

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Інтернет-магазин музичних інструментів»

Виконав студент групи ІПЗ-20-1
ГОЛОВЕЦЬКИЙ Євгеній Володимирович

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ФАНТ Микола Олександрович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГОЛОВЕЦЬКИЙ Євгеній Володимирович**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Інтернет-магазин музичних інструментів»**,
затверджена Наказом закладу вищої освіти від **«26» лютого 2024 р., №74с**

Вихідні дані для роботи: **розробка інтернет магазину музичних інструментів**
Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	16.02.2024р.	16.02.2024р.
2	Вакалюк Т.А.	16.02.2024р.	16.02.2024р.
3	Вакалюк Т.А.	16.02.2024р.	16.02.2024р.

РЕФЕРАТ

Випускна робота бакалавра складається з програмної реалізації інтернет-магазину, а також пояснювальної записки. Пояснювальна записка до дипломної роботи містить 72 сторінок, 47 ілюстрації, 28 таблиць, 14 сторінок додатків та 19 літературних джерел.

Основною метою даної роботи є розробка та впровадження програмного продукту, що надаватиме широкий спектр інструментів для пошуку, вибору та купівлі музичних інструментів в інтернеті.

В роботі визначено ключові завдання, що пов'язані з розробкою інтернет-магазину, проведено аналіз існуючого ринку та наведено сценарії роботи програмного продукту. Представлено огляд загальної структури додатку, описано алгоритми взаємодії між окремими компонентами, а також наведено опис об'єктної структури (діаграму класів) додатку. Описано алгоритми, які використовуються для реалізації основних функцій інтернет-магазину.

					ІПЗ.КР.Б-121-24-ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Головецький			Інтернет магазин музичних інструментів			Літ.	Арк.	Аркушів	
Керівник		Вакалюк Т.А.								3	89
								ФІКТ Гр. ІПЗ-20-1			
Н. контр.											
Зав. каф.											

ABSTRACT

The bachelor's thesis consists of the software implementation of an online store and an explanatory note. The explanatory note for the diploma project contains 72 pages, 47 illustrations, 28 tables, 14 appendix, and 19 literary sources.

The main objective of this work is to develop and implement a software product that provides a wide range of tools for searching, selecting, and purchasing musical instruments online.

The thesis identifies key tasks related to the development of the online store, analyzes the existing market, and presents scenarios for the operation of the software product. An overview of the application's general structure is presented, algorithms for interaction between individual components are described, as well as a description of the object structure (class diagram) of the application. The algorithms used for implementing the main functions of the online store are also described.

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ІНТЕРНЕТ МАГАЗИНУ	8
1.1. Постановка задачі	8
1.2. Аналіз аналогів програмного продукту.....	10
1.3. Вибір архітектури для інтернет магазину	16
1.4. Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення	18
Висновки до першого розділу.....	21
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ ДОДАТКУ	22
2.1.Визначення варіантів використання та об’єктно-орієнтованої структури системи	22
2.2. Проектування бази даних.....	29
2.3. Проектування та реалізація алгоритмів роботи системи.....	34
2.4. Архітектура програмного комплексу.....	38
2.5. Реалізація функціоналу інтернет магазину музичних інструментів.....	40
Висновки до другого розділу	52
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ДОДАТКОМ.....	53
3.1. Порядок розгортання та налаштування системи	53
3.2. Інтерфейс та порядок роботи	54
3.3. Огляд інтерфейсу та інструментів адміністратора.....	61
3.4. Тестування роботи інтернет магазину музичних інструментів.	65
Висновки до третього розділу	68
ВИСНОВКИ.....	69
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс.

БД – база даних.

EF Core – EntityFramework Core – фреймворк для полегшення доступу до бази даних.

ПЗ – програмне забезпечення.

СУБД – система управління базами даних.

MVC – схема розділення даних додатків і керуючої логіки на три окремі компоненти Model-View-Controller CRUD – основні операції, що використовуються для управління даними (Create , Read , Update , Delete).

MSSQL – Microsoft SQL – одна з найпопулярніших реляційних систем управління базами даних.

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми полягає у тому що музика завжди була важливою частиною культури та самовираження. Розвиток інтернет технологій значно понизив поріг входу у світ музики та надав можливість кожному виражати своє бачення через створення чогось унікального. Це набагато збільшило потік як професіоналів так і аматорів, які створюють попит на різноманітне музичне обладнання. Також саме завдяки розвитку музичної культури, все більше проводяться різноманітні події, які теж потребують спеціального обладнання.

Метою є створення якісного сервісу, який надає широкий асортимент товарів та змогу обирати, вивчати та замовляти музичне обладнання. Сервіс має надавати товари як і для початківців, так і складні системи для великих проєктів, типу обладнання для концертів чи клубів. Процес має бути як умога зручнішим та зрозумілим. Для реалізації задуманого потрібно провести аналіз вже існуючих рішень на ринку, визначити сильні та слабкі сторони.

Встановлена мета обумовлює наступні завдання:

- аналіз завдань, які має вирішувати інтернет магазин музичних інструментів,
- проектування архітектури додатку, яка забезпечить максимальну ефективність задуманого функціоналу,
- обґрунтування вибору інструментів розробки,
- дослідження та планування функціоналу додатку, розробка алгоритмів та варіантів використання,
- програмна реалізація проєкту.

Об'єктом дослідження є інтернет-магазини, що спеціалізуються на продажу музичних інструментів, запчастин до них, обладнання для як для початківців, професійних музикантів так і організаторів великих подій.

Предметом дослідження є елементи які стосуються розробки інтернет магазину, такі як асортимент товарів, користувацький досвід, технологічні рішення, також їх переваги у порівнянні з існуючими аналогами на ринку

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ІНТЕРНЕТ МАГАЗИНУ

1.1.Постановка задачі

Метою кваліфікаційної роботи бакалавра є проектування та розробка інтернет-магазину який завдяки сучасним технологіям надасть змогу користувачам обирати та замовляти музичне обладнання та інструменти. Користувачі матимуть можливість переглядати товари в каталозі, шукати бажаний товар, використовувати сортування та фільтрацію. В подальшому можливе додавання товару в коризну та оформлення замовлення.

Сторінки товарів мають надавати повний опис самого продукту, а саме назву, точну характеристику та відеодемонстрацію товару, також для кожного товару будуть підбрані рекомендовані товари з інших категорій. Для повного розуміння товару буде реалізована система рейтингу та відгуків. Кожен користувач матиме змогу залишити власний коментар з оцінкою. Інші користувачі зможуть також оцінювати кожен відгук.

Також для зареєстрованих користувачів буде доступний особистий профіль. Там буде міститися вся інформація про користувача, та можливість змінити її, історія замовлень та списки обраного.

Для реалізації поставленого завдання, основними етапами є:

1. Аналіз проекту та збір вимог:

- дослідження вже існуючих практик реалізації інтернет магазину,
- визначення основного функціоналу інтернет-магазину, створення тіаграм для глибшого розуміння,
- аналіз існуючих аналогів.

2. Вибір та інтеграція технологій:

- оцінка існуючих інструментів для розробки, вибір основного стеку розробки,
- пошук вже розроблених бібліотек та фреймворків,
- інтеграція рішень у проект.

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

3. Розробка системи:

- проектування та розробка бази даних системи, яка буде забезпечувати високу швидкість доступу до даних,
- розробити API яке буде видавати дані для потреб фронт-енду,
- розробити сторінки для реєстрації та входу, реалізувати вхід за допомогою сторонніх сервісів, можливість зміну пароля та логіну,
- розробити каталог з товарами, реалізувати пошук, вибірку за категоріями, сортування та фільтрацію,
- оформити сторінку з товарами, має бути пристунє фото та відео демонстрації товару,
- реалізувати функціонал кошику, передбачити можливість застосування гостей сайту,
- реалізувати процес оформлення замовлення, зібрати всі необхідні дані з користувача,
- додати систему відгуків та оцінок.

4. Тестування системи:

- Переконатися в працездатності системи шляхом перевірки усього функціоналу на предмет несправностей,
- Покриття системи різноманітними тестами, для легкого вирішення можливих проблем.

5. Розгортання:

- Розгортання системи на базі платформи Azure[\[1\]](#).

Розробка системи надасть користувачам можливість користуватися зручним та надійним додатком для пошуку та замовлення музичного обладнання, для їх порівняння та прозорого вибору.

Таким чином процес пошуку та отримання бажаних товарів стане в рази легшим. Як новачки так і професіонали зможуть задовольнити свої потреби в різному обладнанні, завдяки розробленій системі.

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

1.2. Аналіз аналогів програмного продукту

Зараз на ринку існують декілька основних продавців музичного обладнання на території України. Тут при проведемо аналіз функціоналу, товарів та дизайну таких трьох магазинів як jam.ua, muztorg.ua та luxpro.ua.

Jam.ua^[2] - це інтернет-магазин музичних інструментів та обладнання, який пропонує широкий асортимент товарів для музикантів різних рівнів та жанрів.



Рисунок 1.1 – Інтерфейс Jam.ua.

Переваги:

- **Широкий вибір:** На сайті представлено безліч категорій товарів, включаючи струнні, клавішні, духові інструменти, ударні, студійне обладнання та аксесуари.

		Головецький Є.В.			ШЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

- Зручна навігація: Інтуїтивно зрозумілий інтерфейс дозволяє легко знайти потрібні товари, використовуючи фільтри та категорії.
- Акції та знижки: Регулярно проводяться акції та пропонуються знижки на різні товари, що дозволяє зекономити кошти.
- Професійна підтримка: Консультанти магазину готові допомогти з вибором інструментів та обладнання, надаючи професійні рекомендації.

Недоліки:

- Ціни: Деякі товари можуть бути дорожчими порівняно з аналогічними пропозиціями в інших інтернет-магазинах.
- Обмежена інформація про товари: Не завжди доступна детальна інформація та відгуки про продукцію, що може ускладнити вибір.
- Доставка: Можливі затримки з доставкою, особливо при замовленнях великих та об'ємних товарів.

Muztorg.ua [3]— це великий інтернет-магазин, що спеціалізується на продажу музичних інструментів та обладнання. Магазин є офіційним дистриб'ютором товарів від провідних світових брендів, зокрема Gibson, Fender, Yamaha, Korg та багатьох інших.

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

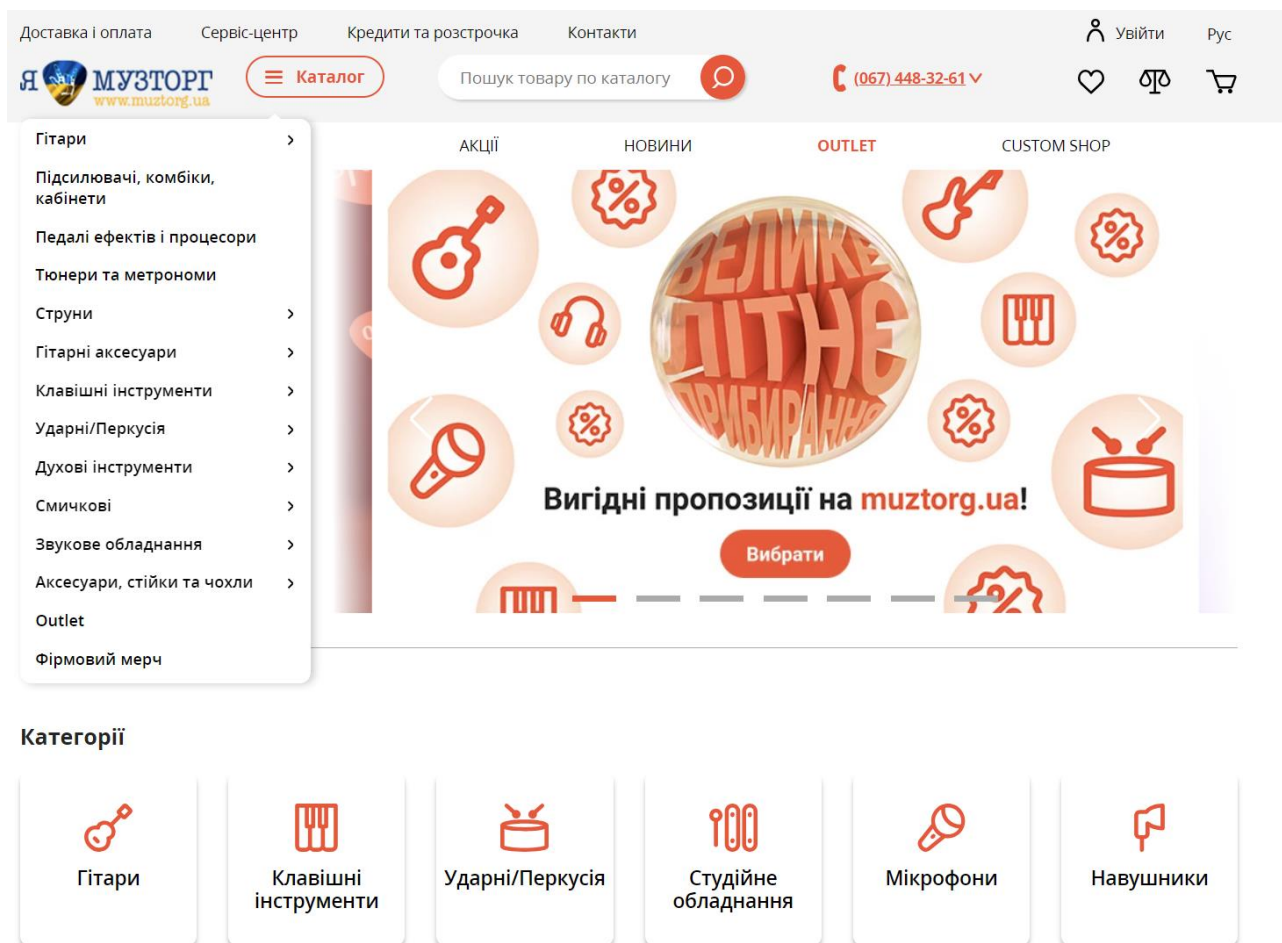


Рисунок 1.2 – Інтерфейс Muztorg.ua.

Переваги:

- **Широкий асортимент:** Магазин пропонує великий вибір музичних інструментів, включаючи гітари, клавішні, ударні інструменти, а також обладнання для студій звукозапису та акустичні системи.
- **Офіційні дистриб'ютори:** Muztorg.ua є офіційним представником багатьох відомих брендів, що гарантує оригінальність та якість продукції.
- **Багато філіалів:** Магазин має мережу фізичних магазинів, де можна особисто ознайомитися з товарами, отримати консультацію та зробити покупку.
- **Програма лояльності:** Існують програми лояльності та бонусні системи, які дозволяють постійним клієнтам отримувати додаткові знижки та переваги.

		Головецький Є.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Вакалюк Т.А.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Недоліки:

- Обмежена кількість бюджетних варіантів: Хоча асортимент великий, вибір бюджетних інструментів може бути обмеженим, що може бути незручним для новачків або тих, хто шукає недорогі варіанти.
- Можливі проблеми з наявністю товарів: Іноді можуть виникати ситуації, коли популярні товари швидко закінчуються на складі, і доводиться чекати на їх повторну поставку.
- Доставка лише по Україні: Магазин здійснює доставку тільки по Україні, що обмежує можливості замовлень для клієнтів з інших країн.

Luxpro.ua^[4] — це інтернет-магазин, який спеціалізується на продажу світлового та звукового обладнання, музичних інструментів, а також техніки для різних шоу та заходів.

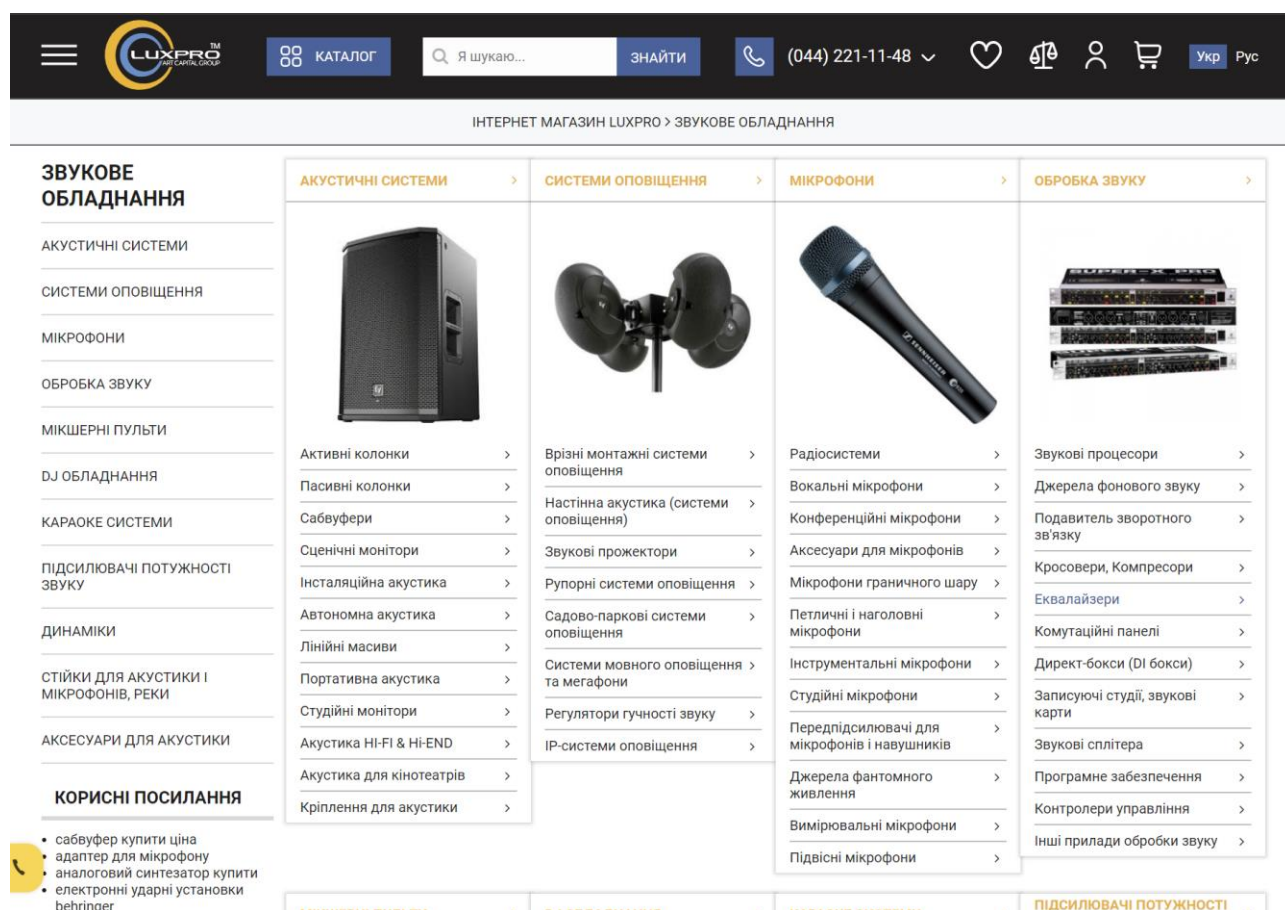


Рисунок 1.3 – Інтерфейс luxpro.ua.

Переваги:

- Сервісний центр: Магазин має власний сервісний центр, де можна провести ремонт та обслуговування придбаного обладнання. Це значно збільшує термін служби техніки та забезпечує її безперебійну роботу.
- Професійний асортимент: магазин орієнтований в основному на професійне студійне обладнання. Це дозволяє клієнтам знайти все необхідне для організації заходів в одному місці.

Недоліки:

- Вузька спеціалізація: пропонує великий вибір світлового та звукового обладнання, музичних інструментів, техніки для ді-джеїв, караоке систем, лазерних шоу та багато іншого. Це дозволяє клієнтам знайти все необхідне для організації заходів в одному місці.
- Високі ціни: товари в Luxpro.ua мають досить високу ціну, оскільки основни асортимент – професійне обладнання. Хоча асортимент включає і бюджетні варіанти, більшість професійного обладнання коштує дорого.

На основні аналізу аналогів програмного продукту було складено порівняльну таблицю:

Таблиця 1.1

	Jam.ua	Muztorg.ua	Luxpro.ua
Приємний інтерфейс	+	+	+
Широкий асортимент	+	+	-
Ціна	+	+	-
Спеціалізація	Загальні товари	Загальні товари	Професійне обладнання

Підтримка користувачів	+	+	+
Визнання	-	+	-
Функціонал сайту	+	+	+

Провівши детальний аналіз аналогічних продуктів можна виділити найбільш успішні та негативні сторони продуктів jam.ua, muztorg.ua та luxpro.ua.

Завдяки цьому проект стане сильним інструментом як для початківців, так і для професіоналів та організаторів заходів, який надаватиме широкий спектр послуг та товарів, матиме приємний дизайн та зручний функціонал.

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

1.3. Вибір архітектури для інтернет магазину

Оскільки проект являє собою інтернет магазин, вже перевіреною практикою є використання onion-архітектури.

Onion архітектура[5] (або архітектура "цибулини") є програмною архітектурною парадигмою, яка підкреслює відокремлення логіки бізнесу від зовнішніх компонентів, таких як інтерфейси користувача, бази даних або веб-сервіси. Ця архітектура має кілька рівнів, які утворюють форму цибулини, де ядро містить основну бізнес-логіку, а зовнішні шари взаємодіють з іншими компонентами системи.

Такий вибір зумовлено тим що веб-додаток використовує багато сторонніх сервісів, і використання цієї парадигми надає чітку структуру розподілу відповідальностей. Вона гарантує що компоненти веб-додатку не будуть конфліктувати між собою та будь ізольовані один від одно. Завдяки цьому внесення змін та додавання нового функціоналу до проекту стає в рази простішим та швидшим. (див. рис. 1.4.)

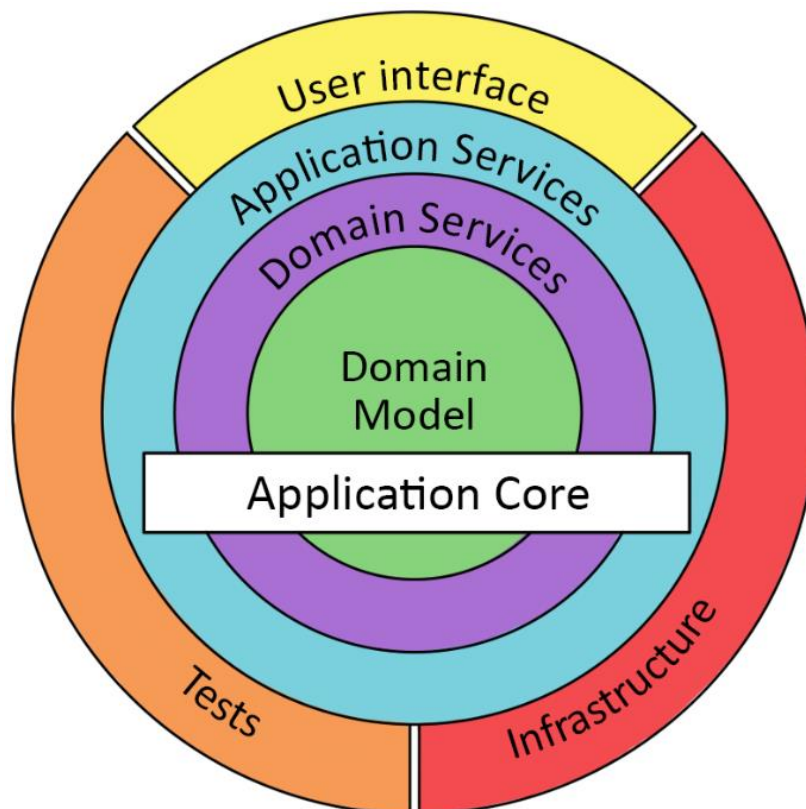


Рисунок 1.4 – Onion архітектура.

Наведена діаграма обраної архітектури проєкту складається з:

Application Core є центром програми і включає:

- **Domain Model:** Містить бізнес-логіку та правила, які визначаються предметною областю програми. Це модель, яка визначає поведінку системи та її бізнес-процеси
- **Domain Services:** Послуги, що забезпечують бізнес-логіку, яка не може бути віднесена до конкретного об'єкта або сутності.
- **Application Services:** Відповідає за реалізацію сценаріїв користувача та взаємодіє з Domain Model. Ці сервіси координують виконання бізнес-логіки та відповідають за реалізацію use cases.

User Interface (Користувацький інтерфейс) — це зовнішній шар, який взаємодіє з користувачем. Він відповідає за відображення інформації та збір даних від користувача. Це може бути веб-інтерфейс, мобільний додаток або будь-яка інша форма інтерфейсу.

Infrastructure (Інфраструктура) — це шар, який відповідає за технічну інфраструктуру, необхідну для роботи програми. Він включає доступ до баз даних, файлові системи, зовнішні сервіси, мережеві з'єднання та інші технічні аспекти.

Tests (Тести) — це шар, що відповідає за тестування всіх аспектів програми. Включає юніт-тести, інтеграційні тести, функціональні тести та інші види тестування, щоб забезпечити якість і надійність програмного забезпечення.

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

1.4. Обґрунтування вибору інструментальних засобів та вимог до апаратного забезпечення

Вибір технологій для розробки бекенду та фронтенду є важливим рішенням, яке впливає на ефективність, масштабованість та зручність підтримки веб-додатку.

Для бекенду основною мовою програмування є C#, сучасна, об'єктно-орієнтована мова програмування, розроблена компанією Microsoft як частина платформи .NET. Вона відома своєю потужністю, продуктивністю та зручністю для розробників.

Основні переваги мови C# [6]:

Об'єктно-орієнтована природа:

- Інкапсуляція, наслідування та поліморфізм: основні принципи ООП які підтримує мова C#.
- Абстракція та інтерфейси: завдяки ним можливо гнучкі та розширювані програми.

Сучасні особливості:

- LINQ [7] та Лямбда-вирази: Ці особливості дозволяють легко та ефективно маніпулювати колекціями даних, роблячи код більш виразним та зручним для читання.
- Асинхронність: дозволяє писати асинхронний код так само легко, як синхронний.

Безпека та надійність:

- Керована пам'ять: Завдяки збору сміття (garbage collection), C# звільняє розробників від необхідності вручну керувати пам'яттю, що знижує ризик утечок пам'яті та інших помилок.
- Безпека типів: Статична типізація в C# допомагає виявляти помилки на етапі компіляції, підвищуючи надійність коду.

Платформа .NET [8] також надає потужний інструмент для веб розробки, а саме ASP.Net [9]. Версія ASP.NET Core, має високу продуктивність завдяки

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

своєму ефективному використанню ресурсів та оптимізації для сучасних веб-додатків.

Основні переваги ASP.Net:

Продуктивність та масштабованість:

- Висока продуктивність: ASP.NET, особливо у версії ASP.NET Core, має високу продуктивність завдяки своєму ефективному використанню ресурсів та оптимізації для сучасних веб-додатків.
- Масштабованість: ASP.NET легко масштабується, підтримуючи як горизонтальне, так і вертикальне масштабування.

Безпека:

- Інтегровані механізми безпеки: ASP.NET пропонує потужні засоби для реалізації автентифікації та авторизації, захисту від загроз, таких як SQL-ін'єкції[10] та Cross-Site Scripting (XSS)[11].
- Підтримка OAuth і OpenID Connect[12]: Проста інтеграція з популярними системами автентифікації.

Підтримка різних типів додатків:

- Розмаїття шаблонів: Підтримка MVC[13], Web API та SignalR[14] надає потужний інструментарій, що дозволяє створювати різноманітні типи додатків під різні типи потреб.

Одним із основних моментів є отримання даних з бази. Перед розробником завжди стоїть задача зробити це процес як умога зручнішими та швидшим. З цим допомагає ORM[15] система Entity Framework Core[16].

Основні переваги Entity Framework Core:

Простота та продуктивність:

- Модель першого коду (Code-First[17]): EF Core дозволяє розробникам визначати моделі даних за допомогою класів C#, а потім автоматично створювати схеми баз даних на основі цих моделей. Це робить процес розробки швидшим і більш інтуїтивним, оскільки не потрібно вручну створювати SQL-скрипти для визначення таблиць.

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

- Підтримка міграцій: За допомогою міграцій розробники можуть легко оновлювати схему бази даних відповідно до змін у моделях даних. Міграції автоматично генерують SQL-скрипти для внесення необхідних змін у базу даних, що значно спрощує процес управління версіями бази даних.

Потужні можливості та розширюваність:

- Інтеграція з іншими технологіями: EF Core легко інтегрується з іншими технологіями, такими як ASP.NET Core, SignalR та інші компоненти екосистеми .NET, що дозволяє створювати комплексні та масштабовані рішення. Це також включає підтримку шаблонів репозиторіїв та одиниць роботи (Unit of Work [\[18\]](#)), що робить архітектуру додатків більш організованою та легкою для підтримки.

Оскільки проект вже використовує інші продукти Microsoft, природним вибором стає MSSQL[\[19\]](#) завдяки глибокій інтеграції з цими технологіями.

Microsoft SQL Server (MSSQL) є одним з найпопулярніших реляційних систем управління базами даних (RDBMS). Використання MSSQL у проекті має ряд вагомих переваг.

Основні причини вибору MSSQL:

- **Висока продуктивність:** MSSQL оптимізований для роботи з великими обсягами даних і складними запитами, забезпечуючи швидкий доступ до даних та високу продуктивність додатків.
- **Інтелектуальна оптимізація запитів:** Вбудовані можливості для оптимізації запитів дозволяють автоматично вибирати найефективніший спосіб виконання запитів, що підвищує продуктивність бази даних.
- **Захист даних:** MSSQL пропонує широкий спектр засобів для забезпечення безпеки даних, включаючи шифрування, аудит доступу та ролі на основі привілеїв. Це дозволяє надійно захищати конфіденційні дані.

		Головецький Є.В.			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до першого розділу

У першому розділі було проведено комплексний аналіз та підготовку до розробки інтернет-магазину музичних інструментів. Розглянуті аспекти включають постановку задачі, аналіз аналогічних програмних продуктів, вибір архітектури та обґрунтування інструментальних засобів.

Було визначено основні цілі та завдання проекту. Основною метою є створення зручної та функціональної платформи для продажу музичних інструментів. Задачі включають реалізацію користувацького інтерфейсу, системи управління товарами, кошика покупок, системи обробки замовлень та інтеграцію платіжних систем.

Проведений аналіз існуючих рішень на ринку інтернет-магазинів музичних інструментів показав, що більшість популярних платформ мають схожий набір функцій, але відрізняються зручністю користування, швидкістю роботи, дизайном та асортиментом товарів. Це дозволило визначити ключові вимоги до нашого проекту, які включають високу продуктивність, зручність інтерфейсу та широкі можливості для пошуку товарів.

Було обрано архітектурний підхід, що базується на Onion Architecture. Цей підхід забезпечує модульність, легкість тестування та підтримки коду. Внутрішні та зовнішні залежності чітко розмежовані, що сприяє створенню більш гнучкої та масштабованої системи. Обраний підхід дозволяє легко адаптувати та розширювати функціональність магазину в майбутньому.

Для реалізації бек-енду проекту було обрано ASP.NET Core, завдяки його високій продуктивності, підтримці кросплатформеності та інтеграції з різними сервісами Microsoft. Взаємодію з базою забезпечуватиме Entity Framework Core. Базу даних планується реалізувати на Microsoft SQL Server (MSSQL), яка забезпечує високу продуктивність, безпеку та надійність.

		Головецький Є.В.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ ДОДАТКУ

2.1.Визначення варіантів використання та об'єктно-орієнтованої структури системи

У цьому розділі представлено визначення варіантів використання та об'єктно-орієнтованої структури системи для інтернет-магазину музичних інструментів. Основна мета розділу – окреслити функціональні можливості системи через варіанти використання, а також надати структурований підхід до проектування програмного продукту. Це включає детальний опис сценаріїв взаємодії користувачів та адміністраторів із системою, а також побудову об'єктно-орієнтованої моделі, що відображає основні сутності системи та їх взаємодію.

В процесі роботи над розділом було визначено основні функціональні можливості системи, які забезпечують необхідну функціональність для користувачів та адміністраторів. Користувачі отримують можливість переглядати та додавати товари до кошика, оформлювати замовлення, залишати відгуки та оцінки, а також керувати своїми обліковими записами. Адміністратори мають змогу керувати товарами, категоріями, брендами, замовленнями та користувачами.

Проаналізувавши аналогічні рішення та наявний інструментарій, була створена діаграма прецедентів та наведений детальний опис варіантів використання.(див.рис.2.1.).

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

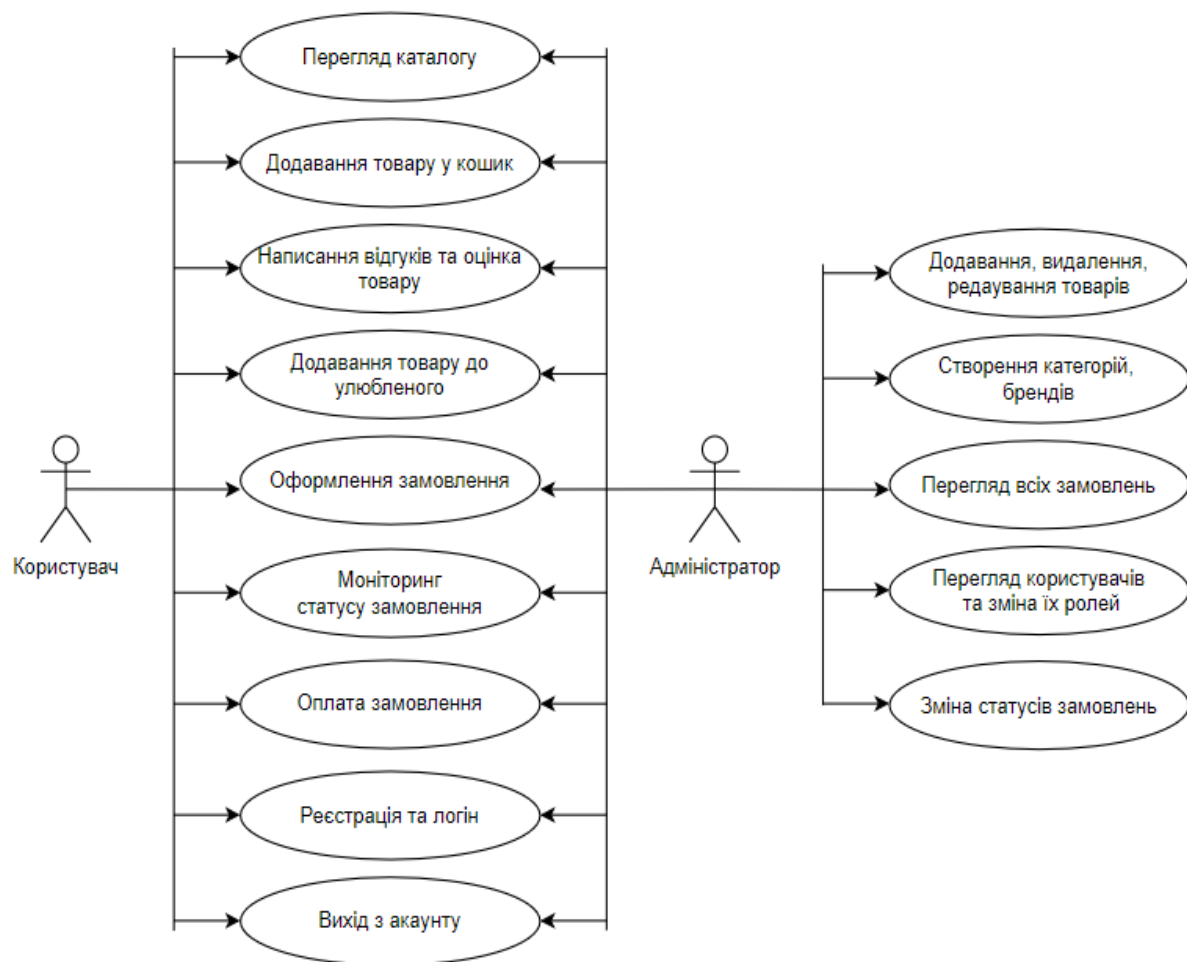


Рисунок 2.1 – Діаграма варіантів використання.

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.14.

Таблиця 2.1

Опис варіанту «Перегляд товару»	
Назва	Перегляд товару
Актори	Користувач, Адміністратор
Короткий опис	Перегляд списку доступних музичних інструментів, переглядати деталі кожного товару, такі як опис, ціна, фотографії, характеристики.

Таблиця 2.2

Опис варіанту «Додавання товару у кошик»	
Назва	Додавання товару у кошик
Актори	Користувач, Адміністратор
Короткий опис	Користувач може додавати обрані товари у кошик для подальшого придбання

Таблиця 2.3

Опис варіанту «Написання відгуків та оцінка товару»	
Назва	Написання відгуків та оцінка товару
Актори	Користувач, Адміністратор
Короткий опис	Користувач може залишати відгуки та оцінювати товари, що допомагає користувачам приймати рішення про покупку.

Таблиця 2.4

Опис варіанту «Додавання товару до улюбленого»	
Назва	Додавання товару до улюбленого
Актори	Користувач, Адміністратор
Короткий опис	Користувач може додавати товари до списку улюблених для швидкого доступу у майбутньому.

Таблиця 2.5

Опис варіанту «Оформлення замовлення»	
Назва	Оформлення замовлення
Актори	Користувач, Адміністратор
Короткий опис	Користувач може оформити замовлення на товари, що знаходяться у кошику, вказавши дані для доставки та оплати.

Таблиця 2.6

Опис варіанту «Оплата замовлення»	
Назва	Оплата замовлення
Актори	Користувач, Адміністратор
Короткий опис	Користувач може здійснити оплату замовлення через інтегровані платіжні системи

Таблиця 2.7

Опис варіанту «Реєстрація та логін»	
Назва	Реєстрація та логін
Актори	Користувач, Адміністратор
Короткий опис	Користувач може створити обліковий запис або увійти у вже існуючий для доступу до персоналізованих функцій.

Таблиця 2.8

Опис варіанту «Вихід з акаунту»	
Назва	Вихід з акаунту
Актори	Користувач, Адміністратор
Короткий опис	Користувач може вийти зі свого облікового запису для завершення сеансу

Таблиця 2.9

Опис варіанту «Додавання, видалення, редагування товарів»	
Назва	Додавання, видалення, редагування товарів
Актори	Адміністратор
Короткий опис	Адміністратор може керувати товарами у каталозі, додаючи нові, редагуючи існуючі або видаляючи товари, які більше не доступні.

Таблиця 2.10

Опис варіанту «Створення категорій, брендів»	
Назва	Створення категорій, брендів
Актори	Адміністратор
Короткий опис	Адміністратор може створювати нові категорії товарів та бренди, що полегшує користувачам навігацію по каталогу.

Таблиця 2.11

Опис варіанту «Перегляд всіх замовлень»	
Назва	Перегляд всіх замовлень
Актори	Адміністратор
Короткий опис	Адміністратор має доступ до перегляду всіх замовлень, що дозволяє моніторити процеси та виконання замовлень.

Таблиця 2.12

Опис варіанту «Перегляд користувачів та зміна їх ролей»	
Назва	Перегляд користувачів та зміна їх ролей
Актори	Адміністратор
Короткий опис	Адміністратор може змінювати статус замовлень

Для забезпечення чіткого розуміння архітектури системи було створено об'єктно-орієнтовану структуру, яка включає основні класи системи, їхні атрибути та методи. Це дозволяє зрозуміти взаємодію між різними компонентами системи та забезпечити належну організацію коду при розробці.

Нижче наводиться об'єктно-орієнтована структура системи на основі діаграми прецедентів, що була розглянута раніше. (див.рис 2.2.). Зоображена діаграма демонструє спроектовані сутності та як вони відносяться один до одного. На основі цієї діаграми в подальшому буде створена база даних системи.

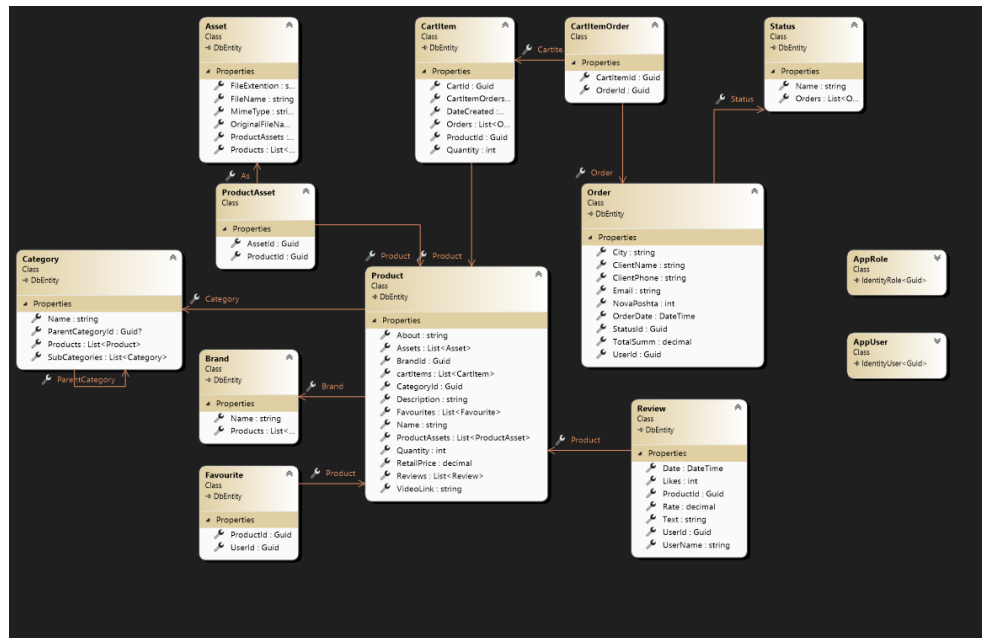


Рисунок 2.2 Сутності, спроектовані після аналізу діаграми прецедентів наведеної вище(див.рис.2.1)

Класс Product

- Відповідає за представлення інформації про товари

Класс Category

- Відповідає за представлення інформації про категорії

Класс Brand

- Відповідає за представлення інформації про бренди

Класс Order

- Відповідає за представлення інформації про замовлення

Класс CartItem

- Відповідає за представлення інформації про елементи корзини

Класс Favorite

- Відповідає за представлення інформації про улюблені товари користувача.

Класс Review

- Відповідає за представлення інформації про відгуки користувачів

Класс Asset

- Відповідає за представлення інформації про файли, пов'язані з товарами

Класс Status

- Відповідає за представлення статусів замовлень

Класс AppUser

- Відповідає за представлення інформації про користувача, включаючи дані для авторизації та контактну інформацію

Класс AppRole

- Відповідає за представлення інформації про користувача, включаючи дані для авторизації та контактну інформацію

Взаємодія між класами

- Користувачі (User) взаємодіють з товарами (Product) через класи CartItem, Favorite, та Order для додавання товарів до кошика, улюблених, та оформлення замовлень.
- Адміністратори (Admin) управляють товарами (Product), категоріями (Category), та брендами (Brand) за допомогою відповідних методів додавання, редагування та видалення.
- Файли (Asset) пов'язані з товарами (Product),
- Товари(Product) мають категорії(Category), бренди(Brand) та відгуки().
- Змовлення(Orders) оформлюються через відповідні елементи кошика(CartItems)

Розглянута діаграма прецедентів ідентифікує ключові дії користувачів та адміністраторів інтернет-магазину музичних інструментів. На основі цих дій була розроблена об'єктно-орієнтована структура системи, яка включає основні класи. Це забезпечує чітке розуміння архітектури системи та дозволяє ефективно планувати подальшу розробку та реалізацію проекту. На основі цієї структури в подальшому буде створена база даних.

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

2.2. Проектування бази даних

У цьому розділі представлено детальне проектування бази даних для інтернет-магазину музичних інструментів. Основна мета цього розділу – розробити реляційну модель бази даних, яка забезпечить ефективне зберігання та обробку даних, необхідних для функціонування системи.

Структура проекту складається з 2 баз даних. Перша містить в собі всю інформацію про користувачів та ролі включаючи дані для авторизації та контактну інформацію користувача. Друга ж зберігає все що пов'язане з товарами, коризною, замовленнями та відгуками.

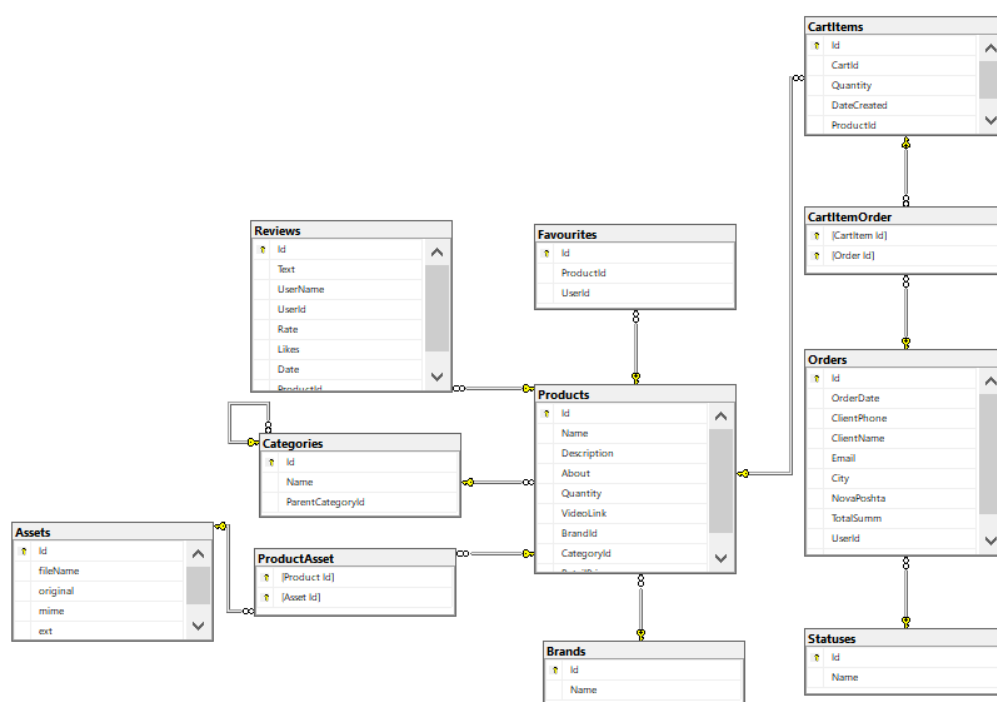


Рисунок 2.3 Діаграма таблиць бази даних «Store_data».

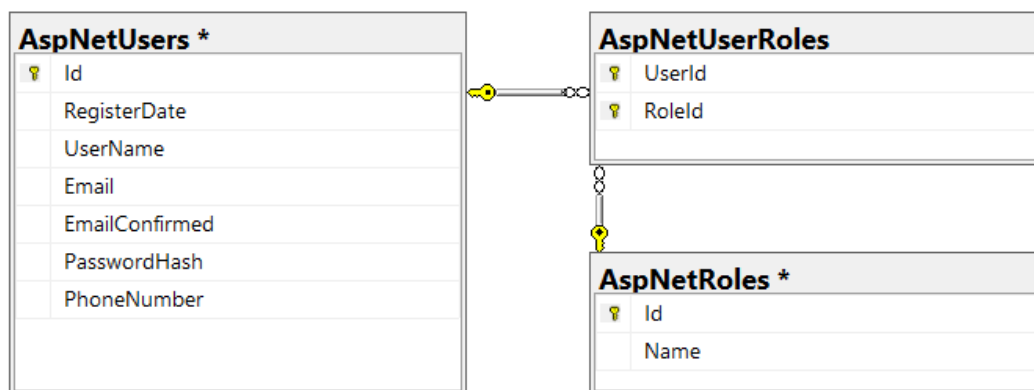


Рисунок 2.4 Діаграма таблиць бази даних «Store_identity».

Окремо розберемо кожну з таблиць представлених у базах даних.

Опис таблиць бази даних «Store_identity»:

Таблиця «AspNetUsers» містить всю інформацію про користувача.

Таблиця 2.13

Таблиця «AspNetUsers»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
UserName	nvarchar	Логін або псевдонім користувача
Email	nvarchar	Пошта
EmailConfirmed	nvarchar	Чи підтверджена пошта
PhoneNumber	nvarchar	Номер телефону
FullName	nvarchar	Повне ім'я
RegisterDate	date	Дата реєстрації

Таблиця «AspNetRoles» містить всю інформацію про ролі користувачів.

Таблиця 2.14

Таблиця «AspNetRoles»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
Name	nvarchar	Назва ролі

Опис таблиць бази даних «Store_data»:

Таблиця «Product» містить всю інформацію про товар.

Таблиця 2.15

Таблиця «Product»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
Name	nvarchar	Назва товару
Description	nvarchar	Опис товару
About	nvarchar	Інформація про товар
Quantity	int	Кількість товару
VideoLink	nvarchar	Посилання на відеодемонстрацію
BrandId	nvarchar	Унікальний номер бренду товару
CategoryId	nvarchar	Унікальний номер категорії
RetailPrice	float	Ціна

Таблиця «Orders» містить всю інформацію про замовлення.

Таблиця 2.16

Таблиця «Orders»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
OrderDate	date	Дата замовлення
ClientPhone	nvarchar	Номер телефону клієнта
ClientName	nvarchar	Ім'я клієнта
Email	nvarchar	Пошта клієнта
NovaPoshta	nvarchar	Адреса відділення доставки
TotalSumm	float	Повна сума замовлення
UserId	nvarchar	Унікальний номер користувача
StatusId	nvarchar	Унікальний номер статусу

Таблиця «CartItems» містить всю інформацію про елемент в корзині.

Таблиця 2.15

Таблиця «CartItems»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
CartId	nvarchar	Унікальний номер корзини
Quantity	int	Кількість позиції в корзині
DateCreated	date	Дата створення
ProductId	nvarchar	Унікальний номер товару

Таблиця «Reviews» містить всю інформацію про відгуки про товар.

Таблиця 2.16

Таблиця «Reviews»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
Text	nvarchar	Текст відгуку
UserId	nvarchar	Унікальний номер користувача
Rate	float	Рейтинг відгуку
Likes	int	Кількість вподобань на відгуку
Date	date	Дата створення
ProductId	nvarchar	Унікальний номер товару

Таблиця «Assets» містить всю інформацію про відгуки про допоміжні файли проекту.

Таблиця 2.17

Таблиця «Assets»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
FileName	nvarchar	Ім'я файлу
Original	nvarchar	Оригінальне ім'я
Mime	nvarchar	Mime-тип
Ext	nvarchar	Розширення файлу

Таблиця «Assets» містить всю інформацію про категорії.

Таблиця 2.18

Таблиця «Categories»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
Name	nvarchar	Назва категорії
ParentCategoryId	nvarchar	Унікальний номер батьківської категорії

Таблиця «Favourites» містить всю інформацію про улюблені товари користувачів.

Таблиця 2.19

Таблиця «Favourites»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
ProductId	nvarchar	Унікальний номер товару який користувач додав до улюбленого
UserId	nvarchar	Унікальний номер користувача який додав товар до улюбленого

Таблиця «Brands» містить всю інформацію про бренди.

Таблиця 2.20

Таблиця «Brands»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
Name	nvarchar	Назва бренду

Таблиця «ProductAssets» містить зв'язок між товаром та файлами.

Таблиця 2.21

Таблиця «ProductAssets»		
Поле	Тип	Призначення
ProductId	nvarchar	Унікальний номер товару
AssetId	nvarchar	Унікальний номер файлу

Таблиця «ProductAssets» містить зв'язок між елементом корзини та замовленням.

Таблиця 2.22

Таблиця «CartItemOrder»		
Поле	Тип	Призначення
CartItemId	nvarchar	Унікальний номер елементу корзини
OrderId	nvarchar	Унікальний номер замовлення

Таблиця «Statuses» містить всю інформацію про статуси.

Таблиця 2.23

Таблиця «Statuses»		
Поле	Тип	Призначення
Id	nvarchar	Унікальний номер
Name	nvarchar	Назва Статусу

2.3. Проектування та реалізація алгоритмів роботи системи.

У цьому розділі розглядається проектування та реалізація ключових алгоритмів, необхідних для функціонування інтернет-магазину музичних інструментів. Алгоритми є основою системи, оскільки вони визначають процеси, що забезпечують виконання основних функцій та інтеракції між користувачами і базою даних.

Метою цього розділу є детальне описання алгоритмів, які реалізують функціональні можливості системи, включаючи реєстрацію та авторизацію

		Головецький Є.В.			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

користувачів, управління товарами, обробку замовлень та забезпечення взаємодії між різними компонентами системи.

Проектування алгоритмів включає розробку логіки, яка забезпечує правильне та ефективне виконання бізнес-процесів, таких як управління користувачами, товарами та замовленнями. Реалізація цих алгоритмів дозволяє інтегрувати різні компоненти системи та забезпечити їх коректну роботу у взаємодії з базою даних.

В ході розробки проекту було створено декілька діаграм активностей для наглядної демонстрації та подальшого розуміння алгоритмів роботи проекту.

Для початку розберемо процес реєстрації нового користувача.

Реєстрація нового користувача є критичною функцією, яка включає перевірку введених даних та створення нового запису в базі даних.

- **Отримання даних:** Система отримує дані користувача (ім'я, електронну адресу, пароль).
- **Перевірка наявності користувача:** Перевіряється, чи існує користувач з такою ж електронною адресою.
- **Присвоєння ролі:** Нови користувач отримує роль за замовчуванням.
- **Збереження даних:** Дані нового користувача зберігаються в таблиці Users.
- **Підтвердження реєстрації:** Користувачу надсилається електронний лист для підтвердження реєстрації.
- **Перехід до вед-додатку:** За результатами перевірки користувач переходить до додатку.

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

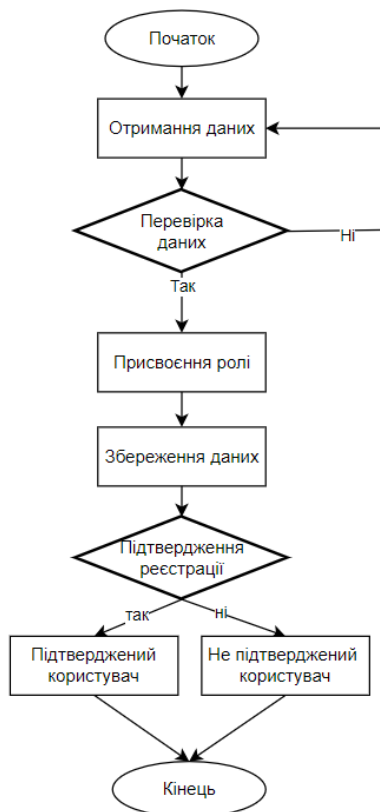


Рисунок 2.5 Діаграма активностей для процесу реєстрації користувача.

Наступним кроком буде логічно розглянути процес оформлення замовлення для авторизованого користувача.

- **Отримання даних:** Система отримує та демонструє наявні товари.
- **Сторінка товару:** Система отримує та демонструє наявну інформацію про обраний товари.
- **Додавання до кошику:** Додавання відповідного товару до кошику
- **Перевірка на авторизацію:** Система перевіряє чи авторизований користувач
- **Створення замовлення:** Створюється нове замовлення з інформацією про товари, користувача та загальну суму.
- **Перевірка даних замовлення:** система перевіряє чи правильно сформоване замовлення
- **Оплата товару:** Користувач оплачує товар
- **Підтвердження замовлення:** При успішній оплаті, замовлення користувача підтверджується

- **Переведення замовлення в очікування:** При провалі оплати замовлення переводиться в статус очікування, з можливістю оплатити потім.

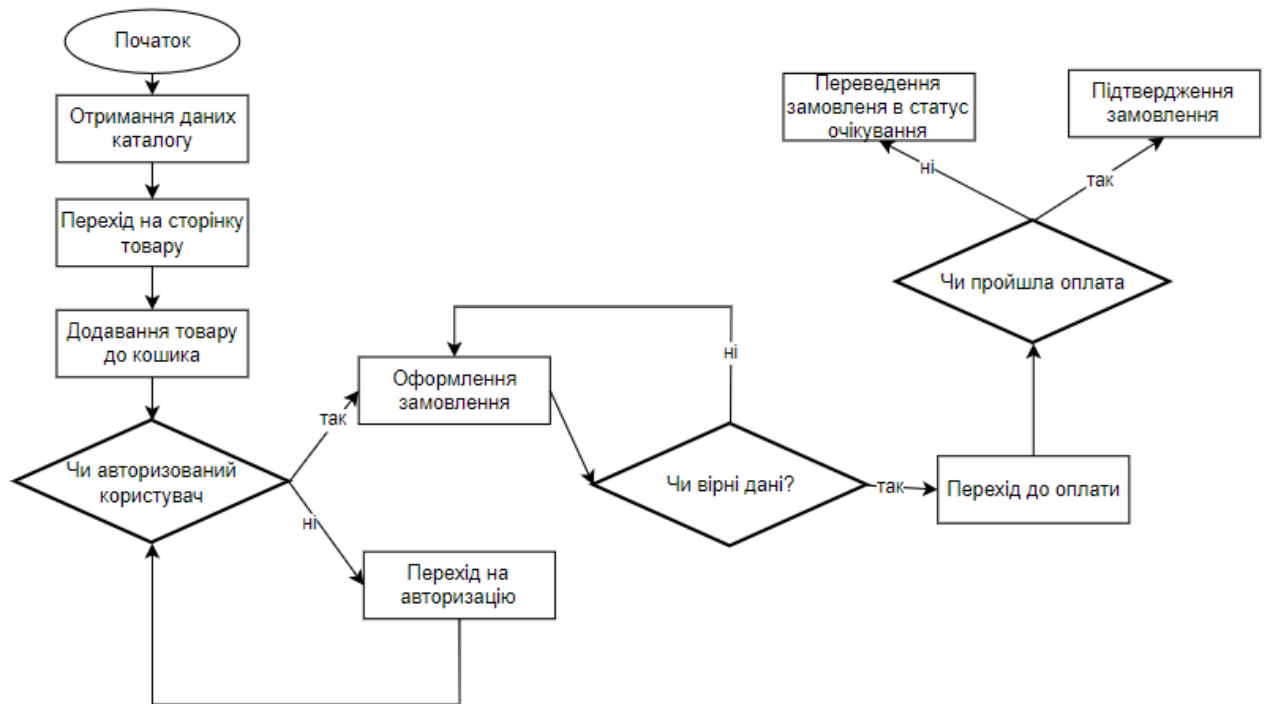


Рисунок 2.6 Діаграма активностей для процесу оформлення замовлення

Розроблені алгоритми забезпечують основні функціональні можливості інтернет-магазину музичних інструментів, включаючи реєстрацію та авторизацію користувачів, управління товарами, обробку замовлень та взаємодію з базою даних. Це дозволяє створити гнучку, масштабовану та ефективну систему, яка задовольняє вимоги користувачів та адміністрації.

2.4. Архітектура програмного комплексу

Архітектура програмного комплексу інтернет-магазину музичних інструментів базується на принципах розділення відповідальностей, що забезпечують чітку структурованість, модульність та масштабованість системи. Основними патернами проектування, використаними в архітектурі, є MVC (Model-View-Controller) та Repository.

Патерн MVC розділяє програму на три взаємопов'язані компоненти: модель, представлення та контролер. Через розмір проекту та ще більше розділення функціоналу, у проекті від цього патерну залишився лише принцип, оскільки функції моделі на себе забрав цілий фреймворк, а контроллери були розділені багато підкласів та патернів. Нижче наведена просунута реалізація патерну на прикладі сутності товару.(див.рис 2.6.).

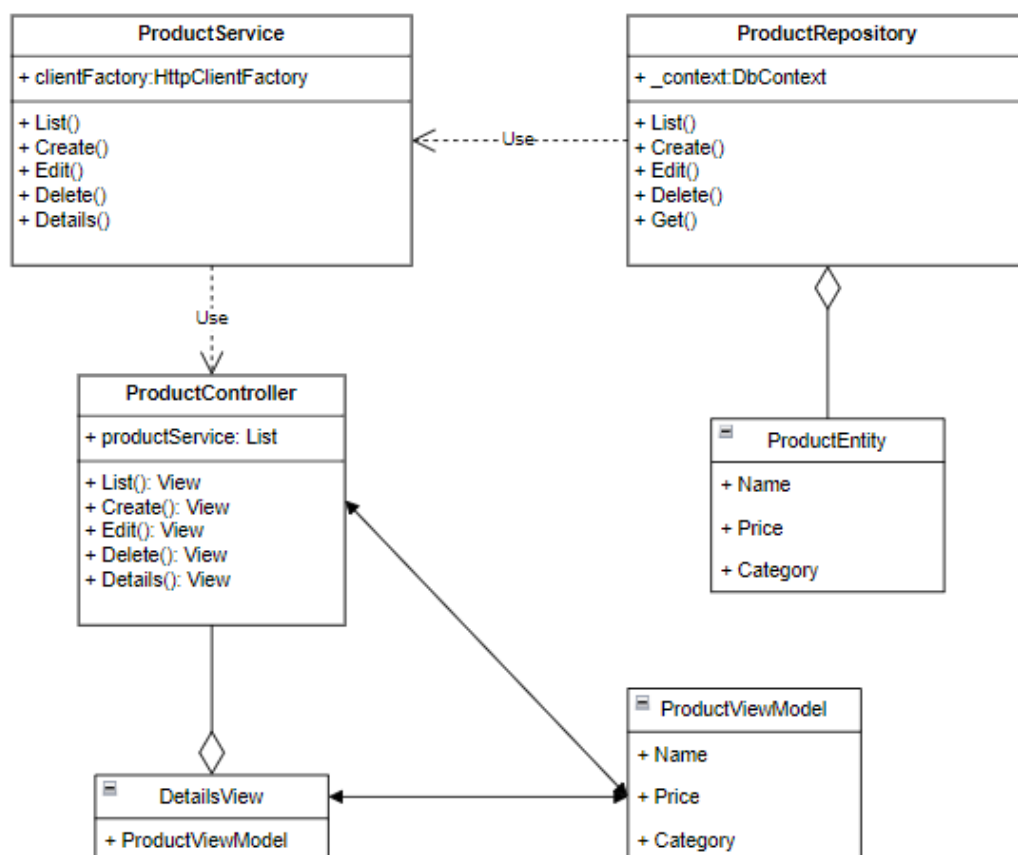


Рисунок 2.7 Діаграма прикладу реалізації патерну MVC.

Патерн Repository надає абстракцію для доступу до даних, дозволяючи працювати з ними, не залежачи від конкретної реалізації сховища даних. Це сприяє кращій організації коду, підвищенню тестованості та зменшенню дублювання коду.

- **Інтерфейс репозиторію:** Визначає контракт для виконання операцій над даними, таких як отримання, збереження, оновлення та видалення.
- **Реалізація репозиторію:** Реалізує інтерфейс та виконує конкретні операції над даними, зокрема доступ до бази даних через ORM (наприклад, Entity Framework Core). (див.рис 2.7.).

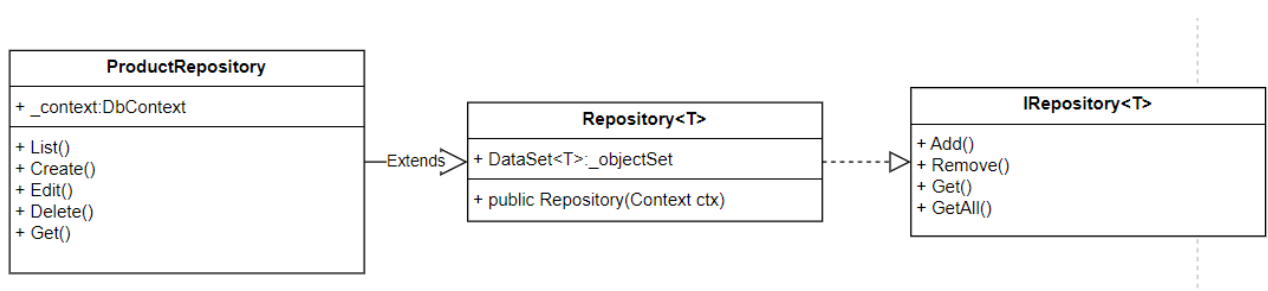


Рисунок 2.8 Діаграма прикладу реалізації патерну Repository.

Використання патернів MVC та Repository дозволяє створити гнучку, масштабовану та легко підтримувану архітектуру програмного комплексу інтернет-магазину. Патерн MVC сприяє розділенню відповідальностей, роблячи систему більш структурованою, тоді як патерн Repository забезпечує абстракцію для доступу до даних, підвищуючи тестованість та реюзабельність коду.

2.5. Реалізація функціоналу інтернет магазину музичних інструментів

Метод **List** – підготовує сторінку для відображення основних товарів. Він надає інформацію для відображення категорій а також фільтрацій та сортуванн.

Він приймає два параметри: «name» (назва продукту для пошуку) та «sortOrder» (порядок сортування). Параметр s»ortOrder» визначає, як буде відсортований список продуктів за іменем, ціною чи категорією.

```
Public async Task<IActionResult> List(Guid? categoryId)
```

Рисунок 2.9 Оголошення методу **List**

Спочатку метод перевіряє чи було обрано категорію товарів, якщо наявна категорія, вона передається на представлення. Відбувається налаштування змінних ViewBag для передачі параметрів сортування в представлення. Виконується виклик сервісу _productService для отримання списку продуктів з сервера. Результат зберігається у змінній «response».

```
APIResponse response = new();
ViewData["Category"] = categoryId;
if (categoryId.HasValue)
{
    CategoryViewModel category = new();
    response = await _categoryService.GetAsync<APIResponse>(categoryId.Value,
HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
    {
        category =
JsonConvert.DeserializeObject<CategoryViewModel>(Convert.ToString(response.Result));
    }
    ViewData["CategoryName"] = category.ParentName + " > " + category.Name;
}
else
{
    ViewData["CategoryName"] = "List";
}
```

Рисунок 2.10 Перевірка категорій.

Далі запитам до API додатку дістаються дані про категорії та бренди. Виконується виклик сервісу «_productService» для отримання списку продуктів з сервера. Результат зберігається у змінній response.

```
List<CategoryViewModel> categories = new();
response = await
_categoryService.GetAllAsync<APIResponse>(HttpContext.Session.GetString(SD.SessionTo
ken));
if (response != null && response.IsSuccess)
{
```

```

        categories =
JsonConvert.DeserializeObject<List<CategoryViewModel>>(Convert.ToString(response.Result));
    }

    List<BrandViewModel> brandss = new();
    response = await
_brandService.GetAllAsync<APIResponse>(HttpContext.Session.GetString(SD.SessionToken
));
    if (response != null && response.IsSuccess)
    {
        brandss =
JsonConvert.DeserializeObject<List<BrandViewModel>>(Convert.ToString(response.Result
));
    }

```

Рисунок 2.11 Отримання інформації про бренди та категорії.

Під кінець отримана інформація передається на представлення. Повертається з відсортованим та/або відфільтрованим списком продуктів.

```

IndexViewModel viewModel = new IndexViewModel
{
    Categories = categories,
    Brands = brandss
};
return View(viewModel);

```

Рисунок 2.12 Передача інформації на представлення.

Процес отримання даних відбувається через звернення до сервісів категорій та брендів, після цього дані передаються на представлення. Завдяки цьому одразу після завантаження сторінки користувач побачить вже заповнену колонку з сортуванням та фільтрами.

Метод **_PartialList** – відповідає за безпосередньо відображення самих товарів. В ньому також наявна обробка даних з форми фільтрації та впорядкування отриманих даних. Даний метод створений для того щоб дати можливість динамічно оновлювати сторінку. Він отримує основні дані, сортує та фільтрує, проводить пошук необхідних товарів.

Основні кроки методу:

- Встановлення розміру сторінки.
- Отримання списку всіх продуктів.
- Фільтрація продуктів за параметром пошуку.
- Фільтрація продуктів за категорією.
- Додаткова фільтрація за даними з форми.

- Реалізація пагінації.
- Повернення часткового представлення.

```

Public async Task<IActionResult> PartialList(Guid? categoryId, int page = 1,
IformCollection formCollection = null)
{
    int pageSize = 3; // elems on page
    List<ProductViewModel> products = new();
    var response = await
_productService.GetAllAsync<APIResponse>(HttpContext.Session.GetString(SD.SessionTok
en));
    if (response != null && response.IsSuccess)
    {products =
JsonConvert.DeserializeObject<List<ProductViewModel>>(Convert.ToString(response.Resu
lt));
    }
    string searchParam = formCollection["searchParam"];
    if (categoryId.HasValue) { ViewData["Category"] = categoryId;
    response = await
_productService.GetAllByCategoryAsync<APIResponse>(categoryId.Value,
HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
    {
        products =
JsonConvert.DeserializeObject<List<ProductViewModel>>(Convert.ToString(response.Resu
lt));}}
    if (formCollection.Count > 0)
    {
        var form = formCollection["formCollection"];
        var brands = formCollection["selectedBrands"].ToList();
        decimal min = Convert.ToDecimal(formCollection["min"]);
        decimal max = Convert.ToDecimal(formCollection["max"]);
        var order = formCollection["order"].ToString();
        products = products.Where(prod => (prod.RetailPrice > min && prod.RetailPrice <
max)).ToList();
        if (brands.Count > 0)
        {
            products = products.Where(prod => brands.Any(brand =>
brand.Equals(prod.BrandName))).ToList();
        }
    }
    var count = products.Count();
    var items = products.Skip((page - 1) * pageSize).Take(pageSize).ToList();
    PageViewModel pageViewModel = new PageViewModel(count, page, pageSize);

    ListViewModel model = new ListViewModel();
    model.Products = items;
    model.PageViewModel = pageViewModel;
    return PartialView("_PartialList", model);
}

```

Рисунок 2.13 Метод **_PartialList**.

Цей метод отримує всі продукти та застосовує різні фільтри на основі параметрів пошуку, категорії та додаткових критеріїв з форми. Він також підтримує сортування та пагінацію. Результат у вигляді часткового представлення повертається до клієнта. Це дозволяє динамічно оновлювати

		Головецький Є.В.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

список продуктів на веб-сторінці без повного перезавантаження сторінки.

Метод **Details** – відповідає за вікно з конкретним товаром. Тут відбувається процес пошуку та отримання товару з бази даних. Тут же підтягуються відгуки та пов'язані товари, і після формування моделі дані передаються на представлення.

Основні кроки методу:

- Ініціалізація моделі детальної інформації::
- Ініціалізація об'єкта продукту::
- Отримання інформації про продукт:
- Отримання списку продуктів з тієї ж категорії:
- Отримання відгуків про продукт:
- Заповнення моделі даними:
- Повернення представлення з моделлю:

```
public async Task<IActionResult> Details(Guid id)
{
    DetailsViewModel model = new DetailsViewModel();
    ProductViewModel product = new();
    var response = await _productService.GetAsync<APIResponse>(id,
        HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
        product =
        JsonConvert.DeserializeObject<ProductViewModel>(Convert.ToString(response.Result));
    List<ProductViewModel> products = new();
    response = await
    _productService.GetAllByCategoryAsync<APIResponse>(product.CategoryId,
        HttpContext.Session.GetString(SD.SessionToken));

    if (response != null && response.IsSuccess)
        products =
        JsonConvert.DeserializeObject<List<ProductViewModel>>(Convert.ToString(response.Result));

    List<ReviewViewModel> reviews = new();
    response = await
    _reviewService.GetAllForProductAsync<APIResponse>(product.Id, HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
        reviews =
        JsonConvert.DeserializeObject<List<ReviewViewModel>>(Convert.ToString(response.Result));
    model.Product = product;
    model.Products = products;
    model.Reviews = reviews; return View(model);
}
```

Рисунок 2.14 Метод **Details**

Це забезпечує користувача повною інформацією про вибраний продукт, включаючи його деталі, відгуки інших користувачів та інші продукти з тієї ж категорії.

		Головецький Є.В.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

За процес додавання товару в кошик відповідає метот **Add**. Спочатку отримується унікальний номер корзини, він або співпадає з користувачем, або якщо користувач не авторизований створюється автоматично. Далі отримуться продукт який необхідно додати і формується вже елемент корзини.

Основні кроки методу:

- Отримання ідентифікатора користувача::
- Отримання інформації про продукт::
- Отримання інформації про елемент кошика:
- Перевірка наявності продукту в кошику та додавання його, якщо він відсутній:
- Отримання відгуків про продукт:
- Перенаправлення на список товарів у кошику:

```
public async Task<IActionResult> Add(Guid id)
{
    var userId = Guid.Parse(GetCartId());
    if (User.Identity.IsAuthenticated)
    {
        userId = Guid.Parse(_userManager.GetUserId(User));
    }
    ProductViewModel product= new();
    var response = await
    _productService.GetAsync<APIResponce>(id,HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess) product =
    JsonConvert.DeserializeObject<ProductViewModel>(Convert.ToString(response.Result));
    if (product == null)
        return Redirect("~/CartItem/List");
    CartViewModel cartView = new();
    response = await _cartItemService.GetAsync<APIResponce>(product.Id,
    HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
    {
        cartView =
    JsonConvert.DeserializeObject<CartViewModel>(Convert.ToString(response.Result));
    }if (cartView == null || cartView.CartId != userId)
    {
        cartView = new CartViewModel
        {
            Id = Guid.NewGuid(),
            CartId = userId,
            Quantity = 1,
            DateCreated = DateTime.Now,
            ProductId = product.Id,
            ProductName = product.Name;
        };
        response = await _cartItemService.CreateAsync<APIResponce>(cartView,
    HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {
            TempData["success"] = "CartItem created successfully";
            return RedirectToAction(nameof(List));
        }return
        Redirect("~/CartItem/List");
    }
```

Рисунок 2.15 Метод **Add**.

Отже, метод **Add** відповідає за додавання продукту до кошика користувача. Він спочатку отримує ідентифікатор користувача, перевіряє наявність продукту та елемента кошика. Якщо продукт ще не додано до кошика, метод створює новий елемент кошика та зберігає його. У разі успішного додавання продукту до кошика, користувача перенаправляють на сторінку зі списком товарів у кошику з відповідним повідомленням про успіх.

Метод **MakeOrder** викликається коли користувач додав усі бажані товари до корзини та продовжив оформлювати замовлення. Нове замовлення формується з урахуванням введених особистих даних користувача.

Основні кроки методу:

- Отримання користувача: Метод починається з отримання поточного автентифікованого користувача за допомогою «_userManager»
- Отримання елементів кошика: Викликається «_cartItemService», щоб отримати всі елементи кошика для даного cartId. Якщо відповідь успішна, елементи кошика десеріалізуються в список cartItems.
- Ініціалізація моделі замовлення: Створюється новий об'єкт «CartOrderViewModel» і заповнюється інформацією про замовлення «OrderViewModel», включаючи ім'я клієнта, телефон і електронну пошту, отримані з користувача, а також елементи кошика.
- Розрахунок підсумкової суми замовлення: Підсумкова сума замовлення «Subtotal» розраховується шляхом множення кількості кожного елемента на його ціну і сумування результатів.
- Повернення представлення з моделлю: Метод повертає представлення «MakeOrder»

```
public async Task<IActionResult> MakeOrder(Guid? cartId)
{
    var user = await _userManager.GetUserAsync(User);
    if (cartId.HasValue)
        ViewData["cartId"] = cartId;
    var cartItems = new List<CartItemViewModel>();
    var response = await _cartItemService.GetAllAsync<ApiResponse>(cartId.Value,
        HttpContext.Session.GetString(SD.SessionToken));
}
```

```

if (response != null && response.IsSuccess)
{
    cartItems =
JsonConvert.DeserializeObject<List<CartItem>>(Convert.ToString(response.Result)
);} CartOrderViewModel model = new CartOrderViewModel();
model.Order = new OrderViewModel();
model.Order.ClientName = user.FullName;
model.Order.ClientPhone = user.PhoneNumber;
model.Order.Email = user.Email;
model.CartItems = cartItems;
ViewBag.Subtotal = 0;
foreach (var item in model.CartItems)
{
    ViewBag.Subtotal += item.Quantity * item.Price;
}

return View("MakeOrder", model)}

```

Рисунок 2.16 Метод **MakeOrder**.

Після оформлення переходу на сторінку оплати та введення платіжних даних, запит відправляється на платіжну систему «LiqPay» яка за результатами оплати видає відповідний результат. Потім замовлення переходить в статус оплаченого.

Основні кроки методу:

- Отримання відповіді від платіжного сервісу: Метод починається з виклику «_liqpayService» для відправки платіжних даних і отримання відповіді, яка десеріалізується в словник.
- Обробка помилки оплати: Якщо результат відповіді містить «result»: «error», у «ViewBag» зберігаються коди помилки і її опис.
- Обробка успішної оплати: Якщо результат відповіді містить «result»: «ok»
- Повернення результату: Метод завершується поверненням представлення «Result».

Метод Checkout обробляє процес оплати, взаємодіючи з платіжним сервісом та сервісами замовлень і статусів. У випадку успішної оплати, статус замовлення оновлюється, а користувачу відображається повідомлення про успішну транзакцію.

```

Public async Task<IActionResult> Checkout(CheckoutViewModel model,string jobId="")
{
    var response = JsonConvert.DeserializeObject<Dictionary<string, string>>(await
    _liqpayService.SendPayment(model.CardNumber));

    if (response["result"] == "error")
    {
        ViewBag.ErrorCode = response["err_code"];
        ViewBag.ErrorDesc = response["err_description"];
    }
    if (response["result"] == "ok")
    {
        OrderViewModel order = new();
        var response = await
        _orderService.GetAsync<APIResponse>(model.RelatedOrderId,
        HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {
            order =
            JsonConvert.DeserializeObject<OrderViewModel>(Convert.ToString(response.Result));
        }
        StatusViewModel status = new();
        response = await _statusService.GetAsync<APIResponse>("Sended",
        HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {
            status =
            JsonConvert.DeserializeObject<StatusViewModel>(Convert.ToString(response.Result));
        }

        order.StatusId = status.Id;
        response = await _orderService.UpdateAsync<APIResponse>(order,
        HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {
        }
        ViewBag.ErrorCode = "Payment successful";
        ViewBag.ErrorDesc = "Traction has reached its destination";
    }

    return View("Result");
}

```

Рисунок 2.17 Метод **Checkout** .

Під час використання всіх цих методів постійно виконувалось підтягування даних з бази, відповідні сервіси відправляли запити на API проекту. Тому було розроблено універсальний метод **SendAsync** який виконує запити до API та автоматично підлаштовується під кожен сервіс додатку.

Метод **SendAsync** виконує асинхронний HTTP-запит до API і обробляє отриману відповідь. Він використовує політику повторних спроб для обробки можливих помилок мережі. Основні кроки методу:

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

- Ініціалізація HTTP-клієнта: Метод починається зі створення клієнта «HTTP» з назвою «StoreApi»
- Створення об'єкта «HttpRequestMessage»: Створюється нове повідомлення HTTP-запиту, задається заголовок Асепт для прийому JSON-відповідей, встановлюється «URI» запиту. Якщо в запиті є дані, вони серіалізуються в JSON і додаються до тіла запиту
- Додавання контенту до запиту (якщо є дані):
- Встановлення методу HTTP-запиту (GET, POST, PUT, DELETE) В залежності від типу API-запиту встановлюється відповідний метод.
- Політика повторних спроб (Retry Policy):
- Виконання HTTP-запиту з політикою повторних спроб: Відправка запиту виконується з використанням політики повторних спроб у разі виникнення помилок, таких як «HttpRequestException».
- Обробка відповіді від API: Якщо відповідь успішна, вона десеріалізується в об'єкт APIResponse. Якщо статус відповіді є «BadRequest» або «NotFound», повертається відповідь з помилкою. У випадку помилки під час десеріалізації, повертається об'єкт відповіді з помилкою. В кінці успішно десеріалізована відповідь повертається як результат.

```

Public async Task<T> SendAsync<T>(APIRequest apiRequest)
{
    try
    {
        var client = httpClient.CreateClient("StoreApi");
        HttpRequestMessage message = new HttpRequestMessage();
        message.Headers.Add("Accept", "application/json");
        message.RequestUri = new Uri(apiRequest.Url);
        if (apiRequest.Data != null){
            message.Content = new
StringContent(JsonConvert.SerializeObject(apiRequest.Data),
Encoding.UTF8, "application/json");}
        switch (apiRequest.ApiType)
        {case SD.ApiType.POST:message.Method = HttpMethod.Post;break;
        case SD.ApiType.PUT:message.Method = HttpMethod.Put;break;
        case SD.ApiType.DELETE:message.Method = HttpMethod.Delete;break;
        default:message.Method = HttpMethod.Get;break;}
        HttpResponseMessage apiResponseMessage = null;
        if (!string.IsNullOrEmpty(apiRequest.Token))
        {DefaultRequestHeaders.Authorization = new
System.Net.Http.Headers.AuthenticationHeaderValue("Bearer",apiRequest.Token );}
        string apiContent = "";
    }
}

```

```

var policy = Policy.Handle<HttpRequestException>().RetryAsync(3);
await policy.ExecuteAsync(async () =>
{
    apiResponseMessage = await client.SendAsync(message);
    apiContent = await apiResponseMessage.Content.ReadAsStringAsync();
});
try
{
    ApiResponse apiResponse =
    JsonConvert.DeserializeObject<ApiResponse>(apiContent);
    if (apiResponseMessage.StatusCode==System.Net.HttpStatusCode.BadRequest
    ||apiResponseMessage.StatusCode == System.Net.HttpStatusCode.NotFound){
        ApiResponse.StatusCode = System.Net.HttpStatusCode.BadRequest;
        ApiResponse.IsSuccess = false;
        var res = JsonConvert.SerializeObject(ApiResponse);
        var returnObj = JsonConvert.DeserializeObject<T>(res);
        return returnObj;}}
    catch (Exception){var expectionResponse =
    JsonConvert.DeserializeObject<T>(apiContent);return expectionResponse;}
    var ApiResponse = JsonConvert.DeserializeObject<T>(apiContent);
    return ApiResponse ;}
catch (Exception e)
{var dto = new ApiResponse
{
    ErrorMessage = new List<string> { Convert.ToString(e.Message) },
    IsSuccess = false,
};

var res = JsonConvert.SerializeObject(dto);
var ApiResponse= JsonConvert.DeserializeObject<T>(res);
return ApiResponse;
}
}

```

Рисунок 2.18 Метод **SendAsync** .

Метод **SendAsync** забезпечує відправку HTTP-запитів до API з налаштуванням заголовків, обробкою токенів авторизації, повторними спробами у разі помилок та обробкою відповідей API. Він надійно обробляє як успішні відповіді, так і помилки, забезпечуючи правильну десеріалізацію даних та відповідні повідомлення про помилки.

Одною з особливостей проекту є можливість залишати відгуки на товари. Виконати таку дію може тільки авторизований користувач. Процес починається на сторінці товару, де користувач може залишити свій коментар та оцінку. Додавання відгуку до бази виконує метод **AddReview**.

Основні кроки:

- Перевірка валідності рейтингу
- Отримання інформації про користувача
- Отримання відгуків користувача
- Перевірка наявності відгуку для даного продукту

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

- Створення нового відгуку
- Збереження відгуку
- Обробка невдалої спроби створення відгуку

```
public async Task<IActionResult> AddReview(string text,int rate, Guid productId)
{
    if (rate>5||rate<0)
    {
        return Ok("Rate only between 0 and 5");
    }
    var user = await _userManager.GetUserAsync(User);
    List<ReviewViewModel> reviews = new();
    var response = await _reviewService.GetAllForUserAsync<APIResponse>(user.Id,
HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
    {
        reviews =
JsonConvert.DeserializeObject<List<ReviewViewModel>>(Convert.ToString(response.Result));
    }
    if (reviews.Count(r=>r.ProductId== productId)>0)
    {
        return Ok("You cant add more reviews");
    }
    ReviewViewModel review = new ReviewViewModel();
    review.Text = text;
    review.Rate = rate;
    review.Date = DateTime.Now;
    review.Likes = 0;
    review.UserName = _userManager.GetUserName(User);
    review.UserId = Guid.Parse(_userManager.GetUserId(User));
    review.ProductId = productId;
    response = await _reviewService.CreateAsync<APIResponse>(review,
HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
    {
        TempData["success"] = "Review created successfully";
        return Ok("Review created successfully");
    }
    return RedirectToAction("ReviewsList",new { productId = productId});
}
```

Рисунок 2.19 Метод **AddReview**

Відправлені запити приймає API проекту. Для прикладу приведено GET метод який відповідають за отримання конкретного товару з бази даних **GetProduct**.

Опис основних кроків.

- Перевірка вхідного параметра id. Якщо він є порожнім (null), метод повертає відповідь з кодом помилки BadRequest, означаючи невірні дані запиту.
- Виконується асинхронний запит до _productRepository, щоб отримати продукт за його ідентифікатором.

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

- Після отримання продукту перевіряється, чи він існує. Якщо продукт не знайдено, метод повертає відповідь з кодом помилки NotFound
- Якщо продукт успішно знайдено, він конвертується в об'єкт ProductViewModel, який потім встановлюється як результат відповіді
- Завершення методу з поверненням об'єкту Ok(_response), де _response містить результат операції та код статусу.

```
Public async Task<ActionResult<APIResponse>> GetProduct(Guid id)
{
    if (id == null)
    {
        return BadRequest();
    }
    var product = await _productRepository.GetItemAsync(p => p.Id == id,
includeProps: "Brand,Category,Assets,Favourites,Reviews");
    if (product == null)
    {
        return NotFound();
    }
    _response.Result = new ProductViewModel(product);
    _response.StatusCode = System.Net.HttpStatusCode.OK;

    return Ok(_response);
}
```

Рисунок 2.20 Методи та **GetProduct**

Відповідні методи та API прописані для кожної сутності в базі даних.

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Висновки до другого розділу

У другому розділі детально розглянуто проектування та реалізацію інтернет-магазину музичних інструментів.

Були визначені основні функціональні можливості системи та створено відповідні діаграми прецедентів, які відображають взаємодію користувачів і адміністраторів з системою. Об'єктно-орієнтована структура системи описує сутності та їх зв'язки, що забезпечує чітке розуміння структури даних та логіки взаємодії між компонентами системи.

Також були розроблені структури бази даних, які включають таблиці для зберігання інформації про користувачів, товари, замовлення та інші важливі об'єкти системи. Це забезпечує надійну та ефективну організацію даних, необхідних для функціонування інтернет-магазину.

Була розглянута архітектура системи на основі патернів MVC та Repository. Патерн MVC забезпечує розділення відповідальностей між різними компонентами системи (модель, представлення, контролер), що підвищує її гнучкість та зручність у підтримці. Патерн Repository надає абстракцію для роботи з даними, що спрощує доступ до бази даних та підвищує тестованість коду.

Отже, другий розділ демонструє всебічний підхід до розробки та впровадження інтернет-магазину музичних інструментів, забезпечуючи міцний фундамент для створення продуктивного, масштабованого та користувацьки орієнтованого програмного рішення.

		Головецький Є.В.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ДОДАТКОМ

3.1. Порядок розгортання та налаштування системи

Перед розгортанням системи необхідно підготувати відповідне середовище, яке включає налаштування серверів, бази даних та всіх необхідних компонентів. Основні етапи підготовки середовища:

Вибір хостингу: Визначення оптимального хостингового провайдера, що забезпечить необхідний рівень продуктивності, надійності та безпеки.

Налаштування серверів: Встановлення та налаштування веб-серверів (наприклад, IIS[\[19\]](#) для ASP.NET Core), конфігурація необхідного програмного забезпечення та служб.

Підготовка бази даних: Створення бази даних у середовищі SQL Server, налаштування підключення та застосування початкової структури таблиць і даних.

Під час аналізу рішень для хостингу, вибір впав на Azure. Azure є продуктом Microsoft, що забезпечує глибоку інтеграцію з .NET та іншими інструментами розробки від Microsoft, такими як Visual Studio та Azure DevOps, що і стало основним фактором при виборі цього інструменту.

Налаштування серверів також відбувається через сервіси Azure. Зручні інструменти керування підписками та робочими групами дозволяють автоматично розміщувати сервери, обирати віртуальні машини для них та виконувати конфігурацію ресурсів доступних серверу.

Бази даних проекту також були розміщені на сервери який виділив Azure. Підключення до бази відбувається завдяки строкам підключення. Створення бази забезпечує EntityFramework який керується з вікна терміналу. При оновленні бази даних створені міграції автоматично змінюють структуру розміщеної бази, що не потребує додаткових дій від розробника.

Оскільки файли, а саме картинки для товарів неможливо зберігати в стандартній базі даних, потрібно використовувати стороннє рішення для хостингу. Azure також надає зручний сервіс для цього. Сервіс BlobStorage.

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Зберігання ключів та іншої чутливої інформації потребує окремого сховища, яке може забезпечити швидкий та зручний доступ і в одночас повну безпеку інформації яка зберігається. Таким рішенням є Azure KeyVault.

Також продемонстровано створені рішення на сервісі Azure.

☐	 Azure subscription 1	Subscription
☐	 musiicstoreclient	App Service
☐	 musiicstorewebapi	App Service
☐	 MusicStoreKeyVault	Key vault
☐	 WebStoreGroup	Resource group
☐	 Store_Identity	SQL database
☐	 DefaultResourceGroup-CUS	Resource group
☐	 ASP-WebStoreGroup-979f	App Service plan
☐	 blobformusicstoreassets	Storage account
☐	 musicstoredbserver	SQL server
☐	 Store_Data	SQL database

Рисунок 3.1 Створені рішення на сервісі Azure.

3.2. Інтерфейс та порядок роботи .

Після переходу на сайт, користувач побачить каталог з наявними товарами. Інтерфейс магазину складається з панелі навігації, через яку користувач може перейти на головну сторінку, кошик, або зареєструватись або увійти до аккаунту.

На головній сторінці також присутнє вікно з сортуванням та фільтрами. Також реалізований пошук та вибірка за категоріями.

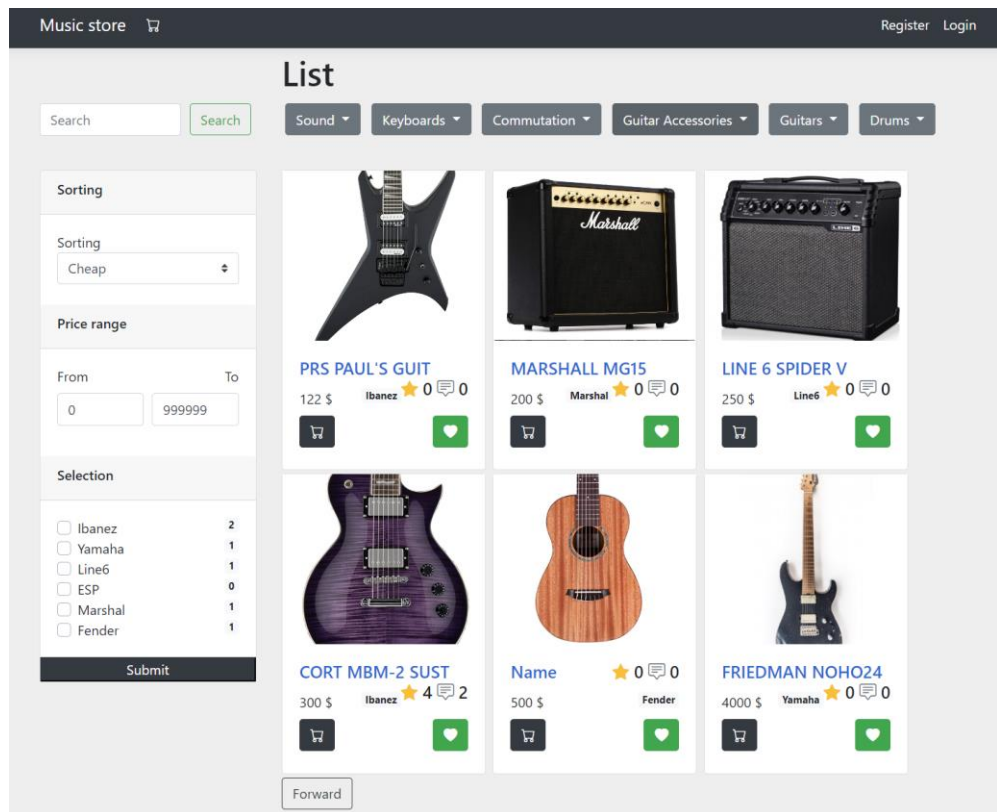


Рисунок 3.2 Загальний вигляд каталогу з товарами.

Користувач також може увійти до вже існуючого або створити новий обліковий запис. На сторінці логіну (рис. 3.4) користувачу буде запропоновано декілька варіантів входу, наприклад увійти за існуючим Google аккаунтом або перейти до створення нового аккаунту в системі (рис. 3.3). На сторінці входу також доступна функція «Забув пароль».

Рисунок 3.3.Вікно реєстрації.

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

Рисунок 3.4.Вікно входу.

При переході на сторінку товару користувач побачить який товар представлено на сторінці, Ліворуч розміщене зображення гітари, що дає користувачам візуальне уявлення про товар. Праворуч від зображення міститься розділ **About**, де наведено короткий опис гітари.

У тексті підкреслюється висока якість гітари, її елегантний дизайн та функціональні особливості, такі як комфортний гриф та яскраве звучання. У верхньому правому куті сторінки присутній іконка зі зіркою та значок "5", що відображає рейтинг або кількість уподобань товару.

Під основною інформацією розміщено розділ **Description** з відео, вбудованим з YouTube, де демонструється гра на цій гітарі. Відео надає користувачам можливість побачити і почути гітару в дії, що допомагає краще зрозуміти її характеристики. Під відео розташований список технічних характеристик гітари.

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

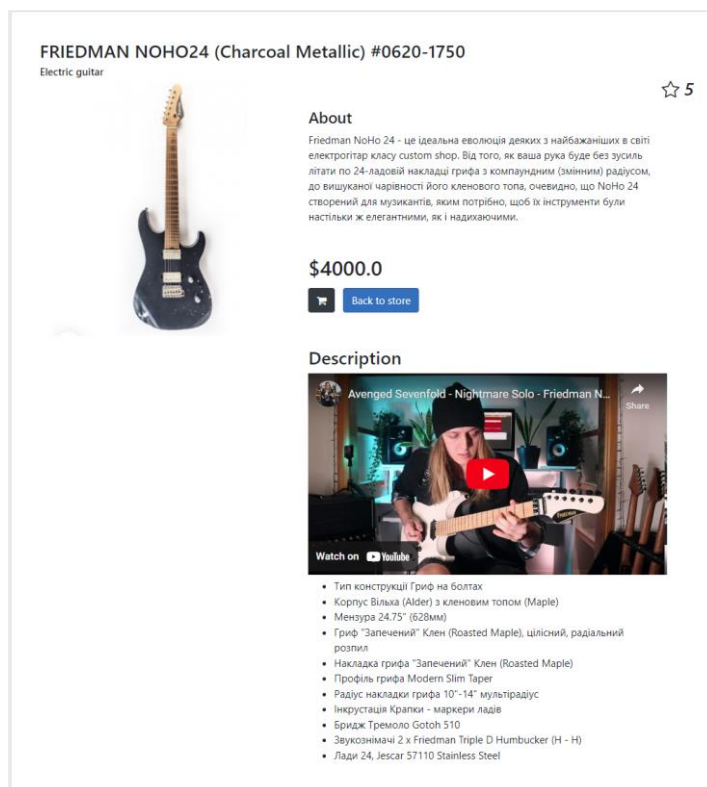


Рисунок 3.5.Вигляд сторінки товару.

Нижче на цій сторінці розміщена секція з відгуками. Користувачі мають можливість залишити відгук з оцінкою від 0 до 5 для товару. Є можливість відсортувати відгуки за вподобаннями інших користувачів.

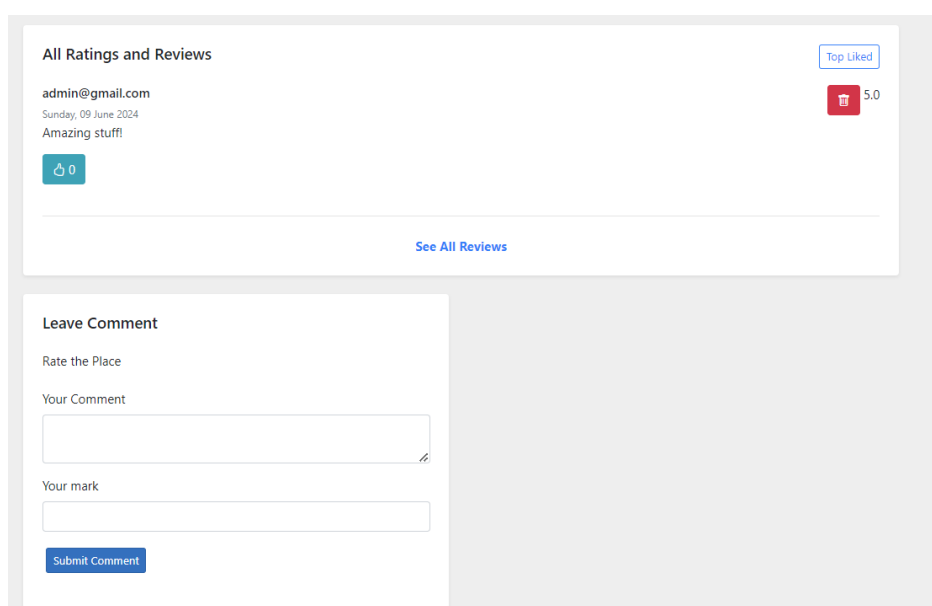


Рисунок 3.6.Вигляд секції з відгуками.

Як на головному каталозі так і на сторінці товару користувач має можливість додати товар в корзину з можливістю продовжити та оформити

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

замовлення. При бажанні можна очистити корзину або повернутися назад до магазину.

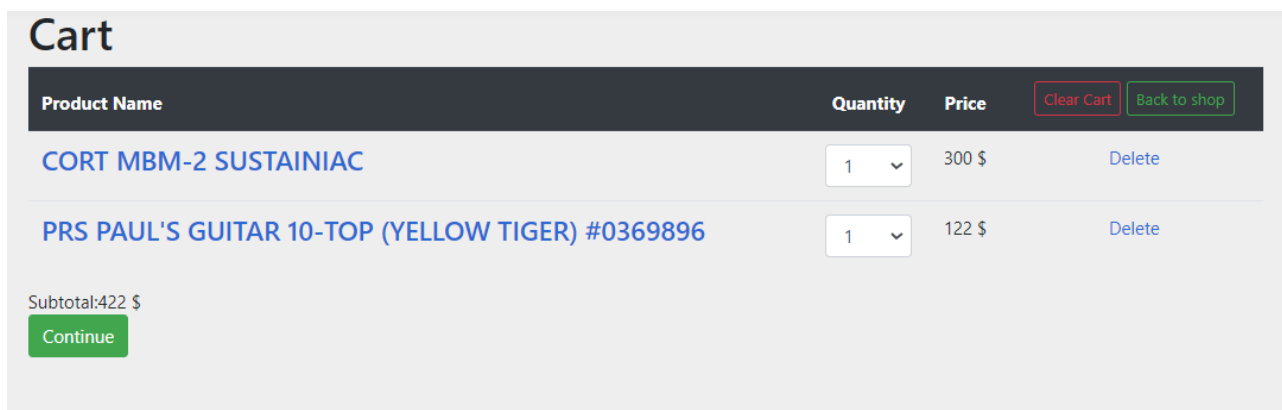


Рисунок 3.7.Корзина з товарами

При додаванні товару в корзину вискакує вікно з підтвердженням того що товар було додано.

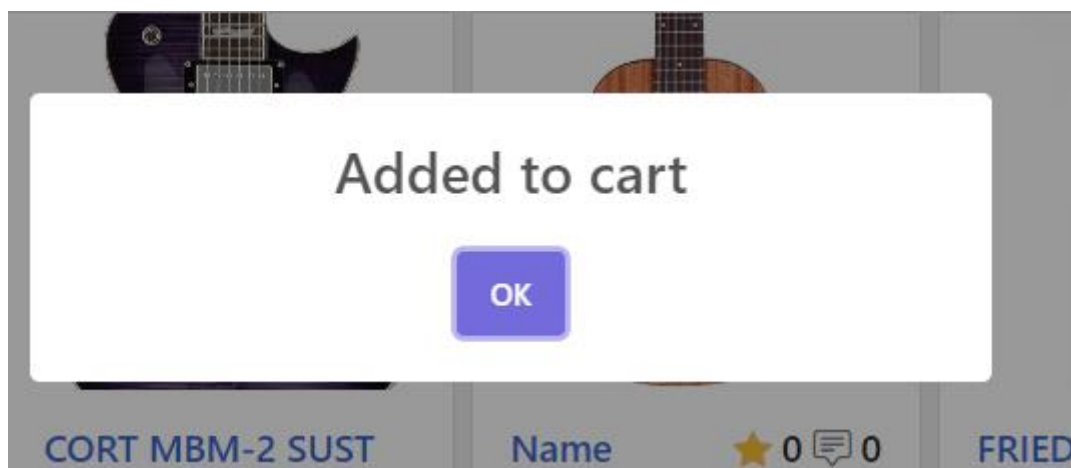


Рисунок 3.8.Повідомлення з підтвердженням

При переході до оформлення замовлення, користувач бачит форму з даними для введення та товари, їх назва та ціна, які знаходились у корзині.

У користувача є можливість оплатити одразу, або відкласти оплату на потім.

Користувач також обирає місто та відділення нової пошти для доставки.

		Головецький Є.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

User details	Your cart
ClientName Eugene Holovetskiy	CORT MBM-2 SUSTAINIAC \$300.0 Cort разом з Меттью
ClientPhone 0989838777	PRS PAUL'S GUITAR 10-TOP (YELLOW TIGER) #0369896 \$122.0 Улюблена гітара Пола
Email admin@gmail.com	MARSHALL MG15 \$200.0 Комбопідсилювач для
City Zhytomyr	Total (USD) 622 \$
NovaPoshta 3	
Go to payment Pay later	

Рисунок 3.9.Сторінка оформлення замовлення товару

Після оформлення замовлення та введення всіх даних користувачу буде запропоновано оплатити товар. Симуляція оплати проводиться через сервіс LiqPay.

Payment for order 5cb5

☒ Credit card

OwnerName

Full name as displayed on card

CardNumber

ExpirationDate

CVV

Рисунок 3.10.Сторінка оплати товару

Кожен користувач має власний профіль, де він може переглянути та змінити дані свого облікового запису, переглянути свої замовлення та список обраних товарів.

Також можна переглянути кількісь залишених користувачем відгуків та загальну кількість замовлень.

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

admin@gmail.com

Reviews 2

Orders 4

Favourite 0

Account details

Orders

Favourites

Account Details

UserName

admin@gmail.com

Email address

admin@gmail.com

Full name

Eugene Holovetskiy

PhoneNumber

0989838777

RegisterDate

mm/dd/yyyy

Save changes

Last Order Details

City:

Zhytomyr

Date:

06/02/2024

Phone:

Name:

Eugene Holovetskiy

Рисунок 3.11.Профіль користувача. Сторінками з деталями про обліковий запис

На вкладці з замовленнями можна побачити номер змовлення, дату його створення, суму та статус.

Є можливість перейти до детального перегляду, або продовжити оплату замовлення.

admin@gmail.com

Reviews 2

Orders 3

Favourite 0

Account details

Orders

Favourites

Your orders

5cb5	6/2/2024 6:57:13 PM	Pending	Pay	800.0\$
2154	6/2/2024 6:58:18 PM	Sended	Details	800.0\$
43bc	6/9/2024 4:40:22 PM	Canceled	Details	800.0\$

Рисунок 3.12.Профіль користувача. Сторінками з деталями про бліковий запис.

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

Користувачі мають можливість додавати товари до обраного. Всі обрані товари можна переглянути на відповідній сторінці профілю. Є можливість додати товар до кошика.

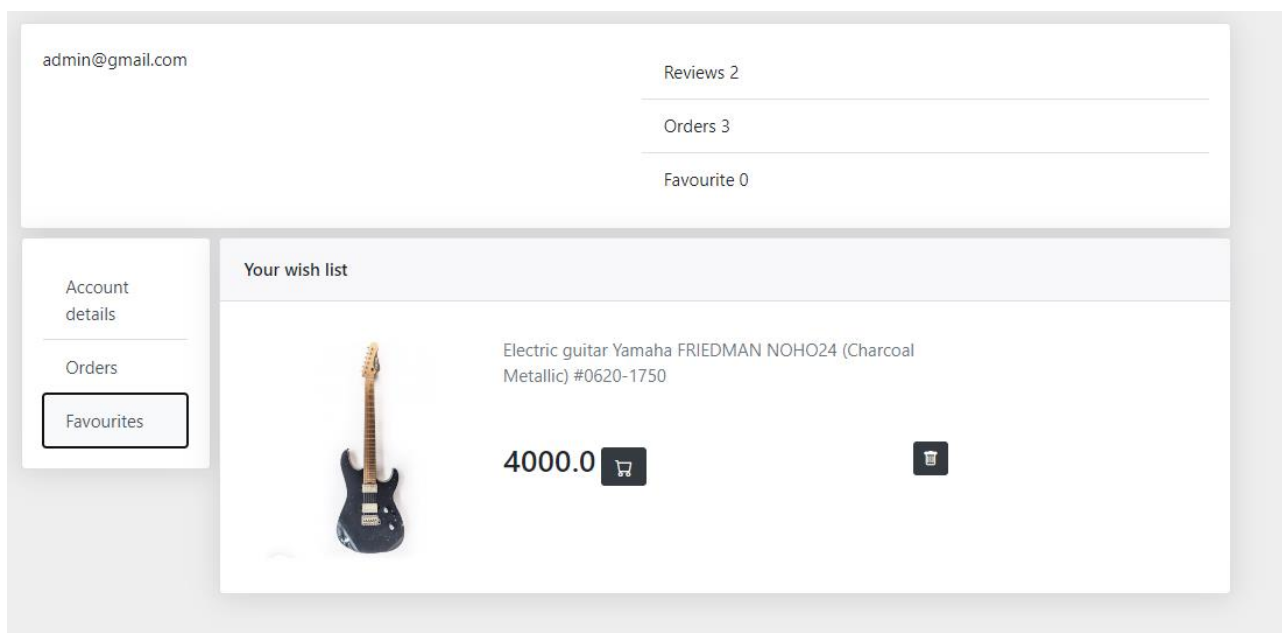


Рисунок 3.13. Профіль користувача. Сторінками з деталями про білковий запис.

3.3. Огляд інтерфейсу та інструментів адміністратора.

При використанні облікового запису адміністратора, додається доступ до панелей управління.

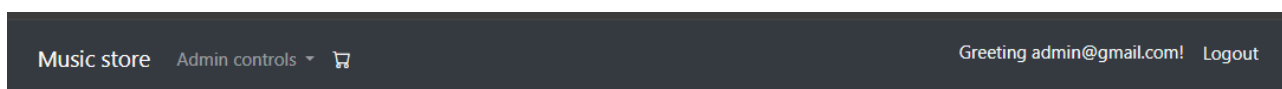


Рисунок 3.14. Змінене навігаційне меню

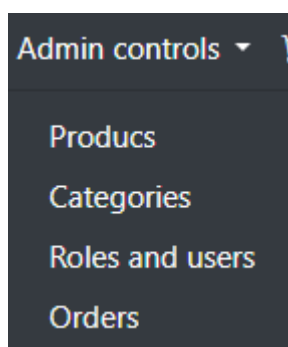


Рисунок 3.15. Посилання переходу на таблиці управління

		Головецький Є.В.			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Вакалюк Т.А.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

Для прикладу буде розібрана панель управління для товарів.

Product list					
Create New					
Search					
ID	Name	Category	RetailPrice	Brand	
eeafcd0bf-bea1-4096-2735-08dc3047778	CORT MSM-2 SUSTAINAC	Electric guitar	300.00	Ibanez	Edit Delete
74676bc9-38d7-4029-4d6e-08dc831b1155	Name	Classical guitar	500.00	Fender	Edit Delete
3257147c-8007-49bc-7020-08dc889c304	PRS PAUL'S GUITAR 10-TOP (YELLOW TIGER) #0369896	Electric guitar	122.00	Ibanez	Edit Delete
905d17a9-1454-4a2b-80a4-08dc88993559	FRIEDMAN NOHO24 (Charcoal Metallic) #0620-1750	Electric guitar	4000.00	Yamaha	Edit Delete
db2733cc-2263-45e2-d059-08dc889a343b	LINE 6 SPIDER V 20 MKII	Guitar Amplifiers	250.00	Line6	Edit Delete
34fecf79-cfaa-4d91-143b-08dc889c7c39	MARSHALL MC15	Guitar Amplifiers	200.00	Marshall	Edit Delete

Рисунок 3.16.Панель управління товарами

Адмін-панель для управління товарами в інтернет-магазині надає адміністраторам зручний інтерфейс для пошуку, сортування, редагування та видалення товарів. Основні компоненти та функції панелі включають:

Поле пошуку: Розташоване у верхній частині панелі, поле пошуку дозволяє адміністраторам швидко знаходити товари за ключовими словами, такими як назва товару, категорія або бренд. Це значно спрощує процес управління великим асортиментом.

Параметри сортування: Панель надає можливість сортування товарів за різними критеріями

Кнопки дій: Для кожного товару доступні кнопки дій, що дозволяють виконувати наступні операції:

Редагування (Edit): (рис 3.18) Перехід на сторінку редагування товару, де можна змінити його деталі, включаючи назву, опис, ціну, зображення та інші атрибути.

Видалення (Delete): (рис 3.19) Можливість видалення товару з бази даних. Перед видаленням з'являється підтвердження, щоб запобігти випадковому видаленню.

Створення (Create new): (рис 3.17) Кнопка для додавання нового товару. Відкриває форму, де можна ввести деталі товару, включаючи назву, опис, ціну, зображення та інші атрибути.

Create

ProductViewModel

Name

Description

About

CategoryId

Acoustic drums

RetailPrice

0.00

BrandId

ESP

VideoLink

Create

Choose Files

No file chosen

Upload

N

Рисунок 3.17.Сторінка створення нового товару.

Edit

ProductViewModel

Id

905d17d1-1454-4a2b-8ba4-08dc88993559

Name

FREEDMAN NOHO24 (Charcoal Metallic) #0620-1750

Description

Тип конструкції Грейф на боці
Корпус Вільям (Alder) з кленовим топом (Maple)

About

Freedman Noho 24 - це ідеальна еволюція деяких з найкращих в світі електронних класу custom shop. Від того, як ваша рука буде тиснути плати по 24-матриці накладці грифа з композитним (змішаним) радіусом, до вишуканої чарівності його кленового топа, очевидно, що Noho 24 створений для музикантів, яким потрібно, щоб їх інструменти були настільки ж елегантними, як і надзвичайними.

CategoryId

Electric guitar

RetailPrice

4000.00

BrandId

Yamaha

VideoLink

https://www.youtube.com/embed/gzeOYV8tqo?si=YB5uic3UKdAMNgH



Рисунок 3.18.Сторінка змінення товару.

Delete

Are you sure you want to delete this?

Id

74676bc9-38d7-4029-4d6e-08dc831b1f55

Name

Name

Description

Description

CategoryName

Classical guitar

RetailPrice

500.00

WholesalePrice

0.00



Submit

Back to List

Рисунок 3.19.Сторінка підтвердження видалення товару.

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

Адміністратор також має функцію створювати змінювати та видаляти категорії та бренди.

Рисунок 3.20.Сторінка підтвердження видалення товару.

Також наявна панель управління обліковими записами користувачів.

Рисунок 3.21.Сторінка керування користувачами.

Адміністратор може змінювати ролі користувачам.

Рисунок 3.22.Зміна ролі користувачам.

Ну і наостанок, адміністратор може видаляти відгуки які залишили інші користувачі. (рис 3.23 – 3.24).

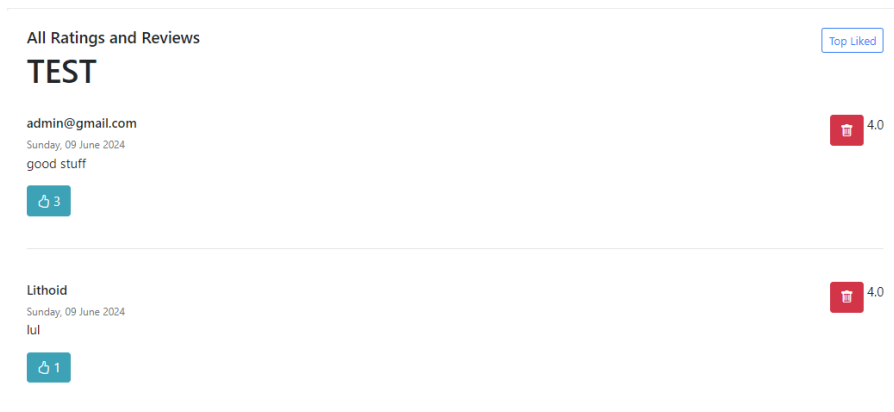


Рисунок 3.23.До видалення відгуку.

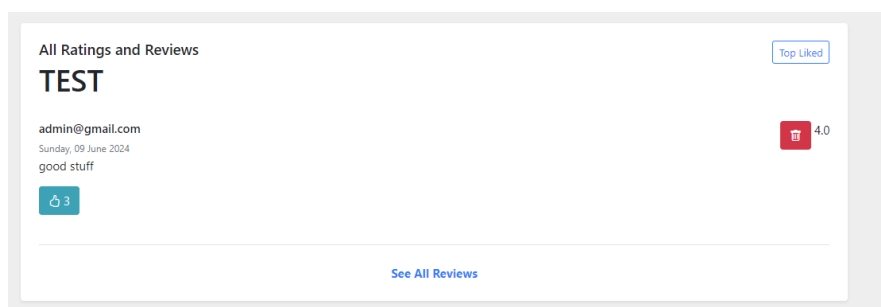


Рисунок 3.24.Після видалення відгуку.

3.4. Тестування роботи інтернет магазину музичних інструментів.

Тестування роботи інтернет-магазину музичних інструментів є критично важливим етапом для забезпечення його правильної функціональності та надійності. Цей процес може включати кілька ключових аспектів, щоб переконатися, що магазин працює оптимально в усіх умовах.

Функціональність веб-сайту та перевірка роботи основних функцій, таких як додавання товарів до кошика, обробка замовлень та оплата мають проходити без збоїв та однаково для всіх користувачів.

Ці тести допомагають забезпечити, що інтернет-магазин працює належним чином і надає задоволення користувачам, забезпечуючи позитивний досвід від покупок онлайн.

Для безпосереднього тестування було обране мануальне тестування.

Його основні переваги:

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

Гнучкість і творчість:

Людина може виявити нюанси та проблеми, які автоматизовані тести можуть пропустити. Вручному тестуванні притаманний більший ступінь творчості та можливість уваги до дрібниць.

Тестування в реальному середовищі:

Мануальне тестування дозволяє спостерігати, як програмне забезпечення веде себе в реальному середовищі з різними обставинами, що є важливим для проектів які взаємодіють з реальними користувачами.

Відсутність необхідності в автоматизації:

Для програм з невеликим обсягом та швидкою зміною вимог ручне тестування може бути ефективнішим та економічнішим, оскільки не потрібно витрачати час на розробку та підтримку автоматизованих сценаріїв тестування.

Перед початком тестування було складено план його проведення який включає в себе перевірку основного та допоміжного функціоналу.

Метою тестування є перевірка наступних функцій:

- Додавання товару в кошик
- Оформлення замовлення
- Оплата замовлення

Таблиця 3.1

Тест «Додавання товару в кошик»	
Опис тесту	Перевірка чи правильно буде додано товар до кошику
Кроки тесту	1. Натискання на кнопку «Додати до кошика» 2. Перехід до кошика
Очікуваний результат	Товар з'явився в одиничному екземплярі у кошику з вірно зазначеними даними
Фактичний результат	Дорівнює очікуваному

Таблиця 3.2

Тест «Оформлення замовлення»	
Опис тесту	Перевірка чи правильно буде оформлене замовлення, та чи правильно введені всі дані
Кроки тесту	1. Перехід від корзини до оформлення замовлення 2. Введення даних користувача 3. Завершення оформлення
Очікуваний результат	Замовлення з'явилося в особистому кабінеті користувача зі статусом «Очікує»
Фактичний результат	Дорівнює очікуваному

Таблиця 3.3

Тест «Оплата замовлення»	
Опис тесту	Перевірка чи правильно буде оплачений товар та чи буде підтверджене замовлення
Кроки тесту	1. Перехід з особистого кабінету до оплати замовлення 2. Введення платіжних даних 3. Отримання підтвердження від сервісу оплати 4. Підтвердження в системі магазину
Очікуваний результат	Позитивний відгук від платіжної системи та статус замовлення «Підтверджено» в системі магазину
Фактичний результат	Дорівнює очікуваному

Ці тести допомагають забезпечити впевненість, що інтернет-магазин працює без помилок та надає якісний функціонал користувачам. Такий підхід допомагає уникнути можливих проблем та помилок, забезпечуючи стабільну та надійну роботу інтернет-магазину музичних інструментів.

Висновки до третього розділу

У третьому розділі було детально розглянуто процес налаштування та розгортання проекту на сервіс Microsoft Azure. Був проведений огляд використаного функціоналу та описані всі кроки необхідні для розгортання. Мало місце і обґрунтування використання обраного рішення, наведені основні переваги при використанні у проекті.

Також був розглянутий інтерфейс та порядок роботи з системою. Наведені приклади інтерфейсу з докладним описом функціоналу. Були розглянуті головна сторінка та процеси які на ній відбуваються, проведені операції сортувань та фільтрацій над товарами. Описані процеси додавання товарів до кошику з подальшим оформленням замовлення. Наведені приклади оплати товару та зовнішній вигляд і функціонал особистого профілю користувача. Окремо були розглянуті аспекти пов'язані з авторизацією та автентифікацією, були наведені відповідні форми та описи процесів які відбуваються. Інтерфейс показав свою зручність та працездатність в задоволенні потреб користувачів.

Були описані принципи роботи облікового запису адміністратора. Наведений основний функціонал та рівень доступу ролі. Показані приклади створення, змін та видалення товарів, показаний доступ адміністратора до категорій та брендів. Описана можливість керування користувачами, зміна їх рівня доступу, та можливість видалити обліковий запис користувача з системи. Наданий функціонал забезпечує повний контроль адміністратора над системою з можливістю впливу на всі моменти системи

Для забезпечення безперебійної роботи системи були проведені тести основної функціоналу проекту. Мануальне тестування показало свої переваги та недоліки. Тестові випадки були описані та перевірені. Результат показав повну працездатність та резистентність системи до несправностей.

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході написання кваліфікаційної роботи бакалавра був повністю розглянутий процес написання інтернет магазину музичних інструментів. Робота охопила всі аспекти та ключові моменти написання системи. Робота була розділена на три частини, завдяки чому зуміла досягти максимальної інформативності в плані описання ходу розробки системи.

Перший розділ приділив максимальну увагу процесу постановки та визначенню поставленої задачі. Була повністю розібрана тема та проведений збір вимог. Дані коригувалися або підтверджувалися під час огляду аналогічних рішень існуючих на ринку. Був проведений максимально детальний аналіз, встановленні переваги та недоліки розглянутих систем та як вони впливали на досвід користувача. Після визначення цілей та мети роботи був підібраний найбільш підходящий інструментарій для розробки. Викристання кожного інструменту є обгрунтованим та підтвердженим проведенням анілзом. Згідно з цими вимогами та обраними інструментами розробки була виділенна найбільш ефективна архітектура додатку, яка забезпечала простоту розробки та надійність кінцевого проекту.

Другий розділ був присвячений визначенню основних варіантів використання системи. Завдяки результатам аналізу в першому розділі, вдалося встановити найбільш корисний та підходящий функціонал проекту, який забезпечить користувча хорошим досвідом використання. Були сформовані основні сутності та встановлена об'єкто-орієнтована модель системи. На основі створених сутностей була спроектована та розроблена база даних проекту. Таблиці та структура організації даних отримали обгрунтування . Були розроблені алгоритми використання системи. Це дозволило виявити найбільш оптимальні шляхи реалізації функціоналу, для забезпечення позитивного користувацьго досвіду та ефективного використання ресурсів системою. На основі отриманих даних був розроблений та описаний основний функціонал додатку. Представлені основні методи для взаємодії компонентів системи між собою.

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

Третій розділ розглянув аспект розгортання та налаштування системи на ресурсі Microsoft Azure. Був обґрунтований вибір саме цього інструменту. Наведений та описаний основний функціонал сервісу. Вдало розгортання істворило можливість переглядати сторінку з будь-якого пристрою. Був розглянутий інтерфейс системи. Наведені та описані приклади використання системи. Завдяки проведеному тестуванню підтвердилась працездатність та безвідбійність усього проекту.

Таким чином дана робота продемонструвала та підтвердила навички в процесі розробки інтернет магазину музичних інструментів.

Завдяки використаному підходу, ефективному виборі інструментів, детальному аналізу рішень вдалось створити перспективний проект з великим потенціалом до розширення та розвитку.

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
		Вакалюк Т.А.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Microsoft .NET[Електронний ресурс] – Режим доступу до ресурсу:
<https://dotnet.microsoft.com/>
2. ASP.NET Web API[Електронний ресурс] – Режим доступу до ресурсу:
<https://learn.microsoft.com/en-us/aspnet/web-api/>
3. ASP.NET Core[Електронний ресурс] – Режим доступу до ресурсу:
<https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-8.0>
4. EntityFramework Core[Електронний ресурс] – Режим доступу до ресурсу:
<https://learn.microsoft.com/en-us/ef/core/>
5. Microsoft SQL Server [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.microsoft.com/en-us/sql-server>
6. Azure Portal[Електронний ресурс] – Режим доступу до ресурсу:
<https://portal.azure.com/#home>
7. DOU спільнота програмістів[Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/>
8. Udemy[Електронний ресурс] – Режим доступу до ресурсу:
<https://www.udemy.com/>
9. W3 schools [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.w3schools.com/>
10. SQL injectons[Електронний ресурс] – Режим доступу до ресурсу:
<https://portswigger.net/web-security/sql-injection>
11. Cross site scripting [Електронний ресурс] – Режим доступу до ресурсу:
<https://owasp.org/www-community/attacks/xss/>
12. OpenID and Oauth 2.0[Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.okta.com/docs/concepts/oauth-openid/>
13. SignalR[Електронний ресурс] – Режим доступу до ресурсу:
<https://learn.microsoft.com/aspnet/signalr/>
14. Microsoft IIS server [Електронний ресурс] – Режим доступу до ресурсу:
<https://learn.microsoft.com/iis>
15. Jam.ua [Електронний ресурс] – Режим доступу до ресурсу: <https://jam.ua/ua/>

		Головецький Є.В.			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

- 16.Muztorg.ua [Електронний ресурс] – Режим доступу до ресурсу:
<https://muztorg.ua/uk/>
- 17.Luxpro.ua [Електронний ресурс] – Режим доступу до ресурсу:
<https://luxpro.ua/>
- 18.Unit of work and repository pattern [Електронний ресурс] – Режим доступу до ресурсу: <https://learn.microsoft.com/ru-ru/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application>
- 19.ORM system [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/>

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

ДОДАТКИ

		Головецький Є.В			ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

Shop.StoreWebApi

```
namespace StoreWebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AssetApiController : ControllerBase
    {
        protected APIResponse _response;
        private IAssetRepository _assetRepository;

        public AssetApiController(IAssetRepository assetRepository)
        {
            _assetRepository = assetRepository;
            _response = new();
        }
        [HttpGet]
        public async Task<ActionResult<APIResponse>> GetAssets()
        {
            var assets = await _assetRepository.GetAllItemsAsync(includeProps: "Products");
            _response.Result = assets.Select(p => new AssetViewModel(p));
            _response.StatusCode = System.Net.HttpStatusCode.OK;

            return Ok(_response);
        }
        [HttpGet("{id:Guid}", Name = "GetAsset")]
        [ProducesResponseType(200)]
        [ProducesResponseType(400)]
        [ProducesResponseType(404)]
        public async Task<ActionResult<APIResponse>> GetAsset(Guid id)
        {
            if (id == null)
            {
                return BadRequest();
            }
            var asset = await _assetRepository.GetItemAsync(p => p.Id ==
id,includeProps:"Products");
            if (asset == null)
            {
                return NotFound();
            }
            _response.Result = new AssetViewModel(asset);
            _response.StatusCode = System.Net.HttpStatusCode.OK;

            return Ok(_response);
        }

        [HttpPost(Name = "CreateAsset")]
        public async Task<ActionResult<APIResponse>> CreateAsset([FromBody] AssetViewModel
model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            if (model == null)
            {
                return BadRequest(model);
            }
            await _assetRepository.AddItemAsync(model);
        }
    }
}
```

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

```

        _response.Result = model;
        _response.StatusCode = System.Net.HttpStatusCode.Created;

        return CreatedAtRoute("GetAsset", new { id = model.Id }, _response);
    }
    [HttpDelete("{id:Guid}", Name = "DeleteAsset")]
    public async Task<ActionResult<APIResponse>> DeleteAsset(Guid id)
    {
        if (id == null)
        {
            return BadRequest();
        }
        var item = await _assetRepository.GetItemAsync(p => p.Id == id);
        if (item == null)
        {
            return NotFound();
        }
        await _assetRepository.DeleteItemAsync(id);
        _response.StatusCode = HttpStatusCode.NoContent;
        _response.IsSuccess = true;
        return Ok(_response);
    }
    [HttpPut("{id:Guid}", Name = "UpdateAsset")]
    public IActionResult UpdateAsset(Guid id, [FromBody] AssetViewModel model)
    {
        if (model == null || id != model.Id)
        {
            return BadRequest();
        }
        _assetRepository.ChangeItemAsync(model);
        _response.StatusCode = HttpStatusCode.NoContent;
        _response.IsSuccess = true;
        return Ok(_response);
    }
}
}
}

```

```

[Route("api/[controller]")]
[ApiController]
public class ProductApiController : ControllerBase
{
    protected APIResponse _response;
    private IProductRepository _productRepository;
    private ICategoryRepository _categoryRepository;

    public ProductApiController(IProductRepository productRepository,
    ICategoryRepository categoryRepository)
    {
        _productRepository = productRepository;
        _categoryRepository = categoryRepository;
        _response = new();
    }
    [HttpGet]
    [ProducesResponseType(200)]
    public async Task<ActionResult<APIResponse>> GetProducts()
    {
        var products = await _productRepository.GetAllItemsAsync(includeProps:
"Brand,Category,Assets,Favourites,Reviews");
        _response.Result = products.Select(p=>new ProductViewModel(p));
        _response.StatusCode = System.Net.HttpStatusCode.OK;

        return Ok(_response);
    }
}

```

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

[HttpGet("GetProductsByCategory/{categoryId:Guid}", Name =
"GetProductsByCategory")]
public async Task<ActionResult<APIResponse>> GetProductsByCategory(Guid categoryId)
{
    var products = await _productRepository.GetAllItemsAsync(p=>p.CategoryId==
categoryId, includeProps: "Brand,Category,Assets,Favourites,Reviews");
    _response.Result = products.Select(p => new ProductViewModel(p));
    _response.StatusCode = System.Net.HttpStatusCode.OK;

    return Ok(_response);
}

[HttpGet("{id:Guid}", Name = "GetProduct")]
[ProducesResponseType(200)]
[ProducesResponseType(400)]
[ProducesResponseType(404)]
public async Task<ActionResult<APIResponse>> GetProduct(Guid id)
{
    if (id==null)
    {
        return BadRequest();
    }
    var product = await _productRepository.GetItemAsync(p=>p.Id==id,includeProps:
"Brand,Category,Assets,Favourites,Reviews");
    if (product == null)
    {
        return NotFound();
    }
    _response.Result = new ProductViewModel(product);
    _response.StatusCode = System.Net.HttpStatusCode.OK;

    return Ok(_response);
}

[HttpPost( Name = "CreateProduct")]
public async Task<ActionResult<APIResponse>> CreateProduct([FromBody]
ProductViewModel model)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }

    if (model == null)
    {
        return BadRequest(model);
    }

    await _productRepository.AddItemAsync(model);

    _response.Result = model;
    _response.StatusCode = System.Net.HttpStatusCode.Created;

    return CreatedAtRoute("GetProduct", new { id = model.Id }, _response);
}

[HttpDelete("{id:Guid}", Name = "DeleteProduct")]
public async Task<ActionResult<APIResponse>> DeleteProduct(Guid id)
{
    if (id==null)
    {
        return BadRequest();
    }

    var product = await _productRepository.GetItemAsync(p => p.Id == id);
    if (product == null)
    {
        return NotFound();
    }
}

```

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        await _productRepository.DeleteItemAsync(id);
        _response.StatusCode = HttpStatusCode.NoContent;
        _response.IsSuccess = true;
        return Ok(_response);
    }
    [HttpPut("{id:Guid}", Name = "UpdateProduct")]
    public async Task<ActionResult<APIResponse>> UpdateProduct(Guid id, [FromBody]
ProductViewModel model)
    {
        if (model == null || id != model.Id)
        {
            return BadRequest();
        }
        await _productRepository.ChangeItemAsync(model);
        _response.StatusCode = HttpStatusCode.NoContent;
        _response.IsSuccess = true;
        return Ok(_response);
    }

    [HttpGet("GetFavouriteForUser/{userId:Guid}", Name = "GetFavouriteForUser")]
    public async Task<ActionResult<APIResponse>> GetFavouriteForUser(Guid userId)
    {
        var products = await _productRepository.GetAllItemsAsync(p =>
p.Favourites.Any(f=>f.UserId== userId), includeProps:
"Brand,Category,Assets,Favourites,Reviews");
        _response.Result = products.Select(p => new ProductViewModel(p));
        _response.StatusCode = System.Net.HttpStatusCode.OK;

        return Ok(_response);
    }

    [HttpGet("GetRelatedProducts/{productId:Guid}", Name = "GetRelatedProducts")]
    public async Task<ActionResult<APIResponse>> GetRelatedProducts(Guid productId)
    {
        var product = await _productRepository.GetItemAsync(p => p.Id == productId,
includeProps: "Brand,Category,Assets,Favourites,Reviews");
        if (product == null)
        {
            return NotFound();
        }

        var parentCategories = await _categoryRepository.GetAllItemsAsync(c =>
c.ParentCategoryId==null, includeProps: "Category");

        var products = await _productRepository.GetAllItemsAsync(includeProps:
"Brand,Category,Assets,Favourites,Reviews");
        _response.Result = products.Select(p => new ProductViewModel(p));
        _response.StatusCode = System.Net.HttpStatusCode.OK;

        return Ok(_response);
    }
}
}

```

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

namespace StoreWebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class OrderApiController : ControllerBase
    {

        protected APIResponse _response;
        private IOrderRepository _orderRepository;

        public OrderApiController(IOrderRepository orderRepository)
        {
            _orderRepository = orderRepository;
            _response = new();
        }

        [HttpGet]
        public async Task<ActionResult<APIResponse>> GetOrders()
        {
            var orders = await _orderRepository.GetAllItemsAsync(includeProps:
"CartItems,Status");
            _response.Result = orders.Select(p => new OrderViewModel(p));
            _response.StatusCode = System.Net.HttpStatusCode.OK;

            return Ok(_response);
        }

        [HttpGet("GetOrdersForUser/{userId:Guid}", Name = "GetOrdersForUser")]
        public async Task<ActionResult<APIResponse>> GetOrdersForUser(Guid userId)
        {
            var orders = await _orderRepository.GetAllItemsAsync(p => p.UserId == userId,
includeProps: "CartItems,Status");
            _response.Result = orders.Select(p => new OrderViewModel(p));
            _response.StatusCode = System.Net.HttpStatusCode.OK;

            return Ok(_response);
        }

        [HttpGet("{id:Guid}", Name = "GetOrder")]
        [ProducesResponseType(200)]
        [ProducesResponseType(400)]
        [ProducesResponseType(404)]
        public async Task<ActionResult<APIResponse>> GetOrder(Guid id)
        {
            if (id == null)
            {
                return BadRequest();
            }
            var order = await _orderRepository.GetItemAsync(p => p.Id == id, includeProps:
"CartItems,Status");
            if (order == null)
            {
                return NotFound();
            }
            _response.Result = new OrderViewModel(order);
            _response.StatusCode = System.Net.HttpStatusCode.OK;

            return Ok(_response);
        }

        [HttpPost(Name = "CreateOrder")]
        public async Task<ActionResult<APIResponse>> CreateOrder([FromBody] OrderViewModel
model)
        {

```

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (!ModelState.IsValid)
        {
            return BadRequest(ModelState);
        }

        if (model == null)
        {
            return BadRequest(model);
        }
        await _orderRepository.AddItemAsync(model);

        _response.Result = model;
        _response.StatusCode = System.Net.HttpStatusCode.Created;

        return CreatedAtRoute("GetOrder", new { id = model.Id }, _response);
    }

    [HttpDelete("{id:Guid}", Name = "DeleteOrder")]
    public async Task<ActionResult<APIResponse>> DeleteOrder(Guid id)
    {
        if (id == null)
        {
            return BadRequest();
        }
        var item = await _orderRepository.GetItemAsync(p => p.Id == id);
        if (item == null)
        {
            return NotFound();
        }
        await _orderRepository.DeleteItemAsync(id);
        _response.StatusCode = HttpStatusCode.NoContent;
        _response.IsSuccess = true;
        return Ok(_response);
    }

    [HttpPut("{id:Guid}")]
    public async Task<ActionResult> Update(Guid id, [FromBody] OrderViewModel model)
    {
        if (model == null || id != model.Id)
        {
            return BadRequest();
        }
        await _orderRepository.ChangeItemAsync(model);
        _response.StatusCode = HttpStatusCode.NoContent;
        _response.IsSuccess = true;
        return Ok(_response);
    }
}

namespace StoreWebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class FavouriteApiController : Controller
    {
        protected APIResponse _response;
        private IFavouriteRepository _favouriteRepository;
    }
}

```

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public FavouriteApiController(IFavouriteRepository favouriteRepository)
{
    _favouriteRepository = favouriteRepository;
    _response = new();
}
[HttpGet]
public async Task<ActionResult<APIResponse>> GetFavourites()
{

    var products = await _favouriteRepository.GetAllItemsAsync();
    _response.Result = products.Select(p => new FavouriteViewModel(p));
    _response.StatusCode = System.Net.HttpStatusCode.OK;

    return Ok(_response);
}
[HttpPost]
public async Task<ActionResult<APIResponse>> CreateFavourite([FromBody]
FavouriteViewModel model)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }

    if (model == null)
    {
        return BadRequest(model);
    }
    await _favouriteRepository.AddItemAsync(model);

    _response.Result = model;
    _response.StatusCode = System.Net.HttpStatusCode.Created;

    return CreatedAtRoute("GetFavourite", new { id = model.Id }, _response);
}

[HttpDelete("{prodId:Guid}/{userId:Guid}", Name = "DeleteFavourite")]
public async Task<ActionResult<APIResponse>> DeleteFavourite(Guid prodId,Guid
userId)
{
    if (prodId == Guid.Empty || userId == Guid.Empty)
    {
        return BadRequest();
    }
    var item = await _favouriteRepository.GetItemAsync(f
=>f.UserId==userId&&f.ProductId==prodId );
    if (item == null)
    {
        return NotFound();
    }
    await _favouriteRepository.DeleteItemAsync(item.Id);
    _response.StatusCode = HttpStatusCode.NoContent;
    _response.IsSuccess = true;
    return Ok(_response);
}

}
}

```

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

namespace StoreWebApi.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ReviewApiController : ControllerBase
    {
        protected APIResponse _response;
        private IReviewRepository _reviewRepository;

        public ReviewApiController(IReviewRepository reviewRepository)
        {
            _reviewRepository = reviewRepository;
            _response = new();
        }

        [HttpGet]
        public async Task<ActionResult<APIResponse>> GetReviews()
        {
            var reviews = await _reviewRepository.GetAllItemsAsync();
            _response.Result = reviews.Select(p => new ReviewViewModel(p));
            _response.StatusCode = System.Net.HttpStatusCode.OK;

            return Ok(_response);
        }

        [HttpGet("{id:Guid}", Name = "GetReview")]
        [ProducesResponseType(200)]
        [ProducesResponseType(400)]
        [ProducesResponseType(404)]
        public async Task<ActionResult<APIResponse>> GetReview(Guid id)
        {
            if (id == null)
            {
                return BadRequest();
            }
            var review = await _reviewRepository.GetItemAsync(p => p.Id == id);
            if (review == null)
            {
                return NotFound();
            }
            _response.Result = new ReviewViewModel(review);
            _response.StatusCode = System.Net.HttpStatusCode.OK;

            return Ok(_response);
        }

        [HttpGet("GetAllForProduct/{productId:Guid}", Name = "GetAllForProduct")]
        public async Task<ActionResult<APIResponse>> GetAllForProduct(Guid productId)
        {
            var reviews = await _reviewRepository.GetAllItemsAsync(p => p.ProductId ==
productId, includeProps: "Product");
            _response.Result = reviews.Select(p => new ReviewViewModel(p));
            _response.StatusCode = System.Net.HttpStatusCode.OK;

            return Ok(_response);
        }

        [HttpGet("GetReviewsForUser/{userId:Guid}", Name = "GetReviewsForUser")]
        public async Task<ActionResult<APIResponse>> GetReviewsForUser(Guid userId)
        {
            var reviews = await _reviewRepository.GetAllItemsAsync(p => p.UserId == userId,
includeProps: "Product");
            _response.Result = reviews.Select(p => new ReviewViewModel(p));
            _response.StatusCode = System.Net.HttpStatusCode.OK;

            return Ok(_response);
        }
    }
}

```

		Головецький Є.В.			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

[HttpPost(Name = "CreateReview")]
public async Task<ActionResult<APIResponse>> CreateReview([FromBody]
ReviewViewModel model)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }

    if (model == null)
    {
        return BadRequest(model);
    }
    await _reviewRepository.AddItemAsync(model);

    _response.Result = model;
    _response.StatusCode = System.Net.HttpStatusCode.Created;

    return CreatedAtRoute("GetCategory", new { id = model.Id }, _response);
}

[HttpDelete("{id:Guid}", Name = "DeleteReview")]
public async Task<ActionResult<APIResponse>> DeleteReview(Guid id)
{
    if (id == null)
    {
        return BadRequest();
    }
    var item = await _reviewRepository.GetItemAsync(p => p.Id == id);
    if (item == null)
    {
        return NotFound();
    }
    await _reviewRepository.DeleteItemAsync(id);
    _response.StatusCode = HttpStatusCode.NoContent;
    _response.IsSuccess = true;
    return Ok(_response);
}

[HttpPut("{id:Guid}", Name = "UpdateReview")]
public IActionResult UpdateReview(Guid id, [FromBody] ReviewViewModel model)
{
    if (model == null || id != model.Id)
    {
        return BadRequest();
    }
    _reviewRepository.ChangeItemAsync(model);
    _response.StatusCode = HttpStatusCode.NoContent;
    _response.IsSuccess = true;
    return Ok(_response);
}
}
}

```

Store.WebApp

```

public class ProductController : Controller
{
    private IProductService _productService;
    private ICategoryService _categoryService;
    private IReviewService _reviewService;
    private IBrandService _brandService;

    private readonly UserManager<AppUser> userManager;

```

		Головецький Є.В			ПІЗ.КР.Б – 121 – 24 – ПІЗ	Арк.
		Вакалюк Т.А.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    public ProductController(IProductService productService, ICategoryService
categoryService, IBrandService brandService, IReviewService reviewService,
userManager<AppUser> userManager)
    {
        _productService = productService;
        _categoryService = categoryService;
        _brandService = brandService;
        _reviewService = reviewService;
        _userManager = userManager;
    }

    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> List(string name,string sortOrder)
    {
        ViewBag.NameSortParm = sortOrder== "Name" ? "name_desc" : "Name";
        ViewBag.PriceSortParm = sortOrder == "Price" ? "price_desc" : "Price";
        ViewBag.CategorySortParm = sortOrder == "Category" ? "category_desc" : "Category";

        List<ProductViewModel> products = new();
        var response = await
_productService.GetAllAsync<APIResponse>(HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {
            products =
JsonConvert.DeserializeObject<List<ProductViewModel>>(Convert.ToString(response.Result));
        }
        switch (sortOrder)
        {
            case "Name":
                products = products.OrderBy(s => s.Name).ToList();
                break;
            case "name_desc":
                products = products.OrderByDescending(s => s.Name).ToList();
                break;
            case "Price":
                products = products.OrderBy(s => s.RetailPrice).ToList();
                break;
            case "price_desc":
                products = products.OrderByDescending(s => s.RetailPrice).ToList();
                break;
            case "Category":
                products = products.OrderBy(s => s.CategoryName).ToList();
                break;
            case "category_desc":
                products = products.OrderByDescending(s => s.CategoryName).ToList();
                break;
        }

        if (!string.IsNullOrEmpty(name))
        {
            products = products.Where(p => p.Name.Contains(name)).ToList();
        }

        return View(products);
    }
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> Create(ProductViewModel product)
    {
        if (product.IsEmpty)
        {
            List<CategoryViewModel> categories = new();
            var response = await
_categoryService.GetAllAsync<APIResponse>(HttpContext.Session.GetString(SD.SessionToken));
            if (response != null && response.IsSuccess)
            {
                categories =
JsonConvert.DeserializeObject<List<CategoryViewModel>>(Convert.ToString(response.Result));
            }
        }
    }

```

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        List<BrandViewModel> brands = new();
        response = await
_brandService.GetAllAsync<APIResponse>(HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {
            brands =
JsonConvert.DeserializeObject<List<BrandViewModel>>(Convert.ToString(response.Result));
        }

        CreateEditProductViewModel model = new CreateEditProductViewModel();
        model.Product = product;
        model.Brands = brands.OrderBy(b => b.Name); ;
        model.Categories = categories.OrderBy(c => c.Name);

        return View(model);
    }

    if (ModelState.IsValid)
    {
        var response = await _productService.CreateAsync<APIResponse>(product,
HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {
            TempData["success"] = "Product created successfully";
            return RedirectToAction(nameof(List));
        }
    }

    return Redirect("Error");
}
[Authorize(Roles = "Admin")]
public async Task<IActionResult> Edit(Guid id)
{
    ProductViewModel product = new();
    var response = await
_productService.GetAsync<APIResponse>(id,HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
    {
        product =
JsonConvert.DeserializeObject<ProductViewModel>(Convert.ToString(response.Result));
    }
    List<CategoryViewModel> categories = new();
    response = await
_categoryService.GetAllAsync<APIResponse>(HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
    {
        categories =
JsonConvert.DeserializeObject<List<CategoryViewModel>>(Convert.ToString(response.Result));
    }

    List<BrandViewModel> brands = new();
    response = await
_brandService.GetAllAsync<APIResponse>(HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
    {
        brands =
JsonConvert.DeserializeObject<List<BrandViewModel>>(Convert.ToString(response.Result));
    }

    CreateEditProductViewModel model = new CreateEditProductViewModel();
    model.Product = product;
    model.Brands = brands.OrderBy(b => b.Name); ;
    model.Categories = categories.OrderBy(c=>c.Name);

    return View(model);
}

```

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> AcceptEdit(CreateEditProductViewModel model)
    {
        if (model!=null)
        {
            var response = await _productService.UpdateAsync<APIResponse>(model.Product,
HttpContext.Session.GetString(SD.SessionToken));
            if (response != null && response.IsSuccess)
            {

return Redirect("~/Product/List");
            }
            return Redirect("~/Product/List");
        }
        [Authorize(Roles = "Admin")]
        public async Task<IActionResult> Delete(Guid id)
        {
            ProductViewModel product = new();
            var response = await _productService.GetAsync<APIResponse>(id,
HttpContext.Session.GetString(SD.SessionToken));
            if (response != null && response.IsSuccess)
            {
                product =
JsonConvert.DeserializeObject<ProductViewModel>(Convert.ToString(response.Result));
            }

            return View(product);
        }
        [Authorize(Roles = "Admin")]
        public async Task<IActionResult> AcceptDelete(Guid? id)
        {
            var response = await _productService.DeleteAsync<APIResponse>(id.Value,
HttpContext.Session.GetString(SD.SessionToken));
            if (response != null && response.IsSuccess)
            {
                TempData["success"] = "Product deleted successfully";
                return Redirect("~/Product/List");
            }
            return Redirect("~/Product/List");
        }

        public async Task<IActionResult> Details(Guid id)
        {
            DetailsViewModel model = new DetailsViewModel();

            ProductViewModel product = new();
            var response = await _productService.GetAsync<APIResponse>(id,
HttpContext.Session.GetString(SD.SessionToken));
            if (response != null && response.IsSuccess)
                product =
JsonConvert.DeserializeObject<ProductViewModel>(Convert.ToString(response.Result));

            List<ProductViewModel> products = new();
            response = await
_productService.GetAllByCategoryAsync<APIResponse>(product.CategoryId,
HttpContext.Session.GetString(SD.SessionToken));
            if (response != null && response.IsSuccess)
                products =
JsonConvert.DeserializeObject<List<ProductViewModel>>(Convert.ToString(response.Result));

            List<ReviewViewModel> reviews = new();
            response = await
_reviewService.GetAllForProductAsync<APIResponse>(product.Id,HttpContext.Session.GetString(
SD.SessionToken));
            if (response != null && response.IsSuccess)

```

		Головецький Є.В			ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
		Вакалюк Т.А.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        reviews =
JsonConvert.DeserializeObject<List<ReviewViewModel>>(Convert.ToString(response.Result));

        model.Product = product;
        model.Products = products;
        model.Reviews = reviews;

        return View(model);
    }
}

public class OrderController : Controller
{

    private ICartItemService _cartItemService;
    private IStatusService _statusService;
    private IOrderService _orderService;
    private ILiqpayService _liqpayService;
    private readonly UserManager<AppUser> _userManager;

    private readonly IHttpClientFactory _httpClientFactory;

    public OrderController(UserManager<AppUser> userManager,
        ICartItemService cartItemService,
        IStatusService statusService,
        IOrderService orderService,
        IHttpClientFactory httpClientFactory, ILiqpayService liqpayService)
    {

        _userManager = userManager;
        _cartItemService = cartItemService;
        _statusService = statusService;
        _orderService = orderService;
        _httpClientFactory = httpClientFactory;
        _liqpayService = liqpayService;
    }
    public IActionResult Index()
    {
        return View();
    }
    public async Task<IActionResult> MakeOrder(Guid? cartId)
    {
        /*if (cartId.HasValue)
        {
            var items = CartViewModel.GetCartItemList(_cartItemRepository, cartId);
            OrderViewModel model = new OrderViewModel()
            {
                Status = "Waiting",
                ClientName = _userManager.GetUserName(User),
                ClientPhone = "123", //_userManager.GetPhoneNumberAsync(user),
                OrderDate = DateTime.Now,
                CartItemIds = items.Select(c => c.Id).ToList()
            };
            if (await _orderRepository.AddItemAsync(model))
            {
                return Redirect("~/Shop/List");
            }
        }
        */
        var user = await _userManager.GetUserAsync(User);

```

		Головецький Є.В			ПІЗ.КР.Б – 121 – 24 – ПІЗ	Арк.
		Вакалюк Т.А.				86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (cartId.HasValue)
        {
            ViewData["cartId"] = cartId;
        }
        var cartItems = new List<CartViewModel>();

        var response = await _cartItemService.GetAllAsync<APIResponse>(cartId.Value,
HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {
            cartItems =
JsonConvert.DeserializeObject<List<CartViewModel>>(Convert.ToString(response.Result)
);

            CartOrderViewModel model = new CartOrderViewModel();
            model.Order = new OrderViewModel();

            model.Order.ClientName = user.FullName;
            model.Order.ClientPhone = user.PhoneNumber;
            model.Order.Email = user.Email;
            model.CartItems = cartItems;

            ViewBag.Subtotal = 0;
            foreach (var item in model.CartItems)
            {
                ViewBag.Subtotal += item.Quantity * item.Price;
            }

            return View("MakeOrder", model);
        }
        public async Task<IActionResult> AcceptOrder(CartOrderViewModel model, Guid?
cartId, string submitButton)
        {
            if (ModelState.IsValid)
            {
                var cartItems = new List<CartViewModel>();

                var response = await
_cartItemService.GetAllAsync<APIResponse>(cartId.Value,
HttpContext.Session.GetString(SD.SessionToken));
                if (response != null && response.IsSuccess)
                {
                    cartItems =
JsonConvert.DeserializeObject<List<CartViewModel>>(Convert.ToString(response.Result)
);

                    StatusViewModel status = new();
                    response = await _statusService.GetAsync<APIResponse>("Pending",
HttpContext.Session.GetString(SD.SessionToken));
                    if (response != null && response.IsSuccess)
                    {
                        status =
JsonConvert.DeserializeObject<StatusViewModel>(Convert.ToString(response.Result));

                        model.Order.Id = Guid.NewGuid();
                        model.Order.OrderDate = DateTime.Now;
                        model.Order.StatusName = status.Name;
                        model.Order.StatusId = status.Id;
                        model.Order.CartItemIds = cartItems.Select(c => c.Id).ToList();
                        model.Order.UserId = Guid.Parse(_userManager.GetUserId(User));

                        foreach (var item in cartItems)
                            model.Order.TotalSumm += item.Price * item.Quantity;
                    }
                }
            }
        }
    }
}

```

		Головецький Є.В			ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
		Вакалюк Т.А.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        response = await _orderService.CreateAsync<APIResponse>(model.Order,
HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
            TempData["success"] = "Order created successfully";

        string jobId = "";
        switch (submitButton)
        {
            case "Go to payment":
                return RedirectToAction("Checkout", new { orderId =
model.Order.Id });
            case "Pay later":
                jobId = BackgroundJob.Schedule(() =>
ChangeStatus(model.Order.Id), TimeSpan.FromSeconds(40));
                return RedirectToAction("List", "Shop");
        }
        return RedirectToAction("List", "Shop");
    }
    else
    {
        var cartItems = new List<CartItemViewModel>();

        var response = await
_cartItemService.GetAllAsync<APIResponse>(cartId.Value,
HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {
            cartItems =
JsonConvert.DeserializeObject<List<CartItemViewModel>>(Convert.ToString(response.Result)
);
            model.CartItems = cartItems;
            ViewBag.Subtotal = 0;
            if (cartId.HasValue)
            {
                ViewData["cartId"] = cartId;
            }
            foreach (var item in model.CartItems)
            {
                ViewBag.Subtotal += item.Quantity * item.Price;
            }
            return View("MakeOrder", model);
        }
    }

    public async Task ChangeStatus(Guid orderId)
    {
        OrderViewModel order = new();
        var response = await _orderService.GetAsync<APIResponse>(orderId, "token must
be here");
        if (response != null && response.IsSuccess)
        {
            order =
JsonConvert.DeserializeObject<OrderViewModel>(Convert.ToString(response.Result));
            StatusViewModel status = new();
            response = await _statusService.GetAsync<APIResponse>("Canceled", "token
must be here");
            if (response != null && response.IsSuccess)
            {

```

		Головецький Є.В.			ПІЗ.КР.Б – 121 – 24 – ПІЗ	Арк.
		Вакалюк Т.А.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        status =
        JsonConvert.DeserializeObject<StatusViewModel>(Convert.ToString(response.Result));
    }
    if (order.StatusName!="Sended")
    {
        order.StatusId = status.Id;
        response = await _orderService.UpdateAsync<APIResponse>(order, "token
must be here");
    }

}

[Authorize(Roles = "Admin")]
public async Task<IActionResult> List()
{

    List<OrderViewModel> orders = new();
    var response = await
_orderService.GetAllAsync<APIResponse>(HttpContext.Session.GetString(SD.SessionToken
));
    if (response != null && response.IsSuccess)
    {
        orders =
        JsonConvert.DeserializeObject<List<OrderViewModel>>(Convert.ToString(response.Result
));
    }
    orders = orders.OrderBy(o => o.OrderDate).ToList();
    return View(orders);
}

[Authorize(Roles = "Admin")]
public async Task<IActionResult> Edit(Guid? id)
{
    OrderViewModel order = new();
    var response = await _orderService.GetAsync<APIResponse>(id.Value,
HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
    {
        order =
        JsonConvert.DeserializeObject<OrderViewModel>(Convert.ToString(response.Result));
    }

    var statuses = new List<StatusViewModel>();
    response = await
_statusService.GetAllAsync<APIResponse>(HttpContext.Session.GetString(SD.SessionToke
n));
    if (response != null && response.IsSuccess)
    {
        statuses =
        JsonConvert.DeserializeObject<List<StatusViewModel>>(Convert.ToString(response.Resul
t));
    }

    EditOrderViewModel model = new EditOrderViewModel();
    model.Order = order;
    model.Statuses = statuses;
}

```

```

        return View(model);
    }
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> AcceptEdit(EditOrderViewModel model)
    {
        if (model.Order != null)
        {
            var response = await _orderService.UpdateAsync<APIResponse>(model.Order,
HttpContext.Session.GetString(SD.SessionToken));
            if (response != null && response.IsSuccess)
            {
                return Redirect("~/Order/List");
            }
        }
        return Redirect("~/Order/List");
    }

    public async Task<IActionResult> Delete(Guid? id)
    {
        var response = await _orderService.DeleteAsync<APIResponse>(id.Value,
HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {

            TempData["success"] = "Order deleted successfully";
            return Redirect("~/Order/List");
        }
        return Redirect("~/Order/List");
    }

    [HttpPost]
    public async Task<IActionResult> Checkout(CheckoutViewModel model, string
jobId="")
    {
        var response = JsonConvert.DeserializeObject<Dictionary<string,
string>>(await _liqpayService.SendPayment(model.CardNumber));

        if (response["result"] == "error")
        {
            ViewBag.ErrorCode = response["err_code"];
            ViewBag.ErrorDesc = response["err_description"];
        }
        if (response["result"] == "ok")
        {
            OrderViewModel order = new();
            var response = await
_orderService.GetAsync<APIResponse>(model.RelatedOrderId,
HttpContext.Session.GetString(SD.SessionToken));
            if (response != null && response.IsSuccess)
            {
                order =
JsonConvert.DeserializeObject<OrderViewModel>(Convert.ToString(response.Result));
            }
            StatusViewModel status = new();
            response = await _statusService.GetAsync<APIResponse>("Sended",
HttpContext.Session.GetString(SD.SessionToken));
            if (response != null && response.IsSuccess)
            {
                status =
JsonConvert.DeserializeObject<StatusViewModel>(Convert.ToString(response.Result));
            }
        }
    }

```

```

        order.StatusId = status.Id;
        response = await _orderService.UpdateAsync<APIResponse>(order,
HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {

        }

        ViewBag.ErrorCode = "Payment successful";
        ViewBag.ErrorDesc = "Traction has reached its destination";
    }

    return View("Result");
}

[HttpGet]
public async Task<IActionResult> Checkout(Guid orderId)
{

    ViewBag.OrderId = orderId.ToString().Substring(32,4);
    return View();
}

public async Task<IActionResult> Details(Guid orderId)
{
    OrderViewModel order = new();
    var response = await _orderService.GetAsync<APIResponse>(orderId,
HttpContext.Session.GetString(SD.SessionToken));
    if (response != null && response.IsSuccess)
    {
        order =
JsonConvert.DeserializeObject<OrderViewModel>(Convert.ToString(response.Result));
    }

    var cartItems = new List<CartItemViewModel>();
    var user = await _userManager.GetUserAsync(User);

    foreach (var itemId in order.CartItemIds)
    {
        response = await _cartItemService.GetAsync<APIResponse>(itemId,
HttpContext.Session.GetString(SD.SessionToken));
        if (response != null && response.IsSuccess)
        {
            CartItemViewModel cartItem =
JsonConvert.DeserializeObject<CartItemViewModel>(Convert.ToString(response.Result));
            cartItems.Add(cartItem);
        }
    }

    CartOrderViewModel model = new CartOrderViewModel();
    model.Order = order;
    model.CartItems = cartItems;

    return View(model);
}
}

```