Форма № Н-9.02

Міністерство освіти і науки України

Відокремлений структурний підрозділ «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»
(повне найменування вищого навчального закладу)

Відділення інформаційних технологій, менеджменту, туризму та підготовки іноземних громадян
(назва відділення)

Циклова комісія комп'ютерної інженерії
(повна назва циклової комісії)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

фахового молодшого бакалавра
(освітньо-професійного ступеня)

на тему: **Розробка веб-сайту інтернет магазину окремого відділу «Нова школа» видавництва «Підручники і посібники»**

Виконав: студент IV курсу, групи КІ-412

Спеціальності 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

_____ Ірина КУДРИК
(ім'я та прізвище)

Керівник Роксолана БОЛЮБАШ
(ім'я та прізвище)

Рецензент _____
(ім'я та прізвище)

Тернопіль – 2025

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
імені ІВАНА ПУЛЮЯ»

Відділення **інформаційних технологій, менеджменту, туризму та підготовки іноземних громадян**
Циклова комісія **комп'ютерної інженерії**
Освітньо-професійний ступінь **фаховий молодший бакалавр**
Освітньо-професійна програма: **Обслуговування комп'ютерних та систем і мереж**
Спеціальність: **123 Комп'ютерна інженерія**
Галузь знань: **12 Інформаційні технології**

ЗАТВЕРДЖУЮ
Голова циклової комісії
комп'ютерної інженерії
_____ Андрій ЮЗЬКІВ
"31" березня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кудрик Ірині Валеріївні
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: **Розробка веб-сайту інтернет магазину окремого відділу «Нова школа» видавництва «Підручники і посібники»**

керівник роботи Болюбаш Роксолана Юріївна
(прізвище, ім'я, по батькові)

Затверджені наказом Відокремленого структурного підрозділу «Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя» від 28.03.2025р № 4/9-166а.

2. Строк подання студентом роботи: 13 червня 2025 року.

3. Вихідні дані до роботи: Visual Studio Code, Mozilla Firefox, Microsoft Edge, React, JavaScript, HTML, CSS, Node.js, Express.js, MariaDB, DBEaver.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): Загальний розділ. Розробка технічного та робочого проєкту. Спеціальний розділ. Економічний розділ. Охорона праці та безпека життєдіяльності.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
   - структурна схема сайту;
   - блок-схема до скрипта;
   - скрипт;
   - таблиця техніко-економічних показників.

6. Консультанти розділів роботи

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ	Богдана МАРТИНЮК викладач		
Охорона праці та безпека життєдіяльності	Володимир ШТОКАЛО викладач		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Отримання і аналіз технічного завдання	01.04	
2	Збір і узагальнення інформації	05.05	
3	Написання першого розділу	16.05	
4	Розробка технічного та робочого проекту	23.05	
5	Написання спеціального розділу	30.05	
6	Розрахунок економічної частини	02.06	
7	Написання розділу охорони праці	04.06	
8	Виконання графічної частини	09.06	
9	Оформлення проєкту	11.06	
10	Погодження нормоконтролю	12.06	
11	Попередній захист роботи	13.06	
12	Захист кваліфікаційної роботи		

7. Дата видачі завдання: 01 квітня 2025 року

Студент _____ Ірина КУДРИК
(підпис) (ім'я та прізвище)

Керівник роботи _____ Роксолана БОЛЮБАШ
(підпис) (ім'я та прізвище)

АНОТАЦІЯ

Основною метою даної кваліфікаційної роботи є розробка вебсайту для інтернет-магазину окремого відділу «Нова школа» видавництва «Підручники і посібники».

Дана кваліфікаційна робота складається з п’яти розділів: у першому — загальному, — розділі проведено аналітичний огляд існуючих рішень, вказано доцільність роботи та розроблено технічне завдання.

Другий розділ є основним, містить матеріал про обрані середовища розробки для створеного вебдодатку у вигляді односторінкового вебсайту, описано структуру та вміст сайту.

Третій розділ — спеціальний, — містить інструкції зі встановлення необхідних середовищ для розробки вебсайту, такі як платформа для використання вебсерверу, бази даних та клієнтської частини вебдодатку. Також реалізовано використання вебсайту завдяки адміністративній панелі.

Четвертий розділ є економічним та містить розрахунки собівартості, ціни та термін окупності розробленої кваліфікаційної роботи.

У п’ятому розділі розглянуті питання з охорони праці та безпеки життєдіяльності.

Графічна частина кваліфікаційної роботи складає чотири плакати формату А3.

Кваліфікаційна робота носить практично-орієнтований характер і містить простий за функціоналом сайт, який розміщений на даний момент на локальному сервері з метою подальшого удосконалення перед повноцінним релізом.

ANNOTATION

The main goal for qualification work is develop a website for online-store for separate department «Nova Shkola» publishing «Pidruchnyky i Posibnyky».

This qualification work have a main five chapters: In first chapter, called general, section realized analytic review and technical documentation.

Second chapter have main content about qualification work and have material about selected web development for created web application like one-paged website, descripted site’s structure and content.

Third chapter called special and have instruction for download all list of used development like platform for using webserver, database and web application’s client part. Also, was realized using website by administration dashboard.

Forth chapter is about economics and contains cost calculations, prices and payback period of the developed qualification work.

In the fifth chapter considers issues of occupational health and safety.

The graphic part of the qualification work consists of four A3 posters.

The qualification work is practically oriented and contains a simple functional site, which is currently hosted on a local server for the purpose of further improvement before a full release.

ЗМІСТ

ВСТУП

Інтернет-магазини стали невід'ємною частиною сучасного ринку, надаючи зручний доступ до товарів і послуг для широкої аудиторії. Особливо важливими є платформи, що спеціалізуються на освітніх продуктах, адже попит на якісні навчальні матеріали стабільно зростає серед батьків, учнів і педагогів. Видавництво «Підручники і посібники» вже має власний веб-сайт, який успішно представляє його діяльність, однак окремий відділ «Нова школа» потребує власної онлайн-платформи для ефективного просування та продажу спеціалізованого асортименту.

Метою даної кваліфікаційної роботи є розробка веб-сайту інтернет-магазину для окремого відділу «Нова школа» видавництва «Підручники і посібники». Сайт призначений для реалізації освітніх товарів, орієнтованих на різні вікові групи: наліпки, розмальовки та посібники для підготовки до школи для дошкільнят; читальники, тренувальники та матеріали для письма для учнів початкової школи; посібники для підготовки до ЗНО/НМТ.

Створення окремого сайту для відділу «Нова школа» дозволяє зосередитися на специфічних потребах цільової аудиторії — батьків, учнів і вчителів, які шукають зручний спосіб придбання навчальних матеріалів. Окрема платформа забезпечує чітке позиціонування асортименту, спрощує навігацію для користувачів і дозволяє реалізувати цільові маркетингові стратегії, не перевантажуючи основний сайт видавництва. Такий підхід підвищує ефективність продажів і сприяє кращій взаємодії з клієнтами, які шукають спеціалізовані освітні продукти.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1. Аналітичний огляд існуючих рішень

Правильно обране середовище розробки відіграє важливу роль у створенні проєкту. І неправильно вважати, що йде мова винятково про редактори коду чи IDE. Насправді, середовища розробки можна поділити на ось такі чотири групи[17]:

1. Операційні системи — найбільш вагомий вибір, адже саме від неї залежать наступні інструменти, які використовуватимуться для оптимізації роботи. Найбільш популярною трійкою, звісно, є Windows, MacOS та Linux. Windows добре підходить для загальної розробки, але часто потребує додаткових інструментів та їхнього налаштування. MacOS часто обирають саме для розробки дизайну та вебпрограмування, а також вже має необхідні інструменти для того аби написати код. Linux же ж є найбільш вдалим варіантом, адже має високі можливості налаштування та чудово підходить для розробки сервера чи для розробки бекенду[17].

2. Редактори коду або IDE є базовими інструментами для написання коду. Перші є інструментом для редагування тексту. Він легкий, простий, має мінімальну кількість функцій, такі як підсвічування синтаксису та найпростіший дебаґ. Проте це чудовий вибір для початківців. До редакторів належать Sublime Text, Visual Studio Code тощо.

IDE підходять для більш досвідчених користувачів з більшою специфікацією проєктів. Вони наділені великою кількістю додаткових вбудованих функцій, до прикладу, автоматичне форматування коду, контроль версій та інші. Сюди належать такі програми, як WebStorm, JetBrains Fleet тощо[3].

3. Браузери також належать до основних інструментів під час веброзробки. Одним з найбільш популярних, звісно, є Google Chrome, який є чудовим інструментом для проведення тестів. Другим найбільш популярним для

веброзробки браузером є Mozilla Firefox, який має водночас окрему версію для розробників під назвою Firefox Developer Edition. Саме ця версія наділена різними вбудованими інструментами, які можуть допомогти із розробкою вебпроєкту і її часто рекомендують для новачків[17].

Звісно, є й інші браузери, такі як Microsoft Edge, який уже за функціоналом є схожий до Chrome, та має вже вбудованих механізм JavaScript для розробки. Він уже містить багато функцій, до прикладу, відображення помилок, попередження та виправлення проблем у коді, покрокова перевірка коду тощо.

4. Інші інструменти. В основному, це все, що можна назвати інструментом розробки: від розширень браузера та редактора коду до окремих інструментів[17]. Так, тут можна віднайти Figma, який використовують у дизайні та прототипуванні, або ж Github, який використовують як окреме сховище. Проте, все ж, є деякі з них обов’язковими під час розробки.

Одним з найбільш важливих інструментів для розробки вебсайтів є база даних задля збереження величезної кількості інформації, такі як: дані про клієнтів, товари, медіа, замовлення тощо[32]. Зазвичай, зберігається вона у вигляді таблиці, документів та інших форматів. Так, саме завдяки базам даних ми спроможні вести соціальні мережі, користуватись послугами різних стрімінгових сервісів, читати блоги та переглядати статті.

Вибір бази даних для сайту залежить від його типу та потреб. Найпопулярнішими є реляційні бази даних — одні з найстаріших типів та, водночас, достатньо простих. Інформація у таких базах зберігається у вигляді таблиці, де за стовпці беруться загальні властивості об’єктів, а у рядках — інформація про той чи інший об’єкт[32].

Запити у реляційних базах даних формуються за допомогою мови програмування SQL[4]. Дана мова дозволяє маніпулювати всією інформацією для функціонування сайту, а саме додавати її, редагувати, переглядати, отримувати вибіркові дані та видаляти за допомогою простих запитів. Це відбувається за допомогою окремих посилань на ті чи інші таблиці. SQL-запити розбиваються на токени — зрозумілі фрагменти, такі як ключові слова або ідентифікатори, — а

сама база даних перевіряє їхній синтаксис, правильну послідовність — зв’язок із вже наявними таблицями та/або стовпцями, — та записує на диск або у пам’ять кожен раз, коли виконується новий запит[4].

Єдиним недоліком реляційної бази даних є її власна перевага: сувора структура відомостей про об’єкти. Так, даний тип є логічним та зрозумілим, проте вимагають фіксованої структури, яка може бути незручною при використані динамічних даних[35]. Проте, не зважаючи на це, реляційні бази даних все ще є фаворитами серед веброзробників, адже вона чудова для зберігання інформації про користувачів, замовлення, вміст сайту тощо.

Аби обійти головний недолік реляційних БД були розроблені нереляційні. Так, ці бази даних використовують не тільки традиційне збереження інформації у вигляді таблиць, але й інші види — до прикладу, у вигляді документів. Вони вже дозволяють зберігати динамічні дані, які можуть змінюватись у будь-який момент часу. Чудово підходять для зберігання великої кількості неструктурованої інформації будь-яких типів та додавання нових у процесі роботи.

Не зважаючи на різноманіття, NoSQL поділяють за чотирма основними групами. Їх розрізняють за моделлю даних та підходом до структури. Найбільш простими є база даних типу «ключ-значення». Тут інформація зберігається у вигляді словника, де ключ — є покажчиком або ж словом, а увесь вміст відповідає за його тлумачення. Такі БД чудово працюють із абсолютно різною інформацією, нехай це буде набір чисел, слів, різних файлів[36].

Документоорієнтовані бази даних зберігають інформацію у структурованих форматах, де інформація все ще зберігається завдяки «ключеві»; графові — у вигляді своєрідної мережі, де об’єкти виступають під виглядом вузлів, або ж вершин, які з’єднані між собою немов павутинка[32]; а ствопчикові передають інформацію за допомогою різних колонок, а не рядків, як у реляційних БД, що забезпечує комфортну роботу з великим обсягом даних[32].

Проте база даних — це лише набір інформації, нехай вона буде структурованою чи ні. Проте аби мати можливість з нею працювати — необхідно

використовувати відповідне програмне забезпечення.

Система управління базою даних, або ж СУБД — це спеціалізоване програмне забезпечення, яке полегшує роботу між користувачем та базою даних. Вони дозволяють створювати та зберігати абсолютно всю інформацію, запобігають її втраті. СУБД надають можливість одночасного перегляду даних різними користувачами, а також — налаштовувати різний доступ для учасників, що гарантує безпеку від лишнього втручання та втрати даних[11].

Для кожного типу баз даних існують і різни види СУБД, які працюють у консольному режимі. Так, для релятивних використовують РСУБД, який є інструментом, що дозволяє працювати з реляційними базами даних, виконуючи необхідні запити, керування та безпеку. Якщо реляційні бази призначені як структуроване сховище інформації про різні об’єкти — набір таблиць, — то РСУБД відповідає за створення та керування цими БД, обробкою запитів від вебсайту, шифрування даних та авторизацію користувачів з різним доступом. Такими є Oracle, MySQL, MariaDB, PostgreSQL та OpenJDK.

СУБД для нереляційних баз даних більш багатогранні та різносторонні завдяки різноманітності самих БД, проте єдиною спільною рисою у них є здатність працювати як з реляційними, так і з не- чи слабоструктурованими даними[11]. Дані СУБД мають перевагу над РСУБД завдяки тому, що розраховані не тільки на SQL-запити, а й на інші способи обробки інформації, адже NoSQL бази даних не мають універсальної мови, оскільки кожен тип БД має власний спосіб записувати та обробляти інформацію. Такими СУБД є MongoDB (документноорієнтовані БД), Amazon DynamoDB та Redis («ключ-значення») HBase (стовпчикові) та інші [34].

Окрім консольного режиму, для СУБД розроблені також і спеціалізоване програмне забезпечення зі графічним інтерфейсом. Це пов’язано з тим, що СУБД зосереджується на серверній частині — зберігання даних, обробка запитів тощо. Для цього достатньо консольного режиму, який часто використовують для автоматизації. Проте для початківців та звичайних користувачів розроблені різні графічні інтерфейси, аби було простіше працювати з даними — це додаткове

програмне забезпечення, яке є окремим інструментом для управління базою даних через зручний графічний інтерфейс.

Найбільш поширеним та якісним у роботі вважається середовище DbVisualizer — кросплатформова програма зі графічним інтерфейсом, яка підтримує велику низку баз даних різних типів, такі як Oracle, MySQL, MongoDB, Azeru тощо, пропонуючи розширений редактор SQL з різними функціями, до прикладу як автоматичне форматування та підказки. Окрім хорошої візуалізації даних, DbVisualizer наділений надійними заходами безпеки. Проте, на жаль, DbVisualizer є платним середовищем.

За аналог можна узяти DBeaver — кросплатформовий інструмент для роботи з базами даних. Дане програмне забезпечення зі графічним інтерфейсом підтримує реляційні, NoSQL і хмарні бази даних. Це безкоштовна програма, яка надає доступ до великої низки БД[3]. Окрім базового графічного інтерфейсу тут також є SQL-редактор, можливість проведення міграції бази даних, моніторинг з’єднання тощо.

Проте, найважливішими інструментами для веброзробки є технології, які охоплюють мови програмування, інструменти та фреймворки. Вони відповідають за відтворення контенту, взаємодію з ним тощо — тобто допомагають створити продуктивний та багатофункціональний вебдодаток[2]. Для веброзробників програмування — це основа їхнього проєкту, завдяки йому вони спроможні реалізувати свій творчий потенціал[16]. То ж цей інструмент поділяють на дві категорії: фронтенд та бекенд технології.

Фронтенд — це частина вебдодатку, з яким взаємодіє напряму користувач. Завдяки цьому інструменту вебсторінки можна структурувати, відтворювати будь-яку інформацію, стилізувати вміст, зробити вебсайт інтерактивним та візуально привабливим[16].

Це можна реалізувати за допомогою таких простих інструментів, як HTML та CSS. Перший є мовою програмування, яка забезпечує розробку базової структури та вмісту у вебдодатку — текст, зображення, покликання тощо[10]. CSS, у свою чергу, відповідає за стилізацію усього вмісту — саме завдяки цьому

елементу сучасні вдалі вебпроєкти мають чудовий вигляд[2]. Це є базовою частиною усіх сайтів і без реалізації бодай одного з цих інструментів сторінка не викликатиметься, або ж виглядатиме некоректно під час її перегляду.

Мови програмування, як додаткові інструменти у веброзробці, відіграють роль комунікації з комп'ютером та віддання команд що та як робити[2]. Вибір коректної мови програмування для вебсайту залежить від різних факторів, такі як характер та причина розробки проєкту, рівень знань розробника та конкретні вимоги. При розробці статичного вебсайту, достатньо використовувати два базових інструменти — HTML і CSS, у той час як для динамічного варта зверну увагу на ще одну. Це може бути як JavaScript, що орієнтований на вебпрограмування, так і Python — більш універсальний у великій кількості сфер[13]. Давайте розглянемо їх більш детально.

JavaScript — це міжплатформова, об'єктно-орієнтована мова програмування, що використовується у розробці динамічних вебсторінок. Вона дозволяє реалізувати такі елементи, як інтерактивні кнопки, анімації форми з валідацією тощо[17]. Задля позначення програми, у цій мові використовують такий термін як «скрипт», які, записавши, виконуються як простий текст[38].

Задля створення динамічної сторінки, для JavaScript було вигадано фреймворки (або ж бібілотеки) — перевірені інструменти, завдяки яким можна створювати інтерактивні та масштабні вебдодатки. Цей інструмент став настільки вподобаним у використанні різними компаніями, що знання та розуміння фреймворків від JavaScript є одним з основних етапів у розвитку веброзробника[15]. Найбільш популярними бібліотеками є AngularJS(Anguar2+) та ReactJs.

Варта також зазначити, що односторінковим сайтом вважається сайт, що складається з єдиної HTML-сторінки (переважно, index.html) та додаткових ресурсів для його роботи. Взаємодія з вебпрограмою відбувається без перезавантаження сторінки, що забезпечує швидку та якісну роботу.

Angular2+ — це фреймворк, випущений компанією Ґуґл, з відкритим кодом та легким синтаксисом[8]. Використовується дана бібліотека у розробці

однсторінкових вебпрограм. Angular наділений різними бібліотеками, які охоплюють великий список додаткових функцій, до прикладу, керування формами, зв’язок клієнт-сервер, маршрутизацію, тестування коду тощо.

React — це одна з бібліотек з відкритим вихідним кодом, яка також призначена для розробки односторінкових застосунків[41]. У першу чергу, React був розробленим для кращого керування рекламою у Facebook та Instagram[22]. На даний момент вважається одним з найкращих фреймворків для створення вебсайтів завдяки великій кількість функцій, що забезпечують якісний користувацький досвід[22]. React використовує віртуальну модель, яка називається DOM — Document Object Model. Ця модель використовується таким чином, щоб змінювати лише деякі елементи проєкту. Завдяки цьому продуктивність вебдодатку зростає, а вебсайт й надалі залишається комфортним у використанні[23].

Бібліотеку React використовують додатково з іншими. Найбільш вживаними фреймворками є: React Router, Redux, Axios та Next.JS.

Фреймворк React Router використовується в односторінкових вебдодатках для налаштування правильної навігації та маршрутизації[19].

Бібліотека Redux — для правильної та чіткої побудови статичного дерева станів. Це допомагає легше віднаходити помилки та їх вирішити[20].

Фреймворк Axios відповідає за здійснення HTTP-запитів з браузера такі як, наприклад, доступ до даних та їхня відправка на сервер[20].

Бібліотека Next.JS допомагає зі створенням серверної частини. Завдяки цьому фреймворку можна здійснювати серверний рендеринг, маршрутизацію та автоматичне розділення коду, що спрощує створення готових до роботи React-додатків[20].

Окрім JavaScript є й інші мови, які використовуються при веброзробці. До прикладу, Python — одна з універсальних мов, яку можна використовувати для різних цілей: автоматизація завдань, аналіз даних, створення вебсайтів та тестування програмного забезпечення[1]. Ця мова програмування є простою та читабельною, що й заохочує початківців до її вивчення[13]; звісно,

універсальність мови також стає вагомою причиною вивчення, адже її можна застосовувати, до прикладу, й у розробці звичайних десктопних додатків. Так само, як і JavaScript, Python володіє власними бібліотеками — Django та Flask, — що полегшують процес веброзробки[27].

Також, однією з мов програмування, яку можна застосувати у розробці вебпроєкту є Java — потужна мова програмування, яка використовується у великій кількості різних проєктів, заодно і у розробці вебпрограм. Завдяки Java можна реалізувати досить великі вебпроєкти-надгіганти[14]. Так, завдяки Java можна реалізувати платформи від банківських та звичайних фінансових сервісів до платформ електронної комерції, або ж як їх ще називають, інтернет-магазини. Завдяки своїй платформонезалежности (так, Java можна використовувати на будь-якій операційній системі) її й вважають універсальною мовою програмування. Як і будь-яка інша мова, у неї також є власний список фреймворків — JavaServerFaces та Apache Structs, — які спрощують розробку вебдодатків[14].

Як вже згадувалось раніше, веброзробка складається з двох основних частин: фронтенд та бекенд. У той період часу, як фронтенд фокусується на візуальній частині вебдодатку, бекенд відповідає за його(вебсайту) функціонування. Двома основними елементами бекенду є база даних а мова програмування[24].

Розробка бекенду відповідає за створення серверної частини вебсайту, яка обробляє все: вхідні дані, саму базу даних, відповідає за авторизацію та автентифікацію користувачів та забезпечує безперебійну роботу інтернет-продукту[27].

Найбільш вживаних мов програмування для розробки бекенду є п’ять, а саме: Python, Java, JavaScript(Node.js), PHP та Go[24].

Як ми вже згадували раніше, Python є однією з найбільш вживаних мов для програмування, проте найбільшу перевагу у веброзробці для написання бекенду —серверну логіку, яка забезпечує роботу вебдодатку, — віддають саме йому, а на фронтенд обирають JavaScript. Популярний він завдяки своїй простоті та

читабельності, а найчастіше з фреймворків використовують Django та Flask задля створення надійних, масштабних та безпечних вебпрограм[21].

Django є найбільш популярним фреймворком з відкритим кодом для Python, який сприяє швидкій розробці. Його цілком достатньо для створення бекенду, розробки адміністративної панелі, автентифікаційної системи, правильної зв'язки з базою даних тощо[25].

Більш простою бібліотекою для Python є Flask — вона більш мінімалістична, не наділена об'єкто-орієнтовною системою чи адміністративною панеллю, або ж автентифікацією, як її товариш вище. Проте, даний фреймворк надає повну свободу розробнику. Завдяки своїй гнучкости, Flask надає розробнику можливість обирати, яке розширення йому необхідне, що чудово підходить для невеликого проєкту[25].

Окрім цього, для розробки бекенду можна обрати і Java. Ця мова програмування пропонує велику кількість різних функцій, наділена надійною системою типізації та бібліотеками, що необхідно розробникам для створення проєктів-надгігантів. Також, до переваг Java відносять багатопотокову одночасну обробку даних, що підвищує продуктивність та швидкість реагування вебдодатку на взаємодію користувача з ним[13].

Як вже згадувалось раніше, JavaScript також використовується у розробці бекенду для вебдодатків. Якщо бути більш точним, то використовується один з його фреймворків, а саме — Express/Node.js.

З офіційного сайту можна взнати, що Node.js — це безкоштовне міжплатформне середовище з відкритим кодом, яке чудово підходить для вебдодатків реального часу. Дана бібліотека розроблена на мові програмування JavaScript, яка дозволяє створювати, налаштовувати та автоматизувати сервери та додатки, використовувати інструменти командного рядка[18]. Node використовується поза контекстом браузера, на відміну від самого JavaScript. Тобто, дана платформа працює безпосередньо на комп'ютерні або серверній ОС, тим самим не потребує специфічних API для браузера, як материнська мова програмування [7].

Node.js, у порівнянні зі традиційними вебсерверами, використовує один потік для обробки кількох клієнтських запитів завдяки своїй асинхронній моделі програмування [18]. Окрім цього, дана платформа надає зручні інструменти для простої взаємодії з клієнтською стороною на основі JavaScript. Використовуючи різні фреймоврки, такі як Express.js та/або Nest.js, розробники спроможні проєктувати потужні та масштабовані вебдодатки[40].

Express — є найбільш популярним фреймворком та основою для решти інших бібліотек для Node.js. Він надає велику низку різних можливостей, такі як обробка запитів з різними HTTP на різних маршрутах та загальні налаштування, такі як порт і розташування шаблонів. Express є достатньо мінімалістичною бібліотекою, до якої можна дозавантажити різні сумісні пакети для кращого досвіду у розробці вебпрограмування. Так, завдяки додатковим бібліотекам можна налаштувати роботу з файлами cookie, сеансами, авторизацією та реєстрацією користувачів тощо[7].

Express.js спрощує маршрутизацію, підтримує створення REST API, додатків реального часу та односторінкових проєктів. Сама бібліотека вимагає базові знання та розуміння JavaScript та Node.js[6].

Одним з найбазовіших інструментів бекенду є PHP (Hypertext Preprocessor) — близько 70% різних вебсайтів функціонують завдяки цій мові програмування. PHP набув значної популярності завдяки побутової думки про його простоту у використанні та незалежности від різних платформ[22].

Як й інші мови програмування, що ми розглядали вище, PHP також наділений фреймворками. Одним з таких є Laravel — серед всіх фреймворків на даній серверній мові програмування вирізняється своїми функціями, у які входять синтаксис, структура, вбудована утиліта автентифікації, різні механізми шаблонів тощо. Дана бібліотека досить поширена серед платформ інтернет-магазинів, CMS та різних мікросервісів[19].

Go, також відомий як Golang — іще одна мова програмування, розроблена корпорацією Ґуґл у далекому 2007 році, набула своєї популярності також завдяки власній простоті та ефективності у створенні серверних систем[26]. Даний

інструмент було розроблено з метою створити типізовану мову програмування, код якої легко буде написати та прочитати. Спочатку Go чудово підходила для проєктів-гігантів, проте, згодом, перетворилась на просту та продуктивну мову, яку використовують уже понад один мільйон розробників[12].

Від самого початку Go розроблялась з метою усунути недоліки уже існуючих мов програмування та задовільнити усі потреби сучасної розробки вебдодатків. Так, деякі хмарні технології, такі як Kubernetes, Docker, Terraform та Prometheus були написані саме завдяки цій мові програмування[12]. Саме завдяки власній простоті, швидкій компіляції, статичній типізації, стандартним фреймворкам — а також з окремо виділеним для тестування, — наднизькому споживанні ресурсів, підтримці паралельності закривається основна ціль даної мови програмування — полегшення веброзробки у великих масштабах[29].

Кожна змінна у Go повинна мати свій чіткий та однозначний тип. Якщо JavaScript є більш гнучкою і не вимагає строгого оголошення, то Golang «полює» за кожним неправильно прописаним типом ще у процесі компіляції проєкту[28].

Надалі для розробки вебсайту «Нова школа» я, як початківець у веброзробці, використовуватиму Visual Studio Code, а із браузерів — Mozilla Firefox та Microsoft Edge. За графічне середовище для роботи з базою даних використовуватиму DBeaver, а за базу даних було обрано MariaDB, яка є простою для зберігання інформації про товари, замовлення, клієнтів тощо. Усе розроблялось на основі HTML/CSS і бібліотек React від JavaScript для оформлення дизайну сайту та примітивного функціонування, а також Node.js з фреймворком Express.js для налаштування серверу та автоматизації сайту.

1.2. Технічне завдання

1.2.1. Найменування та область застосування

Основним завданням вебсайту «Нова школа» є створення зручної та ефективної платформи для продажу освітніх товарів окремого відділу Редакції

газет «Підручники і посібники». Окрім цього, до області застосування також належить і використання різних інструментів для реклами та просування, такі як акції, знижки, рекомендації та персональні пропозиції.

Даний вебсайт належить до виду електронної комерції, або ж ecommerce — різновид платформ, метою яких є торгівля товарами та/або послугами різних компаній чи окремих інтернет-користувачів[30]. Електронну продаж-торгівлю можна вести як на ПК, так і на смартфорнах чи інших розумних пристроях[5].

1.2.2. Призначення розробки

Метою кваліфікаційної роботи є створення зручного та безпечного інтернет-магазину «Нова школа» окремого відділу Редакції газет «Підручники і посібники» для продажу освітніх товарів. За середовище було обрано Visual Studio Code і бібліотеки React для JavaScript та Express для Node.js які є найбільш дружелюбними до початківців. За базу даних була обрана реляційна MariaDB за її простоту та структурованість, а для роботу із нею — DBeaver.

Сам сайт покликаний забезпечити легкий доступ до товарів, зручну навігацію, детальні описи товарів і можливість оформити замовлення.

1.2.3. Вимоги до програмного забезпечення

Основним завданням є створення сайту для окремого відділу «Нова школа» Редакції газет «Підручники і посібники», який забезпечує продаж освітніх товарів.

Вимогами до програмного забезпечення щодо розробки інтернет-магазину є:

Програмні засоби, використані для розробки інтернет-магазину «Нова школа», забезпечують повну сумісність із усіма модулями сайту, враховуючи функціонал комерційної платформи[39]. До функцій входять:

- Реєстрація та авторизація користувачів.

- Пошук і фільтрація товарів.

- Оформлення замовлення, його редагування, спосіб оплати та доставки.

- Адміністративна панель.

Для розробки використовується програмне забезпечення з відкритим кодом, яке дозволяє комерційне використання без придбання ліцензії. Основним середовищем розробки є Visual Studio Code версії 1.100.2, що підтримує розширення для HTML, CSS, JavaScript та бібліотеки React версії 19.0.0, забезпечуючи ефективну роботу з кодом комерційного вебсайту.

Сайт гарантує коректну роботу в браузерах Mozilla Firefox версії 138.0.4 та Microsoft Edge версії 136.0.3240.76. Адаптивність до інших розмірів екрану інтернет-магазину реалізовано за допомогою CSS-медіазапитів.

1.2.4. Техніко-економічні показники

Розробка сайту для інтернет-магазину «Нова школа» відбувалась на ноутбуці ASUS ROG Strix G15 G512LU-HN093 з наступними характеристиками:

- Дисплей 15.6", IPS, 144 Гц.

- Центральний процесор Intel Core i7-10750H.

- 16 ГБ ОЗП.

- Відеокарта NVIDIA GeForce GTX 1660Ti.

- SSD-накопичувач на 512 ГБ (M.2 NVMe).

- Бездротова карта Intel Wi-Fi 6 AX201 160 МГц.

- Операційна система Windows 10.

Локальний сервер для тестування організовано на цьому ж ноутбуці за допомогою Express.js — фреймворк для розробки бекенду та вебдодатків з використанням REST API від Node.js — середовище, що, у свою чергу, виконує функції серверного середовища для розробки та тестування вебсайту, написане на мові програмування JavaScript.

Із програмних засобів було використано:

- HTML 5.

- CSS 3.
- React 19.0.0.
- Node.js 20.19.0.
- Express.js 5.1.0.
- DBeaver 25.0.1.202503231807.
- MariaDB 11.7.2.
- Visual Studio Code 1.100.2.
- Mozilla Firefox 138.0.4.
- Microsoft Edge 136.0.3240.76.

1.2.5. Стадії та етапи розробки

Етапами розробки сайту інтернет-магазину «Нова школа» є:

1. Визначення завдання та мети.
2. Написання технічного завдання та створення карти сайту.
3. Розробка макету дизайну.
4. Верстка сторінок.
5. Наповнення сайту контентом.
6. Тестування та подальше налаштування.
7. Внесення змін.
8. Здача проєкту.

1.2.6. Порядок контролю та прийому сайту

Після завершення розробки вебсайту інтернет-магазину «Нова школа» необхідно провести тестування та аналіз проєкту, щоб переконатися у відповідності сайту заданим вимогам, його функціональності, продуктивності та якості.

До перевірки функціональності входить:

- Перевірка реєстрації та авторизації користувачів як за іменем, так і за

електронною поштою.

- Перевірка пошуку товару та їхньої фільтрації.

- Перевірка оформлення замовлення, а саме додавання товарів до кошику, його редагування, вибір доставки та оплати.

- Перевірка коректності адміністративної панелі, а саме редагування вмісту інтернет-магазину та керування замовленнями.

Також до тестування входить перевірка якісної адаптивності сайту за допомогою медіазапитів на пристроях з різними розширеннями екрану, сумісність з браузерами.

Вимірюється час завантаження сторінок та стабільності роботи при одночасному доступі кількох користувачів. Після розміщення сайту на хостингу проводиться перевірка коректності роботи інтернет-магазину, включаючи коректність обробки замовлень та відображення контенту.

Після проведення усіх тестувальних робіт готується документ, у якому підбивають підсумки завершення розробки, відповідність сайту вимогам і його готовність до експлуатації. Також передається доступ до адміністративної панелі та вихідного коду.

2. РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

2.1 Постановка задачі на розробку програмного забезпечення

Інтернет-магазин окремого відділу «Нова школа» видавництва «Підручники і посібники» створений задля продажу освітньої та дитячої літератури, такої як навчальні посібники, збірники для підготовки до іспитів для випускників старших класів та підготовки до школи для дошкільнят, розмальовки для дошкільнят тощо.

Основною метою вебсайту є забезпечення розробки зручної платформи для користувачів з можливістю переглянути, обрати та замовити товар. Розробку подібного вебдодатку можна вважати заледве не першою сходинкою задля того, аби вийти на ширший ринок, з подальшим покращенням ефективності умов ведення торгівлі та більш простим способом розповсюдження своїх товарів[5].

2.1.1 Огляд середовища розробки вебсервера Node.js

Одна із вагомих причин, чому для локального запуску вебдодатку необхідний бодай примітивний вебсервер є запити, які повинні виконувати фронтенд-частина сайту. Запити формують як код у React, так і сам браузер, наприклад, виведення товарів з бази даних у каталозі. Завдяки серверу можна перевірити, чи все правильно функціонує, навіть якщо він — вебдодаток, — використовує статичні HTML-файли[48].

Розробка власного вебсерверу на відміну від використання локального надає повний контроль над проєктом. Саме розробка власного вебсерверу дозволяє повністю проводити оптимізацію сайту під конкретні потреби розробника.

У цьому питанні для розробки вебсайту для інтернет-магазину «Нова школа», чий фронтенд написаний завдяки React, найкраще підходе платформа Node.js, яка зробила справжній прорив для веброзробників, адже JavaScript, на

основі якої розроблений Node, був винятково клієнтською —фронтенд — частиною. Це дозволяє використовувати одну мову для розробки проєкту, що є чудовим варіантом для веброзробників, які тільки розпочинають свій шлях у сфері вебпрограмування[9].

Завдяки цьому, Node.js знімає потребу у зміні контексту, який вимагається при розробці проєкту на декількох мовах програмування для різних частин сайту, аби не виникало конфліктів між синтаксисом та бібліотеками. Оскільки у даній розробці завдяки серверній платформі Node використовується винятково одна мова програмування — це спрощує в рази роботу.

Також це спрощує і сам код, дозволяючи використовувати функції як на клієнтській, так і на серверній стороні без додаткового їхнього оголошення. Завдяки одній мові програмування не виникає потреби писати схожий код, що сильно економить час. Так, з Node.js використовується JavaScript як для форми у React, так і для API, що зменшує плутанину.

Завдяки асинхронній моделі обробки запитів, Node.js обробляє багато операцій одночасно, не чекаючи завершення попередніх завдань, що збільшує продуктивність вебдодатку. Це є однією з основних його переваг над іншими, більш традиційними серверними платформами, де запити обробляються послідовно, що може вповільнити роботу вебсайту[9].

Для кращої структури використовується функція async/await. Дана функція допомагає працювати з асинхронним кодом та дещо його структурує задля розуміння написаного та роботи серверу. Це допомагає уникнути складних ланцюжків або зворотних викликів, які часто виникають при роботі з API, читанням файлів чи взаємодією з базою даних[49**].

Для кращої роботи з серверною платформою Node є один з його легких фреймворків — Express.js. Дана бібліотека спрощує створення серверних додатків та API, яка побудована на основі модуля http. Express.js використовується для кращої оптимізації роботи таких речей як маршрутизація для обробки різних HTTP-методів; обробки запитів, перевірка автентифікації та авторизації користувачів. На сьогоднішній день, Express.js є невід’ємною

частиною екосистеми Node.js і є найбільш популярним вибором під час розробки серверної частини[6].

Загалом, Express.js можна сприймати як перелік додаткових плагінів, що полегшують розробку вебдодатку і дозволяють простіше вирішувати поставлені завдання. Зокрема, даний фреймворк можна використовувати разом з бібліотеками React.js, а також з базами даних, як MongoDB або MariaDB[37].

2.1.2 Огляд програмних засобів для розробки вебсайту

Задля розробки інтернет-магазину «Нова школа» використовувались набір програмних засобів, які забезпечують якісне створення та підтримку клієнтської та серверної частин проєкту. Використання сучасного технологічного стеку та інструментів дозволило ефективно реалізувати функціонал каталогу книг, системи авторизації, кошик та оформлення замовлень.

До основних програмних засобів належить середовище розробки Visual Studio Code, призначення якого є написання, редагування та налагодження коду для вебдодатку. Дана програма підтримує різні мови програмування, у тому числі й HTML, CSS та JavaScript, які також використовуються у розробці проєкту для кваліфікаційної роботи. У VS Code вбудований термінал, за допомогою якого зручно запускати серверну та клієнтську частини, використовуючи винятково дві команди: npm start для фронтенду та node server.js задля бекенду.

На рисунку 2.1 зображене середовище розробки Visual Studio Code.

Як вище згадувалось, у VS Code використовується використовується бібліотека від JavaScript, а саме — React.js, яка розроблена з метою створення компонентних інтерфейсів користувача, які ще також прозивають UI. Компоненти у даній бібліотеці відповідають функціям у материнській мові програмування. Вони приймають пропси — довільні вхідні дані, — і повертають їх як React-елементи, виводячи необхідну інформацію на екран. Елементами називають найпримітивніші об’єкти у вебпрограмуванні, які описують те, що повинно виводитись на екран[22].

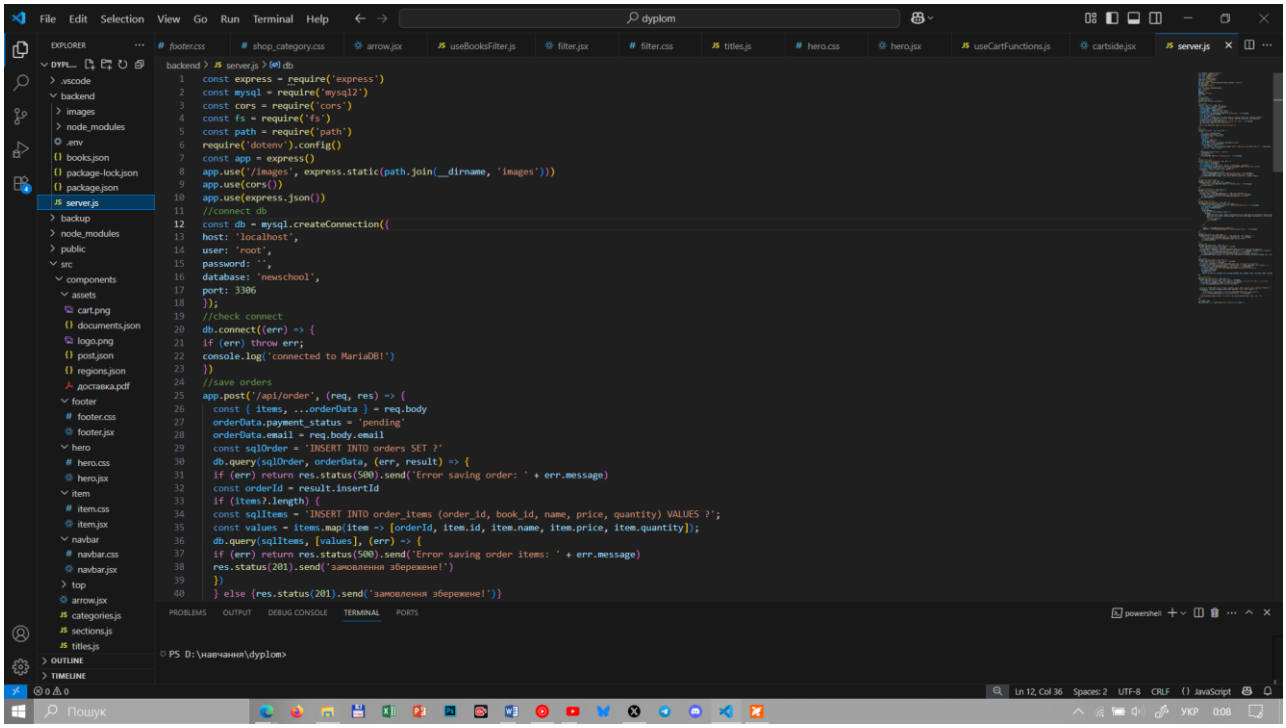


Рисунок 2.1 — середовище розробки VS Code

Окрім цього, використовується одна з бібліотек React, а саме React Router, яка забезпечує навігацію між сторінками без перезавантаження браузера. Це дозволяє створювати плавний UX — користувацький досвід, — де перехід між сторінками відбувається миттєво.

У React також можна створювати та використовувати стан без написання класу. Для цього були розроблені хуки, які дозволяють перевикористовувати логіку між компонентами. Тобто, аби не прописувати повторно в іншому елементі коду ту чи іншу функцію, створюється окремий елемент, який просто викликається у тому місці, де це необхідно. Це дозволяє зробити код менш навантаженим та більш зрозумілим під час його прочитання.. React має як вбудовані хуки, такі як useState, useEffect, useContext, так розробник може створити свій власний задля подальшого використання[22].

Також, бібліотека React чудово інтегрується з Express.js, до прикладу, відправляючи HTTP-запити до серверного API, завдяки тому, що написані вони на одній мові програмування, а саме JavaScript.

Наступним важливим елементом є база даних. У розробці проєкту для

кваліфікаційної роботи використовується така БД, як MariaDB — реляційна база даних з відкритим кодом, яка призначена для зберігання даних про книги, користувачів та замовлення, а вся інформація зберігається у вигляді таблиць. Завдяки своїй суворій моделі, інформація зберігається структуровано, що дозволяє зберігати у собі великий обсяг даних.

Задля кращої роботи з базою даних, використовувалось таке програмне забезпечення, як DBeaver — безкоштовний інструмент для керування різними базами даних, такі як MariaDB, MongoDB, MySQL, Oracle тощо. Для вебсайту «Нова школа» DBeaver використовувався задля створення та налаштування СУБД MariaDB. Сам інтерфейс даної програми є зручним у використанні, а під час запуску програми можна переглянути підказки щоб нагадати собі як правильно працювати з різними базами даних. DBeaver підтримує SQL-редактор, завдяки якому можна проводити складні запити, до прикладу, створювати та редагувати таблиці, видаляти їх тощо.

На рисунку 2.2 зображено інтерфейс програми DBeaver.

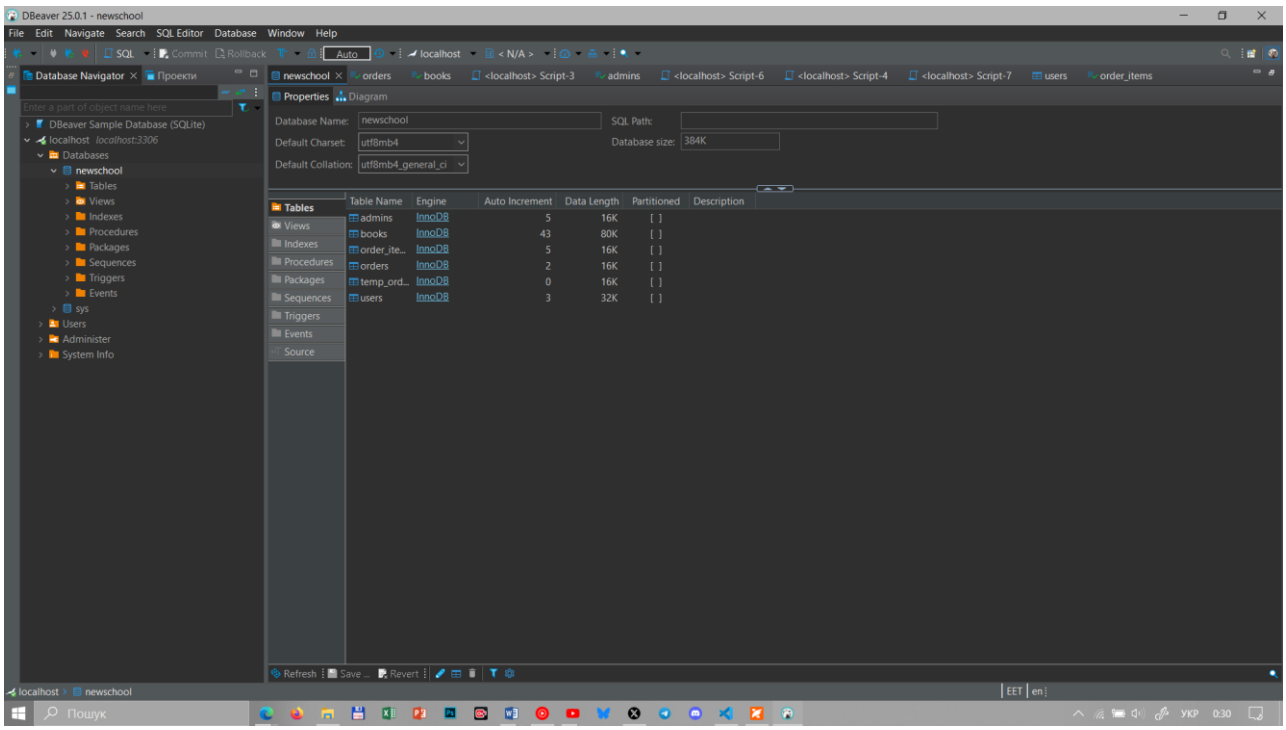


Рисунок 2.2 — графічний інтерфейс DBeaver

2.2 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних

Надалі розглянемо сам інтернет-магазин «Нова школа». Як згадувалось раніше, основною метою вебсайту є продаж освітньої літератури. Користувачі спроможні додавати товари до кошика, формувати замовлення та бачити суму до сплати. Реєстрація та авторизація користувачів також підтримують цю механіку, адже вона полегшує формування клієнтської бази для магазину. Даний аспект реалізовано за допомогою Express.js, що є фреймворком Node.js, який чудово підходе для комерційних операцій та інтеграції баз даних.

Сайт також наділений й інформаційно-довідковою системою, хоча вона не відіграє головну роль, проте є невід'ємною частиною проєкту. Вона відповідає за надання користувачам інформацію про книги, а саме: їхню назву, автора чи авторів, ціну, опис, цільову авдиторію. Сторінки кошика та сформованого замовлення у профілі також надають довідкову інформацію про вибрані товари, їхню кількість і загальну суму. Даний аспект реалізовано за допомогою фреймворку React, який належить до мови програмування JavaScript, який найкраще серед усіх забезпечує зручне відображення інформації.

Задля реалізації даного проєкту для кваліфікаційної роботи, а саме вебсайту інтернет-магазину «Нова школа» видавництва «Підручники і посібники» за основу була обрана розробка сучасного технологічного стеку, що включає у себе React для клієнтської частини та Express.js для серверної. Дані інструменти забезпечують повний контроль над розробкою проєкту та надають вільний простір для творчої реалізації в межах односторінкового вебдодатку. React є однією з найбільш популярних бібліотек JavaScript з відкритим кодом, яка забезпечує створення динамічного та інтерактивного інтерфейсу. Express.js, у свою чергу, є простим фреймворком для Node.js, який спрощує створення REST API для обробки HTTP-запитів. Основними перевагами даного методу є висока продуктивність, повний контроль під час розробки, а також гнучкість для подальшого масштабування.

У процесі виконання кваліфікаційної роботи було розроблено функціонал для інтернет-магазину «Нова школа». Реалізовано такі особливості, як динамічний інтерфейс, створений за допомогою React, який включає каталог продукції, фільтр для каталогу, форму реєстрації та авторизації, кошик та оформлення замовлення; REST API, яке реалізоване завдяки Express.js, що забезпечує обробку запитів для реєстрації користувачів, отримання переліку товарів, їхнє додавання до кошика та формування замовлення; зручна навігація між сторінками завдяки React Router, що забезпечує швидкий доступ до основних розділів сайту.

Під час перегляду головної сторінки сайту, яка зображена на рисунку 2.3, можна ознайомитись з головними категоріями — цільовою авдиторією того чи іншого продукту, — каталогу.

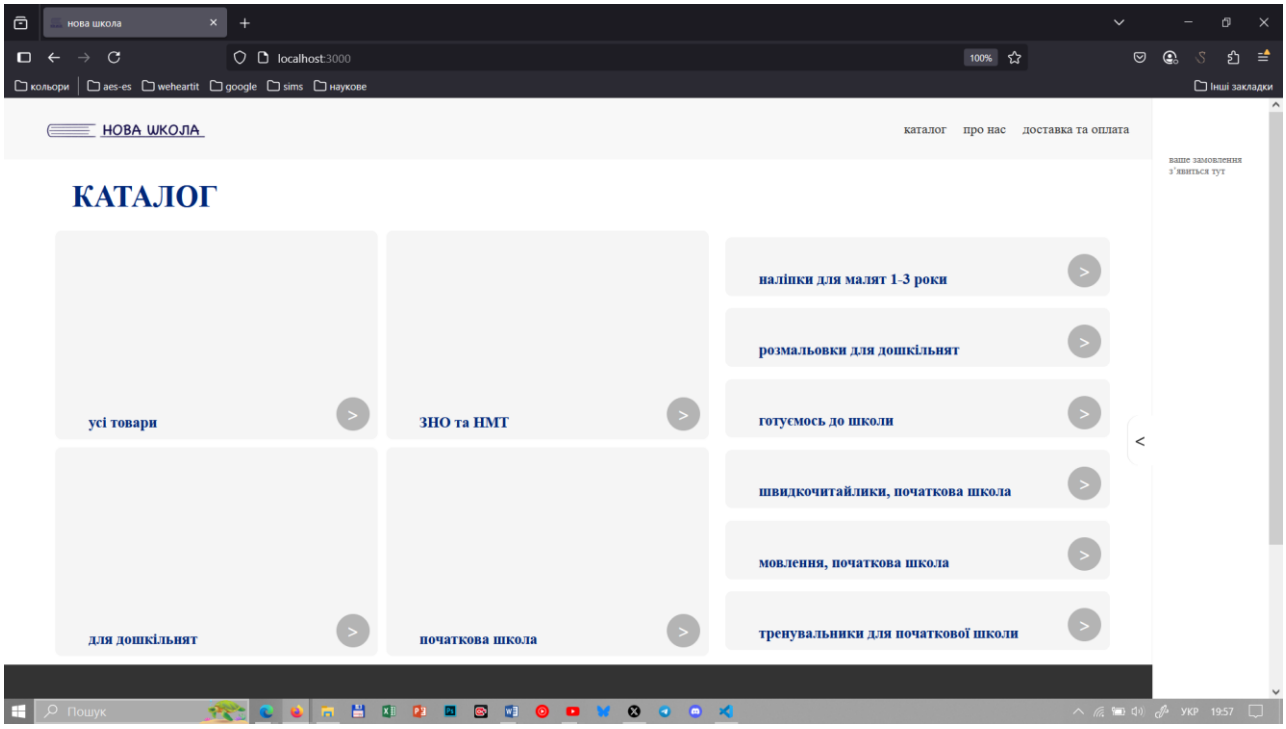


Рисунок 2.3 — головна сторінка сайту

Структура сайту оформлена за стандартним принципом, а саме: header-content-footer, яка забезпечує зручну навігацію відносно вмісту. Сама навігація підтримується бібліотекою React Router, яка забезпечує швидке перемикання між

контентом без перезавантаження.

Блок заголовка — header — реалізовано через компонент Navbar, який є частиною клієнтського інтерфейсу, створеного завдяки React. Сам блок виводить логотип магазину та головне меню з посиланнями на ключові розділи, такі як каталог, інформація про видавництво, до якого належить відділ, а також коротко про організацію доставки та оплати.

Нижній колонтитул організовано через компонент Footer, який зображений на рисунку 2.4, де знаходиться статична інформація, як контактні дані магазину, дублюються посилання на сторінки про доставку та про відділ.

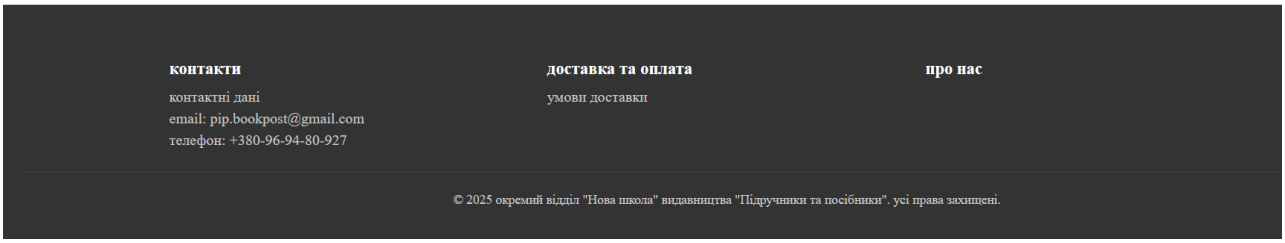


Рисунок 2.4 — футер сайту

Основний блок є динамічним і формується залежно від поточного маршруту. Кожен маршрут відповідає окремій сторінці, які реалізовані як React-компонент, яким є App.js — точка входу для всього додатку, що організовує структуру сторінок та охоплює весь сайт.

Даний компонент відповідає за маршрутизацію сторінок, підключає контексти для автентифікації та кошика, а також відображає основні компоненти інтерфейсу, такі як, навігація, футер, бічний кошик тощо. Задля маршрутизації використовуються такі елементи, як BrowserRouter та Routes, що належать до бібліотеки react-router-dom, завдяки яким можна визначити шляхи для різних сторінок. BrowserRouter є головним компонентом, який дозволяє використовувати маршрутизацію, працює з історією браузера, зміною URL без перезавантаження сторінки, що є головною особливістю односторінкового вебдодатку. Відповідає за імпорт інструментів для налаштування маршрутизації на сайті. Елемент Routes відповідає за групування маршрутів, які записуються у

коді даного компоненту через ключ Route.

На рисунку 2.5 зображено функцію коду App.js.

В компоненті App.js використовуються два основних контексти: AuthProvider та ShopContextProvider. Вони призначені для обміну даними між великою кількістю компонентів, що полегшує роботу коду.

```
function App() {
  return (
    <div>
      <AuthProvider>
        <BrowserRouter>
          <ShopContextProvider>
            <Cartside />
            <Navbar />
            <Routes>
              <Route path="/" element={<Shop />} />
              <Route path="/all_books" element={<ShopCategory />} />
              <Route path="/dlia_doshkilniat" element={<ShopCategory category="для дошкільнят" />}/>
              <Route path="/dlia_doshkilniat/nalipky" element={<ShopCategory subcategory="наліпки" />}/>
              <Route path="/dlia_doshkilniat/rozmaliovky" element={<ShopCategory subcategory="розмальовки" />}/>
              <Route path="/dlia_doshkilniat/pidhotovka_do_shkoly" element={<ShopCategory subcategory="підготовка до школи" />}/>
              <Route path="/pochatkova_shkola" element={<ShopCategory category="початкова школа" />}/>
              <Route path="/pochatkova_shkola/trenuvalnyky" element={<ShopCategory subcategory="тренувальник" />}/>
              <Route path="/pochatkova_shkola/chytannia" element={<ShopCategory subcategory="читання" />}/>
              <Route path="/pochatkova_shkola/movlennia" element={<ShopCategory subcategory="мовлення" />}/>
              <Route path="/zno_nmt" element={<ShopCategory category="зно/нмт" />} />
              <Route path="/login" element={<Login />} />
              <Route path="/cart" element={<Cart />} />
              <Route path="/checkout" element={<Checkout />} />
              <Route path="/profile" element={<Profile />} />
              <Route path="/order-success" element={<Ord_Succ />} />
              <Route path="/book/:id" element={<Books />} />
              <Route path='/pro_nas' element={<AboutUs/>}/>
              <Route path='/dostavka_ta_oplata' element={<Delivery/>}/>
              <Route path="/admin/dashboard" element={<AdminDashboard />} />
            </Routes>
            <Footer />
          </ShopContextProvider>
        </BrowserRouter>
      </AuthProvider>
    </div>
  )}
export default App
```

Рисунок 2.5 — функція компоненту App.js

Задля збереження даних користувача, на сайті є можливість реєстрації та входу у свій власний кабінет, де можна переглянути оформленні раніше замовлення та їхній статус. Дана функція виконується за допомогою контексту AuthProvider, який і забезпечує глобальний стан автентифікації — вхід та вихід з профіля.

AuthProvider створює глобальний контекст для автентифікації користувача у вебдодатку та дозволяє зберігати, отримувати та змінювати інформацію про поточного користувача. Використовуючи функцію export const AuthProvider =

({children}) створюється компонент-обгортка, який надає значення контексту всім дочірнім компонентам. Всередині цієї обгортки використовується булевий стан isAuthenticated, завдяки якому можна перевірити, чи увійшов користувач в особистий кабінет. Вхід перевіряється через функцію currentUser.

Функція const login = (userData, remember)=>{…} викликається під час входу користувача та розділена на декілька елементів, а саме: додає роль користувачу за допомогою перевірки const userWithRole = userData.role?userData:{...userData, role: userData.username?'admin':'user'}; зберігає дані про користувача за допомогою функції localStorage, коли здійснено вхід у профіль, та за допомогою sessionStorage коли користувач вирішив залишитись анонімним; при виході викликається функція const logout=()=>{…}, яка очищає стани стани та видаляє дані користувача.

Дані про користувача зберігаються у базі даних, якою у проєкті виступає реляційна БД — MariaDB. У даній таблиці можна звірити збережену інформацію про покупця, а також можна переглянути коли був зареєстрований його електронний кабінет завдяки колонці «created_at».

На рисунку 2.6 зображено вигляд збережених даних про у MariaDB через графічний інтерфейс DBeaver.

	id	email	password	lastName	phone	role	created_at	firstname
1	1	test@example.com	password123	Test	+380123456789	user	2025-06-15 13:17:41.000	testuser
2	2	kudrykiv03@gmail.com	123454321	кудрик	+380968289682	user	2025-06-15 13:21:22.000	ірина

Рисунок 2.6 — збережені дані про користувачів у MariaDB

Другим контекстом, який використовується у вебпрограмі є ShopContext, що відповідає за управління станом магазину: перелік книг, завантаження даних та операція з кошиком. Даний контекст дозволяє будь-якому компоненту вебдодатку отримувати та змінювати ці дані без необхідності передавати їх через пропси завдяки команді useContext(ShopContext).

У даному контексті використовуються такі стани, або ж state, як books, cartItems, loading. Задля завантаження книг зі серверу, використовується команда

const fetchBooks = async () => {…} — вона відправляє запит на сервер з метою отримати перелік продукції.

Окрім цього, існує збереження кошику між сесіями завдяки функції localStorage.setItem('cartItems', JSON.stringify(cartItems)). Задля збереження під час переходу між різними сторінками, в useEffect використовується такий стан, як window.history.state. через умовний оператор if перевіряється існування даного стану. Завдяки цьому, навіть при закритті вкладки, користувач, знову відкривши сайт, не втрачає перелік обраного раніше товару.

Також, даний контекст має окремо прописані операції для редагування кошику. Так, завдяки addToCart та setCartItems виконується збільшення та зменшення кількості товару у кошику. Операція removeFromCart видаляє книгу з переліку.

Наступним головним елементом у розробці проєкту для кваліфікаційної роботи є каталог, що відображає перелік книг із примітивною можливістю фільтрації за категоріями та підкатегоріями та наділений простою навігацією.

На рисунку 2.7 зображено виведення каталогу у браузері.

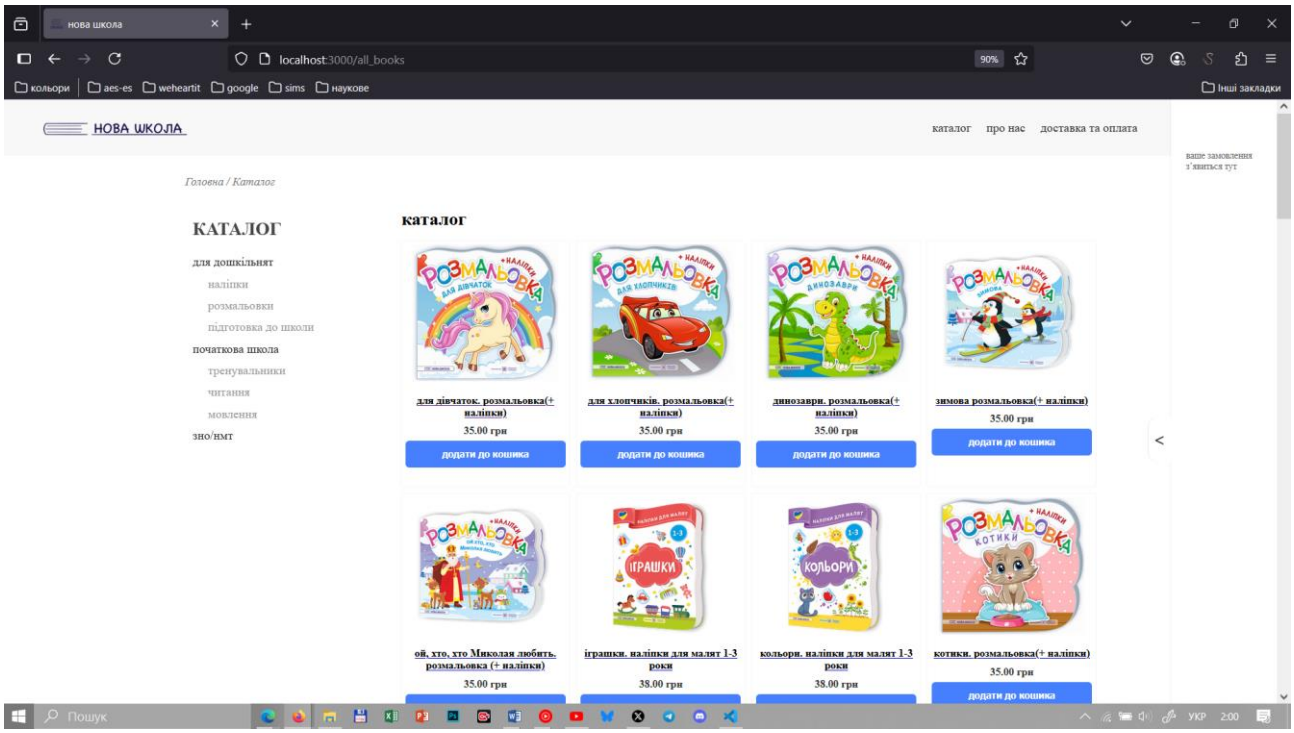


Рисунок 2.7 — вигляд каталогу

Основними елементами даного компоненту є ShopContext, який ми розглядали вище, а також окремі об’єкти для виведення кожної книги окремо, фільтр, перелік категорій та кастомний хук для логіки фільтрації. Давайте розберемо кожен з них.

Розпочнемо з найважливішого, а саме — з книг, адже без товару, інші елементи тут не потрібні.

У даному проєкті цей компонент називається Item, код якого зображений на рисунку 2.8. він розроблений для відображення кожної книги окремо у каталозі товару. Виводить зображення, назву, ціну книги та, завдяки ShopContext і розробленій для цього кнопці, дозволяє додавати її до кошика. Також, тут використовується Link задля переходу на сторінку конкретного товару для детального ознайомлення з ним.

На рисунку 2.9 зображено вигляд окремого товару.

src > components > item > item.jsx > ...

```jsx
import React, { useContext } from "react"; import { Link } from "react-rou
const Item = ({id, name, image, price, quantity}) => {
  const {addToCart} = useContext(ShopContext)
  return (
    <div className="item">
      <Link to={`/book/${id}`}>
        <img src={`http://localhost:3003/${image}`} alt={name} />
        <h3>{name}</h3>
      </Link>
      <p>{price} грн</p>
      <button
        className="add"
        onClick={() => addToCart(id)}
        disabled={quantity === 0}
      ><p className="cart">додати до кошика</p></button>
    </div>)}
export default Item
```

Рисунок 2.8 — код компоненту Item

Задля роботи фільтру був розроблений хук, який у даній роботі називається useBooksFilter. Окрім фільтрації книг за категоріями, даний компонент розроблений з метою синхронізації вибраного фільтру із URL-адресою.

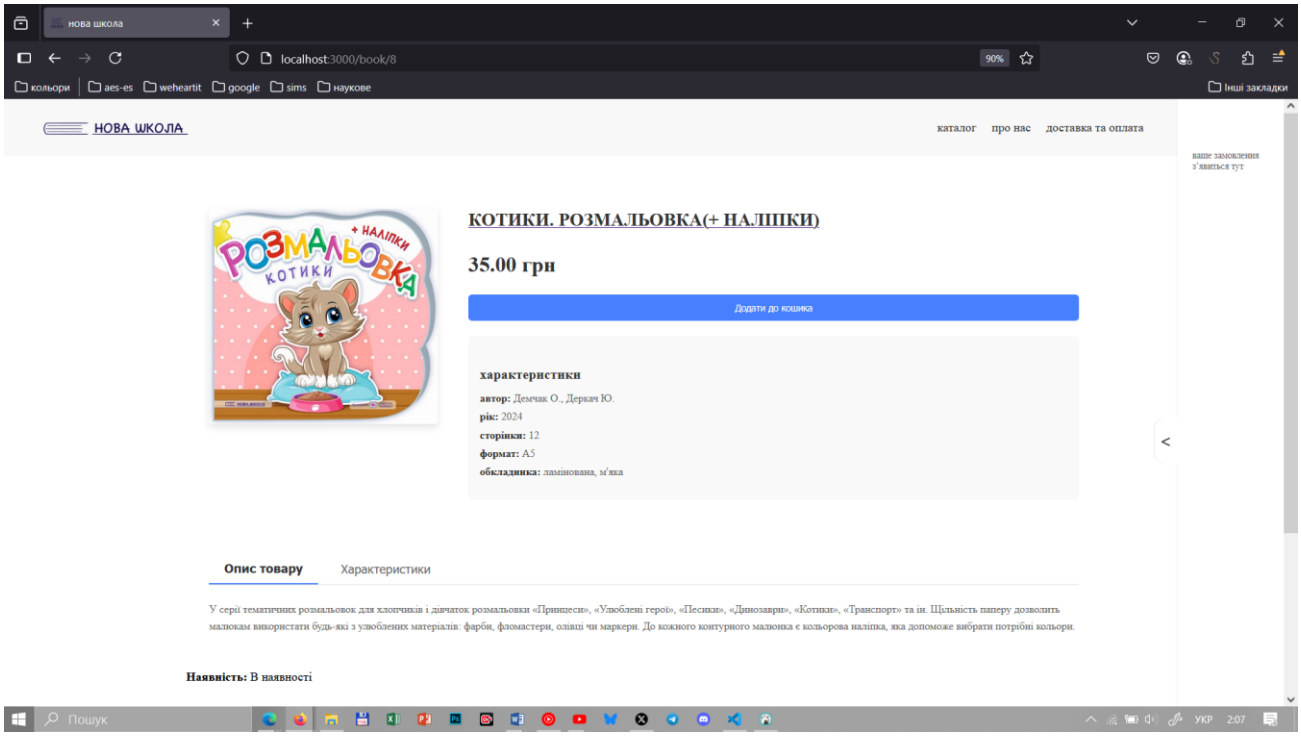


Рисунок 2.9 — вигляд розмальовки «Котики»

Задля даної синхронізації відбувається ініціалізація стану з URL, а саме, при зміні шляху, хук аналізує покликання і встановлює відповідні категорію або підкатегорію. Для цього використовується мапінг всередині ефекту за допомогою умовного оператора if-else. Він аналізує сегменти шляху за допомогою таких відповідностей як urlToCategoryMap, і, на його основі встановлює значення selectedCategory. Таке саме відбувається і з підкатегоріями.

Опісля перевірки на відповідність URL з компонентами категорії або підкатегорії, виводиться масив книг, який ініціалізований як filteredBooks. За замовчуванням, коли нічого не було обрано — на сторінку виводиться перелік усієї продукції. Даний масив також користується умовним оператором, який виводить категорію чи підкатегорію за умови, якщо даний елемент був обраний користувачем.

Даний хук використовується як у каталозі, так і в окремому елементі, а саме у фільтрі. Він реалізований у вигляді бічної панелі та дозволяє користувачу обирати що саме він хоче переглянути на даний момент: увесь каталог, чи тільки якусь конкретну категорію.

На рисунку 2.10 зображено виведення фільтру у браузері.

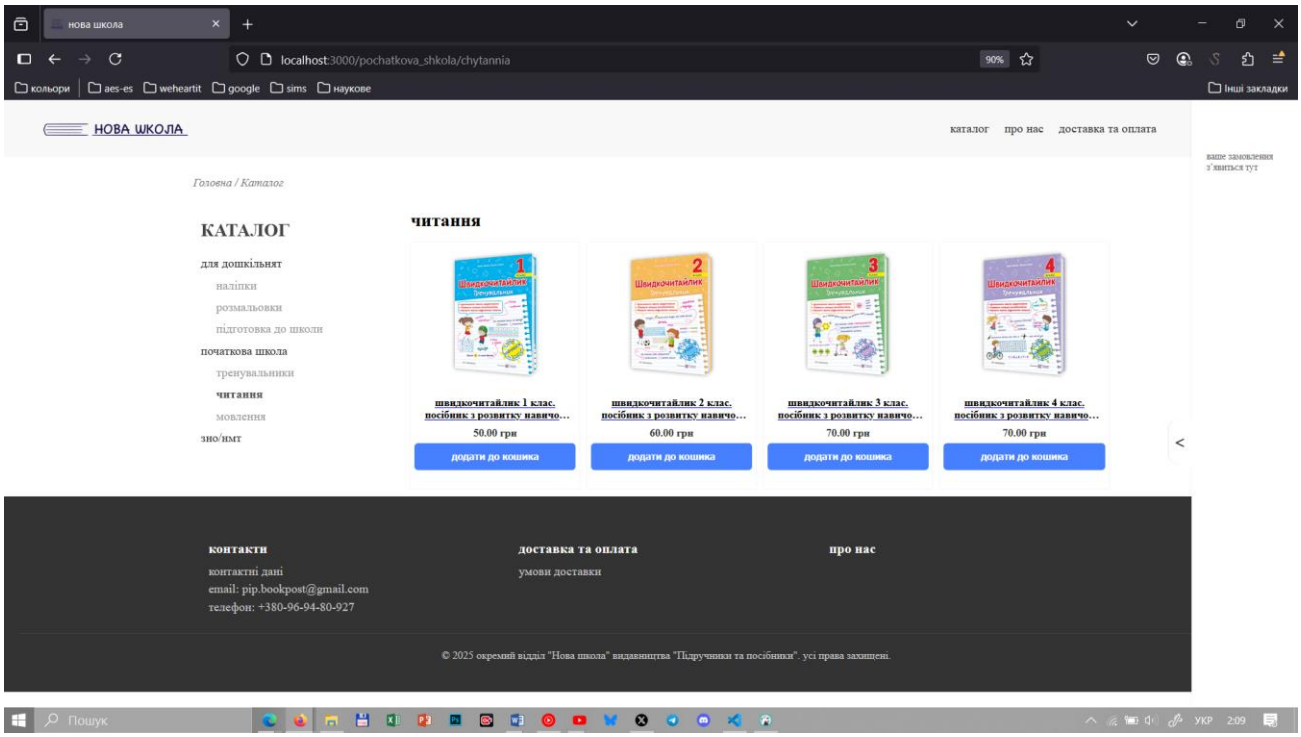
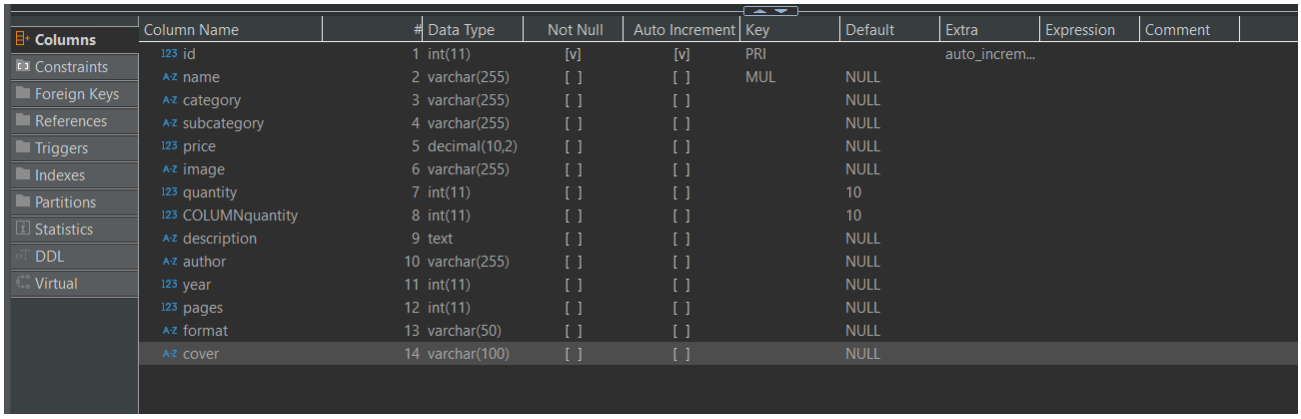


Рисунок 2.10 — вигляд відфільтрованого переліку товарів за підкатегорією «читання»

Каталог формується завдяки інформації, що внесена у таблицю баз даних. Для виведення інформації про товар, програма звертається до неї через серверну частину з метою отримати такі дані, як назва продукції, її рік випуску, автор, опис тощо. На рисунку 2.11 можна ознайомитись з характеристиками даної таблиці.



Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
id	1	int(11)	[v]	[v]	PRI		auto_increm...		
name	2	varchar(255)	[ ]	[ ]	MUL	NULL			
category	3	varchar(255)	[ ]	[ ]		NULL			
subcategory	4	varchar(255)	[ ]	[ ]		NULL			
price	5	decimal(10,2)	[ ]	[ ]		NULL			
image	6	varchar(255)	[ ]	[ ]		NULL			
quantity	7	int(11)	[ ]	[ ]		10			
COLUMNquantity	8	int(11)	[ ]	[ ]		10			
description	9	text	[ ]	[ ]		NULL			
author	10	varchar(255)	[ ]	[ ]		NULL			
year	11	int(11)	[ ]	[ ]		NULL			
pages	12	int(11)	[ ]	[ ]		NULL			
format	13	varchar(50)	[ ]	[ ]		NULL			
cover	14	varchar(100)	[ ]	[ ]		NULL			

Рисунок 2.11 — характеристика таблиці «books»

На рисунку 2.12 можна ознайомитись з виглядом переліку товарів у базі даних.

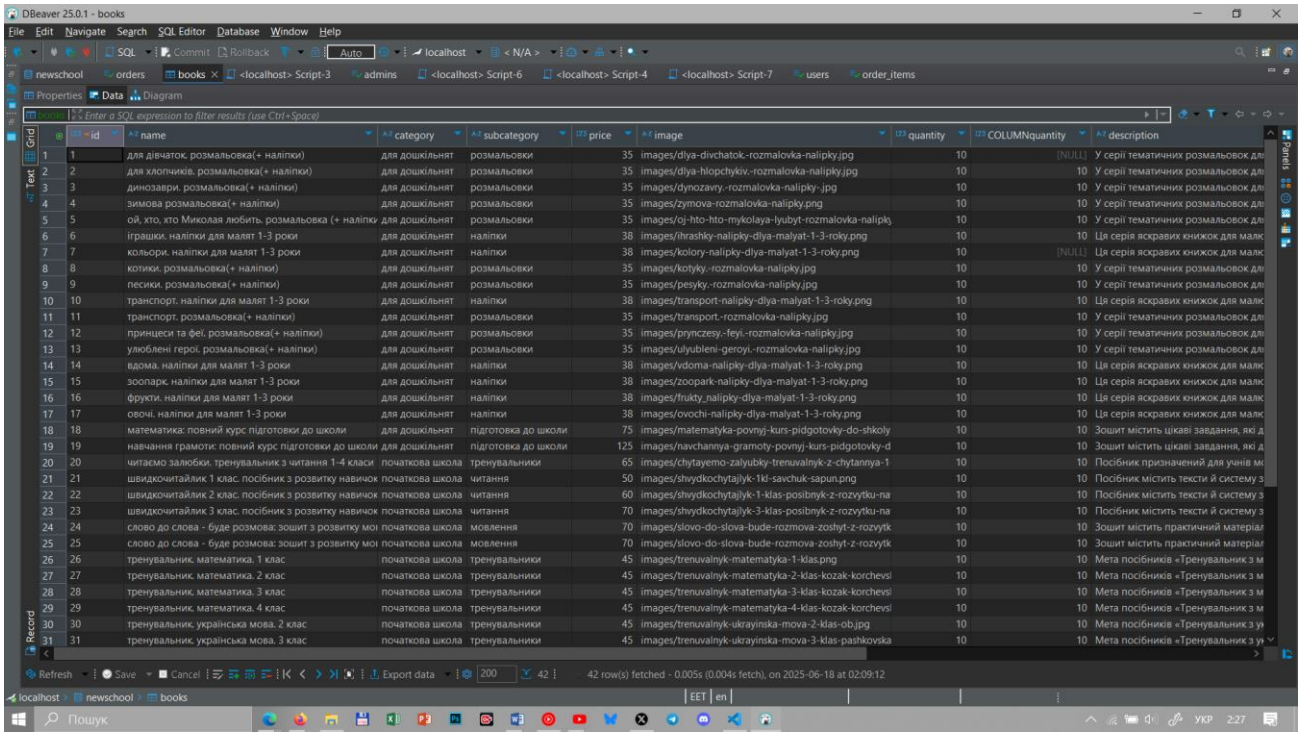


Рисунок 2.12 — перелік товарів у базі даних

Окремим компонентом тут є кошик, про який ми крадькома згадували коли розглядали контекст ShopContext для роботи інтернет-магазину. В ньому — контексті — зберігаєть стан кошика, який містить масив cartItems, а також наділений функціями додавання та видалення товарів, або ж зміною їхньої кількості, виведенням ціни за товар а також загальної суми до сплати. Дані кошика синхронізуються з елементом localStorage, завдяки чому інформація — обраний товар та його кількість зі загальною сумою — не втрачаються після оновлення чи навіть закриття сторінки.

Окремим компонентом тут є кошик, про який ми крадькома згадували коли розглядали контекст ShopContext для роботи інтернет-магазину. В ньому — контексті — зберігаєть стан кошика, який містить масив cartItems, а також наділений функціями додавання та видалення товарів, або ж зміною їхньої кількості, виведенням ціни за товар а також загальної суми до сплати. Дані

кошика синхронізуються з елементом localStorage, завдяки чому інформація — обраний товар та його кількість зі загальною сумою — не втрачаються після оновлення чи навіть закриття сторінки.

Кошик відображається у трьох видах. Перший раз, користувач зустрічає його у вигляді бічної панелі, яка має адаптивну властивість. За замовчуванням, бічна панель згорнута, проте, задля кращого ознайомлення з переліком обраного товару, його можна розгорнути та переглянути вміст. Це відбувається за допомогою кастомного хуку useCartFunctions, у якому це прописано у стані useState як const [isMinimized, setIsMinimized] = useState(true).

Як згадувалось раніше, у роботі кошика також використовуються розроблені окремо хуки, одним з яких є useCartFunctions(див. рис. 2.13), що відповідає за надання зручного інтерфейсу під час роботи з кошиком. Він дозволяє отримувати дані, підраховувати загальну кількість товарів та суму, а також відповідає за роботу кошика та його бічної панелі. Задля цього, він звертається до глобального контексту — ShopContext, — задля правильної реалізації своєї роботи.

src > hooks > JS useCartFunctions.js > [@] default

```
import { useContext, useEffect, useState } from "react"; import { ShopContext } from "../context/shopContext";

const useCartFunctions = () => {
    const {cartItems = [], addToCart, removeFromCart} = useContext(ShopContext);
    const [isMinimized, setIsMinimized] = useState(true);
    //локал. контроль кількості
    const [localCartItems, setLocalCartItems] = useState(cartItems);
    //синхрон. localCartItems із cartItems, якщо вони змінюються
    useEffect(()=> {
        setLocalCartItems(cartItems);
    }, [cartItems])
    const totalItems = localCartItems.reduce((sum, item) => sum + (item.quantity || 1), 0)
    const totalPrice = localCartItems.reduce((sum, item) => sum + item.price * (item.quantity || 1), 0)
    const toggleSidebar = () => {
        setIsMinimized(!isMinimized);
    };
    const handleIncrease = (item) => {
        const updatedItems = localCartItems.map((cartItem) => {
            if (cartItem.id === item.id) {return { ...cartItem, quantity: (cartItem.quantity || 1) + 1 }}
            return cartItem;
        })
        setLocalCartItems(updatedItems)
        addToCart(item.id, 1)
    }
    const handleDecrease = (item) => {if (item.quantity > 1) {addToCart(item.id, -1)} else {removeFromCart(item.id)}}
    return {
        cartItems: localCartItems, isMinimized, totalItems,
        totalPrice, toggleSidebar, handleIncrease, handleDecrease}
}
export default useCartFunctions
```

Рисунок 2.13 — вигляд коду useCartFunctions

Незадовго до формування кошику, або ж просто під час запуску вебсайту, на бічній панелі відображається текст «ваше оформлення з'явиться тут», що допомагає користувачі краще зорієнтуватись з розташуванням кошику на сайті.

Другий стан кошика реалізований у вигляді окремої сторінки, де замовлення відображається у вигляді повної таблиці переліку товарів, завдяки якій можна остаточно звіритись з обраною продукцією та сумою до сплати перед тим, як перейти до кінцевого етапу під час купівлі товарів.

Дана сторінка, як і бічна панель, яку ми розглянули вище, наділена усіма властивостями кошика. За замовчуванням, допоки іще не обрано ніякого товару, сторінка відображає інформацію про відсутність сформованого замовлення та виводить додатково кнопку задля повернення до каталогу.

На рисунку 2.14 можна ознайомитись з виглядом пустого кошику.

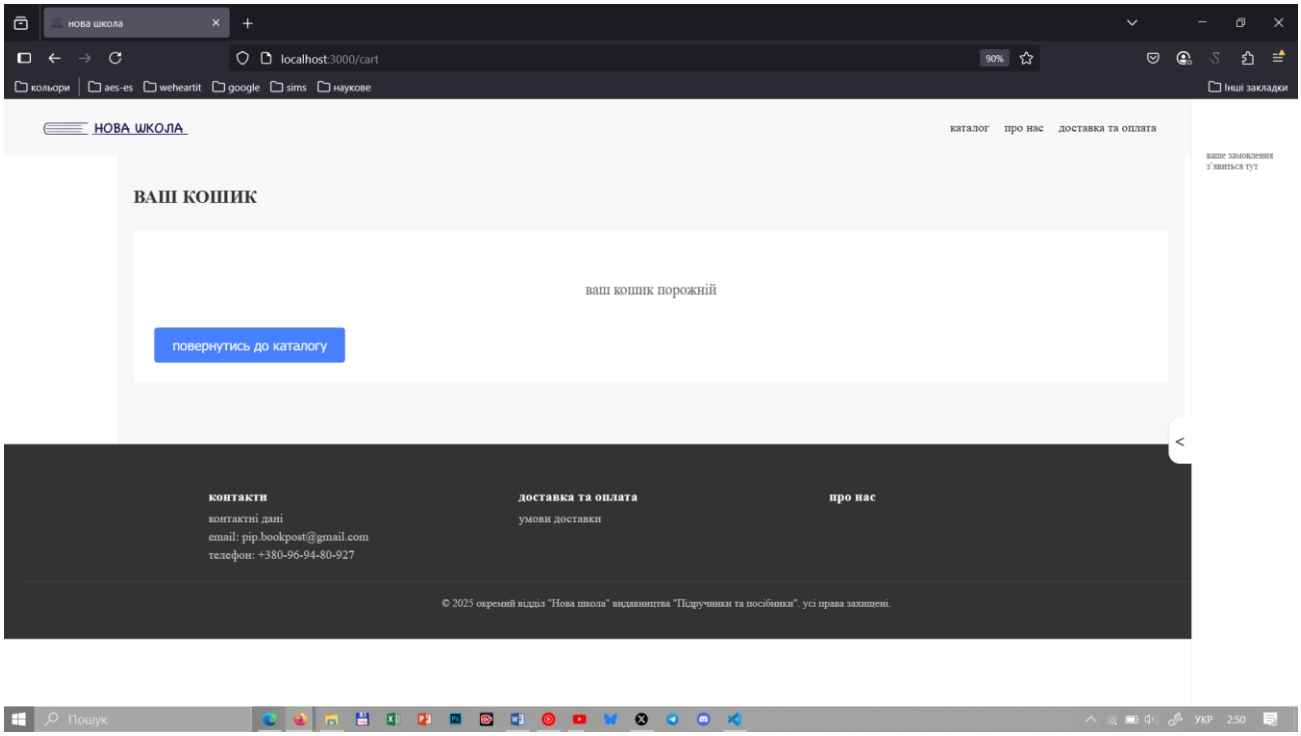


Рисунок 2.14 — вигляд пустого кошику

Наступним елементом є компонент, що реалізовує сторінку оформлення замовлення, що у коді записано як Checkout. Даний компонент дозволяє користувачу переглянути перелік товарів зі сформованого кошика без змоги його

редагування, а також введенням необхідних даних для оформлення оплати та доставки.

Задля роботи даного компоненту використовуються хуки, реалізовані самим React, які відповідають за стан, контекст та ефект —useState, useContext та useEffect відповідно. З хуку useCartFunctions, який ми розглядали вище, отримуються такі дані як перелік товарів у кошику та загальна сума до оплати.

Окрім цього, був створений окремо хук — useCartOrder, який відповідає за роботу з формою замовлення.

У даному компоненті реалізоване найголовніше, а саме — керування станом форми задля остаточної підготовки даних для відправки замовлення. Він (хук) забезпечує зручну роботу з контактними та адресними даними, вибором способу доставки та місця призначення задля відправки замовлення у майбутньому. Форма ініціалізується за домогою стану як formData(див. рис. 2.15), яка наділена відповідними полями. Більшу частину з них необхідно вводити самостійно, проте адресні поля реалізовані як список на основі даних з відповідних json.

```
17  const [formData, setFormData] = useState({
18    lastName: "",
19    firstName: "",
20    email: "",
21    phone: "",
22    region: initialRegion,
23    city: hasCities ? regions.city[initialRegion][0].value : "",
24    delivery: "nova-poshta",
25    branch: "",
26    payment: "visa",
27    comment: "",
28  });
```

Рисунок 2.15 — код ініціалізації форми formData у хуці useCartOrder

За допомогою useEffect хук слідкує за змінами способу доставки, а саме служба, якою це буде організовано, область та місто. Якщо існує декілька відділень за даних критерій у переліку — у коді це використано через умову if (availableBranches.length > 0) {…}, — то, за замовчуванням, обирається перше доступне: branch: availableBranches[0].value.

За допомогою функції handleInputChange(див. рис. 2.16) автоматично оновлюються форми при зміні будь-якого поля. Це стосується радше адресних даних, ніж інших. Тобто, при зміні області видозмінюється перелік міст та відділень, а також, при виборі служби доставки перелік відділень проходять повторну обробку з відсіюванням відділень інших служб з переліку.

```
40  const handleInputChange = (e) => {
41    const { name, value } = e.target;
42    setFormData((prev) => {
43      const newFormData = { ...prev, [name]: value };
44      if (name === "region") {newFormData.city = regions.city[value]?.[0]?.value || ""; newFormData.branch = ""}
45      if (name === "delivery" || name === "city") {newFormData.branch = branchData.branch[value]?.[formData.city]?.[0]?.value || ""}
46      return newFormData})
47  }
```

Рисунок 2.16 — вигляд коду функції handleInputChange у хуці useCartOrder

Компонент Checkout є одним з небагатьох, хто використовує також і контекст задля авторизації користувача або обробки даних про нього задля кращої реалізації співпраці між видавництвом та покупцем. Наприклад, у даному компоненті, під час оформлення замовлення, можна підтягнути автомачно свої власні дані з особистого кабінету за допомогою AuthContext, який використовує стан currentUser.

Після оформлення замовлення, на серверну частину відправляється POST-запит.

Раз ми вже згадали про авторизацію, давайте розглянемо роботу особистого кабінету та реєстрацію. Реєстрація та вхід реалізовані за допомогою однієї сторінки, яка змінює вигляд форми та викликається у проєкті під назвою Login, а сам кабінет — Profile. Обидві сторінки використовують створений окремо хук useLoginSignUp, а для особистого кабінету був розроблений окремо ще і useOrders.

Компонент Login дозволяє перемикатися між формами логіну та реєстрації. За замовчуванням, у вебдодатку викликається форма входу, проте за допомогою текстової кнопки, що реалізована як <span className="switch-link" onClick={() => setIsLoginForm(false)}>зареєструватися відбувається зміна форми з метою подальшої реєстрації користувача.

Задля кращої роботи даного компоненту був реалізований хук useLoginSignUp. Він керує станом форм, обробляє введення та відправку даних на сервер, перемикання між формами, а також вихід з профілю. Задля виведення форми входу використовується булевий стан isLoginForm.

Зміна між формами також відбувєаться завдяки реалізованому обробнику handleRegisterChange, який, коли isLoginForm видає стан false, змінює відповідне поле у формі реєстрації. handleRegisterSubmit перевіряє, чи усі поля заповнені та передає їх у POST-запит на серверну частину. У разі успіху, відбувається зміна форми на відповідні поля для входу, де необхідно ввести лише електронну пошту та пароль, після чого відбувається ще один запит до бекенду, який після перевірки дозволяє особі увійти як звичайний користувач або адміністратор.

На рисунку 2.17 наведено у приклад код обробника handleRegisterSubmit.

```
const handleRegisterSubmit = async (e) => {
  e.preventDefault();
  const { username, lastName, email, phone, password } = registerForm;
  if (!username || !lastName || !email || !phone || !password) {setError('будь ласка, заповніть усі обов'язкові поля.'); return}
  try {
    const response = await fetch('http://localhost:3003/api/user/register', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ username, lastName, email, phone, password }),
    })
    if (!response.ok) throw new Error('Registration failed: ' + (await response.text()))
    const data = await response.json()
    login(data.user, loginForm.remember)
    alert('реєстрація пройшла успішно')
    setError('');
    setRegisterForm({ username: '', lastName: '', email: '', phone: '', password: '' });
    setIsLoginForm(true)
  } catch (err) {setError(err.message)}
}
```

Рисунок 2.17 — елемент коду обробника handleRegisterSubmit хуку useLoginSignUp

Компонент Profile реалізовує сторінку особистого кабінету користувача, де можна переглянути особисті дані, а також список раніше оформлених замовлень. Дані поточного користувача беруться з AuthContext за допомогою стану currentUser, а вигляд замовлень отримуються через створений окремо хук useOrders.

Хук useOrders реалізований задля отримання пелеріку замовлень поточного

користувача зі серверу за допомогою параметра userIdentifier. Даний елемент відправляє GET-запит на сервер з метою отримати дані, використовуючи електронну пошту покупця.

За допомогою асинхронної функції fetchOrders відбувається звернення до бази даних з метою отримати перелік створених раніше замовлень.

На рисунку 2.18 зображено вигляд хуку useOrders.

Рисунок 2.18 — код хуку useOrders

Окрім клієнтської частини також реалізована і примітивна адміністративна панель, яка ініціалізована як AdBoard. У ній реалізовано зміна даних про товар, додавання нового а також перегляд усіх реалізованих замовлень різними користувачами.

Вхід в адміністративну панель відбувається за допомогою checkbox у формі входу, а також за допомогою хуку useLoginSignUp(див. рис. 2.19). Це реалізовано через функцію handleLoginSubmit, який через серверну частину перевіряє якою даний користувач наділений роллю та перенаправляє на відповідну сторінку: звичайного користувача до власного профілю, а адміністратора — до відповідної панелі.

```
const handleLoginSubmit = async (e) => {
  e.preventDefault()
  const {login:loginField, password, remember, admin_check} = loginForm
  if (!loginField || !password) {setError('будь ласка, заповніть усі обов'язкові поля.'); return}
  console.log('Sending login request to:', admin_check?'/api/admin/login':'/api/user/login');
  try {
    const baseUrl = 'http://localhost:3003';
    const endpoint = admin_check ? '/api/admin/login':'/api/user/login';
    const body = admin_check ? {username: loginField, password}:{email: loginField, password}
    const response = await fetch(baseUrl + endpoint, {
      method: 'POST',
      headers: {'Content-Type':'application/json'},
      body: JSON.stringify(body),
    })
    if (!response.ok) throw new Error(await response.text())
    const data = await response.json()
    login(data.user, remember)
    const { from, cartItems, formData } = location.state || {}
    const redirectPath = admin_check ? '/admin/dashboard' : (from || '/profile')
    navigate(redirectPath, { state: { cartItems, formData }, replace: true })
    alert('вхід успішний!')
    setError('')
  } catch (err) {setError(err.message || 'невірне ім'я користувача, email або пароль.')}
}
```

Рисунок 2.19 — використання обробника handleLoginSubmit у хуці useLoginSignUp

AdBoard розділена на дві частини, а саме на перелік товарів та замовлення. За замовчуванням виводиться каталог у вигляді таблиці з переліком усіх наявних на сайті товарів, які отримуються з бази даних, як це зображено на рисунку 2.20.

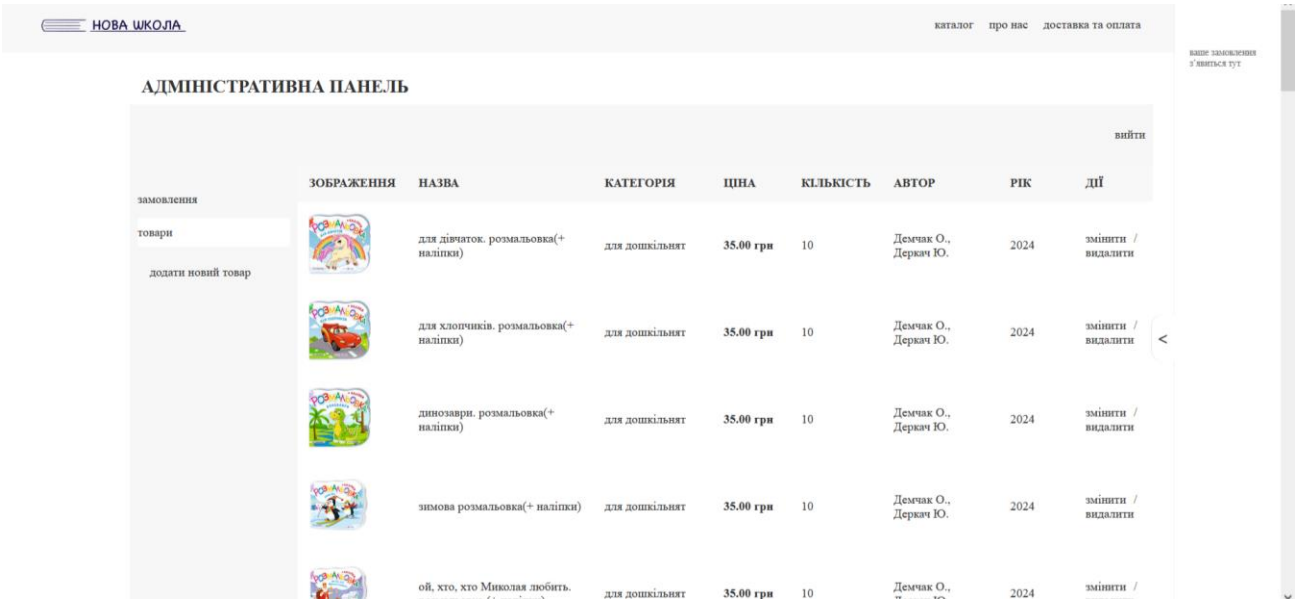


Рисунок 2.20 — вигляд адміністративної панелі

У майбутньому проєкт буде вдосконалюватись з метою більшої автоматизації задля повної реалізації вебдодатку. Серед майбутніх вдосконалень очікується хешування паролів задля кращої безпеки інформації; реалізація платіжної системи з підкріпленням основної мети інтернет-магазину, а саме — продаж-купівля товарів; впровадження адаптивного дизайну для різних екранів за допомогою CSS Grid або Bootstrap. Також планується редагування адміністративної панелі, а саме її автоматизація, редагування та додавання нових категорій тощо.

2.3 Огляд та структура сайту

На головній сторінці перехід на каталог або ж на якусь з категорій товару прописано через динамічний CSS-стиль, який відповідає за зміну розмірів блоку чи смужки при наведенні на них курсором. Кожен блок та смуга є активною та при натисканні переходить на сторінку каталогу(див. рис. 2.21) або обрану категорії/підкатегорію користувачем.

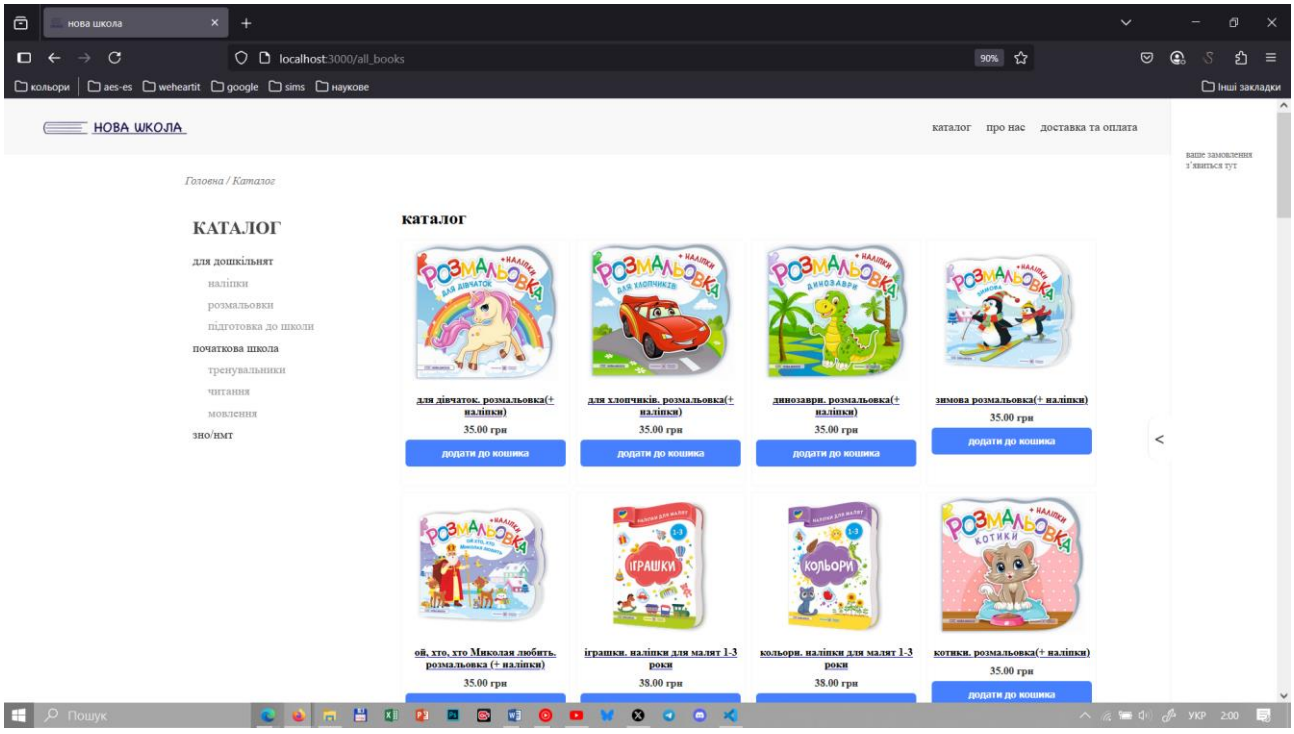


Рисунок 2.21 — вигляд каталогу

Під час вибору товару для того, аби занести його у своє замовлення, за допомогою кнопки «додати до кошика», вибрана продукція з’являється у панелі, що знаходиться праворуч. Дана панель формує кошик, та дозволяє користувачеві переглянути обраний товар, відредагувати його кількість та перейти до оформлення замовлення. При натисканні кнопок «-» та «+» регулюється його кількість. Якщо була обрана книжка у кількості 1 штука, то при натисканні на знак «-» вона зникає з кошика.

На рисунках 2.22 і 2.23 можна ознайомитись з виглядом бічної панелі та формуванню кошика при виборі товару.

Рисунок 2.22 — вигляд пустої бічної панелі

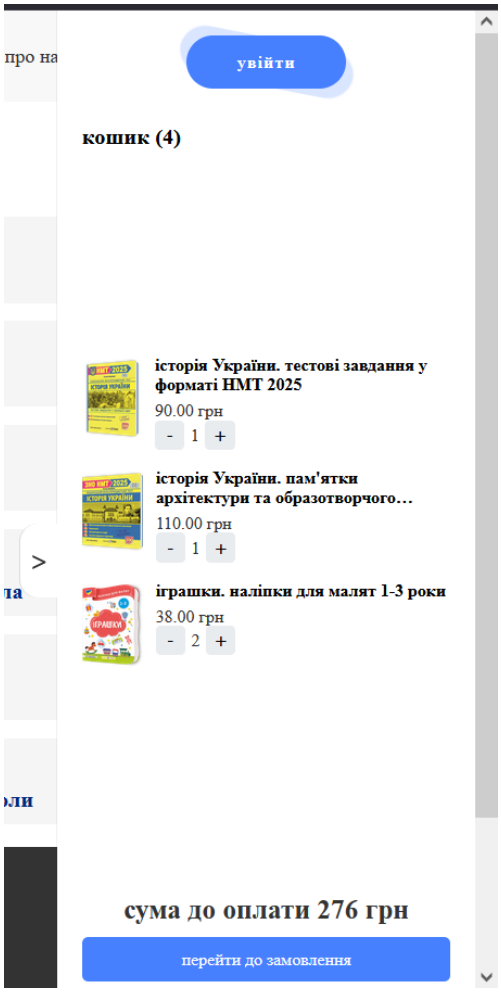


Рисунок 2.23 — вигляд бічної панелі з обраним товаром

При натисканні кнопки у бічній панель «перейти до замовлення» спочатку відкривається окрема сторінка кошика, аби користувач мав змогу ще раз

перевірити обраний ним товар. На даній сторінці замовлення виводить у вигляді таблиці, у якій відображається зображення товару, його назва, обрана покупцем кількість, ціна та сума до сплати. Колонка «ціна» відповідає за ознайомлення покупця з ціною однієї книжки, у той час як колонка «усього» відображає суму за декілька книг однієї позиції.

В самому кінці відображається сума до сплати за всі позиції, а також кнопка «оформити доставку», яка відповідає за відображення останнього етапу оформлення замовлення.

На рисунку 2.24 можна ознайомитись з виглядом кошика.

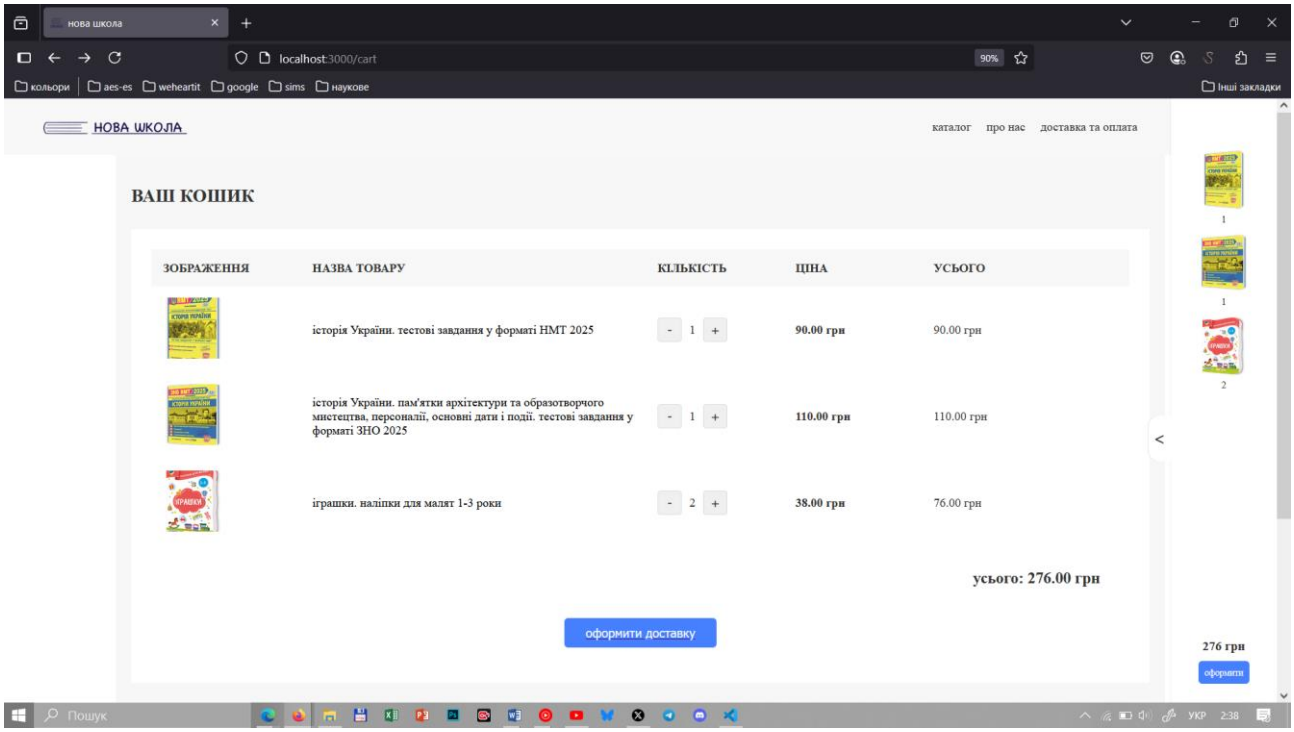


Рисунок 2.24 — вигляд сформованого кошика

Після того, як покупець іще раз переглянув свій кошик, він переходить до оформлення доставки замовлення. Дана сторінка поділена на дві частини: угорі виводиться перелік товарів та його кількість, а також сума до сплати за обраний товар; друга частина — це форма, яку необхідно заповнити.

Дана форма розбита на блоки для кращої читабельності та поділена за тематиками: покупець, доставка, оплата та коментар. Внизу сторінки знаходиться

кнопка «оформити замовлення». У блоці «покупець» користувач може вручну ввести інформацію — контактні дані, — або ж завдяки автозаповненню полів, яку можна реалізувати завдяки доступу до кабінету. Для цього необхідно навести курсив на слово «зареєстровані?» та натиснути на нього. Даний текст перекидає користувача на сторінку входу, аби увійти у свій кабінет. Як тільки користувач ввійде у власний профіль, сайт одразу підтягує інформацію та автоматично прописує її у формі.

Блок «доставка» відповідає за адресні дані, де необхідно зі списку обрати необхідну інформацію. Тут обирається область, місто, комфортна служба доставки та відділення.

Дана сторінка зображена на рисунку 2.25.

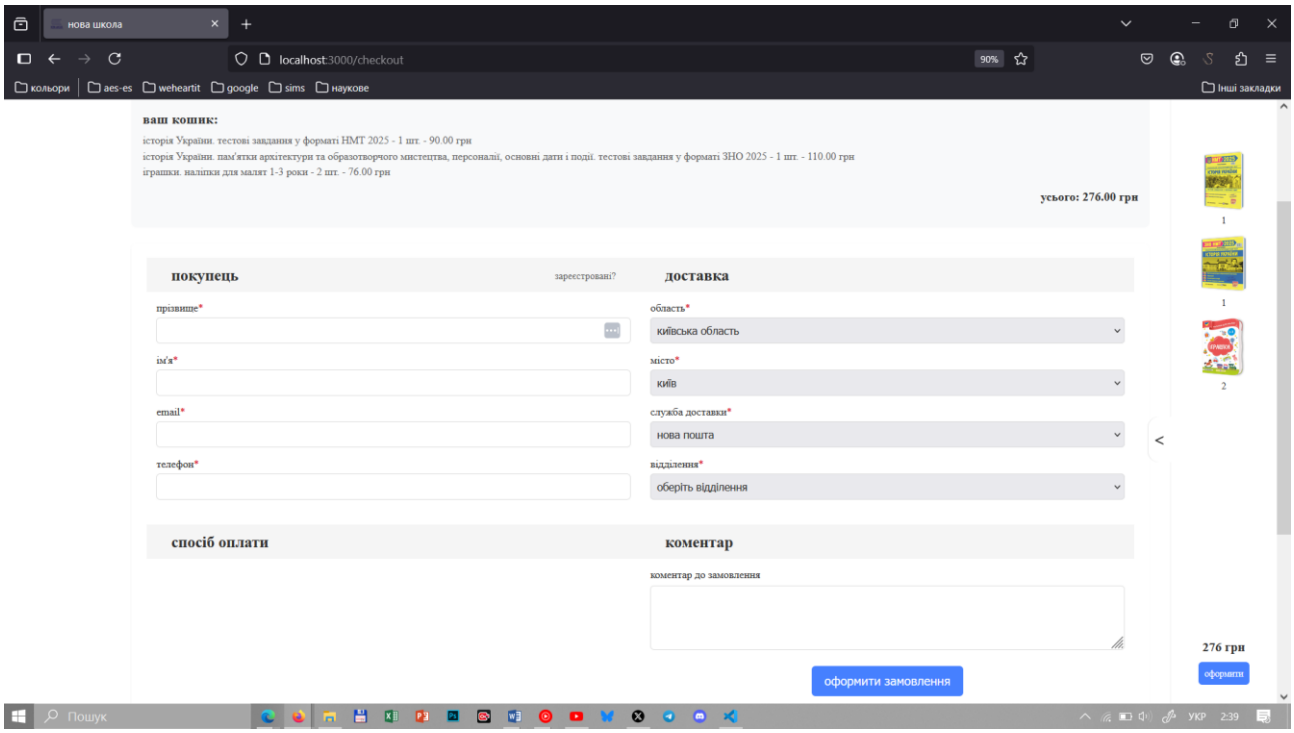


Рисунок 2.25 — вигляд оформлення замовлення

Створення та вхід до власного кабінету виглядає наступним чином: на бічній панелі необхідно натиснути на кнопку «увійти». Вона змінює вміст основного блоку на сторінку входу, де необхідно заповнити невелику форму задля авторизації. Опісля цього, вебдодаток виведе сторінку з вмістом про

користувача та його створені раніше замовлення.

При відсутньому особистому кабінеті його можна створити, перейшовши за покликанням «немає акаунту? зареєструватися». Дане покликання змінить вигляд форми, де треба увести власні дані. Опісля успішної реєстрації вебдодаток змінює сторінку на вхід в особистий кабінет.

З виглядом форм для входу та реєстрації можна ознайомитись на рисунках 2.26 і 2.27 відповідно.

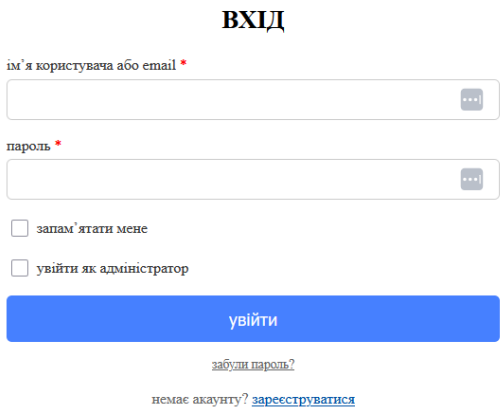


Рисунок 2.26 — вигляд форми входу

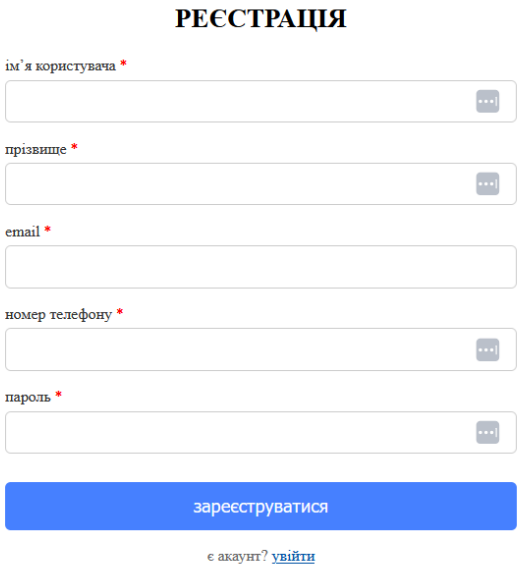


Рисунок 2.27 — вигляд форми реєстрації

Після успішного входу, вебдодаток виводить профіль. Дана сторінка також розділена на дві частини: вгорі короткий перелік даних користувача, а саме ім'я, прізвище, електронна пошта та телефон; а трохи нижче — оформлені раніше замовлення. Дана частина сторінки дозволяє ознайомитись з переліком замовленого товару, сумою до оплати, видом доставки та датою створення замовлення.

З виглядом особистого кабінету можна ознайомитись на рисунку 2.28.

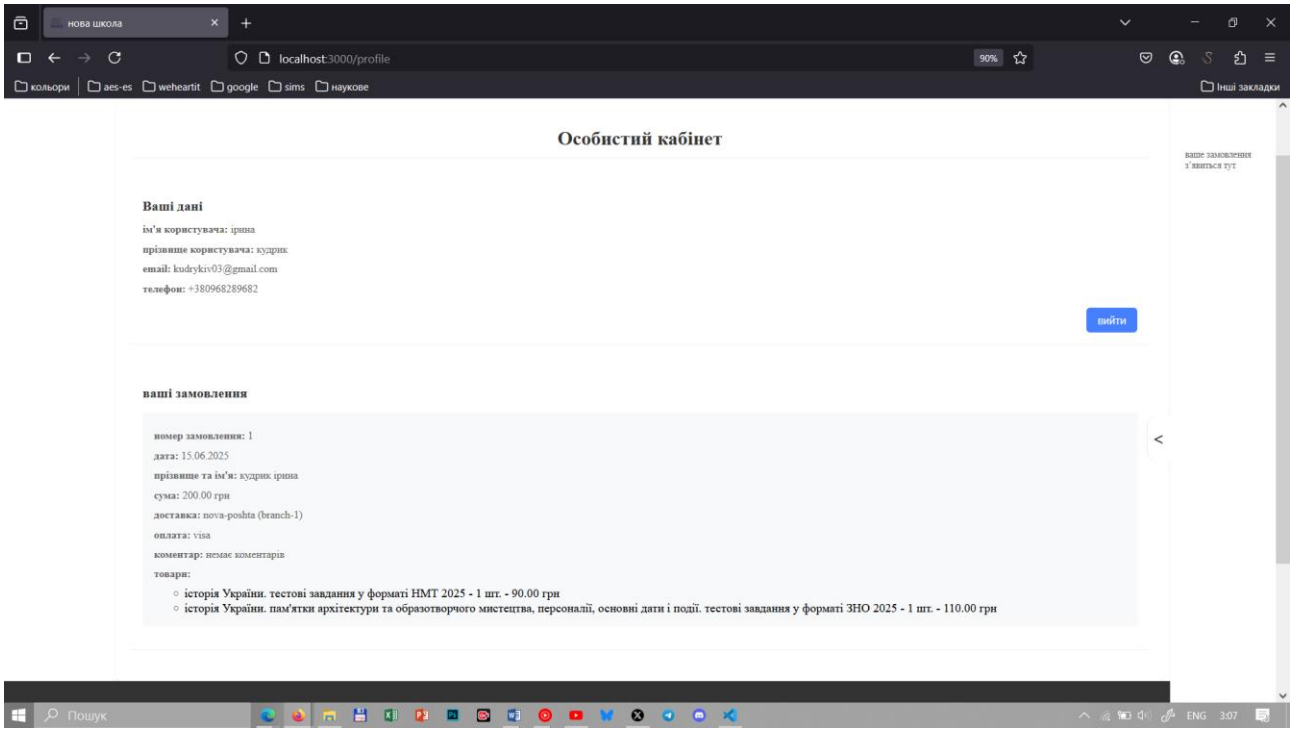


Рисунок 2.28 — вигляд особистого кабінету

2.4 Написання скрипту

Давайте розглянемо розробку ShopCategory.

Даний компонент відповідає за сторінку каталогу товарів з можливістю примітивної фільтрації за категоріями та підкатегоріями. Він відображає список книг у вигляді блоків та дозволяє користувачу обирати цікаву йому категорію за допомогою бічної панель-фільтру. Товар можна додати до кошику та переглянути окремо створену сторінку для кожної з книг.

2.4.1. Аналіз вимог та планування

Головною вимогою до даного скрипту є відображення книг та фільтрація за відповідними категоріями та їхніми підкатегоріями. Категорії «для дошкільнят» та «початкова школа» розділені по три підкатегорії кожна: для дошкільнят — «розмальовки», «наліпки» та «підготовка до школи»; для початкової школи — «тренувальники», «читання», «мовлення». Категорія

«ЗНО/НМТ» не наділена окремими категоріями через нестачу різноманітности товарів.

Перелік книг повинен отримуватися з бази даних MariaDB через GET-запит як API на серверній частині вебдодатку.

Даний компонент повинен бути реалізований за допомогою React як клієнтська частина, Express.js як серверна.

2.4.2. Створення таблиці books у MariaDB

Таблиця books відповідає за збереження переліку товарів та всі їхні характеристики. Створення її можна реалізувати за допомогою SQL-запиту (див. рис. 2.29) із необхідними полями. Опісля цього створюється пуста база даних, у яку необхідно внести всю інформацію. Це можна зробити як вручну, так і за допомогою різних команд.

Задля внесення інформації було використано INSERT INTO books в SQL-скрипті, завдяки якому відбувається внесення даних до таблиці за допомогою запиту VALUES, всередині якого необхідно прописати вручну усі позиції.

```
CREATE TABLE IF NOT EXISTS books (
        id INT AUTO_INCREMENT PRIMARY KEY,
        name VARCHAR(255),
        category VARCHAR(255),
        subcategory VARCHAR(255),
        price DECIMAL(10, 2),
        image VARCHAR(255),
        quantity INT DEFAULT 10,
        description TEXT,
        author VARCHAR(255),
        year INT,
        pages INT,
        format VARCHAR(50),
        cover VARCHAR(100)
    )
ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
```

Рисунок 2. 29 — вигляд SQL-запиту задля створення таблиці books

2.4.3. Розробка серверної частини

У розробці серверної частини використовується фреймворк Express.js на платформі Node на основі мови програмування JavaScript. Дана бібліотека відповідає за створення REST API, який обробляє різні HTTP-запити від фронтенду та комунікує з базою даних MariaDB.

У server.js, що відповідає за серверну частину сайту, прописуємо з'єднання до бази даних, як зображено на рисунку 2.31. База даних MariaDB інтегрується у серверну частину за допомогою функції const mysql = require('mysql2').

```
const db = mysql.createConnection({
host: 'localhost',
user: 'root',
password: '',
database: 'newschool',
port: 3306
});
```

Рисунок 2.31 — елемент коду зі server.js, що відповідає за підключення бази даних

Задля забезпечення даних для компонента ShopCategory, який відображає каталог книг з фільтрацією, необхідно використати GET-запит.

API виконує SQL-запит задля отримання усіх записів з таблиці та повертає JSON-масив об'єктів з отриманими з бази даних полями. У разі помилки повертає статус 500 з текстовим повідомленням.

На рисунку. 2.32 зображено api задля відображення усього переліку книг.

```
app.get('/api/books', (req, res) => {
  db.query('SELECT * FROM books', (err, results) => {
    if (err) return res.status(500).send('Error fetching books: ' + err.message)
    res.json(results)
  })
})
```

Рисунок 2.32 — вигляд API задля отримання переліку товарів з таблиці

Ендпоінт викликається через ShopContextProvider для завантаження всіх книг, які фільтруються на клієнтській стороні. Це відбувається за допомогою функції fetchBooks та використовується в асинхронному коді завдяки функції const response = await fetch('http://localhost:3003/api/books').

2.4.4. Розробка клієнтської частини з використанням React

При створенні ShopCategory у вебдодатку буде повністю пустим та нічого не відображатиме. Нижче наведено код створення пустої сторінки.

```
import React from "react"
const ShopCategory = () => {
  return (<div />)
}
export default ShopCategory
```

На рисунку 2.33 можна побачити вигляд пустої сторінки каталогу.

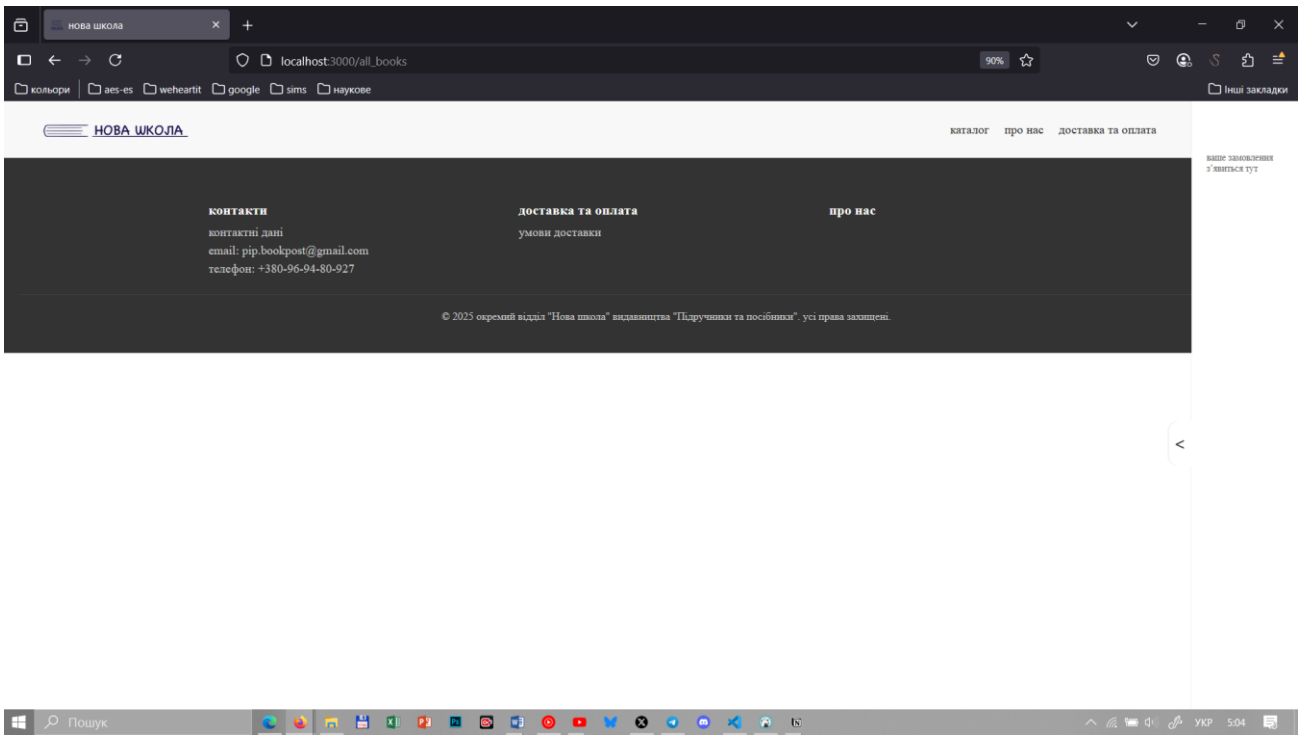


Рисунок 2.33 — вигляд ShopCategory при створенні

Додаємо контекст завдяки функції import{ShopContext} from "../context/shopContext". Всередині компоненту ShopCategory прописуємо що необхідно витягнути з контексту, а саме об’єкт books за допомогою react-хуку useContext:

const {books}=useContext(ShopContext)

У функції return виводимо перелік книг як звичайний список через елемент ul. Додаємо умову існування масиву через перевірку його довжини за допомогою books && books.length>0? . Даний перегляд проводить перевірку існування масиву та звіряється чи не є він порожнім. Тоді функція для кожної книги створює елемент списку з її назвою, як зображено на рисунку 2.34.

Це зроблено з метою переконатись, що ShopCategory має доступ до бази даних та виводить увесь перелік товарів коректно за допомогою ключа {book.name}.

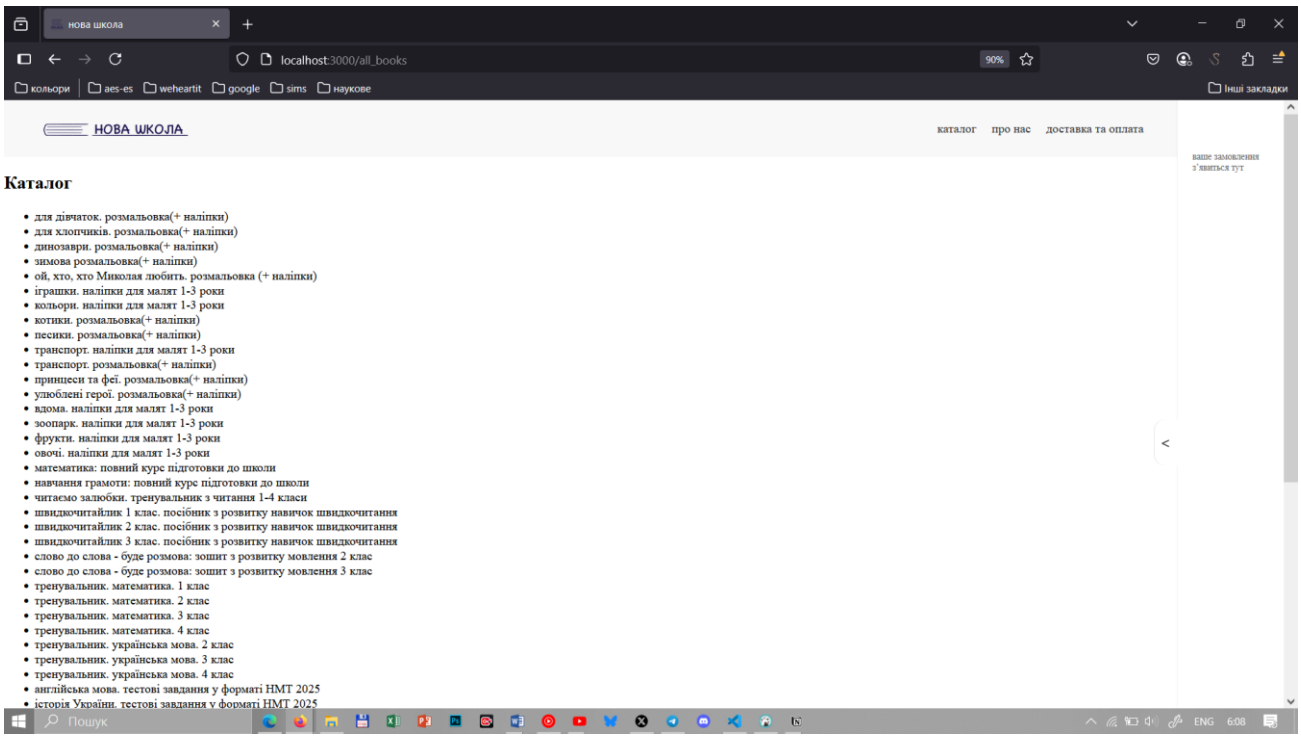


Рисунок 2.34 — зображення виведеного списку

Додаємо до залежностей Item, з якого витягуємо вигляд товарів та реалізовуємо далі ShopCategory як секції через контейнер div.

Задля виведення товарів створюється контейнер з назвою класу books-section всередині контейнера shop-category-container, який відображає масив books. Кожному компоненту передають основні дані про книгу, а саме індекс, назва, зображення та ціна.

На рисунку 2.35 зображено вигляд каталогу без використання CSS-стилів з використанням залежності Item.

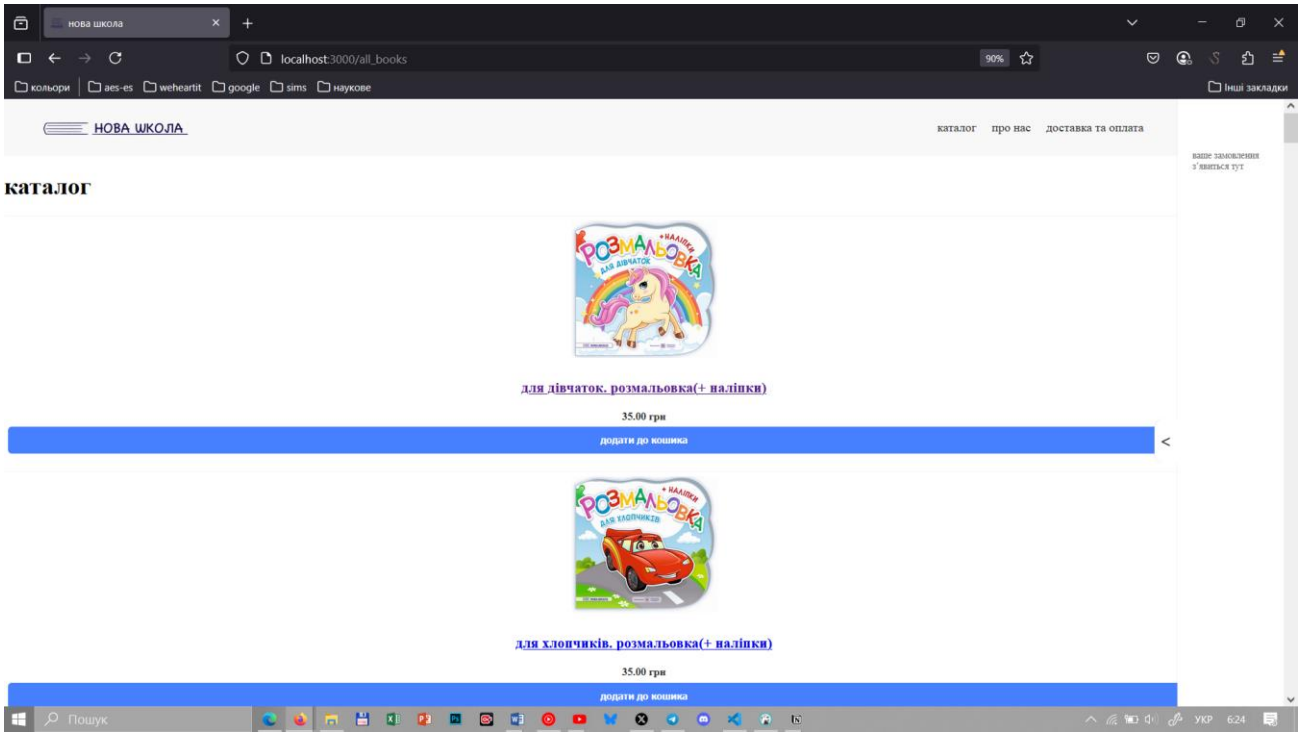


Рисунок 2.35 — вигляд каталогу завдяки компоненту Item

Додаємо до залежностей фільтр та його хук. Завдяки хуку useBookFilter перевіряємо масив books з глобального контексту та повертаємо дані, а саме: масив книг опісля фільтрації filteredBooks; вибрані категорію та підкатегорію selectedCategory та selectedSubcategory відповідно; функції для зміни фільтрів, такі як handleCategoryClick, handleSubCategoryclick та handleShowAll.

У функції return викликаємо сам фільтр, а у books-section змінюємо масив books на filteredBooks задля зміни вигляду товару.

Додаємо до залежностей також масив з переліком заголовків та викликаємо його у функції return.

На рисунку 2.36 зображено вигляд ShopCategory без використання стилів з роботою фільтру.

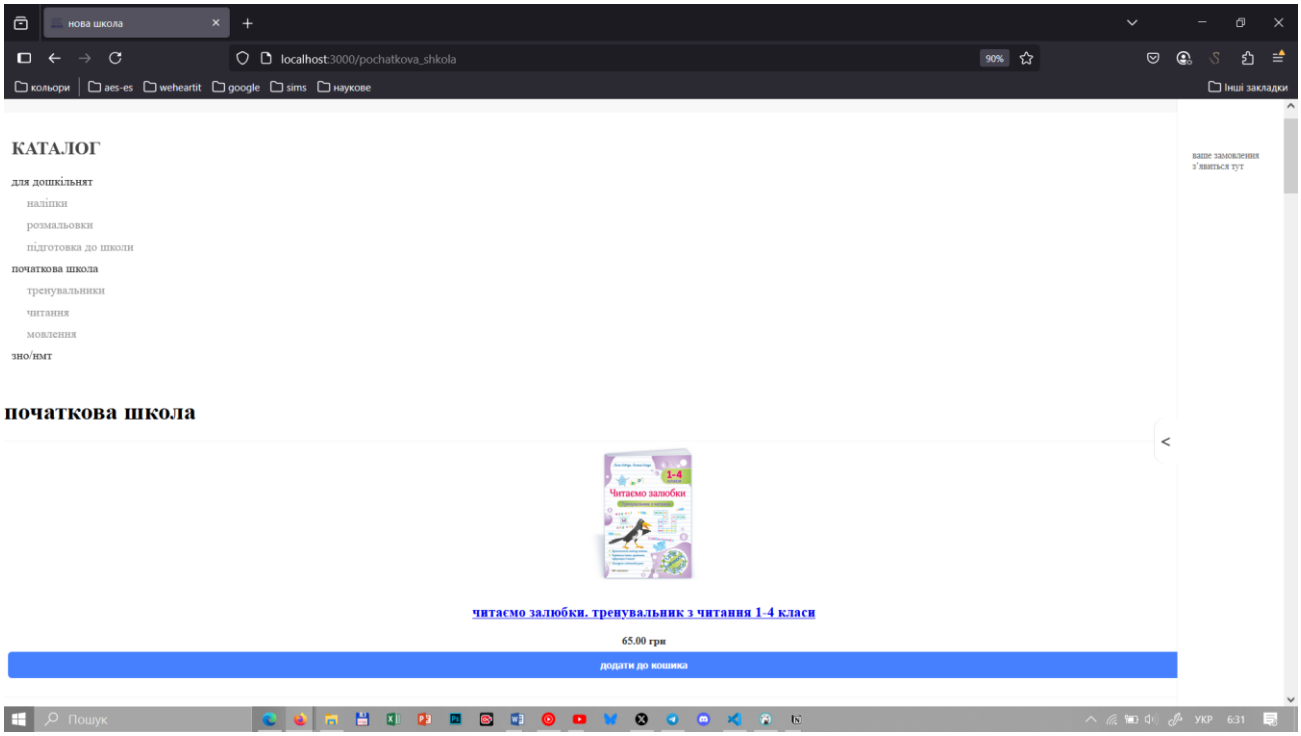


Рисунок 2.36 — вигляд каталогу з використанням фільтру

Прописуємо стилі у shop_category.css та додаємо даний файл до залежностей.

У стилях для сторінки задаємо максимальну ширину, встановлюємо відступи зверху і знизу та центруємо контейнер по горизонталі. Вмісту основного контейнеру — shop-category-container — надаємо гнучкості — display: flex — з відступом між елементами.

Товари відображаються у вигляді сіток. Кількість колонок автоматично підлаштовується під ширину екрану:

grid-template-columns: repeat(auto-fit, minmax(220px, 1fr));

Надаємо відступи між рядками та колонками та вирівнюємо елементи за сіткою. Окрім цього, товарам надаємо динамічності, а саме, у класі item:hover налаштовуємо збільшення масштабу при наведенні.

На рисунку 2.37 наведено кінцевий вигляд каталогу після застосування

стилів.

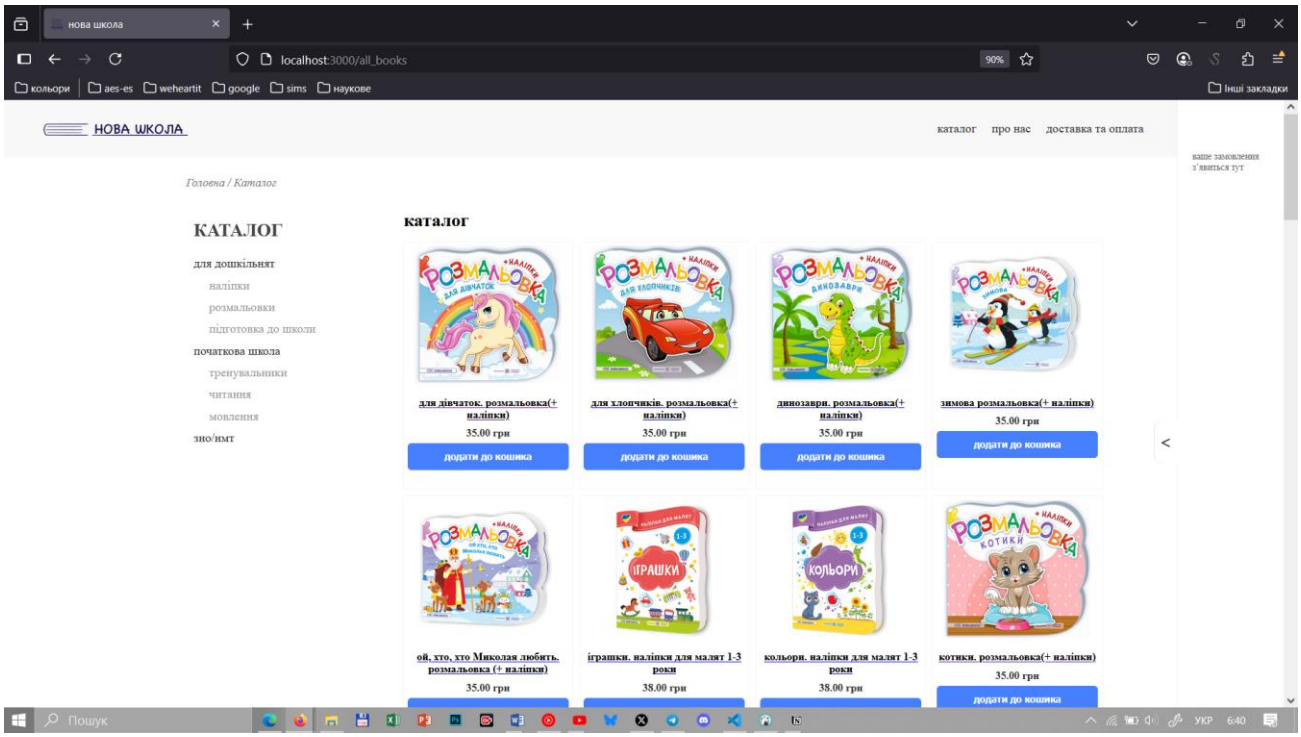


Рисунок 2.37 — кінцевий вигляд каталогу

3. СПЕЦІАЛЬНИЙ РОЗДІЛ

3.1 Інструкція зі встановлення Visual Studio Code

Задля встановлення Visual Studio Code необхідно зайти на їхній офіційний сайт — https://code.visualstudio.com/ та обрати «Завантажити для Windows». Опісля, необхідно запустити інсталятор та обрати необхідні опції та компоненти.

Першим компонентом є вибір мови. За замовчуванням стоїть англійська. Її й обираємо та ідемо далі. Опісля цього необхідно ознайомитись з ліцензією, де коротко описується функціонал VS Code та узгоджується обмін даними між користувачем та корпорацією Microsoft, яка і є розробником даного ПЗ. Наступним кроком є вибір додаткових завдань (Tasks), такі як, наприклад, створення ярлика на робочому столі.

Наприкінці натискаємо кнопку «Install», як зображено на рисунку 3.1. Після завершення процесу встановлення Visual Studio Code буде готовий до використання.

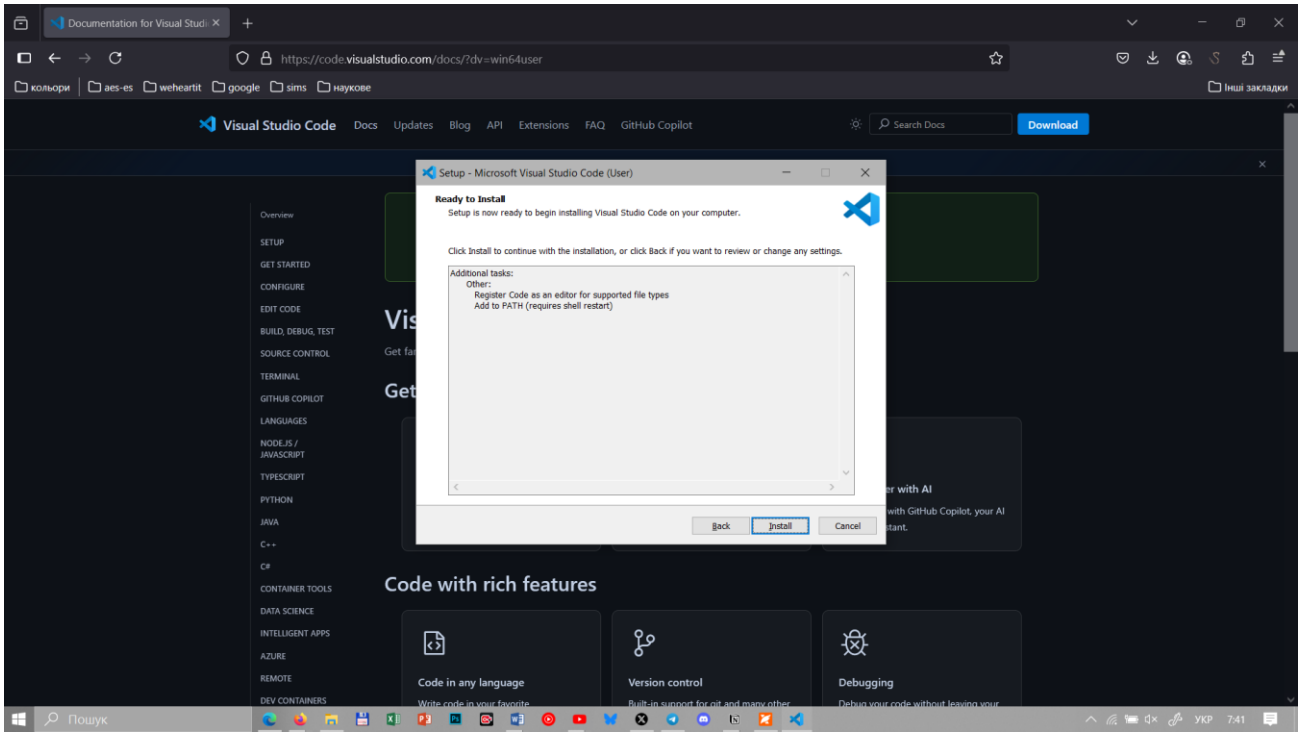


Рисунок 3.1 — інсталяція VS Code

3.2 Інструкція зі встановлення Node.js

Задля встановлення Node.js, необхідно перейти на їхній офіційний сайт — https://nodejs.org/en, та обрати «Download Node.js(LTS)». LTS відповідає за рекомендовану стабільну версію. Станом на травень-червень 2025 року це 22 версія Node.js.

Вигляд офіційного сайту зображено на рисунку 3.2.

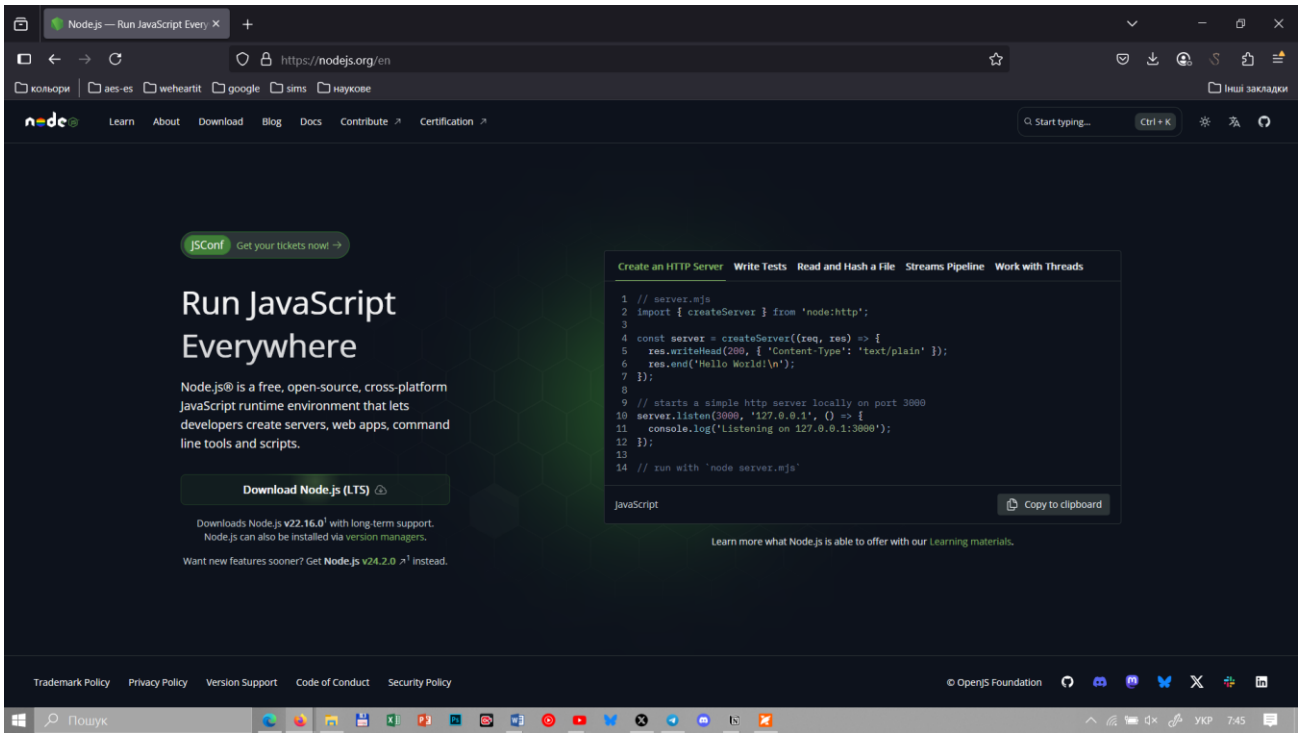


Рисунок 3.2 — вигляд офіційного сайту Node

Ознайомлюємось з ліцензією, після чого необхідно обрати місце розташування. За замовчуванням пропонує на диску С створення нової папки. Наступне вікно пропонує встановлення різних компонентів, наприклад, як npm package manager.

Після встановлення з’явиться консольне вікно, на якому виводиться встановлена версія (див. рис. 3.3).

Задля встановлення express, для серверного проєкту необхідно у консолі прописати npm install express mysql2.

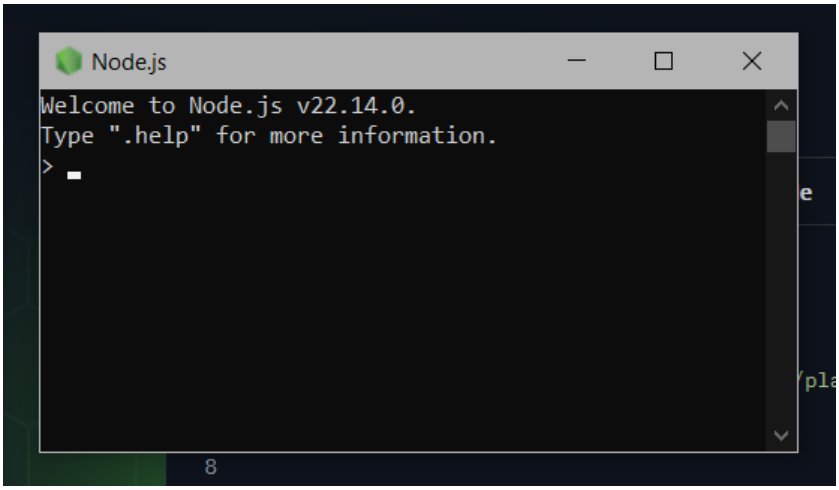


Рисунок 3.3 — вигляд платформи Node.js

Задля запуску серверної частини програми необхідно у терміналі до бекенду прописати наступне:

node server.js

3.3 Інструкція зі встановлення DBeaver та бази даних

Задля встановлення DBeaver, необхідно перейти на їхній офіційний сайт — https://dbeaver.io/, та обрати версію DBeaver Community, яка є безкоштовною у використанні. Опісля цього, необхідно на сайті обрати для якої системи встановити. У даному випадку обираємо Windows(installer), як на рисунку 3.4.

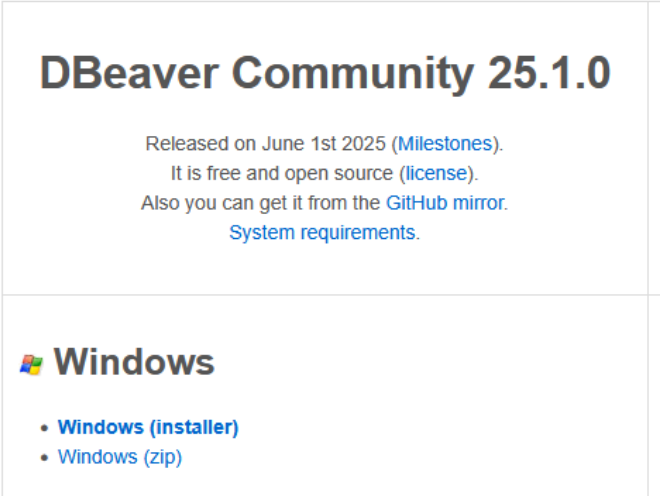


Рисунок 3.4 — вигляд вибору встановлення DBeaver для Microsoft Windows

Опісля завантаження інсталятора, необхідно його відкрити та пройти наступні кроки, а саме знайомство з ліцензією; вибір додаткових компонентів, вибір місце збереження програми та її даних.

Задля запуску бази даних, необхідно у навігаційній панелі, що знаходиться зверху, обрати «Створення нової бази даних», завдяки якому з’явиться перелік запропонованих, як на рисунку 3.5.

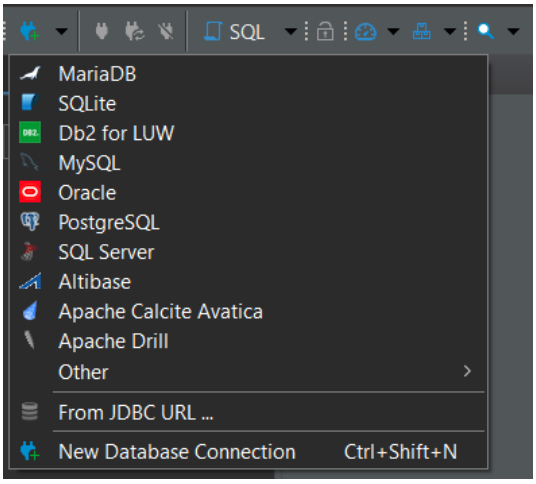


Рисунок 3.5 — перелік запропонованих баз даних

Вигляд вікна для налаштування MariaDB зображено на рисунку 3.6.

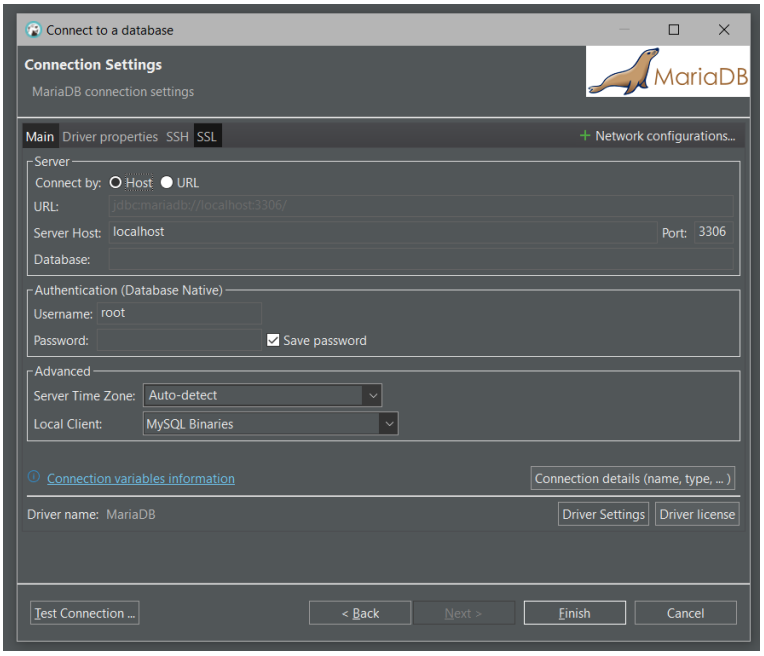


Рисунок 3.6 — вигляд вікна задля налаштування СУБД MariaDB

3.4 Інструкція зі встановлення інших програмних засобів

Для роботи з React, необхідно встановити вбудований набір інструментів. Задля цього у консолі Visual studio Code відносно нашої папки необхідно прописати команду для встановлення. Дана команда запускає останню версію React, ініціалізує проєкт та встановлює необхідні залежності.

Задля встановлення react-router-dom необхідно прописати у консолі наступне(див. рис. 3.7):

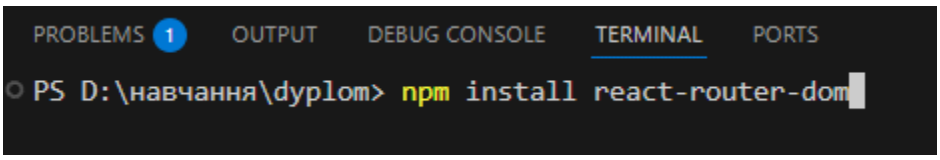


Рисунок 3.7 — встановлення react-router-dom

Опісля цього створиться для обраної папки структура, яка складатиметься з декількох папок. У папці public знаходиться index.html, який виконує основну роль реалізації проєкту як односторінковий вебдодаток, до якого підключаються усі компоненти React. Після неї іде папка src, в якій знаходиться логіка програми.

Задля перевірки роботи необхідно у терміналі прописати наступне(див. рис. 3.8):

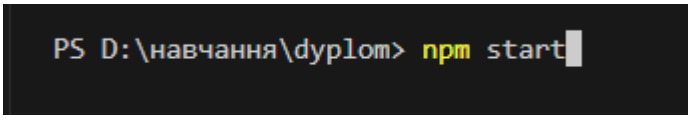


Рисунок 3.8 — запуск програми

3.5 Інструкція з експлуатації вебсайту через адміністративну панель

Задля того, аби увійти у профіль (див. рис. 3.9) необхідно увести у логін admin та пароль як admin_newschool. Внизу поставити галочку у пункті «увійти як адміністратор».

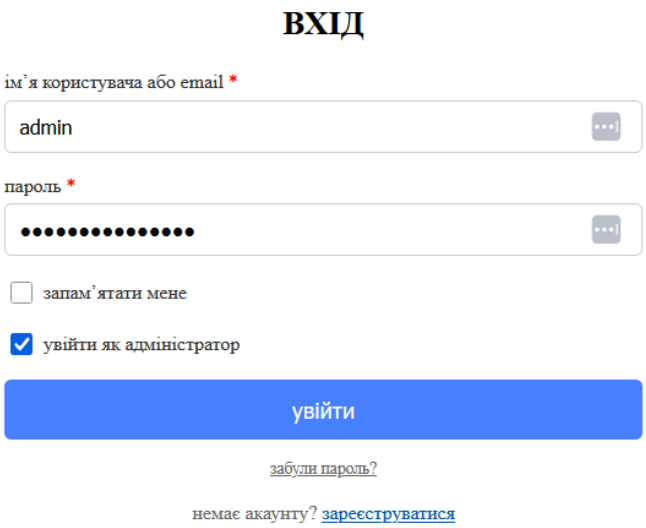


Рисунок 3.9 — вигляд входу під адміністратором

Перед вами відкриється адміністративна панель. За замовчуванням одразу виводиться каталог наявного товару у вигляді таблиці. Ліворуч знаходяться такі категорії як «товари» — каталог, та «замовлення», де можна переглянути зроблені людьми раніше замовлення та їхній статус.

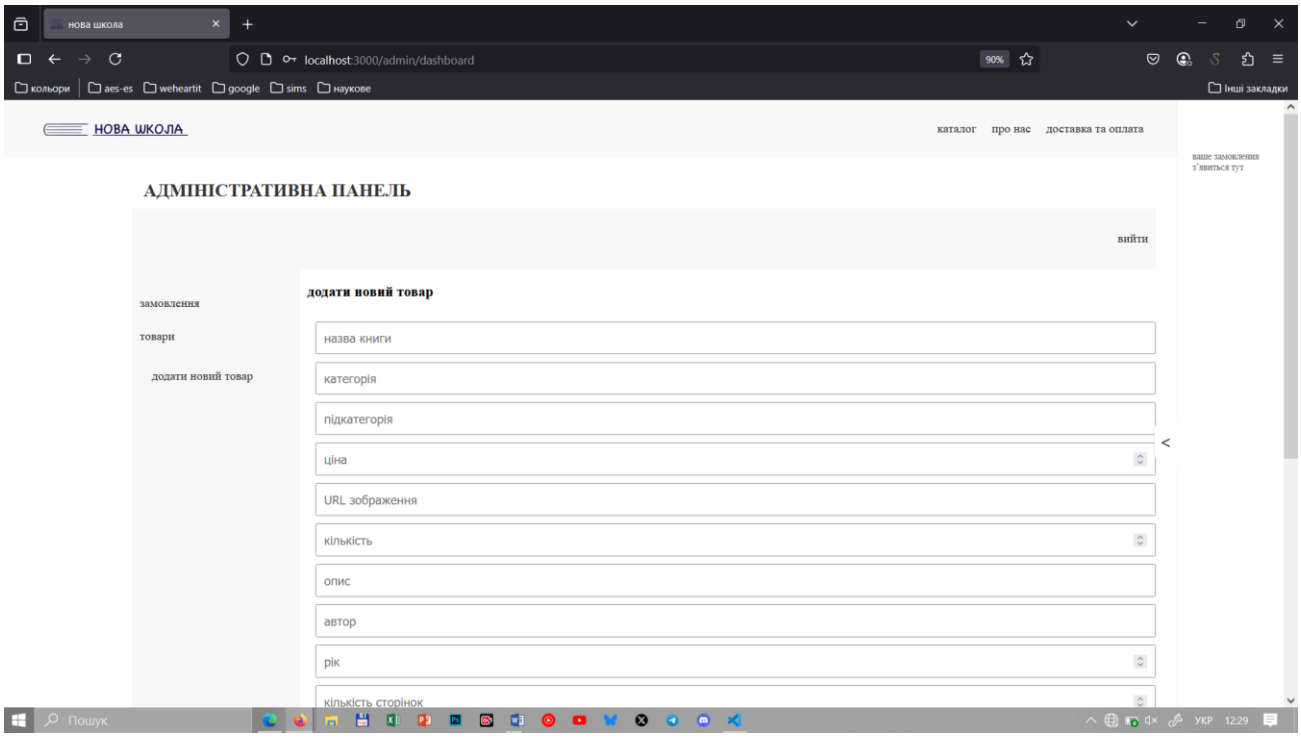


Рисунок3.10 — вигляд форми для додавання нового товару

Нижче категорії «товари» знаходиться текстова кнопка «додати новий товар». При її використанні з’являється форма, яку необхідно заповнити. З виглядом форми можна ознайомитись на рисунку 3.10.

Опісля внесення всіх даних необхідно натиснути внизу кнопку «додати товар». Після цього сторінка оновиться і внизу переліку з’явиться нова позиція.

У другій вкладці виводиться перелік замовлень, де можна редагувати їхній статус. Виводиться ім’я користувача, його електронна пошта, сума за замовлення та його статус. За замовчуванням у замовлення стоїть статус «у процесі». Повна інформація замовлення на даний момент реалізована у базі даних, яка розбита на дві частини.

Вигляд даної панелі реалізовано на рисунку 3.11.

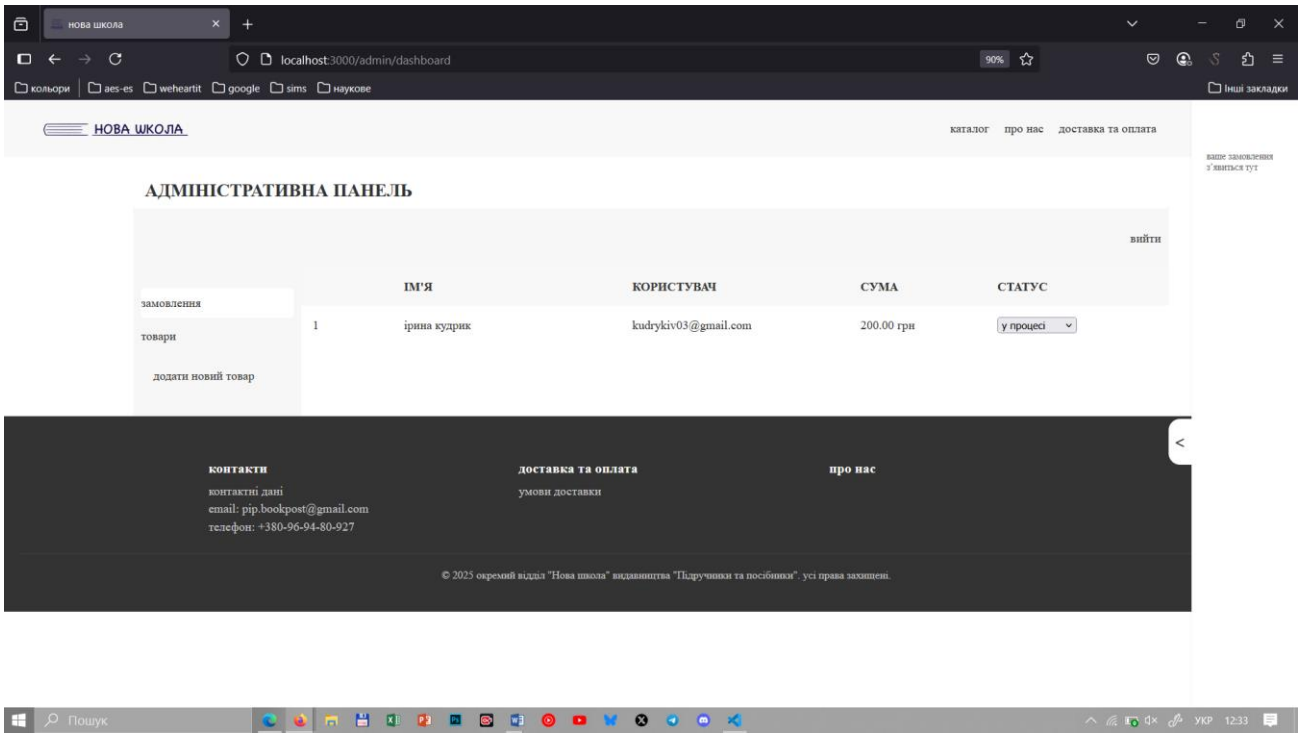


Рисунок 3.11 — вигляд панелі зі замовленнями

В базі даних можна ознайомитись зі замовленнями більш детально. Вони реалізовані через дві взаємопов’язані таблиці, а саме: таблиця «orders», яка відповідає за доставку та оплату самого замовлення з усіма необхідними контактними та адресними данами користувача; таблиця «order_items», у якій

розміщено перелік товарів, що входять у те чи інше замовлення.

На рисунку 3.12 можна ознайомитись з виглядом таблиці «orders», де ми бачимо умови доставки та контактні дані покупця.

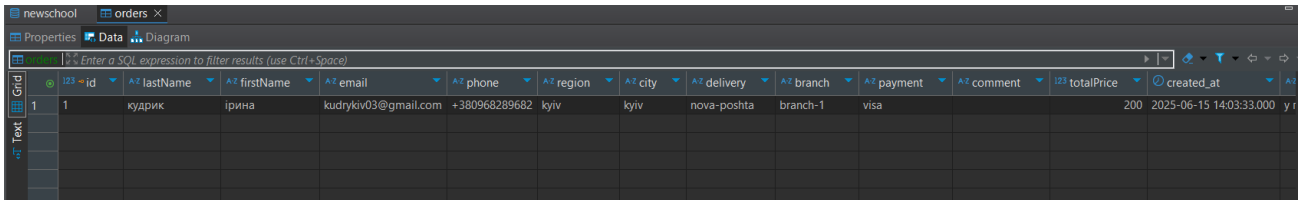


Рисунок 3.12 — вигляд таблиці «orders»

На рисунку 3.13 зображено перелік товарів, на основі яких і було оформлено те чи інше замовлення.

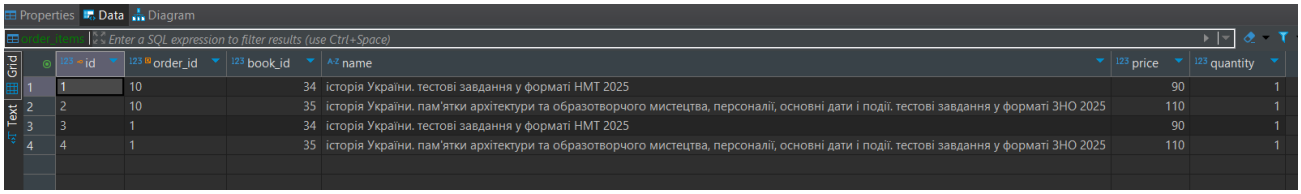


Рисунок 3.13 — вигляд товарів, на які було оформлено замовлення

4 ЕКОНОМІЧНИЙ РОЗДІЛ

Метою економічної частини кваліфікаційної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності розробки вебсайту інтернет-магазину окремого відділу «Нова школа» видавництва «Підручники і посібники», обчислення собівартості та повної ціни проєкту.

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

Для визначення загальної тривалості проведення НДР доцільно дані витрат часу по окремих операціях технологічного процесу зведено у таблиці 1.

Таблиця 4.1 — Середній час виконання НДР та стадії технологічного процесу

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1	видача завдання та постановка задачі	керівник проєкту	5 година
2	аналіз задачі, складання технічного завдання	розробник	7 годин
3	планування	розробник	12 годин
4	проєктування	розробник	9 годин
5	розробка (програмування)	розробник	150 годин
6	розгортання	розробник	12 годин
7	тестування	розробник	8 годин
8	затвердження здачі проєкту	керівник проєкту	2 години
Разом		–	205 годин

Звідси, на розробку проєкту та виконання завдання для розробки вебсайту інтернет-магазину окремого відділу «Нова школа» видавництва «Підручники і посібники» складає 205 годин.

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

Відповідно до Закону України «Про оплату праці» заробітна плата — це «винагорода, обчислена, як правило, у грошовому виразі, яку власник або уповноважений ним орган виплачує працівникові за виконану ним роботу».

Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та господарської діяльності підприємства. Заробітна плата складається з основної та додаткової оплати праці.

Основна заробітна плата нараховується на виконану роботу за тарифними ставками, відрядними розцінками чи посадовими окладами і не залежить від результатів господарської діяльності підприємства.

Додаткова заробітна плата — це складова заробітної плати працівників, до якої включають витрати на оплату праці, не пов’язані з виплатами за фактично відпрацьований час. Нараховують додаткову заробітну плату залежно від досягнутих і запланованих показників, умов виробництва, кваліфікації виконавців. Місячний оклад кожного працівника слід враховувати згідно існуючих на даний час тарифних окладів.

Основна заробітна плата розраховується за формулою:

$$З_{осн.} = Т_с \cdot К_г$$

де $Т_с$ — тарифна ставка, грн;

$К_г$ — кількість відпрацьованих годин.

Звідси, для керівника проєкту $З_{осн.} = 300 \cdot 7 = 2100$ гривень,

для розробника — $З_{осн.} = 50 \cdot 298 = 9900$ гривень.

Додаткова заробітна плата становить 10–15% від суми основної заробітної

плати.

$$З_{дод.} = З_{осн.} \cdot К_{допл.}$$

де, $К_{допл.}$ — коефіцієнт додаткових виплат працівникам 0,1 — 0,15.

Для керівника проєкту $З_{дод} = 2100 \cdot 0,13 = 273$ гривні,

для розробника — $З_{дод} = 9900 \cdot 0,1 = 990$ гривень.

Звідси загальні витрати на оплату праці ($В_{о.п.}$) визначаються за формулою:

$$В_{о.п.} = З_{осн.} + З_{дод.}$$

Для керівника — 2100+273=2373 гривні,

для розробника —9900+990=10890 гривень.

Крім того, слід визначити суму нарахування на заробітну плату:

єдиний соціальний внесок — 22%;

Отже, сума нарахувань на заробітну плату буде становити:

$$В_{с.з.} = ФОП \cdot 0,22$$

де, ФОП — фонд оплати праці, грн.

$ФОП = \Sigma З_{осн.}$, звідси ФОП = 2100+9900=12000 гривень.

Тим самим, для проєкту $В_{с.з.}$ становитиме 12000·0,22=2640 гривень.

Проведені розрахунки витрат на оплату праці зведено у таблиці 4.2.

Таблиця 4.2 — Зведені розрахунки витрат на оплату праці

№ п/п	Категорія працівників	Основна заробітна плата, грн.			Додатк зароб. плата, грн.	Нарах. на ФОП, грн.	Всього витрати на оплату праці, грн.
		Тариф. ставка, грн.	К-сть відпрацьов. год.	Фактично нарах. з/пл., грн.			
1	керівник	300	7	2100	273	-	-
2	розробник	50	198	9900	990	-	-
Разом		350	205	12000	1263	2640	15927,2

4.3 Розрахунок витрат на електроенергію

Затрати на електроенергію 1-ці обладнання визначаються за формулою:

$$З_е = W \cdot T \cdot S$$

де, W — необхідна потужність, кВт;

T — кількість годин роботи обладнання;

S — вартість кіловат-години електроенергії.

Для наявного ноутбука у розробника ASUS ROG Strix G15 G512LU W = 0,2 кВт, T = 170 год.

Вартість кіловат-години електроенергії слід приймати згідно існуючих на даний час тарифів (7 грн).

Звідси,

$$З_е = 0{,}2 \cdot 170 \cdot 7 = 238 \text{ грн}$$

4.4 Розрахунок суми амортизаційних відрахувань

Характерною особливістю застосування основних фондів у процесі виробництва є їх відновлення. Для відновлення засобів праці у натуральному виразі необхідне їх відшкодування у вартісній формі, яке здійснюється шляхом амортизації.

Амортизація — це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення.

Комп’ютери та оргтехніка належать до четвертої групи основних фондів. Для наявного ноутбука у розробника ASUS ROG Strix G15 G512LU на момент купівлі у 2021 році коштував 38000 грн. Оскільки дана сума перевищує 2500 грн — нараховується амортизація. Мінімально допустимі строки корисного використання 2 роки, звідси норма амортизації становить 20%.

Для визначення амортизаційних відрахувань застосовуємо формулу:

$$А = \frac{Б_в \cdot Н_а}{100\%}$$

де А — амортизаційні відрахування за звітний період, грн.;

$Б_в$ — балансова вартість групи основних фондів на початок звітного періоду, грн.;

$Н_а$ — норма амортизації, %.

$$А = \frac{38000 \cdot 0{,}04}{150} \cdot 170 = 1722{,}7$$

Отже, амортизаційні відрахування за рік становитимуть 7600 грн за умови балансової вартості 38000 грн і норми амортизації 20% на рік.

4.5 Обчислення накладних витрат

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 20-60% від суми основної та додаткової заробітної плати працівників.

$$Н_в = В_{о.п} \cdot 0{,}2 \ldots 0{,}6$$

де, $Н_в$ — накладні витрати.

Звідси, $Н_в = 12000 \cdot 0{,}4 = 4800$ гривень.

4.6 Складання кошторису витрат та визначення собівартості НДР

Результати проведених вище розрахунків зведено у таблиці 4.3.

Таблиця 4.3 — Кошторис витрат на НДР

Зміст витрат	Сума, грн.	В % до загальної суми
Витрати на оплату праці (основну і додаткову заробітну плату)	12000,00	50,76
Відрахування на соціальні заходи	2640	8,41
Витрати на електроенергію	238	0,18
Амортизаційні відрахування	1722,7	24,22
Накладні витрати	4800	15,3
Собівартість	21400,7	100

Собівартість($С_в$) НДР розрахуємо за формулою:

$$С_в = В_{о.п.}+В_{с.з.}+З_{м.в.}+З_е+А+Н_в$$

Звідси, $С_в$=21400,7 грн.

4.7 Розрахунок ціни НДР

Ціну НДР можна визначити за формулою:

$$Ц = С_в \cdot (1+Р_{рен}) \cdot (1+ПДВ)$$

де, $Р_{рен.}$ — рівень рентабельності;

$В_{і.н.}$ — вартість носія інформації, грн.;

ПДВ — ставка податку на додану вартість, (20 %).

Звідси, ціна НДР становитиме:

$$Ц = 21400,7 \cdot (1+0,3) \cdot (1+0,2) = 33385,00 \text{ гривні.}$$

4.8 Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва — це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Для визначення ефективності продукту розраховують чисту теперішню вартість (ЧТВ) і термін окупності ($Т_{ОК}$).

$$ЧТВ = -К_в + \sum_{i=1}^{t} \frac{Г_п}{(1+і)^t}$$

де $К_В$ — затрати на проєкт;

$Г_п$ — грошовий потік за t-ий рік;

t — відповідний рік проєкту;

i — величина дисконтної ставки (10…15%).

Якщо ЧТВ ≥ 0, то проєкт може бути рекомендований до впровадження.

$$ЧТВ = -21400,7 + \frac{11984,4}{(1+0,1)} + \frac{11984,4}{(1+0,1)^2} + \frac{11984,4}{(1+0,1)^3} = 8402,7$$

Термін окупності визначається за формулою:

$$Т_{ок} = Т_{пв} + \frac{Н_в}{Г_{пр}}$$

де $Т_{пв}$ — період до повного відшкодування витрат, років;

$Н_в$ — невідшкодовані витрати на початок року, грн;

$Г_{пр}$ — грошовий потік на початок року, грн.

$$Т_{ок} = 2 + \frac{601,4}{11984,4} = 2,1$$

Всі дані розрахунків внесено у зведену таблицю 4.4 техніко-економічних показників.

Таблиця 4.4 — Техніко-економічні показники НДР

№ п/п	Показник	Значення
1.	Собівартість, грн.	21400,7
2.	Плановий прибуток, грн.	11984,4
3.	Ціна, грн.	33385,00
4.	Чиста теперішня вартість, грн.	8402,7
5.	Термін окупності, рік	2,1

Собівартість вебсайту інтернет-магазину окремого відділення «Нова школа» видавництва «Підручники і посібники» становить 21400,7 гривень, а загальна його вартість — 33385,00 гривень. Термін окупності проєкту — 2,1 рік.

Проєкт є економічно доцільним для реалізації в рамках НДР завдяки позитивній ЧТВ і короткому терміну окупності та може бути використаний за основу для подальшого розвитку комерційного проєкту.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

5.1 Статична електрика та захист від неї

Статична електрика виникає внаслідок тертя двох діелектриків або діелектрика і провідника. Під час тертя двох об’єктів на одному, з вищими діелектричними властивостями, накопичується позитивний заряд, а на іншому, з нижчими діелектричними характеристиками, — негативний. Таке явище виникає при терті твердих діелектриків, до яких можна віднести пластмасу, вовняну тканину, папір тощо. До рідких діалектриків, при терті яких також може виникати статична електрика, належать нафта, спирт тощо, а до газоподібних — сухе повітря, різні газоподібні суміші та інші.

Заряди статичної електрики мають властивості накопичуватися на окремих об’єктах, а саме на поверхні обладнання, матеріалі, одязі працівників та інших предметах. Нагромадження заряду призводить до іскрових розрядів, які є пожежо- і вибухонебезпечними. Іскра від потенціалу на тілі людини може досягати 2,5—7,9 мДж, що достатньо для спалаху багатьох речовин, таких як парів ацетону, метану, оксиду вуглецю тощо.

Іскрові розряди статичної електрики відчуваються як поколювання або незначний поштовх, що самі по собі не становлять небезпеки для людини завдяки малій силі струму. Проте, враховуючи елемент несподіванки такого розряду — людина може перелякатись, що може стати причиною нещасного випадку.

Тривала дія статичної електрики на організм людини може порушувати фізіологічні процеси, викликати розлади центральної нервової системи та серцево-судинної системи. Гранично допустима напруженість електричного поля на робочих місцях не повинна перевищувати 60 кВ/м за умови впливу тривалістю до 1 години.

Згідно з Правилами захисту від статичної електрики електростатична безпека вважається задовільною, якщо максимальна енергія зарядів не перевищує 40 % мінімальної енергії спалаху речовини.

Основні методи захисту від статичної електрики включають відведення зарядів у землю, запобігання накопиченню зарядів та їх нейтралізацію. Для зменшення статичних зарядів застосовують: підвищення вологості повітря до 70%, нанесення електропровідних плівок на діелектричні поверхні. Графіт, сажа, металевий порошок, що додаються до виготовлення гуми, знижують заряди статичної електрики. В окремих випадках статичну електрику нейтралізують за допомогою спеціальних приладів — іонізаторів.

5.2 Недостатня, або надмірна освітленість робочого місця

Освітлення є найважливішим показником гігієни праці та невід'ємною частиною наукової організації. Раціональне освітлення виробничих приміщень справляє позитивний психофізичний вплив на робітників, сприяє підвищенню продуктивності праці, гарантує її безпеку та збереженню високої працездатності. Неправильно підібране освітлення погіршує умови зорової роботи, підвищує втому очей, нервової системи, може стати причиною нещасного випадку або захворювання. До 5% травм обумовлене недостатнім чи нераціональним освітленням.

Надмірна та/або недостатня освітленість робочого місця є одними зі шкідливих та небезпечних виробничих факторів. У зв'язку з цим, до освітлення робочого місця ставляться цілком визначені вимоги.

Гігієнічні вимоги, засновані на психофізичних особливостях сприйняття світла і його впливу на організм людини, зводяться до наступного: Рівень освітленості повинен відповідати гігієнічним нормам, які враховують умови здорової праці. Повинна бути забезпечена рівномірність і стійкість рівня освітленості в приміщені, щоб уникнути частої переадаптації і стомлення зору. Спектральний склад світла штучних джерел повинен наближатися до сонячного. Освітленість не повинна створювати блискучості як від самих джерел світла, так і в зоні праці.

Вимоги до освітленості залежать від виду зорової роботи, який

визначається розміром об’єкта, що розглядається. Надалі ми розглядатимемо робочі місця, які оснащені персональними комп’ютерами.

Важливою умовою безпеки людини, що перебуває перед екраном, є правильний вибір візуальних параметрів дисплея та світлотехнічних умов робочого місця. Робота з дисплеями при неправильному виборі яскравості й освітленості екрана, контрастності знаків, їх кольорів, за наявності відблисків на екрані, тремтіння та мерехтіння зображення призводить до зорового стомлення, головного болю, значного психофізіологічного навантаження, погіршення зору.

Залежно від джерел світла освітлення може бути природним, що створюється прямими сонячними променями та розсіяним світлом небосхилу; штучним що створюється електричними джерелами світла та суміщеним, при якому недостатнє за нормами природне освітлення доповнюється штучним.

Природне світло повинно бути бічним(одно або двохстороннє), що здійснюється через вікна у зовнішніх стінах; зорієнтованим, як правило, на північ чи північний схід при коефіцієнті не нижче 1,5%. Вікна приміщень повинні мати регулювальні пристрої для відчинення, а також жалюзі, штори тощо. При виробничій потребі дозволяється експлуатувати ЕОМ у приміщеннях без природного освітлення за узгодженням з органами Держпромгірнагляду та органами й установами санітарно епідеміологічної служби.

Штучне освітлення приміщення з робочими місцями, обладнаними персональними комп’ютерами, має бути всеосяжним і рівномірним. Загальне освітлення передбачає розміщення світильників у верхній зоні приміщення. Місцеве утворюється світильниками, що концентрують світловий потік безпосередньо на робочих місцях. Вони розміщуються збоку (переважно ліворуч), або локально над робочим місцем. При штучному освітленні за джерело світла застосовують, зазвичай, люмінесцентні лампи. Рівень освітленості на робочому місці має становити 300-500 лк. При використанні комбінованого освітлення не допускається відблисків на поверхні екрана та збільшення освітлення екрана вище 300 лк.

5.3 Характеристика скарг працівників та посадових осіб, які працюють більше половини робочого дня за комп'ютером

Працівники та посадові особи, які проводять понад половину робочого часу (більше 4 годин на день) за комп'ютером, часто повідомляють про низку скарг, пов'язаних із тривалим перебуванням у статичній позі, інтенсивним зоровим навантаженням та впливом умов робочого середовища. Основні скарги включають: синдром «сухого ока» — сухість, подразнення та/або почервоніння очей; погіршення гостроти зору, розмитість зображення, відчуття втоми в очах переважно наприкінці робочого дня, головний біль, а також біль та/або скутість у шиї, плечах, спині та попереку через тривале перебування в одній позі; ниючий біль у зап'ястях; підвищена втома і зниження концентрації уваги; дратівливість, тривожність та часте погіршення настрою, погіршення сну та нестача енергії у позаробочий час.

Вищезазначені скарги зумовлені тривалим навантаженням завдяки великій кількості сидячої роботи, малорухливості, недостатнього відпочинку, а також завдяки впливу електромагнітного випромінювання. Основними факторами ризику є проблеми з освітленням робочого місця, некоректне розташування монітора, відсутність регулярних перерв для відпочинку та фізичних вправ.

Особливість роботи осіб, що працюють більше половини дня за комп'ютером є задіяння малої кількості м'язів та збільшене навантаження на центральну нервову систему. До факторів, що характеризують напруженість праці, відносяться: інтелектуальна напруга — напруга, викликана частим зверненням до інтелектуальних процесів у зв'язку з високою щільністю потоку проблемних ситуацій; ступінь монотонності навантажень — одноманітність робочих дій, їх багаторазове повторення і невелика тривалість; нестабільний режим роботи тощо. При інтенсивній інтелектуальній діяльності потреба мозку в енергії підвищується, складаючи 15 ... 20 % від загального обсягу в організмі, що вказує на сильне навантаження на організм. Тривале розумове навантаження впливає на психічну діяльність, погіршує функції уваги (обсяг, концентрація,

переключення), пам’яті (короткочасної і довгострокової), сприйняття (збільшується частота помилок). Окрім цього, різні дослідження свідчать, що нервове напруження впливає на серцево-судинну систему, збільшуючи артеріальний тиск і частоту пульсу, а також на терморегуляцію організму та емоційні стани працівника.

Однією з найкращих профілактик при тривалій роботі за електронно-обчислювальними машинами перерва на відпочинок та легка розминка тіла на робочому місці та активний відпочинок у позаробочий час. Також у профілактиці стомлення і перевтоми значну роль відіграє організація раціонального режиму праці та відпочинку.

Робота за ЕОМ без регламентованої перерви не повинна перевищувати 2 години. При 8-годинному робочому дні регламентовану перерву необхідно встановлювати через 2 години від початку зміни і після обідньої перерви тривалістю 10-15 хвилин.

Під час регламентовані перерви слід використовувати активний відпочинок – комплекс спеціальних профілактично-реабілітаційних вправ, перебування на свіжому повітрі. За можливості, додатково провітрювати приміщення.

Із метою профілактики і попередження захворювань при роботі з ЕОМ необхідно дотримуватись режиму дня, розумно чередувати працю і відпочинок, уміло використовувати фізичні вправи, фактори природного середовища для підвищення резистентності організму, розширення норми реакції.

ВИСНОВОК

Для даної кваліфікаційної роботи було розроблено вебсайт для інтернет-магазину окремого відділу «Нова школа» видавництва «Підручники і посібники». Даний вебдодаток орієнтований на комерційне використання, де користувачі спроможні переглядати та здійснювати покупку книг.

При розробці інтернет-магазину були проаналізовані сучасні вебтехнології задля розробки односторінкового вебдодатку з динамічним та водночас простим інтерфейсом, розроблені спосіб авторизації користувачем, реалізовано спосіб покупки.

Для взаємодії з користувачем було використано бібліотеку React мови програмування JavaScript, а для серверної частини — фреймворк Express на платформі Node.js., чия материнська мова програмування також є JavaScript, що полегшує процес написання коду для вебдодатку.

За базу даних було обрано реляційну MariaDB завдяки її простоті при експлуатації, а для легшої роботи з нею — графічний інтерфейс DBeaver.

Розробка і впровадження проєкту вебсайту інтернет-магазину оккремого відділу «Нова школа» видавництва «Підручники і посібники» є економічно ефективним згідно з розрахунками, наведеними в економічному розділі.

У майбутньому планується подальше покращення та краща автоматизація проєкту, оскільки код є гнучким та відповідає повному контролю розробника.

ЛІТЕРАТУРА

1. 6 Most Popular Back-end Programming Languages. URL: https://www.deazy.com/knowledge-hub/top-6-back-end-languages (дата звернення: 21.05.2025)
2. An Introduction to Web Development Technologies. URL: https://tillerdigital.com/blog/an-introduction-to-web-development-technologies/ (дата звернення: 21.05.2025)
3. DBeaver Community. URL: https://dbeaver.io (дата звернення: 21.05.2025)
4. Difference between SQL, PL/SQL and T-SQL. URL: https://medium.com/@rajesh_data_ai/difference-between-sql-pl-sql-and-t-sql-3f654c499b1f (дата звернення: 20.05.2025)
5. E-commerce Defined: Types, History and Examples. URL: https://www.investopedia.com/terms/e/ecommerce.asp (дата звернення: 22.05.2025)
6. Express.js Tutorial. URL: https://www.geeksforgeeks.org/node-js/express-js/ (дата звернення: 22.05.2025)
7. Express/Node introduction — Learn web development. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/ Extensions/ Server-side/Express_Nodejs/Introduction (дата звернення: 22.05.2025)
8. Getting started with Angular. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Frameworks_libraries/Angular_getting_started (дата звернення: 21.05.2025)
9. How Node.js Works: A comprehensive Guide in 2025. URL: https://nodesource.com/blog/how-nodejs-works (дата звернення: 22.05.2025)
10. HTML: HyperText Markup Language. URL https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 21.05.2025)
11. Introduction of DBMS (Database Management System). URL: https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/ (дата звернення: 21.05.2025)

12. Is Go The Right Choice for Building a Backend? URL: https://getstream.io/blog/go-backend/ (дата звернення: 22.05.2025)
13. Java vs Python for Backend: Which One is Best in 2024? URL: https://medium.com/@carloshardy9687/java-vs-python-for-backend-which-one-is-best-in-2024-9a0901fe3145 (дата звернення: 22.05.2025)
14. Java's Role in Web and App Development: Powering the Digital Experience. URL: https://medium.com/@KanakSengar/javas-role-in-web-and-app-development-powering-the-digital-experience-5162502dd2a8 (дата звернення: 221.05.2025)
15. JavaScript. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 21.05.2025)
16. Master the Art of Web Design: What are the 9 Best Programming Languages for Computer Programming For Web Designers. URL: https://medium.com/@theurikahuthiat/master-the-art-of-web-design-what-are-the-best-programming-languages-for-computer-programming-for-0bfe3570797e (дата звернення: 21.05.2025)
17. My way to the perfect web development environment. URL: https://areknawo.com/my-way-to-the-perfect-web-development-environment/ (дата звернення: 20.05.2025)
18. Node.js — запускайте JavaScript будь-де. URL: https://nodejs.org/uk (дата звернення: 21.05.2025)
19. Node.js vs PHP: Choosing the Right Backend in 2025. URL: https://webandcrafts.com/blog/node-js-vs-php (дата звернення: 22.05.2025)
20. Open Source Web Programming: A Guide for Developers. URL: https://medium.com/@Bigscal-Technologies/top-3-open-source-web-programming-languages-that-will-rule-in-2024-fc9a3291e82c (дата звернення: 21.05.2025)
21. Python Backend Development: A Complete Guide for Beginners. URL: https://www.datacamp.com/tutorial/python-backend-development (дата звернення: 21.05.2025)

22. React. URL: https://react.dev/ (дата звернення: 21.05.2025)
23. React — JavaScript-бібліотека для створення користувацьких інтерфейсів. URL: https://uk.legacy.reactjs.org/ (дата звернення: 21.05.2025)
24. The 5 Best Backend Development Languages to Master. URL: https://roadmap.sh/backend/languages (дата звернення: 21.05.2025)
25. The top 4 Python backend frameworks for building entry level AI projects. URL: https://pieces.app/blog/the-top-4-python-back-end-frameworks-for-your-next-project (дата звернення: 21.05.2025)
26. Top 10 Go Backend Frameworks. URL: https://medium.com/codex/top-10-go-backend-frameworks-88546ba8f02d (дата звернення: 22.05.2025)
27. Top 11 Backend Programming Languages in 2025/ URL: https://webandcrafts.com/blog/backend-languages (дата звернення: 21.05.2025)
28. TypeScript vs Go: Choosing Your Backend Language. URL: https://dev.to/encore/typescript-vs-go-choosing-your-backend-language-2bc5 (дата звернення: 22.05.2025)
29. What are the benefits of using Go for building websites and backend development. URL: https://www.quora.com/What-are-the-benefits-of-using-Go-for-building-websites-and-backend-development (дата звернення: 22.05.2025)
30. What is Ecommerce? Definition, Types, Advantages and Disadvantages. URL: https://sell.amazon.com/learn/what-is-ecommerce (дата звернення: 22.05.2025)
31. What software do I need to build a website? — Learn web development. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Tools_and_setup/What_software_do_I_need (дата звернення: 19.05.2025)
32. База даних: типи, особливості та їхні відмінності. URL: https://dou.ua/lenta/articles/types-of-databases/ (дата звернення: 20.05.2025)
33. Видавництво «Підручники і посібники». URL: https://pp-books.com.ua/
34. Все, що потрібно знати про бази даних для початківців: MySQL, PostreSQL, MongoDB. URL: https://dan-it.com.ua/uk/blog/vse-shho-potribno-znati-pro-bazi-danih-dlja-pochatkivciv-mysql-postgresql-mongodb/ (дата звернення: 20.05.2025)
35. Реляційна база даних — UA5. URL https://ua5.org/database/189-reljacjjna-

baza-danikh.html (дата звернення: 20.05.2025)

36. Розбираємось з реляційними та нереляційними базами даних. URL https://dou.ua/forums/topic/49585/ (дата звернення: 20.05.2025)
37. Створення веб-додатків на EXPRESS.JS: швидко та під ключ. URL: https://brander.ua/technologies/expressjs (дата звернення: 22.05.2025)
38. Сучасний підручник з JavaScript. URL: https://uk.javascript.info/ (дата звернення: 21.05.2025)
39. Технічні вимоги для розробки сайту громадської організації. URL: https://eef.org.ua/wp-content/uploads/2024/03/Tehnichni-vymogy-do-vebsajtu.pdf (дата звернення: 20.05.2025)
40. Що таке Node.js: визначення, особливості та застосування. URL: https://wezom.com.ua/ua/blog/vse-chto-nuzhno-znat-o-nodejs (дата звернення: 22.05.2025)
41. Що таке React JS? Як почати вивчати Реакт? URL: https://brainlab.com.ua/uk/blog-uk/shho-take-react-js-yak-pochaty-vyvchaty-reakt (дата звернення: 21.05.2025)