ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістр

на тему: «**Методологія застосування нейропомічника для розробки програмного коду вебконтенту**»

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи та технології
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти магістр
групи 126ІСТ_мд_2023
Омельченко Денис Вікторович
Керівник: Слюсар Вадим Іванович
Рецензент: Муравльов Володимир Вячеславович

Полтава – 2024 року

ВСТУП

Актуальність даної кваліфікаційної роботи обумовлена стрімким розвитком інформаційних технологій і штучного інтелекту. Нейропомічники, засновані на великих мовних моделях, як ChatGPT і Gemini, стають незамінними інструментами для підтримки розробників у прискоренні та спрощенні рутинних завдань, таких як написання, перевірка та тестування коду. Їх використання сприяє підвищенню продуктивності, зменшенню кількості помилок і вдосконаленню процесів веброзробки, що робить тему роботи актуальною та важливою для сучасної індустрії програмного забезпечення.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в межах науково-дослідної ініціативної тематики «Організаційно-методологічні аспекти впровадження інформаційно-комунікаційних систем і технологій в управлінні діяльністю сучасних організацій та підприємств за умов переходу до цифрової економіки» (ДРН 0123U105060, 2023-2028 рр.), що реалізується на кафедрі інформаційних систем та технологій, тематиці досліджень навчально-дослідної лабораторії інтелектуальних систем, комп'ютерних мереж та інтернет речей кафедри інформаційних систем та технологій Полтавського державного аграрного університету.

Метою кваліфікаційної роботи є підвищення ефективності програмування, покращення якості коду та оптимізації робочих процесів програмістів.

Завданнями кваліфікаційної роботи є:

– аналіз існуючих інструментарію для розробки вебконтенту за допомогою штучного інтелекту;

– дослідження можливостей нейропомічників для оптимізації;

– розробка рекомендацій щодо застосування нейропомічника для розробки вебконтенту.

Об'єктом дослідження є процес автоматизації розробки програмного забезпечення з використанням нейропомічників.

Предметом дослідження є методи та технології застосування нейропомічників для автоматизації та оптимізації процесу розробки програмного коду.

Методами дослідження є аналітичний, інформаційно-пошуковий, методи синтезу, емпіричне дослідження, аналіз літературних джерел, моделювання робочих процесів.

Інформаційна база кваліфікаційної роботи є сукупність наукових та технічних джерел, які висвітлюють сучасні підходи до автоматизації розробки коду та застосування нейронних мереж у веброзробці.

Елементи наукової новизни роботи полягають в методології застосування нейропомічника для підтримки та автоматизації процесів розробки програмного коду.

Практична значущість роботи полягає в обґрунтуванні рекомендацій щодо застосування нейропомічника для розробки вебконтенту – можуть бути використані для подальших досліджень за даною тематикою та при проєктуванні сайту Інтернет.

Апробація результатів відбувалася в рамках V Міжнародної студентської наукової конференції «Міждисциплінарні наукові дослідження та перспективи їх розвитку» (4 жовтня 2024 р., м. Черкаси), VII Міжнародної студентської наукової конференції «Розвиток суспільства та науки в умовах цифрової трансформації» (15 листопада 2024 р., м. Тернопіль).

За результатами досліджень здійснено 2 публікації 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 68 сторінок формату А4. Вона містить 68 рисунків.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ГЕНЕРАТИВНОГО ШТУЧНОГО ІНТЕЛЕКТУ РОБОТИ З ВЕБКОНТЕНТОМ

1.1 Генеративний штучний інтелект, як інструмент для відтворення вебконтенту

Нейропомічник – це система штучного інтелекту, яка допомагає користувачам вирішувати різні проблеми та передавати інформацію [1]. Вона працює за допомогою технологій обробки природної мови, щоб розуміти запитання, запити та надавати відповіді, рекомендації чи інші корисні інструменти (рис. 1.1).

Нейропомічники можуть автоматизувати багато рутинних процесів, таких як заповнення форм, бронювання зустрічей, керування розкладами або ведення баз даних. Це не тільки підвищує продуктивність, але й зменшує ймовірність помилок, які можуть виникнути під час виконання цих завдань.

Рисунок 1.1 – Пошукова система нейропомічника

Коли користувач формулює запит, нейропомічник аналізує його, виділяючи ключові слова та фрази, які можуть вказувати на конкретні теми чи питання, що цікавлять користувача [2]. Наприклад, якщо користувач запитує про новини в певній галузі, нейропомічник може визначити, який аспект цієї галузі його цікавить, хай то нові технології, економічні тенденції, зміни в законодавстві чи інші фактори (рис. 1.2).

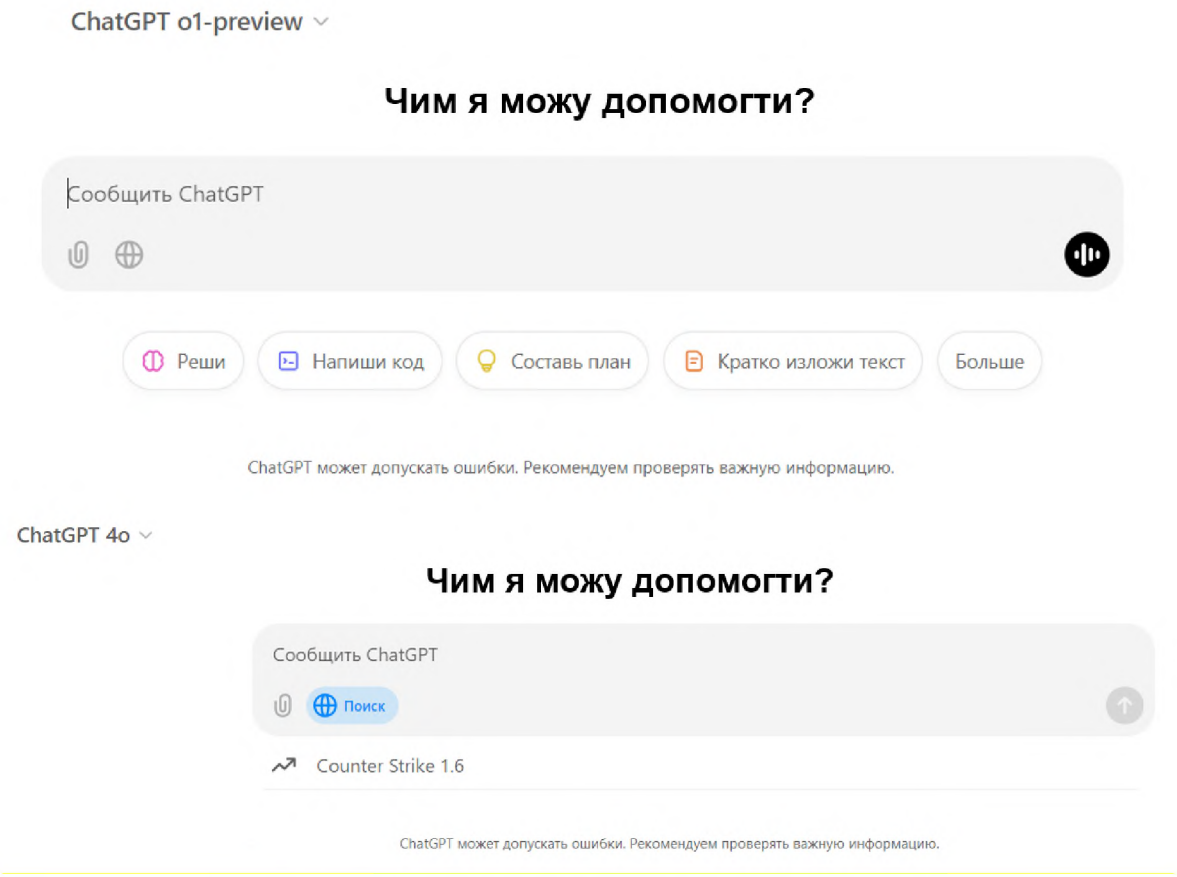


Рисунок 1.2 – Інтерфейс нейропомічника

Однією з головних переваг нейропомічника є його здатність не тільки отримувати доступ до великих обсягів даних, але й глибоко аналізувати їх [3]. Коли нейропомічник здійснює пошук у кількох джерелах, він не обмежується отриманням результатів, а активно оцінює якість отриманої інформації. Це дає змогу відфільтрувати найбільш релевантні та надійні джерела, надаючи користувачеві доступ не просто до масиву даних, а до справді корисної, релевантної та надійної інформації.

Аналіз якості даних починається на етапі відкриття. Нейропомічник оцінює джерела, з яких він черпатиме інформацію. Алгоритми, засновані на обробці природної мови та машинному навчанні, здатні визначати репутацію ресурсів: це можуть бути відомі новинні сайти, авторитетні блоги, наукові статті чи організації з високою репутацією в окремих дисциплінах. Це дозволяє нейропічнику вибирати лише експертів або надійні джерела.

Щоб оцінити вірогідність інформації, нейропомічник може порівняти кілька джерел, що описують ту саму подію чи факт. Якщо інформація відрізняється в різних джерелах, алгоритм може розпізнати ці різночитання та визначити, яке джерело є більш авторитетним. Наприклад, представлена робота з екраном, на якому відображаються новинні статті та дані (рис. 1.3) про певну подію. Це демонструє, як технологія може аналізувати різні джерела та визначати пріоритет.

Рисунок 1.3 – Відображення даних про певну подію

Іншою важливою частиною аналізу є оцінка актуальності інформації. Враховуючи темпи змін на інформаційному ринку, нейропомічник активно відстежують, коли дані публікуються [4]. Наприклад, для запитів, пов'язаних з останніми новинами в галузі, нейропомічник віддаватиме пріоритет останнім публікаціям та фільтруватиме застарілу інформацію. Це гарантує, що користувач отримує лише оновлені дані з практичною цінністю під час запиту. Нейропомічник значно підвищує ефективність процесу виявлення та аналізу інформації. Залежно від контексту запиту, нейропомічник також може використовувати алгоритми машинного навчання для визначення ключових тем і тенденцій, які цікавлять користувача. Це дозволяє адаптувати результати пошуку до особистих уподобань і потреб, надаючи точнішу та релевантнішу інформацію.

Крім того, нейропомічник може реалізовувати фільтри на основі якості джерела даних. Наприклад, публікації з відомих і надійних джерел можуть бути пріоритетними, підвищуючи довіру користувача до отриманих даних. Це особливо важливо в ситуації, коли в Інтернеті багато неперевіреної та фейкової інформації.

Щоб ще більше підвищити свою корисність, нейропомічник може інтегрувати функції, які дозволяють користувачеві зберігати або маркувати цікаві матеріали для подальшого перегляду. Це може включати можливість створювати дайджест особистої інформації або налаштовувати сповіщення для нових публікацій у вибраних темах.

Нарешті, варто зазначити, що нейропомічники стають дедалі розумнішими з розвитком технологій, набуваючи здатності не лише обробляти запити, а й передбачати потреби користувачів, надаючи проактивні та персоналізовані рішення.

Це робить їх незамінним інструментом у сценарії сучасного інформаційного середовища, де швидкість і якість отриманих даних можуть мати значний вплив на прийняття рішень. Зі здатністю швидко обробляти великі обсяги інформації та давати актуальні відповіді.

Таким чином, нейропомічники не тільки змінюють спосіб отримання та обробки інформації, але й створюють нові стандарти для спілкування, навчання та бізнесу. У світі, де інформація постійно збільшується, їх роль стає все більш актуальною та важливою, оскільки вони можуть надати підтримку та поради, необхідні для прийняття обґрунтованих рішень.

1.2 Використання нейропомічників для оптимізацуії коду

Використання нейропомічників для генерації та аналізу коду є важливим інструментом у роботі розробників і команд, що працюють над програмними проєктами сьогодні. Інструменти на основі штучного інтелекту, такі як ChatGPT, Gemini та інші, не тільки автоматизують рутинні завдання, але й допомагають забезпечити кращу якість і швидкість розробки. В цьому підпункті детальніше розглянемо, як нейпомічники можуть допомогти на різних етапах розвитку.

Нейропомічники можуть значно спростити процес створення коду, особливо при роботі з повторюваними завданнями або структурами шаблонів [5]. Прикладом, є написання простих методів Application Programming Interface, такі як обробка запитів GET або POST (рис. 1.4).

Нейропомічники може створювати базові шаблони, які можна швидко налаштувати відповідно до конкретних вимог.

Ще однією важливою перевагою є здатність нейропомічників неправильно аналізувати коди. Розробники часто стикаються з проблемами свого коду, якими можуть бути синтаксичні помилки, неправильний синтаксис або навіть складні помилки, які важко помітити.

Прикладом, існує JavaScript код який викликає асинхронні функції, нейропомічник може вказати на невірне використання await або допомогти зрозуміти, де виклик функції може призвести до неочікуваних результатів. Це значно зменшує час на виправлення помилок і підвищує загальну якість продукту (рис. 1.5).

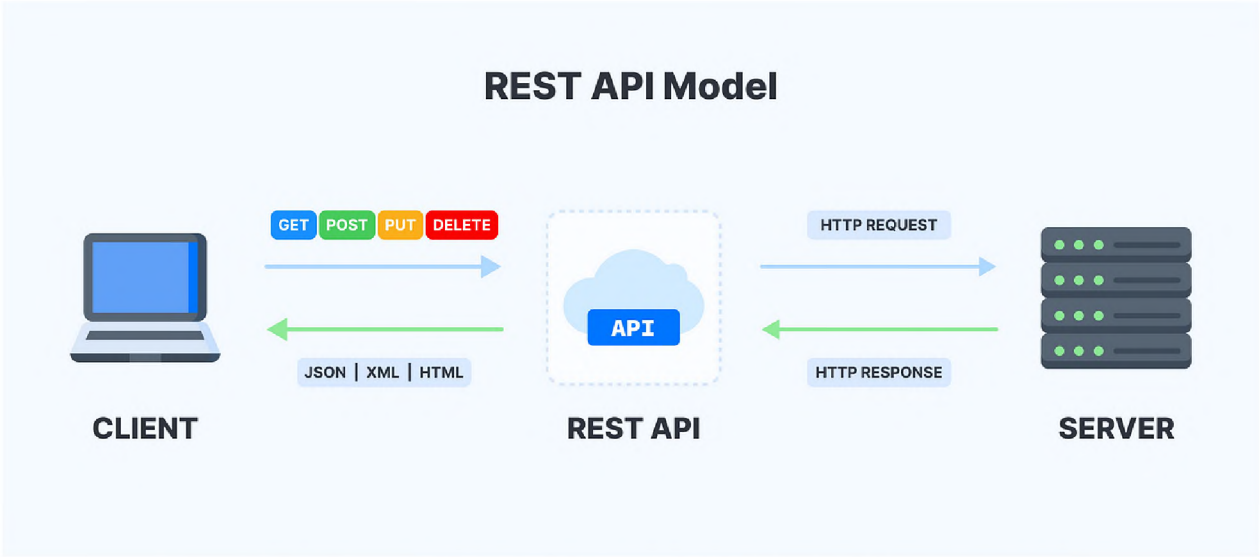


Рисунок 1.4 – Методи Application Programming Interface

Неправильний приклад:

```javascript
function fetchData() {
  const data = await fetch('https://example.com');
  console.log(data);
}
```

Правильний приклад:

```javascript
async function fetchData() {
  const data = await fetch('https://example.com');
  console.log(data);
}
```

Рисунок 1.5 – Вигляд правильних та непральних прикладів формування запитів

Нейропомічники можуть аналізувати наявний код і пропонувати кращі варіанти реалізації, які можуть бути більш ефективними або простішими в обслуговуванні. У разі роботи з базами даних штучний інтелект може запропонувати оптимальні методи запитів, які зменшують час виконання та навантаження на сервер [6].

Для розробників, наприклад, нейропомічники можуть запропонувати оптимізацію рендерингу компонентів або зменшення кількості рендерерів у React, дозволяючи пришвидшити завантаження сторінки та покращити взаємодію з користувачем. Наприклад React.memo який запобігає повторному рендерингу компонента, якщо його пропси не змінилися, це може бути дуже корисним коли функціональні компоненти отримують і передають однакові властивості (рис. 1.6).

```
import React from 'react';

// Компонент, обгорнутий в React.memo
const MyComponent = React.memo(({ name }) => {
  console.log('Рендеринг MyComponent');
  return <div>Hello, {name}!</div>;
});

export default function App() {
  const [count, setCount] = React.useState(0);

  return (
    <div>
      <button onClick={() => setCount(count + 1)}>Increment count: {count}</button>
      <MyComponent name="John" />
    </div>
  );
}
```

Рисунок 1.6 – Приклад запобігання повторному рендерингу компонента

Документування коду є важливим аспектом розробки, але багато розробників уникають цього через брак часу або бажання. Нейропомічники можуть автоматично генерувати коментарі та документацію для коду, полегшуючи його розуміння іншим розробникам.

Наприклад, під час написання нової функції нейропомічник може підготувати опис її призначення, вказавши вхідні параметри та значення, яке вона повертає. Це особливо корисно для документації Application Programming Interface або пояснень, де потрібне чітке розуміння функціональності (рис. 1.7).

```
/**
 * Обчислює площу прямокутника.
 *
 * @param {number} length - Довжина прямокутника.
 * @param {number} width - Ширина прямокутника.
 * @returns {number} Площа прямокутника.
 */
function calculateRectangleArea(length, width) {
  return length * width;
}

// Приклад використання:
console.log(calculateRectangleArea(4, 5)); // Повертає 20
```

Рисунок 1.7 – Вигляд згенерованої функції, де описано її призначення

Таким чином, нейропомічники є потужним інструментом для прискорення розробки програмного забезпечення, полегшуючи процеси написання коду, його оптимізації та забезпечення якості. При розумному використанні вони стають чудовим доповненням до навичок і досвіду розробника, дозволяючи зосередитися на складних та інноваційних аспектах розробки.

1.3 Інтеграція нейропомічника з вебконтентом

Поєднання нейропомічника з вебконтентом включає інтеграцію штучного інтелекту або чатботів, які здатні не тільки аналізувати та розуміти інформацію на вебресурсах, але й взаємодіяти з користувачами в режимі реального часу. Це дозволяє створювати більш інтерактивні вебсторінки, де кожен відвідувач може отримати персоналізовану допомогу, рекомендації чи відповіді на запитання, враховуючи конкретний контент сайту. Підвищення ефективності роботи з вебресурсами шляхом інтеграції нейропомічників базується на значному скороченні часу, який користувач витрачає на пошук інформації.

Традиційно, щоб знайти потрібні дані на вебсайті, користувачі повинні переглядати великі обсяги тексту, скролити через сторінки або навіть використовувати функції пошуку. Однак, за допомогою нейропомічника, ці дії стають набагато простішими та швидшими [7]. Нейропомічник може автоматично визначати, що потрібно знати користувачеві, і надавати точну відповідь на основі контексту вмісту сторінки.

Основна перевага нейропомічника полягає в тому, що він здатний працювати з різними форматами контенту і здатний «розуміти» не тільки текст, але й інші види медіа [8]. Наприклад, якщо користувач шукає інформацію, пов'язану з певним зображенням чи відео, нейропомічник може проаналізувати ці медіа та на основі їх змісту надати відповідь. Це робить взаємодію з сайтом більш інтерактивною та природною.

Основна перевага нейропомічника полягає в тому, що він здатний працювати з різними форматами контенту і здатний «розуміти» не тільки текст, але й інші види медіа. Наприклад, якщо користувач шукає інформацію, пов’язану з певним зображенням або відео, нейропомічник може проаналізувати цей медіафайл і надати відповідь на основі його вмісту. Це робить взаємодію з сайтом більш інтерактивним та природним.

Для текстового вмісту нейропомічник працює з технологіями обробки природної мови (NLP). За допомогою такого алгоритму він може інтерпретувати їх у контексті того, що є на сторінці, а не лише фактичних запитів. Наприклад, якщо користувач вимагає про певний продукт, нейропомічник не тільки шукає ключові слова, але й визначає контекст, у якому ці товари згадуються в тексті. Це призводить до більш точних і детальних відповідей.

Нейропомічник, який працює з текстовим вмістом, використовує технологію, яка дозволяє розуміти природну мову. Це означає, що він не тільки шукає слова в текстах, але може зрозуміти їх значення та контекст, у якому вони вживаються. Наприклад, якщо користувач запитує про товар, нейропомічник не тільки шукатиме слово «товар», а й звертатиме увагу на те, де і як це слово вжито.

Він може зрозуміти, що це опис товару, відгук про нього або якась технічна характеристика. Завдяки цьому підходу нейропомічник може давати більш точні відповіді, які враховують не лише те, що є в тексті, а його загальний зміст.

Крім того, нейропомічник може адаптувати вміст вебсайту до вподобань і потреб кожного користувача. Аналізуючи поведінку відвідувача, наприклад попередні запити або взаємодію з певними розділами сайту, штучний інтелект може надати саме ту інформацію, яка, ймовірно, буде цікавою для користувача [9]. Наприклад, якщо люди часто шукають новини про певні продукти чи послуги, сайт може автоматично відображати відповідні матеріали, статті чи спеціальні пропозиції, які відповідають цим інтересам.

Такий підхід не тільки значно покращує користувацький досвід, але й дозволяє скоротити час, витрачений на пошук необхідної інформації. Нейропомічники, завдяки своїм можливостям обробляти та інтерпретувати текст, зображення, відео та інші типи медіа, є потужним інструментом для створення інноваційних вебсайтів.

Висновки до розділу 1

У цьому розділі було детально розглянуто роль нейропомічників у контексті вебконтенту, їх можливості та застосування на різних рівнях взаємодії з користувачем. Нейропомічники з їх здатністю обробляти та інтерпретувати текст, зображення, відео та інші форми медіа є потужним інструментом для створення інноваційних вебсайтів, здатних взаємодіяти з користувачами.

Перше питання, яке було розглянуто, полягало в тому, як можна використовувати нейропомічника для відтворення вебконтенту.Це містить можливість автоматичного аналізу та організації вмісту на вебсайті, щоб користувачі могли взаємодіяти з ним на зручному та інтуїтивно зрозумілому рівні. Важливо, що нейропомічник дозволяє створювати персоналізовані сторінки, адаптовані до запитів і поведінки користувачів.

Завдяки таким можливостям вебсайти можуть надавати більш точні та релевантні відповіді на запитання, значно підвищуючи простоту та ефективність взаємодії з користувачем.

Також було розглянуто використання нейропомічників для генерації та аналізу сигналів, що важливо для автоматизації багатьох аспектів веброзробки. Нейропомічники можуть допомагати створювати вебресурси шляхом автоматизації процесів кодування, налагодження та тестування. Вони можуть надавати рекомендації щодо оптимізації коду, виявляти помилки або навіть генерувати код на основі заданих параметрів. Це дозволяє розробникам працювати набагато швидше, зменшує кількість рутинних завдань і дозволяє їм зосередитися на більш творчих і складних аспектах розробки.

У розділі також описується поєднання нейропомічників із вебконтентом, що є ключовим фактором у створенні динамічних та адаптивних вебсторінок. Завдяки інтеграції нейропомічників вебсайти можуть не тільки відповідати на прямі запити користувачів, але й контекстно розуміти, який контент буде найбільш корисним для кожного конкретного користувача.

Отже, нейропомічники на вебсайтах забезпечують високий рівень персоналізації та інтерактивності, допомагаючи створювати адаптивні, ефективні та зручні вебресурси. Вони дозволяють автоматизувати значну частину процесів розробки, оптимізації контенту та взаємодії з користувачем, роблячи вебсайти більш ефективними та доступними для ширшої аудиторії. В результаті такі технології допомагають не тільки покращити користувацький досвід, але й підвищити ефективність роботи із загальним вебконтентом.

РОЗДІЛ 2
РОЗРОБКА МЕТОДОЛОГІЇ ЗАСТОСУВАННЯ НЕЙРОПОМІЧНИКА ДЛЯ ПІДТРИМКИ ТА АВТОМАТИЗАЦІЇ ПРОЦЕСІВ РОЗРОБКИ ПРОГРАМНОГО КОДУ

2.1 Поєднання нейропомічника з бібліотекою React.js

React – це бібліотека JavaScript, яка дозволяє створювати потужні інтерфейси для вебдодатків [10]. Його розроблено для швидкого й ефективного відтворення елементів сторінки, що робить його ідеальним для створення масштабованих інтерактивних програм. Це розширена бібліотека компонентів, побудована на JavaScript. React використовує загальний підхід до створення складних користувальницьких інтерфейсів. Цей підхід допомагає розбивати інтерфейси на менші компоненти [11].

Кожен компонент має власні властивості та статус, що дозволяє динамічно змінювати зовнішній вигляд компонента залежно від середовища. Одним із найважливіших компонентів React є віртуальна DOM (Document Object Model), яка дозволяє швидко та ефективно розміщувати елементи інтерфейсу на сторінці.

React також підтримує односпрямований потік даних, який дозволяє даним проходити вгору по ієрархії компонентів, що забезпечує легкість і передбачуваність в управлінні середовищем ресурсів [12]. Завдяки своїй надійності та простоті використання React став незамінним інструментом для розробки вебдодатків для багатьох компаній і організацій (рис. 2.1).

Компонентна архітектура (CBA) – це шаблон проєктування програмного забезпечення, який поділяє програмне забезпечення на окремі компоненти, які можна повторно використовувати в різних частинах програми [13]. Хоча вебкомпоненти є основою CBA, вони не є єдиним способом використання цієї архітектури, існують інші фреймворки, такі як React, Angular і Vue, які використовують компонентну архітектуру для створення вебдодатків [14].

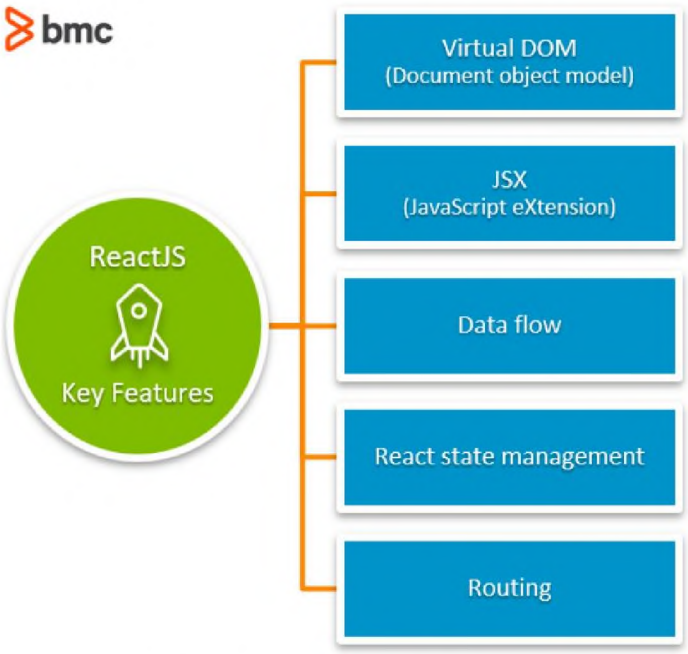


Рисунок 2.1 – Структура бібліотеки React JS

Розглянемо переваги React, які можуть бути корисними на основі думки інших розробників React.

1. Віртуальний DOM (Virtual DOM) технологія, яка використовується в React для оптимізації процесу відтворення екрана сторінки (рис. 2.2). Він полягає у створенні віртуальної копії фактичного DOM, який React створює та оновлює екран. Коли дані змінюються, React спочатку створює віртуальний DOM, який представляє новий тип вмісту. React порівнює цей віртуальний DOM з його станом до зміни та показує різницю між ними. Потім React вносить невеликі зміни у віртуальну DOM, щоб відповідати віртуальній DOM [15].

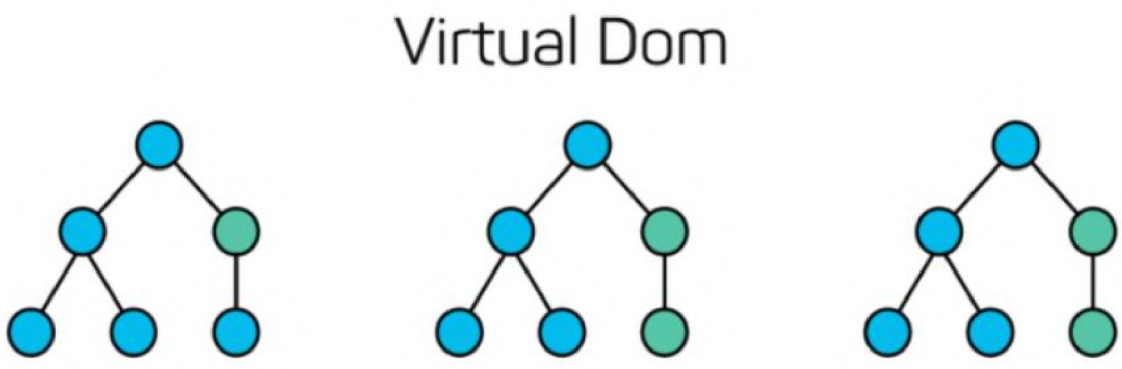


Рисунок 2.2 – Структура віртуального DOM

2. JSX – це синтаксичне розширення для JavaScript, яке React використовує для більш формального визначення інтерфейсу вебдодатка. Це дозволяє розробникам писати код, схожий на HTML, у своєму коді JavaScript, полегшуючи створення складних інтерфейсів користувача та керування ними. React дозволяє розробникам створювати елементи інтерфейсу, які знаходяться в різних частинах програми. Разом JSX і React допомагають створювати динамічні та адаптивні вебдодатки. JSX дозволяє розробникам писати код, який виглядає та виглядає як HTML, але насправді працює як JavaScript. Потім React зосереджується на відображенні цих компонентів у браузері та керує макетом і поведінкою програми [16].

3. Компонентний підхід (рис. 2.3) є базовим принципом для створення React, одного з найпопулярніших фреймворків JavaScript для програмування. Використовуючи об'єктноорієнтований підхід, React допомагає розбити вебдодаток на менші незалежні компоненти, які можна використовувати знову і знову [17]. Кожен компонент має власний внутрішній стан (атрибут), атрибути та вхідні дані (props), а також життєві цикли, які оцінюють поведінку компонента під час створення, оновлення та знищення. Матеріали можна поєднувати та комбінувати для створення більш складних конструкцій. Коли об'єкт змінює форму або отримує нову енергію, він автоматично знову використовується.

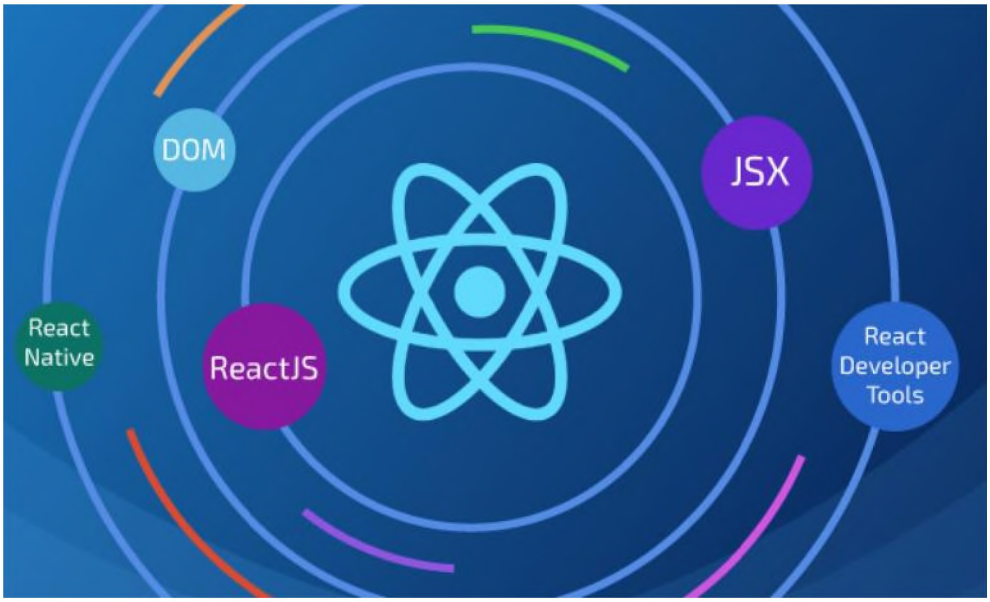


Рисунок 2.3 – Структура віртуального DOM

4. Наявність модулів. У React є метод, у якому функції діляться на невеликі незалежні модулі, які називаються компонентами. Кожен компонент відповідає за свою частину інтерфейсу користувача та може використовуватися в різних частинах програми. Крім того, модульність React дозволяє скоротити час розробки та підвищити продуктивність шляхом повторного використання компонентів [18]. Компоненти можна повторно використовувати в різних областях програми або навіть в інших проєктах, зменшуючи дублювання коду та підвищуючи ефективність розробки. Це зберігає код чистим і охайним, роблячи його простішим і легшим для розуміння (див. рис. 2.3).

5. Широка спільнота. React має велику й активну спільноту розробників, які працюють із бібліотекою React і діляться своїм досвідом із проєктом. Ця група є онлайн і включає дизайнерів, тестувальників та інших експертів. Спільнота React зосереджена на створенні та підтримці бібліотеки React, а також створенні та підтримці екосистеми React, яка включає додаткові бібліотеки, інструменти, фреймворки та плагіни для розширення проєктів React. Організація активно обговорює нові ідеї розвитку та проєкти, співпрацює з промисловістю та надає підтримку стартапам.

Щоб інтегрувати головного помічника ChatGPT у React.js, починаємо зі створення компонента, який представляє інтерфейс користувача для надсилання повідомлень і взаємодії з Application Programming Interface головного помічника. Спершу, стан компонента налаштовано для зберігання всієї необхідної інформації: історії чату, тексту, введеного користувачем, і індикатора завантаження, який відображатиметься під час очікування відповіді.

Кожне повідомлення в розмові містить текст і тип відправника, користувача чи фасилітатора (додаток А, рис. А.1). Це дозволяє зберігати та відображати серіалізовані розмови між користувачем і помічником.

Потім компонент надсилає запит сервера Application Programming Interface нейропомічника за допомогою бібліотеки, такої як axios, яка містить текст повідомлення користувача та ключ Application Programming Interface для автентифікації.

Цей код містить текстове поле та кнопку «Надіслати», що дозволяє користувачеві легко спілкуватися з чатботом. Якщо відповідь ще не отримано, піктограма завантаження відображає повідомлення про завдання, яке надає користувачеві чіткий відгук про статус запиту. Інформаційна історія відображається у текстовому форматі, про користувача та нейропомічника вирівняна по різні сторони екрана, що робить інтерфейс інтуїтивно зрозумілим, візуальне представлення знаходиться на (рис. 2.4).

Ось як працює цей інтерфейс:

– коли користувач натискає «Надіслати», праворуч з'являється нове повідомлення, внизу значок завантаження;

– відповідь бота вирівняна по лівому краю;

– створює ефект діалогу, що робить інтерфейс зрозумілим і зручним для спілкування в реальному часі.

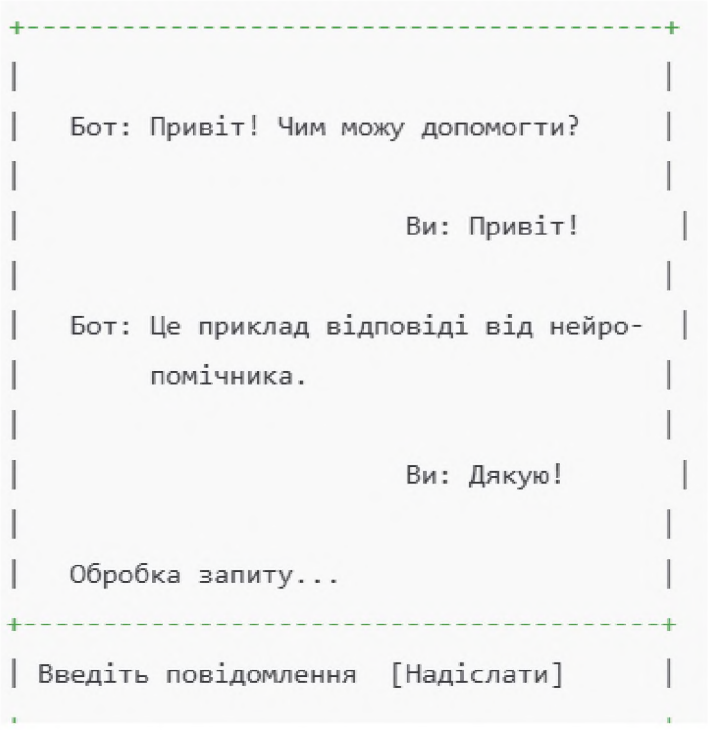


Рисунок 2.4 – Візуальний вигляд тестового компонента

Крім того, після кожного запиту, навіть якщо користувач не надає повне повідомлення, система відповідає за часткову обробку даних і відображення результатів на основі вхідної підмножини [19].

Це не тільки допомагає покращити зручність використання, але й значно покращує зручність використання. Наприклад, вводячи довгі запити, можна надіслати частину тексту на сервер, отримати відповідь і почати обробку, поки решта запиту все ще надсилається, зменшуючи накладні витрати на зв'язок.

Оскільки передача даних у реальному часі необхідна для підтримки безперервного потоку даних, компонент має бути оптимізовано, щоб мінімізувати часову затримку між запитом і відповіддю.

Це можна досягти за допомогою ефективних асинхронних функцій на стороні сервера та оптимізації взаємодії між компонентами на стороні клієнта. Залежно від завантаження Application Programming Interface і швидкості мережі, механізм повторного запиту може стати невід'ємною частиною інтерактивної системи, покращуючи роботу користувача, особливо в умовах непередбачуваних затримок або тимчасових проблем з мережею. Ідея цього механізму полягає в тому, щоб надіслати запит, якщо не зроблено жодної спроби.

2.2 Оптимізація роботи React.js з нейропомічником

Оптимізація роботи React.js з нейропомічником включає в себе кілька аспектів, які забезпечують ефективну взаємодію між компонентами користувацького інтерфейсу та сервером, де працює нейропомічник [20]. Основні принципи оптимізації полягають у зменшенні витрат часу, покращенні продуктивності за високих робочих навантажень і забезпеченні кращої взаємодії з користувачем.

Однією з головних особливостей React є асинхронність, використовуючи асинхронні запити (наприклад, через axios або fetch), можна надсилати запити до нейропомічника, не блокуючи інтерфейс (додаток А, рис. А.2). Це дозволяє користувачам взаємодіяти з додатком, поки дані завантажуються або обробляються у фоновому режимі, що значно покращує взаємодію з користувачем та покращує його досвід.

Асинхронні запити в React зазвичай виконуються в поєднанні з хуками useEffect і useState, які дозволяють контролювати потік даних і відображати відповідні повідомлення або символи [21]. Це забезпечує плавний і ефективний інтерфейс, де користувачі можуть бачити статуси завантажень або повідомлення про помилки в режимі реального часу. Дані динамічно оновлюються, як тільки вони стають доступними, що призводить до швидшого часу відповіді для програм і більшої задоволеності користувачів.

Робота асинхронності в React.js включає такі аспекти як:

– обробка неузгоджених запитів Application Programming Interface;

– нейропомічника дозволяє здійснювати запити без заморожування інтерфейсу;

– дозволяє не блокувати основний потік візуалізації в React, що підвищує загальну продуктивність програми.

Кешування відповідей від нейропомічника може значно покращити ефективність, особливо якщо одна й та сама інформація запитується кілька разів (додаток А, рис. А.3). Можна зберігати отримані дані в локальному стані компонента або використовувати глобальні стейти, такі як Redux або React Context, для кешування результатів (рис. 2.5).

Основні аспекти кешування в React.js:

– зменшується кількість запитів до Application Programming Interface;

– скорочує час завантаження даних;

– знижує навантаження на сервер.

```
{
  "name": "Example Data",
  "description": "This is a cached response."
}
```

Рисунок 2.5 – Приклад результату при успішного кешування даних

Ліниве завантаження та відкладене виконання є потужними методами оптимізації в React.js [22], які можуть скоротити час завантаження програми, зменшити навантаження на систему та покращити взаємодію з користувачем. Це особливо важливо в складних і великомасштабних програмах, таких як чати або інші реальні взаємодії (додаток А, рис. А.4).

У цьому випадку використовував ліниве завантаження, не доводилося щоразу надсилати запит нейропомічнику. Відображає потрібний компонент, а не всі інші компоненти, таким чином зменшуючи навантаження на додаток.

Ключові аспекти лінивого наванження та відкладного виконання:

– покращується початкова продуктивність додатка;
– знижує споживання пам’яті;
– завантаження потрібних компонентів одразу.

Нейропомічник може аналізувати поведінку користувача в реальному часі, визначати, які компоненти є критичними, а які рідко взаємодіють з користувачем [23]. Наприклад, він може використовувати дані про кліки або рухи миші, щоб виявити компоненти, які часто залишаються незмінними. В таких випадках можна автоматично обгортати їх у React.memo або застосовувати useMemo, щоб зменшити кількість рендерів.

React.memo використовується для обгортання функціональних компонентів, які мають запускатися лише тоді, коли їхні залежності змінюються. Це особливо важливо для компонентів, які не переміщують і не змінюють їх часто. Якщо пропси компонента не змінилися, React збереже попередньо створену версію компонента та повторно її використає, уникаючи надмірності. Відображати компонент при зміні ресурсів (рис. 2.6).

```
import React from "react";

// Компонент, обгорнутий в React.memo
const MemoizedComponent = React.memo(({ value }) => {
    console.log("Рендеринг компонента MemoizedComponent");
    return <div>Значення: {value}</div>;
});
```

Рисунок 2.6 – Приклад створення компонента React.memo

useMemo використовується для кешування обчислень, які можуть бути затратними з точки зору продуктивності, щоб уникнути повторного виконання цих обчислень на кожному рендері компонента (рис. 2.7). useMemo запам'ятовує результат виконання функції та перераховує його лише тоді, коли змінюються залежності [24].

```
const memoizedValue = useMemo(() => {
    // Обчислення
    return результат;
}, [залежності]);
```

Рисунок 2.7 – Приклад застосування useMemo

Основні аспекти React.memo та useMemo:

- мінімізує непотрібні рендери компонентів;
- знижує навантаження на браузер;
- підвищує ефективність;
- підвищує загальну швидкість додатка.

Оптимізація React.js за допомогою нейропомічника полягає в зменшенні затримки, покращенні продуктивності та забезпеченні взаємодії користувача з системою. Використання асинхронних запитів, кешування, лінивого завантаження та відкладеного виконання допомагають зробити програму швидшою, ефективнішою та простішою у використанні.

2.3 Інтеграція нейропомічника з кросплатформою Node.js

Node.js потужна та широко використовувана технологія для розробки вебдодатків на стороні сервера. Це середовище JavaScript, яке дозволяє розробникам використовувати мову, яка раніше використовувалася в браузерах, для створення високоякісних програм на стороні сервера [25].

Однією з ключових особливостей Node.js є його асинхронність і залежність від контексту, що робить його ідеальним для ресурсомістких додатків, де це критично важливо, і максимально використовувати паралелізм.

Node.js складається з таких компонентів:

– двигун V8;
– node package manager;
– система модулів;
– Event Loop.

Вбудований двигун V8 є вихідним кодом механізму JavaScript, який Google розробив для використання у браузері Chrome. V8 відомий своєю високою продуктивністю та ефективністю. Виконує код JavaScript ефективніше, ніж інші криптографічні механізми [26].

Основною особливістю V8 є перетворення вхідного коду в машинний. Використання V8 у Node.js дозволяє використовувати JavaScript для створення серверних програм та інших інструментів поза браузером (рис. 2.8). Крім того, Node.js використовує V8 для кращого асинхронного та неблокуючого виконання коду [27].

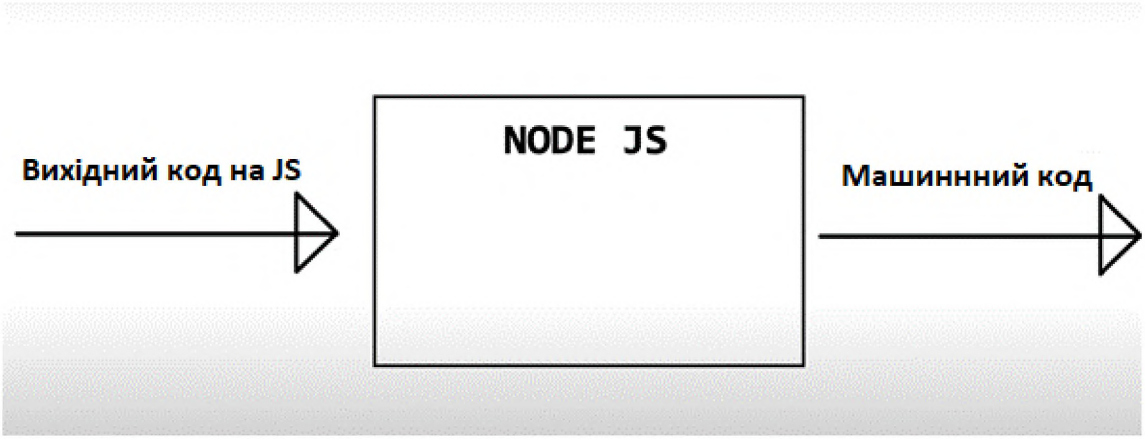


Рисунок 2.8 – Схематичне зображення роботи V8 двигун

Використання Node Package Manager – це менеджер пакетів для платформи Node.js, який відіграє ключову роль в оновленні та управлінні залежностями в проєктах JavaScript (рис. 2.9).

Обираючи npm, розробники можуть легко взаємодіяти з широким спектром бібліотек і інструментів [29]. Однією з основних функцій npm є встановлення пакетів і керування ними. Кожен проєкт може мати файл package.json, який описує список залежностей та іншу інформацію про проєкт. Команда npm install встановлює всі залежності в цей файл, забезпечуючи відтворюваність і легкість роботи з проєктом в різних середовищах. Розробники можуть публікувати власні бібліотеки або інструменти, надаючи широкий спектр готових рішень для різних завдань.

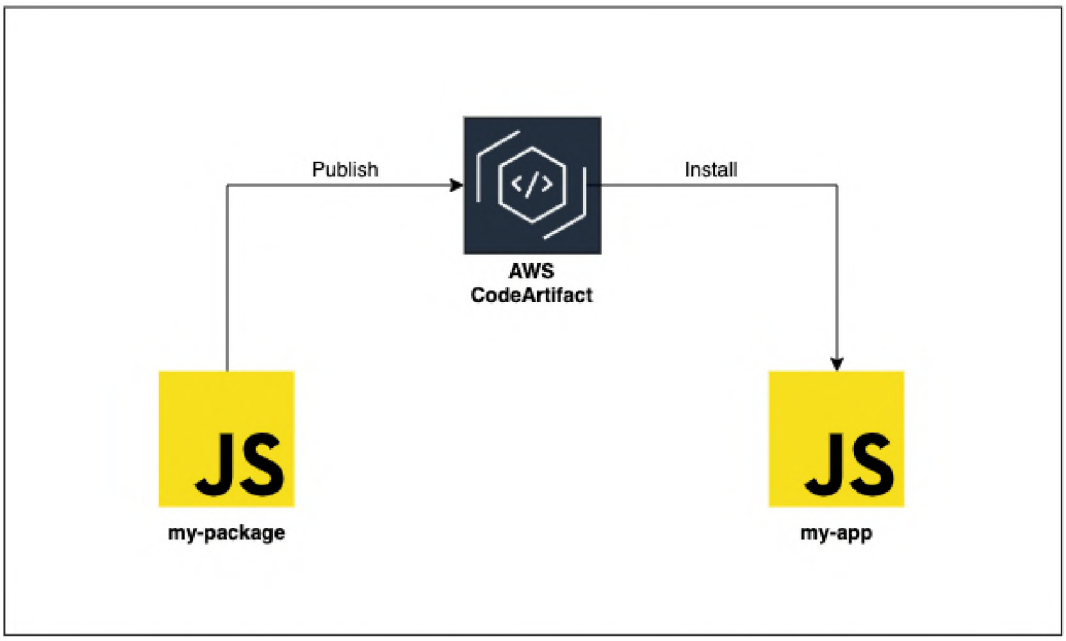


Рисунок 2.9 – Приклад роботи Node Package Manager

Модульна система є однією з основних функцій Node.js і відіграє важливу роль у створенні організованих блоків коду, які можна підтримувати та використовувати.

Базується на стандарті Common JS і дозволяє розбивати код на невеликі незалежні модулі, які можна використовувати в інших частинах програми. Модулі бувають двох типів.

Перший тип вбудовані моделі Node.js має багато вбудованих модулів, таких як file system (для операцій і файлових систем), http (для створення вебсерверів та події (додаток А, рис. А.5 та рис. А.6).

Крім того, розробники можуть використовувати сторонні модулі, що встановлені за допомогою Node Package Manager.

Другий тип анонімні модулі розробники можуть створювати анонімні модулі, які визначені вбудовано в той самий файл, де вони розгорнуті. Це корисно для невеликих блоків коду, які не призначені для використання в інших частинах проєкту [30]. Прикладом буде функція publicFunction, яка повертає змінну, за допомогою module.export, експортуємо в будь-який проєкт (рис. 2.10).

```
const fakeDatabase = {
  users: []
};

// Функція для додавання нового користувача
const addUser = (name, age) => {
  const newUser = {
    id: fakeDatabase.users.length + 1,
    name: name,
    age: age
  };
  fakeDatabase.users.push(newUser);
  return newUser;
};
```

Рисунок 2.10 – Приклад використання анонімного модуля

Event Loop в Node.js – це інструмент, який дозволяє виконувати кілька завдань одночасно, навіть якщо для виконання деяких потрібен час, наприклад завантаження файлу чи надсилання запиту на сервер [31]. Ось як це працює: коли Node.js зустрічає прості інструкції, такі як виконання обчислень або виведення тексту, він негайно їх виконує.

Поки асинхронне завдання працює десь у фоновому режимі, Node.js не переривається й продовжує виконувати інші завдання, як «офіціант» приймає нові замовлення, чекаючи, поки їжа буде готова. Коли асинхронне завдання завершується, Node.js отримує маркер про його завершення (рис. 2.11 і 2.12), що стосується цього завдання, і виконує заплановані події після завершення, наприклад, використовує результат.

```
console.log('Початок')

setTimeout(()=>{
 console.log('Це асинхронне завдання')
}, 1000)

console.log('Кінець')
```

Рисунок 2.11 – Приклад використання асинхронних функцій

```
C:\async>node test.js
Початок
Кінець
Це асинхронне завдання

C:\async>
```

Рисунок 2.12 – Результат асинхронних функцій

Якщо коротко пояснити асинхронні функції не чекають доки інша функція виконається У цьому прикладі (див. рис. 2.11) функція setTimeout не чекає на виконання, але вони виконують роботу з написання коду, який утримується.

Асинхронний код – це код, який виконується не послідовно, а здебільшого у паралельних або непослідовних шляхах виконання [32]. Тобто виконання такого коду може початися, призупинитися і відновитися пізніше, коли виконуються асинхронні операції (див. рис. 2.11).

Синхронний код – це код, який виконується послідовно, одна операція за раз, без паралельних або асинхронних операцій. Усі завдання виконуються по черзі, в якій вони записуються, і наступне завдання не починається до завершення попереднього (рис. 2.13 і 2.14). Наприклад, є черга, де кожен чекає своєї черги, і ніхто не може просунутися вперед, поки не закінчить попередній, так само й працює синхронний код.

У синхронному коді все відбувається в суворій послідовності. Якщо виконується операція, яка потребує багато часу, наприклад отримання даних із сервера або читання файлу, уся програма зупиняється та чекає на завершення операції, перш ніж перейти до наступної операції.

Асинхронний код працює інакше. Тут програма не чекає завершення тривалих операцій, таких як отримання даних або створення великого файлу. Замість цього він переходить до інших завдань і «повертається» до асинхронного завдання після його завершення. В JavaScript можна використовувати такі методи, як setTimeout, promises або async/await, щоб керувати асинхронним виконанням подій. Це дозволяє їм виконувати іншу роботу, поки вони чекають результатів давно назрілої роботи.

Асинхронний код робить програму значно ефективнішою, особливо в середовищах з високим навантаженням, таких як сервери, що обробляють багато одночасних запитів, оскільки вона не затримується на одному завданні й не блокує виконання інших.

```
console.log('Початок')

setTimeout(()=>{
  console.log('Це асинхронне завдання')
}, 1000)

function test() {
   for(let i = 0 ; i < 1e29; i++){}
   console.log("Довга операція завершена")
}

test()

console.log('Кінець')
```

Рисунок 2.13 – Приклад використання синхронних функцій

```
PS C:\async> node test.js
Початок
Довга операція завершена
Кінець
PS C:\async>
```

Рисунок 2.14 – Результат синхронних функцій

Node.js – це дуже швидке та ефективне серверне середовище з багатьма корисними функціями. Його серцем є двигун V8 від Google, який виконує JavaScript, перетворюючи його на машинний код.

Це робить кодування швидшим і ефективнішим. Ще одна важлива функція Node.js, менеджер пакетів, який дозволяє легко додавати необхідні бібліотеки та керувати ними у вашому проєкті [33].

Це дозволяє ефективно знаходити тисячі важливих інструментів, що значно полегшує оновлення. Модульна структура в Node.js дозволяє організовувати код в окремі розділи, які називаються модулями. Це робить ваш код більш структурованим і дозволяє повторно використовувати його в проєктах.

Нарешті, основний механізм роботи Event Loop. Це дозволяє виконувати декілька запитів одночасно, не чекаючи завершення кожного, що робить Node.js ідеальним для проєктів із кількома одночасними запитами.

Ці функції роблять Node.js придатним для створення потужних, масштабованих програм на стороні сервера, які можуть обробляти великі обсяги даних у режимі реального часу.

2.4 Взаємодія з серверами та створення віртуальних запитів

Для виконання HTTP запитів використовується http модуль, який дозволяє надсилати запити на сторінку Application Programming Interface, цей спосіб дуже ефективний, наприклад, простий GET запит (додаток А, рис. А.7).

Основними елементами асинхронних функцій є async та await, а також використання функцій зворотного виклику, промісів (Promises) або бібліотек для асинхронних подій, таких як axios або fetch.

Після встановлених модулів, використовуємо або створюємо асинхронні функції Вони забезпечують більш читабельний, зрозумілий і логічний спосіб роботи з асинхронними операціями.

Асинхронні функції в Node.js є частиною моделі асинхронного програмування, яка реалізована в мові JavaScript за допомогою концепції виклику та інших методів роботи з асинхронним кодом. Асинхронна функція має два параметри такі response (res) та request (req).

Методи які має request:

– method, включає в себе тип метода (GET, POST, DELETE);

– headers, включає вигляд в якому приходить запит на сервер (json, html);

– body, найважливіша частина запиту, вона включає сам запит.

Методи які має response:

– statusCode, відповідає статусу відповіді (200 (OK), 404 (Not Found), 500 (Internal Server Error));

– writeHead, встановлює код статусу та HTTP заголовки;

– send та json, надіслання відповіді клієнту на запит.

Взаємодія з серверами та створення віртуальних запитів на Node.js, особливо з використанням нейромережевих помічників, дозволяє реалізувати інтелектуальні функції для обробки запитів та відповідей. Це корисно для створення чат-ботів, інтелектуальних асистентів та систем рекомендацій, де потрібна швидка й автоматизована обробка даних.

Для задання будь-яких запитів та коректної роботи, Node.js потребує залежності. Всі залежності знаходяться у файлі package.json. Цей файл містить три категорії: назву, запуск, та залежності до файлу. Приклад файлу package.json відображено на (рис. 2.15).

```
{
  "name": "back",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon server.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@sendgrid/mail": "^8.1.1",
    "cors": "^2.8.5",
    "crypto": "^1.0.1",
    "express": "^4.18.3",
    "jsonwebtoken": "^9.0.2",
    "mysql2": "^3.9.2",
    "nodemailer": "^6.9.12",
    "nodemon": "^3.1.0"
  }
}
```

Рисунок 2.15 – Приклад файлу package.json

Для тестування запитів, найактуальніший інструмент Postman, який дозволяє розробникам легко надсилати HTTP запити, отримувати відповіді та аналізувати їх.

Він надає інтуїтивний інтерфейс для створення, відправлення та управління запитами до серверів (рис. 2.16).

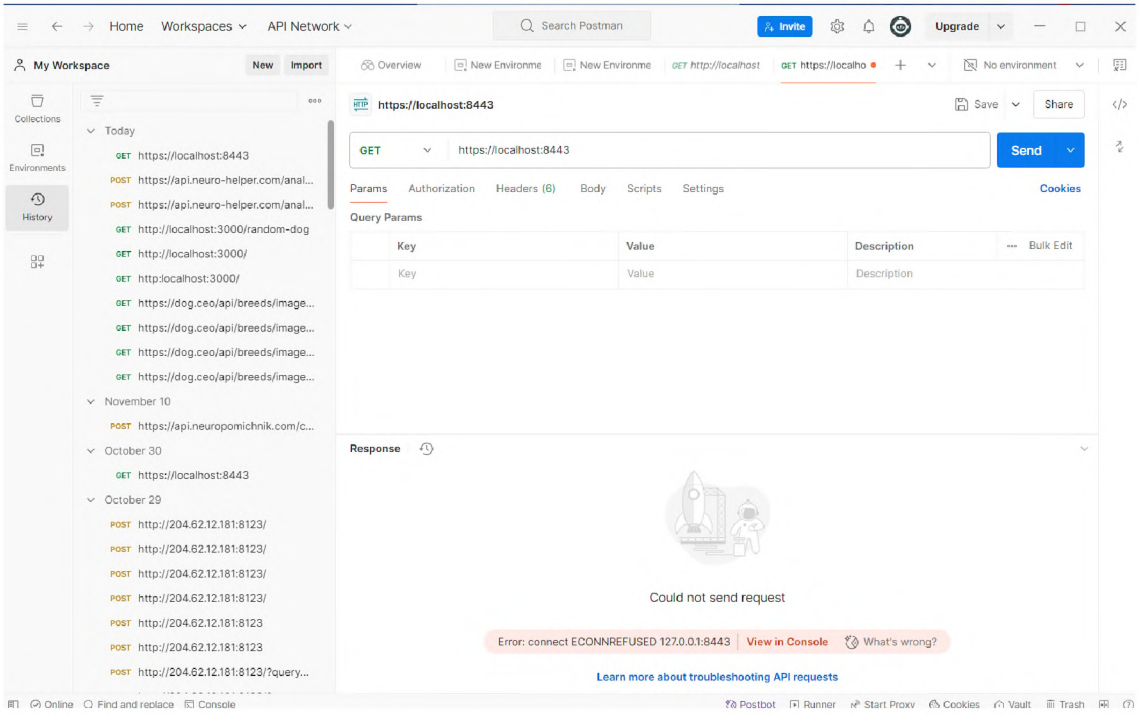


Рисунок 2.16 – Візуальний вигляд інтерфейсу Postman

Основні компоненти Postman:

– створення та надсилання запитів, дозволяє тестувати різні методи HTTP (GET, POST, PUT, DELETE тощо) та передавати в запит параметри, заголовки, аутентифікацію та тіло запиту;

– автоматизація тестування, дозволяє створювати скрипти для автоматизації перевірок;

– колекції, дозволяє групувати запити на колекції;

– перемінні середовища, дозволяє створювати перемінні для тестування.

Також Postman може робити Sql запити баз даних таких як Clickhouse, під'єднався до неї в даному через username та password (рис. 2.17), але Postman може підключатися різними шляхами (рис. 2.18).

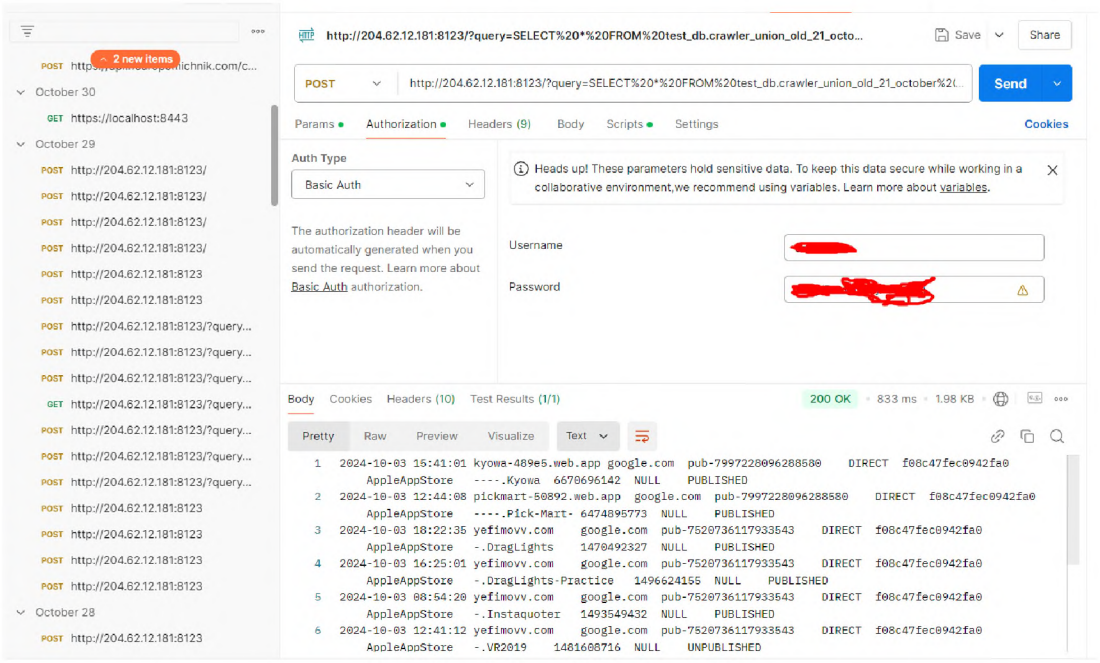


Рисунок 2.17 – Підключення до бази даних Clickhouse через Postman

Головна концепція Postman полягає в тому, щоб надати розробникам і тестувальникам зручний інструмент для роботи з Application Programming Interface.

Мета Postman спростити створення, тестування, документування та автоматизацію запитів до Application Programming Interface, полегшуючи спілкування між клієнтськими додатками та серверами (рис. 2.19).

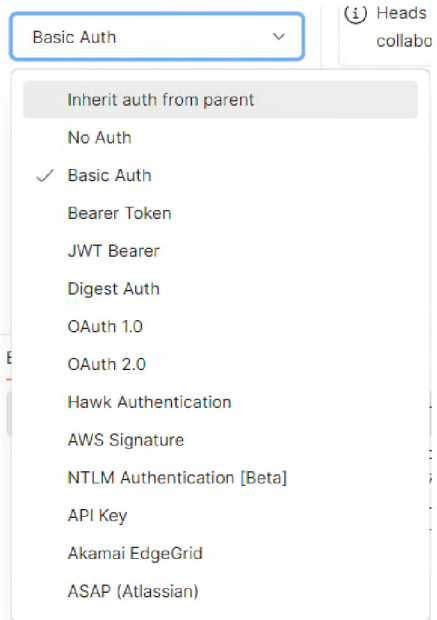


Рисунок 2.18 – Список різних шляхів підключення до бази даних

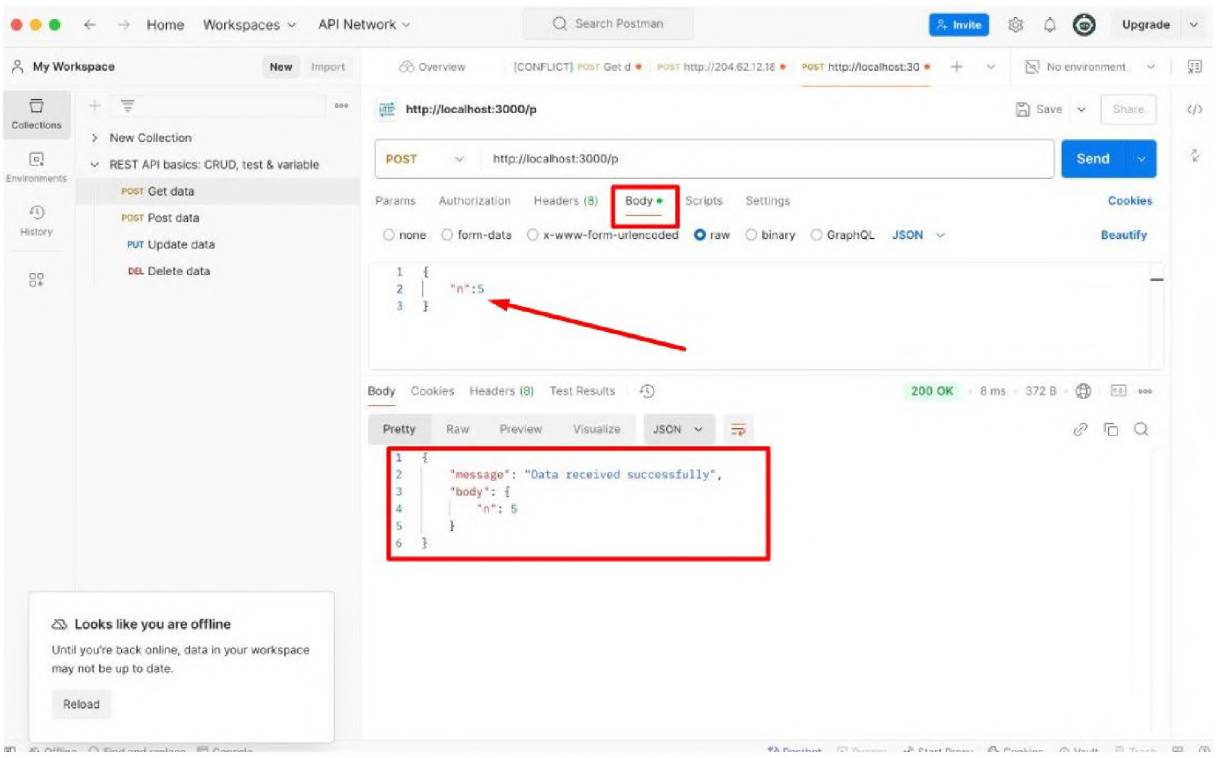


Рисунок 2.19 – Виконання запиту через Postman

Синхронізація сервера та виконання віртуальних запитів у Node.js за допомогою нейропомічників пропонують потужні можливості для створення інтелектуальних програм, які можуть швидко й ефективно виконувати запити в реальному часі. Завдяки асинхронній моделі Node.js ці програми можуть легко взаємодіяти з різними зовнішніми інтерфейсами програмування додатків, базами даних та іншими службами, забезпечуючи швидке маніпулювання даними без затримок. Нейропомічники в поєднанні з обробкою природної мови та інструментами машинного навчання забезпечують більш точні та важливі результати, покращуючи зручність використання.

Висновки до розділу 2

У другому розділі розглядаються принципи проєктування та характеристики написання коду для вебконтенту з використанням нейронних помічників у поєднанні з сучасними бібліотеками та платформами.

Зокрема, вони досліджували, як нейропомічник взаємодіє з React.js і Node.js, представляючи способи оптимізації їх продуктивності та поведінки створення віртуальних запитів для обробки даних у реальному часі.

Це стосується того, як інтеграція нейропомічника з React.js дозволяє покращити користувацькі інтерфейси, роблячи їх більш інтуїтивно зрозумілими та гнучкими, що покращує досвід роботи з програмою. Він пояснює, як налаштувати службу нейропомічника в React.js, щоб забезпечити швидку та легку обробку даних.

Інтеграція нейропомічника з React.js дозволяє розширити функціональність і можливості інтерфейсу користувача. React.js відомий своїми функціями та швидкими оновленнями, він стає ще потужнішим, коли мова йде про можливості нейронної мережі. Нейропомічники можуть обробляти великі обсяги даних, вивчати поведінку користувачів і надавати персоналізовані рекомендації, роблячи вебдодаток ефективнішим.

Підпункт, присвячений Node.js, досліджує способи використання нейропомічників для створення інтелектуальних серверних рішень, які можуть обробляти численні запити та обчислення в реальному часі. Node.js з його асинхронною моделлю обробки запитів і подій (Event Loop) є ідеальною основою для роботи нейронних помічників, які можуть виконувати складні обчислення, аналізувати дані та взаємодіяти з тими, хто його використовує.

У підсумку, розділ демонструє, як поєднання нейропомічників із сучасними вебтехнологіями дозволяє створювати продуктивні, інтерактивні та масштабовані вебдодатки, які відповідають сучасним вимогам користувачів та бізнесу.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО ЗАСТОСУВАННЯ НЕЙРОПОМІЧНИКА ДЛЯ РОЗРОБКИ ВЕБКОНТЕНТУ

3.1 Використання бібліотеки React та кросплатформи Node.js

React може бути використаний для створення фронтенду (клієнтської частини) вебдодатка, а Node.js – для створення серверної частини, яка оброблятиме запити та надаватиме дані (рис. 3.1). Цей підхід дозволяє створювати сучасні вебдодатки, де клієнтська та серверна частини чітко розділені, але працюють разом для забезпечення повної функціональності [34].

Серверна частина Node.js включає:

– використання фреймворків таких Express, або інші для обробки HTTP запитів;

– створювання REST API, яке React.js використовує для взаємодії з сервером;

– обробляння запитів від клієнта та надає відповіді;

– роботу з базами даних.

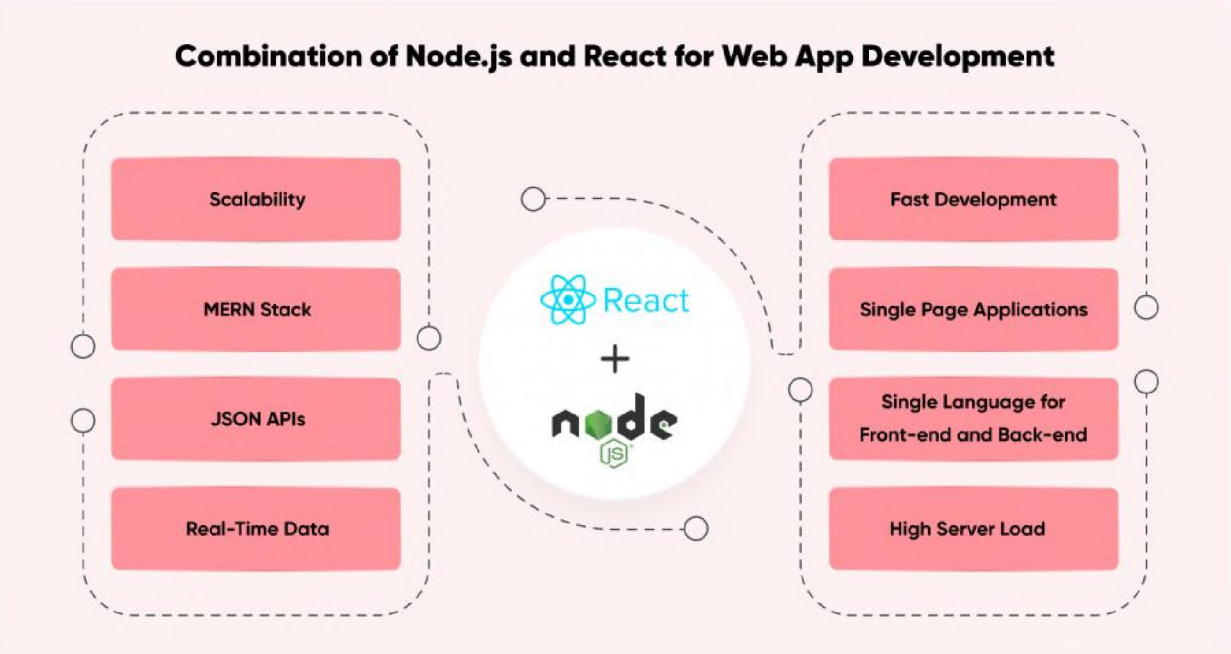


Рисунок 3.1 – Відображення взаємодії між React.js та Node.js

Клієнтська частина React.js включає:

– створення інтерфейсу користувача;

– реалізація HTTP запитів, в основному через модулі fetch або axios;

– використання бібліотеки для управління станом (Redux) або контекст для зберігання даних.

Коли ці дві технології працюють разом, Node.js зазвичай використовується для обробки запитів від клієнта (це можуть бути запити до бази даних, автентифікації або іншої серверної логіки), а React.js – для відображення результатів цих запитів у вебінтерфейсі. Наприклад, користувач взаємодіє з інтерфейсом на React, який надсилає запит на сервер (Node.js), а сервер обробляє його і відправляє відповідь назад (додаток А, рис. А.8 та рис. А.9).

Таким чином, з'єднання Node.js і React.js дозволяє створювати повноцінні вебзастосунки, де одна технологія відповідає за серверну логіку, а інша – за взаємодію з користувачем.

Дослідження даної роботи є створення вебдодатка на цих аспектах та за допомогою нейропомічника для тестування та виявлення помилок чи інших несправностей.

Додаток буде включати детальна авторизація користувача, де користувача в разі випадку що забув пароль може відновити його, при успішному вході в систему він зможе ознайомитися з інформацією та замовити товари.

Для реалізації даної задачі використав такі інструменти: React.js, Node.js, Render, Express.js, Netlify, Clever Cloud, MySQL.

Clever Cloud – це європейська платформа як послуга (PaaS), яка допомагає командам розробників швидко та безпечно розгортати цифрові додатки та сервіси. Вона автоматизує процеси розгортання, масштабування та обслуговування додатків, дозволяючи зосередитися на розробці без необхідності керувати інфраструктурою (рис. 3.2).

Платформа підтримує понад 25 мов програмування, включаючи Java, Node.js, PHP, Python та інші, а також надає керовані бази даних, такі як MySQL, PostgreSQL, MongoDB та Redis.

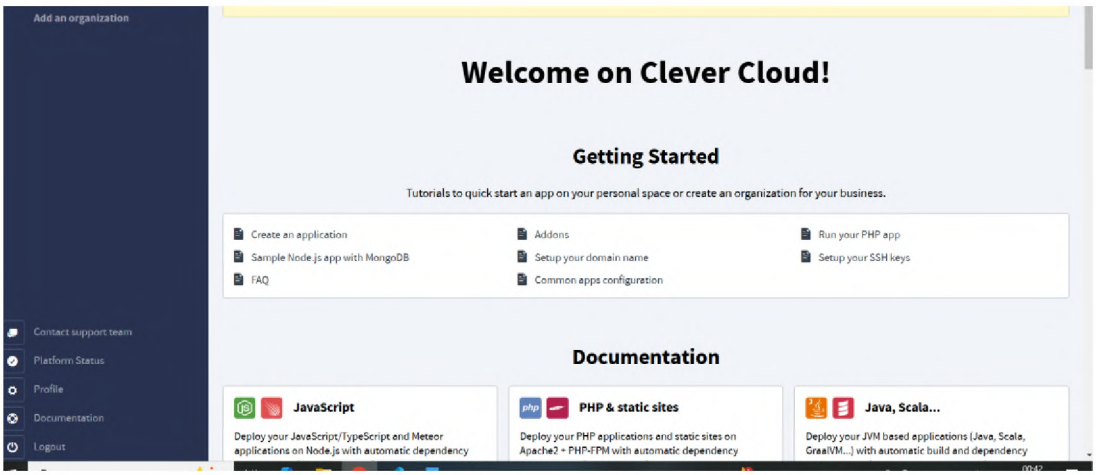


Рисунок 3.2 – Інтерфейс платформи Clever Clound

Clever Cloud забезпечує автоматичне масштабування додатків, високу доступність та безпеку, а також прозоре ціноутворення з оплатою за фактичне використання ресурсів.

Компанія базується в Європі та активно підтримує цифровий суверенітет, співпрацюючи з європейськими технологічними партнерами та дотримуючись високих стандартів безпеки та конфіденційності даних.

Clever Cloud активно використовується для створення вебсайтів, API, мобільних додатків, мікросервісів і великих корпоративних систем. Завдяки можливостям безперервного розгортання (Continuous Deployment) платформа інтегрується з популярними системами управління версіями, такими як GitHub, GitLab або Bitbucket.Clever Cloud ідеально підходить для стартапів, малих і середніх компаній, які шукають просте і надійне рішення для розгортання додатків без необхідності витрачати час на управління інфраструктурою.

Clever Cloud також інтегрує з нейропомічником який оптимізував витрати. Він аналізує, скільки ресурсів використовує ваш додаток, і пропонує способи заощадження, наприклад, змінюючи налаштування або вибираючи дешевші ресурси. Це економить час і гроші. Коли виникають помилки чи проблеми, нейропомічник швидко знаходить їх і пояснює, що саме сталося, а також підказує, як це виправити. Вам не потрібно бути експертом у всіх технічних деталях – платформа вже враховує ваші потреби та намагається спростити роботу користувача.

Завдяки інтеграції зі штучним інтелектом Clever Cloud стає більш гнучким і зручним, дозволяючи зосередитися на розробці додатків, а не на управлінні інфраструктурою.

3.2 Створення інтерактивних компонентів в React.js та Node.js за допомогою нейропомічника

Для створення вебдодатка, створив форму авторизації де кожен клієнт може авторизуватися (рис. 3.3). Авторизація включає поля для реєстрації: username, login, email, password.

Спочатку відправляється запит даних на сервер Node.js та отримує відповідь (рис. 3.4). Дані коли передаються на сервер перевіряються чи існує такий користувач, якщо існує може видати помилку, якщо ні, то користувач отримає JSON Web Token (рис. 3.5).

JSON Web Token (JWT) – це спосіб обміну даними між двома сторонами (наприклад, клієнтом і сервером) у вигляді компактного і безпечного токена. Його часто використовують для аутентифікації в вебдодатках [35]. Після того як користувач входить в систему, сервер генерує цей токен і відправляє його назад клієнту. Клієнт потім використовує цей токен для доступу до захищених частин вебсайту чи сервісу, надсилаючи його кожного разу в запитах.

JSON Web Token складається з трьох частин: заголовка, тіла і підпису. Заголовок містить інформацію про тип токена та алгоритм шифрування.

Перша частина – це заголовок, який містить інформацію про тип токена (зазвичай це JWT) та алгоритм, що використовується для його захисту, наприклад, HS256.

Друга частина – тіло, де зберігаються основні дані, такі як ідентифікатор користувача, термін дії токена або інша важлива інформація. Ці дані не зашифровані, тому їх можна прочитати, якщо декодувати токен.

Третя частина – це підпис, який гарантує, що дані не були змінені.

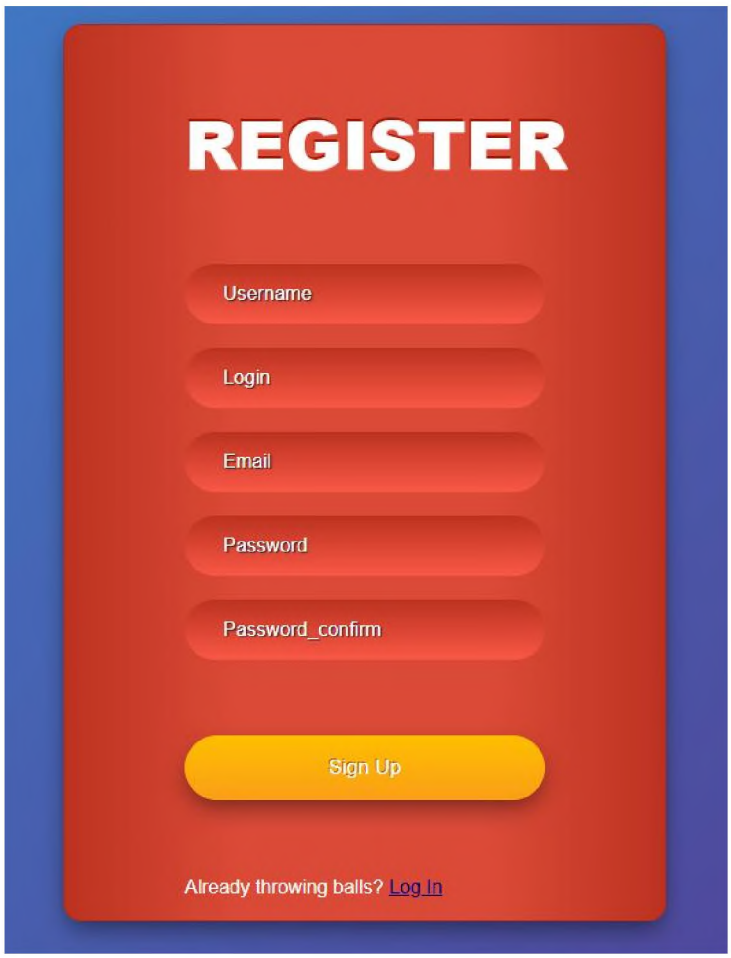


Рисунок 3.3 – Форма авторизації на React.js

```
fetch('http://localhost:5000/register', {
    method: 'POST',
    headers: {
        'Content-type': 'application/json'
    },
    body: JSON.stringify({
        username: state.username,
        login: state.login,
        email: state.email,
        password: state.password,
        confirmPassword: state.confirmPassword
    })
})
.then((response) => {
    if (response.status === 200) {
        SetRegister(true)
        SetIsUser(false)
    }

     else {
        SetIsUser(true)
    }
    return response.json();
})
.then((data) => {
    console.log(data);
```

Рисунок 3.4 – Запит на авторизацію на React.js

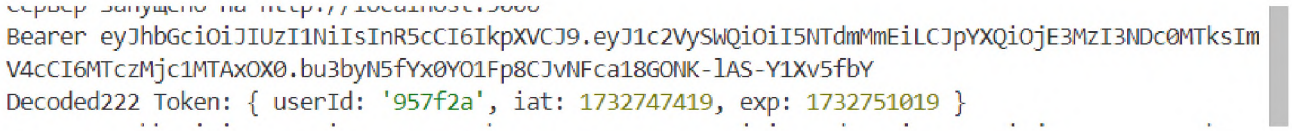


Рисунок 3.5 – Реєстрація даних на сервері

Після того, як отримали дані сервер записує дані в базу даних MySQL яка була створена Clever Cloud (рис. 3.6) більш детально про це хмарне сховище розписано в минулому підпункті.

Структура таблицы | Связи

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Комментарии	Дополнительно	Д
1	username	varchar(255)	utf8_general_ci		Нет	Нем			
2	login	varchar(255)	utf8_general_ci		Нет	Нем			
3	email	varchar(50)	utf8_general_ci		Нет	Нем			
4	password	varchar(8)	utf8_general_ci		Нет	Нем			
5	code_user	varchar(6)	utf8_general_ci		Нет	Нем			
6	token	varchar(255)	utf8_general_ci		Да	NULL			

Рисунок 3.6 – Структура таблиці register в базі даних MySQL

Для успішної авторизації користувач отримує код підтвердження на вказаний email (рис. 3.7), якщо указана пошта при реєстрації вірна, то код прийде, і ви повинні увести його, бо інакше система не пропустить (рис. 3.8).

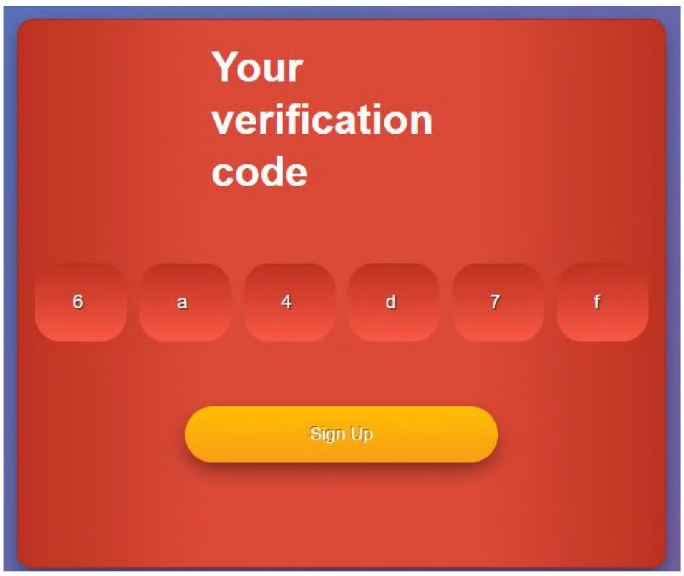


Рисунок 3.7 – Приклад уведення коду авторизації

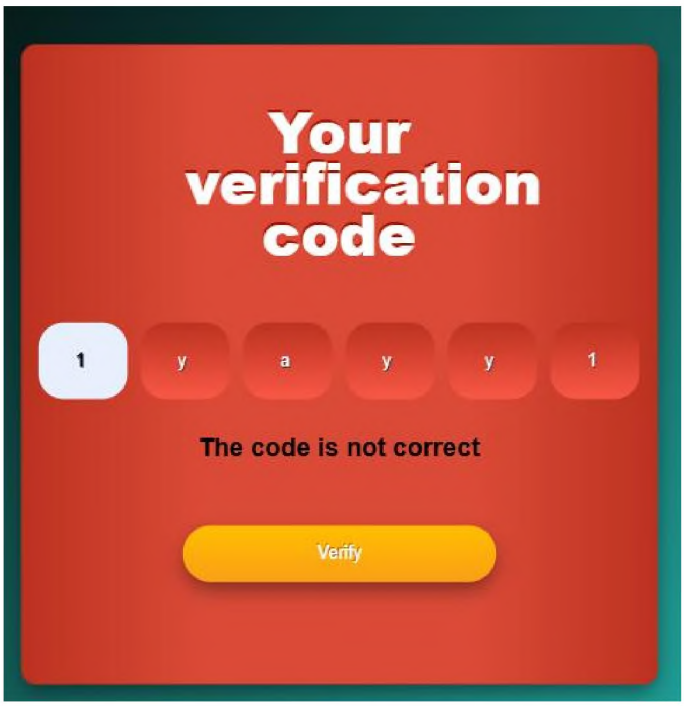


Рисунок 3.8 – Приклада неправильного коду авторизації

Тепер користувач зареєструвався і дані його знаходяться в базі даних (рис. 3.9). Дана база даних складається з трьох таблиць register, register_token, register_user (рис. 3.10). Таблиця register відповідає за реєстрованих користувачів при формі реєстрації (див. рис. 3.6). Наступна таблиця зберігає всі авторизовані токени (рис. 3.11). Остання таблиця для авторизації в нас буде register_user, де зберігається код підтвердження (рис. 3.12).

	username	login	email	password	code_user	token
Удалить	Den25	11	dedomelchenko2@gmail.com	1	11d47d	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV
Удалить	Den2323	1123	dedomelchenko2dg@gmail.com	4	7aa906	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV
Удалить	yaya	115	dedomelchenko4@gmail.com	4	6a4d7f	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV
Удалить	Den258	11e	dedomelchenko2e@gmail.com	1	bf44f0	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV
Удалить	Den2	dedomelchenko@gmail.com	dedomelchenko@gmail.com	5	0ee206	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV
Удалить	Den259	j	dedomelchenko8@gmail.com	5	71cd96	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV

Рисунок 3.9 – Список зареєстрованих користувачів в базі даних

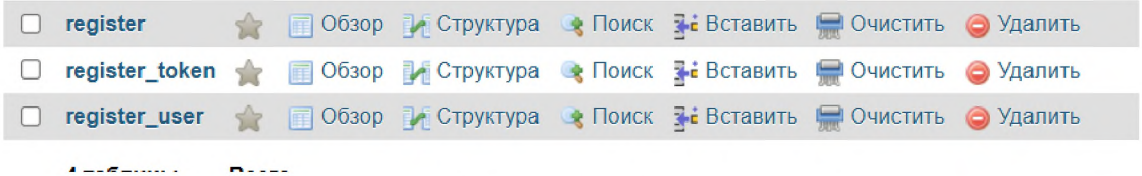


Рисунок 3.10 – Список таблиць в базі даних

Рисунок 3.11 – Фрагмент таблиці авторизованих токенів в базі даних

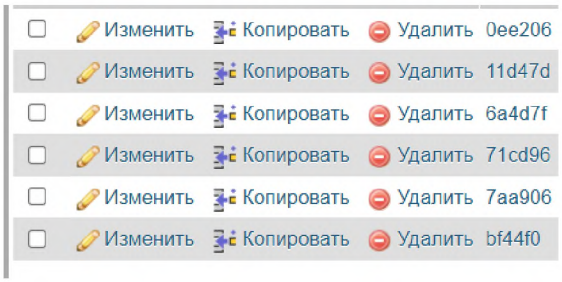


Рисунок 3.12 – Зберігання кодів підтвердження в базі даних

Таблиці register_user та register_token зв'язані з register, за допомогою індексу, тобто коли дані міняються в при авторизації, чи токен, чи код підтвердження автоматично міняється і в таблиці register.

При логізації дані беруться з таблиці register (рис. 3.13), якщо дані уведені правильно, то користувач також отримує листа з кодом для логування.

```
router.post('/login', async (req, res) => {
   const {username, login, email, password} = req.body

    try {
        const [registerUser] = await connection.query(
            `SELECT username, login, email, password FROM register `);

        let isUser = false
        let isError = false

        for (let i = 0; i < registerUser.length; i++) {
            let user = registerUser[i];
            let isMatch = true;
            for (let key in user) {
                if ( user[key] !== req.body[key]) {
                    isError = true;
                    isMatch = false;
                    break;

                }
            }
            if (isMatch) {
                isError = false
                isUser = true;
                break;
            }
        }
```

Рисунок 3.13 – Приклад реального SQL запиту при логуванні користувача

Після авторизації токен кожен раз обновляється в даній реалізації, його валідність перевіряється на сервері, якщо токен не валідний отримуємо повідомлення від сервера що токен не валідний (рис. 3.14).

```
router.post('/verifyToken', async (req, res) => {
    const token = req.headers['authorization'];

    if (!token) {
        return res.status(403).json({ message: 'Token undefined' });
    }

    jwt.verify(token.split(' ')[1], 'secretKey', (err, decodedToken) => {
        if (err) {
            return res.status(403).json({ message: 'Invalid token' });
        } else {
            try {
                console.log('Decoded Token:', decodedToken);
                const userId = decodedToken.userId;
                console.log('User ID:', userId);

                return res.status(200).json({ message: 'Token updated successfully' });
            } catch (error) {
                if (error instanceof jwt.TokenExpiredError) {
                    return res.status(400).json({ message: 'Token expired' });
                }
            }
        }

    });
});
```

Рисунок 3.14 – Перевірка валідності токена

Кожен користувач має можливість змінити свій пароль, дотримуючись певної процедури [36]. Спочатку користувач повинен авторизуватися для того, щоб змінити пароль в системі. Після реєстрації користувач переходить на логізацію де уводить вхідні дані, саме там можна змінити пароль. Щоб змінити пароль натискаємо на кнопку Forgot your password (рис. 3.15).

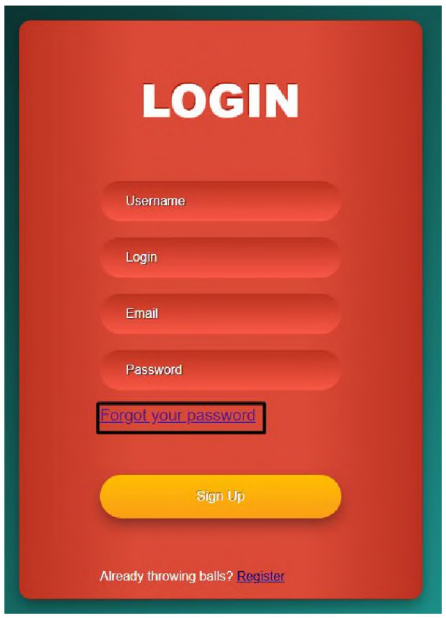


Рисунок 3.15 – Скидання пароля в додатку

Після натискання на зміну пароля користувач переходить на форму, і повинен увести email (рис. 3.16), який є в базі даних зареєстрованих користувачів (див. рис. 3.11), але якщо користувач випадковим чином чи не випадковим спробує увести відсутній email в базі даних то отримає помилку (рис. 3.17).

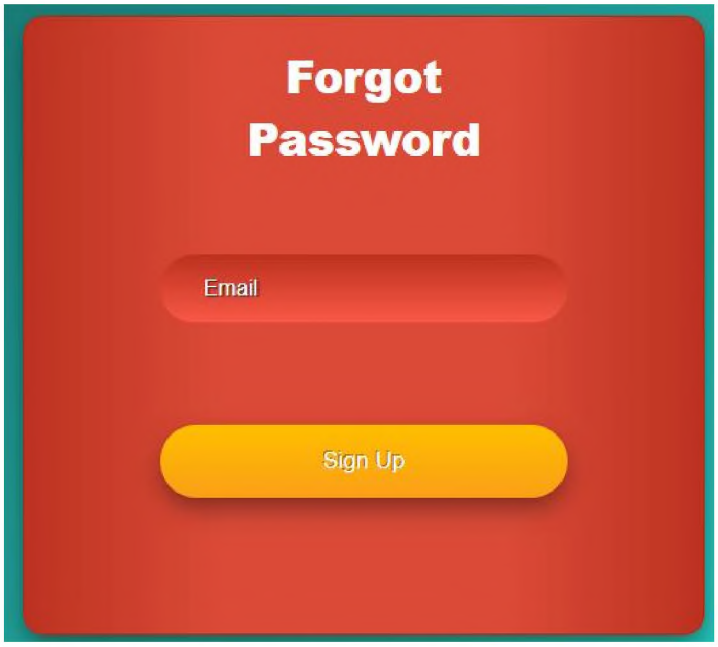


Рисунок 3.16 – Форма для наявного email в базі даних

Рисунок 3.17 – Помилка користувача при не правильному уводі даних

Якщо користувач увів правильний пароль, то він отримає таке повідомлення (рис. 3.18).

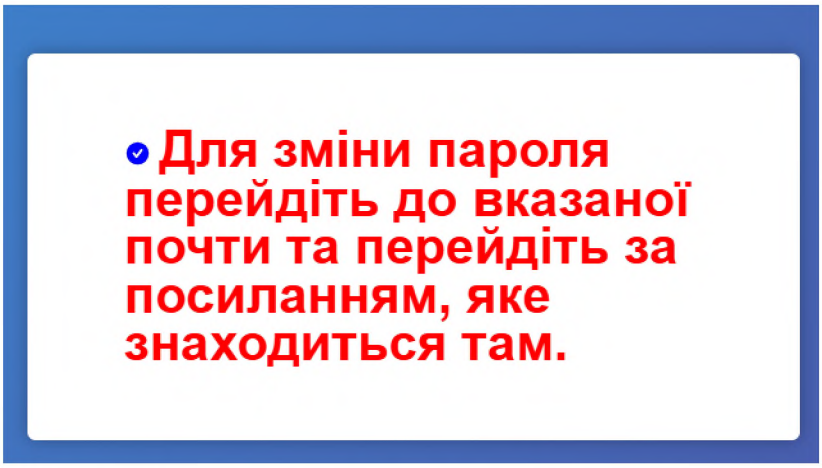


Рисунок 3.18 – Повідомлення про успішне відправлення даних на email

Тепер кожен користувач отримає повідомлення на email, воно буде містити в собі посилання на нову сторінку для зміни пароля (рис. 3.19), якщо повідомлення немає у «Вхідних», то воно значить знаходиться в «Спамі».

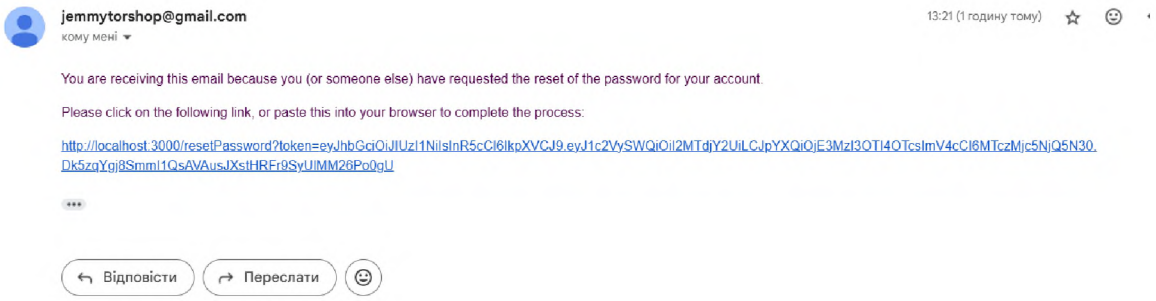


Рисунок 3.19 – Повідомлення на email для зміни пароля

Перейшов за посиланням яке знаходиться на email, користувач перейшов на форму (рис. 3.20), де є поля old password, new password, confirm password. Кожне поле повинно бути заповнене та коректно уведене. Для перевірки, щоб пароль зміниться в базі даних візьмемо користувача який зареєструвався і його email має значення omelchenko4562@gmail.com і він має пароль tryn4 (рис. 3.21) і після зміни пароля буде відображатися в базі даних новий пароль даного користувача за email (рис. 3.22 та 3.23).

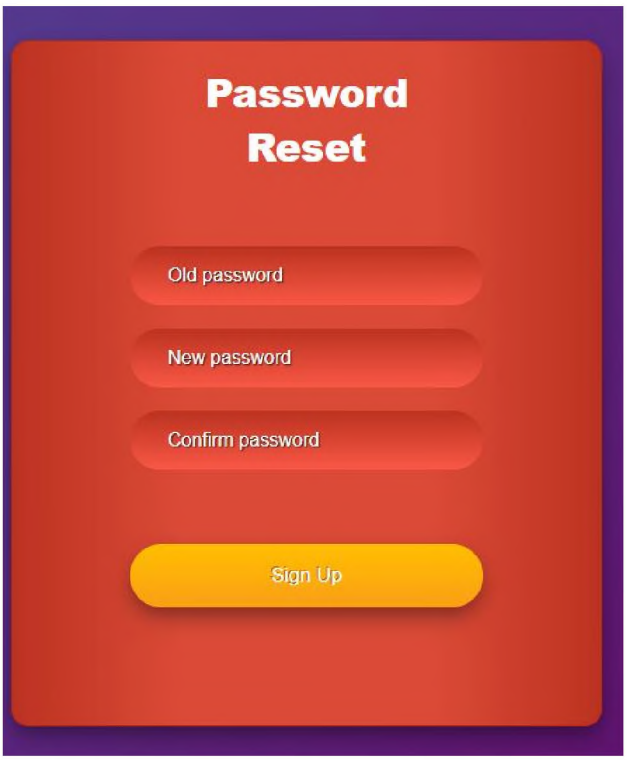


Рисунок 3.20 – Повідомлення на email для зміни пароля

	username	login	email	password
Изменить Копировать Удалить	Den25	11	dedomelchenko2@gmail.com	5
Изменить Копировать Удалить	Den2323	1123	dedomelchenko2dg@gmail.com	4
Изменить Копировать Удалить	yaya	115	dedomelchenko4@gmail.com	1
Изменить Копировать Удалить	Den258	11e	dedomelchenko2e@gmail.com	1
Изменить Копировать Удалить	Den2	dedomelchenko@gmail.com	dedomelchenko@gmail.com	5
Изменить Копировать Удалить	omelchenko99den	h	omelchenko4562@gmail.com	tryn4

Рисунок 3.21 – Користувач omelchenko4562@gmail.com до зміни пароля

Після успішної зміни пароля користувач отримує підтвердження про те, що його пароль було оновлено. Відтепер при вході в систему слід використовувати новий пароль.

Для тестування написаного коду я використав нейропомічника ChatGPT, який дозволяє ефективно оцінити якість реалізації та знайти можливі проблеми. Цей підхід допоміг виявити помилки, які могли залишитися непоміченими, і запропонувати рішення для їх усунення. ChatGPT також аналізує логіку роботи коду, перевіряє його на відповідність вимогам і допомагає уникнути потенційних помилок [37].

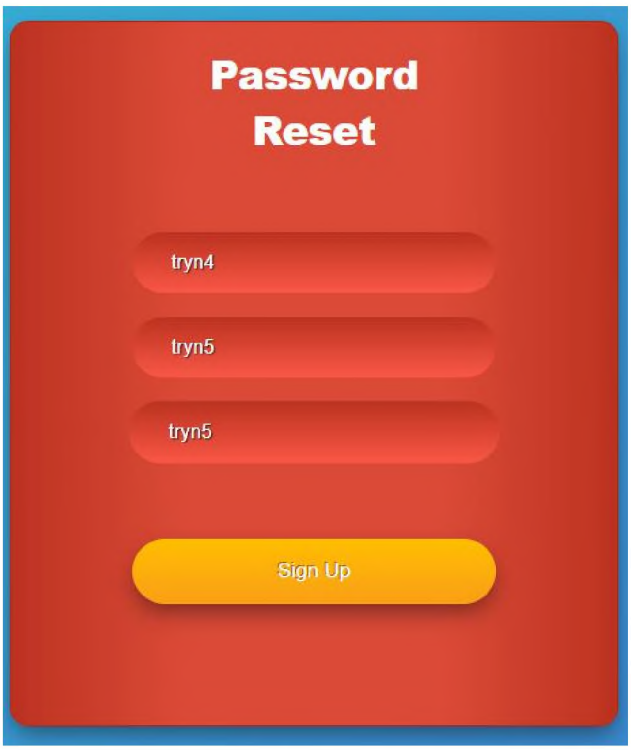


Рисунок 3.22 – Зміна пароля для користувача

☐	Изменить	Копировать	Удалить	Den25	11	dedomelchenko2@gmail.com	5
☐	Изменить	Копировать	Удалить	Den2323	1123	dedomelchenko2dg@gmail.com	4
☐	Изменить	Копировать	Удалить	yaya	115	dedomelchenko4@gmail.com	1
☐	Изменить	Копировать	Удалить	Den258	11e	dedomelchenko2e@gmail.com	1
☐	Изменить	Копировать	Удалить	Den2	dedomelchenko@gmail.com	dedomelchenko@gmail.com	5
☐	Изменить	Копировать	Удалить	omelchenko99den	h	omelchenko4562@gmail.com	tryn5

Рисунок 3.23 – Відображення зміни пароля в базі даних

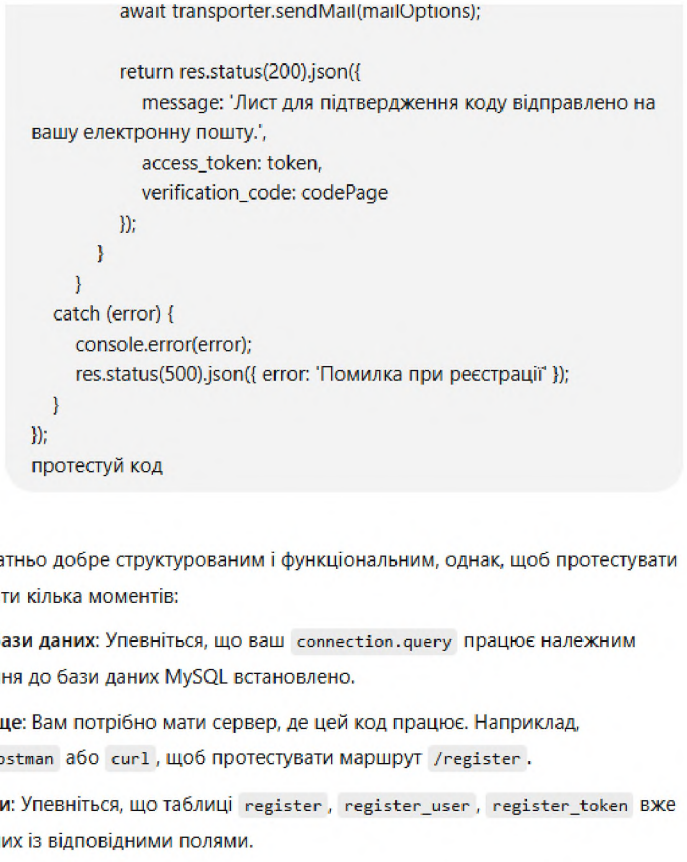
Окрім цього, нейропомічник надає рекомендації щодо оптимізації. Наприклад, він може вказати на надлишковий код, що сповільнює роботу, або запропонувати альтернативні методи реалізації, які покращать продуктивність та читабельність. Якщо певна частина коду є складною для розуміння чи використання, ChatGPT пояснює, як її можна зробити більш інтуїтивною та зрозумілою.

Також під час тестування я отримав поради щодо поліпшення архітектури додатка. Наприклад, нейропомічник може запропонувати. Для тестування форми авторизації використання ChatGPT дозволяє ретельно перевірити логіку її роботи та функціонал [38].

Спочатку я передав у нейропомічник код форми, щоб переконатися, що всі поля обробляються коректно. Наприклад, перевірялися такі аспекти, як правильна валідація введених даних, коректне відображення повідомлень про помилки та відповідність структури коду сучасним стандартам розробки. ChatGPT допоміг протестувати сценарії введення валідних і не валідних даних, таких як порожнє поле, неправильний формат електронної пошти або пароль, що не відповідає мінімальним вимогам.

Для тестування бекенд частини використав саме цього нейропомічника (рис. 3.24). Нейропомічник допоміг проаналізувати логіку роботи API, обробку запитів і відповідей, а також забезпечити дотримання найкращих практик розробки серверної частини.

Під час тестування я передавав код маршрутизації, логіку контролерів, моделі баз даних і середовища налаштувань. ChatGPT проаналізував кожен компонент, звернув увагу на потенційні вузькі місця та надав рекомендації, як їх виправити.



```
            await transporter.sendMail(mailOptions);

            return res.status(200).json({
              message: 'Лист для підтвердження коду відправлено на
вашу електронну пошту.',
              access_token: token,
              verification_code: codePage
            });
          }
        }
      catch (error) {
        console.error(error);
        res.status(500).json({ error: 'Помилка при реєстрації' });
      }
});
протестуй код
```

Ваш код виглядає достатньо добре структурованим і функціональним, однак, щоб протестувати його, потрібно врахувати кілька моментів:

1. **Підключення до бази даних**: Упевніться, що ваш `connection.query` працює належним чином і підключення до бази даних MySQL встановлено.
2. **Тестове середовище**: Вам потрібно мати сервер, де цей код працює. Наприклад, використовуйте `Postman` або `curl`, щоб протестувати маршрут `/register`.
3. **Дані для перевірки**: Упевніться, що таблиці `register`, `register_user`, `register_token` вже створені в базі даних із відповідними полями.

Рисунок 3.24 – Тестування коду за допомогою нейропомічника ChatGPT

У цьому підрозділі розписана детальна процедура авторизації користувача. Спочатку користувач має ввести свій login або email, а також поточний пароль у відповідні поля форми авторизації. Система перевіряє введені дані на відповідність зареєстрованим у базі даних. Якщо login і password збігаються зі збереженими даними, користувач успішно проходить авторизацію та отримує доступ до свого облікового запису.

3.3 Програмна реалізація вебдодатка

Після успішної авторизації користувач автоматично перенаправляється до головного меню додатка, яке є стартовою точкою для подальшої роботи [39]. На цьому етапі сервер перевіряє правильність введених даних, а у відповідь клієнту надсилається токен, що підтверджує його особу. Цей токен використовується для забезпечення безпеки та збереження стану авторизації протягом роботи в додатку.

На головній сторінці розміщена інформація для ознайомлення, яка включає основні відомості про компанію, послуги або товари, що пропонуються. Вся інформація представлена в лаконічному вигляді (рис. 3.25) щоб користувачі могли швидко ознайомитися з основними аспектами діяльності чи пропозицій, що є на сайті.

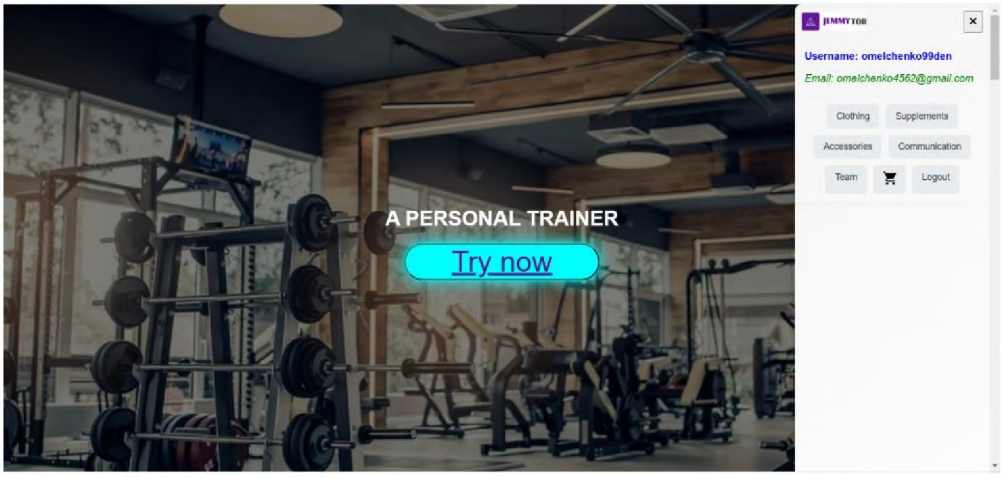


Рисунок 3.25 – Головне меню додатка

Наступним компонентом буде інформація про сайт, яка надає детальніше пояснення про його призначення та функціональність. Цей розділ може включати історію створення сайту, його основні цілі та переваги для користувачів. Тут також можна зазначити ключові можливості ресурсу, його унікальні особливості, а також, можливо, місію або цінності, які він прагне реалізувати. Інформація має бути структурована так, щоб користувачі могли швидко зрозуміти, чому варто використовувати цей сайт і як він може допомогти вирішити їхні потреби (рис. 3. 26 – 3.30).

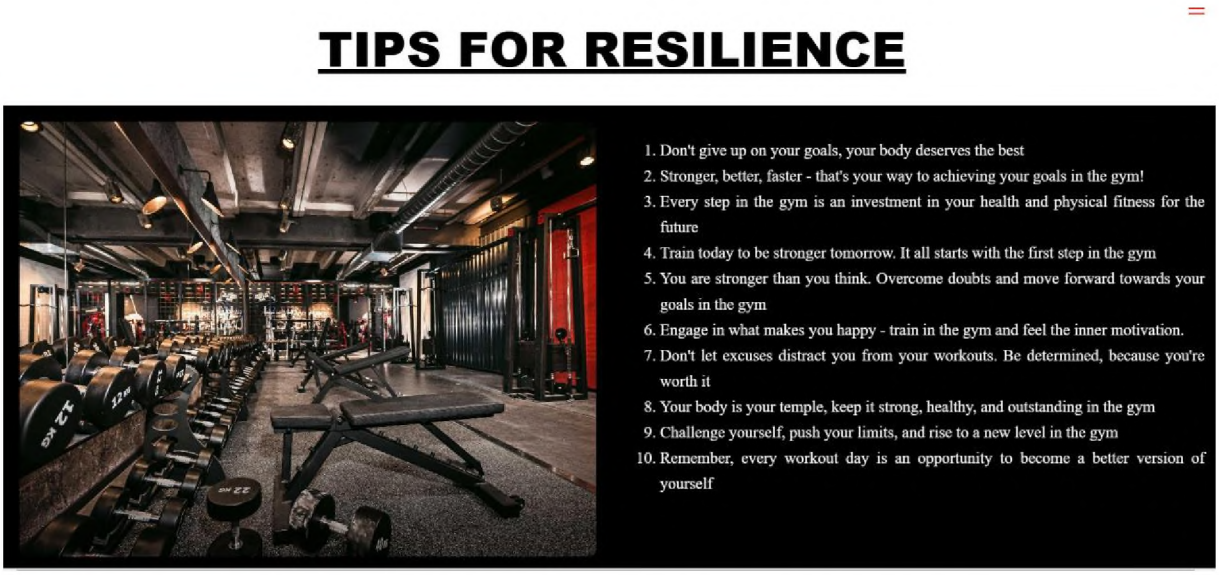


Рисунок 3.26 – Інформація для користувача

Рисунок 3.27 – Реалізація спортивного слайдера

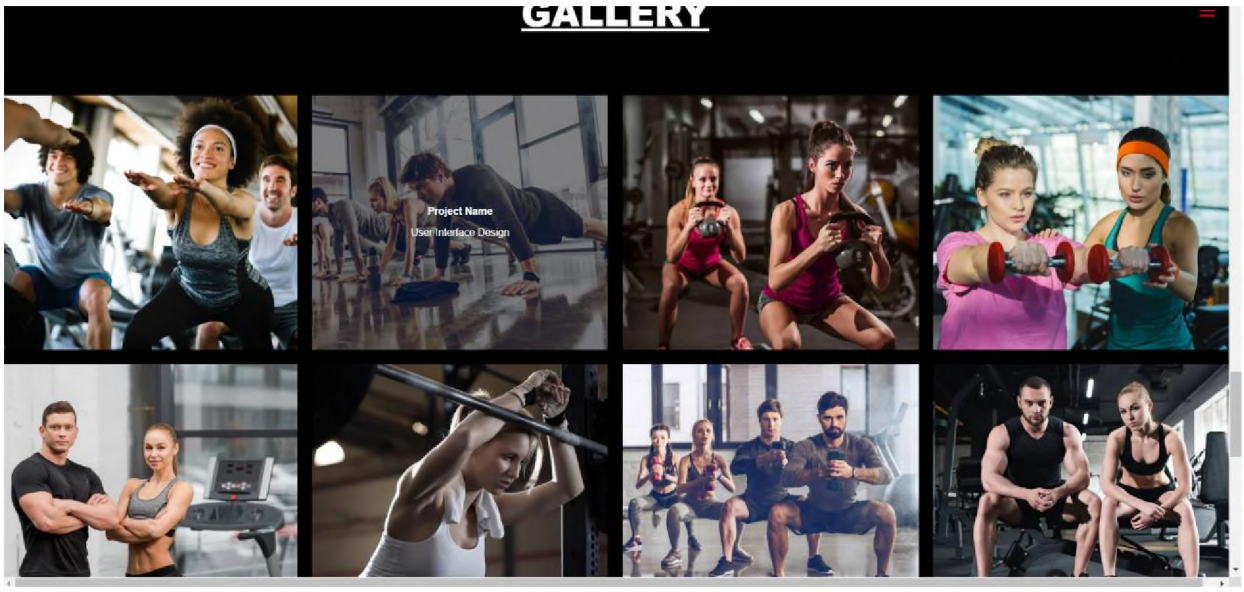


Рисунок 3.28 – Галерея картинок

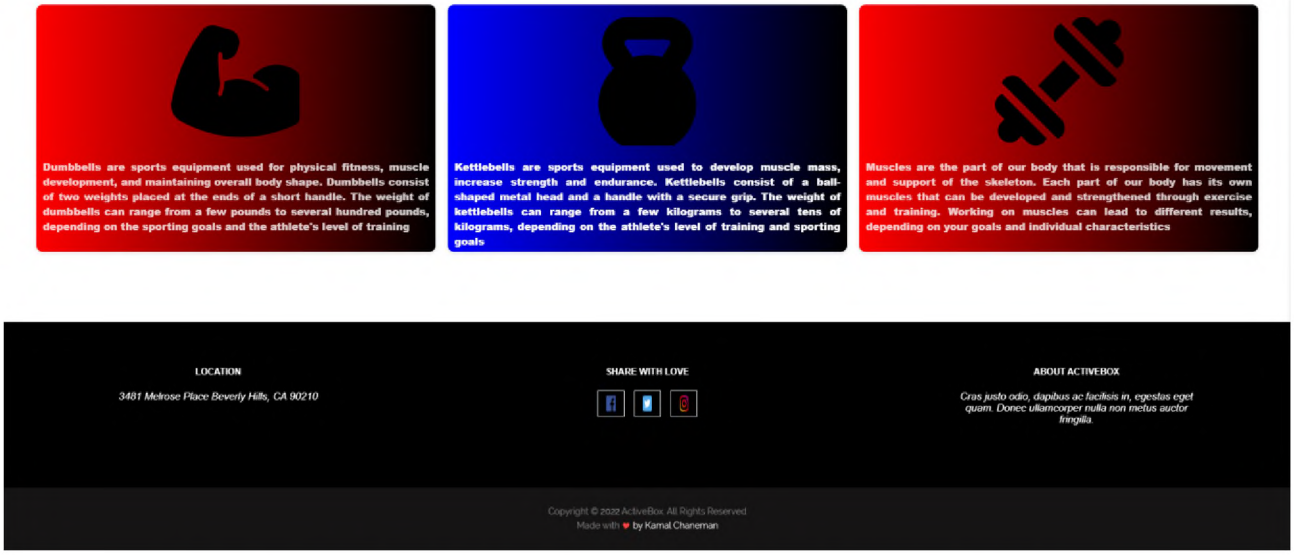


Рисунок 3.29 – Колонтитул та інформація для користувача

Користувач може купити будь-який товар із сайту. Наприклад, на вкладці «Clothing» (рис. 3.30) він обирає потрібний товар.

Після переходу на сторінку товару користувач може переглянути його деталі (рис. 3.31), такі як опис, ціна, назва тощо. Якщо товар підходить, його можна додати до кошика.

Далі залишається перейти до кошика, оформити замовлення, заповнивши дані, вказавши певні дані для зв'язку з клієнтом потрібно вказати ім'я, електронну пошту, номер телефону та адресу.

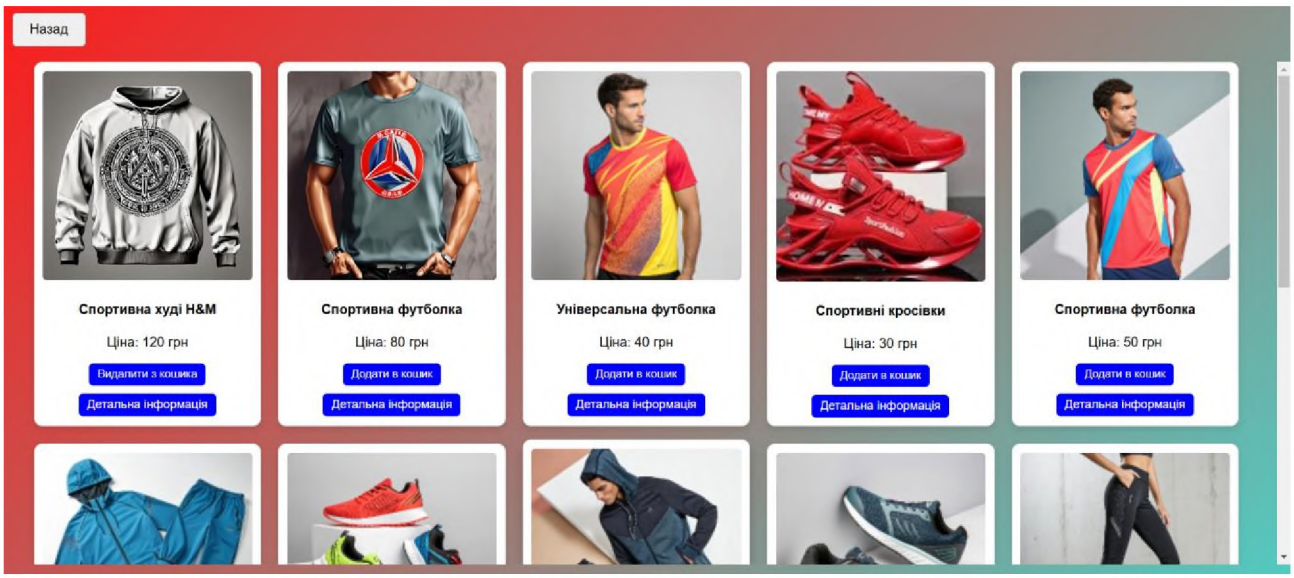


Рисунок 3.30 – Перегляд товарів у вкладці «Clothing»

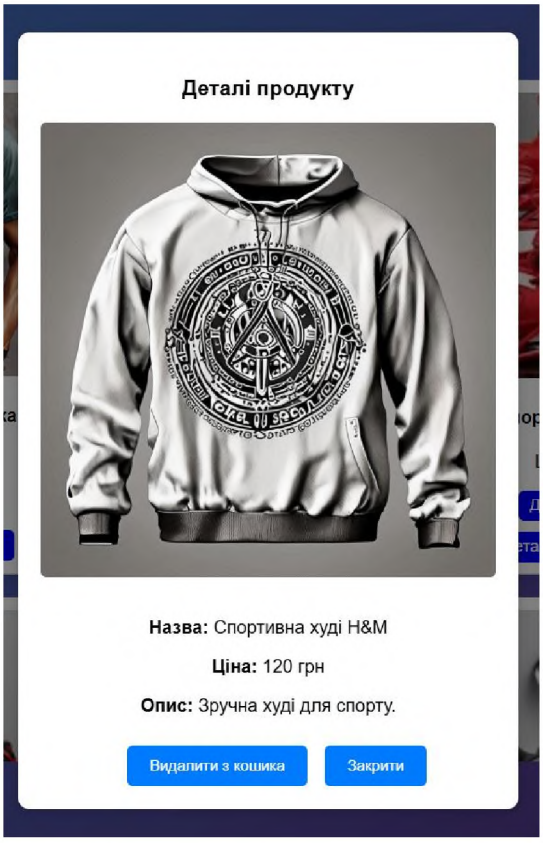


Рисунок 3.31 – Детальний перегляд товару

Коли користувач хоче купити товар, він переходить до кошика. Там він вводить своє ім'я, номер телефону, адрес доставлення, електронну пошту для підтвердження замовлення та вибирає необхідну кількість товару. Після заповнення цих даних замовлення оформлюється (рис. 3.32).

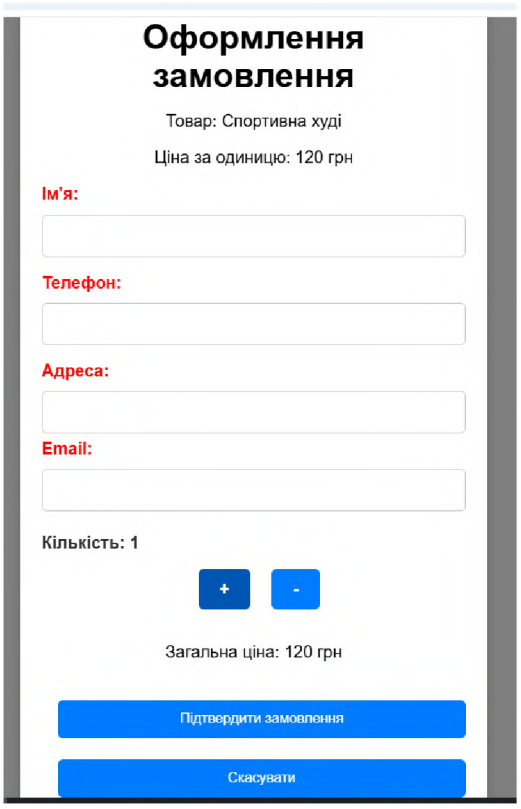


Рисунок 3.32 – Оформлення замовлення користувача

Коли користувач оформлює замовлення, всі введені дані, такі як ім'я, телефон, адреса, електронна пошта та кількість товару, зберігаються в базі даних під назвою «buy» (рис. 3.33).

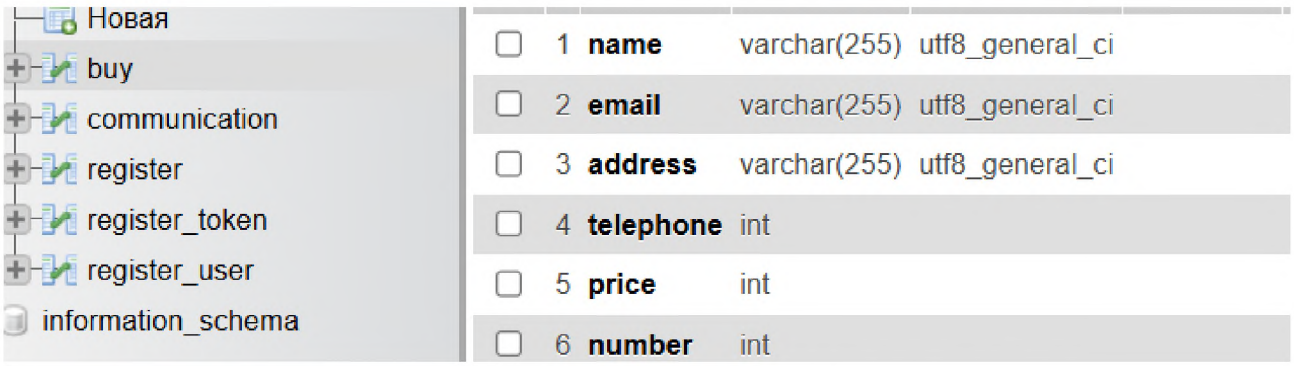


Рисунок 3.33 – Запис замовлення в базу даних buy

Оформити можна не тільки товари з одягом, але з іншими такими як: добавки, аксесуари.

Якщо виникають певні питання є форма зв'язку, де ви залишаєте свої контакти, і з користувачем повинні зв'язатися представники компанії (рис. 3.34).

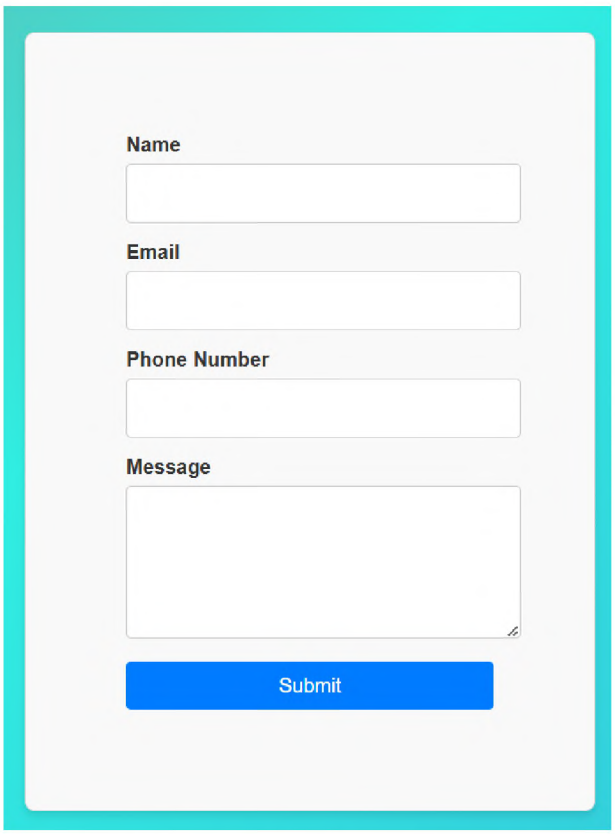


Рисунок 3.34 – Форма зв'язку з клієнтом

3.4 Економічне обґрунтування прийнятих рішень

Економічне обґрунтування використання нейропомічника для вебконтенту базується на аналізі його ефективності та впливу на процес розробки вебдодатків. Впровадження цього інструмента значно оптимізує робочий процес, що безпосередньо відображається на економії часу та ресурсів. Завдяки автоматизації багатьох рутинних завдань, нейропомічник дозволяє зменшити кількість людських помилок і скоротити обсяг ручної роботи, що особливо важливо під час розробки складних і масштабних проєктів.

Застосування нейропомічника також підвищує продуктивність команди розробників. Це досягається шляхом автоматичного генерування типових компонентів, полегшення тестування коду, а також швидкого пошуку та виправлення помилок [40]. Така оптимізація робочого процесу сприяє скороченню термінів розробки, що знижує загальні витрати на проєкт.

Окрім того, використання нейропомічника дозволяє більш ефективно розподілити ресурси. Розробники можуть зосередитися на вирішенні творчих і технічно складних завдань, тоді як рутинні процеси виконує помічник [41]. Це сприяє підвищенню якості кінцевого продукту, оскільки більше уваги приділяється деталям, що впливають на функціональність та зручність для користувачів.

Ще одним важливим аспектом є економія коштів на навчанні персоналу. Нейропомічник може служити інструментом підтримки для менш досвідчених розробників, забезпечуючи підказки та допомогу в реальному часі. Це дозволяє знизити витрати на залучення додаткових спеціалістів або на проведення навчальних програм.

Нижче описано обчислення економічного обґрунтування ефективності результатів. Оцінка інформаційні технології проводиться в два етапи. Перший етап включає оцінку витрат на впровадження вебдодатка, а другий включає оцінку витрат на проєкт впровадження.

$$B_П = B_{ТЗ} + B_{ППЗ} + B_{ОП} + B_{ВСЗ} + B_{ПСП} + B_У + B_{РПЗ} + B_І, \qquad (3.1)$$

де $B_П$ – прямі витрати на проєкт впровадження інформаційних технологій;

$B_{ТЗ}$ – витрати на придбання технічного забезпечення, грн;

$B_{ППЗ}$ – витрати на придбання програмного забезпечення, грн;

$B_{ОП}$ – витрати на оплату праці, грн;

$B_{ВСЗ}$ – витрати на відрахування на соціальні заходи, грн;

$B_{ПСП}$ – витрати за послуги, які виконують сторонні підприємства, грн;

$B_У$ – витрати на управління інформаційними технологіями, грн;

$B_{РПЗ}$–витрати на розробку прикладного програмного забезпечення власними силами, грн;

$B_І$ – інші прямі витрати на впровадження, грн.

Розв'язок. $B_П$ матиме такий вигляд: $B_П = B_{ТЗ} + B_{ППЗ} + B_{ОП} + B_{ВСЗ} + B_{ПСП} + B_У + B_{РПЗ} + B_І = 10000 + 5000 + 25000 + 10000 + 5000 + 10000 + 10000 + 2000 = = 77000$ *грн.*

Таким чином, Валовий прибуток ($В_П$) для спортивного комплексу складає *77000 грн.*

Оцінка непрямих витрат проєкту впровадження ($В_Н$), які є здійсненними поділяються на 2 групи, розраховуються за формулою (грн):

$$В_Н = В_{Н1} + В_{Н2}, \quad (3.2)$$

де $В_{Н1}$ – витрати, пов'язані з простоями ІС з ряду причин;

$В_{Н2}$ – витрати, пов'язані з людським фактором.

Розв'язок: $В_Н = 2000 + 3000 = 5000$ *грн.*

Витрати на утримання спортивного комплексу за період життєвого циклу. Річна вартість визначається за формулою:

$$В_{УТР} = В_{ОП} + В_{ВСЗ} + В_П + В_І, \quad (3.3)$$

де $В_{УТР}$ – щорічні витрати на утримання інформаційних технологій;

$В_{ОП}$ – витрати на оплату праці по підтримці та удосконаленню інформаційної системи, грн;

$В_{ВСЗ}$ – витрати на відрахування на соціальні заходи, грн;

$В_П$ – витрати за послуги сторонніх підприємств, грн;

$В_І$ – інші витрати на утримання інформаційної системи, грн.

Розв'язок: $В_{УТР} = 10000 + 3000 + 7000 + 2000 = 22000$ *грн.*

Виходячи з аналізу, зрозуміло, що загальна вартість проєкту буде визначатися за формулою:

$$В_{ІТ} = В_П + В_Н + В_{УТР} \quad (3.4)$$

де $В_{ІТ}$ – загальні витрати на проєкт впровадження інформаційних технологій.

Розв'язок: $В_{ІТ} = 77000 + 5000 + 22000 = 104000$ *грн.*

Оподаткування прибутку розраховується таким чином:

$$П_Р = Виручка - Затрати \quad (3.5)$$

Розв'язок: $П_Р = 124000 - 104000 = 20000$ *грн.*

Податок на прибуток:

$$Ппод = Пр \cdot Ст_{под} \quad (3.6)$$

де $Ст_{под}$ – ставка податку на прибуток (0,25 – індексне вираження відсотків по податку на прибуток з розрахунку 25 %).

Розв'язок: *Ппод = 20000 · 0,25 = 5000 грн.*

Тепер, розрахуємо чистий прибуток:

$$Пр_ч = Пр – П_{под} \quad (3.7)$$

Розв'язок: $Пр_ч = 20000 – 5000 = 15000$ *грн.*

Всі ці фактори сприяють економічному обґрунтуванню та підвищенню ефективності результатів досліджень при створенні спортивного комплексу з використанням React JS, Node.js, нейропомічники та додаткові хмарні технологогії. Допомагаючи зменшити витрати на розробку та технічне обслуговування, підвищити продуктивність системи, забезпечити швидке реагування на зміни та підвищити простоту використання [42].

Таким чином, впровадження нейропомічника в процес розробки вебконтенту є економічно вигідним рішенням, яке дозволяє значно підвищити ефективність роботи. Він оптимізує внутрішні процеси, знижує витрати на розробку завдяки автоматизації рутинних завдань та зменшенню кількості помилок. Це, в свою чергу, прискорює реалізацію проєктів та покращує їх якість. Нейропомічник також сприяє підвищенню продуктивності команд, дозволяючи розробникам зосередитись на більш складних і творчих аспектах роботи.

Висновки до розділу 3

У третьому розділі детально проаналізовано та реалізовано підхід до створення сучасних вебдодатків із використанням бібліотеки React і платформи Node.js. Зокрема, розглянуто інтеграцію цих технологій для забезпечення кросплатформенності та гнучкості в розробці, що дозволяє створювати динамічні інтерфейси, зручні для користувача.

Практична частина розділу акцентує увагу на розробці інтерактивних компонентів у середовищі React.js із залученням можливостей нейропомічника. Такий підхід дозволив не лише автоматизувати рутинні задачі, але й скоротити час, необхідний для створення функціональних елементів.

Зберігаючи при цьому високу якість коду, платформа Node.js, у свою чергу, забезпечила стабільну серверну частину, що сприяє ефективному управлінню даними та обробці запитів у реальному часі [43].

Програмна реалізація вебдодатка демонструє всі етапи його створення: від проєктування архітектури до впровадження інтерактивних функцій. Особливу увагу приділено адаптації нейропомічника для полегшення роботи з користувацькими запитами, оптимізації коду та інтеграції додаткових функціональних можливостей.

Також у розділі виконано економічне обґрунтування використання нейропомічника в контексті розробки вебконтенту. Доведено, що його впровадження дозволяє значно знизити витрати на розробку шляхом підвищення продуктивності команди, зменшення кількості помилок у коді та пришвидшення часу виконання завдань [44]. Це сприяє покращенню рентабельності проєкту та забезпечує конкурентні переваги.

Таким чином, результати цього розділу підкреслюють важливість використання сучасних технологій, таких як React і Node.js, у поєднанні з нейропомічником для створення інноваційних, економічно вигідних і функціональних вебрішень.

ВИСНОВКИ

У першому розділі було досліджено важливість нейропомічників як інструментів для покращення процесу відтворення вебконтенту. Нейропомічники дозволяють значно полегшити роботу з кодом, автоматизувати аналіз даних і забезпечити більш точне та ефективне створення контенту. Важливими аспектами є використання цих інструментів для генерації та аналізу коду, а також їх інтеграція з вебконтентом для забезпечення високої продуктивності та якості [45]. Другий розділ присвячено технічним аспектам поєднання нейропомічників з популярними бібліотеками та платформами для веброзробки, такими як React.js та Node.js. Виявлено, що такі поєднання дозволяють значно зменшити час розробки та покращити функціональність вебдодатків. Третій розділ зосереджено на практичному застосуванні бібліотеки React та Node.js з використанням нейропомічників.

Практична частина показала, що поєднання цих інструментів дозволяє розробляти високоякісні та ефективні вебдодатки з мінімальними витратами часу.

Дослідження показало, що нейропомічники можуть значно полегшити процес розробки вебконтенту, зокрема у контексті використання сучасних технологій, таких як React.js та Node.js [46]. Вони сприяють оптимізації коду, підвищенню ефективності розробки та інтеграції з іншими технологіями, що робить їх важливими інструментами для сучасних веброзробників.

Таким чином, результатами роботи є методологія застосування нейропомічника для підтримки та автоматизації процесів розробки програмного коду; рекомендацій щодо застосування нейропомічника для розробки вебконтенту. Вони можуть бути використані для подальших досліджень за даною тематикою та при можуть бути використані для подальших досліджень за даною тематикою та при проектуванні сайту Інтернет.