ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти бакалавр

на тему: «Розроблення інтернет-магазину з продажу комп'ютерної техніки із використанням JavaScript-бібліотеки React»

Виконав: здобувач вищої освіти
за освітньо-професійною
програмою Інформаційні
управляючі системи спеціальності
126 Інформаційні системи та
технології
освітнього ступеня бакалавр
групи 126ІСТ_бд_2022[1](стн)
Левченко Ю.І.
Керівник: Копішинська О.П.
Рецензент: Муравльов В.В.

Полтава – 2024 року

ВСТУП

Актуальність кваліфікаційної роботи обумовлена тим, що в сучасному світі розвиток інформаційних технологій стрімко поширюється на різні сфери життя, зокрема, на сферу торгівлі. Інтернет-торгівля з кожним роком набуває все більшого значення, адже все більше людей надають перевагу онлайн-покупкам через зручність, широкий вибір та можливість порівнювати ціни. Це підкреслює актуальність теми кваліфікаційної роботи, яка спрямована на розробку інтернет-магазину. В умовах стрімкого зростання онлайн-продажів, розробка інтернет-магазину з продажу комп'ютерної техніки стає важливим завданням, яке дозволить відповідати потребам сучасного ринку, надаючи клієнтам зручний і ефективний спосіб придбання товарів.

Метою кваліфікаційної роботи є розробка функціонального інтернет-магазину з використанням JavaScript-бібліотеки React, яка дозволить ефективно презентувати та продавати комп'ютерну техніку через онлайн-платформу. Використання React забезпечить динамічність і швидкість роботи інтерфейсу, що є критично важливим для сучасних вебдодатків.

Завданнями кваліфікаційної роботи є:

– проаналізувати сучасний інструментарій JavaScript-бібліотеки React для вибору найбільш підходящих інструментів та методів розробки;

– розробити зручний та інтуїтивно зрозумілий інтерфейс для користувачів, щоб забезпечити позитивний користувацький досвід і простоту у використанні магазину;

– забезпечити швидку та надійну роботу магазину під час великого потоку відвідувачів, щоб уникнути простоїв і затримок у роботі сайту;

– інтегрувати систему безпеки для захисту персональних даних клієнтів та операцій з платежами, що є надзвичайно важливим для захисту конфіденційності та довіри клієнтів;

– аналізувати статистичні дані та відгуки користувачів для постійного вдосконалення функціоналу та підвищення задоволеності клієнтів.

Об'єктом дослідження є процес розробки інтернет-магазину з продажу комп'ютерної техніки з використанням JavaScript-бібліотеки React, що включає всі етапи від проектування до реалізації та тестування.

Предметом дослідження є функціональність та ефективність використання інтернет-магазину для продажу комп'ютерної техніки, а також вплив обраних технологій на загальну продуктивність та безпеку додатку.

Методами дослідження є аналітичний метод для вивчення існуючих рішень та технологій, інформаційно-пошуковий метод для збору та аналізу даних з наукових джерел і відкритих баз бібліотек, а також програмно-прикладний метод для безпосередньої розробки та тестування інтернет-магазину.

Інформаційна база кваліфікаційної роботи складається з наукових джерел, які містять інформацію про сучасні технології розробки вебдодатків, статистичних звітів міжнародних аналітичних компаній щодо трендів інтернет-торгівлі, документації JavaScript-бібліотеки React, а також аналітики стратегій розвитку онлайн-комерції.

Практична значущість роботи полягає у можливості застосування отриманих результатів для створення ефективного та конкурентоспроможного інтернет-магазину з продажу комп'ютерної техніки, що задовольнятиме потреби користувачів та забезпечить успішну комерційну діяльність.

Апробація результатів відбувалася в рамках фокус-груп тестування, де учасники мали змогу оцінити функціональність та зручність використання розробленого інтернет-магазину.

За результатами досліджень здійснено одну публікацію тез доповідей.

Структура кваліфікаційної роботи логічно пов'язана з завданнями досліджень і містить вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг пояснювальної записки кваліфікаційної роботи складає 59 сторінок формату А4. Вона містить 27 рисунків.

РОЗДІЛ 1
РОЗВИТОК СФЕРИ ІНТЕРНЕТ-МАГАЗИНІВ В УКРАЇНІ

1.1 Аналіз сучасного стану та перспективи розвитку електронної торгівлі в Україні

Використання електронної комерції набуває все більшого поширення в сучасному світі. Інноваційні технології використовуються не лише у Європі та США, але й в Україні, яка також не залишається осторонь. Незважаючи на те, що український цифровий бізнес ще тільки починає свій шлях, він має великий потенціал для подальшого розвитку. Що ж зумовлює цей розвиток ринку електронної комерції? По-перше, інформаційні технології впроваджуються майже у всі сфери суспільного життя: економіку, бізнес, освіту, культуру та інші. Електронна комерція відображає протилежність класичному підприємництву, яке ми знаємо. Інтернет-технології проникають у нові сфери та регіони, охоплюють практично всі сектори людської діяльності та забезпечують більш ефективний продаж товарів. Тепер електронний бізнес стає альтернативою традиційним методам. Мережеві структури знаходяться в активному розвитку, що прискорює торгівлю в онлайні. Таким чином, в інтернеті відбувається обмін інформацією, презентація товарів та послуг, а також їхній безпосередній продаж. Це зручно як для покупця, так і для підприємця, оскільки операції можна проводити, не виходячи з дому. Електронна бізнес-платформа дозволяє здійснювати операції з матеріальними та електронними благами. Згідно з опитуванням Deloitte в Україні [1], перед війною протягом двох років спостерігався помітне зростання онлайн-покупок порівняно з офлайн: 22 % респондентів стали частіше здійснювати покупки в інтернеті, в той час як лише 9 % – офлайн. Ця тенденція тільки зростає, і вона в значній мірі обумовлена наслідками пандемії, яка підштовхнула багатьох до онлайн-шопінгу (рис. 1.1). Обсяг ринку електронної комерції в Україні за даними Soul Partners і Baker Tilly зросла на 41 % в 2020 р., досягнувши 4 млрд доларів, що становило 8,8 % від загального обсягу роздрібної торгівлі. Цей ринок майже втричі збільшився з

2016 р., однак аналітики передбачали подвоєння обсягів протягом наступних п'яти років.

У 2021 р. розвиток електронної комерції був сприяний зростанням кількості активних користувачів, поліпшенням їхнього досвіду покупок та збільшенням довіри до онлайн-замовлень [2].

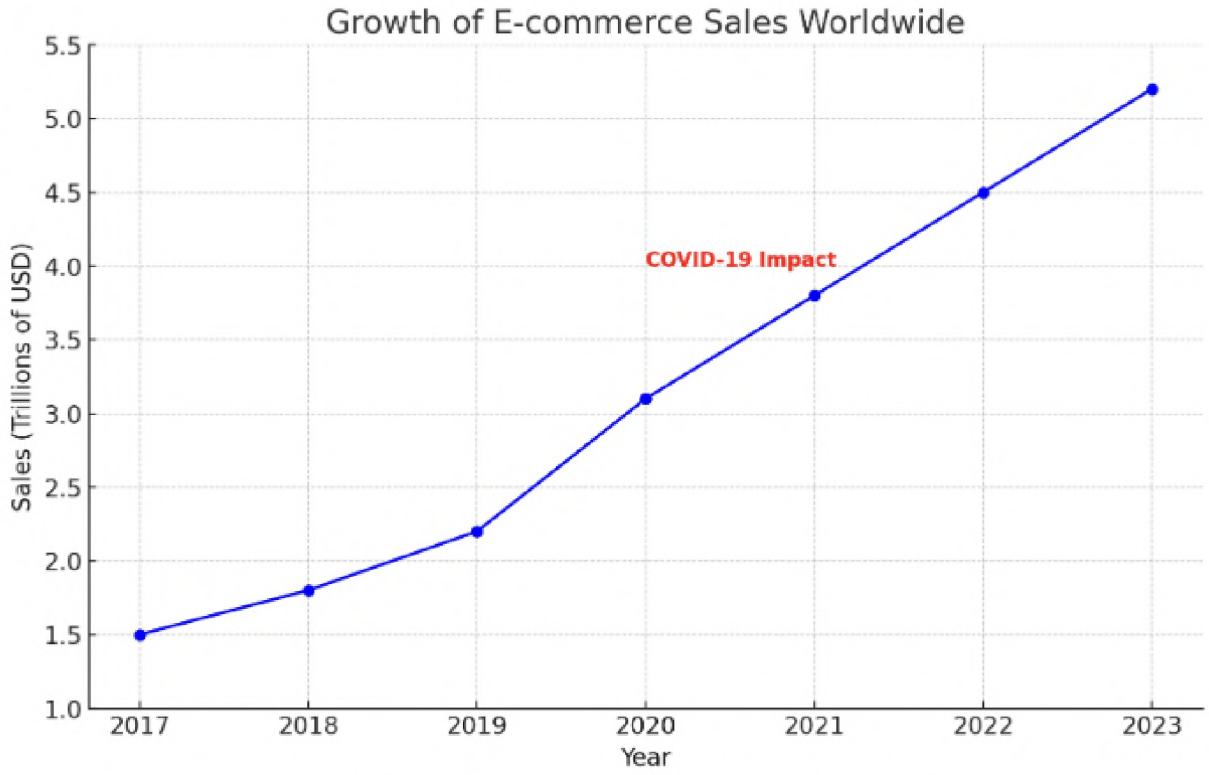

Рисунок 1.1 – Статистика обсягів онлайн покупок [2]

Люди стали активніше використовувати послуги інтернет-магазинів та сервісів доставки, а звичка робити покупки в онлайні, набута під час пандемії, залишилася в повсякденному житті. Простота, швидкість, зручність та безпека залишаються ключовими перевагами електронної комерції. Якщо раніше онлайн-покупки були характерні, переважно, для молоді віком від 18 до 23 років, то зараз цим користуються і представники інших вікових груп, від 25 до 45 років. Це зумовлено зусиллями маркетологів, які активно працюють над привернення більш старшої аудиторії. Ще одним цікавим трендом є здійснення покупок через смартфони, а не через ноутбуки чи комп'ютери. Ця мобільність та можливість здійснювати покупки будь-де і будь-коли сприяють розвитку електронної комерції.

Тому комерційні структури в інтернеті формуються все частіше. Це призводить до створення інноваційного бізнес-середовища, яке надає багато переваг як підприємцям, так і покупцям. Завдяки цьому підприємствам вдається відносно коротким часом впровадити нові технології, налагодити співпрацю з клієнтами, постачальниками та партнерами, а також досягти конкурентоспроможності. Об'єктивно можна стверджувати, що серед двох підприємств ефективніше буде те, яке використовує цифрові технології. Електронна торгівля швидко інтегрується в економіку, тому Україна, подібно до інших цивілізованих країн, прагне регулювати процеси цифровізації на законодавчому рівні. Прозорі алгоритми діяльності в інтернеті є вигідними як для держави, так і для клієнта і виробника. Підприємець чітко знає, яким чином регулюється електронна комерція в Україні на законодавчому рівні, тому він слідує діючим законам і вчасно сплачує податки до державної казни. Відрахування до державного бюджету надходять регулярно, що в свою чергу підвищує рівень ВВП.

1.2 Огляд існуючих платформ та фреймворків для розробки інтернет-магазинів

Розробка вебсайтів може проводитися за допомогою спеціальних конструкторів сайтів або вручну за допомогою фреймворка. Кожен з цих підходів має свої переваги та відмінності.

Розробка сайту на конструкторі дозволяє створити сайт швидко та легко. Використання не потребує спеціальних знань веброзробки та дозволяє швидко створити сайт з готових шаблонів та інструментів. Конструктори сайтів зазвичай мають простий та зрозумілий інтерфейс, що дозволяє створювати сайти навіть користувачам без досвіду веброзробки. Крім того, використання конструктора сайтів може бути дешевше, ніж розробка сайту вручну [3].

Однак, конструктор сайтів має свої обмеження. Використання готових шаблонів та інструментів може обмежити творчість та індивідуальний дизайн

сайту. Крім того, конструктор може мати обмежені можливості розширення та функціональності сайту.

Розробка сайту вручну дозволяє створити більш точний та індивідуальний дизайн, а також забезпечує більшу гнучкість у виборі технологій та функціональності. Це дає можливість розробити унікальний сайт, який повністю відповідає потребам та бажанням клієнта. Крім того, ручна розробка сайту забезпечує більший контроль над кодом і можливість покращення його продуктивності.

Декілька конструкторів сайтів, їх особливості та переваги:

SitePro − це платформа для створення сайтів, яка дозволяє користувачам швидко і легко створювати професійні вебсайти без необхідності мати навички програмування. SitePro пропонує широкий набір інструментів та функцій, які роблять процес розробки сайту доступним для користувачів з різними рівнями технічних знань. Його основною перевагою є, 200 баз шаблонів різних дизайнів. Варіант знайдеться для ресурсів будь-якої тематики: бізнес, кулінарія, подорожі, сфера послуг, лендінг-сторінки, візитки та багато іншого. Крім того, обраний шаблон містить набір тематичних зображень хорошої якості. Їх можно використовувати для створення якісного контенту сайту, який і приваблює користувачів [4].

Wix − це популярна платформа для створення вебсайтів, яка дозволяє користувачам без знань програмування створювати професійні та функціональні вебсайти. Вона надає широкий набір інструментів і функцій, які допомагають як новачкам, так і досвідченим користувачам створювати різноманітні вебресурси. Перевагами є: легкість використання, швидкий запуск та часті оновлення платформи [5].

Elementor − це популярний плагін для WordPress, який дозволяє створювати професійні вебсайти за допомогою візуального конструктора. Він надає користувачам можливість легко налаштовувати сторінки без необхідності писати код, використовуючи функцію перетягування елементів (drag-and-drop). Перевагами є: Elementor сумісний майже з усіма плагінами, тож можливості

відносно сайту майже не обмежені, це кращий конструктор сайтів для блогу, але з ним можна зробити і сайт будь-якої тематики та багато інструкцій і гайдів в мережі, тому що це дуже популярний конструктор для створення сайту. Проте, іншим підходом до створення інтернет-магазинів є використання фреймворків для розробки з нуля [6].

Фреймворк (англ. framework) – це структура або набір платформи, який надає основу для розробки програмного забезпечення. Фреймворк визначає загальну архітектуру програми, встановлює правила та стандарти для взаємодії з компонентами програми, і надає набір інструментів та бібліотек для швидкого створення програм [7]. Основна ідея фреймворку полягає в тому, щоб спростити процес розробки програм, забезпечивши готові структури та інструменти для виконання рутинних задач. Це дозволяє розробникам фокусуватися на бізнес-логіці та унікальних функціях програми, не витрачаючи час на написання повторюваного коду або на роботу з низькорівневими деталями.

На сьогоднішній день існує низка бібліотек та фреймворків, які використовуються для створення вебдодатків. Серед них найбільш поширеними є React, Vue.js, Angular, Svelte, jQuery, Meteor та Backbone. Вони включають у себе різноманітні підходи та технології для ефективної розробки, тестування та роботи з додатками. Використання автоматизованих тестів спрощує процес тестування та дозволяє економити час. Сучасний розвиток цих бібліотек розпочався з виникнення jQuery [8]. Вона вирішувала проблему різноманітних версій JavaScript у різних браузерах, які працювали через маніпуляцію з DOM. Однак у бібліотеки jQuery відсутня була чітка структура для написання повноцінного додатку, та код здебільшого мав некеровану форму.

Починаючи з 2010 р., почалася розробка повноцінних фреймворків для клієнтської частини додатків. Першим із них був Backbone, а потім AngularJS від компанії Google. Ці фреймворки вперше внесли клієнтський роутинг, що дозволило створювати односторінкові додатки (Single Page Application).

Ще одним відомим фреймворком того часу був Ember, який використовував HTML шаблонізатор Handlebars. З появою перших фреймворків

та постійним ускладненням написання клієнтської частини додатків з'явилася необхідність у розширенні області веброзробки для фахівців з фронтенду. Фронтенд-розробник має за завдання проектування користувацьких інтерфейсів для сайтів або додатків та вирішення технічних завдань у цій області. Їм необхідно майстерно володіти HTML, CSS, JavaScript, а також мати базове розуміння SEO, кросбраузерності та юзабіліті. Основна мета фронтенд-розробника – забезпечити швидку, надійну та привабливу роботу вебдодатка [9].

Проведемо короткий аналіз основних конкурентів. У ході дослідження популярних фронтенд-фреймворків було виділено наступні.

Angular – це фреймворк з відкритим кодом, написаний на TypeScript. Розробляється він під керівництвом Angular Team у компанії Google та спільнотою незалежних розробників і корпорацій. Angular є переосмисленою та переробленою версією AngularJS, створеною тією ж командою.

Фреймворк Angular включає в себе:

– модульність: поділ коду на модулі для кращої організації;

– підтримка анімацій для створення плавних інтерфейсів;

– маршрутизація: інструменти для побудови маршрутизації в односторінкових додатках;

– робота з бекендом: засоби для взаємодії з сервером і API;

– зберігання, обробка, відображення даних.

Основні компоненти архітектури Angular: Module, Component, Template, Service, Router, Pipe, Directives (рис. 1.2).

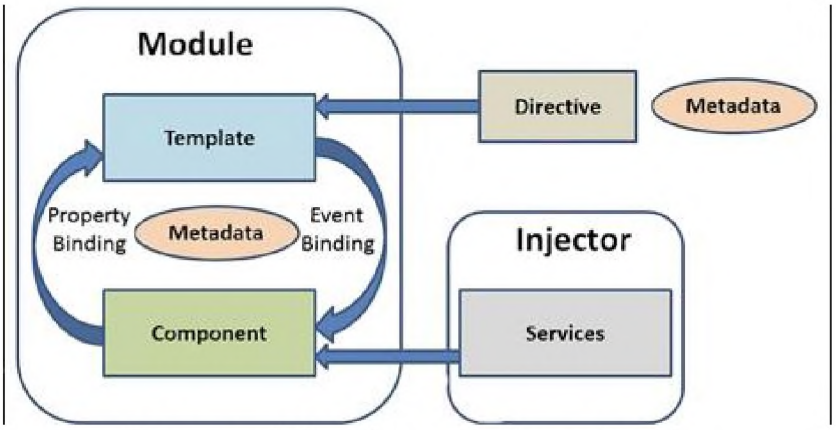

Рисунок 1.2 – Схема роботи Angular

Модулі (Module) є структурними одиницями застосунку, що інкапсулюють певну логіку. В Angular модулі об’єднують компоненти, директиви та сервіси, які пов’язані певною логікою. Прикладами модулів можуть бути модуль для профілю користувача, модуль для написання листа або модуль для перегляду списку листів.

Компоненти (Component) представляють собою класи TypeScript, що зберігають дані та логіку відображення цих даних у шаблоні. Шаблон тісно пов'язаний з компонентом, і дані з компонента можна легко відображати в шаблоні, використовуючи спеціальний синтаксис. Компонент також може отримувати дані безпосередньо з шаблону.

Шаблон (Template) є фрагментом HTML-коду зі спеціальним синтаксисом, який дозволяє вбудовувати дані з компонента без використання innerHTML та подібних методів. Шаблон прописується в компоненті та є частиною його конфігурації.

Сервіси (Service) в Angular є класами TypeScript, які виконують задачі, пов’язані з отриманням, зберіганням та обробкою даних. Наприклад, сервіси можуть виконувати логування, перетворення даних для подальшої передачі в компонент, звернення до бекенду тощо. На відміну від компонентів та директив, сервіси не працюють напряму з представленнями (шаблонами).

Роутер (Router) використовується для переходу між екранами з метою відображення різного контенту. Він відстежує зміни у фрагменті URL в адресному рядку браузера та завантажує відповідні частини застосунку.

Директиви та пайпи є більш специфічними конструкціями, які простіше продемонструвати в коді, ніж описати словами. Ці компоненти складають основу архітектури Angular, забезпечуючи чітку структуру та зручність у розробці вебдодатків [10].

Vue.js – це фреймворк, який знайшов баланс між обмеженнями та гнучкістю. Його ядро в основному вирішує задачі представлення даних, що дозволяє легко інтегрувати його в існуючі проекти поступово. Однак Vue.js

також підходить для створення повноцінних односторінкових додатків (SPA), оскільки має повний набір функціоналу.

Цей фреймворк активно завойовує популярність серед розробників завдяки своїй хорошій документації, низькому рівню входу та невеликому розміру. Vue.js пропонує такі можливості, як CLI, маршрутизація (подібна до Angular), використання Virtual DOM і швидкий час розробки (як у React) [11].

Незважаючи на це, Vue.js має менш гнучкий компонентний підхід порівняно з React, а керування життєвим циклом компонентів, яке відбувається «під капотом», може стати проблемою у великих проєктах через неочевидність та складність відлагодження. Таким чином, Vue.js є відмінним вибором для розробки більшості вебдодатків, але для дуже складних інтерфейсів може бути доцільніше звернутися до React.

Svelte.js − це так званий зникаючий JavaScript-фреймворк. Являє собою принципово новий підхід до розробки фронтенду. На відміну від типових фреймворків, які завантажуються в браузер і починають працювати, Svelte є компілятором. Він перетворює код, написаний з використанням власного синтаксису, в елегантний і оптимізований чистий JavaScript-код. У Svelte відсутні Virtual DOM і абстракції, використовується лише чистий низькорівневий JavaScript.

Основною перевагою, яку надають фреймворки, є те, що в браузер користувача не потрібно завантажувати сам фреймворк: це зменшує навантаження. Відсутність необхідності обробляти Virtual DOM також знижує використання ресурсів, а збірник сміття JavaScript може ефективніше звільняти пам'ять, яку використовує програма.

Проте для великих застосунків, де розмір фреймворку (25–50 чи навіть 170 КБ) становить лише 1–2% від загального розміру застосунку, а Virtual DOM дає більше користі, ніж використовує ресурсів, переваги Svelte можуть бути несуттєвими. Незважаючи на це, Svelte активно набирає популярність і може стати новим подихом у світі фронтенду. Однак, на даний момент, вакансій для розробників, які працюють з Svelte, дуже мало [12].

У ході дослідження існуючих платформ та фреймворків для розробки інтернет-магазинів було виявлено, що кожен з розглянутих інструментів має свої унікальні переваги та недоліки. Аналіз показав, що вибір платформи або фреймворку значною мірою залежить від специфічних вимог проекту, таких як масштабованість, продуктивність, легкість у використанні, підтримка та інтеграція з іншими системами.

Платформи для розробки інтернет-магазинів, такі як SitePro, Wix та Elementor, забезпечують готові рішення, які дозволяють швидко розпочати продажі в інтернеті. Вони пропонують широкий набір функцій, що включає управління товарами, платежами, доставкою та маркетинговими інструментами. Проте, їхні обмеження в гнучкості та кастомізації можуть стати перешкодою для специфічних бізнес-вимог.

Фреймворки для розробки інтернет-магазинів, такі як React, Angular, Vue.js та Svelte, надають розробникам більшу свободу в створенні користувацьких інтерфейсів та інтеграції з бекенд-системами. Використання цих фреймворків дозволяє створювати високоефективні, масштабовані та інтерактивні вебдодатки. Зокрема, React і Vue.js відзначаються простотою у вивченні та високою продуктивністю, тоді як Angular пропонує більш комплексний підхід з багатою функціональністю. Svelte, зі свого боку, забезпечує високу швидкість та оптимізацію за рахунок компіляції коду, що дозволяє зменшити навантаження на браузер.

Таким чином, вибір між платформою та фреймворком для розробки інтернет-магазину повинен базуватися на детальному аналізі вимог проекту та ресурсів команди розробників. Платформи підходять для швидкого запуску та управління стандартними функціями електронної комерції, тоді як фреймворки забезпечують гнучкість та можливість створення унікальних рішень, що відповідають специфічним потребам бізнесу.

1.3 Особливості та переваги використання JavaScript-бібліотеки React для розробки інтернет-магазинів

React - це відкрита JavaScript-бібліотека, розроблена компанією Facebook, яка дозволяє розробникам будувати інтерфейси користувача вебдодатків. Однією з основних особливостей React є використання компонентного підходу до розробки, що дозволяє розділити користувацький інтерфейс на невеликі незалежні компоненти, якими легко керувати і перевикористовувати [13].

Переваги використання React для розробки інтернет-магазинів:

– швидка реакція на зміни: завдяки використанню віртуального DOM React забезпечує швидку реакцію на зміни даних (рис. 1.3). Це дозволяє покупцям отримувати миттєвий зворотний зв'язок при взаємодії з сайтом;

– компонентна структура: React дозволяє розробникам створювати незалежні компоненти, які можна легко перевикористовувати у різних частинах магазину. Це спрощує процес розробки та підтримки вебдодатку;

– односторінкова програма: React підтримує реалізацію односторінкової архітектури, що дозволяє створювати динамічні та інтерактивні інтерфейси без перезавантаження сторінок. Це забезпечує зручність користувачів та покращує загальний досвід взаємодії;

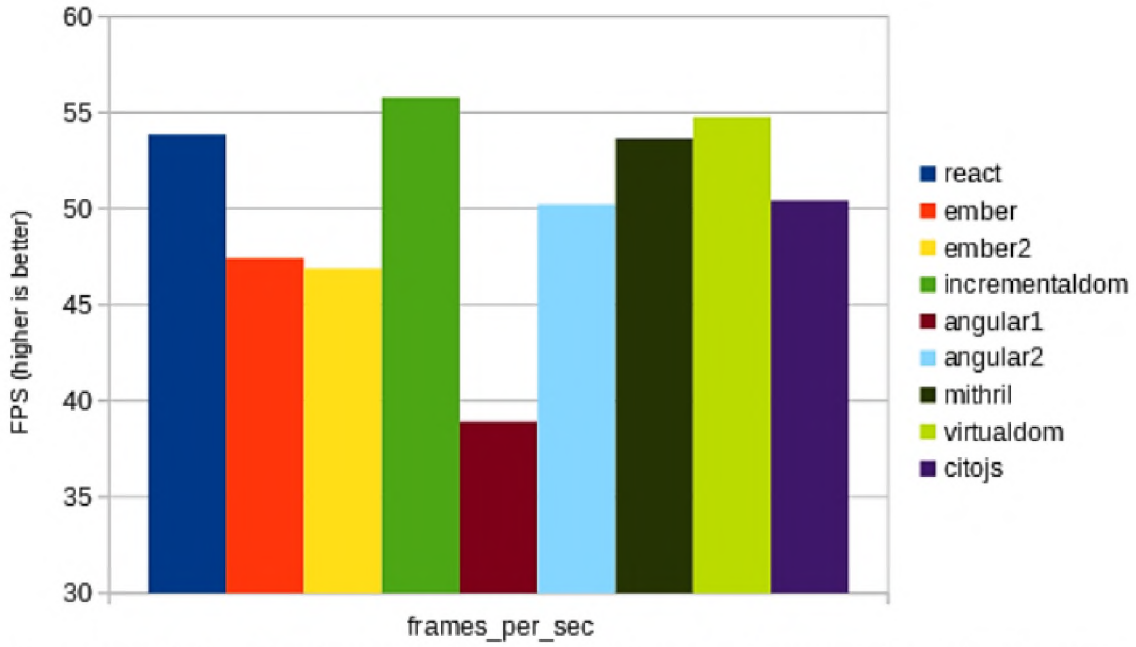

Рисунок. 1.3 – Порівняння швидкості React з іншими фреймворками

– розширюваність: з використанням різноманітних доповнень та бібліотек React може бути легко розширений для вирішення конкретних вимог магазину. Це дозволяє розробникам швидко впроваджувати нові функції та можливості;

– підтримка спільноти: React має велику та активну спільноту розробників, що забезпечує доступ до безлічі документації, рішень та підказок. Це сприяє швидкій і якісній розробці магазину та вирішенню потенційних проблем;

– SEO-оптимізація: React дозволяє реалізувати серверний рендерінг, що сприяє покращенню індексації вебсайту пошуковими системами. Це особливо важливо для інтернет-магазинів, які залежать від органічного трафіку для залучення клієнтів;

– модулярність та гнучкість: Завдяки модульній архітектурі React, розробники можуть легко масштабувати магазин, додавати нові функції та змінювати існуючі без значного впливу на інші частини додатку. Це дозволяє швидко реагувати на зміни в бізнес-вимогах та впроваджувати інновації.

– кросплатформеність: React може бути використаний для розробки не лише вебдодатків, а й мобільних застосунків за допомогою React Native. Це дозволяє створювати єдиний код для різних платформ, що заощаджує час та ресурси розробки;

– безпека: React дозволяє використовувати безпечні методи для маніпуляції з даними, такі як контрольовані компоненти та використання JavaScript Expressions. Це допомагає уникнути можливих уразливостей безпеки та забезпечити захист конфіденційності клієнтів.

JavaScript-бібліотека React виявляється потужним інструментом для розробки інтернет-магазинів, завдяки своїм особливостям та перевагам, таким як швидка реакція на зміни, компонентна структура, односторінкова архітектура, розширюваність та підтримка спільноти. Використання React дозволяє створювати ефективні, швидкі та користувацько-орієнтовані вебплатформи, що відповідають сучасним вимогам електронної комерції [14].

1.4 Оцінка переваг та недоліків VS Code як інтегрованого середовища розробки для побудови вебдодатків

Visual Studio Code, або VSCode — це однин з найпопулярніших інтегрованих середовищ для веброзробки, розроблений компанією Microsoft. Він пропонує широкий спектр функціональних можливостей та розширень, які допомагають веброзробникам підвищити продуктивність та зручність роботи (рис. 1.4).

Однією з ключових переваг VS Code є його вбудована інтеграція з різноманітними інструментами та технологіями веброзробки. Наприклад, VS Code підтримує мови програмування, такі як HTML, CSS, JavaScript, Python та інші. Крім того, він має інтеграцію з Git, що спрощує керування версіями коду та співпрацю з іншими розробниками [15].

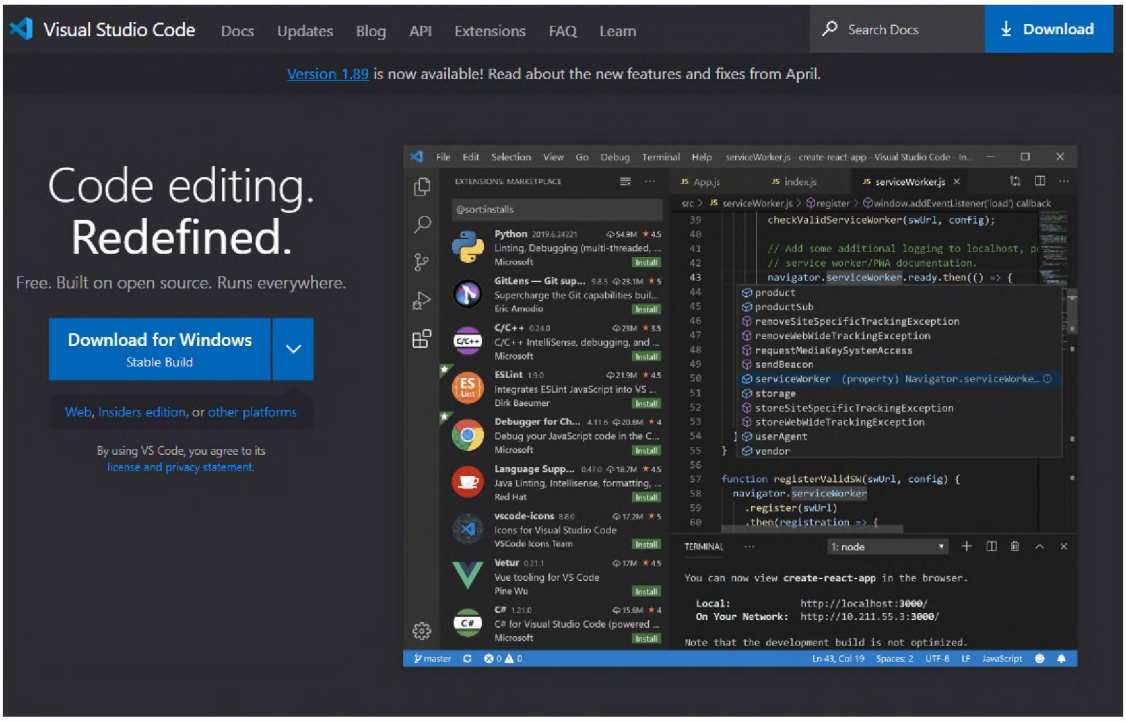

Рисунок. 1.4 – Головна сторінка VS Code

Додатково, VS Code пропонує широкий набір корисних функцій, які полегшують написання коду та зменшують кількість помилок. Наприклад, автодоповнення коду допомагає веброзробникам швидше та точніше писати код. Також має вбудовані засоби для відлагодження коду, що дозволяє розробникам

швидко виявляти та виправляти помилки. Плюс до цього, VS Code має широкий вибір розширень та плагінів, які дозволяють розширити його функціональність та налаштувати під потреби веброзробника. Наприклад, доступні розширення для роботи з популярними фреймворками, такими як React, Angular або Vue, а також для взаємодії з базами даних та іншими інструментами.

Виокремлені переваги VS Code також включають:

– багатофункціональність: можливість працювати з різноманітними типами файлів, включаючи HTML, CSS, JavaScript та інші, а також підтримка більшості популярних фреймворків та бібліотек веброзробки;

– керування Git: вбудована підтримка системи контролю версій Git для легкого виконання комітів, перегляду відмінностей та злиття гілок;

– відлагодження: підтримка відлагодження для багатьох мов програмування для ефективного виявлення та виправлення помилок у коді.

Ще однією важливою рисою VS Code є його можливість розширення функціональності через сторонні розширення та плагіни. Це дозволяє веброзробникам адаптувати середовище до своїх унікальних потреб та використовувати додаткові інструменти, які полегшують їхню роботу. Наприклад, доступні розширення для автоматизації завдань, підтримки різних фреймворків та бібліотек, інтеграції з різними сервісами хмарного сховища, і навіть для розвитку додаткових можливостей відлагодження.

Недоліки VS Code:

– виснаженість ресурсів: у деяких випадках, особливо при використанні багато розширень, VS Code може споживати значні обсяги оперативної пам'яті та процесорних ресурсів, що може призвести до збоїв або зниження продуктивності системи;

– неідеальна підтримка деяких мов програмування: хоча VS Code підтримує багато мов програмування, проте відмічаються деякі обмеження або недоліки у підтримці окремих мов, що може стати проблемою для розробників, що працюють з екзотичними мовами або фреймворками;

– багатофункціональність може бути затяжною для новачків: для новачків, велика кількість функцій та налаштувань може бути викликом, і вони можуть втратитися в обиранні оптимальних інструментів та налаштуваннях для своєї роботи;

– повільне завантаження: у порівнянні з іншими текстовими редакторами, VS Code може завантажуватися повільніше, особливо при великій кількості встановлених розширень;

– питання конфіденційності: деякі користувачі висловлюють занепокоєння щодо даних, які VS Code може збирати та відправляти до серверів Microsoft, що може стати проблемою для тих, хто цінує приватність та безпеку [16].

У результаті, Visual Studio Code стає не лише інструментом для написання коду, але й повноцінним робочим середовищем, що об'єднує в собі потужність, зручність та розширюваність. Він допомагає веброзробникам реалізувати їхні ідеї та створювати високоякісні вебсайти та додатки з мінімальними зусиллями.

Отже, VS Code значно спрощує роботу веброзробників і допомагає створювати якісний код. Завдяки своїм можливостям та розширенням, ця середа розробки є відмінним вибором для розробки вебсайтів та додатків.

РОЗДІЛ 2
ОБГРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ ДЛЯ РОЗРОБЛЕННЯ ІНТЕРНЕТ-МАГАЗИНУ

2.1 Аналіз вимог до функціоналу та інтерфейсу інтернет-магазину комп'ютерної техніки

Недоліки у відображенні та функціоналі сайту можуть викликати серйозні проблеми. Вони призводять до незадоволення користувачів, що може призвести до втрати аудиторії, яка швидко шукає кращі альтернативи. Крім того, складний інтерфейс ускладнює навігацію, що може призвести до того, що користувачі не зможуть знайти потрібну інформацію, що впливає на їхнє задоволення та створює негативне сприйняття бренду. Додатково, нестабільна або неправильна робота функціоналу може призвести до втрати даних користувачів або навіть до збоїв у роботі сайту. Це може стати наслідком втрати клієнтів, особливо у випадку інтернет-магазинів або сервісів, де навіть невелика помилка може коштувати втрати клієнтів. Крім того, недоліки у функціоналі можуть стати причиною кібератак або інших кіберзагроз, загрожуючи як користувачам, так і власникам сайту.

Функціонал інтернет-магазину комп'ютерної техніки повинен відповідати потребам сучасного споживача та забезпечувати зручний та ефективний процес здійснення покупки.

До основних вимог до функціоналу можна віднести:

– каталог товарів: наявність широкого асортименту товарів, їх структуроване представлення та можливість швидкого пошуку за параметрами;

– корзина та оформлення замовлення: зручний механізм додавання товарів до корзини, можливість редагування замовлення та простий процес оформлення покупки;

– оплата та доставка: різноманітні методи оплати та швидка, надійна доставка товарів до клієнтів;

– огляд та оцінки товарів: можливість перегляду детальних описів товарів, відгуків користувачів та надання власних оцінок;

– особистий кабінет користувача: функціонал для реєстрації користувачів, перегляду історії замовлень та зміни особистої інформації.

Інтерфейс інтернет-магазину має бути інтуїтивно зрозумілим та привабливим для користувача. До основних вимог до інтерфейсу можна віднести:

– зручна навігація: легка знахідка потрібних розділів та товарів, інтуїтивно зрозуміла структура сайту;

– привабливий дизайн: естетично оформлений інтерфейс, який привертає увагу користувача та створює позитивне враження;

– адаптивність: можливість коректного відображення сайту на різних пристроях (комп'ютери, планшети, смартфони);

– швидкість реакції: мінімальний час завантаження сторінок та швидка відповідь на дії користувача;

– зручність оформлення замовлення: інтуїтивно зрозумілі форми для заповнення даних та оформлення покупки.

Аналіз вимог до функціоналу та інтерфейсу інтернет-магазину комп'ютерної техніки вказує на необхідність забезпечення високої якості та зручності користування платформою. Оптимальне поєднання функціональності та естетики дозволить привернути та утримати клієнтів, забезпечуючи успішне функціонування бізнесу в умовах онлайн-торгівлі [17] (додаток А).

2.2 Аналіз архітектур вебдодатків

Існує кілька основних архітектурних підходів до розробки вебдодатків. Один із найпоширеніших підходів – монолітна архітектура, в якій весь додаток розгортається як єдиний модуль. Хоча монолітна архітектура може бути простою у розробці та розгортанні, вона не дуже гнучка та масштабована, оскільки будь-яка зміна вимагає оновлення всього додатку.

На противагу цьому, мікросервісна архітектура розбиває додаток на набір незалежних, невеликих служб, кожна з яких відповідає за певну бізнес-логіку або функціональність. Ці мікросервіси можуть бути розроблені та розгорнуті окремо, використовуючи різні технології та мови програмування. Такий підхід забезпечує кращу гнучкість, масштабованість та стійкість до збоїв, оскільки збій в одному мікросервісі не впливає на роботу інших. Однак мікросервісна архітектура може бути складнішою у розробці та управлінні через необхідність забезпечення належної комунікації між мікросервісами та вирішення питань, пов'язаних з розподіленими системами. Схематично описані архітектури показані на рис. 2.1.

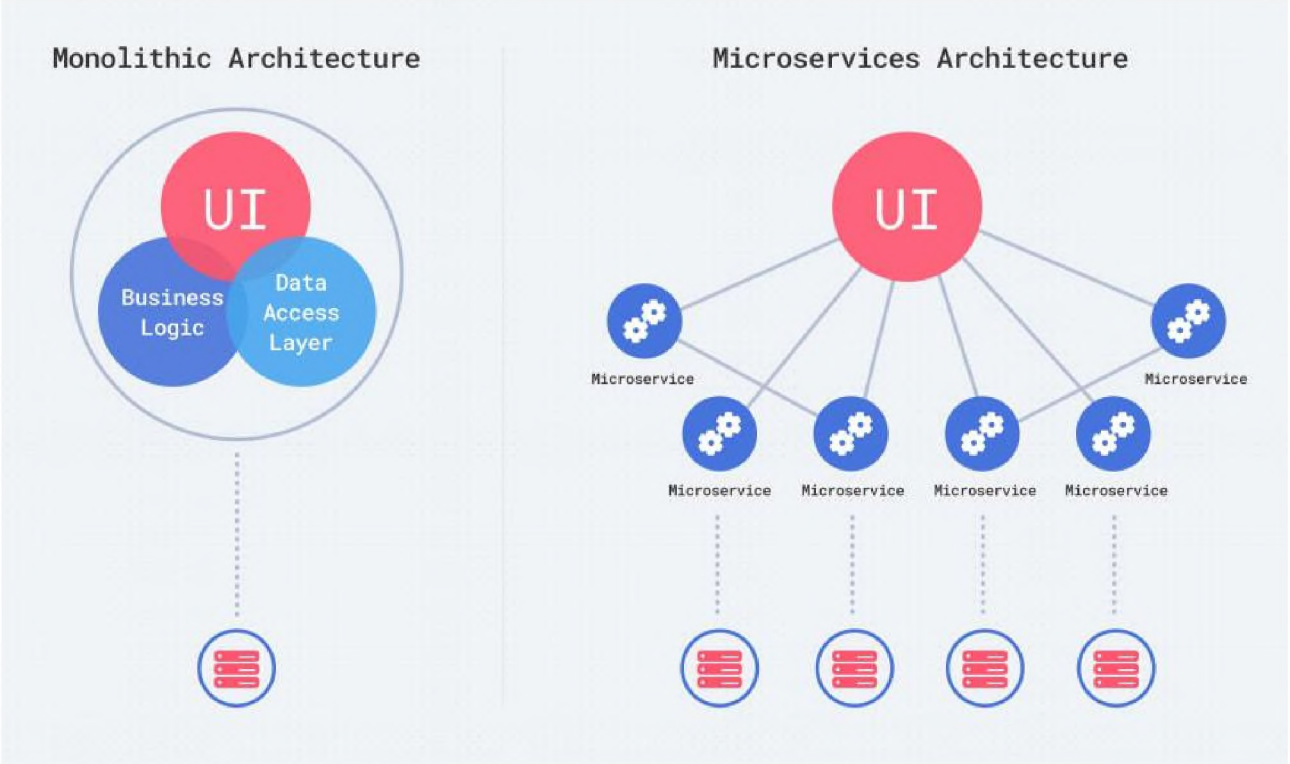

Рисунок. 2.1 – Порівняння мікросервісної та монолітної архітектури

Іншим популярним підходом є клієнт-серверна архітектура, в якій клієнтська частина (веббраузер) відокремлена від серверної частини. Клієнтська частина відповідає за представлення даних та взаємодію з користувачем, тоді як серверна частина займається обробкою даних, логікою додатку та зберіганням даних. Цей підхід забезпечує чітке розмежування обов'язків та полегшує розробку і тестування різних компонентів окремо. Однак він може бути менш

ефективним у випадках, коли потрібна тісна інтеграція між клієнтською та серверною частинами.

Крім того, існує підхід з роздільним клієнтом та API (Application Programming Interface), де клієнтська частина взаємодіє з серверною частиною через чітко визначений API. Цей підхід забезпечує високу гнучкість та можливість підключення різних клієнтів (веб, мобільні додатки тощо) до одного і того ж API. Однак він може бути більш складним у розробці та вимагати ретельного планування та документування API.

Також варто згадати про хмарну архітектуру, яка передбачає розміщення додатку в хмарному середовищі, таких як Amazon Web Services (AWS), Microsoft Azure або Google Cloud Platform. Хмарна архітектура забезпечує масштабованість, високу доступність та можливість гнучкого масштабування ресурсів відповідно до навантаження. Однак вона може бути дорогою та вимагати навичок управління хмарними ресурсами.

Ще одним важливим архітектурним підходом є Event-Driven Architecture (EDA), яка базується на асинхронній обробці подій та повідомлень. У такій архітектурі різні компоненти додатку взаємодіють шляхом публікації та підписки на події або повідомлення через брокер повідомлень. EDA забезпечує високу гнучкість, масштабованість та децентралізацію, але може бути складною у розробці та відстеженні потоків подій.

Крім того, існують різні архітектурні шаблони та підходи, такі як шаблон Model-View-Controller (MVC), Model-View-Presenter (MVP), Model-View-ViewModel (MVVM), шаблон Flux та інші. Вибір відповідного архітектурного підходу залежить від вимог до додатку, масштабованості, продуктивності, гнучкості та рівня складності.

Важливо також враховувати питання безпеки, оскільки вебдодатки часто обробляють конфіденційні дані та можуть бути вразливими до різних типів атак [18]. Належна архітектура безпеки, включаючи такі заходи як шифрування, автентифікацію, авторизацію, захист від ін'єкцій та хакінгу, є вкрай важливою для забезпечення конфіденційності та цілісності даних.

Окрім власне архітектури вебдодатку, важливо також враховувати додаткові аспекти, такі як інтеграція з базами даних, кешування, моніторинг та логування, розгортання та автоматизація, тестування та безперервна інтеграція/безперервне розгортання (CI/CD). Ці аспекти відіграють важливу роль у забезпеченні надійності, продуктивності та якості вебдодатків.

Отже, вибір відповідної архітектури вебдодатку залежить від багатьох факторів, включаючи вимоги до масштабованості, продуктивності, гнучкості, безпеки та складності додатку. Кожен архітектурний підхід має свої переваги та недоліки, і розробники повинні ретельно оцінити їх, щоб обрати найбільш підходящий варіант для конкретного проекту. Крім того, важливо враховувати додаткові аспекти, такі як інтеграція, тестування, розгортання та моніторинг, щоб забезпечити належну якість та надійність вебдодатку.

2.3 Використання можливостей взаємодії між React і сервером для побудови ефективного обміну даними між клієнтом та сервером

Взаємодія між React і сервером є ключовим аспектом побудови сучасних вебдодатків, що забезпечують ефективний обмін даними між клієнтською та серверною частинами. Цей процес вимагає врахування ряду технічних аспектів та вибору оптимальних підходів для досягнення високої продуктивності та надійності.

Одним із головних підходів до взаємодії між React та сервером є використання HTTP-запитів. React компоненти можуть виконувати запити до сервера за допомогою стандартних бібліотек, таких як Axios або Fetch API. Axios надає простий та потужний інтерфейс для обробки запитів та відповідей, забезпечуючи автоматичне перетворення JSON-даних та підтримку обробки помилок. Fetch API, будучи вбудованою в браузери, є легким та ефективним рішенням для здійснення HTTP-запитів, що особливо важливо для малих і середніх додатків [19].

Інший підхід полягає у використанні WebSocket для реалізації реального часу обміну даними. WebSocket дозволяє встановлювати постійне з'єднання між клієнтом і сервером, що значно знижує затримки при обміні повідомленнями порівняно з традиційними HTTP-запитами. Це особливо корисно для додатків, які вимагають оперативного оновлення даних, таких як чати, онлайн-ігри та фінансові трекери. В React можна легко інтегрувати WebSocket через сторонні бібліотеки, такі як Socket.io, що надає зручний API для роботи з WebSocket-з'єднаннями [20].

GraphQL є ще одним потужним інструментом для побудови ефективної взаємодії між React та сервером. GraphQL дозволяє клієнту запитувати лише ті дані, які йому дійсно необхідні, що зменшує об'єм передаваних даних та оптимізує мережевий трафік. Крім того, GraphQL підтримує агрегування запитів, що дозволяє значно зменшити кількість необхідних звернень до сервера. Бібліотека Apollo Client забезпечує зручну інтеграцію GraphQL з React, надаючи можливості кешування, обробки помилок та управління станом додатку [20].

Одним із важливих аспектів обміну даними між React і сервером є управління станом додатку. Використання бібліотек для управління станом, таких як Redux або MobX, дозволяє централізувати обробку даних і забезпечити синхронізацію стану додатку з даними, отриманими від сервера. Redux, наприклад, пропонує концепцію єдиного сховища (store), що дозволяє легко відстежувати зміни стану та спрощує дебагінг. Комбінація Redux з Redux Thunk або Redux Saga дозволяє ефективно обробляти асинхронні дії, такі як HTTP-запити.

Включення React Server Components у взаємодію між React та сервером відкриває нові можливості для побудови високоефективних вебдодатків. React Server Components (RSC) дозволяють виконувати рендеринг частини компонентів на сервері, що знижує навантаження на клієнт та покращує продуктивність (рис. 2.2).

React Server Components дозволяють розподілити обчислювальні навантаження між клієнтом і сервером, що є особливо корисним для великих

додатків з багатими інтерфейсами користувача. Завдяки RSC, частина роботи з обробки даних і рендерингу компонентів виконується на сервері, а клієнту передається вже готовий результат.

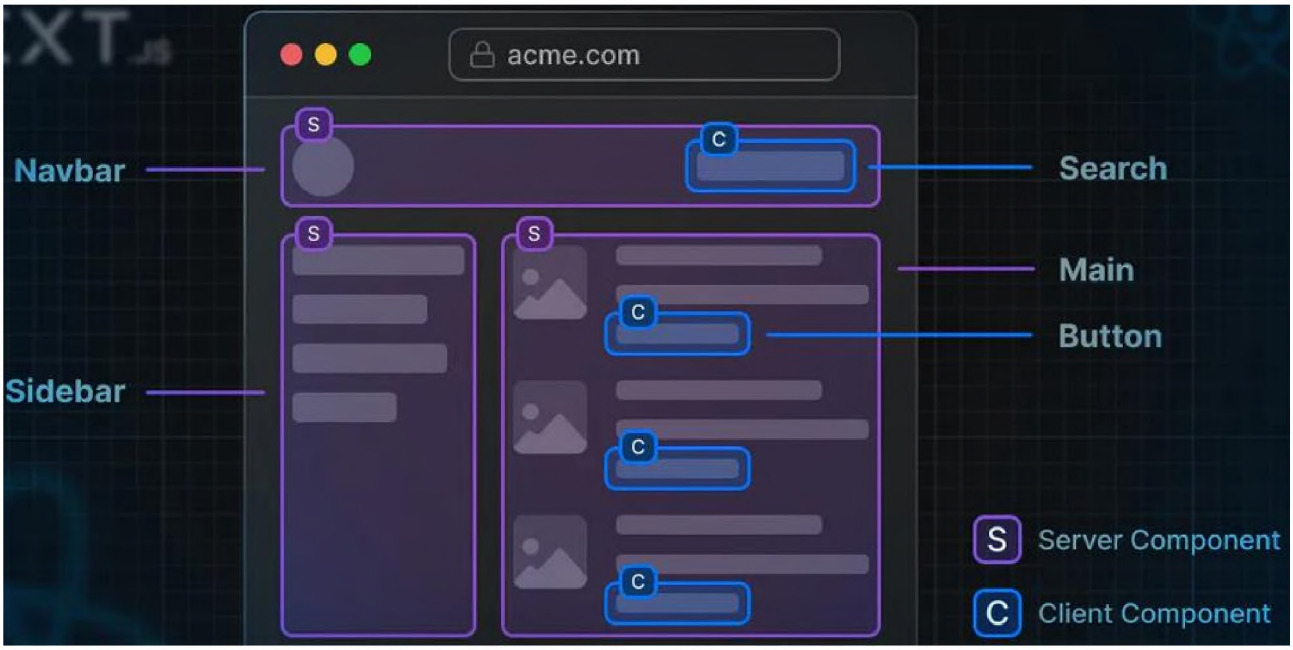

Рисунок 2.2 – Схема роботи React Server Components

Це значно зменшує час, необхідний для завантаження та рендерингу сторінок на клієнтському боці, що особливо помітно на мобільних пристроях та в умовах повільного інтернет-з'єднання.

Використання RSC також сприяє покращенню SEO, оскільки серверний рендеринг забезпечує повний HTML-контент, який може бути індексований пошуковими системами. Це дозволяє покращити видимість вебдодатків у пошукових результатах та підвищити їхню конкурентоспроможність.

Однією з ключових переваг React Server Components є можливість інтеграції з іншими серверними технологіями, такими як Next.js. Next.js надає інструменти для простого використання серверних компонентів, включаючи підтримку статичного та серверного рендерингу, маршрутизації та оптимізації продуктивності. Комбінування Next.js з React Server Components дозволяє створювати потужні та ефективні додатки, що швидко завантажуються і забезпечують плавний користувацький досвід. Одним з основних викликів при використанні RSC є необхідність управління станом додатку між сервером та

клієнтом. Для цього можуть використовуватися різні підходи, такі як збереження стану у глобальному сховищі, використання контексту React або локального стану компонентів. Також важливо забезпечити синхронізацію даних між клієнтом і сервером, що може вимагати використання механізмів обміну повідомленнями або періодичного оновлення даних [21].

Крім того, важливо враховувати питання безпеки та доступності даних при використанні RSC. Оскільки частина логіки додатку виконується на сервері, необхідно забезпечити захист від потенційних атак та несанкціонованого доступу до даних. Це може вимагати впровадження механізмів аутентифікації та авторизації, шифрування даних та регулярних аудитів безпеки.

Таким чином, використання React Server Components у поєднанні з традиційними методами обміну даними між клієнтом і сервером, такими як HTTP-запити, WebSocket та GraphQL, відкриває нові можливості для побудови високоефективних і надійних вебдодатків. Впровадження цих технологій дозволяє оптимізувати продуктивність, покращити користувацький досвід та забезпечити надійність і безпеку даних, що є ключовими аспектами сучасних вебдодатків.

2.4 Аналіз методів підтримки асинхронності та маршрутизації в зв'язці з використанням React та Node.js для створення плавно працюючого інтернет-магазину.

Інтернет-магазини в сучасному світі стали невід'ємною складовою електронної комерції, що стрімко розвивається. Вони надають споживачам можливість отримати доступ до різноманітних товарів та послуг у будь-який час та з будь-якого місця. Однак, в умовах конкуренції важливо мати ефективний та зручний інтерфейс для користувачів. Аналітики сайту Retail дослідили та узагальнили 5 найбільш значних проблем, що впливають на ефектвність функціонування більшості інтернет-магазинів [22]. На першому місці називають саме користувацький досвід, тобто можливість ефективно управляти

користувацьким інтерфейсом (user experience, UX-дизайн). Порушеннями є повільне завантаження графічних елементів, незрозуміла навігація, непродумані лінки тощо. До цього додається відсутність адаптивності, неперевірений контент, просто довгий шлях від замовлення до оформлення іоплати покупки.

Таким чином, головною проблемою при розробці вебдодатку є забезпечення плавності його роботи. Це означає, що сторінки магазину повинні завантажуватися швидко та без перерв, а користувач повинен мати можливість зручно навігуватися по сайту та здійснювати покупки без зайвих труднощів. При великій кількості товарів та відвідувачів це може стати викликом, особливо при використанні асинхронних запитів.

Для вирішення проблеми плавності роботи інтернет-магазину використовуються різні методи, зокрема асинхронність та маршрутизація. У даній роботі проводиться аналіз методів підтримки асинхронності та маршрутизації з використанням React та Node.js з метою створення оптимально працюючого інтернет-магазину.

Однією з основних переваг React є його компонентна архітектура, яка дозволяє розбити інтерфейс на невеликі, повторно використовувані частини - компоненти. Це спрощує розробку, тестування і підтримку коду .

Node.js – це середовище виконання JavaScript, побудоване на движку V8 (той самий, що й у браузері Google Chrome), яке дозволяє виконувати код JavaScript на сервері. Одними з ключових особливостей Node.js є його асинхронна та подієва модель програмування, яка дозволяє ефективно обробляти багатопотокові запити, а також є популярним вибором для розробки серверної частини вебдодатків через його швидкодію та можливість легко взаємодіяти з базами даних, файловою системою та іншими зовнішніми ресурсами. Він також підтримує велику кількість розширень (модулів), що дозволяє розробникам швидко додавати функціональність до своїх додатків [23].

Асинхронність у веброзробці використовується для забезпечення плавності роботи додатків. Плавність – особливий темін, показник, який комплексно описує ефекивну взаємодію користувача із вебзастосунком.

Оскільки вебзастосунки взаємодіють з серверами та базами даних через мережу, блокуючі операції можуть спричинити затримки, що негативно впливає на користувацький досвід. Використання асинхронних запитів дозволяє вебдодаткам продовжувати виконання інших операцій під час очікування відповіді від сервера (рис. 2.3) [24].

```
14  function App() {
15    const dispatch = useDispatch()
16    const [cartProductCount,setCartProductCount] = useState(0)
17
18    const fetchUserDetails = async()=>{
19      const dataResponse = await fetch(SummaryApi.current_user.url,{
20        method : SummaryApi.current_user.method,
21        credentials : 'include'
22      })
23
24      const dataApi = await dataResponse.json()
25
26      if(dataApi.success){
27        dispatch(setUserDetails(dataApi.data))
28      }
29  }
```

Рисунок 2.3 – Приклад використання async

Функція `fetchUserDetails` є асинхронною функцією, яка виконує запит до API для отримання деталей користувача та, у випадку успіху, оновлює стан додатку з отриманими даними. Ось покроковий опис роботи цієї функції:

1. Оголошення асинхронної функції:

const fetchUserDetails = async() => {

Функція оголошена як асинхронна (async), що дозволяє використовувати всередині неї оператор `await` для асинхронних операцій.

2. Виконання запиту до API:

const dataResponse = await fetch(SummaryApi.current_user.url, {

method: SummaryApi.current_user.method,

credentials: 'include' });

`fetch` виконує HTTP-запит до URL, який визначений в `SummaryApi.current_user.url`.

Метод запиту (наприклад, `GET` або `POST`) визначається через `SummaryApi.current_user.method`.

Параметр `credentials: 'include'` означає, що разом із запитом будуть відправлені дані аутентифікації, такі як куки.

Використання `await` призупиняє виконання функції до завершення запиту та отримання відповіді.

3. Отримання даних із відповіді: const dataApi = await dataResponse.json();

Використання `await` для парсингу JSON-відповіді з об'єкта `dataResponse`, що містить відповідь від сервера.

4. Перевірка успішності відповіді: if (dataApi.success) {

- Перевірка, чи була відповідь успішною, з використанням поля `success` з отриманих даних (`dataApi`). Якщо це поле є `true`, то запит був успішним.

5. Оновлення стану додатку:

dispatch(setUserDetails(dataApi.data));

Якщо запит успішний, функція “dispatch” викликає дію `setUserDetails` з даними користувача (`dataApi.data`), що оновлює стан додатку (можливо, у Redux-сторі).

Асинхронність у React, зокрема за допомогою Suspense, дозволяє компонентам чекати завантаження даних без блокування рендерингу інших частин інтерфейсу користувача (UI-дизайн), що забезпечує більш плавний користувацький досвід. Наприклад, під час завантаження даних може відображатися тимчасовий UI, а потім, коли дані будуть готові, відбувається плавне переключення на основний контент.

У React та Node.js існують різні методи для підтримки асинхронності:

1. Promise API та async/await: у Node.js та JavaScript Promise API та async/await є стандартними механізмами для роботи з асинхронним кодом (рис. 2.4). Вони дозволяють виконувати асинхронні операції без блокування потоку виконання.

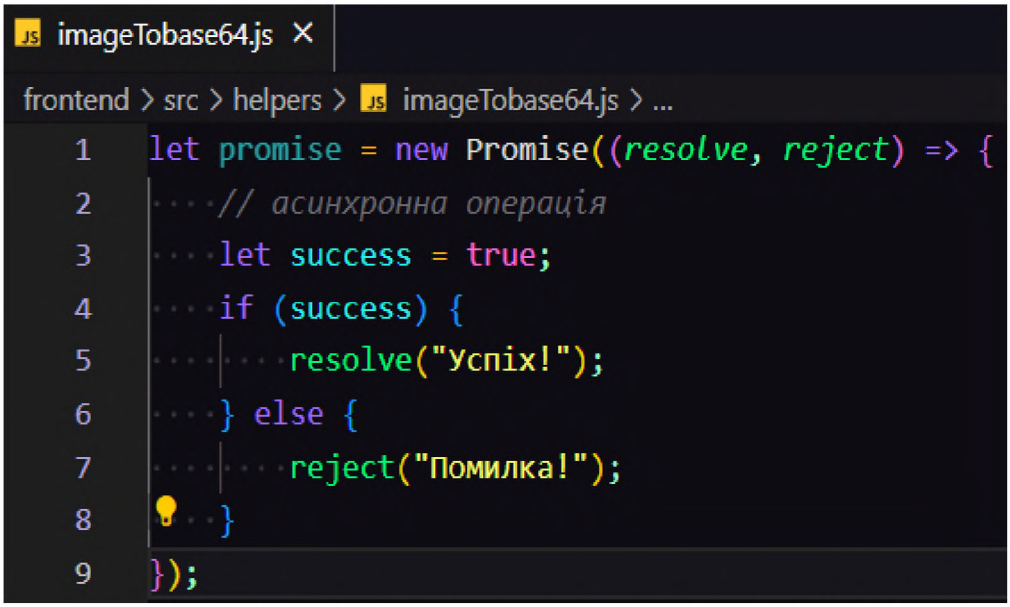

Рисунок 2.4 – Приклад використання promise API

2. Callback функції: це один з найпоширеніших методів у Node.js. Проте використання вкладених або неправильно синхронізованих callback-ів може призвести до так званого "callback hell", що ускладнює розуміння та підтримку коду [25].

3. Event Emitters: Node.js також підтримує механізм подій, який дозволяє створювати та обробляти події асинхронно.

Маршрутизація у веброзробці – це процес визначення шляху для запитів користувачів у вебдодатках. В інтернет-магазинах маршрутизація дозволяє користувачам здійснювати навігацію між різними сторінками, категоріями товарів та оформлювати замовлення.

У React та Node.js маршрутизацію можна забезпечити за допомогою таких інструментів:

1. React Router: бібліотека для React, яка дозволяє визначати маршрути та управляти ними в залежності від URL. Маршрутизація в React, здійснювана за допомогою React Router, дозволяє створювати динамічні односторінкові додатки, які реагують на зміни URL без повного перезавантаження сторінки, що також сприяє плавності роботи додатку.

2. Express.js: популярний фреймворк для Node.js, який дозволяє легко створювати серверні додатки. Він також має вбудовану підтримку маршрутизації. Node.js, у свою чергу, забезпечує потужний бекенд, який може

асинхронно обробляти запити, що дозволяє швидко відповідати на дії користувачів і зменшує час відгуку сервера.

Методи підтримки асинхронності та маршрутизації мають вирішальне значення для розробки плавно працюючого інтернет-магазину. Використання оптимальних методів асинхронного програмування та ефективної маршрутизації дозволяє забезпечити швидку відповідь сервера, зручну навігацію користувача та ефективну роботу додатка в цілому [26].

Аналізуючи ці методи, розробники можуть виявити можливі проблеми з продуктивністю та оптимізувати їх для забезпечення кращого користувацького досвіду та підвищення конкурентоспроможності інтернет-магазину.

Отже, дослідження методів підтримки асинхронності та маршрутизації виявляється критичним для успішної розробки інтернет-магазину на базі React та Node.js. Цей аналіз допомагає забезпечити ефективну відповідь сервера, зручну навігацію користувача та загальну плавність роботи додатка [27].

РОЗДІЛ 3
РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ КОМП'ЮТЕРНОЇ ТЕХНІКИ

3.1 Результати застосування NPM, Node.js, Express та JSX в розробці інтернет-магазину

Інтернет-магазин був розроблений за допомогою IDE Visual Studio Code, одного з найпопулярніших та найпотужніших редакторів коду. VSCode забезпечив зручне та ефективне середовище для написання, налагодження та тестування коду, що значно прискорило процес створення вебзастосунку.

Однією з ключових технологій, використаних у розробці інтернет-магазину, став React – бібліотека для побудови інтерфейсів користувача. React дозволяє створювати динамічні та інтерактивні вебзастосунки, забезпечуючи високу продуктивність та зручність в управлінні компонентами.

Проте, для роботи з React необхідно мати встановлений Node.js. Node.js є серверною платформою, яка дозволяє запускати JavaScript-код поза браузером. Версію з довгостроковою підтримкою (LTS) було завантажено та встановлена з офіційного вебсайту Node.js. LTS була обрана через її стабільність та планований цикл випусків з розширеною системою підтримки і оновленями безпеки. Завдяки Node.js, було отримано можливість використовувати npm (Node Package Manager) – інструмент для управління пакунками та залежностями, що є невід'ємною частиною розробки на React. npm дозволяє легко встановлювати та керувати бібліотеками, потрібними для проекту, що значно спрощує процес розробки та забезпечує можливість використання останніх версій необхідних інструментів та бібліотек [28].

Для початку роботи з програмою було використано команду "npm init", під час якої було вказано назву програми, версію, опис проекту та автора. Це призвело до створення нового файлу «package.json» з вказаною інформацією.

Після встановлення Node.js був встановлений фреймворк Express версії 4.16.1 за допомогою команди: npm install express –save -g.

Крім цього, для створення стандартної структури програми з найбільш часто використовуваними проміжними програмними засобами був використаний Express-генератор. Для його встановлення в програмі була використана наступна команда: npm install express-generator -g.

Після цього новий експрес-додаток згортається за допомогою команди express. При цьому завантажуються мінімально необхідні файли та теки для програми. Після переходу до нещодавно створеної папки “backend” всі необхідні модулі, вже включені до файлу package.json, встановлюються за допомогою команди npm install. Різні версії npm, Node.js і Express, які використовуються при розробці програми (рис. 3.1).

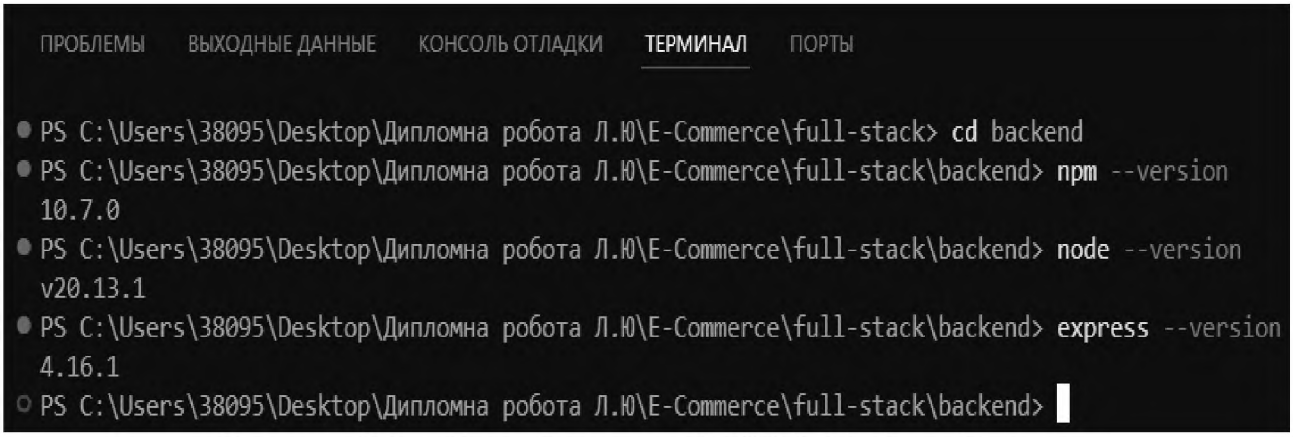

Рисунок 3.1 – Перевірка версій NPM, Node та Express

Під час розробки програми було використано Node.js версії 20.13.1, останню версію з довгостроковою підтримкою (LTS) на той момент, як показано на рис. 3.1. Для керування залежностями використовувалися npm версії 10.7.0, а також фреймворк Express версії 4.16.1.

3.2. Додаткові пакети Node

Окрім основних пакетів та інструментів, які були описані раніше, в проекті було встановлено багато інших пакетів, які забезпечують додаткові функції та можливості програми. Ці додаткові пакети включають бібліотеки для розширення функціональності, підвищення продуктивності та покращення зручності використання програми. Встановлення та налаштування цих пакетів

значно сприяли створенню більш гнучкого та потужного програмного забезпечення.

У реалізації проекту використовувалася аутентифікаційна платформа Auth0, що є рішенням з автентифікації та управління доступом, розробленим для застосування у вебпрограмах та мобільних додатках. Auth0 забезпечує високий рівень безпеки шляхом використання сучасних протоколів аутентифікації, включаючи OAuth та OpenID Connect, та відповідає стандартам безпеки в інтернеті [29]. Його використання дозволило забезпечити надійний механізм ідентифікації користувачів та керування доступом до функцій програми. Був встановлений командою “npm install @auth0/auth0-react” (рис. 3.2).

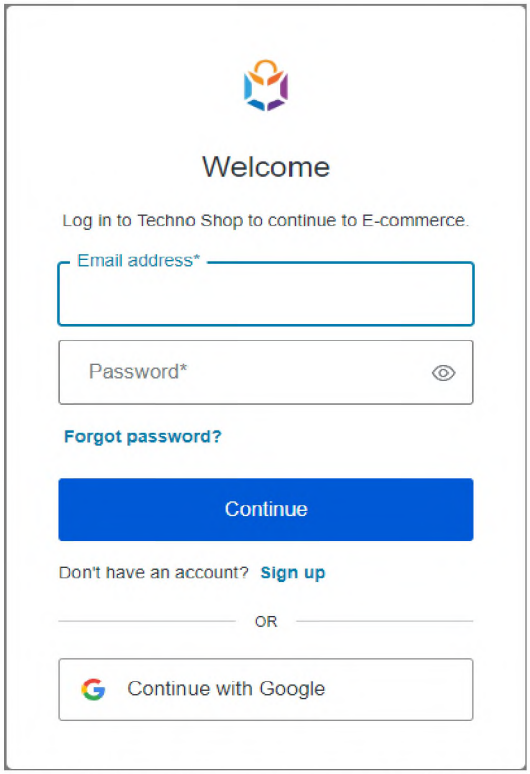

Рисунок 3.2 – Форма реєстрації та логіну

Бачення свого статусу у вебдодатку для автентифікації користувачів має безперечну корисність:

– надійність і безпека: інформування користувача про його поточний статус автентифікації допомагає уникнути недоречних дій, забезпечуючи правильне використання системи. Наприклад, відображення інтерфейсу для

увійти або вийти з раніше аутентифікованого облікового запису зменшує ризик несанкціонованого доступу до даних.

– користувацький досвід: візуалізація статусу автентифікації полегшує взаємодію користувача з додатком. Це дозволяє забезпечити зручність та зрозумілість інтерфейсу, що впливає на загальний позитивний враження від використання продукту.

– керування обліковим записом: завдяки відображенню відповідних елементів інтерфейсу, користувачі можуть легко керувати своїм обліковим записом.

Представлений уривок програмного коду є фрагментом вебдодатка, який використовується для відображення інтерфейсу автентифікації користувачів (рис. 3.3).

```
{
          isAuthenticated ?
        <div className='user'>
          <div className='icon'>
            <CiLogout />
          </div>
            <div className='btn'>
              <button onClick={() => logout({
logoutParams: { returnTo: window.location.origin
 } })}>Вийти</button>
            </div>
        </div>
        :
        <div className='user'>
          <div className='icon'>
            <FiLogIn />
          </div>
          <div className='btn'>
            <button onClick={() =>
loginWithRedirect()}>Увійти</button>
          </div>
        </div>
        }
```

Рисунок 3.3 – Фрагмент коду

Умовний оператор використовується для перевірки статусу аутентифікації, після чого відображаються відповідні елементи інтерфейсу. У випадку, якщо користувач вже аутентифікований, відображається іконка та кнопка для виходу з системи. У протилежному випадку, коли користувач не

аутентифікований, відображається інтерфейс для початку процесу автентифікації за допомогою відповідної іконки та кнопки (рис 3.4 – 3.5)

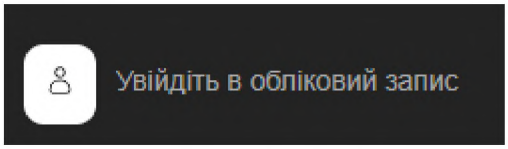

Рисунок 3.4 – Статус користувача у режимі гость

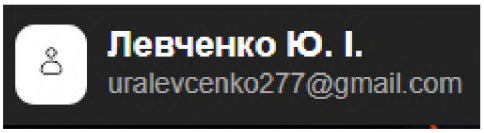

Рисунок 3.5 – Користувач увійшов в обліковий запис

Firebase – це комплексний набір інструментів для розробників мобільних та вебдодатків, створений компанією Google. Він дозволяє розробникам швидко створювати додатки з високою продуктивністю та надійністю завдяки інтегрованим інструментам і сервісам. Тому було вирішено обрати саме цей інструмент для Realtime Database. Firebase Hosting забезпечує швидкий та безпечний вебхостинг для розгортання вебдодатків [30]. Його використовують для різних видів додатків, від невеликих прототипів до великих комерційних проектів. Для цього створювався обліковий запис на вебсайті Firebase (рис. 3.6), після чого був створений кластер для нового проекту.

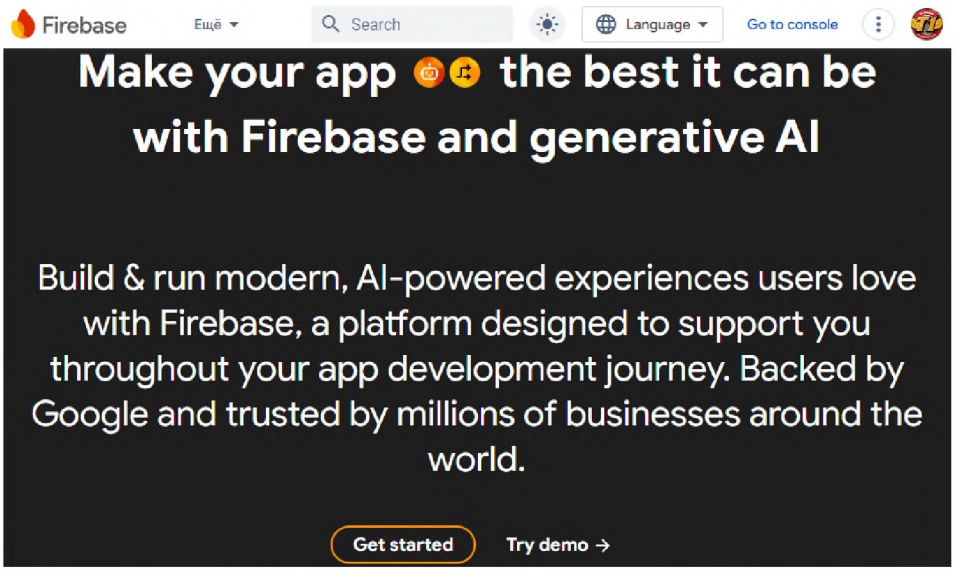

Рисунок 3.6 – Головна сторінку Firebase

Також був створений користувач з правами на читання і запис до бази даних, який використовувався в додатку Node.js. Для забезпечення взаємодії програми з сервером, поточна IP-адреса комп'ютера була додана до списку дозволених IP-адрес. Кластер Firebase показує область, де розташовано сервер, операції читання і записи, кількість активних з'єднань на сервері, а також розмір бази даних, як показано на рис. 3.7.

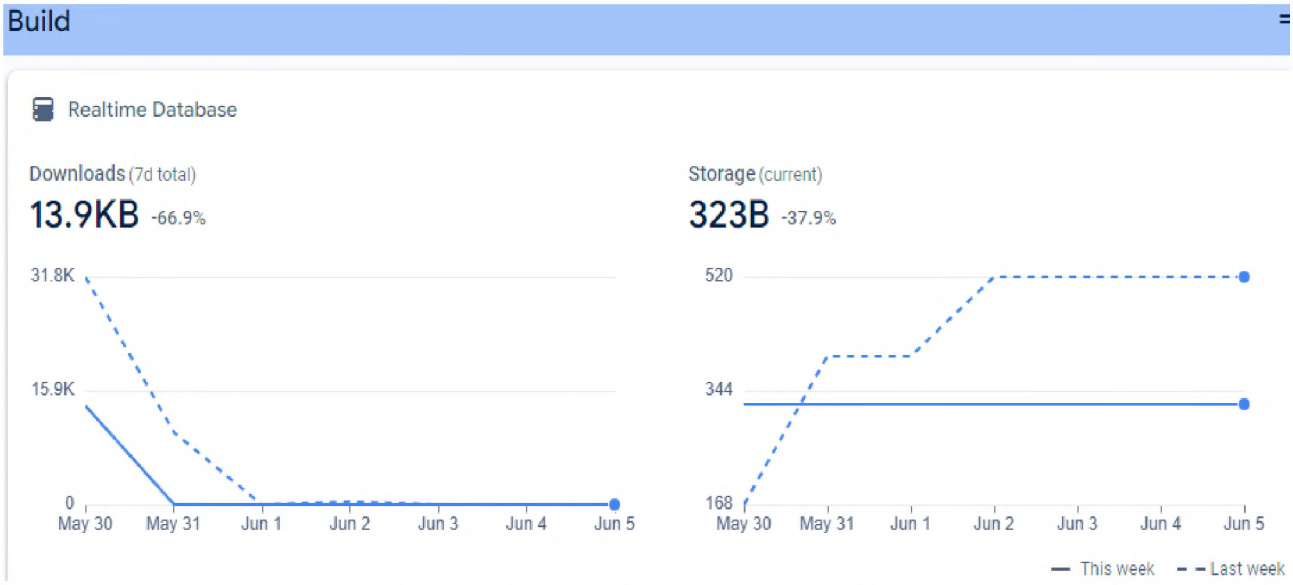

Рисунок 3.7 – Огляд кластера Firebase

Наступним кроком буде застосування Mongoose для спрощення взаємодії з сервером та базою даних. Остання доступна версія Mongoose, на момент встановлення, версія 8.3.4. Її встановлено за допомогою команди "npm install mongoose –save", а зв'язок із базою даних створено, використовуючи URL, ім'я користувача та пароль, створені під час налаштування Firebase. Крім того, були створені необхідні схеми та моделі, про які буде згадано далі.

Для відображення динамічного вмісту на HTML-сторінці обрано вбудований механізм шаблонів JSX (JavaScript XML) – це синтаксичне розширення мови JavaScript, яке було використано у проекі з React. JSX дозволяє писати код, що виглядає як HTML, але при цьому інтегрується з JavaScript, роблячи процес створення користувацьких інтерфейсів більш інтуїтивним та ефективним. Основною особливістю JSX є використання розмітки схожу на

HTML, всередині JavaScript-коду. Це полегшує читання та написання компонентів інтерфейсу [31].

3.3 Програмна логіка та безпека

Середовище розробки, всі пакети Node.js та інструменти були завантажені та встановлені. Після налаштування цієї базової структури для розробки програми розпочався процес розробки та написання коду. Остаточна структура файлової системи проекту показана на рис. 3.8. Вебдодаток складається з кількох папок і файлів, що використовуються для реалізації її логіки. Щоб підтримувати простоту та зрозумілість структури, файли розподілені по різних папках. Моделі, контролери та представлення програми зберігаються в окремих теках відповідно до архітектури MVC.

Файлова структура React додатку, показана на зображенні, виглядає наступним чином. Головна директорія проекту називається “techno-shop”. В ній містяться кілька важливих папок і файлів. Папка “build” містить зібрані (production) файли додатку. Створилась через команду “npm run build”, використовується в екосистемі Node.js для запуску процесу побудови проєкту. Ця команда є частиною системи npm scripts, яка дозволяє розробникам визначати та запускати власні скрипти для автоматизації різних завдань у процесі розробки

Папка “node_modules” містить усі встановлені залежності проекту, керовані npm. Папка “public” зберігає статичні файли, такі як HTML, CSS, JavaScript файли та інші ресурси (зображення, шрифти), необхідні для роботи додатку на стороні клієнта. Папка “src” є головною папкою з вихідним кодом додатку. В ній розташовані кілька файлів і папок: “comp” (папка з компонентами React додатку), “App.js” (основний компонент додатку React), “index.js” (вхідний файл, який рендерить React додаток у DOM), “reportWebVitals.js” (файл для вимірювання продуктивності додатку) та “setupTests.js” містить конфігурації для тестування, забезпечуючи належне середовище для написання і виконання тестів, що допомагає підтримувати високу якість коду.

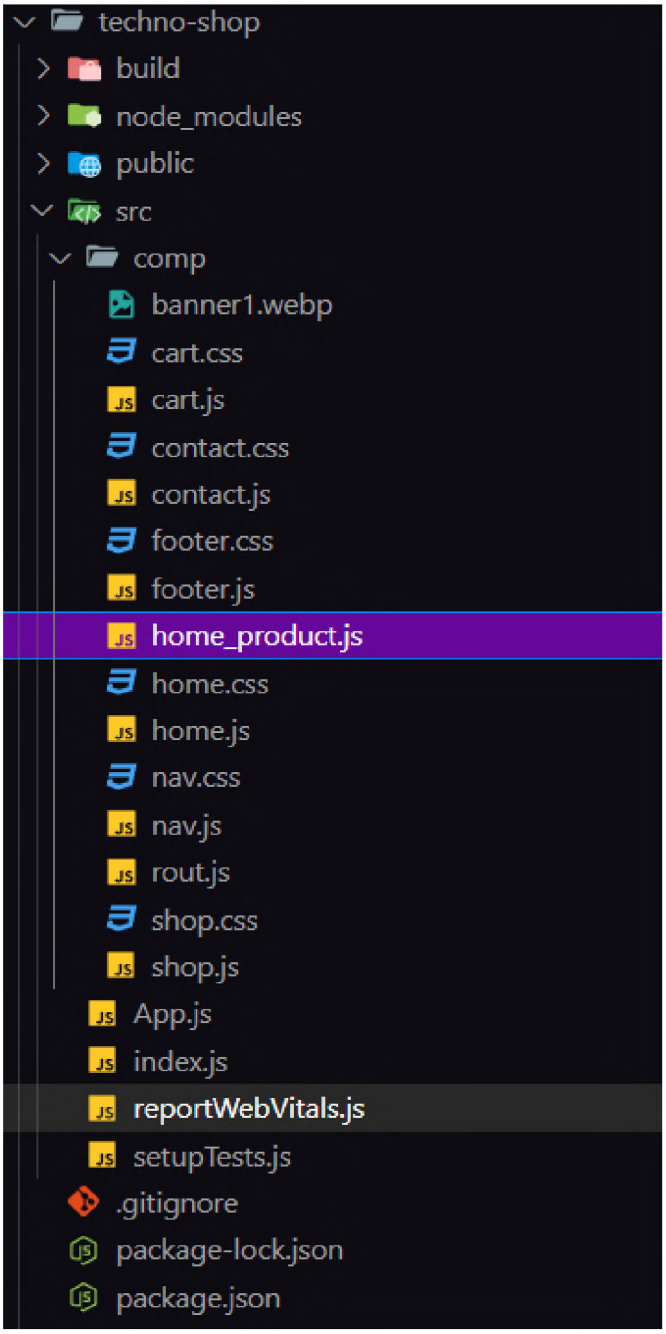

Рисунок 3.8 – Файлова структура програми

Також у кореневій директорії проекту знаходяться кілька важливих файлів. Файл ".gitignore" визначає, які файли та папки не повинні відстежуватися системою контролю версій Git. Файл "package-lock.json" фіксує версії всіх встановлених npm-пакетів для забезпечення однакової збірки серед усіх середовищ. Файл "package.json" містить метадані проекту, включаючи залежності, скрипти та іншу інформацію, що стосується проекту. Файл "README.md" містить інформацію про проект, інструкції щодо встановлення,

використання тощо. Ця структура забезпечує організованість коду, зручність у підтримці та масштабуванні додатку.

Файл App.js є основним компонентом React для додатка, який імпортує необхідні бібліотеки та компоненти, такі як React, useState для керування станом, компоненти навігації (Nav), футера (Footer), маршрутизації (BrowserRouter з react-router-dom), основний компонент маршрутизації (Rout) та список продуктів (Homeproduct) (рис. 3.9).

```
1  import React, { useState } from 'react';
2  import Nav from './comp/nav';
3  import Footer from './comp/footer';
4  import { BrowserRouter } from
   'react-router-dom';
5  import Rout from './comp/rout';
6  import Homeproduct from
   './comp/home_product';
7
8  const App = () => {
9    const [cart, setCart] = useState([]);
10   const [search, setSearch] = useState(
   '');
11
12   const [shop, setShop] = useState(
   Homeproduct);
13   const Filter = (x) => {
14     const catefilter = Homeproduct.
   filter((product) => {
15       return product.cat === x;
16     });
17     setShop(catefilter);
18   };
```

Рисунок 3.9 – Фрагмент коду App.js

Компонент App визначається за допомогою функціонального компонента, в якому використовуються хуки useState для створення трьох станів: cart для збереження продуктів у кошику, search для збереження тексту пошуку та shop для збереження відфільтрованих продуктів.

Для фільтрації продуктів за категоріями використовується функція Filter, яка фільтрує продукти за категоріями та оновлює стан shop відповідно до вибраної категорії (рис. 3.10). Щоб відобразити всі продукти без фільтрації, використовується функція allcatefilter, яка встановлює всі продукти у стан shop.

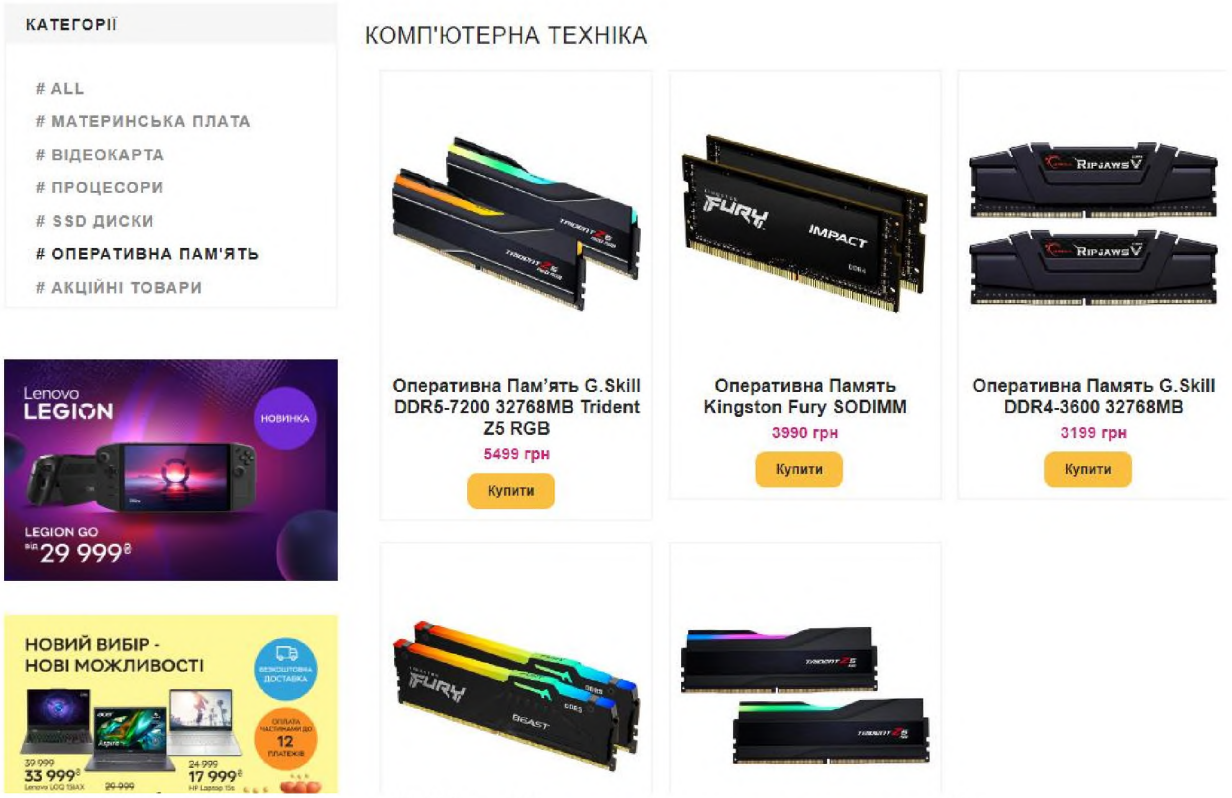

Рисунок 3.10 – Фільтрація товару по кагерої

Функція searchProduct виконує пошук продуктів за назвою, використовуючи текст, введений у поле пошуку (рис. 3.11 – 3.12). Якщо поле пошуку порожнє, користувачеві видається попередження про необхідність введення назви товару, і у стан shop встановлюються всі продукти. Якщо текст пошуку не порожній, відбувається фільтрація продуктів за назвою, яка містить введений текст, і оновлюється стан shop.

```
<div className='search_box'>
    <input
        type='text' value={search} placeholder='Введіть назву товару'
        onChange={(e) => setSearch(e.target.value)}
    ></input>
    <button onClick={searchProduct}>
        <AiOutlineSearch />
    </button>
</div>
```

Рисунок 3.11 – Фрагмент коду searchproduct

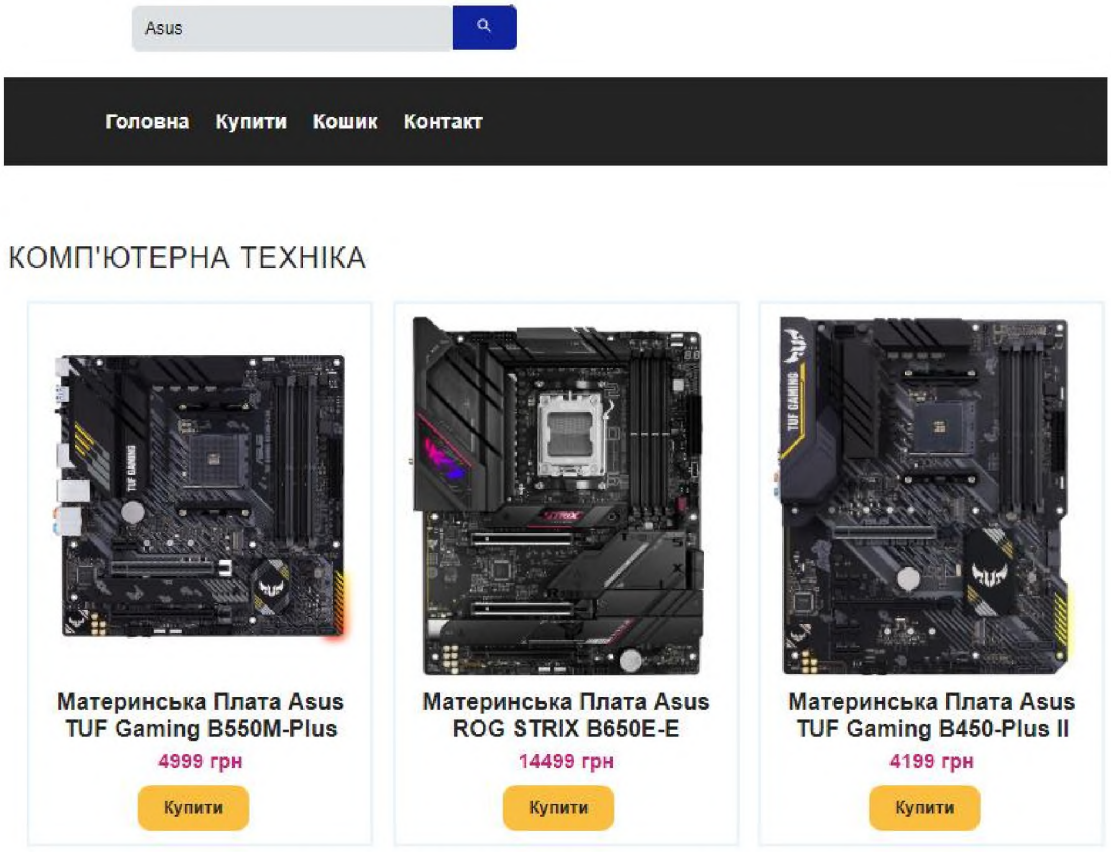

Рисунок 3.12 – Пошук товару за назвою

Для додавання продуктів у кошик використовується функція addtocart, яка перевіряє наявність продукту у кошику. Якщо продукт вже є в кошику, користувачеві видається попередження про те, що товар вже був доданий. Якщо продукту немає у кошику, він додається до кошика з кількістю, встановленою на 1, та оновлюється стан cart(рис. 3.13 – 3.14).

```
const addtocart = (product) => {
  const exist = cart.find((x) => {
    return x.id === product.id;
  });
  if (exist) {
    alert('Цей товар вже був доданий у кошик');
  } else {
    setCart([...cart, { ...product, qty: 1 }]);
    alert('Додано до кошика');
  }
};
console.log(cart);
```

Рисунок 3.13 – Фрагмент коду addtocart

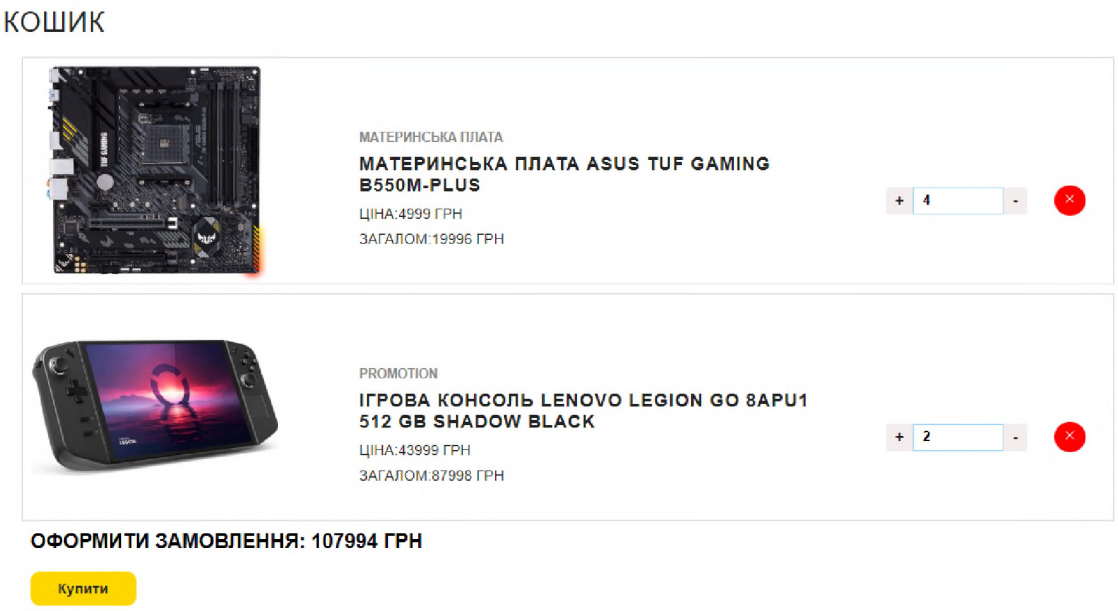

Рисунок 3.14 – Розділ “Кошик”

Основна частина компонента App рендерить компоненти JSX, включаючи маршрутизатор (BrowserRouter), навігаційний бар (Nav) з пропсами для пошуку, основний компонент маршрутизації (Rout) з пропсами для керування кошиком, фільтрації та додавання товарів, а також футер (Footer). Наприкінці компонент App експортується для використання в інших частинах додатка.

Таким чином, компонент App є головним контейнером для інших компонентів та логіки додатка, керуючи станом, фільтрацією, пошуком та додаванням товарів до кошика.

Об'єкт "Homeproduct" представляє собою масив, що містить інформацію про різноманітні комп'ютерні комплектуючі та аксесуари (рис. 3.15). Кожен елемент масиву є об'єктом, що описує один продукт із наступними властивостями:

– id: унікальний ідентифікатор продукту;
– name: назва продукту;
– price: ціна продукту;
– image: шлях до зображення продукту;
– cat: категорія продукту (наприклад, материнська плата, SSD диск, оперативна пам'ять, процесор, відеокарта);

– type: тип продукту, який може мати значення, такі як "new", "featured", "top" або бути порожнім;

– text: опис продукту, що містить технічні характеристики та іншу інформацію.

Наприклад, перший об'єкт у масиві описує материнську плату "Asus TUF Gaming B550M-Plus" з ціною 4999 гривень, зображенням, категорією "материнська плата", типом "new" та докладним описом технічних характеристик.

Масив містить різноманітні продукти, такі як материнські плати, SSD диски, оперативна пам'ять, процесори, відеокарти, а також ігрову консоль. Кожен продукт має свої специфічні характеристики, відмінні ціни та призначення, що допомагає користувачам знайти та порівняти необхідні компоненти для своїх комп'ютерних систем. Масив експортується для використання в інших частинах додатка для відображення продуктів на головній сторінці та в пошукових результатах.

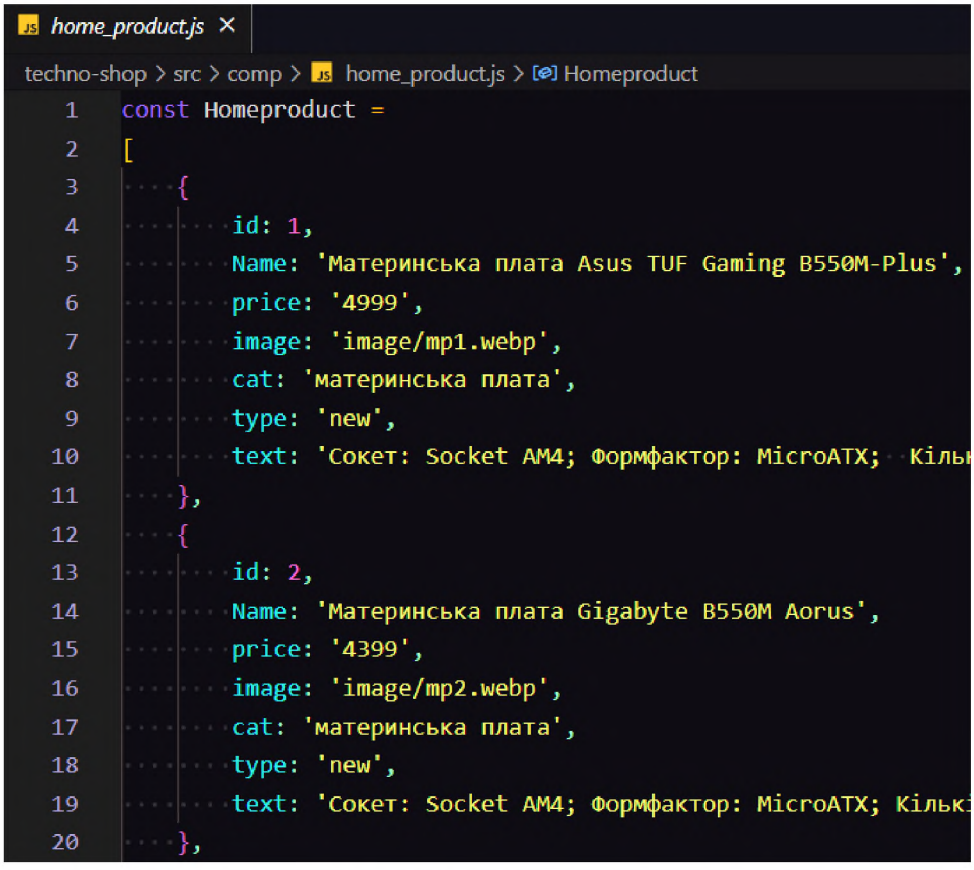

Рисунок 3.15 – Схема моделі товару

Для зворотного зв’язку було розроблено компонент "Contact" у React, який представляє собою форму (рис. 3.16-3.17). Компонент використовує хук useState для управління станом користувача, що включає чотири поля: ім'я, електронна пошта, тема та повідомлення. При кожній зміні вхідних даних функція data оновлює відповідне значення у стані.

```
const send = async (e) =>
    {
        const {Name, email, subject, Message} = user
        e.preventDefault()
        const option = {
            method: 'POST',
            headers: {
                'Content-Type': 'aplication/json'
            },
            body: JSON.stringify({
                Name, email, subject, Message
            })
        }
```

Рисунок 3.16 – Фрагмент коду contact.js

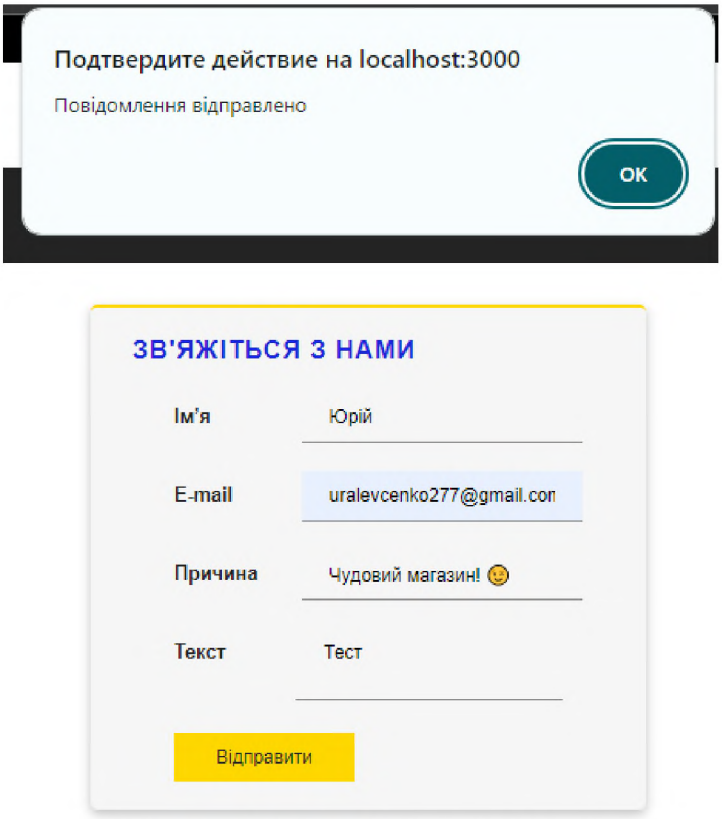

Рисунок 3.17 – Відправка форми

Для кожного поля введення визначені обробники подій onChange, які викликають функцію data, щоб оновити стан компонента відповідно до змін введених користувачем. Кнопка "Відправити" викликає функцію send, яка запобігає стандартній відправці форми, збирає дані зі стану та надсилає їх у форматі JSON на сервер за допомогою методу POST через API fetch. Якщо запит успішний, користувач отримує сповіщення про успішну відправку повідомлення, інакше – повідомлення про помилку.

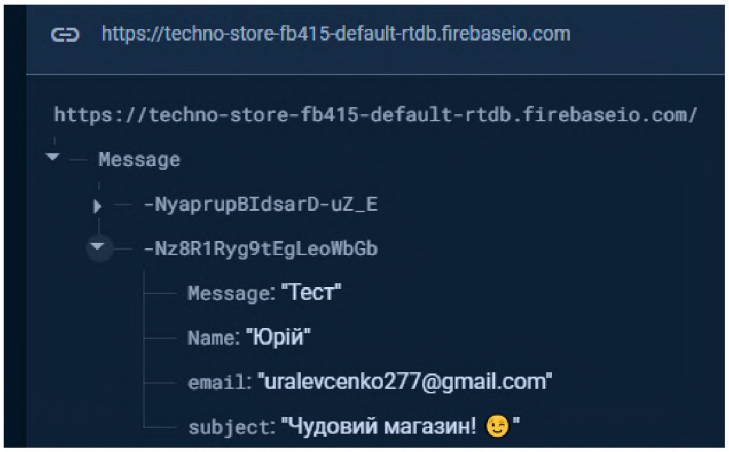

Рисунок 3.18 – Відображення повідомлення в Firebase

Безпека є надзвичайно важливим аспектом для всіх вебдодатків. Уразливість вебдодатків зростає щодня через стрімке збільшення кількості кібератак та порушень даних. Хакери використовують різноманітні методи, такі як фішинг і злом, щоб отримати доступ до облікових даних для входу та інформації про кредитні картки. Крім того, незахищені маршрути полегшують зловмисникам доступ до додатків. Отже, сайти мають бути надійно захищені. Особливо це стосується вебдодатків для електронної комерції, які здійснюють велику кількість фінансових операцій під час купівлі-продажу товарів, тому безпека повинна бути найвищим пріоритетом. Навіть незначне порушення безпеки може призвести до значних збитків для компанії та її клієнтів [32].

Автентифікація - це процес верифікації користувача, який намагається отримати доступ до програми. Користувач проходить верифікацію на сервері за допомогою різних методів, таких як проста форма входу, дані онлайн-банкінгу, відбитки пальців, розпізнавання голосу або обличчя, а також сканування сітківки

ока. Найпоширеніший метод автентифікації - це форма входу [33]. Користувач створює обліковий запис із базовою інформацією, такою як електронна пошта і пароль, який шифрується і зберігається в базі даних (рис. 3.19). Під час входу в систему сервер звіряє надану користувачем інформацію з тією, що зберігається в базі даних. Після успішної перевірки для користувача створюється сеанс, який зберігається у браузері клієнта як файл cookie і надсилається з кожним наступним запитом, зробленим користувачем.

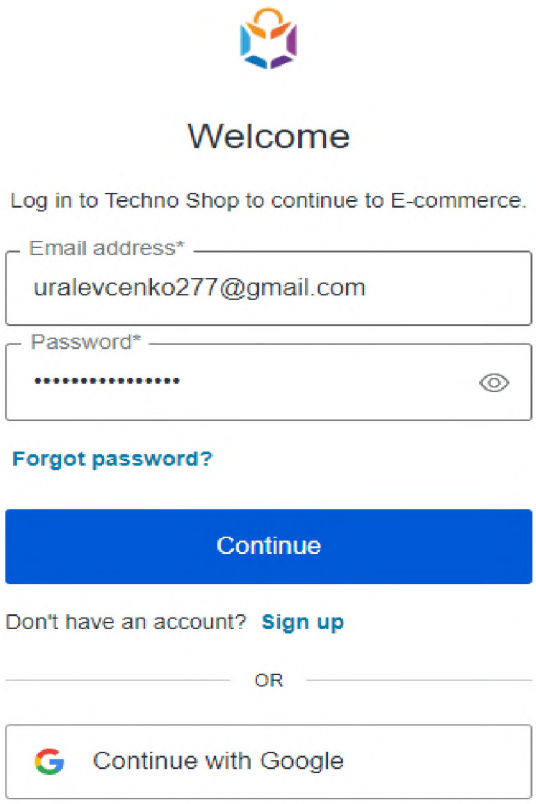

Рисунок 3.19 – Форма аутентифікації

Node.js складається з вбудованих основних модулів та інших сторонніх модулів. Для реалізації різних функціоналів, які недоступні в Node.js за замовчуванням, ці модулі імпортувалися в додаток відповідно до потреб. Для реалізації одного модуля в додатку було потрібно декілька інших модулів, від яких він залежав. Збільшення кількості сторонніх модулів підвищило вразливість додатка до атак та витоків даних.

Сторонні пакети ефективно управлялися за допомогою npm, менеджера пакетів, що входить до складу Node.js. npm автоматично перевіряв всі встановлені бібліотеки. Крім того, команда "npm audit" перевіряла наявність

шкідливих пакетів серед встановлених бібліотек і перелічувала їх. Ще один сторонній інструмент, "snyk", підвищував безпеку додатка, перевіряючи всі відомі вразливості у списку залежностей. Його було встановлено командою "npm install -g snyk" і проект було перевірено за допомогою команди "snyk test" (рис. 3.20) [34].

```
PS C:\Users\38095\Desktop\diplomna robota\techno-shop> snyk test

Testing C:\Users\38095\Desktop\diplomna robota\techno-shop...

Organization:      uralevcenko277
Package manager:   npm
Target file:       package-lock.json
Project name:      techno-shop
Open source:       no
Project path:      C:\Users\38095\Desktop\diplomna robota\techno-shop
Licenses:          enabled

✓ Tested 1278 dependencies for known issues, no vulnerable paths found.
```

Рисунок 3.20 – Тестування проекту за допомогою snyk

Команда «snyk test» була запущена в інструменті командного рядка, де перераховані всі вразливості у зовнішніх залежностях програми, як показано на рис. 3.20. Проект не виявив уразливість.

Розроблений сайт було загружено на хостинг ukraine.com.ua, бо вони пропонують заходи для забезпечення захисту своїх хостингових послуг [35]. Вони забезпечують захист від розподілених атак на відмову в обслуговуванні (DDoS), що допомагає підтримувати стабільну роботу сайтів навіть під час високого навантаження або спроб злому. Також було надано SSL-сертифікат, який забезпечують шифрування даних між користувачем і сервером, що захищає інформацію від перехоплення та несанкціонованого доступу [36]. Вони здійснюють регулярне резервне копіювання даних, що дозволяє відновити інформацію у випадку втрати або пошкодження. Використання систем виявлення та запобігання вторгненням допомагає вчасно виявляти та блокувати спроби несанкціонованого доступу до серверів. Постійнодіючий моніторинг безпеки дозволяє оперативно виявляти і реагувати на потенційні загрози.

Регулярне оновлення програмного забезпечення та систем управління контентом (CMS) з метою закриття відомих вразливостей та підтримання високого рівня безпеки. Використання антивірусного програмного забезпечення для захисту серверів від шкідливого програмного забезпечення та вірусів. Ці заходи допомагають забезпечити надійний захист сайту, хостинг яких здійснюється на платформі ukraine.com.ua.

3.4 Економічна оцінка ефективності розробки інтернет-магазину

Розробка інтернет-магазину є стратегічно важливим кроком для бізнесу, що прагне розширити свою присутність в онлайн-просторі та збільшити обсяг продажів. В даному дослідженні буде проведено економічну оцінку ефективності розробки інтернет-магазину, який реалізовано за допомогою технологій React, Auth0 для аутентифікації, Firebase для зберігання даних та CSS для стилізації. Хостинг та доменне ім'я було зареєстровано на платформі ukraine.com.ua з доменом com.ua.Основні елементи розробки:

– функціонал пошуку товарів: реалізація інтерфейсу для пошуку товарів;

– фільтри пошуку товарів: додавання можливості фільтрації результатів пошуку за різними критеріями;

– аутентифікація за допомогою auth0: забезпечення безпеки користувацьких даних та управління доступом;

– форма зворотного зв'язку з використанням firebase: збір та обробка відгуків від клієнтів;

– стилізація за допомогою css: забезпечення привабливого та зручного користувацького інтерфейсу.

Розробник: Один фахівець; тривалість розробки: 20 годин

Економічні розрахунки:

1. Витрати на розробку:

– середня годинна ставка розробника становить 20 доларів;

– загальні витрати на оплату праці: 20 годин * 20 доларів/год = 400 доларів;

– вартість хостингу на ukraine.com.ua: 50 доларів на рік;

– вартість домену com.ua: 15 доларів на рік;

– загальні витрати на хостинг та домен: 50 доларів + 15 доларів = 65 доларів.

Загальні витрати на розробку: 400 доларів (оплата праці) + 65 доларів (хостинг та домен) = 465 доларів.

2. Очікувані доходи

Припустимо, що інтернет-магазин буде приносити прибуток від продажу товарів та послуг. Для спрощення розрахунків візьмемо середній місячний дохід у 500 доларів. Отже розрахуємо річний дохід: 500 доларів/місяць * 12 місяців = 6000 доларів.

3. Економічна ефективність

Для оцінки економічної ефективності використовуємо показник рентабельності інвестицій ROI (Return on Investment):

ROI = ((Річний дохід – Загальні витрати)/Загальні витрати) * 100%.

Підставимо значення: ROI = ((6000 – 465)-465) *100% ≈ 1189.25%

Розробка інтернет-магазину на React з використанням актуальних технологій є економічно ефективною інвестицією. Витрати на розробку та хостинг склали 465 доларів, тоді як очікуваний річний дохід може досягати 6000 доларів, що забезпечує високий рівень рентабельності інвестицій (1189.25%).

Ефективність цього проекту обґрунтована не лише фінансовими показниками, але й потенційними перевагами для бізнесу, такими як розширення клієнтської бази, підвищення лояльності клієнтів та збільшення обсягів продажу.

ВИСНОВКИ

Головною метою наукової роботи було створення інтернет-магазину комп'ютерної техніки, дослідження повних стекових фреймворків JavaScript та розробка прототипу вебдодатка на їх основі. Було вивчено різні аспекти використання таких фреймворків і платформ, як Node.js, Express та Firebase. Розглянуто переваги та недоліки розробки вебдодатків з використанням цих технологій і проведено порівняння з іншими аналогічними технологіями. Окрім цього, визначено швидкість, продуктивність і рівень безпеки інтернет-магазину.

Досвід, отриманий під час наукової роботи, підтвердив, що бібліотека React, написана мовою JavaScript, є найкращою для розробки вебдодатків. JavaScript надає всі необхідні компоненти для розробки додатків: Node.js для серверної сторони, React та Express для клієнтської сторони, а також Firebase як базу даних. Використання Node.js дозволяє створювати додатки з великою швидкістю та швидким циклом розробки. Крім того, він найкраще підходить для розробки високопродуктивних і масштабованих вебдодатків. Велика кількість бібліотек і пакетів Node.js допомагає розширити можливості цієї платформи без особливих труднощів. З урахуванням постійного додавання нових функцій у JavaScript та щорічних випусків нових версій ECMAScript, JavaScript безумовно є технологією майбутнього.

Однак, JavaScript не є найкращим рішенням для всіх типів додатків. Node.js не може ефективно обробляти важкі обчислення та обробку даних на серверній стороні. Тому високотехнологічні підприємства, що використовують машинне навчання або складні алгоритми, не можуть повністю покладатися на цю технологію. Проте завдяки активній спільноті розробників JavaScript та відкритості його коду, Node.js продовжує розвиватися і вдосконалюватися.