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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення вебсайту інтернет-магазину музичних інструментів ”SoundKitchen”

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Давибіда В.Р.
(підпис) (прізвище та ініціали)

Керівник _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Загородна Н. В.
(підпис) (прізвище та ініціали)

Тернопіль
2025

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

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

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_27_» _січня _ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Давибіда Володимир Романович
(прізвище, ім'я, по батькові)
1. Тема роботи Створення вебсайту інтернет-магазину музичних інструментів "SoundKitchen"
Керівник роботи Матійчук Любомир Павлович, д-р екон. наук, проф кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*07*» *травня* 2025 року № *4/7-444*
2. Термін подання студентом завершеної роботи 22 червня 2025 р.
3. Вихідні дані до роботи Джерела з літератури та інтернету про створення клієнтської та серверної частин вебсайту.
4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області та постановка завдання на розробку. 1.1 Огляд існуючих рішень та їх аналіз. 1.2 Актуальність теми та опис предметної області. 1.3 Постановка завдання розробки програмної системи. 1.4 Висновки до першого розділу. 2 Теоретичні основи та вимоги до розробки програмної системи. 2.1 Загальні поняття та засади розробки програмного забезпечення. 2.1.1 Визначення підходів до розробки та методів дослідження для реалізації проєкту. 2.2 Розробка та визначення вимог до веб-платформи онлайн-магазину музичних інструментів. 2.3 Огляд та обґрунтування вибору стеку технологій для розробки вебсайту інтернет-магазину. 2.4 Висновки до другого розділу. 3 Проектування та реалізація інтернет-магазину "SoundKitchen". 3.1 Архітектура та реалізація програмної системи вебсайту. 3.1.1 Реалізація клієнтської частини системи. 3.1.2 Реалізація серверної частини системи. 3.2 Демонстрація інтерфейсу та функціональних можливостей створеного вебсайту. 3.3 Тестування та оцінка якості системи. 3.4 Висновки до третього розділу. 4 Безпека життєдіяльності, основи охорони праці. 4.1 Ергономічні проблеми безпеки життєдіяльності. 4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК. 4.3 Висновки до четвертого розділу. Висновки. Перелік джерел.Додаток А. Додаток Б.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Огляд існуючих рішень. 2. Актуальність обраної теми. 3. Мета, об'єкт та предмет дослідження. 4. Завдання на розробку вебсайту інтернет-магазину. 5. Теоретичні основи та вимоги до системи. 6. Діаграма варіантів використання. 7. Архітектура програмної системи. 8. Користувацький інтерфейс. 9. Корзина та процес оформлення замовлення. 10. Приклад оплати замовлення. 11. Порівняння можливостей існуючих рішень та розробленого вебсайту інтернет-магазину. 12. Тестування та оцінка якості розробленої системи. 13. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., кандидат технічних наук, доцент кафедри МТ		

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	27.01.2025	
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	08.05.2025-15.05.2025	
3.	Виконання дослідження щодо функціональних та нефункціональних вимог	16.05.2025-17.05.2025	
4.	Розроблення вебсайту інтернет-магазину	17.05.2025-22.05.2025	
5.	Оформлення розділу «Аналіз предметної області та постановка завдання на розробку»	23.05.2025-01.06.2025	
6.	Оформлення розділу «Теоретичні основи та вимоги до розробки програмної системи»	23.05.2025-01.06.2025	
7.	Оформлення розділу «Проектування та реалізація інтернет магазину “SoundKitchen”»	23.05.2025-01.06.2025	
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»	02.06.2025-05.06.2025	
9.	Виконання завдання до підрозділу «Основи охорони праці»	02.06.2025-05.06.2025	
10.	Оформлення кваліфікаційної роботи	06.06.2025-10.06.2025	
11.	Нормоконтроль	11.06.2025-13.06.2025	
12.	Перевірка на плагіат	16.06.2025	
13.	Попередній захист кваліфікаційної роботи	18.06.2025	
14.	Захист кваліфікаційної роботи	27.06.2025	

Студент _____ Давибіда В.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матійчук Л.П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення вебсайту інтернет-магазину музичних інструментів ”SoundKitchen” // Кваліфікаційна робота освітнього рівня «Бакалавр» // Давибіда Володимир Романович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СН-41 // Тернопіль, 2025 // С. 76, рис. – 19, табл. – 2, слайди – 13, додат. – 13, бібліогр. – 46.

Ключові слова: вебсайт, інтернет-магазин, електронна комерція, база даних, mongodb, javascript, nodejs, react.

Кваліфікаційна робота присвячена створенню вебсайту інтернет-магазину музичних інструментів “SoundKitchen”.

У першому розділі роботи обґрунтовано важливість розробки сучасних платформ електронної комерції, які забезпечують комфортний досвід користувачів під час вибору та купівлі музичних інструментів. Проаналізовано сучасні тенденції розвитку онлайн-торгівлі в музичній сфері та здійснено огляд аналогічних рішень на ринку.

У другому розділі розглянуто технічні особливості проєктування вебсайтів, зокрема сучасні підходи до побудови архітектури систем. Сформульовано функціональні й нефункціональні вимоги до вебсайту, а також обґрунтовано вибір технологічного стеку, зокрема мови програмування, фреймворків, бібліотек і систем управління базами даних.

Третій розділ зосереджений на практичній реалізації вебсайту: розробці його структури, функціональних можливостей і користувацького інтерфейсу. Описано механізми роботи системи, її зв’язок із базами даних, а також створення ключових функцій, таких як пошук і сортування товарів, управління кошиком і оформлення замовлень. Проведено тестування вебсайту та перевірено його відповідність визначеним критеріям функціональності.

ANNOTATION

Creation of the “SoundKitchen” Online Store for Musical Instruments // Qualification work of the educational level «Bachelor» // Davybida Volodymyr // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2025 // P. 76 , fig. – 19, tabl. – 2, slides – 13, annexes. – 13, references – 46.

Keywords: website, online store, e-commerce, database, mongodb, javascript, nodejs, react.

The qualification work is dedicated to creation of the website of the online store for musical instruments “SoundKitchen”.

The first chapter of the work justifies the importance of developing modern e-commerce platforms that provide a comfortable user experience when choosing and purchasing musical instruments. Current trends in the development of online commerce in the music industry are analysed and a review of similar solutions on the market is made.

The second chapter discusses the technical features of website design, including modern approaches to building system architecture. The functional and non-functional requirements for a website are formulated, and the choice of a technology stack, including programming languages, frameworks, libraries, and database management systems, is justified.

The third chapter focuses on the practical implementation of the website: the development of its structure, functionality and user interface. It describes the mechanisms of the system, its connection with databases, and the creation of key functions such as search and sorting of goods, cart management, and ordering. The website was tested and its compliance with the defined functionality criteria was verified.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – Інтерфейс програмування додатків.

CORS (Cross-Origin Resource Sharing) – Механізм обміну ресурсами між різними джерелами.

CSS (Cascading Style Sheets) – Каскадні таблиці стилів.

DRY (Don’t Repeat Yourself) – Принцип програмування, який передбачає уникнення дублювання коду шляхом створення модульних, багаторазових компонентів або функцій.

HTTPS (HyperText Transfer Protocol Secure) – Захищений протокол передачі гіпертексту.

JSON (JavaScript Object Notation) – Формат обміну даними.

JWT (JSON Web Token) – Токен для аутентифікації на основі JSON.

KISS (Keep It Simple, Stupid) – Принцип розробки, який наголошує на необхідності створення простих і зрозумілих рішень для уникнення надмірної складності.

REST (Representational State Transfer) – Архітектурний стиль для створення вебсервісів.

UI (User Interface) – Користувацький інтерфейс.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасних умовах електронна комерція набуває дедалі більшого значення, а попит на зручні онлайн-платформи для торгівлі зростає, зокрема в сегменті музичних інструментів. Музиканти різних рівнів, від початківців до професіоналів, потребують доступу до широкого вибору товарів, зрозумілого дизайну та надійного процесу оформлення замовлень. Однак багато існуючих інтернет-магазинів музичних інструментів мають недоліки, такі як обмежені можливості пошуку, складна навігація або відсутність функцій, що відповідають специфічним потребам аудиторії. Таким чином, розробка вебсайту інтернет-магазину “SoundKitchen” із сучасним інтерфейсом, адаптивною верстkою та спеціалізованими функціями є актуальним завданням, що відповідає потребам розвитку електронної комерції та вебтехнологій.

Мета і задачі дослідження. Метою кваліфікаційної роботи на здобуття ступеня бакалавра є розробка вебсайту інтернет-магазину музичних інструментів “SoundKitchen”, який забезпечує ефективну взаємодію користувачів із платформою під час вибору та купівлі товарів. Для реалізації цієї мети визначено такі завдання:

– дослідити сучасні тенденції у сфері онлайн-торгівлі музичними інструментами та проаналізувати поведінку користувачів;

– вивчити конкурентні рішення на ринку, оцінивши їхні переваги та недоліки з погляду функціоналу й дизайну;

– визначити функціональні та нефункціональні вимоги до вебсайту, враховуючи потреби цільової аудиторії;

– розробити архітектуру вебсайту з урахуванням принципів безпеки та зручності;

– створити програмний продукт, використовуючи сучасні інструменти веброзробки;

– провести тестування вебсайту за участю представників цільової аудиторії, зібрати зворотний зв’язок і перевірити відповідність системи встановленим вимогам.

Практичне значення одержаних результатів. Розроблений вебсайт “SoundKitchen” забезпечує зручний доступ до асортименту музичних інструментів, полегшуючи процес вибору та купівлі завдяки інтуїтивно зрозумілому інтерфейсу, системі фільтрації товарів, кошику та зручному оформленню замовлень. Платформа дозволяє користувачам швидко знаходити потрібні товари, отримувати детальну інформацію про них і здійснювати покупки з мінімальними зусиллями. Отримані результати можуть бути використані для створення інших спеціалізованих інтернет-магазинів, а також у навчальних програмах із веброзробки та дизайну користувацького досвіду.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ

1.1 Огляд існуючих рішень та їх аналіз

Ринок інтернет-магазинів музичних інструментів характеризується значною різноманітністю платформ, які пропонують широкий асортимент товарів для музикантів різного рівня підготовки – від початківців до професіоналів. Серед популярних рішень на українському ринку можна виділити такі платформи, як Muztorg [1], Jam [2], Soundmaster [3], LuxPRO [4] та Muzline [5]. Кожна з цих платформ має власні особливості, цільову аудиторію та підходи до організації функціоналу й дизайну, що впливають на користувацький досвід.

Платформа Muztorg позиціонується як частина великої дистриб'юторської мережі, що забезпечує доступ до оригінальних музичних інструментів від відомих виробників. Її сильними сторонами є широкий асортимент, гарантійне обслуговування та інформаційна підтримка клієнтів. Інтерфейс сайту зручний для пошуку товарів за категоріями, а також пропонує детальні описи продуктів. Однак дизайн виглядає дещо застарілим, а навігація може бути ускладненою для нових користувачів через велику кількість категорій і підкатегорій. Крім того, платформа не пропонує додаткових функцій, таких як порівняння характеристик інструментів чи інтеграція з навчальними ресурсами, що обмежує її привабливість для певної аудиторії. На рисунку 1.1 зображено головну сторінку Muztorg.

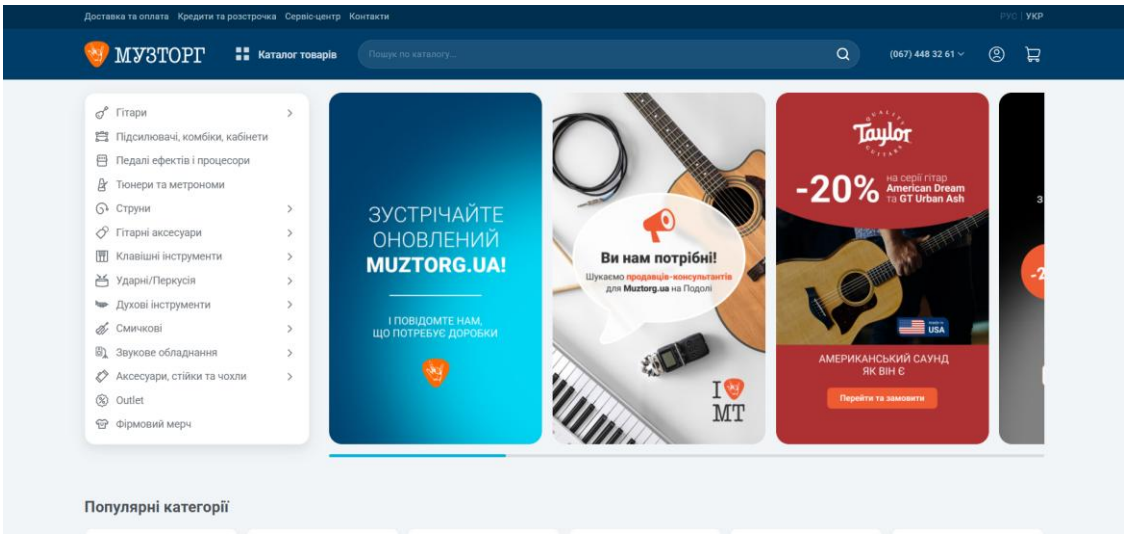

Рисунок 1.1 – Головна сторінка Muztorg

Інтернет-магазин Jam вирізняється співпрацею з численними партнерами по всій Україні, що дозволяє забезпечити швидку доставку та доступ до понад 9000 найменувань товарів. Платформа підтримує професійну консультацію для покупців, що є значною перевагою для новачків. Сайт має сучасний дизайн із чіткою структурою, але функціонал пошуку та фільтрації товарів обмежений, що може ускладнювати швидкий вибір інструментів за специфічними параметрами, наприклад, за типом деревини для гітар чи діапазоном клавіш для синтезаторів. На рисунку 1.2 представлено вигляд головної сторінки Jam.

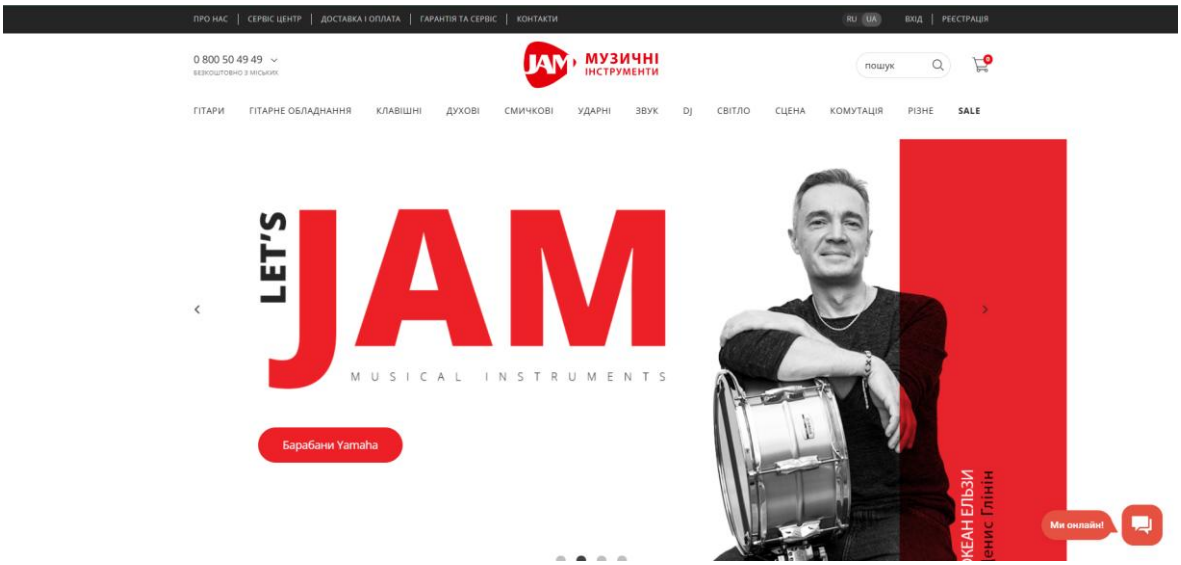

Рисунок 1.2 – Головна сторінка Jam

Soundmaster акцентує увагу на прямій співпраці з виробниками, що дозволяє пропонувати якісні інструменти для різних цілей – від навчальних до професійних. Сайт має зручний інтерфейс із можливістю швидкого перегляду популярних категорій, таких як гітари, клавішні чи перкусія. Проте відсутність розширених фільтрів, таких як сортування за технічними характеристиками, ускладнює процес вибору для користувачів, які шукають інструменти з конкретними параметрами. Крім того, дизайн сайту виглядає стандартним і не пропонує унікальних елементів, які б вирізняли його серед конкурентів. На рисунку 1.3 зображено головну сторінку Soundmaster.

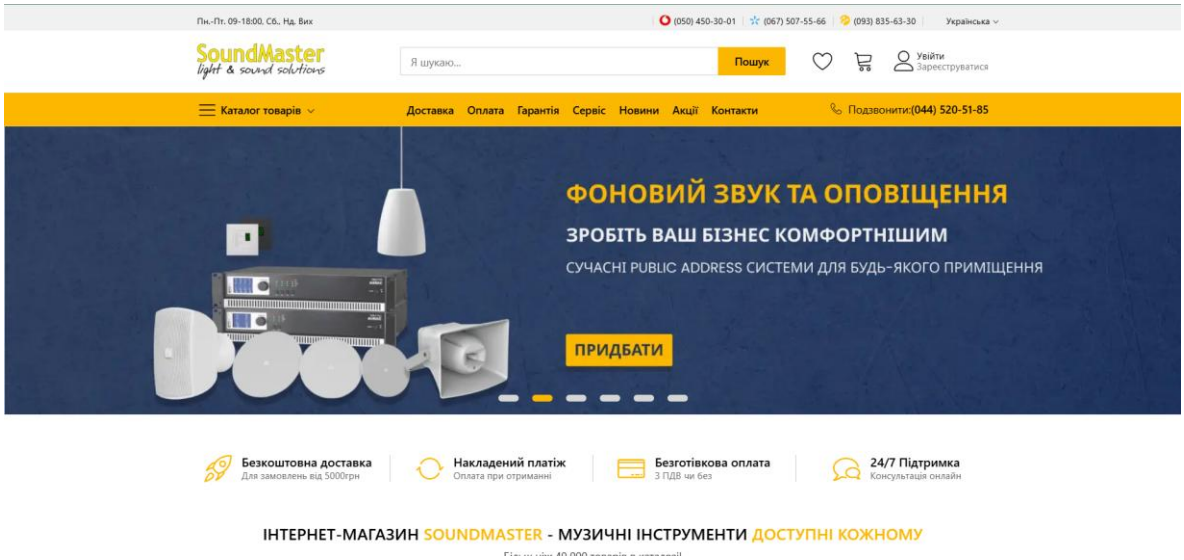

Рисунок 1.3 – Головна сторінка Soundmaster

LuxPRO пропонує широкий асортимент не лише музичних інструментів, а й супутнього обладнання, такого як підсилювачі, кабелі та аксесуари. Сайт вирізняється чіткою організацією каталогу та можливістю замовлення з доставкою по всій Україні. Однак платформа має обмежений функціонал для порівняння товарів, а також не пропонує додаткових сервісів, таких як рекомендації на основі вподобань користувача чи інтеграція з навчальними платформами. Інтерфейс є сучасним, але для деяких користувачів може здаватися перевантаженим через велику кількість рекламних банерів. На рисунку 1.4 представлено головну сторінку LuxPRO.

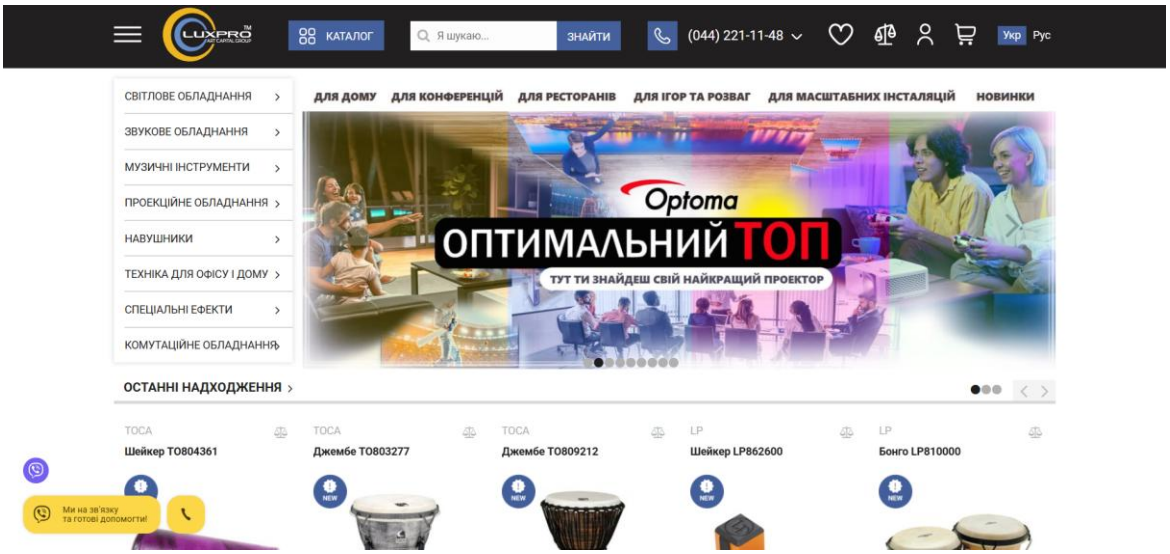

Рисунок 1.4 – Головна сторінка LuxPRO

Muzline позиціонується як лідер ринку з акцентом на доступні ціни та широкий вибір брендів. Сайт пропонує детальні описи товарів і статті про вибір інструментів, що є корисним для новачків. Проте платформа страждає від недостатньо інтуїтивної навігації, а її функціонал для оформлення замовлень може бути ускладненим через багатоступеневий процес. Дизайн сайту є сучасним, але не вирізняється унікальними рішеннями, що могло б покращити користувацький досвід. На рисунку 1.5 зображено вигляд головної сторінки Muzline.

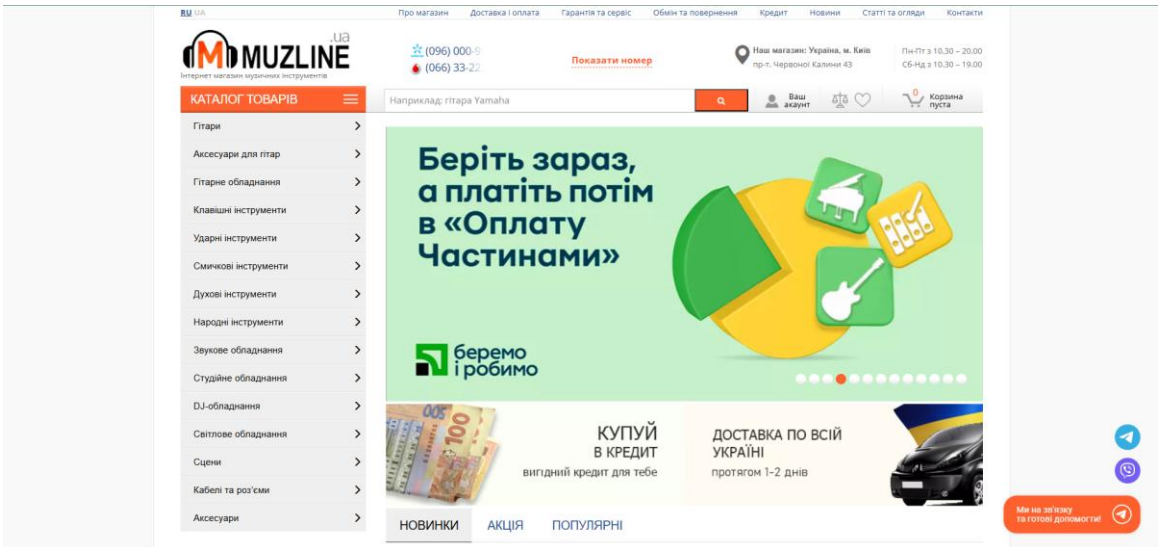

Рисунок 1.5 – Головна сторінка Muzline

Для ґрунтовного порівняння функціональних і дизайнерських особливостей проаналізованих платформ було розроблено таблицю 1.1. Цей аналіз дозволяє виявити ключові переваги та недоліки наявних рішень, а також визначити можливості для створення вебсайту “SoundKitchen”, який би відповідав потребам цільової аудиторії. Порівняння проводилося за основними критеріями, що є важливими для користувачів інтернет-магазинів музичних інструментів: асортимент, зручність навігації, функціонал пошуку та фільтрації, адаптивність дизайну, а також наявність додаткових сервісів, таких як рекомендації.

Таблиця 1.1 Порівняння функціональних і дизайнерських особливостей існуючих рішень та запропонованого вебсайту “SoundKitchen”

Категорія характеристик	**Muztorg, LuxPRO, Muzline**	**Jam, Soundmaster**
Асортимент	Широкий, включає інструменти та супутнє обладнання	Обмежений, з фокусом на популярні категорії
Зручність навігації	Середня, через складну структуру категорій	Висока, завдяки простішій організації
Функціонал пошуку та фільтрації	Обмежений, брак детальних технічних параметрів	Базовий, орієнтований на загальні категорії
Адаптивність дизайну	Часткова, не завжди оптимізована для мобільних пристроїв	Часткова, з обмеженнями на малих екранах
Додаткові сервіси	Консультації, статті, але без персоналізації	Обмежені, фокус на консультаціях

Продовження таблиці 1.1

Інтуїтивність оформлення замовлень	Середня, багатоступеневий процес	Середня, залежить від платформи
Швидкість завантаження	Середня, залежить від оптимізації	Середня, з періодичними затримками
Підтримка користувачів	Обмежена (телефон, email)	Обмежена, з фокусом на консультації

Аналіз існуючих рішень виявив низку проблем, які обмежують їхню ефективність для цільової аудиторії, що підтверджується сучасними рекомендаціями щодо дизайну вебсайтів для електронної комерції [6].

По-перше, більшість платформ мають обмежений функціонал пошуку та фільтрації, що ускладнює вибір інструментів за специфічними технічними характеристиками. Це особливо критично для професійних музикантів, які потребують детальної інформації для прийняття рішення. По-друге, дизайн багатьох платформ не є повністю адаптивним, що створює незручності при використанні на мобільних пристроях, попри зростання популярності мобільного шопінгу.

Крім того, більшість проаналізованих сайтів не пропонують додаткових сервісів, які могли б покращити користувацький досвід, наприклад, персоналізовані рекомендації на основі історії переглядів. Навігація часто є недостатньо інтуїтивною, а процес оформлення замовлень – надмірно складним через велику кількість кроків. Щодо конфіденційності, зберігання даних залежить від політики провайдерів, що може викликати занепокоєння у користувачів, які цінують безпеку.

Результати аналізу вказують на потребу у створенні вебсайту інтернет-магазину музичних інструментів, який би усунув зазначені недоліки. Ідеальний інструмент мав би пропонувати функціонал пошуку та фільтрації, адаптивний і сучасний дизайн, оптимізований для всіх типів пристроїв, а також додаткові сервіси, такі як рекомендації. Вебсайт “SoundKitchen” покликаний заповнити ці

прогалини, забезпечуючи зручний, безпечний і функціональний досвід для користувачів, що відповідає сучасним вимогам електронної комерції.

1.2 Актуальність теми та опис предметної області

У сучасному світі, де технології стрімко розвиваються, а обсяг інформації невпинно зростає, ефективне управління онлайн-торгівлею набуває вирішального значення для успіху бізнесу [7]. Особливо це стосується ніші музичних інструментів, де інтернет-магазин, такий як “SoundKitchen”, має справу з великим асортиментом товарів, обробкою замовлень і запитів клієнтів у найкоротші терміни. Така специфіка роботи вимагає від команди розробників, адміністраторів, менеджерів злагодженої організації, ретельного планування та постійного нагляду за виконанням завдань. Популярність онлайн-покупок зростає, а разом із нею змінюються й підходи до торгівлі, що спонукає фахівців у сфері електронної комерції впроваджувати сучасні інструменти для оптимізації процесів.

Недостатня ефективність у керуванні інтернет-магазином може призвести до серйозних наслідків: зниження продуктивності, перевантаження команди чи навіть зриву ключових термінів, наприклад, під час оновлення товарного каталогу або підготовки до рекламних акцій. Серед основних викликів цієї сфери потреба швидко адаптуватися до змін на ринку, підтримувати актуальність асортименту та забезпечувати якісне обслуговування клієнтів. Для вирішення цих завдань необхідні не лише чітка організація, а й інструменти, які допомагають систематизувати роботу та вчасно нагадувати про важливі етапи. У цьому контексті вебсайт “SoundKitchen” відіграє роль не лише торговельної платформи, а й цифрового помічника, який спрощує управління торгівлею завдяки функціям візуалізації асортименту, відстеження та контролю замовлень.

Управління інтернет-магазином охоплює комплекс процесів, методів та інструментів, спрямованих на організацію роботи для досягнення бізнес-цілей. Воно є основою як для управління проєктами, так і для координації команди, що

особливо важливо для фахівців електронної комерції, які щодня працюють із різноманітними завданнями. Етапи цього процесу [8] адаптовані до специфіки “SoundKitchen” і виглядають так:

– Визначення завдання: постановка чіткої мети й умов, наприклад, створення зручного фільтра для вибору гітар або оновлення переліку аксесуарів у каталозі.

– Встановлення пріоритетів: аналіз терміновості та значущості завдань, щоб ефективно розподілити зусилля, наприклад, приділити більше уваги підготовці до сезонних знижок.

– Планування: розробка плану дій із визначенням строків, ресурсів і відповідальних осіб, від програмістів до контент-менеджерів.

– Реалізація: виконання завдань, таких як налаштування сайту, перевірка його роботи чи додавання нових товарів.

– Контроль: спостереження за прогресом, виявлення проблем і коригування планів для вчасного завершення.

– Фіналізація: підведення підсумків, запуск оновлень (наприклад, нового дизайну сайту) та оцінка їх впливу.

Добре налагоджена система управління, яку пропонує “SoundKitchen”, допомагає впорядкувати робочі процеси та оптимізувати їх. Це сприяє зниженню стресу та дає змогу бачити повну картину роботи й перспективи розвитку. Великі завдання, як-от створення конкурентоспроможного магазину, розбиваються на зрозумілі кроки, прогрес стає очевидним, а реакція на зміни оперативною. Вебсайт поєднує в собі каталог музичних інструментів та систему управління замовленнями [9] та асортиментом товарів. Адміністратори й менеджери можуть легко переглядати, додавати та редагувати будь-які елементи асортименту. Це робить “SoundKitchen” чудовим рішенням для підвищення продажів в умовах сучасної онлайн-торгівлі.

1.3 Постановка завдання розробки програмної системи

На основі проведеного аналізу існуючих рішень та виявлених недоліків на ринку інтернет-магазинів музичних інструментів чітко визначено проблему, яка потребує вирішення в межах цього проєкту. Більшість наявних платформ не повною мірою відповідають потребам користувачів, зокрема через обмежений функціонал пошуку, неінтуїтивну навігацію та відсутність спеціалізованих функцій, таких як детальне порівняння характеристик інструментів чи інтеграція з навчальними ресурсами.

З огляду на це, проблемою, яку необхідно вирішити в межах даної роботи, є відсутність зручного, функціонального та адаптивного вебсайту інтернет-магазину музичних інструментів, який би забезпечував ефективну взаємодію користувачів із платформою, враховуючи специфічні потреби музикантів різних рівнів підготовки.

Метою розробки даної програмної системи є створення вебсайту інтернет-магазину “SoundKitchen”, який забезпечуватиме зручний доступ до асортименту музичних інструментів, інтуїтивний процес вибору та купівлі товарів, а також оптимізуватиме користувацький досвід завдяки сучасному дизайну та спеціалізованим функціям.

Для досягнення мети було окреслено наступні завдання для розробки:

1. Проаналізувати конкурентні рішення на ринку інтернет-магазинів музичних інструментів.

2. Розробити архітектуру вебсайту, що забезпечить його безпеку, зручність використання та можливість інтеграції додаткових функцій.

3. Реалізувати функціонал управління каталогом товарів, що включає створення, перегляд, редагування та видалення записів із зазначенням назви, опису і ціни.

4. Забезпечити зручну систему пошуку та фільтрації товарів за параметрами, такими як категорія, бренд та ціна.

5. Реалізувати функціонал кошика покупок, що дозволить додавати, переглядати, редагувати та видаляти товари перед оформленням замовлення.

6. Забезпечити зручний і швидкий процес оформлення замовлень із мінімальною кількістю кроків та підтримкою різних способів оплати.

7. Впровадити адаптивний дизайн інтерфейсу, що забезпечить коректне відображення вебсайту на пристроях із різними розмірами екранів.

8. Додати функціонал рекомендацій товарів на основі переглянутих або придбаних користувачем позицій.

9. Забезпечити базову безпеку даних користувачів, включаючи захист персональної інформації та безпечне оброблення транзакцій.

Для чіткого розуміння характеристик майбутньої платформи “SoundKitchen” розроблено таблицю, яка відображає заплановані функціональні та якісні особливості системи. Ці характеристики враховують недоліки конкурентів, описані в розділі 1.1, і спрямовані на створення конкурентоспроможного рішення, що відповідає сучасним вимогам електронної комерції.

Таблиця 1.2 характеристики веб-платформи “SoundKitchen”

Категорія характеристик	**Опис**
Асортимент	Широкий асортимент із детальними описами товарів і супутніх аксесуарів
Зручність навігації	Висока зручність завдяки чіткій та інтуїтивній організації каталогу
Функціонал пошуку та фільтрації	Розширений функціонал із фільтрацією за категоріями, знижками та кольором
Адаптивність дизайну	Повна адаптивність для коректного відображення на всіх пристроях.
Додаткові сервіси	Впровадження рекомендацій, системи відгуків і рейтингів.

Продовження таблиці 1.2

Інтуїтивність оформлення замовлень	Висока зручність із мінімальною кількістю кроків і підтримкою різних способів оплати
Швидкість завантаження	Висока швидкість завдяки оптимізації
Підтримка користувачів	Підтримка через email для швидкого вирішення запитів користувачів.

Ця таблиця відображає заплановані характеристики “SoundKitchen”, які враховують потреби цільової аудиторії та сучасні тенденції в електронній комерції. Вони стануть основою для проєктування та реалізації системи, що усуне недоліки конкурентних рішень і забезпечить зручний, безпечний і функціональний користувацький досвід.

1.4 Висновки до першого розділу

Перший розділ кваліфікаційної роботи присвячений аналізу поточного стану ринку інтернет-магазинів, що спеціалізуються на музичних інструментах. У рамках цього дослідження обґрунтовано доцільність розробки вебсайту “SoundKitchen” і визначено ключові завдання для його створення. Аналіз показав, що сфера електронної комерції у сегменті музичних інструментів демонструє активний розвиток, проте існуючі платформи часто мають низку недоліків, серед яких обмежений функціонал пошуку, складна навігація та недостатня адаптованість до специфічних потреб цільової аудиторії, що включає музикантів із різними рівнями підготовки.

Дослідження рішень, які пропонують платформи Muztorg [1], Jam [2], Soundmaster [3], LuxPRO [4] та Muzline [5], дозволило виявити як їхні переваги, так і недоліки. Основними проблемами таких сервісів є відсутність розширених фільтрів для пошуку товарів, обмежена адаптивність інтерфейсу для мобільних пристроїв та недостатня кількість додаткових послуг, зокрема персоналізованих

рекомендацій. Ці недоліки відкривають можливості для створення більш функціонального і зручного рішення.

Огляд предметної області висвітлив значення ефективного управління інтернет-магазином, яке включає організацію асортименту, обробку замовлень і забезпечення високого рівня зручності для користувачів. Вебсайт “SoundKitchen” задумано як платформу, що об’єднає зручний каталог товарів із продуманою системою управління замовленнями, спрямовану на оптимізацію бізнес-процесів.

На основі проведеного аналізу сформовано завдання для розробки програмної системи, до яких належать створення адаптивного дизайну, реалізація функцій пошуку, фільтрації, кошика, оформлення замовлень, рекомендацій товарів і забезпечення безпеки даних користувачів. Виконання цих завдань дозволить розробити вебсайт, що вирішить проблеми існуючих платформ і відповідатиме сучасним вимогам електронної комерції.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ ТА ВИМОГИ ДО РОЗРОБКИ ПРОГРАМНОЇ СИСТЕМИ

2.1 Загальні поняття та засади розробки програмного забезпечення

Створення програмного забезпечення – це складний процес, який включає розробку, перевірку та обслуговування систем, що вирішують конкретні задачі користувачів. У випадку створення вебресурсу для інтернет-магазину музичних інструментів “SoundKitchen” основною метою є розробка зручної, надійної та ефективної платформи, яка відповідатиме сучасним стандартам електронної комерції та задовольнить потреби музикантів різного рівня. У цьому розділі розглянуто ключові поняття та принципи, на яких базується розробка програмного забезпечення для реалізації такого проєкту.

Розробка програмного забезпечення – це процес, що охоплює аналіз потреб користувачів, створення проєктної документації, написання коду, тестування [10] та забезпечення подальшого супроводу. У контексті вебсайту “SoundKitchen” це передбачає розробку платформи, яка забезпечує зручний доступ до вибору музичних інструментів, інтуїтивний пошук і оформлення замовлень, а також надійний захист даних клієнтів. Основні якості ефективного програмного забезпечення включають функціональність, надійність, продуктивність та гнучкість до змін. У рамках цього проєкту досягнення цих характеристик забезпечується завдяки продуманій архітектурі [11], що розділяє функції між клієнтською і серверною частинами, полегшуючи розробку та управління системою.

Для створення “SoundKitchen” доцільно застосовувати гібридну модель життєвого циклу [12], яка поєднує елементи каскадного та ітераційного підходів [13]. Каскадний підхід забезпечує чітку послідовність етапів:

– Збір вимог: визначення потреб користувачів, таких як зручна фільтрація товарів за брендом, ціною чи категорією, та забезпечення адаптивного дизайну.

– Системне проєктування: розробка архітектури, яка включає клієнтську частину (інтерфейс), серверну частину (обробка запитів) і базу даних для зберігання інформації про товари та замовлення.

– Реалізація: створення програмного коду з використанням сучасних технологій [14], таких як, JavaScript, NodeJs, React, Tailwind та інші для інтерфейсу та серверної частини відповідальної за обробку данних.

– Тестування: перевірка функціональності компонентів, таких як кошик, система пошуку та оформлення замовлень, а також сумісності з різними пристроями.

– Розгортання: запуск вебсайту на сервері для доступу користувачів.

– Технічна підтримка: моніторинг роботи системи, виправлення помилок і додавання нових функцій.

Ітераційний процес розробки забезпечує можливість поліпшення системи, враховуючи відгуки користувачів, що сприяє адаптації платформи до вимог і очікувань її цільової аудиторії.

Важливі принципи якісного кодування:

– DRY [15]. Повторювану логіку слід виносити в окремі модулі або компоненти. У проєкті інтернет-магазину це може бути окрема бібліотека для роботи з корзиною чи обробки замовлень.

– KISS [16]. Простота реалізації зменшує ризик помилок і полегшує підтримку. Не варто впроваджувати складні рішення там, де достатньо стандартних можливостей фреймворку.

Також важливим аспектом є управління якістю та ризиками. Контроль якості включає як автоматизоване тестування, так і ручне. Для “SoundKitchen” важливе тестування платежів, коректності відображення каталогу товарів. Оцінка і мінімізація ризиків, регулярний перегляд архітектури, аналіз вразливостей, план відновлення після збоїв, беззаперечно є невідємними складовими життєвого циклу хорошої програмної системи.

2.1.1 Визначення підходів до розробки та методів дослідження для реалізації проєкту

На етапі підготовки до створення вебсайту інтернет-магазину музичних інструментів “SoundKitchen” ключовим завданням є визначення оптимального підходу до розробки та методів дослідження, які забезпечать ефективну реалізацію проєкту. Вибір методології та інструментів ґрунтується на потребах цільової аудиторії – музикантів різного рівня підготовки, а також на вимогах до функціональності, безпеки та зручності платформи, сформульованих у попередніх розділах.

Для розробки “SoundKitchen” планується застосувати гнучку методологію, засновану на принципах Agile [17], із використанням ітеративного підходу. Такий вибір обумовлений кількома факторами:

– необхідність створення прототипів для раннього тестування ключових функцій, таких як пошук товарів чи оформлення замовлень;

– можливість адаптації до змін вимог, що можуть виникнути під час аналізу зворотного зв’язку від потенційних користувачів;

– ефективність для проєктів із динамічними вимогами, де пріоритет надається швидкому отриманню працюючого продукту;

– акцент на створенні цінності для користувачів, що відповідає меті проєкту – забезпечити зручний і функціональний досвід покупки музичних інструментів.

Гнучка методологія дозволить розробляти платформу короткими циклами, що забезпечить можливість раннього виявлення недоліків і коригування функціоналу. Наприклад, на початкових етапах можна буде протестувати систему фільтрації товарів або кошик, щоб переконатися в їхній зручності. На відміну від каскадної моделі, яка передбачає жорстку послідовність етапів і ускладнює внесення змін, Agile [17] забезпечує гнучкість і фокус на ключових функціях, що є критичним для створення конкурентоспроможної платформи.

Запланований процес розробки складатиметься з ітеративних циклів, що охоплюють такі етапи:

Планування: постановка цілей ітерації, визначення пріоритетних завдань (наприклад, розробка каталогу товарів чи системи аутентифікації) та оцінка ресурсів.

– Аналіз і проєктування: уточнення вимог, розробка ескізів інтерфейсу та схем архітектури, таких як структура API чи моделі даних.

– Реалізація: написання коду з дотриманням принципів модульності (DRY [15]) та простоти (KISS [16]) для полегшення подальшої підтримки.

– Тестування: перевірка працездатності компонентів, їхньої сумісності з різними пристроями та відповідності вимогам безпеки.

– Оцінка: збір відгуків від потенційних користувачів для вдосконалення функціоналу перед наступною ітерацією.

Цей підхід дозволить поступово створювати платформу, адаптуючи її до потреб користувачів і сучасних стандартів електронної комерції.

Для обґрунтування технічних рішень і забезпечення якості проєкту планується застосувати набір наукових методів дослідження, які сприятимуть структурованому підходу до розробки.

Системний аналіз буде використано для розкладання проєкту на складові елементи та визначення їхніх взаємозв’язків. Цей метод дозволить представити “SoundKitchen” як систему, що складається з клієнтської частини (інтерфейс на React [18]), серверної частини (Node.js [19], Express) і бази даних (MongoDB [20]). Системний аналіз допоможе спроєктувати модулі, такі як каталог, кошик чи система рекомендацій, і забезпечити їхню ефективну взаємодію.

Порівняльний аналіз застосовуватиметься для вибору технологій і оцінки аналогів, таких як платформи, описані в розділі 1.1. Метод передбачає порівняння характеристик інструментів (наприклад, продуктивність React [18] порівняно з Vue.js, переваги MongoDB [20] над MySQL) та їхньої відповідності

вимогам проєкту. Це дозволить обґрунтувати використання JavaScript-стеку для забезпечення гнучкості й продуктивності платформи.

Емпіричний метод передбачає проведення апробації прототипів серед представників цільової аудиторії. На етапі підготовки планується організувати опитування та тестування макетів інтерфейсу для оцінки зручності навігації, фільтрації товарів і оформлення замовлень. Зібрані дані допоможуть уточнити вимоги до дизайну та функціоналу.

Метод проєктування буде використано для створення концептуальної та детальної архітектури системи. На концептуальному рівні визначатимуться принципи роботи платформи, наприклад, використання RESTful API [21] для зв’язку між фронтендом і бекендом. На детальному рівні розроблятимуться схеми компонентів (наприклад, Navigation, Product) і структури даних (Mongoose-схеми). Використання принципів модульності та дизайн-мислення забезпечить гнучкість системи та інтуїтивність інтерфейсу.

Отже, поєднання гнучкої Agile-методології з ітеративним підходом і науковими методами (системний і порівняльний аналіз, емпіричне тестування, проєктування) створює міцну основу для підготовки до розробки вебсайту “SoundKitchen”. Цей підхід забезпечить створення платформи, яка відповідатиме потребам користувачів і сучасним вимогам електронної комерції.

2.2 Розробка та визначення вимог до веб-платформи онлайн-магазину музичних інструментів

Чітке формулювання вимог до програмної системи є основою для успішного проєктування, розробки та тестування веб-платформи. Вимоги визначають, які функції та якості повинна мати система, щоб відповідати потребам користувачів і бізнес-цілям. У контексті підготовки до створення вебсайту інтернет-магазину музичних інструментів “SoundKitchen” вимоги поділяються на функціональні, що описують конкретні можливості платформи, та нефункціональні, що визначають якісні характеристики системи.

Функціональні вимоги визначають поведінку системи та її можливості, які забезпечують взаємодію користувачів із платформою. Для “SoundKitchen” ці вимоги розроблені з урахуванням потреб цільової аудиторії, описаних у розділі 1.1, та проблематики існуючих рішень, виявленої в аналізі конкурентів.

Управління каталогом товарів: Система повинна забезпечувати можливість перегляду, пошуку, фільтрації та сортування товарів (музичних інструментів і супутнього обладнання). Користувачі зможуть обирати товари за категоріями (наприклад, гітари, клавішні, перкусія), діапозоном знижки або кольором. Кожен товар матиме детальну сторінку з описом, зображеннями, ціною та відгуками, а також супутніми рекомендаціями.

Кошик покупок: Платформа передбачатиме функціонал кошика, що дозволяє додавати, редагувати, видаляти товари та переглядати їхній перелік перед оформленням замовлення. Користувач зможе бачити загальну суму замовлення з урахуванням знижок або акцій, ціну доставки, а також загальну суму знижки.

Оформлення замовлень: Система забезпечить швидкий і зручний процес оформлення замовлень із мінімальною кількістю кроків. Користувачі зможуть вказувати дані для доставки, обирати спосіб оплати (наприклад, карткою, банківським переказом) і отримувати підтвердження замовлення.

Аутентифікація та управління профілем: Платформа надаватиме можливість реєстрації та входу в особистий кабінет, де користувачі зможуть зберігати персональні дані, переглядати історію замовлень.

Система рекомендацій: На основі декількох вхідних параметрів система пропонуватиме персоналізовані рекомендації кожного товару, наприклад, схожі за ціною чи кольором аналоги та аксесуари.

Відгуки та рейтинги: Користувачі матимуть змогу залишати та переглядати відгуки та оцінки для товарів, що допоможе іншим клієнтам приймати рішення про покупку.

Адміністративні функції: Адміністратори платформи зможуть додавати, редагувати та видаляти товари, керувати замовленнями, а також переглядати статистику продажів.

На рисунку 2.1 представлено діаграму варіантів використання (Use Case Diagram), яка відображає основні сценарії взаємодії користувачів і адміністраторів із системою.

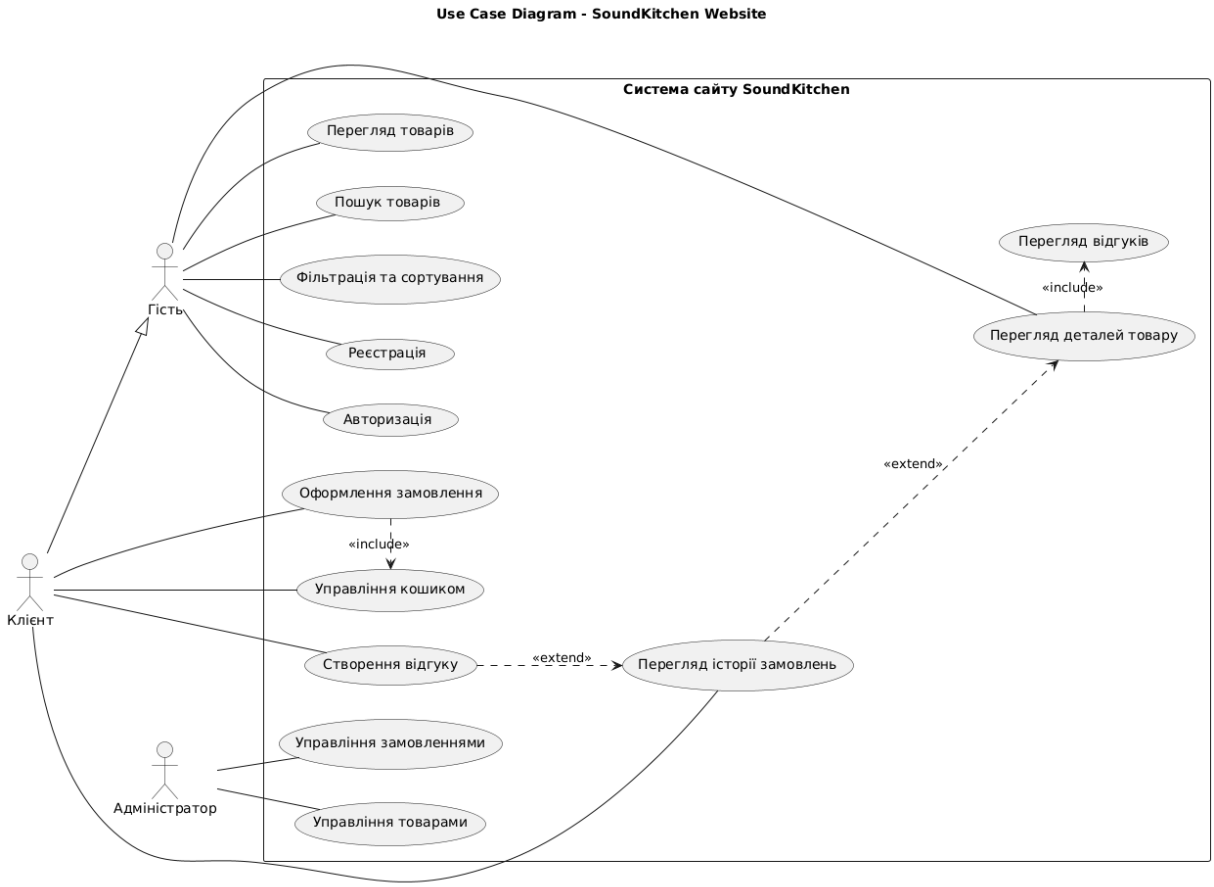

Рисунок 2.1 – Діаграма варіантів використання веб-платформи “SoundKitchen”

Ця діаграма є відображенням порядку взаємодії в реальному часі, і представлена за стандартами Unified Modeling Language [22]. Вона показує акторів: користувачів, адміністраторів та інших учасників, нових або вже зареєстрованих, з центральними функціями призначення системи, які входять до складу платформи також: пошук-механізм пошуку, замовлення-пошук та каталог.

Нефункціональні вимоги, у свою чергу, складають важливу складову роботи над розробкою будь-якої сучасної системи. На відміну від решти, вони являють собою перелік якісних характеристик, що найбільше позитивно або негативно впливає на ефективність, стабільність, зручність використання та загальне відчуття своєї аудиторії в роботі тією чи іншою платформою. Щодо “SoundKitchen”, вимоги цієї категорії були визначені з урахуванням новітніх тенденцій у сфері електронної комерції, а також потреб та інтересів цільової аудиторії.

– Продуктивність: Платформа повинна забезпечувати максимально швидке завантаження сторінок, тривалість якого не перевищує 2–3 секунд навіть за стандартних умов використання. Важливо, щоб система миттєво реагувала на дії користувачів, такі як пошук товарів, додавання їх до кошика чи виконання інших операцій. Це стало можливим завдяки оптимізації запитів до бази даних та інтеграції сучасних ефективних технологій, серед яких noSQL [23] база даних та RESTful API [21].

– Надійність: Одна з головних вимог до платформи полягає в її стабільності роботи. Система повинна функціонувати без збоїв і забезпечувати коректну обробку помилок. Наприклад, у разі введення некоректних даних користувачем платформа має відображати інформативні повідомлення про помилки. Також важливо, щоб навіть у випадку технічних збоїв зберігалася інформація про незавершені замовлення.

– Зручність використання: Розробка інтерфейсу орієнтована на інтуїтивне сприйняття користувачами. Простота у використанні, чітка та логічна навігація, доступність до ключових функцій – усе це забезпечує зручний користувацький досвід. Наприклад, функція фільтрації товарів і оформлення замовлень реалізована таким чином, щоб користувачі могли виконати ці дії в кілька кліків.

– Адаптивність: Завдяки впровадженню адаптивного дизайну платформа буде коректно відображатися на широкому спектрі пристроїв – від настільних комп’ютерів до планшетів і смартфонів. Для цього використовуються передові

інструменти, що дозволяють забезпечити максимально комфортне використання платформи незалежно від типу пристрою.

– Безпека: Захист персональних даних і конфіденційність користувачів є пріоритетом. Для цього платформа використовує такі методи, як HTTPS-протокол, хешування паролів із застосуванням бібліотек, а також аутентифікацію з допомогою токенів. NoSQL база даних побудована з урахуванням найсучасніших вимог до захисту від несанкціонованого доступу.

– Модульність і масштабованість: Завдяки використанню компонентного підходу у розробці і багатошаровій архітектурі платформа має високу гнучкість і можливість масштабування. Це дозволить легко додавати нові функції чи вдосконалювати вже існуючі, наприклад, розширювати систему рекомендацій.

– Локалізація: Відповідно до очікувань користувачів, інтерфейс буде локалізовано українською мовою, що дозволить забезпечити високу залученість цільової аудиторії.

Таким чином, функціональні та нефункціональні вимоги до веб-платформи “SoundKitchen” формують основу для створення сучасного, безпечного та ефективного рішення, яке відповідає потребам ринку електронної комерції. Усі описані аспекти спрямовані на створення зручного інструмента, який задовольнить потреби користувачів і сприятиме популярності проєкту.

На кожному етапі розробки – від аналізу вимог до тестування готової системи – ці критерії будуть служити орієнтиром для створення високоякісного продукту. Вони дозволяють закласти фундамент для довготривалого розвитку, зберігаючи конкурентоспроможність і відповідаючи потребам аудиторії.

2.3 Огляд та обґрунтування вибору стеку технологій для розробки вебсайту інтернет-магазину

Вибір стеку технологій є визначальним етапом підготовки до створення вебсайту, оскільки від цього залежить продуктивність, зручність розробки, масштабованість і якість користувацького досвіду. Сучасні технології

веброзробки пропонують широкий вибір інструментів, кожен із яких має свої сильні сторони та обмеження. Для забезпечення відповідності функціональним і нефункціональним вимогам, описаним у розділі 2.2, необхідно ретельно оцінити доступні технології та обрати ті, що найкраще відповідають потребам проєкту.

Для створення клієнтської частини вебсайту планується використовувати React [18] – бібліотеку JavaScript, яка вирізняється гнучкістю та ефективністю завдяки компонентному підходу. React дозволяє створювати модульні інтерфейси, де кожен елемент, такий як картка товару чи панель навігації, є незалежним компонентом. Це спрощує розробку складних сторінок, таких як каталог товарів чи кошик, і забезпечує швидке оновлення інтерфейсу завдяки віртуальному DOM(Об'єктна Модель Документа від англ. Document Object Model). Для управління станом, наприклад, для відстеження вмісту кошика чи профілю користувача, буде використано Redux [24], який забезпечує централізоване зберігання даних і передбачуваність поведінки системи. React Router [25], у свою чергу, відповідатиме за навігацію між сторінками, такими як домашня сторінка чи сторінка результатів пошуку, забезпечуючи плавний перехід без перезавантаження.

Стилізація інтерфейсу “SoundKitchen” базуватиметься на комбінації Tailwind CSS[26] і Material UI. Tailwind CSS, як утилітарний CSS-фреймворк, дозволяє швидко створювати адаптивний дизайн із високим рівнем кастомізації, що відповідає вимозі зручності використання на різних пристроях — від настільних комп’ютерів до смартфонів. Material UI доповнює Tailwind готовим набором компонентів, які забезпечують єдиний стиль і прискорюють розробку таких елементів, як кнопки чи форми. Ці інструменти разом створюють сучасний і інтуїтивно зрозумілий інтерфейс, локалізований українською мовою, що відповідає потребам цільової аудиторії.

Для серверної частини обрано Node.js [19] із фреймворком Express. Node.js дозволяє використовувати JavaScript як єдину мову для фронтенду та бекенду, що спрощує розробку та інтеграцію компонентів. Express забезпечує створення швидкого та масштабованого RESTful API, яке оброблятиме запити, такі як

пошук товарів, оформлення замовлень чи аутентифікація користувачів. Асинхронна модель Node.js ідеально підходить для обробки великої кількості одночасних запитів, що є важливим для інтернет-магазину з потенційно високим навантаженням. Цей вибір відповідає вимогам продуктивності та модульності, дозволяючи легко додавати нові функції, наприклад, систему рекомендацій.

Для зберігання даних, наприклад інформації про товари, замовлення чи відгуки, планується використовувати MongoDB із бібліотекою Mongoose, як NoSQL-база даних, пропонує гнучкість у роботі з неструктурованими даними, що ідеально підходить для каталогу музичних інструментів. Mongoose спрощує моделювання даних, надаючи зручні схеми для роботи з MongoDB, що полегшує інтеграцію з Node.js і забезпечує швидку обробку запитів, відповідно до вимог продуктивності.

Безпека платформи буде забезпечена за допомогою JWT для аутентифікації користувачів та bcrypt[30] для хешування паролів. Протокол HTTPS захистить передачу даних, а CORS[31] забезпечить коректну обробку міжсайтових запитів. Ці інструменти відповідають нефункціональній вимозі безпеки, гарантуючи захист персональних даних і транзакцій користувачів. Для взаємодії фронтенду з бекендом використовуватиметься бібліотека Axios[28], яка відповідає за зручне виконання API-запитів, що сприяє швидкості та стабільності системи.

Порівняно з альтернативами, такими як Vue.js чи Angular для фронтенду, React[18] вирізняється ширшою екосистемою та великою підтримкою спільноти, що значно спрощує пошук готових рішень та розширення функціоналу. Для бекенду розглядалися Django (Python) і Laravel (PHP), але Node.js із Express виявився найбільш оптимальним завдяки швидкості, легкості інтеграції з JavaScript-фронтендом та підтримці асинхронних операцій. При виборі між MySQL та PostgreSQL, було визначено що реляційні бази даних, поступаються NoSQL у гнучкості для роботи з неструктурованими даними, що є критичним для каталогу товарів із різноманітними атрибутами.

Обраний технологічний стек, включає такі інструменти, як React, Redux, React Router, Tailwind CSS, Material UI, Node.js, Express, MongoDB із Mongoose, JWT, bcrypt, Axios і CORS, є ретельно продуманим рішенням, спрямованим на досягнення оптимального балансу між різними аспектами веб-розробки. Цей набір технологій дозволяє створити ефективне середовище для розробки, забезпечуючи не лише високу продуктивність, але й надійну безпеку, гнучкість адаптації до змін і виняткову зручність роботи як для розробників, так і для кінцевих користувачів.

2.4 Висновки до другого розділу

Другий розділ кваліфікаційної роботи присвячений теоретичній підготовці до розробки вебсайту інтернет-магазину музичних інструментів “SoundKitchen” із застосуванням сучасних принципів і методологій створення програмного забезпечення. У рамках дослідження було здійснено комплексний аналіз ключових аспектів, необхідних для створення функціональної, безпечної та зручної платформи, що відповідає потребам цільової аудиторії — музикантів різного рівня підготовки.

На початку розділу розглянуто загальні засади розробки програмного забезпечення, обґрунтовано вибір гнучкої ітеративної моделі на основі Agile-принципів [17]. Цей підхід обрано через його здатність забезпечувати швидке створення прототипів, адаптацію до змін вимог і орієнтацію на цінність для користувача, що є критичним для проєктів у сфері електронної комерції. Використання системного, порівняльного, емпіричного аналізу та методу проєктування дозволило чітко визначити підходи до розробки та методи дослідження, які ляжуть в основу створення платформи.

Далі було сформульовано функціональні та нефункціональні вимоги до вебсайту “SoundKitchen”. Функціональні вимоги охоплюють ключові можливості системи, такі як управління каталогом товарів, кошик покупок, оформлення замовлень, аутентифікація, система рекомендацій, відгуки та

адміністративні функції. Нефункціональні вимоги визначають якісні характеристики, зокрема продуктивність, надійність, зручність використання, адаптивність, безпеку, модульність і локалізацію. Ці вимоги створюють чітке бачення кінцевого продукту, що відповідає сучасним стандартам і очікуванням користувачів.

На основі аналізу сучасних технологій веброзробки обґрунтовано вибір технологічного стеку для “SoundKitchen”. Для клієнтської частини обрано React [18] із Redux [24] і React Router [25] для створення динамічного та інтуїтивного інтерфейсу, а також Tailwind CSS [26] і Material UI для адаптивного дизайну. Серверна частина базуватиметься на Node.js [19] із фреймворком Express і MongoDB [20] із Mongoose для ефективної обробки даних. Безпека забезпечуватиметься через JWT [29], bcrypt [30], HTTPS і CORS, а взаємодія між фронтендом і бекендом — через Axios [28]. Обраний стек відповідає вимогам продуктивності, безпеки, модульності та масштабованість, а також спрощує розробку завдяки єдиній мові програмування (JavaScript).

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНТЕРНЕТ МАГАЗИНУ “SOUNDKITCHEN”

3.1 Архітектура та реалізація програмної системи вебсайту

На етапі підготовки до створення вебсайту “SoundKitchen” ключовим завданням є розробка архітектури системи, яка забезпечить її ефективність, модульність і відповідність вимогам, сформульованим у розділі 2.2. Архітектура системи базується на модульному підході, що передбачає чіткий розподіл відповідальності між компонентами, спрощуючи процес розробки, тестування та подальшого масштабування. Для забезпечення гнучкості та стабільності застосовується багатошарова архітектура, яка розділяє клієнтську частину, серверну логіку та управління даними.

Архітектура “SoundKitchen” ґрунтується на клієнт-серверній моделі, де клієнтська частина (фронтенд) відповідає за взаємодію з користувачем, а серверна частина (бекенд) обробляє запити та забезпечує роботу з даними. Система спроєктована з урахуванням принципів RESTful [21] дизайну, що забезпечує ефективну взаємодію між фронтендом і бекендом через API. Такий підхід відповідає вимогам модульності, продуктивності та безпеки, описаним у розділі 2.2.

Система поділяється на три основні шари:

1. Рівень представлення (фронтенд): відповідає за створення графічного інтерфейсу користувача, відображення інформації (наприклад, каталогу товарів, кошика) та обробку дій користувача (пошук, вибір товарів, оформлення замовлень).

2. Рівень бізнес-логіки (бекенд): обробляє запити від фронтенду, виконує обчислення (наприклад, розрахунок вартості замовлення з урахуванням знижок) і керує взаємодією між інтерфейсом і базою даних.

3. Рівень даних: забезпечує зберігання, отримання, оновлення та видалення інформації про товари, замовлення, користувачів і відгуки.

Ця структура дозволяє незалежно розробляти та вдосконалювати кожен шар, що спрощує інтеграцію нових функцій, таких як персоналізовані рекомендації чи адміністративні інструменти.

Для реалізації “SoundKitchen” планується використати повноцінний JavaScript-стек, описаний у розділі 2.3, що включає React [18] для фронтенду, Node.js [19] із Express для бекенду та MongoDB [20] із Mongoose для управління даними. Структура проєкту чітко розділена на клієнтську та серверну частини, відображаючи модульний підхід.

Фронтенд (soundkitchen):

– public/: міститиме статичні файли, такі як index.html (точка входу) та favicon (маленьке зображення, яке відображається поруч із веб-адресою в адресному рядку браузера).

– src/: основний вихідний код:

– App.js: головний компонент, що об’єднуватиме сторінки та маршрути.

– index.js: точка входу для ініціалізації React-додатку.

– customer/:

– components/: багаторазові UI-компоненти, такі як Cart, Product, Navigation, OrderDetails.

– pages/: компоненти сторінок, наприклад, HomePage, SearchResultsPage, ProductPage.

– Routers/: конфігурація маршрутів із React Router [25] для навігації.

– State/: управління глобальним станом через Redux [24] для збереження даних кошика чи профілю користувача.

– utils/: утилітарні функції.

Бекенд (soundkitchen_api):

– src/: вихідний код серверної частини:

– index.js: налаштування Express-додатку.

– server.js: ініціалізація сервера та підключення до бази даних MongoDB[20].

– config/: конфігураційні файли для бази даних, JWT [29] і платіжних систем.

– controller/: контролери для обробки API-запитів (наприклад, для пошуку товарів чи оформлення замовлень).

– middleware/: проміжні обробники для аутентифікації (JWT) і CORS.

– models/: Mongoose-схеми для даних (товари, замовлення, користувачі).

– routes/: маршрути API, такі як /products, /orders і тд.

– services/: бізнес-логіка, наприклад, для фільтрації товарів чи створення рекомендацій.

На рисунку 3.1 зображено планову структуру файлів і папок проєкту “SoundKitchen”.

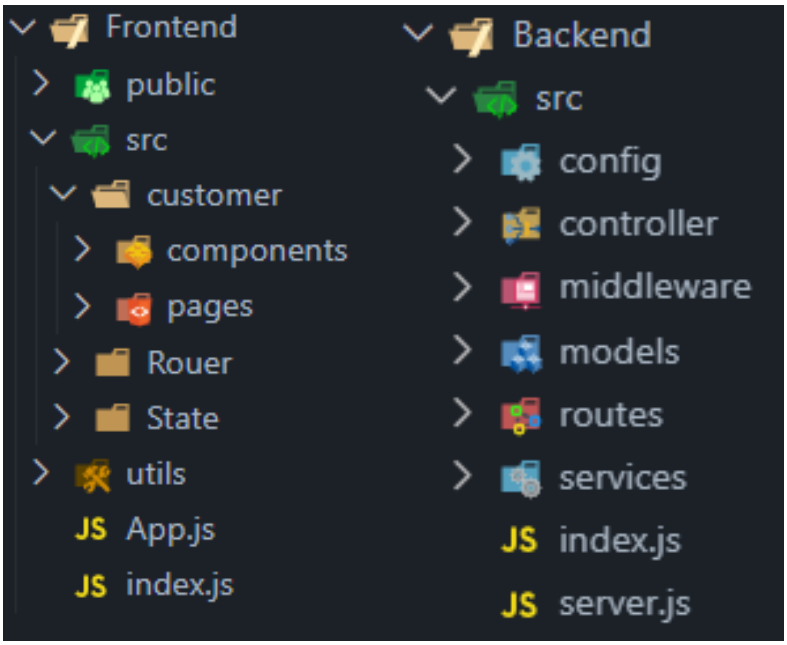

Рисунок 3.1 – Планова структура файлів проєкту вебсайту “SoundKitchen”

Інтерфейс користувача: Планується реалізувати з використанням React для створення модульних компонентів, таких як Product (картка товару з назвою, ціною, кнопкою «Додати до кошика») і Cart (список товарів із можливістю редагування). Tailwind CSS забезпечить адаптивний дизайн, а Material UI [27] надасть готові компоненти (кнопки, форми) для єдиного стилю.

Управління станом: Redux використовуватиметься для централізованого зберігання даних, таких як вміст кошика чи інформація про користувача, із застосуванням Actions і Reducers для обробки змін.

Навігація: React Router забезпечить маршрутизацію між сторінками (наприклад, /home, /search, /product/:id), дозволяючи користувачам переходити між розділами без перезавантаження.

API-взаємодія: Axios використовуватиметься для відправлення запитів до бекенду, наприклад, для отримання каталогу товарів чи оформлення замовлення.

API: Express реалізує RESTful API з ендпоінтами, такими як GET /products (отримання каталогу), POST /orders (створення замовлення), POST /auth/login (аутентифікація). Контролери оброблятимуть запити, викликаючи відповідні сервіси.

Бізнес-логіка: Сервіси міститимуть логіку для обробки запитів, наприклад, фільтрація товарів за категоріями чи створення рекомендацій на основі історії переглядів.

Моделі даних: Mongoose-схеми визначатимуть структуру даних, наприклад, модель Product із полями для назви, ціни, категорії та відгуків.

Безпека: JWT [29] забезпечить аутентифікацію, bcrypt [30] – хешування паролів, CORS [31] – обробку міжсайтових запитів, а HTTPS захист – передачі даних.

MongoDB використовуватиметься для зберігання даних про товари, замовлення, користувачів і відгуки. Гнучка структура NoSQL[23] дозволить легко додавати нові атрибути, наприклад, технічні характеристики інструментів.

Mongoose забезпечить зручну взаємодію з MongoDB, спрощуючи створення, оновлення та видалення записів через API.

План реалізації “SoundKitchen” базується на принципах модульності, DRY [15] і KISS [16], що забезпечують простоту розробки та підтримки. Багаторазові компоненти (наприклад, Cart, Product) і сервіси (наприклад, ProductService, OrderService) зменшують дублювання коду. На фронтенді застосовуватиметься паттерн Container/Presentation для розділення логіки та

відображення, а на бекенді – багатошарова архітектура (маршрути, контролери, сервіси), що забезпечує гнучкість і масштабованість.

Ця архітектура та план реалізації створюють основу для розробки вебсайту “SoundKitchen”, який відповідатиме вимогам продуктивності, безпеки, адаптивності та зручності, сформульованим у розділі 2.2.

3.1.1 Реалізація клієнтської частини системи

Клієнтська частина вебсайту інтернет-магазину музичних інструментів «SoundKitchen» реалізована з використанням бібліотеки React.js [18], що забезпечує створення динамічного, модульного та зручного користувацького інтерфейсу. Архітектура фронтенду побудована на основі компонентного підходу, що сприяє повторному використанню коду, спрощує підтримку та відповідає принципам модульності, описаним у розділі 3.1. Реалізація враховує функціональні та нефункціональні вимоги, сформульовані в розділі 2.2, і використовує технологічний стек, описаний у розділі 2.3 (React, Redux, React Router, Tailwind CSS, Material UI), для забезпечення адаптивності, продуктивності та зручності використання.

Навігація реалізована через компонент Navigation.jsx, який забезпечує адаптивне меню, оптимізоване для десктопних і мобільних пристроїв. Під час ініціалізації навігаційної панелі створюється поле пошуку з підтримкою введення запиту та обробкою URL-параметрів (див. лістинг 3.1).

Лістинг 3.1 – Ініціалізація навігаційної панелі з інтегрованою пошуковою системою

```
import { useState } from 'react';
import { useNavigate } from 'react-router-dom';
export default function Navigation() {
  const [searchQuery, setSearchQuery] = useState('');
  const [showMobileSearch, setShowMobileSearch] = useState(false);
  const navigate = useNavigate();
  const handleSearch = (e) => {
    if (e.key === 'Enter' && searchQuery.trim()) {
```

```
navigate(`/search?query=${encodeURIComponent(searchQuery.trim())}`
);
    }
  };
  return (
    // JSX для навігаційної панелі
  );
}
```

Пошукова система інтегрована в навігаційну панель і використовує URL-параметри для обробки запитів, що дозволяє зберігати та ділитися результатами пошуку. React Router забезпечує плавну навігацію між сторінками, такими як /home, /search чи /product/:id, без перезавантаження сторінки.

Для управління глобальним станом використано Redux Toolkit, який організовано за модулями, що відповідають функціональним областям:

- State/Auth/: аутентифікація користувачів (реєстрація, вхід, управління профілем).
- State/Cart/: управління кошиком (додавання, редагування, видалення товарів).
- State/Order/: обробка замовлень (створення, перегляд, відстеження).
- State/Payment/: управління платежами.
- State/Product/: робота з каталогом товарів.
- State/Review/: управління відгуками.
- State/Rating/: обробка рейтингів.

При ініціалізації модуля платежів створюються константи для обробки дій, пов’язаних із створенням і оновленням платежів (див. лістинг 3.2).

Лістинг 3.2 – Константи для управління діями в модулі платежів

```
export const CREATE_PAYMENT_REQUEST = 'CREATE_PAYMENT_REQUEST';
export const CREATE_PAYMENT_SUCCESS = 'CREATE_PAYMENT_SUCCESS';
export const CREATE_PAYMENT_FAILURE = 'CREATE_PAYMENT_FAILURE';
```

Кожен модуль містить Action Types, Actions і Reducers, що забезпечують передбачуваність і централізоване управління станом.

Каталог товарів реалізований через компонент Product.jsx, який підтримує розширену фільтрацію. Для ініціалізації фільтрів створюється структура даних із параметрами (див. лістинг 3.3).

Лістинг 3.3 – Структура даних для фільтрації товарів за діапазоном знижок

```
export const filterOptions = {
  id: 'discount',
  name: 'Діапазон знижок',
  options: [
    { value: 'all', label: 'Усі товари' },
    { value: '20', label: 'Знижка від 20%' },
    { value: '30', label: 'Знижка від 30%' },
    { value: '40', label: 'Знижка від 40%' },
    { value: '50', label: 'Знижка від 50%' },
  ],
};
```

Компонент підтримує фільтрацію за категоріями (гітари, клавішні, перкусія), брендами, ціновими діапазонами та знижками. Для мобільних пристроїв передбачена висувна панель фільтрів, що забезпечує зручність використання.

Компонент ProductDetails.jsx відображає повну інформацію про товар, включаючи інтерактивні елементи. Під час ініціалізації сторінки товару створюється блок із ціною, знижкою та оригінальною вартістю (див. лістинг 3.4).

Лістинг 3.4 – Відображення ціни та знижки на сторінці товару

```
<div className="flex space-x-4 items-center text-gray-900 mt-4">
  <p className="font-bold">{product?.discountedPrice} ₴</p>
  <p className="opacity-50 line-through">{product?.price} ₴</p>
  <p className="text-green-500 font-bold">-
{product?.discountPercentage}%</p>
</div>
```

Сторінка включає вкладки для відгуків, рейтингів і форми залишення відгуків, а також блок рекомендацій схожих товарів, згенерованих на основі категорії чи бренду.

Система відгуків реалізована через компоненти ReviewForm.jsx і RatingOverview.jsx. Для обробки форми відгуків створюється функція валідації (див. лістинг 3.5).

Лістинг 3.5 – Валідація форми відгуків

```
const handleSubmit = async (e) => {
  e.preventDefault();
  if (formData.rating === 0) {
    setError('Будь ласка, виберіть оцінку');
    return;
  }
  if (!isRatingOnly && formData.review.trim().length < 10) {
    setError('Відгук має містити щонайменше 10 символів');
    return;
  }
  // Логіка відправки відгуку
};
```

Система забезпечує валідацію відгуків і рейтингів, а також перевіряє, чи користувач уже залишав оцінку для цього товару.

Оформлення замовлення реалізовано через компонент Checkout.jsx із багатокроковим інтерфейсом. Під час ініціалізації процесу оформлення створюється список кроків і відображається активний етап (див. лістинг 3.6).

Лістинг 3.6 – Ініціалізація багатокрокового інтерфейсу оформлення замовлення

```
const steps = ['Вхід', 'Дані доставки', 'Підсумок замовлення',
'Оплата'];
export default function Checkout() {
  const [activeStep, setActiveStep] = React.useState(0);
  return (
    <div className="lg:px-12 pt-6">
      <Stepper activeStep={activeStep}>
        {steps.map((label) => (
          <Step key={label}>
            <StepLabel>{label}</StepLabel>
          </Step>
        ))}
      </Stepper>
    </div>
  );
}
```

Кожен крок включає валідацію введених даних, а проміжні результати зберігаються в Redux store для безперервності процесу.

Компонент Order.jsx забезпечує перегляд і фільтрацію замовлень. Для ініціалізації фільтрації замовлень створюється масив статусів і логіка їхньої обробки (див. лістинг 3.7).

Лістинг 3.7 – Фільтрація замовлень за статусом

```
const orderStatus = [
  { label: 'Розміщено', value: 'Placed' },
  { label: 'Відправлено', value: 'Shipped' },
  { label: 'Доставлено', value: 'Delivered' },
  { label: 'Скасовано', value: 'Cancelled' },
];
const filteredOrders = orders?.filter((order) => {
  if (!statusFilter.length) return true;
  return statusFilter.includes(order.orderStatus);
});
```

Платіжна система інтегрована через компонент PaymentSuccess.jsx, який обробляє стани платежів. Під час ініціалізації компонента викликаються дії для оновлення статусу платежу (див. лістинг 3.8).

Лістинг 3.8 – Обробка статусу платежу в компоненті PaymentSuccess

```
import { useEffect } from 'react';
import { useDispatch } from 'react-redux';
export default function PaymentSuccess({ orderId }) {
  const dispatch = useDispatch();
  useEffect(() => {
    if (orderId) {
      dispatch(getOrderById(orderId));
      dispatch(updatePaymentStatus(orderId));
    }
  }, [orderId, dispatch]);
  return (
    // JSX для відображення статусу платежу
  );
}
```

Компонент відображає стани платежу (успішний, в очікуванні, невдалий) із відповідними повідомленнями.

Клієнтська частина розроблена з урахуванням принципів адаптивного дизайну, реалізованих через Tailwind CSS і Material UI. Основні особливості:

- Адаптивні сітки для каталогу товарів.
- Мобільні меню та висувні фільтри.
- Оптимізовані форми для різних розмірів екранів.
- Прогресивне завантаження зображень і контенту.
- Індикатори завантаження для асинхронних операцій.

Фронтенд забезпечує плавну взаємодію через асинхронні API-запити (Axios), кешування даних у Redux store і оптимістичні оновлення інтерфейсу, що покращує сприйняту швидкість роботи.

3.1.2 Реалізація серверної частини системи

Серверна частина вебсайту інтернет-магазину музичних інструментів «SoundKitchen» розроблена з використанням Node.js [19] і фреймворку Express.js, що забезпечує створення надійної, масштабованої та продуктивної системи для обробки бізнес-логіки, взаємодії з базою даних і виконання API-запитів. Реалізація серверної частини відповідає принципам багатошарової архітектури, описаним у розділі 3.1, і враховує функціональні та нефункціональні вимоги, сформульовані в розділі 2.2. Архітектура побудована з урахуванням модульності, що спрощує розробку, тестування та подальше розширення функціоналу платформи.

Серверна частина «SoundKitchen» організована за багатошаровим принципом, що включає такі основні компоненти:

- Маршрути (Routes): визначають API-ендпоінти для обробки HTTPS-запитів.
- Контролери (Controllers): відповідають за обробку запитів, підготовку відповідей і виклик бізнес-логіки.
- Сервіси (Services): містять бізнес-логіку та взаємодіють із базою даних.

– Моделі (Models): визначають структуру даних у MongoDB [20] за допомогою Mongoose-схем.

– Проміжні обробники (Middleware): виконують наскрізні функції, такі як аутентифікація, валідація та обробка CORS.

Ця структура забезпечує чіткий розподіл відповідальності, що відповідає принципам DRY [15] і KISS [16], а також дозволяє легко масштабувати систему, додаючи нові функції, наприклад, інтеграцію з зовнішніми сервісами чи розширення системи рекомендацій.

Основна логіка ініціалізації серверного додатку реалізована у файлі index.js, який налаштовує Express-сервер, підключає необхідні middleware і маршрути. Лістинг 3.9 демонструє базову конфігурацію сервера.

Лістинг 3.9 – Налаштування Express-сервера в index.js

```
const express = require('express');
const cors = require('cors');
const app = express();
app.use(express.json());
app.use(cors());
// Підключення маршрутів
app.use('/auth', require('./routes/auth.routes'));
app.use('/api/users', require('./routes/user.routes'));
app.use('/api/products', require('./routes/product.routes'));
app.use('/api/cart', require('./routes/cart.routes'));
app.use('/api/orders', require('./routes/order.routes'));
app.use('/api/reviews', require('./routes/review.routes'));
app.use('/api/payments', require('./routes/payment.routes'));
module.exports = { app };
```

Запуск сервера здійснюється у файлі server.js, який встановлює підключення до бази даних MongoDB і активує сервер на заданому порті (наприклад, 5454). Лістинг 3.10 ілюструє цей процес.

Лістинг 3.10 – Запуск сервера в server.js

```
const { app } = require('./index');
const { connectToDatabase } = require('./config/db');
const PORT = process.env.PORT || 5454;
app.listen(PORT, async () => {
    try {
        await connectToDatabase();
```

```
        console.log(`Сервер запущено на порті: ${PORT}`);
    } catch (error) {
        console.error(`Помилка запуску сервера:
${error.message}`);
    }
});
```

Файл config/db.js відповідає за підключення до MongoDB[20], використовуючи бібліотеку Mongoose для забезпечення стабільної взаємодії з базою даних.

Безпека системи забезпечується через реалізацію аутентифікації на основі JWT [29] і хешування паролів за допомогою bcrypt [30]. Middleware authenticate.js перевіряє валідність JWT-токенів, як показано в лістингу 3.11.

Лістинг 3.11 – Middleware для аутентифікації

```
const jwt = require('jsonwebtoken');
const userService = require('../services/user.service');

const authenticate = async (req, res, next) => {
    try {
        const token = req.headers.authorization?.split(' ')[1];
        if (!token) {
            return res.status(401).json({ error: 'Токен не надано'
});
        }
        const decoded = jwt.verify(token, process.env.JWT_SECRET);
        const user = await
userService.findUserById(decoded.userId);
        if (!user) {
            return res.status(401).json({ error: 'Користувача не
знайдено' });
        }
        req.user = user;
        next();
    } catch (error) {
        return res.status(500).json({ error: `Помилка
аутентифікації: ${error.message}` });
    }
};
module.exports = authenticate;
```

Контролер auth.controller.js реалізує логіку реєстрації та входу користувачів, як показано в лістингу 3.12.

Лістинг 3.12 – Реєстрація користувача

```
const userService = require('../services/user.service');
const jwtProvider = require('../utils/jwt');
const cartService = require('../services/cart.service');
const register = async (req, res) => {
    try {
        const { email } = req.body;
        const existingUser = await
userService.findUserByEmail(email);
        if (existingUser) {
            return res.status(400).json({ message: `Користувач із
email ${email} уже існує` });
        }
        const user = await userService.createUser(req.body);
        const jwt = jwtProvider.generateToken(user._id);
        await cartService.createCart(user);
        return res.status(201).json({ jwt, message: 'Реєстрація
успішна' });
    } catch (error) {
        return res.status(500).json({ error: `Помилка реєстрації:
${error.message}` });
    }
};
module.exports = { register };
```

Сервіс user.service.js відповідає за створення користувачів із хешуванням паролів і пошук за ідентифікатором чи email, забезпечуючи валідацію даних.

Функціонал кошика реалізований через контролер cart.controller.js і сервіс cart.service.js, які підтримують додавання, оновлення, видалення товарів і очищення кошика після оформлення замовлення. Лістинг 3.13 демонструє додавання товару до кошика.

Лістинг 3.13 – Додавання товару до кошика

```
const cartService = require('../services/cart.service');
const addItemToCart = async (req, res) => {
    try {
        const user = req.user;
        const cartItem = await cartService.addItemToCart(user._id,
req.body);
        return res.status(201).json(cartItem);
    } catch (error) {
        return res.status(500).json({ error: `Помилка додавання до
кошика: ${error.message}` });
    }
};
module.exports = { addItemToCart };
```

Сервіс cart.service.js обробляє логіку кошика, включаючи перевірку доступності товару та оновлення загальної суми.

Обробка замовлень реалізована через контролер order.controller.js і сервіс order.service.js, які підтримують повний життєвий цикл замовлення: створення, оновлення статусу, відстеження та скасування. Лістинг 3.14 ілюструє створення замовлення.

Лістинг 3.14 – Створення замовлення

```
const orderService = require('../services/order.service');
const cartService = require('../services/cart.service');
const Address = require('../models/address.model');
const Order = require('../models/order.model');
const OrderItems = require('../models/orderItems.model');
const createOrder = async (req, res) => {
    try {
        const user = req.user;
        const { shippingAddress } = req.body;
        const order = await orderService.createOrder(user,
shippingAddress);
        return res.status(201).json(order);
    } catch (error) {
        return res.status(500).json({ error: `Помилка створення
замовлення: ${error.message}` });
    }
};
module.exports = { createOrder };
```

Система відгуків реалізована через контролер review.controller.js і сервіс review.service.js, які підтримують створення, перегляд і валідацію відгуків. Лістинг 3.16 показує створення відгуку.

Лістинг 3.16 – Створення відгуку

```
const reviewService = require('../services/review.service');
const createReview = async (req, res) => {
    try {
        const user = req.user;
        const review = await reviewService.createReview(req.body,
user);
        return res.status(201).json(review);
    } catch (error) {
        return res.status(500).json({ error: `Помилка створення
відгуку: ${error.message}` });
    }
```

```
};
module.exports = { createReview };
```

Сервіс забезпечує перевірку на унікальність відгуку від одного користувача для конкретного товару та валідацію текстового вмісту.

Для обробки платежів інтегровано платіжну систему LiqPay [32], яка підтримує створення платіжних посилань і обробку callback-повідомлень. Лістинг 3.17 демонструє створення платіжного посилання.

Лістинг 3.17 – Створення платіжного посилання

```
const LiqPay = require('../utils/liqpay');
const orderService = require('../services/order.service');
const createPaymentLink = async (orderId) => {
    try {
        const order = await orderService.findOrderById(orderId);
        const liqPay = new LiqPay(process.env.LIQPAY_PUBLIC_KEY,
process.env.LIQPAY_PRIVATE_KEY);
        const paymentLinkRequest = {
            // Данні запиту
        };
        const paymentLink = liqPay.cnb_form(paymentLinkRequest);
        return { paymentLink };
    } catch (error) {
        throw new Error(`Помилка створення платіжного посилання:
${error.message}`);
    }
};
module.exports = { createPaymentLink };
```

Callback-обробник у payment.controller.js верифікує статус платежу та оновлює замовлення.

Адміністративна функціональність реалізована через контролер adminOrder.controller.js, який дозволяє переглядати, підтверджувати, відправляти, доставляти або скасовувати замовлення. Лістинг 3.18 показує підтвердження замовлення.

Лістинг 3.18 – Підтвердження замовлення адміністратором

```
const orderService = require('../services/order.service');
const confirmOrder = async (req, res) => {
    try {
```

```
        const orderId = req.params.orderId;
        const order = await orderService.confirmOrder(orderId);
        return res.status(200).json(order);
    } catch (error) {
        return res.status(500).json({ error: `Помилка
підтвердження замовлення: ${error.message}` });
    }
};
module.exports = { confirmOrder };
```

Для зберігання даних вебсайту «SoundKitchen» використовується NoSQL-база даних [23] MongoDB [20], яка забезпечує гнучкість у роботі з неструктурованими даними та відповідає вимогам продуктивності й масштабованості, сформульованим у розділі 2.2. MongoDB обрано через її здатність ефективно обробляти різноманітні атрибути товарів, підтримку швидких запитів і простоту інтеграції з Node.js через бібліотеку Mongoose.

База даних складається з кількох колекцій, що відповідають основним сутностям системи:

– Users: зберігає інформацію про користувачів (email, хешований пароль, ім’я, прізвище, адреси).

– Products: містить дані про товари (назва, ціна, знижка, категорія, бренд, технічні характеристики, відгуки).

– Carts: зберігає кошики користувачів (список товарів, кількість, загальна сума).

– CartItems: зберігає товари додані в кошик.

– Orders: містить інформацію про замовлення (користувач, товари, адреса доставки, статус, загальна сума).

– OrderItems: деталізує товари в замовленнях (ціна, кількість, колір).

– Reviews: зберігає відгуки та рейтинги до товарів (текст, оцінка, автор, дата).

– Addresses: містить адреси доставки (вулиця, місто, поштовий індекс, країна).

– Categories: зберігає дані про категорії (3 різні рівні).

– Reviews та Ratings: містять данні про відгуки та рейтинги товарів.

На рисунку 3.2 зображено схему бази даних, яка ілюструє взаємозв’язки між колекціями.

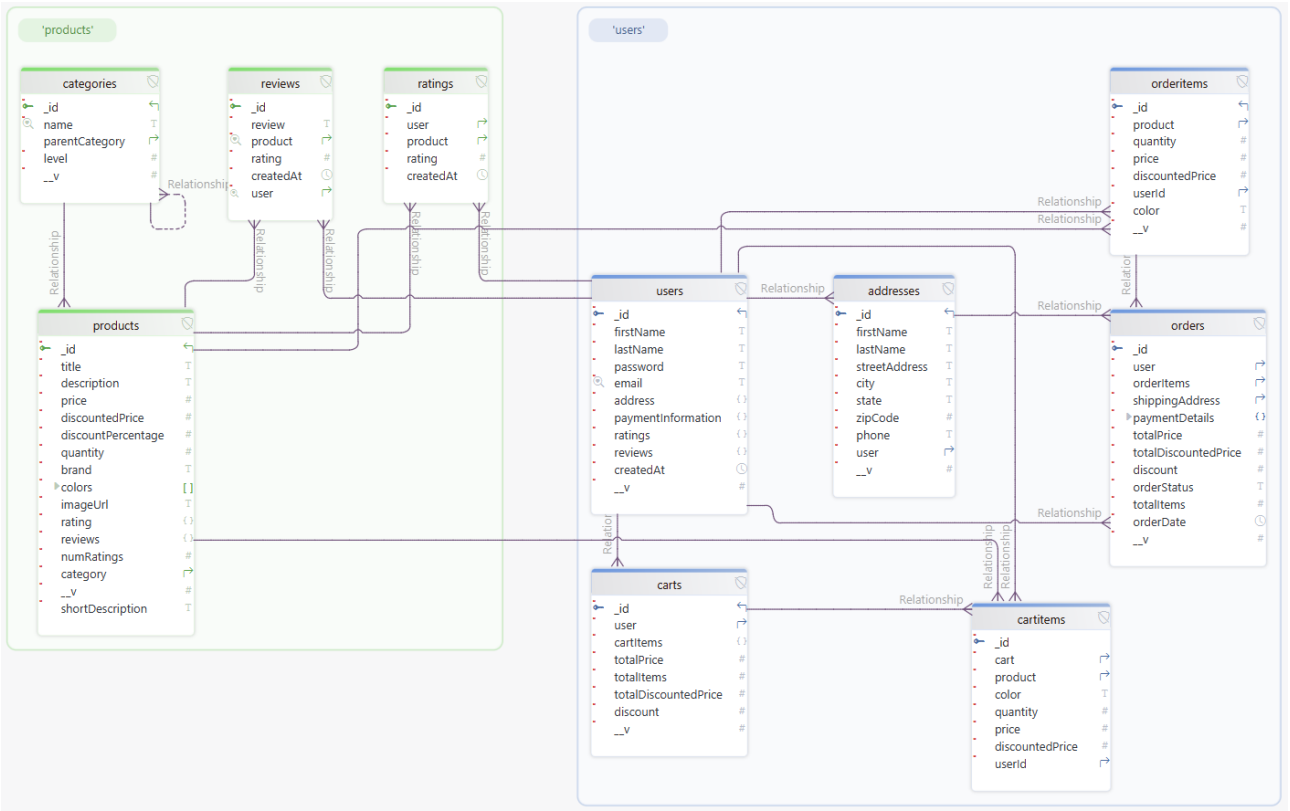

Рисунок 3.2 – Схема бази даних вебсайту «SoundKitchen»

Схема бази даних, реалізована в MongoDB, забезпечує гнучке зберігання інформації про товари, замовлення та користувачів, оптимізуючи швидкість обробки запитів. Використання NoSQL-підходу дозволяє легко масштабувати систему та адаптувати її до зростання асортименту. Структура колекцій підтримує ефективну взаємодію з RESTful API, забезпечуючи надійність і швидкодію платформи.

3.2 Демонстрація інтерфейсу та функціональних можливостей створеного вебсайту

Після розгляду архітектури та реалізації системи цей підрозділ фокусується на демонстрації вебсайту інтернет-магазину музичних інструментів

“SoundKitchen” із позиції користувача. Він ілюструє, як функції, визначені у розділі 2.2, втілюються у візуальному оформленні через інтерфейс, а також описує основні інтерактивні елементи, що сприяють комфортному користуванню платформою. У межах демонстрації розглядаються ключові сценарії взаємодії для клієнтів та адміністраторів, акцентуючи увагу на адаптивності, ефективності й ергономічності інтерфейсу.

При вході на вебсайт користувач потрапляє на головну сторінку, яка є центральним елементом взаємодії. Вона містить адаптивне меню навігації, поле пошуку, банери з акційними пропозиціями, рекомендовані товари та категорії (гітари, клавішні, перкусія тощо). У верхній частині сторінки розташовано логотип магазину та кнопки для входу/реєстрації, переходу до кошика та особистого кабінету. Для залучення уваги на банерах відображаються акції та оголошення.

На рисунку 3.3 зображено головну сторінку вебсайту з основними елементами інтерфейсу.

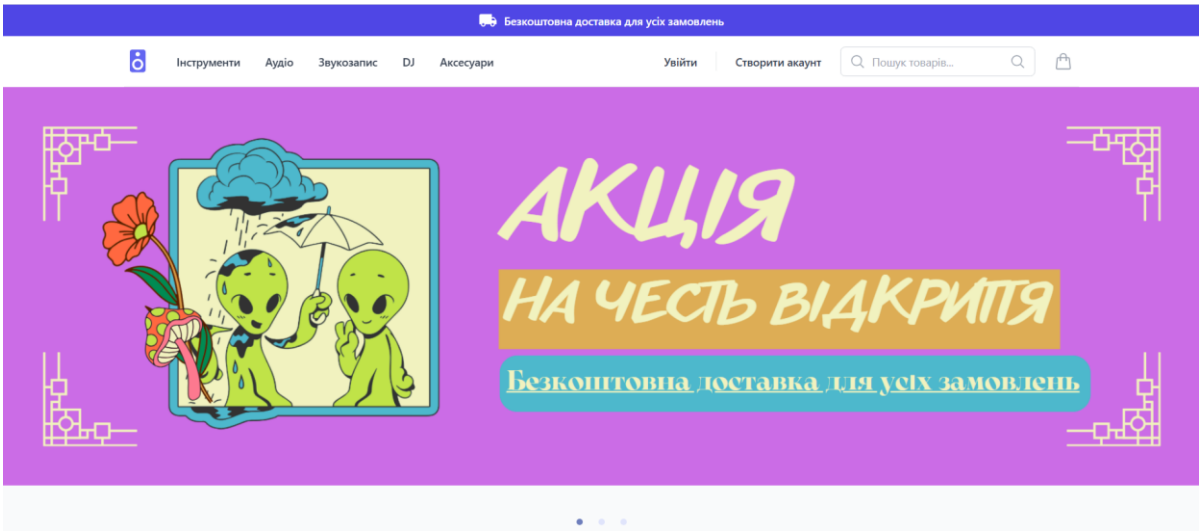

Рисунок 3.3 – Головна сторінка вебсайту

Сторінка переліку товарів є ключовим компонентом платформи, що дозволяє користувачам переглядати музичні інструменти та супутнє обладнання. Для переходу до певної категорії товару, потрібно обрати її у верхньому навігаційному меню, натиснувши на будь яку вказану категорії і перейшовши на

одну з вказаних підкатегорій. Сторінка з товарами містить адаптивну сітку товарів із картками, які включають зображення, назву, ціну та знижку (якщо вона присутня). Для зручності пошуку реалізована система фільтрації, що дозволяє обирати товари за діапазоном знижок, кольором чи наявністю на складі. На мобільних пристроях фільтри відображаються у висувній панелі, що економить простір і забезпечує зручність. На рисунку 3.4 продемонстровано сторінку товарів з фільтрами.

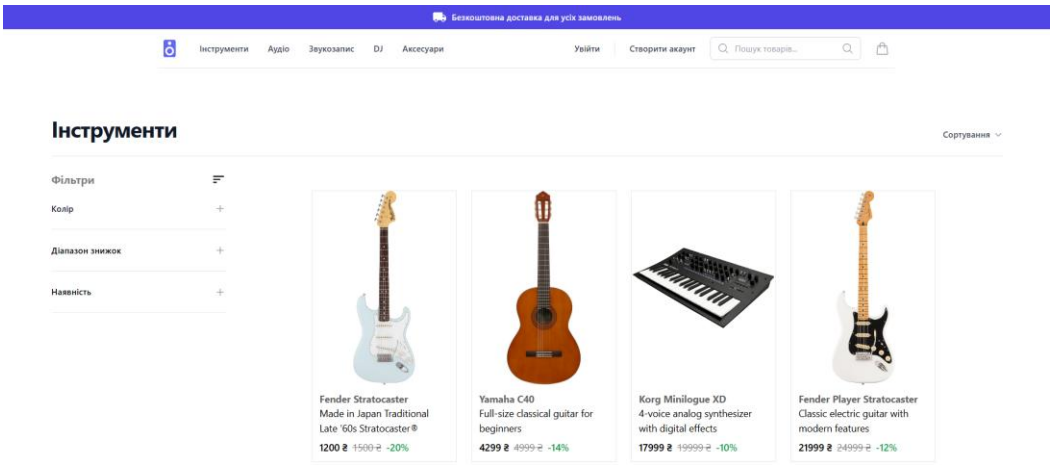

Рисунок 3.4 – Сторінка каталогу з фільтрами

Також на рисунку 3.5 зображено мобільну версію сайту з відкритим меню вибору фільтрів.

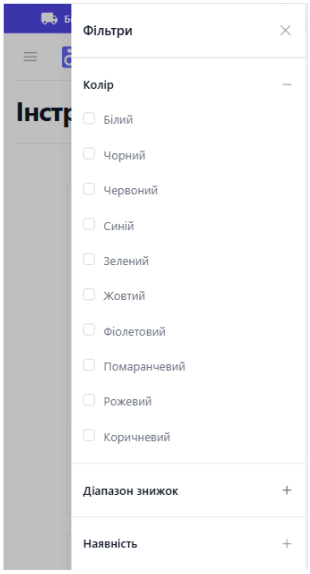

Рисунок 3.5 – Мобільна версія каталогу з фільтрами

Сторінка окремого товару містить детальну інформацію: зображення, опис, характеристики, ціну, знижку (якщо є) та кнопку “Додати до кошика”. У нижній частині сторінки розташовано вкладки для відгуків, рейтингів і форми залишення відгуків, а також блок із рекомендованими товарами, підібраними на основі характеристик товару. Інтерактивні кнопки вибору кольру, забезпечують зручну взаємодію для користувачів. На рисунку 3.6 зображено сторінку товару.

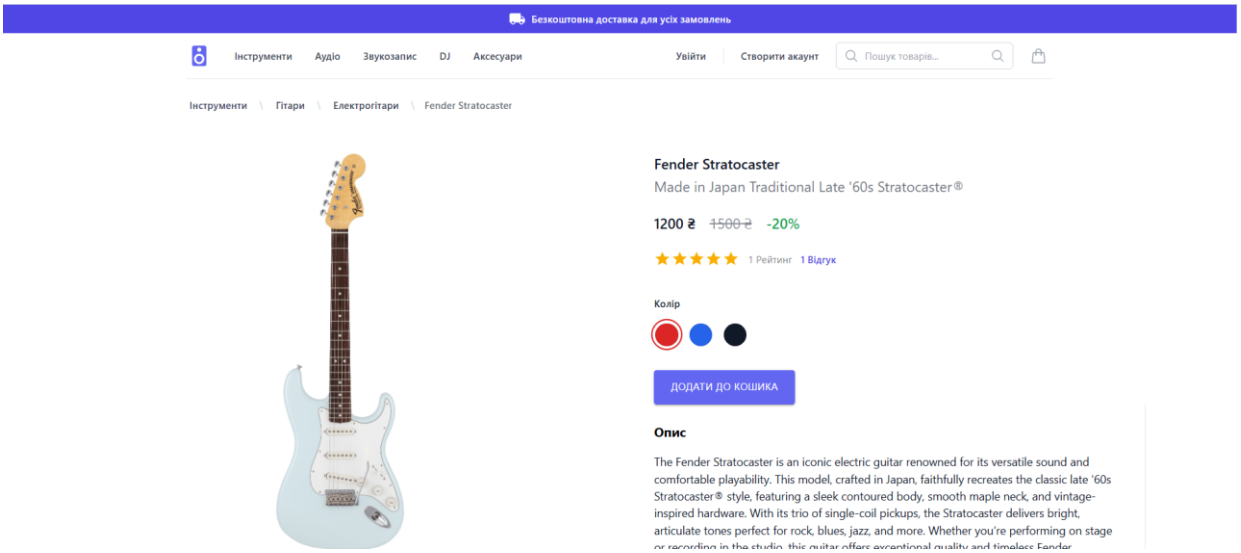

Рисунок 3.6 – Сторінка товару

Користувачі можуть залишати відгуки та оцінки для товарів через форму на сторінці товару. Форма включає поле для тексту відгуку, вибір рейтингу (1–5 зірок) і валідацію (наприклад, мінімальна довжина тексту — 10 символів). Система перевіряє, чи користувач уже залишав відгук для цього товару, щоб уникнути дублювання. Відгуки відображаються у вкладці посортовані за датою. На рисунку 3.7 зображено форму відгуків і список відгуків на сторінці товару.

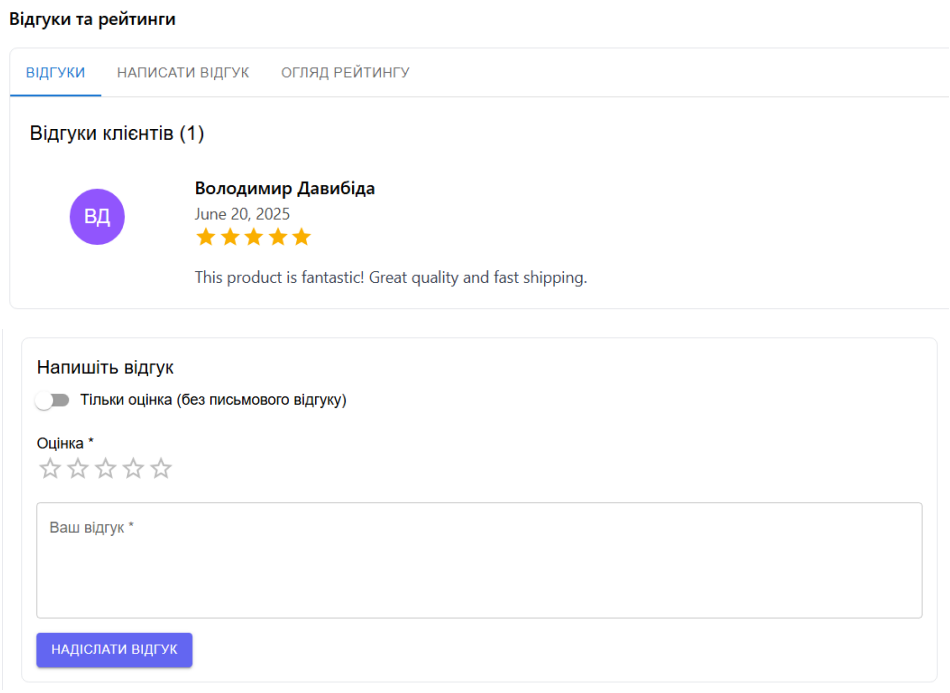

Рисунок 3.7 – Форма та список відгуків

Процес оформлення замовлення реалізований через багатокроковий інтерфейс, що включає етапи: вхід/реєстрація, введення даних доставки, вибір способу оплати та підтвердження замовлення. Кожен етап супроводжується валідацією даних і відображенням активного кроку через компонент Stepper. Після успішного оформлення користувач отримує підтвердження з номером замовлення та деталями.

На рисунку 3.8 зображено багатокроковий інтерфейс оформлення замовлення.

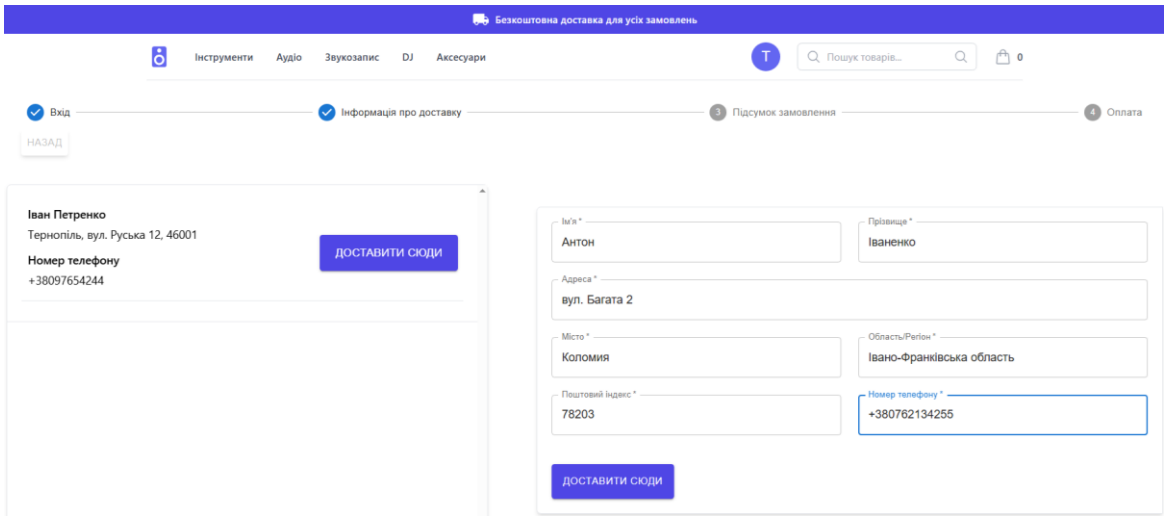

Рисунок 3.8 – Інтерфейс оформлення замовлення

На рисунку 3.9 зображено другий крок підтвердження замовлення з усіма обраними товарами та підсумковою ціною.

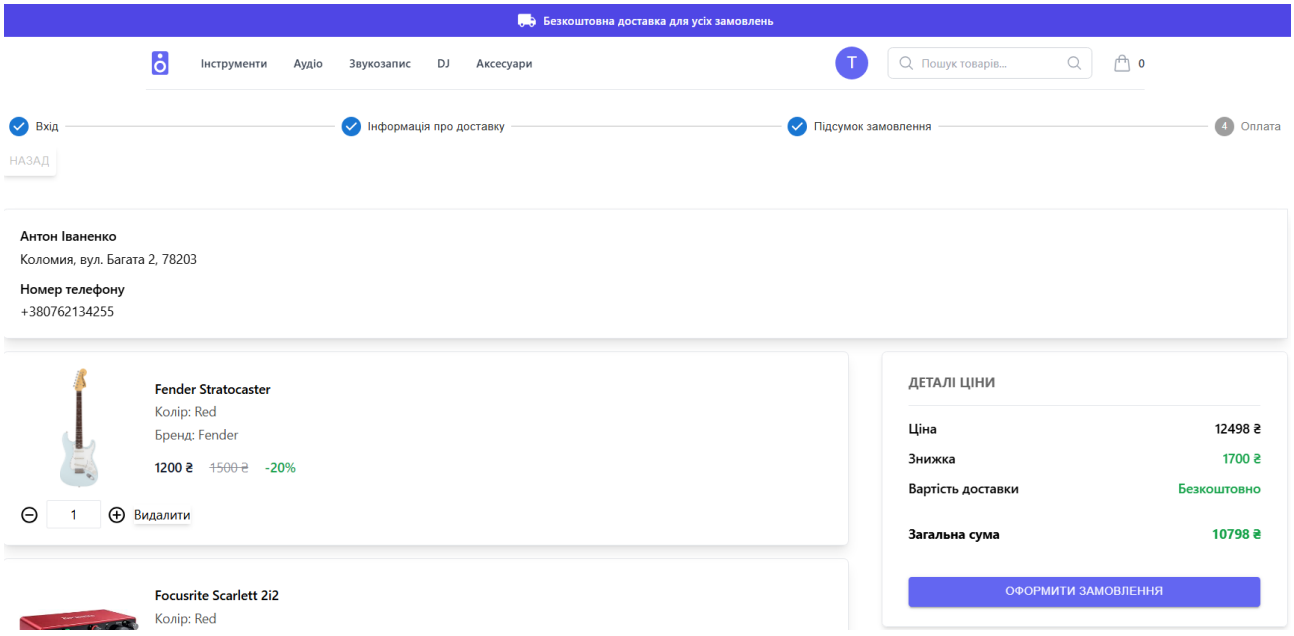

Рисунок 3.9 – Інтерфейс підтвердження замовлення

Сторінка замовлень дозволяє користувачам переглядати історію замовлень, їх статуси та інформацію про них. Сторінка замовлень містить фільтри за статусом (розміщено, відправлено, доставлено, скасовано, повернено), що полегшує пошук. Кожне замовлення супроводжується деталями, такими як дата, сума та статус доставки. На рисунку 3.10 показано сторінку замовлень.

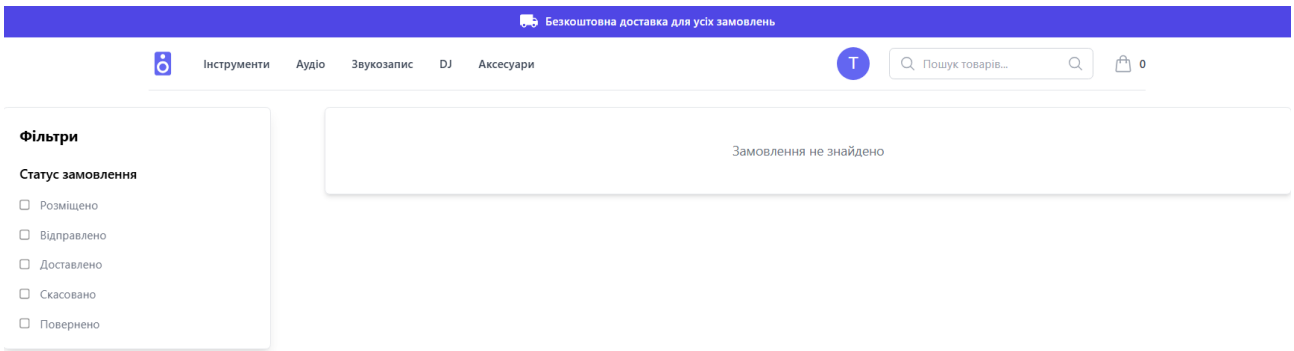

Рисунок 3.10 – Сторінка замовлень

На сторінці кожного товару після секції з відгуками знаходиться розділ з рекомендаціями на основі товару який переглядається на цій сторінці. Рекомендації підбираються на основі бренду, кольору та цінової категорії. На картці кожного підібраного товару є надпис який індентифікує за яким параметром було рекомендовано той чи інший товар. На рисунку 3.11 зображено секцію рекомендацій.

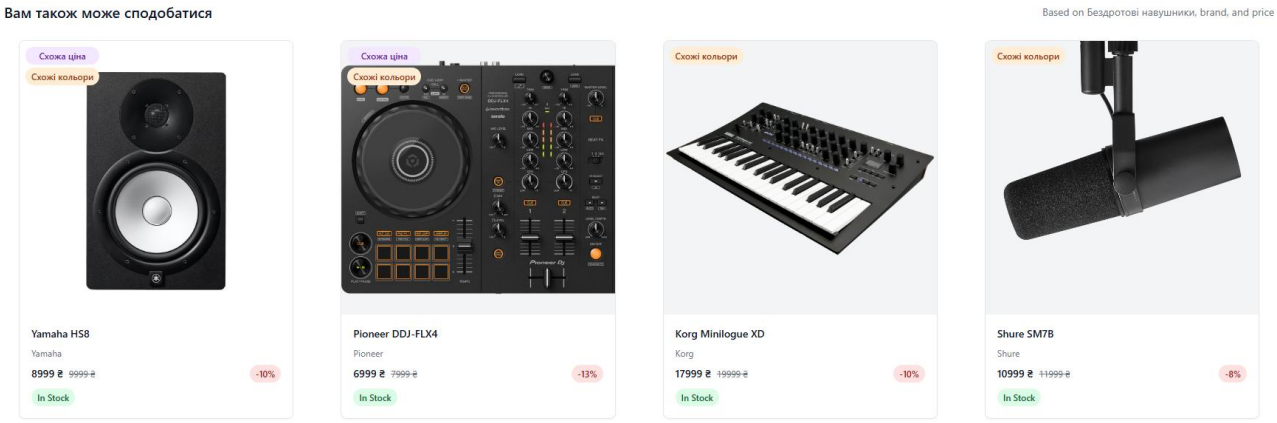

Рисунок 3.11 – Сторінка замовлень

Інтерфейс «SoundKitchen» оптимізований для різних пристроїв завдяки Tailwind CSS [26] і Material UI [27] , а також відповідає стандартам доступності WCAG [33]. Основні особливості адаптивного дизайну:

- Адаптивні сітки для каталогу товарів, які змінюють кількість стовпців залежно від розміру екрана.
- Мобільне меню, що ховається у гамбургер-іконку на смартфонах.
- Висувні панелі фільтрів для мобільних пристроїв.
- Оптимізовані форми введення для сенсорних екранів.
- Прогресивне завантаження зображень для зменшення часу завантаження.

Для підвищення залученості користувачів на головній сторінці відображаються загальні рекомендації, які відображаються одинаково для усіх користувачів. Індикатори завантаження та оновлення інтерфейсу (наприклад,

додавання товару до кошика без затримки) покращують візуальну швидкість роботи.

Ця демонстрація інтерфейсу та функціональних можливостей вебсайту “SoundKitchen” ілюструє, як технічна реалізація, описана в попередніх розділах, перетворюється на зручний і ефективний продукт для користувачів. Платформа забезпечує інтуїтивну навігацію, швидкий доступ до ключових функцій і адаптивність, відповідаючи вимогам сучасних стандартів електронної комерції.

3.3 Тестування та оцінка якості системи

Для оцінки зручності, ефективності та задоволеності користувачів вебсайтом інтернет-магазину музичних інструментів «SoundKitchen» було проведено тестування та апробацію системи серед цільової аудиторії – музикантів різного рівня підготовки, включаючи початківців і професіоналів.

Метою тестування було зібрати якісні та кількісні відгуки, щоб підтвердити відповідність платформи функціональним і нефункціональним вимогам, а також виявити потенційні напрямки для вдосконалення. Оцінка якості проводилася шляхом поширення онлайн-опитування через Google Forms серед учасників, які активно взаємодіяли з прототипом вебсайту протягом тижня. Учасникам було запропоновано виконати типові сценарії використання (пошук товарів, додавання до кошика, оформлення замовлення, перегляд відгуків) і заповнити анонімну форму з питаннями про інтерфейс, зручність, продуктивність, адаптивність і загальне враження.

На рисунку 3.12 зображено приклад онлайн-форми опитування, використаної для оцінки відповідності вимогам вебсайту.

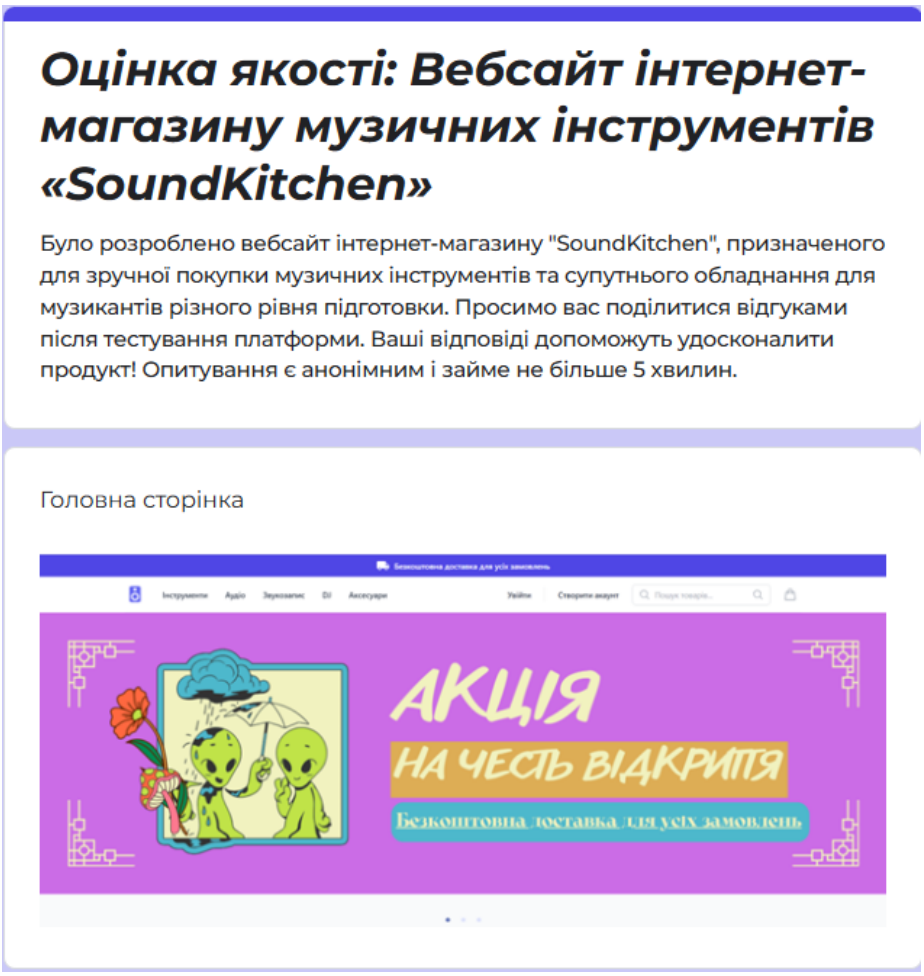

Рисунок 3.12 – Заголовок опитування для оцінки відповідності вимогам вебсайту “SoundKitchen”

Опитування містило оцінювання ключових аспектів взаємодії з платформою:

– Загальне враження від інтерфейсу.
– Інтуїтивність навігації та легкість освоєння.
– Зручність пошуку та фільтрації товарів.
– Ефективність кошика та процесу оформлення замовлення.
– Корисність рекомендацій і відгуків.
– Адаптивність інтерфейсу на різних пристроях.
– Продуктивність і стабільність системи.
– Пропозиції щодо покращення.

Результати опитування показали високий рівень задоволеності користувачів і підтвердили практичну цінність платформи. Нижче наведено узагальнені результати за основними аспектами.

– Загальне враження від інтерфейсу: 85% респондентів оцінили інтерфейс на «5» (чудово), 15% — на «4» (добре).

– Інтуїтивність навігації та легкість освоєння: 90% учасників оцінили навігацію як «дуже інтуїтивну» («5»), 10% — як «інтуїтивну» («4»).

– Зручність пошуку та фільтрації товарів: 80% респондентів оцінили функціонал пошуку та фільтрації на «5», 20% — на «4».

– Ефективність кошика та оформлення замовлення: 85% учасників оцінили кошик і багатокроковий процес оформлення замовлення на «5», 15% — на «4».

– Корисність рекомендацій і відгуків: 80% респондентів оцінили систему рекомендацій і відгуків на «5», 20% — на «4».

– Адаптивність інтерфейсу: 100% учасників оцінили адаптивність на «5».

– Продуктивність і стабільність: 95% респондентів оцінили продуктивність і стабільність на «5», 5% — на «4».

На рисунку 3.13 представлено результати проведеного опитування.

# №	Загальне враження	Інтуїтивність нав	Зручність п	Ефективніст	Корисність	Адаптивності	Продуктивн	Пропозиції щодо покращення
1	5	5	5	5	5	5	5	Сайт чудовий, так тримати!
2	5	5	5	5	5	5	5	Додайте функцію порівняння товарів за характеристиками.
3	5	5	4	5	4	5	4	Інтерфейс дуже зручний, продовжуйте в тому ж дусі!
4	5	5	5	4	4	5	5	Розширте систему рекомендацій, щоб враховувати історію покупок.
5	5	5	5	5	5	5	5	Пошук працює добре, але можна додати автодоповнення запитів.
6	5	4	4	5	5	5	5	Платформа стабільна, жодних нарікань!
7	5	5	5	5	5	5	5	Додайте можливість оформлення замовлення без реєстрації.
8	5	5	5	5	5	5	5	Фільтри зручні, але хотілося б бачити більше параметрів, наприклад, матеріал інструменту.
9	5	4	5	4	4	5	5	Сайт ідеально адаптується до смартфона, молодці!
10	4	5	4	5	5	5	5	Впровадьте темну тему для комфортного використання вночі.
11	5	5	5	5	5	5	5	Кошик інтуїтивний, але можна додати кнопку швидкого очищення.
12	5	5	5	5	5	5	5	Рекомендації корисні, але іноді хотілося б бачити більше аксесуарів.
13	5	5	5	5	5	5	5	Все працює швидко й без збоїв, браво!
14	5	5	5	5	5	5	5	Додайте інтерактивні гайди для нових користувачів.
15	4	5	5	4	5	5	5	Сторінки завантажуються миттєво, супер!
16	5	5	4	5	4	5	5	Хотілося б бачити більше брендів у фільтрах пошуку.
17	4	5	5	5	5	5	5	Відгуки дуже допомагають, але можна додати сортування за рейтингом.
18	5	5	5	5	5	5	5	Сайт виглядає сучасно, дякую за якісну роботу!
19	5	5	5	5	5	5	5	Додайте можливість збереження обраних товарів у список бажань.
20	5	5	5	5	5	5	5	Все на найвищому рівні, чекаю нових функцій!

Рисунок 3.13 – Результати опитування користувачів

Окрім опитування, було проведено ручне тестування ключових функцій системи, включаючи:

Функціональне тестування: Перевірено працездатність пошуку, фільтрації, додавання до кошика, оформлення замовлення, аутентифікації, відгуків і рекомендацій. Усі функції працювали відповідно до вимог заданих у розділі 2.2.

Тестування адаптивності: Використано інструменти браузера (DevTools) для перевірки відображення на екранах різного розміру (від 320px до 1920px). Інтерфейс коректно адаптувався завдяки використанні бібліотеки Tailwind CSS.

Тестування продуктивності: Використано розширення для браузера Lighthouse для оцінки швидкості завантаження сторінок. Середній час завантаження становив 1,8 секунди, що відповідає вимозі продуктивності (2–3 секунди).

Тестування безпеки: Перевірено захист від типових вразливостей (наприклад, SQL-ін’єкцій, XSS) через JWT-аутентифікацію, HTTPS і валідацію вводу. Усі тести пройдено успішно.

Серед пропозицій щодо покращення, отриманих із відкритих відповідей, учасники найчастіше згадували:

– Додавання функції порівняння товарів за характеристиками для зручнішого вибору.

– Розширення системи рекомендацій для врахування історії покупок і персоналізованих уподобань.

– Можливість швидкого оформлення замовлення без обов’язкової реєстрації.

– Впровадження темної теми інтерфейсу для комфортного використання вночі.

– Додавання автодоповнення до пошукової системи для швидшого пошуку товарів.

– Розширення фільтрів за додатковими параметрами, такими як матеріал інструменту чи рік випуску.

– Створення списку бажань для збереження обраних товарів.

Ці відгуки будуть враховані для подальшого розвитку платформи. Результати тестування та оцінки якості підтверджують, що вебсайт «SoundKitchen» відповідає заявленим вимогам, забезпечує зручний користувацький досвід і має потенціал для масштабування. Платформа готова до впровадження та подальшого вдосконалення на основі зворотного зв’язку користувачів.

3.4 Висновки до третього розділу

У третьому розділі кваліфікаційної роботи детально розглянуто процеси проєктування, створення, демонстрації та тестування вебсайту інтернет-магазину музичних інструментів “SoundKitchen”. Система розроблена з використанням модульного та багатошарового підходів, що забезпечує чітке розмежування обов’язків між клієнтською та серверною частинами і рівнем даних. Така структура сприяє зручності розробки, масштабованості та простоті підтримки платформи, відповідаючи основним принципам модульності, DRY [15] і KISS [16], описаним у розділі 3.1.

Клієнтська частина створена з використанням React, Redux [24], React Router [25], Tailwind CSS [26] і Material UI [27], що дозволяє реалізувати динамічний, адаптивний і зручний інтерфейс. У тексті наведено ключові приклади коду, які демонструють функціонал основних компонентів, таких як навігація, каталог товарів, кошик, оформлення замовлень, відгуки та платежі. Серверна частина, розроблена на Node.js [19] із використанням Express і MongoDB [20], забезпечує RESTful API [21] для обробки запитів, бізнес-логіки та роботи з базою даних. Впроваджені механізми аутентифікації, обробки замовлень і відгуків, а також заходи безпеки, такі як використання JWT [29], bcrypt [30] і HTTPS, представлені у кодових прикладах.

Розділ 3.2 включає демонстрацію функціональних можливостей платформи, що підтверджує її зручність і ефективність для користувачів. Основні сторінки, такі як головна, каталог, картка товару, оформлення замовлень

і система рекомендацій, забезпечують інтуїтивну навігацію та доступ до необхідного функціоналу. Адаптивний дизайн, створений із використанням Tailwind CSS [26] і Material UI [27], забезпечує коректне відображення на різних пристроях, а оптимізації, як-от прогресивне завантаження зображень і асинхронні запити, покращують користувацький досвід.

Результати тестування, описані в розділі 3.3, підтвердили відповідність вебсайту функціональним і нефункціональним вимогам. Онлайн-опитування серед цільової аудиторії – виявило високий рівень задоволеності інтерфейсом, зручністю навігації, процесом пошуку й оформлення замовлень, а також адаптивністю. Функціональне тестування, оцінка адаптивності, продуктивності (середній час завантаження 1,8 секунди) і безпеки (захист від XSS та SQL-ін’єкцій) підтвердили стабільність і надійність платформи. Пропозиції користувачів, такі як функція порівняння товарів, можливість швидкого замовлення без реєстрації та розширення системи рекомендацій, лягли в основу майбутніх оновлень.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

Розробка вебсайту інтернет-магазину музичних інструментів “SoundKitchen” вимагає не лише технічної вправності, але й ретельного врахування принципів безпеки життєдіяльності та охорони праці. Тривала робота за комп’ютером, що є основою діяльності розробників, створює потенційні ризики для їхнього здоров’я, зокрема фізичного та психоемоційного перенапруження. У цьому розділі розглянуто ергономічні аспекти безпеки життєдіяльності, пов’язані з організацією робочого місця, та загальні вимоги охорони праці для користувачів персональних комп’ютерів. Запропоновані інженерні рішення спрямовані на створення безпечних і комфортних умов праці, що сприяють підвищенню продуктивності та зниженню професійних ризиків.

4.1 Ергономічні проблеми безпеки життєдіяльності

Ергономіка є ключовим фактором забезпечення безпеки під час тривалої роботи за комп’ютером, що становить основу розробки вебсайтів. Основними ергономічними ризиками для розробників є порушення постави, перенапруження зорового апарату, синдром зап’ястного каналу та зниження працездатності через незручну організацію робочого місця. Ці проблеми можуть призводити до професійних захворювань, якщо не вжити відповідних заходів.

Організація робочого місця відіграє вирішальну роль у зниженні ергономічних ризиків. Робочий стіл має бути спроектований так, щоб монітор розташовувався на рівні очей, а відстань до екрана становила 50–70 см [34]. Це зменшує навантаження на шийний відділ хребта та очі. Крісло з регульованою висотою та підтримкою поперекового відділу [35] забезпечує комфортне положення тіла, що знижує ризик розвитку болю в спині. Клавіатура та миша мають бути розміщені так, щоб руки залишалися в природному положенні, з кутом у ліктьовому суглобі близько 90 градусів. Таке розташування зменшує напруження в зап’ястях і попереджає розвиток синдрому зап’ястного каналу.

Освітлення робочого місця є ще одним важливим аспектом ергономічної безпеки. Недостатнє або надмірне освітлення може викликати зорову втому, головний біль і зниження концентрації. Оптимальний рівень освітлення для роботи за комп’ютером має бути рівномірним, без відблисків на екрані та становити становити 300–500 лк. Для цього доцільно використовувати світильники з розсіяним світлом, які забезпечують комфортне сприйняття зображення на моніторі. Якщо в приміщенні є природне освітлення, необхідно розташувати робоче місце так, щоб уникнути прямих сонячних променів, які створюють контрастні тіні [36].

Режим праці та відпочинку також має суттєвий вплив на безпеку життєдіяльності. Тривала безперервна робота за комп’ютером призводить до накопичення втоми, що негативно позначається на здоров’ї та продуктивності. Рекомендується робити перерви кожні 2 години тривалістю 10–15 хвилин, під час яких слід виконувати вправи для очей, такі як фокусування на віддалених об’єктах, та легку фізичну розминку для м’язів шиї, плечей і спини [37]. Ці заходи сприяють відновленню кровообігу та зниженню фізичного напруження.

Психофізіологічний комфорт є невід’ємною частиною ергономічної безпеки. Монотонність роботи за комп’ютером може викликати емоційне вигорання та зниження мотивації. Для профілактики таких станів рекомендується облаштувати робоче місце елементами, що сприяють психологічному розвантаженню, наприклад, використовувати спокійні кольори в інтер’єрі, такі як світло-зелений або блакитний, які заспокоюють нервову систему. Додавання кімнатних рослин або періодична зміна виду діяльності, наприклад, короткі консультації з колегами, також сприяють підтриманню психоемоційного балансу.

Отже, вирішення ергономічних проблем безпеки життєдіяльності під час розробки вебсайту “SoundKitchen” передбачає комплексний підхід, що включає правильну організацію робочого місця, оптимізацію освітлення, дотримання режиму праці та відпочинку, а також врахування психофізіологічних факторів.

Ці заходи дозволяють знизити ризики для здоров’я розробників і підвищити ефективність їхньої роботи.

4.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

Охорона праці користувачів персональних комп’ютерів є важливим аспектом забезпечення безпеки під час розробки вебсайтів. Робота з комп’ютерною технікою пов’язана з потенційними небезпеками, такими як ураження електричним струмом, вплив електромагнітних полів, а також шкідливі фактори виробничого середовища, зокрема шум, недостатня вентиляція чи неправильне освітлення. Для створення безпечних умов праці необхідно враховувати низку вимог, які стосуються електробезпеки, гігієнічних умов і ергономічної організації робочого місця.

Електробезпека є пріоритетним аспектом під час роботи з комп’ютерами. Усі електроприлади, що використовуються в процесі розробки, повинні бути справними та мати надійну ізоляцію. Для захисту від ураження електричним струмом необхідно забезпечити заземлення комп’ютерного обладнання, що дозволяє відводити надлишкову напругу в разі несправності [38]. Використання пристроїв захисного відключення (ПЗВ) є ефективним рішенням для миттєвого припинення подачі електроенергії при виявленні витоку струму. Регулярна перевірка стану електропроводки та обладнання допомагає своєчасно виявляти та усувати потенційні загрози.

Гігієнічні умови виробничого середовища відіграють важливу роль у забезпеченні безпеки. Приміщення, де працюють розробники, має бути обладнане системою вентиляції, яка забезпечує постійний приплив свіжого повітря та підтримує комфортну температуру в межах 20–24°C. Відносна вологість повітря повинна бути на рівні 40–60%, а швидкість руху повітря – не більше 0,3 м/с, щоб уникнути пересушування слизових оболонок очей і дихальних шляхів [39]. Для зниження рівня шуму, який може створювати комп’ютерна техніка чи зовнішні джерела, рекомендується використовувати

шумопоглинальні матеріали, такі як акустичні панелі, або розміщувати робочі місця подалі від джерел шуму.

Електромагнітні поля, що генеруються комп'ютерами, можуть негативно впливати на здоров'я, викликаючи втому, головний біль або порушення сну. Для зменшення цього впливу необхідно використовувати сучасні монітори з низьким рівнем випромінювання та дотримуватися рекомендованої відстані до екрана (50–70 см). Періодичні перерви в роботі [37] дозволяють знизити накопичувальний ефект електромагнітного впливу, а правильне розташування обладнання, наприклад, уникнення розміщення системних блоків поблизу робочого місця, додатково зменшує ризик.

Організаційні заходи з охорони праці включають регулярні медичні огляди для розробників, які постійно працюють із комп'ютерами. Такі огляди дозволяють своєчасно виявляти порушення зору, опорно-рухового апарату чи інші професійні захворювання. Проведення інструктажів з охорони праці є обов'язковим для ознайомлення працівників із правилами безпечної експлуатації обладнання та діями в аварійних ситуаціях. Наприклад, у разі виявлення несправності комп'ютера необхідно негайно відключити його від мережі та повідомити відповідальну особу [40].

Таким чином, вимоги охорони праці для користувачів ПК під час розробки вебсайтів охоплюють електробезпеку, гігієнічні умови, захист від електромагнітних полів і організаційні заходи. Ці рішення забезпечують безпеку та сприяють ефективній роботі.

4.3 Висновки до четвертого розділу

Аналіз ергономічних проблем безпеки життєдіяльності та вимог охорони праці для користувачів персональних комп'ютерів у процесі розробки вебсайту "SoundKitchen" дозволив розробити комплексний підхід до забезпечення безпечних і комфортних умов праці. Продуманий дизайн робочого місця, включаючи оптимальне розташування обладнання та використання

регульованого крісла, знижує фізичне навантаження та попереджає професійні захворювання, такі як зорова втома чи порушення постави. Належне освітлення та психофізіологічний комфорт, досягнутий завдяки спокійним кольорам і рослинам, сприяють підтриманню продуктивності та зниженню стресу.

Заходи електробезпеки, такі як заземлення обладнання та використання пристроїв захисного відключення, мінімізують ризик ураження струмом. Комфортні гігієнічні умови, включаючи вентиляцію, оптимальну температуру та зниження шуму, створюють сприятливе виробниче середовище. Захист від електромагнітних полів забезпечується сучасними моніторами та правильним розташуванням обладнання. Регулярні медичні огляди та інструктажі з охорони праці дозволяють своєчасно виявляти ризики та підвищувати обізнаність працівників.

Запропоновані рішення є практичними та спрямованими на довгострокове забезпечення безпеки. Вони не лише знижують ризики для здоров’я, але й сприяють ефективній реалізації проєкту. Для підтримання безпеки рекомендується регулярно перевіряти стан обладнання, проводити інструктажі та залучати розробників до профілактичних оглядів. Такий підхід гарантує стабільну роботу над проєктом “SoundKitchen” і може бути адаптований для інших ІТ-проєктів із тривалою роботою за комп’ютером.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи освітнього рівня «Бакалавр» було розроблено вебсайт інтернет-магазину музичних інструментів «SoundKitchen», призначений для забезпечення зручного та ефективного користувацького досвіду для музикантів різного рівня підготовки.

У першому розділі кваліфікаційної роботи:

– визначено актуальність створення спеціалізованого інтернет-магазину музичних інструментів, враховуючи зростання попиту на онлайн-покупки та обмеженість асортименту в локальних магазинах;

– проведено порівняльний аналіз конкурентних рішень, що дозволив виявити їхні недоліки, такі як складна навігація чи відсутність персоналізованих рекомендацій;

– обґрунтовано необхідність розробки нового вебсайту, який відповідає потребам цільової аудиторії;

сформульовано проблему, мету та завдання створення програмної системи.

У другому розділі кваліфікаційної роботи:

– досліджено принципи розробки вебдодатків, включаючи моделі життєвого циклу та сучасні підходи до проєктування;

– обґрунтовано вибір гнучкої Agile-методології[17] та відповідних методів дослідження для проєкту;

– сформульовано функціональні вимоги (управління каталогом, кошик, оформлення замовлень, рекомендації) та нефункціональні вимоги (продуктивність, безпека, адаптивність);

– обґрунтовано вибір технологічного стеку (React, Node.js[19], MongoDB[20]) для створення ефективної та масштабованої платформи.

У третьому розділі кваліфікаційної роботи:

– описано архітектуру системи, яка базується на багатошаровій клієнт-серверній моделі з чітким розподілом фронтенду, бекенду та рівня даних;

– продемонстровано користувацький інтерфейс і функціональні можливості, включаючи адаптивний дизайн, інтуїтивну навігацію, систему відгуків і багатокроковий процес оформлення замовлень;

– проведено тестування системи серед цільової аудиторії, що підтвердило її зручність, продуктивність (час завантаження 1,8 секунди) і відповідність вимогам, а також виявило напрямки для вдосконалення, такі як додавання порівняння товарів.

У розділі «Безпека життєдіяльності, основи охорони праці» проаналізовано заходи забезпечення електробезпеки для ІТ-фахівців, включаючи профілактику електричних аварій і використання сертифікованого обладнання для зниження ризиків ураження струмом. Окремо розглянуто вплив ефективної організації робочого процесу, зокрема застосування гнучких методологій управління проєктами, на зменшення фізичних і психоемоційних навантажень. Такі підходи сприяють профілактиці перевтоми, запобіганню професійного вигорання та підтриманню здоров’я й продуктивності розробників.

ПЕРЕЛІК ДЖЕРЕЛ

1 Muztorg.ua [Електронний ресурс]. Режим доступу: https://muztorg.ua/uk/ (дата звернення: 10.05.2025).

2 Jam.ua: Музичні інструменти [Електронний ресурс]. Режим доступу: https://jam.ua/ua/ (дата звернення: 10.05.2025).

3 SoundMaster.ua [Електронний ресурс]. Режим доступу: https://soundmaster.ua/ua (дата звернення: 10.05.2025).

4 Інтернет-магазин LuxPRO™ [Електронний ресурс]. Режим доступу: https://luxpro.ua/ (дата звернення: 10.05.2025).

5 Музичні інструменти в MuzLine магазин №1 [Електронний ресурс]. Режим доступу: https://muzline.ua/ (дата звернення: 10.05.2025).

6 Best Practices for E-commerce Website Design [Електронний ресурс]. Режим доступу: https://www.shopify.com/blog/ecommerce-website-design (дата звернення: 23.05.2025).

7 Ecommerce (Learn About the Evolution of Online Shopping) [Електронний ресурс]. Режим доступу: https://www.bigcommerce.com/articles/ecommerce/ (дата звернення: 23.05.2025).

8 Project Management Life Cycle: The 5 Phases Explained [Електронний ресурс]. Режим доступу: https://www.coursera.org/articles/project-management-lifecycle (дата звернення: 23.05.2025).

9 Що таке система управління замовленнями [Електронний ресурс]. Режим доступу: https://www.logos3pl.com/uk/glossary/order-management-system/ (дата звернення: 23.05.2025).

10 Тестування програмного забезпечення [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення (дата звернення: 12.05.2025).

11 Software Architecture Guide [Електронний ресурс]. Режим доступу: https://martinfowler.com/architecture/ (дата звернення: 12.05.2025).

12 What is SDLC? - Software Development Lifecycle Explained [Електронний ресурс]. Режим доступу: https://aws.amazon.com/what-is/sdlc/ (дата звернення: 12.05.2025).

13 Cascading and iterative models of software and application development [Електронний ресурс]. Режим доступу: https://webwizard.com.pl/en/blog/2021/11/11/cascading-and-iterative-models-of-software-and-application-development/ (дата звернення: 12.05.2025).

14 MERN Stack Explained [Електронний ресурс]. Режим доступу: https://www.mongodb.com/resources/languages/mern-stack (дата звернення: 12.05.2025).

15 Don't repeat yourself [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/Don%27t_repeat_yourself (дата звернення: 12.05.2025).

16 KISS principle [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/KISS_principle (дата звернення: 12.05.2025).

17 What is Agile? | Agile 101 [Електронний ресурс]. Режим доступу: https://agilealliance.org/agile101/ (дата звернення: 12.05.2025).

18 Learn React [Електронний ресурс]. Режим доступу: https://react.dev/learn (дата звернення: 12.05.2025).

19 Node.js Docs [Електронний ресурс]. Режим доступу: https://nodejs.org/docs/latest/api/ (дата звернення: 12.05.2025).

20 MongoDB Docs [Електронний ресурс]. Режим доступу: https://www.mongodb.com/docs/manual/ (дата звернення: 12.05.2025).

21 What is a REST API? [Електронний ресурс]. Режим доступу: https://www.redhat.com/en/topics/api/what-is-a-rest-api/ (дата звернення: 12.05.2025).

22 UML-Diagrams.org: Use Case Diagrams Tutorial [Електронний ресурс]. Режим доступу: https://www.uml-diagrams.org/use-case-diagrams.html (дата звернення: 23.05.2025).

23 Клачко Л. Ю. Класифікація баз даних NoSQL в залежності від моделі представлення даних // Збірник тез доповідей VI Міжнародної науково-технічної конференції молодих учених та студентів „Актуальні задачі сучасних технологій“, 16-17 листопада 2017 року. — Т. : ТНТУ, 2017. — Том II. — С. 207. (дата звернення: 09.06.2025).

24 Redux Toolkit Documentation [Електронний ресурс]. Режим доступу: https://redux-toolkit.js.org/ (дата звернення: 23.05.2025).

25 React Router Documentation [Електронний ресурс]. Режим доступу: https://reactrouter.com/en/main (дата звернення: 23.05.2025).

26 Tailwind CSS Documentation [Електронний ресурс]. Режим доступу: https://tailwindcss.com/docs/installation (дата звернення: 23.05.2025).

27 Material UI Documentation [Електронний ресурс]. Режим доступу: https://mui.com/material-ui/getting-started/ (дата звернення: 23.05.2025).

28 Axios Documentation [Електронний ресурс]. Режим доступу: https://axios-http.com/docs/intro (дата звернення: 23.05.2025).

29 JSON Web Tokens (JWT) [Електронний ресурс]. Режим доступу: https://jwt.io/introduction (дата звернення: 23.05.2025).

30 bcrypt Documentation [Електронний ресурс]. Режим доступу: https://www.npmjs.com/package/bcrypt (дата звернення: 23.05.2025).

31 CORS (Cross-Origin Resource Sharing) [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS (дата звернення: 23.05.2025).

32 LiqPay Documentation [Електронний ресурс]. Режим доступу: https://www.liqpay.ua/documentation/ (дата звернення: 23.05.2025).

33 Web Accessibility Guidelines (WCAG) [Електронний ресурс]. Режим доступу: https://www.w3.org/WAI/standards-guidelines/wcag/ (дата звернення: 23.05.2025).

34 ДСТУ ISO 9241-5:2004. Ергономічні вимоги до роботи з відеотерміналами в офісі. Частина 5. Вимоги до компонування робочого місця та до робочої пози [Електронний ресурс] – Режим доступу:

https://dnaop.com/html/2286/doc-ДСТУ_ISO_9241-5_2004 (дата звернення: 05.06.2025).

35 ДСТУ 7951:2015. Дизайн і ергономіка. Крісло оператора. Загальні ергономічні вимоги [Електронний ресурс] – Режим доступу: https:// dnaop.com/html/61769/doc-ДСТУ_7951_2015 (дата звернення: 05.06.2025).

36 ДБН В.2.5-28-2018 “Природне і штучне освітлення” [Електронний ресурс] – Режим доступу: https://e-construction.gov.ua/laws_detail/3074958732556240833 (дата звернення: 05.06.2025).

37 ISO 9241-5:2024 “Ergonomics of human-system interaction” [Електронний ресурс] – Режим доступу: https://www.iso.org/standard/86222.html (дата звернення: 05.06.2025).

38 Правил улаштування електроустановок [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/rada/show/v0476732-17 (дата звернення: 05.06.2025).

39 ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 05.06.2025).

40 НПАОП 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/rada/show/va042282-99 (дата звернення: 05.06.2025).

41 Шимчук, Г. В., Маєвський, О. В., & Назаревич, О. Б. (2016). Конспект лекцій з дисципліни «Розподілені системи моніторингу та керування».

42 Шимчук, Г., Голотенко, О., & Золотий, Р. З. (2022). Основні проблеми та загрози хмарної безпеки. Матеріали X науково-технічної конференції „Інформаційні моделі, системи та технології “Тернопільського національного технічного університету імені Івана Пулюя, 59-60.

43 Шимчук, Г. В., Маєвський, О. В., & Назаревич, О. Б. (2016). Методичні вказівки до самостійної роботи студентів та модульного контролю знань з

дисципліни «Розподілені системи моніторингу та керування» для студентів освітнього рівня «бакалавр» спеціальності 125–«Кібербезпека».

44 Шимчук, Г. В., Маєвський, О. В., Назаревич, О. Б., & Стадник, М. А. (2016). Конспект лекцій з дисципліни «Грід-системи та технології хмарних обчислень» для студентів освітніх рівнів «спеціаліст»,«магістр» 122 «Комп'ютерні науки та інформаційні технології».

45 Шимчук, Г., Шевченко, Н., Швирло, К., & Гарматюк, Н. (2025). Система відновлення даних у бездротових сенсорних мережах на основі машинного навчання. Herald of Khmelnytskyi National University. Technical sciences, 353(3.2), 246-250.

46 Шимчук, Г. В., Маєвський, О. В., & Назаревич, О. Б. (2016). Конспект лекцій з дисципліни Комп'ютерна графіка для студентів освітнього рівня «бакалавр» спеціальності 125 «Кібербезпека».

ДОДАТКИ

Додаток А

Приклади коду серверної частини сайту

Лістинг А1 – вміст файлу index.js

```
const express = require('express');
const cors = require('cors');
const app = express();
app.use(express.json());
app.use(cors());
app.get("/", (req, res) => {
    return res.status(200).send({
        message: "API працює справно",
        status: true
    });
});
const authRoutes = require('./routes/auth.route');
app.use('/auth', authRoutes.router);
const userRoutes = require('./routes/user.route');
app.use('/api/users', userRoutes.router);
const productRoutes = require('./routes/product.route');
app.use('/api/products', productRoutes.router);
const adminProductRoutes = require('./routes/adminProduct.route');
app.use('/api/admin/products', adminProductRoutes.router);
const cartRoutes = require('./routes/cart.route');
app.use('/api/cart', cartRoutes.router);
const cartItemRoutes = require('./routes/cartItem.route');
app.use('/api/cart_items', cartItemRoutes.router);
const orderRoutes = require('./routes/order.route');
app.use('/api/orders', orderRoutes.router);
const adminOrderRoutes = require('./routes/admin.route');
app.use('/api/admin/orders', adminOrderRoutes.router);
const reviewRoutes = require('./routes/review.route');
app.use('/api/reviews', reviewRoutes.router);
const ratingRoutes = require('./routes/rating.route');
app.use('/api/ratings', ratingRoutes.router);
const paymentRoutes = require('./routes/payment.route');
app.use('/api/payments', paymentRoutes.router);
const categoryRoutes = require('./routes/category.route');
app.use('/api/categories', categoryRoutes.router);
module.exports = {
    app
};
```

Лістинг А2 – вміст файлу server.js

```
const { app } = require(".");
const { connectToDatabase } = require("./config/db");

const PORT = 5454;
```

```
app.listen(PORT, async () => {
    await connectToDatabase();
    console.log("API Сервер запущений на порті: ", PORT);
});
```

Лістинг А3 – вміст файлу config/db.js

```
const mongoose = require("mongoose")
const mongodbURL = process.env.MONGODB_URL;
const connectToDatabase = () => {
    return mongoose.connect(mongodbURL);
}
module.exports = { connectToDatabase };
```

Лістинг А4 – вміст файлу config/jwtProvider.js

```
const jwt = require('jsonwebtoken');
const SECRET_KEY = process.env.SECRET_KEY;
const generateToken = (userId) => {
    const token = jwt.sign({ userId }, SECRET_KEY, { expiresIn:
"48h" });
    return token;
}
const getUserIdFromToken = (token) => {
    const decoded = jwt.verify(token, SECRET_KEY);
    return decoded.userId;
}
module.exports = {
    generateToken,
    getUserIdFromToken
};
```

Лістинг А5 – вміст файлу config/liqPayClient.js

```
const LiqPay = require('../libs/liqPay');
const public_key = process.env.PUBLIC_KEY;
const private_key = process.env.PRIVATE_KEY;
const liqPay = new LiqPay(
    public_key,
    private_key
);
module.exports = {
    liqPay
};
```

Лістинг А6 – вміст файлу middleware/authenticate.js

```
const jwtProvider = require("../config/jwtProvider");
const userService = require("../services/user.service");
const authenticate = async (req, res, next) => {
```

```
    try {
        const token = req.headers.authorization?.split(" ")[1];
        if (!token) {
            return res.status(401).send({ error: "Токен не
знайдено!" })
        }
        const userId = jwtProvider.getUserIdFromToken(token);
        const user = await userService.findUserById(userId);
        req.user = user;
    } catch (error) {
        return res.status(500).send({ error: error.message })
    }
    next();
}
module.exports = {
    authenticate
}
```

Лістинг А7 – вміст файлу services/cart.service.js

```
const Cart = require('../models/cart.model');
const CartItem = require('../models/cartItem.model');
const Product = require('../models/product.model');

async function createCart(user) {
    try {
        const cart = new Cart({ user });
        const createdCart = await cart.save();
        return createdCart;
    } catch (error) {
        throw new Error(`Error creating cart: ${error.message}`);
    }
};
async function findUserCart(userId) {
    try {
        let cart = await Cart.findOne({ user: userId });
        let cartItems = await CartItem.find({ cart: cart._id
}).populate("product");
        cart.cartItems = cartItems;
        let totalPrice = 0;
        let totalDiscountedPrice = 0;
        let totalItems = 0;
        for (let cartItem of cart.cartItems) {
            totalPrice += cartItem.price;
            totalDiscountedPrice += cartItem.discountedPrice;
            totalItems += cartItem.quantity;
        }
        cart.totalPrice = totalPrice;
        cart.totalItems = totalItems;
        cart.discount = totalPrice - totalDiscountedPrice;
        cart.totalDiscountedPrice = totalDiscountedPrice;
        return cart;
```

```
    } catch (error) {
        throw new Error(`Помилка знаходження корзини користувача:
${error.message}`);
    }
};
async function addItemToCart(userId, req) {
    try {
        const cart = await Cart.findOne({ user: userId });
        const product = await Product.findById(req.productId);
        const isPresent = await CartItem.findOne({ cart: cart._id,
product: product._id, color: req.color, userId });
        if (!isPresent) {
            const cartItem = new CartItem({
                product: product._id,
                cart: cart._id,
                quantity: 1,
                userId,
                price: product.price,
                color: req.color,
                discountedPrice: product.discountedPrice
            });
            const createdCartItem = await cartItem.save();
            cart.cartItems.push(createdCartItem);
            await cart.save();
            return "Item added to cart";
        } else {
            isPresent.quantity += 1;
            isPresent.price = isPresent.quantity * product.price;
            isPresent.discountedPrice = isPresent.quantity *
product.discountedPrice;
            await isPresent.save();
            return "Item quantity updated in cart";
        }
    } catch (error) {
        throw new Error(`Error adding item to cart:
${error.message}`)
    };
};
async function clearCart(userId) {
    try {
        const cart = await Cart.findOne({ user: userId });
        if (!cart) throw new Error("Cart not found");
        await CartItem.deleteMany({ cart: cart._id });
        cart.cartItems = [];
        cart.totalPrice = 0;
        cart.totalItems = 0;
        cart.totalDiscountedPrice = 0;
        cart.discount = 0;
        await cart.save();
        return { message: "Корзина очищена успішно" };
    } catch (error) {
```

```
        throw new Error(`Помилка очищення корзини:
${error.message}`);
    }
}
module.exports = {
    createCart,
    findUserCart,
    addItemToCart,
    clearCart
};
```

Лістинг А9 – вміст файлу services/cart.controller.js

```
const cartService = require("../services/cart.service");
const findUserCart = async (req, res) => {
    const user = await req.user;
    try {
        const cart = await cartService.findUserCart(user._id);
        return res.status(200).send(cart);
    } catch (error) {
        return res.status(500).send({ error: error.message })
    }
}
const addItemToCart = async (req, res) => {
    const user = await req.user;
    try {
        const cartItemStatus = await
cartService.addItemToCart(user._id, req.body);
        return res.status(201).send(cartItemStatus);
    } catch (error) {
        return res.status(500).send({ error: error.message })
    }
}
const clearCart = async (req, res) => {
    const user = await req.user;
    try {
        const result = await cartService.clearCart(user._id);
        return res.status(200).send(result);
    } catch (error) {
        return res.status(500).send({ error: error.message });
    }
};
module.exports = {
    findUserCart,
    addItemToCart,
    clearCart
}
```

Лістинг А10 – вміст файлу services/cart.route.js

```
const express = require('express');
```

```
const { authenticate } = require('../middleware/authenticate');
const cartController = require("../controller/cart.controller");
const router = express.Router();
router.get("/", authenticate, cartController.findUserCart);
router.put("/add", authenticate, cartController.addItemToCart);
router.delete("/clear", authenticate, cartController.clearCart);
module.exports = {
    router
}
```

Лістинг А11 – вміст файлу models/cart.model.js

```
const mongoose = require('mongoose');
const cartSchema = new mongoose.Schema({
    user: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "users",
        required: true
    },
    cartItems: [{
        type: mongoose.Schema.Types.ObjectId,
        ref: "cartItems",
        required: true
    }],
    totalPrice: {
        type: Number,
        required: true,
        default: 0
    },
    totalItems: {
        type: Number,
        required: true,
        default: 0
    },
    totalDiscountedPrice: {
        type: Number,
        required: true,
        default: 0
    },
    discount: {
        type: Number,
        required: true,
        default: 0
    }
});
const Cart = mongoose.model('cart', cartSchema);
module.exports = Cart;
```

Додаток Б

Приклади коду кілєнтської частини сайту

Лістинг Б1 – вміст файлу App.js

```
import { Route, Routes, useLocation } from 'react-router-dom';
import './App.css';
import CustomerRouters from './Routers/CustomerRouters';
import ScrollToTop from "./customer/components/ScrollToTop";
function App() {
    const { pathname } = useLocation();
    return (
        <div>
            <ScrollToTop pathName={pathname} />
            <Routes>
                <Route path='/*' element={<CustomerRouters
/>}></Route>
            </Routes>
        </div>
    );
}
export default App;
```

Лістинг Б2 – вміст файлу index.js

```
import ReactDOM from 'react-dom/client';
import { Provider } from 'react-redux';
import { BrowserRouter } from 'react-router-dom';
import App from './App';
import './index.css';
import reportWebVitals from './reportWebVitals';
import { store } from './State/store';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
    <BrowserRouter>
        <Provider store={store}>
            <App />
        </Provider>
    </BrowserRouter>
);

reportWebVitals();
```

Лістинг Б3 – вміст файлу customer/components/ScrollToTop.js

```
import { useEffect } from 'react';
export default function ScrollToTop(pathName) {
  useEffect(() => {
    window.scrollTo(0, 0);
  }, [pathName]);
```

```
  return null;
}
```

Лістинг Б4 – вміст файлу utils/recentlyViewed.js

```
const RECENTLY_VIEWED_KEY = 'soundkitchen_recently_viewed';
const MAX_RECENTLY_VIEWED = 10;
export const addToRecentlyViewed = (product) => {
  try {
    const recentlyViewed = getRecentlyViewed();

    const filteredProducts = recentlyViewed.filter(p => p._id !==
product._id);

    const updatedProducts = [product,
...filteredProducts].slice(0, MAX_RECENTLY_VIEWED);

    localStorage.setItem(RECENTLY_VIEWED_KEY,
JSON.stringify(updatedProducts));
  } catch (error) {
    console.error('Error saving to recently viewed:', error);
  }
};
export const getRecentlyViewed = () => {
  try {
    const stored = localStorage.getItem(RECENTLY_VIEWED_KEY);
    return stored ? JSON.parse(stored) : [];
  } catch (error) {
    console.error('Error loading recently viewed:', error);
    return [];
  }
};
export const clearRecentlyViewed = () => {
  try {
    localStorage.removeItem(RECENTLY_VIEWED_KEY);
  } catch (error) {
    console.error('Error clearing recently viewed:', error);
  }
};
export const getRecentlyViewedExcluding = (productId) => {
  const recentlyViewed = getRecentlyViewed();
  return recentlyViewed.filter(product => product._id !==
productId);
};
```

Лістинг Б5 – вміст файлу customer/pages/HomePage/HomePage.jsx

```
import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import HomeCarousel from
"../../components/HomeCarousel/HomeCarousel";
```

```
import HomeSectionCarousel from
"../../components/HomeSectionCarousel/HomeSectionCarousel";
import { findCategoryProducts } from
"../../../State/Product/Action";
const HomePage = () => {
    const dispatch = useDispatch();
    const { categoryProducts, categoryProductsLoading } =
useSelector((state) => state.product);
    const categories = [
        { key: 'featured', path: ["Інструменти", "Гітари"], name:
"🎸 Рекомендовані гітари", priority: 1 },
        { key: 'deals', path: ["Аудіо", "Навушники"], name: "🔥
Гарячі пропозиції - Аудіообладнання", priority: 2 },
        { key: 'recording', path: ["Звукозапис", "Мікрофони"],
name: "🎙️ Професійне обладнання для запису", priority: 3 },
        { key: 'keyboards', path: ["Інструменти", "Клавішні"],
name: "🎹 Клавішні та синтезатори", priority: 4 },
        { key: 'trending', path: ["DJ", "Контролери"], name: "📈
Популярне DJ обладнання", priority: 5 },
        { key: 'drums', path: ["Інструменти", "Барабани"], name:
"🥁 Барабани та перкусія", priority: 6 }
    ];    useEffect(() => {
        categories.forEach(category => {
            let sortType = "price_low";

            if (category.key === 'deals' || category.key ===
'featured') {
                sortType = "discount_high";
            } else if (category.key === 'trending') {
                sortType = "newest";
            }

            const tryFetchProducts = async (categoryPath) => {
                try {
                    await
dispatch(findCategoryProducts(category.key, {
                        categoryPath: categoryPath,
                        colors: [],
                        discount: category.key === 'deals' ? "10"
: "all",
                        sort: sortType,
                        stock: "all",
                        pageNumber: 1,
                        pageSize: 15,
                    }));
                } catch (error) {
                    console.warn(`Failed to fetch products for
${categoryPath.join(' -> ')}, trying fallback...`);

                    const englishFallback = {
                        'featured': ["Instruments", "Guitars"]
```

```
                        'deals': ["Audio", "Headphones"],
                        'recording': ["Recording", "Microphones"],
                        'keyboards': ["Instruments", "Keyboards"],
                        'trending': ["DJ", "Controllers"],
                        'drums': ["Instruments", "Drums"]
                    };

                    if (englishFallback[category.key]) {
                        try {
                            await
dispatch(findCategoryProducts(category.key, {
                                categoryPath:
englishFallback[category.key],
                                colors: [],
                                discount: category.key === 'deals'
? "10" : "all",
                                sort: sortType,
                                stock: "all",
                                pageNumber: 1,
                                pageSize: 15,
                            }));
                        } catch (fallbackError) {
                            console.error(`Failed to fetch
products for category ${category.key} even with fallback:`,
fallbackError);
                        }
                    }
                }
            };

            tryFetchProducts(category.path);
        });
    }, [dispatch]);
    const sortedCategories = [...categories].sort((a, b) =>
a.priority - b.priority);
    return (
        <div className="bg-gray-50">
            <HomeCarousel />
            {/* Product Carousels */}
            <div className="space-y-16 py-10 px-5 lg:px-10 max-w-
8xl mx-auto">
                {sortedCategories.map((category) => {
                    const products =
categoryProducts[category.key]?.content || [];
                    const isLoading =
categoryProductsLoading[category.key];

                    return (
                        <div key={category.key} className="bg-
white rounded-xl shadow-sm p-6">
                            {isLoading ? (
                                <div className="animate-pulse">
```

```
                                    <div className="h-8 bg-gray-
200 rounded w-1/3 mb-6"></div>
                                    <div className="flex space-x-
4">
                                        {[1, 2, 3, 4, 5].map((i)
=> (
                                            <div key={i}
className="flex-shrink-0 w-64">
                                                <div className="h-
48 bg-gray-200 rounded mb-4"></div>
                                                <div className="h-
4 bg-gray-200 rounded mb-2"></div>
                                                <div className="h-
4 bg-gray-200 rounded w-3/4"></div>
                                            </div>
                                        ))}
                                    </div>
                                </div>
                            ) : products.length > 0 ? (
                                <HomeSectionCarousel
                                    data={products}
                                    sectionName={category.name}
                                />
                            ) : null}
                        </div>
                    );
                })}
            </div>
        </div>
    );
};
export default HomePage;
```

Лістинг Б6 – вміст файлу Routers/CustomerRouters.jsx

```
import React from "react";
import { Route, Routes } from "react-router-dom";
import HomePage from '../customer/pages/HomePage/HomePage';
import Footer from "../customer/components/Footer/Footer";
import Navigation from
"../customer/components/Navigation/Navigation";
import Cart from "../customer/components/Cart/Cart";
import Product from "../customer/components/Product/Product";
import ProductDetails from
"../customer/components/ProductDetails/ProductDetails";
import Checkout from "../customer/components/Checkout/Checkout";
import Order from "../customer/components/Order/Order";
import OrderDetails from
"../customer/components/OrderDetails/OrderDetails";
import PaymentSuccess from
"../customer/components/Payment/PaymentSuccess";
```

```
import SearchResultsPage from
"../customer/pages/SearchResultsPage";
const CustomerRouters = () => {
    return (
        <div>
            <div>
                <Navigation />
            </div>
            <Routes>
                <Route path="/" element={<HomePage
/>}></Route>
                <Route path="/cart" element={<Cart />}></Route>
                <Route path="/products" element={<Product
/>}></Route>
                <Route path="/:levelOne/:levelTwo/:levelThree"
element={<Product />}></Route>
                <Route path="/:levelOne/:levelTwo"
element={<Product />}></Route>
                <Route path="/:levelOne" element={<Product
/>}></Route>
                <Route path="/product/:productId"
element={<ProductDetails />}></Route>
                <Route path="/checkout" element={<Checkout
/>}></Route>
                <Route path="/account/orders" element={<Order
/>}></Route>
                <Route path="/account/order/:orderId"
element={<OrderDetails />}></Route>
                <Route path="/payment/:orderId"
element={<PaymentSuccess />}></Route>
                <Route path="/search" element={<SearchResultsPage
/>}></Route>
            </Routes>
            <div>
                <Footer />
            </div>
        </div>
    );
};
export default CustomerRouters;
```