Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Веб-застосунок для продажу музичних платівок
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.200122.01.05.ПЗ

Виконав студент ІІІ курсу, група ІПЗ-20-1 _____ Підпис Войченко Р.О Ініціали, прізвище

Керівник др. фіз.-мат. наук., проф Науковий ступінь, звання _____ Підпис Бедратюк Л.П Ініціали, прізвище

Нормоконтролер канд. техн. наук, доцент Науковий ступінь, звання _____ Підпис Гурман І.В Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Підпис Л. П. Бедратюк Ініціали, прізвище

1 червня 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
02 01 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Войченку Роману Олександровичу
Прізвище, ім'я, по батькові студента

1. Тема кваліфікаційної роботи Веб-застосунок для продажу музичних платівок

Керівник кваліфікаційної роботи Бедратюк Леонід Петрович, др. ф.-м. н., проф
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 05.02.2023 р. № 6
2. Строк подання студентом роботи на кафедру 01.06.2023 р.
3. Вихідні дані до роботи Матеріали переддипломної практики
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі,
проектування програмного забезпечення
програмна реалізація
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Гурман Іван Васильович, канд. техн. наук, доцент	31.05.23	31.05.23
Антиплагіат	Гурман Іван Васильович, канд. техн. наук, доцент	31.05.23	31.05.23

7. Дата видачі завдання « 05 » лютого 2023 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою дипломного проектування, визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12– 31.12.2022	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2023	
3 Проектування програмного забезпечення	01.02 – 28.02 2023	
4 Програмна реалізація з використанням відповідних засобів розробки	01.03 – 10.04.2023	
5 Тестування програмного забезпечення	11.04 – 30.04.2023	
6 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 – 25.05.2023	
7 Попередній захист КвР	травень 2023 (згідно графіка)	
8 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2023	
9 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КвР	з 01.06.2023	

Студент _____ Войченко Р.О
Підпис Ініціали, прізвище

Керівник роботи _____ Бедратюк Л.П
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломного проекту: Веб-застосунок для продажу музичних платівок.

Автор проекту: Войченко Роман Олександрович.

Керівник проекту: Бедратюк Леонід Петрович.

Пояснювальна записка: 66 с.,32 рис.,2 табл., 4дод., 41 джерело.

Графічна частина: 15 слайдів.

Music, E-COMMERCE, JAVASCRIPTREST, MONGODB, NODEJS, ANGULAR, KARMA, RXJS,

Метою кваліфікаційної робти є розробка веб застосунку, яки дозволятиме його користувачам автоматизувати процес пошуку та купівлі вінілових платівок, а також полегшить обробку замовлень для адміністратора системи. Система має сучасний, інтуїтивно-зрозумілий користувачам інтерфейс.

У кваліфікаційній роботі було проаналізовано предметну область задачі, з'ясовані причини через які частого виникпоть проблеми при замовленні вінілових платівок, було проведено аналіз існуючих рішень альтернативних програмних застосунків, а також в записці були порівняні можливі архітектурні рішення для подібних застосунків та можливі патерни які використовуються для розробки веб-застосунків, бцли визначені модулі для системи та здійснена їх програмна реалізація.

Для розробки веб-додатку була використана мова програмування JavaScript, фреймворки Angular та Node.jsта база даних MongoDB.

В результаті було розроблено веб-застосунок для продажу музичних платівок.

01.06.2023р
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ. 200122.01.05.ПЗ	Пояснювальна записка			
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічні документи			
4	А3	КвРІПЗ. 200122.01.05.Е8	UML-діаграма варіантів	1		
			використання			
5	А3	КвРІПЗ. 200122.01.05.Е8	UML-діаграма	1		
			Deployment			
6	А3	КвРІПЗ. 200122.01.05.Е8	Схема БД	1		

					КвРІПЗ.200122.01.05.ПЗ		
Змін.	Арк.	№ докум.	Підпис	Дата			
Виконав		Войченко Р.О		1.06	Веб-застосунок для продажу музичних платівок	Літ.	Арк. 1 / Аркушів 101
Керівник		Бедратюк Л.П		1.06			
Н. контр.		Гурман І. В		1.06	Відомість документів	ХНУ, ІПЗ-20-1	
Зав. каф.		Бедратюк Л.П.		1.06			

ЗМІСТ

ВСТУП

Веб-магазини стали необхідною складовою сучасного електронного бізнесу, який забезпечує зручність та доступність для покупців у будь-який час та місце. У рамках даної кваліфікаційної роботи було розроблено веб-застосунок для продажу музичних платівок. У сучасному світі музичні платівки повертаються в тренд, привертаючи увагу меломанів і колекціонерів. Ця кваліфікаційна робота пропонує створення інтернет-магазину, який спростить процес придбання музичних платівок та надасть зручну платформу для ознайомлення з багатим асортиментом.

Метою проекту є створення функціонального та зручного інтерфейсу для замовлення та оплати музичних платівок в мережі Інтернет. Крім того, проект передбачає розробку системи адміністрування, що дозволить додавати нові платівки, відслідковувати стан замовлень та керувати іншими аспектами магазину.

Однією з головних задач проекту є покращення зручності та швидкості процесу замовлення музичних платівок. Завдяки веб-застосунку покупець зможе швидко знайти потрібний альбом, дізнатися інформацію про його наявність та ціну, вибрати зручний спосіб доставки та оплати.

Також, веб-магазин дозволить вирішити проблему обмеженої кількості точок продажу музичних платівок та відсутності можливості придбати певний альбом в окремих регіонах. Даний проект вирішує ці проблеми шляхом створення централізованого магазину, доступного для покупців з будь-якої точки світу.

У сучасному світі музичні виконавці випускають свої альбоми на вінілі з кількох причин:

– Аудіо якість: Вінілові платівки відтворюють аналоговий звук, що може надати більш теплий та природний звуковий діапазон порівняно з цифровими форматами. Вони мають потужний звук з глибшою динамікою, що може підкреслити музичні нюанси та емоції.

– Естетика та витонченість: Вінілові платівки є предметами мистецтва, оскільки мають обкладинки, великі формати та деталізацію. Вони додають

витонченості та ексклюзивності до альбому, що може привернути увагу фанатів та колекціонерів.

– Колекціонування та відтворення цінності: Вінілові платівки є популярними серед колекціонерів, оскільки вони мають історичну цінність та можуть стати раритетами. Випуск альбому на вінілі може підвищити його цінність та привернути увагу фанатів, які шукають ексклюзивні та обмежені видання.

– Ритуал слухання: Вінілові платівки надають особливий ритуал слухання музики.

– Маркетинг та брендування: Випуск альбому на вінілі може бути стратегічним кроком для виконавця, який допомагає підсилити його бренд та створити попит.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Веб-застосунок для продажу вінілових платівок належить до сфери електронної комерції та музичної індустрії. Він спрямований на надання інтернет-користувачам можливості придбати вінілові платівки з різних жанрів та виконавців. Огляд предметної області включає аналіз його ключових аспектів.

Досліджується стан ринку вінілових платівок, його тенденції, зростання популярності цього формату звукозапису, аналіз конкуренції та інших факторів, що впливають на ринкові умови.

Вивчається цільова аудиторія веб-застосунку, включаючи музичних ентузіастів, колекціонерів, діджеїв та інших зацікавлених осіб. Аналізуються їхні потреби, вподобання, звички, покупки та очікування від веб-застосунку. Встановлюються основні вимоги та виклики, що виникають при продажу вінілових платівок через веб-магазин, включаючи управління запасами, доставку. Аналізуються існуючі веб-магазини, які спеціалізуються на продажу вінілових платівок. Вивчається їхній асортимент, цінова політика, пропозиції та особливості обслуговування клієнтів. Це дозволяє виявити конкурентні переваги та можливості для вдосконалення веб-застосунку, який буде розроблятися в рамках кваліфікаційної роботи. Аналізуються важливі правові аспекти, пов'язані з продажем товарів онлайн, включаючи законодавство про захист споживачів, авторські права, захист даних та інші вимоги, які необхідно враховувати при розробці та експлуатації веб-магазину.

На основі проведеного аналізу предметної області визначається проблема, яку необхідно вирішити через розробку веб-застосунку з продажу вінілових платівок. Наприклад, це може бути недостатня кількість доступних веб-магазинів з

віниловими платівками, незручний процес пошуку та вибору продукту, складнощі у здійсненні покупок тощо.

В результаті здійснення змістовного аналізу предметної області веб-застосунку для продажу вінілових платівок були виявлені наступні важливі аспекти:

- Зростання популярності вінілових платівок на сучасному ринку звукозапису.
- Наявність конкуренції від інших веб-магазинів, що продають вінілові платівки.
- Потреба в удосконаленні процесу пошуку, вибору та замовлення продуктів.
- Правові аспекти, які необхідно враховувати при розробці та експлуатації веб-магазину.

Описані висновки стануть основою для подальшої розробки ПЗ, яке буде вирішувати виявлену проблему та вдосконалювати процес продажу вінілових платівок у веб-застосунку. Для цього будуть використані відповідні методи та технології, що дозволять створити ефективну та зручну платформу для покупців

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Вінілові платівки знову набувають популярності у сучасному світі. Ось кілька причин, що сприяють цьому:

- Аудіо якість: Багато людей цінують теплоту та багатство звуку, який відтворюється на вінілових платівках. Вони надають більш природний звуковий діапазон та аналоговий характер, який додає особливого шарму до музичного відтворення.
- Колекціонування: Вінілові платівки приваблюють колекціонерів та меломанів. Це унікальний спосіб збирати і слухати музику, а також додає елемент витонченості та індивідуальності до музичного досвіду.

– Арт-дизайн: Обкладинки вінілових платівок часто є справжніми шедеврами мистецтва. Вони можуть містити красиві ілюстрації, фотографії або витончений дизайн, що робить їх бажаними предметами для колекціонування.

– Ретро-тренд: В останні роки спостерігається загальний ретро-тренд, коли багато людей повертаються до старовинних технологій і речей, які мають відчуття ностальгії та автентичності. Вінілові платівки відображають цей тренд і стають популярними серед молоді та дорослих.

– Соціальний аспект: Слухання вінілових платівок може бути соціальним досвідом. Багато магазинів музики організовують вечірки, де люди збираються разом, обмінюються платівками та насолоджуються музикою.

Провівши детальний аналіз предметної області можна розглянути сайти аналоги. Нижче наведено приклад веб застосунку аналогу для продажу вінілу. Приклад наведено на рисунку 1.1.

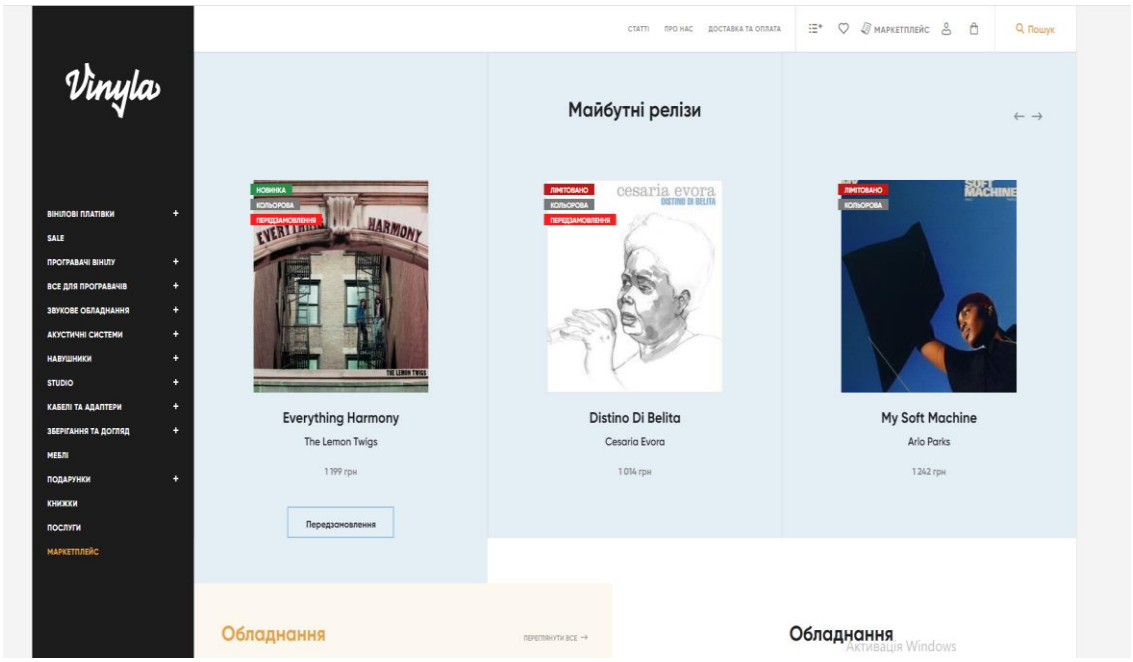


Рисунок 1.1 - інтерфейс сайту «Vinyla» - https://vinyla.com/

Серед функціоналу даного веб-застосунку можна виділити такі важливі елементи.

– Каталог продукції: Сайт може мати широкий вибір вінілових пластинок, які доступні для покупки. Каталог може бути організований за жанрами музики,

артистами, альбомами та іншими критеріями, що спрощують пошук конкретного запису.

– Пошук та фільтрація: Сайт може надавати функцію пошуку, що дозволяє користувачам швидко знайти вінілові пластинки за назвою артиста, назвою альбому або іншими ключовими словами. Також можуть бути наявні фільтри, які дозволяють користувачам обмежити пошук за ціною, жанром, станом пластинки тощо.

– Інформація про продукт: Кожна вінілова пластинка може мати окрему сторінку з докладною інформацією про неї, включаючи зображення обкладинки, трекліст, роки випуску, виробника тощо. Також можуть бути надані відгуки користувачів та рейтинги продуктів.

– Кошик покупок: Користувачі можуть додавати пластинки до свого кошика покупок, зберігати їх там та вирішувати, скільки екземплярів кожного запису вони бажають придбати. Після цього вони можуть перейти до оформлення замовлення та здійснити покупку.

– Користувацький обліковий запис: Сайт може надавати можливість створення особистого облікового запису, де користувачі зможуть зберігати свої контактні дані, переглядати історію замовлень, слідкувати за статусом доставки та зберігати свої уподобання.

– Оплата та доставка: Сайт може підтримувати різні способи оплати, такі як кредитні картки, електронні гаманці або платіжні системи. Також можуть бути надані різні варіанти доставки, що дозволяють користувачам вибрати найзручніший спосіб отримання своїх покупок.

Вцілому розглянувши даний застосунок можна зробити висновок що він задовільняє більшість потреб користувачів які хочуть придбати собі вінілові платівки різних жанрів та для різних цілей. На рисунку 1.2 можна ознайомитись зі сторінкою перегляду конкретної платівки для сайту аналогу.

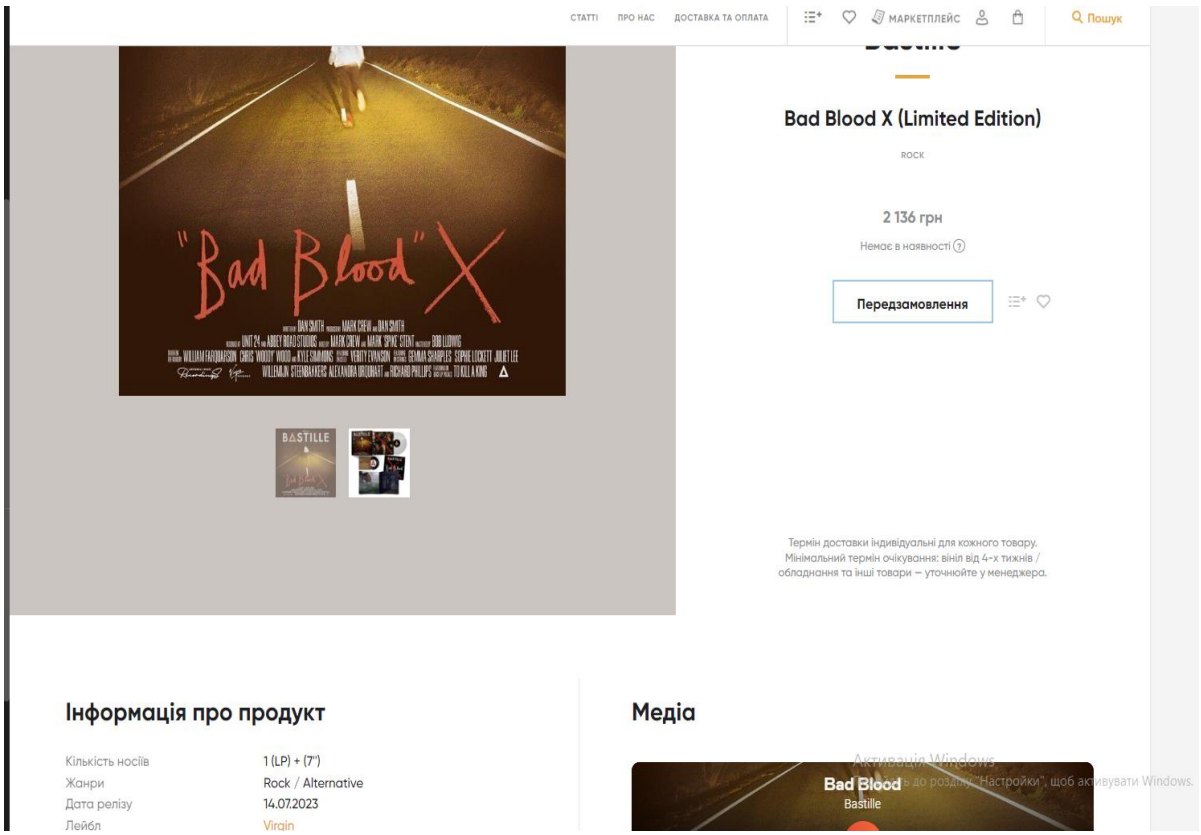


Рисунок 1.2 – сторінка перегляду конкретної платівки - https://vinyla.com/

Для більш детального аналізу проведемо розбір ще деяких сайтів аналогі. Аигляд головної сторінки сайту аналогу представлений на рисунку 1.3.

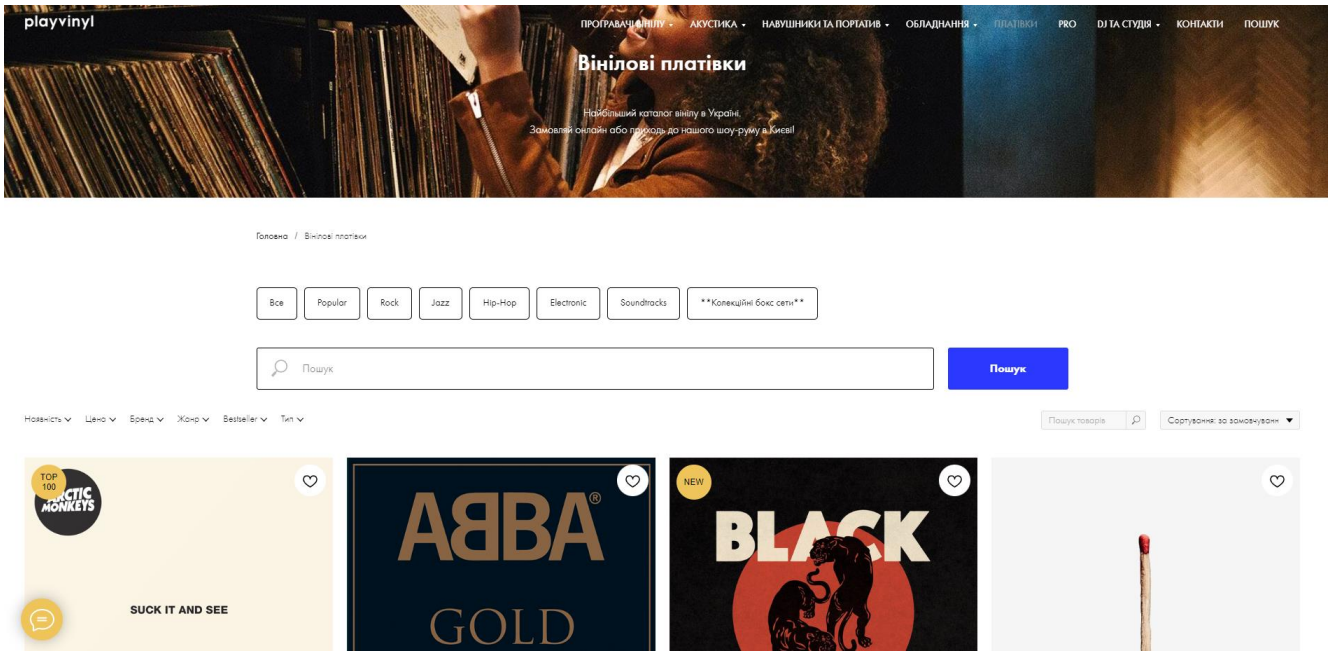


Рисунок 1.3 – Головна сторінка на сайті - https://playvinyl.com.ua/

Існує кілька чинників, які можуть зацікавити людей у покупці вінілових платівок на веб-сайтах:

– Широкий вибір: Веб-сайти пропонують великий асортимент вінілових платівок з різних жанрів та епох. Це дозволяє покупцям знайти саме ті альбоми, виконавців або рідкісні видання, які їх цікавлять.

– Рідкісність та ексклюзивність: Веб-сайти можуть пропонувати рідкісні або обмежені видання вінілових платівок, які можуть бути важкодоступними в традиційних магазинах. Це привертає колекціонерів та фанатів, які шукають унікальність та ексклюзивність.

– Аудіофільський досвід: Вінілові платівки надають унікальний аудіофільський досвід, звукову якість та атмосферу, яку багато людей цінують. Вони можуть насолоджуватися більш широким звуковим діапазоном, виразністю і теплотою звуку, які не завжди можуть бути досягнуті в інших форматах.

Сайти аналоги мають досить широкий асортимент товарів з ними можна ознайомитись на рисунку 1.4.

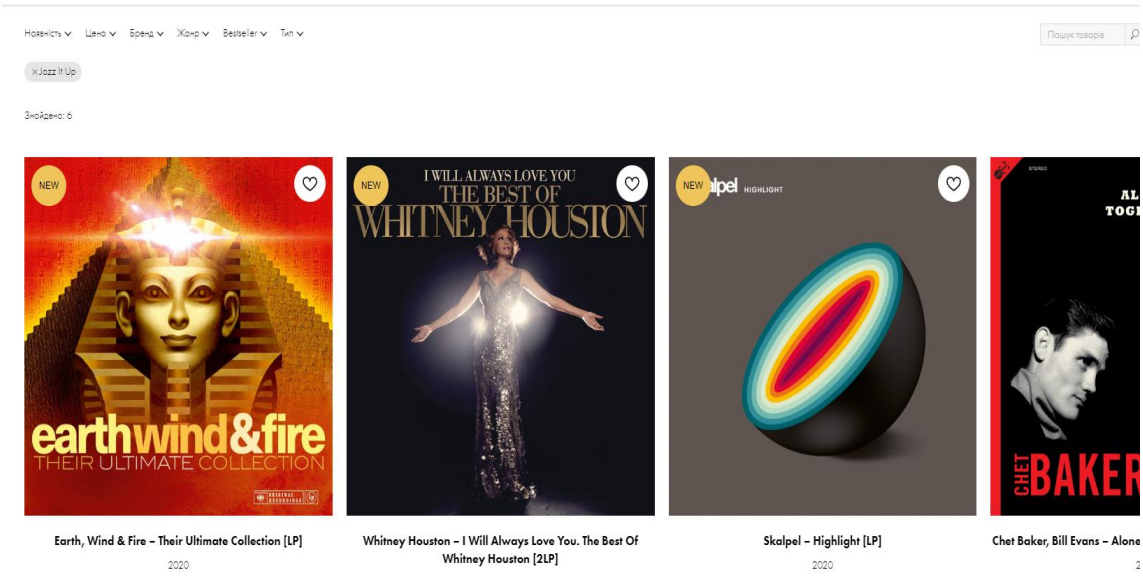


Рисунок 1.4 – Асортимент платівок на сайті - https://playvinyl.com.ua/

Серед цікавих рішень які надають застосунки аналоги можна виділити те що деякі з них займаються продажем не лише вінілу а й різних музичних пристроїв таких як програвачі вінілу, музичні колонки, акустика та інші прилади для прослуховування музики. Це надає цікаві можливості для масштабування бізнесу і

залучення нових клієнтів до веб-застосунку. Реалізацію на сайті аналозі можна побачити на рисунках 1.5 та 1.6.

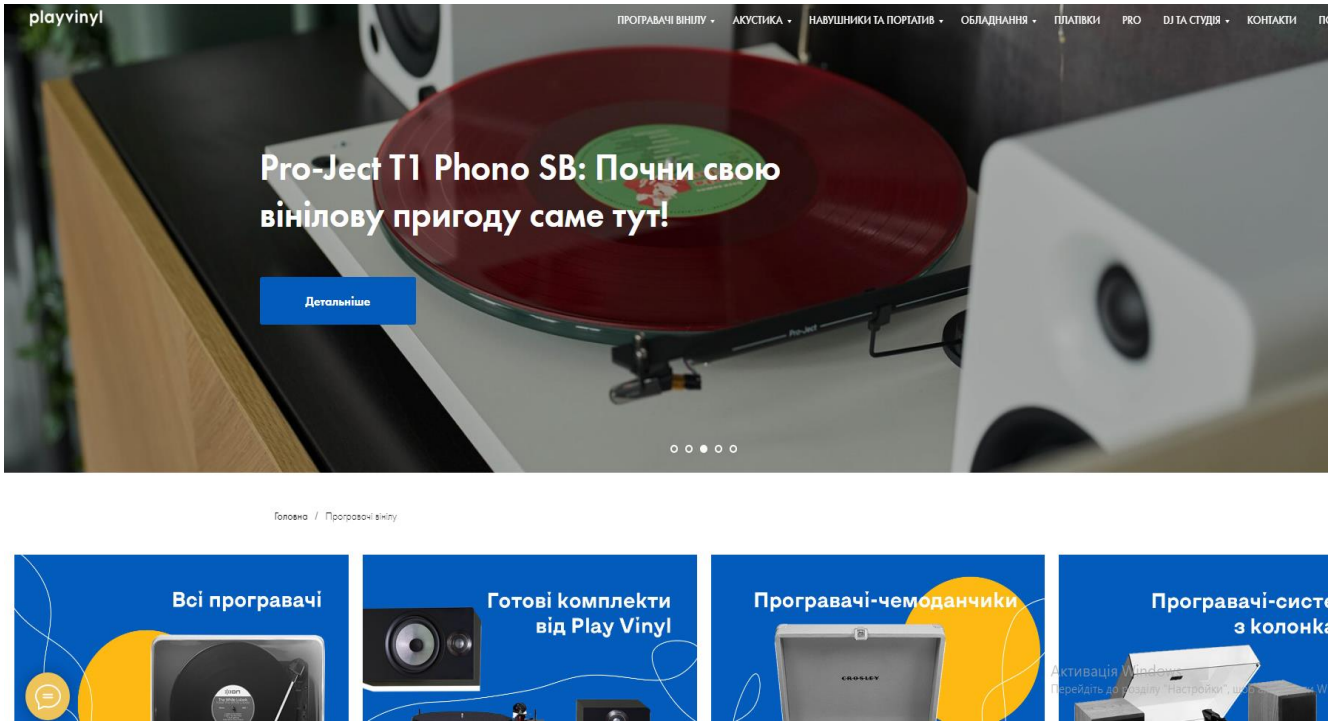


Рисунок 1.5 – Продаж вінілових програвачів - https://playvinyl.com.ua/

На цій сторінці ми бачимо сторінку продажу апаратури для прослуховування вінілу що може бути дуже цікавою пропозицією для меломанів.

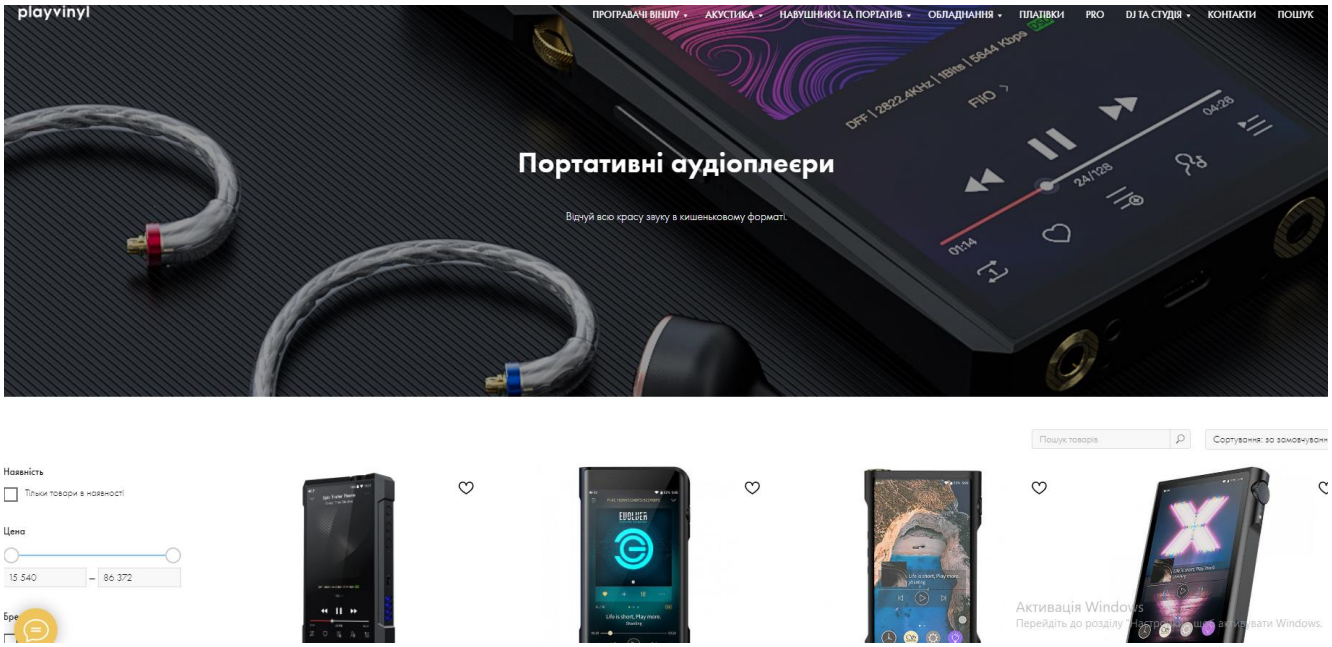


Рисунок 1.6 – Продаж музичних плеєрів - https://playvinyl.com.ua/

Виконавши детальний аналіз всіх схожих веб-застосунків, ми можемо порівняти їх за різними характеристиками. Основною характеристикою є зручність використання сайту, зокрема швидкість знаходження потрібного продукту у переліку. Надалі, значну увагу слід приділити оцінці інтерфейсу, як першого, з чим користувач зустрічається, а також кількості та ефективності реклами на сайті. Деяким важливим функціоналом є можливість пошуку та фільтрування, а також самостійний вибір платівки таонлайн-оплати для зручності користувачів.

Серед негативних факторів можна відзначити те, що багато рішень мають присутність реклами, яка може бути або пов'язана з вінілом, або ні. Крім того, більшість застосунків не мають можливості оплати, і користувач може лише надіслати свої контактні дані, після чого оператор зв'язується з ним по телефону або поштою. Таким чином, після проведення детального аналізу існуючих схожих рішень та їх характеристик, були визначені основні характеристики, функціональні та нефункціональні вимоги до розроблюваного проекту.

1.3 Аналіз вимог до програмного забезпечення та розробка технічного завдання

Аналіз вимог до програмного забезпечення є важливим кроком у розробці кваліфікаційної роботи на тему "Веб-магазин з продажу музичних платівок". Цей розділ присвячений визначенню функціональних і нефункціональних вимог до системи, а також створенню технічного завдання, що описує ці вимоги та вказує на технічні аспекти реалізації проекту.

Пошук та перегляд платівок: Система повинна мати можливість швидкого та зручного пошуку платівок за різними критеріями (назва, виконавець, жанр тощо). Користувачі повинні мати можливість переглядати деталі про платівку, включаючи обкладинку та трек-лист.

Додавання товарів у кошик: Користувачам має бути надана можливість додавати платівки до кошика для подальшого оформлення замовлення.

Оформлення замовлення: Система повинна мати функціонал оформлення замовлення, включаючи введення доставкової адреси, методи оплати та підтвердження замовлення.

Керування інвентарем: Система повинна вести облік наявності платівок та автоматично оновлювати їх кількість після кожного замовлення. У разі вичерпання запасів певної платівки, система повинна відображати її як недоступну для замовлення.

Визначення нефункціональних вимог:

Ефективність та продуктивність: Система повинна забезпечувати швидкий та ефективний пошук платівок, швидке оформлення замовлення та мінімальну затримку при завантаженні сторінок.

Безпека: Система повинна забезпечувати безпеку персональних даних користувачів, використовуючи захисні механізми, такі як шифрування паролів та захист від SQL-ін'єкцій.

Сумісність та масштабованість: Система повинна бути сумісною з різними веб-браузерами та мобільними пристроями. Крім того, вона повинна бути масштабованою, щоб забезпечити можливість росту та розширення магазину у майбутньому.

Ось перелік функціональних можливостей, які може надавати веб-застосунок для продажу музичних платівок:

- Перегляд каталогу: Користувач може переглядати каталог вінілових платівок, відсортований за жанрами, виконавцями або іншими категоріями.
- Пошук: Користувач може виконувати пошук за назвою, виконавцем або ключовими словами, щоб швидко знайти певну вінілову платівку.
- Детальна інформація: Користувач може переглядати детальну інформацію про кожну вінілову платівку, включаючи обкладинку, опис, рік випуску, трекліст та інші характеристики.
- Кошик покупок: Користувач може додавати вінілові платівки до свого кошика, де вони зберігаються до оформлення замовлення.

– Оформлення замовлення: Користувач може оформити замовлення на придбання вінілових платівок з кошика, вказавши доставкові дані та обираючи метод оплати.

– Обліковий запис користувача: Користувач може створити обліковий запис, де він зможе зберігати свої контактні дані, переглядати історію замовлень та змінювати налаштування свого профілю.

Таблиця 1.1 – Опис акторів розроблюваного ПЗ

Актор	Опис
Користувач	Основний актор, який відвідує веб-магазин, переглядає каталог вінілових платівок, шукає та купує пластинки. Може створювати обліковий запис та зберігати свої уподобання.
Адміністратор	Відповідає за управління веб-магазином. Може додавати, видаляти або редагувати продукти у каталозі, керувати замовленнями, відгуками користувачів та іншими аспектами сайту.

Діаграма варіантів використання складається з акторів, прецедентів та зв'язків між ними. Актори - це зовнішні сутності (користувачі, системи, пристрої), які взаємодіють з системою. Прецеденти - це конкретні функції або операції, які виконуються системою від імені акторів.

Діаграма варіантів використання дозволяє чітко визначити функціональність системи та взаємодію з її користувачами. Вона є важливим інструментом для зрозуміння потреб та вимог користувачів і використовується для подальшого аналізу, проектування та розробки системи. Розроблену діаграму варіантів використання можна побачити нижче на рисунку 1.7.

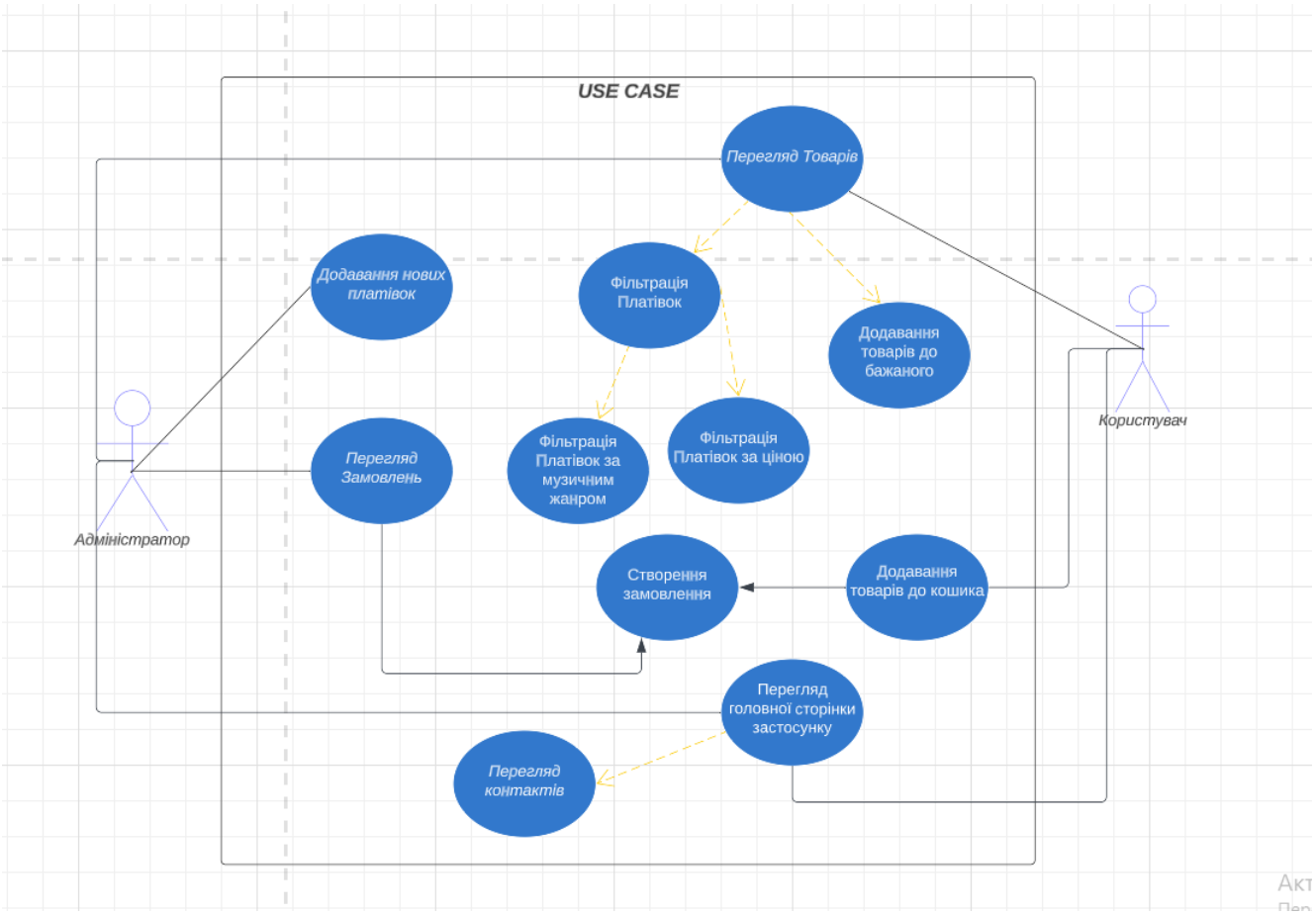


Рис 1.7 – Діаграма варіантів використання.

Після проведення аналізу вимог до програмного продукту, було створено технічне завдання, яке можна знайти у додатку А. Це технічне завдання послужить основою для подальшої розробки застосунку.

1.4 Висновки. Постановка задачі

У даному розділі були розглянуті особливості предметної області, виявлені проблеми, які існують в цій області, а також обговорені шляхи їх вирішення. Були проаналізовані наявні програмні засоби, виявлені їх переваги та недоліки. Був складений список функціональних та нефункціональних вимог до розроблюваного проекту.

Виконавши аналіз усіх вимог та предметної області кваліфікаційної роботи ми можемо створити список завдань які будуть стояти перед нами під час виконання кваліфікаційної роботи.

Ось список задач для кваліфікаційної роботи на тему "Веб-застосунок для продажу музичних платівок":

– Аналіз вимог та формулювання функціональних та нефункціональних вимог до системи.

– Проектування бази даних для зберігання інформації про платівки, використовуючи MongoDB.

– Розробка серверної частини застосунку з використанням популярного веб-фреймворку, наприклад, Node.js з Express.

– Реалізація клієнтської частини застосунку за допомогою Angular.

– Розробка функціонального застосунка який після розгортання зможе задовільняти усі потреби як власника так і користувачів.

Висновки:

Провівши детальний аналіз предметної області зі сторення веб-застосунку з продажу музичних платівок можна прийти до висновку що ця сфера має велику перспективу розвитку адже все більше і більше музичних виконавців випускають свої пісні на вінілових платівках. Ринок має великі перспективи для подальшого розвитку. Проведення анлізу існуючих рішень дозволило виявити переваги та недоліки сайтів аналогів. З проведеного аналізу можна прийти до висновків що попит на вініл та різного типу аудіоапаратуру є високим і ніша електронної комерції в цьому напрямку є перспективною і доступною для входу навіть для невеликих бізнесів. Була проведена розробка технічного завдання, що дозволило визначити основний функціонал для застосунку та розробити функціонал який в майбутньому може бути використаний для його розширення. В загальному можна прийти до висновку, що Розробка веб-застосунку для продажу музичних платівок має потенціал бути успішним проектом, оскільки ця галузь має своїх прихильників і клієнтів.

2. Проектування Програмного забезпечення

2.1 Аналіз та вибір архітектури веб-додатка

Клієнт-серверна архітектура є популярним підходом для розробки веб-додатків, включаючи застосунки для продажу музичних платівок. В цій архітектурі взаємодія між користувачем (клієнтом) і додатком відбувається через комунікацію між клієнтською та серверною частинами.

Клієнтська частина, яка зазвичай представлена веб-браузером, відповідає за відображення інтерфейсу користувача. Користувачі взаємодіють з додатком, виконуючи різні дії, такі як перегляд каталогу платівок, додавання товарів до кошика, оформлення замовлення та інші. Клієнтська частина передає запити на сервер для обробки та отримує відповіді, які вона відображає для користувача.

Серверна частина додатку містить бізнес-логіку, базу даних та інші компоненти, необхідні для обробки запитів від клієнта. Вона отримує запити від клієнтської частини, обробляє їх, взаємодіє з базою даних та іншими сервісами, які можуть бути потрібні для виконання запитів. Після обробки запиту серверна частина повертає відповідь клієнту, який відображає її користувачеві.

Клієнт-серверна архітектура дозволяє відокремити логіку відображення та бізнес-логіку, що забезпечує модульність та масштабованість системи. Наприклад, зміна інтерфейсу користувача може бути виконана без впливу на серверну частину. Крім того, цей підхід дозволяє використовувати різні типи клієнтів (веб-браузери, мобільні додатки) для взаємодії з серверною частиною.

В контексті веб-додатку для продажу музичних платівок, клієнт-серверна архітектура дозволить розробити функціональність, яка дозволить користувачам швидко переглядати каталог платівок, здійснювати покупки, керувати замовленнями та виконувати інші операції, взаємодіючи з сервером через зручний та інтуїтивно зрозумілий інтерфейс.

У клієнт-серверній архітектурі, клієнт і сервер взаємодіють за допомогою протоколу HTTP.

Протокол HTTP (Hypertext Transfer Protocol) є протоколом передачі гіпертекстових документів через мережу Інтернет. Він використовується для комунікації між клієнтом (клієнтською програмою або браузером) і сервером (веб-сервером).

HTTP базується на моделі "клієнт-сервер", де клієнт ініціює запити, а сервер надає відповіді на ці запити. Запити і відповіді між клієнтом і сервером передаються у вигляді повідомлень, що містять заголовки та необов'язковий тіловий контент.

Структура HTTP-запиту складається з:

- Заголовка методу (GET, POST, PUT, DELETE тощо), який вказує тип операції, яку слід виконати на ресурсі сервера.
- URL-адреси ресурсу, до якого відбувається запит.
- Заголовків запиту, які містять додаткову інформацію про запит (наприклад, тип контенту або параметри авторизації).

Структура HTTP-відповіді включає:

- Код статусу, що показує результат виконання запиту (наприклад, 200 OK для успішного запиту, 404 Not Found для незнайденого ресурсу і т.д.).
- Заголовки відповіді, які містять інформацію про відповідь сервера (наприклад, тип контенту або дата модифікації).
- Тіло відповіді, яке може містити додаткові дані або ресурси, пов'язані з запитом.

HTTP є безстіковим протоколом, що означає, що кожний запит клієнта і відповідь сервера розглядаються як окремі незалежні операції. Комунікація між клієнтом і сервером здійснюється за принципом "запит-відповідь", де клієнт відправляє запит, а сервер надсилає відповідь на цей запит.

Протокол HTTP є основою для багатьох веб-технологій, таких як HTML, CSS, JavaScript та RESTful API. Він дозволяє клієнтам отримувати веб-сторінки, передавати дані на сервер, взаємодіяти з веб-сервісами та здійснювати різні дії в Інтернеті.

Клієнтська частина:

– Фронтенд додаток будується з використанням фреймворка Angular. Angular забезпечує розробку потужних та динамічних веб-інтерфейсів, які можуть включати функціональність перегляду каталогу платівок, кошика покупок, оформлення замовлення та інші операції, необхідні для користувачів.

– Клієнтський додаток взаємодіє з серверною частиною через HTTP-запити. Він відправляє запити на сервер для отримання даних про платівки, додавання їх до кошика, оформлення замовлення та інші операції. Після отримання відповіді від сервера, клієнтська частина оновлює відображення для користувача.

Серверна частина:

– Бекенд додаток будується з використанням платформи Node.js. Node.js дозволяє розробляти серверні додатки на JavaScript і забезпечує швидке та ефективне виконання запитів.

– База даних MongoDB використовується для зберігання даних про платівки, користувачів, замовлення та іншу інформацію, необхідну для функціонування магазину. MongoDB є NoSQL базою даних, яка забезпечує гнучкість та швидкий доступ до даних.

– Серверна частина отримує HTTP-запити від клієнтської частини, обробляє їх та взаємодіє з базою даних MongoDB для отримання необхідної інформації. Вона відповідає на запити клієнтів, надсилаючи їм відповіді, які містять необхідні дані або статуси операцій.

– Серверна частина також відповідає за обробку платежів, авторизацію користувачів та забезпечення безпеки додатку.

Загалом, клієнт-серверна архітектура для веб-магазину з продажу музичних платівок, розроблена з використанням Angular, Node.js та Mongo DB, дозволяє створити потужний та функціональний застосунок, який надає користувачам зручний інтерфейс для перегляду та придбання платівок, а серверна частина забезпечує ефективну обробку запитів та доступ до необхідної інформації.

2.2 Опис структури даних та моделі бази даних

Для веб-магазину з продажу музичних платівок, побудованого на стеку технологій Angular, Node.js та MongoDB, можна використовувати наступну структуру даних та модель бази даних:

Структура даних:

- Користувачі: зберігання інформації про зареєстрованих користувачів, таку як ім'я, електронна пошта, адреса доставки, історія замовлень тощо.
- Платівки: зберігання деталей про кожну музичну платівку, такі як назва, виконавець, жанр, рік видання, ціна, наявність у складі, зображення обкладинки тощо.
- Кошик: зберігання інформації про платівки, які користувач додав до кошика, включаючи їх кількість та загальну вартість.
- Замовлення: зберігання деталей про кожне оформлене замовлення, включаючи вибрані платівки, користувача, адресу доставки, статус оплати тощо.

Сучасні веб-магазини зазвичай використовують бази даних для зберігання своїх даних, таких як інформація про товари, користувачів, замовлення та інші важливі деталі. Бази даних забезпечують структуроване та ефективне зберігання даних, а також дозволяють легко виконувати операції пошуку, оновлення та видалення даних.

Ось кілька сучасних видів баз даних, які часто використовуються в веб-розробці:

Реляційні бази даних (Relational databases - RDBMS):

- Найпопулярнішими реляційними базами даних є MySQL, PostgreSQL, Oracle, Microsoft SQL Server і SQLite.
- Вони використовують табличну структуру для зберігання даних та залежностей між ними за допомогою ключів.
- Реляційні бази даних дозволяють виконувати складні запити, забезпечують цілісність даних та підтримують транзакції.

NoSQL бази даних:

– NoSQL (Not Only SQL) бази даних, такі як MongoDB, Cassandra, CouchDB, розроблені для зберігання та обробки великого обсягу нереляційних даних.

– Вони забезпечують гнучку схему даних, горизонтальне масштабування та високу продуктивність.

– NoSQL бази даних добре підходять для веб-магазинів зі збільшуваною кількістю даних та потребою в швидкому доступі до них.

Ключ-значення (Key-Value) бази даних:

– Key-Value бази даних, такі як Redis, Riak, Amazon DynamoDB, зберігають дані у вигляді ключів та відповідних значень.

– Ці бази даних добре підходять для швидкого зберігання і отримання даних, особливо для кешування та роботи з сесіями.

– Документ-орієнтовані бази даних:

– Документ-орієнтовані бази даних, такі як MongoDB, Couchbase, Elasticsearch, зберігають дані у вигляді JSON-подібних документів.

– Ці бази даних забезпечують гнучкість у зберіганні та операціях з даними, підтримують індексацію та пошук по текстовому змісту.

Колоночно-орієнтовані бази даних:

– Колоночно-орієнтовані бази даних, такі як Apache Cassandra, HBase, Vertica, зберігають дані у вигляді колонок, а не рядків.

– Ці бази даних підходять для ситуацій, де потрібен швидкий доступ до певних колонок даних або горизонтальне масштабування.

Кожен вид бази даних має свої переваги і відповідає різним потребам проекту. Вибір конкретної бази даних залежить від вимог проекту, обсягу даних, швидкодії та інших факторів.

СКБД (Системи Керування Базами Даних) - це програмні продукти, які дозволяють створювати, зберігати, оновлювати та керувати базами даних. Основним завданням СКБД є забезпечення ефективного та безпечного доступу до даних, забезпечення цілісності даних та виконання операцій з базою даних.

Ось кілька популярних СКБД та їх особливості:

MySQL:

- Відкрита реляційна СКБД, яка надає надійність та високу продуктивність.
- Має широку підтримку та активну спільноту користувачів.
- Підтримує розподілені транзакції, реплікацію та кластеризацію.

PostgreSQL:

- Розширена реляційна СКБД з акцентом на розширюваність та стандартизованість.
- Підтримує розширені функції, такі як JSON-підтримка, географічні запити та повнотекстовий пошук.
- Має потужну систему реплікації та можливості паралельного виконання запитів.

Oracle Database:

- Комерційна реляційна СКБД, яка використовується в багатьох підприємствах.
- Має велику функціональність, підтримує розподілені транзакції, кластеризацію та резервне копіювання даних.
- Забезпечує високу надійність та швидкодію.

MongoDB:

- Документ-орієнтована СКБД, яка зберігає дані у вигляді JSON-подібних документів.
- Має гнучку схему даних та високу продуктивність.
- Підтримує горизонтальне масштабування та реплікацію даних.

Redis:

- Ключ-значення СКБД, яка зберігає дані у вигляді пар ключ-значення.
- Дуже швидка та ефективна для кешування та швидкого доступу до даних.
- Підтримує багатофункціональність, таку як сортування, публікація-підписка та геопросторові операції.

Після вивчення переваг і недоліків різних типів баз даних, було вирішено, що кваліфікаційна робота буде розроблятися з використанням нереляційної бази даних MongoDB. Це дозволить забезпечити швидкий доступ до даних, незалежно від їх обсягу, і усуне потребу детального проектування таблиць та встановлення зв'язків між ними, що в свою чергу зменшить терміни розробки та проектування.

Засновуючись на раніше створеній діаграмі варіантів використання (див. рисунок 1.11), було визначено, які дані будуть зберігатись у базі даних.

Модель бази даних:

- MongoDB використовує документ-орієнтовану модель даних, де інформація зберігається у вигляді документів у форматі JSON. Кожен документ містить ключі та значення, що представляють різні атрибути та їх значення для конкретного об'єкта.
- Для моделювання структури даних можна використовувати колекції (collections), які відповідають таблицям у реляційних базах даних. Кожна колекція може містити набір документів, які представляють записи про певні об'єкти.
- Наприклад, можна мати колекцію "users" для зберігання документів, які представляють користувачів, колекцію "records" для зберігання документів про музичні платівки, колекцію "carts" для зберігання документів про кошики користувачів та колекцію "orders" для зберігання документів про замовлення.
- Кожен документ в колекції має унікальний ідентифікатор, який дозволяє однозначно ідентифікувати запис.

Така структура даних та модель бази даних дозволять зберігати та організовувати інформацію про платівки, користувачів, кошики та замовлення в ефективний спосіб, що підходить для веб-магазину з продажу музичних платівок.

Колекція "users" (користувачі):

- _id: ObjectId (унікальний ідентифікатор користувача)
- name: string (ім'я користувача)
- email: string (електронна пошта користувача)
- address: string (адреса доставки користувача)

- Role Id : string (Ідентифікатор ролі користувачі)

Колекція "record" (платівки):

- _id: ObjectId (унікальний ідентифікатор платівки)
- Name: string (назва платівки)
- artist: string (виконавець)
- genre: string (жанр)
- year: number (рік видання)
- price: number (ціна)
- stock: number (наявна кількість в запасі)
- image: string (URL-адреса зображення обкладинки)
- genre: string (Музичний жанр)

Колекція "carts" (кошики користувачів):

- _id: ObjectId (унікальний ідентифікатор кошика)
- userId: ObjectId (ідентифікатор користувача, пов'язаний з кошиком)
- recordId: ObjectId(ідентифікатор платівки, пов'язаний з кошиком)
- items: [{ vinylId: ObjectId (ідентифікатор платівки) quantity: number (кількість) }] (масив предметів у кошику)

Колекція "Order History" (замовлення):

- _id: ObjectId (унікальний ідентифікатор замовлення)
- userId: ObjectId (ідентифікатор користувача, пов'язаний з замовленням)
- RecordId: ObjectId (ідентифікатор Платівки, пов'язаний з замовленням)
- Price: number (Ціна замовлення)
- Quontity: number(кількість товарів)

2.3 Проектування серверної частини веб-додатка

Монолітна архітектура - це традиційний підхід до розробки програмного забезпечення, де весь застосунок розгортається як єдиний, неділенний блок, в якому логіка, функціональність та компоненти програми знаходяться разом.

Основні характеристики монолітної архітектури:

– Єдиний виконуваний файл: Весь код додатку, включаючи логіку, базу даних і інтерфейс користувача, компілюється та розгортається як один виконуваний файл.

– Централізоване керування: Всі компоненти додатку, такі як база даних, інтерфейси користувача, обробники запитів, розташовані в межах одного процесу.

– Прямі виклики: Взаємодія між компонентами здійснюється шляхом прямих викликів функцій або методів, зазвичай в межах пам'яті.

Переваги монолітної архітектури:

– Простота розробки: Розробка та тестування монолітних додатків є відносно простими, оскільки весь код знаходиться в одному місці.

– Легше масштабування: Монолітні додатки можуть бути легко масштабовані вертикально, додаючи більш потужне обладнання або горизонтально, запускаючи кілька копій додатку.

Недоліки монолітної архітектури:

– Велика складність: Зі зростанням розміру додатку, розробка, тестування та розгортання можуть стати складними завданнями.

– Залежність компонентів: Зміни в одному компоненті можуть вплинути на решту додатку, що може ускладнити розвиток і підтримку.

– Обмежена масштабованість: Монолітні додатки мають обмежену горизонтальну масштабованість, оскільки весь додаток має бути розгорнутий разом.

– Виділення ресурсів: Розгортання та масштабування вимагає виділення ресурсів для всього додатку, навіть якщо лише деякі його частини активно використовуються.

У зв'язку з розвитком мікросервісної архітектури, монолітна архітектура часто замінюється або переходить до більш гнучких та розділенних моделей розробки програмного забезпечення. Зі схемою монолоітної архітектури можна ознайомитись на рисунку 2.1.

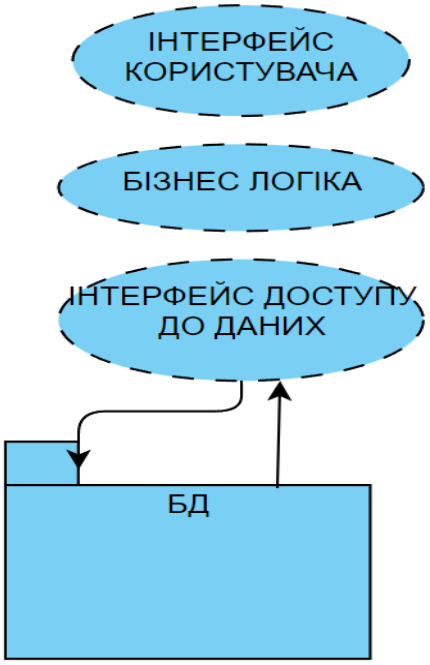


Рис 2.1 – Монолітна архітектура

Мікросервісна архітектура - це підхід до розробки програмного забезпечення, в якому застосунок розбивається на невеликі та автономні сервіси, які взаємодіють між собою через мережу.

Основні характеристики мікросервісної архітектури:

– Розподілена структура: Застосунок складається з набору незалежних сервісів, які можуть бути розгорнуті та масштабовані окремо.

– Автономність: Кожен сервіс є автономним та може бути розроблений, випробуваний, розгорнутий та масштабований окремо від інших сервісів.

– Взаємодія через API: Сервіси взаємодіють між собою за допомогою стандартизованих API, таких як REST або мікрослужби.

– Орієнтованість на бізнес-функції: Кожен сервіс фокусується на конкретній бізнес-функції або функціональності.

– Легке масштабування: Кожен сервіс може бути масштабований окремо в залежності від навантаження, що сприяє ефективному використанню ресурсів.

Переваги мікросервісної архітектури:

– Гнучкість: Кожен сервіс може бути розроблений і вдосконалений незалежно, що сприяє швидкому розвитку та впровадженню нових функцій.

– Масштабованість: Можливість масштабування окремих сервісів забезпечує гнучке керування навантаженням та ефективне використання ресурсів.

– Легша підтримка: Виправлення помилок та внесення змін стають простішими, оскільки вони обмежені до конкретних сервісів, а не до всього додатку.

Недоліки мікросервісної архітектури:

– Складність управління: Розробка та управління багатьма сервісами можуть бути складними, оскільки вони вимагають моніторингу, координації та узгодження.

– Взаємодія через мережу: Комунікація між сервісами через мережу може призводити до затримок та проблем зі швидкодією.

– Комплексність тестування: Випробування та налагодження мікросервісного додатку вимагає комплексного тестування кожного сервісу окремо та взаємодії між ними.

Мікросервісна архітектура є популярним підходом до розробки додатків у сучасному програмуванні, особливо у великих та складних проектах, де вимагається гнучкість, масштабованість та швидкість розробки.

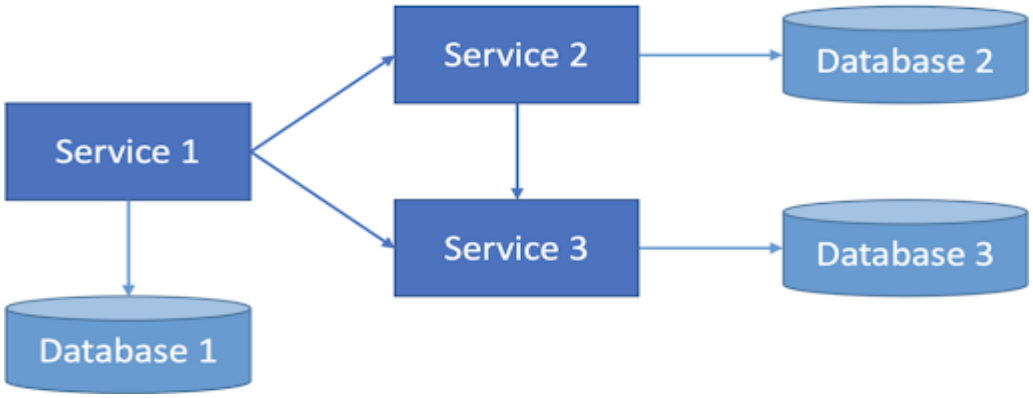


Рисунок 2.2 –мікросервісна архітектура

Отже, можна прийти до висновку, що монолітна архітектура є доцільним вибором для розробки простих та невеликих програмних систем, які не вимагають значних ресурсів. З іншого боку, мікросервісна архітектура підходить для складних систем з великою кількістю незалежних модулів. Таким чином, для даної кваліфікаційної роботи була обрана монолітна архітектура, враховуючи середню

складність проекту та обмежений час розробки. Модулі цієї системи мають високий рівень взаємодії, тому розбити їх на окремі сервіси було б недоцільно та зайняло багато часу на розгортання та тестування.

2.4 Проектування клієнтської частини веб-додатка

Для створення дизайну веб-застосунку було вирішено розглянути існуючі рішення для дизайнів веб-застосунків та обрати один із них для основи дизайну.

Прототипізація є важливим етапом у розробці програмного забезпечення (ПЗ). Вона включає створення прототипу або пробного варіанту ПЗ з метою дослідження, валідації та отримання зворотного зв'язку від користувачів, замовників або зацікавлених сторін.

Прототип є приблизною та функціональною моделлю кінцевого продукту, яка дозволяє перевірити концепції, взаємодію, функціональність та інтерфейс перед реалізацією остаточного рішення.

Почнемо зі створення верхньої частини сторінки “хедеру”, хедер повинен містити у собі логотип веб-застосунку який при натисканні буде вести на головну сторінку, кнопка перегляду всіх платівок, кнопка кошика та можливість реєстрації (рисунок 2.3).

Рисунок 2.3 – Хедер веб сайту для не авторизованого користувача

Коли користувач скористався кнопкою авторизації і зайшов на веб сайти вигляд хедеру зміниться і замість кнопки входу з’явиться кнопка виходу(рисунок 2.4).

Рисунок 2.4 – Хедер веб сайту для авторизованого користувача

Для користувача адміністратора зовнішній вигляд хедера буде мати майже такий самий вигляд за виключенням того що зявиться кнопка завантаження нових платівок (рисунок 2.5)

Рисунок 2.5 – Хедер веб сайту для адміністратора

Далі почнемо з розробки головної сторінки веб-застосунку. Головна сторінка відповідальна за репрезентацію додатку перед користувачами. На головній сторінці розміщені такі елементи:

- Хедер
- Меню швидкої навігації
- Блок з Назвою веб-застосунку

Макет головної сторінки зображено на рисунку 2.6.

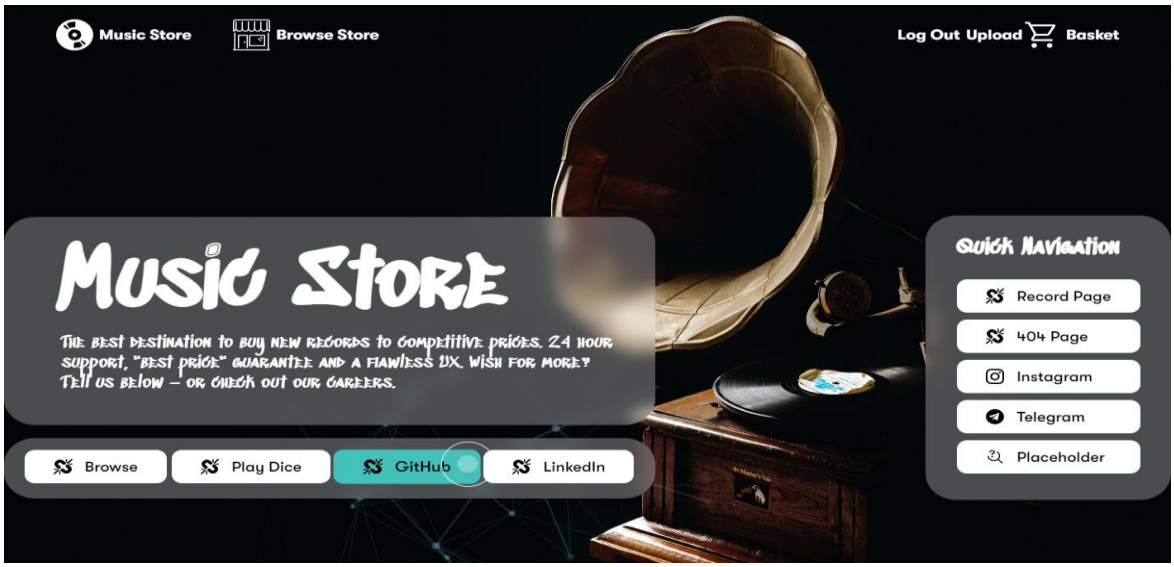


Рисунок 2.6 – Головна сторінка застосунку

Також так як головна сторінка має зацікавити користувача ми додамо до проекту canvas анімацію та змінимо курсор користувача. Зміни побачимо на рисунку 2.7.

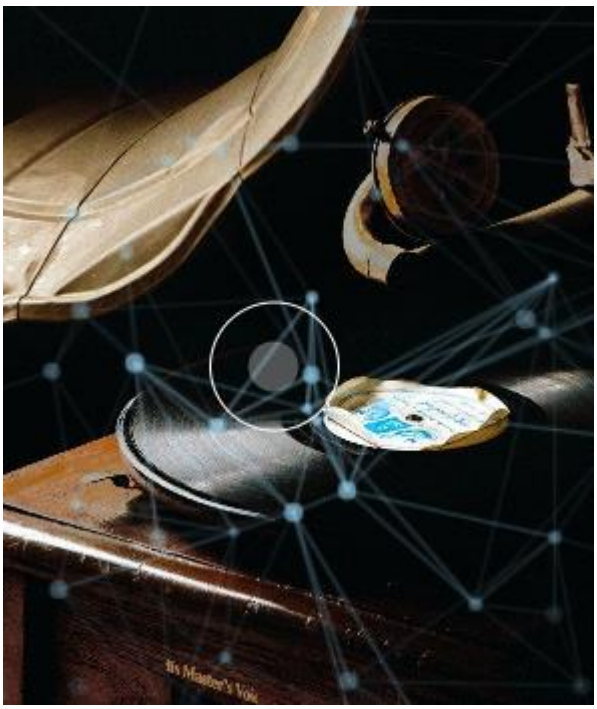


Рисунок 2.7 – Canvas та анімований курсор застосунку

Canvas - це веб-технологія HTML5, яка надає можливість рендерити графічні елементи та анімацію безпосередньо на сторінці веб-сайту. Вона представляє собою прямокутну область на веб-сторінці, яку можна програмно керувати за допомогою JavaScript.

За допомогою Canvas можна створювати різноманітні графічні ефекти, малювати лінії, фігури, текст, розміщувати зображення та виконувати анімацію. Все це досягається шляхом маніпулювання пікселями на внутрішньому "канвасі" (полотні), яке можна відображати на екрані.

Canvas надає потужний і гнучкий інтерфейс для роботи з графікою в реальному часі. Вона здатна працювати зі швидкістю 60 кадрів на секунду та надає можливості для обробки подій, таких як кліки мишею або натискання клавіш.

Наступним кроком проектування буде створення макету перегляду всіх доступних платівок, на цій сторінці користувач зможе ознайомитись з усіма доступними платівками. Тут буде розміщений блок фільтрації доступних платівок за жанром музики можливість фільтрування та скидання цих фільтрів, а також

будуть розміщені карточки платівок клацнувши на які можна буде ознайомитись з більш детальною інформацією (рисунок 2.8).

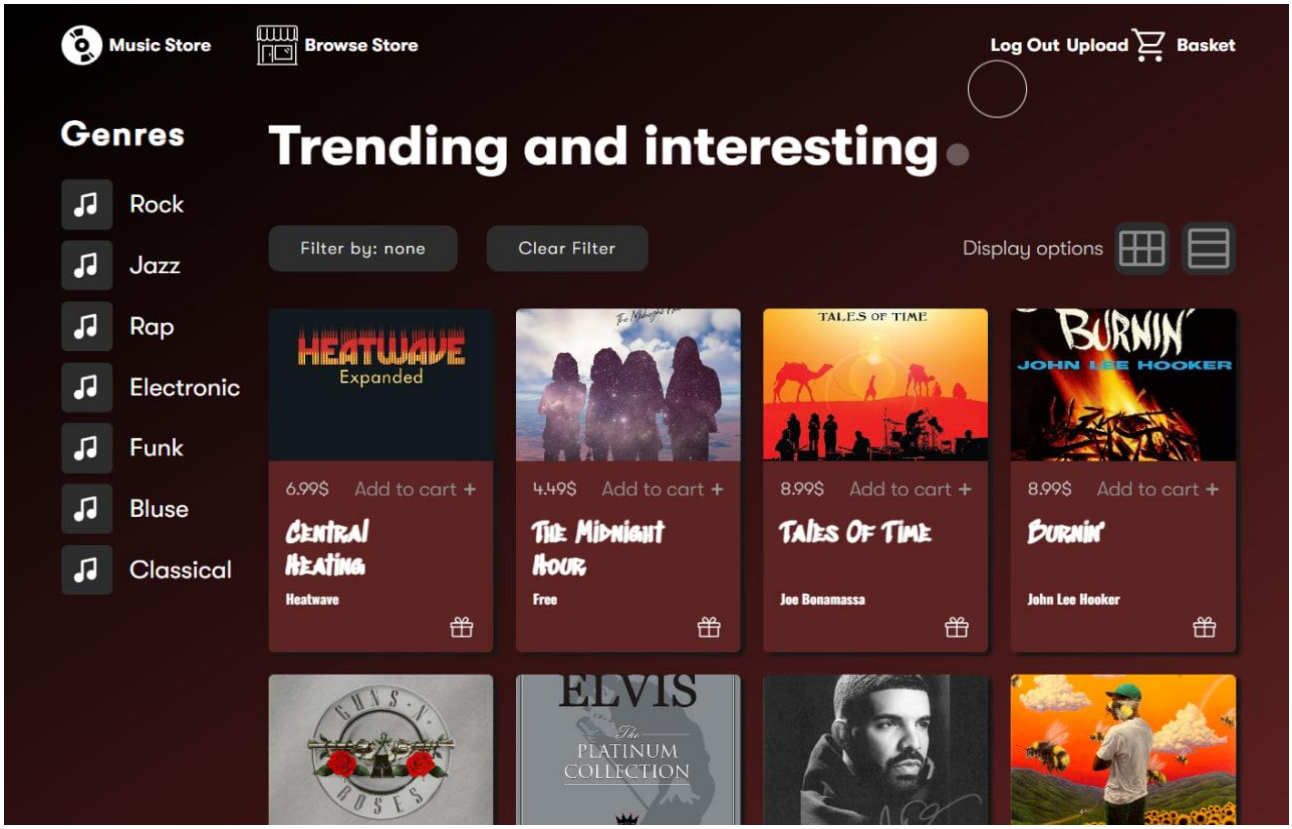


Рисунок 2.8 – Сторінка перегляду платівок

Ця сторінка дозволяє користувачу ознайомитись з усім асортиментом платівок які є на сайті а також відортувати їх за своїми вподобаннями.

Наступним етапом буде розробка макету для докладного огляду інформації про Платівку. У лівій частині сторінки буде розташований слайдер, що дозволить переглядати фотографії. У правій частині буде розміщена таблиця з детальним описом платівки, її ціною. Внизу сторінки буде розташована кнопка, яка дозволить додати платівку у кошик. Макет сторінки з детальною інформацією можна побачити на рисунку 2.9.

Рисунок 2.8 – Детальна інформація про платівку

Далі створимо макет для сторінки кошика де користувач зможе переглянути додані платівки та видалити їх з кошика. Детальніше з макетом цієї сторінки можна ознайомитись на рисунку 2.9.

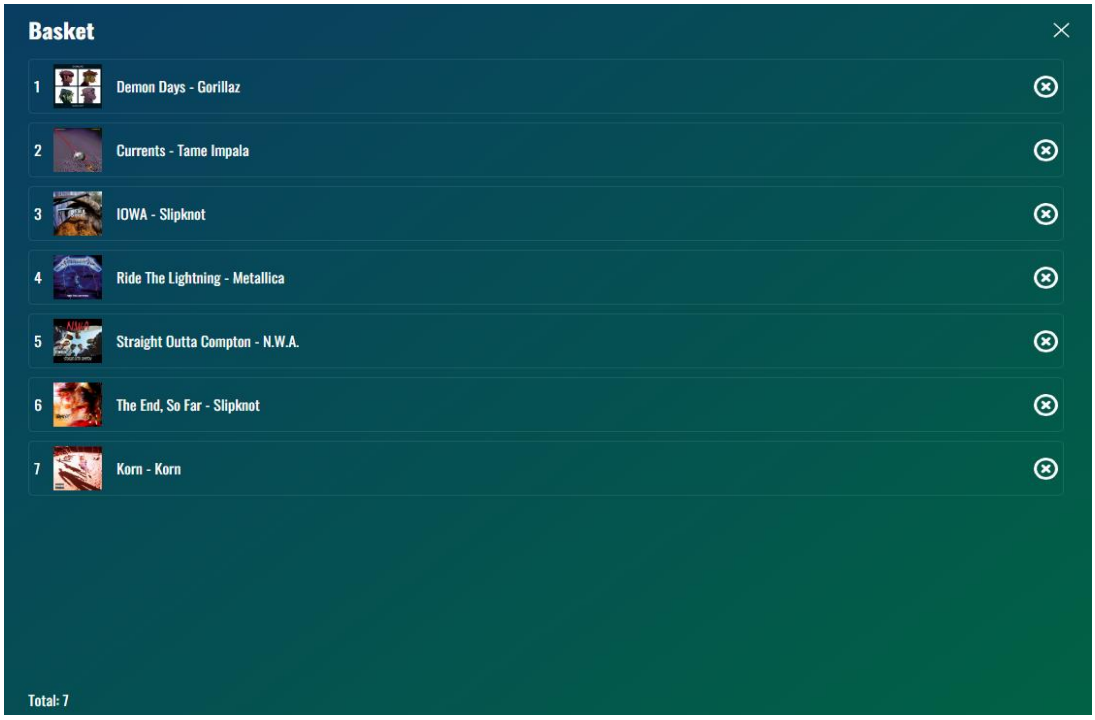


Рисунок 2.9 – Кошик з доданими платівками

2.5 Аналіз та вибір технологій і методів реалізації веб-додатка

Аналізуючи наявне програмно-технічне забезпечення для веб-застосунку з продажу музичних платівок у предметній області, було виявлено використання такого стеку технологій:

- Angular: Цей фреймворк використовується для побудови користувацького інтерфейсу магазину з продажу латівок. Angular надає потужні інструменти для розробки SPA (односторінкових додатків), дозволяючи створювати динамічні та ефективні веб-додатки.
- Ngrx: Це бібліотека стану, яка базується на концепції Redux і використовується разом з Angular для керування станом додатку. Вона забезпечує централізоване управління станом додатку і спрощує роботу зі змінними стану.
- SCSS: Це препроцесор CSS, який дозволяє використовувати змінні, вкладені стилі, міксіни та інші функції, що полегшують розробку та підтримку стилів магазину. SCSS допомагає зменшити дублювання коду і зробити CSS більш організованим та зрозумілим.
- Karma: Це інструмент для запуску автоматизованих тестів JavaScript у різних браузерах. Використовуючи Karma, можна переконатися, що функціональність магазину з продажу латівок працює коректно на різних платформах і в різних середовищах.
- Node.js є відкритим середовищем виконання JavaScript, яке дозволяє розробникам виконувати JavaScript-код на стороні сервера. За допомогою Node.js ви можете створювати швидкі та масштабовані веб-додатки, мережеві сервіси та інші застосунки.

Використання цього стеку технологій дозволяє побудувати сучасний, ефективний та масштабований магазин з продажу платівок зі зручним користувацьким інтерфейсом та добре організованим станом додатку.

Нижче наведена порівняльна таблиця між Angular та React:

Таблиця 2.1 – Порівняння можливих рішень

Критерій	**Angular**	**React**
Тип	Фреймворк	бібліотека
Мова програмування	TS	JS
Розмір	Великий	Менший
Швидкодія	Швидше	Нижче
Компонентна структура	Шаблон-компонент	JSX-компонент
Компонентна архітектура	Інтегрована (вбудовані модулі)	Зовнішня (використовується Redux або інші сторонні бібліотеки)

Ця таблиця надає загальну порівняльну інформацію між Angular та React. Вибір між ними залежить від потреб проекту, розміру команди розробників та рівня їхнього досвіду, а також від екосистеми та підтримки, які найкраще підходять для вашого конкретного проекту.

Angular має декілька переваг для створення веб-застосунків з продажу музичних платівок:

– Модульність: Angular пропонує модульну структуру, що дозволяє легко організовувати та управляти функціональністю веб-застосунку. Ви можете розділити ваш застосунок на різні модулі, такі як каталог платівок, кошик покупок, оплата тощо, що спрощує розробку та підтримку коду.

– Розширюваність: Angular надає можливість розширювати функціональність застосунку за допомогою власних компонентів, директив та сервісів. Це означає, що ви можете легко створювати власні компоненти для відображення музичних платівок, фільтрів, сортування тощо, що сприяє виразному та гнучкому дизайну інтерфейсу.

– Двостороннє зв'язування даних: Angular надає потужну можливість двостороннього зв'язування даних. Це означає, що зміни, зроблені в моделі даних, автоматично відображаються в інтерфейсі користувача і навпаки. Для прикладу, ви можете оновлювати кошик покупок в реальному часі, коли користувач додає або видаляє платівки.

– Підтримка SPA (односторінкових додатків): Angular підтримує розробку односторінкових додатків (SPA), що дозволяє створювати багатосторінкові додатки з більш зручним користувацьким інтерфейсом. Завдяки SPA, веб-застосунок з продажу музичних платівок може забезпечити швидку та безперервну навігацію між різними розділами, зберігаючи стан застосунку.

– Ефективна обробка даних: Angular пропонує можливості для ефективної обробки даних, таких як фільтрація, сортування та пагінація. Ви можете використовувати функціональність Angular для швидкого та точного відображення музичних платівок залежно від потреб користувача.

– Кросс-платформова підтримка: Застосунки, розроблені з використанням Angular, можуть бути легко виконуватись на різних платформах, таких як веб, мобільні пристрої або настільні комп'ютери. Це дозволяє широко поширити ваш застосунок та забезпечити доступ користувачам на різних пристроях.

Загалом, використання Angular для створення веб-застосунку з продажу музичних платівок дозволить вам отримати потужну та гнучку платформу для розробки, яка забезпечить зручний інтерфейс, ефективну обробку даних та кросс-платформову підтримку.

MongoDB є нереляційною (NoSQL) базою даних, яка може мати декілька переваг для веб-застосунків з продажу музичних платівок. Ось кілька переваг використання MongoDB:

– Гнучкість у моделюванні даних: MongoDB дозволяє гнучко моделювати дані, оскільки не вимагає фіксованої схеми. Це означає, що ви можете зберігати різноманітні дані про музичні платівки без необхідності строго дотримання зв'язків або структур даних.

– Швидкодія: MongoDB є дуже швидкою базою даних, оскільки вона використовує індекси та кешування для швидкого доступу до даних. Це особливо важливо в веб-застосунках, де швидкий доступ до інформації про музичні платівки є ключовим.

– Масштабованість: MongoDB дозволяє легко масштабувати базу даних горизонтально, додавати нові сервери та розподіляти навантаження між ними. Це робить його ідеальним вибором для веб-застосунків, які можуть зростати та масштабуватись з часом.

– Географічна реплікація: MongoDB підтримує географічну реплікацію, що дозволяє розподілити дані між різними фізичними місцями. Це може бути корисним для веб-застосунків з продажу музичних платівок, які мають клієнтів з різних регіонів світу.

– Простота використання: MongoDB має дружній до розробників інтерфейс і добре документовану документацію. Це робить його легким у використанні та інтеграції з веб-застосунками.

Загалом, використання MongoDB у веб-застосунку з продажу музичних платівок може забезпечити гнучкість, швидкодію, масштабованість та простоту використання для ефективного зберігання та доступу до даних.

Node.js є середовищем виконання JavaScript на серверній стороні, і використання його для бекенду веб-застосунку з продажу музичних платівок має кілька переваг:

– Єдиностронність мови програмування: Використання Node.js дозволяє розробляти як фронтенд, так і бекенд веб-застосунку за допомогою однієї мови програмування - JavaScript. Це спрощує розробку, спільне використання коду та зменшує поріг входження для розробників.

– Швидкодія та масштабованість: Node.js побудований на потужному та ефективному двигуні V8 JavaScript. Він надає незвичайно швидку обробку запитів, що особливо важливо для веб-застосунків з великою кількістю користувачів та

великим обсягом даних. Крім того, Node.js легко масштабується горизонтально, що дозволяє розподіляти навантаження між багатьма серверами.

– Нестопінгова архітектура: Node.js використовує нестопінгову архітектуру, що означає, що він може обробляти багато запитів одночасно без блокування. Це забезпечує високу масштабованість та продуктивність веб-застосунків.

– Багатофункціональність: Node.js має широкий вибір пакетів та модулів у своєму екосистемі, що дозволяє розробникам ефективно будувати різноманітні функціональності для веб-застосунків з продажу музичних платівок. Наявність пакетного менеджера npm робить доступ до цих ресурсів ще зручнішим.

– Розширюваність: Node.js підтримує використання асинхронного програмування та події, що дозволяє розробникам створювати швидкодіючі та масштабовані застосунки. Такий підхід дозволяє ефективно обробляти багато запитів одночасно та реалізовувати розподілені функції.

В цілому, використання Node.js для бекенду веб-застосунку з продажу музичних платівок забезпечує швидку розробку, швидкодію, масштабованість та ефективну роботу з великими обсягами даних.

Висновки:

Провівши проектування програмного продукту можна зробити ряд висновків що до проектування. Перш за все було прийнято рішення використовувати клієнт-серверну архітектуру для створення застосунку проаналізовані її переваги та недоліки. Був проведений аналіз та створення структури даних. Була спроектована клієнтська частина застосунка яка буде повністю задовольняти потреби користувачів застосунку та покупців. Розроблена клієнтська частина має можливість своїм візуалом заволікати нових користувачів та зацікавлювати їх у купівля вінілу. Після проведення існуючих рішень був проведений аналіз технологій та обраний найкращий стек технологій для БД, клієнтської та серверної частин. Отже даний стек технологій та обрана архітектура дозволять вирішитити усі проблем та задачі які стоять перед розробкою ПЗ.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка бази даних

Node.js дозволяє легко взаємодіяти з базами даних MongoDB. Для цього потрібно встановити драйвер MongoDB Node.js, який дозволить підключитися до бази даних та виконувати запити. Щоб встановити цей драйвер, виконайте наступну команду:

```
npm install mongodb
```

Після цього, вам потрібно створити кластер MongoDB Atlas і завантажити зразки даних для використання.

Створення бази даних MongoDB для веб-магазину музичних платівок включає наступні кроки:

- Встановлення та налаштування MongoDB: Завантажте та встановіть MongoDB на свій сервер або скористайтеся облачним сервісом MongoDB. Налаштуйте параметри доступу та безпеки, які відповідають вашим потребам.
- Створення бази даних: Запустіть MongoDB та створіть нову базу даних для вашого веб-магазину музичних платівок. Надайте їй ім'я, яке буде відображати призначення бази даних.
- Визначення колекцій: Колекції в MongoDB відповідають таблицям у реляційних базах даних. Визначте колекції, які будуть зберігати дані вашого веб-магазину, наприклад, колекцію "Платівки" для зберігання інформації про музичні платівки.
- Визначення схеми даних: Визначте структуру даних для кожної колекції у MongoDB. Наприклад, для колекції "Платівки" можна визначити поля, такі як "назва", "виконавець", "жанр", "ціна" тощо. Визначте типи даних та інші обмеження, які необхідні для кожного поля.
- Запис даних: Додайте записи до вашої бази даних, вставивши дані про музичні платівки в колекцію "Платівки". Кожен запис буде представляти окрему музичну платівку та її атрибути.

– Запити до бази даних: Виконуйте запити до бази даних MongoDB, щоб отримувати, оновлювати та видаляти дані. Наприклад, ви можете виконувати запити для пошуку платівок за жанром, сортування за ціною, підрахунку кількості платівок тощо.

– Індексування: Встановіть індекси для полів, які використовуються у пошукових запитах чи для забезпечення унікальності даних. Індексування може покращити швидкість пошуку та ефективність роботи з базою даних.

– Захист даних: Забезпечте безпеку вашої бази даних MongoDB шляхом встановлення прав доступу, аутентифікації та інших заходів безпеки. Використовуйте ролі та дозволи, щоб обмежити доступ до бази даних лише для авторизованих користувачів.

Зі створеною схемою БД можна ознайомитись за допомогою діаграми класів представленої на рисунку 3.1.

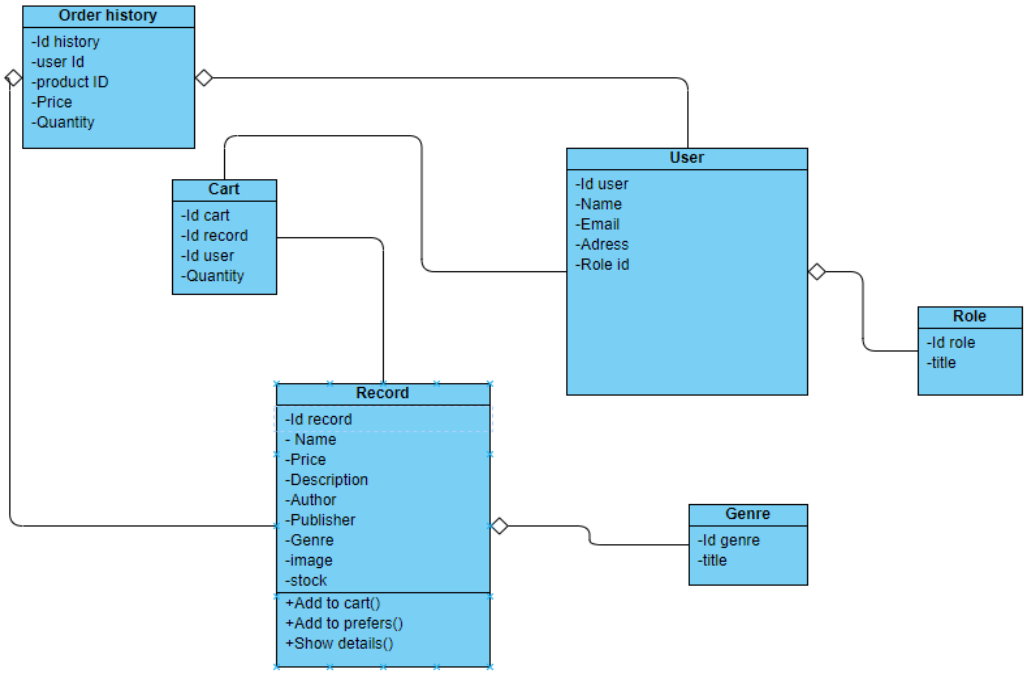


Рисунок 3.1 – Діаграма класів

3.2 Розробка програмних модулів

Архітектура Angular проекту для веб-застосунку з продажу музичних платівок може бути розроблена з використанням таких ключових компонентів та підходів:

Компоненти:

- Головний компонент (App Component): Це основний компонент, який включає інші компоненти та управляє загальною структурою сторінки.
- Компоненти для відображення списку платівок, деталей платівок, кошика та інших важливих функціональних елементів магазину.
- Компоненти для форм (наприклад, реєстрація, оформлення замовлення), які взаємодіють з користувачем для введення даних та виконання дій.

Сервіси:

- Сервіси для отримання та обробки даних про платівки з сервера або з бази даних, такі як сервіс для отримання списку платівок, сервіс для додавання до кошика тощо.
- Сервіси для авторизації та аутентифікації користувачів.
- Сервіси для керування станом додатку, наприклад, сервіс для управління кошиком, додаванням і видаленням платівок тощо.

Модулі:

- Основний модуль (App Module): Це головний модуль, який об'єднує всі компоненти, сервіси та інші модулі додатку.
- Модулі для реєстрації, авторизації та інших функціональних блоків додатку.

Маршрутизація:

- Використовуйте модуль маршрутизації (Angular Router) для визначення та керування шляхами сторінок в додатку. Він дозволяє встановлювати адреси URL для різних сторінок та налаштовувати, який компонент повинен бути відображений при переході за певним шляхом.

Розробку модулів почнемо з опису структури проекту зображеної на рисунку 3.1

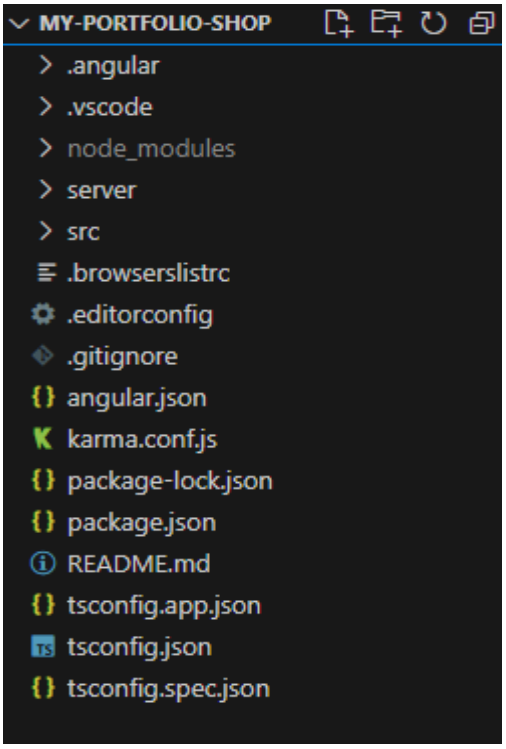


Рисунок 3.2 – Структура програми

Застосунок містить у собі 2 основних папки server та src. Server вміщає у собі логіку роботи з бекенд частиною та БД, а src клієнтську логіку.

Папка models створена для опису структури в об’єкта в базі даних. В проекті є три моделі: User, Order, Record. Їх представлено на рисунку 3.2.

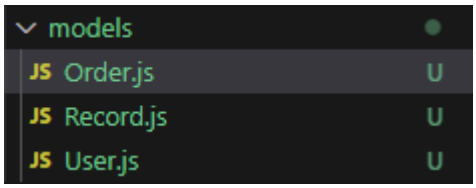


Рисунок 3.3 – папка моделей

Приклад створення моделі з Платівкою:

```
const mongoose = require('mongoose');
// Створення схеми моделі
```

```
const vinylSchema = new mongoose.Schema({

  title: {
    type: String,
    required: true
  },
  artist: {
    type: String,
    required: true
  },
  genre: {
    type: String,
    required: true
  },
  price: {
    type: Number,
    required: true
  },
  quantity: {
    type: Number,
    default: 0
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});

// Створення моделі на основі схеми
const Vinyl = mongoose.model('Vinyl', vinylSchema);

// Експорт моделі для використання в інших модулях
module.exports = Vinyl;
```

Папка сервіси містить усі сервіси які використовує Angular застосунок.

Сервіс в Angular є класом, який надає функціональність та оброблює бізнес-логіку для компонентів або інших сервісів. Він може містити методи для отримання або збереження даних, взаємодії з зовнішніми API, обробки подій та багато іншого.

Приклад сервіса що керує записом обраного обєкта в Subject RxJS.

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';
@Injectable({
  providedIn: 'root'
})
export class ChosenObjectService {
  chosenObject: BehaviorSubject<any> = new BehaviorSubject<any>(null);
```

```
  chosenFilter: BehaviorSubject<any> = new BehaviorSubject<any>(null);
  chosenSorting: BehaviorSubject<any> = new BehaviorSubject<any>(null);
  constructor() { }
}
```

Далі розглянемо папку src зображену на рисунку 3.4.

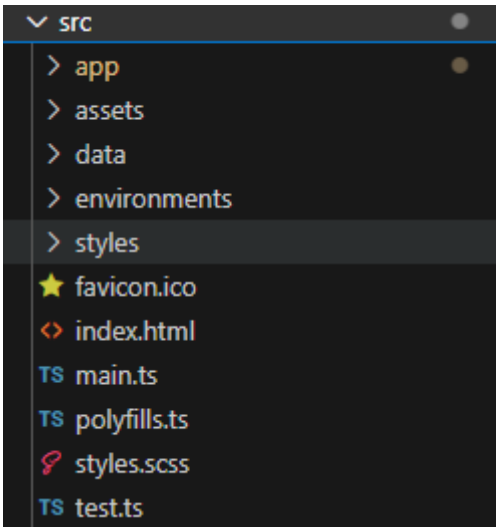


Рисунок 3.4 – src папка

У папці src в Angular проекті знаходиться вихідний код додатку. Вона містить усі необхідні файли для розробки та збирання Angular додатку. Ось загальний опис важливих файлів та папок в папці src:

- app: Ця папка містить компоненти, шаблони, стилі та інші файлів пов'язані з основними функціональностями додатку. Вона зазвичай містить файл app.component.ts, який є головним компонентом додатку.
- assets: В цій папці зберігаються статичні файли, такі як зображення, шрифти або конфігураційні файли. Вони можуть бути використані в додатку.
- environments: Ця папка містить конфігураційні файли для різних середовищ (наприклад, розробка, продакшн). Вони визначають змінні середовища, такі як URL API або ключі доступу.
- styles: У цій папці знаходяться файли стилів, такі як CSS або SASS. Файл styles.scss є головним файлом стилів, який використовується в додатку.
- index.html: Це основний HTML-файл додатку, який містить загальну структуру сторінки і включає JavaScript та CSS файли.

– main.ts: Цей файл є точкою входу додатку. Він імпортує головний модуль додатку і запускає його.

– polyfills.ts: У файлі polyfills.ts включені поліфіли, які забезпечують підтримку функціональності в різних браузерах.

– angular.json: Це конфігураційний файл Angular, в якому визначаються налаштування проекту, такі як шляхи до файлів, конфігурації збирання та інші параметри.

На рисунку 3.5 зображена папка assets що містить в собі усі зображення які використовуються у веб-застосунку

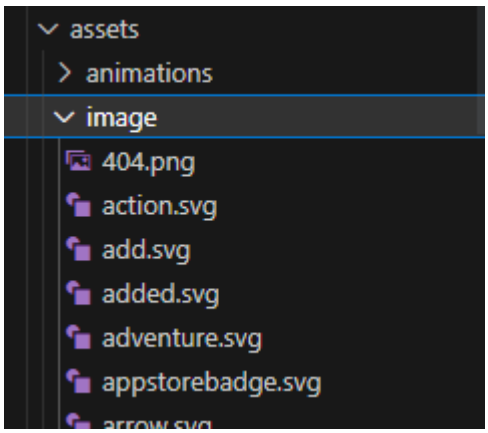


Рисунок 3.5 – assets папка

Також важливою складовою сучасних проектів є візуальна частина. Цікаві візуальні ефекти досягаються за допомогою різних анімацій, canvas, transition. Візуальні ефекти будуть розміщені у папці utils. Структура папки utils зображена на рисунку 3.6.

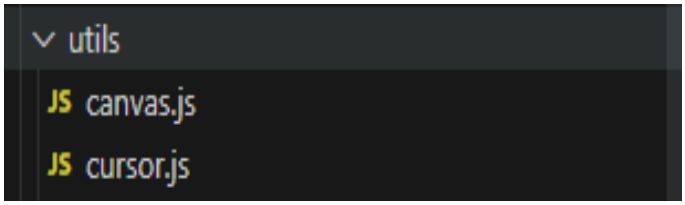


Рисунок 3.6 – структура utils

Далі розглянемо частинку коду яка відповідає за створення кастомного курсора.

```
export function Cursor() {
   var cursor = document.querySelector('.cursor');
   var cursorinner = document.querySelector('.cursor2');
   var a = document.querySelectorAll('a');

   document.addEventListener('mousemove', function(e){
   var x = e.clientX;
   var y = e.clientY;
   cursor.style.transform = `translate3d(calc(${e.clientX}px - 50%), calc(${e.clientY}px - 50%), 0)`
   });
```

Event Listener (слухач подій) - це функція або метод, який призначений для відстеження та реагування на певні події, які виникають в програмі або на веб-сторінці.

У веб-розробці, елементи сторінки (наприклад, кнопки, посилання, поля введення) можуть генерувати різні події, такі як клік, наведення курсора, введення тексту та інші. Щоб реагувати на ці події, можна використовувати event listener.

Event Listener слідкує за виникненням певної події та виконує задану функцію або метод, коли подія стає активною. В JavaScript, для додавання event listener до елемента, використовуються методи, такі як addEventListener().

Файл index.html виступає як точка входу для додатку. В ньому знаходиться кореневий компонент (app-root component) проекту, який є обгорткою для всіх інших компонентів.

Приклад index.html

```
<!doctype html>
<html lang="en">
  <script src="http://cdnjs.cloudflare.com/ajax/libs/gsap/1.18.0/TweenMax.min.js"></script>
<head>
  <meta charset="utf-8">
  <title>MyPortfolioShop</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Далі розглянемо структуру папки components. Ця папка буде містити усі необхідні компоненти для створення застосунку всередині папка components розгалужена на папку з примітивами, групами та сторінками.

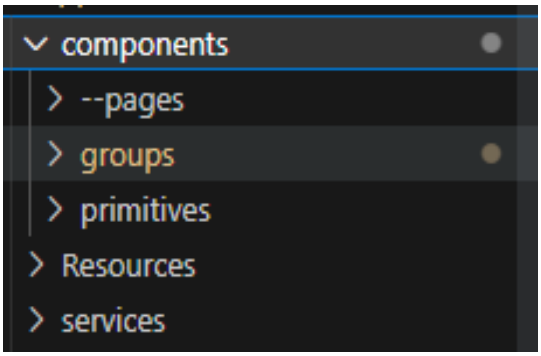


Рисунок 3.7 –структура папки components

Далі розглянемо приклад компоненту примітива кнопки.

App-button.component.html

```
<div class="button" (click)="changeState()">
   <ng-container *ngIf="state">
      <p class="title">Add to cart</p>
   </ng-container>
   <ng-container *ngIf="!state">
      <p class="title added">Added</p>
   </ng-container>
</div>
```

Логіка роботи з компонентом сконцентрована в TS файлі.

App-button.component.ts

```
import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';
@Component({
  selector: 'app-pr-add-to-card-button',
  templateUrl: './pr-add-to-card-button.component.html',
  styleUrls: ['./pr-add-to-card-button.component.scss']
})
export class PrAddToCardButtonComponent implements OnInit {
  @Input() state!: boolean;

  @Output() stateChanger: EventEmitter<boolean> = new EventEmitter();
  constructor() { }

  ngOnInit(): void {
  }
  changeState(){
    this.stateChanger.emit(!this.state);
  }
}
```

Цей компонент має певні властивості та приймає параметри.

@Input та @Output - це дві декоратори, які використовуються в Angular для забезпечення зв'язку та передачі даних між компонентами.

– @Input: Цей декоратор використовується для прийому даних від батьківського компонента до дочірнього компонента. Він дозволяє передати значення зовнішнього компонента в компонент, в якому він використовується. Щоб використовувати @Input, потрібно оголосити властивість в дочірньому компоненті з декоратором @Input перед нею. Це дозволяє зовнішньому компоненту передати значення через цю властивість.

– @Output: Цей декоратор використовується для передачі даних від дочірнього компонента до батьківського компонента. Він дозволяє сповіщати батьківський компонент про події або зміни, які відбуваються у дочірньому компоненті. Для використання @Output потрібно оголосити об'єкт типу EventEmitter у дочірньому компоненті та призначити йому значення змінної або події, яка буде викликатися. Батьківський компонент може підписатися на цю подію і реагувати на неї.

Використання @Input та @Output дозволяє створювати компоненти, які взаємодіють один з одним та передають дані між собою, що допомагає створювати більш модульний та зрозумілий код.

Стилізація компонентів виконується за допомогою Scss.

SCSS (Sassy CSS) - це розширена версія CSS, яка додає багато корисних функцій і можливостей для полегшення розробки стилів. SCSS використовує синтаксис, сумісний з CSS, але додає додаткові можливості, такі як змінні, вкладені селектори, міксіни, наслідування і багато іншого.

SCSS дозволяє писати більш структурований, повторно використовуваний і керований код стилів, що робить його потужним інструментом для розробників веб-інтерфейсів.

3.3 Керівництво користувача

Керівництво користувача для веб-застосунку з продажу музичних платівок:

Для користувача:

Вітання на головній сторінці: При отриманні доступу до веб-застосунку, ви отримаєте привітання на головній сторінці. Тут ви знайдете основні елементи навігації та можливість шукати та переглядати музичні платівки.

Перейшовши на головну сторінку додатку користувач зможе ознайомитись з веб-застосунком. Важливо щоб інтерфейс був інтуєтивно зрозумілим та надавав інформацію про послуги та товари які надає застосунок. З виглядом головної сторінки можна ознайомитись на рисунку 3.8.

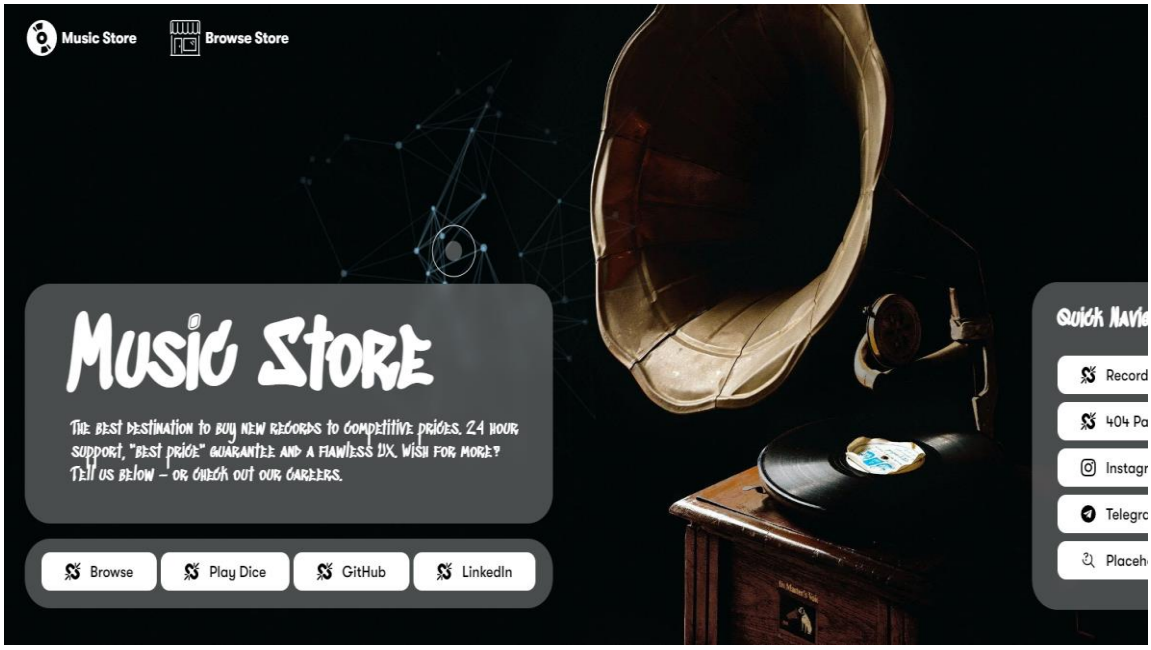


Рисунок 3.8 – Голован сторінка додатку

Серед іншого функціоналу до якого має доступ користувач можна виділити такі пункти.

Пошук музичних платівок: Використовуйте функцію пошуку на головній сторінці, щоб знайти платівки за назвою, виконавцем або жанром. Просто введіть ключове слово в поле пошуку і натисніть Enter або натисніть на кнопку пошуку.

Перегляд деталей платівки: Клікніть на будь-яку музичну платівку, щоб переглянути її деталі, такі як обкладинка, назва, виконавець, жанр, ціна та опис. Також ви можете переглядати фотографії альбома та іншу додаткову інформацію.

Рисунок 3.9 –деталі музичної платівки

Додавання платівки в кошик: Якщо ви хочете придбати платівку, просто клікніть на кнопку "Додати в кошик" на сторінці деталей платівки. Ваша платівка буде додана в кошик.

Корзина: У верхньому меню є пункт "Кошик". Клікніть на нього, щоб переглянути зміст вашої кошика та здійснити оформлення замовлення.

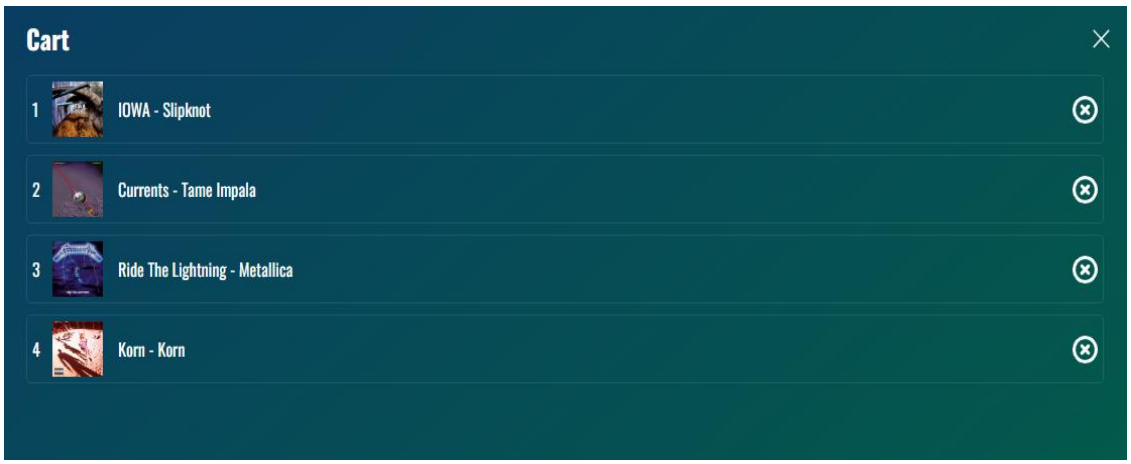


Рисунок 3.10 – Вигляд кошика користувача

Для адміністратора система надає такі функції.

Увійдіть в систему: Використовуйте свої облікові дані, щоб увійти в систему адміністратора. У вас повинні бути окремі облікові дані для доступу до адмін-панелі. Авторизація користувача зображена на рисунку 3.11.

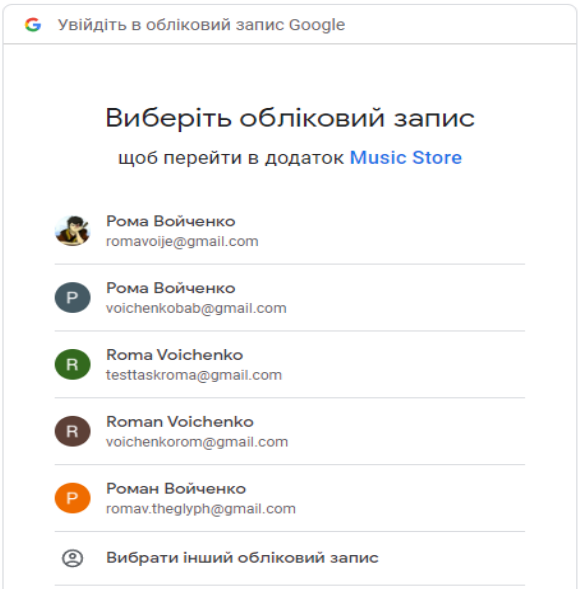


Рисунок 3.11 - авторизація

Адмін-панель: Після успішного входу ви потрапите на адмін-панель, де знайдете різноманітні опції для управління веб-застосунком. На рисунку 3.12 зображено автьоризованого користувача застосунку.

Рисунок 3.12 – авторизація

На рисунку 3.13 зображено втгляд галереї продуктів для адміністратора.

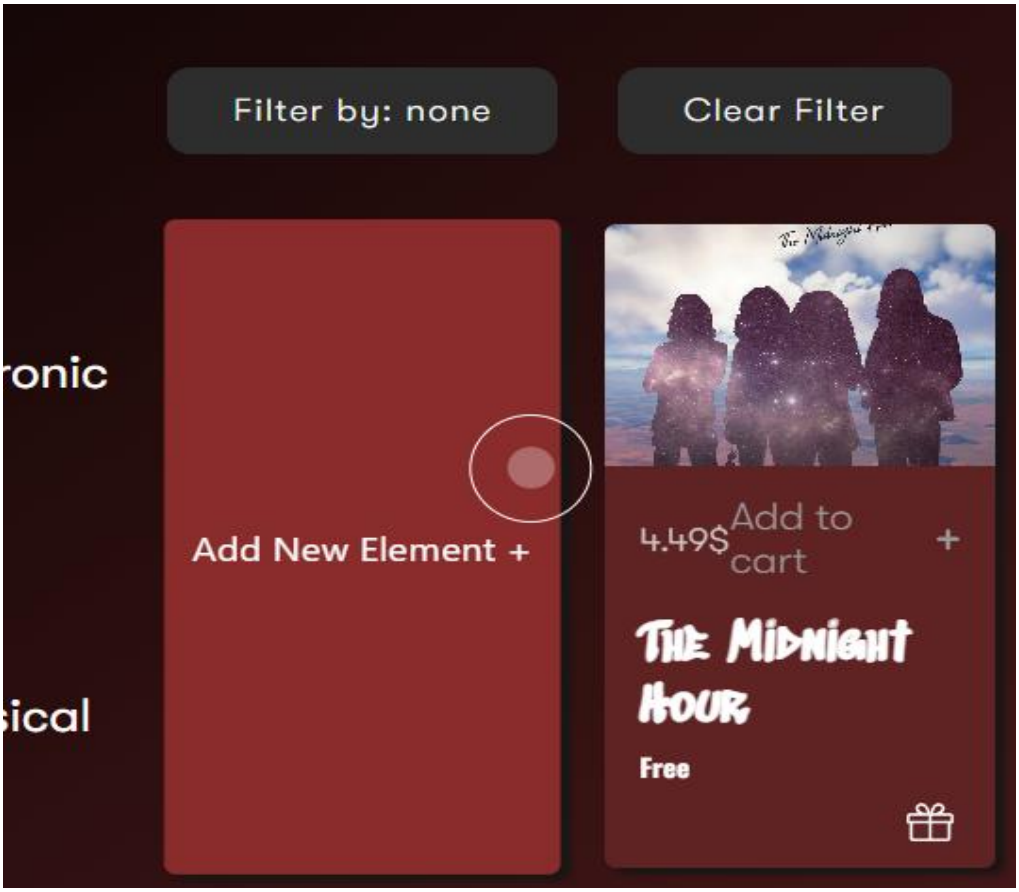


Рисунок 3.13 – Галерея платівок для адміністратора

Для економії бюджету частина адміністрування застосунку буде розроблена з чисто функціональної частини і не має важливого значення створення гарного веб дизайну для застосунку. Це дозволить економити час та бюджет розробникам застосунку. Також було вирішено що клієнтська частина адміністрування застосунку має бути максимально візуально наближена до корстувацької.

Додавання нових платівок: У адмін-панелі є можливість додавати нові платівки до магазину. Заповніть всю необхідну інформацію, таку як назва, виконавець, жанр, ціна, опис та фотографії платівки. На рисунку 3.14 зображена форма для додавання нової платівки.

Рисунок 3.14 – Форма додавання нового товару адміністратора

Редагування та видалення платівок: Ви можете редагувати існуючі платівки, вносячи зміни до їх інформації або видаляти платівки, якщо вони більше не доступні для продажу.

3.4 Технічні характеристики веб-застосунку

Мінімальні технічні характеристики для пристрою для перегляду веб-застосунку. Ось загальні рекомендації:

– Операційна система: Пристрій повинен мати підтримку сучасних операційних систем, таких як Windows, macOS, Linux, Android або iOS.

– Веб-браузер: Найновіша версія сумісного веб-браузера (наприклад, Google Chrome, Mozilla Firefox, Safari, Microsoft Edge). Для оптимального відображення та функціональності веб-застосунку рекомендується використовувати оновлену версію браузера.

– Процесор: Мінімум двоядерний процесор з частотою не менше 1.8 ГГц.

– Оперативна пам'ять (RAM): Рекомендовано мати не менше 4 ГБ оперативної пам'яті для забезпечення плавної роботи веб-застосунку та запобігання зависанням або повільному завантаженню сторінок.

– Дисплей: Мінімальне розширення екрану 1280x800 пікселів для забезпечення зручного відображення веб-інтерфейсу.

– Інтернет-підключення: Стабільне та достатньо швидке Інтернет-підключення для завантаження сторінок, зображень та інших ресурсів веб-застосунку.

– Вбудовані функції: Пристрій повинен мати підтримку JavaScript та включені вбудовані функції, такі як відтворення звуку та відео, для повного функціоналу веб-застосунку.

3.5 Розгортання та встановлення системи

Для успішного розгортання цього проекту необхідно мати певні компоненти встановлені на вашому комп'ютері, зокрема Node.JS і MongoDB. Почніть з відкриття папки проекту і відкриття двох терміналів. Для запуску клієнтської частини виконайте наступні кроки:

– Перейдіть до папки "client" в терміналі за допомогою команди "cd client".

– Виконайте команду "npm install" для встановлення залежностей проекту.

– Запустіть команду "npm start" для запуску клієнтської частини.

Для запуску серверної частини виконайте такі кроки:

– Перейдіть до папки "server" в терміналі за допомогою команди "cd server".

– Виконайте команду "npm install" для встановлення залежностей проекту.

– Запустіть команду "node index.js" для запуску серверної частини.

У цьому описі було пояснено процес створення бази даних та її підключення до веб-додатку. Також була розглянута декомпозиція системи на модулі та надано детальний опис кожного модуля. Були визначені технічні характеристики інтернет-платформи. Надано алгоритм для встановлення та використання розробленого програмного продукту.

3.6 Тестування веб-застосунку

Існує кілька підходів для тестування Angular веб-застосунків, включаючи тести одиниць (unit tests), інтеграційні тести (integration tests) та e2e-тести (end-to-end tests). Кожен з цих підходів має свої особливості і використовується для різних цілей. Ось короткий опис кожного з них:

- Тести одиниць (Unit tests): Це тести, які перевіряють окремі компоненти, сервіси, директиви або пайпи в ізоляції від інших частин додатку. Вони дозволяють перевірити правильність роботи окремих модулів і функціональностей. Для тестування одиниць використовуються фреймворки, такі як Karma та Jasmine.
- Інтеграційні тести (Integration tests): Це тести, які перевіряють взаємодію між компонентами, сервісами, директивами тощо. Вони дозволяють перевірити, чи працюють різні частини додатку разом як очікується. Для інтеграційних тестів використовуються фреймворки, такі як TestBed та Jasmine.
- E2E-тести (End-to-end tests): Це тести, які симулюють поведінку користувача і перевіряють функціональність додатку в реальному середовищі. Вони охоплюють багато компонентів і перевіряють, чи працює застосунок від початку до кінця згідно з очікуваннями користувача. Для e2e-тестування використовуються фреймворки, такі як Protractor або Cypress.

Комбінація цих підходів дозволяє забезпечити широкий спектр тестування функціональності, якість коду та взаємодію різних компонентів в Angular веб-застосунку з продажу вінілових платівок. Кожен підхід має свої переваги і може бути використаний для покриття різних аспектів додатку.

Ось приклад юніт-тесту для даного компонента меню музичних жанрів за допомогою Karma:

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { GrGenresMenuComponent } from './gr-genres-menu.component';
import { ProductService } from 'src/app/services/product.service';
import { BehaviorSubject } from 'rxjs';
describe('GrGenresMenuComponent', () => {
  let component: GrGenresMenuComponent;
  let fixture: ComponentFixture<GrGenresMenuComponent>;
  let productService: ProductService;
```

```
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [GrGenresMenuComponent],
      providers: [ProductService]
    })
      .compileComponents();
  });
  beforeEach(() => {
    fixture = TestBed.createComponent(GrGenresMenuComponent);
    component = fixture.componentInstance;
    productService = TestBed.inject(ProductService);
    fixture.detectChanges();
  });
  it('повинен створити компонент', () => {
    expect(component).toBeTruthy();
  });
  it('повинен емітувати вибраний жанр у методі selectGenre', () => {
    spyOn(productService.subject, 'next');
    const genre = { name: 'Rock', icon: 'assets/image/action.svg' };
    component.selectGenre(genre);
    expect(productService.subject.next).toHaveBeenCalledWith(genre);
  });
});
```

У цьому тесті ми створюємо екземпляр `GrGenresMenuComponent` і перевіряємо, що він успішно створений. Також ми тестуємо метод `selectGenre`, використовуючи спай в наступний метод `productService.subject`, і перевіряємо, що він був викликаний з очікуваним об'єктом жанру.

Зверніть увагу: Перед запуском тестів переконайтеся, що імпортуєте необхідні залежності та налагоджуєте оточення для виконання тестів.

Нижче на рисунку 3.15 представдена папка з конфігом для karma.

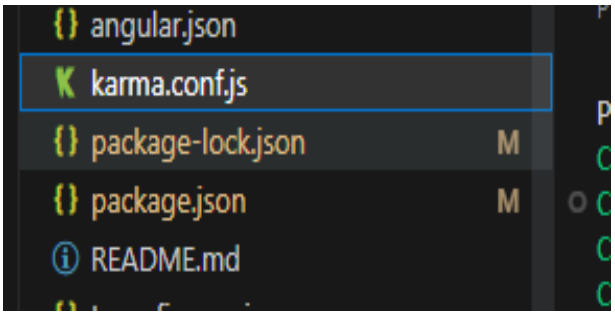


Рисунок 3.15 – Karma config

Успішний Karma тест повертає повідомлення "SUCCESS" або "All tests pass", яке відображається в консолі або з'являється в інтерфейсі тестування. Це означає, що всі тести пройшли успішно і відповідають очікуванням. Крім того, ви отримаєте

повний звіт про пройдені тести, включаючи кількість успішних та неуспішних тестів, час виконання тестів і іншу інформацію, що допомагає оцінити стан вашого проекту.

Ось приклади тестових наборів для веб-застосунку з продажу музичних платівок:

Набір тестів для перевірки функціональності додавання товару в корзину:

- Тест на додавання товару в корзину при натисканні кнопки "Додати в корзину".
- Тест на перевірку кількості товарів у корзині після додавання.
- Тест на перевірку правильності відображення обраного товару в корзині.
- Набір тестів для перевірки пошуку товарів:
- Тест на пошук товару за назвою.
- Тест на перевірку правильності відображення результатів пошуку.
- Тест на перевірку поведінки пошуку при введенні неправильного запиту.
- Набір тестів для перевірки сортування товарів:
- Тест на сортування товарів за ціною у зростаючому порядку.
- Тест на перевірку правильності відображення відсортованих товарів.
- Набір тестів для перевірки процесу оформлення замовлення:
- Тест на заповнення форми замовлення з правильними даними.
- Тест на перевірку валідації форми замовлення з неправильними даними.
- Тест на успішне оформлення замовлення і отримання підтвердження.
- Набір тестів для перевірки аутентифікації та авторизації:
- Тест на вхід в систему з правильними обліковими даними.
- Тест на перевірку відображення правильних даних користувача після входу.
- Тест на перевірку поведінки системи при спробі доступу до захищених ресурсів без авторизації.

Ці тестові набори покривають основні функціональність веб-застосунку з продажу музичних платівок і допоможуть перевірити правильність його роботи. Звичайно, ви можете розширити ці набори або додати додаткові тести залежно від потреб вашого проекту.

Висновки:

Провівши програмну реалізацію застосунку був зроблений ряд висновків. Програмна реалізація додатків мала деякі складнощі, але вцілому була виконана в повному обсязі та має перспективи для масштабуваня та додавання нового функціоналу до застосунку. Під час програмної реалізації була створена БД, можливість авторизації користувача. Був розроблений код для серверної та клієнтської частини додатку. Булир розроблені різні клієнтські анімації та створена архітектура проекту.У розділі тестування застосунку були створені тестові набори та написані юніт тести що покращить безпеку та роботу застосунку, а також створить базу для його подальшого розширення та додавання нового функціоналу.

ВИСНОВКИ

Висновки для кваліфікаційної роботи на тему "Веб-застосунок для продажу музичних платівок":

Під час виконання кваліфікацінйої роботи було проаналізовано предметну область, також була доведена актуальність розробки даного проекту і визначені причини, через які часто виникають проблеми у людей, які не користуються веб-додатками у сфері електронної комерції. На підставі проведення аналізу схожих застосунків, було створено список їх переваг і недоліків та визначено функціональні та нефункціональні вимоги до програмного продукту. На основі цих вимог сформовано технічне завдання а також варіанти використання додатку.

У рамках даної роботи був розроблений веб-застосунок з продажу музичних платівок. Застосунок надає можливість користувачам переглядати, шукати та придбавати вінілові платівки з різних жанрів.

Для реалізації проекту було використано Angular framework для фронтенду та Node.js з використанням Express.js для бекенду. MongoDB була обрана як база даних для зберігання інформації про платівки та замовлення користувачів.

Веб-застосунок має інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам з легкістю шукати платівки за жанрами, переглядати детальну інформацію про товари та додавати їх у кошик. Також, користувачі можуть оформляти замовлення та виконувати оплату.

В процесі розробки веб-застосунку були враховані принципи дизайну та взаємодії, забезпечуючи зручну навігацію та позитивний користувацький досвід.

Для забезпечення якості та надійності програмного забезпечення, було проведено тестування різних функціональних модулів, таких як додавання товару в корзину, пошук, сортування та оформлення замовлення.

В результаті роботи були досягнуті поставлені цілі, а саме створено функціональний веб-застосунок для продажу музичних платівок, який надає зручний інтерфейс користувачам та забезпечує необхідну функціональність.

Проект має потенціал для подальшого розширення, наприклад, додавання функціоналу аутентифікації користувачів, можливості додавання відгуків та оцінок до товарів, а також інтеграцію з платіжними системами для забезпечення онлайн-оплати.

В цілому, розроблений веб-застосунок є ефективним інструментом для продажу музичних платівок, забезпечуючи зручний спосіб пошуку та придбання вінілових записів. Використання сучасних технологій розробки та дизайну дозволило створити продукт, який відповідає потребам користувачів та може бути успішно використаний у комерційних цілях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бедратюк Л. П., Радельчук Г. І., Форкун Ю. В., Яшина О. М. "Дипломний проект: методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення»". Хмельницький: ХНУ, 2020. – 77с.

2. Aguilar A. "E-commerce Website Optimization: Why 95% of Your Website Visitors Don't Buy, and What You Can Do About It."

3. Застосунок аналог Vinyla. URL: https://vinyla.com/ (дата звернення - 26.05.23).

4. Застосунок аналог Vinyl Records https://vinylrecords.com.ua/(дата звернення - 26.05.23).

5. Chaffey D., Ellis-Chadwick F. "Digital Marketing: Strategy, Implementation and Practice."

6. Clark K., Perrin B. "Web Design Playground: HTML & CSS the Interactive Way."

7. Collis L., Gibson C. "Running a Web Design Business: Defining Your Business, Managing Your Clients, and Getting Paid."

8. Corral A. "Mastering E-commerce Analytics: Maximizing the Online Shopping Experience."

9. Cumming S., Van Niekerk A. "Mastering Laravel: Build Scalable Laravel Apps."

10. Davis J. "Web Design with HTML, CSS, JavaScript and jQuery Set."

11. Deason K. "Dropshipping E-commerce Business Model 2020: A Step-by-Step Guide With The Latest Techniques On How To Start Building, Growing and Scaling Your Shopify and Online Store in No Time."

12. Eckstein R. "Mastering Firebase for Android Development."

13. Elgammal M. "Mastering Web Development with Angular."

14. Fielding J., Taylor R., Hughes L. "PHP & MySQL: Novice to Ninja."

15. Goel V. "React and React Native: A Complete Hands-On Guide to Modern Web and Mobile Development with React.js, 3rd Edition."

16. González J. A. G., Peña R. M. "Django 2 by Example: Build powerful and reliable Python web applications from scratch."

17. Kauffman R. J., Li T. "E-commerce 2019: Business, Technology, Society."

18. Kuhn M., Johnson M., Hoffman T. "Modern PHP: New Features and Good Practices."

19. Lewis R., Seale W. "Building a RESTful Web Service with Spring."

20. Long L. "Full-Stack React, TypeScript, and Node: Build Scalable Full-Stack JavaScript Web Applications."

21. Loshin P. "Designing Enterprise Blockchain Applications: A Hands-On Approach."

22. McDonough B., Pollock P. "Professional WordPress: Design and Development."

23. Nielsen J., Budiu R. "Mobile Usability."

24. Oliver R. "HTML & CSS for Complete Beginners: A Step-by-Step Guide to Learning HTML5 and CSS3."

25. Pons J. "The Vue Handbook: A thorough introduction to Vue.js."

26. Powell M., Gallagher B. "Beginning PHP 7: From Novice to Profess."

27. Souders S. "High-Performance Web Sites: Essential Knowledge for Front-End Engineers."

28. Taft D. R., Williams R. L. "Java for Students."

29. Thomas D., Busch B. "Building Progressive Web Applications: Bringing the Power of Native to the Browser."

30. Welling L., Thomson L. "PHP and MySQL Web Development."

31. West A. "Mastering Firebase for Android Development: Build dynamic, real-time Android apps using Firebase."

32. Williams C. "CSS Master: Organized, Fast and Efficient CSS for Modern Websites."

33. Yank K. "HTML5 and CSS3 Visual QuickStart Guide."

34. Zietlow J., Davis M. "Building Progressive Web Apps: Bringing the Power of Native to the Browser."

35. Літтл С., Музетті А. "Майстерність розробки React: Нарахування веб-додатків з використанням React та Redux."

36. Макарова О., Черкашина Ю. "Торгівельний дизайн: засоби підвищення ефективності магазину."

37. Роуз Б. "E-commerce: Mastering the Basic Principles of E-commerce."

38. Сімсон Г., Шингарев І. "Розробка інтернет-магазину на Magento 2."

39. Томлінсон С. "WordPress for Beginners 2020: A Visual Step-by-Step Guide to Mastering WordPress."

40. Чапаєва І. Є., Стефаненка О. О. "Розробка та моделювання інтернет-магазину на платформі OpenCart."

41. Шафер С. "The E-commerce Book: Building the E-Empire."

ДОДАТОК А
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки веб застосунку для продажу музичних платівок. Технічне завдання розроблено у відповідності до стандарту ГОСТ 19.201–78.

1 Підстава для розробки

Підставою для розробки є «Завдання на кваліфікаційну роботу», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки: Веб-застосунок для продажу музичних платівок.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням додатку є клієнтська частина для купівлі музичних платівок.

2.2 Експлуатаційне призначення

Програма повинна експлуатуватися на будь-яких пристроях, на яких є браузер. Кінцевим користувачем додатку може виступати будь-яка особа.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Для звичайного користувача:

- перегляд інформаційних блоків про певну платівку;
- перегляд галереї платівок
- можливість оформлення замовлення;
- окремі сторінки під різні платівки;
- моливість сорування за музичними жанрами;
- можливість переглянути контакти застосунку
- можливість перегляду веб-сайту на різних мовах;

Для адміністратора:

- авторизація;
- створення ролей адміністраторів з відповідними доступами до редагування вмісту сайту;
- можливість створення та редагування музичних платівок;
- можливість перегляду замовлень;
- можливість завантаження фото у галереї на веб-сайті;

3.2 Вимоги до надійності

Веб-додаток повинен забезпечувати такі вимоги до надійності:

- обробляти невірні дії користувача і попереджати його про можливі наслідки;
- можливість самостійно відновлюватись у разі збою;
- можливість резервного копіювання бази даних.

3.3 Умови експлуатації та вимоги до технічних засобів

Веб-додаток повинен працювати на всіх пристроях, які мають браузер та стабільний доступ до мережі «Інтернет»: смартфонах, планшетах та комп’ютерах. Браузер може бути будь-яким: Safari, GoogleChrome, Opera, MozillaFirefox, тощо.

Для роботи платформи без збоїв, потрібно, щоб пристрій, на якому вона запускається задовольняв наступні мінімальні вимоги:

- 1 гб внутрішньої пам’яті;
- 2 Гб оперативної пам’яті;
- 4-ядерний процесор;
- доступ до мережі «Інтернет» зі швидкістю мінімум 20 Мбіт/с.

3.4 Вимоги до інформаційної та програмної сумісності

Для розробки веб-застосунку був використаний фронтенд JavaScript фреймворк Angular.js, бекенд Node.js, та база даних MongoDB.

3.5 Спеціальні вимоги

Програма повинна мати зручний та зрозумілий зовнішній інтерфейс користувача.

4 Вимоги до програмної документації

У момент здачі проекту замовнику надається наступний набір документів:

- текст програми;
- опис програми;
- технічне завдання;
- керівництво користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки веб-застосунку для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.23 – 31.01.23	Обґрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02.23 – 26.02.23	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури
Технічний проект 29.02.23 – 19.03.23	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка докладного алгоритму; розробка структури програми
Робочий проект 20.03.23 – 15.04.23	Розробка програмного забезпечення	Реалізація програмного забезпечення; відлагодження; проведення попереднього тестування

Кінець таблиці А.1

1	2	3
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 23.04.23 – 30.04.23	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і розгортання програмного забезпечення
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням

6 Порядок контролю та приймання

Контроль здійснюється кінцевими користувачами системи, підключеними на етапі тестування додатку.

ДОДАТОК Б
(обов’язковий)

ДІАГРАМИ

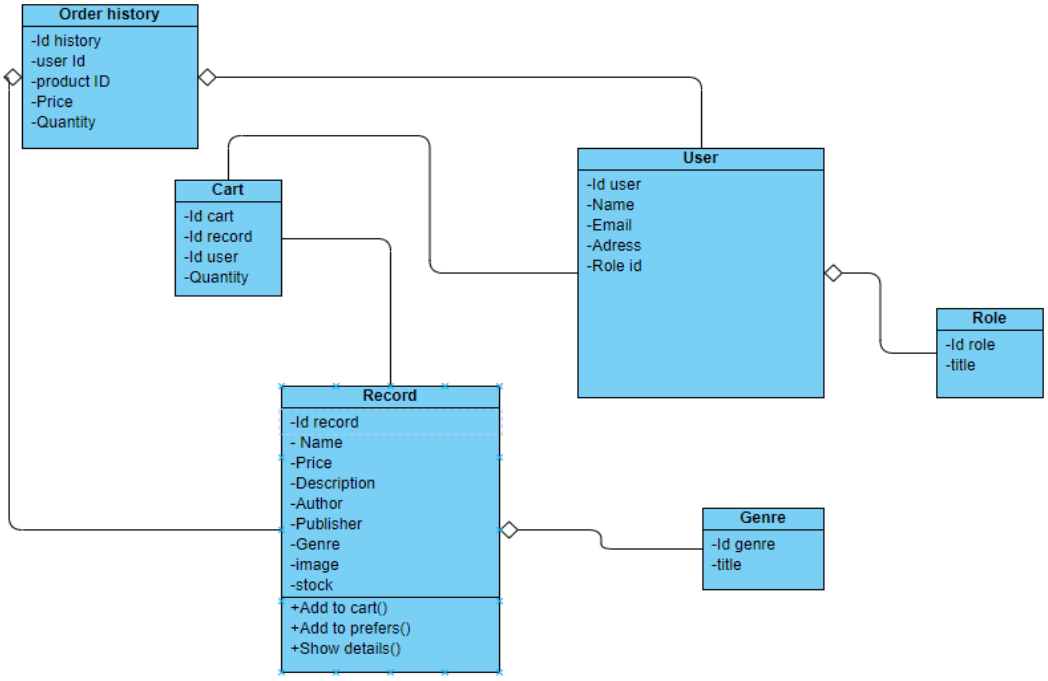


Рисунок Б.1 – Схема бази даних

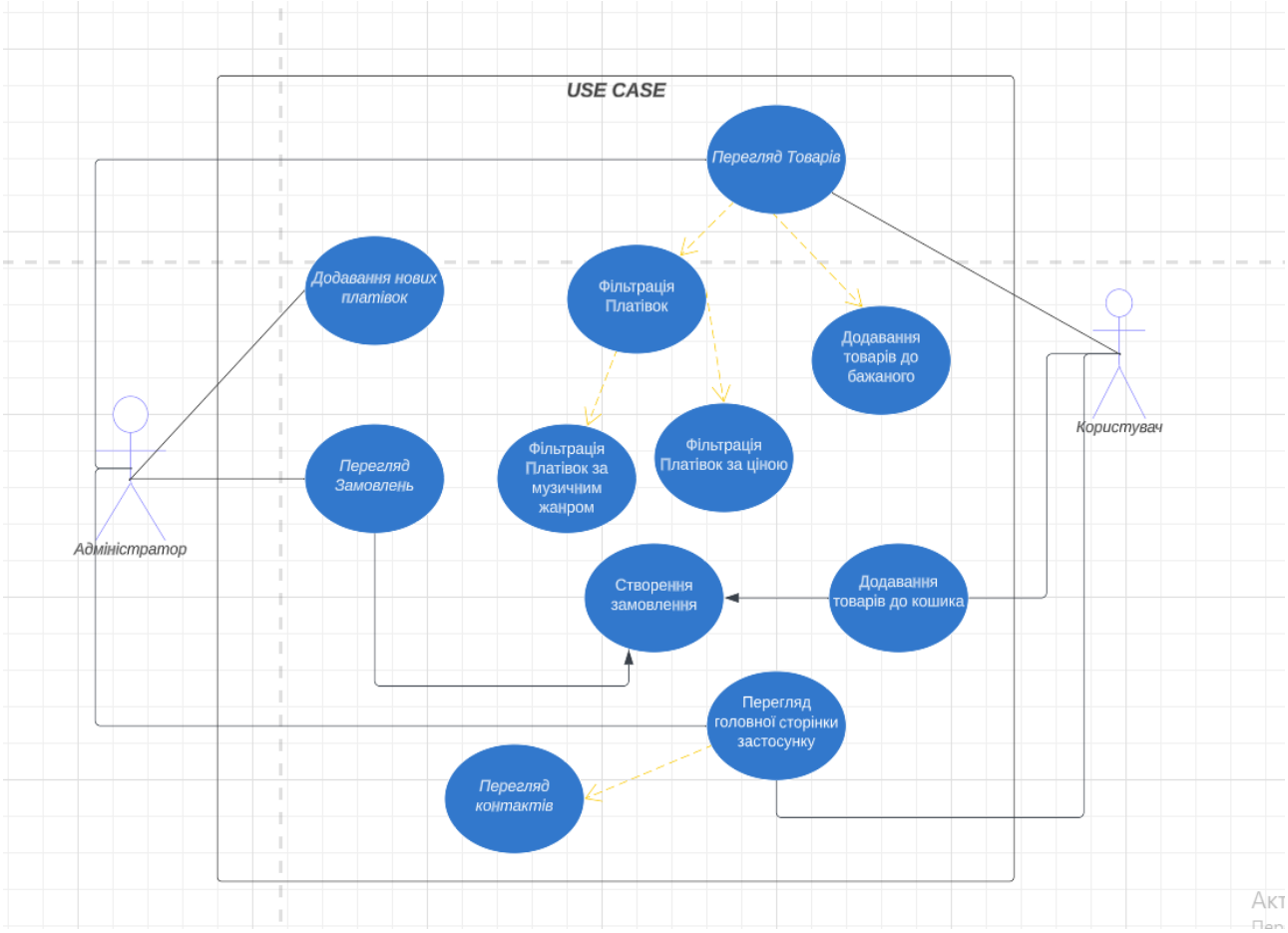


Рисунок Б.2 – Діаграма варіантів використання

ДОДАТОК В
(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

App.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomePageComponent } from './components/--pages/home-page/home-page.component';
import { GrNavigationComponent } from './components/groups/gr-navigation/gr-
navigation.component';
import { PrCtaButtonComponent } from './components/primitives/pr-cta-button/pr-cta-
button.component';
import { GrHeaderComponent } from './components/groups/gr-header/gr-header.component';
import { PrGhostButtonComponent } from './components/primitives/pr-ghost-button/pr-ghost-
button.component';
import { StorePageComponent } from './components/--pages/store-page/store-page.component';
import { GrProductCardComponent } from './components/groups/gr-product-card/gr-product-
card.component';
import { GrProductsGridComponent } from './components/groups/gr-products-grid/gr-products-
grid.component';
import { PrAddToCardButtonComponent } from './components/primitives/pr-add-to-card-button/pr-
add-to-card-button.component';
import { GrGenresMenuComponent } from './components/groups/gr-genres-menu/gr-genres-
menu.component';
import { GrFilterRowComponent } from './components/groups/gr-filter-row/gr-filter-row.component';
import { GrMainTitleComponent } from './components/groups/gr-main-title/gr-main-title.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { ProductPageComponent } from './components/--pages/product-page/product-
page.component';
import { GrGameInfoComponent } from './components/groups/gr-game-info/gr-game-
info.component';
// import { AngularFireModule } from '@angular/fire';
import { AngularFireModule } from '@angular/fire/compat';
import { AngularFireDatabaseModule } from '@angular/fire/compat/database';
import { AngularFireAuthModule } from '@angular/fire/compat/auth';
import { environment } from '../environments/environment';
import { PrSignInComponent } from './components/primitives/pr-sign-in/pr-sign-in.component';
import { PrLogOutComponent } from './components/primitives/pr-log-out/pr-log-out.component';
@NgModule({
  declarations: [
    AppComponent,
    HomePageComponent,
    GrNavigationComponent,
    PrCtaButtonComponent,
    GrHeaderComponent,
    PrGhostButtonComponent,
    StorePageComponent,
    GrProductCardComponent,
    GrProductsGridComponent,
    PrAddToCardButtonComponent,
```

```
    GrGenresMenuComponent,
    GrFilterRowComponent,
    GrMainTitleComponent,
    ProductPageComponent,
    GrGameInfoComponent,
    PrSignInComponent,
    PrLogOutComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    AngularFireModule.initializeApp(environment.firebase),
    AngularFireDatabaseModule,
    AngularFireAuthModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

App.component.html

```
<div class="cursor"></div>
<div class="cursor2"></div>
<!-- <div class="backGround"></div> -->
<app-gr-header></app-gr-header>
<!-- <app-home-page></app-home-page> -->
<!-- <app-store-page></app-store-page> -->


<router-outlet></router-outlet>
```

Home-page.component.html

```
<div class="main" [@slideInOut]>
  <div class="home">
    <!-- <div class="video"></div> -->
    <div id="large-header" class="large-header video">
      <canvas id="demo-canvas"></canvas>
    </div>
    <div class="container">
       <!-- <div class="left">
          <div class="splash">
            <h1>Music Store</h1>
            <p class="intro">The best destination to buy new records to competitive prices. 24 hour
support, "best price" guarantee and a flawless UX. Wish for more? Tell us <span
class="here">below</span> — or check out our <span class="careers">careers.</span></p>
```

```
        </div>

        <div class="buttons">
            <app-pr-cta-button [text]="'Browse'"  [svgName]="'notfound'"></app-pr-cta-button>
            <app-pr-cta-button [text]="'Play Dice'"  [svgName]="'notfound'"></app-pr-cta-button>
            <app-pr-cta-button [text]="'GitHub'"  [svgName]="'notfound'"></app-pr-cta-button>
            <app-pr-cta-button [text]="'LinkedIn'"  [svgName]="'notfound'"></app-pr-cta-button>
        </div>
      </div> -->
      <app-gr-main-title></app-gr-main-title>
      <app-gr-navigation></app-gr-navigation>
   </div>
  </div>
</div>
```

Home-page.component.scss

```
#large-header {
   background-image: url('https://wallpaperaccess.com/full/2616605.jpg');
}
.main-title {
   position: absolute;
   margin: 0;
   padding: 0;
   color: #f9f1e9;
   text-align: center;
   top: 50%;
   left: 50%;
   -webkit-transform: translate3d(-50%,-50%,0);
   transform: translate3d(-50%,-50%,0);
}
.demo-1 .main-title {
   text-transform: uppercase;
   font-size: 4.2em;
   letter-spacing: 0.1em;
}

.main-title .thin {
   font-weight: 200;
}
@media only screen and (max-width : 768px) {
   .demo-1 .main-title {
      font-size: 3em;
   }
}
   >.main{
      max-width: 1440px;
      margin: auto
   }
```

```
* {
    margin: 0;
    padding: 0;
}
:host{
    background: tan;
}
.overlap {
    position: absolute;
    width: 100%;
    height: 100%;
    background-color: #0f1011;
    z-index: 10;
}
html,
body {
    margin: 0;
    width: 100vw;
}
.parent * {
    pointer-events: none;
}
.container {
    display: flex;
    justify-content: space-between;
    align-items: center;
    height: calc(100vh - 76px);
}
.careers {
    color: #45c1bc;
    font-family: "GT Medium";
    cursor: pointer;
}
a {
    text-decoration: none;
}
.left,
.right {
    width: 50vw;
    height: 90vh;
    display: flex;
    flex-direction: column;
    justify-content: center;
}
.right {
    align-items: flex-end;
}
.left .buttons .browseBtn {
    background-color: #45c1bc;
```

```
}
.buttonsRight {
    margin-right: 40px;
    position: relative;
    border-radius: 20px;
    padding: 22px 30px 30px 30px;
    box-sizing: border-box;
    gap: 5px;
    display: flex;
    flex-direction: column;
    backdrop-filter: blur(6px);
}
.buttonsRight button {
    width: 100%;
    justify-content: flex-start;
    padding-left: 48px;
    gap: 15px;
}
.buttonsRight h2 {
    font-size: 26px;
    margin-bottom: 14px;
}
.technologies {
    height: 18px;
    width: 18px;
}
.video {
    position: fixed;
    right: 0;
    bottom: 0;
    min-width: 100%;
    min-height: 100%;
    z-index: -1;
    background-attachment: fixed;
    //background: url("../../../../assets/image/photo_2023-04-06_21-49-32.jpg");
    background: black;
    background-repeat: no-repeat;
    background-position: center;
    background-size: cover;
  }
.splash {
    display: flex;
    flex-direction: column;
    position: relative;
    width: 630px;
    height: 240px;
    box-sizing: border-box;
    padding: 22px 35px 55px 50px;
    margin-left: 44px;
```

```
    border-radius: 30px;
    z-index: 6;
    backdrop-filter: blur(8px);
}
.buttons {
    display: flex;
    position: relative;
    width: 630px;
    height: 70px;
    box-sizing: border-box;
    padding: 14px 20px 15px 20px;
    margin-left: 44px;
    margin-top: 15px;
    border-radius: 30px;
    z-index: 6;
    backdrop-filter: blur(8px);
    justify-content: space-between;
}
.cta {
    background-color: #fff;
    border: 1px transparent;
    border-radius: 18px;
    color: black;
    padding: 10px 28px;
    display: flex;
    gap: 12px;
    font-family: "GT Medium";
    font-size: 16px;
    transition: 0.3s all;
    cursor: pointer;
}

.cta:hover {
    transform: scale(1.03);
    background-color: #45c1bc;
}
.cta * {
    pointer-events: none;
}
.cta .linkedin {
    padding-bottom: 1px;
}
.lastChild {
    gap: 10px;
}
.cta span {
    padding-top: 1px;
}
.ctaSVG {
```

```
    height: 18px;
    width: 18px;
}
.buttons::after,
.splash::after,
.buttonsRight::after {
    content: "";
    position: absolute;
    border-radius: 30px;
    inset: 0.01%;
    height: inherit;
    width: inherit;
    z-index: -1;
    background-color: white;
    opacity: 0.285;
}
.intro {
    color: white;
    padding-left: 5px;
    font-family: "GT Regular";
    font-size: 19px;
    opacity: 1;
}
splash h1 {
    font-family: "GT Bold";
    font-size: 90px;
    color: white;
}
.home h2 {
    color: white;
    font-family: "GT Bold";
}
@media only screen and (max-width: 1000px) {
    .container {
        display: flex;
        flex-direction: column;
        justify-content: flex-start;
        align-items: center;
    }
    .left,
    .right {
        width: auto;
        height: auto;
        margin-top: 100px;
    }
    .buttonsRight,
    .buttons,
    .splash {
        margin-right: 0px;
```

```
        margin-left: 0px;
    }
}
@media screen and (max-width: 700px) {
    .left,
    .splash {
        width: 450px;
    }
    .right {
        margin-top: 20px;
        margin-bottom: 100px;
    }
    .buttons {
        width: 276px;
        height: 190px;
        align-self: center;
        flex-direction: column;
    }
    .buttons button {
        width: 100%;
        gap: 20px;
        padding-left: 75px;
    }
    .splash {
        height: 225px;
    }
    .splash h1 {
        font-size: 56px;
    }
}
@media screen and (max-width: 500px) {
    .left,
    .splash {
        width: 276px;
    }
    .splash {
        padding: 10px 35px 10px 35px;
        height: 60px;
        justify-content: center;
    }
    .splash h1 {
        font-size: 36px;
    }
    .splash p {
        display: none;
    }
}
}
```

Home-page.component.ts

```
import { Component, OnInit } from '@angular/core';
// import {myFunction} from '../../../../assets/utils/canvas.js'
import particleAnimation from './particleAnimation.js';
import  {Cursor}  from 'src/assets/utils/cursor.js';
import { slideInOut } from 'src/assets/animations/animation.js';

@Component({
  selector: 'app-home-page',
  templateUrl: './home-page.component.html',
  styleUrls: ['./home-page.component.scss'],
  animations: [slideInOut]
})
export class HomePageComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
      particleAnimation();
      Cursor();
  }

}
```

Моделі

Record.js

```
const mongoose = require('mongoose');
// Створення схеми моделі
const vinylSchema = new mongoose.Schema({

  title: {
    type: String,
    required: true
  },
  artist: {
    type: String,
    required: true
  },
  genre: {
    type: String,
    required: true
  },
  price: {
    type: Number,
    required: true
  },
  quantity: {
    type: Number,
```

```
    default: 0
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});
// Створення моделі на основі схеми
const Vinyl = mongoose.model('Vinyl', vinylSchema);

// Експорт моделі для використання в інших модулях
module.exports = Vinyl;
```

Order.js

```
const mongoose = require('mongoose');
// Схема моделі Order
const orderSchema = new mongoose.Schema({
  customerName: {
    type: String,
    required: true
  },
  email: {
    type: String,
    required: true
  },
  address: {
    type: String,
    required: true
  },
  phone: {
    type: String,
    required: true
  },
  products: [
    {
      name: {
        type: String,
        required: true
      },
      price: {
        type: Number,
        required: true
      },
      quantity: {
        type: Number,
        required: true
      }
    }
  ],
```

```
  totalAmount: {
    type: Number,
    required: true
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});
// Створення моделі Order на основі схеми
const Order = mongoose.model('Order', orderSchema);
// Експорт моделі Order
module.exports = Order;
```

User.js

```
const mongoose = require('mongoose');
// Схема моделі User
const userSchema = new mongoose.Schema({
  username: {
    type: String,
    required: true,
    unique: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  },
  isAdmin: {
    type: Boolean,
    default: false
  },
  createdAt: {
    type: Date,
    default: Date.now
  }
});
// Створення моделі User на основі схеми
const User = mongoose.model('User', userSchema);
// Експорт моделі User
module.exports = User;
```

App-gr-header.component.html

```
<header class="header">
   <div class="left-side">
      <a routerLink="" routerLinkActive="active" ariaCurrentWhenActive="page">
         <app-pr-ghost-button [text]="'Music Store'" [svgName]="'record'"></app-pr-ghost-button>
      </a>
      <a routerLink="/store-page" routerLinkActive="active" ariaCurrentWhenActive="page">
         <app-pr-ghost-button [text]="'Browse Store'" [svgName]="'store'"></app-pr-ghost-button>
      </a>
   </div>
   <div class="right-side">
      <app-pr-sign-in></app-pr-sign-in>
      <app-pr-ghost-button [text]="'Basket'" [svgName]="'basket'"></app-pr-ghost-button>
   </div>
</header>
```

App-gr-header.component.ts

```
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'app-gr-header',
  templateUrl: './gr-header.component.html',
  styleUrls: ['./gr-header.component.scss']
})
export class GrHeaderComponent implements OnInit {
  constructor() { }
  ngOnInit(): void {
  }
}
```

App-gr-products-grid.component.ts

```
import { ChangeDetectorRef, Component, OnInit } from '@angular/core';
import { Subscription } from 'rxjs';
import { ProductService } from 'src/app/services/product.service';
import games from 'src/data/records';
import { cloneDeep } from 'lodash';
@Component({
  selector: 'app-gr-products-grid',
  templateUrl: './gr-products-grid.component.html',
  styleUrls: ['./gr-products-grid.component.scss']
})
export class GrProductsGridComponent implements OnInit {

  gamesList: any = [...games];
  filteredGames: any = [...this.gamesList];

  //private productService: ProductService;
```

```
  selectedItemSubscriber: Subscription = new Subscription();

  constructor(
    private productService: ProductService
    //productService: ProductService
  ) {
    this.productService = productService

  }
  ngOnInit(): void {
    this.selectedItemSubscriber = this.productService.subject.subscribe(config => {
      console.log(config);
      if(config){
        this.filteredGames = cloneDeep(this.gamesList).filter((v: { genre: string }) =>
v.genre.toLowerCase() === config.name.toLowerCase());
      } else if(config === null) {
        this.filteredGames = cloneDeep(this.gamesList);
      }
      console.log("games",this.filteredGames);
    });
  }
}
```

App-gr-genres.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { GrGenresMenuComponent } from './gr-genres-menu.component';
import { ProductService } from 'src/app/services/product.service';
import { BehaviorSubject } from 'rxjs';
describe('GrGenresMenuComponent', () => {
  let component: GrGenresMenuComponent;
  let fixture: ComponentFixture<GrGenresMenuComponent>;
  let productService: ProductService;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [GrGenresMenuComponent],
      providers: [ProductService]
    })
      .compileComponents();
  });
  beforeEach(() => {
    fixture = TestBed.createComponent(GrGenresMenuComponent);
    component = fixture.componentInstance;
    productService = TestBed.inject(ProductService);
    fixture.detectChanges();
  });
  it('повинен створити компонент', () => {
    expect(component).toBeTruthy();
  });
```

```
  it('повинен емітувати вибраний жанр у методі selectGenre', () => {
    spyOn(productService.subject, 'next');
    const genre = { name: 'Rock', icon: 'assets/image/action.svg' };
    component.selectGenre(genre);
    expect(productService.subject.next).toHaveBeenCalledWith(genre);
  });
});
```

Utils.js

```
export function Cursor() {
    var cursor = document.querySelector('.cursor');
    var cursorinner = document.querySelector('.cursor2');
    var a = document.querySelectorAll('a');
    document.addEventListener('mousemove', function(e){
    var x = e.clientX;
    var y = e.clientY;
    cursor.style.transform = `translate3d(calc(${e.clientX}px - 50%), calc(${e.clientY}px - 50%), 0)`
    });
    document.addEventListener('mousemove', function(e){
    var x = e.clientX;
    var y = e.clientY;
    cursorinner.style.left = x + 'px';
    cursorinner.style.top = y + 'px';
    });
    document.addEventListener('mousedown', function(){
    cursor.classList.add('click');
    cursorinner.classList.add('cursorinnerhover')
    });
    document.addEventListener('mouseup', function(){
    cursor.classList.remove('click')
    cursorinner.classList.remove('cursorinnerhover')
    });
    a.forEach(item => {
    item.addEventListener('mouseover', () => {
        cursor.classList.add('hover');
    });
    item.addEventListener('mouseleave', () => {
        cursor.classList.remove('hover');
    });
    })
}
```

ParticlerAnimation.js

```
export default function particleAnimation() {
    var width, height, largeHeader, canvas, ctx, points, target, animateHeader = true;
    // Main
    initHeader();
    initAnimation();
```

```
addListeners();

function initHeader() {
    width = window.innerWidth;
    height = window.innerHeight;
    target = {x: width/2, y: height/2};
    largeHeader = document.getElementById('large-header');
    largeHeader.style.height = height+'px';
    canvas = document.getElementById('demo-canvas');
    canvas.width = width;
    canvas.height = height;
    ctx = canvas.getContext('2d');
    // create points
    points = [];
    for(var x = 0; x < width; x = x + width/20) {
        for(var y = 0; y < height; y = y + height/20) {
            var px = x + Math.random()*width/20;
            var py = y + Math.random()*height/20;
            var p = {x: px, originX: px, y: py, originY: py };
            points.push(p);
        }
    }
    // for each point find the 5 closest points
    for(var i = 0; i < points.length; i++) {
        var closest = [];
        var p1 = points[i];
        for(var j = 0; j < points.length; j++) {
            var p2 = points[j]
            if(!(p1 == p2)) {
                var placed = false;
                for(var k = 0; k < 5; k++) {
                    if(!placed) {
                        if(closest[k] == undefined) {
                            closest[k] = p2;
                            placed = true;
                        }
                    }
                }

                for(var k = 0; k < 5; k++) {
                    if(!placed) {
                        if(getDistance(p1, p2) < getDistance(p1, closest[k])) {
                            closest[k] = p2;
                            placed = true;
                        }
                    }
                }
            }
        }
```

```
            p1.closest = closest;
        }
        // assign a circle to each point
        for(var i in points) {
            var c = new Circle(points[i], 2+Math.random()*2, 'rgba(255,255,255,0.3)');
            points[i].circle = c;
        }
    }
    // Event handling
    function addListeners() {
        if(!('ontouchstart' in window)) {
            window.addEventListener('mousemove', mouseMove);
        }
        window.addEventListener('scroll', scrollCheck);
        window.addEventListener('resize', resize);
    }
    function mouseMove(e) {
      // console.log(e);
        var posx = 0, posy = 0;
        if (e.pageX || e.pageY) {
            posx = e.pageX;
            posy = e.pageY;
        }
        else if (e.clientX || e.clientY)    {
            posx = e.clientX + document.body.scrollLeft + document.documentElement.scrollLeft;
            posy = e.clientY + document.body.scrollTop + document.documentElement.scrollTop;
        }
        target.x = posx;
        target.y = posy;
    }
    function scrollCheck() {
        if(document.body.scrollTop > height) animateHeader = false;
        else animateHeader = true;
    }
    function resize() {
        width = window.innerWidth;
        height = window.innerHeight;
        largeHeader.style.height = height+'px';
        canvas.width = width;
        canvas.height = height;
    }
    // animation
    function initAnimation() {
        animate();
        for(var i in points) {
            shiftPoint(points[i]);
        }
    }
    function animate() {
```

```
    if(animateHeader) {
        ctx.clearRect(0,0,width,height);
        for(var i in points) {
            // detect points in range
            if(Math.abs(getDistance(target, points[i])) < 4000) {
                points[i].active = 0.3;
                points[i].circle.active = 0.6;
            } else if(Math.abs(getDistance(target, points[i])) < 20000) {
                points[i].active = 0.1;
                points[i].circle.active = 0.3;
            } else if(Math.abs(getDistance(target, points[i])) < 40000) {
                points[i].active = 0.02;
                points[i].circle.active = 0.1;
            } else {
                points[i].active = 0;
                points[i].circle.active = 0;
            }
            drawLines(points[i]);
            points[i].circle.draw();
        }
    }
    requestAnimationFrame(animate);
}

function shiftPoint(p) {
    TweenLite.to(p, 1+1*Math.random(), {x:p.originX-50+Math.random()*100,
        y: p.originY-50+Math.random()*100, ease:Circ.easeInOut,
        onComplete: function() {
            shiftPoint(p);
        }});
}

// Canvas manipulation
function drawLines(p) {
    if(!p.active) return;
    for(var i in p.closest) {
        ctx.beginPath();
        ctx.moveTo(p.x, p.y);
        ctx.lineTo(p.closest[i].x, p.closest[i].y);
        ctx.strokeStyle = 'rgba(156,217,249,'+ p.active+')';
        ctx.stroke();
    }
}

function Circle(pos,rad,color) {
    var _this = this;

    // constructor
    (function() {
```

```
            _this.pos = pos || null;
            _this.radius = rad || null;
            _this.color = color || null;
        })();

        this.draw = function() {
            if(!_this.active) return;
            ctx.beginPath();
            ctx.arc(_this.pos.x, _this.pos.y, _this.radius, 0, 2 * Math.PI, false);
            ctx.fillStyle = 'rgba(156,217,249,'+ _this.active+')';
            ctx.fill();
        };
    }

    // Util
    function getDistance(p1, p2) {
        return Math.pow(p1.x - p2.x, 2) + Math.pow(p1.y - p2.y, 2);
    }

}
```

ДОДАТОК Г
(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

LARANA, INC.

Кваліфікаційна робота на тему

"Веб-застосунок для продажу музичних платівок"

Виконав студент групи ІПЗС20-1
Войченко Роман

Виконав студент: Войченко Роман Олександрович
Керівник: Бедратюк Л.П. д-р фіз.-мат. наук, проф.

Актуальність теми

Веб-магазини стали необхідною складовою сучасного електронного бізнесу, який забезпечує зручність та доступність для покупців у будь-який час та місце. У рамках даної дипломної роботи було розроблено веб-магазин для продажу музичних платівок.

Метою проекту є створення функціонального та зручного інтерфейсу для замовлення та оплати музичних платівок в мережі Інтернет. Крім того, проект передбачає розробку системи адміністрування, що дозволить додавати нові платівки, відслідковувати стан замовлень та керувати іншими аспектами магазину.

Однією з головних задач проекту є покращення зручності та швидкості процесу замовлення музичних платівок. Завдяки веб-магазину покупець зможе швидко знайти потрібний альбом, дізнатися інформацію про його наявність та ціну, вибрати зручний спосіб доставки та оплати.

Також, веб-магазин дозволить вирішити проблему обмеженої кількості точок продажу музичних платівок та відсутності можливості придбати певний альбом в окремих регіонах. Даний проект вирішує ці проблеми шляхом створення централізованого магазину, доступного для покупців з будь-якої точки світу.

Мета та завдання проекту

При розробці дипломного проекту "Веб-магазин для продажу музичних платівок" потрібно вирішити такі задачі:
1)Розробити зручний та простий інтерфейс користувача, щоб покупці могли швидко та легко знайти потрібні платівки, дізнатися про їх характеристики та ціну.
2)Забезпечити безпеку платежів, щоб клієнти могли здійснювати оплату з використанням різних методів, таких як банківські карти, електронні гроші, PayPal тощо.
3)Розробити систему замовлень та доставки, щоб клієнти могли обрати зручний спосіб доставки та знати про стан свого замовлення.
4)Забезпечити можливість додавання нових товарів та оновлення інформації про наявність та ціни товарів, щоб магазин завжди мав актуальну інформацію.
5)Розробити систему адміністрування магазину, щоб адміністратори могли керувати замовленнями, товаром, налаштуваннями магазину та іншими аспектами роботи магазину.
6)Забезпечити оптимізацію сайту для пошукових систем, щоб магазин мав хорошу видимість в пошукових системах та залучав більше клієнтів.
7)Забезпечити адаптивність сайту для різних пристроїв, щоб користувачі могли зручно користуватися сайтом на мобільних пристроях, планшетах та комп'ютерах.
8)Забезпечити захист персональних даних клієнтів та відвідувачів магазину від несанкціонованого доступу та злочинних дій.

1 Дослідження предметної області та постановка задачі

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Веб-магазин з продажу вінілових платівок належить до сфери електронної комерції та музичної індустрії. Він спрямований на надання інтернет-користувачам можливості придбати вінілові платівки з різних жанрів та виконавців. Огляд предметної області включає аналіз його ключових аспектів.
Досліджується стан ринку вінілових платівок, його тенденції, зростання популярності цього формату звукозапису, аналіз конкуренції та інших факторів, що впливають на ринкові умови. Вивчається цільова аудиторія веб-магазину, включаючи музичних ентузіастів, колекціонерів, діджеїв та інших зацікавлених осіб. Аналізуються їхні потреби, вподобання, звички покупок та очікування від веб-магазину. Встановлюються основні вимоги та виклики, що виникають при продажу вінілових платівок через веб-магазин, включаючи управління запасами, доставку.

Angular-аналіз обраних рішень

Плюси:

- **Можливість чітко групувати елементи, використовуючи модулі.**
- **Є повноцінним фреймворком, що має все необхідне для написання сучасних вебзастосунків і навіть більше.**
- **Підтримка Typescript «з коробки». По суті фреймворк є побудованим з прицілом на підтримку Typescript, як однієї з головних ідей.**
- **Використання MVVM принципів побудови застосунків, що дає змогу чітко розмежовувати реалізацію бізнес-логіки від її представлення.**
- **Завдяки наявності модулів — чудово масштабується, якщо попередньо чітко планувати структуру застосунку й слідувати їй.**
- **Dependency Injection — чудовий спосіб передачі лише необхідних залежностей (у вигляді сервісів), у компоненти й інші елементи Angular. Особливо розквітає, коли починаєте писати всілякого роду тести для ваших компонентів, бо дозволяє легко підмінити будь-яку з цих залежностей необхідною вам імплементацією.**
- **Документація актуальна і вичерпна, в плані доступних класів, методів і типів. Також містить масу додаткової інформації, зокрема такої, як:**
  - **Angular coding style guide — офіційний гайд по рекомендованому синтаксису, конвенціях, структурі застосунку.**
  - **Best Practices, як рекомендації з написання Безпечних і Доступних застосунків, а також підходів для розбиття коду на множину окремих бандлів — Lazy-loading feature modules.**
  - **І багато чого ще, з повним переліком доступної інформації можна ознайомитись тут.**
- **Кількість інформації, доступної в інтернеті щодо будь-якого аспекту роботи, чи певної бібліотеки.**
- **Велике ком'юніті.**
- **Широкий вибір бібліотек під будь-яку задачу і смак.**
- **Регулярно оновлюється і вдосконалюється.**
- **Дуже просунуте CLI, котре при цьому також піддається деякій кастомізації (для прикладу Blueprints, для кастомізації генерації компонентів та інших елементів і файлів).**
- **Підтримується Google.**

Мінуси:

- **Оскільки структура передбачає доволі великий обсяг знань, яким необхідно володіти аби почати писати на Angular, крива входу є доволі крутою. Як наслідок, мінусом є висока складність, порівняно з іншими фреймворками й бібліотеками.**
- **В документації подекуди катастрофічно бракує прикладів використання того чи іншого функціоналу, оскільки знання його сигнатури буває недостатньо.**
- **Офіційна документація доступна лише сімома мовами. Українська в їх переліку відсутня.**
- **Інколи низька швидкість роботи, яку дуже просто «зіпсувати». Але при цьому, якщо знати про те, як працює Angular Change Detection і як в цілому оптимізувати швидкість роботи й завантаження вебзастосунків, стає більш ніж конкурентною.**
- **Також мінусом, на мою думку, є пряме мутування властивостей класу для зміни або оновлення стану компонента.**

Таблиця варіантів використання системи

Use case UML діаграма вінілових платівок може включати наступні елементи:

1. **Користувач:**
   - **Переглядати список доступних платівок**
   - **Пошукати платівки за жанром, виконавцем або назвою**
   - **Додавати платівки до кошика**
   - **Видаляти платівки з кошика**
   - **Оформляти замовлення**
   - **Здійснювати оплату**
   - **Отримувати підтвердження замовлення**
2. **Адміністратор:**
   - **Додавати нові платівки до магазину**
   - **Видаляти платівки з магазину**
   - **Редагувати інформацію про платівки (жанр, виконавець, назва, ціна тощо)**
   - **Керувати замовленнями (підтверджувати, відміняти, відправляти)**
3. **Система:**
   - **Відображення списку доступних платівок**
   - **Пошук платівок за жанром, виконавцем або назвою**
   - **Оновлення інформації про наявність платівок**
   - **Розрахунок суми замовлення**
   - **Підтвердження та відправлення підтвердження замовлення**
   - **Видалення замовлення**
   - **Оповіщення адміністратора про нове замовлення**

Ця текстова інформація описує основні акторів та їх можливі взаємодії з системою у контексті онлайн магазину з продажу вінілових платівок.

2. Проектування Програмного забезпечення

Архітектура Angular проекту для веб-магазину з продажу музичних платівок може бути розроблена з використанням таких ключових компонентів та підходів:

1.Компоненти:

Головний компонент (App Component): Це основний компонент, який включає інші компоненти та управляє загальною структурою сторінки.

Компоненти для відображення списку платівок, деталей платівок, кошика та інших важливих функціональних елементів магазину.

Компоненти для форм (наприклад, реєстрація, оформлення замовлення), які взаємодіють з користувачем для введення даних та виконання дій.

2.Сервіси:

Сервіси для отримання та обробки даних про платівки з сервера або з бази даних, такі як сервіс для отримання списку платівок, сервіс для додавання до кошика тощо.

Сервіси для авторизації та аутентифікації користувачів.

Сервіси для керування станом додатку, наприклад, сервіс для управління кошиком, додаванням і видаленням платівок тощо.

3.Модулі:

Основний модуль (App Module): Це головний модуль, який об'єднує всі компоненти, сервіси та інші модулі додатку.

Модулі для реєстрації, авторизації та інших функціональних блоків додатку.

4.Маршрутизація.

2.2 Опис декомпозиції

Виберемо модульну декомпозицію для проекту веб-магазину вінілових платівок. Описатимемо основні модулі, які можуть бути частиною системи:

1. **Модуль Каталогу:**
   - **Відповідає за відображення списку доступних платівок**
   - **Містить функціонал пошуку платівок за жанром, виконавцем або назвою**
2. **Модуль Кошика:**
   - **Відповідає за додавання та видалення платівок з кошика користувача**
   - **Зберігає обрані платівки між сеансами користувача**
3. **Модуль Замовлення:**
   - **Обробляє оформлення замовлення користувачем**
   - **Включає функціонал розрахунку суми замовлення та оплати**
4. **Модуль Адміністрування:**
   - **Доступний тільки для адміністраторів**
   - **Містить функціонал додавання, видалення та редагування платівок у магазині**
   - **Управління замовленнями, включаючи підтвердження, відміну та відправлення замовлення**
5. **Модуль Аутентифікації та Авторизації:**
   - **Забезпечує безпечний доступ до функціоналу для зареєстрованих користувачів**
   - **Включає функції реєстрації, входу та управління обліковими записами**
6. **Модуль Оповіщення:**
   - **Відповідає за надсилання повідомлень користувачам та адміністраторам про замовлення, статуси тощо**
7. **Модуль Бази Даних:**
   - **Забезпечує зберігання та доступ до інформації про платівки, замовлення, користувачів тощо**

Кожен з цих модулів може бути реалізований як окремий компонент системи зі своїми функціями та взаємодіями з іншими модулями

2.3 Опис залежностей

У проекті веб-магазину вінілових платівок можуть існувати наступні типи залежностей:

1. **Міжмодульні залежності:**
   - **Модуль Каталогу може залежати від модуля Бази Даних для отримання інформації про наявні платівки.**
   - **Модуль Кошика залежить від модулів Каталогу та Бази Даних для додавання та видалення платівок з кошика.**
   - **Модуль Замовлення залежить від модуля Кошика для отримання вибраних платівок та від модулів Аутентифікації та Бази Даних для отримання даних користувача та збереження замовлення.**
2. **Міжпроцесні залежності:**
   - **Процес оформлення замовлення залежить від процесу авторизації користувача та процесу обробки платежу.**
3. **Залежності всередині даних:**
   - **Дані про платівки та замовлення залежать від даних, збережених у базі даних.**
4. **Залежності між станами:**
   - **Стан замовлення може змінюватись від "Нове" до "Підтверджене" до "Відправлене" в залежності від взаємодії з модулями Адміністрування та Аутентифікації.**
5. **Міжрівневі залежності:**
   - **Модуль Клієнтського інтерфейсу може залежати від модулів Каталогу та Коши**

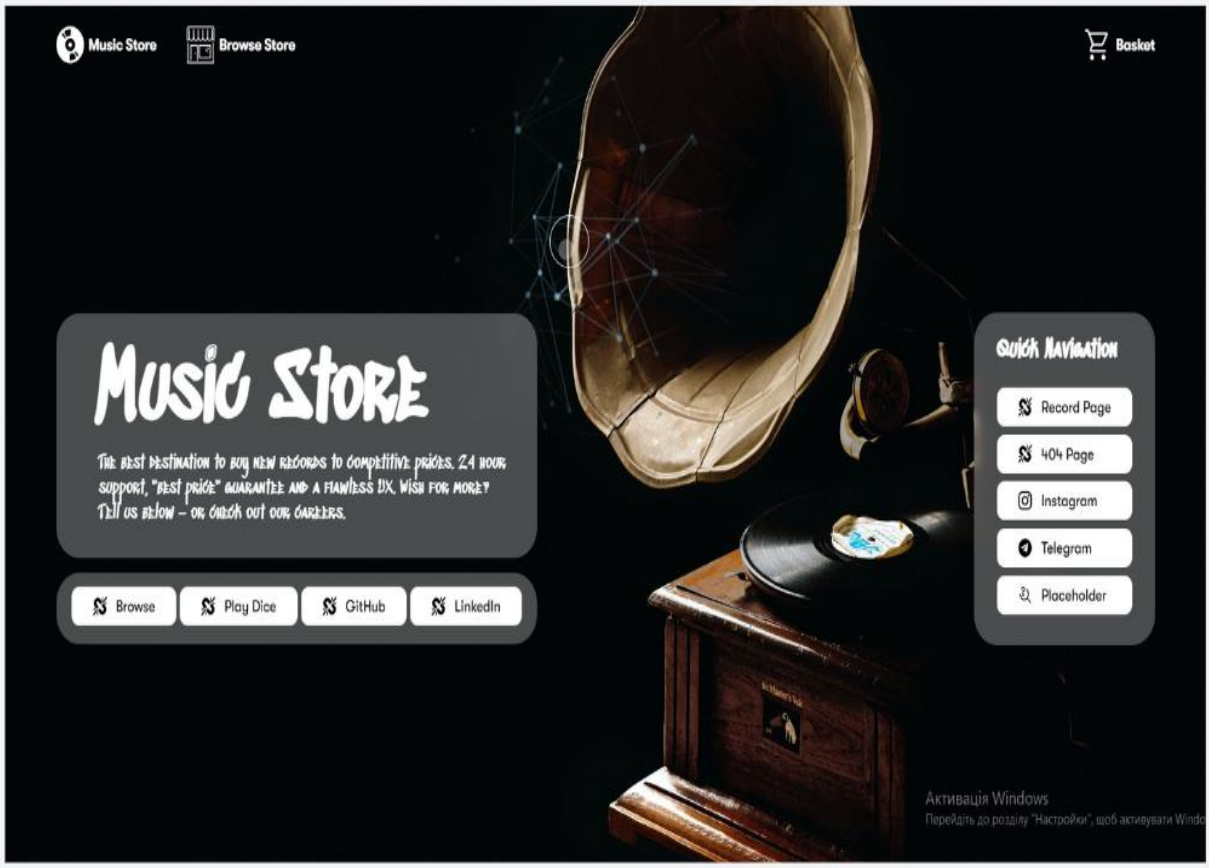
3 Програмна реалізація

3.1 Детальне проектування модулів

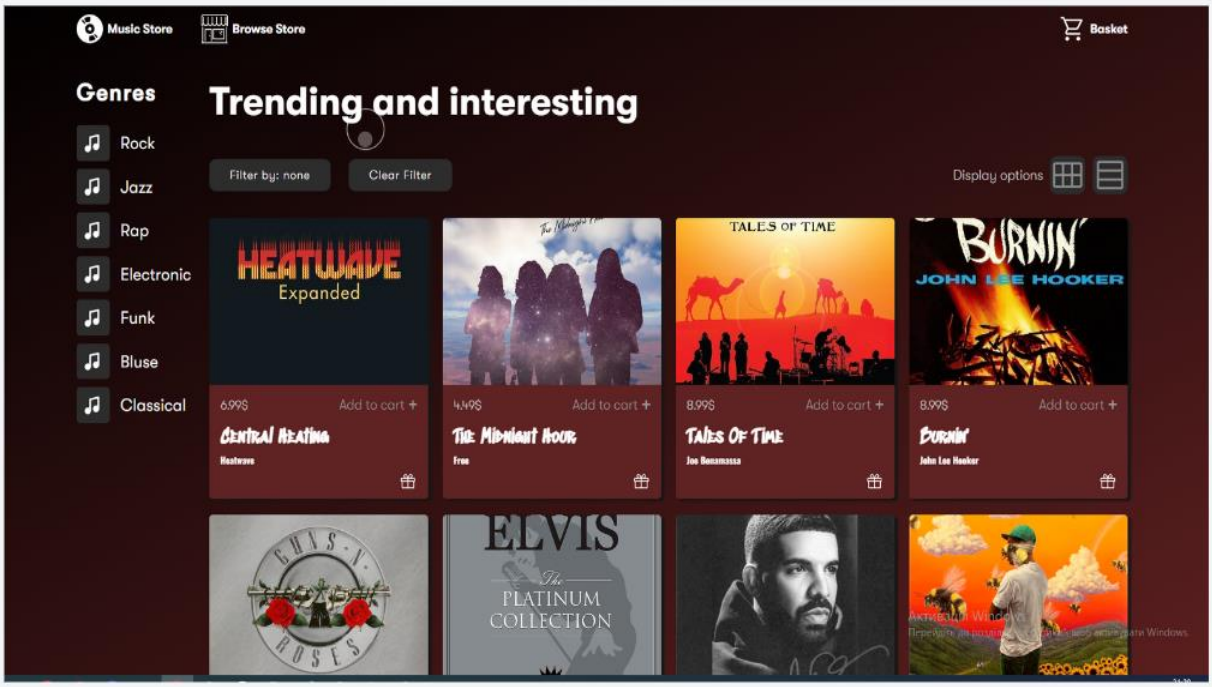
У детальному проектуванні модулів веб-магазину вінілових платівок використовуються такі структурні елементи:

1. **Модуль Каталогу:**
   - **Клас Vinyl: містить властивості платівки, такі як назва, виконавець, жанр, ціна тощо.**
   - **Клас Catalog: містить методи для отримання списку платівок з бази даних, фільтрації за жанром або виконавцем.**
2. **Модуль Кошика:**
   - **Клас Cart: представляє кошик користувача та містить методи для додавання, видалення та отримання платівок з кошика.**
   - **Клас CartItem: представляє окрему платівку в кошику з відповідними властивостями, такими як кількість та ціна.**
3. **Модуль Замовлення:**
   - **Клас Order: представляє замовлення користувача та містить властивості, такі як список платівок, адреса доставки, статус тощо.**
   - **Клас OrderManager: містить методи для обробки замовлення, розрахунку суми, збереження замовлення у базі даних.**
4. **Модуль Адміністрування:**
   - **Клас AdminPanel: надає інтерфейс адміністратора для додавання, редагування та видалення платівок з магазину.**
   - **Клас OrderManagement: містить методи для керування замовленнями, такі як підтвердження, відміна, відправлення замовлення.**
5. **Модуль Аутентифікації та Авторизації:**
   - **Клас User: містить дані користувача, такі як ім'я, електронна пошта, пароль тощо.**
   - **Клас AuthManager: містить методи для реєстрації, входу та управління обліковими записами користувачів.**

3.1 Детальне Проектування модулів. Вигляд модуля головної сторінки додатку



3.1 Вигляд модуля каталогу платівок



3.1 Вигляд модуля кошика

Модуль Кошика:

Клас Cart: представляє кошик користувача та містить методи для додавання, видалення та отримання платівок з кошика.
Клас CartItem: представляє окрему платівку в кошику з відповідними властивостями, такими як кількість та ціна.

3.1 Вигляд модуля картки окремого товару

Модуль Замовлення:

Клас Order: представляє замовлення користувача та містить властивості, такі як список платівок, адреса доставки, статус тощо.
Клас OrderManager: містить методи для обробки замовлення, розрахунку суми, збереження замовлення у базі даних.

4 Тестування

Розділ верифікації та валідації ПЗ для веб-магазину вінілових платівок включає наступні етапи:

1. **Обґрунтування вибору методів тестування:**
   - **Враховуючи характеристики проекту, такі як інтерфейс користувача, функціональність, безпека, варто використовувати комбінацію методів тестування, включаючи модульні тести, інтеграційні тести, системні тести та регресійні тести.**
   - **Важливо також провести тестування продуктивності для переконанняся, що система може обробляти навантаження в реальному часі.**
2. **Розробка тестових наборів даних:**
   - **Створення реалістичних тестових наборів даних для перевірки функціональності та коректності роботи системи.**
   - **Тестові набори мають включати різноманітні варіації платівок, замовлень, користувачів та сценаріїв взаємодії з системою.**
3. **Аналіз результатів тестування:**
   - **Оцінка відповідності результатів тестування вимогам та очікуванням.**
   - **Виявлення дефектів, помилок та проблем, що виникають під час тестування.**
   - **Оцінка покриття тестами різних функціональних та нефункціональних аспектів системи.**

Узагальнюючі висновки:

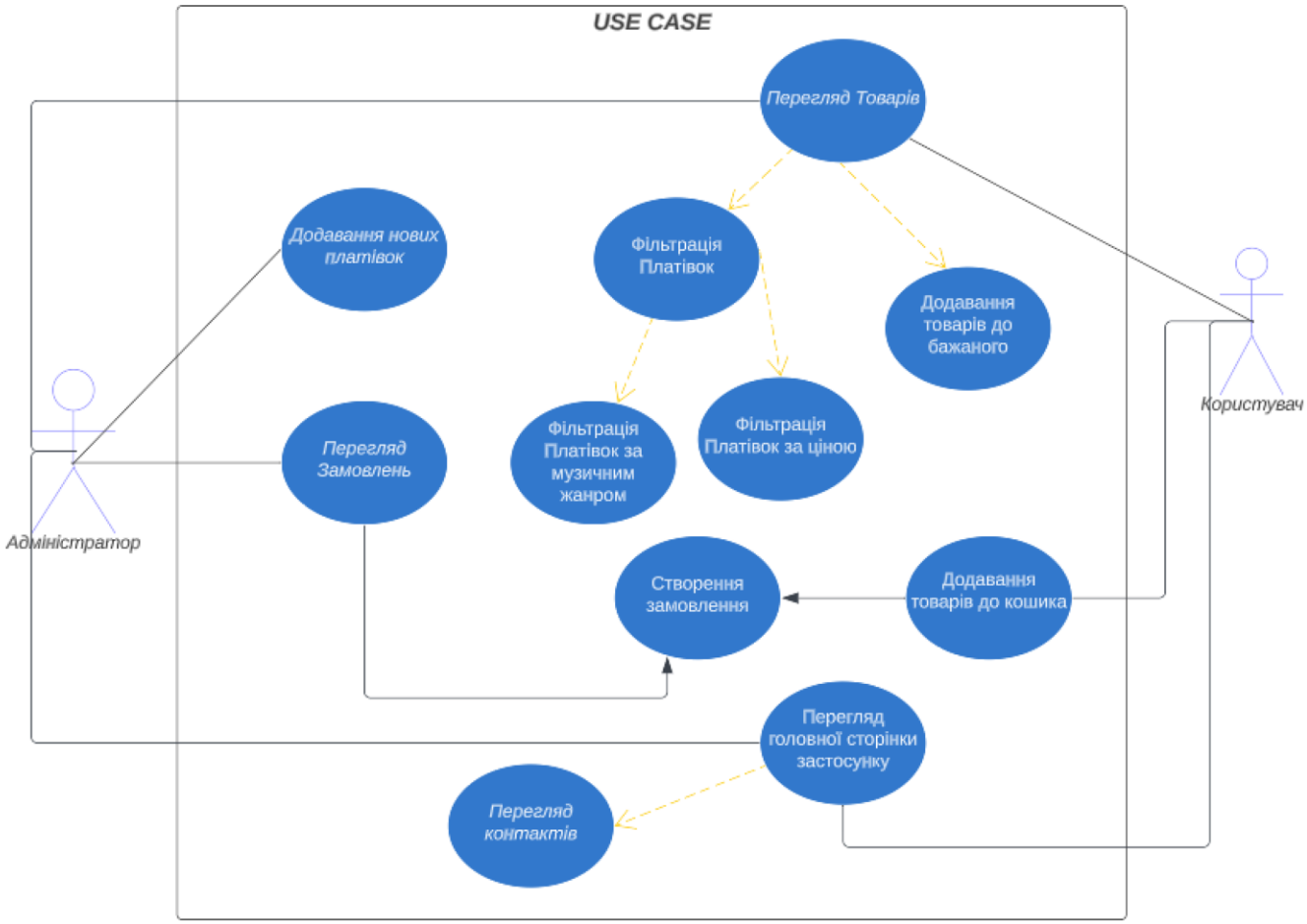
- **Використання різних методів тестування дозволило забезпечити високу якість та надійність системи.**
- **Розроблені тестові набори даних дозволили виявити і виправити помилки та недоліки у роботі системи.**
- **Результати тестування свідчать про задовільне функціонування веб-магазину вінілових платівок.**

Висновки:

1. **У рамках проекту був розроблений функціональний веб-магазин вінілових платівок, що дозволяє користувачам переглядати та замовляти продукцію.**
2. **Були використані сучасні технології та інструменти для розробки, такі як HTML, CSS, JavaScript для фронтенду.**
3. **Реалізовано основні функції, такі як перегляд каталогу вінілових платівок, додавання товарів до кошика, оформлення замовлення та аутентифікація користувачів.**
4. **Було проведено тестування системи з використанням реалістичних тестових наборів даних та різних методів тестування.**
5. **В результаті тестування виявлено і виправлено деякі помилки та недоліки у функціонуванні системи.**
6. **Веб-магазин продемонстрував задовільну продуктивність під час тестування.**
7. **Застосування принципів дизайну та інтуїтивно зрозумілий інтерфейс користувача забезпечують зручну навігацію та зручність використання системи.**
8. **Результати проекту підтверджують досягнення поставлених цілей та вимог до функціональності та надійності веб-магазину вінілових платівок.**
9. **Для подальшого розвитку проекту можна розглянути додавання нових функцій, покращення безпеки, оптимізацію продуктивності та підтримку різних пристроїв та браузерів.**
10. **В цілому, проект є успішним, він відповідає поставленим вимогам та може бути використаний в якості робочого прототипу веб-магазину вінілових плат**

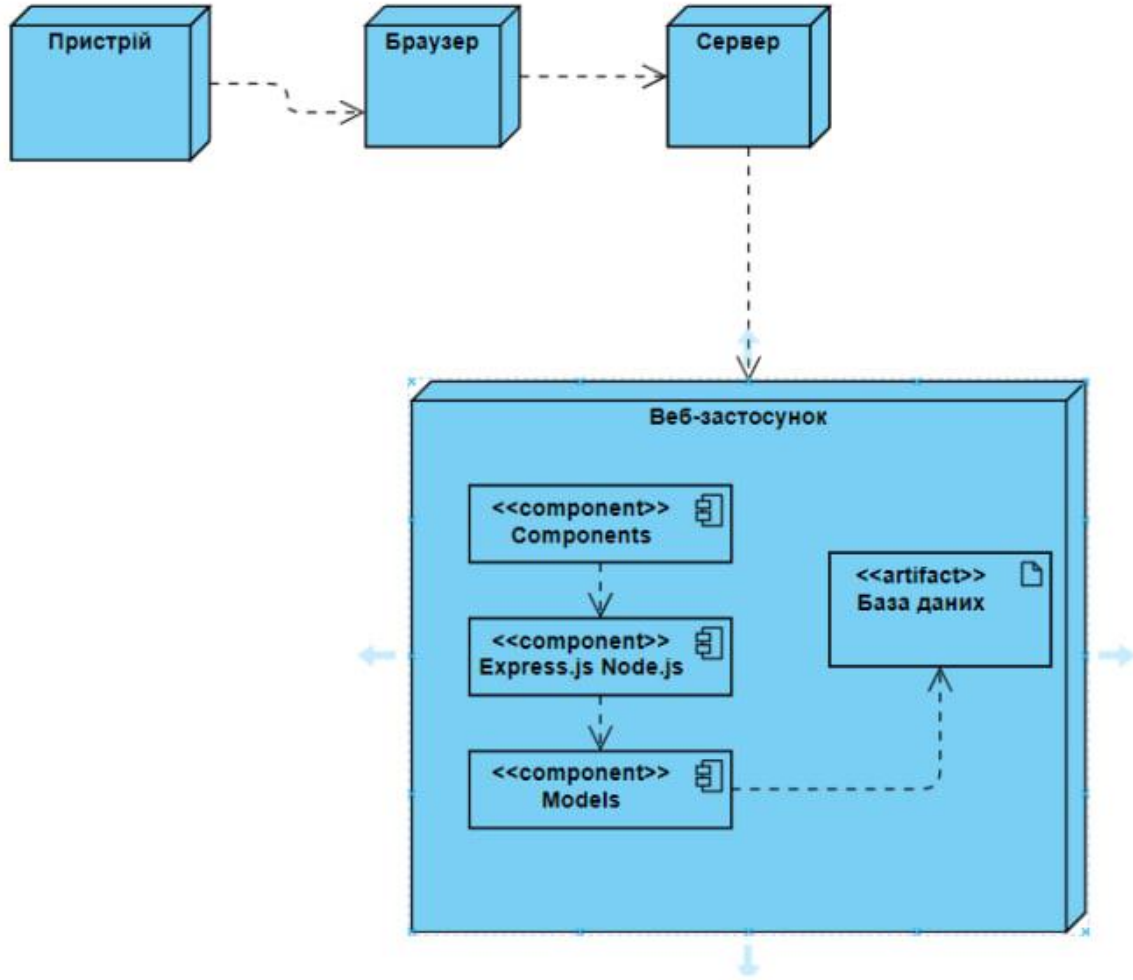
ГРАФІЧНА ЧАСТИНА

Рисунок 1 - Діаграма варіантів використання



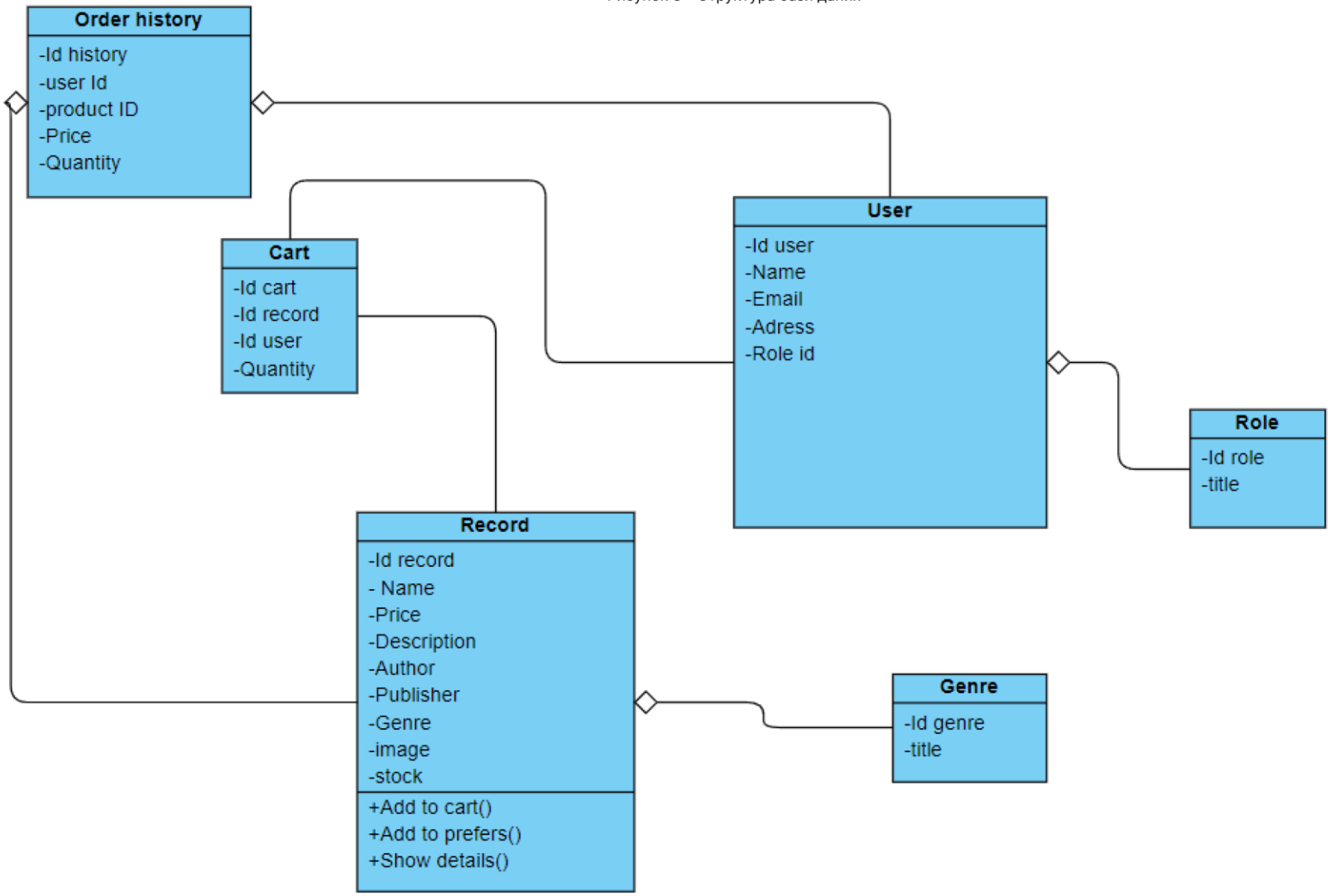
					КвРІП3.200122.01.05.Е8	
Зм.	Арк.	№докум.	Підпис	Дата	USE-CASE Діаграма Веб-застосунок для продажу музичних платівок	Літера / Маса / Масштаб
Розробив		Войченко Р.О.		22.05.23		
Керівник		Бедратюк Л.П.		22.05.23		Аркуш 1 / Аркушів 3
Консульт.						
Н.Контр.		Гурман І.В		22.05.23		ХНУ, ІПЗс-20-1
Зав.каф.		Бедратюк Л.П.		22.05.23		

Рисунок 2 - Діаграма розгортання



					КвРІП3.200122.01.05.Е8	
					Deployment- діаграма Веб-застосунок для продажу музичних платівок	Літера / Маса / Масштаб
Зм.	Арк.	№докум.	Підпис	Дата		
Розробив		Войченко Р.О.		22.05.23		Аркуш 2 / Аркушів 3
Керівник		Бедратюк Л.П.		22.05.23		
Консульт.						
Н.Контр.		Гурман І.В		22.05.23		ХНУ, ІПЗс-20-1
Зав.каф.		Бедратюк Л.П.		22.05.23		

Рисунок 3 - Структура бази даних



					КвРІП3.200122.01.05.Е8	
Зм.	Арк.	№докум.	Підпис	Дата	Схема БД Веб-застосунок для продажу музичних платівок	Літера / Маса / Масштаб
Розробив		Войченко Р.О.		22.05.23		
Керівник		Бедратюк Л.П.		22.05.23		Аркуш 3 / Аркушів 3
Консульт.						
Н.Контр.		Гурман І.В		22.05.23		ХНУ, ІПЗс-20-1
Зав.каф.		Бедратюк Л.П.		22.05.23		

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗс20-1
Войченка Т. О.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Веб. застосунок для продажу музичних посібок.

(керівник роботи – Бедратюк Леонід Петрович
Прізвище, ім'я, по батькові)

05.02.23
Дата

Підпис студента

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Войченка Р. О.
Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗс-20-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

30.05.23
дата

підпис

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

щодо дотримання академічної доброчесності

Цією декларацією я, Войченко Роман Олександрович
Прізвище, ім'я, по батькові

студент III курсу факультету інформаційних технологій,
здобувач вищої освіти (шифр та назва спец-ті, курс, академічна група)/науково-педагогічний працівник (назва кафедри)

кафедри інженерії програмного забезпечення
назва факультету

підтверджую, що ознайомився (- лась) з Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті та Кодексом академічної доброчесності і **зобов'язуюсь** дотримуватися їх вимог під час освітнього процесу, проведення наукової діяльності, виконання організаційно-адміністративних функцій тощо.

Усвідомлюю, що у разі порушення мною принципів академічної доброчесності нестиму відповідальність перед академічною спільнотою ХНУ згідно з нормами, визначеними Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, законодавства України.

«05» лютого 2023 р.

Підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

ID: 114390 Назва: БКР Веб-застосунок для продажу музичних платівок Додано в БД: 2023-05-31 Автора: Войченко Р.О. Керівники: Бедратюк Л.П. Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	75697	695	4691 (6%)	52 (7%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1015349370

Дата перевірки:
31.05.2023 17:55:10 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
31.05.2023 17:59:05 EEST

ID користувача:
100005589

Назва документа: **ВОЙЧЕНКО-ІПЗс20-1-КВР**

Кількість сторінок: **67** Кількість слів: **11599** Кількість символів: **93887** Розмір файлу: **11.38 MB** ID файлу: **1015017362**

7.03%
Схожість

Найбільша схожість: **2.92%** з джерелом з Бібліотеки (ID файлу: **1011233803**)

5.21% Джерела з Інтернету	490	Сторінка 69
4.57% Джерела з Бібліотеки	118	Сторінка 72

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Веб-застосунок для продажу музичних платівок

Автор: Войченко Роман Олександрович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Бедратюк Леонід Петрович д. ф.-м.н., проф

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо та в назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 7,03% і адресується до 608 джерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Керівник _____ Л. П. Бедратюк

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Войченко Роман Олександрович

Тема Веб-застосунок для продажу музичних платівок

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень _____; кількість сторінок записки _____

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі було проведено аналіз предметної області з продажу музичних платівок, визначено усі функціональні та нефункціональні вимоги. Було доведено що розробка ПЗ у цій ніші є доцільною. Був проведений аналіз уже існуючих рішень для електроної комерціїв цій ніші. Була спроектована архітектура та структура застосунку. Було створено програмний код програми та проведено її тестування. За результатами тестування можна вважати що програмне забезпечення працює коректно та задовільняє поставленим вимогам

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі було доведено що тема є актуальною було визначено мету створення застосунку та актуальність теми, визначено мету та завдання дипломного проектування. У першому розділі було проведено аналіз предметної області та розглянуто існуючі рішення та визначені вимоги до розроблюваного застосунку. У другому розділі проведено аналіз сучасних архітектур, розглянуто їх переваги і недоліки та визначено, що система буде відповідати монолітній архітектурі та моделі клієнт-сервер. У третьому розділі була проведено налаштування середовища реалізована архітектура застосунку та було виконано написання програмного коду та проведено тестування застосунку.У висновках для розділу було описано висновки що до проведеної роботи що до розробки застосунку.

4. Позитивні сторони роботи Позитивними сторонами кваліфікаційної роботи є актуальність теми адже ринок вінілу зараз швидко зростає а існуюча пропозиція не завжди задовільняє попит. Також використання сучасного стеку технологій дозволить легко розвивати та масштабувати бізнес у майбутньому.

5. Негативні сторони роботи У роботі сортування музичних платівок відбувається лише за декількома критеріями, що не покриває усіх можливих характеристик музичних платівок для полегшення пошуку користувачам.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики проектування. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження _____

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує оцінку «добре»

РЕЦЕНЗЕНТ (прізвище, ім'я, по-батькові, посада, місце роботи) Мартинюк Валерій Володимирович, зав. каф. АКІТаР

"1" 06 2023 р. _____
(підпис)