МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА САЙТУ ЕЛЕКТРОННОЇ КОМЕРЦІЇ З ЗАСТОСУВАННЯМ MYSQL, EXPRESS ТА REACT»**

Виконав: студент 4 курсу, групи 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Є.І. Кривохлябов
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н., Мильцев О.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри комп'ютерних наук, д.т.н., доцент, Шило Г.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кривохлябову Єгору Івановичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка сайту електронної комерції з застосуванням MySQL, Express та React

керівник роботи Мильцев Олександр Михайлович, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Аналіз вимог та розробка сайту.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою докладу

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.01.2024	
2.	Збір вихідних даних.	05.02.2024	
3.	Обробка методичних та теоретичних	23.02.2024	
	джерел.		
4.	Розробка першого та другого розділу.	29.03.2024	
5.	Розробка третього розділу.	17.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	17.06.2024	

Студент _____ Є.І. Кривохлябов
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.М. Мильцев
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка сайту електронної комерції з застосуванням MySQL, Express та React»: 49 с., 37 рис., 5 табл., 10 джерел.

API, EXPRESS, JEST, KNEX, MYSQL, OBJECTION, REACT, TYPESCRIPT.

Об’єкт дослідження – процес розробки продукту.

Предмет дослідження – процес розробки комерційного продукту в контексті його етапів та виявлення потенційних труднощів, що можуть виникнути під час цього процесу.

Мета роботи – провести аналіз етапів розробки комерційного продукту та виявити потенційні труднощі, що можуть виникнути під час цього процесу.

Метод дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

Результати та їх новизна: досліджено етапи розробки комерційного продукту, виявлено ключові проблеми та застосовано нові методи для збору та аналізу вимог до програмного забезпечення. Ці зміни спрямовані на покращення управління процесом розробки та підвищення ефективності програмного забезпечення.

Взаємозв’язок з іншими роботами: ця робота базується на дослідженні процесу розробки комерційного продукту та виявлені труднощі, що виникають під час цього процесу.

Значення та висновки: дослідження виявило основні етапи розробки комерційного продукту та потенційні труднощі на них. Це корисно для практичних діяльностей у програмній розробці, дозволяючи розробляти стратегії розв’язання проблем на ранніх стадіях. Робота спрямована на покращення ефективності та якості розробки ПЗ, що важливо для IT-індустрії.

SUMMARY

Bachelor’s qualifying paper «E-commerce Website Development Using MySQL, Express and React»: 49 pages, 37 figures, 5 tables, 10 references.

API, EXPRESS, JEST, KNEX, MYSQL, OBJECTION, REACT, TYPESCRIPT.

The object of the study is the product development process.

The subject of the study is the process of developing a commercial product in the context of its stages and identifying potential difficulties that may arise during this process.

The aim of the study is to analyze the stages of commercial product development and identify potential difficulties that may arise during this process.

The methods of research are collecting and analyzing software requirements, modeling methods, design, construction, and software testing.

Results and novelty: the stages of commercial product development are investigated, key problems are identified, and new methods for collecting and analyzing software requirements are applied. These changes are aimed at improving the management of the development process and increasing the efficiency of software.

Relationship to other works: this work is based on a study of the commercial product development process and the difficulties encountered during this process.

Significance and conclusions: the study has identified the main stages of commercial product development and potential difficulties at them. This is useful for practical activities in software development, allowing to develop strategies for solving issues at early stages. The work is aimed at improving the efficiency and quality of software development, which is important for the IT industry.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 7
1 Огляд вимог та планування проєкту 9
1.1 Аналіз вимог замовника 9
1.2 Висновки до розділу 1 14
2 Проєктування та реалізація бази даних 15
2.1 Концептуальна модель бази даних..... 15
2.2 Практична реалізація бази даних 17
2.3 Висновки до розділу 2 29
3 Проєктування та реалізація вебзастосунка..... 30
3.1 Реалізація сайту..... 30
3.2 Демонстрація реалізованого інтерфейсу 38
3.3 Висновки до розділу 3 46
Висновки 48
Перелік посилань..... 49

ВСТУП

У сучасному світі розробка програмного забезпечення визнається однією з найбільш динамічних та інноваційних галузей. З кожним роком вимоги до програмних продуктів зростають, а користувачі стають більш вимогливими. В цьому контексті розробка комерційних продуктів набуває особливої ваги, вимагаючи від розробників не лише високого технічного рівня, але й глибокого розуміння вимог ринку та потреб споживачів.

Зростаюча конкуренція та стрімкий розвиток технологій в електронній комерції вимагають від розробників не лише технічної кмітливості, але й здатності гармонійно поєднувати інструменти для досягнення високої якості продукту.

Метою кваліфікаційної роботи є аналіз етапів розробки комерційного продукту. У ролі конкретного прикладу вибрано процес створення e-commerce сайту, оскільки цей вид продукту широко представлений в аутсорсингових компаніях та є актуальним об’єктом для дослідження. Серед ключових етапів розробки виокремлюють наступні:

- аналіз вимог;
- проєктування;
- розробка;
- тестування;
- впровадження;
- підтримка та оновлення.

Об’єктом дослідження є процес розробки продукту e-commerce сайту. Предметом дослідження є ідентифікація ключових етапів та вирішення потенційних труднощів у цьому процесі.

У першому розділі роботи детально розглядається технічне завдання від замовника, а також обгрунтовується вибір технологічного стеку для реалізації проєкту.

У другому розділі проводиться глибокий аналіз предметної області, включаючи створення схеми даних на основі виділених сутностей та ретельний аналіз взаємозв’язків між ними.

У третьому розділі надається вичерпний опис програмної реалізації на кожному етапі проєкту. Здійснюється роз’яснення деталей та обґрунтування вибору конкретних рішень при імплементації. Розглядаються приклади виконання, що ілюструють реалізацію кожного етапу та демонструють ключові аспекти програмного проєкту.

1 ОГЛЯД ВИМОГ ТА ПЛАНУВАННЯ ПРОЄКТУ

1.1 Аналіз вимог замовника

Замовник висловлює бажання створення e-commerce сайту, який буде використовувати технології Express, Eslint [6], Knex [4], MySQL [3], Objection [7], TypeScript [5]. Метою проєкту є створення функціонального інтернет-магазину, що задовольнить вимоги користувачів. У рамках цього проєкту передбачено реалізацію основних сторінок, які включають в себе:

- **логін та реєстрація**: можливість входу та реєстрації користувачів для персонального кабінету;
- **головна сторінка**: перегляд доступних продуктів, реалізація пошуку та фільтрації за різними параметрами, зміна вигляду списку товарів;
- **сторінка продуктів**: опис товару, характеристики товару, динаміка цін на товар, секція для коментарів користувачів;
- **сторінка користувача**: можливість користувача редагувати свій особистий профіль;
- **стрінка улюблених продуктів**: сортування та фільтрація товарів у розділі «Вибране».

У ролі аргументів на користь обраного інструментарію, замовник визначив наступні переваги:

- **TypeScript**: зменшення кількості помилок завдяки статичній типізації, підвищення читабельності та облегшення обслуговування коду, підтримка сучасних стандартів ECMAScript та екосистеми Node.js [1, 10];
- **MySQL**: висока швидкодія та надійність у роботі з об’ємними даними, розширена підтримка операцій та оптимізацій для складних запитів, широкий спектр інструментів та документації для спрощення розробки та адміністрування;

- **Knex**: абстракція бази даних для роботи із SQL безпосередньо в коді, підтримка різних СУБД;
- **Objection**: легке використання та читабельний код при роботі з базою даних, вбудована підтримка відносин між таблицями;
- **Express**: мінімальний та гнучкий фреймворк для швидкої розробки, велика активна спільнота;
- **Eslint**: створення єдиної стилістики коду в проєкті, виявлення та усунення можливих помилок та покращення якості коду.

Отже, після детального вивчення обраного інструментарію можна визначити, що продукт націлений на вдосконалення обробки запитів для комерційного вебсайту, зокрема, у сфері мережевих операцій, таких як взаємодія з базою даних чи обмін повідомленнями через сокети. Варто відзначити, що основний акцент робиться на швидкій та ефективній роботі з мережею, сприяючи оптимізації введення та виведення даних. Не менш важливо відзначити, що обрана інструментальна платформа сприяє зручності для розробників, допомагаючи виявляти та виправляти помилки ще на етапі розробки, що позитивно позначається на якості та ефективності процесу створення продукту.

Замовник також встановлює конкретні вимоги до деяких компонентів системи:

- **валідація пароля при реєстрації**: дотримання визначеної довжини, використання лише латинських символів, зберігання пароля у БД в шифрованому вигляді виключаючи збереження його у сирому форматі;
- **аутентифікація**: використання JWT-токенів для ефективного та безпечного механізму аутентифікації;
- **Role-Based Access Control (RBAC)**: адміністратор, звичайний користувач;
- **головна сторінка**: коли користувач вказує дані у полі для пошуку – демонструвати рекомендації, натиснувши на Breadcrumbs – користувач має перейти до цього розділу, нескінченне прокручування або розбиття на сторінки.

Дозволи для кожної ролі зведені у таблиці 1.1. та візуалізовано у рисунку 1.1.

Таблиця 1.1 – Дозволи для кожної ролі користувача

Дія	Адмін	Користувач	Опис
Переглядати продукти	+	+	Відображати список продуктів
Улюблене CRUD	+	+	Додати до списку вибраного: відображення списку, додати/прибрати товар зі списку, видалити товар зі списку
Переглядати \додавати коментарі	+	+	Відображення коментарів та додавання нових
Видаляти коментарі	+	–	Можливість видаляти коментарі будь-якого користувача
Панель адміна	+	–	Статистика (у вигляді графіків і таблиць): за новими користувачами, за товарами, що найбільш переглядаються, за товарами в кількості «обраних» у користувачів і т.д.
Продукти CRUD	+	–	Окрема сторінка з редагуванням списку всіх товарів: – додавання нового товару; – зміна назви/опису/характеристик/ціни товару і т.д.; – видалення товару.

Продовження таблиці 1.1

Дія	Адмін	Користувач	Опис
Користувачі CRUD	+	–	Окрема сторінка з редагуванням списку всіх користувачів: – зміна ролі користувача; – видалення користувача.

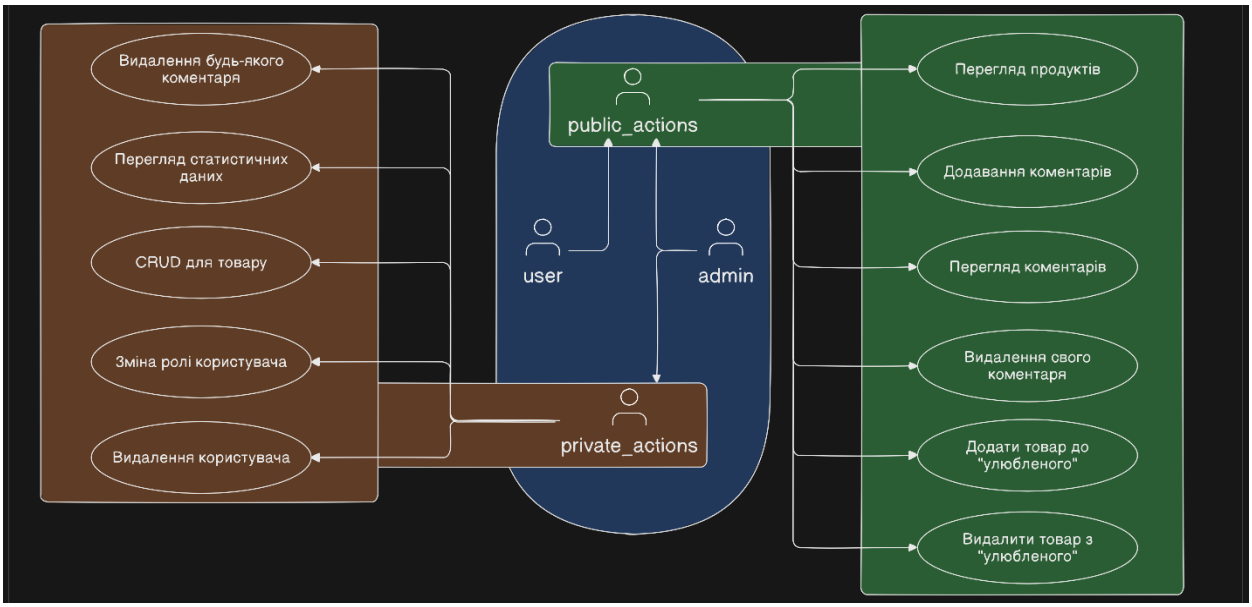

Рисунок 1.1 – Діаграма прецедентів

Для спрощення процесу розробки слід відзначити, що замовник вже володіє готовим дизайном проєкту у Figma (див. рис. 1.2 – 1.5).

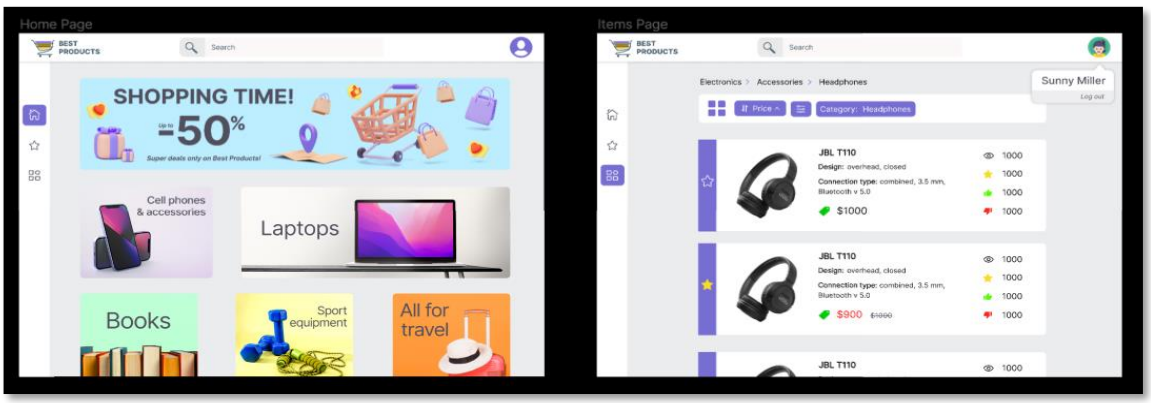

Рисунок 1.2 – Шаблон головної сторінки та списку продуктів

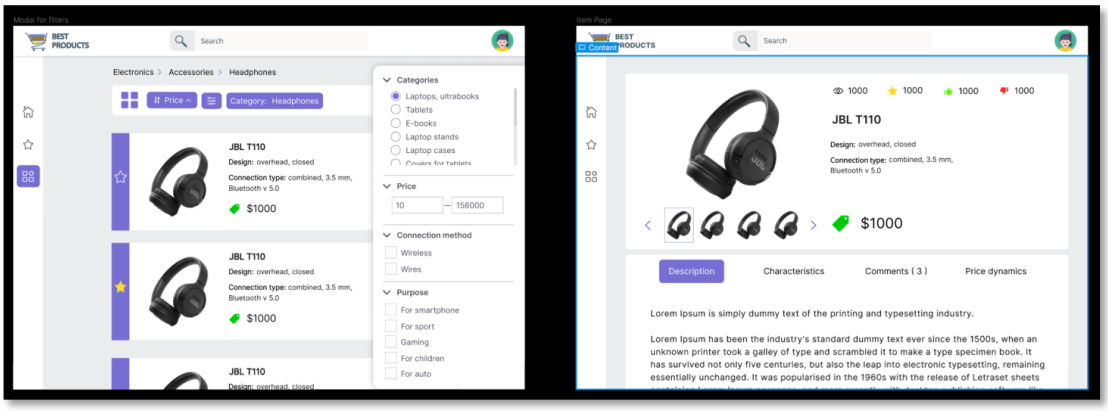

Рисунок 1.3 – Шаблон налаштувань списку продуктів та сторінка продукту

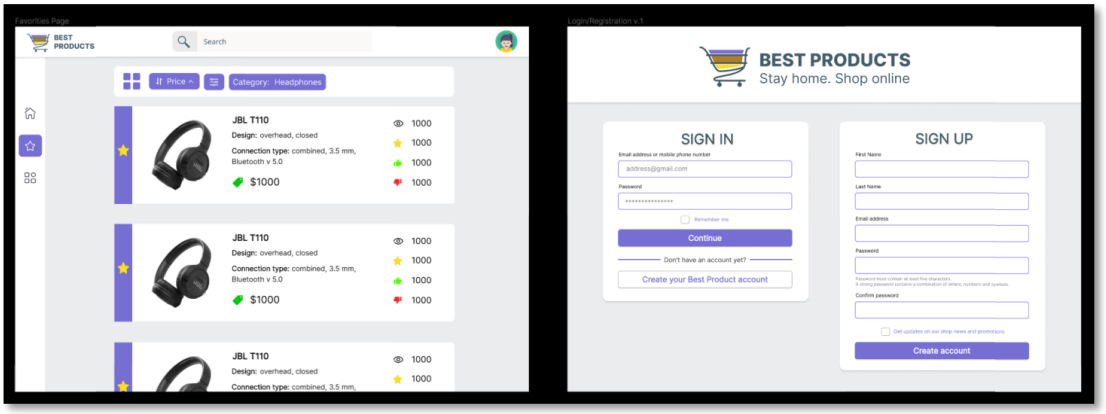

Рисунок 1.4 – Шаблон сторінки «улюблених» продуктів та форми аутентифікації та реєстрації

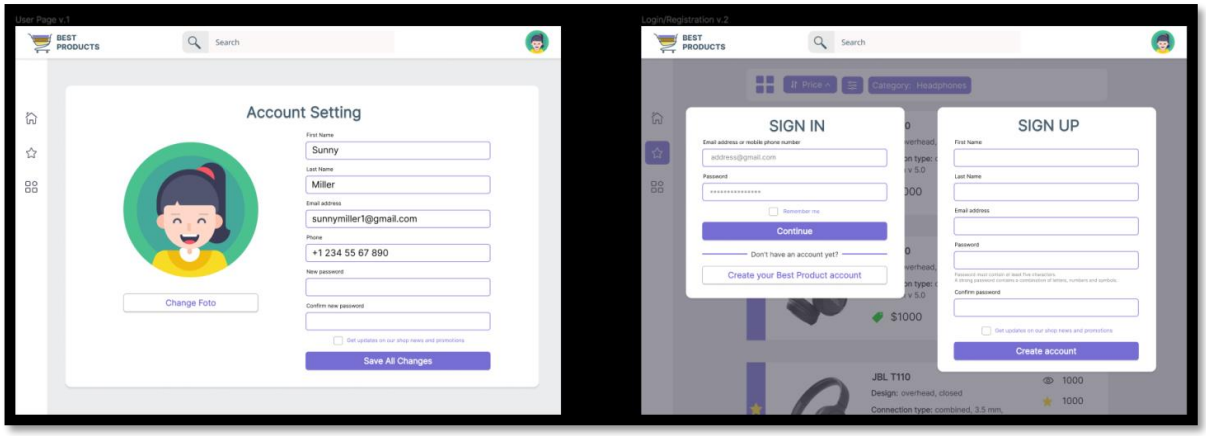

Рисунок 1.5 – Шаблон сторінки користувача та форми аутентифікації та реєстрації

1.2 Висновки до розділу 1

На цьому етапі було проведено детальний аналіз вихідних даних, включаючи технічне завдання від замовника, інструментарій для розробки та дизайн сторінок сайту. В результаті цього аналізу отримана глибоке розуміння вимог та очікувань замовника щодо функціонала та дизайну сайту. Це дозволить ефективно реалізувати необхідний функціонал та відтворити бажаний дизайн, забезпечуючи високу якість та відповідність вимогам замовника.

2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

2.1 Концептуальна модель бази даних

Отже, на зараз ми маємо технічне завдання та інструментарій для виконання, тому, першим кроком, змоделюємо сутності з якими ми будемо мати справу. Також для кожної сутності одразу буде створений код для міграцій.

Для простоти розуміння, спочатку буде продемонстровано загальну модель усіх таблиць (див. рис. 2.1).

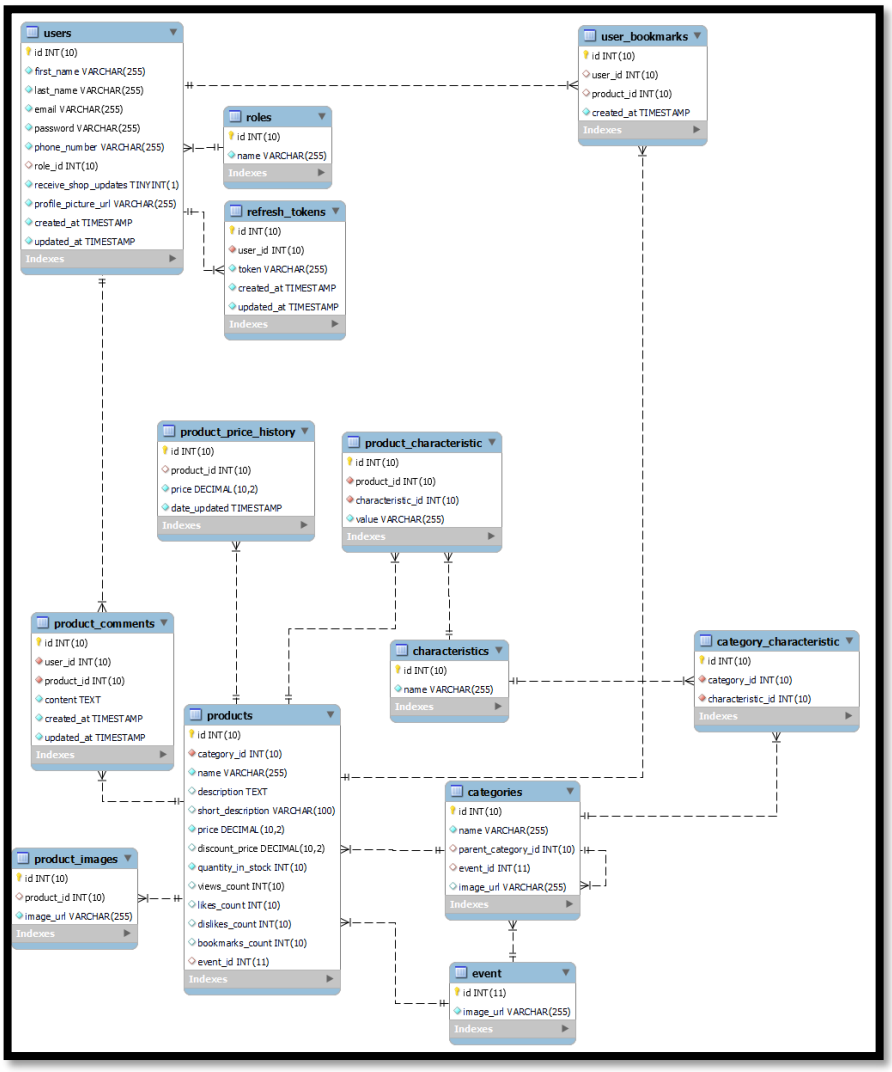

Рисунок 2.1 – Схема бази даних

Також для більшого розуміння, додано глобальні арени залежних таблиць (див. рис. 2.2). Цей підхід допомагає в ідентифікації та аналізі взаємозв'язків між таблицями, що дозволяє ефективніше керувати та оптимізувати базу даних з урахуванням її складної структури.

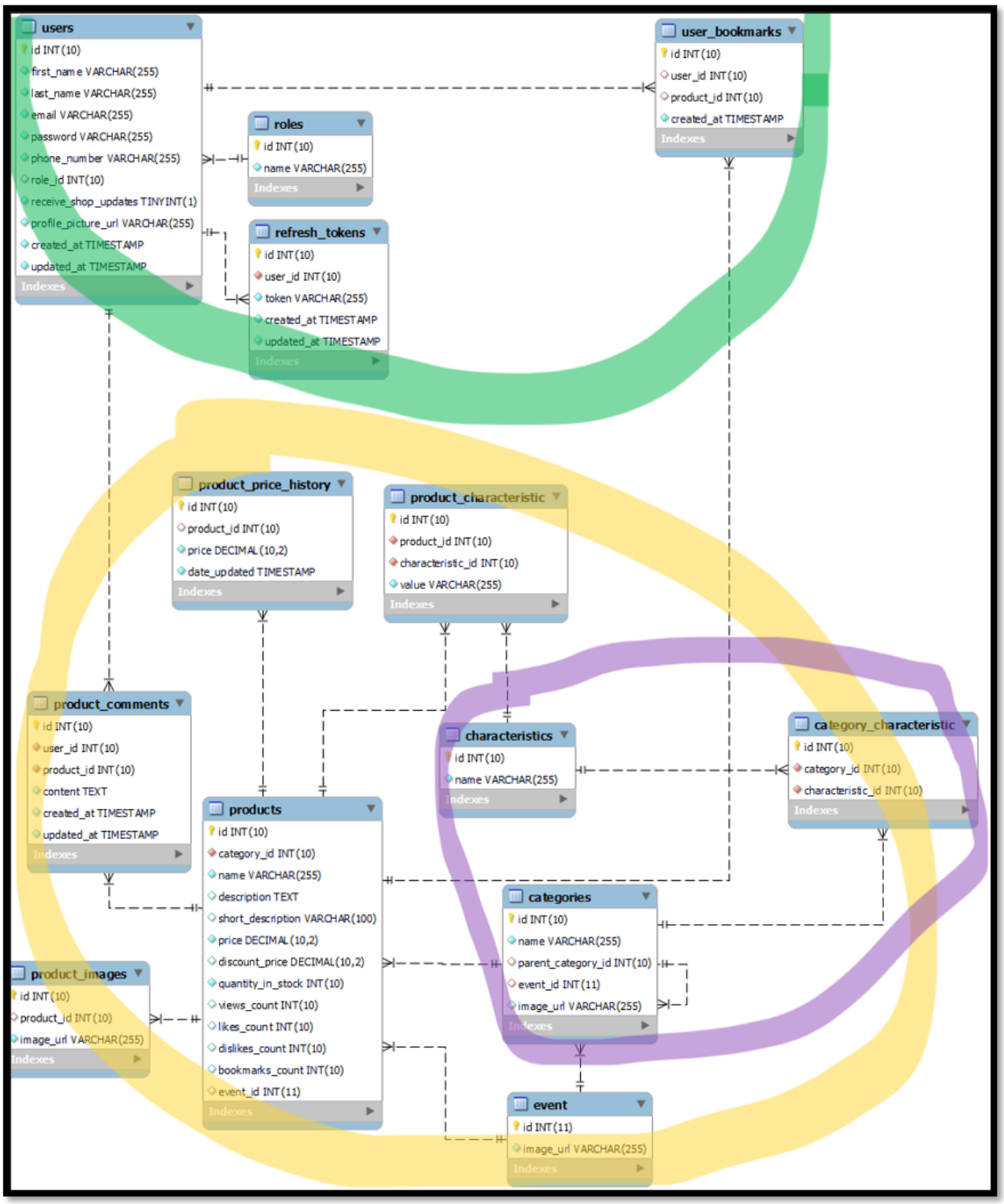

Рисунок 2.2 – Арени відповідальності

2.2 Практична реалізація бази даних

Модель продукту є ключовим компонентом системи та визначає основні характеристики товарів, що пропонуються в електронному магазині (див. рис. 2.3).

Детальний опис моделі включає наступні атрибути:

- **id:** унікальний ідентифікатор продукту, автоматично збільшується;
- **category_id:** ідентифікатор категорії, до якої належить продукт, зовнішній ключ, пов’язаний з таблицею категорій;
- **name:** назва продукту, яка не може бути порожньою;
- **description:** повний опис продукту, може містити велику кількість тексту;
- **short_description:** короткий опис продукту, обмежений 100 символами;
- **price:** ціна продукту у вигляді десяткового числа з фіксованою точністю 10.2;
- **discount_price:** знижена ціна продукту у випадку акції або знижки;
- **quantity_in_stock:** кількість одиниць товару на складі;
- **views_count:** кількість переглядів продукту, за замовчуванням ініціалізується нулем;
- **likes_count:** кількість позитивних відгуків про продукт, за замовчуванням ініціалізується нулем;
- **dislikes_count:** кількість негативних відгуків про продукт, за замовчуванням ініціалізується нулем;
- **bookmarks_count:** кількість разів, коли продукт додано в обране, за замовчуванням ініціалізується нулем;
- **event_id:** ідентифікатор події, пов’язаної з продуктом, зовнішній ключ, пов’язаний з таблицею подій. Може бути пустим у випадку, якщо продукт не пов’язаний з жодною подією.

Ця модель включає два зовнішні ідентифікатори: **category_id** та **event_id**. Це зроблено для того, щоб кожен продукт мав чітко визначену

категорію та, за необхідності, мав можливість пов’язатися із конкретною акційною подією, наприклад, чорною п’ятницею.

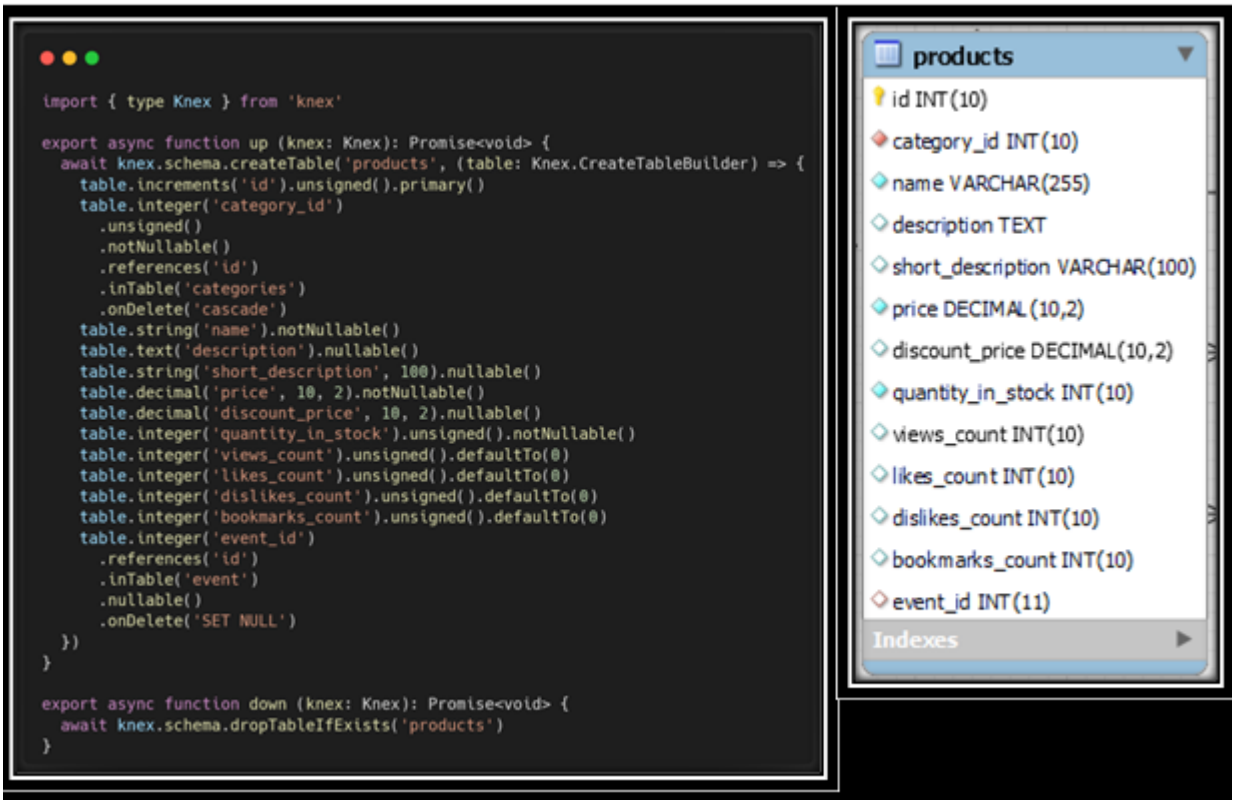

Рисунок 2.3 – Код міграції та таблиця products у базі даних

Така структура дозволяє ефективно каталогізувати продукти та забезпечує гнучкість у визначенні їх асоціацій з різними категоріями та подіями. Тому, розглянемо ці дві моделі.

Почнемо з моделі категорій (див. рис. 2.4):

- **id:** унікальний ідентифікатор категорії, що виступає основним ключем таблиці;
- **name:** назва категорії товарів;
- **parent_category_id:** зовнішній ключ, що вказує на батьківську категорію, до якої може відноситися дана;
- **event_id:** зовнішній ключ, який може бути пов’язаний із конкретною подією, що має відношення до категорії;
- **image_url:** URL-адреса зображення, що представляє категорію.

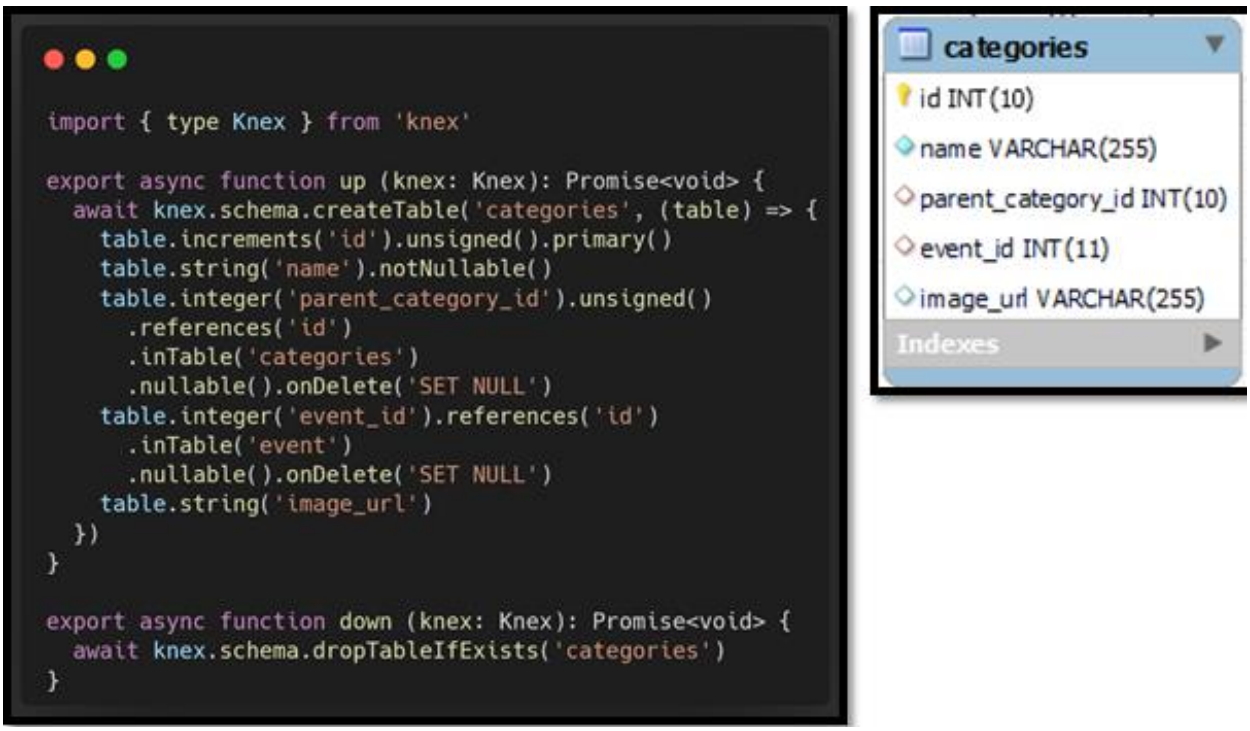

Рисунок 2.4 – Код міграції та таблиця categories у базі даних

Ця модель категорій передбачає можливість рекурсивної організації підкатегорій через поле parent_category_id. Ця можливість дозволяє створювати складні ланцюжки підкатегорій, що є важливим аспектом для реалізації різноманітних функціоналів. Наприклад, вона допомагає у забезпеченні ефективної роботи з breadcrumbs, що дає можливість користувачам з легкістю відстежувати своє поточне місце в ієрархії категорій, незалежно від її глибини і складності. Такий підхід дає більшу гнучкість та зручність у використанні, роблячи ієрархію категорій більш прозорою.

Тепер розглянемо модель акцій\подій (див. рис. 2.5):

– **id:** унікальний ідентифікатор події, що виступає основним ключем таблиці;
– **image_url:** URL-адреса зображення, яка представляє подію чи акцію.

Для простоти моделювання було вирішено мати тільки фото акції, що буде демонструватись на сайті. За потрібності, додаткові параметри можна додати пізніше.

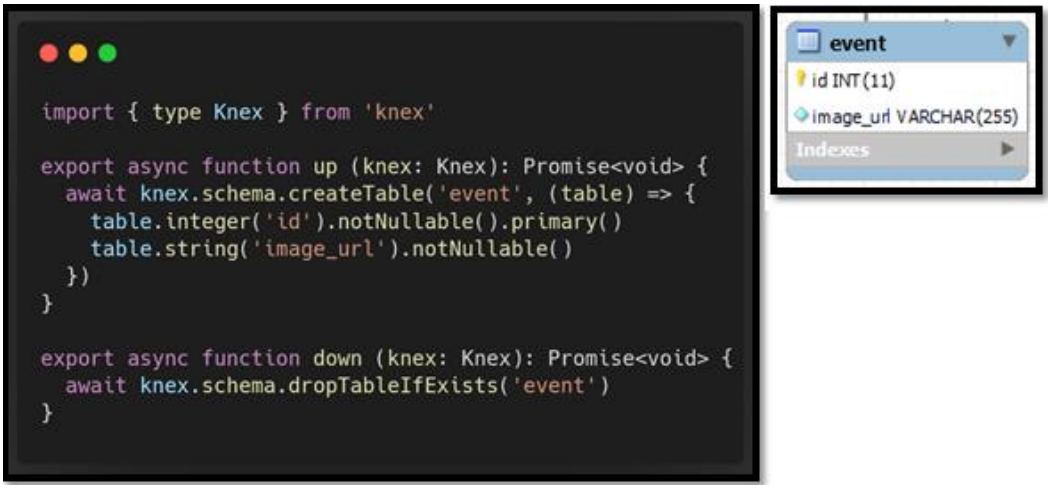

Рисунок 2.5 – Код міграції та таблиця event у базі даних

Тепер додамо таблицю для зображень продукту, що відзначається своєю важливою особливістю – можливістю динамічного додавання, видалення та зміни зображень для кожного окремого продукту (див. рис. 2.6). Ця функція дозволяє легко керувати графічним вмістом, забезпечуючи високий рівень гнучкості та персоналізації для кожного елементу асортименту:

– **id:** унікальний ідентифікатор зображення, що виступає основним ключем таблиці;
– **product_id:** зовнішній ключ, який посилається на ідентифікатор конкретного продукту в таблиці «products». Використовується для встановлення зв’язку між зображенням та продуктом;
– **image_url:** URL-адреса зображення, яке представляє конкретний продукт. За замовчуванням встановлено URL-адресу для зображення «no_image.png», яка використовується, якщо для продукту не надано іншого зображення.

Реалізація цієї моделі відзначається своєю простотою, що дозволяє легко адаптувати її для міграції до різних баз даних. Однак варто врахувати потенційну проблему, якою є збільшення кількості записів. Деякі бази даних пропонують ефективне розв’язання цього питання за допомогою типу даних «масив».

Наприклад, ми можемо зберігати масив зображень продукту в одному

запису, уникнувши необхідності створення окремого запису для кожного зображення. Проте важливо зазначити, що не всі бази даних підтримують цей тип реалізації.

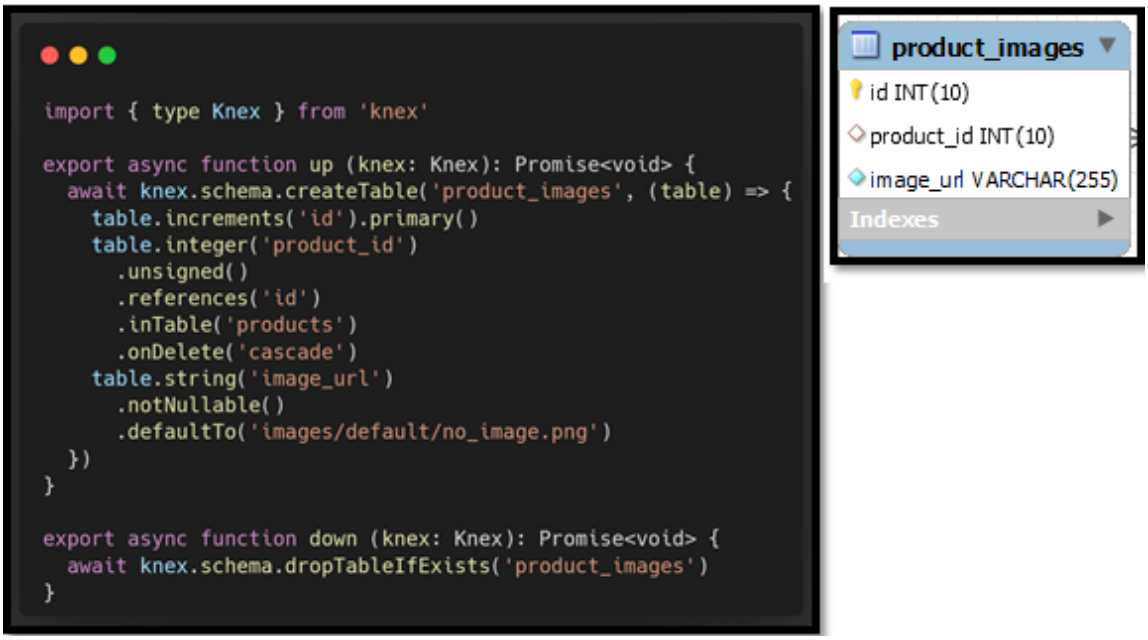

Рисунок 2.6 – Код міграції та таблиця product_images у базі даних

Далі розглянемо модель історії ціни продукту (див. рис. 2.7):

- **id:** унікальний ідентифікатор запису історії цін, що виступає основним ключем таблиці;
- **product_id:** зовнішній ключ, який посилається на ідентифікатор конкретного продукту в таблиці «products». Використовується для встановлення зв’язку між історією цін та конкретним продуктом;
- **price:** ціна продукту на момент збереження запису історії цін;
- **date_updated:** дата та час оновлення ціни, автоматично фіксується при зміні ціни продукту.

Мотивацією для створення такої моделі є можливість відстежування кожного моменту оновлення ціни, який відображається у створенні нового запису в таблиці. Зберігаючи повну історію подібних змін, ми отримуємо можливість виконувати різноманітні обчислення, такі як обчислення середньої ціни за тиждень, місяць чи рік. Ця можливість аналізу та статистики дозволяє

зробити більш об’єктивні рішення, спираючись на динаміку змін цін у різні періоди часу.

Рисунок 2.7 – Код міграції та таблиця product_price_history у базі даних

Також для кожного продукту має сенс мати якісь характеристики: колір, вага, матеріал, т.д. Тому, створимо модель для цього (див. рис. 2.8):

- **id:** унікальний ідентифікатор запису характеристики продукту, що виступає основним ключем таблиці;
- **product_id:** зовнішній ключ, який посилається на ідентифікатор конкретного продукту в таблиці «products». Використовується для встановлення зв’язку між характеристикою та конкретним продуктом;
- **characteristic_id:** зовнішній ключ, який посилається на ідентифікатор конкретної характеристики в таблиці «characteristics». Використовується для встановлення зв’язку між характеристикою та конкретним продуктом;
- **value:** значення характеристики продукту.

Ця модель надає нам можливість гнучко налаштовувати характеристики продуктів. Однак важливо відзначити, що для використання цієї моделі потрібно встановлювати ідентифікатор характеристики (characteristic_id), яку

ми бажаємо включити. Розглянемо деталі моделі характеристик (рис. 2.9):

– **id:** унікальний ідентифікатор запису характеристики, що виступає основним ключем таблиці;
– **name:** назва характеристики, яка ідентифікує та описує конкретний аспект продукту.

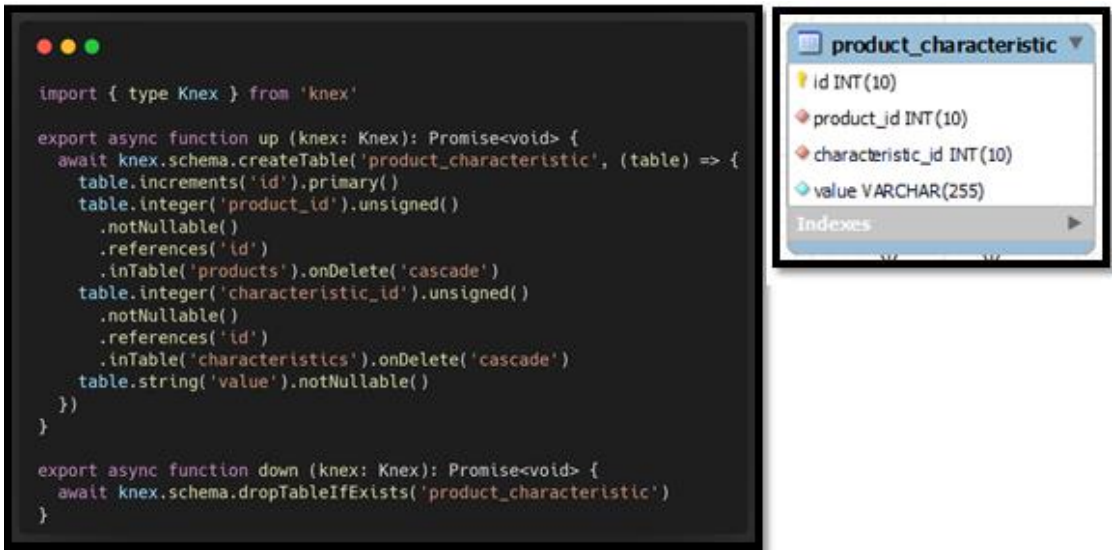

Рисунок 2.8 – Код міграції та таблиця product_characteristic у базі даних

Рисунок 2.9 – Код міграції та таблиця characteristics у базі даних

Ця модель є досить простою, оскільки для її належного функціонування вистачає чітко визначеної назви характеристики, такої як «колір», «матеріал» і т.д. Мотивація для створення цієї таблиці детально розглядається в наступній

моделі, оскільки саме вона вимагає використання цієї проміжної таблиці – характеристика до кожної категорії (рис. 2.10):

- **id:** унікальний ідентифікатор запису в таблиці, який виступає основним ключем;
- **category_id:** зовнішній ключ, який посилається на ідентифікатор категорії в таблиці «categories». Визначає зв’язок між характеристиками та конкретною категорією;
- **characteristic_id:** зовнішній ключ, який посилається на ідентифікатор характеристики в таблиці «characteristics». Встановлює зв’язок між характеристикою та конкретною категорією.

```
import { type Knex } from 'knex'

export async function up (knex: Knex): Promise<void> {
  await knex.schema.createTable('category_characteristic', (table) => {
    table.increments('id').primary()
    table.integer('category_id').unsigned()
      .notNullable()
      .references('id')
      .inTable('categories')
      .onDelete('CASCADE')
    table.integer('characteristic_id').unsigned()
      .notNullable()
      .references('id')
      .inTable('characteristics')
      .onDelete('CASCADE')
  })
}

export async function down (knex: Knex): Promise<void> {
  await knex.schema.dropTableIfExists('category_characteristic')
}
```

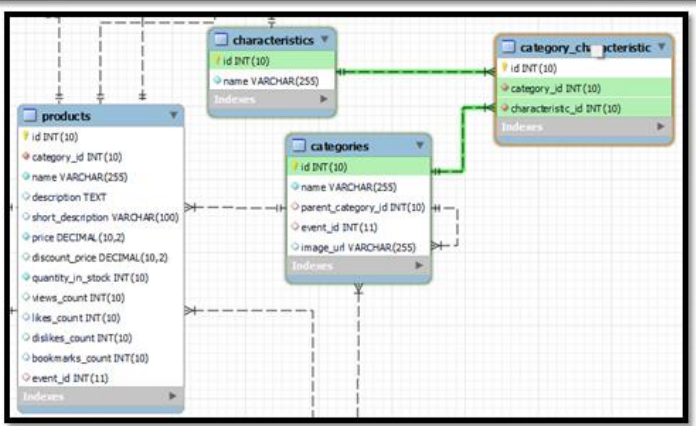

Рисунок 2.10 – Код міграції та таблиця category_characteristic у базі даних

Отже, мета створення цієї моделі полягає в контролі за тим, які характеристики може мати кожна окрема категорія продуктів. Наприклад, для

ноутбука необхідно визначити тип процесора, розмір екрана та колір корпусу, в той час, як для книги важливі характеристики можуть включати автора, кількість сторінок та колір обкладинки.

Цікаво відзначити, що якщо ноутбук та книга можуть мати спільну характеристику – колір. З цією метою була розроблена модель загальних характеристик, яка дозволяє перевикористовувати вже наявні характеристики та спрощує процес конфігурації нових продуктів.

Наступна модель – коментарі до продукту (див. рис. 2.11):

– **id:** унікальний ідентифікатор запису в таблиці, який виступає основним ключем;
– **user_id:** зовнішній ключ, який посилається на ідентифікатор користувача в таблиці «users». Визначає автора коментаря;
– **product_id:** зовнішній ключ, який посилається на ідентифікатор продукту в таблиці «products». Визначає продукт, до якого написаний коментар;
– **content:** зберігає текст коментаря користувача;
– **timestamps:** автоматично зберігає час створення та оновлення запису.

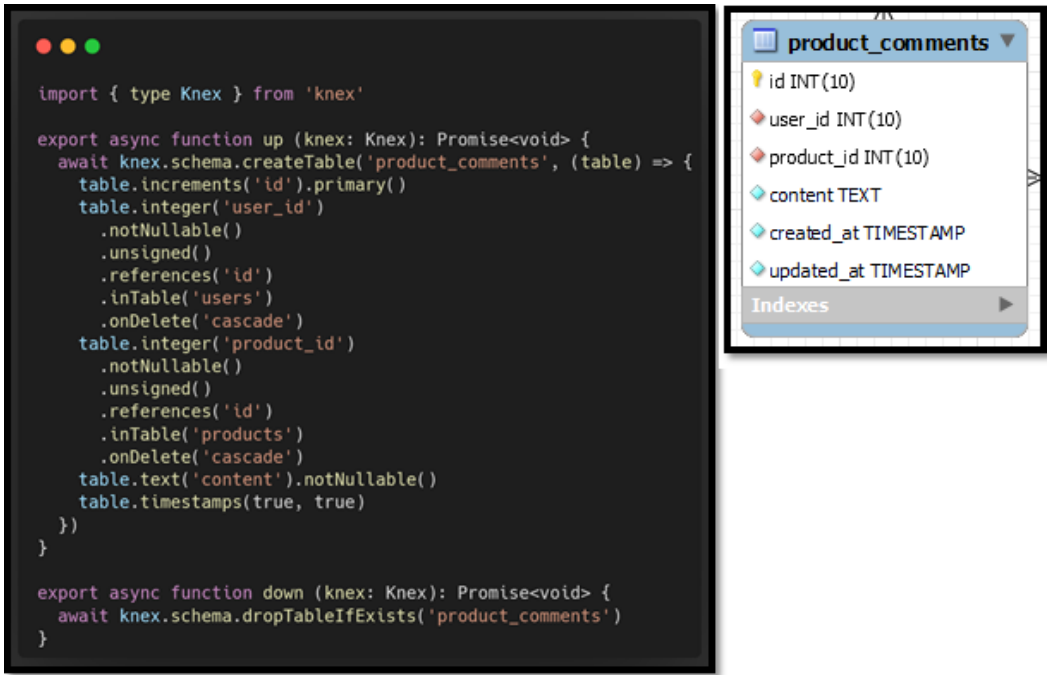

Рисунок 2.11 – Код міграції та таблиця product_comments у базі даних

Ця модель дозволяє легко зберігати численні коментарі до різних продуктів. Зауважимо два зовнішні ідентифікатори – product_id та user_id. Якщо таблиця продуктів вже існує, то таблиці користувачів ще немає. Таким чином, створимо її зараз (див. рис. 2.12):

- **id:** унікальний ідентифікатор запису в таблиці, який служить основним ключем;
- **first_name:** ім’я користувача;
- **last_name:** прізвище користувача;
- **email:** електронна пошта користувача;
- **password:** хеш пароля користувача для забезпечення безпеки;
- **phone_number:** номер телефону користувача;
- **role_id:** зовнішній ключ, який посилається на ідентифікатор ролі користувача в таблиці «roles». Визначає роль користувача в системі;
- **receive_shop_updates:** визначає, чи користувач хоче отримувати оновлення від магазину;
- **profile_picture_url:** URL-адреса зображення профілю користувача;
- **timestamps:** визначають момент часу створення та оновлення запису в таблиці.

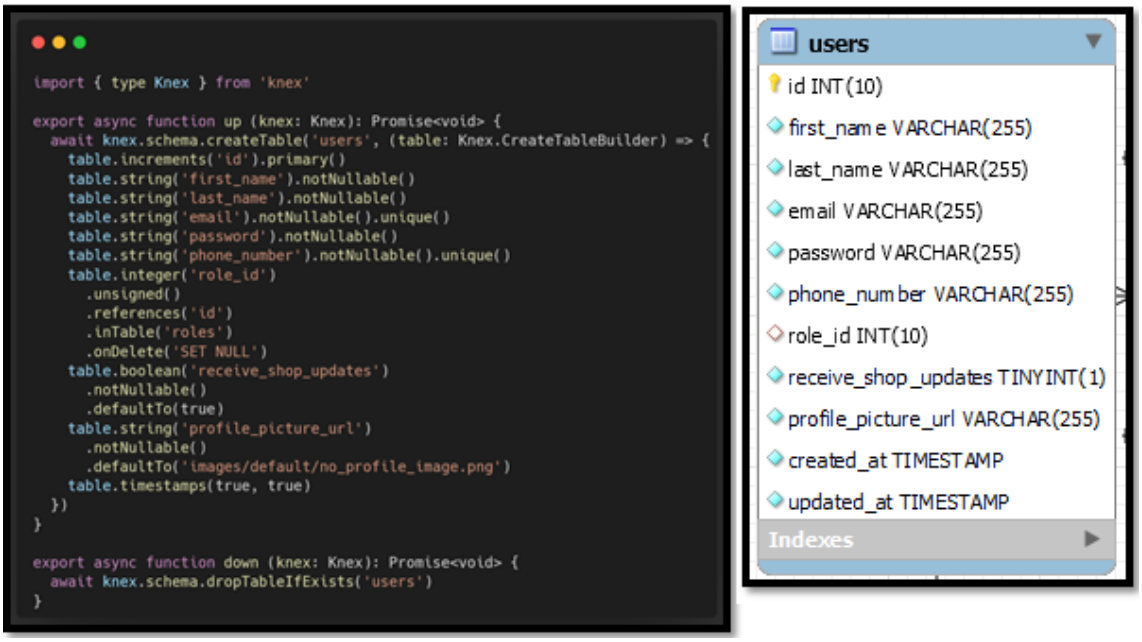

Рисунок 2.12 – Код міграції та таблиця users у базі даних

Ця модель користувачів представляє стандартний підхід до зберігання основної інформації про користувачів. Однак слід відзначити, що для повноцінного функціонування системи необхідно додати модель для ролей, оскільки у цій моделі присутній зовнішній ключ role_id. Це дозволить належним чином визначити ролі користувачів у системі (див. рис. 2.13):

– **id:** унікальний ідентифікатор запису в таблиці, що є основним ключем;
– **name:** назва ролі, яка однозначно ідентифікує тип ролі в системі.

Рисунок 2.13 – Код міграції та таблиця roles у базі даних

Відповідно до вимог технічного завдання, передбачено можливість у користувачів мати список улюблених продуктів. З цією метою розглянемо створення моделі, в якій будуть зберігатися закладки користувачів (див. рис. 2.14):

– **id:** унікальний ідентифікатор запису в таблиці, що є основним ключем;
– **user_id:** зовнішній ключ, що посилається на ідентифікатор користувача в таблиці «users». Визначає, якому користувачеві належить дана закладка;
– **product_id:** зовнішній ключ, що посилається на ідентифікатор продукту в таблиці «products». Визначає, який продукт був доданий в закладки користувачем;
– **created_at:** вказує час створення запису про закладку.

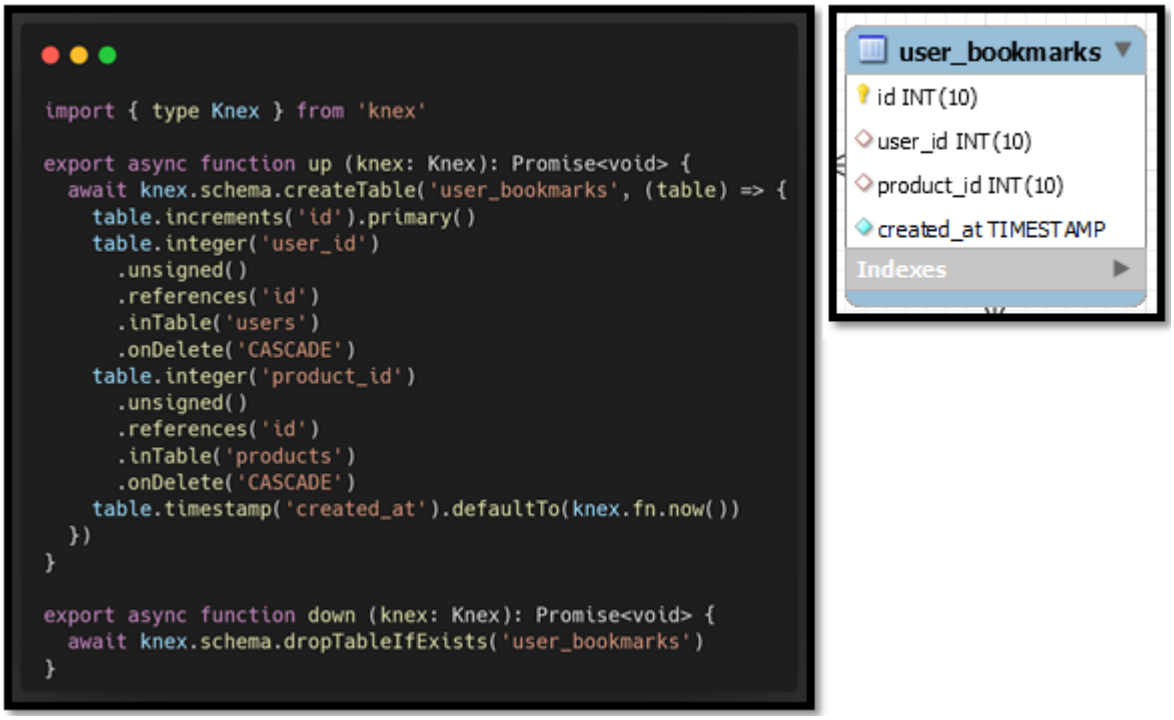

Рисунок 2.14 – Код міграції та таблиця user_bookmarks у базі даних

Ця архітектура моделі дозволяє нам гнучко додавати та видаляти записи відповідно до потреб системи.

Головний етап створення моделей в рамках проєкту успішно завершено, і вони належним чином відповідають основному функціоналу сайту. Проте, перед завершенням цієї фази розробки залишився лише один важливий крок – впровадження останньої моделі, яка відповідатиме за зберігання токенів оновлення токена доступу (див. рис. 2.15):

- **id:** унікальний ідентифікатор запису в таблиці, який є основним ключем;
- **user_id:** зовнішній ключ, що посилається на ідентифікатор користувача в таблиці «users». Вказує, якому користувачеві належить даний токен оновлення;
- **token:** унікальний токен оновлення, який використовується для обміну на новий токен доступу після його закінчення терміну дії;
- **created_at та updated_at:** вказують час створення та останнього оновлення запису про токен оновлення.

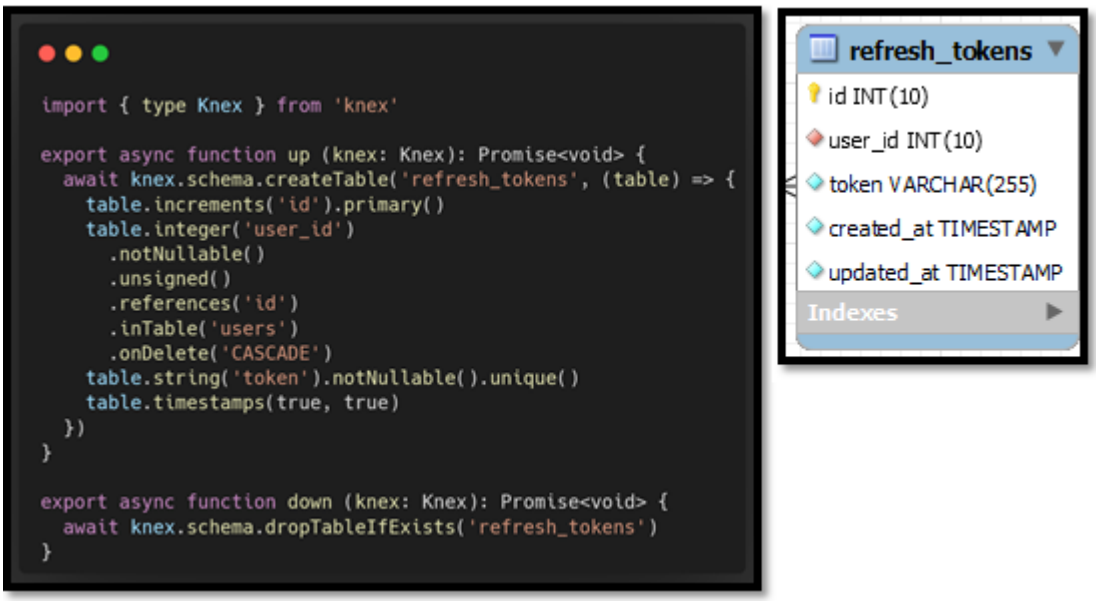

Рисунок 2.15 – Код міграції та таблиця refresh_tokens у базі даних

Сайт використовує два види токенів: токен доступу для отримання приватних ресурсів та токен оновлення для оновлення токена доступу. Цей підхід забезпечує безпеку та автентифікацію в додатку, забезпечуючи безперервний доступ до його функціонала.

2.3 Висновки до розділу 2

Було проведено детальний аналіз етапів розробки комерційного продукту. Зокрема, вже на початковому етапі, маючи технічне завдання від замовника, були створені моделі та на їх основі розроблені таблиці для бази даних MySQL. Додатково були розроблені міграції для кожної з моделей, що дозволить ефективно управляти змінами в структурі бази даних та забезпечити її консистентність. Крім того, детально прояснено особливості створених моделей, а також вказано причини їх обрання у саме такому вигляді, що сприятиме якості та ефективності подальшої розробки та управління проєктом.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКА

3.1 Реалізація сайту

Тепер перейдемо до реалізації. Почнемо з серверної частини. Важливо відзначити, що обрано класичну архітектуру MVC, яка не лише широко використовується, але й зарекомендувала себе як надійний та ефективний підхід до розробки програмного забезпечення.

MVC відзначається чітким розділенням логіки програми на моделі (дані), вигляд (представлення) та контролери (логіку обробки). Ця архітектура полегшує розробку, зберігає код організованим та сприяє його більшій модульності. Специфічно для нашого проєкту, вона дозволяє нам ефективно керувати серверною логікою, забезпечуючи гнучкість та зрозумілість в процесі розробки.

Загалом, архітектура маршруту запиту на сервер виглядає наступним чином:

– сервер приймає запит та направляє його до відповідного роуту;
– запит проходить один чи кілька middleware;
– запит потрапляє до контролера та оброблює запит.

Враховуючи кращі практики DDD (Domain-driven design), також буде доцільно використовувати патерн сервісу та репозиторію для роботи з бізнес-логікою.

Тепер створимо API для взаємодії з сервером (див. табл. 3.1 – 3.4).

Таблиця 3.1 – API користувачів

Метод	**API**	**Опис**
POST	/api/users/registration	Реєстрація нового користувача у системі.

Продовження таблиці 3.1

Метод	**API**	**Опис**
POST	/api/users/login/	Аутентифікація користувача та отримання доступу до системи.
POST	/api/users/myInfo/	Отримання особистої інформації поточного користувача.
GET	/api/users/bookmarks/	Отримання списку закладок, пов’язаних із поточним користувачем.
POST	/api/users/bookmarks/:id	Додавання закладки до колекції користувача за вказаним ідентифікатором.
DELETE	/api/users/bookmarks/:id	Видалення закладки з колекції користувача за вказаним ідентифікатором.
GET	/api/users/statistics/	Отримання статистичних даних щодо активності користувача.
GET	/api/users/table/	Отримання табличної інформації про користувачів.
DELETE	/api/users/	Видалення користувача з системи.
PATCH	/api/users/myInfo	Оновлення особистої інформації поточного користувача.
PATCH	/api/users/:id	Оновлення інформації про користувача за вказаним ідентифікатором.
GET	/api/users/refresh	Оновлення токена доступу користувача.
POST	/api/users/logout	Вихід користувача із системи.

Таблиця 3.2 – API продуктів

Метод	**API**	**Опис**
GET	/api/products/:productId/comments	Отримання коментарів для конкретного продукту за вказаним ідентифікатором.
GET	/api/products/:productId/characteristics	Отримання характеристик конкретного продукту за вказаним ідентифікатором.
GET	/api/products/:productId/images	Отримання зображень для конкретного продукту за вказаним ідентифікатором.
GET	/api/products/:productId/priceHistory	Отримання історії цін для конкретного продукту за вказаним ідентифікатором.
POST	/api/products/:productId/comments	Додавання коментаря до конкретного продукту з валідацією та твердою аутентифікацією.
DELETE	/api/products/:productId/comments/:commentId	Видалення коментаря до конкретного продукту за вказаним ідентифікатором з твердою аутентифікацією.
GET	/api/products/statistics	Отримання статистики продуктів з твердою аутентифікацією та валідацією.
GET	/api/products/table	Отримання табличних даних про продукти з твердою аутентифікацією та валідацією.

Продовження таблиці 3.2

Метод	**API**	**Опис**
DELETE	/api/products/	Видалення всіх продуктів з твердою аутентифікацією та валідацією.
PATCH	/api/products/:productId	Оновлення інформації про конкретний продукт за вказаним ідентифікатором з твердою аутентифікацією та валідацією.
POST	/api/products/	Додавання нового продукту з твердою аутентифікацією та валідацією.
PATCH	/api/products/:productId/comments/:commentId	Оновлення коментаря до конкретного продукту за вказаним ідентифікатором з твердою аутентифікацією та валідацією.
GET	/api/products/	Отримання всіх продуктів з додатковою аутентифікацією та перевіркою валідності запиту.

Таблиця 3.3 – API категорій

Метод	**API**	**Опис**
GET	/api/categories/	Отримання всіх категорій з вказаною валідацією.
GET	/api/categories/:categoryId/path	Отримання шляху категорії за вказаним ідентифікатором з валідацією.

Продовження таблиці 3.3

Метод	**API**	**Опис**
GET	/api/categories/:categoryId/ characteristics	Отримання характеристик категорії за вказаним ідентифікатором з валідацією.

Таблиця 3.4 – API подій

Метод	**API**	**Опис**
GET	/api/events/	Отримання всіх акцій.

Для кращого уявлення процесу, представимо собі обробку запиту (див. рис. 3.1). Припустимо, що користувач надсилає запит на сервер. Перш ніж продовжити, сервер перевіряє введені дані, щоб переконатися, що вони відповідають вимогам системи. Якщо дані не коректні або не повні, сервер повертає помилку валідації, надаючи користувачеві можливість виправити помилки та відправити запит знову.

У разі успішної перевірки, запит передається до контролера – компонента системи, що керує потоком запитів. Контролер, отримавши запит, викликає відповідний сервіс, який, своєю чергою, отримує доступ до репозиторію, де зберігаються дані користувачів.

Якщо дані відповідають очікуваному формату та умовам системи, сервіс генерує токен доступу, надаючи користувачеві доступ до системи.

Файлова структура backend коду (див. рис. 3.2) відображає організацію серверної реалізації за принципами патерну MVC (Model-View-Controller). Ця структура містить в себе такі ключові директорії, як controllers, database, middleware, models, routes, services та utils. Кожна з цих директорій відповідає за певний аспект роботи додатка, допомагаючи забезпечити чітку розділеність обов’язків та ефективне управління проєктом. Наприклад, директорія controllers містить контролери, які відповідають за обробку запитів, тоді як директорія models містить моделі, які визначають структуру даних застосунку.

Попри те, що клієнтська реалізація базується на React [2] і не має окремого шару view, зазначена файлова структура backend коду допомагає зберігати код організованим та легким для розширення та підтримки.

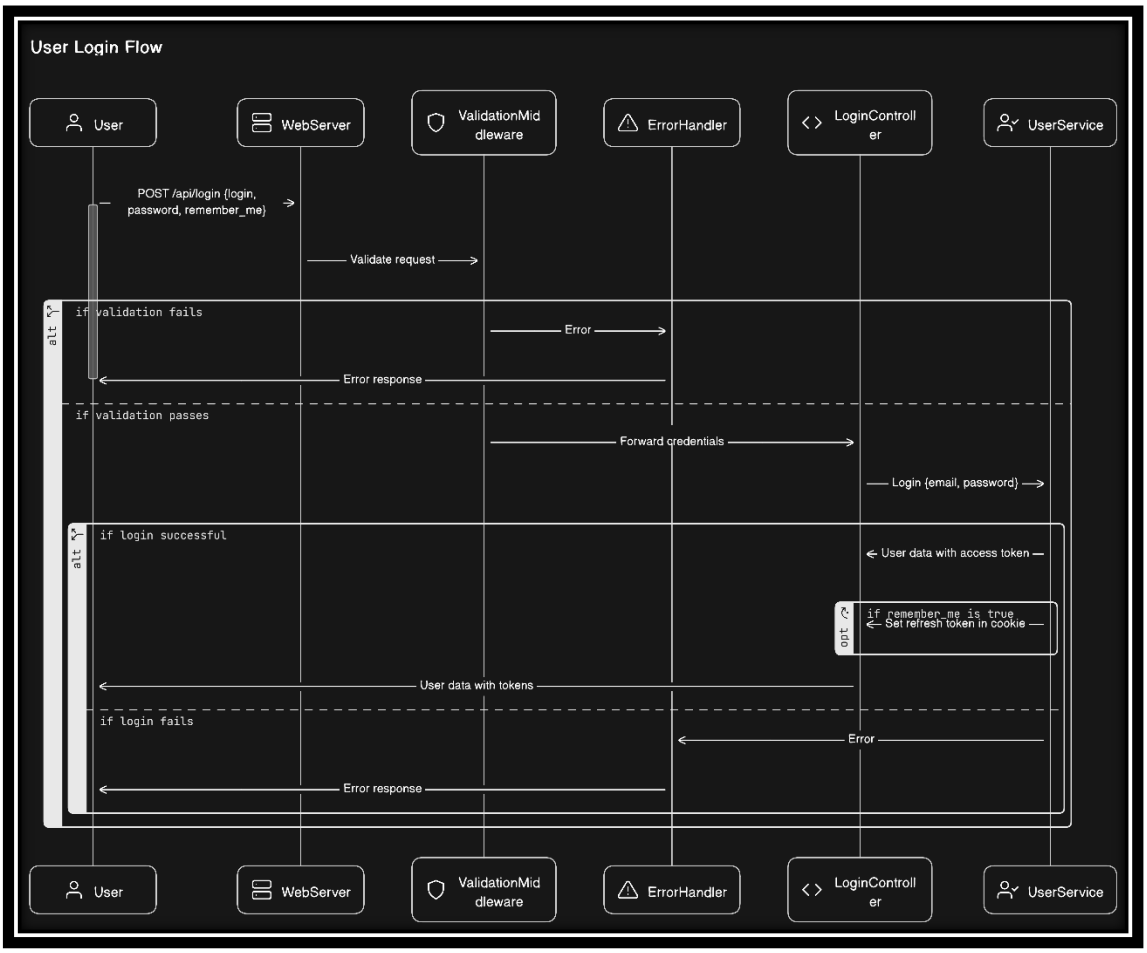

Рисунок 3.1 – Приклад обробки запиту [8]

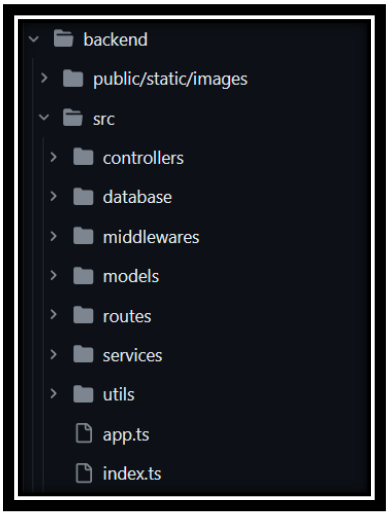

Рисунок 3.2 – Файлова структура backend

Під час розробки було приділено значну увагу важливості тестування коду. У результаті, було проведено повне тестування API з особливим акцентом на перевірку можливостей сервера щодо валідації наданих даних. Наприклад, ми переконалися у неможливості користувача підвищити свою роль шляхом простого «оновлення прізвища», де до запиту також була додана нова роль

Застосування функціонала тестування було інтегровано з використанням сервісу CI/CD на GitHub, що забезпечує автоматизовану перевірку якості коду з кожним новим оновленням коду. Крім того, ми надали можливість локального запуску тестів для розробників для ще більш ефективного контролю якості (див. рис. 3.3), що сприяє забезпеченню стабільності та надійності програмного забезпечення.

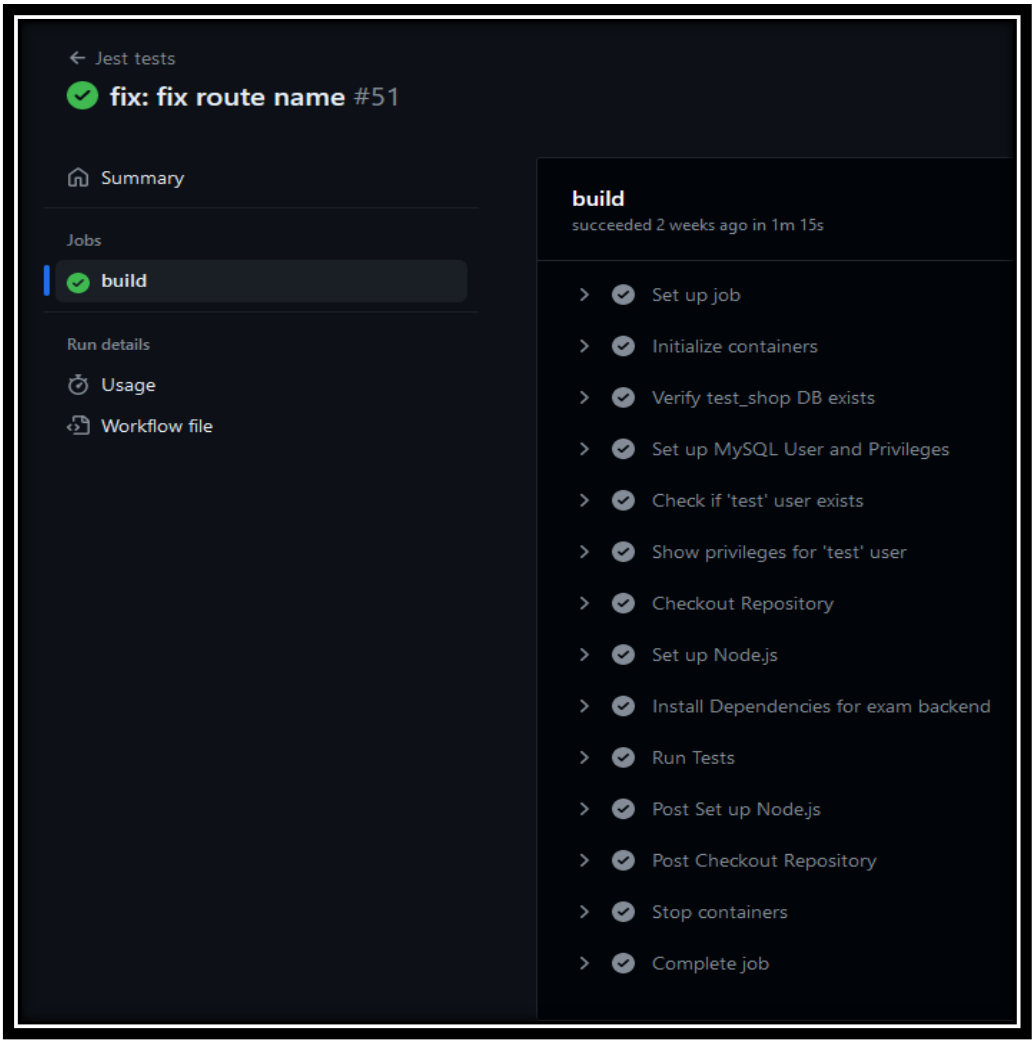

Рисунок 3.3 – Приклад успішної збірки проєкту [9]

У той час як серверна частина дотримується традиційної архітектури MVC, клієнтська частина цього проєкту використовує більш сучасний підхід: feature-Sliced Design (FSD). Цей архітектурний патерн пропонує явні переваги в управлінні складністю коду, що ідеально узгоджується з цілями проєкту.

FSD структурує код навколо функціоналу, групуючи пов’язані компоненти, логіку та стилі в автономні фрагменти. Це сприяє модульності, багаторазовому використанню та чіткому володінню ділянками коду.

Багаторівнева архітектура: FSD розділяє інтерфейс на шари, кожен з яких має певні обов’язки:

- **зрізи**: інкапсулюють окремі функції та пов’язані з ними компоненти, логіку та стилі;
- **сегменти**: поділяють фрагменти на менші блоки коду, які можна використовувати повторно, покращуючи організацію та зручність супроводу;
- **шари**: групування фрагментів на основі їхньої функціональності, що зазвичай включає рівень інтерфейсу користувача, логічний рівень і рівень даних.

Файлова структура frontend коду (див. рис. 3.4):

- **app:** ця директорія містить основний код фронтенду, включаючи маршрутизацію, компоненти верхнього рівня та інші ключові елементи;
- **pages:** у цій директорії розміщені компоненти сторінок, відповідальні за візуальний контент та інтерактивність окремих сторінок сайту;
- **shared:** ця директорія слугує сховищем для загальних компонентів, стилів та функцій, які використовуються в різних місцях клієнтської частини. це сприяє повторному використанню коду та покращує узгодженість інтерфейсу;
- **widgets:** у цій директорії розміщені автономні та повторно використовувані компоненти, які виконують специфічні функції, такі як кнопки, форми, таблиці тощо.

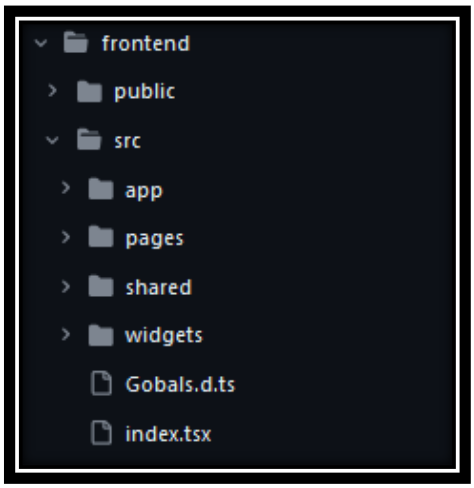

Рисунок 3.4 – Файлова структура frontend

3.2 Демонстрація реалізованого інтерфейсу

Шаблон сайту був представлений на етапі аналізу вимог замовника. Важливо відзначити, що, хоча замовник не висловлював конкретної вимоги до мобільної адаптації, такий функціонал все ж було впроваджено. Також була додана проста панель адміністратора.

Отже, спочатку клієнт відкриває сайт та бачить головну сторінку (див. рис. 3.5).

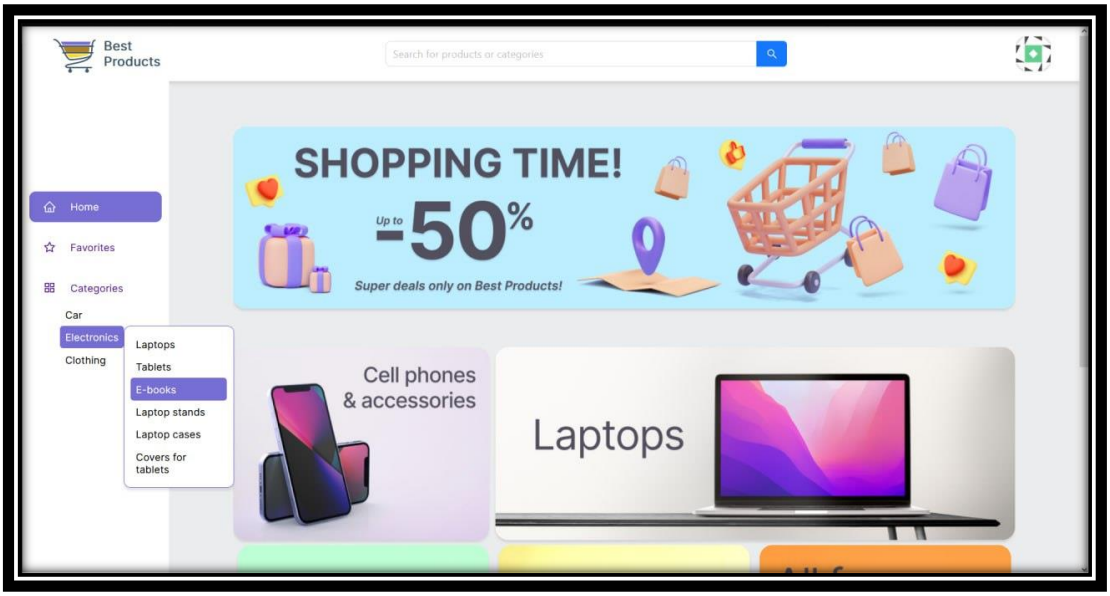

Рисунок 3.5 – Головна сторінка

Після завершення цього кроку, коли він вже відчув, що все готово до подальшої роботи, настав час перевірити налаштування свого акаунту з цікавою можливістю налаштування. Він уважно проглядає всі опції, прагнучи переконатися, що кожен параметр відповідає його особистим вподобанням і потребам.

Цікавим є те, що в рамках цього процесу він може одночасно активувати можливість отримання оновлень і рекомендацій, щоб завжди бути в курсі новин та мати доступ до рекомендацій, що стане важливим інструментом для ефективного використання платформи (див. рис. 3.6).

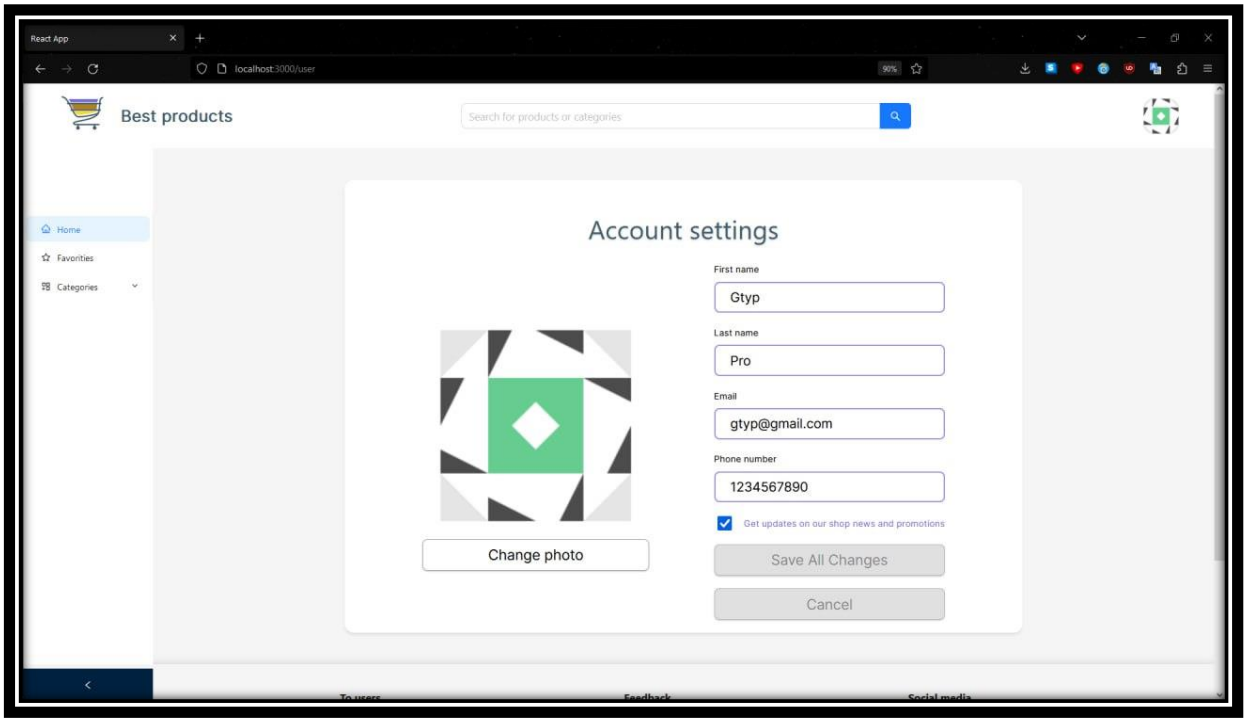

Рисунок 3.6 – Сторінка користувача

Після акуратної та ретельної перевірки всіх параметрів і налаштувань, які були належним чином налаштовані згідно з вимогами та потребами користувача, що виконує функції адміністратора в системі, виникає необхідність провести докладний та глибокий аналіз статистичної інформації. Цей аналіз спрямований на отримання всебічного розуміння поточного стану системи та ефективного прийняття обґрунтованих рішень.

У процесі аналізу необхідно враховувати широкий спектр показників, включаючи, але не обмежуючись, продуктивність, надійність, витрати ресурсів, а також рівень безпеки системи. Вивчення цих аспектів дозволяє адміністратору отримати повну картину функціонування системи та здійснити обґрунтовані рішення, що відповідають вимогам та цілям організації.

Під час аналізу інформації можуть застосовуватися різноманітні методи та інструменти, включаючи аналіз даних, статистичні моделі, а також візуалізацію даних за допомогою графіків, діаграм, табличних звітів та інших засобів (див. рис. 3.7, 3.8).

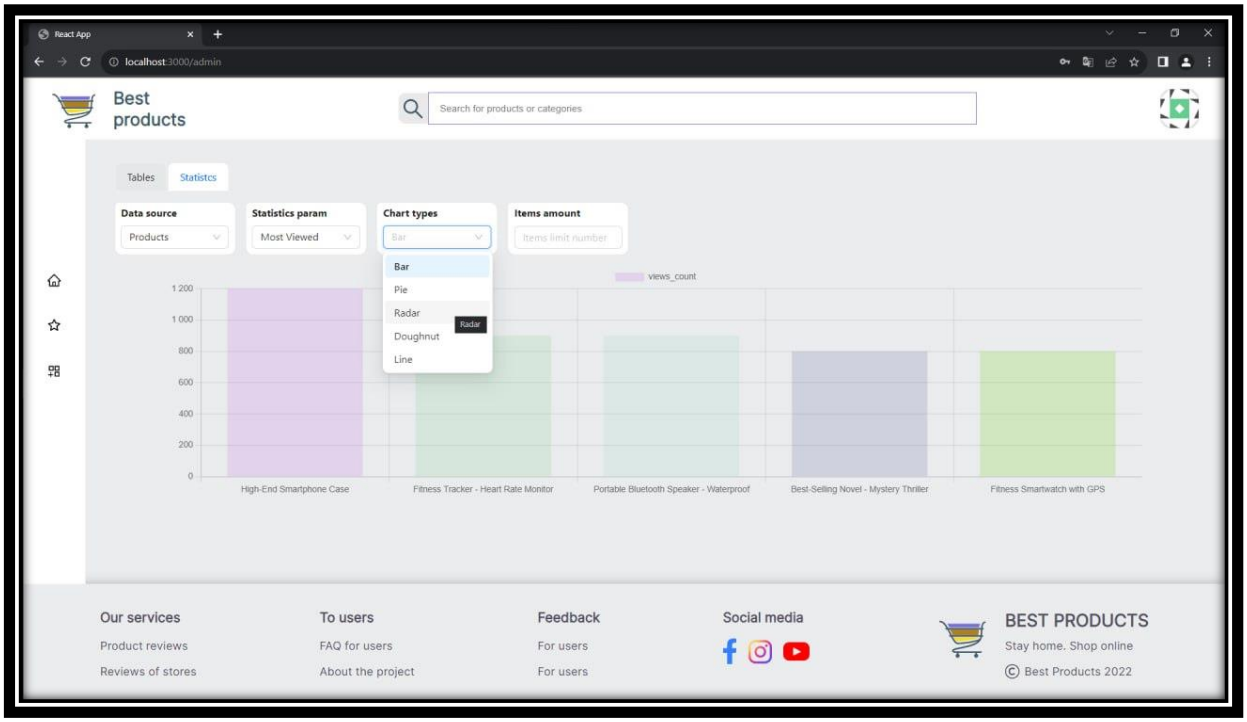

Рисунок 3.7 – Сторінка статистичної інформації

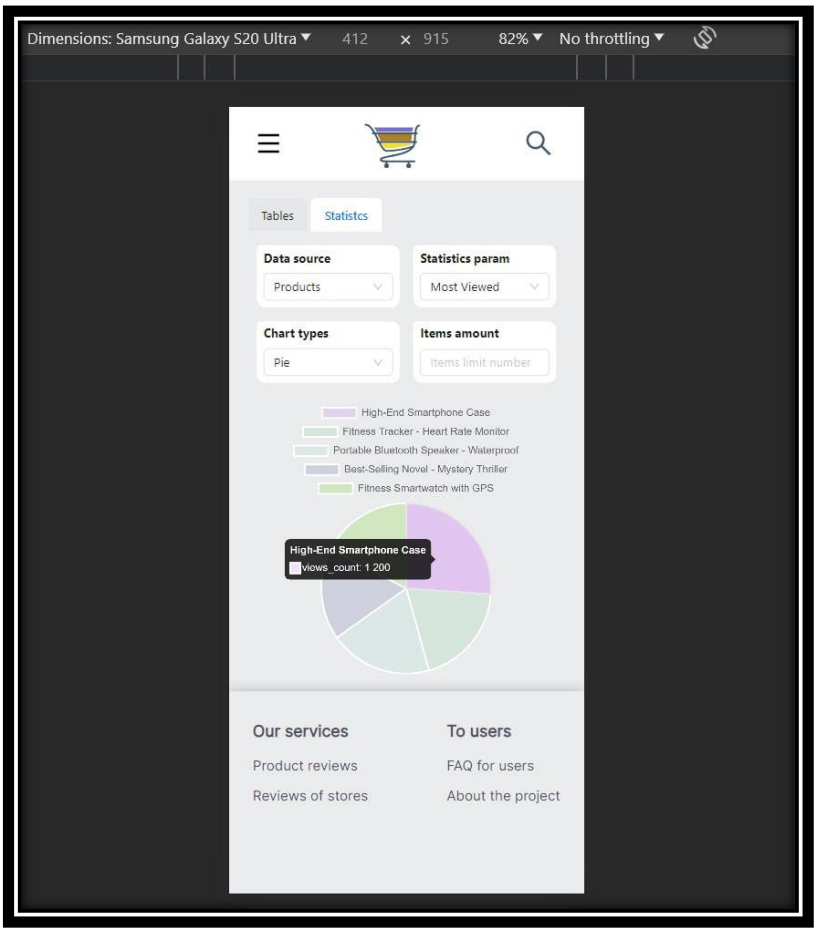

Рисунок 3.8 – Адаптивна сторінка статистичної інформації

Після перевірки статистики, користувач бажає знайти деякий продукт, тому він вирішує побачити список за певною категорією (див. рис. 3.9 – 3.10).

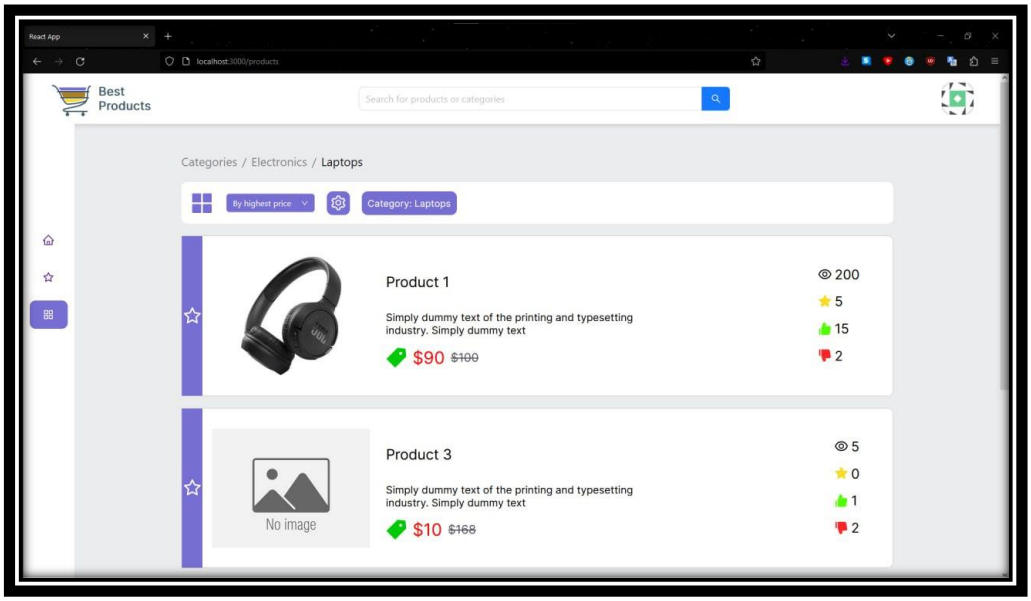

Рисунок 3.9 – Список продуктів

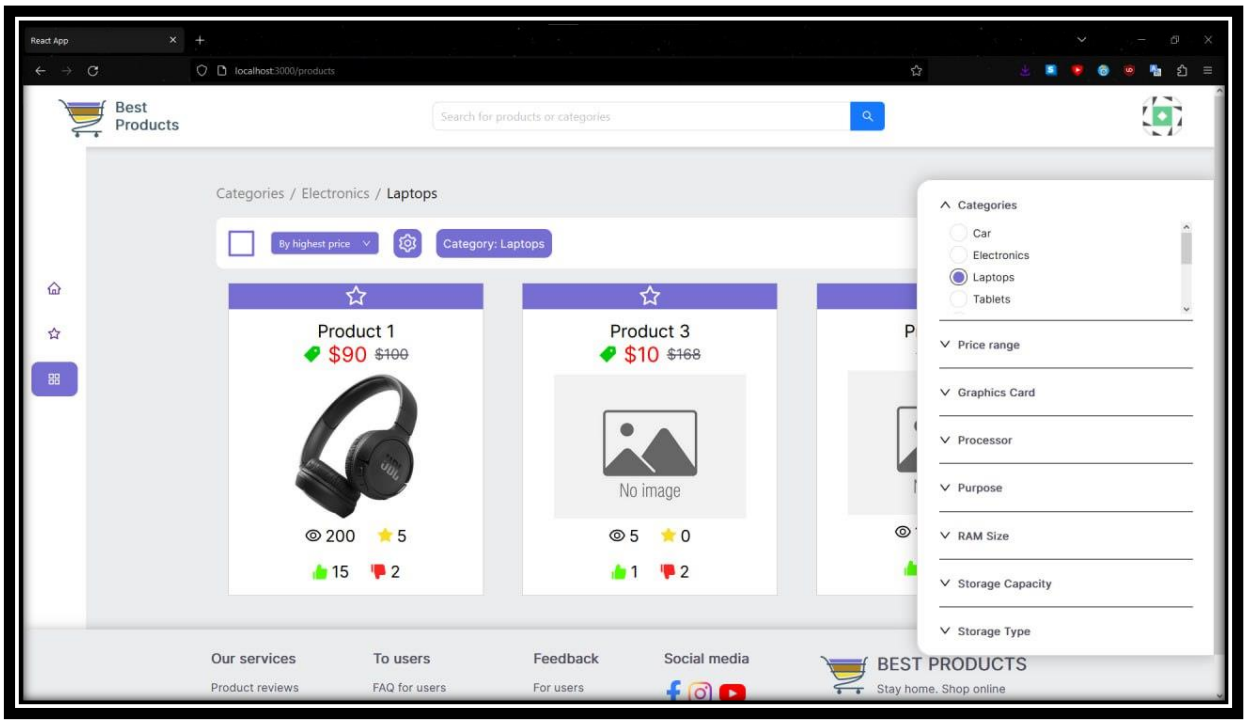

Рисунок 3.10 – Налаштування списку продуктів

Після спроб знайти продукт у списку, користувач вирішив знати продукт за назвою (див. рис. 3.11, 3.12).

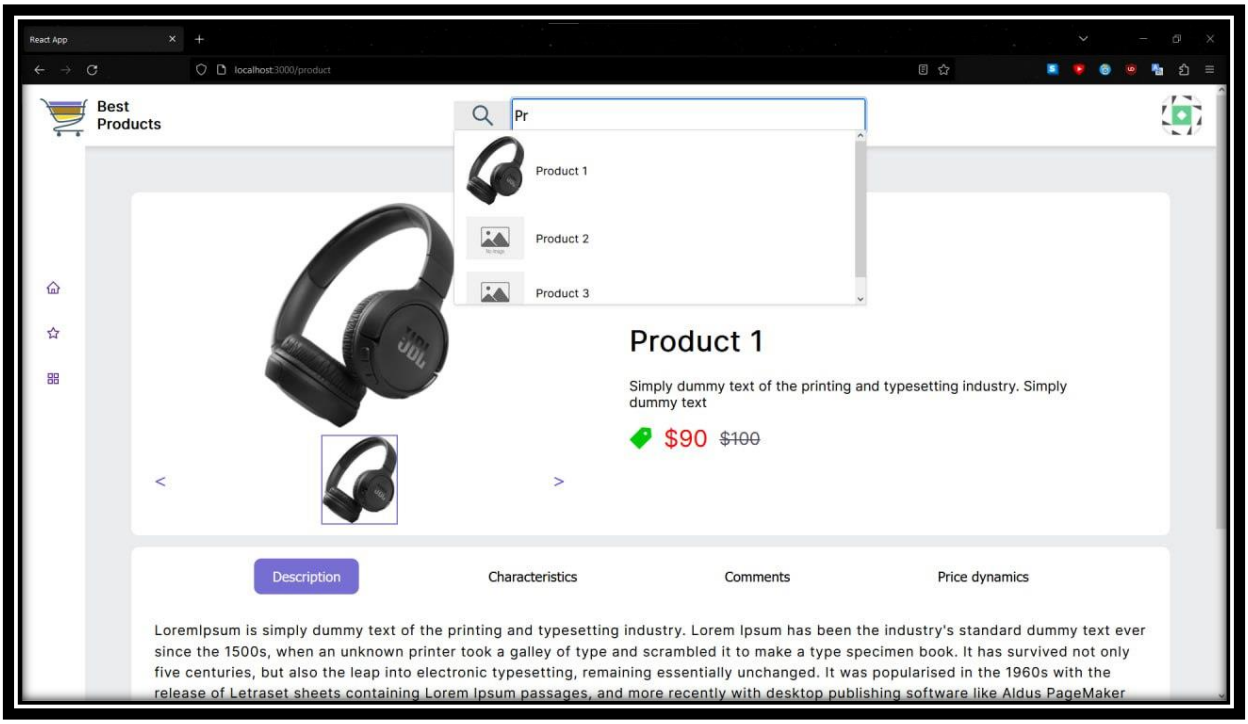

Рисунок 3.11 – Рекомендації у пошуку

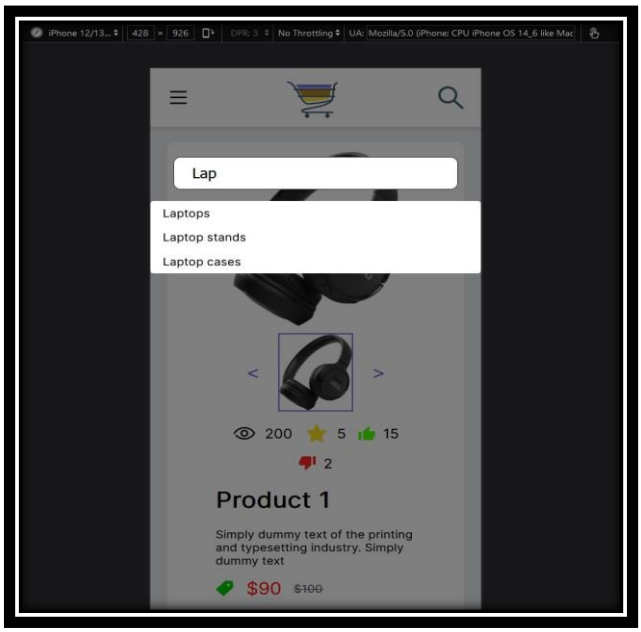

Рисунок 3.12 – Адаптивний пошук

Також користувач бажає переглянути деякий продукт та дані до нього, адже попередній аналіз – є важливою частиною перед придбанням (див. рис. 3.13, 3.14).

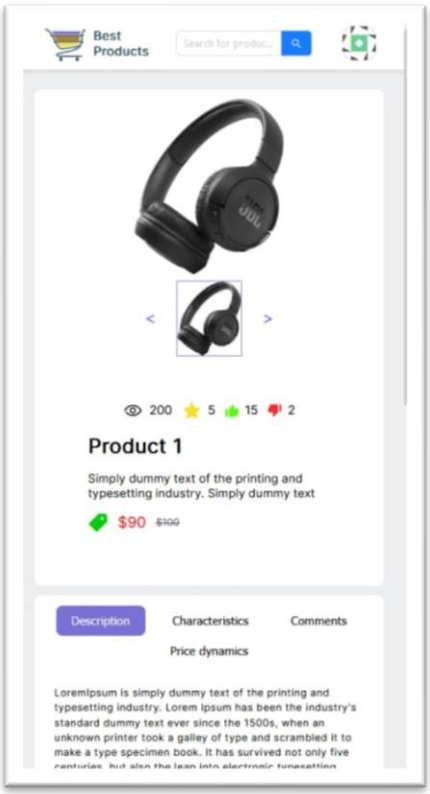

Рисунок 3.13 – Сторінка продукту

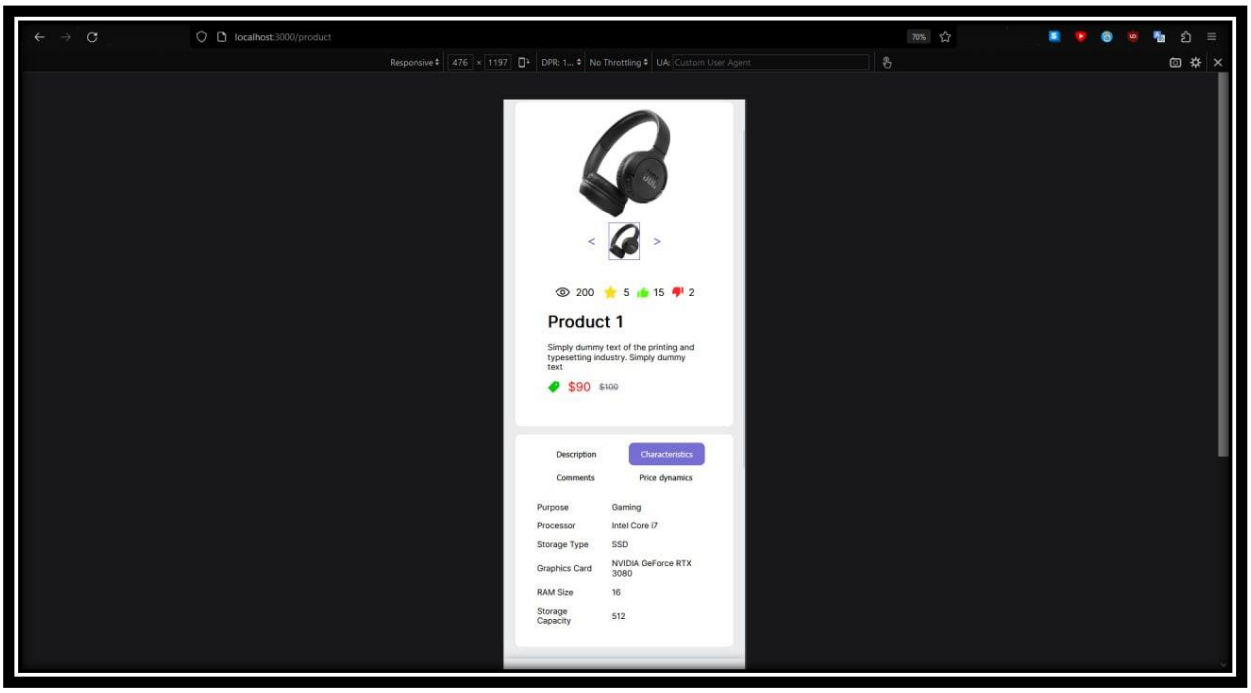

Рисунок 3.14 – Інформація про продукт

У результаті, користувачу також потрібно побачити думки щодо цього продукту, тому він вирішив переглянути коментарі (див. рис. 3.15, 3.16).

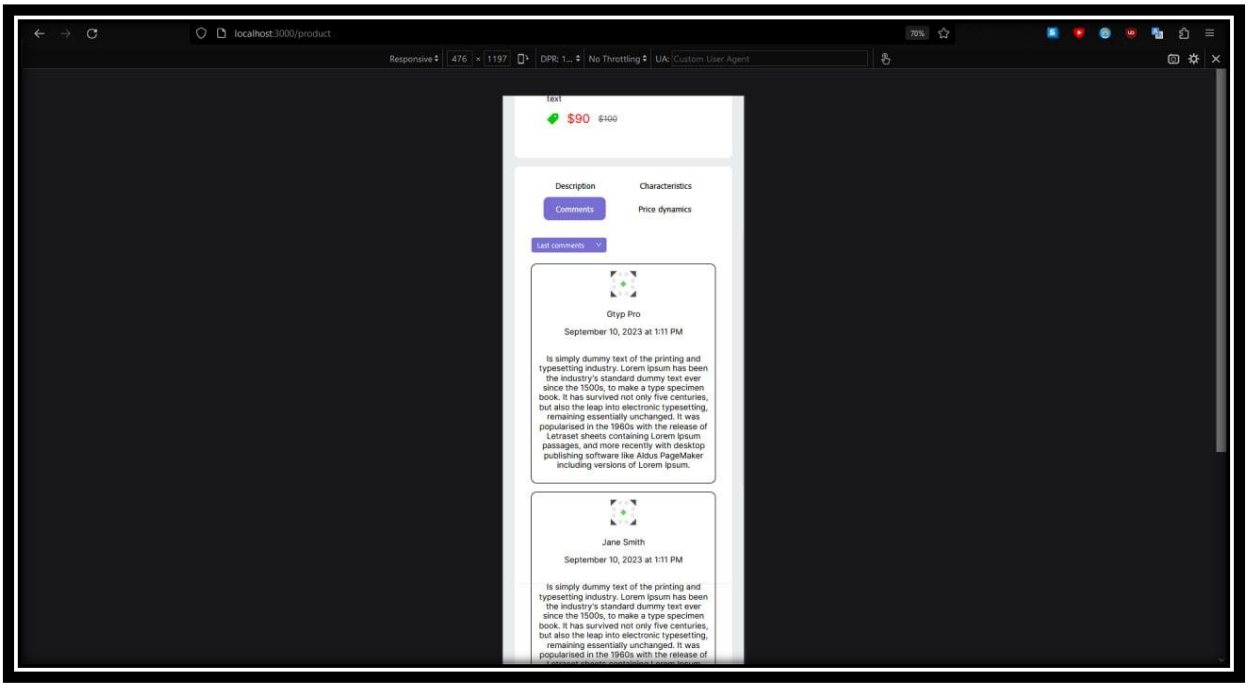

Рисунок 3.15 – Коментарі до продукту

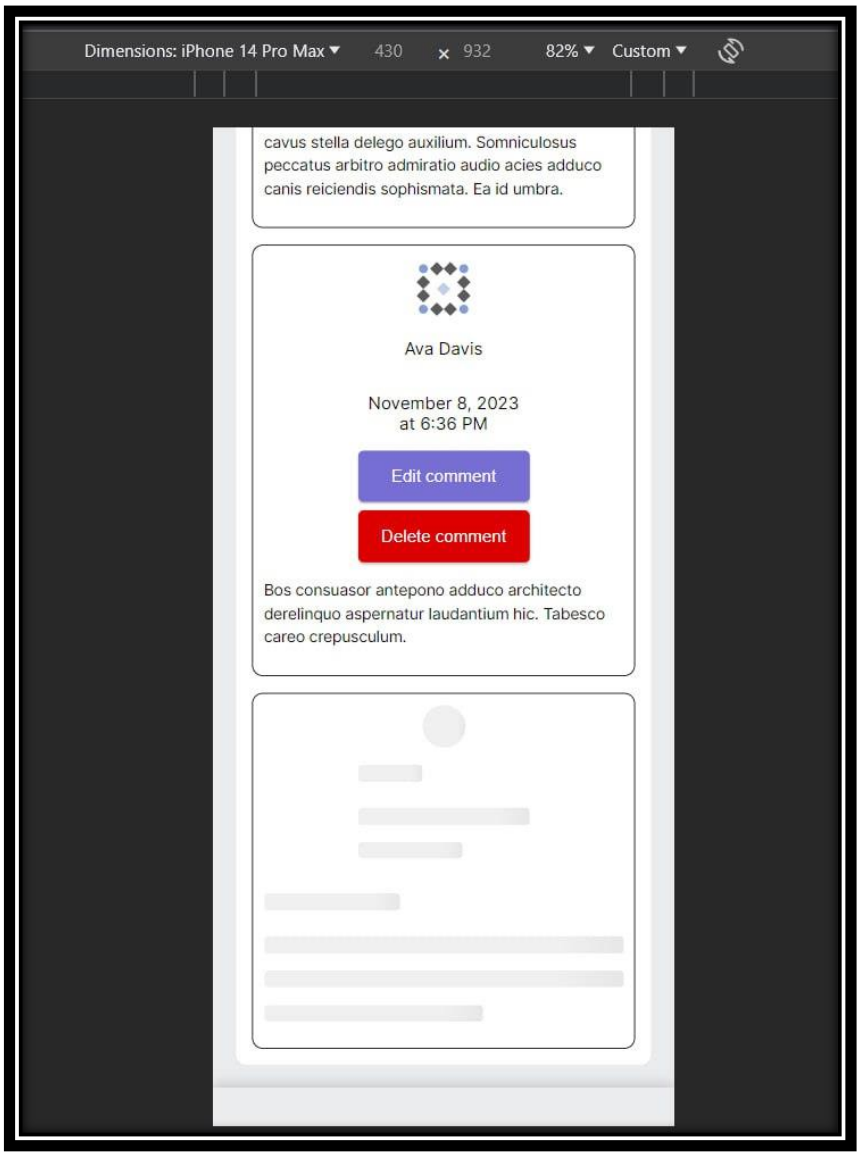

Рисунок 3.16 – Завантаження додаткових коментарів

Врешті решт, коли всі необхідні завдання виконано, важливо, щоб користувач завершив роботу у застосунку правильним чином, забезпечивши таким чином безпеку своїх даних та дотримання внутрішніх процедур.

Після успішного завершення роботи він має здійснити вихід із системи, щоб переконатися, що ніякі зміни не будуть втрачені та щоб підготувати застосунок до наступних користувачів або завдань.

Для зручності користувача, меню виходу доступне у бічній панелі застосунку (див. рис. 3.17), де він може знайти всі необхідні опції для завершення роботи та виходу з програми.

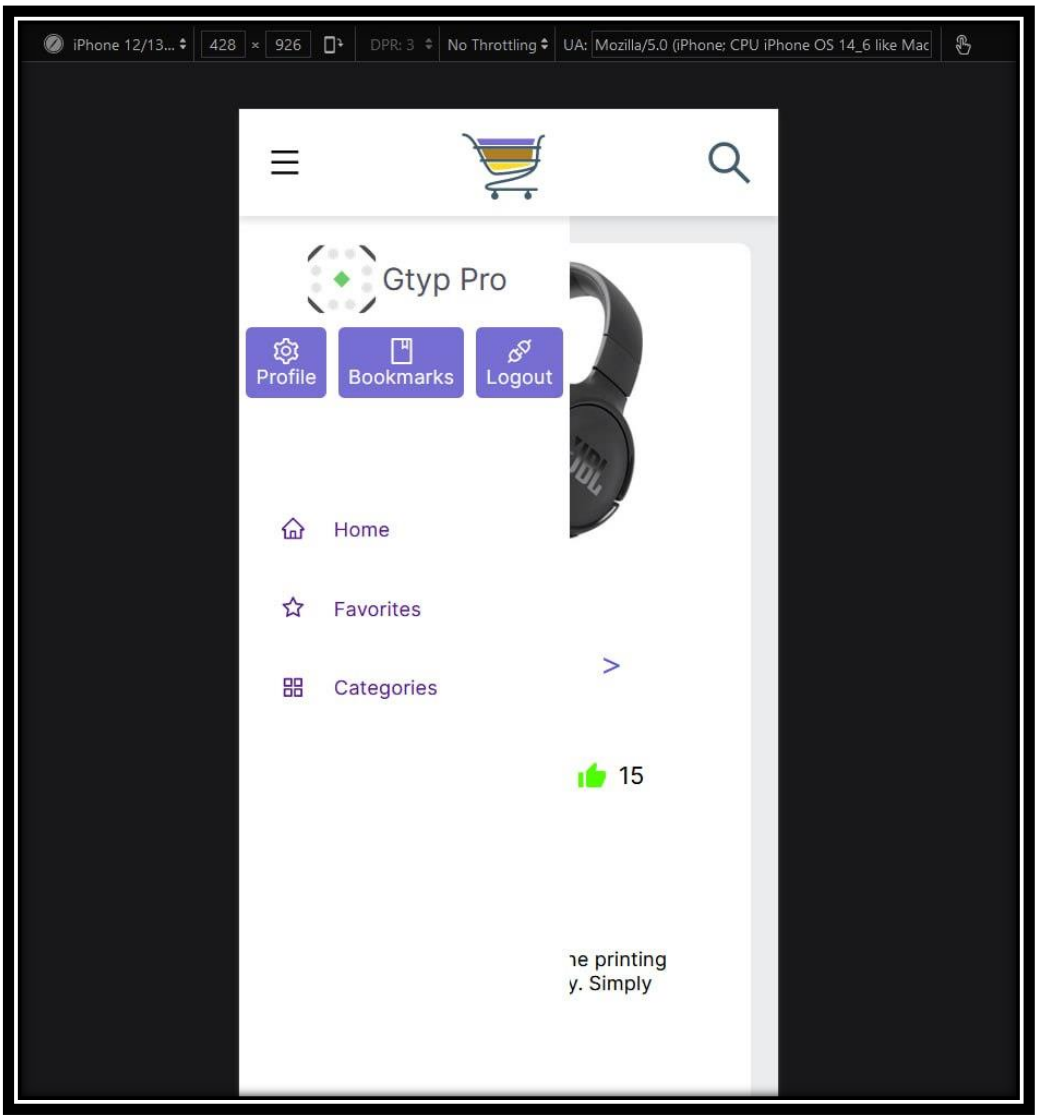

Рисунок 3.17 – Бічна панель

3.3 Висновки до розділу 3

У третьому розділі дослідження було проведено вичерпний опис програмної реалізації на кожному етапі проєкту. Це включало розробку як серверної, так і клієнтської частин, а також детальний аналіз архітектурних рішень для обох компонентів системи. Кожен етап реалізації був ретельно проаналізований, роз’яснені його деталі та обґрунтовані вибори конкретних рішень. Надані приклади виконання ілюстрували практичне застосування кожного етапу та виділяли ключові аспекти програмного проєкту. Додатково,

було розроблено та реалізовано систему CI/CD для автоматизованого тестування серверної частини, що значно сприяло забезпеченню якості продукту. Крім того, в розділі були надані фото реалізованого інтерфейсу, таблиці з прикладами запитів на сервер та API роутами, що дозволило зрозуміти та оцінити функціональні можливості програмного забезпечення більш детально.

ВИСНОВКИ

В процесі аналізу вимог та розгортання інтернет-магазину за допомогою TypeScript, MySQL, Knex, Objection, Express та Eslint було встановлено, що розроблений продукт відповідає поставленим завданням та вимогам замовника.

Найбільше вражає те, що навіть при відсутності вимог щодо мобільної адаптації, було розв’язане це питання. Крім того, додавання панелі адміністратора стало доповненням до функціонала.

Важливим етапом було ретельне вивчення та аналіз технічного завдання від потенційного замовника, що стосується створення інтернет-магазину. Застосовано технологічний стек, який виявився дієвим для реалізації поставлених завдань. Висвітлено ключові моменти, такі як валідація даних, ролі користувачів, CRUD-операції та коментарі до товарів.

Зазначаючи, що використовувались не лише теоретичні знання, але й практичні навички, кожен етап розробки було висвітлено докладно. Реалізація CI/CD, використання тестів та конкретні приклади коду в деталях демонструють гарні приклади автоматизації та забезпечення безпеки програмного забезпечення.

Оглядаючи роботу в цілому, можна зробити висновок, що створений інтернет-магазин відповідає вимогам, викладеним на етапі аналізу, і являє собою функціональний продукт, готовий задовольнити потреби користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Fast, unopinionated, minimalist web framework for Node.js. URL: https://expressjs.com/en/starter/installing.html (дата звернення: 09.02.2024).
2. React documentation. URL: https://react.dev/learn (дата звернення: 19.02.2024).
3. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 21.02.2024).
4. Knex.js. SQL query builder. URL: https://knexjs.org/guide/ (дата звернення: 05.03.2024).
5. TypeScript Documentation. URL: https://www.typescriptlang.org/docs/ (дата звернення: 11.03.2024).
6. ESLINT Documentation. URL: https://eslint.org/docs/latest/ (дата звернення: 27.03.2024).
7. Objection.js. Installation. URL: https://vincit.github.io/objection.js/guide/installation.html (дата звернення: 05.03.2024).
8. DiagramGPT. URL: https://www.eraser.io/diagramgpt (дата звернення: 12.03.2024).
9. GitHub Actions documentation. URL: https://docs.github.com/en/actions (дата звернення: 11.03.2024).
10. Node.js v21.5.0 documentation. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 14.03.2024).