

МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Проектування та розробка наукового калькулятора

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи КБ-20-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»



(підпис)

Максюта Роман
Олегович
(прізвище, ім'я, по батькові)

Науковий керівник



(підпис)

Погореловська Ірина
Дем'янівна, к.е.н.,
доцент
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Керівник освітньо-
професійної програми



(підпис)

Омельчук Антон
Анатолійович, к.т.н.
(прізвище, ім'я, по батькові, науковий
ступінь, вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних
технологій і систем протокол від «22» травня 2024 № 15

Завідувач кафедри



(підпис)

Філоненко Михайло
Миколайович, к.ф.-м.н.,
доцент
(прізвище, ім'я, по батькові,
науковий ступінь, вчене звання)

Реєстрація № 32

27.05.2024

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту

(до захисту, на доопрацювання)

22.05.2024

(дата)



(підпис)

Результат захисту

4 (добре)

(оцінка)

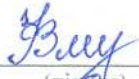
07.06.2024

(дата)



(підпис)

Голова комісії:



(підпис)

В.М. Франчук

Члени комісії:



(підпис)

М.М. Філоненко



(підпис)

С.М. Грищенко



(підпис)

В.О. Ніжегородцев

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	22		07.06.2024 Франчук В.М.  Філоненко М.М.  Грищенко С.М.  Ніжегородцев В.О.  Суховерхівка Л.П. 
2. Дотримання вимог щодо оформлення	від 0 до 10	6		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	3		
4. Рецензент	від 0 до 5	4		
РАЗОМ БАЛІВ	від 0 до 50	37		

Оцінювання кваліфікаційної роботи науковим керівником - 38 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
75	C	4 (добре)	07.06.2024	Франчук В.М.  Філоненко М.М.  Грищенко С.М.  Ніжегородцев В.О.  Суховерхівка Л.П. 

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую

Завідувач кафедри  Філоненко
М.М.

«23» жовтня 2023

**Завдання на кваліфікаційну роботу
здобувачу вищої освіти**

Максюті Роману Олеговичу
(прізвище, ім'я, по батькові)

Тема кваліфікаційної роботи: Проектування та розробка наукового калькулятора

затверджена наказом ректора від «20» жовтня 2023 № 1411

1. Строк здачі закінченої роботи «20» травня 2024

Мета роботи: розробка програмного забезпечення для наукового калькулятора

Об'єкт дослідження: технології розробки наукового калькулятора

Предмет дослідження: програмне забезпечення для реалізації функцій наукового калькулятора

Орієнтовна база дослідження: методи, технології, засоби проектування архітектури програмного забезпечення та реалізації функціональності онлайн калькулятора

Вихідні дані до роботи: вимоги до функціоналу наукового калькулятора

Перелік графічного матеріалу: лістинги коду, копії екранів результатів презентація роботи

2. Зміст кваліфікаційної роботи (перелік питань за кожним розділом):

Розділ 1. Теоретичні аспекти розробки наукового калькулятора. 1.1. Основні функції онлайн калькулятора. 1.2. Формування вимог до онлайн-калькулятора

Розділ 2. Технології розробки програмного забезпечення онлайн-калькулятора. 2.1. Аналітичний огляд технологій розробки програмного забезпечення онлайн калькулятора. 2.2. Аналітичний огляд існуючих конструкторів онлайн калькулятора.

Розділ 3. Практична реалізація онлайн калькулятора.

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактич о
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.10.2023	До 15.10.2023
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	29.12.2023	29.12.2023
3	II розділ (аналіз експериментальної частини про методики досліджень)	15.02.2024	15.02.2024
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	15.04.2024	15.04.2024
5	Висновки	01.05.2024	01.05.2024
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	06.05.2024	06.05.2024
7	Отримання відгуку керівника	28.05.2024	28.05.2024
8	Отримання зовнішньої рецензії	30.05.2024	30.05.2024

3. Календарний план виконання роботи

4. Дата видачі завдання «23» жовтня 2023

5. Керівник кваліфікаційної роботи


(підпис)

Погореловська І

(прізвище, ініціали)

6. Керівник освітньо-професійної програми


(підпис)

Омельчук А. А.

(прізвище, ініціали)

7. Завдання до виконання отримав
здобувач вищої освіти


(підпис)

Максюта Р. О.

(прізвище, ініціали)

АНОТАЦІЯ

Максюта Р.О. Проектування та розробка наукового калькулятора

Мета роботи: Розробка програмного забезпечення наукового калькулятора Завдання, які були виконання для досягнення визначеної мети:

- проаналізовано теоретичні аспекти розробки наукового калькулятора;
- **сформовано вимоги до функціоналу та дизайну онлайн-калькулятора;**
- **здійснено аналітичний огляд технологій розробки програмного забезпечення онлайн калькулятора;**
- **здійснено аналітичний огляд існуючих конструкторів онлайн калькулятора;**
- виконано практичну реалізацію **онлайн калькулятора.**

Об'єкт дослідження: технології розробки наукового калькулятора

Предмет дослідження: програмне забезпечення для реалізації функцій наукового калькулятора

Інформаційною базою досліджень є методи, технології, засоби проектування архітектури програмного забезпечення та реалізації функціональності онлайн калькулятора.

Науково-практична значимість одержаних результатів: кваліфікаційна робота містить як теоретичні аспекти проектування та розробки онлайн калькулятора, так і конкретні результати її виконання — розроблений науковий онлайн калькулятор.

Дипломна робота містить 55 сторінок, 1 таблицю, 46 рисунків, список використаних джерел з 26 найменувань.

Ключові слова: веб-розробка, онлайн калькулятор, програмне забезпечення, конструктор, фреймворки.

Рік виконання кваліфікаційної роботи 2024

Рік захисту роботи 2024

ABSTRACT

Maksyuta R.O, Design and development of a scientific calculator

Purpose of work: Development of scientific calculator software

Tasks that were performed to achieve the specified goal:

- the theoretical aspects of the development of a scientific calculator were analyzed;
- the requirements for the functionality and design of the online calculator were formed;
- an analytical review of online calculator software development technologies was carried out;
- an analytical review of existing online calculator designers was carried out;
- the practical implementation of the online calculator was completed.

Object of research: technologies of scientific calculator development

The subject of research: software for implementing the functions of a scientific calculator

The research information base is methods, technologies, means of designing the software architecture and implementing the functionality of the online calculator.

Scientific and practical significance of the obtained results: the qualification work contains both theoretical aspects of designing and developing an online calculator, as well as specific results of its implementation — a developed scientific online calculator.

The thesis contains 55 pages, 1 table, 46 figures, a list of used sources with 26 names.

Keywords: web development, online calculator, software, designer, frameworks.

The year of performance of qualification work is 2024 The year of job protection is 2024

ЗМІСТ

ВСТУП.....	3
РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ НАУКОВОГО ОН-ЛАЙН КАЛЬКУЛЯТОРА	5
1.1. Онлайн калькулятор як система управління контентом (CMS)	5
1.2. Функціональні вимоги до калькулятора.....	7
РОЗДІЛ 2. ТЕХНОЛОГІЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОНЛАЙН КАЛЬКУЛЯТОРА	12
2.1. Аналітичний огляд технологій розробки програмного забезпечення онлайн калькулятора.....	12
2.2. Аналітичний огляд існуючих конструкторів онлайн калькулятора.....	17
РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ОНЛАЙН КАЛЬКУЛЯТОРА	23
ВИСНОВКИ.....	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	55

ВСТУП

Актуальність теми "Проектування та розробка наукового калькулятора" полягає у необхідності створення потужного і функціонального інструменту для наукових обчислень, який відповідає сучасним вимогам і впроваджує передові технології. У сучасному світі, де наукові дослідження та технічні розрахунки є невід'ємною частиною багатьох галузей, науковий калькулятор відіграє важливу роль у забезпеченні точних обчислень та аналізу даних.

Стан і ступінь розробки проблеми в спеціальній літературі свідчать про наявність досліджень та розробок в галузі наукових калькуляторів. Аналіз літературних джерел показує, що існують різні підходи до проектування та розробки наукових калькуляторів, включаючи використання різних алгоритмів, методів оптимізації та рішень інтерфейсу. Однак, існує потреба у подальшому дослідженні та вдосконаленні таких калькуляторів, зокрема у напрямку покращення їх функціональності, ефективності та зручності використання.

Метою роботи є розробка програмного забезпечення для наукового калькулятора

Завдання, які були виконані для досягнення визначеної мети:

- проаналізовано теоретичні аспекти розробки наукового калькулятора;
- сформовано вимоги до функціоналу та дизайну онлайн-калькулятора;
- здійснено аналітичний огляд технологій розробки програмного забезпечення онлайн калькулятора;
- здійснено аналітичний огляд існуючих конструкторів онлайн калькулятора;
- виконано практичну реалізацію онлайн калькулятора.

Об'єктом дослідження є технології розробки наукового калькулятора.

Предмет дослідження: програмне забезпечення для реалізації функцій наукового калькулятора

Методи дослідження включають аналіз наукової літератури, вивчення існуючих рішень у галузі наукових калькуляторів, математичне моделювання, розробку та реалізацію алгоритмів і програмного забезпечення, проведення експериментів для оцінки функціональності та ефективності калькулятора.

Інформаційною базою досліджень є методи, технології, засоби проєктування архітектури програмного забезпечення та реалізації функціональності онлайн калькулятора.

Науково-практична значимість одержаних результатів: кваліфікаційна робота містить як теоретичні аспекти проєктування та розробки онлайн калькулятора, так і конкретні результати її виконання — розроблений науковий онлайн калькулятор.

Апробація результатів роботи: за результатами досліджень було підготовлено тези доповіді «Особливості розробки калькулятора для сайту», які опубліковані у збірнику тез І-ї Всеукраїнської конференції молодих вчених «Математичні методи та цифрові технології в економіці», яка відбулася у Державному податковому університеті 15.03.2024 [25]

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ НАУКОВОГО ОН-ЛАЙН КАЛЬКУЛЯТОРА

1.1. Основні функції онлайн калькулятора

“Калькулятор на веб-сайті — це окремий програмний модуль, який забезпечує додаткову функціональність для відвідувачів сайту, зокрема спрощує та пришвидшує комінікацію, сприяє формуванню користувацького досвіду, покращує відвідуваність сайту та його конверсію” [25]. Найбільш популярними функціями онлайн калькуляторів є:

Розрахунок вартості: калькулятори можуть допомогти користувачам оперативно оцінити вартість товарів або послуг, які вони хочуть придбати. Наприклад, калькулятор вартості доставки, кредитний калькулятор розрахунку щомісячних платежів, калькулятор вартості послуг з ремонту або будівництва тощо.

Планування та бюджетування: калькулятори можуть допомогти користувачам у плануванні своїх фінансів або бюджетуванні. Наприклад, калькулятори розрахунку витрат на подорожі, на щоденні витрати, на іпотеку тощо.

Генерація лідів (англ. Lead, потенційний клієнт): калькулятори можуть бути ефективним інструментом для генерації лідів. Зокрема, при використанні калькулятора користувач може отримати пропозицію підписатися на розсилку, придбати пробну версію продукту, отримати запрошення на вебінар, або в обмін на результати розрахунків залишити свої контактні дані.

Навчальні цілі: калькулятори можуть використовуватися для освітніх цілей, щоб допомогти користувачам зрозуміти складні математичні або фінансові концепції. Наприклад, калькулятор кредитного рейтингу або калькулятор інвестицій.

Науковий онлайн-калькулятор - це веб-додаток, який дозволяє користувачам швидко виконувати складні математичні обчислення та здійснювати різні арифметичні операції на веб-сайті. Він може бути використаний для простих обчислень, таких як додавання та віднімання чисел, або для складніших функцій, таких як обчислення виразів з

використанням тригонометрії, логарифмів або степенів. Онлайн-калькулятори дозволяють швидко та точно робити обчислення, заощаджуючи час і зусилля. Деякі онлайн-калькулятори надають графічні представлення математичних концепцій, що допомагає кращому їх розумінню та засвоєнню.

Основні властивості онлайн калькулятора:

1. Доступність: онлайн-калькулятори можуть бути доступні з будь-якого пристрою, який має підключення до Інтернету, такого як комп'ютер, смартфон або планшет. Це дозволяє користувачам використовувати калькулятор у будь-якому місці і в будь-який час.
2. Зручність: онлайн-калькулятори зазвичай мають простий та інтуїтивно зрозумілий інтерфейс, що робить їх легкими у використанні. Більшість з них надають кнопки для виконання різних арифметичних операцій, а також можуть мати додаткові функції, такі як збереження результатів чи конвертація одиниць вимірювання.
3. Розширені можливості: деякі онлайн-калькулятори можуть мати розширені можливості, які дозволяють виконувати складніші математичні операції або використовувати спеціальні функції. Наприклад, деякі калькулятори можуть мати вбудовану підтримку для розрахунку відсотків, обчислення квадратного кореня або виконання конвертацій валют.
4. Придатність для навчання: онлайн-калькулятори можуть бути корисними для навчання та викладання математики. Вони можуть надати візуальне представлення різних математичних концепцій та допомогти користувачам зрозуміти, як працюють різні операції.
5. Кросплатформність, тобто сумісність з різними операційними системами і архітектурами процесорів забезпечується.

Однак, важливо пам'ятати, що онлайн-калькулятори мають свої обмеження. Вони зазвичай базуються на певних алгоритмах та припущеннях і можуть бути чутливими до помилок або недоліків у програмуванні. Також

варто звернути увагу на безпеку при використанні онлайн-калькуляторів, особливо при обробці чутливої інформації, такої як фінансові дані [17].

Завдяки доступності та зручності використання онлайн-калькулятори стали незамінними інструментами для багатьох людей і професійних груп, включаючи студентів, вчителів, фінансових аналітиків, інженерів та багатьох інших.

Незалежно від мети використання, онлайн-калькулятори є потужними інструментами, які полегшують наше повсякденне життя та сприяють розвитку навичок розуміння й використання математики. Тобто, онлайн-калькулятори стали невід'ємною частиною нашого цифрового світу.

1.2. Формування вимог до онлайн-калькулятора

Проектування і впровадження калькулятора, як програмного забезпечення, відбувається в кілька етапів:

1) Обговорення проєкту з клієнтом. Визначення функціональних вимог.

Фахівці аналізують дані, які беру участь в розрахунках, групують всі змінні з метою коректного визначення завдань і підготовки технічного завдання.

2) Складання докладного технічного завдання.

Це один із найважливіших етапів, який визначає перелік майбутніх робіт. Для того щоб калькулятор коректно функціонував, розробники повинні чітко розуміти, які формули повинні застосовуватися при розрахунках.

3) Створення прототипу калькулятора та проектування дизайну.

У прототипі виділяються ключові елементи калькулятора, після чого виконується їх візуалізація для кращого розуміння.

4) Виконується автоматизація обчислень та тестування.

5) Здійснюється інтеграція калькулятора на сайт.

Розроблений програмний модуль має бути органічною частиною веб-ресурсу, на якому його планується розмістити, вписатися в його структуру та оформлення [2].

Відповідно до поставленої теми, сформулюємо такі вимоги:

1. Калькуляторні функції:

- Додавання, віднімання, множення та ділення чисел.
 - Обчислення квадратного кореня, кубічного кореня, логарифму, натурального логарифму та інших математичних функцій.
 - Розрахунок факторіала числа.
 - Піднесення числа до деякого степеня.
 - Обчислення тригонометричних функцій (синус, косинус, тангенс) в радіанах або градусах залежно від вибраного режиму.
2. Вибір режиму калькулятора:
- Можливість перемикавання між звичайним і науковим режимами калькулятора.
 - В звичайному режимі доступні лише базові математичні операції, а в науковому режимі - додаткові функції і операції.
3. Робота зі змінними:
- Збереження результату попереднього обчислення для використання в подальших розрахунках.
 - Використання збереженого результату за допомогою кнопки "ANS" (попередня відповідь).
4. Введення та виведення даних:
- Введення чисел та математичних операцій за допомогою кнопок калькулятора.
 - Відображення введених операцій та результату на екрані калькулятора.
5. Керування кутовою одиницею:
- Перемикавання між режимом радіанів та градусів.
 - Відображення активного режиму на екрані калькулятора.
6. Структура та оформлення:
- Розташування кнопок калькулятора в рядках та стовпцях.
 - Відображення активного режиму калькулятора.
7. Вимоги до дизайну:

- Дизайн додатку повинен бути консервативним – використовувати неяскаві, пастельні кольори і тони;
- Дизайн додатку має бути лаконічним, стильним та сучасним.

8. Вимоги до шрифтового оформлення додатку

- Основними шрифтовими гарнітурами є лише гарнітури з стандартних «безпечних» шрифтів.
- Шрифти, які використовуються для оформлення графічних елементів сайту. Не повинні суперечити корпоративному стилю компанії.
- Розмір шрифтів повинен забезпечувати зручність сприйняття тексту при мінімально допустимому розмірі екрану.

9. Браузери

- Веб-додаток повинен забезпечувати коректне відображення даних в наступних браузерах:
- Opera (версія 10.0i вище);
- Safari (версія 4.0 і вище);
- Mozilla Firefox (версія 3.0 і вище);
- Google Chrome;
- Microsoft Edge.

Ці вимоги визначають функціональні можливості, взаємодію з користувачем та зовнішній вигляд веб-додатку калькулятора.

Нефункціональні вимоги описують характеристики системи, які не стосуються безпосередньо її функціональності, але впливають на якість, надійність, продуктивність та інші аспекти системи [4]. Нefункціональні вимоги можна сформулювати так:

- Ефективність: Код повинен бути оптимізований для досягнення максимальної продуктивності та швидкодії обчислень.
- Надійність: Код повинен бути стабільним та надійним, не схильним до помилок або аварійного завершення.

- Сумісність: Код повинен бути сумісним з різними веб-браузерами та здатним працювати на різних платформах.
- Безпека: Код повинен бути захищеним від потенційних загроз безпеки, таких як вразливості, атаки злому або витоки даних.
- Читабельність: Код повинен бути добре структурованим, мається на увазі зрозумілим та документованим, щоб забезпечити легку читабельність та розуміння для інших розробників.
- Тестованість: Код повинен бути написаний з урахуванням можливості легкої автоматизованої перевірки та тестування, для забезпечення якості продукту.
- Масштабованість: Код повинен бути гнучким та масштабованим, здатним до розширення та керування зростанням обсягу даних або функціональності.
- Документація: Код повинен бути добре задокументованим, з вказівками щодо використання, функцій, залежностей та інших важливих аспектів.

Водночас, розробка професійних калькуляторів для сайтів залишається актуальною і потребує знання та розуміння всіх особливостей етапів розробки веб-додатків. Розробка онлайн-калькулятора складається з наступних етапів: “визначення вимог до функціональності та інтерфейсу; проєктування інтерфейсу користувача; вибір технологій (мови програмування, фреймворки, бази даних тощо), які найкраще відповідають вимогам проєкту; розробку фронтенд частини калькулятора, яка реалізуватиме логіку обчислень та взаємодію з користувачем; розробку бекенд частини для обробки запитів користувачів, зберігання даних або виконання складних операцій; тестування калькулятора для перевірки його працездатності, коректності обчислень та відповідності вимогам, захищеності від поточних загроз та збереження конфіденційності користувачів; оптимізація та вдосконалення після тестування; розгортання

калькулятора на сервері або хмарній платформі, а також його подальша підтримка чи додавання нового функціоналу” [23].

РОЗДІЛ 2. ТЕХНОЛОГІЇ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОНЛАЙН КАЛЬКУЛЯТОРА

2.1. Аналітичний огляд технологій розробки програмного забезпечення онлайн калькулятора

До аналітичного огляду технології розробки веб-додатків входить огляд різних аспектів, таких як мови програмування, фреймворки, засоби розробки та інфраструктури. Ось кілька ключових технологій, які широко використовуються в розробці веб-додатків:

- **“HTML (HyperText Markup Language)** є основною мовою розмітки, яка використовується для створення веб-сторінок. Вона визначає структуру та семантику контенту, що дозволяє браузерам правильно інтерпретувати та відображати цей контент для користувачів. HTML є фундаментом веб-розробки та виступає основою для впровадження інших технологій, таких як CSS і JavaScript. Один з основних принципів HTML - це використання тегів. Теги є елементами розмітки, які визначають різні типи контенту на веб-сторінці. Кожен тег має свою семантику та функціональність. Наприклад, `

` використовується для відображення абзаців тексту, `

`-`` - для заголовків різного рівня, `` - для вставки зображень і т.д. Один з сильних боків HTML - це його семантика. HTML надає спеціальні семантичні теги, які допомагають пояснити смислову структуру контенту. Наприклад, `` використовується для визначення заголовку сторінки або розділу, `` - для навігаційного меню, `` - для підвалу сторінки і т.д. Використання семантичних тегів не тільки поліпшує доступність веб-сторінок для користувачів з обмеженими можливостями, але й сприяє зрозумілості коду для розробників та пошукових систем. HTML також надає можливості для створення посилань, форм, таблиць, списків, вкладок і багатьох інших елементів інтерфейсу. Завдяки HTML можна створювати інтерактивні веб-

сторінки, які взаємодіють з користувачем та надають зручність використання. Одним з важливих аспектів використання HTML є його сумісність з різними браузерами. Хоча раніше були розбіжності в тому, як браузери інтерпретують HTML, сучасні стандарти та специфікації HTML допомагають забезпечити єдину та зрозумілу інтерпретацію тегів. Однак, розробники повинні бути уважними та використовувати правильний синтаксис та структуру HTML, щоб забезпечити, що їхні веб-сторінки працюють на різних браузерах” [12].

- “CSS (Cascading Style Sheets) є мовою стилізації, яка використовується для оформлення та вигляду веб-сторінок. Вона працює у поєднанні з HTML та іншими технологіями, дозволяючи розробникам контролювати вигляд та макет елементів на сторінці. Одним з головних завдань CSS є забезпечення розділення змісту та представлення на веб-сторінці. Використовуючи CSS, розробники можуть встановлювати різні стилі, такі як кольори, шрифти, розміри, розташування і фонові зображення для елементів HTML. Завдяки CSS можна змінювати вигляд сторінки цілком або окремих елементів з легкістю. Це дозволяє створювати привабливі та професійні веб-дизайни. CSS працює на основі принципу каскаду. Це означає, що стилі можуть бути визначені на різних рівнях, і вони будуть застосовуватися в залежності від їх пріоритетності та специфічності. Наприклад, ви можете встановити загальні стилі для всіх абзаців на сторінці, а потім перевизначити їх для конкретного абзацу з використанням ідентифікатора чи класу. Цей принцип надає велику гнучкість та контроль над виглядом елементів на сторінці. CSS також надає різноманітні можливості для розмітки та позиціонування елементів на сторінці. Зокрема, можна використовувати різні типи позиціонування, такі як статичне, абсолютне, відносне та фіксоване, щоб контролювати положення елементів на сторінці. Крім того, CSS дозволяє розміщувати елементи у різних колонках, використовувати гнучкі сітки та вирівнювати елементи по центру чи по

краю. CSS також підтримує анімацію та переходи, що дозволяють створювати рухливі та привабливі ефекти на веб-сторінках. За допомогою ключових кадрів та перехідних ефектів можна створювати плавні зміни стилів або положення елементів, що додає динамічності та взаємодії до веб-додатків. Одним з важливих аспектів CSS є його можливість для адаптивного дизайну. CSS надає засможливості для створення реактивного та адаптивного веб-дизайну, що дозволяє сторінкам пристосовуватися до різних пристроїв та екранних розмірів. За допомогою медіа-запитів та різних розміток елементів, розробники можуть створювати веб-додатки, які оптимально відображаються на різних пристроях, таких як комп'ютери, планшети або смартфони. CSS також має велику кількість вбудованих функцій та можливостей для розширення. Існує багато CSS-фреймворків, таких як Bootstrap чи Foundation, які надають готові компоненти та стилі для швидкого розроблення веб-додатків. Крім того, CSS можна поєднувати з JavaScript для створення більш складних та інтерактивних функцій на сторінці. Однак, при використанні CSS також існують певні виклики та обмеження. Час виконання CSS може займати багато ресурсів, особливо при великому обсязі стилів або складних селекторах. Це може призводити до погіршення продуктивності веб-додатків, особливо на мобільних пристроях з обмеженими ресурсами. Крім того, підтримка CSS може відрізнятися в різних браузерях, що може вимагати додаткового коду для забезпечення сумісності. У підсумку, використання CSS в веб-додатках надає розробникам потужні засоби для стилізації та оформлення сторінок. Вона дозволяє контролювати вигляд та макет елементів на сторінці, створювати анімацію та переходи, а також реагувати на різні пристрої та розміри екранів. CSS є важливою складовою веб-розробки, яка допомагає створювати привабливі та функціональні веб-додатки” [15].

- **“JavaScript:** JavaScript є мовою програмування, яка використовується для додавання інтерактивності та динамічних функцій до веб-сторінок. Він може використовуватись для маніпулювання DOM (Document Object Model), взаємодії з користувачем, валідації форм, виконання AJAX-запитів та багатьох інших завдань. JavaScript надає можливості для реалізації динамічної інтеракції з користувачем, маніпулювання вмістом сторінок та взаємодії з сервером. JavaScript забезпечує широкі можливості для створення функціональних, інтерактивних та динамічних веб-додатків. Одним з головних аспектів використання JavaScript є можливість обробки подій. JavaScript дозволяє реагувати на дії користувача, такі як натискання кнопок, введення тексту або переміщення миші. За допомогою обробників подій, розробники можуть визначити, які дії повинні відбутися при виникненні певної події. Наприклад, можна створити функцію, яка буде виконуватися при натисканні на кнопку і змінювати вміст сторінки або відправляти дані на сервер. JavaScript також дозволяє маніпулювати вмістом сторінки та структурою документа. Це означає, що розробники можуть додавати, видаляти або змінювати елементи HTML та CSS на сторінці за допомогою JavaScript. Такий підхід дозволяє створювати динамічні ефекти, анімацію та змінювати вміст сторінки без перезавантаження всієї сторінки. Наприклад, можна динамічно змінювати зображення, текст або стилі елементів на сторінці. JavaScript також має вбудовані можливості для роботи з об'єктами та масивами даних. Розробники можуть створювати нові об'єкти, виконувати операції з ними, додавати та видаляти властивості. Масиви даних дозволяють зберігати та маніпулювати наборами значень. JavaScript також надає можливості для роботи з рядками, числами, датами та іншими типами даних. Одним з важливих аспектів JavaScript є можливість взаємодії з сервером. За допомогою технологій, таких як AJAX (асинхронний JavaScript та XML), розробники можуть взаємодіяти з сервером без

перезавантаження сторінки. Це дозволяє отримувати та відправляти дані на сервер асинхронно, що забезпечує більш плавну та швидку взаємодію з веб-додатком. Наприклад, можна використовувати бібліотеку jQuery або вбудовані можливості JavaScript для виконання AJAX-запитів і отримання або відправки даних на сервер без перезавантаження сторінки. JavaScript також підтримує роботу з кукі (cookies) та локальним сховищем (localStorage / sessionStorage). Це дозволяє зберігати дані на боці клієнта і використовувати їх пізніше. Наприклад, можна зберегти вибрану користувачем мову або налаштування в локальному сховищі, щоб використовувати їх при кожному завантаженні сторінки. JavaScript також підтримує роботу з регулярними виразами, що дозволяє виконувати швидкий та потужний пошук та заміну тексту за певними шаблонами. Регулярні вирази особливо корисні при валідації форм, обробці введення від користувачів або виконанні складних операцій з рядками. Одним з найбільш популярних веб-фреймворків, побудованих на JavaScript, є React. React дозволяє розробникам ефективно створювати складні інтерфейси користувача, розділяючи їх на компоненти, які можна повторно використовувати. React використовує віртуальний DOM (Document Object Model) для ефективного оновлення та маніпуляції структурою сторінок. JavaScript також дозволяє використовувати зовнішні бібліотеки та фреймворки для розширення можливостей мови. Наприклад, бібліотека D3.js дозволяє створювати складні та інтерактивні візуалізації даних, а бібліотека Axios допомагає здійснювати HTTP-запити до сервера. Узагалі, використання JavaScript у веб-додатках відкриває безліч можливостей для створення інтерактивних, динамічних та функціональних веб-додатків. Вона дозволяє розробникам створювати багатошарові та складні інтерфейси, взаємодіяти з сервером та здійснювати потужну обробку даних. Завдяки широкій підтримці та активній спільноті розробників,

JavaScript залишається однією з найпопулярніших мов програмування для веб-розробки” [11].

- Фреймворки: Фреймворки, такі як React, Angular та Vue.js, надають набір інструментів та бібліотек для розробки веб-додатків. Вони спрощують роботу зі створенням компонентів, керуванням станом додатку, маршрутизацією та взаємодією з API [9].
- Бекенд-технології: Для розробки веб-додатків потрібні також серверні технології. Наприклад, Node.js дозволяє використовувати JavaScript на серверній стороні, а фреймворки, такі як Express або Flask, надають засоби для створення серверних додатків та API [5].
- Бази даних: Для зберігання та керування даними веб-додатків використовуються різні бази даних, такі як MySQL, PostgreSQL, MongoDB та інші. Вони дозволяють зберігати, організовувати та отримувати доступ до даних, необхідних для роботи додатків.
- Хмарні сервіси: Хмарні сервіси, такі як Amazon Web Services (AWS), Microsoft Azure або Google Cloud Platform, надають інфраструктуру та послуги для розгортання, масштабування та керування веб-додатками. Вони дозволяють зосередитися на розробці додатку, не хвилюючись про інфраструктуру [6].

2.2. Аналітичний огляд існуючих конструкторів онлайн калькулятора

Аналітичний огляд існуючих конструкторів онлайн калькулятора дозволяє розглянути функціональність та можливості різних інструментів, які надаються для створення калькуляторів в Інтернеті. Ось кілька популярних конструкторів онлайн калькуляторів, які варто врахувати:

- Calconic: <https://www.calconic.com/> “Calconic є потужним конструктором калькуляторів, який дозволяє легко створювати різноманітні типи калькуляторів, включаючи цінові, фінансові, наукові та багато інших. Він надає інтуїтивний інтерфейс для створення

калькуляторів шляхом перетягування та використання готових блоків із заздалегідь визначеними функціями” [22]. Calconic також пропонує можливість налаштування дизайну калькулятора та інтеграцію з різноманітними платформами (рис 2.1.).

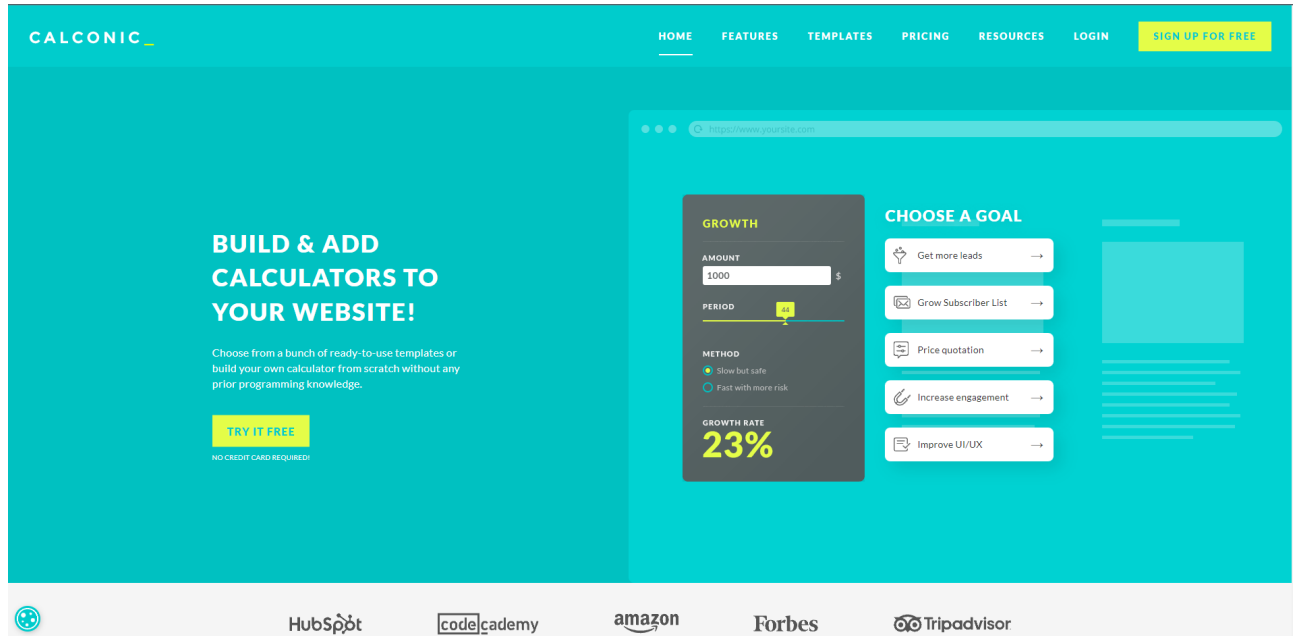


Рис. 2.1. Сервіс Calconic [22]

- Calculoid. “Calculoid є іншим зручним конструктором онлайн калькуляторів, який дозволяє швидко створювати інтерактивні калькулятори для різних цілей, таких як фінанси, нерухомість, спорт та інше. Цей конструктор надає багато вбудованих функцій, таких як математичні операції, формули, валідація даних та інтеграція з платіжними системами” [24]. Calculoid також пропонує зручний редактор для налаштування дизайну та вигляду калькулятора (Рис 2.2.).

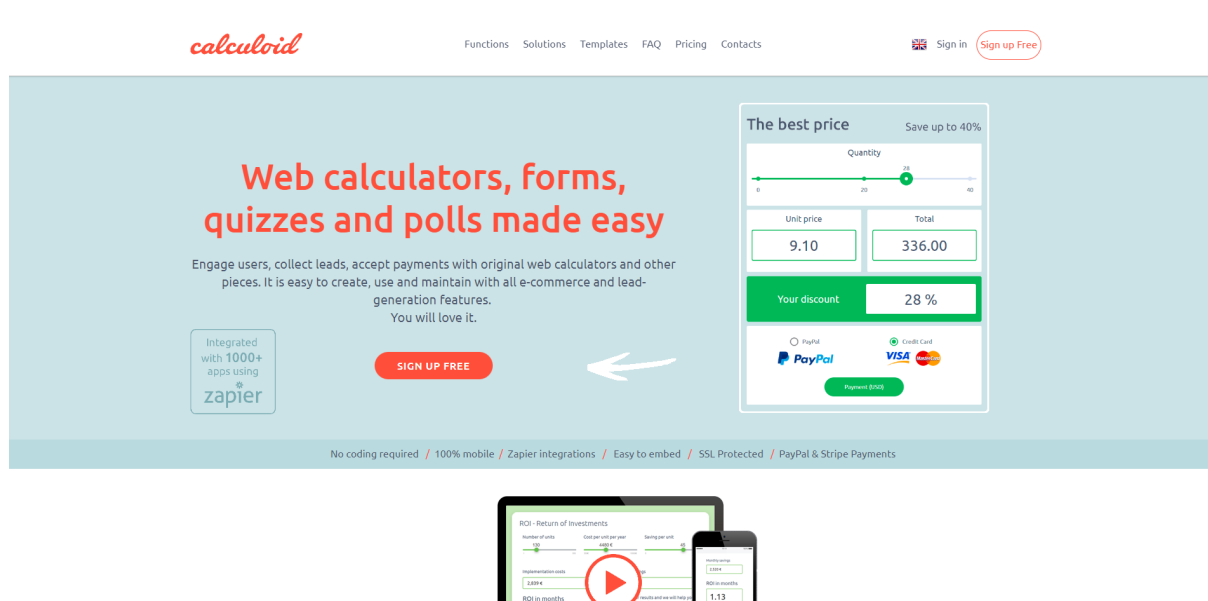


Рис. 2.2. Сервіс Calculoid [24]

- Zoho Creator. “Theo Zoho Creator є універсальною платформою розробки, яка дозволяє створювати не тільки калькулятори, але й повноцінні веб-додатки. <https://www.zoho.com/creator/index1.html> За допомогою Zoho Creator, розробники можуть створювати складні калькулятори зі зв'язками між різними полями, автоматизованими розрахунками та інтеграцією з іншими додатками” [23]. Ця платформа також надає можливості для налаштування вигляду, правильності та взаємодії калькулятора з користувачем (Рис 2.3.).

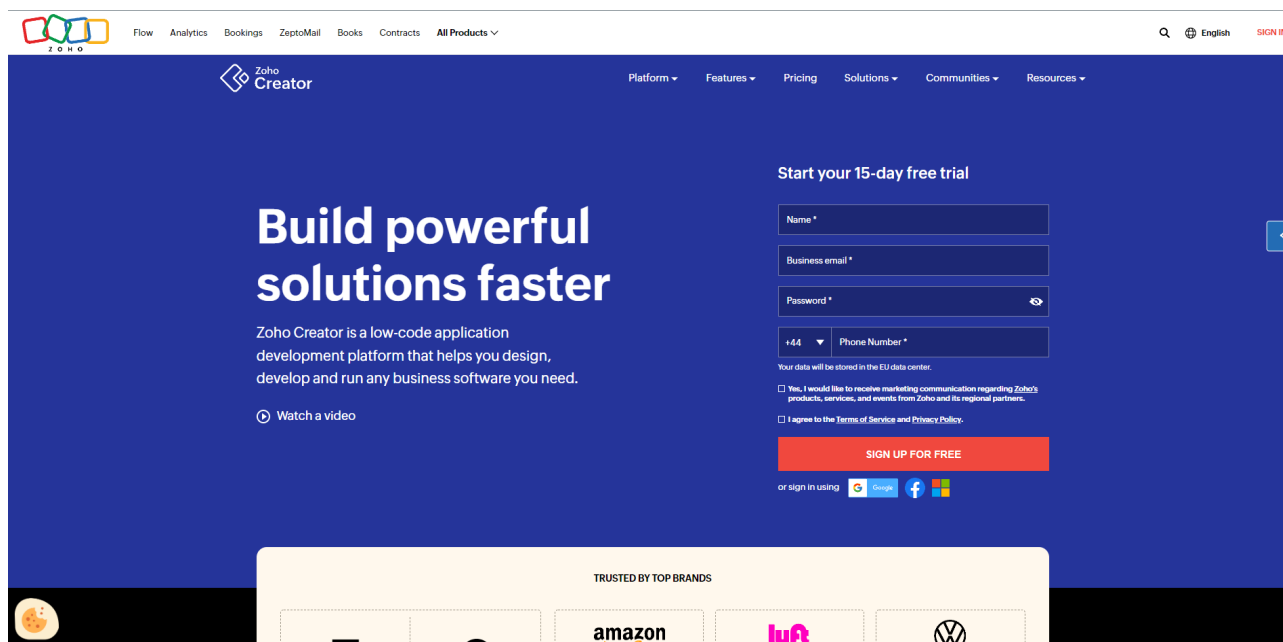


Рис. 2.3. Сервіс Zoho Creator [23]

- JSCalc.io: <https://jscalc.io>. “JSCalc.io є простим, але потужним конструктором калькуляторів, який дозволяє розробникам створювати настраювані калькулятори з використанням JavaScript. Цей конструктор надає можливість створювати різноманітні типи калькуляторів, включаючи фінансові, інженерні та інші” [25]. JSCalc.io також дозволяє налаштовувати зовнішній вигляд калькулятора та інтегрувати його на веб-сторінку (рис. 2.4.).

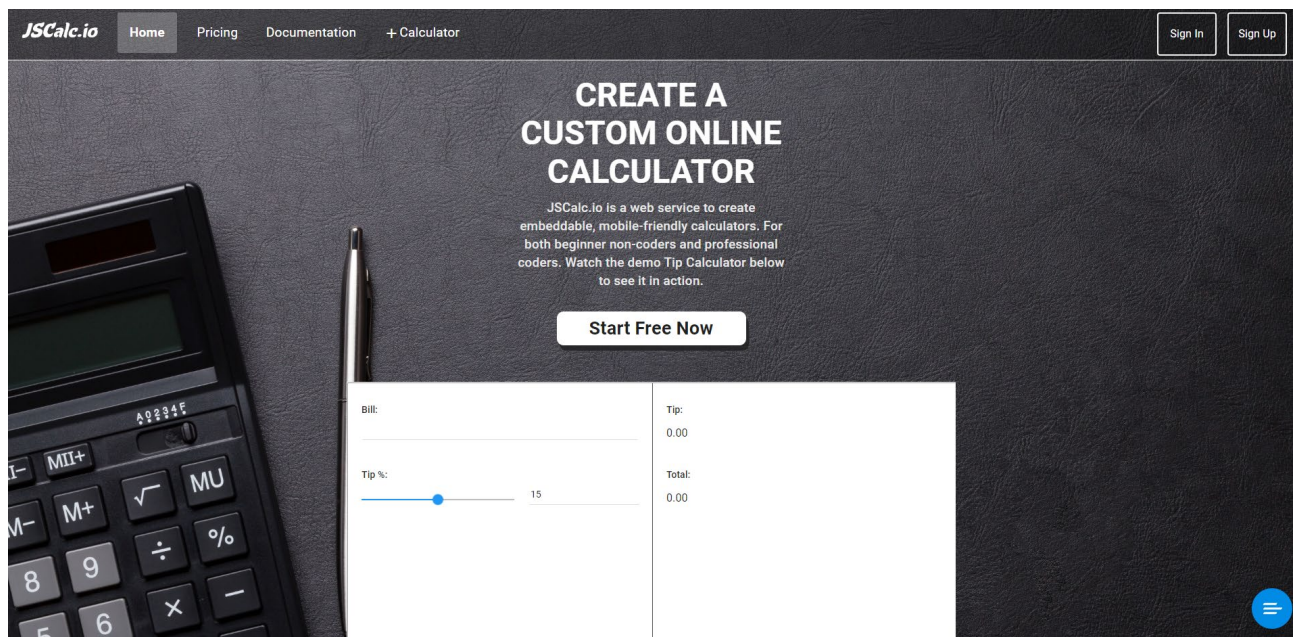


Рис. 2.4. Сервіс JSCalc.io [25]

Ці конструктори онлайн калькуляторів надають різні рівні функціональності та налаштувань. Деякі з них спрощують процес створення калькулятора за допомогою готових блоків та інтуїтивного інтерфейсу, тоді як інші дозволяють розробникам більше контролю та налаштувань, але можуть вимагати більшої технічної експертизи.

Перед вибором конструктора онлайн калькулятора варто врахувати такі фактори, як потреби вашого проекту, рівень технічних навичок, бюджет та можливості для масштабування та інтеграції з іншими системами.

Необхідно також звернути увагу на підтримку та оновлення конструктора, оскільки це може вплинути на забезпечення безпеки та функціональності вашого калькулятора. У кінцевому підсумку, вибір

функціонала онлайн калькулятора буде визначатися специфічними потребами проєкту та конкретними вимогами до функціональності, налаштувань та можливостей інтеграції.

Ці конструктори дають можливість людям без досвіду у веб-програмуванні створити свій власний калькулятор для веб-сайту, використовуючи готові елементи за допомогою візуального редактора. Однією з переваг такого підходу є простота і зручність використання, але часто ці конструктори обмежені у функціоналі і дозволяють створювати лише стандартні калькулятори з простими формулами розрахунку.

Порівняння можливостей JavaScript та фреймворків наведені у табл.2.1.

Вибір між JavaScript та фреймворками залежить від конкретних потреб проєкту. Якщо планується створити простий веб-додаток без складної логіки та великої кількості сторінок, то використання простого JavaScript може бути оптимальним вибором. Буде повний контроль над кодом і дизайном, і не буде потреб в додаткових залежностях.

Таблиця 2.1

Порівняння JavaScript та фреймворків за даними [11, 16, 17]

Використання JavaScript	Простота: JavaScript мова досить проста для вивчення, особливо для розробників, які вже мають досвід у програмуванні. Вона має простий синтаксис та може бути використана для широкого спектру завдань.
	Незалежність від фреймворків: JavaScript може бути використаний без фреймворків для створення простих веб-додатків. Він дозволяє розробникам мати повний контроль над дизайном та архітектурою додатку.
	Велика спільнота: JavaScript є однією з найпопулярніших мов програмування, тому має велику та активну спільноту розробників. Це означає, що завжди є доступ до ресурсів, документації та підтримки.

Використання фреймворків	Прискорена розробка: Фреймворки, такі як Angular, React та Vue.js, надають готові рішення для багатьох типових задач, таких як маршрутизація, управління станом та взаємодія з сервером. Вони спрощують розробку, зменшуючи потребу в написанні багатьох повторюваних частин коду.
	Стандартизація: Фреймворки можуть накладати стандарти та кращі практики на розробку, що полегшує співпрацю розробників та забезпечує однорідність коду.
	Велика спільнота: Популярні фреймворки мають велику та активну спільноту розробників. Це забезпечує доступ до багатьох ресурсів, бібліотек, модулів та підтримки.

Враховуючи все вищезазначене та потреби розроблюваного веб додатку, можна прийти до висновку, що використання JavaScript є найбільш актуальним, це обумовлено наступними факторами:

- Поглиблення та узагальнення навичок володіння JavaScript;
- Простота розробки;
- Оптимізація веб-додатку;
- Велика підтримка спільноти, тобто легке оновлення.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ОНЛАЙН КАЛЬКУЛЯТОРА

Для успішного конструювання та розробки інтерфейсу використаємо вже зарекомендовані золоті правила, запропоновані Theo Mandel, яких необхідно дотримуватися при оформленні інтерфейсу [18].

1. Надати користувачеві контроль:

- Визначте режими взаємодії таким чином, щоб не змушувати користувача до непотрібних або небажаних дій: користувач повинен бути в змозі легко увійти і вийти з режиму з невеликими або без зусиль.
- Забезпечити гнучку взаємодію: різні люди будуть використовувати різні механізми взаємодії, деякі можуть використовувати клавіатурні команди, деякі можуть використовувати мишу, деякі можуть використовувати сенсорний екран і т.д., отже, всі механізми взаємодії повинні бути забезпечені.
- Дозволити взаємодії з користувачем бути переривчастим і незмінним: коли користувач робить послідовність дій, користувач повинен мати можливість перервати послідовність, щоб зробити якусь іншу роботу без втрати виконаної роботи. Користувач також повинен мати можливість скасувати операцію.
- Оптимізувати взаємодію як прогрес рівня кваліфікації та дозволити налаштувати взаємодію: Просунутий або висококваліфікований користувач повинен мати можливість налаштувати інтерфейс, як хоче користувач, що дозволяє різні механізми взаємодії, щоб користувач не відчував себе нудно, використовуючи той же механізм взаємодії.
- Приховати технічні внутрішні дані від випадкових користувачів: користувач не повинен знати про внутрішні технічні деталі системи. Він повинен взаємодіяти з інтерфейсом тільки для того, щоб виконувати свою роботу.
- Дизайн для безпосередньої взаємодії з об'єктами, які з'являються на екрані: користувач повинен мати можливість використовувати об'єкти і

маніпулювати об'єктами, які присутні на екрані для виконання необхідного завдання. Таким чином, користувач відчуває себе легко контролювати екран.

2. Зменшити навантаження на пам'ять користувача:

- Зниження попиту на короткочасну пам'ять: коли користувачі беруть участь у деяких складних завданнях, попит на короткочасну пам'ять є значним. Так інтерфейс повинен бути розроблений таким чином, щоб зменшити запам'ятовування раніше зроблених дій, заданих входами і результатами.
- Встановити значущі значення за замовчуванням: Завжди початковий набір типових значень повинен бути наданий середньому користувачеві, якщо користувачеві потрібно додати деякі нові функції, то він повинен мати можливість додавати необхідні функції.
- Визначте інтуїтивно зрозумілі ярлики: Mnemonics повинна використовуватися користувачем. Mnemonics означає комбінації клавіш для виконання деяких дій на екрані.
- Візуальне розташування інтерфейсу має базуватися на реальній метафорі: Все, що ви представляєте на екрані, якщо це метафора реальної сутності, то користувачі легко зрозуміють.
- Розкривати інформацію прогресивно: інтерфейс повинен бути організований ієрархічно, тобто на головному екрані інформація про завдання, об'єкт або якусь поведінку повинна бути представлена спочатку на високому рівні абстракції.

3. Зробити інтерфейс послідовним:

- Дозволити користувачеві помістити поточну задачу в значущий контекст: багато інтерфейсів мають десятки екранів. Тому важливо послідовно надавати показники, щоб користувач знав про виконану роботу. Користувач також повинен знати, з якої сторінки перейшов на поточну сторінку і з поточної сторінки, куди можна перейти.

- Підтримувати узгодженість у сімействі додатків: розробка деяких програм повинна слідувати і впроваджувати той же дизайн, правила, щоб узгодженість підтримувалася серед додатків.
- Якщо минулі інтерактивні моделі створили очікування користувачів не вносять змін, якщо немає вагомої причини.

У цілому, дизайн інтерфейсу користувача є ключовим компонентом програмної інженерії, оскільки він може мати значний вплив на зручність використання, ефективність та досвід користувача програми. Інженери-програмісти повинні дотримуватися найкращих практик та принципів проектування для створення інтерфейсів, орієнтованих на користувача, послідовних, простих та доступних [14].

Підготуємо стек «Calc», який буде містити три основні папки:

- Css – стилі додатку;
- Img – зображення;
- Js – скрипти додатку;

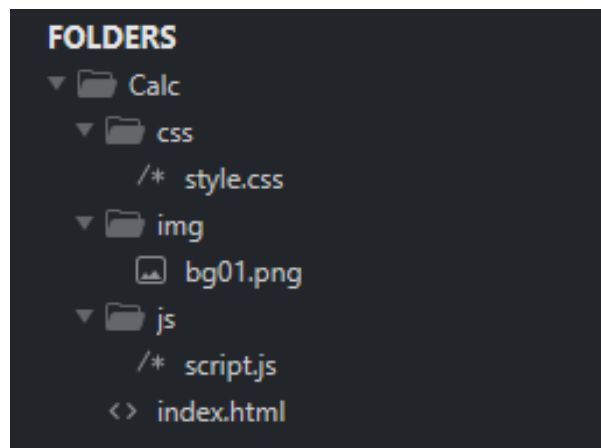


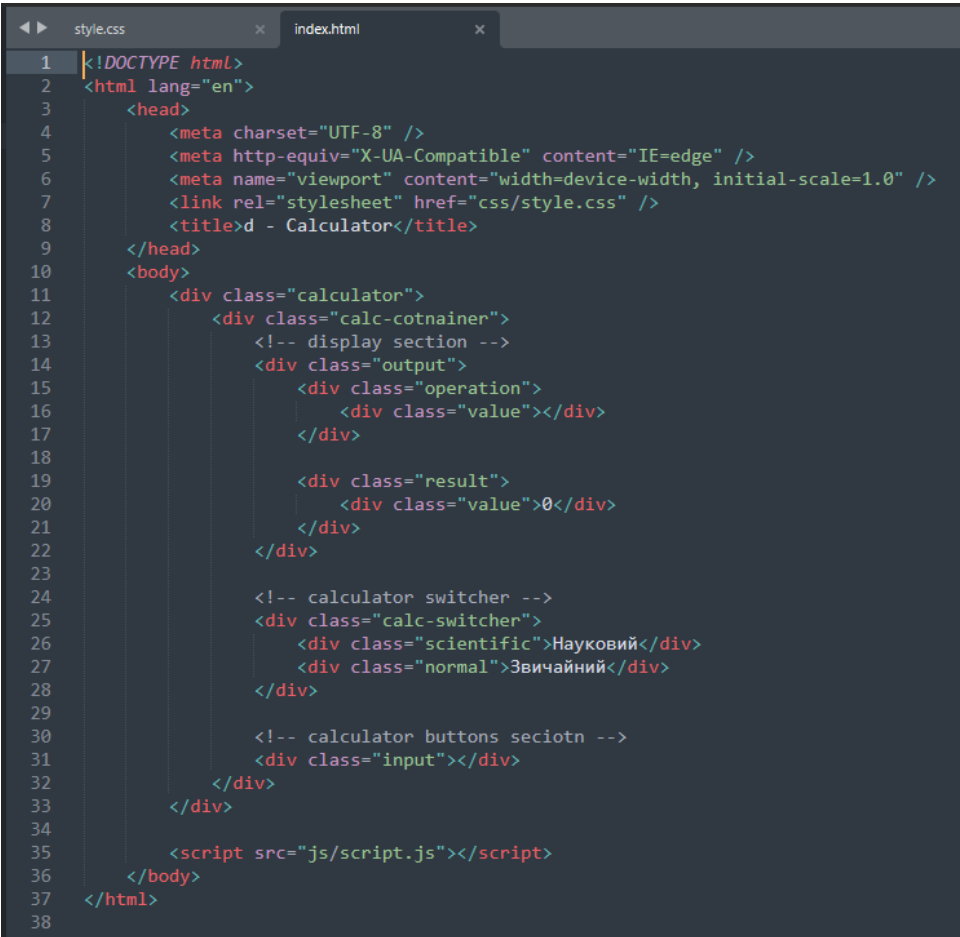
Рисунок 3.1 – Стек Calc

Першим кроком в реалізації веб додатку, буде створення файлу `index.html`, який буде створювати структуру веб сторінки калькулятора, функціонал якого буде реалізований за допомогою `js` а інтерфейс за допомогою `css` [15].

Файл `index.html` (Рис 3.2.) представляє собою структуру HTML-сторінки з калькулятором. Він містить основні теги HTML, такі як

`<!DOCTYPE>`, ``<html>``, ``<head>`` та ``<body>``, які формують структуру сторінки.

У ``<head>`` визначено метатеги, такі як кодування символів (``<meta charset="UTF-8">``), налаштування відображення на пристроях різних розмірів (``<meta name="viewport" content="width=device-width, initial-scale=1.0">``) та заголовок сторінки (``<title>Калькулятор</title>``). Також в цьому розділі підключено зовнішній файл стилів ``style.css`` за допомогою тегу ``<link rel="stylesheet" href="style.css">`'. Це означає, що стилі, які визначені у файлі ``style.css``, будуть застосовуватись до елементів сторінки.



```

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8" />
5     <meta http-equiv="X-UA-Compatible" content="IE=edge" />
6     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7     <link rel="stylesheet" href="css/style.css" />
8     <title>d - Calculator</title>
9   </head>
10  <body>
11    <div class="calculator">
12      <div class="calc-container">
13        <!-- display section -->
14        <div class="output">
15          <div class="operation">
16            <div class="value"></div>
17          </div>
18          <div class="result">
19            <div class="value">0</div>
20          </div>
21        </div>
22
23        <!-- calculator switcher -->
24        <div class="calc-switcher">
25          <div class="scientific">Науковий</div>
26          <div class="normal">Звичайний</div>
27        </div>
28
29        <!-- calculator buttons section -->
30        <div class="input"></div>
31      </div>
32    </div>
33
34    <script src="js/script.js"></script>
35  </body>
36 </html>
37
38

```

Рисунок 3.2 – Лістинг коду файлу index.html

У ``<body>`` розташований блок з класом ``calculator``, який містить внутрішній блок з класом ``calc-container`'. Це місце, де розміщується вміст калькулятора.

Нарешті, в ``<footer>`` розташований блок з текстом "Науковий | Звичайний", який, вказує на режими роботи калькулятора.

Отже, було реалізовано структуру майбутньої сторінки яку ще можна назвати каркасом.

```
let SCIENTIFIC_MODE = true;
const scientificMode = document.querySelector(".scientific");
const normalMode = document.querySelector(".normal");

scientificMode.classList.add("active-calc");
scientificMode.addEventListener("click", () => {
    SCIENTIFIC_MODE = true;
    scientificMode.classList.add("active-calc");
    normalMode.classList.remove("active-calc");
    disableAdvaceKey();
});

normalMode.addEventListener("click", () => {
    SCIENTIFIC_MODE = false;
    scientificMode.classList.remove("active-calc");
    normalMode.classList.add("active-calc");
    disableAdvaceKey();
});
```

Рис. 3.3. Лістинг коду реалізації переключення режимів

Блок коду (рис. 3.3.) відповідає за перемикання режиму калькулятора між науковим та звичайним. Змінна `SCIENTIFIC_MODE` визначає поточний режим (`true` – науковий режим, `false` – звичайний режим). При натисканні на елемент із класом `"scientific"` науковий режим включається, додається клас `"active-calc"` і викликається функція `disableAdvaceKey()`. При натисканні на елемент з класом `"normal"` звичайний режим вмикається, видаляється клас `"active-calc"` і також викликається функція `disableAdvaceKey()`.

```
const OPERATIONS = ["+", "-", "*", "/"];
const POWER = "POWER(";
const FACTORIAL = "FACTORIAL(";
const CUBE = "CUBE(";

let data = {
  operation: [],
  formula: [],
};
let ans = 0;
```

Рисунок 3.4 – Лістинг коду оголошення змінних та констант

У цьому блоці коду оголошуються деякі змінні та константи. OPERATIONS містить перелік операцій калькулятора. POWER, FACTORIAL і CUBE представляють рядки, які будуть використовуватися для формування математичних виразів [28].

Змінна data являє собою об'єкт, що містить масиви operation і formula, які будуть використовуватися для зберігання операцій і формул калькулятора відповідно. Змінна ans використовуватиметься для зберігання попереднього результату обчислень. Тепер необхідно створити кнопки, які будуть організувати функціонал веб-додатку.

```
40 let calculator_buttons = [
41   {
42     name: "rad",
43     symbol: "Rad",
44     formula: false,
45     type: "key",
46   },
47   {
48     name: "deg",
49     symbol: "Deg",
50     formula: false,
51     type: "key",
52   },
53   {
54     name: "cube",
55     symbol: "x³",
56     formula: POWER,
57     type: "math_function",
58   },
59   {
60     name: "cube-root",
61     symbol: "∛",
62     formula: "Math.cbrt",
63     type: "math_function",
64   },
65   {
66     name: "x-inverse",
67     symbol: "x⁻¹",
68     formula: POWER,
69     type: "math_function",
70   },
71   {
72     name: "quad",
73     symbol: "x⁴",
74     formula: POWER,
75     type: "math_function",
76   },
77   {
78     name: "factorial",
79     symbol: "x!",
80     formula: FACTORIAL,
81     type: "math_function",
82   },
83   {
84     name: "x-inverse",
85     symbol: "x⁻¹",
86     formula: POWER,
87     type: "math_function",
88   },
89   {
90     name: "quad",
91     symbol: "x⁴",
92     formula: POWER,
93     type: "math_function",
94   },
95   {
96     name: "factorial",
97     symbol: "x!",
98     formula: FACTORIAL,
99     type: "math_function",
100  },
101 ];
```

Рисунок 3.5 – Лістинг коду створення кнопок

У першому рядку визначається масив `calculator_buttons`, який містить об'єкти з інформацією про кожну кнопку. Кожен об'єкт представляє кнопку калькулятора і має наступні властивості:

- `name`: назва кнопки
- `symbol`: символ, що відображається на кнопці
- `formula`: формула, пов'язана з кнопкою (це може бути математична функція або оператор)
- `type`: тип кнопки (ключ, математична функція, число або оператор)

Наступні рядки містять функцію `createCalculatorBtns()`, яка використовує масив `calculator_buttons` для створення кнопок калькулятора в HTML-кодi. Функція проходить через кожен елемент масиву `calculator_buttons` і створює кнопку з відповідною назвою та символом. Кнопки додаються до відповідних рядків у HTML-структурі калькулятора [27].

Далі є функція `disableAdvanceKey()`, яка вимикає розширений режим калькулятора, якщо змінна `SCIENTIFIC_MODE` має значення `false`. Функція проходить через всі рядки з класом `"advance-keys"` (рядки, які містять кнопки розширеного режиму) і вимикає кнопки, які належать до розширеного режиму, якщо калькулятор працює в звичайному режимі. Якщо калькулятор працює в розширеному режимі, кнопки залишаються активними [26].

Загалом, цей код створює кнопки для калькулятора на основі заданого масиву `calculator_buttons` і надає можливість вимикати кнопки розширеного режиму в залежності від обраного режиму калькулятора (Рис 3.6.).

```

317 function createCalculatorBtns() {
318     let btnPerRow = 6;
319     let addedBtns = 0;
320
321     calculator_buttons.forEach((button) => {
322         if (button.name == "ANS") {
323             btnPerRow = 5;
324             addedBtns++;
325             const lastRow = document.querySelector(".row:last-child");
326             lastRow.style.marginBottom = "16px";
327             const allRow = document.querySelectorAll(".row");
328
329             allRow.forEach((row) => {
330                 row.classList.add("advance-keys");
331             });
332         }
333
334         if (addedBtns % btnPerRow == 0) {
335             inputElement.innerHTML += `<div class="row"></div>`;
336         }
337
338         const row = document.querySelector(".row:last-child");
339
340         row.innerHTML += `<button id="${button.name}">
341             ${button.symbol}
342         </button>`;
343
344         addedBtns++;
345     });
346 }
347 createCalculatorBtns();
348
349
350 function disableAdvaceKey() {
351     const advanceKey = document.querySelectorAll(".advance-keys");
352     advanceKey.forEach((key) => {
353         key.childNodes.forEach((btn) => {
354             if (!SCIENTIFIC_MODE) {
355                 btn.disabled = true;
356             } else {
357                 btn.disabled = false;
358             }
359         });
360     });
361     data.formula = [];
362     data.operation = [];
363     updateOutputOperation(data.operation.join(""));
364     updateOutputResult(0);
365 }

```

Рисунок 3.6 – Лістинг функціонування кнопок веб-додатку

```

let RADIAN = true;
const radBtn = document.getElementById("rad");
const degBtn = document.getElementById("deg");

radBtn.classList.add("active-angle");

function angleToggler() {
    radBtn.classList.toggle("active-angle");
    degBtn.classList.toggle("active-angle");
}

```

Рисунок 3.7. Лістинг коду встановлення змінних радіанів або градусів

У цьому фрагменті коду встановлюється змінна RADIAN зі значенням true, яка вказує, чи використовувати радіани чи градуси. Також отримуються посилання на кнопки з id "rad" та "deg" та присвоюються змінним radBtn та degBtn відповідно. На кнопці "rad" також додається клас "active-angle".

Функція angleToggler() використовується для перемикання між радіанами та градусами при натисканні на кнопки. Вона змінює класи кнопок "rad" та "deg", щоб відображати активний стан (Рис 3.7.).

```

inputElement.addEventListener("click", (e) => {
    const targetBtn = e.target;

    calculator_buttons.forEach((button) => {
        if (button.name == targetBtn.id) calculator(button);
    });
});

```

Рисунок 3.8 – Лістинг коду обробки подій кнопок

У цьому фрагменті коду створюється подія "click" для контейнера inputElement, яка відбувається при клацанні на кнопку в калькуляторі. Коли відбувається подія, визначається кнопка, на яку було клацнуто (targetBtn). За допомогою циклу forEach, перевіряється кожна кнопка в масиві calculator_buttons [27]. Якщо ім'я кнопки співпадає з id кнопки, на яку клацнули, викликається функція calculator() з цією кнопкою в якості аргументу (Рис 3.8.).

```

26 ▼ function calculator(button) {
27 ▼     if (button.type == "operator") {
28         data.operation.push(button.symbol);
29         data.formula.push(button.formula);
30 ▼     } else if (button.type == "number") {
31         data.operation.push(button.symbol);
32         data.formula.push(button.formula);
33 ▼     } else if (button.type == "trigo_function") {
34         data.operation.push(button.symbol + "(");
35
36         data.formula.push(button.formula);
37 ▼     } else if (button.type == "math_function") {
38         let symbol, formula;
39 ▼         if (button.name == "factorial") {
40             symbol = "!";
41             formula = button.formula;
42
43             data.operation.push(symbol);
44             data.formula.push(formula);
45 ▼         } else if (button.name == "power") {
46             symbol = "^(";
47             formula = button.formula;
48
49             data.operation.push(symbol);
50             data.formula.push(formula);
51 ▼         } else if (button.name == "square") {
52             showPowerOnUi(data, button.formula, 2);
53 ▼         } else if (button.name == "cube") {
54             showPowerOnUi(data, button.formula, 3);
55 ▼         } else if (button.name == "quad") {
56             showPowerOnUi(data, button.formula, 4);
57 ▼         } else if (button.name == "x-inverse") {
58             showPowerOnUi(data, button.formula, -1);
59 ▼         } else {
60             symbol = button.symbol + "(";
61             formula = button.formula + "(";
62
63             data.operation.push(symbol);
64             data.formula.push(formula);
65         }
66 ▼     } else if (button.type == "key") {
67 ▼         if (button.name == "clear") {
68             data.operation = [];
69             data.formula = [];
70

```

Рис. 3.9. Лістинг коду обробки подій кнопок

У цій функції calculator() обробляється натискання на кнопку калькулятора. Залежно від типу кнопки (оператор, число, тригонометрична

функція і т. д.), виконуються відповідні дії (Рис 3.9.). Наприклад, якщо кнопка є оператором, символ та формула кнопки додаються до масивів `data.operation` та `data.formula`.

Після обробки кнопки оновлюється відображення операції за допомогою функції `updateOutputOperation()`.

```

5  function powerBaseGetter(formula, powerIndexes) {
6      let powerBases = [];
7
8      powerIndexes.forEach((powerIndex) => {
9          let base = []; // current base
10         let previousIndex = powerIndex - 1;
11         let parenthesisCount = 0;
12
13         while (previousIndex >= 0) {
14             if (formula[previousIndex] == "(") parenthesisCount--;
15             if (formula[previousIndex] == ")") parenthesisCount++;
16
17             let isOperator = false;
18             OPERATIONS.forEach((operator) => {
19                 if (formula[previousIndex] == operator) isOperator = true;
20             });
21
22             let isPower = formula[previousIndex] == POWER;
23
24             if ((isOperator && parenthesisCount == 0) || isPower) break;
25
26             base.unshift(formula[previousIndex]);
27             previousIndex--;
28         }
29         powerBases.push(base.join(""));
30     });
31     return powerBases;
32 }
33

```

Рис.3.10 Лістинг коду функції `powerBaseGetter`

Функція «`powerBaseGetter(formula, powerIndexes)`» (Рис 3.10.) витягує бази степенів з формули. Вона приймає параметри `formula` (формула) - рядок з математичною формулою та `powerIndexes` (індекси степенів) - масив, який містить індекси символів степенів у формулі. Функція обходить формулу назад від кожного індексу степеня, витягує базу степеня та повертає масив баз степенів.

Функція «factorialNumGetter(formula, factorialIndexes)» (рис 3.11.) витягує числа для факторіала з формули. Вона приймає параметри formula (формула) - рядок з математичною формулою та factorialIndexes (індекси факторіалів) - масив, який містить індекси символів факторіалів у формулі. Функція обходить формулу від кожного індексу факторіала, витягує число перед факторіалом та повертає масив об'єктів, які містять інформацію про числа для заміни.

```

37 function factorialNumGetter(formula, factorialIndexes) {
38   let factNumbers = [];
39   let factSequence = 0;
40
41   factorialIndexes.forEach((factIndex) => {
42     let number = []; // current base
43     let nextIndex = factIndex + 1;
44     let nextInput = formula[nextIndex];
45
46     if (nextInput == FACTORIAL) {
47       factSequence++;
48       return;
49     }
50     let firstFactIndex = factIndex - factSequence;
51     let previousIndex = firstFactIndex - 1;
52     let parenthesisCount = 0;
53
54     while (previousIndex >= 0) {
55       if (formula[previousIndex] == "(") parenthesisCount--;
56       if (formula[previousIndex] == ")") parenthesisCount++;
57
58       let isOperator = false;
59       OPERATIONS.forEach((operator) => {
60         if (formula[previousIndex] == operator) isOperator = true;
61       });
62
63       if (isOperator && parenthesisCount == 0) break;
64
65       number.unshift(formula[previousIndex]);
66       previousIndex--;
67     }
68
69     let numberStr = number.join("");
70     const factorial = "factorial(";
71     const clsoeParenthesis = ")";
72     let times = factSequence + 1;
73
74     let toReplace = numberStr + FACTORIAL.repeat(times);
75     let replacement =
76       factorial.repeat(times) + numberStr + clsoeParenthesis.repeat(times);
77
78     factNumbers.push({
79       toReplace: toReplace,
80       replacement: replacement,
81     });
82     factSequence = 0;
83   });
84   return factNumbers;
85 }
86

```

Рисунок 3.11 – Лістинг коду функції factorialNumGetter

Функція «search(formula, keyword)» (рис 3.12.) шукає вказане ключове слово у формулі. Вона приймає параметри formula (формула) - рядок з

математичною формулою та keyword (ключове слово) - рядок, яке потрібно знайти. Функція реалізує формулу та повертає масив індексів, де знайдено ключове слово.

```

89  function search(formula, keyword) {
90      let searchResult = [];
91      formula.forEach((item, index) => {
92          if (item == keyword) searchResult.push(index);
93      });
94
95      return searchResult;
96  }
97

```

Рисунок 3.12 – Лістинг коду функції search

Функція `updateOutputOperation(operation)` оновлює відображення операції на користувацькому інтерфейсі. Вона приймає параметр `operation` (операція) - рядок, який потрібно відобразити.

```

88
89  function search(formula, keyword) {
90      let searchResult = [];
91      formula.forEach((item, index) => {
92          if (item == keyword) searchResult.push(index);
93      });
94
95      return searchResult;
96  }
97
98
99  function updateOutputOperation(operation) {
100      outputOperationElement.innerHTML = operation;
101  }
102
103
104  function updateOutputResult(result) {
105      outputResultElement.innerHTML = result;
106  }
107
108
109  function showPowerOnUi(data, formula, powerNum) {
110      data.operation.push("^(");
111      data.formula.push(formula);
112
113      data.operation.push(powerNum);
114      data.formula.push(powerNum);
115  }
116

```

Рисунок 3.13 – Лістинг коду функції оновлення даних в інтерфейсі

Функція «updateOutputResult(result)» оновлює відображення результату на користувацькому інтерфейсі. Вона приймає параметр result (результат) - рядок, який потрібно відобразити.

Функція «showPowerOnUi(data, formula, powerNum)» додає степінь до користувацького інтерфейсу. Вона приймає параметри data (дані) - об'єкт, який містить масиви operation (операції) та formula (формули), formula (формула) - рядок з математичною формулою та powerNum (степінь) - число, яке потрібно додати.

Функція «trigo(callback, angle)» (Рис 3.14.) виконує тригонометричну функцію. Вона (рис. 3.14) приймає параметри callback (Функція powerBaseGetter(formula, powerIndexes) використовується для отримання основи степеня з математичної формули. Вона приймає два параметри: formula - рядок, що містить формулу, і powerIndexes - масив, який містить індекси символів степеня у формулі. Функція обходить формулу від кожного індексу степеня назад, витягує основу степеня та повертає масив основ степеня.

```

118 function trigo(callback, angle) {
119     if (!RADIAN) {
120         angle = (angle * Math.PI) / 180;
121     }
122     return callback(angle);
123 }
124

```

Рисунок 3.14 – Лістинг коду функції trigo

Функція «inv_trigo(callback, value)» (рис 3.15.) також приймає два аргументи: callback (функція зворотного виклику) та value (значення). Усередині функції відбувається перевірка імені функції callback. Якщо ім'я не є "atan", відбувається додаткова перевірка значення value. Якщо значення менше -1 або більше 1, виводиться попередження і повертається NaN (не число). Потім функція callback викликається переданим значенням і повертається результат. Якщо глобальна змінна RADIAN не визначена, результат перетворюється з радіанів на градуси [23].

```
function inv_trigo(callback, value) {
  if (callback.name !== "atan") {
    if (value < -1 || value > 1) {
      alert("Please enter a number in the range between -1 to 1 of acos");
      return NaN;
    }
  }

  let angle = callback(value);

  if (!RADIAN) {
    angle = (angle * 180) / Math.PI;
  }
  return angle;
}
```

Рисунок 3.15 – Лістинг коду функції inv_trigo

Функція «factorial(number)» (Рис 3.16.) приймає один аргумент number (число) та обчислює факторіал цього числа. У середині функції відбувається перевірка, чи число цілим. Якщо число є цілим, викликається функція gamma(number + 1). Якщо число дорівнює 0 чи 1, повертається 1. Потім виконується цикл, у якому обчислюється факторіал числа. Якщо результат дорівнює нескінченності (Infinity), повертається нескінченність. В іншому випадку повертається результат.

```
function factorial(number) {
  if (number % 1 !== 0) return gamma(number + 1);

  if (number === 0 || number === 1) return 1;
  let result = 1;
  for (let i = number; i > 0; i--) {
    result = result * i;
    if (result === Infinity) return Infinity;
  }
  return result;
}
```

Рисунок 3.16 – Лістинг коду функції factorial

Функція gamma(n) (Рис 3.17.) є реалізацією гамма-функції. У середині функції визначено деякі константи g та p. Якщо n менше 0.5, виконується

рекурсивний виклик функції `gamma(1 - n)` з подальшим виконанням математичного виразу для обчислення гамма-функції. Якщо `n` більше або 0.5, то виконується алгоритм Ланцоша для обчислення гамма-функції. Результат обчислень повертається із функції.

```
function gamma(n) {
    var g = 7,
        p = [
            0.9999999999980993, 676.5203681218851, -1259.1392167224028,
            771.32342877765313, -176.61502916214059, 12.507343278686905,
            -0.13857109526572012, 9.9843695780195716e-6, 1.5056327351493116e-7,
        ];
    if (n < 0.5) {
        return Math.PI / Math.sin(n * Math.PI) / gamma(1 - n);
    } else {
        n--;
        var x = p[0];
        for (var i = 1; i < g + 2; i++) {
            x += p[i] / (n + i);
        }
        var t = n + g + 0.5;
        return Math.sqrt(2 * Math.PI) * Math.pow(t, n + 0.5) * Math.exp(-t) * x;
    }
}
```

Рисунок 3.17 – Лістинг коду функції `gamma`

Наступним етапом у файлі «`style.css`» опишемо стилі оформлення веб-додатку. В файлі опишемо такі селектори та властивості:

1. Селектор `"*"` (Рис 3.18.):

- `padding: 0;` і `margin: 0;` обнуляють внутрішні і зовнішні відступи для всіх елементів.
- `background-origin: padding-box;` встановлює початок фонового зображення всередині елемента.

```
* {
    padding: 0;
    margin: 0;
    background-origin: padding-box;
}
```

Рисунок 3.18 – Лістинг коду селектору `"*"`

2. Селектор `"input, textarea, button, select"` (Рис 3.19.):

-webkit-tap-highlight-color: rgba(0,0,0,0); вимикає підсвічування елементів при натисканні на них на пристроях з сенсорним екраном.

```
input,
textarea,
button,
select,
a {
  -webkit-tap-highlight-color: rgba(0,0,0,0);
}
```

Рисунок 3.19 – Лістинг коду селектору "input, textarea, button, select"

3. Селектор "html" (Рис 3.20.):

- font-size: 16px; встановлює базовий розмір шрифту для всієї сторінки.
- height: 100%; встановлює висоту HTML-елемента на 100% висоти батьківського елемента.

```
html {
  font-size: 16px;
  height: 100%;
}
```

Рисунок 3.20 – Лістинг коду селектору "html"

4. Селектор "body" (Рис 3.21.):

- font-family: "Roboto", sans-serif; задає шрифт для тексту на сторінці, використовуючи шрифт Roboto.
- background-color: #f7f7f7; встановлює колір фону для елемента body.
- background-image: url("../img/bg01.png"); встановлює фонове зображення для елемента body, використовуючи зображення з шляху "../img/bg01.png".

```
body {
  font-family: "Roboto", sans-serif;
  background-color: #f7f7f7;
  background-image: url("../img/bg01.png");
}
```

Рисунок 3.21 – Лістинг коду селектору "body"

5. Селектор ".calculator" (Рис 3.22) :

- display: flex; встановлює елементи всередині .calculator як гнучкий контейнер.
- justify-content: center; вирівнює елементи по горизонталі по центру.
- align-items: center; вирівнює елементи по вертикалі по центру.
- height: calc(100vh - 25px); встановлює висоту .calculator на 100% висоти видимої області мінус 25 пікселів.

```
.calculator {
  display: flex;
  justify-content: center;
  align-items: center;
  height: calc(100vh - 25px);
}
```

Рисунок 3.22 – Лістинг коду селектору "calculator"

6. Селектор ".calc-cotnainer" (Рис 3.23.):

- width: 350px; встановлює ширину .calc-cotnainer на 350 пікселів.
- height: auto; автоматично визначає висоту .calc-cotnainer в залежності від його вмісту.
- background: #fff; встановлює колір фону для .calc-cotnainer на білий.
- border-radius: 30px; встановлює радіус кутів для .calc-cotnainer на 30 пікселів.
- box-shadow: 0 15px 53px 14px rgba(48, 81, 85, 0.1); встановлює тінь блоку .calc-cotnainer з заданими параметрами.

```
.calc-cotnainer {
  width: 350px;

  height: auto;
  background: #fff;

  border-radius: 30px;
  box-shadow: 0 15px 53px 14px rgba(48, 81, 85, 0.1);
}
```

Рисунок 3.23 – Лістинг коду селектору "calc-cotnainer"

7. Селектор ".output" (Рис 3.24):

- width: 100%; встановлює ширину .output на 100%.
- background: #345b82; встановлює колір фону для .output на темно-синій.
- margin-bottom: 10px; встановлює зовнішній відступ знизу для .output на 10 пікселів.

```
.output {  
  width: 100%;  
  background: #345b82;  
  margin-bottom: 10px;  
}
```

Рисунок 3.24 – Лістинг коду селектору "output"

8. Селектор ".operation" (Рис 3.25.):

- width: 100%; встановлює ширину .operation на 100%.
- height: 85px; встановлює висоту .operation на 85 пікселів.
- color: #f3ededb2; встановлює колір тексту для .operation на світло-сірий.
- font-size: 1.3em; встановлює розмір шрифту для .operation.
- font-weight: bold; встановлює насиченість шрифту для .operation на жирний.
- position: relative; встановлює позицію .operation як відносну.
- overflow: hidden; приховує вміст, якщо він виходить за межі .operation.
- .operation .value:
 - position: absolute; встановлює позицію .value як абсолютну.
 - opacity: 0.9; встановлює прозорість для .value на 0.9.
 - bottom: 25px; встановлює відстань від нижнього краю .operation до .value на 25 пікселів.
 - right: 15px; встановлює відстань від правого краю .operation до .value на 15 пікселів.
 - white-space: nowrap; встановлює властивість white-space на nowrap, щоб уникнути перенесення тексту на новий рядок.

- overflow: hidden; приховує вміст .value, якщо він виходить за межі.

```
.operation {
  width: 100%;
  height: 85px;

  color: #f3ededb2;

  font-size: 1.3em;
  font-weight: bold;

  position: relative;
  overflow: hidden;
}
```

Рисунок 3.25 – Лістинг коду селектору "operation"

9. Селектор ".result" (Рис 3.26.):

- width: 100%; встановлює ширину .result на 100%.
- height: 100px; встановлює висоту .result на 100 пікселів.
- color: #f1f1f1; встановлює колір тексту для .result на білий.
- font-size: 1.7em; встановлює розмір шрифту для .result.
- font-weight: bold; встановлює насиченість шрифту для .result на жирний.
- overflow: hidden; приховує вміст, якщо він виходить за межі .result.
- .result .value:
- position: absolute; встановлює позицію .value як абсолютну.
- top: 50%; встановлює відстань від верхнього краю .result до .value на 50%.
- transform: translateY(-50%); зсуває .value на 50% вгору, щоб вирівняти його по вертикалі.
- right: 15px; встановлює відстань від правого краю .result до .value на 15 пікселів.
- white-space: nowrap; встановлює властивість white-space на nowrap, щоб уникнути перенесення тексту на новий рядок.
- overflow: hidden; приховує вміст .value, якщо він виходить за межі.

```

.result {
  width: 100%;
  height: 100px;

  color: #f1f1f1;

  font-size: 1.7em;
  font-weight: bold;

  overflow: hidden;
  position: relative;
}

.result .value {
  position: absolute;

  top: 50%;
  transform: translateY(-50%);
  right: 15px;

  white-space: nowrap;
  overflow: hidden;
}

```

Рисунок 3.25 – Лістинг коду селектору "result"

10. Селектор ".calc-switcher" (Рис 3.26.):

- display: flex; встановлює елементи .calc-switcher як гнучкий контейнер.
- justify-content: space-between; вирівнює елементи в середині .calc-switcher рівномірно по горизонталі.
- align-items: center; вирівнює елементи в середині .calc-switcher по вертикалі.
- .calc-switcher label:
- font-size: 1.2em; встановлює розмір шрифту для міток в .calc-switcher.
- cursor: pointer; встановлює стилізований курсор для міток, щоб показати, що вони є інтерактивними елементами.
- display: none; приховує вхідний елемент.
- .calc-switcher input:checked + label:
- color: #345b82; змінює колір тексту мітки, якщо вхідний елемент перевірено.
- font-weight: bold; встановлює насиченість шрифту для мітки, якщо вхідний елемент перевірено [21].

```

.calc-switcher {
  display: flex;
  border-bottom: 1px solid rgba(7, 24, 104, 0.726);
  margin: 0 15px;
  padding-bottom: 8px;
}

.calc-switcher div {
  margin-right: 10px;
  font-size: 0.8rem;
  text-transform: uppercase;
  color: rgba(255, 17, 0, 0.562);
  cursor: pointer;
  opacity: 0.8;
  font-weight: 500;
}

.calc-switcher div:hover {
  color: rgb(255, 17, 0);
}

.calc-switcher div.active-calc {
  color: rgb(255, 17, 0);
  font-weight: 700;
}

```

Рисунок 3.26 – Лістинг коду селектору "calc-switcher"

11. Селектор ".input" (Рис 3.27.):

- Відповідає елементам з класом "input".
- Правила стилів:
 - border-radius: 0 0 25px 25px; - встановлює радіуси кутів елемента (верхні кути без закруглення, нижні кути з закругленням);
 - height: 100%; - встановлює висоту елемента відсотками від батьківського контейнера;
 - padding-bottom: 10px; - встановлює внутрішній відступ унизу елемента;
 - padding: 10px; - встановлює внутрішній відступ для всіх сторін елемента.

```

.input {
  /* background-color: #fff; */

  border-radius: 0 0 25px 25px;
  height: 100%;
  padding-bottom: 10px;
  padding: 10px;
}

```

Рисунок 3.27 – Лістинг коду селектору "input"

12. Селектор ".row" (Рис 3.28):

- Відповідає елементам з класом "row".
- Правила стилів:
 - `display: flex;` - встановлює елементи в один рядок;
 - `justify-content: space-between;` - розташовує елементи рівномірно по горизонталі з однаковим проміжком між ними;
 - `align-items: center;` - вирівнює елементи по вертикалі по центру.

```
.row {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
}
```

Рисунок 3.28 – Лістинг коду селектору ".row"

13. Селектор ".row button" (Рис 3.29):

- Відповідає кнопкам, які є дочірніми елементами елемента з класом "row".
- Правила стилів:
 - `width: 42px;` - встановлює ширину кнопки;
 - `height: 42px;` - встановлює висоту кнопки;
 - `font-size: 1.1em;` - встановлює розмір шрифту кнопки;
 - `border: none;` - видаляє рамку кнопки;
 - `outline: none;` - видаляє контур при отриманні фокусу кнопкою;
 - `border-radius: 10px 10px;` - встановлює радіуси кутів кнопки (закруглення всіх кутів);
 - `background-color: transparent;` - встановлює прозорий фон кнопки;
 - `margin: 5px;` - встановлює зовнішній відступ для кнопок;
 - `cursor: pointer;` - встановлює курсор у вигляді покажчика;
 - `color: #474141;` - встановлює колір тексту кнопки;
 - `background-color: #334ddf13;` - встановлює фоновий колір кнопки з півпрозорим ефектом;

- transition: all 0.1s ease-in-out; - встановлює плавний перехід для всіх змін стилів кнопки.

```
.row button {
  width: 42px;
  height: 42px;

  font-size: 1.1em;

  border: none;
  outline: none;
  border-radius: 10px 10px;
  background-color: transparent;

  margin: 5px;

  cursor: pointer;
  color: #474141;
  background-color: #334ddf13;
  transition: all 0.1s ease-in-out;
}
```

Рисунок 3.29 – Лістинг коду селектору ".row button"

14. Селектор ".row button:hover" (Рис 3.30):

- Відповідає кнопкам, які наведені курсором миші.

- Правила стилів:

- font-weight: bold; - встановлює жирний шрифт для тексту кнопки при наведенні курсора миші.

```
.row button:hover {
  font-weight: bold;
}
```

Рисунок 3.30 – Лістинг коду селектору ".row button:hover"

15. Селектор ".row #calculate" (Рис 3.31.):

- Відповідає елементу з ідентифікатором "calculate", який є дочірнім елементом елемента з класом "row".

- Правила стилів:

- color: #fff; - встановлює колір тексту на білий;

- background-color: #46e0bc; - встановлює фоновий колір на світло-зелений;
- width: 111px; - встановлює ширину елемента 111 пікселів;
- border-radius: 50px; - встановлює радіуси кутів елемента (закруглення всіх кутів).

```
.row #calculate {
  color: #fff;
  background-color: #46e0bc;
  width: 111px;
  border-radius: 50px;
}
```

Рисунок 3.31 – Лістинг коду селектору ".row #calculate"

16. Селектор ".row #delete" (Рис 3.32):

- Відповідає елементу з ідентифікатором "delete", який є дочірнім елементом елемента з класом "row".
- Правила стилів:
 - color: #fff; - встановлює колір тексту на білий;
 - background-color: #e05046; - встановлює фоновий колір на червоний.

```
.row #delete {
  color: #fff;
  background-color: #e05046;
}
```

Рисунок 3.32 – Лістинг коду селектору ".row #delete"

17. Селектор "#comma" :

- Відповідає елементу з ідентифікатором "comma".
- Правила стилів:
 - font-size: 2rem; - встановлює розмір шрифту 2 рема.

18. Селектори "#multiplication, #subtraction, #addition, #division, #clear" (Рис 3.33.):

- Відповідають елементам з ідентифікаторами "multiplication", "subtraction", "addition", "division", "clear".
- Правила стилів:
 - background-color: #fc9507; - встановлює фоновий колір на помаранчевий;
 - color: white; - встановлює колір тексту на білий.

```
#multiplication,
#subtraction,
#addition,
#division,
#clear {
    background-color: #fc9507;
    color: white;
}
```

Рисунок 3.33 – Лістинг коду селектору "#multiplication, #subtraction, #addition, #division, #clear"

19. Селектор ".input button span" (Рис 3.34.):

- Відповідає елементам span, які є дочірніми елементами кнопок, що знаходяться в елементі з класом "input".
- Правила стилів:
 - font-size: 0.6em; - встановлює розмір шрифту 0.6 ем;
 - position: absolute; - встановлює абсолютне позиціонування.

```
.input button span {
    font-size: 0.6em;
    position: absolute;
}
```

Рисунок 3.34 – Лістинг коду селектору ".input button span"

20. Селектор ".active-angle" (Рис 3.35.):

- Відповідає елементам з класом "active-angle".
- Правила стилів:
 - background-color: #1d1d1db2 !important; - встановлює фоновий колір з напівпрозорістю [11];

- color: #fff !important; - встановлює колір тексту на білий;
- border-radius: 10px !important; - встановлює радіуси кутів елемента (закруглення всіх кутів).

```
.active-angle {
  background-color: #1d1d1db2 !important;
  color: #fff !important;
  border-radius: 10px !important;
}
```

Рисунок 3.35 – Лістинг коду селектору ".active-angle"

21. Селектор ".advance-keys button" (Рис 3.36.):

- Відповідає кнопкам, які знаходяться в елементі з класом "advance-keys".
- Правила стилів:
 - color: rgb(243, 128, 34); - встановлює колір тексту на оранжевий;
 - font-size: 0.9rem; - встановлює розмір шрифту 0.9 ремпродовження...
 - font-weight: 600; - встановлює жирний шрифт для кнопок;
 - width: 45px; - встановлює ширину кнопок 45 пікселів;
 - height: 30px; - встановлює висоту кнопок 30 пікселів;
 - transition: all 0.2s ease-in-out; - встановлює плавний перехід для зміни стилів.

```
.advance-keys button {
  color: rgb(243, 128, 34);
  font-size: 0.9rem;
  font-weight: 600;
  width: 45px;
  height: 30px;
  transition: all 0.2s ease-in-out;
}
```

Рисунок 3.36 – Лістинг коду селектору ".advance-keys button"

22. Селектор ".advance-keys button:hover" (Рис 3.37.):

- Відповідає кнопкам, які знаходяться в елементі з класом "advance-keys" при наведенні курсору на них.

- Правила стилів:

- font-weight: 800; - змінює жирність шрифту при наведенні курсору.

23. Селектор ".advance-keys button:disabled":

- Відповідає вимкненим кнопкам, які знаходяться в елементі з класом "advance-keys".

- Правила стилів:

- opacity: 0.6; - встановлює прозорість кнопки на 0.6 (зменшує її яскравість);

- color: rgba(243, 128, 34, 0.829); - встановлює колір тексту з прозорістю.

```
.advance-keys button:disabled {
  font-weight: 800;
}
```

Рисунок 3.37 – Лістинг коду селектору ".advance-keys button:disabled"

23. Селектор "footer p" (Рис 3.38.):

- Відповідає елементам <p>, які знаходяться в елементі "footer".

- Правила стилів:

- font-size: 13.86px; - встановлює розмір шрифту 13.86 пікселів;

- text-align: center; - встановлює вирівнювання тексту по центру;

- color: gray; - встановлює колір тексту на сірий.

```
footer p {
  font-size: 13.86px;
  text-align: center;
  color: gray;
}
```

Рисунок 3.38 – Лістинг коду селектору "footer p"

24. Селектор "footer p a" (Рис 3.39.):

- Відповідає посиланням <a>, які знаходяться в елементі "footer" в межах елементів <p>.

- Правила стилів:

- text-decoration: none; - вимикає підкреслення для посилань;

- color: rgb(255, 65, 7); - встановлює колір тексту посилань на помаранчевий.

```
footer p a {
  text-decoration: none;
  color: rgb(255, 65, 7);
}
```

Рисунок 3.39 – Лістинг коду селектору "footer p a "

25. Медіа-запит "@media (max-width: 400px)":

Цей (Рис 3.40.) медіа-запит налаштовує вигляд елементів на пристроях з максимальною шириною вікна перегляду не більше 400 пікселів.

```
@media (max-width: 400px) {
  .calc-container {
    max-width: 300px;
    min-width: 250px;

    border-radius: 20px;
  }
  .row button {
    width: 40px;
    height: 40px;
    margin: 5px;
    outline: none;
  }
  .advance-keys button {
    font-weight: 600;
    width: 40px;
    height: 25px;
  }

  .result {
    height: 80px;
  }
  .row #calculate {
    width: 100px;
    border-radius: 50px;
  }

  html {
    font-size: 13px;
  }
}
```

Рисунок 3.40 – Лістинг коду медіа-запиту "@media (max-width: 400px)"

У даному розділі представлена практична реалізація веб-додатку наукового калькулятора (рис 3.41). Було створено структуру веб-сторінки калькулятора за допомогою HTML-коду і задано стилі за допомогою CSS. Калькулятор має два режими: науковий і звичайний. Режим можна перемикати за допомогою відповідних елементів на сторінці

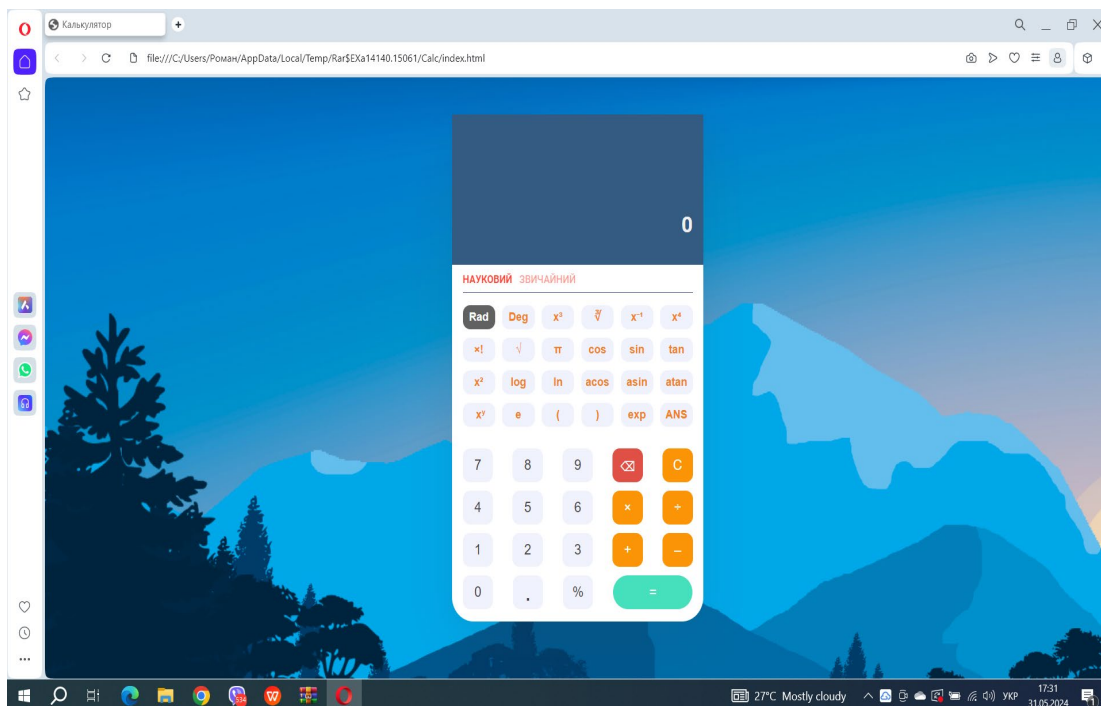


Рис.3.41. Вигляд калькулятора

Також були оголошені змінні та константи, які використовуються для зберігання операцій, формул та попереднього результату обчислень. Були створені кнопки калькулятора на основі заданого масиву, які надають функціонал калькулятора.

Реалізована можливість вимикання розширеного режиму калькулятора, якщо калькулятор працює в звичайному режимі, а також функцію для перемикання між використанням радіанів і градусів.

Загалом, цей розділ надає загальний опис практичної реалізації веб-додатку наукового калькулятора з використанням HTML, CSS і JavaScript.

ВИСНОВКИ

У процесі написання даної роботи були виявлені та розглянуті різні теоретичні і практичні аспекти проблеми проектування та розробки наукового калькулятора.

Проведені дослідження включали аналіз наукової літератури, вивчення існуючих рішень, які відрізняються функціональністю, можливостями та інтерфейсом у галузі наукових калькуляторів, математичне моделювання, розробку та реалізацію алгоритмів і програмного забезпечення, а також проведення експериментів для оцінки функціональності та ефективності калькулятора. Результати досліджень дозволили оцінити важливість і актуальність проектування та розробки наукових калькуляторів у сучасному науково-технічному середовищі.

Основні положення, що відрізняють це дослідження від напрацювань інших авторів, включають:

- Використання сучасних технологій і передових рішень у процесі розробки калькулятора.
- Практична реалізація наукового калькулятора, який може знайти застосування у різних галузях, таких як фізика, математика, інженерія, комп'ютерні науки та інші.

Пропонується також звернути увагу на можливості започаткування подальших досліджень у цій галузі, зокрема:

- Вивчення інших алгоритмів та методів для розробки наукових калькуляторів з метою забезпечення ще більшої точності та швидкості обчислень.
- Дослідження можливості інтеграції калькулятора з іншими науковими інструментами для забезпечення комплексного аналізу даних.
- Розробка інтерфейсних рішень, що спрощують використання калькулятора та роблять його більш доступним для користувачів різних рівнів.

- Проведення додаткових експериментів і тестувань для підтвердження ефективності та надійності розробленого калькулятора.

Необхідно зазначити, що розробка калькулятора є відповідальним завданням, оскільки навіть одна помилка в розрахунках може негативно вплинути на репутацію не лише сайту, але й компанію, для якої він був створений.

Отже, сучасні інформаційні технології дозволяють створювати професійні онлайн-калькулятори з необхідним функціоналом для різноманітних сфер діяльності. Важливо знати їх можливості та вміти їх використовувати.

Загальні підсумки дослідження свідчать про досягнення поставлених у вступі завдань і цілей. Розроблений науковий калькулятор відповідає сучасним вимогам і забезпечує широкий спектр функцій для наукових обчислень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Корнієнко, Ю. В., Одеров, А. В. Програмний код: написання, тестування, оптимізація. Київ: Видавництво "Діалектика", 2020. 272 с.
2. Василенко, Г. В., Шмат, В. М. Проектування програмного забезпечення. Київ: Видавництво "Центр учбової літератури", 2018. 344 с.
3. Стадник, Р. В. Основи веб-розробки. Київ: Видавництво "КМ-Букс", 2017. 256 с.
4. Недашківський О.Л. Планування та проектування інформаційних систем. / О.Л. Недашківський. – К.: ДУТ, 2014. – 215 с.
5. Роберт С. М. Чистий Код / Сесіл Мартін Роберт., Фабула. 2019. 448 с.
6. Дакетт Д. HTML та CSS. Розробка і дизайн веб-сайтів. / Джон Дакетт. – Світ: Київ, 2021. – 480 с.
7. Фрімен Е. Head First. Програмування на JavaScript / Е. Фрімен, Е. Робсон. – Харків: Фабула, 2022. – 672 с.
8. Head First. Патерни проектування / Е.Фрімен, Е. Робсон, Б. Бейтс, К. Сієрра. – Харків: Фабула, 2020. – 672 с.
9. Васильєв О. Алгоритми / Олексій Васильєв. – Київ: Ліра-К, 2022. – 424 с.
10. МакГрат М. JavaScript для початківців. 6-те видання / Майк МакГрат. – Київ: Print2print, 2022. – 232 с.
11. Головчук А. Ф. Інженерна та комп'ютерна графіка / А. Ф. Головчук, О. І. Кепко, Н. М. Чумак. – Київ: Центр учбової літератури, 2021. – 160 с.
12. Брюханова Г. Комп'ютерні дизайн-технології: навчальний посібник / Галина Брюханова. – Київ: Центр учбової літератури, 2021. – 180 с.
13. Стейнер К. Тотальна автоматизація. Як комп'ютерні алгоритми змінюють світ / Крістофер Стейнер. – Київ: Наш Формат, 2018. – 280 с.
14. Мельник Р. Програмування веб-застосувань (фронт-енд та бек-енд) / Роман Мельник. – Львів: Львівська політехніка, 2018. – 248 с.
15. Myers M. A Smarter Way to Learn JavaScript / Mark Myers. – Південа Кароліна: CreateSpace Independent Publishing Platform, 2014. – 256 с.

16. Frisbie M. Professional JavaScript for Web Developers (Tech Today) 5th / Matt Frisbie. – Birmingham: wrox, 2023. – 1104 с.
17. Флэнаган Д. JavaScript. Повне керівництво 7-е видання / Девід Флэнаган. – Київ: Науковий світ, 2022. – 722 с.
18. 1. Катренко А.В. Управління ІТ-проектами . Книга 1. Стандарти, моделі та методи управління проектами: підручник. Львів: «Новий Світ-2000», 2013. 550 с.
19. Порівнюємо React, Angular і Vue. URL: <https://dou.ua/forums/topic/39933/>
20. Топ-10 мов програмування, які будуть актуальні в 2024. URL: <https://career.softserveinc.com/uk-ua/stories/top-10-programming-languages-to-learn-in-2023>
21. Петренко В.О. Огляд кросплатформених фреймворків для розробки мобільного додатку. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2022/04/37.pdf>
22. Calconic. URL: <https://www.calconic.com/>
23. Сервіс Zoho Creator платформа створення онлайн калькуляторів та веб додатків <https://www.zoho.com/creator/index1.html>
24. Calculoid конструктор онлайн калькуляторів. URL: <https://www.calculoid.com/>
25. JSCalc.io конструктор онлайн калькуляторів, використанням JavaScript <https://jscale.io>
26. Максюта Р. Особливості розробки калькулятора для сайту // **Збірник тез І-а Всеукраїнська конференція молодих вчених «Математичні методи та цифрові технології в економіці»** .URL: <https://dpu.edu.ua/novyny-kafedry-kibernetiky-ta-prykladnoi-matematyky/provedena-i-vseukrainska-konferentsiia-molodykh-vchenykh-matematychni-metody-ta-tsyfrovii-tekhnologii-v-ekonomitsi>