

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

# Кафедра системного програмування і спеціалізованих комп’ютерних систем

**Лабораторна робота №2**

з дисципліни

з дисципліни **WEB-дизайн**

на тему: “ Розробка функціональності Web-додатка мовою Javascript ”

Виконала: студентка ІII курсу

ФПМ групи КВ-01

Соліна Т.А.

Київ - 2023

**Мета**: ознайомитись із засобами мови Javascript та навчитись їх застосовувати побудови Web-інтерфейсу користувача.

**Загальне завдання**: розробити функціональність для статичних сторінок Web-додатку першої лабораторної роботи із використанням шаблону MVC

**Інструменти розробки**: мови HTML5, CSS3, Javascript.

**Назва та опис роботи додатка: Сайт “Калькулятор”**

Опис роботи:

Головне завдання сайта: калькулятор

При розробці дизайну обрано веселкові кольори, які домінують на сторінці. Такі кольори обрано з розрахунком на тематику логотипу unicorn (единоріг), сайт буде використовуватись для роботи.

При вході на сайт користувач, якщо він не авторизований, то потратляє не сторінку реєстрації, звідти може перейти на сторінку входу, якщо він вже має свою сторінку.

Після авторизації користувач потрапляє на сторінку профіля, де він може побачити дані про свій обліковий запис.

Потім, за допомогою статичного сайдбара він може потрапити на сторінку калькулятора і обрати вид калькулятора, яким він хоче користуватися.

Sidebar вміщує в собі фото користувача, яке поки що є фото за замовчуванням, при натисненні на яке користувач потрапляє на сторінку користувача. Також присутні кнопки “About”, яка переправляє користувача на сторінку з інформацією про калькулятори. Кнопка “Log out” (кнорпка виходу) переносить користувача на сторінку входу.

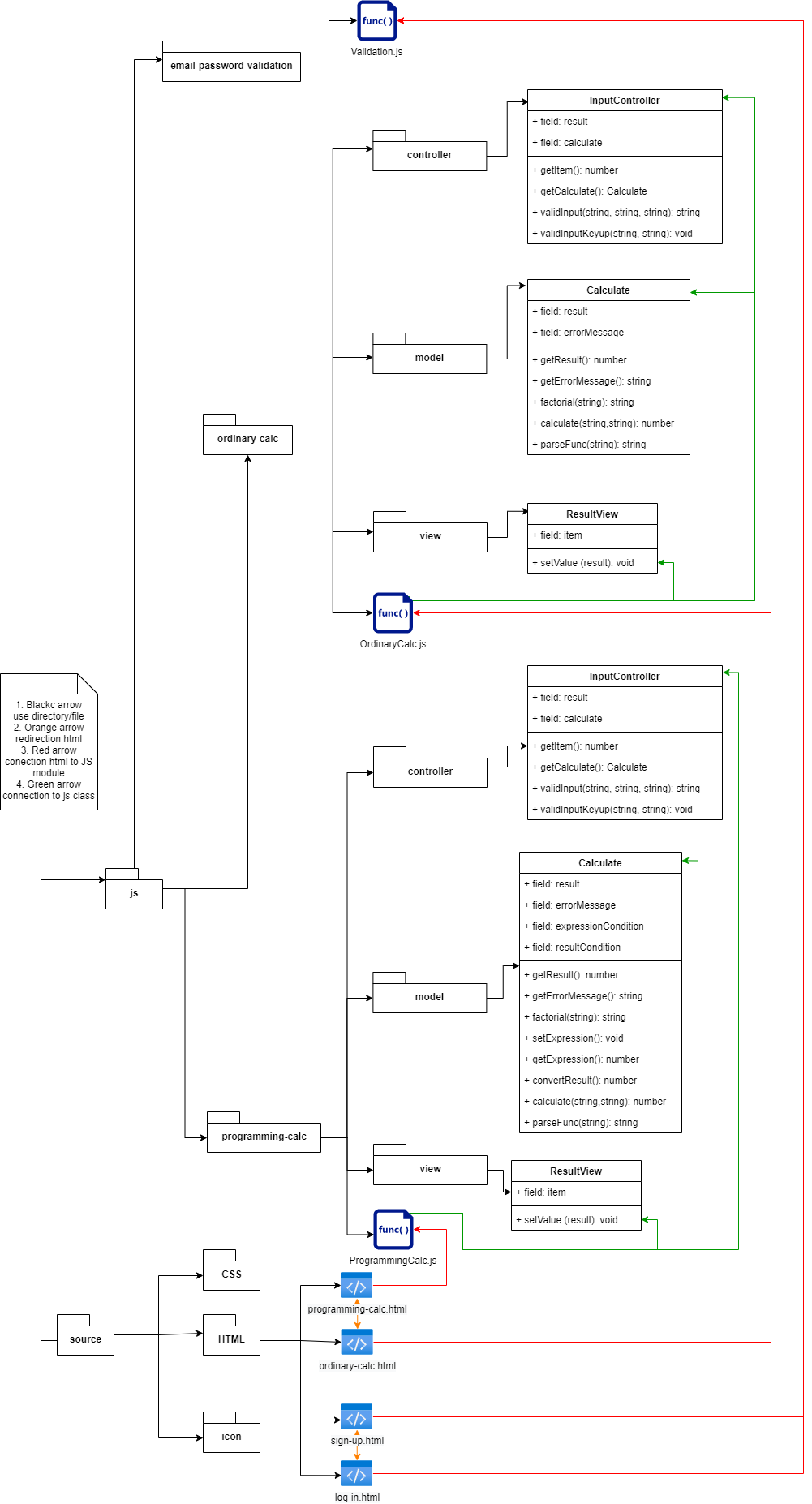
Navbar вміщує в собі логотип додатка а також кнопку про додаток.

Опис основної функціональності: калькулятор

Калькулятор може працювати в двох режимах роботи. Звичайному та в програмованому. Зміна режимів може відбуватися за допомогою сайдбара.

У звичайному режимі калькулятор може виконувати обчислення шляхом додавання, віднімання, множення або ділення результуючого (значення “Result”) значення та введеного (значення “Expression”). Вибравши функцію відсотка можна визначити введений відсоток від результуючого значення. Значення логарифмів (“log” - десятковий логарифм, “ln” - натуральний логарифм), факторіала та кореня квадратного обчислюється наступним чином. Виконується зазначена раніше операція над числом в полі “Expression” та виконується над ним і результатом операція “Operation”.

У програмованому режимі можна виконувати обрахунки всі ті ж самі обрахунки, що і в звичайному тільки в різних системах числення : двійковій, вісімковій, десятковій та шістнадцятковій. Можна також виконувати переведення між різними системами числення, для чого необхідно просто поміняти значення “DEC” (за замовчуванням) на іншу потрібну систему числення. Також є можливість вести обчислення між різними системами числення. Наприклад, до числа в двійковій системі числення додати число в десятковій, вісімковій чи шістнадцятковій системі числення.

**Діаграма класів JavaScript додатка:**

**Скріни статичних веб сторінок додатка:**

Рис.1 - сторінка реєстрації

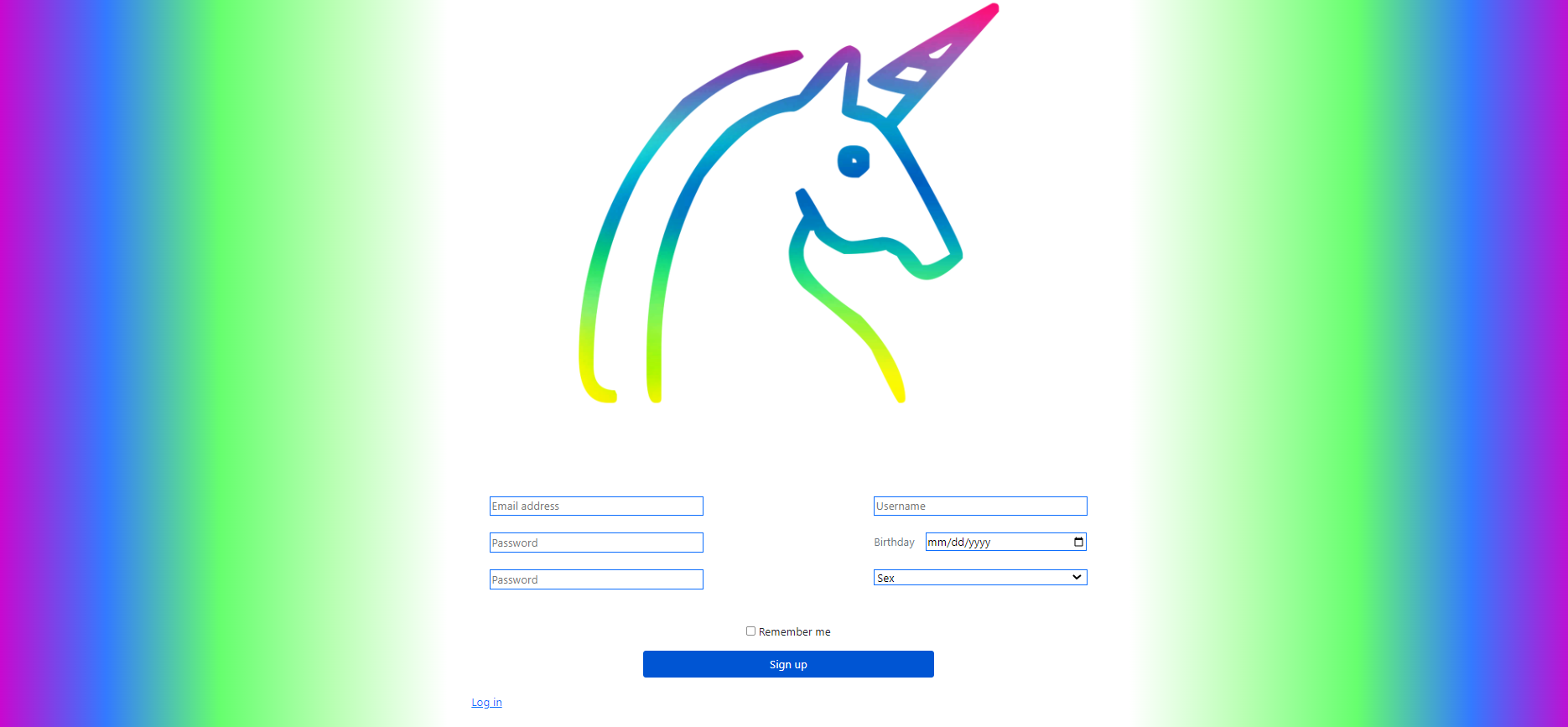


Рис.2 - сторінка входу:

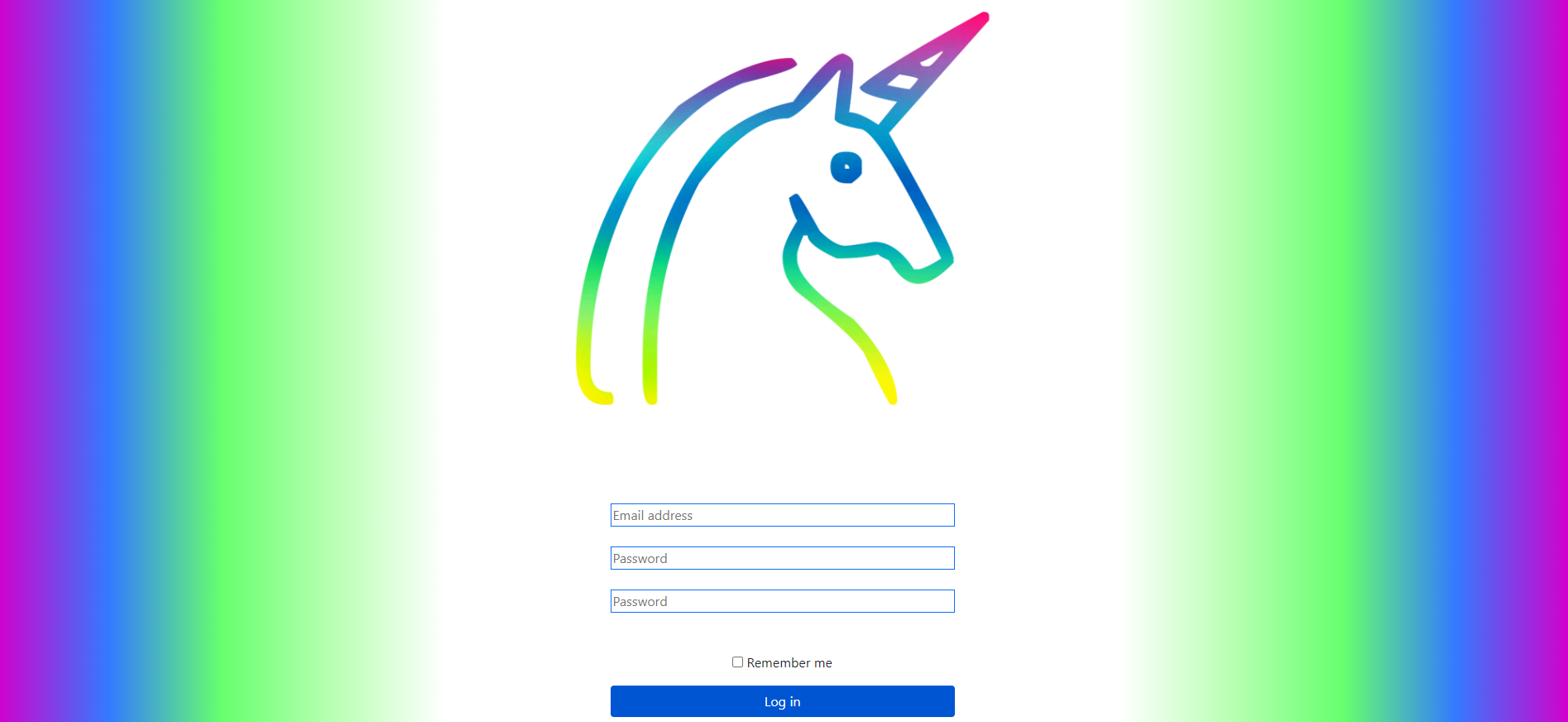


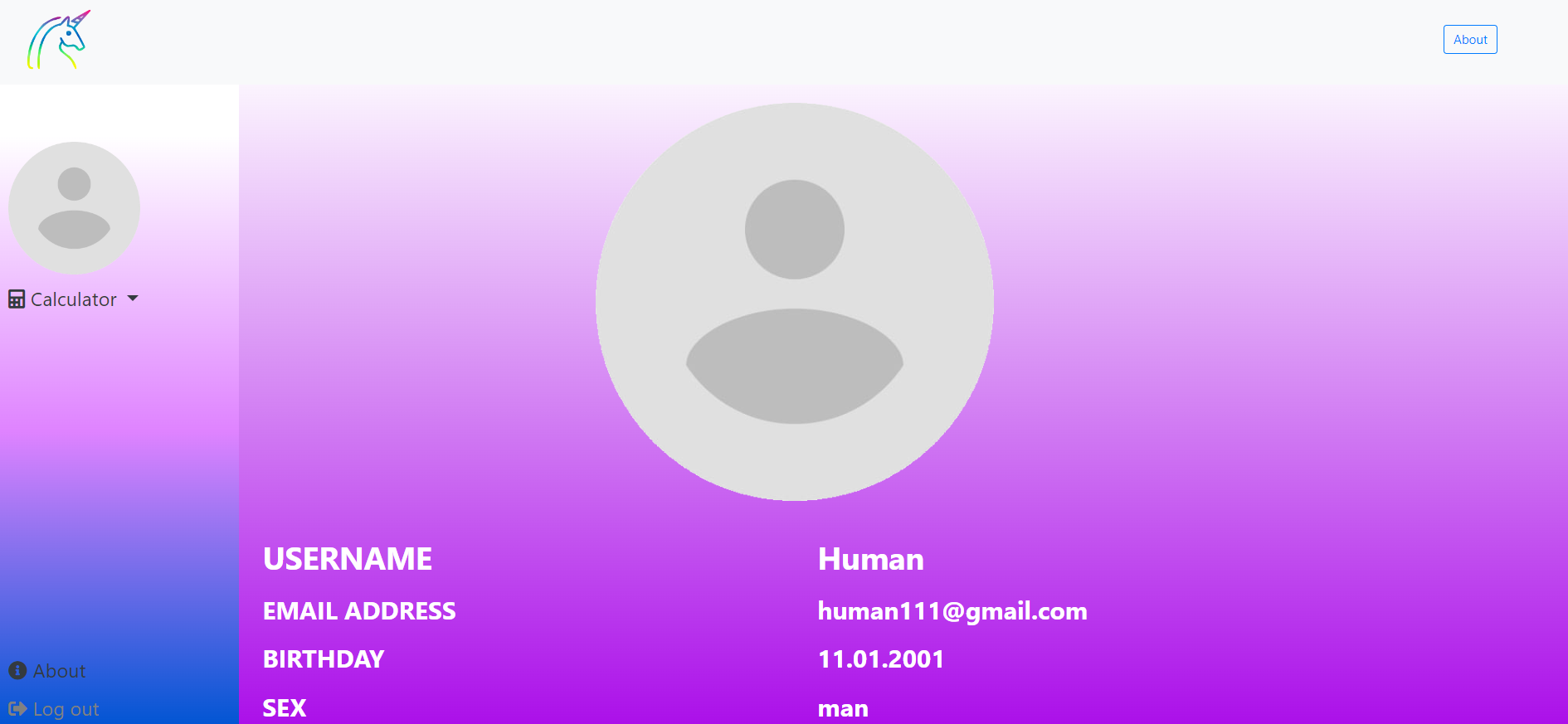
Рис.3 - сторінка профіля

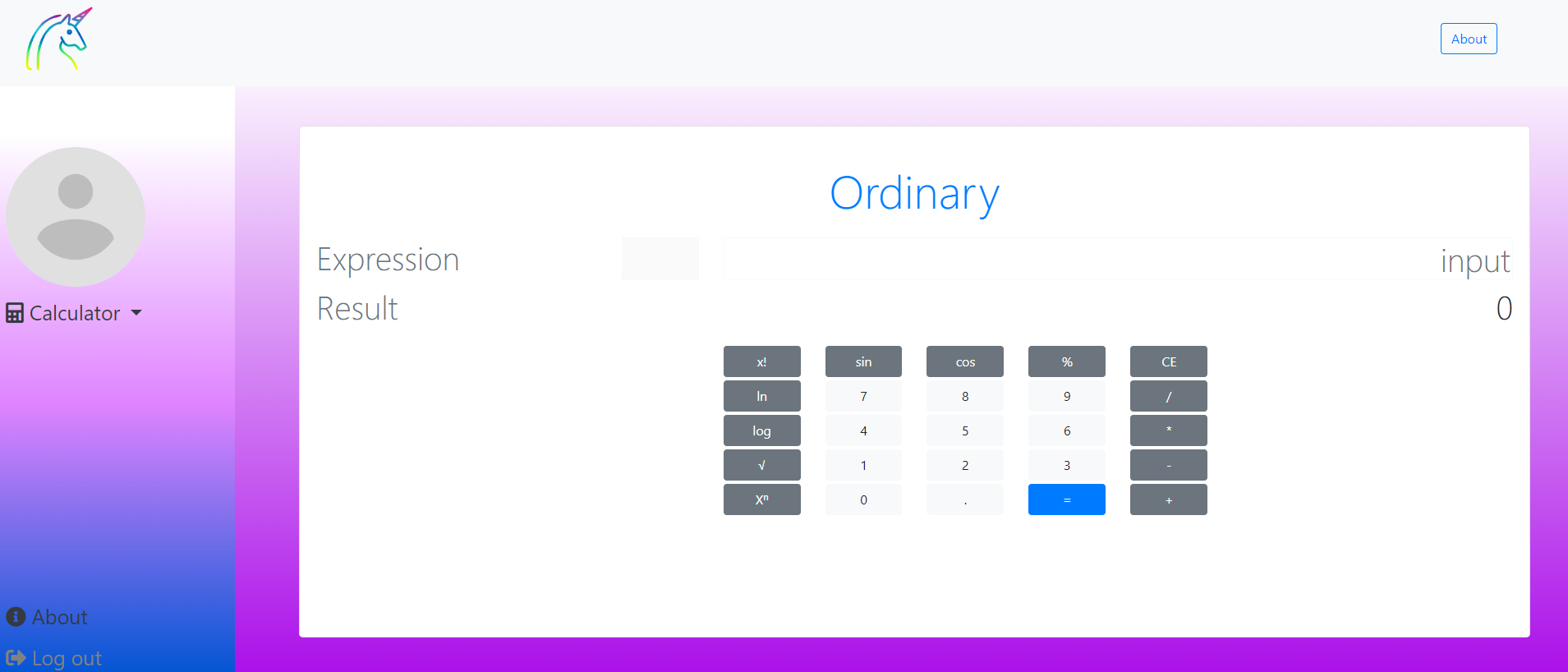
Рис.4 - сторінка звичайного калькулятора

Рис.5 - сторінка програмістського калькулятора

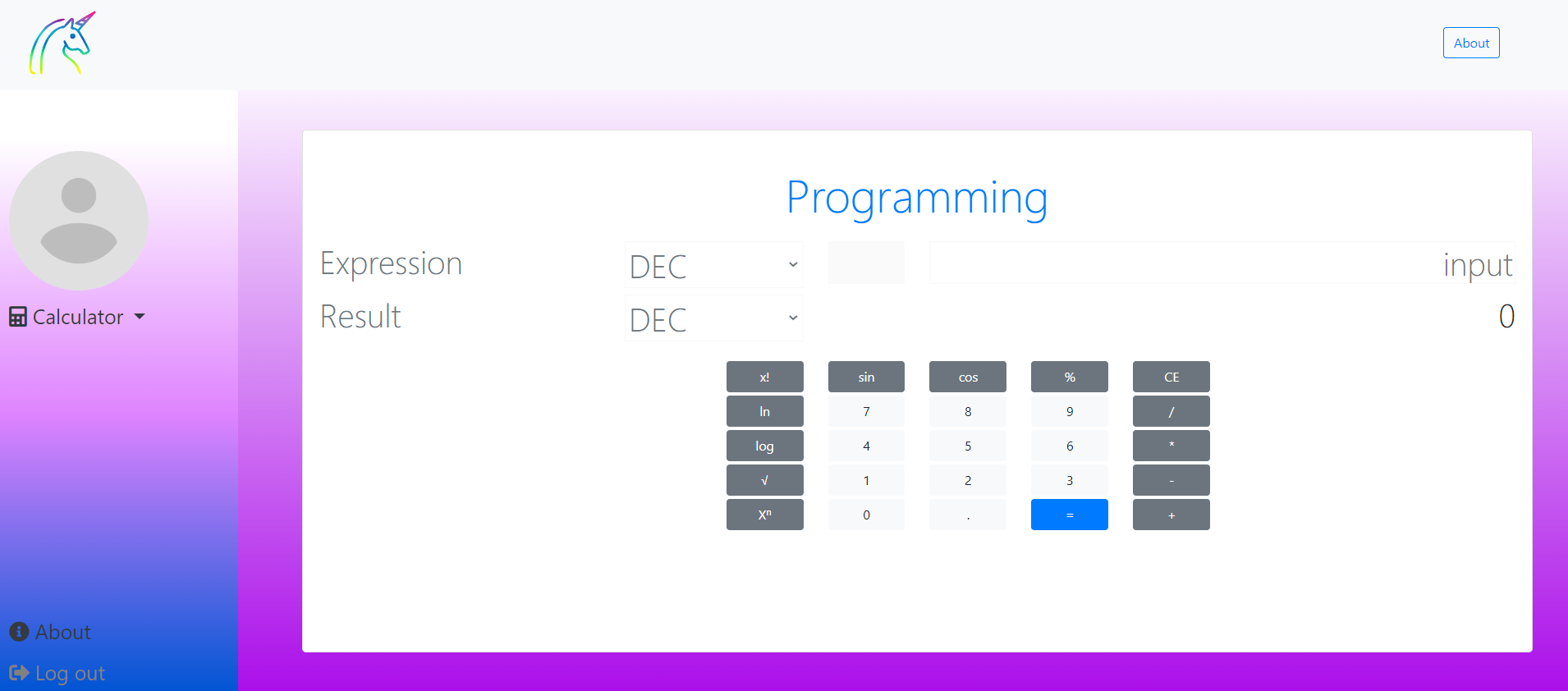
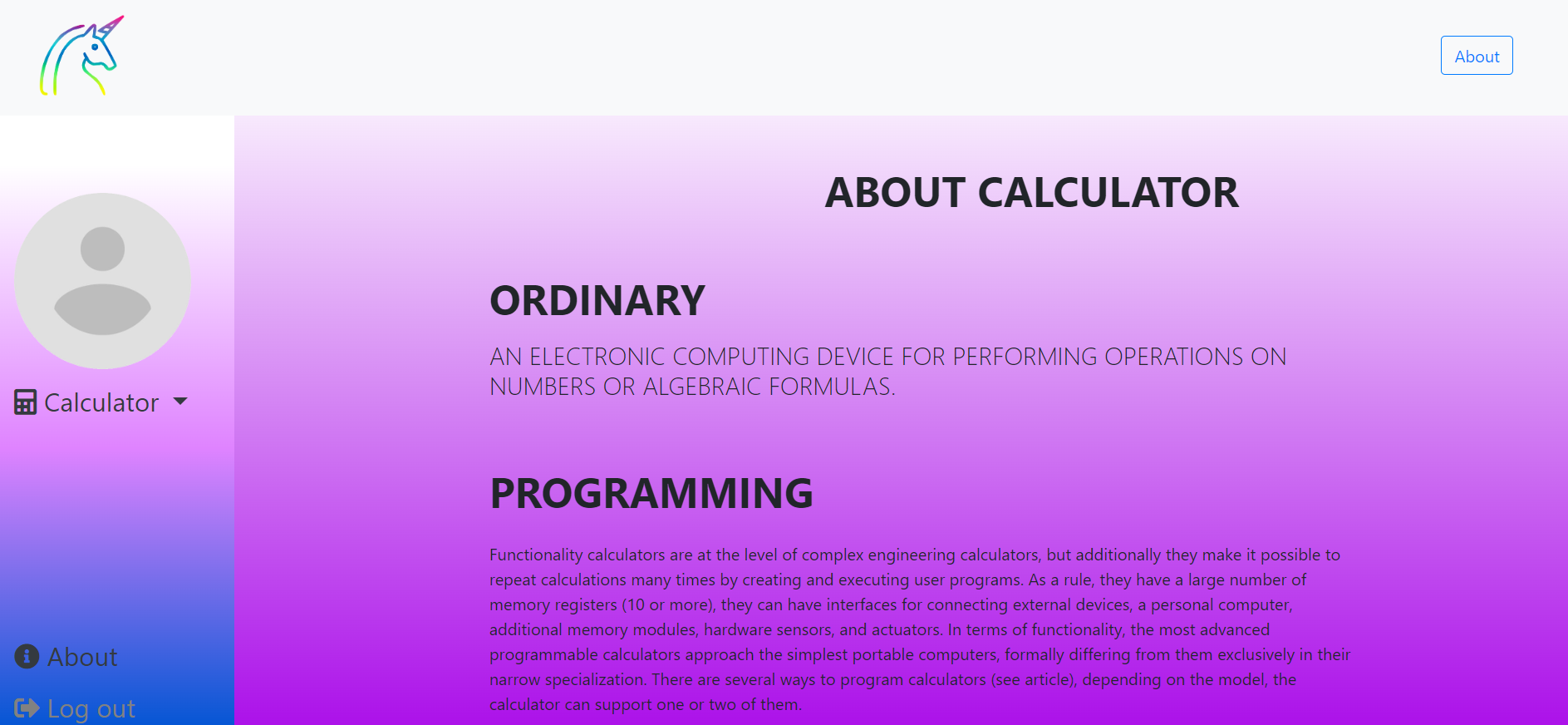


Рис.6 - сторінка про додаток (про калькулятор) 

**Текст файлів JS робочої сторінки**

OrdinaryCalc.js

import InputController from './controller/InputController.js';

import Calculate from './model/Calculate.js';

import ResultView from './view/ResultView.js';

let inputController = new InputController(document.getElementById("expression"), new Calculate());

let calculateExpression = inputController.getCalculator();

let viewInput = new ResultView(document.getElementById("result"));

let expression = document.getElementById('expression');

let opereation = document.getElementById('operation');

viewInput.setValue(inputController.getItem().value);

function handlerKeydownExpression(key, expression) {

    opereation.value = inputController.validInput(key,

        opereation.value, expression);

    viewInput.setValue(calculateExpression.getResult());

    document.getElementById("error").textContent = calculateExpression.getErrorMessage();

}

function handlerOperation(operand) {

    let lastExpression = expression.value;

    expression.value = '';

    opereation.value = operand;

    viewInput.setValue(calculateExpression.calculate(lastExpression, operand));

}

document.getElementById('equal').addEventListener('click',

    function() {

    viewInput.setValue(calculateExpression.calculate(inputController.getItem().value,

        document.getElementById('operation').value));

    expression.value = '';

});

expression.addEventListener('keydown',

    function(event) {

    handlerKeydownExpression(event.key, expression);

});

expression.addEventListener('keyup',

    function(event) {

    inputController.validInputKeyup(event.key, expression);

})

document.getElementById('erase').addEventListener('click',

    function() {

    let erasExpression = expression.value;

    expression.value = erasExpression.substr(0, erasExpression.length - 1);

});

document.getElementById("factorial").addEventListener('click',

    function() {

    viewInput.setValue(calculateExpression

        .calculate(calculateExpression.factorial(expression.value), opereation.value));

    expression.value = '';

});

document.getElementById("sin").addEventListener('click',

    function() {

    expression.value = 'sin('.concat(expression.value);

    handlerKeydownExpression(')', expression);

});

document.getElementById("cos").addEventListener('click',

    function() {

    expression.value = 'cos('.concat(expression.value);

    handlerKeydownExpression(')', expression);

});

document.getElementById("log").addEventListener('click',

    function() {

    expression.value = 'log('.concat(expression.value);

    handlerKeydownExpression(')', expression);

});

document.getElementById("ln").addEventListener('click',

    function() {

    expression.value = 'ln('.concat(expression.value);

    handlerKeydownExpression(')', expression);

});

document.getElementById("sqrt").addEventListener('click',

    function() {

    expression.value = 'sqrt('.concat(expression.value);

    handlerKeydownExpression(')', expression);

});

document.getElementById("pow").addEventListener('click',

    function() {

    expression.value = 'pow('.concat(expression.value).concat('^2');

    handlerKeydownExpression(')', expression);

});

document.getElementById('plus').addEventListener('click',

    function(){ handlerOperation('+') });

document.getElementById('minus').addEventListener('click',

    function(){ handlerOperation('-') });

document.getElementById('multiply').addEventListener('click',

    function(){ handlerOperation('\*') });

document.getElementById('division').addEventListener('click',

    function(){ handlerOperation('/') });

document.getElementById('percent').addEventListener('click',

    function(){ handlerOperation('%') });

document.getElementById('zero').addEventListener('click',

    function() {expression.value += 0; });

document.getElementById('one').addEventListener('click',

    function() {expression.value += 1; });

document.getElementById('two').addEventListener('click',

    function() {expression.value += 2; });

document.getElementById('three').addEventListener('click',

    function() {expression.value += 3; });

document.getElementById('four').addEventListener('click',

    function() {expression.value += 4; });

document.getElementById('five').addEventListener('click',

    function() {expression.value += 5; });

document.getElementById('six').addEventListener('click',

    function() {expression.value += 6; });

document.getElementById('seven').addEventListener('click',

    function() {expression.value += 7; });

document.getElementById('eighth').addEventListener('click',

    function() {expression.value += 8; });

document.getElementById('nine').addEventListener('click',

    function() {expression.value += 9; });

document.getElementById('point').addEventListener('click',

    function() {expression.value += '.'; });

controller / InputController.js

import Calculate from '../model/Calculate.js';

export default class InputController {

    constructor(item, calculate) {

        this.item = item;

        this.calculate = calculate;

    }

    getItem() { return this.item; }

    getCalculator() { return this.calculate; }

    validInput(key, lastValue, expression) {

        let sign = lastValue;

        if (key == = '+' | key == = '-' | key == = '/' | key == = '\*' | key == = '%') {

            sign = key;

            this.calculate.calculate(expression.value, key);

            expression.value = '';

        }

        if (key == = '!') {

            this.calculate.calculate(this.calculate.factorial(expression.value), lastValue);

            expression.value = '';

        }

        if (key == = ')') {

            this.calculate.calculate(this.calculate.parseFunc(expression.value), lastValue);

            expression.value = '';

        }

        if (key == = 'Enter') {

            this.calculate.calculate(expression.value, lastValue);

            expression.value = '';

        }

        return sign;

    }

    validInputKeyup(key, expression) {

        if (key == = '+' | key == = '-' | key == = '/' | key == = '\*' | key == = '%'

            | key == = '!' | key == = ')' | key == = 'Enter') {

            expression.value = '';

        }

    }

}

model / Calculate.js

export default class Calculate {

    constructor() {

        this.result = 0;

        this.errorMessage = '';

    }

    getResult() { return this.result; }

    getErrorMessage() { return this.errorMessage; }

    factorial(expression) {

        expression = parseFloat(expression.replace('+', '').replace('-', '')

            .replace('\*', '').replace('/', '').replace('%', ''));

        let res = 1;

        for (let i = 2; i <= expression; i++) {

            res \*= i;

        }

        return res.toString();

    }

    calculate(expression, operation) {

        if (expression.match(/ [0 - 9] / )) {

            switch (operation) {

            case '+':

                this.result += parseFloat(expression.replace('+', ''));

                break;

            case '-':

                this.result -= parseFloat(expression.replace('-', ''));

                break;

            case '\*':

                this.result \*= parseFloat(expression.replace('\*', ''));

                break;

            case '/':

                this.result /= parseFloat(expression.replace('/', ''));

                break;

            case '%':

                this.result \*= parseFloat(expression.replace('%', '')) / 100;

                break;

            default:

                this.result = parseFloat(expression);

                break;

            }

            this.errorMessage = '';

        }

        else if (expression.length != 0) {

            console.log(expression);

            this.errorMessage = 'Incorrect input expression';

        }

        return this.result;

    }

    parseFunc(expression) {

        let res = 1;

        expression = expression.replace(' ', '');

        if (expression.search(/ log / i) != -1) {

            res = Math.log10(parseFloat(expression.replace('log(', '')));

        }

        else if (expression.search(/ ln / i) != -1) {

            res = Math.log(parseFloat(expression.replace('ln(', '')));

        }

        else if (expression.search(/ cos / i) != -1) {

            res = Math.cos(parseFloat(expression.replace('cos(', '')));

        }

        else if (expression.search(/ sin / i) != -1) {

            res = Math.sin(parseFloat(expression.replace('sin(', '')));

        }

        else if (expression.search(/ ^ / i) != -1) {

            res = expression.substring(0, expression.indexOf('^'));

            res = Math.pow(res, expression.substring(expression.indexOf('^') + 2, expression.length));

        }

        if (expression.search(/ sqrt / i) != -1) {

            res = Math.sqrt(parseFloat(expression.replace('sqrt(', '')));

        }

        return res.toString();

    }

}

view / ResultView.js

export default class ResultView {

    constructor(item) {

        this.item = item;

    }

    setValue(result) {

        this.item.textContent = result;

    }

}