ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних технологій

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Розробка структури та програмна реалізація інтерфейсу WEB-редактора»

Студента 4 курсу, 10 групи, Ковтун Богдан Володимирович

спеціальності 122 «Комп'ютерні науки»

_____
підпис студента

Науковий керівник
кандидат педагогічних наук, доцент _____ Базурін Віталій Миколайович
підпис керівника

Гарант освітньої програми
кандидат технічних наук, професор _____ Демідов Павло Георгійович
підпис керівника

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«12» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу (проект) студентці

Ковтуна Богдана Володимировича
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка структури та програмна реалізація інтерфейсу WEB-редактора»
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *30 травня2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка структури та програмної реалізації інтерфейсу WEB-редактора.
Об'єкт дослідження: процес створення WEB-редактора.
Предмет дослідження: методи програмної реалізації інтерфейсу та розробки структури WEB-редактора.
4. Перелік графічного матеріалу_____
_____
_____
_____
5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Базурін В.М.	15.12.2022 р.	15.12.2022 р.
2	Базурін В.М.	15.12.2022 р.	15.12.2022 р.
3	Базурін В.М.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (проекту) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Аналіз існуючих рішень для розроблення дизайну сайтів

1.1. Аналіз стану проблеми

1.2. Інструментальні засоби для створення і дизайну сайтів

1.3. Аналіз характеристик існуючих веб-редакторів

РОЗДІЛ 2. Розробка моделі програми

2.1. Загальна концепція програми

2.2. Функціональна модель програми

РОЗДІЛ 3. Веб-редактор як один з видів інструментального програмного забезпечення

3.1. Інформаційно-логічна модель веб-редактора

3.2. Особливості програмної реалізації

3.4. Технологія використання і тестування веб-редактора

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*04.10.2022*
2	*Розробка та затвердження завдання на*	*15.12.2022*	*15.12.2022*

	випускну кваліфікаційну роботу		
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*РОЗДІЛ 1. Аналіз існуючих рішень для розроблення дизайну сайтів*	*28.02.2023*	*28.02.2023*
5	*РОЗДІЛ 2. Розробка моделі програми*	*06.04.2023*	*06.04.2023*
6	*РОЗДІЛ 3. Веб-редактор як один з видів інструментального програмного забезпечення*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 -01.06.2023*	
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2022 р.

Керівник випускної кваліфікаційної роботи (проекту)
Базурін В.М.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Ковтун Б.В.
(прізвище, ініціали, підпис)

9. Відгук керівника випускної кваліфікаційної роботи (проекту)

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник випускної кваліфікаційної роботи (проекту)
_____ 30.05.2023 р.
(підпис, дата)

10. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента _____
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.
(підпис, прізвище, ініціали)

«_____»_____2023 р.

Анотація

У випускній кваліфікаційній роботі здійснений комплексний аналіз, розробка структури та інтерфейсу WEB-редактора з метою спрощення навчання веб-розробки, підвищення ефективності роботи з веб-сайтами та їх редагуванням. Проведено робота над теоретичними аспектами та засадами розробки WEB-редактора. Розроблена структура WEB-редактора яка направлена на зручність використання. Створений інтерфейс, який може бути адаптовним під будь-якого користувача, простий у розумінні та має лише корисний функціонал. Обрані слушні інструменти та мови програмування для швидкого та ефективного вдосконалення WEB-редактора.

Ключові слова: WEB-редактор, веб-сайт, користувач, інтерфейс, мова програмування.

Anotation

In the final qualification work, a comprehensive analysis, development of the structure and interface of the WEB-editor was carried out with the aim of simplifying the learning of web development, increasing the efficiency of working with websites and their editing. Work was carried out on theoretical aspects and principles of WEB-editor development. The structure of the WEB editor has been developed, which is aimed at ease of use. The created interface, which can be adapted to any user, is easy to understand and has only useful functionality. Appropriate tools and programming languages are selected for quick and effective improvement of the WEB editor.

Keywords: Web-editor, website, user, interface, programming language.

ЗМІСТ

ВСТУП

Стрімкий розвиток сучасних інформаційних технологій та Інтернету вимагає постійного вдосконалення та розширення функцій веб-додатків. Розробка веб-сторінок вимагає вміння працювати з веб-редакторами, які надають можливість створювати, редагувати та оформляти контент.

У цьому контексті актуальним і важливим завданням є розробка структури та програмної реалізації інтерфейсу WEB-редактора. WEB-редактори є основними інструментами для веб-розробників, дизайнерів і контент-менеджерів, які працюють з веб-сторінками. Вони забезпечують зручний і ефективний спосіб редагування веб-вмісту, створення професійних інтерфейсів користувача, допомагають навчати нових співробітників у веб-розробці. Відповідно, це сприяє ефективнішому виконанню завдань веб-дизайнерів, розробників програмного забезпечення та інших фахівців, які займаються створенням веб-проектів. Зручний та функціональний веб-редактор сприяє покращенню якості веб-сторінок, полегшує редагування та оновлення контенту, що в свою чергу сприяє покращенню взаємодії з користувачами та досягненню поставлених цілей. Дана дипломна робота присвячена структурі інтерфейсу WEB-редактора та розробці програмної реалізації для підвищення продуктивності та зручності Web-розробки. Він вирішує актуальну проблему підвищення ефективності та зручності роботи з веб-контентом, відкриває нові можливості для професіоналів у сфері веб-розробки, допомагає досягти високої якості веб-технологій.

Мета і завдання дослідження. Метою даного дослідження є розробка структури та програмна реалізація інтерфейсу WEB-редактора. Для досягнення поставленої мети необхідно було вирішити наступні **завдання:**

- Проаналізувати проблеми та існуючі рішення проблем для розроблення дизайну сайтів;
- дослідити інструментальні засоби та їх характеристики для створення і дизайну сайтів;
- Проаналізувати та вивчити функціональні та нефункціональні вимоги до WEB-редактора;
- дослідити потреби та основні функції користувачів, які повинен виконувати редактор;
- написати програмний код для реалізації функціональності WEB-редактора;
- перевірка працездатності та стабільності роботи WEB-редактора;

Об'єкт дослідження: процес створення WEB-редактора.

Предмет дослідження: методи програмної реалізації інтерфейсу та розробки структури WEB-редактора.

Методи дослідження: Теоретичною основою дослідження є загальнонауковий аналітичний метод, метод спілкування з потенційними користувачами, опитування, вивчення документацій інструментів для веб-розробки. Для практичного вирішення поставлених задач використовувалися такі методи:

- загальнонауковий аналітичний метод (розділ 1);
- методи математичного моделювання для комплексної оцінки існуючих веб-редакторів (питання 2.1, 2.2);
- методи теорії БД для формування інформаційно-логічної моделі предметної області та БД;
- методи алгоритмічного програмування, для реалізації інтерфейсу веб-редактора.

Практичне значення. Отримані результати, можуть бути використані користувачами веб-редакторів, веб-розробниками, людьми, що вивчають веб-дизайн. Програмна реалізація, надає можливості використання редактору для навчання як студентам, так і вчителям, для більш досвідчених розробників – можливість удосконалити функції редактору.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 20 найменувань, додатків і містить 54 сторінок основного тексту, 11 рисунків і 4 таблиці.

РОЗДІЛ 1.
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ РОЗРОБЛЕННЯ ДИЗАЙНУ САЙТІВ

1.1. Аналіз стану проблеми

Незважаючи на те, що Інтернет увійшов у наше життя досить давно і впевнено, багато підприємців і навіть великих компаній не розуміють, що їм принесе створення власного сайту, адже існують і інші перевірені способи самореклами: телебачення, радіо, ЗМІ, банери, листівки тощо.

Кожен сучасний бізнес має веб-сайт. Це один із факторів репутації, адже саме в Інтернеті потенційні клієнти в першу чергу будуть шукати інформацію про компанію. І якщо він не має принаймні однієї платної сторінки, це виглядає підозріло – наскільки цей бізнес провалений, якщо він не може створити навіть невеликий веб-ресурс?

Актуальність створення веб-сайту також полягає в тому, що якщо потреба якомога швидше донести інформацію до якомога більшої кількості людей, немає кращого способу, ніж використання власного веб-сайту. Веб-ресурси дозволяють стисло і водночас вичерпно представити інформацію про компанію та її товари чи послуги. Сайт також може повідомляти про новини компанії, зміни цін або години роботи, а також містити коментарі вдячних клієнтів.

Не кожному бізнесу потрібен великий портал зі складним дизайном і функціоналом. Іноді зустрічається невеликий сайт-візитка, який можна створити самостійно або замовити у експертів за невелику плату.

Актуальність розробки сайту пояснюється такими факторами:

- Швидкість подання інформації широкому колу осіб;

- Поліпшення іміджу компанії та підвищення її популярності;
- Можливість організувати зворотний зв'язок із клієнтами;
- Оперативний зв'язок з філіями та представниками в різних кінцях країни та за кордоном;
- Організація маркетингових досліджень;
- Реклама та залучення покупців та клієнтів;
- Збільшення трафіку.

Необхідно розуміти, що жоден інший ресурс не дасть стільки переваг, скільки власний сайт, чи це візитівка, Інтернет-магазин чи будь-який інший веб-ресурс.

Розробка веб-сайтів стала необхідністю для багатьох компаній, підприємств та організацій, які прагнуть бути присутніми в Інтернеті. Це пов'язано зі значним збільшенням кількості веб-сайтів у всіх сферах діяльності. Це призвело до зростання конкуренції серед веб-дизайнерів, які прагнуть створювати сайти, що виділяються і привертають увагу цільової аудиторії. Клієнти стають все більш вимогливими та обізнаними в питаннях дизайну веб-сайтів. Вони хочуть, щоб їхні веб-сайти були привабливими, функціональними та відповідали останнім тенденціям. Крім того, щоб привернути увагу і отримати конкурентну перевагу, веб-сайт повинен бути унікальним і відрізнятися від своїх конкурентів. Нові тренди та тенденції рухають розвиток веб-дизайну вперед, і розуміння цих тенденцій є важливим для веб-дизайнерів, оскільки вони впливають на те, як користувачі бачать і сприймають веб-сайт. Деякі з найважливіших тенденцій у сучасному веб-дизайні

1) Мінімальний дизайн: Ця тенденція стосується спрощення дизайну, використання мінімальних елементів та уникнення непотрібних деталей.

Мінімальний дизайн робить веб-сайт чистим і цілеспрямованим, полегшуючи користувачам розуміння контенту.

2) Типографіка: Розробка та використання виразної та привабливої типографіки стає все більш популярною. Великі, жирні шрифти і незвичайні поєднання шрифтів надають веб-сайтам характер і стиль.

3) Градієнти і переходи: Градієнти і плавні переходи кольорів додають веб-дизайну глибини і візуальної привабливості. Градієнти можуть створювати реалістичні ефекти і підкреслювати ключові елементи сайту.

4) Мультимедійні елементи: Використання мультимедійних елементів, таких як відео та анімація, стає все більш поширеним. Мультимедійні елементи, такі як відео та анімація, можуть додати динамічності та інтерактивності веб-сайту і забезпечити більш цікавий користувацький досвід.

5) Мобільний дизайн: Зі стрімким розвитком мобільних пристроїв мобільний дизайн стає все більш важливим. Розробка веб-сайтів з адаптивним дизайном стала пріоритетним завданням для веб-дизайнерів [2].

Розробка веб-сайтів в умовах зростаючої конкуренції стала серйозним викликом для веб-дизайнерів. Веб-дизайнерам необхідно йти в ногу з останніми тенденціями та інноваціями у веб-дизайні та вдосконалювати свої навички і знання. Щоб привабити клієнтів і виділитися серед конкурентів, веб-дизайнери повинні бути креативними, естетичними, аналітичними та адаптивними до мінливих вимог. Також з'явилися нові технології та інструменти для веб-дизайну, такі як HTML5, CSS3, бібліотеки та фреймворки JavaScript. Ці технології дозволяють веб-дизайнерам створювати більш інтерактивні та вражаючі ефекти, використовуючи анімацію, ефекти паралаксу та скролінгу. Ці технології пропонують нові можливості для

творчості та експериментів у веб-дизайні, допомагаючи створювати унікальні та цікаві веб-сайти.

Ще однією важливою тенденцією є зростаюче значення веб-доступності. Веб-дизайнери починають усвідомлювати важливість того, щоб веб-сайти були доступними для всіх користувачів, включаючи людей з обмеженими можливостями. Це означає врахування таких аспектів, як доступність для користувачів з вадами зору та опорно-рухового апарату, використання чіткого та розбірливого контенту, а також відповідне маркування. Технологічні та інноваційні зміни у веб-дизайні відкривають нові можливості для створення естетично привабливих, функціональних і зручних для користувачів веб-сайтів. Вони підкреслюють важливість постійного професійного розвитку та використання новітніх технологій у процесі розробки веб-дизайну [17].

Поглиблений аналіз також може виявити недоліки та прогалини в існуючих рішеннях і визначити сфери, які потребують подальшого вдосконалення. Проблема відсутності легкої та доступної персоналізації в дизайні веб-сайтів полягає в тому, що багато веб-сайтів не пропонують можливості кастомізації інтерфейсу і не враховують індивідуальні потреби користувачів. У більшості випадків веб-сайти мають типовий дизайн, структуру та вбудовані функції, які не дозволяють користувачам пристосувати зовнішній вигляд сайту до власних потреб. Втрата цієї конкурентної переваги є однією з основних проблем, з якими стикаються веб-дизайнери та розробники. Якщо веб-сайт не оновлюється та не використовує новітні технології, він може виглядати застарілим і менш привабливим для користувачів. Крім того, застарілі технології можуть обмежити продуктивність веб-сайту з точки зору швидкості, адаптованості до різних пристроїв, безпеки тощо.

Ще одним фактором, який призводить до втрати конкурентної переваги, є недостатня увага до досвіду користувача. Веб-сайти, яким бракує зручної навігації, швидкого доступу до інформації та достатньої візуальної привабливості, ризикують втратити відвідувачів. Користувачі стають все більш вимогливими та очікують від своїх веб-сайтів якісного користувацького досвіду.

Проаналізувавши потреби користувачів, веб-редактор мусить задовольняти такі функції:

- Підсвічування синтаксису для різних мов програмування. Це дозволяє користувачам переглядати різні елементи коду (ключові слова, змінні, функції тощо) у відповідних кольорах, що полегшує читання та редагування коду.
- Функція автозаповнення допомагає користувачам швидко вводити код, надаючи підказки та заповнюючи фрагмент коду або назву функції. Це зменшує кількість помилок і підвищує продуктивність розробника.
- Вбудовані інструменти для перевірки синтаксичних помилок та інших проблем у коді. Наприклад, він може виділяти неправильні рядки коду або невідомі змінні, допомагаючи розробникам виявляти та виправляти помилки на ранній стадії.
- Веб-редактор повинен дозволяти відкривати, зберігати, копіювати та переміщувати файли та папки в проекті. Це також включає підтримку керування версіями файлів і можливість роботи зі сховищами контролю версій, такими як Git.
- Праця з кількома файлами одночасно, відкриваючи кожен файл на окремій вкладці чи панелі. Це корисно для редагування багатофайлових

проектів або для швидкого переходу між різними фрагментами коду. Підтримка плагінів і розширень:

- Розширення їх функціональність за допомогою плагінів або розширень. Це дозволяє налаштувати редактор під власні потреби та використовувати додаткові інструменти, такі як система управління проектами, інтеграція із зовнішніми службами, налагоджувач тощо.
- Забезпечення портативності та доступності, наприклад через веб-браузер або мобільний додаток, є важливим фактором для багатьох користувачів.

Щоб зробити процес створення веб-сайту простішим, зручнішим і швидшим, програмісти створили веб-редактори для себе та своїх колег.

Веб-редактор - програма для редагування документів HTML. Вони бувають двох видів: текстові та візуальні. Текстові використовуються для самостійного компіляції коду, вони більш складні, тому їх використовують професіонали для створення веб-сторінок [1]. Візуальні редактори також називають їх WYSIWYG (What You See Is What You Get), їх набагато простіше організувати, вони вже мають базові елементи, а при редагуванні код змінюється автоматично. Це означає, що з візуальним редактором може працювати навіть фахівець. Але є проблема. При роботі з візуальним редактором код сторінки стає великим, що збільшує її вагу. Але це можна виправити під час редагування цього коду, і для цього потрібно знати HTML. Найзручнішим варіантом для розробників веб-сайтів є гібридний режим, коли на одній половині екрана відображається код, а на іншій – інтерфейс сайту [7]. Це означає, що два типи веб-редакторів добре працюють разом і доповнюють один одного.

Однак існують не тількі повноцінні редактори, які треба встановлювати на локальний комп'ютер. Онлайн-редактори — види веб-редактора, які дозволяють користувачам створювати та редагувати веб-сайти без встановлення додаткового програмного забезпечення на своїх комп'ютерах. Вони доступні із веб-браузера та забезпечують простий у використанні інтерфейс для роботи з кодом і веб-елементами.

Якщо для роботи з веб-редакторами треба відподідні знання та навички, то є інші інструменти створення веб-сайтів, які цього не потребують. Веб-конструктори та системи управління контентом (CMS) є популярними засобами для створення і публікації веб-сайтів. Ці інструменти дозволяють користувачам без технічних навичок створювати та керувати своїми веб-проектами.

Конструктор сайту — це інтерактивний інструмент, який дозволяє створювати веб-сайти шляхом перетягування готових блоків, шаблонів, зображень і тексту. Зазвичай він має інтуїтивно зрозумілий інтерфейс, який дозволяє швидко створювати та налаштовувати веб-сторінки. Веб-конструктор має обмежені можливості, але ідеально підходить для непрограмістів, любителів і малого бізнесу.

CMS — це платформа, яка дозволяє створювати, редагувати та керувати вмістом веб-сайту. Він пропонує широкий спектр функцій, таких як створення сторінок, керування меню, додавання медіафайлів, керування користувачами тощо. CMS зазвичай базується на зручному для користувача інтерфейсі та використовує базу даних для зберігання вмісту. Вони підходять як для малих, так і для великих веб-проектів.

В даний час існує велика кількість безкоштовних веб-редакторів. Питання про те, що вибрати, не тільки об'єктивний, але і суб'єктивний.

Об'єктивними показниками є інтерфейс, функціональність, швидкість. До суб'єктивних показників можна зарахувати комфорт, звичку, лояльність.

Однак слід зазначити, що лише створенням сайту справа не обмежиться. Його буде необхідно розвивати та підтримувати, збільшувати конверсію та своєчасно поповнювати, а це завдання не з легких.

1.2. Інструментальні засоби для створення і дизайну сайтів

Будь-який сайт складається із зовнішньої та внутрішньої частин. Зовнішня частина – це дизайн, а внутрішня – програмний код. Іншими словами, сайт є програмою, і її створення має спиратися на певні технології, виконуватись із використанням тих чи інших мов програмування. Основними програмними засобами для створення сайтів є:

- Веб-конструктори, є популярними інструментами для створення сайтів без необхідності програмування. Вони надають широкий вибір готових шаблонів та інтерфейс, за допомогою якого можна вносити зміни до вмісту, дизайну та структури сайту;

- CMS (Системи управління контентом) - потужний інструмент для створення та керування веб-сайтами. Вони надають можливість легко створювати та редагувати вміст, управляти меню, розміщувати медіафайли та керувати різними аспектами веб-сайту;

- Графічні редактори є необхідними інструментами для створення та редагування графічного контенту на веб-сайтах, такого як зображення, ілюстрації та графічні елементи. Вони дозволяють розробникам створювати привабливий дизайн та графічні ефекти.

- Кодові редактори незамінний інструмент для розробників, що створюють сайти та будь-який веб-продукт за допомогою коду. Вони надають велику кількість корисних функцій, включаючи окрему середу розробки;

Веб-конструктори що є популярними серед користувачів:

Wix — один із найпопулярніших веб-конструкторів, який пропонує широкий спектр можливостей для створення професійних веб-сайтів. За допомогою нього можна створити свій веб-сайт з нуля або скористатися одним із готових шаблонів, які можна налаштувати відповідно до власних потреб. Wix пропонує простий редактор із функцією перетягування, який дозволяє легко додавати та впорядковувати елементи на веб-сторінці. Використовуючи його, отримаєте налаштування власного дизайну, мультимедійні елементи, оптимізація веб-сайту для пошукових систем та багато іншого.

Squarespace відомий своїми стильними та професійними шаблонами, які можна використовувати для створення чудових веб-сайтів. Можливість налаштовувати параметри SEO, щоб покращити видимість вашого веб-сайту та всі інші можливості конструктора також представлені в ньому.

Weebly — ще один популярний конструктор веб-сайтів, який зосереджується на простоті використання та швидкому впровадженні веб-сайту. Він пропонує багатий функціонал повноцінного конструктора сайтів, а також є можливість додати функціональність за допомогою плагінів.

Серед систем управління контентом найчастіше використовують:

WordPress - популярна система керування вмістом (CMS), яка використовується для створення багатьох типів веб-сайтів, від блогів до корпоративних порталів. Він пропонує велику кількість тем і плагінів, які

дозволяють налаштувати дизайн і функціональність вашого веб-сайту. WordPress має зручний інтерфейс керування, легку публікацію та оновлення вмісту, а також вбудовані інструменти оптимізації SEO.

Joomla - система управління контентом, яка використовується для розробки різноманітних типів веб-сайтів. Вона надає багатофункціональну платформу з великою кількістю розширень та шаблонів. Joomla дозволяє налаштовувати структуру сайту, керувати контентом, створювати блоги, форуми, магазини і багато іншого. Її інтерфейс є досить складним для новачків, але вона надає багато можливостей для досвідчених користувачів, які шукають більшу гнучкість та розширення.

Графічні редактори для розробки дизайну сайтів:

Adobe Photoshop має досить широку сферу використання. З допомогою графічного редактора можна високоякісно створювати та обробляти зображення (зміна насиченості, накладання фільтрів, обрізання, коригування світла, виправлення перспективи тощо); працювати з текстом та нескладними фігурам, задаючи їм стилі для оформлення; обробляти медіафайли, анімацію. Зокрема, розширена версія Adobe Photoshop Extended дозволяє працювати з 3D-файлами. Використовуючи програму, можна також перетворювати плоскі фотографії у тривимірні об’єкти (піраміда, конус, циліндр і т.д.) [8].

Платформу Canva використовують для розробки графіки, презентацій, плакатів, мудбордів, логотипів, оформлення контенту для соцмереж та багато іншого. Користувачам доступний великий набір шрифтів, безкоштовних шаблонів та зображень. Користувачам доступна велика кількість шрифтів, безкоштовних шаблонів і зображень. Платні версії Canva Pro та Canva for Teams мають такі додаткові функції: видалення фону; доступні преміум шаблони, анімація; завантаження дизайнів, як шаблонів; додавання тегів до медіафайлів та інше [9].

Adobe Illustrator CC — провідна програма для розробки векторної графіки та дизайну. Цей сервіс дуже універсальний і використовується для створення логотипів, малюнків, значків, бейджів, ілюстрацій для публікацій, рекламних проспектів, макетів сайтів тощо. Adobe Illustrator CC містить широкий спектр кольорів у різних відтінках; доступна синхронізація з програмами Adobe; пошук шрифтів; сенсорна робоча область; інструменти для встановлення та підключення точок; інструменти навігації (збільшення/зменшення, перетягування ілюстрацій); інструменти малювання [10].

Figma надає можливість створювати дизайн-макети веб-сторінок або мобільних додатків. Ви можете створювати та редагувати шари, векторні графічні елементи, текст, фони та інші дизайн-елементи, інтерактивні прототипи веб-сайтів або мобільних додатків, додавати переходи між сторінками, встановлювати анімацію, створювати взаємодію з елементами і тестувати взаємодію користувача [16].

Редактори коду, що надають зручне середовище для написання коду та мають різноманітні функції для продуктивності розробника:

Visual Studio Code (VS Code) є одним з найпопулярніших редакторів коду, який надає розширений функціонал для розробки різних типів програмного забезпечення, розпізнає різні мови програмування і надає підсвічування синтаксису для полегшення читання та редагування коду. Редактор має вбудований термінал, що дозволяє виконувати команди безпосередньо з редактора, що полегшує роботу з проектами та підтримує розробників [12].

Sublime Text є популярним редактором коду, який відомий своєю швидкодією та легкістю використання. Завдяки швидким клавішним скороченням і можливості налаштування власних команд, Sublime Text

дозволяє швидко виконувати різні дії без застосування миші. Редактор має потужну пошукову функцію, яка дозволяє швидко знаходити та замінювати текст у великих проектах або файлах [13].

Atom — розширюваний текстовий редактор, розроблений GitHub. Він має відкритий код і пропонує широкі можливості редагування та програмування. Atom має вбудовану підтримку системи контролю версій Git, що дозволяє взаємодіяти зі своїм репозиторієм, переглядати зміни, вносити зміни та фіксувати їх безпосередньо з редактора [14].

А також онлайн-редактори коду, з якими можна працювати у браузерах з будь-якого місця та пристрою:

CodePen - один із найпопулярніших онлайн-редакторів коду. Він підтримує розробку HTML, CSS і JavaScript і має потужний редактор із попереднім переглядом у реальному часі. CodePen дозволяє користувачам створювати власні проекти, співпрацювати над кодом з іншими користувачами та ділитися результатами зі спільнотою. Він також пропонує багато готових до використання шаблонів і бібліотек, які допоможуть вам швидко розпочати роботу.

JSFiddle - онлайн-редактор коду, призначений для розробки проектів JavaScript. Він забезпечує зручне середовище для написання, налагодження та тестування коду JavaScript. JSFiddle має вбудовану підтримку різних бібліотек і фреймворків, таких як jQuery, React і AngularJS. Крім того, це дозволяє вам співпрацювати над проектами та бачити результати в режимі реального часу.

1.3. Аналіз характеристик існуючих веб-редакторів

Для проведення детального аналізу характеристик існуючих веб-редакторів використаємо таблицю 1.3, де описано функціональні переваги та

недоліки редакторів коду один між одним. До порівняння наведені: CodePen - онлайн-редактор, що можна запустити з браузеру, VS Code – повноцінний, локальний редактор коду, WebEditor2007 – система управління контентом та Wix – онлайн конструктор сайтів. Порівнюємо та аналізуємо представників з кожної категорії іструментів для веб-розробки.

Таблиця 1.3 Порівняльна характеристика веб-редакторів.

Функціонал	CodePen	VS Code	WebEditor 2007	Wix
Підтримка мов	HTML, CSS, Js	Багато мов	HTML, CSS	HTML, CSS
Автодоповнення коду	+	+	-	+
Перевірка синтаксису	+	+	-	+
Інтеграція з Git	-	+	+	+
Розширюваність	Обмежена	Велика	Обмежена	Обмежена
Спільна робота	Обмежена	+	-	+
Візуальний редактор	+	-	+	+
Швидкодія	Середня	Висока	Низька	Середня
Хмарне зберігання	+	-	-	+

Варто зауважити, що деякі з можливостей цих редакторів можуть варіюватися в залежності від версії, налаштувань та плагінів, що використовуються. Детальний аналіз таблиці дозволяє зробити такі висновки:

CodePen - редактор підтримує різні мови програмування, має автозаповнення коду та перевірку синтаксису. Він підходить для швидкого прототипування та демонстрації коду в онлайн-середовищі.

VS Code - веб-редактор має велику кількість функцій і розширень, які роблять його потужним інструментом розробки. Він підтримує кілька мов програмування, має вбудовану підтримку Git і розширюваність.

WebEditor2007 - ця система управління контентом має базову функціональність, обмежену підтримку мови програмування та візуальний редактор. Його можна використовувати для зміни простих HTML і CSS, але може бути недостатньо для більш складних проектів.

Wix - як веб-конструктор Wix пропонує візуальний редактор, який дозволяє створювати веб-сайти без програмування. Він підтримує HTML і CSS, а також пропонує хмарне сховище та функції командної співпраці.

Кожен із цих веб-редакторів має свої переваги та обмеження. Вибір веб-редактора залежить від власних потреб і проекту. Якщо вам потрібен простий інструмент для швидкого прототипування, CodePen може стати хорошим вибором. Для професійного розвитку з багатим функціоналом і розширюваністю VS Code є чудовим вибором. WebEdit2007 можна використовувати для базового редагування, а Wix — це зручний веб-конструктор для створення веб-сторінок без програмування.

РОЗДІЛ 2.
РОЗРОБКА МОДЕЛІ ПРОГРАМИ

2.1. Загальна концепція програми

Для розробки веб-сайтів використовують як онлайн редактори так і повнофункціональні, що встановлюються локально на комп'ютер.

Онлайн-редактор - це веб-програма, яка дозволяє створювати та редагувати код безпосередньо у веб-браузері. Це дозволяє користувачам створювати веб-сайти без необхідності встановлення спеціалізованого програмного забезпечення на своїх пристроях. Онлайн-редактори зазвичай пропонують зручний інтерфейс, підсвічування синтаксису коду, автозаповнення та інші функції, які спрощують процес розробки веб-сайту як для новачків, так і для досвідчених розробників. Вони дозволяють швидко отримати доступ до редактора з будь-якого пристрою з підключенням до Інтернету. З іншого боку, повнофункціональні редактори коду, встановлені локально на комп'ютері, забезпечують більш розширену функціональність і контроль над процесом розробки. Вони надають можливість редагувати код безпосередньо на робочій станції, надаючи різноманітні інструменти для аналізу коду, налагодження та оптимізації. Повнофункціональний редактор може бути більш продуктивним і ефективним під час роботи над великими проектами та інтеграції з іншими інструментами розробки [4].

Порівнявши види редакторів, загальна концепція програми полягає в створенні онлайн веб-редактора для створення, тестування та редагування веб-додатків. Програма надасть можливіть розробникам легко та швидко

вносити зміни, розробляти прості веб-додатки не використовуючи та не завантажуючи повноцінні редактори.

Основна ідея створення програми – надати можливіть веб-розробникам створювати та візуалізовувати свої веб-додатки без необхідності встановлення локального середовища розробки, швидко та зручно вносити правки, легко та ефективно вчитися.

Визначення цільової аудиторії є важливим кроком у розробці веб-редактора, оскільки це допомагає враховувати потреби, вимоги та очікування користувачів. Розуміючи цільову аудиторію, можна розробити функції, інтерфейс користувача та функції, які найкраще відповідають потребам користувачів. Аналізуючи ці фактори, отримаємо веб-редактор націлений на загальну цільову аудиторію, що включає:

- Веб-розробник: Вони професіонали у створенні веб-сайтів і веб-додатків. Веб-розробники можуть використовувати веб-редактор для швидкого редагування коду, внесення правок, створення структури сторінки, додавання елементів, створення стилів і візуального налаштування веб-елементів.
- Студент: Веб-редактори корисні для студентів, які вивчають веб-розробку або суміжні галузі. Це дозволяє студентам практикуватися та експериментувати з кодом, створювати веб-сторінки та переглядати результати в режимі реального часу.
- Непрофесійні користувачі: Люди, які не мають професійних навичок веб-розробки, але потребують редагування HTML-коду. Власник сайту або блогер. Веб-редактори дозволяють змінювати власний веб-вміст, додавати нові елементи, форматувати текст та вносити інші зміни.

- Дизайнер: Веб-редактор корисний для дизайнерів, які працюють над веб-проектами та хочуть вносити власні зміни в HTML-код, наприклад стиль і розміщення елементів на сторінці.
- Вчитель: Педагоги, які займаються веб-розробкою, можуть використовувати веб-редактор як навчальний інструмент для демонстрації та виконання практичних завдань зі створення веб-сторінок.

Визначивши цільову аудиторію програми, а саме студенти та вчителі, можна перейти до методів розробки. У таблиці 2.1 описані декілька методів, що допоможуть у створенні веб-редактору.

Таблиця 2.1. Методи розробки.

Методи розробки	Опис
Ітеративний підхід	Розробка веб-редактора проводиться шляхом послідовного виконання ітерацій. Кожна ітерація включає аналіз, проектування, реалізацію та тестування.
Водопадний підхід	Розробка веб-редактора виконується послідовно, від одного етапу до іншого. Кожний етап (аналіз, проектування, реалізація, тестування) завершується, перш ніж розпочнеться наступний.
Agile-методології	Веб-редактор розробляється згідно з принципами гнучкого управління проектом. Розробка відбувається у коротких ітераціях, що дозволяє гнучко адаптуватися до змін і вимог користувачів.

Компонентний підхід	Розробка веб-редактора зосереджена на створенні незалежних компонентів, які можуть бути використані повторно і забезпечують високу модульність та розширюваність системи.
Модель-представлення-контролер (MVC)	Веб-редактор розбивається на три основні компоненти: модель (дані), представлення (інтерфейс) та контролер (логіка). Ця модель дозволяє розділити відповідальності та спростити розробку та підтримку редактора.

Методи розробки мають вирішити потреби цільової аудиторії, забезпечуючи їм зручність, ефективність та задоволення від використання редактора. Комбінування різних методів розробки дозволить створити веб-редактор, який відповідає потребам та очікуванням цільової аудиторії.

Для розробки веб-редактора використаний компонентний підхід. Цей підхід дозволяє розділити функціональність редактора на окремі компоненти. Це допомагає краще організувати код і полегшує розуміння структури проекту. Кожен компонент може виконувати свою певну функцію, що спрощує розробку, тестування та модифікацію різних частин системи. Готові компоненти можна повторно використовувати в інших проектах або в різних частинах того самого проекту. Це забезпечує ефективне використання ресурсів і прискорює процес розробки. Компоненти дозволяють легко додавати нову функціональність до системи без необхідності змінювати існуючий код. Це дозволяє динамічно розширювати редактор, додаючи нові функції та модулі під час розробки проекту. Можна тестувати компоненти незалежно, що дозволяє виявляти та виправляти помилки в різних частинах

системи. Це спрощує процес тестування та покращує якість програмного забезпечення. щоб забезпечити найкраще поєднання ефективності, якості та досвіду користувача.

Веб-редактори мають складну структуру, що включає різні компоненти та взаємозв'язки між ними. На Рис. 2.1. зображена діаграма, що описує ці зв’язки.

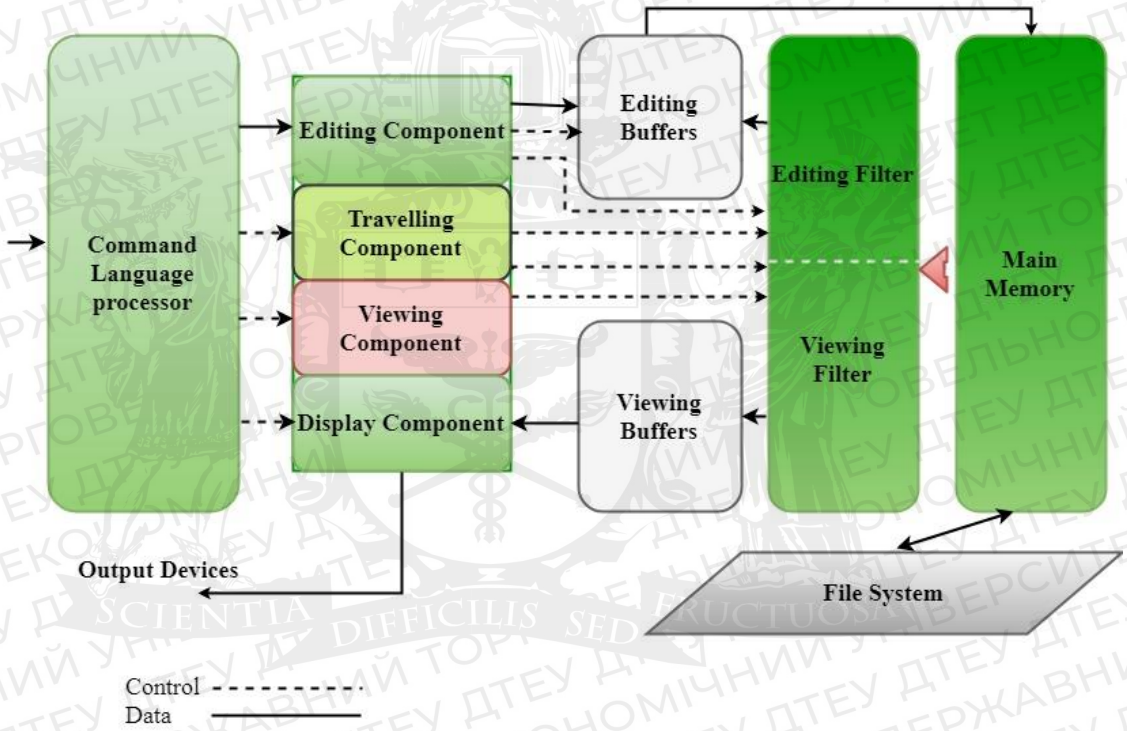

Рис. 2. 1. Структура компонентів редактора.

‘Command Language processor’ виконує функції редагування та перегляд. Це передбачає передавання, редагування, перегляд і показ. Операції редагування ‘editing’ задаються користувачем, а операції відображення ‘display’ – редактором. Компоненти передачі та перегляду викликаються редактором або самим користувачем під час виконання операцій.

‘Editing component’ - це модуль, який займається завданнями редагування. Поточна область редагування визначається його поточним

покажчиком, пов’язаним із компонентом. Коли виконується команда редагування, ‘editing component’ викликає ‘editing filter’, генерує новий буфер редагування. Буфер редагування ‘editing buffer’ містить документ для редагування в поточному місці покажчика редактора.

Під час перегляду документа початок області для перегляду визначається поточним покажчиком перегляду. ‘Viewing component’ - це набір модулів, які використовуються для перегляду. Поточний перегляд можна встановити або скинути залежно від останньої операції.

Коли відображення потрібно оновити, компонент перегляду викликає фільтр перегляду, створює новий буфер і містить документ, який потрібно переглянути за допомогою поточного буфера перегляду. Потім буфер перегляду передається компоненту відображення, який створює відображення шляхом відображення буфера. Буфери редагування та перегляду можуть бути ідентичними або повністю роз’єднаними. Буфери редагування та перегляду також можуть частково перекриватися або міститися один в одному. Компонент редактора взаємодіє з документом від користувача на двох рівнях: основна пам'ять і файлова система диска

Чітко визначена функціональна структура є основою ефективного розроблення і підтримки веб-редактора. Основні складові включають:

- HTML-структура;
- CSS-стилі;
- JavaScript-скрипти;
- Бібліотеки та фреймворки;
- Інтерфейс користувача;
- Функціональні модулі.

HTML-структура визначає структуру інтерфейсу та розташування елементів. Використовуючи різні теги такі як ‘ ’, ‘<button>’, ’<input>’ та

інші створюються інтерактивні елементи інтерфейсу, розмічається розділи та структура сайту, придаються різні значення тексту, додається медіа-контент на веб-сторінку. Чудово для демонстрації типового макету сайту, однак без CSS-стилів це все виглядає примітивно.

CSS (каскадні таблиці стилів) використовується для стилізації та компонування веб-сторінок - наприклад, для зміни шрифту, кольору, розміру та інтервалу вмісту, поділу його на кілька стовпців або додавання анімації та інших декоративних елементів. З цим інструментом веб-редактор буде інтуїтивно зрозумілий, зменшить час на адаптацію користувача та стане легким у користуванні, що підвищить продуктивність.

Сценарії JavaScript використовуються для надання динамічних функцій редактору. Вони відповідають за обробку подій, взаємодію з користувачем, зберігання та обробку даних. Скрипт JavaScript містить різні функції, які реалізують функціональні можливості редактора. Сценарії JavaScript взаємодіють зі структурою HTML і стилями CSS для створення інтерактивних і функціональних веб-редакторів.

Ці три компоненти (структура HTML, стилі CSS, сценарії JavaScript) разом створюють повний веб-редактор. Вони визначають зовнішній вигляд, функціональність і поведінку редактора, дозволяючи користувачам створювати, редагувати та налагоджувати веб-код зручно й ефективно.

Бібліотеки та фреймворки веб-розробки є незамінними інструментами для створення складних і функціональних веб-додатків. Вони надають готові компоненти, функції та інструменти, які спрощують розробку, забезпечують швидке розгортання та допомагають підтримувати якість і ефективність проекту.

Функціональні модулі веб-розробки — це додаткові компоненти або пакети, які додають функціональність веб-додаткам. Вони пропонують готові рішення для різних завдань і допомагають прискорити процес розробки.

2.2 Функціональна модель програми

Для створення функціональної моделі програми слід визначити вимоги, котрі програма повинна вміти робити та як функціональні компоненти залежать один від одного.

Вимоги до функціоналу редактору:

- Редагування коду в різних режимах;
- Автоматична підсвітка синтаксису та автодоповнення;
- Перегляд та попередній перегляд;
- Управління мовами;
- Спільний, корпоративний доступ;
- Доступ до кастомізації робої середи;
- Підтримка клавіатурних швидких команд для зручної роботи.

Функціонал створення коду в онлайн редакторі повинен забезпечити зручну та ефективну роботу розробників при написанні програмного коду. Саме онлайн редактори надають зручне інтерфейс-середовище для написання та редагування. Редактор підтримує функції форматування тексту, автодоповнення, вирівнювання та автоматичне виявлення синтаксичних помилок. Функція редагування коду автоматично показує пропозиції та варіанти коду, коли користувач починає вводити певний символ або ключове слово. Наприклад, введення "f" може показати параметри, що починаються з "for", "function" та інші. Це зменшує кількість помилок і швидше записує

необхідний код. Функкція підсвітки синтаксису підсвічує різні елементи коду різними кольорами, що допомагає розробникам краще розуміти структуру та логіку програми. Після введення HTML-коду в редакторі, можна переглянути його результат. Існує декілька підходів для реалізації цього процесу:

1. Автоматичне оновлення:

Цей підхід включає той факт, що кожен раз, коли код змінюється, редактор автоматично оновлює візуалізацію результату. Це дозволяє користувачам негайно побачити вплив своїх змін без необхідності ручного оновлення.

2. Кнопка «Переглянути» або «Оновити»:

У цьому варіанті користувач має можливість вручну запустити процес перегляду результатів, натиснувши відповідну кнопку. У той же час редактор скомпілює та відобразить поточну версію веб-сторінки на основі останнього збереженого коду.

3. Інтерактивний перегляд:

Такий підхід дозволяє користувачеві візуалізувати результати під час модифікації коду. У цьому режимі зміни коду миттєво візуалізуються, дозволяючи користувачам спостерігати за впливом своїх дій.

4. Розділений екран:

У деяких редакторах ви можете розділити екран на дві частини, одна з яких показує код, а інша - результати в реальному часі. Це дозволяє користувачеві одночасно змінювати код і спостерігати за результатами на одному екрані.

5. Проникливе розуміння:

Цей підхід полягає в тому, що користувачі можуть попередньо переглянути результати своїх змін перед збереженням або публікацією. Це

дає змогу остаточно перевірити зовнішній вигляд вашого веб-сайту та переконатися, що він правильно відображається.

Проаналізувавши методи реалізації процесу перегляду результату, прийняте рішення використати метод інтерактивного перегляду у поділеному екрані та кнопки «Переглянути». Інтерактивний режим перегляду дозволить користувачу швидко побачити, як виглядатиме веб-сторінка одночасно з редагуванням коду у відповідному вікні, а кнопка «Переглянути» служитиме для відкриття веб-сторінки у новій вкладці, щоб користувач зміг оглянути та проаналізувати код.

Для забезпечення зручного використання та полегшення роботи з онлайн редактором, була реалізована підтримка клавіатурних команд за допомогою JavaScript. Він зазвичай використовується для створення сценаріїв веб-сайтів, надання клієнтам можливості взаємодіяти з користувачами, керування браузерами, асинхронного обміну даними з сервером і зміни структури та зовнішнього вигляду сторінки. Ця функціональність дозволяє користувачам використовувати спеціальні комбінації клавіш для виконання певних дій без необхідності використання миші або інших елементів інтерфейсу.

При реалізації цієї функції було використано події клавіатури і обробник подій, що дозволяють відслідковувати натискання клавіш користувачем. Наприклад, при натисканні комбінації "Ctrl + S", запускається функція для збереження робочого проекту. Для зручності користувачів також були додані інші клавіші, які виконують певні дії, такі як відміна останньої дії, переміщення по коду, відкриття допоміжних панелей тощо. Коли натискаються відповідні клавіші або комбінації клавіш, виконуються відповідні дії, які можуть включати виклик функцій, зміну редактора, взаємодію з користувачем тощо. Крім того, користувачеві можуть

відображатися коментарі, наприклад повідомлення про успішну реєстрацію або скасування дії.

Підтримка клавіатурних команд реалізована з урахуванням кросбраузерності та можливих варіацій клавіш і їх кодів для різних браузерів, що забезпечує продуктивність у різних середовищах.

Також було влучно реалізована панель інструментів для швидкого редагування та створення документів. Кожна кнопка заміняє собою тег або групу тегів, що значно пришвидшує та полегшує роботу і не займає багато місця на екрані. Реалізовані кнопки роботи з документом, що включають створення, відкриття та зберігання документу. Скасування та повторення скасованої дії, форматування тексту та вставки різних елементів, таких як зображення, посилання тощо. Вся панель інструментів підвищить продуктивність роботи та навчання, що відповідає вимогам редактора.

Процес збереження та завантаження проекту може здійснюватися за допомогою різних методів і технологій. Основний принцип полягає в тому, щоб дозволити користувачам зберігати свої проекти на сервері та використовувати їх пізніше, щоб змінювати або завантажувати їх на свій пристрій, змінювати код і спостерігати за результатами на одному екрані.

РОЗДІЛ 3.
ВЕБ-РЕДАКТОР ЯК ОДИН З ВИДІВ ІНСТРУМЕНТАЛЬНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Інформаційно-логічна модель веб-редактора

Загальна концепція інформаційно-логічної моделі редактора являє собою представлення структури та організацію даних, що обробляються в редакторі, а також зв’язки між різними елементами редактора. Для розробки структури програмного забезпечення слід створити діаграму класів. Вона використовується в моделюванні класів та атрибутів і взаємозв'язках між ними. Діаграма складаєтсья з класів – блоки, що описують структуру та поведінку об'єктів і містять атрибути; атрибутів, що представляють дані, які зберігаютсья в об'єктах класу; методів та взаємозв'язків.

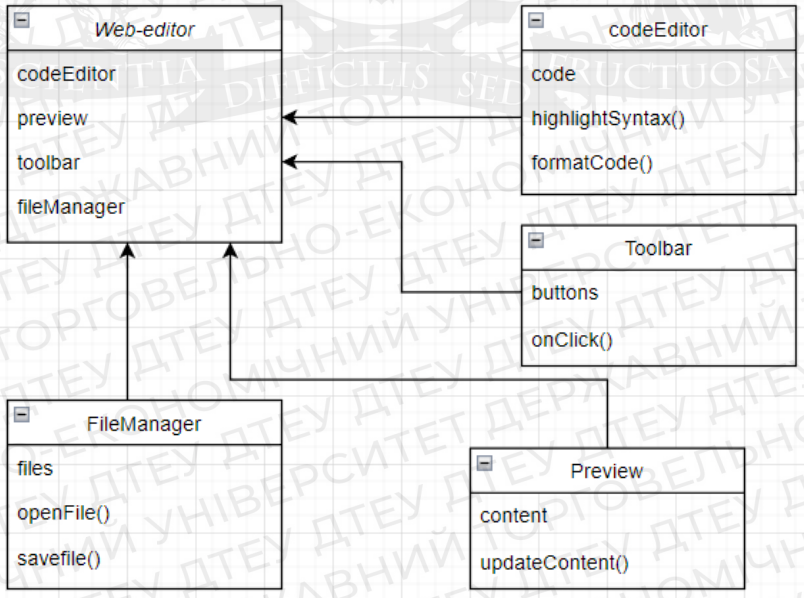

Рис. 3.1. Діаграма класів веб-редактора.

У цій діаграмі (Рис. 3.1.) маємо головний клас ‘WebEditor’, який включає в себе різні компоненти веб-редактора. ‘WebEditor’ має залежність від ‘CodeEditor’, ‘Toolbar’ та ‘Preview’. ‘CodeEditor’ відповідає за редагування коду і має методи для підсвічування синтаксису та форматування коду. ‘Toolbar’ включає кнопки і відповідає за обробку подій кнопок. ‘Preview’ відображає попередній перегляд зміненого коду. Діаграма класів дає нам уявлення про структуру системи, вона показує, як організовані класи, об’єкти та їхні зв’язки. Це дозволяє нам визначити основні класи системи, їхні властивості та методи, а також зрозуміти, як вони взаємодіють один з одним.

Однак, щоб зрозуміти, як саме працює система, які операції виконує користувач, як відбувається взаємодія між користувачем і системою, нам потрібна діаграма активності. Діаграми діяльності використовуються для моделювання послідовності дій, реалізації процесів, взаємодії зовнішніх суб’єктів.

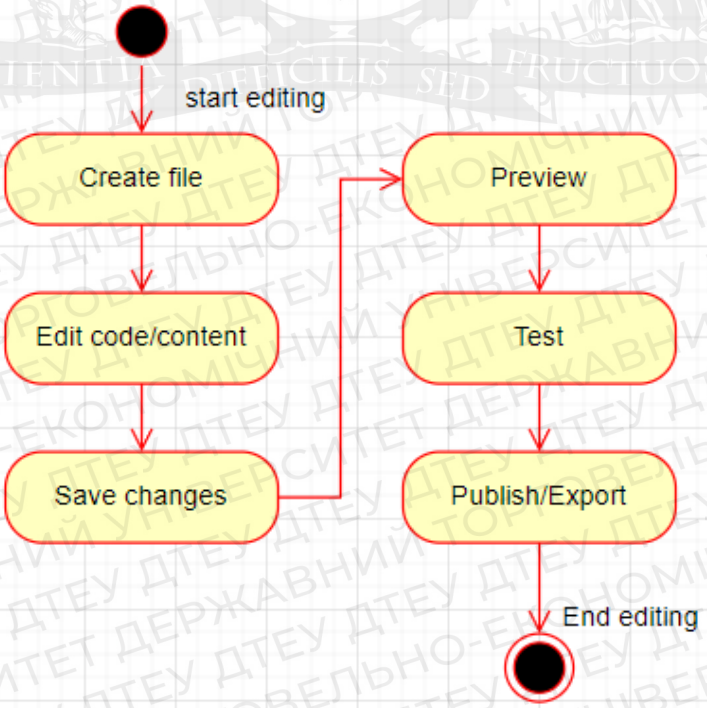

Рис. 3.11. Діаграма активності користувача

Процес починається з того, що користувач вибирає «Start editing», щоб розпочати роботу з веб-редактором. Користувачі мають можливість створити новий файл або вибрати наявний файл для редагування. Цей крок представлено дією «Create file». Основною діяльністю є крок «Edit code/content», де користувач може вносити зміни до коду або вмісту файлу. Після внесення необхідних змін користувач може зберегти зміни, вибравши дію «Save changes». Після збереження змін користувачі можуть забажати попередньо переглянути, а після - протестувати файл, щоб переконатися, що він виглядає та працює належним чином. Це представлено операціями «Preview» та «Test» відповідно. Якщо все в порядку, користувач може продовжити публікацію або розгортання файлу, щоб зробити його доступним для інших. Цей крок представлено операцією «Publish/Export». Нарешті, користувач може завершити процес редагування, вибравши дію «End editing».

Діаграма діяльності зосередженна на моделюванні послідовності дій і процесів, визначаючи, які дії виконуються, гілки та умови, що розглядаються, і які шляхи вибираються під час взаємодії з системою. Вона показує хід дій та їх взаємозв’язки, допомагаючи зрозуміти логіку та послідовність подій. Діаграма послідовності, з іншого боку, фокусується на моделюванні взаємодії між об’єктами або компонентами системи. Вона визначає, як об’єкти взаємодіють один з одним, які повідомлення надсилаються та отримуються між ними, а також порядок, у якому відбуваються ці події. Діаграми послідовності допомагають визначити зв’язки та зв’язок між об’єктами системи. Вона включає такі етапи: (Рис. 3.12)

1) Користувач відкриває редактор у веб-браузері.

2) Редактор завантажує необхідні компоненти та інтерфейс для користувача.

3) Користувач вводить код у вікно редактора.

4) Редактор відправляє введений код на аналіз та підсвічування синтаксису.

5) Після аналізу та підсвічування синтаксису, редактор відображає візуальне представлення коду для користувача.

6) Користувач може редагувати код, додавати нові елементи або виправляти помилки.

7) Редактор надсилає оновлені дані для аналізу синтаксису та підсвічування.

8) Процеси редагування та підсвічування синтаксису повторюються, поки користувач не завершить редагування.

9) Користувач може зберегти свій код, ініціюючи процес збереження на сервері або локальному пристрої.

10) Редактор підтверджує успішне збереження та надає відповідну повідомлення користувачеві.

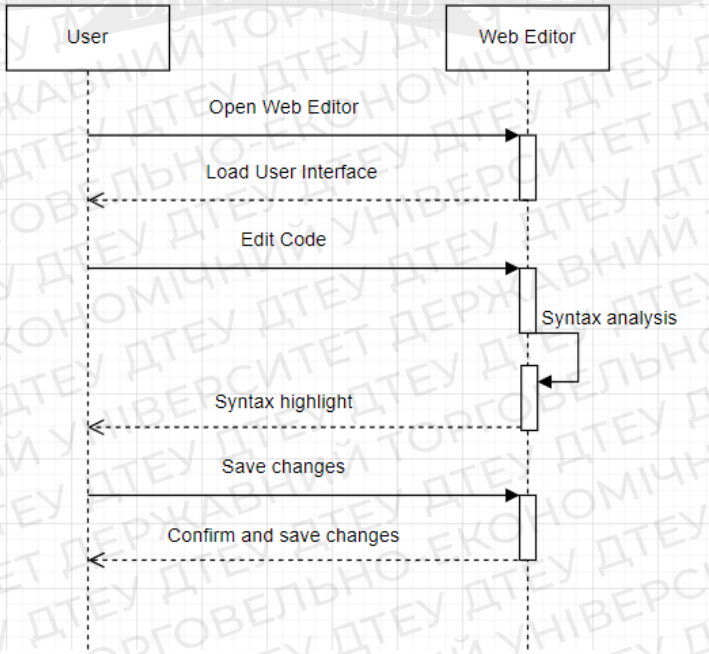

Рис. 3.12. Діаграма послідовності.

UML діаграми можна застосовувати на всіх етапах аналізу бізнес-систем і життєвого циклу розробки додатків. Різні типи діаграм, які підтримує UML, і найбагатший набір можливостей для представлення певних аспектів системи роблять UML популярним способом опису як програмного забезпечення, так і бізнес-систем. Діаграми використовуються для представлення системи (як бізнесу, так і програмного забезпечення) таким чином, щоб її можна було легко перевести в програмний код. Крім того, UML був створений спеціально для оптимізації процесу розробки програмних систем, дозволяючи багаторазово підвищити ефективність їх впровадження та значно підвищити якість кінцевого продукту. Діаграми покращують відстеження проекту та полегшують розробку документів [18].

На підставі зазначеного матеріалу, можна зробити висновок, що веб-редактор матиме загальні фукції, адже призначений для тих хто вчить та навчається розробці. Просте застосуванняя та зручна взаємодія є важливими частинами моделі.

3.2 Особливості програмної реалізації

Описуємо процес реалізації програмного забезпечення, який забезпечує роботу редактора і надає користувачам необхідні функції та можливості що були зазначені у попередніх розділах.

Технічний стек:

Для проекту ми будемо використовувати такий стек технологій:

- React.js – для інтерфейсу;
- TailwindCSS – для стилів;

- Judge0 – для компіляції та виконання нашого коду;
- RapidAPI – для швидкого розгортання коду Judge0;
- Редактор Monaco – редактор коду, який підтримує проект.

Як говорить офіційний слоган, React - це бібліотека для створення інтерфейсів користувача.[11] React не є фреймворком - він навіть не розроблений спеціально для web. Він використовується для візуалізації та поєднання з іншими бібліотеками. Наприклад, React Native можна використовувати для створення мобільних додатків; React 360 можна використовувати для створення програм віртуальної реальності; Крім того, є й інші варіанти.

Розробники використовують React у тандемі з ReactDOM для створення веб-додатків. Основною метою React є мінімізація помилок, які виникають при розробці інтерфейсу користувача. Це досягається за рахунок використання компонентів - незалежних фрагментів логіки, які описують частину інтерфейсу користувача. І ці компоненти поєднуються, щоб створити повноцінний інтерфейс користувача. React видаляє більшу частину роботи з візуалізації, дозволяючи вам зосередитися на дизайні. розробка документів [11].

TailwindCSS - це бібліотека CSS, яка спрощує оформлення HTML подібно до Bootstrap - шляхом додавання різних класів. Але на відміну від Bootstrap, який додає готові елементи, такі як кнопки, сповіщення та панелі навігації, класи TailwindCSS націлені на конкретну властивість. У TailwindCSS немає вбудованої кнопки, її потрібно створити самостійно [19].

Judge0 API — це надійна та масштабована онлайн система виконання з відкритим кодом. Його можна використовувати для створення багатьох типів

додатків, від конкурентоспроможних платформ програмування, освітніх і рекрутингових платформ до онлайн-редакторів коду тощо [20].

Компонент редактора коду в основному складається з Monaco Editor, який є пакетом NPM, який ми можемо використовувати та налаштовувати. Компоненти ‘Editor’ надходять із @monaco-editor/react пакета, який дозволяє нам запускати редактор коду із 85vh заданою висотою.

Компонент ‘Editor’ містить купу реквізитів:

- language: Мова, для якої ми потребуємо підсвічування синтаксису та інтелектуального аналізу;
- theme: кольори та фон фрагмента коду;
- value: фактичне значення коду, яке надходить у редактор коду;
- onChange: це запускається, коли змінюється значення в редакторі коду. Нам потрібно зберегти змінене значення в стані, щоб пізніше можна було викликати Judge0API для його компіляції.

Редактор отримує реквізит onChange, language, code і theme від свого батьківського компонента, яким є Landing.js. Кожного разу, коли value в редакторі коду змінюється, ми викликаємо onChange обробник, який присутній у батьківському Landing компоненті.

Створення Landing компоненти, яка складається з трьох частин:

1. У Actions Bar якому є компоненти, що розкриваються, Languages та Themes;
2. Компонент Code Editor Window;
3. Компоненти Output and Custom Input.

Як розглядалося раніше, компонент CodeEditorWindow враховуватиме код (який постійно змінюється) і метод, onChange, який відстежуватиме зміни

в коді. Встановлюємо стан code і відстежуємо зміни. Компонент CodeEditorWindow також враховує language prop, який є поточною вибраною мовою, для якої нам потрібні підсвічування синтаксису та intellisense. Масив languageOptions, який відстежує прийнятні мовні атрибути Monaco Editor, а також обробляє компіляцію. Кожен languageOptions об'єкт містить ключі id, name, label та value. Весь languageOptions масив можна взяти та помістити в спадне меню та надати як параметри (Рис. 3.2). Кожного разу, коли стан спадного меню змінюється, onSelectChange метод відстежує вибране id та змінює стан відповідно.

Рис. 3.2. Компонент LanguageDropdown та ThemeDropdown.

Для спадного списку використовуємо пакет react-select , який відповідає за спадні списки та їхні обробники змін. React select приймає defaultValue і options як основні параметри. Options - це масив (передаємо його languageOptions тут), який автоматично показує всі відповідні спадні значення.

Prop defaultValue - це значення за замовчуванням, яке надається компоненту. Залишаємо JavaScript як мову за замовчуванням. Щоразу, коли

користувач змінює мову, мова змінюється за допомогою onSelectChange зворотного виклику.

Компонент ThemeDropdown дуже схожий на LanguageDropdown компонент (з інтерфейсом користувача та пакетом react-select). Пакет monaco-themes надає велику кількість тем, які використовуватимемо для зовнішнього вигляду редактору коду. Функція defineTheme відповідає за різні теми, які користувач може вибрати. Функція defineTheme фактично встановлює тему редактора Monaco за допомогою дії monaco.editor.defineTheme (theme, themeData). Цей рядок коду відповідає за фактичну зміну тем у вікні коду Monaco Editor. Функція defineTheme викликається за допомогою зворотного виклику onChange, який використовували раніше в ThemeDropdown.js компоненті. Функція handleThemeChange() перевіряє, чи є тема light або dark. Ці теми за замовчуванням доступні в MonacoEditor компоненті, і не потрібно викликати defineTheme() метод них. Якщо ні, просто викликаємо defineTheme() компонент і встановлюємо стан вибраної теми (Рис. 3.21).

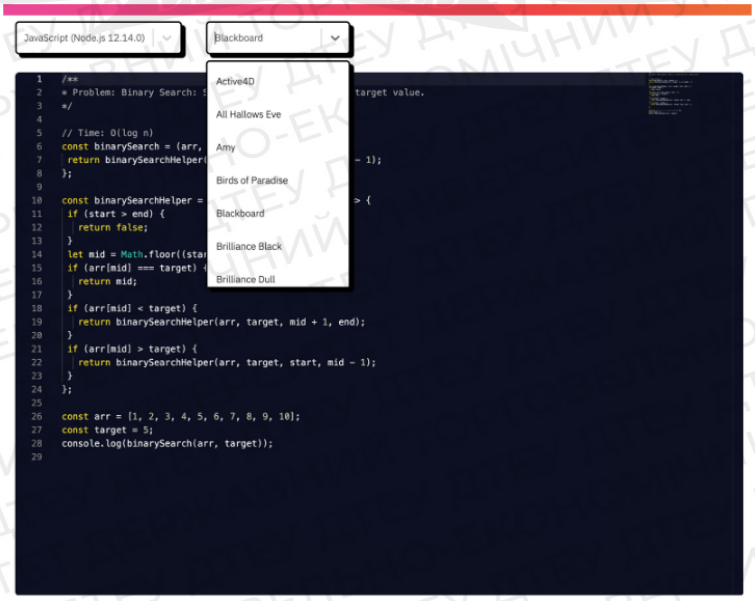

Рис. 3.21. Робота ThemeDropdown.

Компіляція коду різними мовами за допомогою Judge0. Judge0 - це проста система виконання коду з відкритим кодом, з якою можна взаємодіяти. Підписка на базовий план надасть RAPIDAPI_HOST та RAPIDAPI_KEY, які потрібні для викликів API до системи виконання коду.

Послідовність компіляції виглядає наступним чином:

- Compile and Execute після натискання кнопки handleCompile() викликається метод;
- Функція handleCompile() викликає Judge0 RapidAPI серверну частину за submissions URL-адресою з параметрами тіла languageId, source_code, і customInput;
- також options приймає host і secret як заголовки;
- Запит submissionPOST реєструє наш запит на сервері та створює процес. Відповідь на post запит token можна використовувати для перевірки статусу нашого виконання;
- Після повернення результатів можна умовно перевірити, чи результати є успішними чи невдалими, і показати результати на екранах виводу.

Компонент вікна виведення відображає лише відповідні сценарії успіху чи невдачі. Метод getOutput() визначає, яким буде колір тексту і що має бути надруковано:

Якщо statusId = 6 = помилка компіляції. Для цього API повертає дані compile_output, які можна використовувати для відображення помилки.

Якщо statusId = 3 = є сценарій успіху, а саме Accepted. API повертає a stdout, що означає стандартний вихід. Це використовується для відображення даних, які повертаються з коду, який надали в API.

Якщо statusIdє = 5 = помилка Time Limit Exceeded. В коді існує умова нескінченного циклу або він перевищує стандартний час у секундах для виконання коду.

Для кожного іншого стану стандартний stderr об’єкт надає можливість використовувати для відображення помилок.

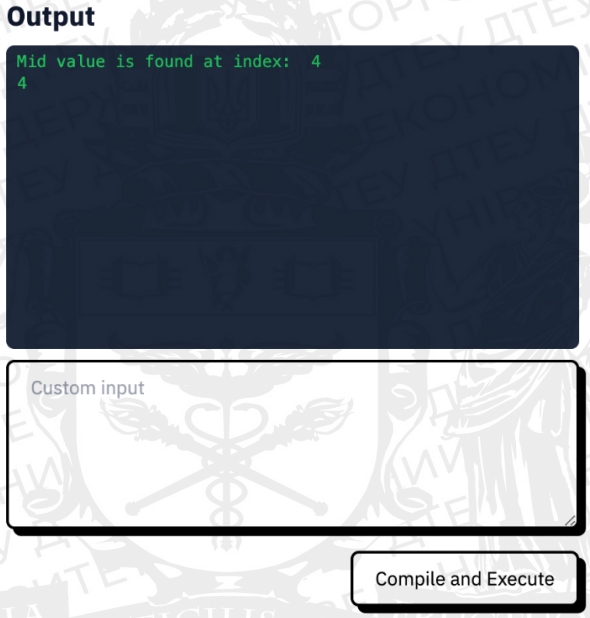

Рис. 3.22. Сценарій успіху програми.

Рис. 3.23. Вихідні деталі компонента.

Компонент OutputDetails служить для відображення деталей, пов'язаних із фрагментом коду, який спочатку був скомпільований. Дані вже встановлено в змінній стану 'outputDetails'. Усі 'time', 'memory' та 'status.description' отримані з відповіді API, зберігаються та відображаються в 'outputDetails'.

Остання річ у додатку – це використання 'ctrl+enter' для компіляції нашого коду, і для цього створюється спеціальний hook. Ініціалізуємо hook цільовим ключем, щоб переконатися, що обидві кнопки натиснуті одночасно. Отже, щоразу, коли користувач натискає 'cntrl+enter' один за одним або одночасно, буде викликаний метод handleCompile().

Отже, користувач зможе обрати бажану мову програмування. Після написання коду він зможе скомпілювати власний код і побачити вихідні дані та результати у вікні виводу. Можливість побачити або успіх, або невдачу для фрагментів коду. Все видно у вікні виведення коду. Користувач може додавати власні введення до своїх фрагментів коду, може бачити відповідні деталі коду, який було виконано.

3.3 Технологія використання і тестування веб-редактора

Для програмної реалізаї поставленого завдання був використаний редактор коду Monaco Editor, який також підтримує VS Code. Для розробки інтерфейсу – React.js, бібліотеку мови програмування JavaScript. А за стилізацію відповідала бібліотека TailwindCSS. Як результат, створений онлайн веб-редактор з полем для вводу коду на різних мовах програмування, (Рис. 3.3) та окремим полем для отримання результату (Рис. 3.22). Також для зручності роботи з редактором протягом великого періоду часу було створено

окрема кнопка для вибору теми, щоб задовольнити потреби кастомізації для кожного користувача (Рис. 3.32), підсвітка синтаксису та автодоповнення для швидкості роботи (Рис. 3.33) та окреме розташування вихідніх деталей, що знаходяться під полем виводу (Рис. 3.23). Усі компоненти/фрагменти коду для багаторазового використання, використання hooks - щоб скомпілювати код за допомогою подій клавіатури.

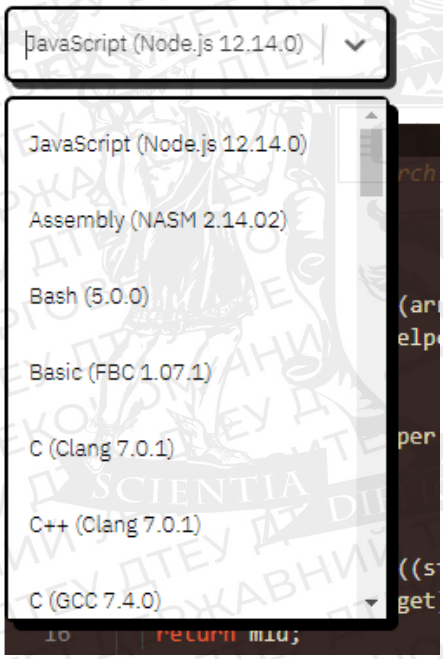

Рис. 3.3. Список мов програмування.

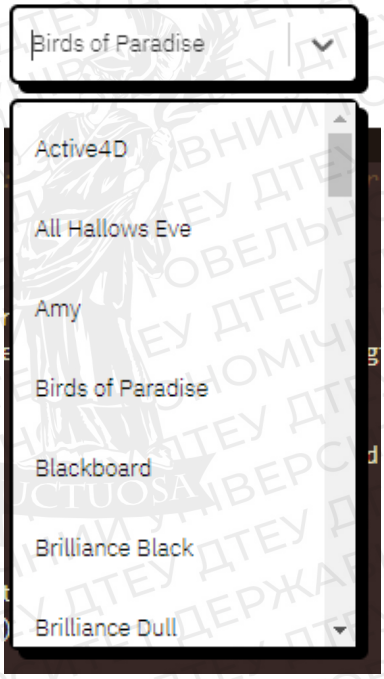

Рис. 3.32. Список тем.

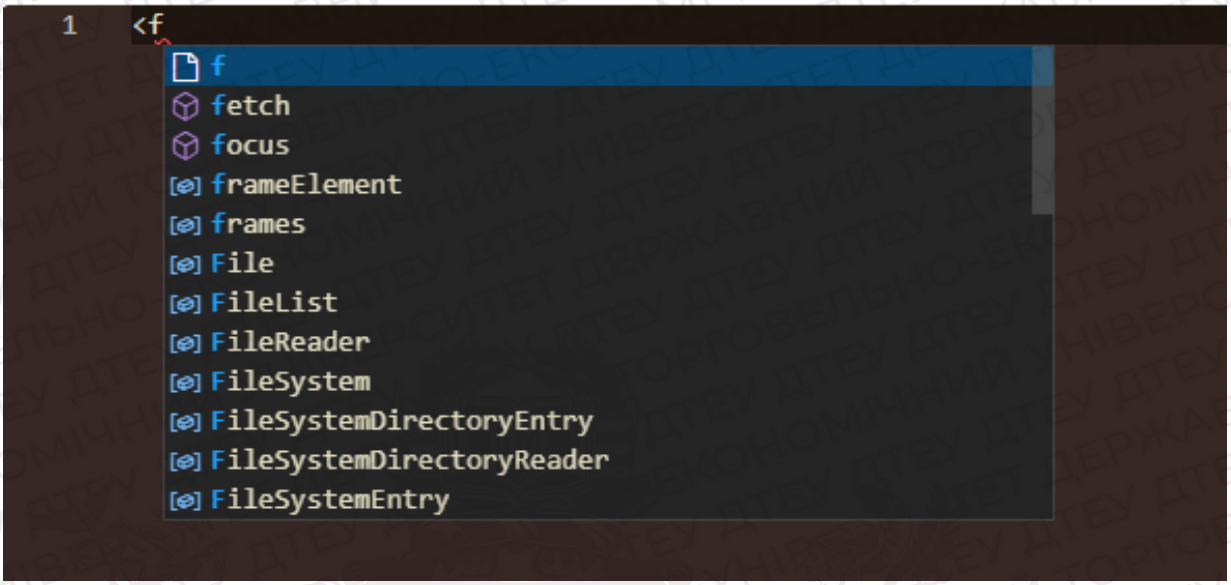

Рис. 3.33. Автодоповнення та підсвітка синтаксису.

Для встановлення та налаштування редактору слід виконати декілька пунктів: (за умови встановленого node.js)

1. Npm install;
2. Зареєструватися на RapidAPI та отримати ключі API (Рис. 3.34);
3. Створіть файл .env;
4. Додайте ключі API у файл .env;
5. Npm запустить проект.

Для перевірки працездатності та коректної роботи редактору був складений Test Case (табл. 3.1) та Negative Test Case (табл 3.2).

Таблиця 3.1. Test Case

Номер	Action	Expected result	Test result
1	Вставити програмний код	Код вставляється правильно	Код вставився правильно
2	Редагувати програмний код	Зміни в коді зберігаються коректно	Зміни в коді зберіглися коректно
3	Виконати програмний код	Код виконується без помилок	Код виконався без помилок
4	Перевірка відображення помилок	Помилки відображаються належним чином	Помилки відобразилися належним чином
5	Скопіювати програмний код	Код успішно копіюється до буфера обміну	Код успішно скопіювався до буфера обміну
6	Вставити скопійований код з буфера обміну	Скопійований код вставляється правильно	Скопійований код вставився правильно
7	Змінити мову програмного коду	Мова програмування змінена	Мова програмування змінилася
8	Змінити оформлення	Оформлення успішно змінене	Оформлення змінилося

Таблиця 3.2. Negative Test Case

Номер	Action	Expected result	Test result
1	Вставити некоректний програмний код	Код не вставляється або відображається помилка	Код вставився та відобразилася помилка
2	Редагувати програмний код з помилками	Зміни в коді не зберігаються або відбуваються помилки	Змани в коді зберіглися та відобразилася помилка
3	Виконати некоректний програмний код	Код виконується з помилками або не виконується	Код виконався та відобразилася помилка
4	Перевірити відображення помилок	Код виконується з помилками або не виконується	Код виконався та відобразилася помилка
5	Скопіювати порожній програмний код	Нічого не копіюється або копіюється некоректний вміст	Нічого не скопіювалося
6	Вирізати порожній програмний код	Нічого не вирізається або вирізається некоректний вміст	Нічого не вирізається
7	Вставити некоректний код з буфера обміну	Код не вставляється або відображається помилка	Код вставився та відобразилася помилка
8	Виконати програмний	Код виконується з	Код виконався та

	код відмінною від вказаної мови	помилками або не виконується	відобразилася помилка

Загалом отримано простий веб-редактор, який має базові функції та буде розширювати свій функціонал, за прикладом розроблених. Базові вимоги та функції виконує задовільно. Для початку навчання мов програмування дуже зручний, не прив’язаний до локального пристрою, що дає можливість використання будь-де.

ВИСНОВКИ

У випускній кваліфікаційній роботі представлено результати теоретичних і прикладних досліджень, що полягають у розробці структури та програмної реалізації інтерфейсу WEB-редактора, що надасть можливість для внесення термінових змін у проекти, навчанню веб-розробці, користування з будь якого пристрою чи браузеру та виконання командних робіт. В результаті проведених досліджень були отримані такі **висновки**:

1. Дослідження інструментальних засобів для створення та дизайну сайтів (Wix, Wordpress, WebEditor2007, VS Code, Atom, Sublime Text та інших) дозволило ознайомитися з різноманітними засобами і їх характеристиками. Це дало змогу визначити найбільш підходящі засоби для реалізації функціоналу WEB-редактора.

2. Аналіз функціональних та нефункціональних вимог до WEB-редактора дозволив зрозуміти, які основні вимоги повинні бути задоволені цим редактором. Також було вивчено потреби та основні функції користувачів, які він повинен виконувати, а саме: підсвідчування синтаксису, функція автозаповнення, перевірка синтаксису, способи кастомізації.

3. Використання сучасних веб-технологій (React.js, TailwindCSS, HTML5, JavaScript, Judge0, RapidAPI) дозволяє створити потужний та функціональний редактор.

4. Перевірка працездатності та стабільності роботи WEB-редактора була проведена за допомогою Test Case та Negative Test Case, щоб переконатися, що програмна реалізація функціональності відповідає очікуванням та працює ефективно.

5. Веб-редактор має потенціал застосування для навчання у сфері розробці програмного забезпечення. Підтримка багатьох мов програмування розширить область користувачів-початківців, які навчаються.

6. Розробка структури та програмної реалізації інтерфейсу веб-редактора є складним процесом, що вимагає поєднання теоретичних знань, технологічних рішень та вміння взаємодіяти з користувачем.

Розроблений веб-редактор надає користувачу можливість виконувати такі дії: вибір мови програмування; вибір кастомізації в області написання коду; писати, редагувати та взаємодіяти з власним програмним кодом на багатьох мовах програмування; кастомізувати результат через окрему область; відразу бачити результат у виділеному вікні.

Функціональні можливості редактора включають: підсвітку синтаксису, підказка, виправлення помилок, підтримка багатьох мов програмування, вивід інформації про дані, автодоповнення.

За результатами тестування можна зробити висновок про те, що створений редактор функціонує стабільно та коректно.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Brown D., Communicating Design: Developing Web Site Documentation for Design and Planning // New Riders Publishing. – 2018.

2. Smith G., Web Design: The Evolution of the Digital World 1990–Today // TASCHEN. – 2017.

3. Shaffer D., Web Design Playground: HTML & CSS the Interactive Way // SitePoint. – 2018.

4. Gábor R.,Web Design: Start Here // Smashing Magazine. – 2017.

5. Castro E., HTML and CSS: Visual QuickStart Guide // Peachpit Press. - 2019.

6. Duckett J., Web Design with HTML, CSS, JavaScript, and jQuery Set – 2014.

7. Ater T., Building Progressive Web Apps: Bringing the Power of Native to the Browser 1st Edition. - 2017.

8. Документація Adobe Photoshop [Електронний ресурс] - Режим доступу до ресурсу: https://helpx.adobe.com/photoshop/user-guide.html.

9. Документація Canva [Електронний ресурс] - Режим доступу до ресурсу: https://www.canva.com/help/.

10. Документація Adobe Illustrator [Електронний ресурс] - Режим доступу до ресурсу: https://helpx.adobe.com/illustrator/user-guide.html.

11. Документація React [Електронний ресурс] - Режим доступу до ресурсу: https://reactjs.org/docs/.

12. Документація Visual Studio Code [Електронний ресурс] - Режим доступу до ресурсу: https://code.visualstudio.com/docs.

13. Документація Sublime Text [Електронний ресурс] - Режим доступу до ресурсу: https://www.sublimetext.com/docs/.

14. Документація Atom [Електронний ресурс] - Режим доступу до ресурсу: https://flight-manual.atom.io/.

15. Документація Visual Studio [Електронний ресурс] - Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/.

16. Документація Figma [Електронний ресурс] – Режим доступу до ресурсу: https://www.figma.com/resources/docs/.

17. Levy J., UX Strategy: How to Devise Innovative Digital Products that People Want 1st Edition. - 2015.

18. Wikipedia - Unified Modeling Language [Електронний ресурс] – Режим доступу до ресурсу https://uk.wikipedia.org/wiki/Unified_Modeling_Language#%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F.

19. Документація Tailwind CSS [Електронний ресурс] - Режим доступу до ресурсу: https://tailwindcss.com/docs.

20. Документація Judge0 [Електронний ресурс] - Режим доступу до ресурсу: https://ce.judge0.com/.

ДОДАТОК

ДОДАТОК А

Програмний код реалізації компоненту редактору коду

```
// CodeEditorWindow.js

import React, { useState } from "react";

import Editor from "@monaco-editor/react";

const CodeEditorWindow = ({ onChange, language, code, theme }) => {
 const [value, setValue] = useState(code || "");

 const handleEditorChange = (value) => {
  setValue(value);
  onChange("code", value);
 };

 return (
  <div className="overlay rounded-md overflow-hidden w-full h-full shadow-
4xl">
   <Editor
    height="85vh"
    width={`100%`}
    language={language || "javascript"}
    value={value}
    theme={theme}
    defaultValue="// some comment"
    onChange={handleEditorChange}
   />
  </div>
 );
};
export default CodeEditorWindow;
```

ДОДАТОК В

Програмний код реалізації Landing компоненту

```
// Landing.js

import React, { useEffect, useState } from "react";
import CodeEditorWindow from "./CodeEditorWindow";
import axios from "axios";
import { classnames } from "../utils/general";
import { languageOptions } from "../constants/languageOptions";

import { ToastContainer, toast } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";

import { defineTheme } from "../lib/defineTheme";
import useKeyPress from "../hooks/useKeyPress";
import Footer from "./Footer";
import OutputWindow from "./OutputWindow";
import CustomInput from "./CustomInput";
import OutputDetails from "./OutputDetails";
import ThemeDropdown from "./ThemeDropdown";
import LanguagesDropdown from "./LanguagesDropdown";

const javascriptDefault = `// some comment`;

const Landing = () => {
  const [code, setCode] = useState(javascriptDefault);
  const [customInput, setCustomInput] = useState("");
  const [outputDetails, setOutputDetails] = useState(null);
  const [processing, setProcessing] = useState(null);
  const [theme, setTheme] = useState("cobalt");
  const [language, setLanguage] = useState(languageOptions[0]);

  const enterPress = useKeyPress("Enter");
  const ctrlPress = useKeyPress("Control");
```

```
const onSelectChange = (sl) => {
  console.log("selected Option...", sl);
  setLanguage(sl);
};

useEffect(() => {
  if (enterPress && ctrlPress) {
    console.log("enterPress", enterPress);
    console.log("ctrlPress", ctrlPress);
    handleCompile();
  }
}, [ctrlPress, enterPress]);
const onChange = (action, data) => {
  switch (action) {
    case "code": {
      setCode(data);
      break;
    }
    default: {
      console.warn("case not handled!", action, data);
    }
  }
};
const handleCompile = () => {
  // We will come to the implementation later in the code
};

const checkStatus = async (token) => {
  // We will come to the implementation later in the code
};

function handleThemeChange(th) {
  // We will come to the implementation later in the code
}
useEffect(() => {
  defineTheme("oceanic-next").then((_) =>
```

```
    setTheme({ value: "oceanic-next", label: "Oceanic Next" })
  );
}, []);

const showSuccessToast = (msg) => {
  toast.success(msg || `Compiled Successfully!`, {
    position: "top-right",
    autoClose: 1000,
    hideProgressBar: false,
    closeOnClick: true,
    pauseOnHover: true,
    draggable: true,
    progress: undefined,
  });
};
const showErrorToast = (msg) => {
  toast.error(msg || `Something went wrong! Please try again.`, {
    position: "top-right",
    autoClose: 1000,
    hideProgressBar: false,
    closeOnClick: true,
    pauseOnHover: true,
    draggable: true,
    progress: undefined,
  });
};

return (
  <>
    <ToastContainer
      position="top-right"
      autoClose={2000}
      hideProgressBar={false}
      newestOnTop={false}
      closeOnClick
      rtl={false}
```

```
        pauseOnFocusLoss
        draggable
        pauseOnHover
      />
      <div className="h-4 w-full bg-gradient-to-r from-pink-500 via-red-500 to-
yellow-500"></div>
      <div className="flex flex-row">
        <div className="px-4 py-2">
          <LanguagesDropdown onSelectChange={onSelectChange} />
        </div>
        <div className="px-4 py-2">
          <ThemeDropdown handleThemeChange={handleThemeChange}
theme={theme} />
        </div>
      </div>
      <div className="flex flex-row space-x-4 items-start px-4 py-4">
        <div className="flex flex-col w-full h-full justify-start items-end">
          <CodeEditorWindow
            code={code}
            onChange={onChange}
            language={language?.value}
            theme={theme.value}
          />
        </div>

        <div className="right-container flex flex-shrink-0 w-[30%] flex-col">
          <OutputWindow outputDetails={outputDetails} />
          <div className="flex flex-col items-end">
            <CustomInput
              customInput={customInput}
              setCustomInput={setCustomInput}
            />
            <button
              onClick={handleCompile}
              disabled={!code}
              className={classnames(
```

```
            "mt-4 border-2 border-black z-10 rounded-md shadow-
[5px_5px_0px_0px_rgba(0,0,0)] px-4 py-2 hover:shadow transition duration-200
bg-white flex-shrink-0",
            !code ? "opacity-50" : ""
          )}
        >
          {processing ? "Processing..." : "Compile and Execute"}
        </button>
      </div>
      {outputDetails && <OutputDetails outputDetails={outputDetails} />}
    </div>
  </div>
  <Footer />
 </>
 );
};
export default Landing;
```

ДОДАТОК С

Програмний код реалізації LanguageDropdown компоненту

```
// LanguageDropdown.js

import React from "react";
import Select from "react-select";
import { customStyles } from "../constants/customStyles";
import { languageOptions } from "../constants/languageOptions";

const LanguagesDropdown = ({ onSelectChange }) => {
 return (
  <Select
   placeholder={`Filter By Category`}
   options={languageOptions}
   styles={customStyles}
   defaultValue={languageOptions[0]}
   onChange={(selectedOption) => onSelectChange(selectedOption)}
  />
 );
};

export default LanguagesDropdown;
```

ДОДАТОК D

Програмний код реалізації ThemeDropdown компоненту

```
// ThemeDropdown.js

import React from "react";
import Select from "react-select";
import monacoThemes from "monaco-themes/themes/themelist";
import { customStyles } from "../constants/customStyles";

const ThemeDropdown = ({ handleThemeChange, theme }) => {
 return (
  <Select
   placeholder={`Select Theme`}
   // options={languageOptions}
   options={Object.entries(monacoThemes).map(([themeId, themeName]) => ({
    label: themeName,
    value: themeId,
    key: themeId,
   }))}
   value={theme}
   styles={customStyles}
   onChange={handleThemeChange}
  />
 );
};

export default ThemeDropdown;
```

ДОДАТОК D

Програмний код реалізації Output компоненту

```
import React from "react";

const OutputWindow = ({ outputDetails }) => {
 const getOutput = () => {
  let statusId = outputDetails?.status?.id;

  if (statusId === 6) {
   // compilation error
   return (
    <pre className="px-2 py-1 font-normal text-xs text-red-500">
     {atob(outputDetails?.compile_output)}
    </pre>
   );
  } else if (statusId === 3) {
   return (
    <pre className="px-2 py-1 font-normal text-xs text-green-500">
     {atob(outputDetails.stdout) !== null
      ? `${atob(outputDetails.stdout)}`
      : null}
    </pre>
   );
  } else if (statusId === 5) {
   return (
    <pre className="px-2 py-1 font-normal text-xs text-red-500">
     {`Time Limit Exceeded`}
    </pre>
   );
  } else {
   return (
    <pre className="px-2 py-1 font-normal text-xs text-red-500">
     {atob(outputDetails?.stderr)}
    </pre>
```

```
    );
  }
 };
 return (
  <>
   <h1 className="font-bold text-xl bg-clip-text text-transparent bg-gradient-to-r
from-slate-900 to-slate-700 mb-2">
    Output
   </h1>
   <div className="w-full h-56 bg-[#1e293b] rounded-md text-white font-normal
text-sm overflow-y-auto">
    {outputDetails ? <>{getOutput()}</> : null}
   </div>
  </>
 );
};

export default OutputWindow;
```

ДОДАТОК Е

Програмний код реалізації KeyBoard Events

```
// useKeyPress.js

import React, { useState } from "react";

const useKeyPress = function (targetKey) {
 const [keyPressed, setKeyPressed] = useState(false);

 function downHandler({ key }) {
  if (key === targetKey) {
   setKeyPressed(true);
  }
 }

 const upHandler = ({ key }) => {
  if (key === targetKey) {
   setKeyPressed(false);
  }
 };

 React.useEffect(() => {
  document.addEventListener("keydown", downHandler);
  document.addEventListener("keyup", upHandler);

  return () => {
   document.removeEventListener("keydown", downHandler);
   document.removeEventListener("keyup", upHandler);
  };
 });
 return keyPressed;
};

export default useKeyPress;
```

```
// Landing.js

...
...
...
const Landing = () => {
  ...
  ...
    const enterPress = useKeyPress("Enter");
    const ctrlPress = useKeyPress("Control");
  ...
  ...
}
```