command, then press **Enter**:

  node src/test/custom-grader.js

- This command will **execute the validation script** and display the test results for your HTML file in the terminal.

Mandatory Assessment Guidelines:

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to**
   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. **This editor Auto Saves the code.**
4. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
5. **This is a web-based application, to run the application on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
   Note: The application will not run in the local browser
6. **You can follow series of command to setup environment once you are in your project-name folder:**
   a. **npm install -> Will install all dependencies -> takes 10 to 15 min.**
   b. **node src/test/custom-grader.js -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min.**
7. **Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on "Submit Assessment" after you are done with code.**