Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА БАГАТОФУНКЦІОНАЛЬНОГО CSS-ГЕНЕРАТОРА СТИЛІВ З ВИКОРИСТАННЯМ REACT

DEVELOPMENT OF A MULTIFUNCTIONAL CSS STYLE GENERATOR USING REACT

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗс-21
Дерелюк Тимур Юрійович

Керівник:
Ящук Андрій Анатолійович

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

_____

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«20» 12 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Дерелюку Тимуру Юрійовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *«Розробка багатофункціонального CSS-генератора стилів з використанням React»*
Керівник роботи: *доцент Ящук А.А.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *React, JavaScript, HTML5, CSS3, Node.js, NPM, Json, Netlify, методичні вказівки до виконання кваліфікаційної роботи бакалавра*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз аналогів, формування вимог, побудова UML-діаграм, вибір технологій розробки, розробка інтерфейсу, реалізація функціоналу, тестування, створення документації, SEO-оптимізація, забезпечення безпеки*
5. Перелік графічного матеріалу: *17 рисунків, 1 додаток*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Яцук А. А.*		
Специфікація вимог до розробленої системи	*Яцук А. А.*		
Розробка об'єкта проектування	*Яцук А. А.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	2,34 %		
Академічна доброчесність	*Яцук А. А.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	Виконано
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	Виконано
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	Виконано
4	Розробка функціонально-структурної схеми роботи об'єкта проектування	до 29.03.2025 р.	Виконано
5	Практична реалізація об'єкта проектування	до 26.04.2025 р.	Виконано
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	Виконано
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	Виконано

Здобувач вищої освіти _____ Тимур ДЕРЕЛЮК

Керівник кваліфікаційної роботи _____ Андрій ЯЩУК

АНОТАЦІЯ

Дерелюк Т. Ю. Розробка багатофункціонального CSS-генератора стилів з використанням React. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків і пропозицій, списку використаних джерел та додатків.

У першому розділі здійснено аналіз розробки сайтів для будівельних компаній і були поставлені завдання до кваліфікаційної роботи. У другому розділі були розглянуті вимоги до розробки, вибрані технології, описані методики та алгоритми створення, а також виконано проєктування сайту. В третьому розділі розроблено багатофункціональний генератор CSS-стилів та викладено для публічного використання. У висновках підсумовано інформацію, представлену в попередніх розділах.

Ключові слова: JavaScript, вебзастосунок, сайт, генератор стилів, CSS.

ABSTRACT

Dereliuk T. Development of a Multifunctional CSS Style Generator Using React. Manuscript. Qualification work of the bachelor's program "Software Engineering" in the specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's thesis consists of an introduction, three chapters, conclusions and suggestions, list of references and additions.

The first chapter analyzes the development of websites for construction companies and sets the tasks for the qualification work. The second section discusses the requirements for development, selects the technologies, describes the methods and algorithms of creation, and designs the site. In the third section, a multifunctional CSS style generator is developed and made available for public use. The conclusion summarizes the information presented in the previous sections.

Keywords: JavaScript, web application, website, style generator, CSS.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі веброзробка стрімко розвивається, і створення якісного, естетично привабливого та адаптивного дизайну сайтів стало обов'язковим елементом будь-якого онлайн-проєкту. CSS-стилі є основним інструментом оформлення вебсторінок, і їх правильне використання істотно впливає на зручність користування, швидкість завантаження сторінок та загальне враження від сайту.

Ручне написання CSS-коду часто є трудомістким і потребує від розробника глибоких знань, уважності та досвіду, особливо при створенні адаптивної верстки, анімацій чи складних ефектів. У таких випадках онлайн-генератори стилів стають ефективним інструментом, що прискорює розробку, зменшує кількість помилок і дозволяє зосередитися на креативних аспектах проєкту.

Виходячи з цього, розробка багатофункціонального генератора CSS-стилів є надзвичайно актуальною темою в наш час, оскільки вона повністю відповідає сучасним вимогам розробки, а саме до її швидкості, зручності використання інструментів та забезпечення високої якості вебпродуктів.

Метою кваліфікаційної роботи бакалавра є розробка генератора CSS стилей для полегшення розробки інших проектів розробниками, які будуть використовувати технології CSS.

Об'єктом кваліфікаційної роботи бакалавра є багатофункціональний генератор CSS-стилів.

Предметом кваліфікаційної роботи бакалавра є дослідження послідовного процесу розробки, засобів та технологій реалізації багатофункціонального генератора CSS-стилів для полегшення розробки інших проектів, які використовують технологію CSS стилей.

Для реалізації поставленої мети необхідно вирішити наступні завдання:

– здійснити аналіз доцільності створення розробки, а також провести порівняння з наявними аналогами;

– визначити вимоги до функціоналу вебзастосунку та побудувати UML-діаграми, що відображатимуть функціональні й нефункціональні характеристики системи;

– розглянути сучасні інструменти та технології веброзробки, обравши найбільш відповідні до поставлених вимог та обґрунтувати вибір архітектурних рішень та програмних засобів;

– реалізувати багатофункціональний CSS-генератор з використанням бібліотеки React та інших актуальних технологій веброзробки;

– провести тестування створеного вебзастосунку на відповідність сформульованим вимогам і виконати налагодження у разі виявлених відхилень;

– розробити доступну та актуальну інструкцію для користувачів із розгортання вебзастосунку;

– здійснити оптимізацію вебзастосунку шляхом побудови семантичної структури та врахування вимог пошукових систем, із подальшою перевіркою рівня оптимізованості;

– впровадити базові заходи кібербезпеки з метою захисту від поширених атак та вразливостей, зокрема використання HTTPS-протоколу, обмеження доступу до локального середовища, а також реалізацію перевірки і фільтрації вхідних даних.

Розробка багатофункціонального CSS-генератора стилів спрощує та прискорює створення вебоформлення, дозволяючи генерувати код без ручного введення. Багатофункціональний вебзастосунок базується на сучасних технологіях SPA та UX/UI, забезпечуючи інтуїтивний і зручний інтерфейс для користувачів.

Результати дослідження і розробки були представлені на X Міжнародній науково-практичній конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві» (ІТОНВ-2025), м. Луцьк, 23-24 травня 2025 р. (додаток А).

РОЗДІЛ 1
АНАЛІЗ РОЗРОБКИ БАГАТОФУНКЦІОНАЛЬНОГО CSS-ГЕНЕРАТОРА СТИЛІВ З ВИКОРИСТАННЯМ REACT І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У сучасному світі веброзробки стильове оформлення інтерфейсів є невід'ємною складовою, що безпосередньо впливає на користувацький досвід, доступність та візуальне сприйняття вебзастосунків. Одним із основних інструментів для стилізації вебінтерфейсів є мова каскадних таблиць стилів (CSS). Однак ручне написання CSS-коду часто є трудомістким процесом, що вимагає високої точності, знання синтаксису та часу, особливо при роботі з адаптивним дизайном, анімаціями та складними UI-компонентами.

Задля вирішення цих проблем, спільнота розробників активно створює та використовує CSS-генератори – інструменти, що дозволяють генерувати стилі за допомогою графічного інтерфейсу або заздалегідь визначених параметрів. Такі генератори значно спрощують процес створення стилів, зменшують кількість помилок та дозволяють пришвидшити розробку. Водночас, більшість з існуючих рішень мають обмежену функціональність, не підтримують гнучке налаштування або не мають сучасної архітектури для масштабування та повторного використання компонентів.

Варто відзначити, що розробка багатофункціонального генератора CSS-стилів, насамперед є розробкою вебзастосунку, тому важливо провести аналіз.

Вебзастосунок – це інтерактивна програма, яка працює у веббраузері та взаємодіє з користувачем через графічний інтерфейс. У структурі типового вебзастосунку можна виокремити такі ключові компоненти:

– інтерфейс користувача (UI) – графічна частина, з якою взаємодіє користувач;

– сховище стану (State Management) – сучасний механізм, який надає змогу зберігати, передавати та обробляти дані між компонентами вебзастосунку;

– логіка клієнта (Frontend logic) – реалізована з використанням сучасних технологій веброзробки, таких як React.

Сучасні вебзастосунки ґрунтуються на логічно структурованій архітектурі, яка забезпечує чітку організацію роботи між користувацьким інтерфейсом, внутрішніми процесами та обробкою даних. Як правило, у таких застосунках використовуються підходи, що дозволяють максимально ефективно обробляти запити користувачів та реагувати на них у реальному часі.

Ключові процеси в архітектурі типового вебзастосунку можна описати так:

– першим етапом завжди виступає процес отримання браузером структури сторінки, яка була сформована на основі HTML, CSS , JavaScript та інших використовуваних технологіях;

– після завантаження сторінки користувач може взаємодіяти з її елементами – натискати кнопки, змінювати значення форм тощо;

– кожна дія користувача викликає відповідні зміни у стані застосунку. У React, наприклад, це реалізується через механізм компонентів і хуків;

– у разі потреби застосунок надсилає запити до серверної частини, отримує відповіді через API та використовує їх для оновлення відображення;

– оновлення інтерфейсу. Дані, отримані або змінені під час взаємодії, відображаються у графічному інтерфейсі без повного перезавантаження сторінки;

– зберігання інформації. У разі необхідності дані можуть бути збережені – як на стороні клієнта (наприклад, у локальному сховищі), так і на сервері.

Важливим є виділити архітектурні підходи у веброзробці. Найпоширенішими архітектурними підходами у веброзробці є:

– монолітна архітектура передбачає, що всі частини застосунку – інтерфейс, логіка та база даних – об’єднані в одному цілісному рішенні. Це простий варіант для невеликих проєктів, хоча з масштабуванням можуть виникати труднощі;

– мікросервісна архітектура розділяє застосунок на незалежні сервіси, кожен з яких виконує окрему задачу. Такий підхід підвищує гнучкість, спрощує оновлення та дозволяє масштабувати окремі частини системи;

– SPA (Single Page Application) – застосунок, який завантажується один раз і працює без перезавантажень сторінки. Це забезпечує швидкий та плавний досвід для користувача;

– serverless-модель дозволяє запускати окремі функції без керування сервером. Це зручно для невеликих, подієво-орієнтованих задач і зменшує навантаження на інфраструктуру.

З огляду на сучасні підходи до побудови вебзастосунків, особливо у контексті зручності інтерфейсу та швидкості взаємодії, дедалі більшої актуальності набувають інструменти, що автоматизують рутинні етапи розробки. Одним із таких інструментів є CSS-генератор стилів – зручне рішення для швидкого створення візуального оформлення елементів без необхідності писати код вручну [1].

CSS-генератор стилів – це інструмент, який дозволяє автоматично створювати фрагменти CSS-коду на основі параметрів, заданих користувачем через зручний інтерфейс. Замість того, щоб вручну прописувати властивості стилю, такі як кольори, тіні, рамки, шрифти чи анімації, розробник або дизайнер може обрати потрібні значення зі списків, слайдерів або полів вводу – і одразу отримати готовий код.

Основна ідея розробки багатофункціонального CSS-генератора стилів полягає в тому, щоб спростити рутинну частину роботи зі стилями та зменшити кількість помилок, які часто трапляються при ручному кодуванні. Такий інструмент особливо корисний на етапах прототипування, швидкої розробки, або коли потрібно створити візуально привабливі елементи без поглибленого занурення у CSS.

Розглянемо більш детально практичну доцільність використання CSS-генератора стилів:

– використання вебзастосунку, який надає генерацію CSS-стилів, допомагає більшій кількості людей гарно оформити інтерфейс сторінки;

– будь-які стилі, та готовий код до них, створюються за лічені секунди;

– використання генератора CSS-стилів сприяє дотриманню уніфікованого стилю в проєкті;

– користувач одразу бачить результат своїх налаштувань та може відразу почати використовувати стилі в своєму проєкті.

У процесі розробки багатофункціонального CSS-генератора за основу планується використати бібліотеку React, яка надає широкі можливості для створення інтерактивного, гнучкого та масштабованого інтерфейсу. Завдяки компонентному підходу та ефективному керуванню станом, React дозволяє будувати застосунки, що легко підтримуються і розширюються. Архітектура майбутнього генератора буде побудована за принципами SPA (Single Page Application), що забезпечить плавну та безперервну взаємодію користувача з інтерфейсом без потреби перезавантаження сторінки.

Основні особливості архітектури запланованого застосунку:

– уся функціональність буде розбита на незалежні модулі, такі як генератор кнопок, тіней, градієнтів тощо. Це дає змогу спрощувати масштабування, полегшує підтримку і забезпечує повторне використання коду;

– для збереження та оновлення параметрів стилів у реальному часі використовуватимуться React hooks (useState, useReducer) або зовнішні бібліотеки, як-от Redux чи Zustand, залежно від складності компонентів;

– кожен генератор матиме зручні елементи керування – поля введення, повзунки, перемикачі – завдяки яким користувач зможе налаштувати потрібні параметри без необхідності працювати з кодом напряму;

– у результаті користувач зможе отримати валідний CSS-код, який легко скопіювати чи експортувати для подальшого використання у власних проєктах.

Такий підхід дозволить створити не просто інструмент для стилізації, а повноцінну платформу, орієнтовану на зручність, гнучкість та швидкість розробки.

Сьогодні в інтренет-просторі є досить багато різноманітних вебзастосунків з генерацією CSS-стилів, деякі дають більше можливостей, а деякі навпаки, тому буде доречним провести аналіз та дізнатися, які привілеї вони мають та можливості вони мають перед нашим додатком як розробляється.

«WebCodeTools» – це зручний вебзастосунок, що об’єднує різноманітні інструменти для розробників, серед яких є й необхідний нам CSS-генератор. Проєкт створювався з метою надати швидкий доступ до корисних рішень, які можна застосовувати у власних розробках. Початкова версія мала обмежену функціональність та простіший дизайн, без підтримки колірної схеми (рис. 1.1).

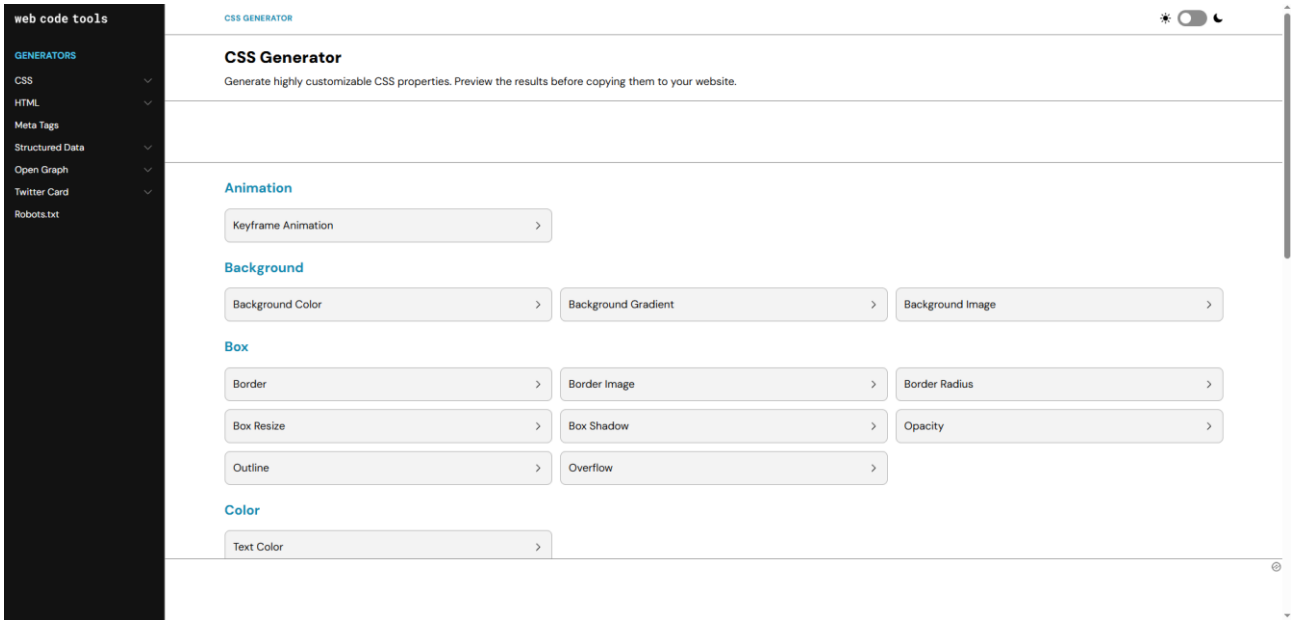

Рисунок 1.1 – Головна сторінка CSS-генератора «WebCodeTools» [2]

Серед переваг в цьому вебзастосунку можна відзначити величезний функціонал який дає можливості отримувати розробнику потрібні функції у подальшій розробці, також до переваг варто віднести можливість змінювати колірну схему сайту в залежності від освітлення, ще потрібно відзначити можливість отримувати динамічно код та мати інформацію до його подальшого використання.

Недоліком сайту є не гнучка верстка під менші екрани та деякі анімації відображаються не зовсім коректно.

«CSS3 Generator» – це досить простий, але при тому функціональний генератор, який дозволяє розробнику отримати потрібні йому стилі швидко та надійно (рис. 1.2). Сайт майже не оновлюється, але стара версія сайту не мала гарної анімації на головній сторінці та мала менший функціонал. Розробником сайту є Randy Jensen, який на головній сторінці зробив посилання сайт, де можна побачити усі його робити та отримати доступ до них.

Рисунок 1.2 – Головна сторінка сайту «CSS3 Generator» [3]

Перевагами сайту є зрозумілий інтерфейс та його швидкодія завдяки спрощенню дизайну сайта до мінімуму, а також мінімізація інших елементів сайту задля пришвидшення завантаження сайту на різних приладах.

Серед недоліків варто відзначити не малу кількість так званих «багів», які можуть заважати користувачу використовувати сайт у власних цілях. Також недоліком є не коректно розроблений інтерфейс відображення згенерованих стилів.

«CSSmatic» – це генератор CSS-стилів, розроблений за допомогою таких сервісів як: freepik та thumbr.it. Почав свій життєвий цикл сайт аж у 2013 році, за цей час багато чого змінилось від дизайну до технологій які використовувались на сайті.

Сайт має досить зрозумілий інтерфейс, але не великий функціонал у зрівнянні з попередніми аналогами які ми розглядали (рис. 1.3).

Рисунок 1.3 – Головна сторінка вебзастосунку «CSSmatic» [4]

Серед переваг варто відзначити особливу функцію під назвою «Noise Texture», завдяки ній розробник може створювати фону з так званим «шумом» і після цього використовувати у дизайні. Також перевагою сайту є його швидкодія завдяки простому дизайну.

Недоліком сайту є відсутність багатьох необхідних функції присунтіх CSS генератора, також було знайдено декілька неприємних багів в інтерфейсі. До недоліку також важливо віднести відсутність кнопки зміни колірної схеми в залежності від освітлення.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Аналіз сучасного стану веброзробки свідчить про зростання попиту на інструменти, що спрощують та прискорюють створення інтерфейсів. Особливо актуальним є автоматизоване генерування CSS-стилів, яке дозволяє зменшити

рутинну роботу розробника, підвищити якість стилізації та забезпечити візуальну уніфікацію елементів. На ринку існує низка подібних рішень, проте більшість з них обмежені у функціональності, не мають сучасного дизайну або не забезпечують достатньої гнучкості у налаштуваннях.

У зв’язку з цим, виникає потреба у створенні сучасного вебзастосунку – багатофункціонального CSS-генератора, який дозволяє користувачеві формувати стилі за допомогою інтуїтивного інтерфейсу з підтримкою перегляду результатів у реальному часі. Такий інструмент має бути реалізований з використанням бібліотеки React, що дозволяє створювати компонентну структуру, зручно керувати станом програми та будувати односторінковий застосунок (SPA), який забезпечить швидку та зручну взаємодію.

У рамках кваліфікаційної роботи бакалавра ставиться завдання:

- проаналізувати необхідність даної розробки, а також відповідні аналоги;
- визначити вимоги до розробки, та створити UML-діаграми, які візуально будуть відображати функціональні та нефункціональні вимоги;
- розглянути сучасні технології веброзробки та обрати найкращі рішення які будуть відповідати вимогам;
- реалізувати розробку багатофункціонального CSS-генератора з використанням бібліотеки React, та інших технологій сучасної веброзробки;
- провести тестування розробленого вебзастосунку на відповідність функціональним та нефункціональним вимогам з подальшим налагодженням;
- створити зрозумілу та актуальну інструкцію з розгортання вебзастосунку для користувача;
- провести детальну оптимізацію вебзастосунку побудувавши семантичну структуру, а також дотримавшись відповідних вимог пошукових систем з подальшою перевіркою додатку на оптимізованість;
- реалізувати засоби безпеки додатку для уникнення популярних атак та уразливостей, реалізувавши такі методи та кроки: впровадження HTTPS-протоколу, обмеження доступу до локального середовища, створення обробки форм введення.

Результатом роботи має стати вебзастосунок, який не лише виконує поставлені функції, а й відповідає сучасним вимогам до продуктивності та зручності користувацького інтерфейсу.

Висновки до розділу 1

У цьому розділі було проведено аналіз сучасного стану розробки вебзастосунків та визначено актуальність створення інструментів, що автоматизують роботу з CSS-стилями. Було розглянуто основні архітектурні підходи до побудови вебзастосунків, зокрема SPA-модель, яка є доцільною для реалізації генератора стилів. Окрему увагу приділено характеристиці CSS-генераторів як засобу спрощення процесу стилізації інтерфейсу та підвищення ефективності розробки.

На основі проведеного аналізу сформульовано завдання – створити багатофункціональний вебзастосунок на базі React для інтерактивного налаштування CSS-стилів. Запропоноване рішення відповідає сучасним вимогам до швидкодії, гнучкості та зручності, а його актуальність обумовлена зростаючою потребою у швидкому прототипуванні. Генератор стане корисним як для початківців, так і для досвідчених розробників.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Основне завдання даної роботи полягає у створенні вебзастосунку, який дозволятиме користувачам значно спростити та автоматизувати процес створення типових CSS-стилів. Такий інструмент має забезпечити зручний інтерфейс для формування стилів без потреби ручного написання коду, що особливо корисно для розробників-початківців, дизайнерів або тих, хто хоче швидко протестувати різні варіанти оформлення. Крім того, має бути передбачено можливість збереження та повторного використання створених стилів, що дозволяє підвищити ефективність роботи та забезпечити послідовність у візуальному оформленні вебзастосунків.

Для аналізу та визначенню вимог до розроблюваного багатофункціонального CSS-генератора стилів, необхідно:

– проаналізувати функціонал популярних CSS-генераторів, таких як CSS3 Generator, WebCodeTools, CSSmatic;

– визначити сильні та слабкі сторони подібних сервісів з точки зору користувача та розробника;

– сформувати перелік ключових функцій, які мають бути реалізовані в CSS-генераторі;

– встановити технічні вимоги до реалізації застосунку з використанням бібліотеки React, з урахуванням сучасних підходів до розробки SPA-додатків;

– визначити інструменти, необхідні для розробки, тестування та розгортання застосунку;

– розробити можливість масштабування функціональності, зокрема додавання нових генераторів або підтримки нових CSS-функцій.

Функціональні вимоги:

– реалізація окремих генераторів для різних CSS-ефектів (box-shadow, border-radius, gradient, transform, filter);

– можливість попереднього перегляду стилізованого елементу в реальному часі;

– підтримка ручного налаштування параметрів через інтерфейс (слайдери, чекбокси, поля вводу);

– автоматичне формування готового CSS-коду з можливістю копіювання.

Нефункціональні вимоги:

– зручний, адаптивний та інтуїтивно зрозумілий інтерфейс користувача;

– швидкодія та висока чутливість інтерфейсу до змін параметрів;

– сумісність з основними браузерами (Chrome, Firefox, Edge, Safari);

– можливість розширення проєкту завдяки додаванню нових компонентів;

– мінімальні затримки при оновленні стану інтерфейсу (SPA-підхід);

– надійність роботи застосунку без критичних помилок чи збоїв.

Таким чином, під час розробки проєкту особлива увага приділяється зручності використання, гнучкості налаштувань та швидкості взаємодії користувача з інструментом, що повністю відповідає сучасним вимогам до розробки вебінтерфейсів.

Для візуального сприйняття та формалізації вимог, а також проєктування архітектури вебзастосунку, було обрано наступні UML діаграми:

– діаграма-прецедентів – візуально показує функціональну залежність між акторами та прецедентами;

– діаграма-послідовності – відображає взаємодії об’єктів впорядкованих за часом;

– діаграма-діяльності – показує нам динамічні аспекти поведінки системи.

На рисунку 2.1 відображено діаграму-прецедентів, вона відображає загальний процес взаємодії користувача з вебзастосунком CSS-генератора, починаючи з вибору потрібного генератора та введення даних, і закінчуючи

отриманням згенерованого результату. Вона демонструє, як введена інформація обробляється вебсервісом: проходить перевірку, рендериться в інтерфейсі та надсилається користувачу. Також показано взаємодію з хостингом, яка забезпечує стабільну обробку запитів. Діаграма ілюструє логіку побудови системи та послідовність її ключових процесів.

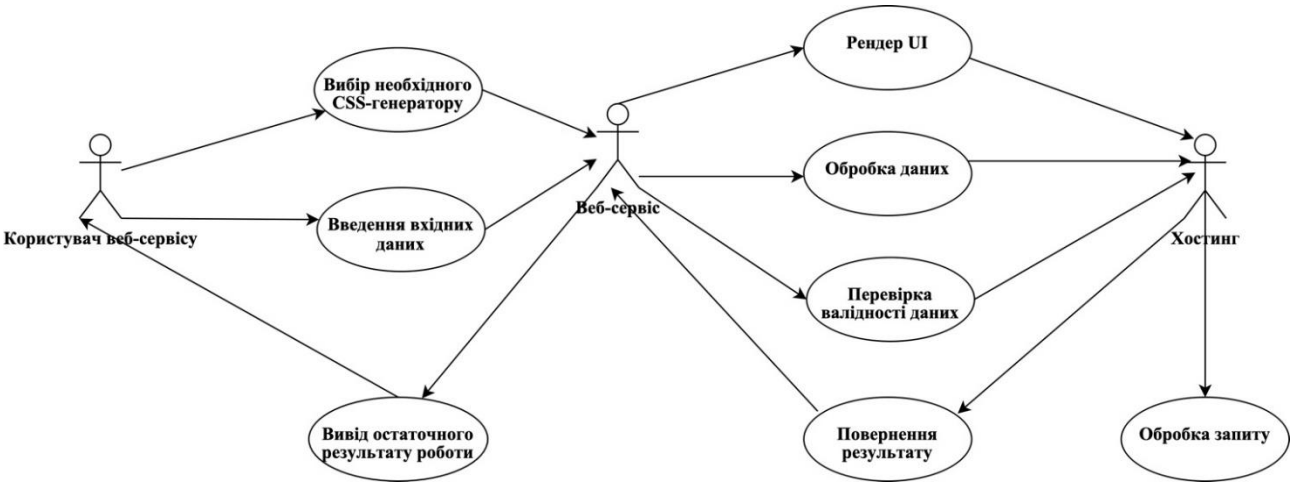

Рисунок 2.1 – Діаграма-прецедентів додатку

Наступною діаграмою (рис. 2.2) виступає діаграма-послідовності. Ця діаграма відображає послідовність взаємодії між користувачем, вебсервісом і сервером у процесі вибору CSS-генератора та обробки введених даних. Вона демонструє, як після вибору генератора та введення параметрів ініціюється HTTP-запит до сервера, де відбувається перевірка коректності даних. У разі успішної перевірки система повертає відповідь 200 OK і генерує відповідний інтерфейс, якщо дані некоректні, сервер відповідає кодом 400 Bad і пропонує спробувати ще раз.

Діаграма демонструє послідовність обробки запиту з урахуванням перевірки введених даних. Якщо дані відповідають заданим умовам, виконується подальша обробка (генерація коду, оновлення інтерфейсу тощо). У разі помилки – система повертає відповідне повідомлення або перериває виконання. Такий підхід забезпечує коректність і стабільність роботи застосунку.

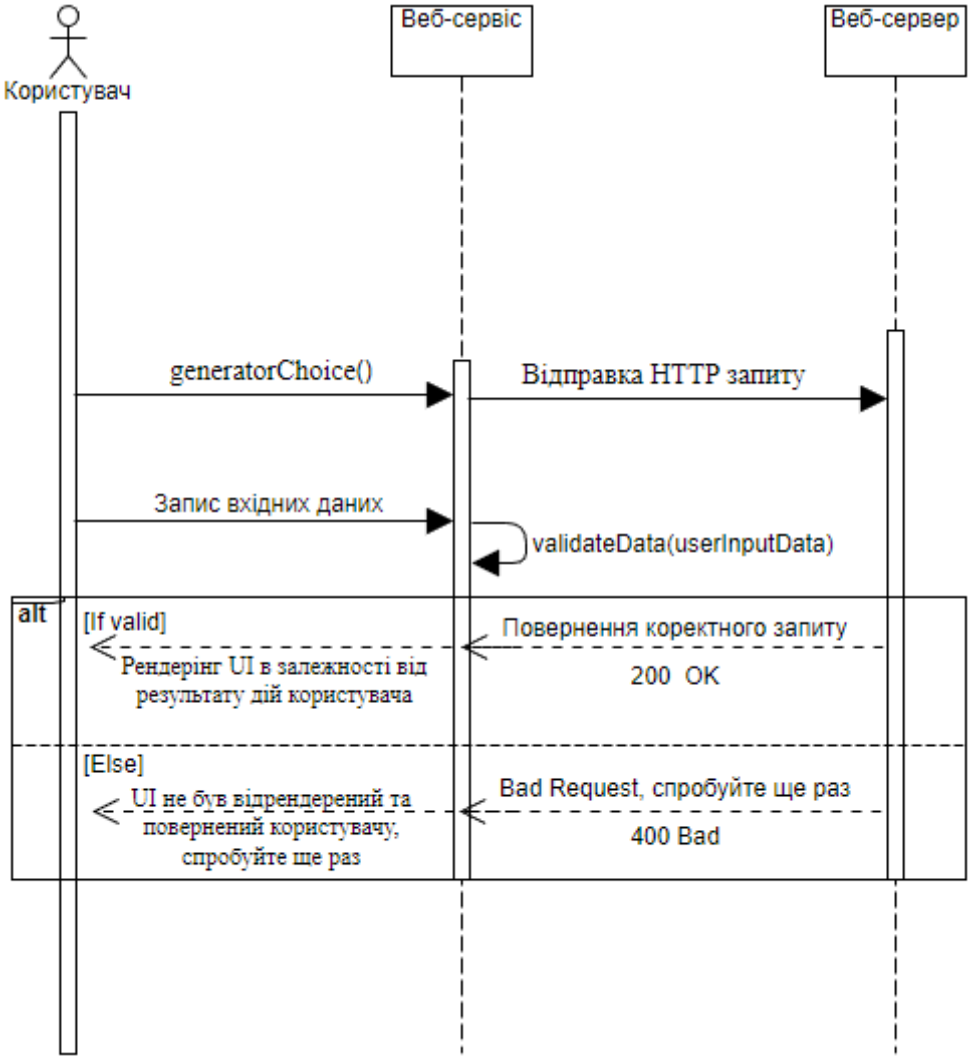

Рисунок 2.2 – Діаграма-послідовності додатку

Діаграма діяльності (рис. 2.3) ілюструє процес обробки запиту користувача в системі CSS-генератора, починаючи з вибору генератора і введення даних, і завершуючи отриманням результату. Вона демонструє, як введені дані передаються на вебсервіс, проходять перевірку, у разі валідності дані надсилаються на хостинг для обробки. У разі помилки на стороні хостингу система повертає повідомлення про критичну помилку: якщо ж обробка успішна – інтерфейс оновлюється, а користувач отримує результат. Діаграма наочно відображає логіку взаємодії між основними компонентами системи та обробку можливих виняткових ситуацій.

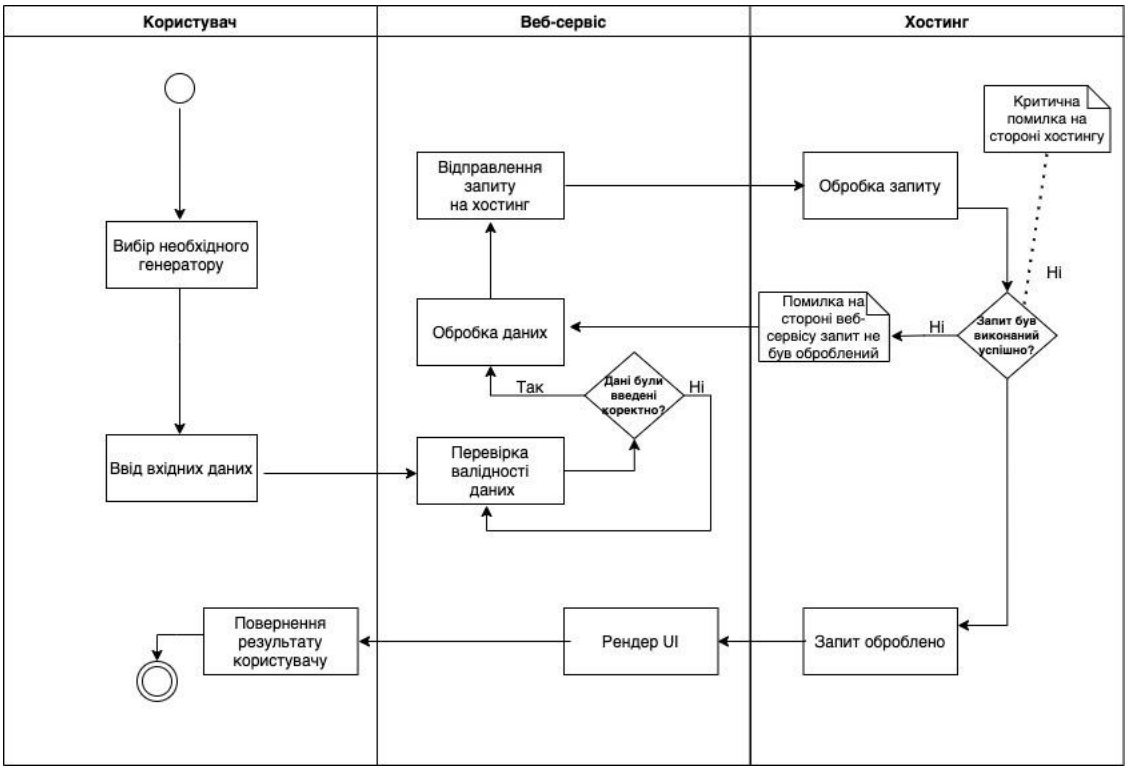

Рисунок 2.3 – Діаграма-діяльності додатку

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Розробка сайтів та вебзастосунків є надзвичайно важливою в сучасному цифровому світі, оскільки Інтернет став не тільки головним джерелом інформації, але й основним каналом комунікації між бізнесом та його клієнтами. Якісно розроблений сайт чи вебзастосунок може забезпечити компанії значну перевагу над конкурентами, привернути нових клієнтів і збільшити прибуток.

Розробка сайтів та вебзастосунків включає в себе весь процес створення сайту – від планування його структури і дизайну до написання коду і тестування на різних пристроях і браузерах. На даний момент існує безліч технологій і інструментів, що дозволяють створювати якісні та ефективні вебсайти.

HTML (HyperText Markup Language) – це стандартизована мова розмітки, яка використовується для створення структури вебсторінок та їхнього відображення у браузері. Веббраузери отримують HTML-документи з сервера

через протоколи HTTP/HTTPS або зчитують їх з локального носія, після чого перетворюють код на візуальний інтерфейс для користувача.

Ідея HTML бере початок із 1980 року, коли фізик Тім Бернерс-Лі, працюючи у CERN, розробив прототип системи INQUIRE для зручного обміну науковими документами. У 1989 році він запропонував створити гіпертекстову інформаційну систему, яка згодом стала основою Інтернету. Наприкінці 1990 року Бернерс-Лі створив першу версію HTML, а також програмне забезпечення для веббраузера і сервера. Хоча проєкт офіційно не був схвалений у CERN, він став визначальним кроком у розвитку Всесвітньої павутини. У своїх нотатках Бернерс-Лі вже тоді вказував на широке застосування гіпертексту, зокрема у створенні електронних енциклопедій [5].

Ключова ідея мови HTML – організація вмісту за допомогою елементів, що не зазнала жодних змін з ранніх часів Всесвітньої павутини. Більше того, навіть дуже старі вебсторінки без проблем обробляються в найсучасніших браузерах (включно з тими, які не існували на момент створення цих сторінок, наприклад Firefox або Chrome).

HTML-розмітка складається з основних структурних елементів, серед яких ключову роль відіграють теги та їх атрибути, типи даних, символи та сутності. Більшість тегів використовуються парами – наприклад, <h1> та </h1>, де перший є відкриваючим тегом, а другий – закриваючим. Водночас існують і самозакриваючі теги, такі як <img>, вони не потребують пари.

Увесь вміст HTML-документа розміщується між тегами <html> і </html>, а безпосереднє візуальне наповнення сторінки – всередині блоку <body>. Наприклад, тег <h1> Це заголовок </h1> позначає заголовок першого рівня, який браузер відображає як найбільш пріорітетний. Елемент виконує функцію контейнера, в якому зручно групувати інші елементи для організації структури сторінки [6].

CSS (Каскадні таблиці стилів) є потужною технологією, що дозволяє визначати правила оформлення HTML документів, розділяючи їх від змісту. Ця прогресивна технологія має довгу історію розвитку і постійно вдосконалюється.

Основне призначення CSS полягає в тому, щоб відокремити візуальне оформлення вебсторінки від її змісту. Це включає налаштування розташування елементів, кольорової гами, шрифтів тощо. Такий підхід підвищує гнучкість у керуванні стилями, полегшує доступність контенту та дозволяє використовувати спільне форматування для кількох сторінок через зовнішній файл стилів «.css». Завдяки цьому зменшується повторення коду, спрощується підтримка структури документа і забезпечується швидше завантаження сторінок за рахунок кешування стилів.

CSS вперше було запропоновано в 1994 році, саме тоді, коли Інтернет почав справді набувати популярності. У той час браузери надавали користувачеві різноманітні можливості щодо стилів – наприклад, параметри представлення в Mosaic дозволяли користувачеві визначати всі види сімейства шрифтів, розміру та кольору для кожного елемента. Нічого з цього не було доступно для авторів документів: все, що вони могли зробити, це позначити частину вмісту як абзац, як заголовок певного рівня, як попередньо відформатований текст або один із кількох інших типів елементів [6-7].

Переваги CSS:

– CSS сприяє публікації вмісту в різних форматах презентацій на основі номінальних параметрів. Номінальні параметри включають явні вподобання користувачів, різні веббраузери, тип пристрою, який використовується для перегляда вмісту (настільний комп’ютер або мобільний пристрій), географічне розташування користувача та багато інших змінних;

– таблиця стилів – незалежно від того, внутрішня вона чи зовнішня – дозволяє задавати оформлення одразу для групи HTML-елементів на основі їхнього типу, класу або взаємозв’язків з іншими елементами. Це значно ефективніше, ніж повторювати стилі для кожного окремого елемента вручну. Зовнішній файл стилів може зберігатися в кеші браузера та використовуватися повторно на різних сторінках, що зменшує обсяг передаваних даних і пришвидшує завантаження сайту;

– коли CSS використовується ефективно, з точки зору успадкування та «каскадування», глобальна таблиця стилів може використовуватися для впливу та елементів стилів на всьому сайті. Якщо виникає ситуація, що стиль елементів слід змінити або відкоригувати, ці зміни можна внести шляхом редагування правил у загальній таблиці стилів. До CSS такий вид обслуговування був більш складним, дорогим і трудомістким;

– вебзастосунки можуть зберігати CSS-файли локально завдяки офлайн-кешу, що дозволяє користувачам переглядати сайти навіть без підключення до Інтернету. Крім того, кешування сприяє швидшому завантаженню сторінок і загальному підвищенню продуктивності вебресурсу.

SASS (Syntactically Awesome Stylesheets) – це мова стилів, створена Хемптоном Кетліном, яка є розширенням CSS і працює як препроцесор. Вона компілюється у звичайний CSS-код, надаючи розробникам ширші можливості для організації стилів. SASS дозволяє використовувати змінні, вкладені правила, міксини, імпорт інших файлів та вбудовані функції для роботи з кольорами та іншими значеннями. Завдяки цьому мова є більш потужною, гнучкою та зручною у підтримці, зберігаючи при цьому повну сумісність із CSS.

Розширення файлів SASS може бути « .sass » або « .scss », залежно від обраного синтаксису. Однак, веббраузери не розуміють цих форматів напряму, тому для використання їх у вебпроєктах потрібно скомпілювати їх до звичайного CSS. Компілятор SASS перетворює SASS-файли в зрозумілий для браузерів CSS-код. Цю роль компілятора може виконувати серверний JavaScript або програма, встановлена на робочій машині, які відстежують зміни в робочих файлах і автоматично компілюють їх у CSS.

SCSS є препроцесором CSS, який допомагає уникнути повторень в коді CSS та ефективно використовувати час. Він вважається потужнішим та стабільнішим, порівняно з CSS, і дозволяє чітко та структурно описувати стиль документа. Це сприяє швидшому та якіснішому виконанню роботи. SCSS надає багато особливостей, і він часто використовується замість звичайного CSS. Зокрема, SCSS дозволяє визначати змінні, які починаються зі знаку « $ » і

присвоювати їм значення за допомогою двокрапки. Також він підтримує вкладений код, який відсутній в CSS. Формат SCSS представляється у вигляді окремих файлів з розширенням « .scss », наприклад, style.scss.

SASS забезпечує низку розширених можливостей, які роблять написання стилів більш зручним і ефективним порівняно зі звичайним CSS. Зокрема, вкладення дозволяє групувати селектори всередині батьківських, що підвищує читабельність і зменшує дублювання коду – можливість, якої не підтримує стандартний CSS.

Змінні в SASS працюють аналогічно до змінних у традиційних мовах програмування: вони мають типи, області видимості та дозволяють зберігати значення, які часто повторюються – наприклад, кольори, розміри шрифтів або відступи. Це значно спрощує оновлення та підтримку стилів.

Крім того, SCSS підтримує розширену роботу з операторами, у тому числі з математичними виразами, операцією ділення з остачею « % » та іншими – навіть для типів даних, таких як кольори чи рядки. І хоча CSS також має функцію calc (), SCSS є гнучкішим і швидшим у написанні обчислень.

Ще однією перевагою є наявність великої кількості вбудованих функцій, які охоплюють обробку кольорів, рядків, чисел. Наприклад, функції random (), round (), darken () та інші дозволяють автоматизувати обчислення та покращити логіку стилізації [8].

JavaScript (скорочено JS) – це високорівнева мова програмування, яка відповідає специфікації ECMAScript. Вона підтримує кілька парадигм програмування, включно з об’єктно-орієнтованою на основі прототипів, імперативною та функціональною. JavaScript характеризується синтаксисом з використанням фігурних дужок, динамічною типізацією та підтримкою функцій як об’єктів першого класу.

JavaScript є багатопарадигменною мовою програмування, яка підтримує імперативний, функціональний і подієво-орієнтований підходи. Вона надає низку вбудованих API для роботи з текстом, датами, регулярними виразами, структурами даних і об'єктною моделлю документа (DOM). При цьому функції

введення-виведення (наприклад, доступ до мережі, файлових систем або графіки) не входять до самої мови, а реалізуються середовищем виконання, найчастіше – веббраузером.

Разом із HTML і CSS, JavaScript становить основу сучасної веброзробки. Він забезпечує інтерактивність вебсторінок і є ключовим елементом у створенні динамічних вебзастосунків. Більшість сучасних вебсайтів використовують JavaScript для реалізації поведінки на стороні клієнта, а всі основні браузери містять вбудовані рушії JavaScript для його виконання [9].

У листопаді 1996 року компанія Netscape передала мову JavaScript до організації ECMA International, щоб розробити єдину специфікацію, яка б стала спільним стандартом для всіх розробників браузерів. Це призвело до появи першої офіційної версії стандарту – ECMAScript, яка була опублікована в червні 1997 року. У наступні роки з'явилися ECMAScript 2 (1998) та ECMAScript 3 (1999). Паралельно в 2000-х роках почалась робота над ECMAScript 4, однак її завершити не вдалося.

У цей період браузер Internet Explorer від Microsoft досяг майже повної домінації на ринку – близько 95 %, що зробило мову JavaScript де-факто стандартом для вебсценаріїв. Спочатку Microsoft брала участь у стандартизації, але згодом припинила співпрацю з ECMA, що фактично зупинило розробку ECMAScript 4.

Сучасний JavaScript позиціонується як безпечна мова програмування, яка не дозволяє прямого доступу до пам'яті або процесора. Це пов'язано з тим, що спочатку мова створювалась для роботи у веббраузерах, де такі можливості були непотрібними.

Можливості JavaScript значною мірою визначаються середовищем, у якому вона виконується. Наприклад, у середовищі Node.js JavaScript отримує доступ до функцій для роботи з файловою системою, мережевими запитами та іншими можливостями серверного програмування.

Натомість у браузерному середовищі мова обмежена функціоналом, пов'язаним із взаємодією з вебсторінками, користувачем і сервером через

інтернет-протоколи. У такому режимі JavaScript не має прямого доступу до системних ресурсів комп'ютера – це обумовлено вимогами безпеки. Наприклад, вона може закрити лише ті вкладки, які були відкриті скриптом.

У початковому задумі JavaScript була мовою для браузерів і вебвзаємодії, тому її можливості поза цим контекстом обмежені без додаткових інструментів чи платформ [10].

Після ґрунтовного вивчення JavaScript можна вдосконалювати свої навички практично у будь-якому напрямку. Для створення сучасних вебзастосунків варто освоїти фреймворки та бібліотеки, які дозволять користуватися ефективними наборами функціональних класів. До переліку найпоширеніших технологій цієї групи входять jQuery, Angular та React. Також варто звернути увагу на надбудови TypeScript, Dart та CoffeeScript. Залежно від обраної технології, вона може сприяти підвищенню лаконічності, структурованості або читабельності програмного коду.

React – це відкрита JavaScript-бібліотека, призначена для створення інтерфейсів користувача, зокрема в односторінкових вебзастосунках. React було розроблено компанією Meta у співпраці зі спільнотою, щоб ефективно вирішувати проблему часткового оновлення контенту без повного перезавантаження сторінки.

Головна мета React – забезпечити швидку, зручну та масштабовану розробку інтерфейсів. Ця бібліотека дозволяє створювати складні додатки з динамічним відображенням даних, зосереджуючись виключно на візуальній частині. React реалізує принципи, близькі до архітектури MVC, і легко інтегрується з іншими бібліотеками або фреймворками, такими як AngularJS чи Redux. Завдяки модульному підходу та гнучкості React може використовуватись як окрема бібліотека, так і частина більш складної екосистеми для управління станом і логікою додатка.

React було створено Джорданом Волке, розробником із компанії Facebook, під впливом XHP – HTML-фреймворку для PHP. Уперше бібліотека з'явилася в

новинній стрічці Facebook у 2011 році, а вже за рік була інтегрована у блозі Instagram.

Однією з ключових особливостей React є використання віртуального DOM (VDOM) замість прямої роботи з DOM браузера. Це дозволяє ефективно визначати та оновлювати лише ті частини інтерфейсу, які зазнали змін. React зберігає копію віртуального DOM, порівнює її з новим станом (процес diff) і вносить лише необхідні оновлення в реальний DOM. Завдяки цьому розробник може писати код так, ніби вся сторінка оновлюється повністю, тоді як React самостійно оптимізує процес оновлення [11-12].

Такий підхід забезпечує декларативність API: розробник описує бажаний стан інтерфейсу, а бібліотека забезпечує його відображення, автоматично керуючи змінами, подіями та атрибутами. Поняття «віртуального DOM» у контексті React стосується елементів React – об’єктів, що описують структуру та стан інтерфейсу, і є ключем до ефективної та продуктивної роботи бібліотеки.

Під час додавання нових елементів до інтерфейсу користувача React створює віртуальну модель DOM у вигляді дерева, де кожен елемент інтерфейсу є окремим вузлом. У разі змін в інтерфейсі формується нове віртуальне дерево, яке порівнюється з попередньою версією. На основі цього порівняння React визначає найоптимальніший спосіб оновлення реального DOM. Завдяки цьому зміни застосовуються лише до тих частин сторінки, які справді змінилися, що суттєво знижує кількість маніпуляцій із реальною DOM і, відповідно, покращує продуктивність [13].

JSX – розширення JavaScript, що надає синтаксичний цукор для викликів функцій і побудови об’єктів, особливо React.createElement (). На перший погляд код JSX може здатися чимось на зразок шаблонизатора або HTML, але це не так. JSX будує елементи React, дозволяючи вам користуватися всією потужністю JavaScript. JSX – відмінний інструмент для запису компонентів React. Його основні переваги:

– поліпшене сприйняття з боку розробника (DX, Developer Experience), код простіше читається через його виразність, обумовлену XML-подібним

синтаксисом, що краще підходить для представлення вкладених декларативних структур;

– підвищення ефективності роботи команд – недосвідченим розробникам (наприклад, вебдизайнерам) простіше змінювати цей код, тому що JSX нагадує розмітку HTML, яка їм вже знайома;

– зниження навантаження на пальці та зменшення синтаксичних помилок – розробникам доводиться набирати менше коду, а це означає, що вони зроблять менше помилок і їм доведеться виконувати менше роботи [14].

React надає можливість керувати поведінкою компонента і налаштовувати його на підставі подій його життєвого циклу. Ці події поділяються на такі категорії:

– події підключення – при підключенні елемента React (екземпляра компонента класу) до вузла DOM;

– події оновлення – відбуваються при оновленні елемента React у результаті зміни значень його властивостей або стану;

– події відключення – відбуваються при від’єднанні елемента React від моделі DOM.

Починаючи з версії React 16.8, у бібліотеці з’явилася нова концепція – хуки. React-хуки є спеціальними функціями, які дозволяють використовувати стан та логіку життєвого циклу всередині функціональних компонентів, без потреби у класах.

React надає набір вбудованих хуків, серед яких найбільш поширеними є useState – для роботи зі станом компонента, та useEffect – для керування побічними ефектами. Інші корисні хуки, такі як useContext та useReducer, також описані в офіційній документації React і дозволяють гнучко реалізовувати логіку компонента.

Використання хуків загалом означає наявність побічних ефектів, які дозволяють компоненту взаємодіяти зі своїм станом та зовнішніми системами, такими як введення-виведення. Побічні ефекти включають будь-яку зміну стану,

яка видима поза межами функції, за винятком значення, яке повертається функцією.

Загалом рекомендується використовувати функціональні компоненти та хуки, а не компоненти, засновані на класах. Функціональні компоненти зазвичай мають більш компактний код у порівнянні з компонентами на основі класів. Вони мають кращу організацію коду, що сприяє більшій читабельності і легкості перевикористання. Також їх легше тестувати.

SPA (Single Page Application), або односторінковий застосунок – це популярний підхід до створення сучасних вебресурсів, за якого основні HTML-, CSS- та JavaScript-файли завантажуються одразу під час першого відкриття сайту. Надалі всі дії користувача, такі як переходи між сторінками, відбуваються без повного перезавантаження сторінки – браузер просто оновлює вміст динамічно, що забезпечує швидку та безперервну взаємодію [12-13].

Односторінкові застосунки на JavaScript-фреймворках підвищують зручність інтерфейсу для відвідувача завдяки безперервності. Перейшовши на сторінку вебсайту, ви відразу зможете визначити – це SPA або багатосторінкова програма: необхідно просто натиснути на кілька посилань у навігації. Багатосторінковий додаток буде перезавантажуватися, змушуючи весь інтерфейс користувача швидко блимати в залежності від вмісту, відбувається подібне через оновлення сайту. Навпаки, SPA плавно відображається завжди у будь-який момент, оскільки така програма просто показує користувачеві інший контент без оновлення сторінки вебсайту в його браузері.

Node.js – це середовище виконання JavaScript з відкритим вихідним кодом, яке дозволяє запускати код поза межами браузера. Платформу було створено у 2009 році Райаном Далем для розробки високопродуктивних і масштабованих мережевих застосунків. До появи Node.js подібні завдання виконувалися за допомогою таких серверів, як Apache HTTP Server, однак вони мали обмеження у роботі з великою кількістю одночасних з’єднань.

Даль критикував традиційний підхід, притаманний Apache, де обробка запитів відбувалася послідовно, що призводило або до блокування потоку, або до

перевантаження системи через велику кількість паралельних стеків виконання. Як альтернативу він запропонував неблокуючу, подієво-орієнтовану модель, яку й реалізував у Node.js.

Node.js поєднує у собі JavaScript-движок Google V8, цикл обробки подій та низькорівневі API введення/виведення, що дозволяє ефективно обробляти велику кількість одночасних запитів і забезпечувати високу продуктивність. Вперше проєкт було представлено 8 листопада 2009 року на конференції JSConf Europe. У січні 2010 року, менеджер пакетів було введено для середовища Node.js з назвою NPM. Менеджер пакетів полегшує програмістам публікацію та спільний доступ до вихідного коду пакетів Node.js і призначений для спрощення встановлення, оновлення та видалення пакетів.

Переваги використання Node.js:

– є доступним для освоєння, особливо для тих, хто вже знайомий із JavaScript та об'єктно-орієнтованим програмуванням. Це одна з причин, чому багато розробників обирають саме це середовище;

– реалізує концепцію «JavaScript everywhere», що дозволяє використовувати одну мову як для фронтенду, так і для бекенду. Це зменшує кількість коду, файлів і спрощує підтримку проєкту;

– в середовищі Node.js компіляція зазвичай відбувається швидше, ніж у багатьох альтернатив, що дозволяє скоротити терміни створення продукту та швидше виводити його на ринок – особливо важливо при запуску MVP;

– був створений як масштабована альтернатива Apache. Завдяки неблокуючій моделі обробки запитів він здатен ефективно працювати з великою кількістю одночасних підключень, що критично для бізнесів, які планують зростання;

– розвинене ком’юніті забезпечує потужну підтримку, обмін досвідом і розробку інструментів з відкритим кодом, які значно полегшують і прискорюють процес створення застосунків [15].

NPM (Node Package Manager) – це менеджер пакетів для JavaScript, який є невід’ємною частиною екосистеми Node.js. Він складається з інтерфейсу

командного рядка (CLI) та онлайн-реєстру, що містить як публічні, так і приватні пакети. Через CLI-інтерфейс користувачі можуть встановлювати, оновлювати та керувати залежностями у своїх проєктах, а також публікувати власні модулі.

NPM автоматично встановлюється разом із Node.js і вважається рекомендованим інструментом для роботи з бібліотеками. Його реєстр містить сотні тисяч JavaScript-пакетів, побудованих переважно на основі формату CommonJS із метаданими у форматі JSON.

Попри велику кількість доступних модулів, реєстр NPM не має суворої системи перевірки перед публікацією. Тому пакети можуть суттєво відрізнятись за якістю чи рівнем безпеки. Оцінювати надійність допомагають статистика завантажень, кількість залежностей та відгуки спільноти. У разі виявлення порушень політики пакети можуть бути вилучені на основі скарг користувачів.

Крім публічного реєстру, багато компаній використовують NPM для управління внутрішніми залежностями у приватних проєктах, що спрощує організацію розробки у великих командах [16].

JSON (JavaScript Object Notation) – це текстовий формат для обміну даними між комп’ютерними системами, який є простим для читання людиною та зручним для машинної обробки. Формат дає змогу описувати об’єкти та інші структуровані дані, що робить його придатним для передачі інформації в мережевому середовищі, зокрема під час серіалізації.

Формат JSON був розроблений і популяризований Дугласом Крокфордом. Найширше застосування JSON отримав у веброзробці, зокрема в поєднанні з технологією AJAX. У цьому контексті JSON замінив XML завдяки своїй легкості, меншому обсягу та природному поєднанню з JavaScript, оскільки дані у форматі JSON можна напряму перетворити в об’єкти JavaScript.

Структурно JSON базується на двох основних конструкціях: наборі пар «ключ/значення» (аналогічно до об'єктів, словників або хеш-таблиць у різних мовах програмування) та впорядкованому списку значень (що відповідає масиву чи списку). Завдяки цим властивостям JSON став стандартом для обміну даними в сучасних вебзастосунках.

У більшості мов програмування впорядкований список значень реалізується як масив, список, вектор або послідовність. У JSON використовуються такі форми структур даних:

– об’єкт – це набір пар «ключ-значення», який починається фігурною дужкою { і закінчується }. Кожна пара розділяється комою, а ключ і значення відділяються двокрапкою. У ролі значень можуть виступати різні типи даних;

– масив – це впорядкована колекція значень, яка розміщується між квадратними дужками [і], при цьому елементи також відділяються комами;

– значення може бути рядком у подвійних лапках, числом, логічним значенням (true, false), null, а також іншим об’єктом або масивом. Допускається вкладеність структур;

– рядок у JSON – це послідовність символів Unicode, обмежена подвійними лапками. У випадку спеціальних символів використовуються escape-послідовності зі зворотною косою рискою « \ ».

Ці базові конструкції дозволяють описувати гнучкі та зрозумілі структури даних для обміну між клієнтом і сервером [17].

Висновки до розділу 2

У другому розділі було визначено вимоги до розроблюваного програмного забезпечення та окреслено основні етапи його проєктування. Було проведено аналіз функціональних можливостей подібних інструментів, сформульовано перелік ключових завдань і визначено функціональні та нефункціональні вимоги до вебзастосунку, орієнтованого на автоматизацію створення CSS-стилів.

З метою кращого розуміння структури майбутньої системи та принципів її роботи, проведено попереднє проєктування архітектури програмного забезпечення з використанням нотації UML. Зокрема, за допомогою діаграми прецедентів, діаграми послідовності та діаграми діяльності було відображено основні сценарії взаємодії користувача із системою, внутрішню логіку обробки

запитів, перевірку даних та обробку результатів на стороні вебсервісу та хостингу.

Під час вибору технологій для реалізації вебзастосунку було обґрунтовано доцільність використання HTML5, CSS3 (SASS) та JavaScript як базових мов веброзробки, що забезпечують основу для створення інтерфейсу. Окрім цього, застосування сучасних бібліотек і фреймворків – зокрема React для клієнтської частини та Node.js для серверної логіки, яка дозволяє побудувати гнучку, адаптивну та динамічну архітектуру, яка легко масштабується відповідно до зростання функціональних потреб проєкту, а також NPM для використання необхідних при розробці пакетів та JSON для структурованості даних.

РОЗДІЛ 3
РОЗРОБКА БАГАТОФУНКЦІОНАЛЬНОГО CSS-ГЕНЕРАТОРА СТИЛІВ З ВИКОРИСТАННЯМ REACT

3.1 Практична реалізація об’єкта проєктування

Розробка багатофункціонального CSS-генератора стилів виконувалась з використанням бібліотеки React, яка дала змогу побудувати інтерфейс на основі гнучкої компонентної структури. Завдяки реалізації у форматі односторінкового застосунку (SPA), усі дії користувача виконуються без оновлення сторінки, що забезпечує швидку реакцію інтерфейсу та загалом робить взаємодію з додатком більш зручною та динамічною.

Важливо було реалізувати функціональні, та нефункціональні вимоги додатку, усі задачі було досягнуто за допомоги компонентів бібліотеки React. Саме компонентний підхід React, а також односторінкова безперервність, дозволила розробити багатофункціональний CSS-генератор стилів. Також, важливо відзначити використання пакетів NPM, які полегшили деякі етапи розробки, а також дали змогу зменшити наватаження додатку, та досягти максимальної швидкості роботи генератора і відповідної оптимізації.

Основними вимогами до багатофункціонального CSS-генератора стилів є:

– зручний та зрозумілий для використання (User-friendly) інтерфейс додатку;

– розподілення генераторів по відповідним групам, а також функціям;

– можливість зберігати у буфері обміну згенерований код CSS-стилю;

– адаптивність додатку для мобільних пристроїв;

– перегляд згенерованого CSS-стилю в реальному часі.

Бібліотека React стала головною при створенні багатофункціонального CSS-генератора. Для запуску додатку використовується команда «npx create-react-app», після відтворення цієї команди встановлюються відповідні залежності для маршрутизації між генераторами, а також для коректного управління станами компонентів, та для використання React-хуків.

На рисунку 3.1 зображено, як відбувається розгортання застосунку, також діаграма дає можливість показати компоненти та об’єкти, які використовуються при роботі багатофункціонального CSS-генератора стилів, нижче наведено відповідну діаграму розгортання.

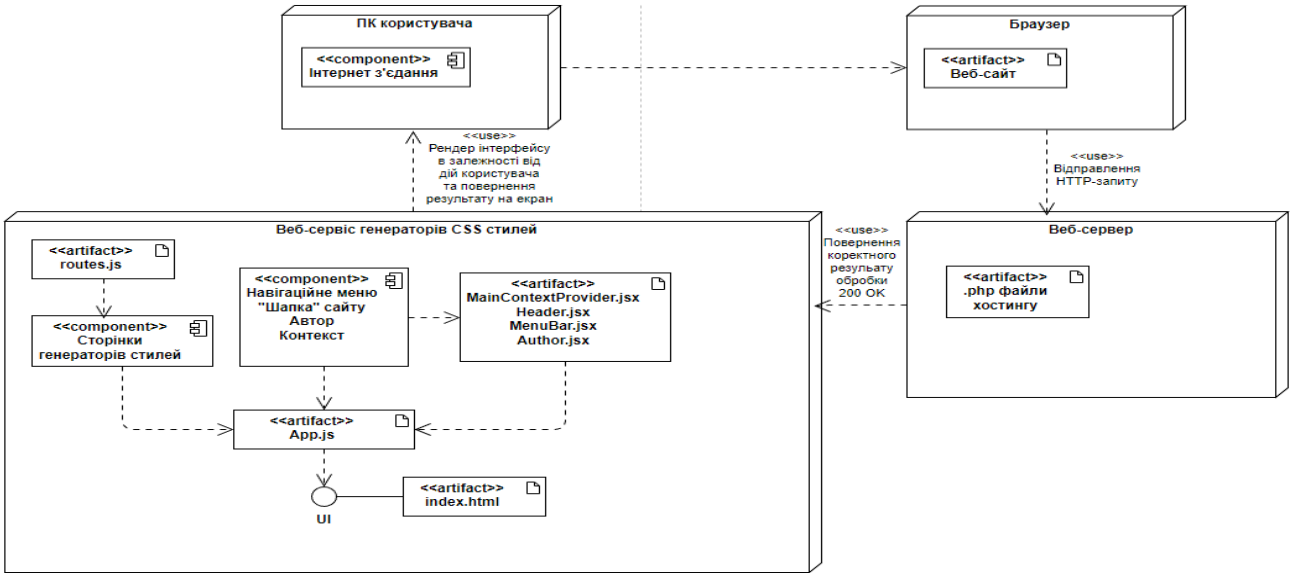

Рисунок 3.1 – Діаграма розгортання

Основні компоненти багатофункціонального додатку:

– app – головний компонент додатку, який відповідає за його розгоратання, коректну маршрутизацію по генераторам, та глобальні налаштування додатку;

– home – цей компонент є головною сторінкою застосунку, яка відображає групи навігаційних кнопок, сформованих на основі даних з масиву. Кожна кнопка веде до відповідного маршруту, а їхній вигляд оформлено за допомогою бібліотеки react-awesome-button. Анімація входу та виходу реалізована через framer-motion, а контекст MainContext використовується для керування станом меню;

– mainContextProvider – цей компонент створює глобальний контекст для керування станами в React-застосунку. Він надає загальний доступ до таких змінних, як: відкриття/закриття меню, копіювання згенерованого коду в буфер обміну, а також анімаційні варіанти для елементів;

– menuBar – даний компонент відповідає за відображення навігаційного меню у React-застосунку. Він використовує контекст MainContext, щоб отримати стан для керування видимістю меню та функцію для його закриття. Меню формується динамічно на основі масиву, де кожен пункт та його підпункти (маршрути) відображаються як посилання з бібліотеки react-router-dom.

Наступний фрагмент коду (ліст. 3.1) описує головний компонент React-застосунку App. Він використовує хук useRoutes для підключення маршрутизації та обгортає весь застосунок у контекст MainContextProvider, що надає глобальні стани. Компоненти Header, MenuBar, Author та динамічний контент (routes) відображаються всередині AnimatePresence для забезпечення анімацій при переходах між сторінками. Компонент відповідає за загальну структуру та логіку відображення інтерфейсу.

Лістинг 3.1 – Компонент App

```
//...

function App() {
    let routes = useRoutes(router);

    return (
        <>
            <MainContextProvider>
                <AnimatePresence>
                    <Header />
                    <div className='container_App'>
                        <MenuBar />
                        {routes}
                    </div>
                    <Author />
                </AnimatePresence>
            </MainContextProvider>
        </>
    );
}
export default App;
```

кінець лістингу 3.1

Кодова реалізація компоненту Home (ліст. 3.2) відповідає за відображення головної сторінки вебзастосунку. Він отримує анімаційні параметри

(mainVariant) та функцію закриття меню (closeMenu) з глобального контексту. За допомогою motion.div з бібліотеки framer-motion реалізовано анімовану появу контенту. Компонент проходить по масиву Datas і для кожної групи відображає її назву та пов’язаний список навігаційних кнопок, створених з використанням AwesomeButton. Останнім додається компонент GitHubLink, що веде до репозиторію розробника. Компонент забезпечує зручну навігацію до генераторів стилів та візуально структурує доступний функціонал.

Лістинг 3.2 – Компонент Home

```
//...

export default function Home() {
    const { mainVariant, closeMenu } = useContext(MainContext);

    return (
        <motion.div
            variants={mainVariant}
            initial="hidden"
            animate="visible"
            exit="exit"
            className='home_Container'>
            {Datas.map((data) => (
                <div key={data.id} className="group_container">
                    <span className="title_group">{data.title}</span>
                    <div className="group_btn">
                        {data.routeAddress.map((routes) => (
                            <Link onClick={closeMenu} key={routes.id}
to={routes.route}>

<AwesomeButton>{routes.name}</AwesomeButton>
                            </Link>
                        ))}
                    </div>
                </div>
            ))}
            <GitHubLink />
        </motion.div>
    );
}
```

кінець лістингу 3.2

Компонент MainContextProvider відображенний у лістингу 3.3 забезпечує глобальний контекст для застосунку, надаючи спільний доступ до кількох станів і функцій: активація меню (menuActive), повідомлення про копіювання (copyClickText), функція анімації переходів між сторінками (mainVariant) і прокрутка вгору при закритті меню (closeMenu). Усі ці значення передаються дочірнім компонентам через MainContext.Provider, що дозволяє централізовано керувати поведінкою інтерфейсу.

Лістинг 3.3 – Компонент MainContextProvider

```
//...

export const MainContext = createContext()

export default function MainContextProvider({ children }) {
    const [menuActive, setMenuActive] = useState(false)
    const [copyClickText, setcopyClickText] = useState(false)

    const btnCopyTextChange = () => {
        setcopyClickText(true)
        setTimeout(() => {
            setcopyClickText(false)
        }, 3000);
    }

    const closeMenu = () => {
        setMenuActive(false)
        window.scrollTo({ top: 0, behavior: 'smooth' });
    }

    const mainVariant = {
        hidden: {
            y: "-5rem",
            opacity: 0
        },
        visible: {
            y: 0,
            opacity: 1
        },
        exit: {
            y: "5rem",
            opacity: 0
        }
    }
    return (
        <MainContext.Provider
```

```
            value={{
                menuActive,
                setMenuActive,
                copyClickText,
                setcopyClickText,
                btnCopyTextChange,
                mainVariant,
                closeMenu
            }}
        >
            {children}
        </MainContext.Provider>
    )
}
```

кінець лістингу 3.3

Компонент MenuBar відповідає за побудову бічного навігаційного меню. Він отримує стан menuActive та функцію closeMenu з глобального контексту MainContext. Меню формується динамічно на основі масиву Datas, де кожна група має назву та перелік маршрутів. При натисканні на будь-який пункт меню викликається closeMenu, що закриває меню та повертає користувача до верхньої частини сторінки. Компонент забезпечує швидку навігацію по застосунку та адаптивну взаємодію з інтерфейсом.

Кодову реалізацію цього компоненту, можна побачити у лістингу 3.4.

Лістинг 3.4 – Компонент MenuBar

```
//...

export default function MenuBar() {
    const { menuActive, closeMenu } = useContext(MainContext);

    return (
        <div className={`menu ${menuActive && 'active'}`}>
            <Link onClick={closeMenu} to={'/'}
className='btn_Menu'>Home</Link>
            {Datas.map((data) => (
                <div key={data.id} className="group_Menu">
                    <span className="title_Menu">{data.title}</span>
                    <div className="Menu_btn">
                        {data.routeAddress.map((routes) => (
                            <Link
                                key={routes.id}
```

```
                            onClick={closeMenu}
                            to={routes.route}
                            className='btn_Menu'>
                            {routes.name}
                        </Link>
                    ))}
                </div>
            </div>
        ))};
```

кінець лістингу 3.4

Структура вебзастосунку повинна бути складною, але водночас зрозумілою та змістовною. Головна сторінка має багато інструментів CSS-стилів, які в свою чергу діляться на групи такі як: background, box, filter, text, transform, flexbox.

Основними сторінками додатку є сторінки інструментів створення CSS-стилів, таких як:

– background color – інструмент для створення стилів кольору заднього фону;

– background gradient – інструмент для створення стилів градієнтних кольорів заднього фону та вибору їх напрямку (радіальний або лінейний);

– border – інструмент для створення стилів рамки елементу в свою чергу він має усі можливі типи стилізування рамки;

– border radius – інструмент для створення стилів заокруглення рамки;

– box shadow – інструмент для створення стилів тіні блоку, має можливість детально налаштувати тінь, наприклад: додати blur, змістити її горизонтально або вертикально і.т.д;

– opacity – інструмент для створення стилів видимості елемента;

– blur – інструмент для створення стилів створення ефекту розмиття картинки;

– sepia – інструмент для створення стилів налаштування властивостей насиченості картинки;

– brightness – інструмент для створення стилів налаштування властивостей яркості картинки;

– contrast – інструмент для створення стилів налаштування властивостей контрасту картинки;

– grayscale – інструмент для створення стилів налаштування властивостей фільтра сірого кольору картинки;

– heu-rotate – інструмент для створення стилів, які змінюють кольоровість зображення за рахунок повороту відтінку на колі;

– invert – інструмент для створення стилі інвертування кольору у картинки;

– saturate – інструмент для створення стилів налаштування властивостей насиченністю кольорів картинки;

– drop-shadow – інструмент для створення стилів тіні картинки та детального налаштування її властивостей;

– font-size – інструмент для створення стилів зміни розміру тексту у px;

– text-color – інструмент для створення стилів кольору тексту;

– text-decoration – інструмент для створення стилів декоративних елементів текст, наприклад таких як: перекреслений текст, підкреслений текст і.т.д;

– text-transform – інструмент для створення стилів трасформування букв у тексті, наприклад у заглавні чи навпаки;

– letter-spacing – інструмент для створення стилів інтервалу між буквами;

– line-height – інструмент для створення стилів інтервалу між строками;

– text-align – інструмент для створення стилів маніпулювання положенням тексту, завдяки цій властивості, можна наприклад відцентрувати текст або вирівняти його по лівому краю;

– font-weight – інструмент для створення стилів налаштування жирності тексту;

– translateX, translateY – інструменти для створення стилів, які дають змогу маніпулювати положенням елемента по осі X та Y;

– scaleX, scaleY – інструменти для створення стилів, які дають змогу маніпулювати розтягуванням елемента по осі X та Y;

– skewX, skewY – інструменти для створення стилів, які дають змогу маніпулювати трасформування елементу змінюючи його наклон у 2D-просторі по осі X та Y;

– rotate – інструмент для створення стилів обертання елементу по напрямку часової стрілки, може містити й від'ємні числа обертання;

– flexbox – потужний інструмент, який дає змогу користувачу маніпулювати полеженням елемента завдяки потрібним властивостям, наприклад: відцентрувати елемент по вертикалі та горизонталі, поставити елемент в кінець контейнера і т.д.

Проєкт повинен бути доступним 24/7 та мати uptime у 100 %, щоб дати змогу користувачам використовувати інструменти подані у додатку у своїх цілях та інтересах.

На головній сторінці вебзастосунку користувач бачить перелік доступних генераторів CSS-стилів, зручно згрупованих за типами властивостей. Така організація дозволяє швидко орієнтуватися в інтерфейсі та знаходити потрібний інструмент без зайвих зусиль. Наприклад, у групі Box зібрані генератори, які відповідають за візуальне оформлення блокових елементів, зокрема Border, Border Radius, Box Shadow та Opacity. Кожна з цих категорій відкриває доступ до окремого генератора, де користувач може змінювати параметри за допомогою зручних елементів керування – повзунків, перемикачів або полів введення – і миттєво бачити результат. Такий підхід не лише підвищує зручність користування, а й дозволяє ефективно експериментувати зі стилями без потреби вручну писати CSS-код. На рисунку 3.2 наведено інтерфейс головної сторінки вебзастосунку.

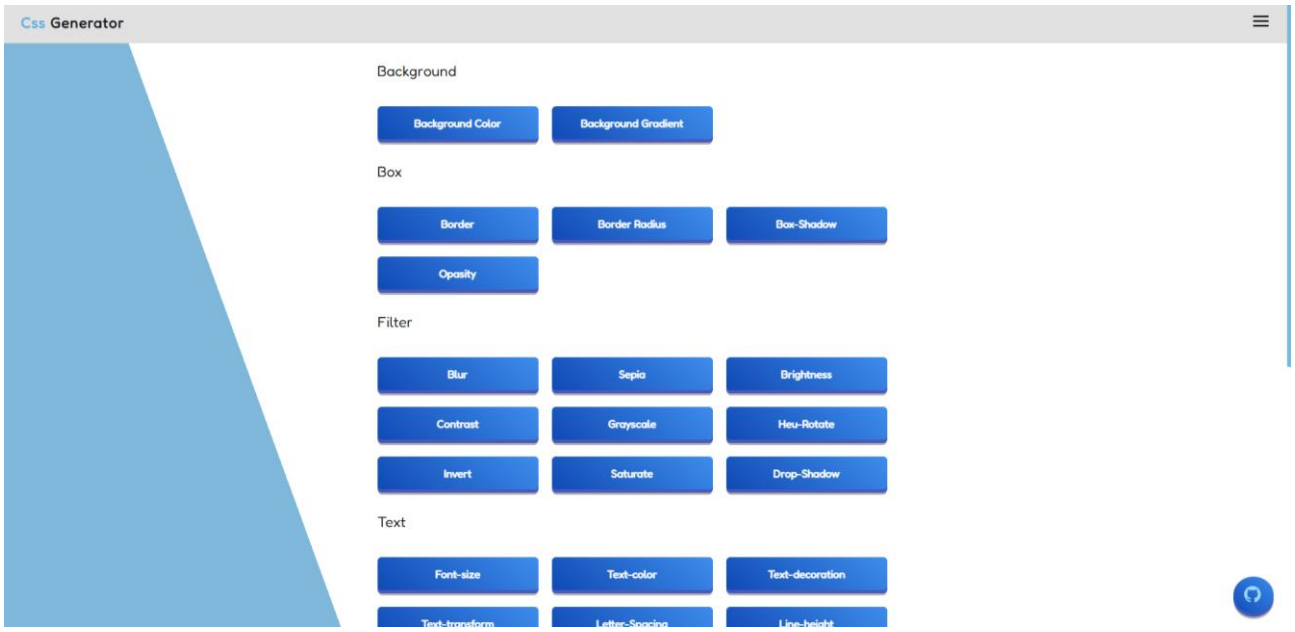

Рисунок 3.2 – Головна сторінка багатофункціонального CSS-генератора

Вебзастосунок має зручну головну сторінку з доступом до генераторів стилів та адаптивне бургер-меню, яке полегшує навігацію (рис. 3.3). Це меню дає змогу швидко перейти до потрібного генератора або повернутися на головну сторінку без оновлення сторінки. Воно адаптується до різних пристроїв, зберігаючи інтуїтивність, а також автоматично закривається після вибору пункту, покращуючи взаємодію.

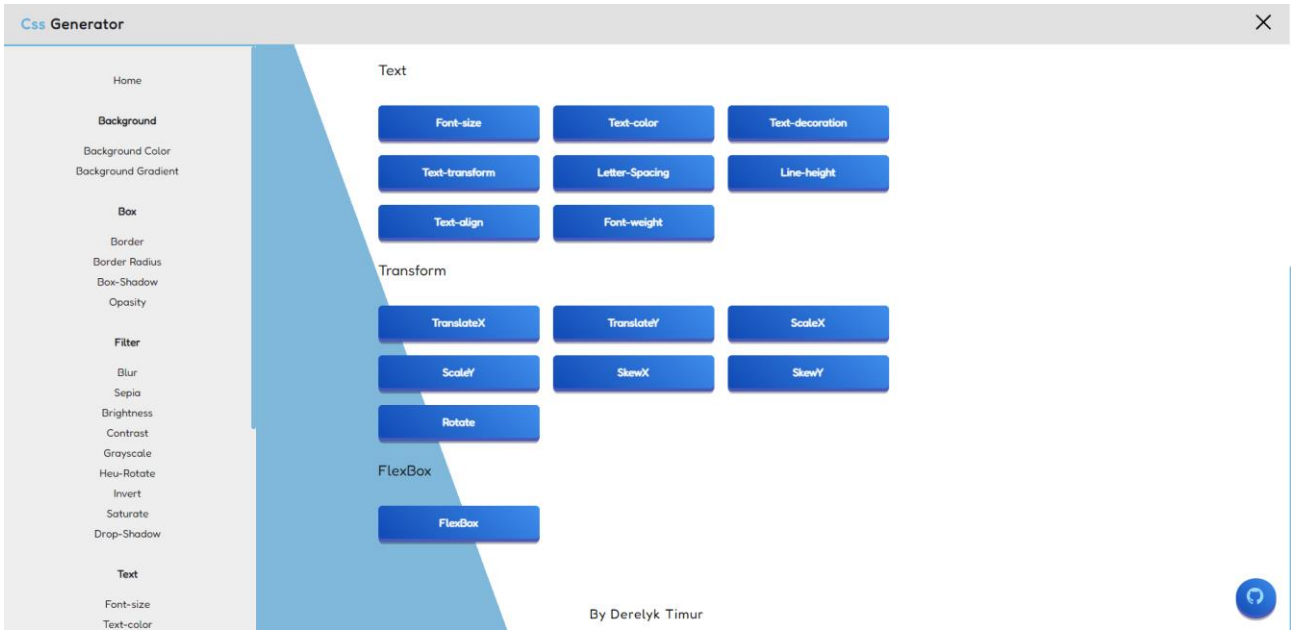

Рисунок 3.3 – Відображення «бургер-меню» вебзастосунку

Після переходу до обраного генератора стилів, користувачу відкривається зручний та зрозумілий інтерфейс, у якому він може налаштовувати параметри відповідно до своїх потреб. Кожен генератор містить набір інтуїтивних елементів керування, що дозволяють точно змінювати властивості CSS без необхідності вручну писати код. Однією з ключових особливостей є динамічний попередній перегляд: усі зміни миттєво відображаються на елементі прямо в інтерфейсі, що дає змогу користувачу бачити результат у реальному часі (рис. 3.4).

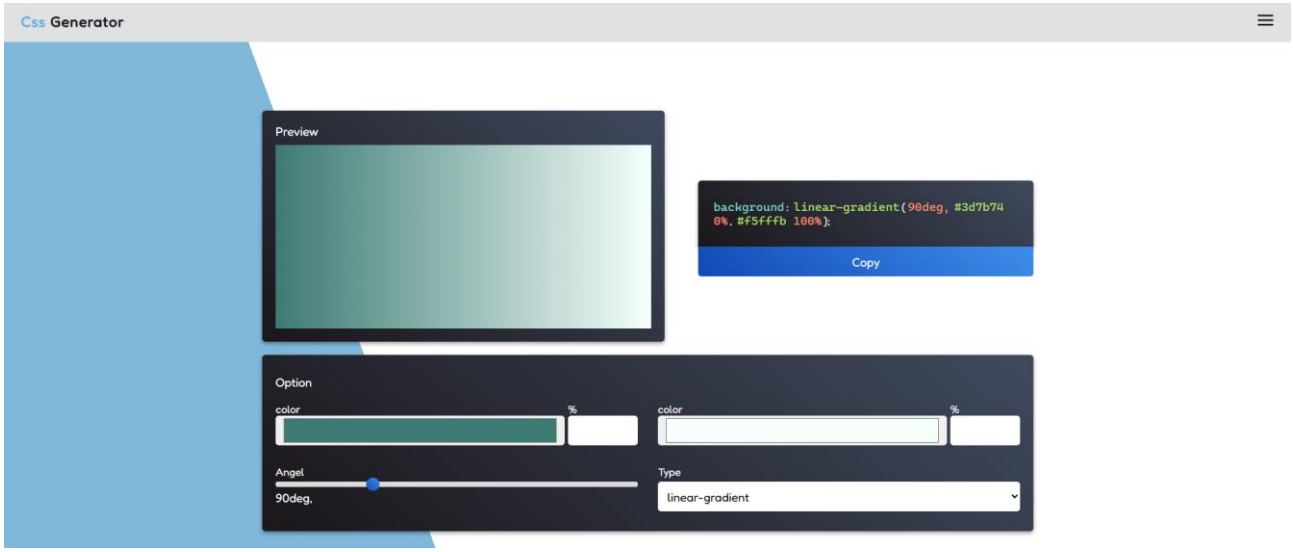

Рисунок 3.4 – Сторінка генератора CSS-стилів Background Gradient

Серед доступних у вебзастосунку генераторів стилів є інструмент із групи Filter, зокрема – генератор Blur (рис. 3.5). Його функція полягає у застосуванні ефекту розмиття на основі алгоритму Гауса до вибраного елемента, найчастіше зображення чи блоку контенту. Користувач має можливість гнучко налаштовувати інтенсивність розмиття за допомогою відповідного повзунка або введення значення вручну в межах від 0px до 100px, залежно від бажаного візуального ефекту. Значення 0px відповідає повній відсутності розмиття, тоді як вищі значення створюють поступово сильніший ефект. Такий параметр є корисним для досягнення низки дизайнерських цілей: створення м’яких візуальних переходів між елементами, виділення активних чи важливих компонентів, затемнення фону за активним вікном або формою.

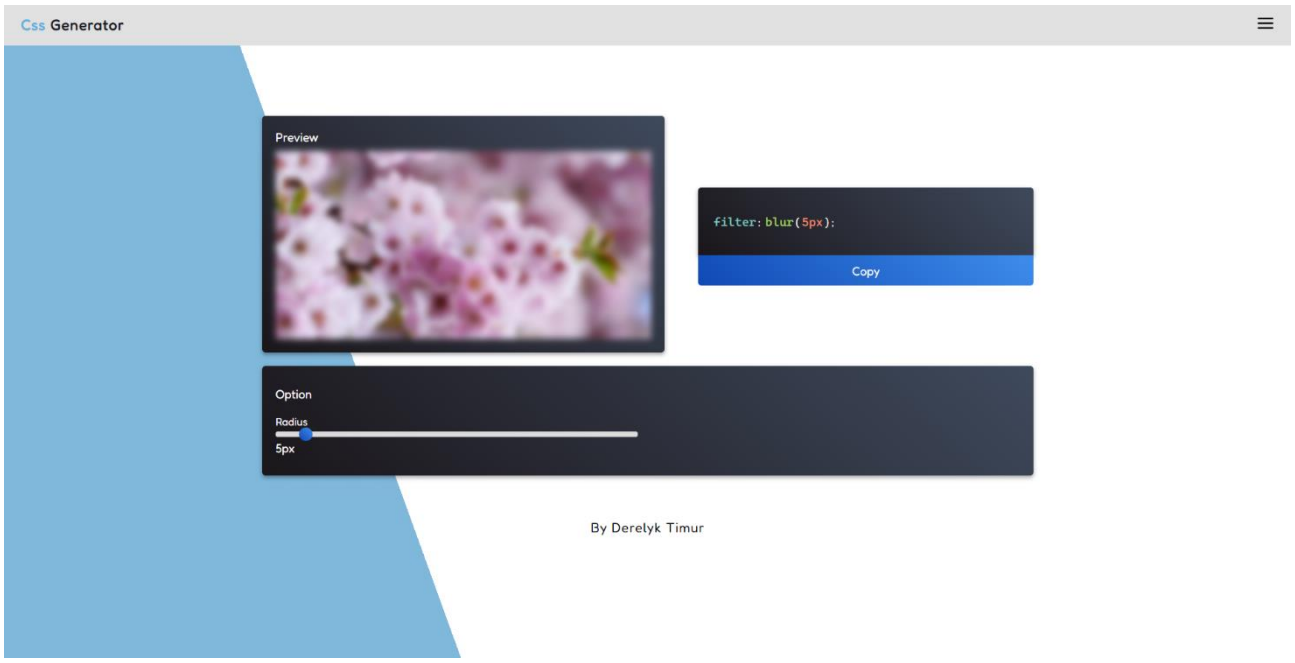

Рисунок 3.5 – Сторінка генератора CSS-стилів Blur

Один із генераторів, реалізованих у межах вебзастосунку – це Flexbox Generator (рис. 3.6), який належить до групи компонувальних властивостей CSS. Його призначення полягає у створенні гнучкого контейнера з можливістю точно налаштувати вирівнювання та напрямок розміщення дочірніх елементів за допомогою властивостей display: flex, flex-direction, justify-content і align-items.

Інтерфейс генератора дозволяє змінювати параметри в реальному часі через випадаючі списки.

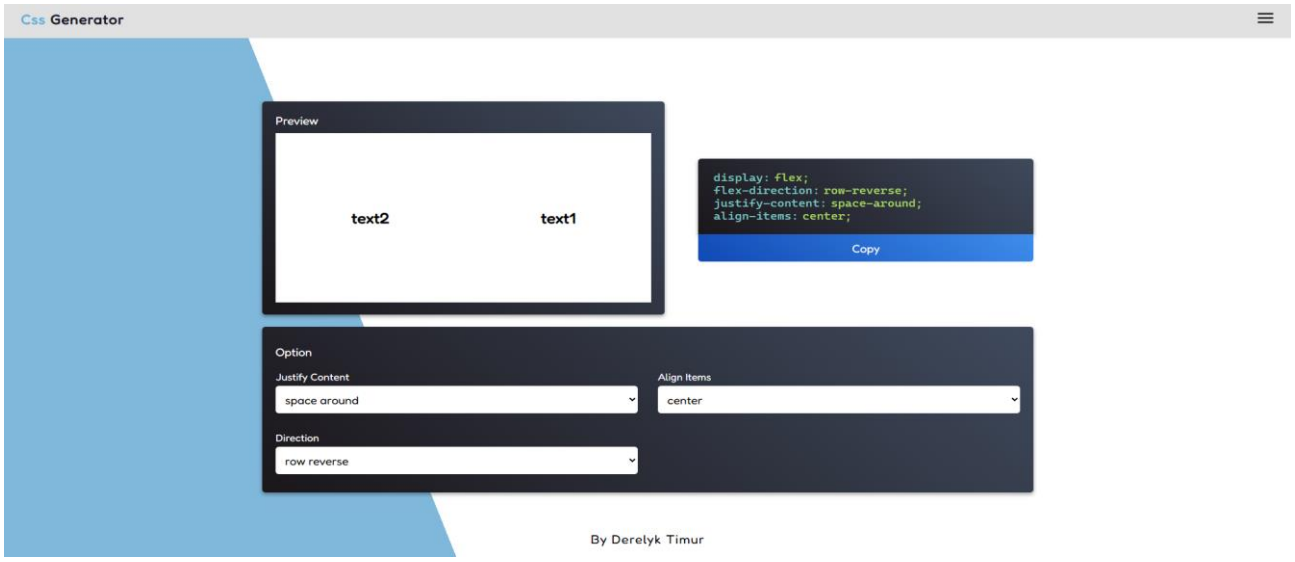

Рисунок 3.6 – Сторінка генератора CSS-стилів FlexBox

3.2 Тестування та налагодження інформаційно-комп’ютерної системи

Одним із ключових етапів життєвого циклу програмного забезпечення є тестування та налагодження, які відіграють визначальну роль у забезпеченні надійності, стабільності та коректності функціонування створеної системи. У межах цієї кваліфікаційної роботи було проведено цілеспрямоване тестування багатофункціонального CSS-генератора стилів, розробленого з використанням бібліотеки React, з метою виявлення та усунення можливих помилок на різних етапах взаємодії з користувачем.

Тестування охоплювало як функціональні, так і нефункціональні аспекти роботи додатку. Зокрема, особливу увагу було приділено перевірці коректності відображення інтерфейсу, динамічного оновлення значень, генерації CSS-коду відповідно до введених параметрів, а також стабільності роботи контекстного середовища, яке забезпечує глобальний доступ до змінних у всьому застосунку.

Проведення функціонального тестування вебзастосунку включала:

– коректність роботи кожного генератора: формування CSS-стилю та його відображення, відповідність згенерованого коду введеним параметрам;

– динамічне оновлення прев’ю елементів на основі змін користувача в режимі реального часу;

– працездатність системи навігації, зокрема бургер-меню та переходів між генераторами;

– коректну інтеграцію кнопки копіювання коду та відповідність вмісту буфера обміну очікуваному результату;

– відображення GitHub-посилання та функціональність зовнішнього переходу.

Тестування проводилося вручну шляхом послідовного проходження усіх можливих сценаріїв взаємодії з інтерфейсом. У результаті було виявлено низку незначних недоліків, пов’язаних із поведінкою стилів на мобільних пристроях та некоректним відображенням певних станів при зміні розмірів вікна браузера, які були оперативно усунені на етапі налагодження.

Проведення нефункціонального тестування вебзастосунку включала:

– тестування адаптивності – застосунок перевірявся на різних розширеннях екрана та пристроях (десктоп, планшет, смартфон), щоб гарантувати коректне відображення інтерфейсу;

– оцінку продуктивності – вимірювалася швидкість завантаження генераторів та часу оновлення CSS-коду при зміні параметрів;

– перевірку сумісності – додаток тестувався у популярних браузерах (Chrome, Firefox, Edge, Safari) для забезпечення однакової поведінки;

– тестування UX – оцінювалася зручність взаємодії з системою з позиції кінцевого користувача.

Для локального тестування було використано вбудовані інструменти React Developer Tools та браузерні інспектори, що дозволило в реальному часі відстежувати зміну станів компонентів, перевіряти DOM-структуру та переглядати стилі.

Для стабільного функціонування вебзастосунку та забезпечення повної інтерактивності інтерфейсу користувача, система повинна відповідати таким мінімальним вимогам:

– windows 10 або новіша, macOS 10.13+, сучасні дистрибутиви Linux;

– сучасна версія Google Chrome, Mozilla Firefox, Microsoft Edge або Safari (підтримка ES6, Flexbox, та Web APIs обов’язкова);

– роздільна здатність екрана, не менше 1024×768 px (для комфортної роботи рекомендовано Full HD);

– стабільне інтернет-з’єднання (необхідне для завантаження ресурсів та переходу за зовнішніми посиланнями, зокрема GitHub);

– оперативна пам’ять від 4 ГБ і більше;

– наявність сучасного браузера з підтримкою JavaScript та локального збереження (Local Storage);

– чотирьох’ядерний процесор (мінімум), рекомендовано з частотою від 1,5 ГГц.

3.3 Інструкція користувача з розгортання вебзастосунку

Після завершення етапу розробки багатофункціонального генератора стилів, постає питання його публікації в мережі. Оскільки вебзастосунок було реалізовано за допомогою фреймворку React, використання платформи Netlify стало найкращим рішенням для розміщення додатку.

Netlify – це сучасна хмарна платформа, яка значно спрощує процес розгортання вебпроєктів. Вона надає автоматизовані інструменти для створення, тестування та розповсюдження додатків без необхідності додаткових конфігурацій серверів. Netlify використовує модульний підхід, що позбавляє розробників необхідності налаштовувати власну внутрішню інфраструктуру і дозволяє їм зосередитися лише на зовнішній стороні своїх проєктів.

Netlify підтримує безперервну інтеграцію (CI) – кожного разу, коли розробник оновлює свій репозиторій GitHub, проєкт автоматично перезбирається і розгортається на сервері. Це дозволяє підтримувати вебзастосунки в актуальному стані без необхідності повторювати ручне розгортання, що особливо зручно, коли розробник активно розробляє або впроваджує зміни. Крім того, платформа забезпечує базову безпеку, адаптивну маршрутизацію, підключення за допомогою SSL-сертифікатів та інші сервіси, важливі для виробничих версій вебзастосунків.

Процес завантаження проєкту в рамках кваліфікаційної роботи передбачає створення нового репозиторію GitHub (рис. 3.7), та підключення його до облікового запису Netlify. Після налаштування параметрів збірки (вказівки гілок, каталогів збірки та команд компіляції) вебзастосунок автоматично збирається та публікується на платформі з використанням наданого домену. Такий підхід забезпечує прозорість розгортання та гнучкість контролю версій, а також дозволяє швидко вносити зміни в будь-який час.

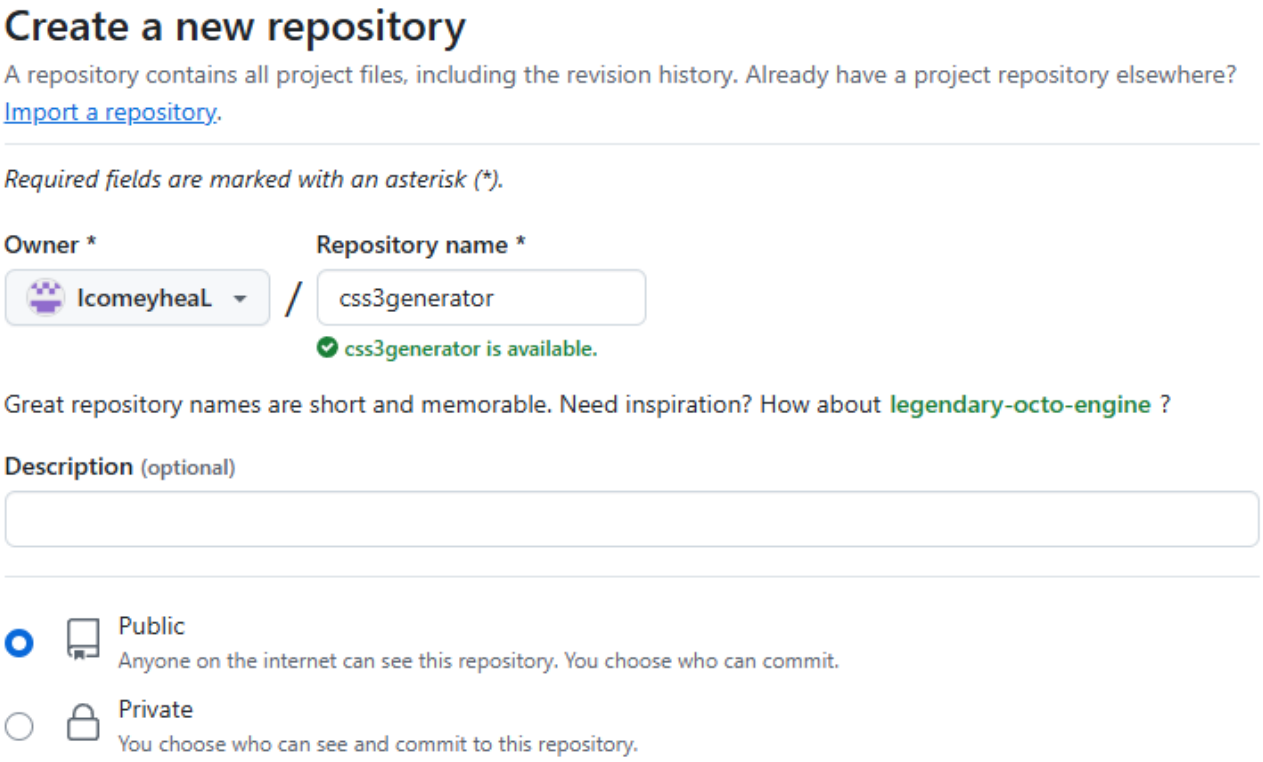

Рисунок 3.7 – Створення нового git-репозиторію

Для збереження історії змін та спільної роботи над проєктом вихідні файли вебзастосунку було завантажено до Git-репозиторію (рис. 3.8). Це забезпечило контроль версій, централізований доступ до коду та можливість відновлення попередніх станів. Окрім цього, використання Git-репозиторію стало необхідним етапом для подальшого розгортання проєкту на хостинговій платформі Netlify, яка підтримує автоматичне розгортання з Git.

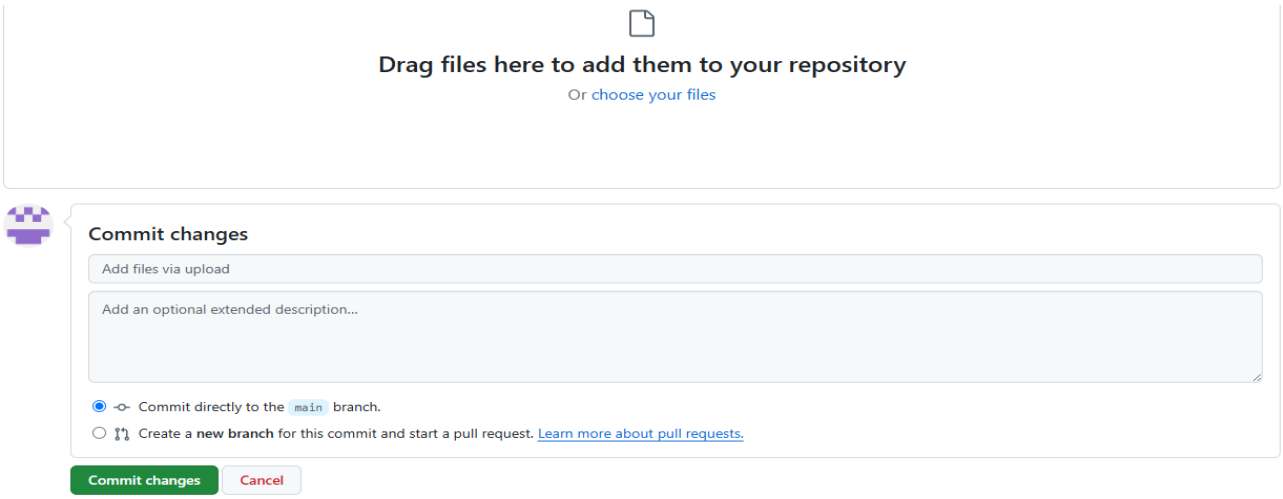

Рисунок 3.8 – Завантаження файлів до репозиторію

Після розробки вебзастосунку файли проєкту готові до розміщення на хостинговій платформі Netlify, що дозволяє легко і швидко розгортати статичні вебресурси та односторінкові додатки (SPA).

Початковим етапом розгортання є створення нового проєкту в середовищі Netlify, а потім імпорт вихідного коду з git-репозиторію, в якому зберігається поточна версія вебзастосунку (рис. 3.9).

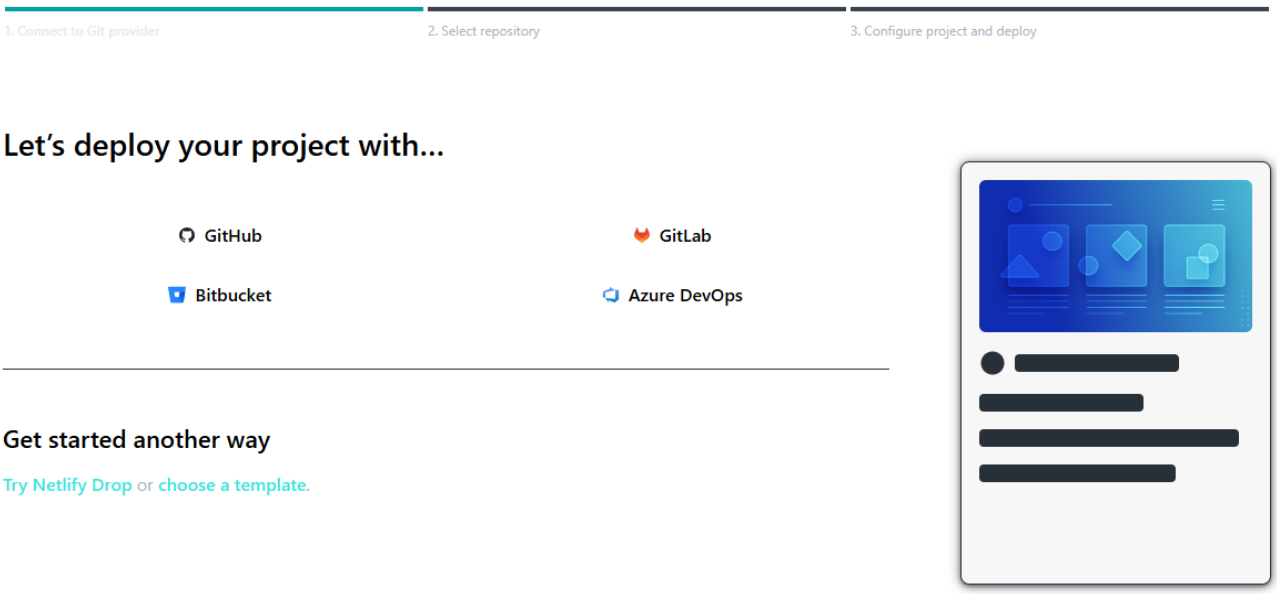

Рисунок 3.9 – Розгортання проєкту на хостингу Netlify

При імпорті репозиторію платформа Netlify автоматично аналізує структуру проєкту, встановлює необхідні залежності та налаштовує середовище, в якому буде збиратися додаток. При цьому користувачі можуть задати індивідуальні параметри розгортання, включаючи вибір гілки репозиторію, вказівку команди, яка буде збирати проєкт, і визначення каталогу, де буде опублікований готовий додаток.

Після завершення процесу збірки та обробки файлів Netlify автоматично публікує вебзастосунок на власних серверах і надає унікальне посилання, яке робить проєкт доступним для перегляда в Інтернеті. В результаті розробники можуть отримати повний доступ до своїх додатків без необхідності складного налаштування сервера (рис. 3.10).

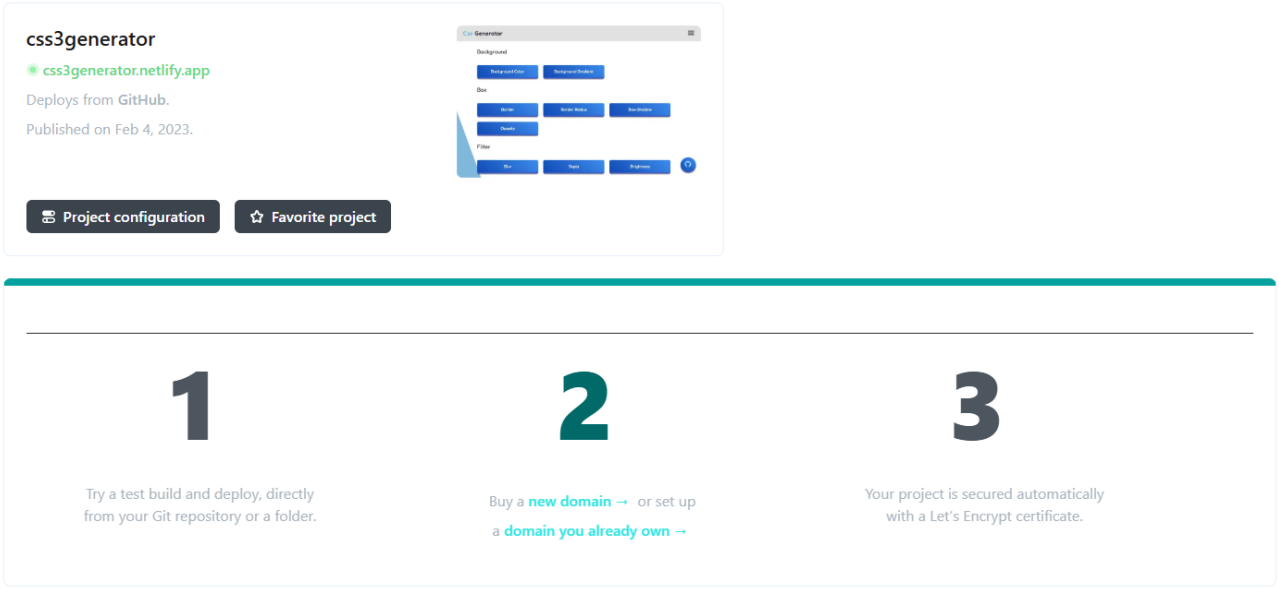

Рисунок 3.10 – Розгорнутий проєкт на хостинг Netlify

Користувачі можуть отримати доступ до повного функціоналу вебзастосунку за допомогою посилань, які вони отримують. Завдяки вбудованим механізмам автооптимізації, що надаються платформою Netlify, інтерфейс залишається стабільним і коректно відображається на різних типах пристроїв. Як результат, процес розгортання є максимально простим, швидким і безпечним, що повністю відповідає сучасним практикам DevOps для малих і середніх вебпроєктів.

3.4 SEO інформаційно-комп'ютерної системи

У контексті сучасного веброзробництва, пошукова оптимізація (Search Engine Optimization) відіграє ключову роль у забезпеченні видимості та доступності проєкту в мережі Інтернет. Навіть найзручніший та найфункціональніший вебзастосунок не зможе ефективно виконувати свою роль, якщо користувачі не знатимуть про його існування. Саме тому на етапі реалізації інформаційно-комп'ютерної системи були враховані базові принципи як технічної, так і контентної SEO-оптимізації.

Оскільки розробка реалізована з використанням React, особливу увагу було приділено проблемі індексації односторінкових застосунків (SPA), які зазвичай не дружні до пошукових систем через динамічне завантаження контенту. Для вирішення цього виклику застосовано такі підходи:

– мінімізація та стиснення файлів – використовувались сучасні методи збірки, що автоматично мінімізують JavaScript, CSS та HTML, зменшуючи розмір сторінки та прискорюючи її завантаження;

– коректна структура DOM – забезпечено семантичну побудову розмітки, з використанням тегів <main>, <section>, <header>, <footer>, що підвищує зрозумілість вмісту для пошукових систем;

– мета-теги та Open Graph – для кожної сторінки налаштовано базові мета-теги (title, description, viewport) та OG-теги для покращення відображення в соціальних мережах;

– friendly-URL – завдяки налаштуванню маршрутизації на рівні сервера (через Netlify), користувачі отримують доступ до зрозумілих та зручних URL-адрес без динамічних параметрів;

– адаптивність – повна підтримка мобільних пристроїв, що враховується Google при ранжуванні (Mobile-First Indexing).

Контентна складова SEO орієнтована на створення зрозумілого, тематично-релевантного і структурованого вмісту. У межах реалізації проєкту було враховано:

– наявність ключових слів – усі текстові блоки, включно з описом генератора, кнопками, підказками та заголовками, містять релевантні запити, що відповідають типовому пошуку користувачів (наприклад, « CSS generator », « online style builder », « visual CSS editor »);

– заголовки різних рівнів (h1-h3) – структура контенту дотримується ієрархії, що полегшує сприйняття інформації не лише для відвідувачів, а й для пошукових алгоритмів;

– альтернативний текст для зображень (alt) – усі зображення в інтерфейсі (іконки, приклади) мають текстові описи, що покращує доступність та індексацію;

– внутрішні переходи – навігація реалізована так, щоб забезпечити логічний зв’язок між розділами застосунку, що позитивно впливає на поведінкові фактори.

Особливу увагу приділено оптимізації швидкодії сайту як одному з ключових факторів SEO. Для оцінки продуктивності використовувався Google Lighthouse, що дозволило провести комплексне тестування та досягти високих результатів, які підтверджують технічну якість реалізації (рис. 3.11).

Рисунок 3.11 – Результат перевірки Lighthouse

Завдяки впровадженим технічним і контентним заходам SEO-оптимізації, вебзастосунок зберігає високу функціональність і зручність для кінцевого користувача, водночас залишаючись цілком доступним для пошукових систем. Це дозволяє ефективно індексувати вміст застосунку, що прямо впливає на його видимість у результатах пошуку. У поєднанні з адаптивним дизайном, швидким завантаженням сторінок і якісно структурованим контентом, це робить вебресурс більш привабливим як для нових користувачів, так і для пошукових алгоритмів. У перспективі така оптимізація дає змогу органічно розширювати аудиторію без потреби у витратних рекламних кампаніях.

3.5 Захист інформаційно-комп’ютерної системи

У сучасному середовищі кіберзагроз захист інформації та комп’ютерних систем є не лише рекомендацією, а й необхідним компонентом будь-якого вебзастосунку. Забезпечення конфіденційності, цілісності та доступності даних користувачів є основою довіри до цифрових продуктів. При розробці вебзастосунків ми приділяємо особливу увагу безпеці на рівні коду та під час взаємодії з кінцевими користувачами.

У рамках проєкту реалізовано низку базових, але критично важливих заходів технічного захисту:

– захист від XSS-атак (Cross-Site Scripting) – усі дані, що вводяться користувачем, проходять обробку перед виведенням на сторінку. Зокрема, була впроваджена фільтрація та екранування потенційно небезпечних символів, що унеможливлює вставку шкідливих скриптів у інтерфейс додатку;

– обмеження доступу до локального середовища – застосунок не зберігає чутливу інформацію в браузері без необхідності та не виконує сторонніх запитів без чітко визначеного джерела, що зменшує ризики витоку даних;

– використання HTTPS – публічна версія додатку розгорнута на Netlify з автоматичним SSL-сертифікатом, що гарантує шифрування всього трафіку між користувачем і сервером, захищаючи дані від перехоплення;

– перевірка інтеграції з GitHub – автентифікація та управління вихідним кодом здійснювались через захищені канали, з використанням облікових записів із двофакторною автентифікацією.

Особлива увага приділяється забезпеченню безпечної та прозорої взаємодії між користувачем і системою:

– обробка форм введення – всі форми, що використовуються для налаштування параметрів CSS, оснащені вбудованими механізмами валідації, які запобігають некоректному або зловмисному введенню даних;

– відсутність необхідності реєстрації та зберігання персональних даних – на поточному етапі реалізації додаток не потребує та не обробляє персональні дані користувача, що значно знижує потенційний ризик зловживань;

– надійність структури коду – завдяки компонентному підходу та суворій структурі проєкту на React вдалося зменшити кількість потенційних точок вразливості під час взаємодії з інтерфейсом.

Таким чином, впроваджені заходи безпеки створюють міцний фундамент для подальшого розвитку вебзастосунків. Ці заходи забезпечують безпеку взаємодії користувачів, захищають систему від типових атак і створюють основу для впровадження більш складних механізмів безпеки при розширенні проєкту або розробці нового функціоналу.

Висновки до розділу 3

У рамках третього розділу було реалізовано багатофункціональний вебзастосунок для генерації CSS-стилів із використанням React. Реалізовано ключові компоненти інтерфейсу: панель налаштування параметрів, блок попереднього перегляду, механізм автоматичної генерації та експорту CSS-коду. Завдяки компонентній структурі забезпечено гнучкість, адаптивність і зручність у використанні. Проведене тестування за допомогою Google Lighthouse засвідчило високу швидкодію, стабільність та відповідність сучасним вебстандартам. Додатково створено покрокову інструкцію з розгортання проєкту через GitHub і Netlify.

У процесі реалізації також було впроваджено базові принципи технічної та контентної SEO-оптимізації, що підвищує видимість застосунку в пошукових системах. Для забезпечення захищеної взаємодії з користувачем використано фільтрацію введених даних, шифрування трафіку та обмеження потенційно небезпечних дій у середовищі браузера. Усі ці заходи дозволили створити не лише зручний у користуванні, але й безпечний і сучасний вебінструмент для frontend розробників.

ВИСНОВКИ

У результаті поетапного формування та реалізації завдань, пов’язаних із створенням багатофункціонального CSS-генератора стилів на базі бібліотеки React, було закладено методологічну та технологічну основу кваліфікаційної роботи. Розробка ґрунтувалася на попередньому аналізі актуальності та доцільності створення подібного вебзастосунку в умовах сучасної веброзробки. На початковому етапі було проведено аналіз та порівняння з існуючими аналогами, зокрема такими як Webcodetools, CSS3 Generator та CSSmatic. Це дозволило виявити основні обмеження наявних рішень – недостатню функціональність, відсутність гнучкості, складність масштабування й брак інтерактивного попереднього перегляду. Кожен етап – від дослідження предметної області до практичної реалізації – був спрямований на створення інтуїтивного, ефективного та зручного вебзастосунку для генерування CSS-стилів.

На основі здійсненого аналізу визначено ключові функціональні й нефункціональні вимоги до вебзастосунку: підтримка CSS-параметрів, прев’ю в реальному часі, адаптивність, продуктивність і зручність використання. React було обрано завдяки його компонентному підходу й динамічному оновленню інтерфейсу без перезавантаження. Для моделювання створено UML-діаграми, що відображають структуру системи та взаємодію користувача, що забезпечило логічну архітектуру застосунку.

Для розробки багатофункціонального CSS-генератора було використано React для побудови SPA-архітектури, HTML і SASS – для структурування та стилізації інтерфейсу, JavaScript – для реалізації динамічної логіки. Node.js разом із NPM забезпечили керування залежностями, а JSON використовувався для зберігання конфігураційних даних. Обраний стек технологій гарантує швидкодію, масштабованість та зручність у використанні.

У процесі реалізації вебзастосунку створено набір функціональних компонентів: панель налаштувань параметрів стилів, блок попереднього

перегляду, система генерації CSS-коду, копіювання в буфер обміну, адаптивна навігація тощо. Інтерфейс побудовано відповідно до принципів UX/UI-дизайну, що забезпечує інтуїтивність і доступність для широкого кола користувачів, включаючи новачків у веброзробці. Особливу увагу приділено адаптивності, що дозволяє користуватись додатком як із десктопів, так і з мобільних пристроїв.

Розроблений вебзастосунок було протестовано на відповідність визначеним функціональним і нефункціональним вимогам. Перевірено коректність роботи генераторів стилів, адаптивність інтерфейсу, швидкодію та стабільність при різних сценаріях взаємодії. Виявлені недоліки були усунуті в процесі налагодження, що дозволило досягти стабільної та зручної у використанні версії вебзастосунку.

Створено користувацьку інструкцію, яка містить покроковий опис розгортання додатку на платформі Netlify за допомогою репозиторію GitHub, що забезпечує простоту використання навіть для недосвідчених користувачів.

Особливу увагу було приділено оптимізації: додаток успішно пройшов аудит Google Lighthouse, що підтвердив його продуктивність, доступність і відповідність SEO-вимогам. Під час розробки також було побудовано семантичну структуру HTML та враховано вимоги пошукових систем для покращення індексації та оптимізації.

Також реалізовано базові засоби безпеки, серед яких: фільтрація введених даних, уникнення прямого HTML-рендерингу, використання HTTPS, обмеження дій потенційно небезпечних скриптів за допомогою CSP-заголовків. Це забезпечує надійний рівень захисту при взаємодії з користувачем.

Підсумовуючи, розробка CSS-генератора стилів була успішно реалізована й дала змогу створити зручний та ефективний інструмент для швидкої генерації та подальшого використання CSS-стилів. Його впровадження здатне суттєво покращити робочий процес веброзробників, сприяти підвищенню якості оформлення інтерфейсів та забезпечити комфортний і інтуїтивний досвід використання для кінцевих користувачів вебзастосунком.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Як створити вебзастосунок: типи, переваги, принцип роботи – Wezom. URL: https://wezom.com.ua/ua/blog/kak-sozdat-veb-prilozhenie (дата звернення: 10.02.2025).

2. CSS Code Generator for Effortless Web Development. Web Code Tools. The best tools for web development. URL: https://webcode.tools/css-generator (date of access: 12.02.2025).

3. CSS3Generator by @RandyJensen. URL: https://css3generator.com/ (date of access: 12.02.2025).

4. CSSmatic by @e98cuenc and freepik. URL: https://www.cssmatic.com/ (date of access: 12.05.2025).

5. Учасники проєктів Вікімедіа. HTML – Вікіпедія. URL: https://uk.wikipedia.org/wiki/HTML (дата звернення: 16.05.2025).

6. Durocher D. HTML / CSS QuickStart Guide: The Simplified Beginners Guide to Developing a Strong Coding Foundation, Building Responsive Websites, and Mastering the Fundamentals of Modern Web Design. ClydeBank Media LLC, 2021. 361 p.

7. W3Schools Online Web Tutorials About CSS. URL: https://www.w3schools.com/css/ (date of access: 21.02.2025).

8. Учасники проєктів Вікімедіа. Sass – Вікіпедія. URL: https://uk.wikipedia.org/wiki/Sass (дата звернення: 04.03.2025).

9. JavaScript Guide – JavaScript MDN. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (date of access: 07.03.2025).

10. Flanagan D. JavaScript: The Definitive Guide, 7th Edition. O'Reilly Media, 2020. 706 p.

11. React – Wikiwand. URL: https://www.wikiwand.com/uk/articles/React (date of access: 11.03.2025).

12. Quick Start – React. URL: https://react.dev/learn (date of access: 15.03.2025).

13. Stefanov S. React: up and Running: Building Web Applications. O'Reilly Media, Incorporated. 2021. 222 p.

14. Вступ до JSX – React. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/docs/introducing-jsx.html (дата звернення: 21.03.2025).

15. Node. js Design Patterns: Design and Implement Production-Grade Node. js Applications Using Proven Patterns and Techniques. De Gruyter GmbH Walter. 2020. 664 p.

16. NPM – Docs. URL: https://docs.npmjs.com/ (date of access: 02.04.2025).

17. Учасники проєктів Вікімедіа. JSON – Вікіпедія. URL: https://uk.wikipedia.org/wiki/JSON (дата звернення: 06.04.2025).

ДОДАТКИ

Додаток А – Сертифікат про участь у міжнародній науково-практичній конференції з проблем вищої освіти і науки

Міністерство освіти і науки України
Волинська обласна рада
Луцька міська рада
Луцький національний технічний університет
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»
Національний університет «Львівська політехніка»
Військовий інститут телекомунікацій та інформатизації імені Героїв Крут (м. Київ)
Тернопільський національний педагогічний університет імені В. Гнатюка
Рівненський державний гуманітарний університет
Навчально-науковий інститут «Українська інженерно-педагогічна академія» Харківського національного університету ім. В.Н. Каразіна
Люблінська політехніка (Польща)
Фрайберзька гірнича академія (Німеччина)
Гронінгенський університет (Нідерланди)
Кавказький університет (Грузія)
Політехнічний університет Браганси (Португалія)

ITOHB

ТЕЗИ ДОПОВІДЕЙ
X Міжнародної науково-практичної конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ITOHB-2025)

23-24 травня 2025 р.

Луцьк 2025

застосунку типу блог, де React відповідає за динамічне відображення контенту, а Node.js із MongoDB – за зберігання даних та реалізацію API. Реалізовано повний цикл обробки статей: створення, редагування, видалення та отримання через API. Такий підхід підтвердив ефективність React як основного засобу створення сучасного frontend для вебплатформ у поєднанні з гнучкими backend-рішеннями на базі JavaScript.

Отже, React у поєднанні з Node.js та MongoDB становить ядро сучасної вебплатформи, що дозволяє реалізувати високодинамічні застосунки зі складною логікою, повністю керованим станом та широкими можливостями для масштабування і подальшої розробки.

Список використаних джерел

1. React. *React*. URL: https://react.dev/.
2. API - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Glossary/API.
3. DOM - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Glossary/DOM.

УДК 004.42

БАГАТОФУНКЦІОНАЛЬНИЙ ГЕНЕРАТОР CSS-СТИЛІВ ЯК ІНСТРУМЕНТ ПРИСКОРЕННЯ РОЗРОБКИ ВЕБІНТЕРФЕЙСІВ

Дерелюк Тимур Юрійович
Луцький національний технічний університет, здобувач вищої освіти спеціальності ІПЗ, derelyk2004@gmail.com
Ящук Андрій Анатолійович
Луцький національний технічний університет, к.т.н., доцент кафедри інженерії програмного забезпечення,
a.yashchuk@lutsk-ntu.com.ua

MULTIFUNCTIONAL CSS-STYLE GENERATOR AS A TOOL TO SPEED UP WEB INTERFACE DEVELOPMENT

Dereliuk Tymur Yuriyovych
Lutsk National Technical University, higher education student in the educational program of Software Engineering, derelyk2004@gmail.com
Yashchuk Andrii Anatoliyovych
Lutsk National Technical University, PhD, associate professor of Software Engineering Department, a.yashchuk@lutsk-ntu.com.ua

This abstract introduces a feature-rich CSS style generator designed to simplify and speed up the development of interfaces. Implemented with React, the tool offers real-time style customization, previewing, and automatic code generation. The project solves common problems of existing generators and provides an intuitive, scalable solution for developers.

Keywords: *CSS, style generator, React, front-end tools, code automation, web development, user interface customization*

Зі зростанням складності вебтехнологій зростає й потреба в інструментах, що спрощують створення та стилізацію інтерфейсів. CSS відіграє ключову роль у зовнішньому вигляді вебдодатків, але ручне написання стилів часто є трудомістким і помилковим, особливо для початківців. Тому актуальною проблемою є створення інструментів, які могли б автоматизувати цей процес і забезпечити більшу зручність для розробників.

Аналіз існуючих онлайн-генераторів CSS-стилів показав, що наявні рішення мають низку обмежень. Зокрема, WebCodeTools [1] пропонує набір утиліт для веброзробників, включаючи CSS-генератор, однак не має підтримки попереднього перегляду та адаптивного інтерфейсу. CSS3 Generator [2], хоча й забезпечує базову функціональність, має застарілий дизайн та обмежений набір параметрів. CSSmatic [3], створений ще у 2013 році, відзначається простим інтерфейсом, але поступається в функціональності сучасним аналогам.

Всі розглянуті сервіси не дозволяють масштабування, не підтримують роботу з кількома елементами одночасно і не

мають інтерактивної візуалізації змін. Це обґрунтовує потребу у створенні більш гнучкого та сучасного інструменту для генерації CSS-стилів.

Метою розробки було створення багатофункціонального генератора CSS-стилів, який дозволяв би користувачам в режимі реального часу створювати та налаштовувати стилі для елементів інтерфейсу і отримувати згенерований код без ручного втручання.

Генератор був реалізований у вигляді односторінкового вебдодатку з використанням бібліотеки React [4]. Інтерфейс розділений на дві основні області: головна сторінка з вибором генераторів і вікно попереднього перегляду згенерованих стилів (рис. 1).

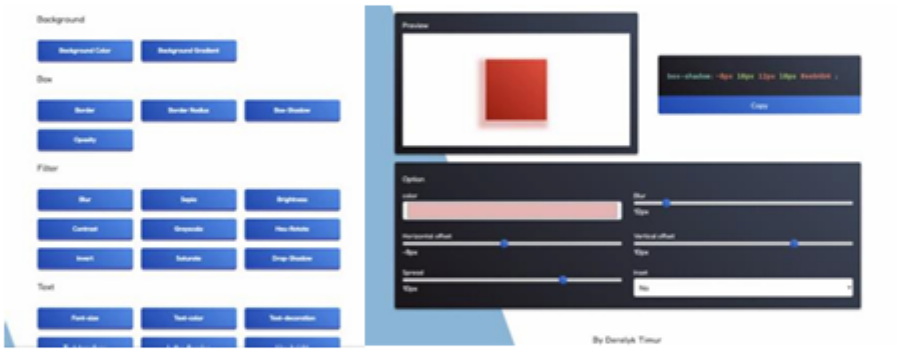

Рисунок 1 – Інтерфейс розробленого генератора стилів

Зміни параметрів стилю (колір, розмір, тінь тощо) миттєво відображаються у вигляді оновленого елемента, а CSS-код генерується автоматично і доступний для копіювання. Розроблений вебдодаток вирізняється гнучкою структурою та зручним інтерфейсом. На відміну від інших генераторів, він враховує недоліки існуючих рішень — відсутність адаптивного перегляду, обмежений функціонал і складність масштабування. Завдяки React-хукам реалізовано динамічне оновлення та підтримку кількох блоків одночасно.

Окрему увагу приділено адаптивності – інтерфейс коректно працює як на десктопах, так і на мобільних пристроях. Завдяки використанню flexbox і медіа-запитів усі елементи налаштовуються відповідно до ширини екрана. Інтерактивність забезпечується хуками useState та useEffect, що дозволяє оновлювати CSS-код у реальному часі без перезавантаження сторінки.[5].

Однією з ключових функцій вебдодатку є можливість одночасної роботи з кількома генераторами стилів. Користувач може задавати окремі параметри для кнопок, заголовків, контейнерів чи текстових блоків і зручно перемикатися між ними. Це дозволяє створювати складні дизайни без виходу з середовища генератора.

Перевагою також є зрозумілий синтаксис згенерованого CSS-коду, який легко інтегрується у проєкти. У подальшому планується додати підтримку SCSS та функцію збереження стилів у хмарі.

Тестування показало стабільну роботу генератора у сучасних браузерах (Chrome, Firefox, Edge), коректність CSS-коду, адаптивність та продуктивність навіть при зміні великої кількості параметрів. Для налагодження використовувались Chrome DevTools і React Developer Tools.

У перспективі планується інтеграція генератора з платформами дизайну, такими як Figma та Adobe XD, а також створення браузерних розширень для зручного використання під час верстки. Це дозволить глибше впровадити інструмент у робочі процеси розробників.

Генератор має значний потенціал у сфері освіти — його можна використовувати для навчання основам CSS без потреби писати код вручну. Візуалізація змін у реальному часі допомагає краще засвоїти принципи специфікації селекторів, каскадування та успадкування.

Також інструмент буде корисним для дизайнерів, які хочуть налаштувати вигляд елементів без знання коду. Це сприяє кращій взаємодії між дизайном і розробкою та підвищує ефективність командної роботи над цифровими продуктами.

Особливу увагу приділено продуктивності. Завдяки оптимізації рендерів, мемоізації компонентів і обмеженню кількості активних станів, генератор стабільно працює навіть на слабких пристроях і при великій кількості одночасних змін стилів.

Підсумовуючи, розроблений багатофункціональний CSS-генератор поєднує сучасні вебтехнології з акцентом на зручність, продуктивність і гнучкість. Він вирішує проблеми ручного стилювання та закладає основу для подальшого розвитку автоматизованих інструментів, доступних як для досвідчених розробників, так і для новачків.

Список використаних джерел

1. WebCodeTools – The best code generators for developers. URL: https://webcode.tools/ (дата звернення: 07.05.2025).

2. CSS3 Generator by Randy Jensen. URL: https://css3generator.com (дата звернення: 07.05.2025).

3. The ultimate CSS tools for web designers. URL: https://www.cssmatic.com (дата звернення: 07.05.2025).

4. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/ (дата звернення: 09.05.2025).

5. Огляд хуків – React. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/docs/hooks-overview.html#gatsby-focus-wrapper (дата звернення: 09.05.2025).

СЕРТИФІКАТ

Кафедра цифрових освітніх технологій

Кафедра інженерії програмного забезпечення

Даний сертифікат засвідчує, що

Дерелюк Тимур Юрійович

брав(-ла) участь у

X Міжнародній науково-практичній конференції
з проблем вищої освіти і науки
"Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2025)"
загальним обсягом 15 годин (0,5 кредитів ECTS),

яка проводилася 23-24 травня 2025 року у м.Луцьк
на базі Луцького національного технічного університету

Ректор ЛНТУ _____ **Ірина ВАХОВИЧ**

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

№ ІТОНВ-2025-49

Програма конференції: https://itonv.lntu.edu.ua/files/2025/program-itonv-2025.pdf