**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра інтелектуальних технологій**

**Модульна контрольна робота №1**

З дисципліни «Крос-платформне програмування»

**Варіант №1**

Виконала студентка групи КН-32

Гнатюк Анна Михайлівна

Перевірила:

Федусенко Олена Володимирівна

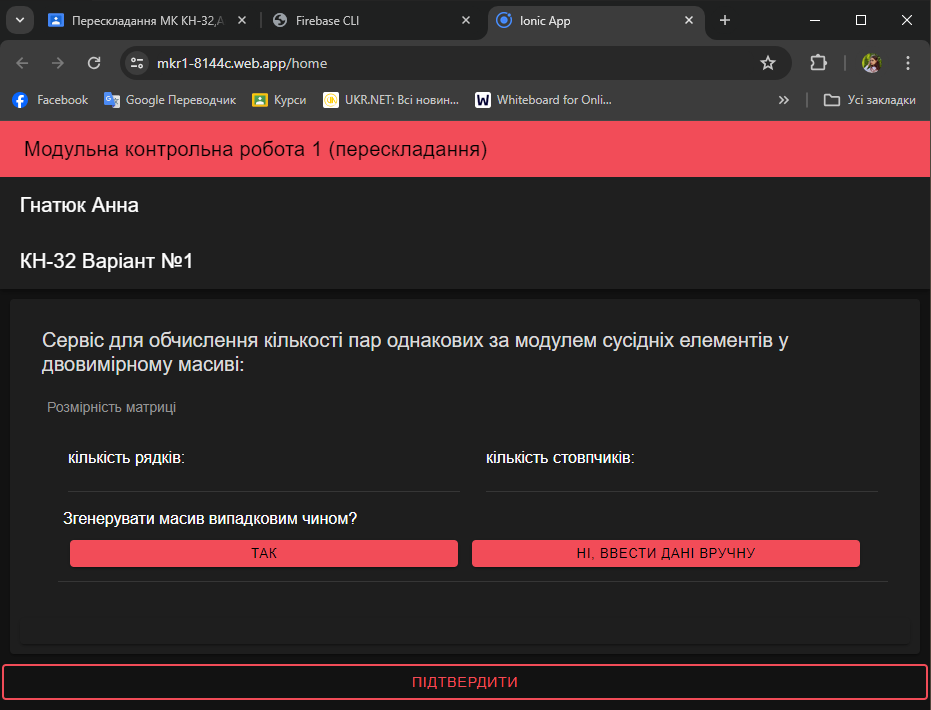
**Київ – 2024**

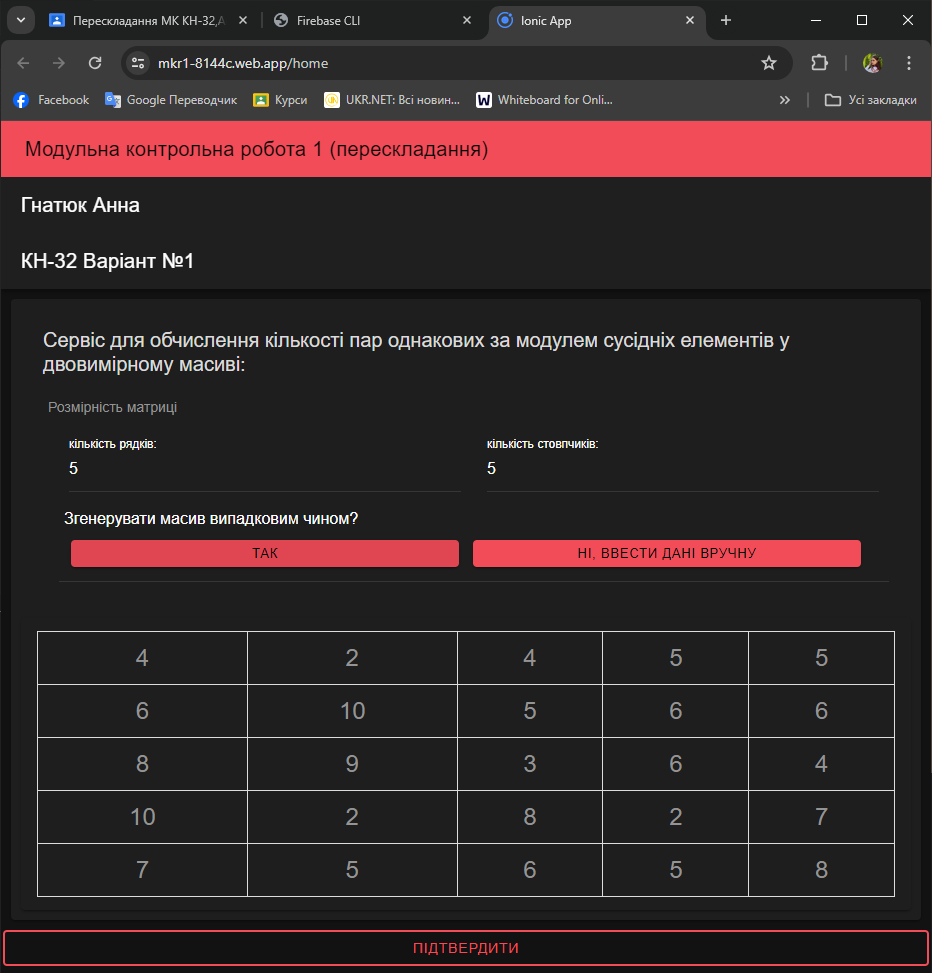
**Завдання:**

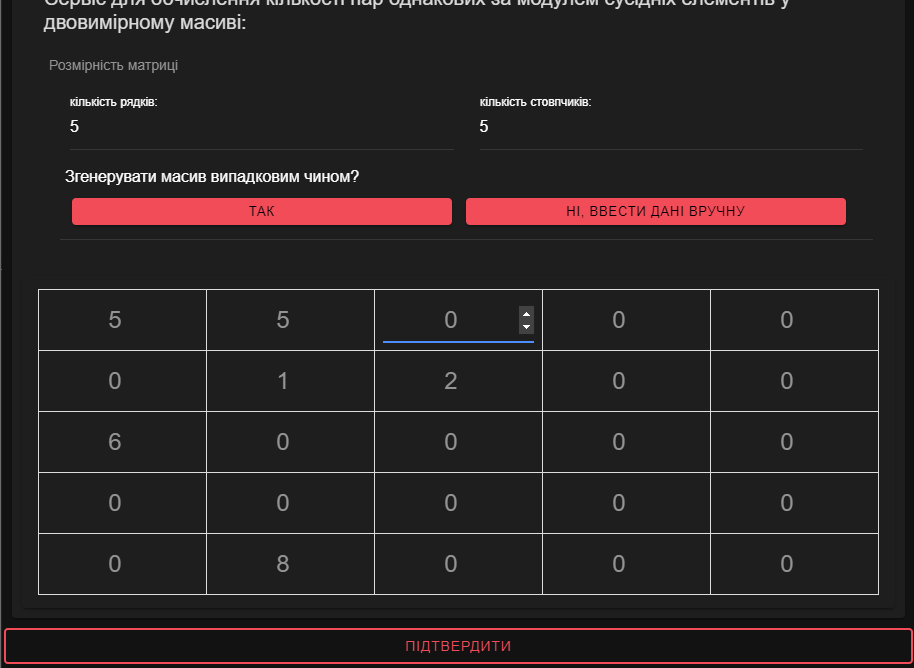
Розробити сервіс для обчислення кількості пар однакових за модулем сусідніх елементів у двовимірному масиві. В якості сусідніх можуть виступати як елементи, що знаходяться в одному стовпчику так і елементи, що знаходяться в одному рядку. Однакові елементи потрібно виокремити кольором.

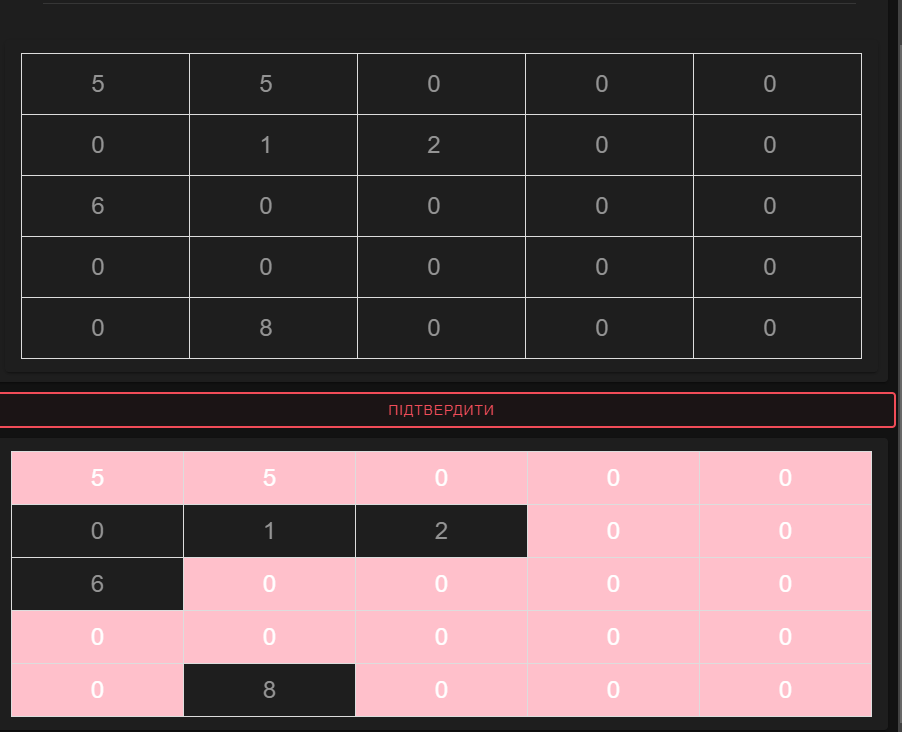
Форма для введення повинна спочатку запитати розмірність масиву, потім запититати чи потрібно згенерувати масив

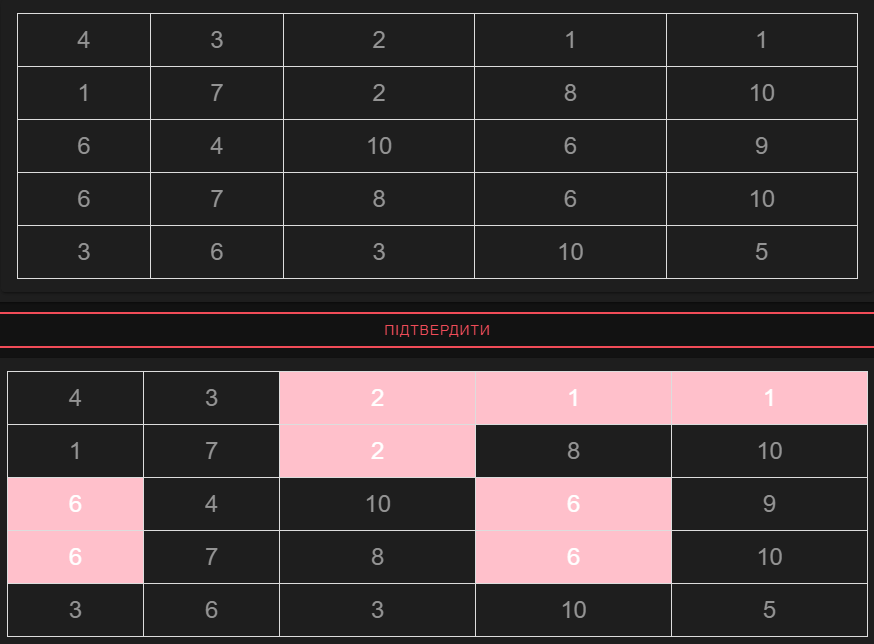
**Виконання:**











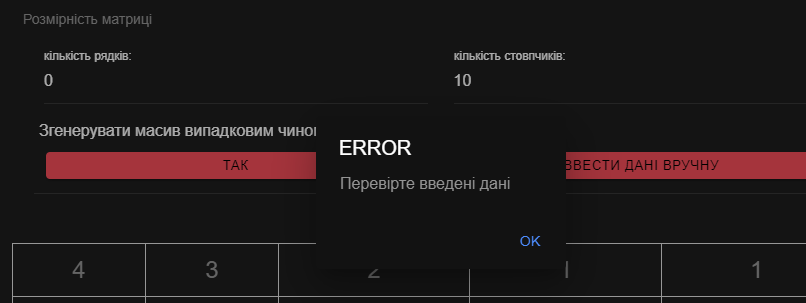


Рисунок 1-6 – результат роботи застосунку на хостингу

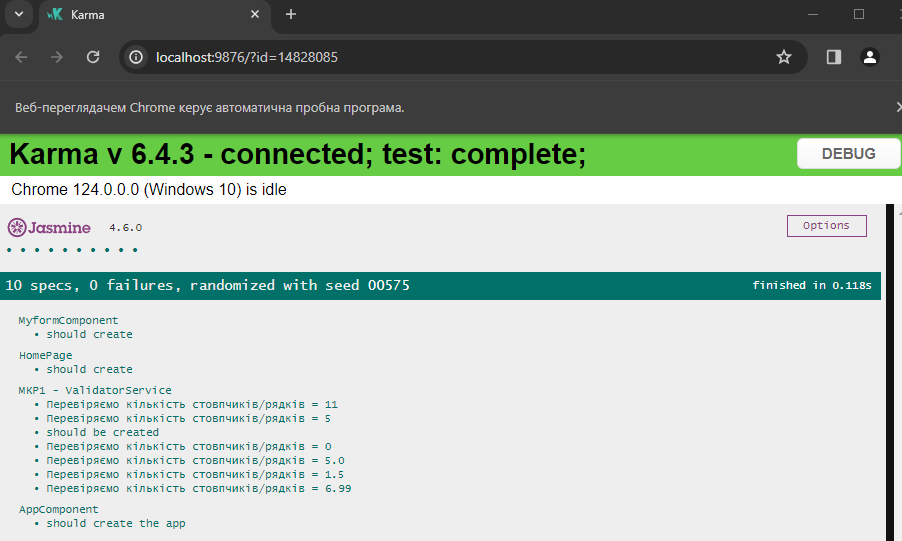


Рисунок 7 – результат проходження тестів

Код тестів (validator.service.spec.ts):

import { TestBed } from '@angular/core/testing';

import { ValidatorService } from './validator.service';

describe('МКР1 - ValidatorService', () => {

  let service: ValidatorService;

  beforeEach(() => {

    TestBed.configureTestingModule({});

    service = TestBed.inject(ValidatorService);

  });

  it('should be created', () => {

    expect(service).toBeTruthy();

  });

  it('Перевіряємо кількість стовпчиків/рядків = 0', () => {

    let x = service.validate\_amount(0);

    expect(x).toBe(false);

  });

  it('Перевіряємо кількість стовпчиків/рядків = 11', () => {

    let x = service.validate\_amount(11);

    expect(x).toBe(false);

  });

  it('Перевіряємо кількість стовпчиків/рядків = 5', () => {

    let x = service.validate\_amount(5);

    expect(x).toBe(true);

  });

  it('Перевіряємо кількість стовпчиків/рядків = 1.5', () => {

    let x = service.validate\_amount(1.5);

    expect(x).toBe(false);

  });

  it('Перевіряємо кількість стовпчиків/рядків = 5.0', () => {

    let x = service.validate\_amount(5.0);

    expect(x).toBe(true);

  });

  it('Перевіряємо кількість стовпчиків/рядків = 6.99', () => {

    let x = service.validate\_amount(6.99);

    expect(x).toBe(false);

  });

});

**Додатки:**

Посилання на GitHub:

<https://github.com/ffructose/CPP_Hnatiuk_KN-32_Lab1/tree/main/Module>

Посилання на хостинг:

<https://mkr1-8144c.web.app/home>

Код програми:

**app\home\home-routing.module.ts**

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

import { RouterModule, Routes } from '@angular/router';

import { HomePage } from './home.page';

const routes: Routes = [

  {

    path: '',

    component: HomePage,

  }

];

@NgModule({

  imports: [RouterModule.forChild(routes)],

  exports: [RouterModule]

})

export class HomePageRoutingModule {}

**app\home\home.module.ts**

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

import { CommonModule } from '@angular/common';

import { IonicModule } from '@ionic/angular';

import { FormsModule } from '@angular/forms';

import { HomePage } from './home.page';

import { HomePageRoutingModule } from './home-routing.module';

import { MyformComponent } from '../myform/myform.component';

import { ReactiveFormsModule } from '@angular/forms';

@NgModule({

  imports: [

    CommonModule,

    FormsModule,

    IonicModule,

    HomePageRoutingModule,

    ReactiveFormsModule

  ],

  declarations: [HomePage, MyformComponent]

})

export class HomePageModule {}

**app\home\home.page.html**

<ion-header [translucent]="true">

  <ion-toolbar color="danger">

    <ion-buttons slot="start">

      <ion-menu-button color="dark" menu="first"></ion-menu-button>

    </ion-buttons>

    <ion-title>

      Модульна контрольна робота 1 (перескладання)

    </ion-title>

  </ion-toolbar>

  <ion-toolbar>

    <ion-title>

      Гнатюк Анна

    </ion-title>

  </ion-toolbar>

  <ion-toolbar>

    <ion-title>

      КН-32 Варіант №1

    </ion-title>

  </ion-toolbar>

</ion-header>

<ion-content [fullscreen]="true">

  <app-myform (operationResult)="handleOperationResult($event)"></app-myform>

  <ion-card \*ngIf="operationResults?.printResult === true">

    <ion-card-content>

      <table class="styled-table">

        <tbody>

          <tr \*ngFor="let row of operationResults?.array; let i = index">

            <td \*ngFor="let cell of row; let j = index" [ngClass]="{'highlight': highlightedCells.has(i + '-' + j)}">

              {{ cell }}

            </td>

          </tr>

        </tbody>

      </table>

    </ion-card-content>

  </ion-card>

</ion-content>

**app\home\home.page.scss**

#container {

  text-align: center;

  position: absolute;

  left: 0;

  right: 0;

  top: 50%;

  transform: translateY(-50%);

}

#container strong {

  font-size: 20px;

  line-height: 26px;

}

#container p {

  font-size: 16px;

  line-height: 22px;

  color: #8c8c8c;

  margin: 0;

}

#container a {

  text-decoration: none;

}

.styled-table {

  width: 100%;

  border-collapse: collapse;

}

.styled-table td {

  border: 1px solid #ddd;

  padding: 8px;

  text-align: center;

  font-size: 24px;

}

.highlight {

  background-color: pink;

  color: white;

}

**app\home\home.page.spec.ts**

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { IonicModule } from '@ionic/angular';

import { HomePage } from './home.page';

describe('HomePage', () => {

  let component: HomePage;

  let fixture: ComponentFixture<HomePage>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [HomePage],

      imports: [IonicModule.forRoot()]

    }).compileComponents();

    fixture = TestBed.createComponent(HomePage);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

**app\home\home.page.ts**

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

import { MatrixOperationResult } from '../myform/matrix-operation-result.interface';

@Component({

  selector: 'app-home',

  templateUrl: 'home.page.html',

  styleUrls: ['home.page.scss'],

})

export class HomePage {

  highlightedCells: Set<string> = new Set();

  operationResults!: MatrixOperationResult;

  constructor(private cdr: ChangeDetectorRef) { }

  handleOperationResult(result: MatrixOperationResult) {

    this.operationResults = result;

    this.highlightAdjacentMatches();

    this.cdr.detectChanges();  // Mark the component for change detection

  }

  highlightAdjacentMatches() {

    this.highlightedCells.clear();

    let rows = this.operationResults.array.length;

    let columns = this.operationResults.array[0].length;

    for (let i = 0; i < rows; i++) {

      for (let j = 0; j < columns; j++) {

        if (j + 1 < columns && this.operationResults.array[i][j] === this.operationResults.array[i][j + 1]) {

          this.highlightedCells.add(`${i}-${j}`);

          this.highlightedCells.add(`${i}-${j + 1}`);

        }

        if (i + 1 < rows && this.operationResults.array[i][j] === this.operationResults.array[i + 1][j]) {

          this.highlightedCells.add(`${i}-${j}`);

          this.highlightedCells.add(`${i + 1}-${j}`);

        }

      }

    }

  }

}

**app\myform\Service\myValidator.ts**

import { AbstractControl, ValidatorFn } from "@angular/forms";

import { ValidatorService } from "./validator.service";

export function amountValidator(): ValidatorFn {

  return (

    control: AbstractControl

  ): { [key: string]: boolean } | null => {

    let validator = new ValidatorService()

    let valid =

      !control.value || validator.validate\_amount(control.value)

    return valid ? null : { date: true }

  }

}

**app\myform\Service\validator.service.spec.ts**

import { TestBed } from '@angular/core/testing';

import { ValidatorService } from './validator.service';

describe('МКР1 - ValidatorService', () => {

  let service: ValidatorService;

  beforeEach(() => {

    TestBed.configureTestingModule({});

    service = TestBed.inject(ValidatorService);

  });

  it('should be created', () => {

    expect(service).toBeTruthy();

  });

  it('Перевіряємо кількість стовпчиків/рядків = 0', () => {

    let x = service.validate\_amount(0);

    expect(x).toBe(false);

  });

  it('Перевіряємо кількість стовпчиків/рядків = 11', () => {

    let x = service.validate\_amount(11);

    expect(x).toBe(false);

  });

  it('Перевіряємо кількість стовпчиків/рядків = 5', () => {

    let x = service.validate\_amount(5);

    expect(x).toBe(true);

  });

  it('Перевіряємо кількість стовпчиків/рядків = 1.5', () => {

    let x = service.validate\_amount(1.5);

    expect(x).toBe(false);

  });

  it('Перевіряємо кількість стовпчиків/рядків = 5.0', () => {

    let x = service.validate\_amount(5.0);

    expect(x).toBe(true);

  });

  it('Перевіряємо кількість стовпчиків/рядків = 6.99', () => {

    let x = service.validate\_amount(6.99);

    expect(x).toBe(false);

  });

});

**app\myform\Service\validator.service.ts**

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

@Injectable({

  providedIn: 'root'

})

export class ValidatorService {

  validate\_amount(value: number) {

    if (value > 10 || value < 1 || !Number.isInteger(value)) return false

    return true;

  }

  constructor() { }

}

**app\myform\matrix-operation-result.interface.ts**

export interface MatrixOperationResult {

    printResult: boolean;

    array: number[][];

  }

**app\myform\myform.component.html**

<form [formGroup]="myForm" (ngSubmit)="onSubmit()">

  <ion-card>

    <ion-card-content>

      <ion-card-header>

        <ion-card-title> Сервіс для обчислення кількості пар однакових за модулем сусідніх елементів у двовимірному

          масиві: </ion-card-title>

      </ion-card-header>

      <ion-card-content>

        <ion-grid>

          <ion-row>

            Розмірність матриці

          </ion-row>

          <ion-row>

            <ion-col>

              <ion-item>

                <ion-label position="floating">кількість рядків:</ion-label>

                <ion-input id="amount-rows" type="number" formControlName="amountRows"></ion-input>

              </ion-item>

            </ion-col>

            <ion-col>

              <ion-item>

                <ion-label position="floating">кількість стовпчиків:</ion-label>

                <ion-input id="amount-cols" type="number" formControlName="amountCols"></ion-input>

              </ion-item>

            </ion-col>

          </ion-row>

        </ion-grid>

        <ion-item>

          <ion-grid>

            <ion-row>

              <ion-label position="stuck">Згенерувати масив випадковим чином?</ion-label>

            </ion-row>

            <ion-row>

              <ion-col>

                <ion-button (click)="fillRandomNumbers()" color="danger" expand="block">

                  Так

                </ion-button>

              </ion-col>

              <ion-col>

                <ion-button (click)="fillZeroes()" color="danger" expand="block">

                  Ні, ввести дані вручну

                </ion-button>

              </ion-col>

            </ion-row>

          </ion-grid>

        </ion-item>

      </ion-card-content>

    </ion-card-content>

    <ion-card \*ngIf="isRandom">

      <ion-card-content>

        <table class="styled-table">

          <tbody>

            <tr \*ngFor="let row of array">

              <td \*ngFor="let cell of row">

                {{ cell }}

              </td>

            </tr>

          </tbody>

        </table>

      </ion-card-content>

    </ion-card>

    <ion-card \*ngIf="!isRandom">

      <ion-card-content>

        <table class="styled-table">

          <tbody>

            <tr \*ngFor="let row of array; let i = index">

              <td \*ngFor="let cell of row; let j = index">

                <ion-input type="number" [value]="cell" (input)="handleInput(i, j, $event)"></ion-input>

              </td>

            </tr>

          </tbody>

        </table>

      </ion-card-content>

    </ion-card>

  </ion-card>

  <ion-button (click)="onSubmit()" type="submit" color="danger" expand="block" fill="outline"

    [disabled]="!myForm.valid">

    Підтвердити

  </ion-button>

</form>

**app\myform\myform.component.scss**

.styled-table {

    width: 100%;

    border-collapse: collapse;

  }

  .styled-table td {

    border: 1px solid #ddd;

    padding: 8px;

    text-align: center;

    font-size: 24px;

  }

  .highlight {

    background-color: pink;

    color: white;

  }

**app\myform\myform.component.spec.ts**

import { ComponentFixture, TestBed, waitForAsync } from '@angular/core/testing';

import { IonicModule } from '@ionic/angular';

import { MyformComponent } from './myform.component';

describe('MyformComponent', () => {

  let component: MyformComponent;

  let fixture: ComponentFixture<MyformComponent>;

  beforeEach(waitForAsync(() => {

    TestBed.configureTestingModule({

      declarations: [ MyformComponent ],

      imports: [IonicModule.forRoot()]

    }).compileComponents();

    fixture = TestBed.createComponent(MyformComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  }));

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

**app\myform\myform.component.ts**

import { Component, OnInit, Output, EventEmitter } from '@angular/core';

import { FormControl, FormGroup, FormBuilder } from '@angular/forms';

import { Validators } from '@angular/forms';

import { ValidatorService } from './Service/validator.service';

import { amountValidator } from './Service/myValidator';

import { AlertController } from '@ionic/angular';

import { MatrixOperationResult } from './matrix-operation-result.interface';

@Component({

  selector: 'app-myform',

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

  styleUrls: ['./myform.component.scss'],

})

export class MyformComponent implements OnInit {

  myForm!: FormGroup;

  array: number[][] = [];

  finalArray: number[][] = [];

  isRandom!: boolean;

  printResult!: boolean;

  @Output() operationResult = new EventEmitter<MatrixOperationResult>();

  constructor(private fb: FormBuilder, private alertController: AlertController) {

    this.myForm = this.fb.group({

      amountRows: ['', [amountValidator()]],

      amountCols: ['', [amountValidator()]],

    });

  }

  fillRandomNumbers() {

    this.isRandom = true;

    this.printResult = false;

    let rows = this.myForm.value.amountRows;

    let columns = this.myForm.value.amountCols;

    let valid = new ValidatorService();

    if (!valid.validate\_amount(rows) || !valid.validate\_amount(columns)) {

      this.presentAlert("Перевірте введені дані");

    } else {

      this.array = []; // Очищення масиву перед заповненням

      for (let i = 0; i < rows; i++) {

        let row: number[] = [];

        for (let j = 0; j < columns; j++) {

          let randomValue = Math.floor(Math.random() \* (10 - 1 + 1)) + 1;

          row.push(randomValue);

        }

        this.array.push(row);

      }

      console.log("submit");

    }

  }

  fillZeroes() {

    this.isRandom = false;

    this.printResult = false;

    let rows = this.myForm.value.amountRows;

    let columns = this.myForm.value.amountCols;

    let valid = new ValidatorService();

    if (!valid.validate\_amount(rows) || !valid.validate\_amount(columns)) {

      this.presentAlert("Перевірте введені дані");

    } else {

      this.array = []; // Очищення масиву перед заповненням

      for (let i = 0; i < rows; i++) {

        let row: number[] = [];

        for (let j = 0; j < columns; j++) {

          row.push(0);

        }

        this.array.push(row);

      }

      console.log("submit");

    }

  }

  handleInput(rowIndex: number, colIndex: number, event: any) {

    this.array[rowIndex][colIndex] = parseFloat(event.target.value);

  }

  onSubmit() {

    this.finalArray = this.array;

    this.printResult = true;

    this.operationResult.emit({

      printResult: this.printResult,

      array: this.finalArray,

    });

  }

  async presentAlert(message: string) {

    const alert = await this.alertController.create({

      header: 'ERROR',

      subHeader: '',

      message: message,

      buttons: ['OK']

    });

    await alert.present();

  }

  ngOnInit() { }

}

**app\app-routing.module.ts**

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

import { PreloadAllModules, RouterModule, Routes } from '@angular/router';

const routes: Routes = [

  {

    path: 'home',

    loadChildren: () => import('./home/home.module').then( m => m.HomePageModule)

  },

  {

    path: '',

    redirectTo: 'home',

    pathMatch: 'full'

  },

];

@NgModule({

  imports: [

    RouterModule.forRoot(routes, { preloadingStrategy: PreloadAllModules })

  ],

  exports: [RouterModule]

})

export class AppRoutingModule { }

**app\app.component.html**

<ion-app>

  <ion-router-outlet></ion-router-outlet>

</ion-app>

**app\app.component.scss**

**app\app.component.spec.ts**

import { CUSTOM\_ELEMENTS\_SCHEMA } from '@angular/core';

import { TestBed } from '@angular/core/testing';

import { AppComponent } from './app.component';

describe('AppComponent', () => {

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [AppComponent],

      schemas: [CUSTOM\_ELEMENTS\_SCHEMA],

    }).compileComponents();

  });

  it('should create the app', () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app).toBeTruthy();

  });

});

**app\app.component.ts**

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

@Component({

  selector: 'app-root',

  templateUrl: 'app.component.html',

  styleUrls: ['app.component.scss'],

})

export class AppComponent {

  constructor() {}

}

**app\app.module.ts**

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

import { BrowserModule } from '@angular/platform-browser';

import { RouteReuseStrategy } from '@angular/router';

import { IonicModule, IonicRouteStrategy } from '@ionic/angular';

import { AppComponent } from './app.component';

import { AppRoutingModule } from './app-routing.module';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

@NgModule({

  declarations: [AppComponent],

  imports: [ReactiveFormsModule, BrowserModule, IonicModule.forRoot(), AppRoutingModule, FormsModule],

  providers: [{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy }],

  bootstrap: [AppComponent],

})

export class AppModule {}