Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-системи для вивчення мови програмування JavaScript

Виконав(ла): студент(ка) 4 курсу, групи СП-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

		Гавриленко А. В.
	(підпис)	(прізвище та ініціали)
Керівник		Петрик М. Р.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Стоянов Ю. М.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Петрик М. Р.
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

Тернопіль
2025

АНОТАЦІЯ

Кваліфікаційна робота обсягом 46 сторінки містить 8 рисунків, 2 додатка, 38 джерел за списком літератури. Об’єктом дослідження є веб-сайти як засоби цифрової взаємодії.

Ключові слова: JavaScript, веб-система, контент-аналіз, SEO, SPA, MongoDB, Node.js, програмне забезпечення, веб-інтерфейс, авторизація.

Мета роботи – створення веб-системи для аналізу та оптимізації веб-контенту із застосуванням мови програмування JavaScript.

У процесі розробки використано метод контент-аналізу, структурно-семантичний підхід, модульне та функціональне тестування. Застосовано сучасні технології: Node.js, MongoDB, HTML, CSS, SPA-архітектуру, REST API, Git. Розроблена система дозволяє перевіряти веб-сайти на унікальність, наявність технічних помилок та відповідність SEO-вимогам. Вона реалізована як односторінковий додаток з функціоналом реєстрації, автентифікації, збереження історії перевірок та генерації звітів.

Система забезпечує інтерактивну взаємодію з користувачем, має сучасний адаптивний інтерфейс і може бути впроваджена на VDS-сервері з використанням GitHub для віддаленого керування кодом. Робота взаємопов’язана з галузями веб-розробки, цифрового маркетингу та інформаційних технологій. Результати можуть бути використані в освітніх платформах, SEO-компаніях і проектах контент-менеджменту.

Розробка сприяє автоматизації перевірки контенту, підвищує ефективність сайтів та знижує витрати на SEO-аналіз. Практична значимість полягає у створенні інструменту, що відповідає сучасним вимогам цифрового ринку. Подальший розвиток може включати інтеграцію з аналітичними сервісами та мобільну адаптацію.

ABSTRACT

The qualification paper consists of 46 pages and includes 8 figures, 2 appendixes, and 38 sources listed in the bibliography. The object of the research is websites as tools of digital interaction.

Keywords: JavaScript, web system, content analysis, SEO, SPA, MongoDB, Node.js, software, web interface, authentication.

The aim of the work is to develop a web system for analyzing and optimizing web content using the JavaScript programming language.

The development process involved the use of content analysis methodology, a structural-semantic approach, as well as modular and functional testing. Modern technologies were applied, such as Node.js, MongoDB, HTML, CSS, SPA architecture, REST API, and Git. The developed system allows checking websites for uniqueness, technical errors, and compliance with SEO requirements. It is implemented as a single-page application with functionality for user registration, authentication, saving the history of checks, and generating reports.

The system provides interactive user engagement, features a modern adaptive interface, and can be deployed on a VDS server using GitHub for remote code management. The work is closely related to the fields of web development, digital marketing, and information technology. The results can be applied in educational platforms, SEO companies, and content management projects.

The development promotes automation of content checking, improves website efficiency, and reduces the cost of SEO analysis. Its practical significance lies in creating a tool that meets the modern demands of the digital market. Further development may include integration with analytical services and mobile adaptation.

ЗМІСТ

ВСТУП

У сучасному цифровому середовищі поняття веб-сайту займає ключову позицію та визначається як сукупність програмного й технічного забезпечення з унікальною адресою в мережі Інтернет. Цей ресурс містить інформаційні матеріали, якими розпоряджається певний суб'єкт, та надає фізичним і юридичним особам доступ до них, а також до інших онлайн-послуг.

Зазвичай веб-сайти розміщуються на одному або декількох серверах. Основними рисами сайту виступають наступні характеристики:

1. Щодо інформаційного змісту:
- наявність концепції;
- логічна структура подачі (дискурсивність);
- здатність переконувати користувача;
- інтерактивність взаємодії;
- мовна дія та комунікативний ефект.
2. З точки зору практичної мети функціонування: прагматизм.
3. З організаційного боку: структурованість і системність.
4. З погляду унікальності: відмінність серед інших веб-ресурсів.
5. З урахуванням динаміки сучасного світу: актуальність та технічна оптимізація.

У цьому дослідженні увага зосереджена саме на тих характеристиках, що безпосередньо пов’язані з послугами веб-розробників. Зокрема, йдеться про засоби автоматизованого аналізу контенту, перевірки на унікальність, а також технічної оптимізації для покращення функціонування сайту та досягнення його практичних цілей.

Особливу увагу приділено поняттю SEO (скорочено від Search Engine Optimization) – комплексу дій, спрямованих на покращення позицій сайту у результатах пошуку за запитами користувачів. Цей процес є складовою частиною ширшого підходу – SEM (Search Engine Marketing), що включає розподіл інтернет-

трафіку на основі релевантності контенту та підвищення його видимості серед конкурентів. Оптимізація неможлива без урахування структури веб-сайту як інтертекстуального явища. Сайт функціонує у формі взаємопов'язаних інформаційних вузлів, які перенаправляють користувача на додаткові джерела, як усередині, так і поза межами ресурсу. Така мережа вимагає узгодженості контенту, новизни, лаконічності та відповідності сучасним потребам користувача.

Ключову роль у підтримці ефективності відіграють оновлення: як повного наповнення ресурсу, так і окремих інформаційних сегментів. Особливо важливими є ключові слова та гіперпосилання, які забезпечують навігацію та доступ до додаткових матеріалів. Їх актуальність та релевантність мають підтримуватись постійно. Отже, проблема оптимізації веб-сайтів є однією з найактуальніших, оскільки вона безпосередньо впливає на присутність ресурсу в онлайн-просторі. Автоматизовані системи аналізу контенту дозволяють виявити слабкі місця, надати рекомендації щодо оновлення структури та підвищити ефективність взаємодії з користувачем.

Цей проєкт присвячено розробці програмних інструментів для аналізу ефективності роботи сайтів, зокрема для покращення їх SEO- та SEM-параметрів. Було вивчено існуючі методи перевірки контенту та на основі отриманих результатів сформульовано практичні рішення для оптимізації. Розробка реалізована з використанням JavaScript – популярної мови програмування для веб-середовища. Фронтенд частину створено на основі JavaScript, HTML та CSS, що забезпечує сучасний інтерфейс користувача. Серверна частина побудована на платформі Node.js, яка дозволяє ефективно працювати з даними та забезпечує масштабованість проєкту.

Проєкт використовує нереляційну базу даних MongoDB для зберігання та подальшого аналізу даних. MongoDB – оптимальний вибір для платформи Node.js завдяки тісній інтеграції з екосистемою JavaScript (особливо через бібліотеку Mongoose), простоті використання та високій структурованості даних.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ПЕРЕДУМОВИ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.

1.1. Вхідні поняття дослідження та визначення предметної області.

У сучасних умовах розвитку цифрових технологій та поглиблення глобалізаційних і інтеграційних процесів в економіці світу, значно зростає роль інформації. Без грамотно вибудованого інформаційного середовища ефективна економічна взаємодія стає майже неможливою. Саме у цій площині перетинаються сфери економіки та інформаційних технологій, формуючи явище інтернет-маркетингу — діяльності, що базується на використанні інтернет-ресурсів для задоволення потреб споживачів.

Ця діяльність реалізується через сайти, де взаємодія між бізнесом і клієнтом здійснюється за допомогою спеціально створеного інформаційного контенту. Контент — це сукупність цифрових матеріалів, розміщених у мережевих інформаційних системах[2]. Він формується на основі стратегій, що покликані надати ресурсу відмінності та індивідуальності в інформаційному середовищі. Саме унікальність — ключова риса якісного контенту, яка дозволяє сайту бути конкурентоспроможним і потрапити до топових позицій у результатах пошуку. Контент має бути оригінальним і не копіювати інші ресурси, оскільки дублювання веде до зниження ефективності сайту в умовах високої конкуренції. Не менш важливим аспектом є актуальність інформації, тобто її відповідність сучасним запитам користувачів. Актуальність забезпечується через постійне оновлення матеріалів та SEO-оптимізацію, яка включає оновлення ключових слів, заголовків, сніпетів тощо. Ці елементи відіграють вирішальну роль у видимості сайту в пошукових системах.

Досягнення унікальності й актуальності створює основу для реалізації інших важливих властивостей контенту, зокрема:

- Цінність — контент має надавати користувачам важливу інформацію, яку складно знайти в інших джерелах.

- Доступність — інформація має бути подана у зручному форматі, у відповідному місці й у потрібний момент.
- Мотиваційність — матеріали повинні спонукати користувача до дій: переходів, підписок, покупок тощо.

Цінний контент — це той, що не тільки відповідає очікуванням користувача, але й сприяє досягненню цілей бізнесу. Він має бути релевантним, корисним і викликати бажання взаємодіяти. Саме така інформація сприяє створенню інтерактивного середовища, де між користувачем і платформою виникає продуктивна комунікація. Доступність контенту забезпечується тим, що він вчасно з'являється в результатах пошуку, у відповідних соціальних мережах або рекламних кампаніях. Мотиваційна складова — це влучні заголовки, зображення, кнопки дій, які викликають бажання «клікнути» чи здійснити інші цільові дії. Кожен інформаційний елемент сайту має виконувати певну функцію: збільшення підписок, залучення нових користувачів, продаж, перенаправлення на інші сторінки. Таким чином, контент повинен бути продуманий, структурований і мати чітку мету [2].

Оцінити якість контенту допомагають такі критерії:

- Оригінальність — перевірка на відсутність плагіату через спеціалізовані сервіси.
- Мовна грамотність — правильність орфографії та пунктуації завдяки автоматичним редакторам.
- Фактологічна точність — перевірка достовірності даних.
- Форматування — структурованість і читабельність тексту.
- SEO-оптимізація — природне включення ключових слів у назви, підзаголовки та основний текст.
- Коректність зовнішніх посилань — перевірка, щоб всі лінки працювали й вели до потрібних ресурсів.

Усі ці заходи мають реалізовуватись не тільки для новоствореного, а й для наявного контенту — це дозволить підтримувати якість ресурсу на високому рівні. Оптимально — здійснювати перевірки ще до публікації, щоб уникнути втрати

довіри з боку пошукових систем і користувачів. Щоб сайт добре працював і займав високі позиції в Google, потрібно регулярно перевіряти якість контенту. Низькоякісний текст може відштовхнути читачів і знизити рейтинг у пошуку. Існують спеціальні онлайн-сервіси, які допомагають виявити слабкі місця у текстах та покращити їх.

Основні ознаки, що сигналізують про проблеми з контентом:

- Зовнішні посилання — потрібно переконатися, що всі посилання на інші сайти або сторінки працюють.
- Вхідні посилання — слідкуйте, щоб сайти, які посилаються на вас, були надійними. Посилання з підозрілих джерел можуть зашкодити.
- Показник відмов — якщо люди швидко залишають сторінку, ймовірно, вони не знайшли потрібної інформації.
- Час на сторінці — короткий час свідчить, що текст не зацікавив відвідувача.
- Є трафік, але немає результатів (реєстрацій, покупок) — отже, контент не переконує діяти.
- Багато переглядів, але мало кліків — назви й опис товарів або статей не приваблюють.
- Користувачі виходять у процесі реєстрації або покупки — можливо, щось неясно або текст не викликає довіри.

Ці показники не завжди означають, що текст поганий. Але вони дають підказку: щось варто змінити. І найкраще — використовувати онлайн-інструменти, які допомагають все це швидко перевірити.

Корисні сервіси для перевірки:

- SEO Review Tools — показує, наскільки добре текст підготовлений для пошукових систем.
- Contentseochecker.com — визначає, чи відповідає текст темі і наскільки легко його читати.
- Yoast SEO (для WordPress) — підказує, що в тексті добре, а що краще змінити [2].

- Hemingway Editor — робить текст простішим і зрозумілішим.
- Apollo Insights — аналізує поведінку користувачів і показує, як контент впливає на відвідувачів.

Якщо контент не працює — що робити?

1. Переконайтесь, що текст розміщений у правильному місці і його бачить потрібна аудиторія.
2. Виправте технічні або візуальні проблеми, які можуть заважати читати або діяти.
3. Перегляньте сам зміст: можливо, текст варто переписати, зробити більш цікавим і корисним.

Що важливо знати:

Google оцінює сайти за принципами E-A-T:

- Експертність
- Авторитетність
- Надійність

Це означає, що текст має бути написаний фахівцем, містити перевірену інформацію і викликати довіру.

Щоб контент добре сприймався Google і мав шанси на високі позиції в пошуку, важливо, щоб його створювали люди, які реально розуміються на темі. Це можуть бути як професіонали з освітою, так і ті, хто має реальний досвід, наприклад, вирощував щось сам, подорожував, чи просто багато років займається певною справою. Головне, щоб людина справді зналася на темі, про яку пише. Google також звертає увагу на те, наскільки автор, сам текст і сайт виглядають авторитетними й правдивими. Якщо в людини є профіль, її ім’я згадується на інших сайтах або вона веде YouTube-канал, це тільки підсилює довіру. Важливо, щоб текст був добре структурований, логічний, без каші з абзаців. Заголовки повинні відповідати змісту, в них мають бути ключові слова. Те саме стосується HTML-розмітки — сайт має бути правильно зібраний, із відповідними тегами і стилями, щоб усе було зручно і для людей, і для пошукових систем. Якщо все зробити грамотно, контент буде виглядати якісним, сайт вигідно виділятиметься серед

конкурентів, а Google з більшою ймовірністю покаже його у топі.

1.2. Метод контент-аналізу як комплексний засіб перевірки веб-сайтів

Якість вебсайту визначається не лише його наповненням, але й тим, наскільки ефективно організована структура, наскільки зміст релевантний, корисний і відповідає очікуванням користувачів. Комплексне оцінювання цифрового контенту передбачає аналіз як з точки зору інформативності, так і з погляду структурно-семантичної цілісності, з урахуванням вимог прагматичності. На підставі такого аналізу можна сформувати конкретні рекомендації для покращення унікальності матеріалу та оптимізації вмісту сайту. Це, своєю чергою, сприяє підвищенню видимості ресурсу у пошукових системах і його конкурентоспроможності на ринку.

Одним з найпоширеніших і дієвих інструментів для такої оцінки є контент-аналіз — метод, що дозволяє системно перевірити інформацію на глибину, точність, актуальність та відповідність низці критеріїв. Цей підхід бере свій початок у працях Б. Берельсона та Г. Ласуела, а сучасну методологічну базу розвинули такі науковці як Р. Віхалемм, Ю. Вооглайд, М. Лаурістін, а також українські дослідники І. Секунова, А. Согорін, Г. Квіт, К. Шіковець та інші.

У розумінні Г. Квіта та К. Шіковець, контент-аналіз — це структурована обробка текстової інформації з кількісною та якісною інтерпретацією її форми й змісту. Це дозволяє системно оцінити ступінь відповідності контенту низці критеріїв: новизні, змістовній оригінальності, корисності для аудиторії, логіці подачі матеріалу, відповідності формату, релевантності ключових слів і коректності мовного оформлення.

Більшість дослідників поділяють контент-аналіз на два основні типи:

- кількісний — з акцентом на статистичні показники контенту;
- якісний — з фокусом на глибинну семантичну й тематичну

інтерпретацію.

У практиці найчастіше застосовують комбінований підхід, де кількісні параметри (повторюваність ключових слів, частотність тем, частка релевантних фрагментів) доповнюють якісним аналізом, що враховує смислову структуру тексту, цінність поданої інформації, а також її логічну та емоційну складову.

Контент-аналіз також дозволяє формувати аналітичні гіпотези про фактори, що можуть впливати на побудову інформаційного посилу, очікуваний ефект серед цільової аудиторії, а також про місцезнаходження або потреби потенційної клієнтури. У процесі дослідження можуть застосовуватись дві аналітичні парадигми:

- когнітивно-дискурсивна — орієнтується на смислові категорії та концептуальні сітки;
- структурно-семантична — базується на аналізі зв’язків між інформаційними одиницями в контенті.

У межах когнітивного підходу формується система ключових понять і значень, які структурно організовують сприйняття змісту. Водночас, у практиці контент-аналіз найчастіше проводиться саме у семантичному вимірі, з акцентом на виділення “смислового ядра” сайту, яке відображає основну інформаційну суть ресурсу [3].

Незалежно від обраної методології, обов’язковим елементом є оцінка прагматичних характеристик сайту: чи є контент доступним, логічно побудованим, цінним для користувача, і як він представлений з точки зору дизайну, шрифтів, маркувань та зручності читання. Для автоматизації контент-аналізу можна застосовувати такі спеціалізовані інструменти, як IBM Content Analytics, TextAnalyst, Галактика Zoom та інші. Окремо варто зазначити, що результати контент-аналізу можуть бути використані як основа для проведення SWOT-аналізу (визначення сильних і слабких сторін сайту, можливостей і загроз його позиціонування). Це дозволяє розробити ефективні рекомендації для оптимізації структури, семантики та змісту ресурсу, що сприятиме його кращій видимості в пошукових системах та підвищенню лояльності з боку цільової аудиторії. [3].

1.3. Обґрунтування технологій та архітектури

Необхідність у серверній логіці зумовлена кількома ключовими факторами. По-перше, процес обробки та аналізу контенту потребує значних обчислювальних ресурсів, тому його доцільно виконувати на серверному рівні, що дозволяє знизити вимоги до клієнтських пристроїв і забезпечити кращу продуктивність. По-друге, реалізація функції збереження результатів аналізу для подальшого доступу потребує використання бази даних — компоненту, який традиційно розміщується на сервері. І нарешті, для забезпечення безпеки даних користувачів необхідна система аутентифікації, яка перевіряє логін і пароль при доступі до персонального кабінету, що також реалізується на стороні сервера.

Для написання серверної логіки використано JavaScript у середовищі Node.js. Цей вибір обумовлений високою швидкістю обробки запитів, надійністю, а також активною підтримкою з боку глобальної спільноти розробників, що забезпечує стабільність та постійне вдосконалення платформи. Використання асинхронного підходу до програмування через async/await і Promise API дає змогу ефективно обробляти одночасні запити без блокування виконання коду. Для зберігання даних у проєкті обрано MongoDB — гнучку та масштабовану нереляційну систему керування базами даних. Вона забезпечує зручну роботу з JSON-подібними структурами, підтримує шардінг (розподіл даних між кількома вузлами), агрегацію даних, а також зберігання великих обсягів інформації. Враховуючи широке використання MongoDB разом із Node.js, їхня сумісність значно полегшує розробку — зокрема завдяки бібліотеці Mongoose, яка надає зручний інтерфейс для взаємодії з БД. Клієнтська частина розроблена у вигляді SPA-додатку (Single Page Application). Це дозволяє працювати з програмою без встановлення окремого ПЗ — потрібен лише сучасний веб-браузер. Після початкового завантаження сторінки користувач може перемикатись між розділами без перезавантаження сайту — завдяки динамічному маршрутизатору, реалізованому на JavaScript. Вся логіка працює у браузері, а зміна вмісту сторінки відбувається швидко й плавно [5].

Зв’язок між клієнтом і сервером здійснюється через REST API. Клієнт надсилає запити серверу у фоновому режимі (використовуючи XMLHttpRequest або fetch), після чого отримує відповідь у форматі JSON. Цей формат легко перетворюється у JavaScript-об’єкти на фронтенді та бекенді, що спрощує обробку та обмін даними.

Рисунок 1.3.1 – Принцип роботи архітекти RESTAPI

Програмне забезпечення логічно розділене на три функціональні модулі:

1. Модуль управління та верифікації користувацьких даних: Включає функції реєстрації та авторизації.
2. Модуль аналізу веб-контенту: Виконує аналіз вмісту веб-сторінок.
3. Модуль генерації звітів: Автоматично формує результати аналізу [5].

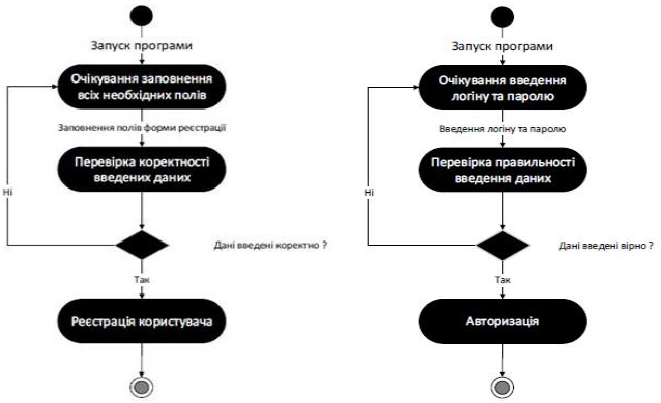

Рисунок 1.3.2 – Функціонування модуля збереження та валідації персональних даних.

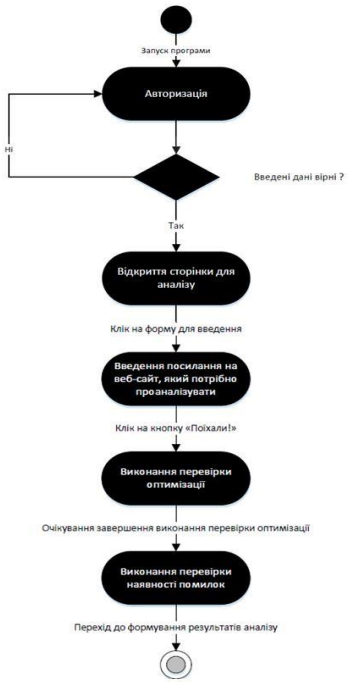

Рисунок 1.3.3 – Функціонування модуля проведення контент-аналізу веб-сторінки

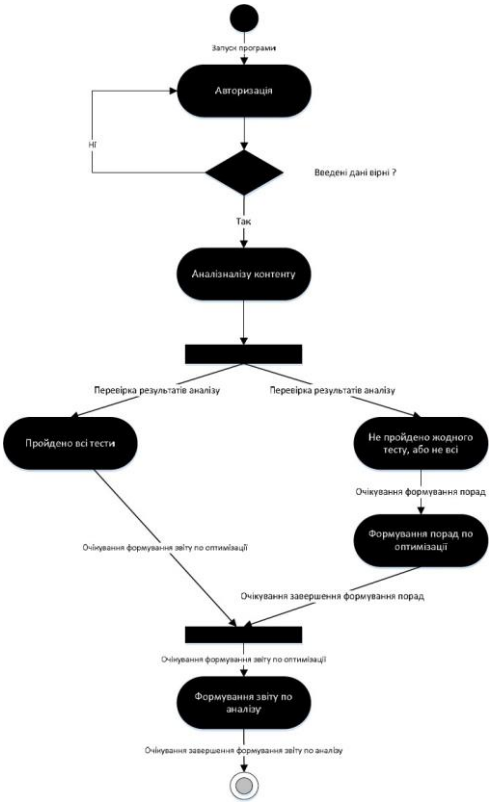

Рисунок 1.3.4. – Функціонування модуля автоматичної генерації підсумків аналізу

РОЗДІЛ 2. ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ

2.1. Застосовані технології розробки

Проєкт реалізовано з використанням повного стеку JavaScript — сучасної об’єктно-орієнтованої мови програмування з динамічною типізацією, яка є ключовою у сфері веб-розробки. Вона забезпечує гнучку та ефективну побудову програмних рішень, що легко масштабуються та підтримуються з часом.

Фронтенд-частина (інтерфейс користувача) створена на базі стандартного набору вебтехнологій:

- JavaScript — відповідає за інтерактивність та логіку поведінки елементів сторінки;
- HTML (HyperText Markup Language) — структурна мова розмітки, яка формує каркас сайту;
- CSS (Cascading Style Sheets) — інструмент для оформлення та позиціонування елементів, що забезпечує візуальну привабливість інтерфейсу.

Серверна частина (бекенд) побудована на Node.js — платформі з відкритим вихідним кодом, що дозволяє реалізовувати логіку обробки даних на тій же мові, що й фронтенд, — JavaScript. Це спрощує розробку та покращує узгодженість між частинами системи. Для зберігання та обробки даних використовується документно-орієнтована база даних MongoDB, яка добре поєднується з Node.js та підтримується через ORM-бібліотеку Mongoose. Ця база даних не вимагає жорсткої схеми та забезпечує гнучку структуру збереження інформації, що ідеально підходить для вебпроєктів.

Для серверної емуляції роботи браузера застосовується бібліотека Phantom.js — інструмент, що дозволяє відкривати сторінки та взаємодіяти з ними програмно, без необхідності графічного інтерфейсу. Це корисно для тестування, сканування або автоматичного збору інформації з вебресурсів. Для валідації HTML та CSS-коду сторінок було використано онлайн-сервіс W3C Markup Validation Service, який дозволяє проаналізувати якість і відповідність коду вебстандартам, виявити

помилки та отримати рекомендації щодо їх виправлення.

Елементи графічного дизайну інтерфейсу створено у Adobe Photoshop CC 2019 (версія 20.0.4). Зовнішній вигляд системи має важливе значення, адже саме з ним користувач взаємодіє щодня — зручність і привабливість напряму впливають на успіх програмного продукту [4].

Система також виконує перевірку важливих параметрів вебсторінок, зокрема:

- наявності favicon (іконки сайту у вкладці браузера),
- підтримки OpenGraph-протоколу,
- правильного використання тегів title, h1-h6, meta, author, а також атрибуту charset,
- наявності ключових слів (keywords) та заголовків,
- а також відповідності розмітки стандартам шляхом аналізу візуального відображення сайту у вікні браузера (viewport-аналіз).

Для прикладу розглянемо одну з функцій, які використовуються для перевірки:

```
QualitySEO.prototype.validateFavicon = function (result) {
  return this.page.evaluate((result) => {
    let items = 0, mark = 0, tags = document.getElementsByTagName("link");
    content = "Please add favicon to the page for better user experience.";
    for (let i = 0; i < tags.length; i++) {
      const rel = tags[i].getAttribute("rel").split(" ");
      if (rel && rel.includes("icon")) {
        content = ++items ? tags[i].getAttribute("href") : content;
        break;
      }
    }
    if (items && content) {
      mark = 1;
      message = "Good.";
    } else {
      mark = 0;
      message = "There is no favicon.";
    }
    result.favicon = { mark, content, message, items };
    return result;
  }, result);
};
```

Лістинг 2.1.1 – Функція для перевірки favicon на сторінці за допомогою JavaScript

Функція validateFavicon призначена для автоматичного визначення наявності зображення favicon на веб-ресурсі. Її робота ґрунтується на аналізі всіх HTML-елементів link, виявлених на сторінці. Якщо атрибут rel одного з таких елементів містить слово «icon», це вказує на те, що сайт використовує favicon. Така перевірка є важливою при створенні інструментів для сканування або валідації сайтів, оскільки favicon — це один із ключових елементів візуальної ідентифікації ресурсу.

Окрім програмної реалізації, важливим аспектом є дотримання технічних вимог як на стороні сервера, так і на стороні клієнта. Це дозволяє забезпечити коректну, безперебійну та швидку роботу програмного забезпечення, а також сприяє формуванню позитивного досвіду користувача.

Для серверної частини бажаними характеристиками є наявність потужного процесора, такого як AMD Ryzen 9 5900X із частотою 3.7 ГГц, оперативної пам’яті обсягом не менше 32 ГБ стандарту DDR3 або DDR4, а також накопичувача типу SSD або HDD з обсягом від 512 ГБ. Важливою є й швидкість мережевого підключення — не менше 1 Гбіт/с, що особливо актуально для високонавантажених веб-систем. Клієнтська частина передбачає використання стаціонарного комп’ютера або ноутбука з операційною системою Windows, Ubuntu чи MacOS, а також можливе використання смартфона або планшета з актуальними версіями Android або iOS. Для повноцінної роботи рекомендована швидкість інтернет-з’єднання повинна бути не нижчою за 5 Мбіт/с. Обсяг оперативної пам’яті має становити щонайменше 8 ГБ для настільних пристроїв та 3 ГБ для мобільних. Браузер користувача також повинен бути сучасним — це можуть бути Google Chrome, Mozilla Firefox, Safari або Opera в останніх версіях. Розроблена програмна система реалізована у форматі односторінкового веб-додатку (SPA – *Single-Page Application*). Це означає, що основний код програми (HTML, CSS і JavaScript) завантажується лише один раз — під час першого відкриття сторінки. Надалі взаємодія з інтерфейсом здійснюється без повного перезавантаження сторінки завдяки програмному маршрутизатору, реалізованому за допомогою JavaScript. Динамічне завантаження вмісту та обмін даними з сервером відбувається у фоновому режимі через нативний об’єкт XMLHttpRequest, що дозволяє

забезпечити асинхронну комунікацію між клієнтом і сервером [4].

Такий підхід до реалізації клієнт-серверної взаємодії дозволяє досягти високої швидкодії, зменшити навантаження на мережу та підвищити зручність використання веб-застосунку.

2.2. Програмна реалізація бази даних.

Для реалізації даного програмного рішення було обрано MongoDB — сучасну, кросплатформенну базу даних з відкритим вихідним кодом, яка належить до категорії нереляційних (NoSQL) систем. На відміну від класичних реляційних баз, вона не використовує табличну структуру. Натомість її внутрішня організація побудована на основі документів у форматі, схожому на JSON, з динамічними схемами даних, які мають формат BSON (Binary JSON) — бінарного подання об’єктів JavaScript. Завдяки такій структурі MongoDB значно краще справляється з обробкою гнучких, різнотипних даних та забезпечує високу швидкодію при масштабуванні, що робить її придатною як для невеликих проєктів, так і для масштабних розподілених систем з великою кількістю одночасних підключень. MongoDB була створена у жовтні 2007 року компанією «10gen» (пізніше перейменованою на MongoDB Inc.). Спочатку система розроблялася як складова частина хмарної платформи (PaaS), однак уже в 2009 році її вихідний код було відкрито. Відтоді MongoDB стала однією з провідних NoSQL БД у світі, яку використовують численні глобальні сервіси, серед яких — eBay, The New York Times, SourceForge тощо [4].

MongoDB поширюється за умовами вільної ліцензії Affero GNU GPL, тоді як драйвери бази доступні згідно з умовами ліцензії Apache. Також існують варіанти комерційного використання з підтримкою від розробника.

Основні функціональні можливості MongoDB:

- Ad-hoc запити: підтримка фільтрації даних за значеннями полів,

регулярними виразами, діапазонами тощо;

- Індексація: можна індексувати будь-яке поле в документі для підвищення швидкості пошуку;
- Реплікація: підтримка створення декількох синхронізованих копій бази;
- Шардинг (розподіл даних): дозволяє масштабувати базу горизонтально, розміщуючи дані на різних серверах за допомогою ключів шардингу;
- Агрегація: підтримка операцій масової обробки даних за допомогою Map і Reduce;
- Зберігання файлів: MongoDB може виконувати функцію файлової системи з усіма перевагами — включно з розподіленим зберіганням даних.

У цьому проєкті MongoDB було інтегровано в середовище Node.js із застосуванням ORM-бібліотеки Mongoose, яка полегшує роботу зі схемами, валідацією та взаємодією з колекціями бази.

Основні структури даних (схеми):

- User – інформація про користувача;
- Analysis – дані, пов’язані з аналітикою [4].

```
const mongoose = require("mongoose"), crypto = require("crypto"),
Schema = mongoose.Schema, validatePresenceOf = (val) => val && val.length;
const User = new Schema({
    email: {
      type: String,
      validate: [validatePresenceOf, "an email is required"],
      index: { unique: true },
    },
    name: String, surname: String, hashed_password: String, salt: String
  },
  { collection: "users" }
);
User.virtual("password").set(function (password) {
    this._password = password;
    this.salt = this.makeSalt();
    this.hashed_password = this.encryptPassword(password);
  }).get(function () { return this._password; });
User.method("authenticate", function (plainText) {
  return this.encryptPassword(plainText) === this.hashed_password;
});
User.method("makeSalt", () => String(Math.round(Date.now() * Math.random())));
User.method("encryptPassword", (password) => {
  if (!password) return "";
  try {
    return crypto.createHmac("sha256", this.salt).update(password).digest("hex");
  } catch (err) { return ""; }
});
module.exports = mongoose.model("User", User);
```

Лістинг 2.2.1 – Створення схеми користувача в Mongoose з валідацією email і хешуванням пароля при збереженні

Система захищає дані користувачів, надаючи до них доступ лише після успішної перевірки автентичності.

Наведений нижче фрагмент коду демонструє механізм авторизації на сервері:

```
const mongoose = require("mongoose"), Schema = mongoose.Schema;
const Analysis = new Schema({
    info: {},
    user_id: String,
    date: { type: Date, default: Date.now },
  },
  { collection: "analyzes" }
);
module.exports = mongoose.model("Analyzes", Analyzes);
```

Лістинг 2.2.2 – Mongoose-схема для збереження загальної інформації, ідентифікатора користувача та дати створення запису

Сесія — це проміжок часу, протягом якого користувач залишається авторизованим у системі. Цей процес підтримується як на клієнтській, так і на серверній стороні. Після того як користувач вводить свій логін і пароль та натискає кнопку входу, на сервер надсилається XHR-запит (XmlHttpRequest), що містить облікові дані у форматі JSON.

Сервер, отримавши цей запит, перевіряє правильність логіну та паролю, звіряючи їх із записами в базі даних. Якщо обліковий запис не знайдено, сервер повертає статус-код 401 Unauthorized, який означає відмову в доступі. У разі успішної автентифікації сервер відповідає статусом 200 OK і надсилає у відповідь два токени:

- Access-token (токен доступу) – використовується для підтвердження авторизації під час наступних запитів
- Refresh-token (токен оновлення) – застосовується для продовження дії сесії без повторного входу

Ці токени зазвичай зберігаються у LocalStorage браузера, що дозволяє клієнту залишатися авторизованим до закінчення терміну дії сесії. Наприклад, якщо на сервері сесія обмежена 20 хвилинами, після цього періоду бездіяльності

користувача система автоматично його розлогінює. Якщо ж refresh-токен регулярно оновлюється — сесія може бути продовжена безперервно.

Чому авторизація є критично важливою?

Механізм авторизації — це один з основних елементів захисту будь-якої сучасної програмної системи. Він гарантує, що доступ до персональних або конфіденційних даних мають лише уповноважені користувачі. Це особливо актуально для сервісів, які працюють з аналітикою вебконтенту, наприклад, у сфері SEO.

Адже дані, пов’язані з оцінкою технічного чи маркетингового стану вебресурсу, можуть бути висококонкурентними. Якщо така інформація потрапить до рук сторонніх осіб (наприклад, конкурентів), вони зможуть використати її для недобросовісної оптимізації власних сайтів, отримавши несправедливу перевагу.

Тому в системах SEO-аналізу захист інформації від несанкціонованого доступу має ключове значення. Неналежна реалізація безпеки може призвести не лише до витоку даних, а й до втрати довіри користувачів, репутаційних збитків або навіть провалу всього проєкту [6].

2.3. Віддалене розміщення коду програми

Для того щоб програмна система була постійно доступна через інтернет і могла функціонувати безперебійно, її слід розгортати на віртуальному сервері типу VPS або VDS (від англ. Virtual Private Server / Virtual Dedicated Server). Звичайні тарифи спільного хостингу (shared hosting) для цього зазвичай не підходять, оскільки для роботи системи необхідна повна підтримка Node.js, що потребує доступу до сервера з правами адміністратора. У більшості випадків запуск Node.js вимагає підключення через SSH-протокол, який надає можливість встановлювати середовище виконання, бібліотеки, керувати файлами та конфігураціями на рівні системи. На деяких дешевих shared-хостингах Node.js також доступний

(наприклад, через cPanel або спеціальні інтерфейси), але такий варіант не рекомендований для серйозних або критичних проєктів через обмеження в продуктивності, слабку масштабованість, низький рівень безпеки та обмежені можливості адміністрування.

З огляду на вищевказане, оптимальним рішенням для надійного хостингу є VDS-сервер — віртуальний виділений сервер, що забезпечує максимальну стабільність, контроль над системним середовищем та високий рівень безпеки. VPS (віртуальний приватний сервер) може стати більш бюджетною альтернативою, зберігаючи більшість ключових переваг, але поступаючись у деяких аспектах захисту та гнучкості [6].

Щоб забезпечити контроль над змінами в коді під час розробки, особливо у командній роботі, доцільно впровадити систему контролю версій Git. Це дозволяє:

- створювати незалежні гілки з альтернативними версіями коду;
- об’єднувати зміни з різних гілок;
- відновлювати попередні коміти;
- контролювати внесені зміни в коді кожного розробника.

Для спільної роботи над програмним продуктом необхідно мати віддалене сховище репозиторію, до якого матимуть доступ усі учасники проєкту. Найпопулярніші платформи для цього — GitHub та GitLab, які підтримують як відкриті, так і приватні репозиторії.

Розробники можуть локально завантажити (клонувати) репозиторій на свій комп’ютер, вносити зміни, а потім синхронізувати їх із сервером. Це забезпечує ефективну та контрольовану командну розробку з можливістю роботи над різними функціональностями паралельно.

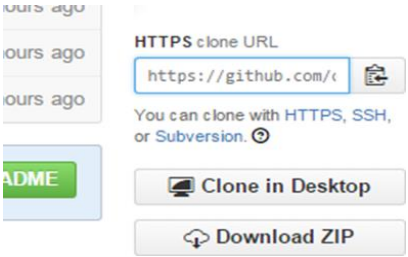

Рисунок 2.3.1 – Функціонал завантаження та клонування репозиторію на Github

2.4. Специфіка користування програмним забезпеченням

Для коректної роботи розробленого програмного продукту рекомендовано використовувати сучасні версії популярних веб-браузерів, зокрема Google Chrome, Mozilla Firefox, Safari, Opera та інші, які підтримують новітні веб-технології.

Під час використання системи виконується аналіз вмісту веб-сторінок, з метою перевірки їх унікальності та оцінки інших параметрів, що мають значення для пошукової оптимізації (SEO). У цьому розділі описано основні функціональні можливості, доступні користувачеві в межах системи. Реєстрація користувача:

Процес реєстрації є необхідною передумовою для створення персонального облікового запису в системі. Якщо користувач не авторизований і вперше заходить на сайт, йому пропонується увійти або створити новий акаунт.

При виборі опції створення облікового запису система перенаправляє користувача на форму реєстрації, де потрібно вказати такі дані: [11].

- Електронна адреса (використовується як логін);
- Ім’я та прізвище;
- Пароль.
- Система валідує введену інформацію згідно з наступними вимогами:
- Пароль має містити не менше 10 символів;
- Обов’язкова наявність хоча б однієї великої та малої літери, цифри та спеціального символу;
- Email повинен бути вказаний у коректному форматі (наприклад, name@example.com);
- Поля з ім’ям та прізвищем не можуть залишатися порожніми.

Після натискання кнопки «Зареєструватися» система перевіряє правильність заповнення форми. У разі помилки користувач бачить повідомлення з поясненням проблеми, і залишається на сторінці для внесення виправлень.

Якщо всі дані введені правильно, система:

- Надсилає підтвердження на вказану електронну адресу у вигляді листа

з інструкцією щодо активації акаунта;

- Виводить на екрані повідомлення про необхідність перевірити пошту;
- Перенаправляє користувача на сторінку входу.

Такий механізм забезпечує високий рівень безпеки, а також захист від несанкціонованого доступу або створення фейкових облікових записів. Уся інформація зберігається у зашифрованому вигляді згідно з вимогами GDPR та інших норм безпеки.

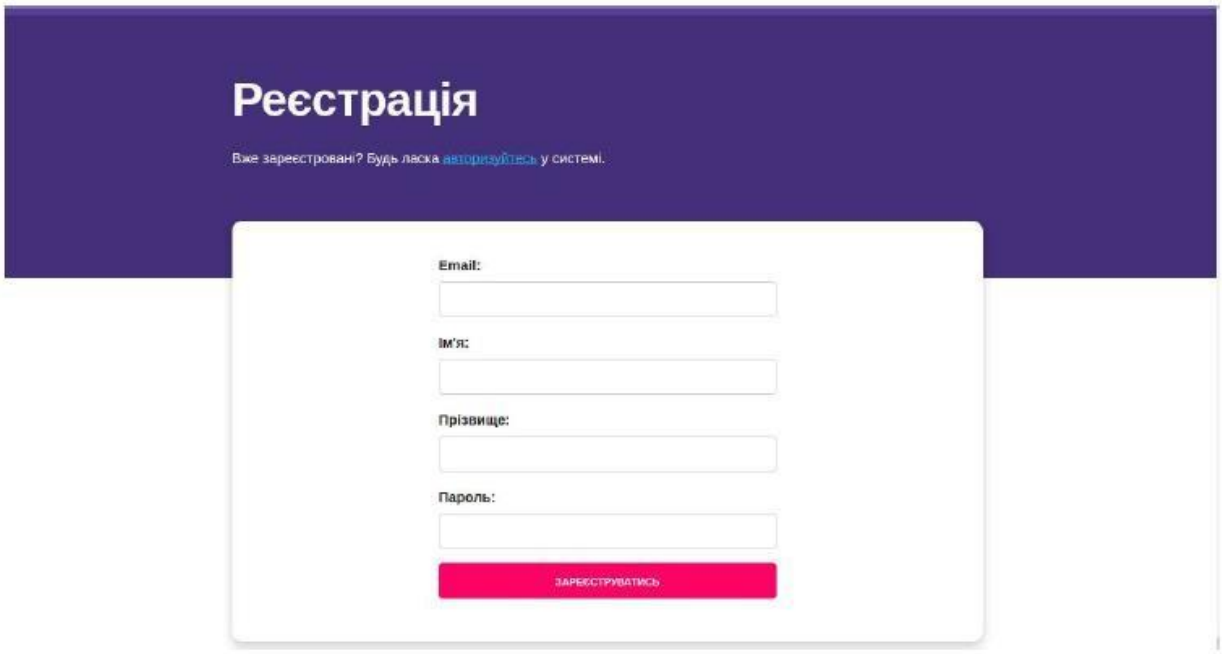

Рисунок 2.4.1 – Реєстрація облікового запису.

У разі, якщо користувач уже має створений обліковий запис, він може пройти процедуру входу до системи. Для цього на сторінці входу необхідно ввести електронну адресу та пароль, а потім натиснути кнопку «Увійти». Після цього формується запит до серверної частини програми. Залежно від правильності введених даних можливі два варіанти розвитку подій: [12].

1. Помилкові дані — якщо вказані email або пароль не відповідають жодному акаунту, користувач побачить сповіщення про те, що такий обліковий запис не знайдено.
2. Успішна перевірка — у випадку коректних облікових даних система

проводить автентифікацію, після чого користувач автоматично перенаправляється на головну сторінку застосунку.

Перевірка сайту на унікальність і SEO-показники

Однією з основних функцій системи є аналіз вмісту веб-сайтів — зокрема перевірка унікальності контенту та оцінка параметрів, що впливають на пошукову оптимізацію (SEO). Для запуску цієї функції користувач має бути авторизованим і знаходитись на головній сторінці програми. Щоб розпочати перевірку, достатньо ввести URL-адресу сайту або сторінки, яку потрібно проаналізувати, у відповідне поле, після чого натиснути кнопку «Старт». Система розпочне автоматичний збір та обробку SEO-даних для подальшого відображення результатів.

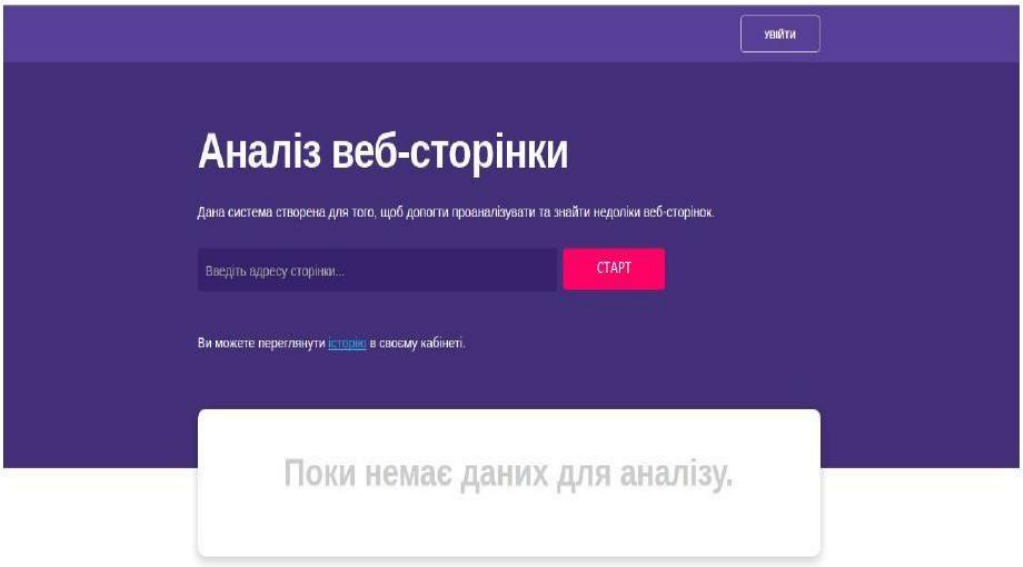

Рисунок 2.4.2 – Головна сторінка

Завершивши аналіз, система автоматично відображає на головній сторінці звіт, що містить виявлені помилки та рекомендації щодо покращення SEO-оптимізації веб-сайту [12].

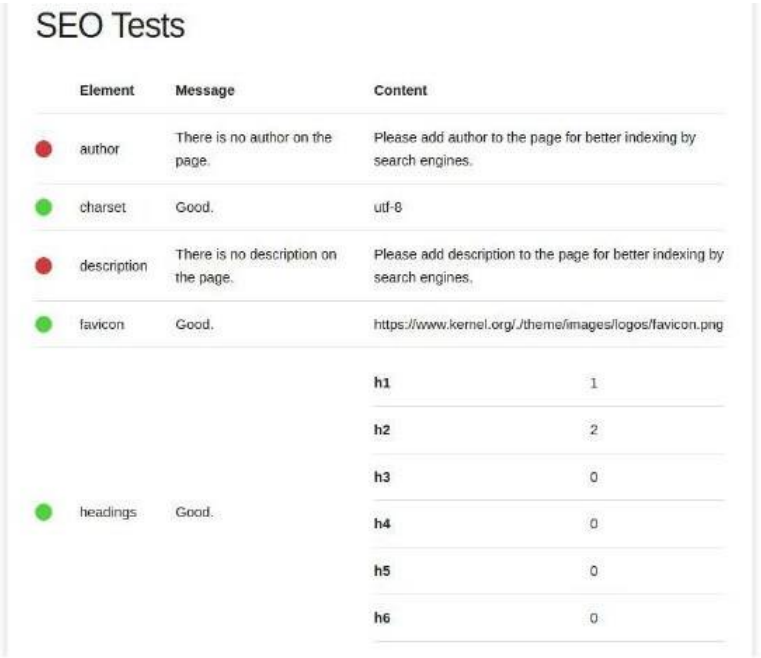

Рисунок 2.4.3 – Результати аналізуSEO-оптимізації.

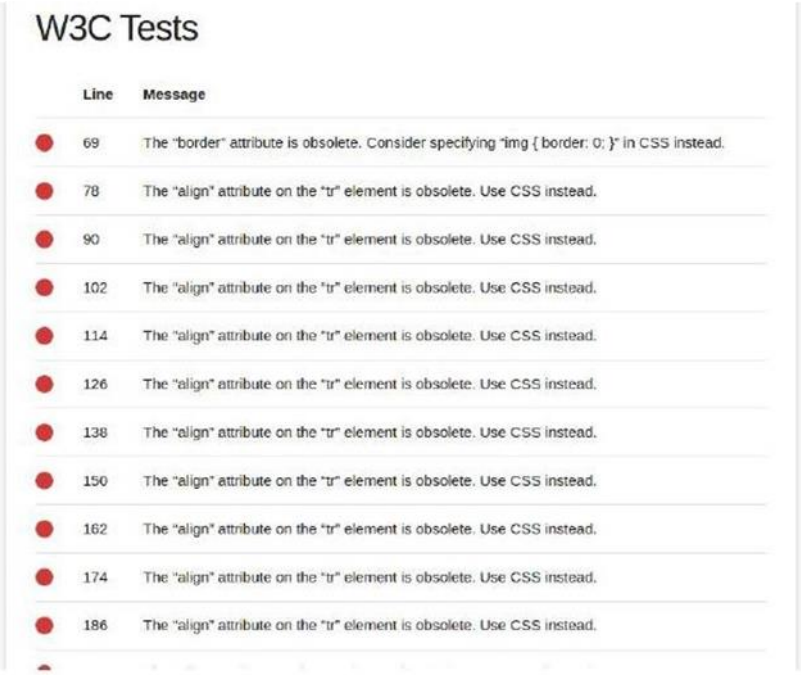

Рисунок 2.4.4. – Резульати виявлення помилок у CSS-коді.

Система дозволяє переглядати історію попередніх перевірок веб-сайтів. Дані зберігаються в базі даних, а доступ до історії здійснюється через головну сторінку (див. Рисунок 2.4.2).

2.5. Тестування програмного забезпечення

Для забезпечення надійності програмного забезпечення було проведено як модульне, так і системне автоматизоване тестування.

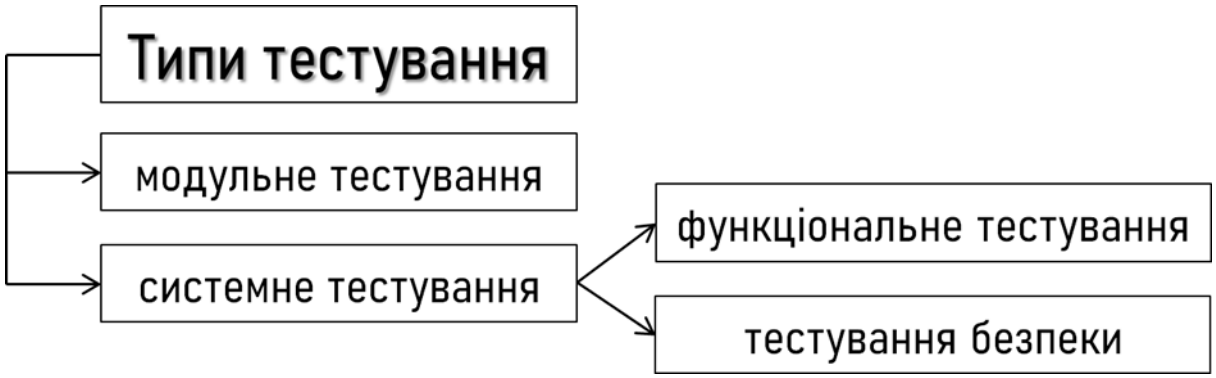

Рисунок 2.5.1. – Типи тестування.

Для проведення модульного тестування використовувався Ranorex Studio 9.0, графічний фреймворк від Ranorex GmbH, який підтримує тестування різних типів програм – від настільних до мобільних та веб-застосунків [8].

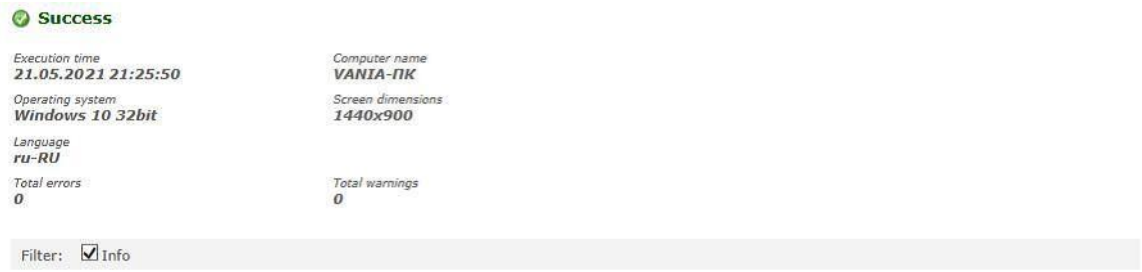

Рисунок 2.5.2 – Підсумок виконання модульного тестування програмної системи.

У ході проведення функціонального тестування було ретельно перевірено роботу всіх основних користувацьких сценаріїв, включаючи процеси реєстрації, входу в систему, ініціалізації аналізу контенту веб-ресурсів на предмет унікальності та відповідності SEO-вимогам, а також перегляд результатів аналізу, який містить як виявлені недоліки, так і автоматично сформовані рекомендації

щодо покращення оптимізації сайту. Усі перевірки були завершені успішно, що підтвердило стабільність функціональних компонентів [8].

Щодо тестування на захищеність, система була піддана імітації несанкціонованих дій, таких як спроби входу з некоректними обліковими даними, внесення змін до бази даних із недійсними токенами автентифікації, а також дії з боку користувачів, які не мають відповідних прав доступу. За підсумками перевірки встановлено, що програмне забезпечення надійно захищене від стороннього втручання й демонструє високий рівень стійкості до спроб зловмисного використання з боку неавторизованих або обмежених у правах користувачів.

РОЗДІЛ 3 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

3.1 Надзвичайні ситуації метеорологічного характеру.

Серед численних природних загроз особливу небезпеку становлять метеорологічні явища, які здатні порушити нормальне функціонування інфраструктури, завдати шкоди населенню та довкіллю. Їхня непередбачуваність та інтенсивність зумовлюють необхідність глибокого вивчення і системного підходу до їхнього попередження та подолання наслідків.

З кожним роком через глобальні зміни клімату метеорологічні надзвичайні ситуації стають дедалі частішими й інтенсивнішими. Особливо це стосується регіонів із континентальним кліматом, де перепади температур, опади та вітрові навантаження можуть змінюватися дуже стрімко. Україна не є винятком: протягом останніх років у різних областях фіксувалися буревії, зливи, аномальні снігопади, спека, заморозки на ґрунті тощо [17].

Відповідно до ДК 019:2010 «Класифікатор надзвичайних ситуацій», метеорологічні надзвичайні ситуації мають код 20300–20339 та охоплюють такі категорії:

1. Атмосферні явища: сильні зливи, снігопади, град, шквали, бурі, тумани, ожеледь, налипання мокрого снігу.
2. Температурні аномалії: сильні морози, спека, заморозки, тривалі засухи.
3. Вітрові навантаження: смерчі, буревії, пилові бурі (швидкість вітру понад 25 м/с).

Основні причини виникнення таких явищ пов’язані з атмосферними фронтами, циклонічною активністю, глобальними змінами клімату та локальними природними аномаліями. Серед антропогенних чинників варто відзначити вирубку лісів, порушення гідрологічного режиму, інтенсивне будівництво та забруднення атмосфери.

Метеорологічні надзвичайні ситуації можуть мати такі наслідки:

1. Пошкодження інфраструктури (електромереж, дорожнього покриття, будівель).

2. Пожежна небезпека у разі тривалих засух.

3. Загроза здоров'ю і життю людей (теплові удари, переохолодження, обмороження).

4. Збитки для сільського господарства (вимерзання культур, посуха, ураження градом).

5. Порушення логістики та транспортного сполучення.

Згідно з вимогами ДСТУ 7743:2015, метеорологічні надзвичайні ситуації мають відстежуватися за допомогою систем спостереження, які забезпечують:

- регулярний моніторинг атмосферних процесів;
- оперативне інформування служб реагування;
- раннє попередження населення про загрозу;
- координацію дій між різними службами цивільного захисту.

Основні заходи реагування включають: [25].

1. Своєчасне інформування населення через ЗМІ, інтернет, мобільні додатки.

2. Підготовку до сезонних ризиків (очищення дахів від снігу, підготовка укриттів, утеплення тощо).

3. Евакуацію населення з небезпечних територій у разі загрози життю.

4. Відновлення пошкодженої інфраструктури після завершення НС.

Нормативне регулювання та державна політика у сфері реагування на метеорологічні надзвичайні ситуації базуються на:

- Кодексі цивільного захисту України;
- ДК 019:2010 – Класифікатор надзвичайних ситуацій;
- ДСТУ 7743:2015 – вимоги до систем раннього виявлення НС;
- ДСТУ 3891:2013 – терміни та визначення понять у сфері НС.

Крім того, важливу роль відіграють наукові дослідження, метеорологічне моделювання, автоматизовані системи раннього оповіщення, а також співпраця з міжнародними організаціями (наприклад, Всесвітньою метеорологічною

організацією).

Таким чином, метеорологічні явища можуть призводити до масштабних надзвичайних ситуацій з тяжкими наслідками для суспільства. Їхнє ефективне попередження та мінімізація шкоди можливі лише за умов функціонування системи моніторингу, чіткого дотримання нормативів (ДСТУ), а також належної обізнаності населення. Забезпечення стійкості інфраструктури та підвищення адаптивності до кліматичних змін є стратегічними завданнями у сфері цивільного захисту.

3.2 Санітарно-гігієнічні вимоги до умов праці.

Організація належних санітарно-гігієнічних умов праці є однією з ключових складових забезпечення безпечного, здорового та продуктивного робочого процесу. Вимоги до цих умов регламентуються як міжнародними стандартами, так і національними нормативно-правовими актами. Серед основних документів, що регламентують санітарно-гігієнічні вимоги, слід виокремити: ДБН В.2.5-28:2018 «Природне і штучне освітлення», ДСТУ ISO 45001:2019 «Системи управління охороною здоров'я та безпекою праці», накази МОЗ України № 341 від 18.02.2022 та № 2205 від 25.09.2020, ДСТУ 8862:2019 щодо гігієнічних вимог побутових виробів, а також санітарні норми і правила МОЗ України. Ці документи є чинними та обов'язковими для застосування у сфері охорони праці станом на 2024–2025 роки [19].

Санітарно-гігієнічні умови праці охоплюють комплекс вимог, спрямованих на створення безпечного та комфортного робочого середовища, що сприяє збереженню здоров'я працівників та попередженню професійних захворювань. Основними складовими таких умов є:

- Мікроклімат. Відповідно до рекомендацій Наказу МОЗ № 341 (2022) та стандарту ДСТУ ISO 45001:2019, оптимальні параметри мікроклімату в робочих

приміщеннях мають відповідати наступним нормам: температура повітря — 22–25°C, відносна вологість — 40–60%, швидкість руху повітря — не більше 0,1 м/с. Забезпечення цих параметрів сприяє комфортному перебуванню працівників і знижує ризик захворювань дихальних шляхів.

- Освітлення. Згідно з вимогами ДБН В.2.5-28:2018 та Наказу МОЗ № 2205 (2020), природне й штучне освітлення повинно відповідати необхідним рівням освітленості залежно від типу діяльності. Для офісних приміщень нормативна освітленість становить не менше 300 лк. Освітлення має бути рівномірним, без мерехтіння та надмірного контрасту, що знижує втомлюваність зорового апарату.
- Площа та об’єм приміщення. Відповідно до рекомендацій МОЗ та ДСТУ ISO 45001:2019, кожен працівник, який працює з комп’ютерною технікою, повинен мати робоче місце з площею не менше 6 м² та об’ємом приміщення не менше 20 м³. Це сприяє запобіганню скупченню та забезпеченню належного повітрообміну [20].
- Вентиляція та повітрообмін. Згідно з ДСТУ ISO 45001:2019, усі робочі приміщення мають бути обладнані системами ефективної вентиляції, що забезпечують кратність повітрообміну відповідно до встановлених норм, а також сприяють видаленню шкідливих речовин, тепла й вологи. Забезпечення належної вентиляції запобігає накопиченню шкідливих факторів у повітрі робочої зони.
- Забезпечення питною водою. Наказ МОЗ № 341 (2022) регламентує вимоги до якості та доступності питної води на робочих місцях. Роботодавець зобов’язаний забезпечити наявність джерел питної води, що відповідає санітарно-гігієнічним нормам за мікробіологічними та хімічними показниками. Вода має бути доступною у достатній кількості протягом усього робочого часу.
- Чистота приміщень і санітарно-побутові умови. Приміщення повинні регулярно прибиратися з використанням дозволених мийних та дезінфекційних засобів, згідно з ДСТУ 8862:2019. Санітарно-побутові умови включають облаштування санвузлів, душових, кімнат для відпочинку, які мають відповідати чинним нормативам і бути укомплектовані відповідними засобами гігієни.

- Ергономіка робочого місця. Відповідно до ДСТУ ISO 6385:2005, робоче місце має бути спроєктоване з урахуванням антропометричних характеристик працівника. Висота столу, крісла, розташування монітора, освітлення, наявність підставки для ніг мають відповідати ергономічним нормам, щоб зменшити навантаження на опорно-руховий апарат, зір і нервову систему.
- Контроль шкідливих факторів. Здійснення регулярного контролю концентрації пилу, хімічних речовин, рівня шуму та вібрацій здійснюється відповідно до ДСТУ ГОСТ 12.1.003:2009 та Наказу МОЗ України № 614 (оновленого у 2022–2023 роках). При необхідності роботодавець проводить санітарно-гігієнічну оцінку умов праці та впроваджує заходи для усунення або мінімізації шкідливих факторів.

Таким чином, дотримання санітарно-гігієнічних вимог до умов праці є не лише юридичною вимогою, а й невід’ємною частиною системи управління охороною здоров’я на підприємстві. Це сприяє збереженню працездатності працівників, зниженню рівня професійних захворювань, підвищенню продуктивності праці та загального рівня безпеки на робочому місці [22].

.

ВИСНОВКИ

У сучасному цифровому середовищі, яке характеризується постійними глобальними змінами, веб-сайти відіграють ключову роль як інструменти взаємодії в економічних, політичних та бізнес-процесах. Їхній вміст повинен відповідати високим вимогам якості: бути унікальним, актуальним, змістовним, доступним для сприйняття, логічно структурованим і водночас привабливим для користувача. Особливу важливість ці параметри набувають у контексті пошукового маркетингу, адже саме контент є основою для просування веб-ресурсів у пошукових системах. З метою досягнення високих позицій у результатах пошуку необхідно регулярно здійснювати перевірку та оптимізацію інформації, розміщеної на сайтах. Через збільшення обсягів контенту, що потребує аналізу, постає потреба в автоматизованих рішеннях, здатних ефективно оцінювати його якість. У межах даного кваліфікаційного проєкту було здійснено розробку програмної системи для перевірки та аналізу веб-контенту відповідно до принципів пошукової оптимізації. У ході реалізації було досліджено предметну область, визначено критерії якості контенту, обґрунтовано вибір технологій для серверної та клієнтської частин системи, зокрема використання JavaScript, Node.js, HTML, CSS та бази даних MongoDB. Розроблена система пройшла модульне і комплексне тестування, що підтвердило її працездатність. Проєкт демонструє ефективність застосування сучасних вебтехнологій для створення інструменту, який відповідає актуальним вимогам цифрового ринку та сприяє підвищенню якості інформаційного наповнення веб-ресурсів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Берко А. Ю. Інтелектуальна система управління контентом сайтів електронного бізнесу/А. Ю. Берко, В. М. Дорош, Л. В. Чирун // Вісник Національного університету «Львівська політехніка». – Львів: Львівська політехніка, 2011. –№ 715:Інформаційні системи та мережі. –С.13–24.

2. Давиденко І. С. Бабюк Н. П. Аналіз мови програмування JavaScript. / LI Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії, Вінниця: ВНТУ, 2022. 2 с.

3. MDN Web Docs. [Електронний ресурс]. – Режим доступу: https://develope r.mozilla.org/

4. W3Schools. [Електронний ресурс]. – Режим доступу: https://www.w3schoo ls.com/

5. JavaScript.info. [Електронний ресурс]. – Режим доступу: https://uk.JavaScr ipt.info/

6. Роббінс Д. HTML5: кишеньковий довідник 5-е видання. / Д. Роббінс – Київ: Діалектика, 2015. 192 с.

7. Мейер Е., Уейл Е. CSS повний довідник. / Е. Мейер, Е. Уейл – Київ: Діалектика, 2017. 1088 с.

8. Уроки SASS / SCSS. [Електронний ресурс]. – Режим доступу: https://itpro ger.com/course/sass

9. Webpack: керівництво для початківців. [Електронний ресурс]. – Режим доступу: https://habr.com/ua/post/514838/.

10. Що таке Node.js. [Електронний ресурс]. – Режим доступу: https://uk.thea strologypage.com/node-js

11. Романюк О. Н. Організація баз даних і знань [Текст] : навчальний посібник / О. Н. Романюк, Т. О. Савчук. - Вінниця : УНІВЕРСУМ-Вінниця, 2003. – 217 с. 56

12. MySQL Workbench. [Електронний ресурс]. – Режим доступу:

https://www.mysql.com/products/workbench/

13. Гурлі Д. HTTP: повний посібник. / Д. Гурлі – Массачусетс: О'Рейлі Медіа 2002. – 656 с.

14. CORS. [Електронний ресурс]. Режим доступу: https://developer.mozilla.o rg/ua/docs/Web/HTTP/CORS

15. Ляхов О.Л. Методи тестування і оцінки якості програмного забезпечення. / О. Л. Ляхов, О.О. Бородіна – Полтава: ПолтНТУ, 2015. – 372 с.

16. Коротун Т. М. Моделі і методи тестування програмних систем / Проблеми програмування. – 2007. –№2. – С. 75–84.

17. Наказ Державного комітету інформаційної політики, телебачення і радіомовлення України і Державний комітету зв'язку та інформатизації України «Про затвердження Порядку інформаційного наповнення та технічного забезпечення Єдиного веб-порталу органів виконавчої влади та Порядку функціонування веб-сайтів органів виконавчої влади» – URL: https://zakon.rada.gov.ua/laws/show/z1021-02/print

18. Наказ Міністерства соціальної політики «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – URL: https://zakon.rada.gov.ua/laws/show/z0508-18/print.

19. Сапега Л. І. Особливості контент-маркетингу як самостійного елементу просування в Інтернеті / Сапега Л. І., Співаковська Т. В. // Актуальні проблеми економіки та управління: збірник наукових праць молодих вчених. – 2014. – Вип. 8.

20. Сєкунова І. О. Суть та значення контент-аналізу у дослідженні інформаційних матеріалів президентських кампаній в Україні. – Бібліотекознавство. Документознавство. Інформологія. – 2013. – N 5. – С.56–59.

21. Согорін, Андрій Анатолійович. Окремі аспекти забезпечення валідності результатів контент-аналізу як методу дослідження соціального дискурсу / А.А. Согорін // Укр. соціум. – 2016. –№2. – С. 41–47.

22. Терещенко В. В. Аналіз сучасних методик пошукової оптимізації / Вісник КрНУ. – 2015. – Вип. 6 (95). – С. 48–54.

23. Тонкіх І.Ю. Критерії якості контенту інтернет-медіа / І. Ю. Тонкіх // Обрії друкарства: Науковий журнал. Електронне видання. – К., 2018. – №1 (6). – С. 209–217.

24. Щербаков Є. В. Аналіз класів мови програмування Javascript / Є. В. Щербаков, М. Є. Щербакова // Вісник Східноукраїнського національного університету імені Володимира Даля. – 2017. – № 8. – С. 90-93. – Режим доступу: http://nbuv.gov.ua/UJRN/VSUNU_2017_8_18

25. ДБН В.2.5-28:2018 «Природне і штучне освітлення». — Київ: Мінрегіонбуд, 2018. — 133 с. — Діє з 01.03.2019 на підставі Наказу № 264 від 03.10.2018. – [Електронний ресурс]. – Режим доступу: https://dbn.co.ua/

26. ДСТУ ISO 45001:2019 «Системи управління охороною здоров'я та безпекою праці. Вимоги та настанови щодо застосування». — Київ: УкрНДНЦ, 2019. — чинний стандарт на 2025 рік. – [Електронний ресурс]. – Режим доступу: https://online.budstandart.com/

27. Наказ МОЗ України № 341 від 18.02.2022 «Про затвердження змін до ДСНП «Гігієнічні вимоги до води питної...»». — Набрав чинності 22.03.2022. – [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0400-22

28. Роз'яснення Державної служби України з питань праці (Держпраці) щодо умов праці з ПК та санітарно-гігієнічних норм (2022–2024 рр.). – [Електронний ресурс]. – Режим доступу: https://dsp.gov.ua/vydacha-dozvoliv/

29. Наказ МОЗ України № 2205 від 25.09.2020 «Про гігієнічні вимоги до умов праці з комп'ютерною технікою». — чинний (станом на 2024 рік). – [Електронний ресурс]. – Режим доступу: https://moz.gov.ua/

30. ДСТУ 8862:2019 «Побутові вироби. Гігієнічні вимоги та методи контролю». — Київ: УкрНДНЦ, 2019. — чинний стандарт. – [Електронний ресурс]. – Режим доступу: https://online.budstandart.com/

31. ДСТУ ISO 6385:2005 «Основні принципи ергономіки для проектування систем праці». — Київ: Держспоживстандарт, 2005. — чинний. – [Електронний ресурс]. – Режим доступу: https://online.budstandart.com/

32. ДСТУ ГОСТ 12.1.003:2009 «ССБТ. Шкідливі речовини. Класифікація та загальні вимоги безпеки». — Київ: Держспоживстандарт, 2009. — чинний. – [Електронний ресурс]. – Режим доступу: https://online.budstandart.com/

33. Наказ МОЗ України № 614 (ред. 2022–2023 рр.) «Про затвердження порядку проведення медичних оглядів працівників певних категорій». — чинний. – [Електронний ресурс]. – Режим доступу: https://moz.gov.ua/

34. Закон України «Про правовий режим надзвичайного стану» від 16.03.2000 № 1550-III. – [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/1550-14#Text

35. Закон України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру» від 08.06.2000 № 1809-III. – [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/1809-14#Text

36. Кодекс цивільного захисту України від 02.10.2012 № 5403-VI. – [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/5403-17#Text

37. Дистанційний курс «Кваліфікаційні роботи бакалаврів» сайту дистанційного навчання ТНТУ [Електронний ресурс]. – Режим доступу: URL: https://dl.tntu.edu.ua/bounce.php?course=5329

38. Методичні вказівки до виконання дипломної роботи освітнього рівня - бакалавр студентами усіх форм навчання для напряму підготовки 121 – Інженерія програмного забезпечення/ Укладачі : Петрик М.Р.,Михалик Д.М., Кінах Я.І., Гладьо С.В., Цуприк Г.Б. – Тернопіль : Вид-во ТНТУ імені Івана Пулюя, 2016 – 28 с.

ДОДАТКИ

Додаток А – Лістинг коду

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";


// === Дані уроків ===
const lessons = [
  {
    id: 1,
    title: "Змінні в JavaScript",
    content:
      "Використовуються ключові слова var, let, const для оголошення змінних.",
    code: `let name = "Олександр";\nconst age = 20;\nvar city = "Львів";`
  },
  {
    id: 2,
    title: "Функції в JavaScript",
    content:
      "Функції дозволяють виконувати повторювані дії багаторазово.",
    code: `function sayHello(name) {\n  return "Привіт, " + name + "!";\n}`
  },
  {
    id: 3,
    title: "Масиви",
    content:
      "Масиви дозволяють зберігати кілька значень в одній змінній.",
    code: `let colors = ["червоний", "зелений", "синій"];`
  }
];

// === Компонент одного уроку ===
const LessonItem = ({ title, content, code }) => (
  <div style={styles.lessonCard}>
    <h3 style={styles.lessonTitle}>{title}</h3>
    <p>{content}</p>
    <pre style={styles.codeBlock}>{code}</pre>
  </div>
);

// === Сторінка з уроками ===
const Lessons = () => (
  <div style={styles.page}>
    <h2 style={styles.heading}>Уроки з JavaScript</h2>
    {lessons.map((lesson) => (
      <LessonItem
        key={lesson.id}
        title={lesson.title}
        content={lesson.content}
        code={lesson.code}
      />
    ))}
```

```
    <Link to="/" style={styles.link}>
      ⬅ Назад на головну
    </Link>
  </div>
);

// === Головна сторінка ===
const Home = () => (
  <div style={styles.page}>
    <h1 style={styles.heading}>Вивчення мови програмування JavaScript</h1>
    <p style={styles.paragraph}>
      Онлайн-платформа для ознайомлення з основами JavaScript. Переглядайте
уроки, вивчайте код, запускайте приклади.
    </p>
    <Link to="/lessons" style={styles.button}>
      Почати навчання
    </Link>
  </div>
);

// === Основний застосунок ===
const App = () => (
  <Router>
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/lessons" element={<Lessons />} />
    </Routes>
  </Router>
);

// === Простий CSS у JS стилі ===
const styles = {
  page: {
    padding: "2rem",
    fontFamily: "Arial, sans-serif",
    maxWidth: "800px",
    margin: "0 auto",
    background: "#f9fafb"
  },
  heading: {
    fontSize: "2rem",
    fontWeight: "bold",
    marginBottom: "1rem"
  },
  paragraph: {
    fontSize: "1.1rem",
    color: "#333"
  },
  button: {
    marginTop: "1rem",
    display: "inline-block",
    padding: "0.75rem 1.5rem",
    background: "#2563eb",
    color: "#fff",
    textDecoration: "none",
    borderRadius: "8px"
  },
  lessonCard: {
```

```
    background: "#fff",
    padding: "1rem",
    borderRadius: "8px",
    marginBottom: "1rem",
    boxShadow: "0 2px 6px rgba(0,0,0,0.1)"
  },
  lessonTitle: {
    fontSize: "1.25rem",
    fontWeight: "bold",
    marginBottom: "0.5rem"
  },
  codeBlock: {
    background: "#f3f4f6",
    padding: "0.75rem",
    borderRadius: "4px",
    whiteSpace: "pre-wrap",
    fontFamily: "monospace",
    marginTop: "0.5rem"
  },
  link: {
    display: "inline-block",
    marginTop: "1rem",
    color: "#2563eb",
    textDecoration: "none"
  }
};

// === Рендеринг додатку ===
ReactDOM.render(<App />, document.getElementById("root"));
```

Додаток Б – Диск з роботою