**App.component.html part of calculator**

<div class="calculator">

<input type="text" [(ngModel)]="display" readonly>

<div class="button-panel">

<div class="button-row">

<button (click)="clear()">C</button>

<button (click)="appendToDisplay('7')">7</button>

<button (click)="appendToDisplay('8')">8</button>

<button (click)="appendToDisplay('9')">9</button>

<button (click)="calculate()">=</button>

</div>

<div class="button-row">

<button (click)="appendToDisplay('4')">4</button>

<button (click)="appendToDisplay('5')">5</button>

<button (click)="appendToDisplay('6')">6</button>

<button (click)="appendToDisplay('+')">+</button>

<button (click)="appendToDisplay('-')">-</button>

</div>

<div class="button-row">

<button (click)="appendToDisplay('1')">1</button>

<button (click)="appendToDisplay('2')">2</button>

<button (click)="appendToDisplay('3')">3</button>

<button (click)="appendToDisplay('\*')">\*</button>

<button (click)="appendToDisplay('/')">/</button>

</div>

<div class="button-row">

<button (click)="appendToDisplay('0')">0</button>

<button (click)="appendToDisplay('.')">.</button>

<button (click)="squareRoot()">√</button>

<button (click)="appendToDisplay('^')">^</button>

<button (click)="clearDisplay()">CE</button>

</div>

</div>

</div>

**Alternate html file**

<div class="calculator">

<input type="text" [(ngModel)]="display" readonly>

<div class="button-panel">

<div class="button-row">

<button (click)="clear()" class="clear-button">C</button>

<button (click)="appendToDisplay('7')" class="number-button">7</button>

<button (click)="appendToDisplay('8')" class="number-button">8</button>

<button (click)="appendToDisplay('9')" class="number-button">9</button>

<button (click)="calculate()" class="equal-button">=</button>

</div>

<div class="button-row">

<button (click)="appendToDisplay('4')" class="number-button">4</button>

<button (click)="appendToDisplay('5')" class="number-button">5</button>

<button (click)="appendToDisplay('6')" class="number-button">6</button>

<button (click)="appendToDisplay('+')" class="operator-button">+</button>

<button (click)="appendToDisplay('-')" class="operator-button">-</button>

</div>

<div class="button-row">

<button (click)="appendToDisplay('1')" class="number-button">1</button>

<button (click)="appendToDisplay('2')" class="number-button">2</button>

<button (click)="appendToDisplay('3')" class="number-button">3</button>

<button (click)="appendToDisplay('\*')" class="operator-button">\*</button>

<button (click)="appendToDisplay('/')" class="operator-button">/</button>

</div>

<div class="button-row">

<button (click)="appendToDisplay('0')" class="number-button">0</button>

<button (click)="appendToDisplay('.')" class="number-button">.</button>

<button (click)="squareRoot()" class="function-button">√</button>

<button (click)="appendToDisplay('^')" class="operator-button">^</button>

<button (click)="clearDisplay()" class="clear-button">CE</button>

</div>

</div>

</div>

**.css file**

.calculator {

width: 300px;

margin: 0 auto;

padding: 20px;

background-color: #f1f1f1;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

}

input {

width: 100%;

padding: 10px;

font-size: 20px;

text-align: right;

border: none;

border-radius: 5px;

margin-bottom: 10px;

}

.button-panel {

display: flex;

flex-direction: column;

}

.button-row {

display: flex;

justify-content: space-between;

margin-bottom: 10px;

}

button {

width: 60px;

height: 60px;

border: none;

border-radius: 50%;

font-size: 20px;

cursor: pointer;

}

.number-button {

background-color: #e0e0e0;

color: #333;

}

.operator-button {

background-color: #ff8c00;

color: #fff;

}

.function-button {

background-color: #4caf50;

color: #fff;

}

.clear-button {

background-color: #f44336;

color: #fff;

}

.equal-button {

background-color: #2196F3;

color: #fff;

}

Connect web api to angular app

Create a service in Angular to communicate with the .NET Core Web API. Generate the service using Angular CLI:

ng generate service calculator

2. In the generated service (e.g., **calculator.service.ts**), define methods that make HTTP requests to your API endpoints. For example:

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class CalculatorService {

private apiUrl = 'https://your-api-url'; // Replace with your API URL

constructor(private http: HttpClient) { }

add(num1: number, num2: number): Observable<number> {

return this.http.get<number>(`${this.apiUrl}/add?num1=${num1}&num2=${num2}`);

}

// Implement other calculator operations (subtract, multiply, divide, etc.)

}

**Step 4: Create a Component**

Create a component in your Angular app to design the calculator user interface and handle user interactions. For example:

ng generate component calculator

**Step 5: Implement the Calculator Component**

In your calculator component (e.g., **calculator.component.ts**), use the **CalculatorService** to perform calculator operations and update the UI:

import { Component } from '@angular/core';

import { CalculatorService } from './calculator.service';

@Component({

selector: 'app-calculator',

templateUrl: './calculator.component.html',

styleUrls: ['./calculator.component.css']

})

export class CalculatorComponent {

num1: number;

num2: number;

result: number;

display: string = ‘ ‘; // Initialize the display with an empty string

constructor(private calculatorService: CalculatorService) { }

clear() // this is the clear function

{

this.display = ''; // Clear the display

}

add() {

this.calculatorService.add(this.num1, this.num2)

.subscribe(result => {

this.result = result;

});

}

// Implement other calculator operations (subtract, multiply, divide, etc.)

}

**Step 7: Run Your Angular App**

Start your Angular app using:

ng serve

To implement the **showHistory** function that displays the calculation history, you can create a modal dialog using a third-party library like Angular Material or a simple HTML and CSS-based modal. Below is a simplified example of how to create a basic history display using HTML and CSS:

**calculator.component.html:**

<div class="calculator">

<!-- Calculator input and buttons (previous code) -->

<div class="button-row">

<button (click)="clear()" class="clear-button">C</button>

<button (click)="showHistory()" class="history-button">History</button>

<button (click)="calculate()" class="equal-button">=</button>

</div>

</div>

<!-- History modal -->

<div id="historyModal" class="modal">

<div class="modal-content">

<span class="close" (click)="closeHistory()">&times;</span>

<h2>Calculation History</h2>

<ul>

<li \*ngFor="let calculation of history">{{ calculation }}</li>

</ul>

</div>

</div>

**calculator.component.ts:**

import { Component, ElementRef } from '@angular/core';

@Component({

selector: 'app-calculator',

templateUrl: './calculator.component.html',

styleUrls: ['./calculator.component.css']

})

export class CalculatorComponent {

// Properties and methods as before...

constructor(private elementRef: ElementRef) {}

showHistory() {

// Display the history modal

const modal = this.elementRef.nativeElement.querySelector('#historyModal');

modal.style.display = 'block';

}

closeHistory() {

// Close the history modal

const modal = this.elementRef.nativeElement.querySelector('#historyModal');

modal.style.display = 'none';

}

// Other methods for calculator operations...

}

**calculator.component.css:**

/\* Style for the history modal \*/

.modal {

display: none;

position: fixed;

z-index: 1;

left: 0;

top: 0;

width: 100%;

height: 100%;

overflow: auto;

background-color: rgba(0, 0, 0, 0.7);

}

.modal-content {

background-color: #f1f1f1;

margin: 15% auto;

padding: 20px;

border: 1px solid #888;

width: 50%;

}

.close {

color: #aaa;

float: right;

font-size: 28px;

font-weight: bold;

cursor: pointer;

}

.close:hover {

color: black;

}