

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНТЕРНЕТ-
МАГАЗИНУ МУЗИЧНИХ ІНСТРУМЕНТІВ ТА АКСЕСУАРІВ
SOFTWARE DEVELOPMENT FOR AN ONLINE STORE OF MUSICAL
INSTRUMENTS AND ACCESSORIES

Виконав(ла): студент(ка) 4 курсу, групи КНТ-211

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

РОМАНОВСЬКА Є.І

(ПРІЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О

(ПРІЗВИЩЕ та ініціали)

Рецензент БАБЕНКО Н.В.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н., проф.
Сергій СУББОТІН
« » 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

РОМАНОВСЬКОЇ Єлизавети Іванівни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного забезпечення інтернет-магазину музичних інструментів та аксесуарів Software Development for an Online Store of Musical Instruments and Accessories

керівник проєкту (роботи) к.т.н., доцент, СТЕПАНЕНКО Олександр Олексійович

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «28» квітня 2025 року № 209

2. Строк подання студентом проєкту (роботи) 1 червня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області та постановка завдання, 2. Розробка архітектури програми, 3. Реалізація програмного забезпечення, 4. Тестування та експлуатація програми, 5. Технічне завдання, 6. Текст програми

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтролер	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання «15» квітня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Реалізація програмного забезпечення	4-5 тижні	Розділ 3
5	Тестування та експлуатація програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

_____ Єлизавета РОМАНОВСЬКА
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проекту (роботи)

_____ Олександр СТЕПАНЕНКО
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
98 с., 7 табл., 31 рис., 3 дод., 38 джерел

МАГАЗИН МУЗИЧНИХ ТОВАРІВ, ВЕБЗАСТОСУНОК, БАЗА
ДАНИХ, REACT JS, NODE JS, MONGODB.

Об'єкт роботи – розробка вебзастосунку для електронної комерції у сфері продажу музичних інструментів та аксесуарів.

Предмет роботи – методи, технології та інструменти розробки інтернет-магазину, що реалізує функції пошуку, фільтрації, перегляду та підбору товарів.

Мета роботи – створення інтернет-магазину, призначеного для продажу музичних інструментів і аксесуарів.

Матеріали, методи і технічні засоби: компонентний підхід і архітектура SPA (Single Page Application), REST -архітектура, персональний комп'ютер із ОС Windows, платформа Node.js, бібліотека React.js, база даних MongoDB Atlas.

Результати роботи. Досліджено актуальні рішення у сфері електронної комерції музичних товарів, проаналізовано вимоги до сучасного інтернет-магазину, розроблено архітектуру програмного забезпечення та реалізовано вебзастосунок Chord на основі технологій React.js, Node.js і MongoDB. Показано доцільність використання SPA та REST-архітектури для забезпечення гнучкості й масштабованості системи. Встановлено ефективність інтеграції сучасних механізмів захисту даних (хешування паролів, JWT-автентифікація).

Висновки. Поставленої мети досягнуто — створено сучасний інтернет-магазин із функціоналом, який відповідає вимогам ринку. Рекомендовано впроваджувати платформу у сфері музичної торгівлі.

Галузь використання. Результати роботи можуть застосовуватись у музичних магазинах, навчальних закладах, стартапах із онлайн-продажу товарів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 98 p., 7 tables, 31 figures, 3 appendix, 38 sources

MUSIC STORE, WEB APPLICATION, DATABASE, REACT JS, NODE JS, MONGODB.

The object of work is to develop a web application for e-commerce in the field of selling musical instruments and accessories.

The subject of the work is methods, technologies, and tools for developing an online store that implements the functions of searching, filtering, browsing, and selecting goods.

Purpose - to create an online store designed to sell musical instruments and accessories.

Materials, methods and techniques: component approach and architecture of SPA (Single Page Application), REST architecture, personal computer with Windows OS, Node.js platform, React.js library, MongoDB Atlas database.

Results. We have studied current solutions in the field of e-commerce of music products, analyzed the requirements for a modern online store, developed a software architecture and implemented the Chord web application based on React.js, Node.js and MongoDB technologies. The expediency of using SPA and REST-architecture to ensure the flexibility and scalability of the system is shown. The effectiveness of integrating modern data protection mechanisms (password hashing, JWT authentication) is established.

Conclusions. The goal has been achieved - a modern online store with functionality that meets market requirements has been created. It is recommended to implement the platform in the field of music trade.

Field of application. The results of the work can be used in music stores, educational institutions, and online sales startups.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	9
Вступ	10
1 Аналіз предметної області та постановка завдання	12
1.1 Характеристика ринку електронної комерції в Україні та світі	12
1.2 Аналіз існуючих онлайн-магазинів музичних інструментів	14
1.3 Вимоги до розробки сайту	16
1.4 Постановка задачі	21
1.5 Висновки до розділу 1	23
2 Розробка архітектури програми	25
2.1 Вибір інструментів та технологій розробки	25
2.1.1 Вибір технологій для розробки клієнтської частини	25
2.1.2 Вибір технологій для розробки серверної частини	26
2.1.3 Вибір середовища розробки та допоміжних інструментів	27
2.2 Вибір і проектування бази даних	30
2.3 Висновки до розділу 2	33
3 Реалізація програмного забезпечення	35
3.1 Реалізація клієнтської частини	35
3.2 Реалізація серверної частини	43
3.3 Панель адміністратора	50
3.4 Висновки до розділу 3	53
4 Тестування та експлуатація програми	55
4.1 Призначення програми	55
4.2 Умови виконання програми	55
4.3 Робота з програмою	56
4.3.1 Користування клієнтською частиною	56
4.3.2 Робота з панеллю адміністратора	63
Висновок	66

Перелік джерел посилання	68
Додаток А Технічне завдання.....	71
Додаток Б Текст програми.....	75
Додаток В Слайди презентації	90

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CSS	– Cascading Style Sheets;
HTML	– Hypertext Markup Language;
JS	– JavaScript;
NoSQL	– Not Only SQL;
REST	– Representational State Transfer;
SPA	– Single Page Application;
SQL	– Structured Query Language;
UI/UX	– User Interface/User Experience;
БД	– База даних;
ОС	– Операційна система;
ПЗ	– Програмне забезпечення;

ВСТУП

У сучасному світі розвиток електронної комерції набуває все більшого значення. Зростання кількості користувачів Інтернету та вдосконалення цифрових технологій створюють умови для активного впровадження онлайн-торгівлі у різні сфери економіки. Торгівля музичними інструментами та аксесуарами також переходить в онлайн-середовище. Проте існуючі платформи часто не враховують особливостей цільової аудиторії, мають обмежену функціональність або застарілий інтерфейс. Це створює потребу у розробці адаптивного, зручного та функціонального інтернет-магазину, який задовольняв би потреби музикантів і поціновувачів музичного обладнання.

Проблема, що розглядається у роботі, полягає у створенні доступного й ефективного способу взаємодії між продавцем музичних товарів і покупцем в онлайн-просторі. Відомі міжнародні платформи, як Thomann, Sweetwater, Guitar Center, задають високі стандарти, проте для локального ринку України важливо враховувати мовну адаптацію, логістику, специфіку попиту та цінову політику. В Україні діють кілька відомих магазинів із онлайн-платформами, таких як «Гітарний дім», «Музторг», «Soundmaster» та «Jam», проте їхні сайти не завжди відповідають сучасним стандартам зручності й адаптивності.

Актуальність теми обумовлена зростанням попиту на онлайн-торгівлю, особливо в умовах повномасштабної війни та пандемій, що обмежують фізичний доступ до магазинів. Молоді музиканти очікують інтуїтивно зрозумілі цифрові рішення для швидкого пошуку, порівняння та придбання товарів. Тому розробка інтернет-магазину з сучасним дизайном, гнучкою архітектурою та зручним інтерфейсом є актуальним і соціально значущим завданням.

Об'єктом дослідження є процес розробки веб-застосунку для електронної комерції у сфері продажу музичних інструментів та аксесуарів.

Предметом — методи, технології та інструменти розробки інтернет-магазину, що реалізує функції пошуку, фільтрації, перегляду та підбору товарів.

Метою роботи є створення функціонального та адаптивного інтернет-магазину під назвою Chord, призначеного для продажу музичних інструментів і аксесуарів із урахуванням потреб користувачів, технічної ефективності та сучасних стандартів веб-розробки.

Для досягнення мети було поставлено завдання:

- провести огляд сучасного стану електронної комерції, зокрема вітчизняних і зарубіжних рішень;
- проаналізувати вимоги до інтернет-магазину та сформулювати технічне завдання;
- розробити структуру бази даних для зберігання інформації про товари та користувачів;
- проаналізувати доступні інструменти і реалізувати клієнтську та серверну частини;
- провести тестування функціоналу та проаналізувати результати.

Методика дослідження включає аналіз наукових публікацій, технічної документації, існуючих рішень, розробку ПЗ із використанням сучасних фронтенд- і бекенд-технологій, тестування системи та перевірку відповідності вимогам.

Результати роботи можуть бути використані для створення комерційних і освітніх онлайн-платформ у галузі музики, впровадження у діяльність музичних магазинів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Характеристика ринку електронної комерції в Україні та світі

Сегмент електронної комерції як у світі, так і в Україні, демонструє динамічне зростання, що підтверджується сучасними статистичними звітами. Згідно з даними Digital 2025: Global Overview Report, на початку 2025 року чисельність користувачів Інтернету у світі досягла 5,64 мільярда, що становить майже 69% населення планети. Важливо зазначити, що за даними Statista, протягом останніх двох десятиліть спостерігається постійне зростання глобальної інтернет-аудиторії: якщо у 2005 році у світі налічувалося приблизно 1,02 мільярда користувачів Інтернету, то у 2024 році ця цифра сягнула 5,5 мільярда осіб (Рисунок 1.1) [1]. За 2024 рік кількість інтернет-користувачів у світі збільшилася на 144 мільйони, що є свідченням сталого зростання цифрової інтеграції населення [1] – [2].

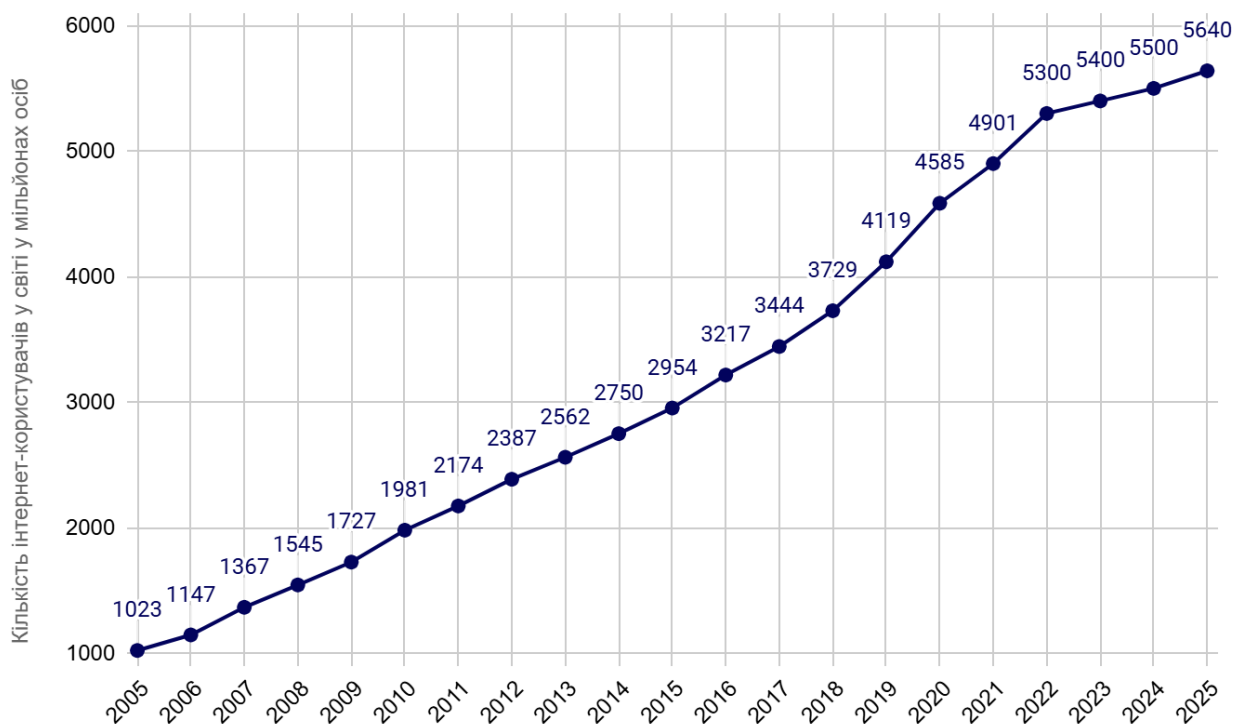


Рисунок 1.1 – Кількість інтернет-користувачів у світі, 2005-2025 рр., млн осіб

В українському сегменті мережі Інтернет на початку 2025 року налічувалося 31,5 мільйона користувачів, тобто понад 82% населення країни. Упродовж року цей показник виріс на 690 тисяч (+2,2%). Варто також зазначити, що на території України зафіксовано 56,4 мільйона мобільних підключень, що перевищує загальну чисельність населення в 1,47 рази. Це свідчить про надзвичайно високий рівень проникнення мобільних технологій та їхнє повсякденне використання [3].

Зростання електронної комерції є наслідком постійного підвищення цифрової грамотності громадян, розвитку інфраструктури онлайн-розрахунків і логістики, а також зміцнення довіри до онлайн-покупок. За аналітичними даними Digital 2025: Online Shopping Still Popping, у 2024 році світовий обсяг продажів через Інтернет збільшився на понад 500 мільярдів доларів, що дорівнює річному приросту в 14,6%. Кількість онлайн-покупців перевищила 2,5 мільярда, а це на 200 мільйонів більше, ніж роком раніше (+8,5%) [4].

Додатково, згідно зі статистикою Statista (2025), у 2024 році глобальні роздрібні продажі в електронній комерції досягли оцінки у шість трильйонів доларів США. Прогнози вказують на зростання цього показника на 31% у найближчі роки з очікуваним наближенням до восьми трильйонів доларів у 2028 році [5]. Це ще раз підтверджує актуальність і потенціал розвитку e-commerce на світовому ринку.

Серед найперспективніших напрямків онлайн-торгівлі вирізняється сегмент музичних інструментів та аксесуарів, адже багато музикантів та шанувальників музики надають перевагу саме покупкам через Інтернет. В Україні цей ринок представлений такими платформами, як «Гітарний дім», «Музторг», «Soundmaster» та «Jam», які пропонують широкий асортимент продукції. Проте низка платформ досі не відповідає сучасним стандартам зручності та мобільної адаптації, що ускладнює користувацький досвід.

Таким чином, враховуючи тенденцію зростання популярності онлайн-покупок і зростаючі вимоги споживачів до функціональних можливостей інтернет-магазинів, створення сучасної платформи Chord є важливим кроком

для покращення якості сервісу та задоволення потреб українських музикантів і поціновувачів музичних інструментів.

1.2 Аналіз існуючих онлайн-магазинів музичних інструментів

Для розробки сучасного онлайн-магазину музичних інструментів і аксесуарів надзвичайно важливо детально проаналізувати існуючі рішення на ринку. Це дозволяє виявити переваги та недоліки конкурентів, а також сформулювати чіткі вимоги до власного програмного продукту.

Світові лідери ринку — «Thomann», «Guitar Center», «Sweetwater» — демонструють максимально широкий спектр функцій: повністю адаптивний дизайн, розгалужену систему фільтрів, списки бажань, порівняння товарів, відеоогляди, індивідуальні консультації та додаткові сервіси, такі як гарантія і безкоштовна доставка, проте вони не націлені на український ринок [6] – [8].

Українські інтернет-магазини характеризуються досить широким асортиментом, проте функціонально та технічно мають значні відмінності. Для порівняння було обрано відомі онлайн-магазини, такі як «Гітарний дім», «SoundMaster», «МузТорг» та «JAM».

«Гітарний дім» [9] має досить простий і зрозумілий інтерфейс, який, однак, уже виглядає дещо застарілим. Сайт має дві версії: версія для мобільних пристроїв і повна. Респонсивність практично відсутня: при зменшенні ширини вікна браузера сайт не перебудовує структуру елементів, що призводить до незручного перегляду сторінок. Проте магазин має хороший функціонал, що включає онлайн-консультації, можливість додавати товари у список бажань, порівнювати обрані моделі, а також містить численні відеоогляди товарів, що позитивно впливає на користувацький досвід і сприяє зростанню довіри покупців.

Онлайн-магазин «SoundMaster» [10] характеризується дещо кращою адаптивністю порівняно з попереднім ресурсом, проте також має суттєві недоліки. Інтерфейс сайту виконаний у нейтральному стилі, однак не завжди

зручно відображається на мобільних пристроях. Респонсивність реалізована лише частково: деякі блоки й елементи сторінок адаптуються під мобільні пристрої, але цей процес нерідко супроводжується помилками й незручностями. Функціональність: є можливість додавати товари у список бажань, порівнювати їх між собою та отримувати онлайн-консультації.

Магазин «МузТорг» [11] можна вважати одним із провідних українських ресурсів у цій сфері. Його інтерфейс може виглядати дещо застарілим, проте є інтуїтивно зрозумілим. Значною перевагою є повна реалізація респонсивності: сторінки автоматично й комфортно адаптуються під будь-які розміри екранів, що робить використання з мобільних пристроїв зручним і приємним. МузТорг має список бажань, порівняння товарів, онлайн-консультації та відеоогляди на популярні товари.

Інтернет-магазин «JAM» [12] демонструє найвищий рівень серед аналізованих українських онлайн-платформ. Сучасний, мінімалістичний дизайн сайту повністю відповідає сучасним стандартам UI/UX, а реалізація респонсивності є взірцевою. Платформа ефективно пристосовується до перегляду на різних пристроях, забезпечуючи користувачам комфортне взаємодію з магазином як на комп'ютерах, так і на смартфонах. JAM пропонує користувачам максимально розширені функції: додавання товарів до списку бажань, порівняння за технічними характеристиками, детальні відеоогляди та професійні онлайн-консультації. Завдяки цьому JAM є чудовим прикладом того, як сучасні інтернет-магазини можуть задовольняти очікування клієнтів.

Розроблений у рамках цієї дипломної роботи інтернет-магазин Chord орієнтований на досягнення рівня, близького до лідерів українського ринку. Chord має інтуїтивний інтерфейс, зручний каталог, ефективну фільтрацію та сучасний дизайн, створений із урахуванням актуальних трендів у сфері веб-дизайну. Платформа повністю респонсивна й забезпечує зручну взаємодію з користувачем на пристроях будь-якого формату. Список бажань реалізовано якісно, що дозволяє клієнтам зручно відкладати та пізніше переглядати цікаві для них товари. Водночас, аналіз конкурентів дозволяє окреслити подальші

напрямки розвитку Chord, зокрема: впровадження порівняння товарів, онлайн-консультацій та інтеграції відеооглядів (Таблиця 1.1). Це не є критичним недоліком, оскільки базовий функціонал магазину вже забезпечує комфортний і повноцінний досвід користування, але додавання зазначених функцій у майбутньому суттєво підвищить конкурентоспроможність платформи на ринку.

Таблиця 1.1 – Порівняння існуючих рішень з розробленим

	Гітарний дім	МузТорг	Soundmaster	Jam	Chord
Сучасний дизайн	+	±	±	+	+
Респонсивність	-	+	±	+	+
Список Бажань	+	+	+	+	+
Порівняння товарів	+	+	+	+	-
Онлайн- консультації	+	+	+	+	-
Відеоогляди	+	+	-	+	-

1.3 Вимоги до розробки сайту

Створення інтернет-магазину Chord визначалося прагненням зробити платформу максимально зручною та функціональною для різних категорій користувачів — від початківців до досвідчених музикантів і педагогів. В основі

вимог лежить орієнтація на потреби сучасного користувача, який цінує швидкий доступ до якісної інформації, простоту взаємодії та надійність у питаннях захисту персональних даних.

Функціональні вимоги:

а) актуальний і деталізований каталог:

1) усі товари в каталозі Chord мають не лише базову інформацію (назва, категорія, підкатегорія, бренд, ціна, зображення), а й розширені характеристики, які залежать від підкатегорії інструмента;

2) кожен продукт супроводжується детальним описом і якісними фотографіями, що дозволяє музикантам швидко знаходити саме ті товари, які відповідають їхнім потребам, порівнювати моделі й робити обґрунтований вибір без ризику помилитися;

б) розвинена система категоризації, фільтрації, сортування та пошуку: користувачі можуть відбирати продукцію за різними властивостями, а функція пошуку за назвою є особливо цінною для тих, хто вже визначився із моделлю інструмента – це дозволяє знайти потрібний товар буквально за кілька секунд;

в) персоналізований простір для зареєстрованих користувачів: авторизовані користувачі мають змогу створювати та зберігати власний список бажань і кошик, ці дані фіксуються у базі даних і залишаються доступними незалежно від пристрою чи сесії, відтак, повертаючись на сайт після певного часу, користувач побачить свої обрані товари та попередні вибори, що особливо важливо для планування майбутніх покупок чи порівняння кількох варіантів;

г) тимчасовий простір для гостей: незареєстровані відвідувачі також можуть додавати товари до кошика і списку бажань, однак ці дані зберігаються лише протягом однієї сесії, а після оновлення сторінки або закриття браузера перелік вибраних товарів обнуляється, що стимулює користувача до реєстрації для збереження виборів надовше;

д) адміністративна панель для керування асортиментом: адміністратор отримує простий інтерфейс для додавання, видалення й перегляду товарів, що

дає змогу оперативно оновлювати пропозиції відповідно до попиту чи наявності на складі;

е) Безпека облікових записів:

1) збереження паролів у зашифрованому вигляді – це гарантує безпеку персональних даних та відповідає сучасним вимогам до захисту інформації в онлайн-сервісах;

2) реалізація авторизації через унікальний e-mail (логін) і пароль, що спрощує процес ідентифікації та дозволяє ефективно уникати дублювання облікових записів;

3) забезпечення захищеної автентифікації та ідентифікації користувачів – це особливо важливо для безпечної взаємодії із персональними та адміністративними функціями сайту.

Нефункціональні вимоги:

- сучасний та інтуїтивно зрозумілий інтерфейс, що відповідає принципам UX/UI-дизайну – сайт повинен бути привабливим і зрозумілим навіть для користувачів без спеціальної підготовки, що підвищує ймовірність повторних відвідувань;

- адаптивний (респонсивний) дизайн, який забезпечує коректну роботу сайту на будь-яких пристроях – це критично для музикантів, які часто користуються смартфонами й планшетами у процесі навчання чи гастролей;

- стабільність роботи основного функціоналу у процесі тестування (відсутність критичних помилок під час типового використання);

- надійність зберігання та захисту персональних даних користувачів, що дозволяє уникнути витоків і зловживань інформацією;

- відсутність критичних затримок або збоїв при роботі з каталогом товарів, пошуком, фільтрацією, списком бажань і кошиком – це забезпечує комфорт при взаємодії з платформою й формує позитивний імідж магазину.

Обмеження та особливості:

- система не підтримує оформлення й обробку замовлень та оплати, що дозволяє зосередити увагу на якісній реалізації основних функцій платформи;
- користувачі не мають можливості редагувати особисті дані через інтерфейс сайту – це тимчасове обмеження, яке знижує складність системи на першому етапі розвитку;
- адміністративна панель доступна за прямим посиланням, без додаткової аутентифікації за ролями – такий тимчасовий підхід обрано для спрощення структури прав доступу та швидкої адаптації на старті проекту;
- для взаємодії з базою даних використовуються дві схеми: товар (Product) і користувач (User) – це рішення сприяє простоті розробки та обслуговування системи, але залишає простір для подальшої деталізації моделі даних.

Для візуалізації функціональних вимог системи на етапі проєктування розроблено три діаграми прецедентів. Кожна відображає одного з трьох користувачів (адміністратор, гість, зареєстрований користувач) інтернет-магазину Chord та основні сценарії його взаємодії із системою (Рисунки 1.2 - 1.4).



Рисунок 1.2 – Діаграма прецедентів адміністратора

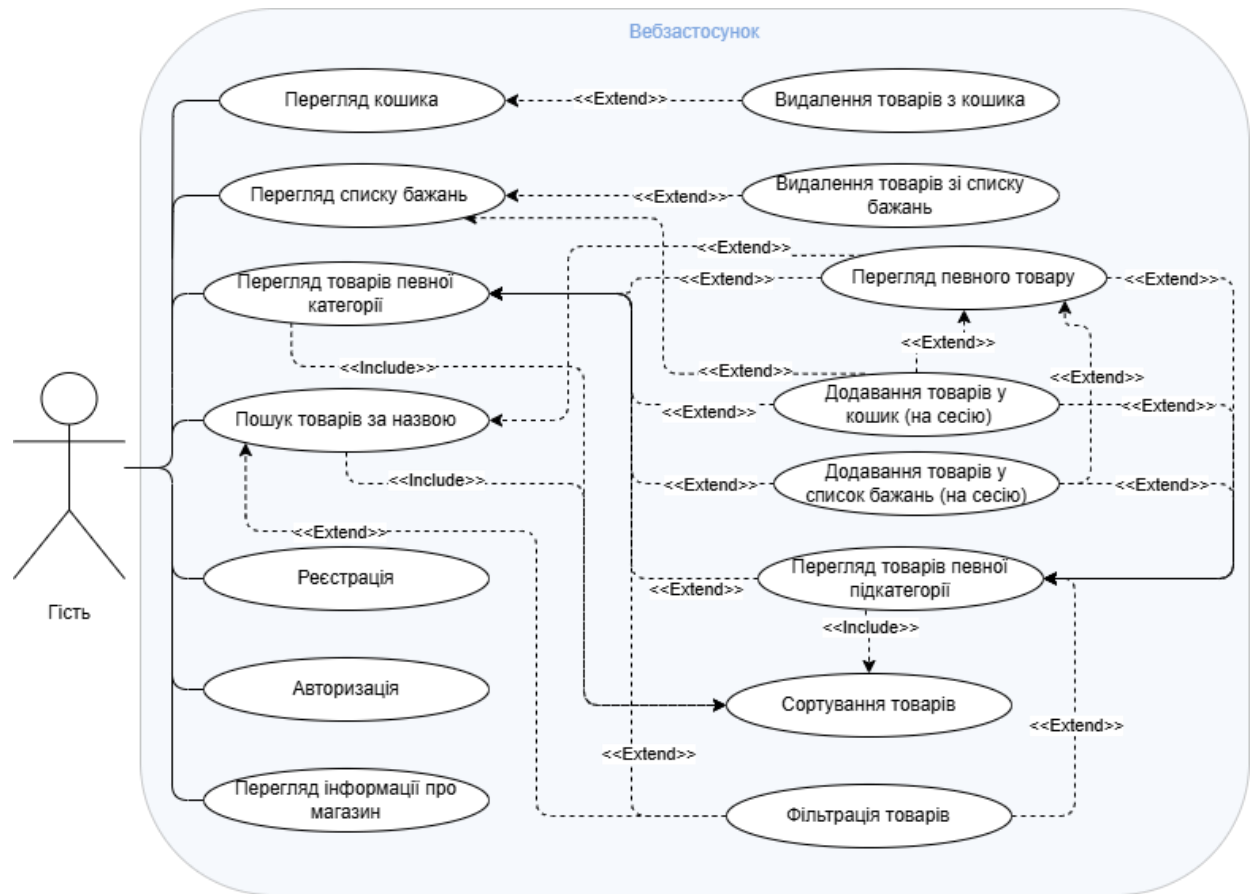


Рисунок 1.3 – Діаграма прецедентів гостя (неавторизованого користувача)

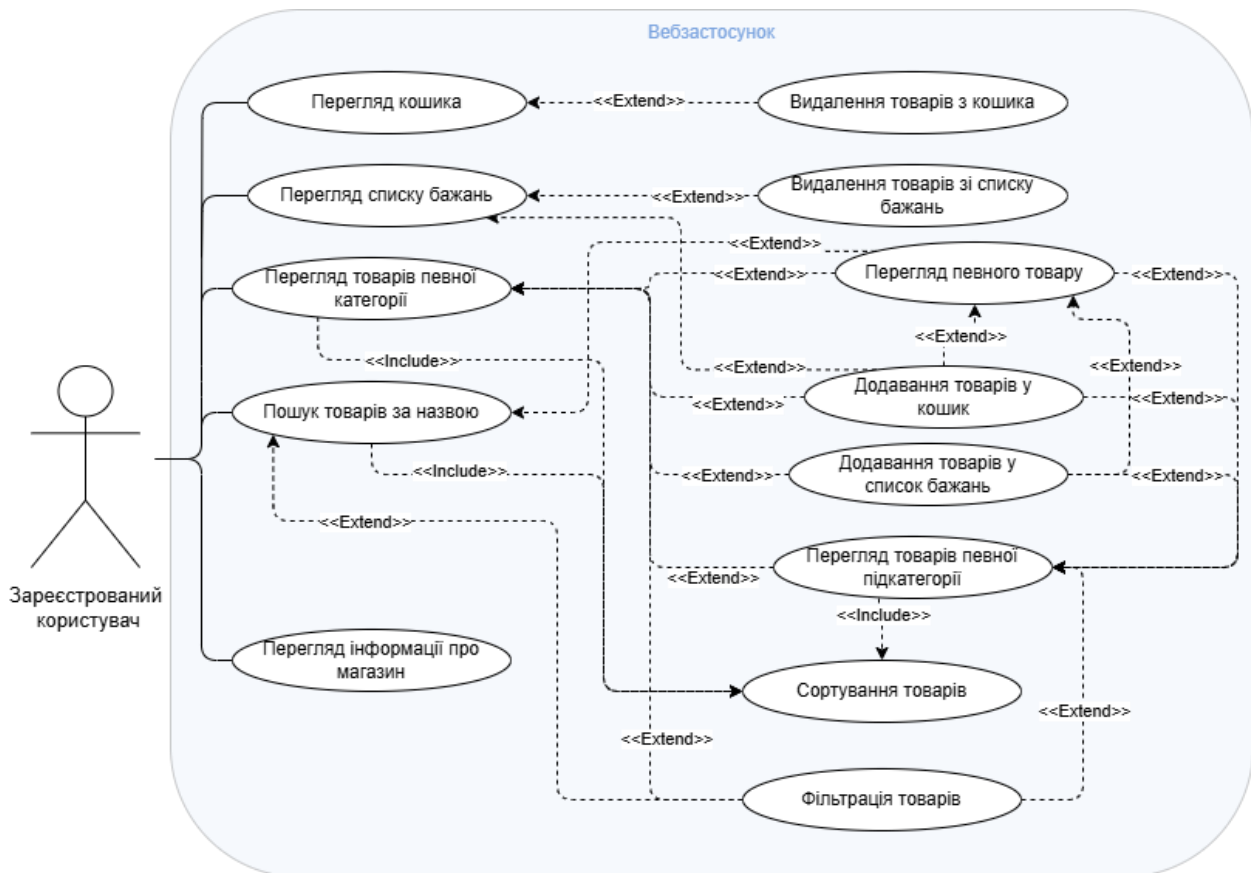


Рисунок 1.4 – Діаграма прецедентів зареєстрованого користувача

Таким чином, визначені вимоги до розробки інтернет-магазину Chord формують підґрунтя для створення веб-застосунку, який не лише відповідає очікуванням сучасних користувачів, але й забезпечує простоту, комфорт і безпеку взаємодії з платформою, залишаючи потенціал для подальшого функціонального розвитку у майбутньому.

1.4 Постановка задачі

В умовах сучасного цифрового середовища ключовим завданням дипломної роботи є розробка програмного забезпечення для інтернет-магазину музичних інструментів та аксесуарів Chord, яке має відповідати актуальним вимогам ринку, забезпечувати високу зручність використання, інформаційну безпеку, ефективну взаємодію із користувачами та створювати основу для подальшого розвитку платформи.

Постановка задачі в даному проєкті ґрунтується на комплексному аналізі потреб цільової аудиторії, тенденцій у сфері електронної комерції та функціональних сценаріїв, що є типовими для сучасних інтернет-магазинів. Необхідно сформулювати концепцію веб-застосунку, визначити його конкурентні переваги та обґрунтувати актуальність для ринку музичних товарів. Важливо описати та реалізувати основні сценарії використання системи, які мають охоплювати базові операції користувача — реєстрацію, авторизацію, перегляд каталогу, пошук, фільтрацію, роботу зі списком бажань і кошиком, а також забезпечити дії адміністратора, пов'язані з оновленням асортименту. При цьому слід передбачити адаптацію системи до реальних потреб кінцевого споживача: інтерфейс має бути не лише функціональним, а й інтуїтивно зрозумілим, швидким та доступним із різних пристроїв.

На наступному етапі необхідно провести аналіз сучасних підходів до проектування архітектури веб-застосунків. Зокрема, потрібно розглянути різні варіанти поділу системи на клієнтську та серверну частини, порівняти популярні бібліотеки та фреймворки для створення користувацького

інтерфейсу, такі як React.js, Vue.js чи Angular, оцінити їх переваги та недоліки для розробки саме інтернет-магазину. Важливо проаналізувати можливості різних серверних платформ і бібліотек для побудови бекенду — серед таких можуть бути Node.js з Express, Python з Django або Flask, PHP з Laravel тощо. Необхідно обґрунтувати вибір бази даних, зокрема розглянути як реляційні (наприклад, MySQL, PostgreSQL), так і нереляційні (наприклад, MongoDB) системи для організації зберігання та обробки даних.

Окрему увагу слід приділити питанням інформаційної безпеки. На цьому етапі варто дослідити сучасні методи захисту облікових записів користувачів, такі як збереження паролів у зашифрованому вигляді за допомогою спеціалізованих бібліотек, а також технології автентифікації та авторизації. Необхідно порівняти їх ефективність і доцільність використання у контексті розробки онлайн-магазину. Необхідно реалізувати адміністративну панель для керування товарами — додавання, перегляду й видалення позицій із каталогу, при цьому відповідно до вимог роботи не впроваджуючи розмежування доступу за ролями.

Окремим завданням є організація взаємодії між клієнтською та серверною частинами для забезпечення цілісності й узгодженості даних, стабільної роботи ключових функцій — формування списку бажань, кошика, реалізації пошуку й фільтрації товарів за різними параметрами, відображення повної та актуальної інформації про кожен товар.

Завершальним етапом передбачається проведення комплексного тестування всіх основних модулів системи для виявлення й усунення можливих помилок, підвищення стабільності та надійності роботи веб-застосунку. У процесі тестування необхідно оцінити працездатність системи під навантаженням, стабільність інтерфейсу на різних пристроях, а також відповідність реалізованого функціоналу раніше сформульованим вимогам.

Усі визначені завдання тісно пов'язані між собою і спрямовані на досягнення кінцевої мети — створення сучасного, якісного, безпечного й масштабованого інтернет-магазину музичних інструментів, здатного

задовольнити очікування різних категорій користувачів і залишати простір для розвитку системи у майбутньому.

1.5 Висновки до розділу 1

Проведений аналіз ринку електронної комерції показав, що цей сегмент активно зростає як у світі, так і в Україні, проте має свої особливості. Зокрема, для українських користувачів залишається важливою мовна локалізація, адаптація до локальних умов доставки й оплати, а також доступність сервісу на різних типах пристроїв.

Дослідження функціональних можливостей відомих зарубіжних онлайн-магазинів, таких як Thomann, Guitar Center і Sweetwater, дозволило виявити їх основні переваги: сучасний дизайн, інтуїтивний і гнучкий інтерфейс, наявність розгалуженої системи фільтрів, можливість порівняння характеристик, інтеграція відеооглядів і онлайн-консультацій. Однак ці платформи часто не враховують особливостей українського ринку.

Аналіз українських аналогів («Гітарний дім», «Soundmaster», «МузТорг», «JAM») показав, що іноді вони поступаються зарубіжним магазинам, зокрема, деякі сайти мають застарілий або неадаптивний інтерфейс. Проте функціонал часто виходить на один рівень зі світовими лідерами.

На основі цього аналізу було зроблено кілька практичних висновків для власного проєкту.

По-перше, необхідно впровадити сучасний, повністю адаптивний інтерфейс, що забезпечить коректну роботу як на комп'ютерах, так і на мобільних пристроях.

По-друге, доцільно реалізувати розширені можливості фільтрації та пошуку товарів, зокрема швидкий пошук за назвою, багаторівневу категоризацію та фільтри за параметрами, які є очікуваними для користувачів сучасних платформ.

Водночас для підвищення довіри користувачів і комфорту використання платформи було вирішено поставити в пріоритет забезпечення захисту персональних даних, а також надійність авторизації й збереження акаунтів.

Окремо визначено, що на першому етапі розвитку платформи можна відмовитися від надлишкового функціоналу (наприклад, порівняння товарів чи інтеграції онлайн-оплати), зосередившись на стабільній роботі основних модулів та можливості швидкого масштабування.

Таким чином, результати аналізу ринку й існуючих рішень дозволили чітко сформулювати пріоритети для розробки інтернет-магазину Chord: сучасний адаптивний дизайн, зручна навігація й гнучкі інструменти для пошуку та фільтрації товарів, а також простота взаємодії та безпека користувачів стали ключовими орієнтирами для подальших етапів роботи.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір інструментів і технологій розробки

Проектування архітектури програмного забезпечення інтернет-магазину Chord вимагало впровадження сучасних технологій, здатних забезпечити не лише високу продуктивність, а й простоту масштабування, підтримку динамічних інтерфейсів і можливість ефективної взаємодії між різними модулями системи. На першому етапі був проведений порівняльний аналіз найбільш релевантних технологічних рішень, які відповідають вимогам до створення гнучких і надійних веб-застосунків.

2.1.1 Вибір технологій для клієнтської частини

Особливості предметної області — необхідність реалізації фільтрації, пошуку, структурування каталогу, а також забезпечення зручної взаємодії користувача з інтерфейсом — зумовили вибір серед трьох найбільш поширених бібліотек: React.js, Vue.js та Angular [13] – [16] (Таблиця 2.1).

Таблиця 2.1 - Порівняльна характеристика бібліотек для клієнтської частини

	React.js	Vue.js	Angular
1	2	3	4
Архітектура	Компонентна	Компонентна	Компонентна (MVVM)
Продуктивність	Висока	Висока	Висока
Поріг входу	Середній	Низький	Високий
Документація	Вичерпна, активна спільнота	Добра, менш розгалужена	Офіційна, складна

Гнучкість	Дуже висока	Висока	Висока
-----------	-------------	--------	--------

Продовження таблиці 2.1

1	2	3	4
Широта екосистеми	Максимальна	Менша	Висока
Масштабованість	Відмінна	Добра	Відмінна
Підтримка e-commerce бібліотек	Велика	Обмежена	Помірна

За результатами аналізу було встановлено, що React.js забезпечує найкращу гнучкість архітектурних рішень для інтернет-магазину з динамічним каталогом, складною системою фільтрації та потребою у швидкому розширенні функціоналу.

2.1.2 Вибір технологій для серверної частини

Для розробки серверної частини системи порівнювалися Node.js (Express.js), Python (Django/Flask) і Java (Spring Boot) [17] – [22]. Результати аналізу наведені у Таблиці 2.2.

Таблиця 2.2 - Порівняльна характеристика бібліотек для серверної частини

	Node.js (Express.js)	Python (Django/Flask)	Java (Spring Boot)
1	2	3	4
Мова програмування	JavaScript	Python	Java

Підтримка REST API	Проста, модульна	Django – повна, Flask – базова	Вбудована
--------------------	------------------	--------------------------------	-----------

Продовження таблиці 2.2

1	2	3	4
Продуктивність	Висока (асинхронна)	Висока (синхронна)	Дуже висока
Поріг входу	Низький/середній	Django – середній, Flask – низький	Високий
Гнучкість	Дуже висока	Django – обмежена, Flask – висока	Висока
Швидкість розробки	Висока	Django – середня, Flask – висока	Середня
Єдність зі стеком	Повна (JS всюди)	Відсутня	Відсутня
Застосування у стартапах	Дуже поширене	Помірне	Рідко

Враховуючи переваги Node.js та Express.js щодо продуктивності, асинхронної обробки запитів і єдності стеку з клієнтською частиною, було обрано саме цю комбінацію як найкращу для гнучкої, масштабованої архітектури інтернет-магазину Chord [20] – [22]. Завдяки цьому вдалося досягти ефективної організації RESTful API, простоти обробки даних та легкості масштабування проєкту.

2.1.3 Вибір середовища розробки та додаткових технологій

У процесі створення вебзастосунку Chord, окрім вибору основних технологій для клієнтської й серверної частин, важливу роль відіграли засоби

організації робочого простору, тестування API й управління базою даних. Аналіз сучасних інструментів дозволив підібрати середовище розробки та допоміжні програми, що найкраще відповідають структурі й завданням проєкту. Порівняння основних характеристик середовищ розробки наведено у Таблиці 2.3 [23] – [26].

Враховуючи вимоги проєкту до гнучкої інтеграції з сучасним JavaScript-стеком, підтримки роботи з кількома модулями (frontend, backend), роботи з git, npm і MongoDB, а також потребу у численних розширеннях, як основне середовище розробки було обрано Visual Studio Code [26]. Додаткові плагіни (prettier, ES7+ React/Redux/React-Native snippets, JavaScript (ES6) code snippets, HTML CSS Support) забезпечили ефективну організацію коду.

Таблиця 2.3 – Порівняння середовищ розробки

	Visual Studio Code	WebStorm	Sublime Text	Atom
1	2	3	4	5
Підтримка JavaScript	Відмінна	Відмінна	Дуже добра	Дуже добра
Розширення/плагіни	Дуже багато (безкоштовно)	Багато (частина платна)	Обмежено	Обмежено
Інтеграція з git	Глибока, інтуїтивна	Вбудована	Через додаткові пакети	Через додаткові пакети
Вартість	Безкоштовно	Платно (30 днів пробного періоду)	Безкоштовно/платно	Безкоштовно
Продуктивність	Висока	Висока	Висока	Задовільна

Підтримка плагінів для Node.js, React	Відмінна	Відмінна	Обмежена	Обмежена
---------------------------------------	----------	----------	----------	----------

Продовження таблиці 2.3

1	2	3	4	5
Документація	Дуже розвинена	Розвинена	Помірна	Застаріла

Менеджмент залежностей здійснювався через npm [27], що дозволило стандартизувати роботу зі сторонніми бібліотеками та скриптами для всіх модулів проєкту.

Контроль версій здійснювався через git [28]. У ході розробки було створено окремі репозиторії для основних модулів (frontend, backend, admin), налагоджено .gitignore-файли для виключення службових і зайвих даних, регулярно виконувалися коміти з детальними повідомленнями, що забезпечувало можливість відстежувати зміни, повертатися до попередніх версій коду, а також працювати над кількома гілками функціоналу незалежно. Для роботи з git використовувалися як інтегровані інструменти у VS Code, так і окремий вебінтерфейс GitHub.

Для тестування API, відлагодження запитів і перевірки відповідей на стороні сервера системно використовувався Postman [29], що надало змогу ефективно контролювати взаємодію між клієнтською і серверною частинами, оперативно перевіряти авторизацію, валідацію й роботу з базою даних.

Управління базою даних здійснювалося через веб-інтерфейс MongoDB Atlas [30], що дозволяло швидко переглядати й змінювати документи, створювати колекції й наповнювати їх тестовими даними у хмарному середовищі. Додатково для аналізу можливостей розробки використовувалося розширення MongoDB for Visual Studio Code [31], яке надає змогу інтегрувати

роботу з MongoDB безпосередньо в редакторі коду, хоча основна робота велася через вебінтерфейс Atlas.

Використовувалися допоміжні бібліотеки для бекенд частини, а саме:

- cors – для організації політики міждоменного доступу й коректної взаємодії між клієнтською та серверною частинами;
- bcrypt – для безпечного хешування паролів користувачів і захисту персональних даних;
- JSON Web Token (JWT) – для реалізації сучасної схеми автентифікації, контролю доступу до захищених ресурсів;
- multer – для ефективної роботи з файлами, зокрема для завантаження зображень товарів на сервер.

Застосування цих бібліотек дозволило підвищити безпеку обробки даних, захистити доступ до облікових записів і забезпечити сучасний рівень функціональності вебзастосунку [32] – [36].

2.2 Вибір і проєктування бази даних

Під час вибору оптимальної системи керування базою даних для інтернет-магазину Chord було проаналізовано два принципово різних підходи – реляційні (SQL) та нереляційні (NoSQL) бази даних (Таблиця 2.4). Особливості предметної області – різноманітність категорій і підкатегорій товарів, наявність унікальних параметрів для різних груп продукції, а також потреба в масштабованості – стали визначальними при обґрунтуванні рішення [37].

Таблиця 2.4 – Порівняння реляційних і нереляційних БД

	Реляційні БД (SQL)	Нереляційні БД (NoSQL)
1	2	3
Тип структури	Таблична, фіксована	Гнучка, документна або

	схема	ключ-значення
Гнучкість у зміні схеми	Обмежена	Висока
Масштабованість	Вертикальна	Горизонтальна
Оптимально для	Транзакції, суворі зв'язки	Динамічні, різномірні дані, швидкий ріст

Продовження таблиці 2.4

1	2	3
Приклад	PostgreSQL, MySQL	MongoDB, CouchDB
Продуктивність при великих обсягах даних	Знижується при складних зв'язках	Зберігається завдяки денормалізації
Формат зберігання	Таблиці (рядки/стовпці)	Документи (JSON/BSON)
Актуальність для e-commerce	Традиційно (за фіксованої моделі)	Сучасний стандарт для кастомних/гнучких платформ

Враховуючи те, що для інтернет-магазину Chord необхідна підтримка великої кількості категорій музичних інструментів із різними параметрами, а також можливість динамічно розширювати або змінювати набір властивостей без складних міграцій, найкращим вибором стала документна NoSQL база даних MongoDB [38].

Документно-орієнтована модель дозволяє зберігати структури даних різної вкладеності та складності у вигляді окремих документів JSON, де кожен товар може мати індивідуальний набір характеристик у полі details.

Такий підхід не лише спрощує зберігання розширених описів і параметрів, а й забезпечує легке масштабування в разі збільшення асортименту або інтеграції нових категорій у майбутньому.

Архітектура бази даних інтернет-магазину Chord побудована на основі двох основних колекцій: Product та Users.

Сутність Product визначає певний товар. Основні поля:

- id – числовий ідентифікатор товару (унікальний, автоінкремент);
- name – назва товару (українською);
- name_eng – назва англійською мовою;
- image – шлях до зображення товару;
- category – основна категорія (наприклад, “guitars”);
- subcategory – підкатегорія (наприклад, “acoustic_guitar”);
- price – поточна ціна;
- old_price – попередня ціна (для акцій, знижок);
- brand – виробник;
- color – колір (опціонально);
- madeIn – країна виробника (опціонально);
- description – текстовий опис товару;
- details – об’єкт із параметрами, специфічними для даної підкатегорії (наприклад, кількість струн, матеріал корпусу, тип клавіш тощо);
- date – дата додавання в систему (за замовчуванням – поточна).

Завдяки зберіганню специфічних характеристик у полі details забезпечується повна підтримка динамічних і різнорідних атрибутів для різних груп інструментів. Це дозволяє оперативно додавати нові властивості чи навіть підкатегорії без втручання в загальну схему бази даних.

Сутність Users визначає зареєстрованого користувача. Основні поля:

- name – ім’я користувача;
- email – електронна пошта (унікальний логін);
- password – хешований пароль (bcrypt);
- cartData – об’єкт (map), у якому ключем є id товару, а значенням – кількість обраних одиниць;
- wishlistData – масив з id товарів у списку бажань;
- date – дата створення облікового запису.

Використання об’єкту для cartData дозволяє швидко оновлювати кількість товарів у кошику, а масиву для wishlistData – зберігати будь-які обрані

позицій. Дані користувача зберігаються захищено (пароль – обов’язково хешується).

Всі операції адміністрування (додавання, видалення, перегляд товарів) реалізуються через інтерфейс адміністратора, однак окрема сутність “адміністратор” у структурі бази даних відсутня; авторизація на адміністративні дії здійснюється за прямим доступом до відповідної панелі.

Всі товари зберігаються у колекції Product без жорстких зовнішніх ключів – це відповідає гнучкості NoSQL. Для кожного користувача зберігається окремий набір товарів у кошику і в списку бажань.

MongoDB забезпечує базовий захист від NoSQL-ін’єкцій за рахунок використання бібліотеки Mongoose для оголошення схем.

Завдяки відсутності жорстких зовнішніх зв’язків та підтримці горизонтального масштабування, БД легко адаптується під ріст кількості даних чи появу нових груп товарів.

Вибір документно-орієнтованої NoSQL бази даних MongoDB обумовлений потребою у швидкому розширенні асортименту, підтримці різноманітних і унікальних характеристик для різних груп музичних товарів, а також простотою інтеграції з сучасним стеком JavaScript. Структура БД Chord дозволяє забезпечити ефективну роботу пошуку, фільтрації, гнучке управління даними користувачів та надійний захист інформації.

2.3 Висновки до розділу 2

У цьому розділі було детально обґрунтовано вибір інструментів, технологій та архітектурних рішень для розробки інтернет-магазину музичних інструментів Chord. Проведений порівняльний аналіз бібліотек і фреймворків для клієнтської частини дозволив визначити React.js як найефективніше рішення завдяки його гнучкості, масштабованості й розвинутій екосистемі, що особливо важливо для побудови складного інтерфейсу з динамічним каталогом і багаторівневою фільтрацією. Аналіз серверних технологій виявив Node.js у

поєднанні з Express.js як оптимальну комбінацію для реалізації високопродуктивного бекенду із сучасною підтримкою асинхронної обробки запитів, ефективною інтеграцією з JavaScript-стеком та легкістю розширення функціоналу [13–22].

У процесі вибору середовища розробки було проаналізовано основні сучасні IDE та редактори. Найкращим рішенням для проєкту стала Visual Studio Code завдяки розвиненій підтримці розширень, глибокій інтеграції з git, зручності роботи з багатьма мовами програмування та наявності безкоштовних інструментів для Node.js, React і MongoDB. Додаткові інструменти — такі як Prettier, Postman та MongoDB Atlas — забезпечили ефективність розробки, тестування та управління даними [23] – [31]

Особливу увагу було приділено впровадженню сучасних бібліотек для захисту та управління користувацькими даними: cors для налаштування політик міждоменного доступу, bcrypt для хешування паролів, JWT для реалізації автентифікації, multer для роботи з файлами [32] – [36]. Для управління версіями коду застосовано git як основний інструмент відстеження змін і організації командної роботи, що сприяло впорядкованості розробки та швидкому поверненню до попередніх версій у разі потреби [28].

Обґрунтовано переваги використання документно-орієнтованої бази даних MongoDB, яка найкраще задовольняє потреби інтернет-магазину з великою кількістю різноманітних і динамічно змінних властивостей товарів. Структура БД, побудована на двох основних колекціях — Product і Users — забезпечує просте масштабування, легке додавання нових атрибутів і високу швидкість пошуку та фільтрації [37] – [38].

Завдяки цим технологічним рішенням, архітектура інтернет-магазину Chord характеризується високою гнучкістю, надійністю та готовністю до масштабування, а також відповідає сучасним вимогам безпеки й продуктивності e-commerce систем. Це створює підґрунтя для подальшого розвитку платформи та впровадження нових функцій відповідно до потреб користувачів.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Реалізація клієнтської частини

Клієнтська частина програмного забезпечення інтернет-магазину Chord розроблена з використанням бібліотеки React.js та побудована за принципами компонентної архітектури. Основна структура проєкту передбачає розподіл на окремі компоненти, сторінки та контекст для централізованого управління даними (Рисунок 3.1).

Проєкт структуровано таким чином:

- Components – окремі елементи інтерфейсу (Header, Footer, Filter, Item, Breadcrumbs, Sorter тощо);
- Pages – сторінки сайту (Home, Product, ShopCategory, Cart, Wishlist, LogInSignUp, інформаційні сторінки);
- Context – глобальний контекст ShopContext, через який здійснюється управління станом кошика, списку бажань та товарами;
- Utils – допоміжні файли для перекладу, конфігурацій;
- CSS – стилі для сторінок;
- Assets – іконки, логотипи.

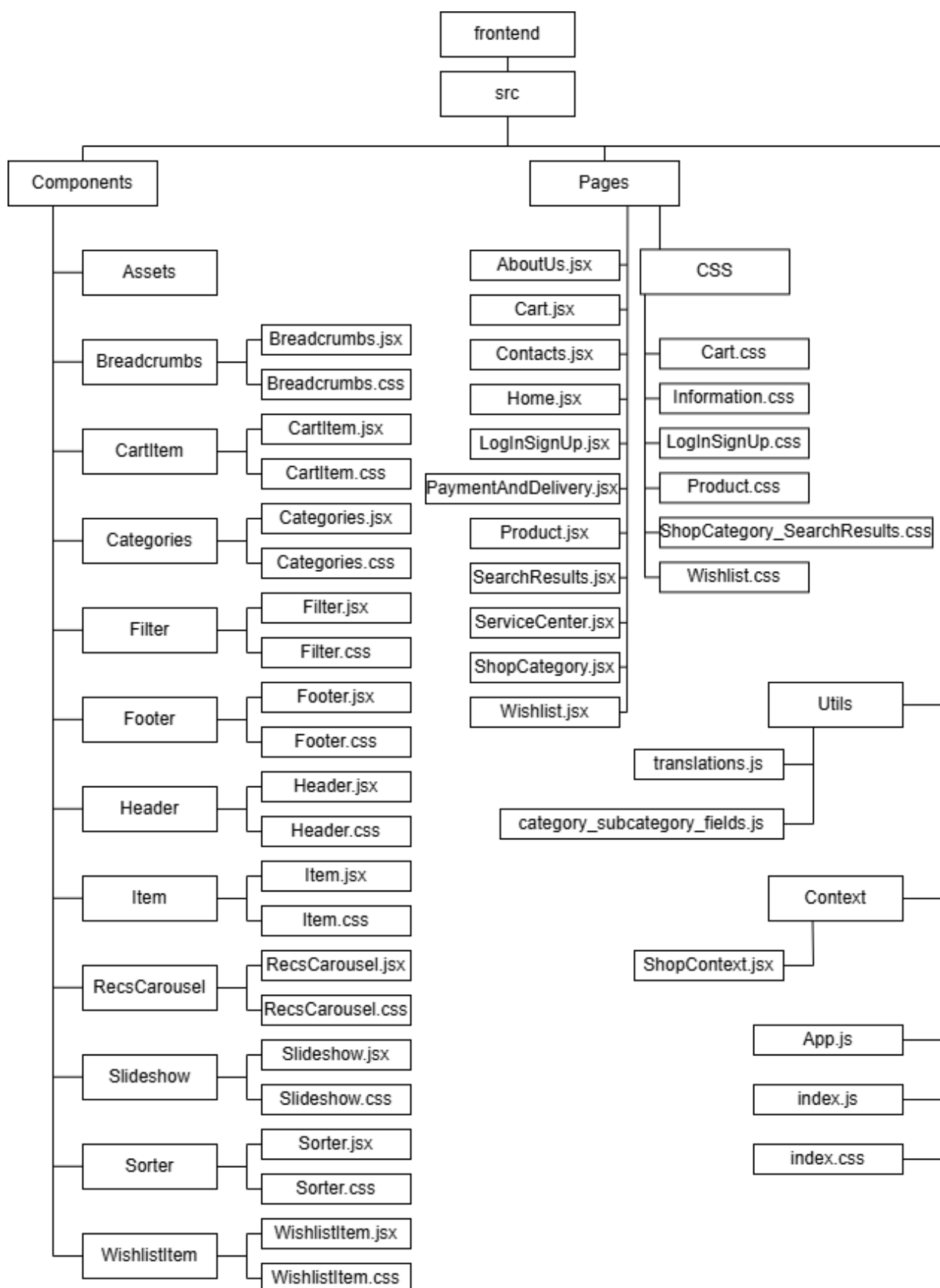


Рисунок 3.1 – Структура фронтенду

Клієнтська частина реалізована як SPA (Single Page Application). Для організації маршрутизації використовується React Router. Основний файл App.js описує всі маршрути.

Для централізованого управління даними про товари, кошик і wishlist у клієнтській частині використовується контекст ShopContext. Основна ідея — надати глобальний доступ до всіх основних операцій із товарами та їхнім станом для будь-якого компонента застосунку.

ShopContext містить такі основні дані та функції:

- масив усіх товарів, що він отримує з бази даних за допомогою бекенду (all_products);
- об'єкт поточного стану кошика map{id : numberOfItems} (cartItems);
- масив індексів товарів у списку бажань (wishlistItems);
- функції для додавання й видалення з кошика та wishlist;
- методи ініціалізації даних з сервера при старті застосунку;
- допоміжні функції для підрахунку суми/кількості товарів, очищення кошика тощо.

Ініціалізація товарів із сервера відбувається в useEffect після завантаження сторінки (Лістинг 3.1):

Лістинг 3.1 – Ініціалізація масиву товарів та стану кошика/вішлиста при завантаженні застосунку

```
useEffect(() => {
  //ініціалізація масиву товарів
  fetch("http://localhost:4000/allproducts")
    .then((response) => response.json())
    .then((data) => setAll_products(data));

  //Синхронізація стану кошика
  if (localStorage.getItem("auth-token")) {
    fetch("http://localhost:4000/getcart", {
      method: "POST",
      headers: {
        Accept: "application/form-data",
        "auth-token": `${localStorage.getItem("auth-
token")}`,
        "Content-Type": "application/json",
      },
    })
      .then((response) => response.json())
      .then((data) => setCartItems(data));
  }
}, []);
```

```

    },
    body: "",
  })
  .then((response) => response.json())
  .then((data) => setCartItems(data));

  //Синхронізація стану вішлиста
  fetch("http://localhost:4000/getwishlist", {
    method: "POST",
    headers: {
      Accept: "application/form-data",
      "auth-token": `${localStorage.getItem("auth-
token")}``,
      "Content-Type": "application/json",
    },
    body: "",
  })
  .then((response) => response.json())
  .then((data) => setWishlistItems(data));
}

```

Додавання товару до кошика — одна з базових функцій ShopContext, яка змінює локальний стан, а для авторизованих користувачів також синхронізується з сервером (Лістинг 3.2).

Лістинг 3.2 – Функція додавання товару до кошика

```

const addToCart = (itemId) => {
  setCartItems((prev) => ({ ...prev, [itemId]: prev[itemId]
+ 1 }));
  if (localStorage.getItem("auth-token")) {
    fetch("http://localhost:4000/addtocart", {
      method: "POST",
      headers: {
        Accept: "application/form-data",
        "auth-token": `${localStorage.getItem("auth-
token")}``,
        "Content-Type": "application/json",
      },
      body: JSON.stringify({ itemId: itemId }),
    })
    .then((response) => response.json())
    .then((data) => console.log(data));
  }
};

```

Інші функції, такі як `removeFromCart`, `addToWishlist`, `removeFromWishlist`, `updateCartItemCount`, реалізовані за аналогічним принципом:

- зміна стану у контексті;
- синхронізація з сервером для збереження даних для авторизованого користувача.

Весь контекст `ShopContext` “обгортає” головний компонент `App`, що дає доступ до глобальних даних для всіх сторінок і компонентів.

Головна сторінка (`Home.jsx`) є “вітриною” магазину, куди потрапляє користувач при першому вході. Вона містить кілька ключових блоків, кожен із яких реалізовано окремим компонентом:

- слайдер рекламних пропозицій (`Slideshow`) – побудований на бібліотеці `Splide` і має налаштування автопрокручування, а також переходи до сторінок відповідних товарів;
- блок «Рекомендуємо»(`RecsCarousel`) – “карусель” із рекомендованими товарами, де кожен елемент рендериться за допомогою компонента `Item`, рекомендації динамічно завантажуються з backend через `fetch`-запит, а кількість товарів на сторінку адаптується під різні розміри екрану;
- популярні категорії (`Categories`) – кожна категорія оформлена у вигляді великої плитки з фоновим зображенням товару з цієї категорії та навігаційним підписом, що з’являється під час наведення курсору на плитку, перехід до каталогу здійснюється через `React Router`.

`Product.jsx` реалізує відображення повної інформації про вибраний товар:

- фото;
- назва;
- ціна;
- бренд;
- колір;
- країна виробництва;
- параметри, специфічні до підкатегорії.

Також наявні кнопки для додавання до кошика і списку бажань інтегровані з ShopContext.jsx.

Breadcrumbs дозволяє легко повернутись на потрібну сторінку.

Header.jsx — один із найскладніших і функціонально насичених компонентів:

- містить логотип, пошук, кошик, список бажань, іконка акаунта та кнопки увійти/zareєструватись або кнопка виходу (дропдаун для маленьких екранів) для авторизованих користувачів, дропдаун-каталог;
- пошук інтегрований із навігацією (пошуковий запит передається через router);
- дропдаун-меню з категоріями підтримує hover/клік, перекривається “overlay” (напівпрозорий фон) для фокусування уваги на меню;
- всі кнопки і посилання інтегровані з React Router;
- реалізовано адаптивність: змінюється структура header і порядок елементів на мобільних пристроях.

Адаптивність і перенесення елементів забезпечується через useEffect з додатковою логікою вставки DOM-елементів, що не типово для React (але ефективно для складного UI). Результат переміщення елементів та різниця Header на великих і малих екранах представлені на рисунках 3.2 та 3.3.

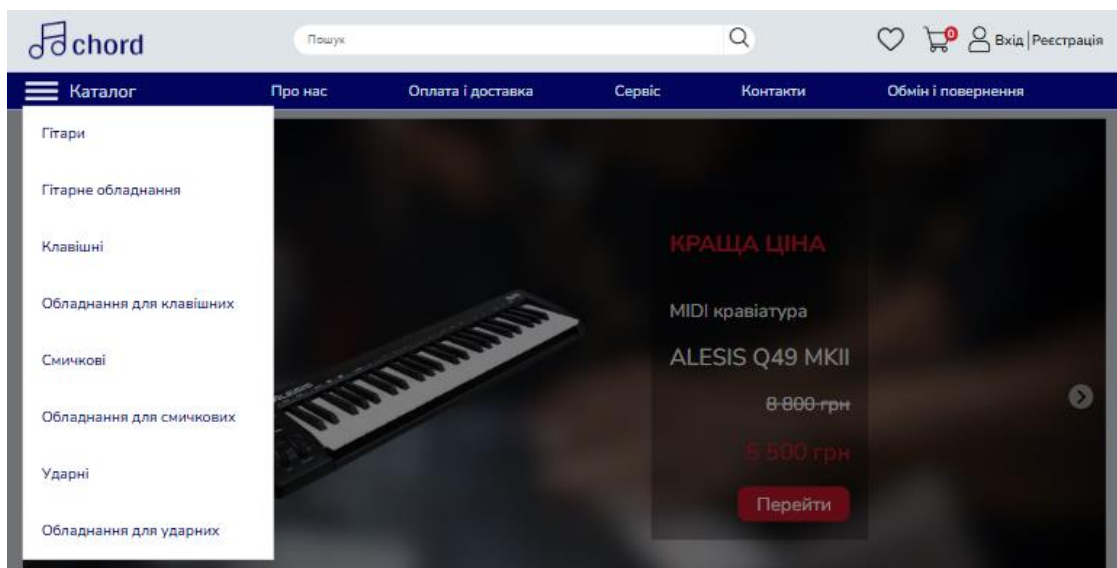


Рисунок 3.2 – Хедер при ширині екрану більше 800px

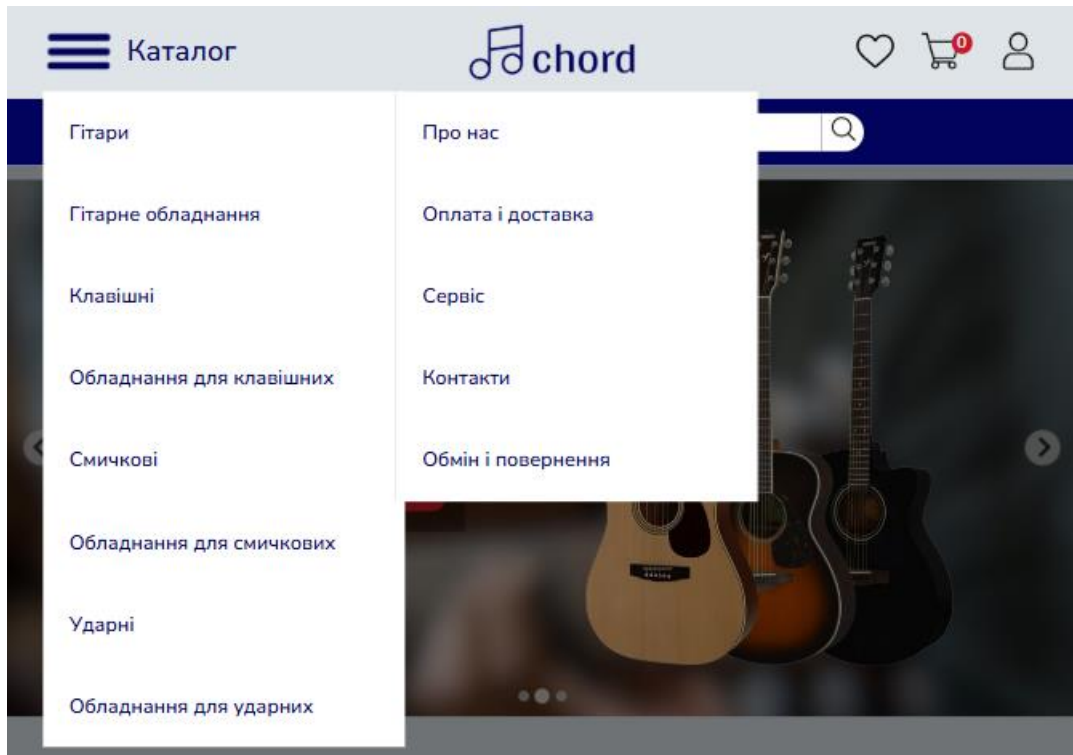


Рисунок 3.3 – Хедер при ширині екрану менше 800px

Фільтри на сторінці каталогу формуються автоматично залежно від отриманих товарів і їхніх властивостей.

Filter.jsx буде доступні фільтри та price range, з урахуванням специфічних параметрів для кожної підкатегорії (конфігурація — у category_subcategory_fields.js) (Лістинг 3.3).

Лістинг 3.3 – Динамічне формування фільтрів залежно від полів товарів

```
useEffect(() => {
  const dynamicDetailsFilters = {};
  const dynamicBasicFilters = {};
  let min = Infinity;
  let max = -Infinity;
  products.forEach((product) => {
    Object.entries(product).forEach(([key, value]) => {
      if (key === "price") {
        min = Math.min(min, value);
        max = Math.max(max, value);
      } else if (key === "details" && typeof value ===
"object") {
```

```

        Object.entries(value).forEach(([detailsKey,
detailsValue]) => {
            if (!dynamicDetailsFilters[detailsKey])
                dynamicDetailsFilters[detailsKey] = new Set();

dynamicDetailsFilters[detailsKey].add(detailsValue);
        });
    } else if (!excludedKeys.includes(key)) {
        if (!dynamicBasicFilters[key])
dynamicBasicFilters[key] = new Set();
        dynamicBasicFilters[key].add(value);
    }
    });
    // ...
}, [products]);

```

Форма входу/реєстрації реалізована як компонент `LogInSignUp.jsx`. Валідація та обробка помилок виконуються як на клієнті, так і на сервері. Повідомлення про помилки валідації відображаються під відповідними полями (Лістинг 3.4).

Лістинг 3.4 – Валідація форми та логіка логінізації в `LogInSignUp.jsx`

```

const validateForm = () => {
    const newErrors = {};
    if (!formData.email) newErrors.email = "Це поле
обов'язкове";
    else if (!/^[^@]+@^[^@]+\.[^@]+$/ .test(formData.email))
        newErrors.email = "Невірний формат електронної адреси";
    if (!formData.password) newErrors.password = "Це поле
обов'язкове";
    if (!isLogin && !formData.username) newErrors.username =
"Це поле обов'язкове";
    setErrors(newErrors);
    return Object.keys(newErrors).length === 0;
};

const login = async () => {
    if (!validateForm()) return;
    let responseData;
    await fetch("http://localhost:4000/login", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(formData),
    })
        .then((response) => response.json())
        .then((data) => (responseData = data));

```

```

    if (responseData.success) {
      localStorage.setItem("auth-token", responseData.token);
      window.location.replace("/home");
    } else {
      setServerError(responseData.errors);
    }
  };

```

Усі елементи вебзастосунку виконані в єдиному візуальному стилі, з акцентом на простоту навігації й привабливість для користувача. Компоненти повністю адаптивні, що забезпечує коректне відображення на різних пристроях та екранах.

3.2 Реалізація серверної частини

Серверна частина інтернет-магазину Chord створена як окремий програмний модуль, розміщений у директорії backend. Вона реалізована на базі Node.js із застосуванням Express.js, MongoDB, Mongoose, JWT, bcrypt, multer і cors. У файлі index.js концентровано всі налаштування серверної частини, зокрема підключення бібліотек, запуск сервера, підключення до MongoDB, опис основних схем Mongoose, реалізацію API, а також логіку обробки файлів (зображень). Сервер стартує на порті 4000.

Для коректної роботи із сучасними браузерами та клієнтською частиною на React у проєкті використовується механізм cors, а для обробки JSON-запитів — стандартну бібліотеку Express.

Підключення бібліотек, старт сервера та з'єднання з базою представляється у Лістингу 3.5.

Лістинг 3.5 – ініціалізація сервера і підключення до MongoDB

```

const port = 4000;
const express = require("express");
const app = express();
const mongoose = require("mongoose");
const jwt = require("jsonwebtoken");
const multer = require("multer");
const path = require("path");

```

```

const cors = require("cors");
const bcrypt = require("bcrypt");

app.use(express.json());
app.use(cors());

//Database connection with mongoDB
mongoose.connect(
  "mongodb+srv://Elizabeth:IOrTDz3HlLauPMm3@cluster0.8rbqyyb.mongodb
  .net/chord_musicshop"
);

```

Для завантаження зображень реалізовано сховище через бібліотеку `multer`, яка розміщує всі файли у піддиректорії `upload/images`. Ім'я файлу формується автоматично із зазначенням типу, мітки часу та розширення. Всі завантажені файли доступні для клієнта за шляхом `/images/<ім'я_файлу>`. (Лістинг 3.6).

Лістинг 3.6 – Налаштування `multer` для зберігання зображень

```

//image storage engine
const storage = multer.diskStorage({
  destination: "./upload/images",
  filename: (req, file, cb) => {
    return cb(
      null,
      `${file.fieldname}_${Date.now()}${path.extname(file.originalname)}
    `);
  },
});

const upload = multer({ storage: storage });

// creating upload endpoint for images
app.use("/images", express.static("upload/images"));

app.post("/upload", upload.single("product"), (req, res) => {
  res.json({
    success: 1,
    image_url:
      `http://localhost:${port}/images/${req.file.filename}`,
  });
});

```

У файлі `index.js` визначено дві ключові схеми: `Product` і `Users`.

Схема `Product` використовується для зберігання інформації про товари. Особливість реалізації — наявність поля `details`, де можуть зберігатися унікальні характеристики залежно від підкатегорії інструмента (Таблиця 3.1).

Таблиця 3.1 – Поля сутності `Product`

Поле	Тип	Призначення
1	2	3
<code>Id</code>	<code>Number</code>	Унікальний ідентифікатор
<code>name</code>	<code>String</code>	Назва товару українською
<code>name_eng</code>	<code>String</code>	Назва англійською
<code>image</code>	<code>String</code>	Шлях до зображення
<code>category</code>	<code>String</code>	Основна категорія
<code>subcategory</code>	<code>String</code>	Підкатегорія
<code>price</code>	<code>Number</code>	Ціна
<code>old_price</code>	<code>Number</code>	Стара ціна (якщо наявна знижка)
<code>brand</code>	<code>String</code>	Бренд
<code>color</code>	<code>String</code>	Колір
<code>madeIn</code>	<code>String</code>	Країна виробника
<code>description</code>	<code>String</code>	Опис
<code>details</code>	<code>Object</code>	Об'єкт із параметрами, специфічними для підкатегорії
<code>date</code>	<code>Date</code>	Дата додавання

Додавання товару реалізовано POST-запитом `/addproduct`. Новий `id` формується на основі останнього доданого товару в базі (Лістинг 3.7).

Лістинг 3.7 – Додавання нового товару

```
//creating API for adding products
```

```

app.post("/addproduct", async (req, res) => {
  let products = await Product.find({});
  let id;
  if (products.length > 0) {
    let last_product_array = products.slice(-1);
    let last_product = last_product_array[0];
    id = last_product.id + 1;
  } else {
    id = 1;
  }
  console.log("Received body:", req.body);
  const product = new Product({
    id: Number(id),
    name: req.body.name,
    name_eng: req.body.name_eng,
    image: req.body.image,
    category: req.body.category,
    subcategory: req.body.subcategory,
    price: Number(req.body.price),
    old_price: Number(req.body.old_price),
    brand: req.body.brand,
    color: req.body.color,
    madeIn: req.body.madeIn,
    description: req.body.description,
    details: req.body.details || {}, // Save the details
object
  });
  console.log("Product to save:", product);
  await product.save();
  console.log("Saved");
  res.json({
    success: true,
    name: req.body.name,
  });
});

```

Видалення товару відбувається через POST-запит `/removeproduct` за переданим `id`.

Отримання всіх товарів — GET-запит `/allproducts` повертає весь асортимент.

Усі користувачі зберігаються у колекції `Users`. Ключові поля представлені у Таблиці 3.2.

Таблиця 3.2 – Поля сутності Users

Поле	Тип	Призначення
name	String	Ім'я користувача
email	String	Е-mail, використовується як логін
password	String	Хешований пароль
cartData	Object	Дані кошика, map {id: кількість}
wishlistData	Array	Массив id товарів у списку бажань
date	Date	Дата створення акаунта

При реєстрації (POST /signup) перевіряється унікальність логіну, пароль хешується за допомогою bcrypt, для нового користувача створюється "порожній" кошик і список бажань (Лістинг 3.8).

Лістинг 3.8 – Реєстрація користувача з хешуванням паролю

```
//Creating endpoint for user registration
app.post("/signup", async (req, res) => {
  let check = await Users.findOne({ email: req.body.email });
  if (check) {
    return res.status(400).json({
      success: false,
      errors: "Обліковий запис з таким логіном вже існує",
    });
  }
  let cart = {};
  for (let i = 0; i < 300; i++) {
    cart[i] = 0;
  }
  let wishlist = [];
  // Хешування паролю перед збереженням
  const saltRounds = 10;
  const hashedPassword = await bcrypt.hash(req.body.password,
saltRounds);

  const user = new Users({
    name: req.body.username,
    email: req.body.email,
    password: hashedPassword,
    cartData: cart,
    wishlistData: wishlist,
  });
  await user.save();
```

```

const data = {
  user: {
    id: user.id,
  },
};

const token = jwt.sign(data, "secret_mshop");
res.json({ success: true, token });
});

```

При вході в акаунт на основі email і паролю генерується JWT-токен для авторизації клієнта (Лістинг 3.9).

Лістинг 3.9 – Логінізація користувача

```

//creating endpoint for user login
app.post("/login", async (req, res) => {
  let user = await Users.findOne({ email: req.body.email });
  if (user) {
    /** @type {boolean} */
    const passCompare = await
bcrypt.compare(req.body.password, user.password);
    if (passCompare) {
      const data = {
        user: {
          id: user.id,
        },
      };
      const token = jwt.sign(data, "secret_mshop");
      res.json({ success: true, token });
    } else {
      res.json({ success: false, errors: "Невірний пароль"
});
    }
  } else {
    res.json({ success: false, errors: "Невірний логін" });
  }
});

```

Для доступу до персональних даних (кошик, wishlist) реалізований middleware fetchUser, який перевіряє валідність токена у заголовку auth-token. У разі невірного або відсутнього токена запит відхиляється з відповідним повідомленням(Лістинг 3.10).

Лістинг 3.10 – Middleware для авторизації

```
// creating middleware to fetch uder
const fetchUser = async (req, res, next) => {
  const token = req.header("auth-token");
  if (!token) {
    res.status(401).send({ errors: "Please authenticate using
valid token" });
  } else {
    try {
      const data = jwt.verify(token, "secret_mshop");
      req.user = data.user;
      next();
    } catch (error) {
      res.status(401).send({ errors: "Please authenticate
using valid token" });
    }
  }
};
```

Додавання/видалення товарів у кошик здійснюється через POST-запити /addtocart, /removefromcart, для wishlist — /addtowishlist, /removefromwishlist. Оновлення даних відбувається шляхом пошуку користувача по id і зміні відповідного поля.

Отримання стану кошика/списку бажань — POST-запити /getcart, /getwishlist.

Також як додаткова функція реалізований endpoint /recommended, що повертає останні додані товари для блоку рекомендацій на головній сторінці.

Весь бекенд Chord написаний у вигляді єдиного серверного файлу без розділення на окремі папки routes/controllers. Всі моделі й бізнес-логіка знаходяться у index.js. Реалізація дозволяє:

- додавати, видаляти й переглядати товари, включаючи їх унікальні характеристики;
- реєструвати й авторизувати користувачів з безпечним зберіганням паролів;
- керувати персональним кошиком і списком бажань;
- додавати зображення товарів через HTTP;
- захищати особисті дані через JWT middleware.

Завдяки такій структурі backend повністю забезпечує вимоги функціоналу онлайн-магазину та залишається гнучким для подальшого розширення.

3.3 Панель адміністратора

Панель адміністратора інтернет-магазину Chord створена як окремий фронтенд-застосунок на базі React.js та інтегрується із серверною частиною через REST API. Головна мета адмін-панелі — забезпечити простий та інтуїтивний інтерфейс для оперативного управління асортиментом товарів магазину.

Структура адмінки побудована модульно, кожен функціонал винесено в окремі компоненти, що розташовані у відповідних папках:

- AddProduct — додавання нових товарів;
- ListProduct — перегляд і видалення товарів;
- Sidebar — навігаційний елемент, що забезпечує швидкий перехід між розділами;
- Navbar — на даному етапі декоративний елемент.

Вся навігація між компонентами організована через Sidebar, що дозволяє перемикатися між режимами роботи.

Головна функція — створення нової позиції у каталозі. Форма додавання товару побудована так, щоб динамічно підлаштовуватися під обрану категорію й підкатегорію. Для цього використовується файл `category_subcategory_fields.js`, у якому визначені зв'язки категорія – підкатегорії та підкатегорія – параметри. Всі характеристики, окрім категорії і підкатегорії, мають свої input-поля. Категорії і підкатегорії обираються з декількох варіантів через тег `select`. Після вибору категорії поле підкатегорії оновлюється, щоб вибір був саме між підкатегоріями потрібної категорії. Усі поля з характеристиками, що залежать від підкатегорії, оновлюються подібним чином – після вибору підкатегорії або нової категорії.

Після вибору зображення товару картинка завантажується на сервер через endpoint `/upload` із застосуванням `fetch` (Лістинг 3.11).

Лістинг 3.11 – завантаження зображення з форми додавання

```
// Handle image upload
let formData = new FormData();
formData.append("product", image);
await fetch("http://localhost:4000/upload", {
  method: "POST",
  headers: {
    Accept: "application/json",
  },
  body: formData,
})
  .then((resp) => resp.json())
  .then((data) => {
    responseData = data;
  });
```

Після заповнення усіх полів і натискання кнопки “Додати” формується запит на backend із усіма даними нового товару (Лістинг 3.12).

Лістинг 3.12 – Відправлення нового товару на сервер

```
fetch("http://localhost:4000/addproduct", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify(productData),
});
```

Після успішного додавання дані у формі обнуляються, з’являється повідомлення про успіх. Під час валідації перевіряється чи не пусті обов’язкові поля.

У компоненті `ListProduct.jsx` реалізовано:

- Запит на бекенд для отримання всіх товарів (`/allproducts`);
- Відображення отриманого списку у вигляді таблиці або списку карток з короткою інформацією;
- Кожен товар має кнопку видалення — натискання відправляє POST-запит `/removeproduct` з `id` вибраного товару (Лістинг 3.13).

Лістинг 3.13 – Видалення товару з каталогу

```
const removeProduct = async (id) => {
  await fetch("http://localhost:4000/removeproduct", {
    method: "POST",
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ id: id }),
  });
  await fetchInfo();
};
```

Після видалення товар одразу зникає зі списку на сторінці, не потребуючи оновлення всієї сторінки.

Sidebar.jsx відповідає за зручність переходу між сторінками адмінки. Він містить пункти меню, які ведуть до форм додавання товару або перегляду всіх товарів, Navbar виконує допоміжну роль.

Інтерфейс виконаний у єдиному стилі, з використанням іконок із папки assets. Дизайн орієнтований на мінімалізм і швидкий доступ до основних функцій без зайвих етапів або переходів.

Панель адміністратора Chord дозволяє додавати нові товари із гнучким налаштуванням параметрів, переглядати повний каталог та оперативно видаляти позиції. Архітектура реалізована максимально простою для подальшого розширення функцій (наприклад, редагування або пошук за фільтрами), а всі ключові процеси автоматизовано для економії часу адміністратора. Такий підхід забезпечує швидке управління асортиментом магазину і високу якість обслуговування.

3.4 Висновки до розділу 3

У цьому розділі було безпосередньо реалізовано функціонал інтернет-магазину музичних інструментів Chord, що охоплює клієнтську частину, серверну логіку та адміністративну панель.

В процесі розробки клієнтської частини вдалося впровадити інтуїтивний, багатокомпонентний інтерфейс із сучасною структурою та гнучкою навігацією. Значну увагу приділено реалізації динамічних фільтрів, які автоматично підлаштовуються під категорію й підкатегорію товарів, а також адаптивному дизайну, що забезпечує коректну роботу на будь-яких пристроях. Також була розроблена синхронізація кошика й списку бажань між різними пристроями для зареєстрованих користувачів, що підвищує зручність та персоналізацію користування платформою.

Розробка серверної частини забезпечила повноцінну підтримку REST API для всіх ключових операцій із товарами та користувачами. Реалізовано надійне зберігання даних у MongoDB із чітким розділенням сутностей Product і Users. Для безпеки користувацьких даних використовується хешування паролів через bcrypt і авторизація через JWT, що гарантує захист облікових записів. Сервер обробляє завантаження зображень, що дозволяє швидко оновлювати асортимент магазину з боку адміністратора.

Панель адміністратора стала окремим додатком, який суттєво спрощує управління каталогом товарів. Адміністратор має можливість додавати товари з динамічно сформованими параметрами, переглядати повний перелік продукції, а також оперативно видаляти непотрібні позиції. Інтерфейс побудований таким чином, щоб навіть без спеціальних навичок можна було швидко редагувати наповнення магазину, а всі зміни моментально відображаються у клієнтській частині завдяки асинхронній роботі з сервером.

Окремо варто відзначити напрями, у яких система інтернет-магазину Chord може бути розширена у майбутньому. По-перше, доцільним буде впровадження ролей користувачів та багаторівневої системи доступу, що дозволить відокремити права адміністратора, менеджерів і звичайних користувачів. Це підвищить безпеку й керованість у випадку масштабування команди. Додатковою перевагою стане реалізація функціоналу редагування вже існуючих товарів у адміністративній панелі, а також можливість масового імпорту та експорту даних для зручнішого оновлення великого каталогу. Ще

одним перспективним напрямом розвитку є інтеграція платіжних систем безпосередньо у клієнтську частину, що дозволить автоматизувати обробку замовлень. А також реалізувати окремий особистий кабінет для користувача з можливістю перегляду історії покупок та змін персональних даних.

У результаті виконання всіх підпунктів розділу 3 було створено цілісну систему, де всі компоненти ефективно взаємодіють між собою. Реалізований функціонал дозволяє швидко масштабувати платформу, розширювати каталог і додавати нові властивості до товарів без складних змін у структурі БД чи логіці взаємодії. Отриманий продукт повністю відповідає поставленим вимогам і дає можливість подальшого розвитку проекту у напрямку розширення функціоналу та автоматизації онлайн-торгівлі музичними інструментами.

4 Експлуатація та тестування програми

4.1 Призначення програми

Програмне забезпечення Chord розроблене для автоматизації процесу онлайн-продажу музичних інструментів та аксесуарів. Платформа орієнтована на широкий спектр користувачів: як музикантів-початківців і професійних виконавців, так і викладачів музичних шкіл, творчих колективів і всіх, хто цікавиться якісними музичними товарами.

Система забезпечує зручний доступ до каталогу продукції, дозволяє легко знаходити потрібні товари завдяки гнучким фільтрам, зберігати вибране у кошику або списку бажань. Передбачена можливість створення персонального кабінету для збереження даних кошика та списку бажань.

Для співробітників магазину реалізована адміністративна панель, що дає змогу оперативно переглядати, додавати, й видаляти товари. Адмін-панель побудована таким чином, щоб процес керування асортиментом був простим і не вимагав спеціальних технічних навичок.

4.2 Умови виконання програми

Програмний комплекс Chord складається з трьох окремих модулів: клієнтської частини, серверної частини та адміністративної панелі, кожен із яких має власні залежності та сценарії запуску.

Для коректної роботи всіх модулів необхідно встановити Node.js версії не нижче 16.0, а також менеджер пакетів npm.

Frontend (основний сайт для користувачів):

- всі залежності встановлюються командою `npm install` у директорії frontend;
- для запуску застосунку використовується команда `npm start`;

- підтримується робота у сучасних версіях браузерів Google Chrome, Mozilla Firefox, Microsoft Edge, Opera.

Backend (серверна частина):

- залежності встановлюються через `npm install` у директорії `backend`;
- для запуску використовується команда `node ./index.js`;
- сервер “слухає” порт 4000 та забезпечує обробку всіх запитів від клієнта й адмін-панелі;
- база даних розміщена у хмарному середовищі MongoDB Atlas, додаткового налаштування локального СУБД не потребується.

Admin panel (адміністративна панель):

- встановлення залежностей — командою `npm install` у папці `admin`;
- для розробки й тестування використовується команда `npm run dev` (платформа Vite).

Мінімальні системні вимоги:

ПК/ноутбук із процесором Intel i3, AMD Ryzen 3 або вище, не менше 4 ГБ оперативної пам’яті, 1 ГБ вільного місця на диску, стабільний доступ до Інтернету.

Операційна система — Windows, Linux або MacOS (не критично завдяки кросплатформенності Node.js і React).

Система не вимагає підключення сторонніх сервісів, окрім доступу до Інтернету для роботи з хмарною базою даних. Для забезпечення стабільної роботи і повної сумісності рекомендується використовувати останні LTS-версії Node.js та сучасні браузери.

4.3 Робота з програмою

4.3.1 Користування клієнтською частиною

Інтерфейс інтернет-магазину Chord створено для максимальної зручності та швидкого доступу до основних функцій як для звичайного користувача, так і для адміністратора. У цьому пункті наведено типові сценарії

роботи з клієнтською частиною і адмін-панеллю, а також приклади валідації, повідомлень про помилки й відображення результатів дій користувача.

Головна сторінка (Рисунок 4.1) містить слайдер із рекламними пропозиціями, блок рекомендованих товарів та плитки популярних категорій. Панель навігації у верхній частині забезпечує швидкий доступ до основних розділів магазину, а пошуковий рядок дозволяє швидко знайти потрібний товар за назвою. При додаванні товарів у кошик інкрементується лічильник біля іконки кошику .

При наведенні або натисканні на пункт “Каталог” відкривається дропдаун-меню з переліком основних категорій, що дозволяє легко орієнтуватися у великому асортименті.

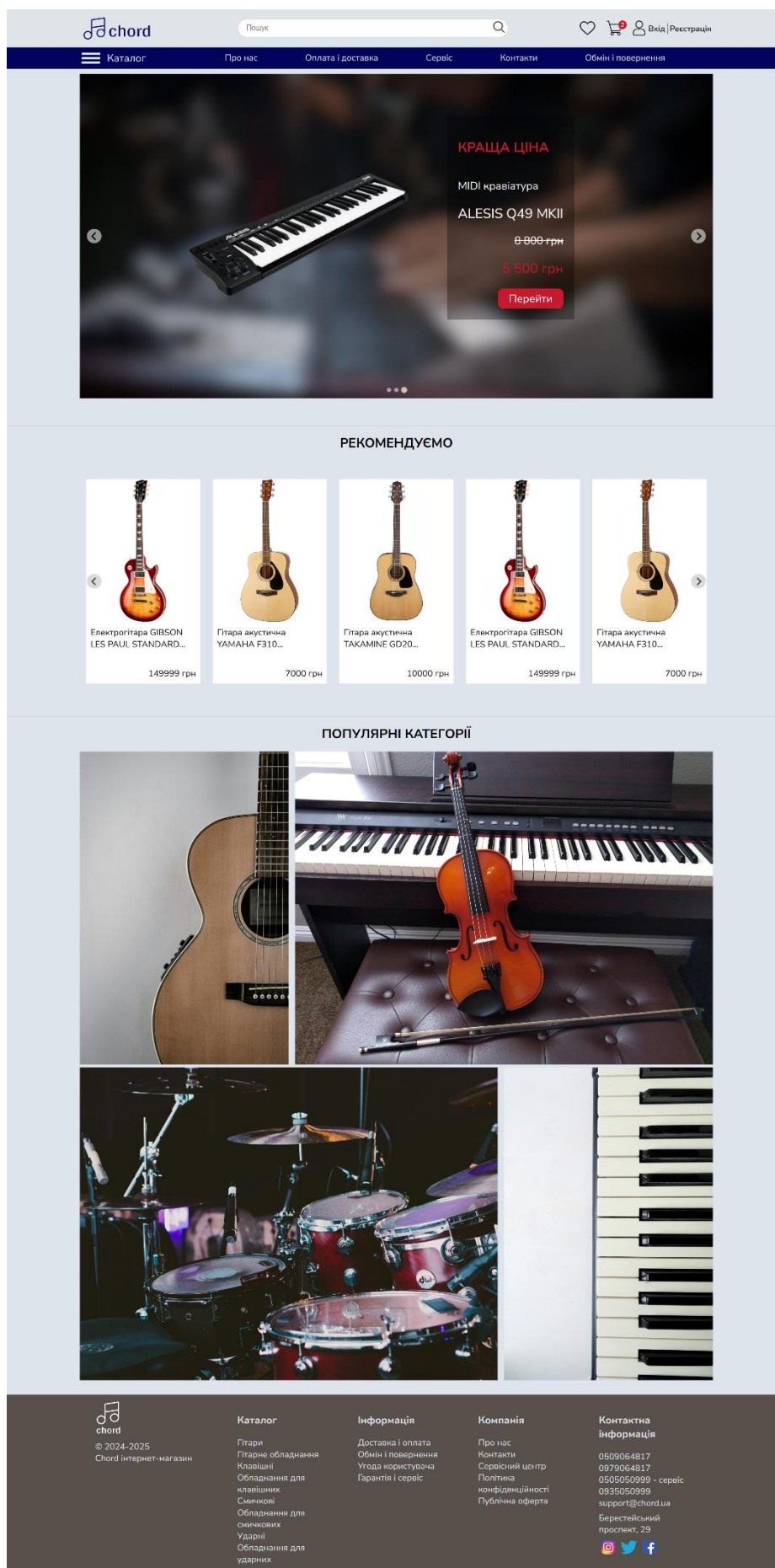


Рисунок 4.1 – Головна сторінка

Каталог (Рисунок 4.2) дає змогу відфільтрувати товари за параметрами: ціна, бренд, колір специфічні характеристики (наприклад, матеріал корпусу чи кількість струн для гітар), а також можна перейти до каталогу певної підкатегорії, натиснувши на її картку. Всі фільтри формуються динамічно залежно від вибраної підкатегорії. На картці товару при наведенні з'являють кнопки, щоб додати товар у вішлист/кошик.

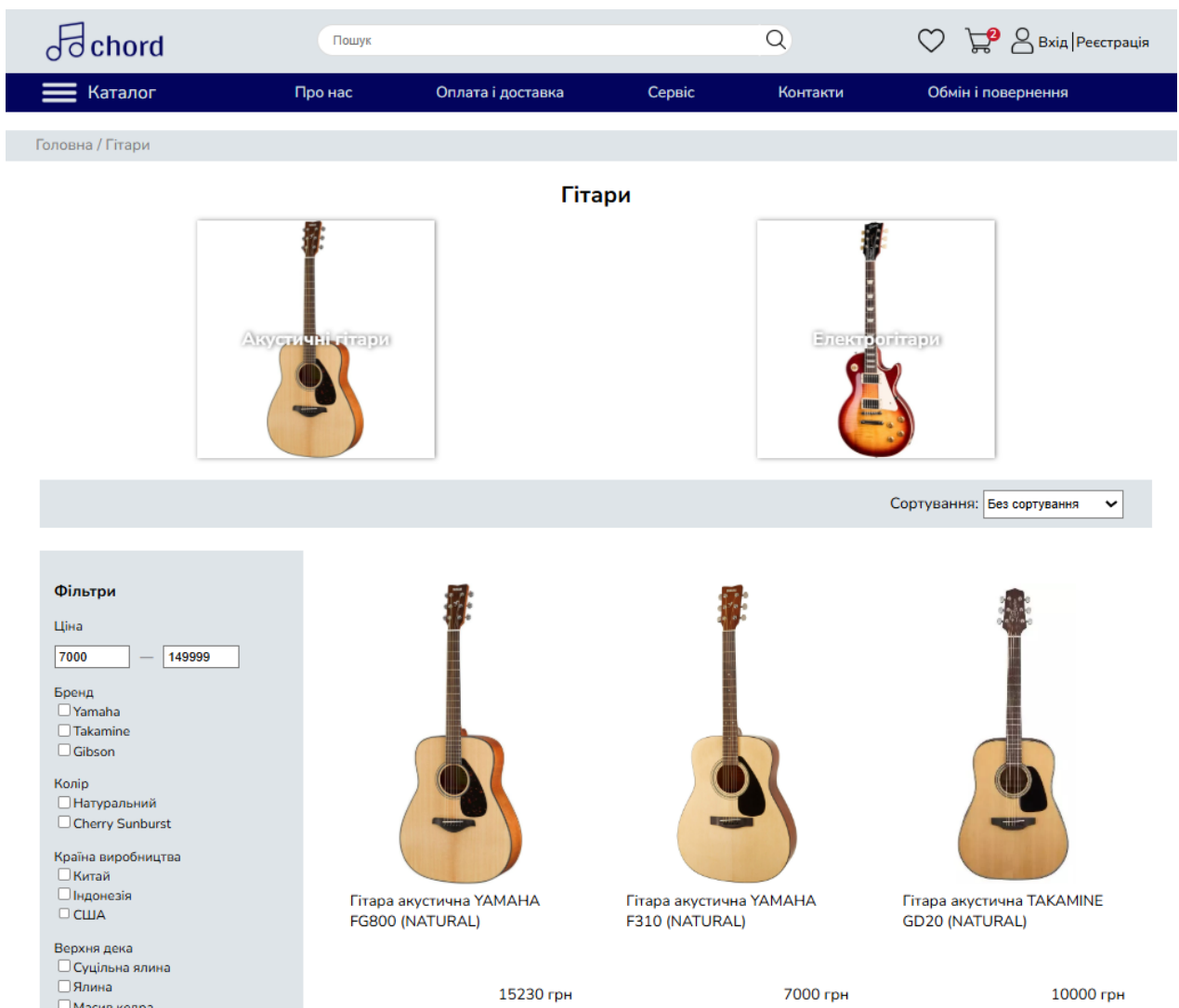


Рисунок 4.2 – Сторінка каталогу за категорією «Гітари»

Сторінка окремого товару (Рисунок 4.3) містить всю інформацію про товар:

- фотографію;
- розгорнутий опис;

– список усіх характеристик, включаючи ті, що специфічні до підкатегорії.

Кнопки “Додати до кошика” і “Додати до списку бажань” дають змогу зберегти вибраний товар для подальшого придбання.



The screenshot shows the product page for a Gibson Les Paul guitar on the 'chord' website. The header includes the 'chord' logo, a search bar, and links for 'Вхід' and 'Реєстрація'. The navigation bar lists 'Каталог', 'Про нас', 'Оплата і доставка', 'Сервіс', 'Контакти', and 'Обмін і повернення'. The breadcrumb trail reads: 'Головна / Гітари / Електрогітари / Електрогітара GIBSON LES PAUL STANDARD '50S (CHERRY SUNBURST)'. The product image is a red Gibson Les Paul guitar. The title is 'Електрогітара GIBSON LES PAUL STANDARD '50S (CHERRY SUNBURST)'. The price is '149999 грн'. The description states: 'Gibson Les Paul Standard '50s — це втілення класичного рок-звучання. Махагонова дека й гриф, паліандрова накладка та хамбакери забезпечують потужний, теплий тон з глибоким сустейном. Ідеальна гітара для блюзу, року та хард-року.' There is a 'Додати в кошик' button and a heart icon for wishlists.

Характеристики

Бренд	Gibson
Країна виготовлення	США
Колір	Cherry Sunburst
Матеріал корпусу	Червоне дерево
Матеріал грифа	Червоне дерево
Матеріал накладки	Паліандр
Звукознімачі	Humbucker

Рисунок 4.3 – Сторінка товару

У кошику є можливість змінювати кількість доданого товару, видаляти позиції, побачити суму усіх обраних товарів, а також присутні кнопки оформлення замовлення (на даному етапі замовлення не реалізовано) та переходу на головну сторінку для продовження покупок (Рисунок 4.4).

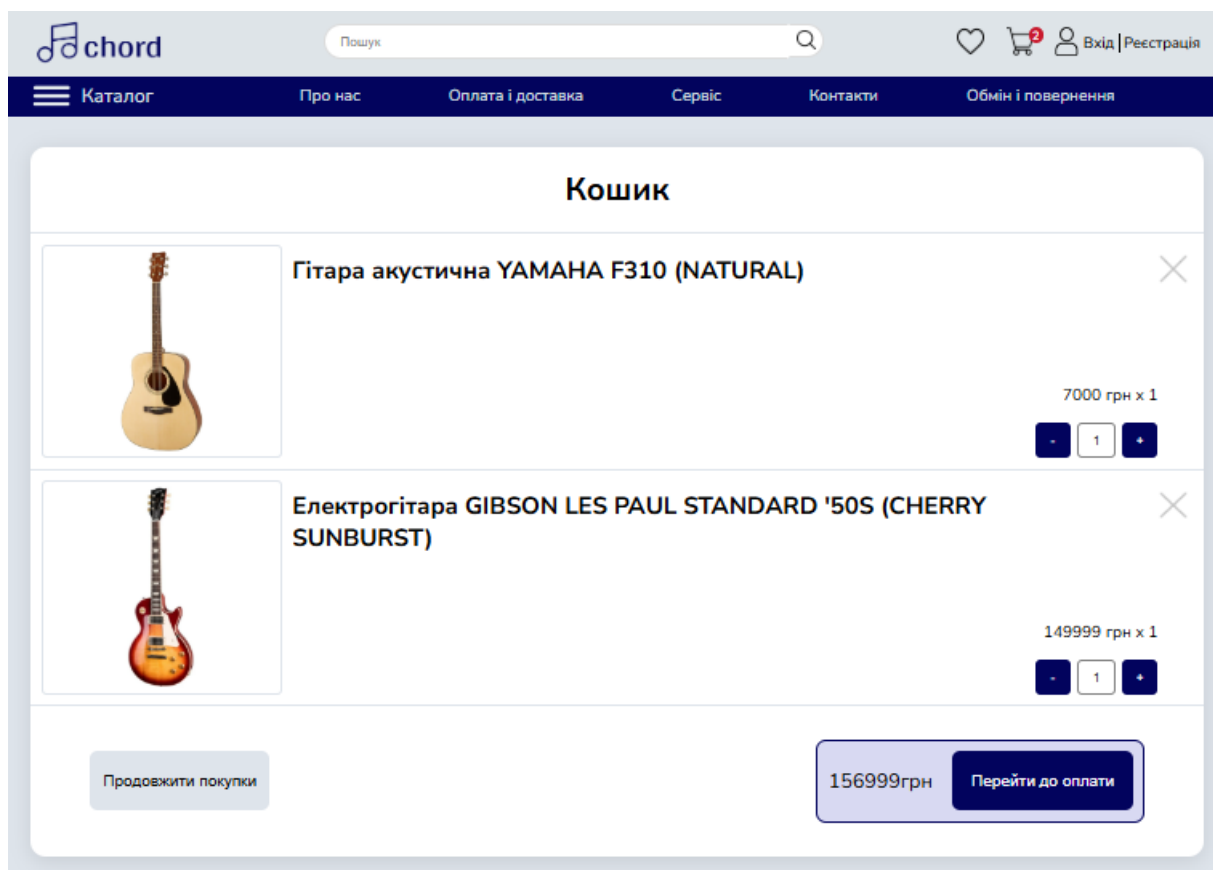


Рисунок 4.4 – Сторінка кошику

Список бажань (wishlist) (Рисунок 4.5) дозволяє зберігати товари для майбутніх покупок або просто відслідковувати вподобане. Є можливість додати один або всі товари до кошика.

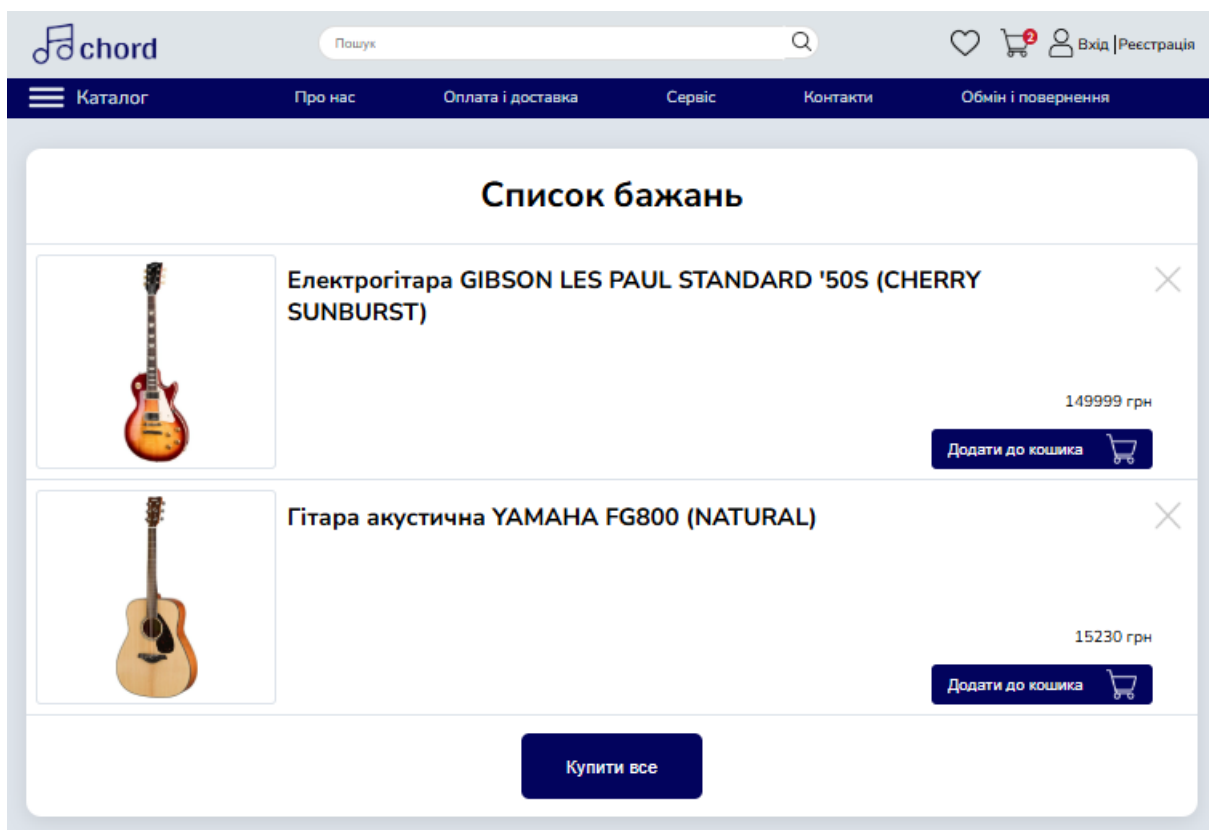


Рисунок 4.5 – Сторінка списку бажань користувача

Щоб зберегти дані кошика та списку бажань необхідно зареєструватись або, за наявності акаунту, увійти у систему. Для реєстрації необхідно ввести ім'я користувача, логін(email) та пароль, для входу – тільки логін і пароль. Всі поля перевіряються на коректність – при помилках відображаються підказки (Рисунок 4.6).

Всі інформаційні сторінки доступні через навігаційне меню. Сторінка результатів пошуку має таку ж структуру, як і каталог, з відповідним заголовком.

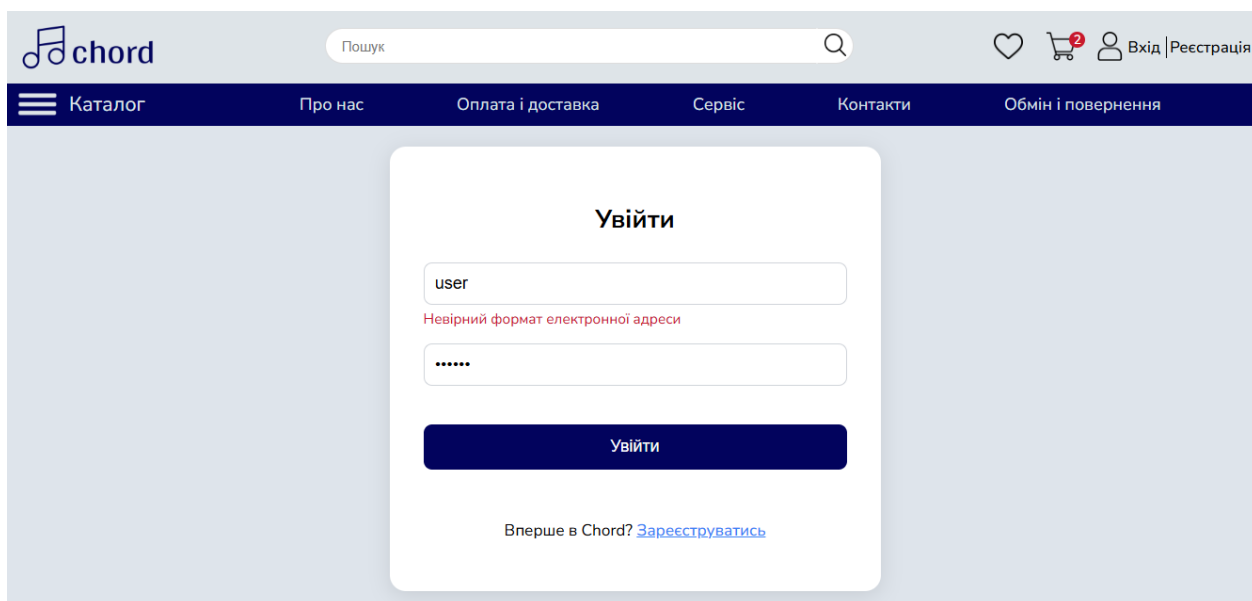


Рисунок 4.6 – Сторінка входу з помилкою «Невірний формат електронної адреси»

4.3.2 Робота з панеллю адміністратора

Адміністративна панель системи Chord дає змогу адміністратору додавати, переглядати та видаляти товари у магазині через простий інтуїтивний інтерфейс.

Після входу адміністратор бачить бічну панель з двома основними пунктами: «Додати товар» та «Список товарів». На менших екранах бічна панель переходить у верхню частину екрану.

Весь актуальний перелік товарів відображається у панелі «Список товарів» у вигляді таблиці. Адміністратор може переглянути основну інформацію, а також одним кліком видалити товар. Після видалення сторінка оновлюється автоматично (Рисунок 4.7).

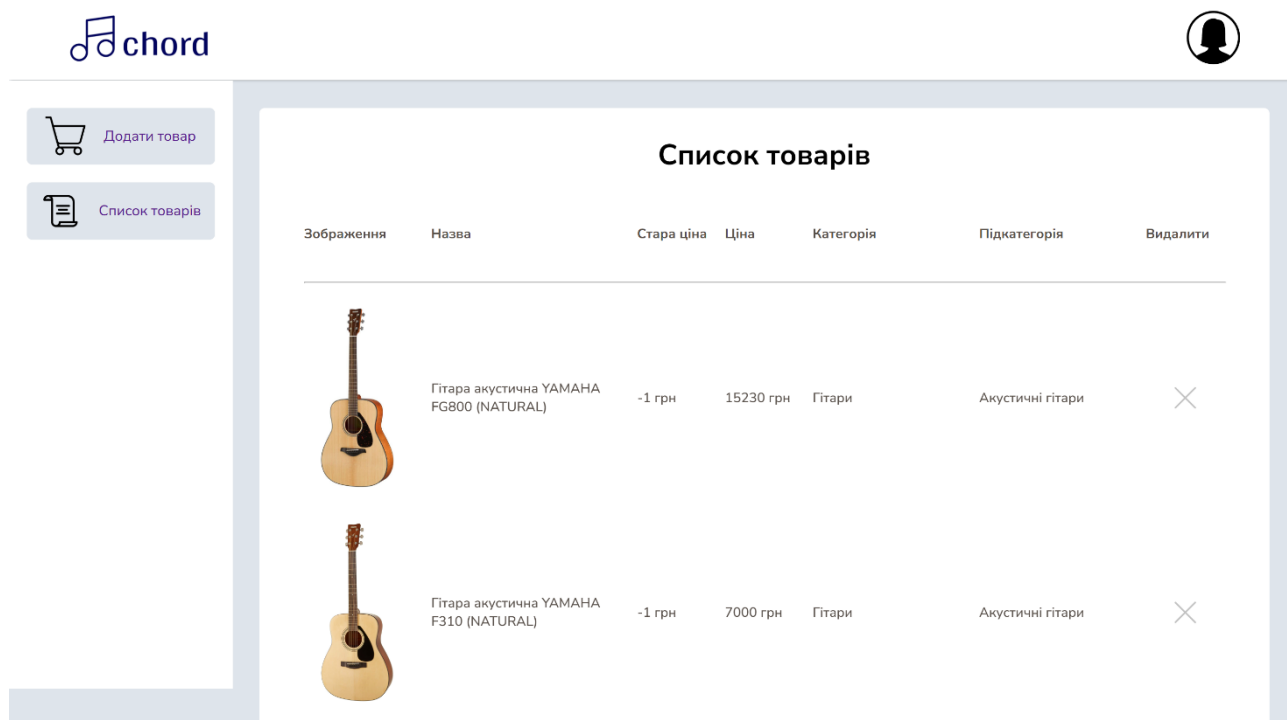


Рисунок 4.7 – Панель зі списком усіх товарів

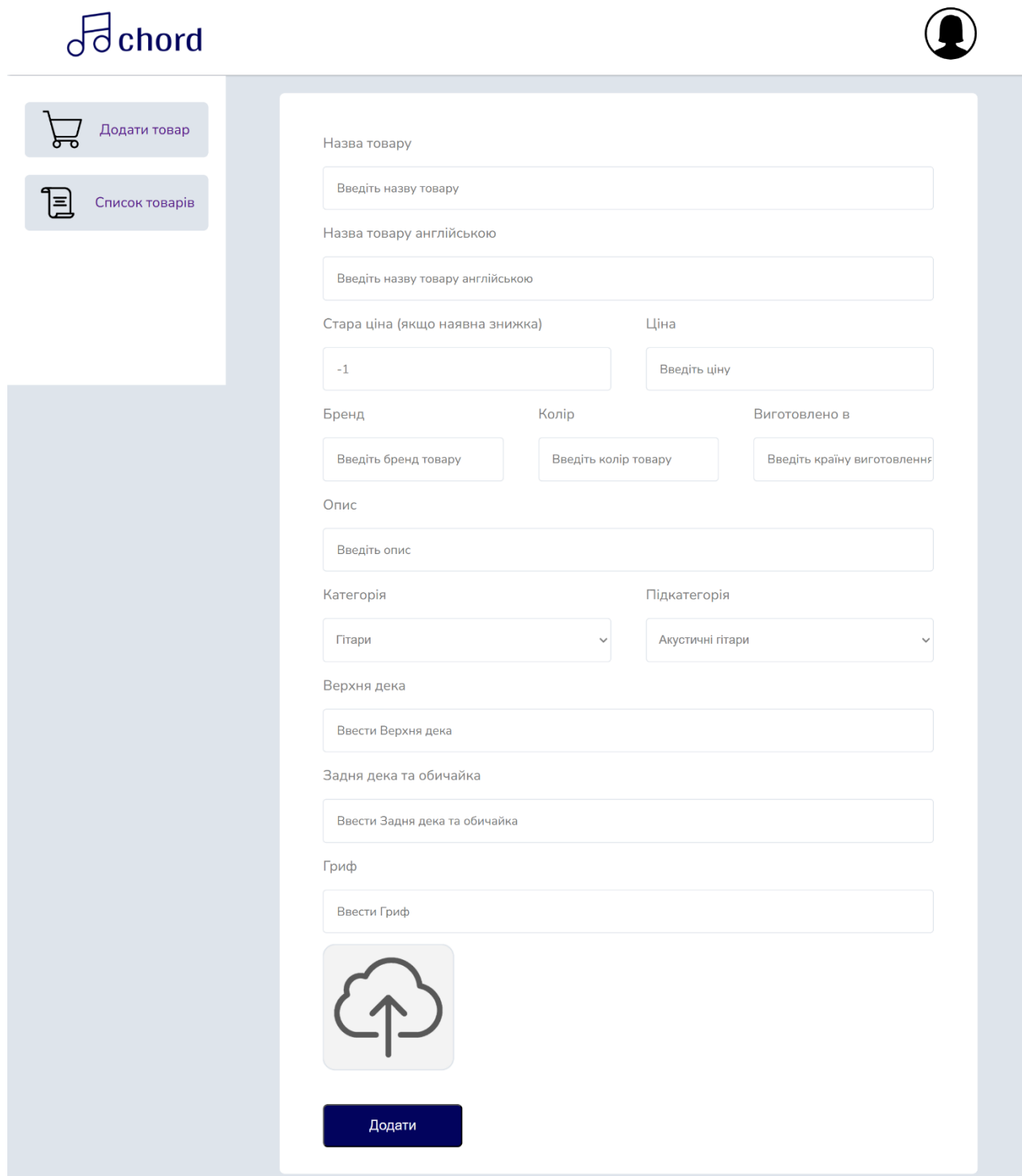
У панелі «Додати товар» адміністратор заповнює усі необхідні поля інформацією про товар, який необхідно додати:

- назва товару;
- назва англійською;
- за наявності знижки – стара ціна;
- поточна ціна;
- бренд;
- колір;
- країна виготовлення;
- опис;
- категорія;
- підкатегорія;
- поля, специфічні до підкатегорії;
- фото.

Усі поля, що залежать від підкатегорії, оновлюються під час нового вибору підкатегорії, в свою чергу поле підкатегорії оновлюється під час вибору

категорії. Тож список завжди залишається коректним і обрати підкатегорію, що належить іншій категорії, або некоректний параметр неможливо (Рисунок 4.8).

Далі потрібно натиснути на кнопку «Додати», щоб додати товар. При невдачі або успіху висвічується інформаційне повідомлення.



chord

Додати товар

Список товарів

Назва товару

Введіть назву товару

Назва товару англійською

Введіть назву товару англійською

Стара ціна (якщо наявна знижка)

-1

Ціна

Введіть ціну

Бренд

Введіть бренд товару

Колір

Введіть колір товару

Виготовлено в

Введіть країну виготовлення

Опис

Введіть опис

Категорія

Гтари

Підкатегорія

Акустичні гітари

Верхня дека

Ввести Верхня дека

Задня дека та обичайка

Ввести Задня дека та обичайка

Гриф

Ввести Гриф

Додати

Рисунок 4.8 – Панель додавання товару

ВИСНОВОК

У роботі вирішено актуальну задачу створення програмного забезпечення для інтернет-магазину музичних інструментів та аксесуарів, що відповідає сучасним вимогам ринку електронної комерції та орієнтований на потреби цільової аудиторії.

Проведено огляд та аналіз сучасного стану електронної комерції, зокрема особливостей розвитку ринку музичних товарів в Україні та світі. Виявлено ключові тенденції, вимоги користувачів і недоліки існуючих онлайн-платформ.

Сформульовано функціональні та нефункціональні вимоги до інтернет-магазину, що враховують особливості користувацького досвіду, захист персональних даних, адаптивність і масштабованість платформи.

Розроблено архітектуру веб-застосунку на основі сучасного стеку технологій: React.js, Node.js, MongoDB. Обґрунтовано доцільність використання SPA-підходу та REST-архітектури для досягнення гнучкості та продуктивності системи.

Створено функціональний інтерфейс з динамічною фільтрацією, багаторівневою категоризацією та сучасним візуальним оформленням. Реалізовано персональні списки бажань і кошик користувача з надійним зберіганням даних.

Реалізовано серверну частину з безпечним зберіганням паролів (хешування), автентифікацією користувачів за допомогою JWT, захистом від несанкціонованого доступу. Налаштовано взаємодію клієнтської та серверної частин через RESTful API.

За результатами роботи отримано програмний продукт, готовий до впровадження у сфері музичної торгівлі, освітніх закладів чи стартапів із онлайн-продажу. Практична цінність розробки полягає у можливості її масштабування, доопрацювання й інтеграції додаткового функціоналу.

Перспективи подальших досліджень полягають у розширенні функціоналу (додавання модуля онлайн-замовлень і оплати, реалізація системи порівняння товарів, впровадження ролей користувачів), а також у вдосконаленні алгоритмів персоналізації та підвищенні рівня захисту даних.

Таким чином, мети роботи досягнуто, всі поставлені завдання виконано. Отримані результати можуть бути рекомендовані до впровадження в практичну діяльність музичних магазинів та освітніх проєктів у сфері онлайн-торгівлі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ani Petrosyan. Number of internet users worldwide from 2005 to 2024 [Electronic resource] // Statista. – 2025. – Access mode: <https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/>.
2. Kemp, S. Digital 2025: Global Overview Report [Electronic resource] // DataReportal. – 2025. – Access mode: <https://datareportal.com/reports/digital-2025-global-overview-report>.
3. Kemp, S. Digital 2025: Ukraine [Electronic resource] // DataReportal. – 2025. – Access mode: <https://datareportal.com/reports/digital-2025-ukraine>.
4. Kemp, S. Digital 2025: Online Shopping Still Popping [Electronic resource] // DataReportal. – 2025. – Access mode: <https://datareportal.com/reports/digital-2025-sub-section-online-shopping>.
5. Statista Research Department. Global retail e-commerce sales 2022–2028. [Electronic resource]. – 2025. – Access mode: <https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/>.
6. Thomann [Electronic resource]. – Access mode: <https://www.thomannmusic.com/>.
7. Guitar Center [Electronic resource]. – Access mode: <https://www.guitarcenter.com/>.
8. Sweetwater [Electronic resource]. – Access mode: <https://www.sweetwater.com/>.
9. Гітарний дім [Електронний ресурс]. – Режим доступу: <https://guitarhouse.com.ua/>.
10. SoundMaster [Електронний ресурс]. – Режим доступу: <https://soundmaster.ua/ua>.
11. МузТорг [Електронний ресурс]. – Режим доступу: <https://muztorg.ua/uk/>.
12. JAM [Electronic resource]. – Режим доступу: <https://jam.ua/ua/>.

13. React – A JavaScript library for building user interfaces [Electronic resource]. – Access mode: <https://react.dev/>.
14. Vue.js – The Progressive JavaScript Framework [Electronic resource]. – Access mode: <https://vuejs.org/>.
15. Angular – One framework. Mobile & desktop [Electronic resource]. – Access mode: <https://angular.io/>.
16. Kumar S., Kumar A., Kumar S., Kumar M. A Comparative Study of Web Application Frameworks for Modern Web Development // MAKH International Journal of Information Technology. – 2024. – Vol. 3, No. 1. – P. 1–10.
17. Django Project Documentation [Electronic resource]. – Access mode: <https://docs.djangoproject.com/>.
18. Flask Documentation [Electronic resource]. – Access mode: <https://flask.palletsprojects.com/>.
19. Spring Boot Reference Documentation [Electronic resource]. – Access mode: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>.
20. Node.js Documentation [Electronic resource]. – Access mode: <https://nodejs.org/>.
21. Express.js Documentation [Electronic resource]. – Access mode: <https://expressjs.com/>.
22. Choma D., Chwaleba K., Dzieńkowski M. The efficiency and reliability of backend technologies: express, Django, and spring boot // Informatyka Automatyka Pomiary w Gospodarce i Ochronie Środowiska. – 2023. – Vol. 13, No. 4. – P. 90–95.
23. WebStorm Documentation [Electronic resource]. – Access mode: <https://www.jetbrains.com/help/webstorm>.
24. Sublime Text Documentation [Electronic resource]. – Access mode: <https://www.sublimetext.com/docs/>.
25. Atom Documentation [Electronic resource]. – Access mode: <https://flight-manual.atom.io/>.

26. Visual Studio Code Documentation [Electronic resource]. – Access mode: <https://code.visualstudio.com/docs>.
27. npm Documentation [Electronic resource]. – Access mode: <https://docs.npmjs.com/>.
28. Git Documentation [Electronic resource]. – Access mode: <https://git-scm.com/doc>.
29. Postman Documentation [Electronic resource]. – Access mode: <https://learning.postman.com/docs>.
30. MongoDB Atlas Documentation [Electronic resource]. – Access mode: <https://www.mongodb.com/docs/atlas>.
31. MongoDB for Visual Studio Code [Electronic resource]. – Access mode: <https://marketplace.visualstudio.com/items?itemName=mongodb.mongodb-vscode>.
32. cors Documentation [Electronic resource]. – Access mode: <https://expressjs.com/en/resources/middleware/cors.html>.
33. bcrypt Documentation [Electronic resource]. – Access mode: <https://www.npmjs.com/package/bcrypt>.
34. jsonwebtoken Documentation [Electronic resource]. – Access mode: <https://www.npmjs.com/package/jsonwebtoken>.
35. multer Documentation [Electronic resource]. – Access mode: <https://www.npmjs.com/package/multer>.
36. Open Web Application Security Project. OWASP Top 10 – 2021: The Ten Most Critical Web Application Security Risks [Electronic resource]. – Access mode: <https://owasp.org/www-project-top-ten/>.
37. Ali W., Shafique M.U., Majeed M.A., Raza A. Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics // International Journal of Computer Applications. – 2019. – Vol. 177, No. 4. – P. 7–13.
38. MongoDB Documentation [Electronic resource]. – Access mode: <https://www.mongodb.com/docs/>

ДОДАТОК А
Технічне завдання

Вступ

Розробляється програмний продукт — веб-застосунок для електронної комерції у сфері продажу музичних інструментів та аксесуарів, призначений для індивідуальних користувачів і невеликих магазинів. Область застосування — онлайн-торгівля, навчальні заклади, творчі спільноти.

А.1 Підстави для розробки

Підставою для виконуваної розробки в межах дипломної кваліфікаційної роботи бакалавра є завдання, що було визначене за темою «Розробка програмного забезпечення інтернет-магазину музичних інструментів та аксесуарів» і затверджене у відповідності з наказом №209 від 28квітня 2025 року за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма призначена для організації онлайн-продажу музичних інструментів і аксесуарів, забезпечення пошуку, фільтрації, перегляду каталогу, формування списку бажань і кошика, а також адміністрування асортименту товарів.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

- актуальний каталог товарів із деталізованими характеристиками для кожної підкатегорії;
- динамічна фільтрація, пошук, сортування товарів;

- персоналізований простір для зареєстрованих користувачів: wishlist, кошик;
- тимчасове збереження вибраних товарів для гостей;
- адміністративна панель для додавання, перегляду, видалення товарів;
- безпечна реєстрація, аутентифікація, збереження паролів у зашифрованому вигляді (bcrypt), автентифікація через JWT.

A.3.2 Вимоги до надійності

- надійність зберігання даних користувачів і товарів;
- відсутність критичних збоїв при типових операціях;
- контроль коректності введення і обробки даних.

A.3.3 Умови експлуатації

Програмний комплекс Chord працює на будь-якому ПК або ноутбуці з операційною системою Windows, Linux чи MacOS. Необхідна стабільна робота підключення до Інтернету для всіх модулів (frontend, backend, admin). Система експлуатується в умовах звичайного офісу чи домашнього використання. Обслуговування не потребує спеціальної кваліфікації: користувачами є адміністратор магазину та звичайні покупці; запуск/зупинка модулів здійснюється за допомогою `npm/node` стандартними командами.

A.3.4 Вимоги до складу та параметрів технічних засобів

Мінімальні вимоги до апаратного забезпечення:

- ПК/ноутбук із процесором Intel i3, AMD Ryzen 3 або вище;
- 4 ГБ оперативної пам'яті;
- 1 ГБ вільного місця на HDD/SSD;

- стабільний доступ до Інтернету.

Операційна система — Windows, Linux, MacOS.

A.3.5 Вимоги до інформаційної та програмної сумісності

Клієнтська частина реалізована на основі React.js, серверна частина працює під Node.js (версія не нижче 16.0) із використанням Express.js, для зберігання даних використовується база даних MongoDB Atlas, встановлення залежностей здійснюється через менеджер пакетів npm. Система не потребує інтеграції зі сторонніми сервісами, окрім підключення до Інтернету.

ДОДАТОК Б
Текст програми

Б.1 Текст файла Header.jsx

```

import { Link, useNavigate } from "react-router-dom";
import "../Header.css";
import { useState, useEffect, useContext, useRef } from
"react";
import { ShopContext } from "../../Context/ShopContext";

import logo from "../Assets/logo.png";
import cart_icon from "../Assets/shopping-cart.png";
import log_in_icon from "../Assets/log_in_sign_up.png";
import wishlist_logo from "../Assets/heart.png";
import search_icon from "../Assets/search.png";
import hamburger_icon from "../Assets/hamburger.png";
import hamburger_black_icon from
"../Assets/hamburger_02035d.png";

export const Header = () => {
  const { getNumberOfItemsInCart } = useContext(ShopContext);
  const [isHovered, setIsHovered] = useState(false);
  const [isAccountHovered, setIsAccountHovered] =
useState(false);
  const [isMobileView, setIsMobileView] =
useState(window.innerWidth <= 800);
  const [searchTerm, setSearchTerm] = useState("");
  const inputRef = useRef();
  const navigate = useNavigate();

  const handleSearch = () => {
    if (searchTerm.trim()) {
      navigate(`/search?q=${encodeURIComponent(searchTerm.trim())}`);
    }
  };

  // Toggle based on screen size
  useEffect(() => {
    const handleResize = () => {
      setIsMobileView(window.innerWidth <= 800);
      const mainContent = document.querySelector(".header-
main-content");
      const menuContent = document.querySelector(".header-
menu-content");
      const menuContentContainer = document.querySelector(
        ".header-menu-content-container"
      );
      const menu = document.querySelector(".menu");
      const mainTools = document.querySelector(".main-
tools");

```

```

const searchContainerMV = document.querySelector(
  ".main_search_container_mobileView"
);
const search = document.querySelector(".main-search");

if (window.innerWidth <= 800) {
  if (!searchContainerMV.contains(search)) {
    searchContainerMV.prepend(search);
  }
  if (!mainContent.contains(menuContent)) {
    mainContent.prepend(menuContent);
  }
  menu.classList.add("dropdown");
  menu.classList.remove("default");
} else {
  if (!mainTools.contains(search)) {
    mainTools.prepend(search);
  }
  if (!menuContentContainer.contains(menuContent)) {
    menuContentContainer.prepend(menuContent);
  }
  menu.classList.remove("dropdown");
  menu.classList.add("default");
}
};

window.addEventListener("resize", handleResize);
handleResize();

return () => window.removeEventListener("resize",
handleResize);
}, []);

return (
  <>
  <div
    className={`overlay ${isHovered ? "visible" : ""}`}
    style={isHovered ? { zIndex: 7 } : { zIndex: 1 }}
  ></div>

  <header
    className="header"
    style={isHovered ? { zIndex: 8 } : { zIndex: 1 }}
  >
    <div className="header-main-content-container">
      <div className="header-main-content">
        <Link to={"/home"} className="main-logo">
          <img src={logo} alt="Logo" />
        </Link>

        <div className="main-tools">
          <div className="main-search">
            {

```

```

        <input
            ref={inputRef}
            type="text"
            placeholder="Пошук"
            aria-label="Пошук"
            value={searchTerm}
            onChange={ (e) =>
setSearchTerm(e.target.value) }
            onKeyDown={ (e) => {
                if (e.key === "Enter") {
                    handleSearch();
                }
            }}
        />
    }
    <button
        onClick={ (e) => {
            if (!isMobileView) {
                handleSearch();
            }
        }}
    >
        <img src={search_icon} alt="Search" />
    </button>
</div>

<Link to={"/wishlist"} className="main-tools-
wishlist">
    <img src={wishlist_logo} alt="" />
</Link>
<Link to={"/cart"} className="main-tools-cart">
    <img src={cart_icon} alt="" />
    <div className="main-cart-count">
        {getNumberOfItemsInCart()}
    </div>
</Link>

<div
    className="main-tools-account"
    onMouseEnter={() => {
        setIsAccountHovered(true);
    }}
    onMouseLeave={() => {
        setIsAccountHovered(false);
    }}
>
    {localStorage.getItem("auth-token") ? (
        <>
            <Link
                className="authorization-icon"
                to={" "}>
                <img src={log_in_icon} alt="" />
            </Link>
            <div

```

```

        className="account_dropdown_mobileView"
        style={
            isMobileView && isAccountHovered
            ? { display: "flex" }
            : { display: "none" }
        }
    >
    <button
        className="exit-button"
        onClick={() => {
            localStorage.removeItem("auth-
token");

            window.location.replace("/home");
        }}
    >
        Вихід
    </button>
</div>
<ul className="account-links exit">
    <li className="authorization-exit">
        <button
            className="exit-button"
            onClick={() => {
                localStorage.removeItem("auth-
token");

                window.location.replace("/home");
            }}
        >
            Вихід
        </button>
    </li>
</ul>
</>
) : (
    <>
        <Link to={"/log_in"}>
            <img src={log_in_icon} alt="" />
        </Link>
        <ul className="account-links enter">
            <li className="authorization-link">
                <Link to={"/log_in"}>Вхід</Link>
            </li>
            <li>
                <Link
to={"/sign_up"}>Реєстрація</Link>
            </li>
        </ul>
    </>
    </div>
</div>
</div>
</div>
</div>

```

```

<div className="header-menu-content-container">
  <div
    className="header-menu-content"
    onMouseEnter={() => isMobileView    &&
setIsHovered(true)}
    onMouseLeave={() => isMobileView    &&
setIsHovered(false)}
  >
    <div
      className="vertical-menu"
      onMouseEnter={() => window.innerWidth > 800 &&
setIsHovered(true)}
      onMouseLeave={() =>
        window.innerWidth > 800    &&
setIsHovered(false)}
    >
      <div className="vertical-menu-title">
        <div
          className="vertical_menu_icon_container">
            <img
              className="vertical_menu_black_icon"
              src={hamburger_black_icon}
              alt=""
            />
            <img
              className="vertical_menu_icon"
              src={hamburger_icon}
              alt=""
            />
          </div>
          <p
            className="vertical_menu_title_text">Каталог</p>
          </div>
          <ul
            className="dropdown"
            style={{ display: isHovered ? "grid" : "none"
}}
          >
            <li className="dropdown-child">
              <Link to={"/guitars"}>Гітари</Link>
            </li>
            <li className="dropdown-child">
              <Link
                to={"/guitar_equipment"}>Гітарне
обладнання</Link>
            </li>
            <li className="dropdown-child">
              <Link
to={"/keyboards_pianos"}>Клавішні</Link>
            </li>
            <li className="dropdown-child">
              <Link to={"/keyboards_pianos_equipment"}>

```

```

        Обладнання для клавішних
      </Link>
    </li>
    <li className="dropdown-child">
      <Link
to={"/string_instruments"}>Смичкові</Link>
    </li>
    <li className="dropdown-child">
      <Link to={"/string_instruments_equipment"}>
        Обладнання для смичкових
      </Link>
    </li>
    <li className="dropdown-child">
      <Link
to={"/percussion_drums"}>Ударні</Link>
    </li>
    <li className="dropdown-child">
      <Link to={"/percussion_drums_equipment"}>
        Обладнання для ударних
      </Link>
    </li>
  </ul>
</div>

<ul
  className="menu"
  style={{
    display: isHovered
      ? isMobileView
        ? "grid"
        : "flex"
      : isMobileView
        ? "none"
        : "flex",
  }}
>
  <li className="menu-child">
    <Link to={"/about_us"}>Про нас</Link>
  </li>
  <li className="menu-child">
    <Link to={"/payment_and_delivery"}>Оплата і
доставка</Link>
  </li>
  <li className="menu-child">
    <Link to={"/service_center"}>Сервіс</Link>
  </li>
  <li className="menu-child">
    <Link to={"/contacts"}>Контакти</Link>
  </li>
  <li className="menu-child">
    <Link href="#">Обмін і повернення</Link>
  </li>
</ul>

```

```

        </div>
    </div>

    <div
className="main_search_container_mobileView"></div>
    </header>
</>
);
};

```

Б.2 Текст файла ShopCategory.jsx

```

import { useContext, useState, useEffect, useRef } from
"react";
import { useParams, Link } from "react-router-dom";
import { ShopContext } from "../Context/ShopContext";
import { Item } from "../Components/Item/Item";
import { Filter } from "../Components/Filter/Filter";
import { Breadcrumbs } from "../Components/Breadcrumbs/Breadcrumbs";
import { Sorter } from "../Components/Sorter/Sorter";
import {
    categoryTranslations,
    subcategoryTranslations,
} from "../Utils/translations";

import "../CSS/ShopCategory_SearchResults.css";

export const ShopCategory = () => {
    const { all_products } = useContext(ShopContext);
    const { category, subcategory } = useParams();

    const [sortOption, setSortOption] = useState("");
    const [sortedProducts, setSortedProducts] = useState([]);
    const [filteredProducts, setFilteredProducts] =
useState([]);
    const [filteredByCategory, setFilteredByCategory] =
useState([]);
    const [isFiltersVisible, setIsFiltersVisible] =
useState(false);
    const [isMobile, setIsMobile] = useState(window.innerWidth
< 900);

    const filterRef = useRef(null);

    const excludedKeys = [
        "name",
        "name_eng",
        "image",
        "old_price",
    ]

```

```

    "category",
    "subcategory",
    "id",
    "_id",
    "date",
    "description",
    "__v",
  ];

  // Handle window resize
  useEffect(() => {
    const handleResize = () => {
      setIsMobile(window.innerWidth < 900);
    };

    window.addEventListener("resize", handleResize);
    return () => window.removeEventListener("resize",
handleResize);
  }, []);

  // Handle click outside for mobile
  useEffect(() => {
    if (!isMobile) return;

    const handleClickOutside = (event) => {
      if (filterRef.current // filterRef.current is null
!filterRef.current.contains(event.target)) {
        setIsFiltersVisible(false);
      }
    };

    document.addEventListener("mousedown",
handleClickOutside);
    return () => {
      document.removeEventListener("mousedown",
handleClickOutside);
    };
  }, [isMobile]);

  // Filter products by category and subcategory
  useEffect(() => {
    const filtered = all_products.filter(
      (item) =>
        item.category === category &&
        (!subcategory || item.subcategory === subcategory)
    );
    setFilteredByCategory(filtered);
  }, [category, subcategory, all_products]);

  const subcategories = [
    ...new Set(
      all_products
        .filter((item) => item.category === category)

```

```

        .map((item) => item.subcategory)
    ),
];

return (
    <>
        <Breadcrumbs />
        <div
            className="shopCategory_searchResults_page_container">
                <h2 className="shopCategory_searchResults_headline">
                    {subcategory
                        ? `${subcategoryTranslations[subcategory] ||
subcategory}`
                        : `${categoryTranslations[category] ||
category}` }
                </h2>

                {!subcategory && (
                    <div
                        className="shopCategory_subcategory_container">
                            {subcategories.map((sub) => {
                                const matchedItem = all_products.find(
                                    (item) => item.subcategory === sub &&
item.category === category
                                );
                                return (
                                    <Link
                                        to={`/${category}/${sub}`}
                                        className="shopCategory_subcategory_box"
                                        key={sub}
                                        style={{
                                            backgroundImage:
`url (${matchedItem?.image})`,
                                            backgroundSize: "contain",
                                            backgroundPosition: "center",
                                            backgroundRepeat: "no-repeat",
                                        }}
                                    >
                                        <div className="subcategory_name">
                                            {subcategoryTranslations[sub] || sub}
                                        </div>
                                    </Link>
                                );
                            })}
                        </div>
                    )}

                    <div
                        className="shopCategory_searchResults_sorter_wrap"
                        style={isFiltersVisible ? { justifyContent: "end" }
: {}}
                    >
                        <button

```

```

        onClick={() => setIsFiltersVisible(true)}
        className={
          !isFiltersVisible
            ? `filters_button button_visible`
            : `filters_button button_NotVisible`
        }
      >
        Фільтри
      </button>

      <Sorter
        products={filteredProducts}
        sortOption={sortOption}
        setSortOption={setSortOption}
        setSortedProducts={setSortedProducts}
      />
    </div>

    <div
      className="shopCategory_searchResults_container">
      <div
        ref={filterRef}
        className={
          isFiltersVisible ? `filters_visible` :
          `filters_notVisible`
        }
      >
        <Filter
          products={filteredByCategory}
          onFilteredChange={setFilteredProducts}
          excludedKeys={excludedKeys}
        />
      </div>

      <div
        className="shopCategory_searchResults_right_col">
        <div
          className="shopCategory_searchResults_products">
            {sortedProducts.map((item) => (
              <Item
                key={item.id}
                id={item.id}
                name={item.name}
                name_eng={item.name_eng}
                image={item.image}
                price={item.price}
                old_price={item.old_price}
                category={item.category}
                subcategory={item.subcategory}
              />
            ))}
          </div>
        </div>
      </div>

```

```

        </div>
    </div>
</>
);
};

```

Б.2 Текст файлу Filter.jsx

```

import { useEffect, useState } from "react";
import "../Filter.css";
import { parametersTranslations } from
"../../Utils/translations";

export const Filter = ({
  products,
  onFilteredChange,
  excludedKeys = [
    "_id",
    "name",
    "name_eng",
    "image",
    "old_price",
    "id",
    "description",
    "date",
    "__v",
  ],
}) => {
  const [availableFilters, setAvailableFilters] =
useState({});
  const [activeBasicFilters, setActiveBasicFilters] =
useState({});
  const [activeDetailsFilters, setActiveDetailsFilters] =
useState({});
  const [detailsKeys, setDetailsKeys] = useState([]);
  const [priceRange, setPriceRange] = useState([0, 0]);
  const [selectedPrice, setSelectedPrice] = useState([0, 0]);

  // Extract filters and price range from products
  useEffect(() => {
    if (products.length === 0) {
      setAvailableFilters({});
      setDetailsKeys([]);
      setPriceRange([0, 0]);
      setSelectedPrice([0, 0]);
      return;
    }

    const dynamicDetailsFilters = {};
    const dynamicBasicFilters = {};

```

```

let min = Infinity;
let max = -Infinity;

products.forEach((product) => {
  Object.entries(product).forEach(([key, value]) => {
    if (key === "price") {
      min = Math.min(min, value);
      max = Math.max(max, value);
    } else if (key === "details" && typeof value ===
"object") {
      Object.entries(value).forEach(([detailsKey,
detailsValue]) => {
        if (!dynamicDetailsFilters[detailsKey])
          dynamicDetailsFilters[detailsKey] = new Set();

dynamicDetailsFilters[detailsKey].add(detailsValue);
      });
      } else if (!excludedKeys.includes(key)) {
        if (!dynamicBasicFilters[key])
dynamicBasicFilters[key] = new Set();
        dynamicBasicFilters[key].add(value);
      }
    });
  });

  const available = {};
  for (const key in dynamicBasicFilters) {
    available[key] = Array.from(dynamicBasicFilters[key]);
  }

  const availableDetails = {};
  for (const key in dynamicDetailsFilters) {
    availableDetails[key] =
Array.from(dynamicDetailsFilters[key]);
  }

  setAvailableFilters({ ...available, ...availableDetails
});

  setDetailsKeys(Object.keys(dynamicDetailsFilters));
  setPriceRange([min, max]);
  setSelectedPrice([min, max]);
}, [products]);

// Apply filters
useEffect(() => {
  const filtered = products.filter((item) => {
    if (item.price < selectedPrice[0] || item.price >
selectedPrice[1])
      return false;

    const matchesBasic =
Object.entries(activeBasicFilters).every(
      ([key, values]) => {

```

```

        if (!values.length) return true;
        return values.includes(item[key]);
    }
    );

    const matchesDetails =
Object.entries(activeDetailsFilters).every(
    ([key, values]) => {
        if (!values.length) return true;
        return values.includes(item.details?.[key]);
    }
    );

    return matchesBasic && matchesDetails;
});

    onFilteredChange(filtered);
    }, [products, selectedPrice, activeBasicFilters,
activeDetailsFilters]);

const handleCheckboxChange = (key, value, isDetail = false)
=> {
    const current = isDetail
        ? activeDetailsFilters[key] || []
        : activeBasicFilters[key] || [];

    const newValues = current.includes(value)
        ? current.filter((v) => v !== value)
        : [...current, value];

    if (isDetail) {
        setActiveDetailsFilters({ ...activeDetailsFilters,
[key]: newValues });
    } else {
        setActiveBasicFilters({ ...activeBasicFilters, [key]:
newValues });
    }
};

const handlePriceChange = (e, type) => {
    const val = Number(e.target.value);
    const newPrice = [...selectedPrice];
    if (type === "min") {
        newPrice[0] = Math.min(val, newPrice[1]);
    } else {
        newPrice[1] = Math.max(val, newPrice[0]);
    }
    setSelectedPrice(newPrice);
};

return (
    <div className="filter">
        <h3>Фільтри</h3>

```

```

<div className="filter-section">
  <label>Ціна</label>
  <div className="price-inputs">
    <input
      type="number"
      value={selectedPrice[0]}
      min={priceRange[0]}
      max={selectedPrice[1]}
      onChange={ (e) => handlePriceChange(e, "min")}
    />
    <span>—</span>
    <input
      type="number"
      value={selectedPrice[1]}
      min={selectedPrice[0]}
      max={priceRange[1]}
      onChange={ (e) => handlePriceChange(e, "max")}
    />
  </div>
</div>

=> {
  (Object.entries(availableFilters).map(([key, values]))
    const isDetail = detailsKeys.includes(key);
    return (
      <div className="filter-section" key={key}>
        <label>{parametersTranslations[key]} ||
key}</label>
        {values.map((val) => (
          <div key={val}>
            <input
              type="checkbox"
              id={`_${key}-${val}`}
              checked={
                isDetail
                ?
activeDetailsFilters[key]?.includes(val) || false
                :
activeBasicFilters[key]?.includes(val) || false
              }
              onChange={() => handleCheckboxChange(key,
val, isDetail)}
            />
            <label                                htmlFor={`_${key}-
_${val}`}>{val}</label>
          </div>
        ))}
      </div>
    );
  )}
</div>
);
};

```

ДОДАТОК В
Слайди презентації

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ

Розробка програмного забезпечення інтернет -
магазину музичних інструментів та аксесуарів
Software Development for an Online Store of
Musical Instruments and Accessories

Виконала: Керівник:

студентка групи КНТ211 к.т.н., доцент
Єлизавета РОМАНОВСЬКА Олександр СТЕПАНЕНКО

Рисунок В.1 – Слайд 1

Вступ

- Об'єкт роботи – розробка вебзастосунку для електронної комерції у сфері продажу музичних інструментів та аксесуарів .
- Предмет роботи – методи, технології та інструменти розробки інтернет -магазину, що реалізує функції пошуку, фільтрації, перегляду та підбору товарів.
- Мета роботи – створення інтернет -магазину, призначеного для продажу музичних інструментів і аксесуарів.

2

Рисунок В.2 – Слайд 2

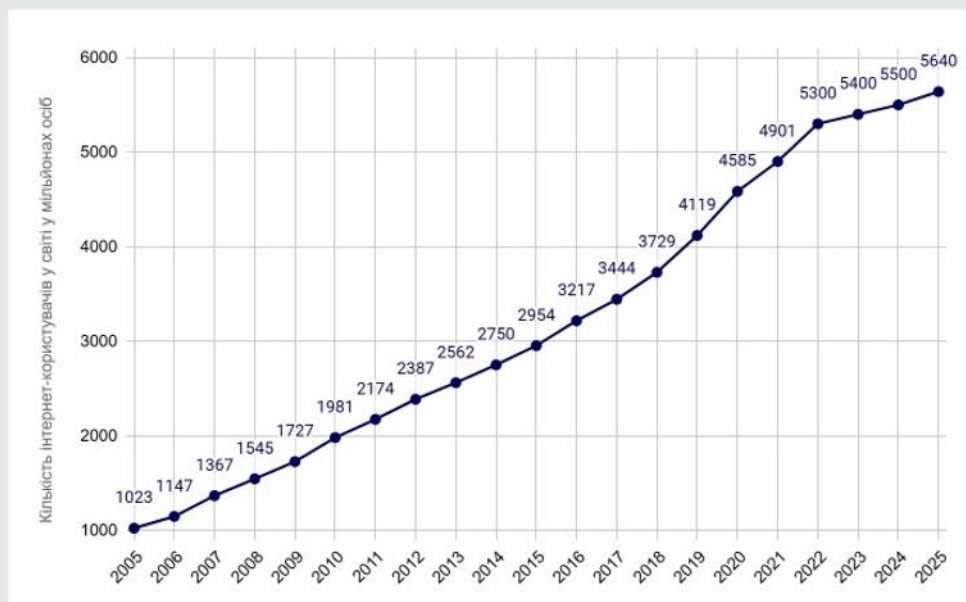
Завдання роботи

- провести огляд сучасного стану електронної комерції, зокрема вітчизняних і зарубіжних рішень;
- проаналізувати вимоги до інтернет-магазину та сформулювати технічне завдання;
- розробити структуру бази даних для зберігання інформації про товари та користувачів;
- проаналізувати доступні інструменти і реалізувати клієнтську та серверну частини;
- провести тестування функціоналу та проаналізувати результати.

3

Рисунок В.3 – Слайд 3

Кількість інтернет-користувачів у світі, 2005-2025 рр., млн осіб



4

Рисунок В.4 – Слайд 4

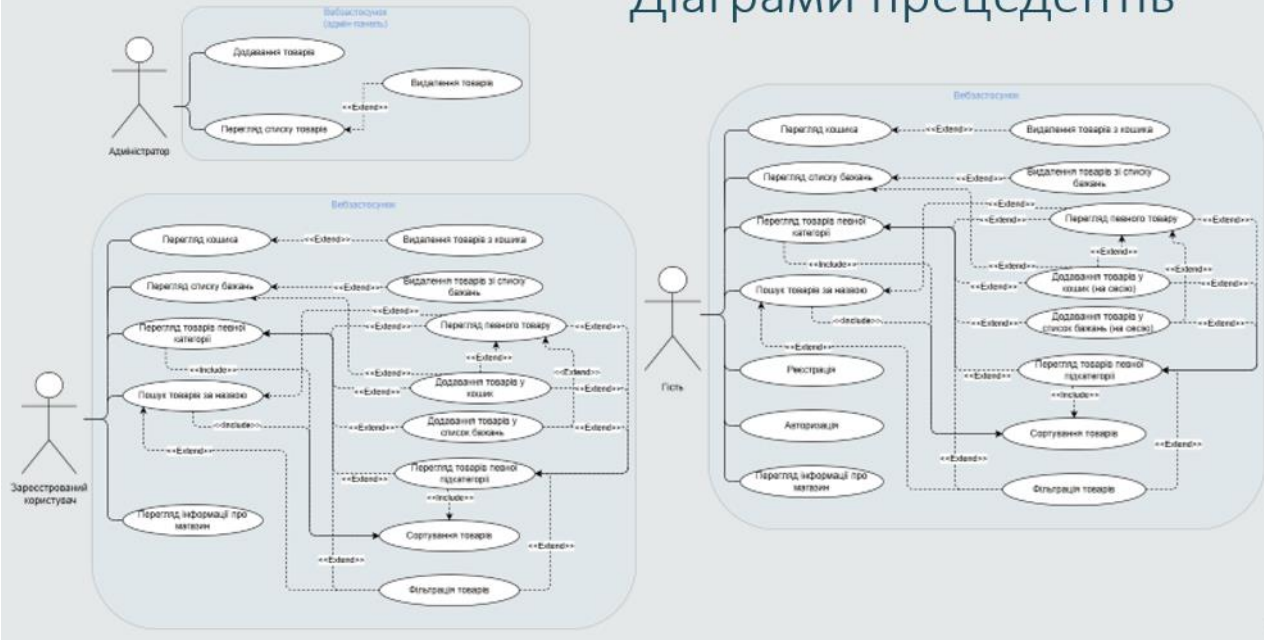
Порівняння існуючих рішень з розробленим

	Гітарний дім	МузТорг	Soundmaster	Jam	Chord
Сучасний дизайн	+	±	±	+	+
Респонсивність	-	+	±	+	+
Список Бажань	+	+	+	+	+
Порівняння товарів	+	+	+	+	-
Онлайн-консультації	+	+	+	+	-
Відеоогляди	+	+	-	+	-

5

Рисунок В.5 – Слайд 5

Діаграми прецедентів



6

Рисунок В.6 – Слайд 6

	React.js	Vue.js	Angular
Архітектура	Компонентна	Компонентна	Компонентна (MVVM)
Продуктивність	Висока	Висока	Висока
Поріг входу	Середній	Низький	Високий
Документація	Вичерпна, активна спільнота	Добра, менш розгалужена	Офіційна, складна
Гнучкість	Дуже висока	Висока	Висока
Широта екосистеми	Максимальна	Менша	Висока
Масштабованість	Відмінна	Добра	Відмінна
Підтримка e-commerce бібліотек	Велика	Обмежена	Помірна

Порівняльна характеристика бібліотек для клієнтської частини

7

Рисунок В.7 – Слайд 7

	Node.js (Express.js)	Python (Django/Flask)	Java (SpringBoot)
Мова програмування	JavaScript	Python	Java
Підтримка REST API	Проста, модульна	Django – повна, Flask – базова	Вбудована
Продуктивність	Висока (асинхронна)	Висока (синхронна)	Дуже висока
Поріг входу	Низький/середній	Django – середній, Flask – низький	Високий
Гнучкість	Дуже висока	Django – обмежена, Flask – висока	Висока
Швидкість розробки	Висока	Django – середня, Flask – висока	Середня
Єдність зі стеком	Повна (JS всюди)	Відсутня	Відсутня
Застосування у стартапах	Дуже поширене	Помірне	Рідко

Порівняльна характеристика бібліотек для серверної частини

8

Рисунок В.8 – Слайд 8

Порівняння середовищ розробки

	Visual Studio Code	WebStorm	Sublime Text	Atom
Підтримка JavaScript	Відмінна	Відмінна	Дуже добра	Дуже добра
Розширення/плагіни	Дуже багато (безкоштовно)	Багато (частин платна)	Обмежено	Обмежено
Інтеграція з git	Глибока, інтуїтивна	Вбудована	Через додаткові пакети	Через додаткові пакети
Вартість	Безкоштовно	Платно (30 днів пробного періоду)	Безкоштовно/платно	Безкоштовно
Продуктивність	Висока	Висока	Висока	Задовільна
Підтримка плагінів для Node.js, React	Відмінна	Відмінна	Обмежена	Обмежена
Документація	Дуже розвинена	Розвинена	Помірна	Застаріла

Рисунок В.9 – Слайд 9

Порівняння реляційних і нереляційних БД

	Реляційні БД (SQL)	Нереляційні БД (NoSQL)
Тип структури	Таблична, фіксована схема	Гнучка, документна або ключ-значення
Гнучкість у зміні схеми	Обмежена	Висока
Масштабованість	Вертикальна	Горизонтальна
Оптимально для	Транзакції, суворі зв'язки	Динамічні, різномірні дані, швидкий ріст
Приклад	PostgreSQL, MySQL	MongoDB, CouchDB
Продуктивність при великих обсягах даних	Знижується при складних зв'язках	Зберігається завдяки денормалізації
Формат зберігання	Таблиці (рядки/стовпці)	Документи (JSON/BSON)
Актуальність для e-commerce	Традиційно (за фіксованої моделі)	Сучасний стандарт для кастомних/гнучких платформ

Рисунок В.10 – Слайд 10

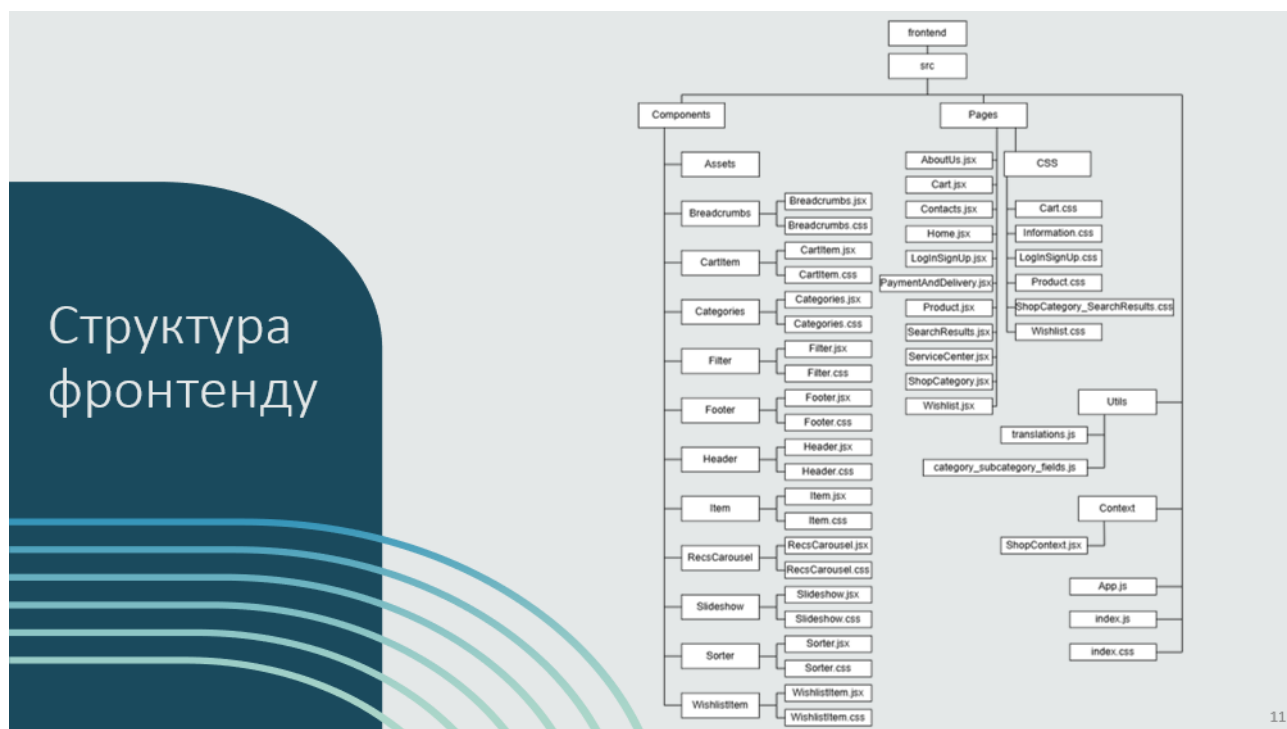


Рисунок В.11 – Слайд 11

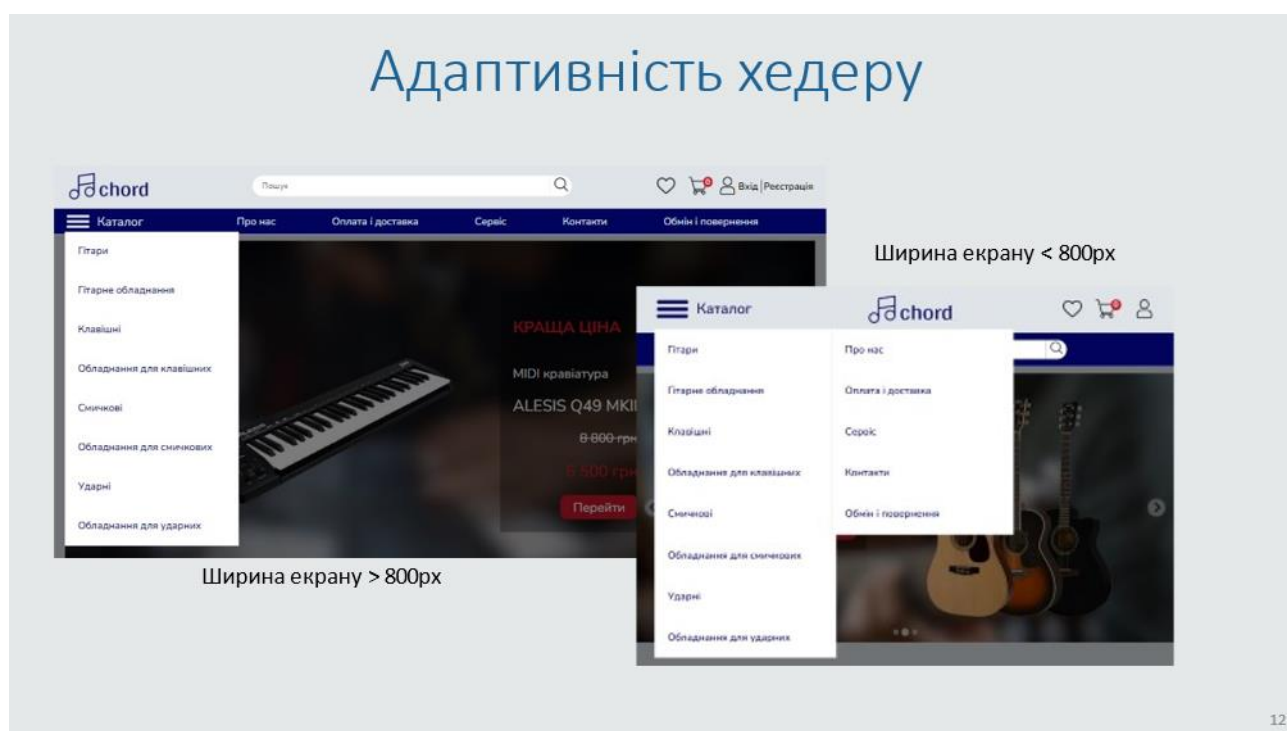


Рисунок В.12 – Слайд 12

Поля сутності Product

Поле	Тип	Призначення
Id	Number	Унікальний ідентифікатор
name	String	Назва товару українською
name_eng	String	Назва англійською
image	String	Шлях до зображення
category	String	Основна категорія
Subcategory	String	Підкатегорія
price	Number	Ціна
old_price	Number	Стара ціна (якщо наявна знижка)
brand	String	Бренд
color	String	Колір
madeIn	String	Країна виробника
description	String	Опис
details	Object	Об'єкт із параметрами, специфічними для підкатегорії
date	Date	Дата додавання

База даних

Поля сутності Users

Поле	Тип	Призначення
name	String	Ім'я користувача
email	String	E-mail, використовується як логін
password	String	Хешований пароль
cartData	Object	Дані кошика (ідентифікатори)
wishlistData	Array	Масив ід товарів у списку бажань
date	Date	Дата створення акунта

13

Рисунок В.13 – Слайд 13

Інтерфейс клієнтської частини

14

Рисунок В.14 – Слайд 14

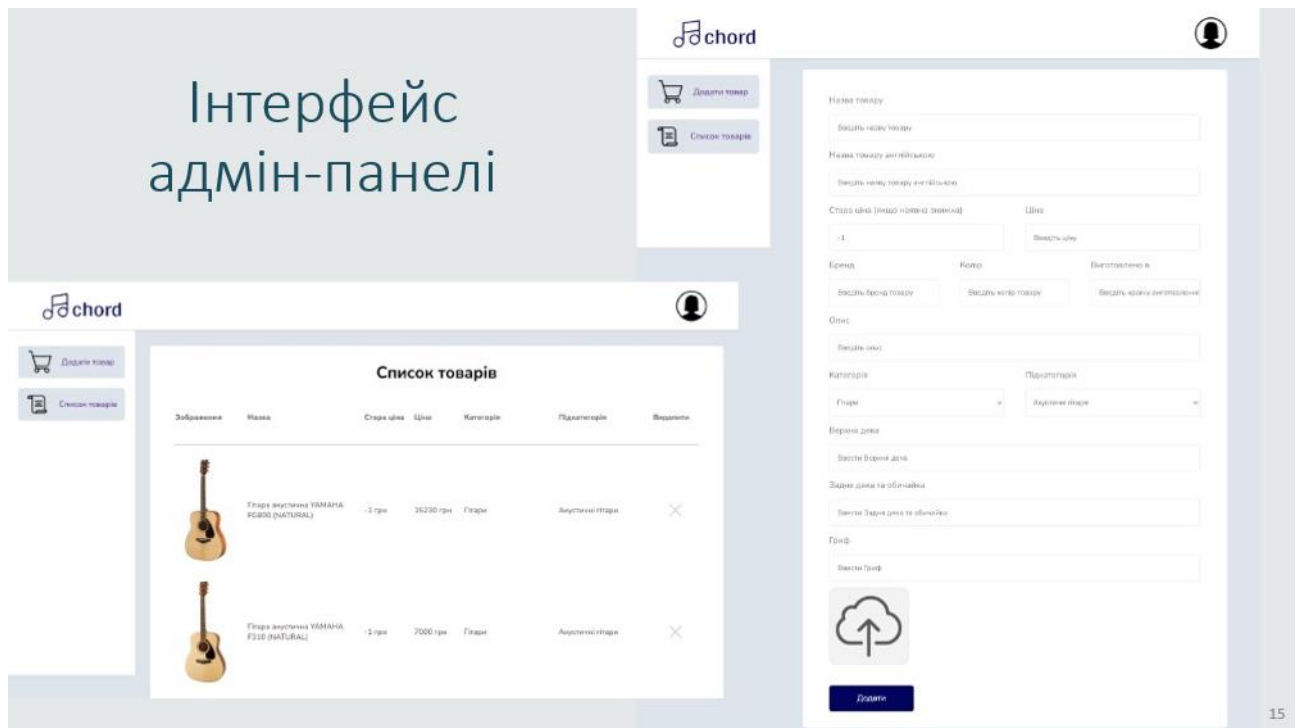


Рисунок В.15 – Слайд 15

Висновок

- створено програмне забезпечення для інтернетмагазину музичних інструментів і аксесуарів відповідно до сучасних вимог та потреб користувачів;
- проведено аналіз ринку електронної комерції та визначено ключові тенденції у сфері музичних товарів;
- сформульовано функціональні та нефункціональні вимоги до системи;
- розроблено архітектуру вебзастосунку на основі React.js, Node.js та MongoDB;
- реалізовано зручний інтерфейс із динамічною фільтрацією, категоризацією, списком бажань і кошиком користувача;
- забезпечено безпечну автентифікацію користувачів та захист даних через JWT;
- розробку можна масштабувати та інтегрувати з додатковими модулями;
- мети роботи досягнуто, всі поставлені завдання виконано.

Рисунок В.16 – Слайд 16