**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Сумський державний університет**
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)

червня 2024 р.

# КВАЛІФІКАЦІЙНА РОБОТА

**на здобуття освітнього ступеня бакалавр**

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна система управління контентом та замовленнями для роздрібної торгівлі»

здобувача групи ІН-06-2 Бараболікової Анни Віталіївни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Анна БАРАБОЛІКОВА
(підпис)

Старша викладачка кафедри комп'ютерних наук, доц., кт.н. Альона МОСКАЛЕНКО _____
(підпис)

**Суми – 2024**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Сумський державний університет**
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
_____
(підпис)
червня 2024_ р.

**КВАЛІФІКАЦІЙНА РОБОТА на здобуття освітнього ступеня бакалавр**
**здобувача групи ІН-06-2 Бараболікової Анни Віталіївни**
зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної
програми «Інформатика»
на тему: «Інформаційна система управління контентом та замовленнями для роздрібної торгівлі»

здобувачки групи ІН-06-2 Бараболікової Анни Віталіївни

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Анна Бараболікова
_____
(підпис)

Старша викладачка кафедри
комп'ютерних наук, доц., кт.н. Альона Москаленко
_____
(підпис)

**Суми – 2024**

**Сумський державний університет**

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«Затверджую»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

_____

(підпис)

# ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

**на здобуття освітнього ступеня бакалавра**

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-06-2 Бараболікова Анна Віталіївна

1. Тема роботи: «Інформаційна система управління контентом та замовленнями для роздрібної торгівлі»

затверджую наказом по СумДУ від *22» квітня 2024 р. № 0414-VI*

2. Термін здачі здобувачем кваліфікаційної роботи *до 01 червня 2024 року*

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

**1** *Інформаційний огляд, аналіз предметної області, постановка задачі, моделювання інформаційної системи.*

**2** *Вибір методів рішення задачі.*

**3** *Програмна реалізація.*

**4** *Взаємодія користувача з системою.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| | | | |

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання Керівник

_____ _____

(підпис) (підпис)

# КАЛЕНДАРНИЙ ПЛАН

| № п/п | Назва етапів кваліфікаційної роботи | Термін виконання | Примітка |
|---|---|---|---|
| *1* | *Затвердження теми роботи* | 22.04.2024 | |
| *2* | *Вивчення та аналіз задачі* | 7.05.2024 | |
| *3* | *Поглиблене дослідження бібліотек, інструментів, технологій, що будуть використовуватися* | 10.05.2024 | |
| *4* | *Програмна реалізація системи* | 19.05.2024 | |
| *5* | *Аналіз отриманих результатів* | 22.05.2024 | |

Здобувач вищої освіти _____ (підпис)

Керівник _____ (підпис)

# АНОТАЦІЯ

**Записка**: 60 стр., 41 рисунок, 3 додаток, 13 використаних джерел.

**Обґрунтування актуальності теми роботи** – тема кваліфікаційної роботи є актуальною, оскільки присвячена створенню сучасної інформаційної системи для роздрібної торгівлі, враховуючи аспект зростання інтернет-торгівлі, зростання користування мобільними пристроями, зростання вимог користувачів в цілому.

**Об’єкт дослідження** — інформаційні системи управління контентом (CMS) та замовленнями (OMS) для роздрібної торгівлі.

**Мета роботи** — розробка та дослідження інформаційної системи управління контентом та замовленнями для роздрібної торгівлі.

**Методи дослідження** — аналіз користувацького досвіду, порівняльний аналіз з іншими ринковими учасниками.

**Результати** —створена система, яка відповідає всім поставленим вимогам. Вона має зручний інтерфейс для адміністраторів та клієнтів, дозволяє ефективно управляти контентом сайту та замовленнями, а також генерувати звіти про продажі.

**ЗМІСТ**

# ВСТУП

В сучасному світі, де веб-технології пронизали майже всі сфери життя, електронна комерція стала невід'ємною частиною торгівлі.

Для споживачів інтернет-магазини відкривають доступ до широкого спектру товарів та послуг у будь-який час і з будь-якої точки світу, роблячи покупки зручними та вигідними.

З іншого боку, для власників бізнесу онлайн-торгівля розширює можливості для реалізації товарів, полегшуючи організацію торгівлі та стимулюючи розвиток інноваційних технологій.

Торгівля – це багатогранний процес, який включає в себе оновлення товарів, співпрацю з брендами, створення акцій, контроль замовлень, роботу з клієнтами та багато іншого.

Ефективне управління всіма цими процесами потребує чіткої структури та автоматизації.

Сьогодні багато роздрібних підприємств стикаються з проблемою неефективності існуючих систем управління контентом та замовленнями.

Вирішення проблем, що виникли, за допомогою стандартних CMS (Система управління контентом) часто призводить до значного ускладнення сайту через необхідність встановлення багатьох додаткових плагінів та модулів.

Саме тому виникає гостра потреба в розробці комплексної інформаційної системи, яка оптимізує процеси управління контентом та замовленнями.

Ця дипломна робота присвячена дослідженню та розробці такої системи для роздрібних підприємств.

Мета дослідження:

Розробка комплексної інформаційної системи для оптимізації процесів управління контентом та замовленнями в роздрібних торговому підприємстві.

Оцінка ефективності впровадження розробленої системи на прикладі конкретного роздрібного підприємства.

Об'єкт дослідження:

Інформаційна система для управління контентом та замовленнями в роздрібному торговому підприємстві.

Предмет дослідження:

Процеси і методи управління контентом і замовленнями у роздрібній торгівлі з використанням інформаційних технологій.

Очікується, що результати дослідження:

Дозволять оптимізувати процеси управління контентом та замовленнями в роздрібних підприємствах, підвищивши їх ефективність.

Створять модель комплексної інформаційної системи, яку можна буде адаптувати до потреб інших роздрібних підприємств.

Допоможуть роздрібним підприємствам зміцнити свої позиції на ринку та розширити свою онлайн-присутність.

В ході дослідження буде використано широкий спектр методів та інструментів, таких як:

Аналіз літератури та наукових джерел.

Вивчення існуючих систем управління контентом та замовленнями.

Аналіз бізнес-процесів роздрібних підприємств.

Проектування та розробка комплексної інформаційної системи.

Тестування розробленої системи.

# 1 АНАЛІТИЧНИЙ ОГЛЯД

## 1.1 Сучасні технології розробки інформаційного і програмного забезпечення веб-орієнтованих систем електронної комерції

Сучасні веб-орієнтовані системи електронної комерції (ЕК) базуються на складних програмних та інформаційних технологіях, що забезпечують їх функціональність, масштабованість, безпеку та зручність використання. До ключових технологій, які використовуються в розробці.

Однією з таких є мови програмування: Вибір мови програмування залежить від складності проекту та необхідних функціональних можливостей. Популярними мовами програмування для розробки ЕК є PHP, Python, Java, JavaScript, C#.

Фреймворки веб-розробки: Фреймворки надають готові компоненти та бібліотеки, що спрощують та прискорюють процес розробки. Популярними фреймворками для ЕК є Laravel (PHP), Django (Python), Spring (Java), React (JavaScript), ASP.NET (C#).

Системи керування контентом (CMS): CMS дозволяють адмініструвати контент веб-сайту без глибоких знань програмування. Популярними CMS для ЕК є WordPress, Joomla!, Drupal, Shopify, Magento.

Бази даних: Бази даних використовуються для зберігання інформації про продукти, замовлення, користувачів та інші дані ЕК. Популярними базами даних для ЕК є MySQL, PostgreSQL, MongoDB.

Хмарні технології: Хмарні технології, такі як Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform, дозволяють масштабувати ЕК, забезпечувати її високу доступність та безпеку.

Також важливими є платіжні системи: Платіжні системи дозволяють користувачам оплачувати замовлення онлайн. Популярними платіжними системами для ЕК є Liqpay, WayForPay, Privat24, Google Pay, Apple Pay.

## 1.2Огляд та аналіз існуючих тематичних інтернет-магазинів

В ході аналізу тематичних інтернет-магазинів було обрано декілька веб-сайтів, які займаються роздрібною торгівлею.

Одним із таких сайтів є інтернет-магазин «Vmv-stroy» представлений на рисунку 1.2

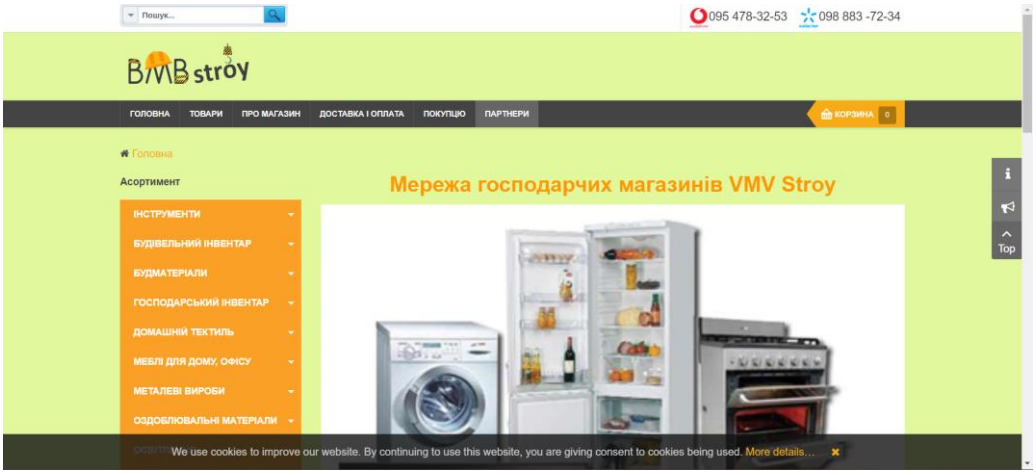



Рисунок 1.2 – Інтернет-магазин «Vmv-stroy»

Перший мінус, який відразу кидається в очі – це застарілий дизайн та великий банер з зображенням що містить мало інформації та незручне розташування пошуку. Для підвищення прихильності користувача було б краще оновити дизайн та розмістити пошук в більш доступному для користувача місці.

Другий мінус це відсутність списку бажань. Користувач може відібрати потрібний йому товар, що б через деякий час швидше та легше оформити замовлення.

Сучасні веб-додатки: виклики та шляхи вдосконалення

Сьогодні існує безліч веб-додатків, що пропонують подібні функції. Проте, багато з них мають певні недоліки, які негативно впливають на досвід користувачів. До таких недоліків належать:

Застарілий дизайн: Непривабливий та незручний інтерфейс може викликати недовіру у потенційних клієнтів та негативно вплинути на імідж компанії.

Незручність використання: Незрозумілі інструкції, складні навігаційні меню та технічні помилки можуть значно ускладнити користувачам процес здійснення замовлення, пошуку інформації чи отримання необхідних послуг.

Неактуальна інформація: Застарілі дані про товари, ціни, умови доставки чи інші важливі аспекти можуть призвести до розчарування користувачів та втрати їхньої довіри.

Щоб вирішити ці проблеми та покращити якість веб-додатків, необхідно:

Розробити сучасний та привабливий дизайн, який буде відповідати очікуванням користувачів та підкреслювати імідж компанії.

Спростити та оптимізувати інтерфейс, зробивши його максимально зручним та інтуїтивно зрозумілим.

Регулярно оновлювати інформацію, щоб вона завжди була актуальною та відповідала дійсному стану справ.

Проводити тестування та виявляти помилки, щоб забезпечити безперебійну роботу веб-додатку.

## 1.3Постановка задачі

Метою даної дипломної роботи є розробка інформаційної веб-орієнтованої системи електронної комерції, що буде відповідати сучасним вимогам та потребам ринку. Для досягнення поставленої мети необхідно виконати наступні завдання:

1) Розробити інформаційну модель системи:
2) Визначити структуру бази даних:
3) Виконати програмну реалізацію:
4) Провести тестування розробки

Очікується, що результатом даної дипломної роботи буде створена веб-орієнтована система електронної комерції, яка буде:

Функціональною та зручною у використанні.

Масштабованою та гнучкою.

Безпечною та надійною.

Відповідною сучасним тенденціям та потребам ринку.

# 2ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

## 2.1Діаграма діяльності

Діаграма діяльності - це графічне зображення виконаного набору дій процедурної системи та розглядається варіація діаграми стану[1].

На рисунку 2.1 представлена діаграма діяльності оформлення замовлення в інтернет-магазині. Ця діаграма корисна для розуміння того, як працює процес оформлення замовлення, а також для виявлення потенційних проблем або покращення ефективності. Рисунок 2.1 було створено за допомогою програми EdrawMax.

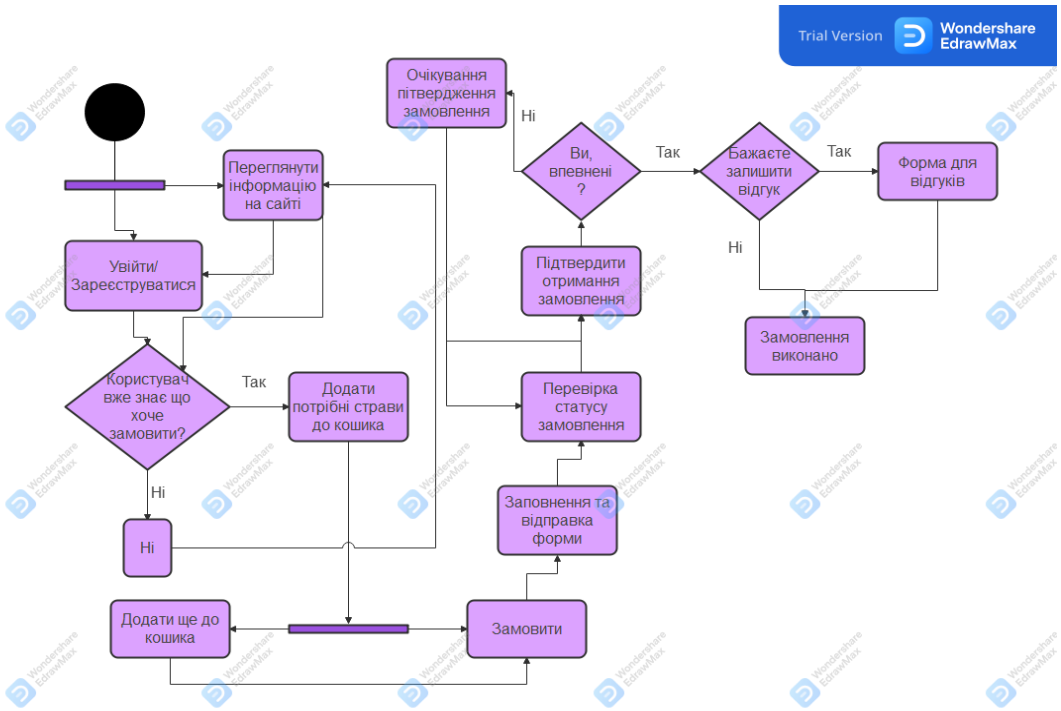



Рисунок 2.1 - Діаграма діяльності

На рисунку 2.1 зображена діаграма діяльності для оформлення замовлення. Користувач може обрати різні шляхи для отримання свого замовлення і відстежувати етапи розробки замовлення.

Опис діаграми:

Початок: Користувач обирає товар і додає його до кошика.

Перевірка кошика: Користувач переглядає кошик і може змінити кількість товарів або видалити їх.

Вхід до системи: Якщо користувач ще не зареєстрований, йому потрібно буде увійти до системи або створити новий обліковий запис.

Введення даних про доставку: Користувач вводить свою адресу доставки та контактну інформацію.

Вибір способу оплати: Користувач обирає спосіб оплати замовлення.

Підтвердження замовлення: Користувач підтверджує замовлення і вводить дані своєї платіжної картки.

Обробка замовлення: Система обробляє замовлення із підтвердженням користувача.

Доставка: Замовлення доставляється користувачеві.

## 2.2Діаграма розгортання

Діаграма розгортання UML - це графічний інструмент, який використовується для моделювання архітектури системи[2]. Вона зображує компоненти системи та їх зв'язки між собою.

На рисунку 2.2 представлена діаграма розгортання системи електронної комерції. Ця діаграма може бути корисною для розуміння того, як влаштована система, а також для виявлення потенційних проблем або покращення масштабованості. Рисунок 2.2.1 було створено за допомогою програми EdrawMax.

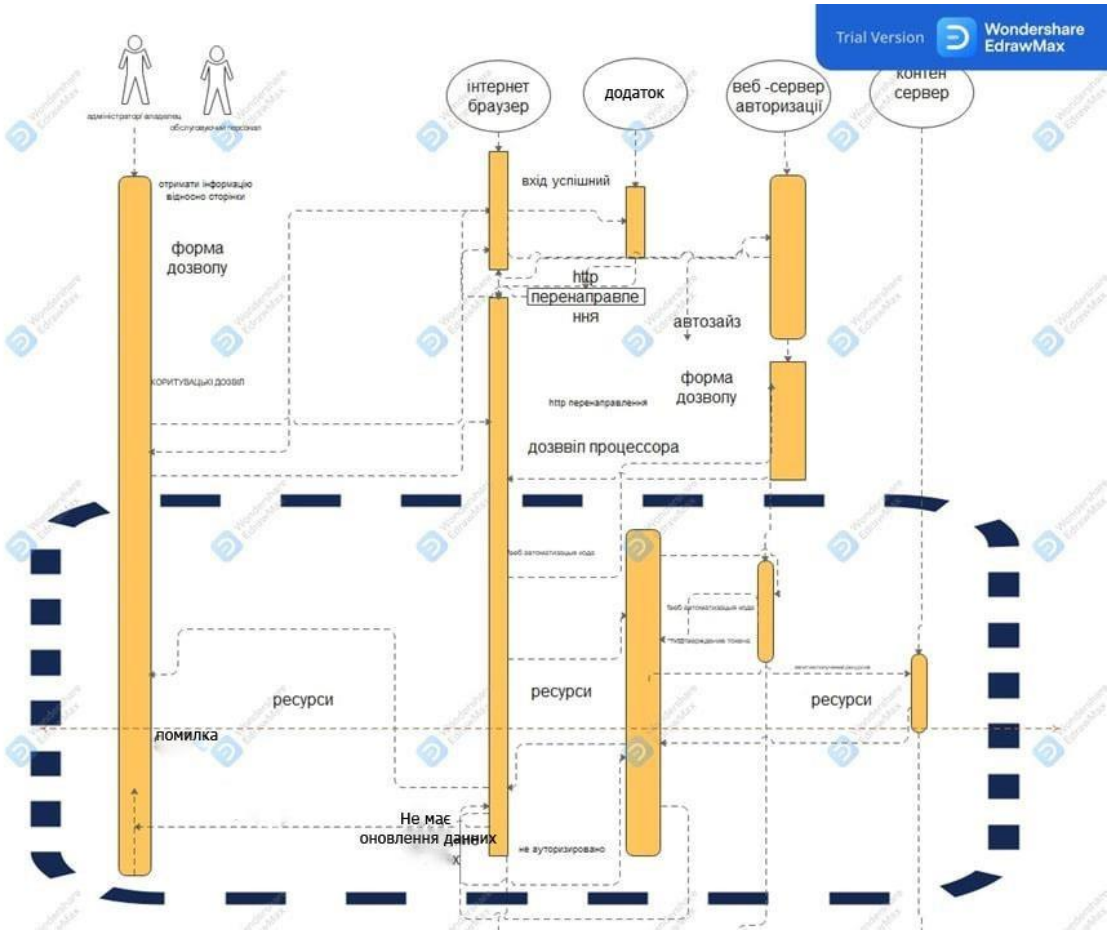



Рисунок 2.1 - Діаграма розгортання

## 2.3Діаграма послідовності

Діаграма послідовності UML - це графічний інструмент, який використовується для моделювання поведінки системи[3]. Вона зображує об'єкти системи та повідомлення, які вони надсилають один одному.

На рисунку 2.3 представлена діаграма послідовності оформлення замовлення в інтернет-магазині. Ця діаграма може бути корисною для розуміння того, як взаємодіють різні частини системи під час оформлення замовлення, а також для виявлення потенційних проблем або покращення ефективності. Рисунок 2.3 було створено за допомогою програми EdrawMax.

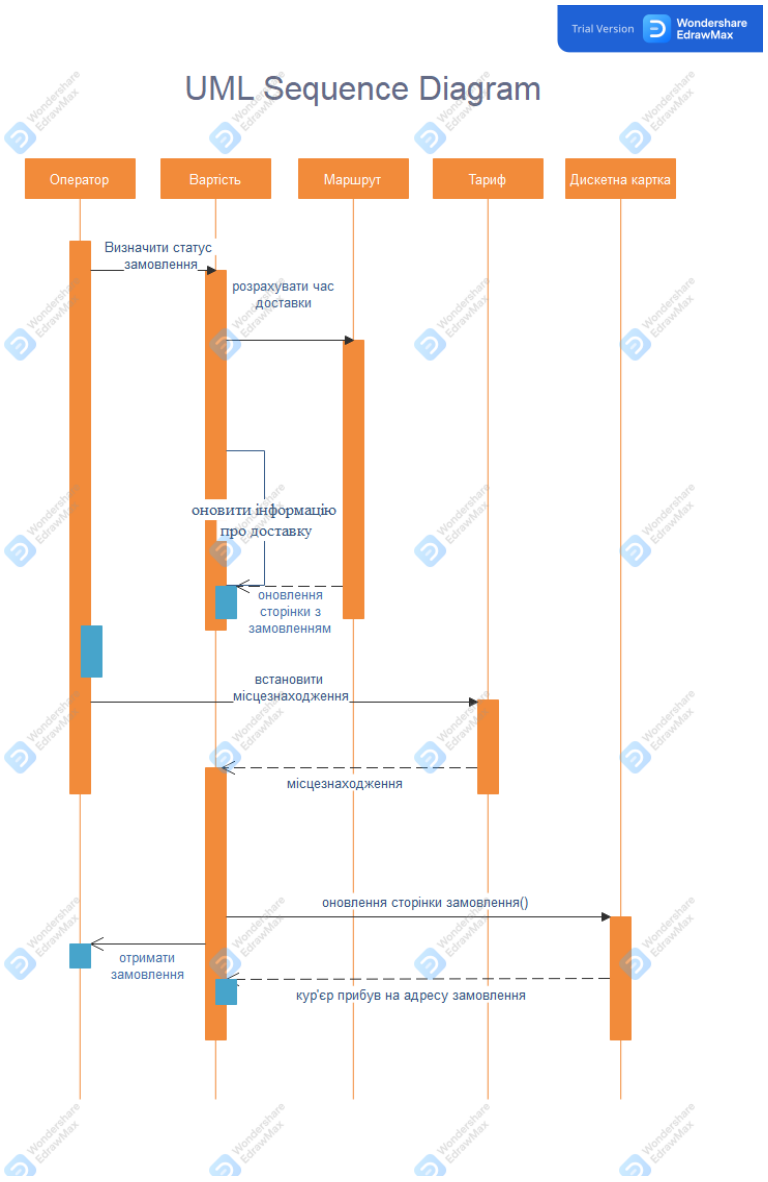



Рисунок 2.3 - Діаграма послідовності

## 2.4Структура програми

Шаблон MVC описує простий спосіб побудови структури програми, метою якого є відокремлення бізнес-логіки від інтерфейсу користувача. В результаті додаток легше масштабується, тестується, супроводжується і звичайно ж реалізується.

Розглянемо концептуальну схему шаблону MVC (на мій погляд – це найбільш вдала схема з тих, що я бачила):

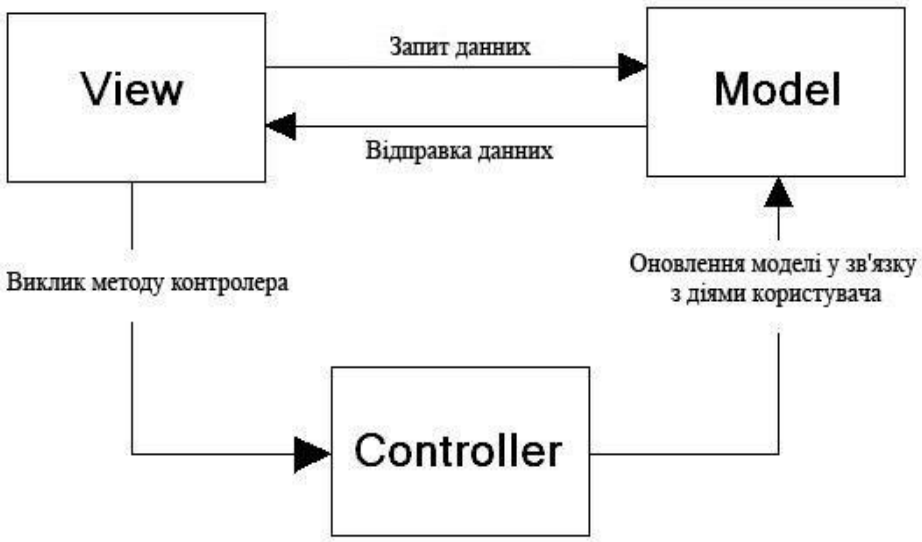



Рисунок 2.4 - Приклад роботи MVC

В архітектурі MVC модель надає дані та правила бізнес-логіки, подання відповідає за інтерфейс користувача, а контролер забезпечує взаємодію між моделлю та поданням.

Типову послідовність роботи MVC-програми можна описати так:

При заході користувача на веб-ресурс, скрипт ініціалізації створює екземпляр програми та запускає його на виконання.

При цьому відображається вигляд, скажімо, головної сторінки сайту.

Програма отримує запит від користувача та визначає запитані контролер та дію. У разі головної сторінки виконується дія за замовчуванням (index).

Додаток створює екземпляр контролера та запускає метод дії,

в якому, наприклад, містяться виклики моделі, які зчитують інформацію з бази даних.

Після цього дія формує подання з даними, отриманими з моделі і виводить результат користувачеві.

Модель містить бізнес-логіку програми і включає методи вибірки (це можуть бути методи ORM), обробки (наприклад, правила валідації) і надання конкретних даних, що часто робить її дуже товстою, що цілком нормально.

Модель не має безпосередньо взаємодіяти з користувачем. Усі змінні, що стосуються запиту користувача, повинні оброблятися в контролері.

Модель не повинна генерувати HTML або інший код відображення, який може змінюватись залежно від потреб користувача. Такий код має оброблятися у видах.

Одна й та сама модель, наприклад: модель аутентифікації користувачів може використовуватися як у користувальницькій, так і в адміністративній частині програми. У такому разі можна винести загальний код в окремий клас і успадковуватись від нього, визначаючи в спадкоємцях специфічні для додатків методи.

Вигляд використовується для визначення зовнішнього відображення даних, отриманих з контролера та моделі.

Види містять HTML-розмітку та невеликі вставки PHP-коду для обходу, форматування та відображення даних.

Не повинні безпосередньо звертатися до бази даних. Цим мають займатися моделі.

Не повинні працювати з даними, отриманими із запиту користувача. Це завдання має виконувати контролер.

Може безпосередньо звертатися до властивостей та методів контролера чи моделей, щоб отримати готових до виведення даних.

Види зазвичай поділяють на загальний шаблон, що містить розмітку, загальну для всіх сторінок (наприклад, шапку і підвал) та частини шаблону, які використовують для відображення даних, що виводяться з моделі або відображення форм введення даних.

Контролер — зв’язуюча ланка, що з'єднує моделі, види та інші компоненти робочого додатка. Контролер відповідає за обробку запитів користувача.

Контролер не повинен містити SQL-запитів. Їх краще тримати у моделях. Контролер не повинен містити HTML та іншу розмітку. Її варто виносити у види.

У добре спроектованому MVC-додатку контролери зазвичай дуже тонкі і містять лише кілька десятків рядків коду. Чого не скажеш про Stupid Fat Controllers (SFC) у CMS Joomla. Логіка контролера досить типова і більшість її виноситься в базові класи.

Моделі, навпаки, дуже товсті та містять більшу частину коду, пов'язану з обробкою даних. структура даних та бізнес-логіка, що міститься в них, зазвичай досить специфічна для конкретної програми.

У більшості випадків, взаємодія користувача з web-програмою проходить за допомогою переходів за посиланнями. Подивіться зараз на адресний рядок браузера – за цим посиланням ви отримали цей текст. За іншими посиланнями, наприклад, що знаходяться праворуч на цій сторінці, ви отримаєте інший вміст.

Таким чином, посилання надає конкретну команду web-додатку.

Сподіваюся, ви вже встигли помітити, що різні сайти можуть мати різні формати побудови адресного рядка. Кожен формат може відображати архітектуру веб додатків. Хоча це не завжди так, але в більшості випадків це явний факт. Розглянемо два варіанти адресного рядка, якими показується якийсь текст і профіль користувача.

Перший варіант:

www.example.com/article.php?id=3 www.example.com/user.php?id=4

Тут кожен сценарій відповідає за виконання певної команди.

Другий варіант:

www.example.com/index.php?article=3 www.example.com/index.php?user=4

А тут усі звернення відбуваються в одному сценарії index.php.

Підхід з безліччю точок взаємодії ви можете спостерігати на форумах з двигуном phpBB. Перегляд форуму відбувається через сценарій viewforum.php,

перегляд топіка через viewtopic.php тощо. Другий підхід з доступом через один фізичний файл сценарію можна спостерігати в моїй улюбленій CMS MODX, де всі звернення проходять через index.php.

Ці два методи суттєво відрізняються. Перший характеризується використанням шаблону контролера сторінок (Page Controller), тоді як другий підхід застосовує патерн контролера запитів (Front Controller). Контролер сторінок підходить для сайтів зі спрощеною логікою. Натомість, контролер запитів об'єднує всі дії з обробки запитів в одному місці, що надає йому додаткові можливості, завдяки яким можна вирішувати більш складні завдання, ніж ті, що вирішуються за допомогою контролера сторінок.

Маршрутизація URL дозволяє налаштувати додаток на прийом запитів з URL, які не відповідають реальним файлам програми, а також використовувати ЧПК, які семантично значущі для користувачів та кращі для пошукової оптимізації. Наприклад, для звичайної сторінки, що відображає форму зворотного зв'язку,

URL міг би виглядати так: http://www.example.com/contacts.php?action=feedback

З використанням движка маршрутизації URL для відображення тієї ж інформації налаштувати програму на прийом таких запитів: http://www.example.com/contacts/feedback

Тут contacts є контролер, а feedback - це метод контролера contacts, що відображає форму зворотного зв'язку і т.д.

Також варто знати, що маршрутизатори багатьох веб-фреймворків дозволяють створювати довільні маршрути URL (вказати, що означає кожна частина URL) та правила їх обробки.

Також для створення видів я використаю шаблонізатор Blade, HTML, CSS, JS, PHP

Blade - це простий, але потужний двигун шаблонів, що входить до складу Laravel. На відміну від деяких шаблонізаторів PHP, Blade не обмежує вас у використанні звичайного "сирого" коду PHP у ваших шаблонах. Насправді всі шаблони Blade компілюються у звичайний PHP-код і хешуються до тих пір, поки не будуть змінені, що означає, що Blade додає фактично нульове навантаження вашому додатку. [4]

Також при створенні веб-додатку буде застосовано такі мови як HTML (мова розмітки), CSS (мова стилів), JavaScript .

HTML (HyperText Markup Language) – це мова розмітки, використовувана для створення структури та визначення вмісту веб-сторінок [5]. Ця мова програмування дозволяє створити структуру документу використовуючи теги включаючи заголовки параграфи таблиці форми і т.д.

CSS (Cascading Style Sheets) – це мова стилізації, використовувана для задання зовнішнього вигляду веб-сторінок, стилю та макету [6]. Використання цієї мови програмування робить можливим відокремлення стилів веб документу від його структури і змінювати вигляд, наприклад розмір шрифтів, відступів, кольорів. CSS забезпечує контроль над виглядом веб-сторінок.

JavaScript – це мова програмування, яка використовується для розширення функціональності веб-сторінок та взаємодії з користувачем [7]. Дозволяє створювати динамічні ефекти, валідацію форм, асинхронну комунікацію з сервером та багато іншого.

PHP (PHP: Hypertext Preprocessor) – це скриптова мова програмування, спеціально розроблена для веб-розробки. PHP використовується для створення динамічних веб-сторінок, обробки форм, роботи з базами даних та багатьох інших завдань на серверному боці [8].

## 2.5Опис бази даних

В якості сервера бази даних для цього проекту було обрано MySQL. Це вільна та відкрита система керування реляційними базами даних (СУБД), яка

є популярним вибором для розробників веб-сайтів та програмного забезпечення.

Переваги роботи

Безкоштовність та відкритість: MySQL доступна безкоштовно для використання та модифікації, що робить її привабливим вибором для багатьох проектів.

Висока продуктивність: MySQL відома своєю високою продуктивністю та масштабованістю, що робить її підходящим вибором для сайтів з високим трафіком.

Широке розповсюдження: MySQL є однією з найпоширеніших СУБД в світі, що означає, що вам буде легко знайти інформацію та підтримку.

Простота використання: MySQL має простий та інтуїтивно зрозумілий інтерфейс командного рядка, а також зручний веб-інтерфейс керування, такий як phpMyAdmin.

Для роботи з MySQL використовується phpMyAdmin. Це безкоштовний веб-інтерфейс керування базами даних для MySQL, який дозволяє вам легко створювати, редагувати та видаляти бази даних, таблиці, поля та записи. phpMyAdmin також надає зручні інструменти для виконання SQL-запитів, імпорту та експорту даних та налаштування параметрів сервера.

У базі даних використовуються такі основні типи полів:

INT: Цей тип використовується для зберігання цілих чисел.

VARCHAR: Цей тип використовується для зберігання рядків символів з фіксованою довжиною.

TEXT: Цей тип використовується для зберігання рядків символів без обмеження довжини.

DATE: Цей тип використовується для зберігання дат.

DATETIME: Цей тип використовується для зберігання дат та часу.

DECIMAL: Цей тип використовується для зберігання десяткових чисел.

BOOLEAN: Цей тип використовується для зберігання логічних значень (TRUE/FALSE).

У базі даних використовувалися такі типи зв'язків:

Один до одного: Цей тип зв'язку використовується, коли один запис в одній таблиці може бути пов'язаний з одним записом в іншій таблиці.

Один до багатьох: Цей тип зв'язку використовується, коли один запис в одній таблиці може бути пов'язаний з багатьма записами в іншій таблиці.

Багато до багатьох: Цей тип зв'язку використовується, коли один запис в одній таблиці може бути пов'язаний з багатьма записами в іншій таблиці, і навпаки.

База даних складається з наступних таблиць:

Користувачі (tp_users):

Зберігає інформацію про користувачів системи, включаючи ім'я користувача, електронну пошту, пароль (в хешованому вигляді) та інші релевантні дані.

Токени відновлення пароля (tp_password_reset_tokens):

Зберігає тимчасові токени, які дозволяють користувачам відновлювати втрачені паролі.

Продукти (tp_products):

Зберігає основну інформацію про продукти, такі як назва, опис, ціна, зображення тощо.

Мета-дані продуктів (tp_product_metas):

Зберігати додаткові дані про продукти, які не поміщаються в основній таблиці продуктів: ключові слова, мета-описи для SEO.

Категорії продуктів (tp_product_categories):

Зберігає інформацію про категорії продуктів, до яких вони належать.

Підкатегорії продуктів (tp_product_sub_categories):

Зберігає інформацію про підкатегорії продуктів для більш детальної класифікації.

Вибране (tp_wishlists):

Зберігає список продуктів, які користувачі додали до свого "Вибраного".

Аналіз продуктів (tp_product_analyses):

Може зберігати дані щодо аналізу поведінки користувачів з продуктами (перегляди, додавання до кошика тощо).

Країни (tp_countries):

Зберігає інформацію про країни, які необхідні для оформлення замовлень або відображення на сайті.

Замовлення (tp_orders):

Зберігає інформацію про замовлення, включаючи дані про клієнта, товари в замовленні, загальну суму, статус замовлення тощо.

Платіжні шлюзи (tp_payment_gateways):

Зберігає інформацію про платіжні шлюзи, які використовуються для обробки платежів.

Замовлені товари (tp_order_product):

Зберігає зв'язок між таблицями "Замовлення" та "Продукти", визначаючи які товари входять до якого замовлення.

Купони знижок (tp_discount_coupons):

Зберігає інформацію про знижки і купони, включаючи код купона, розмір знижки, термін дії тощо.

Токени особистого доступу (tp_personal_access_tokens):

Зберігає токени, які дозволяють стороннім додаткам отримувати доступ до API системи.

Списки розсилок (tp_email_lists):

Зберігає списки електронних адрес для розсилок новин або рекламних повідомлень.

Електронні адреси підписників (tp_subscriber_emails):

Зберігає електронні адреси користувачів, які підписалися на розсилки.

Провайдери електронної пошти (tp_email_providers):

Зберігати інформацію про провайдерів електронної пошти, які використовуються для розсилок.

Коментарі (tp_comments):

Зберігає коментарі користувачів до продуктів, статей або інших елементів системи.

Сторінки (tp_pages):

Зберігає інформацію про статичні сторінки сайту, та їхній контент.

Налаштування (tp_settings):

Зберігає різні налаштування системи, такі як валюта, часовий пояс, налаштування електронної пошти тощо.

Рейтинги (tp_ratings):

Зберігає рейтинги продуктів, залишені користувачами.

Мови (tp_languages):

Зберігає інформацію про мови, які підтримуються системою.

Контент головної сторінки (tp_home_contents):

Зберігає інформацію про контент головної сторінки сайту, банери, блоки з товарами, текстові блоки тощо.

Контактна форма (tp_contactus):

Зберігає дані, що надходять через контактну форму на сайті, такі як ім'я, електронна пошта, повідомлення користувача.

Невиконані задачі (tp_failed_jobs):

Зберігає інформацію про фонові задачі, які не вдалося виконати успішно.

Міграції (tp_migrations):

Зберігає інформацію про стан міграцій бази даних, що використовується для управління змінами структури таблиць.

Відгуки (tp_testimonials):

Зберігає інформацію про відгуки користувачів або клієнтів.

Описана структура бази даних відповідає всім вимогам проекту, забезпечуючи ефективне зберігання, керування та доступ до даних. Використання MySQL як сервера бази даних, разом з phpMyAdmin та чітко продуманою архітектурою, робить базу даних надійною, масштабованою та зручною у використанні.

# 3. ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

## 3.1Вибір засобів програмної реалізації

Для створення даного програмного забезпечення було розглянуто декілька фреймворків таких як Laravel, Symfony і Yii2 кожен з яких має свої переваги та недоліки.

Плюси та мінуси фреймворків: Laravel

Плюси :

Чудова IoC (Інверсія управління); Зручна система міграцій; Інтегрована система модульного тестування; Вбудований шаблонізатор Blade; Дуже гнучке формування роутів; Дуже гнучкі можливості для написання REST API; Дуже швидко розвивається; Море документації на будь-яку тему; Популярний на Заході; Консоль налагодження із коробки зі стеком викликів; RBAC; ACL плагіни

Мінуси

Великий функціонал працює через фасади, і IDE-системи не бачать методів та властивостей у деяких класах, показуючи попередження; Вивчається трохи складніше; Немає інтегрованих генераторів інтерфейсів Symfony

Плюси

Відноситься так само до Yii. До того ж, документація Symfony наполягає на тому, що це не MVC-фреймворк. Вбудована підтримка Codeception дозволяє писати функціональні та приймальні тести

YAML як плюс фреймворку

Деякі з компонентів Symfony2 тепер реалізуються у великих проектах, таких як Drupal та PhpBB.

Велика спільнота розробників; Багато готових модулів (бандлів); Детальна та зрозуміла документація; Досить висока швидкість роботи ядра; Низька складність компонентів;

Плюси

Візуальний генератор коду; Найбільш інтуїтивна MVC-архітектура. Особливо з нуля його вчити приємно • Використовує стандартні способи вирішення задач, що зменшує або усуває заплутаність коду; Полегшує підтримку коду за допомогою загальної архітектури та методів • Має активну спільноту розробників, які підтримують фреймворк, узагальнені завдання та нові можливості; Скорочує час, що витрачається на рутинні завдання, такі як перевірка форм та безпека; Легко налаштувати для кращої продуктивності; Невибагливий до ресурсів; Різні варіанти хешування; Проста інтеграція сторонніх бібліотек та класів; Якісні інструменти безпеки; Поводження, хелпери, можливість розширення базового функціоналу тощо; Підходить для проектів будь-якої складності та масштабу.

Мінуси

Не дуже гнучке формування роутів

Занадто склеєні бібліотеки для frontend'a з backend'ом

Проаналізувавши всі плюси та мінуси для розробки веб ресурсу я обрала Laravel адже він один з найпопулярніших та містить всі потрібні мені рішення.

## 3.2 Програмна реалізація

### 3.2.1 Розробка класів

Клас - це такий шаблон, за яким створюються об’єкти. Він допомагає групувати разом пов’язані змінні та функції[10].

Клас - логічне опис чого-небудь, шаблон, за допомогою якого можна створювати реальні екземпляри цього самого чогось. Іншими словами, це просто опис того, якими мають бути створені суті: якими властивостями і методами повинні володіти. Приклад класу «User».

Цей клас User визначає модель користувача в Laravel framework. Він успадковується від класу Authenticatable пакета Laravel Illuminate\Foundation\Auth, що надає базову функціональність для автентифікації користувачів.

Властивості:

$fillable: Масив атрибутів користувача, які можна призначати масово під час створення або оновлення користувача. Це такі поля, як email, password, full_name тощо.

$hidden: Масив атрибутів користувача, які не повинні бути включені до JSON або масивів даних, що повертаються API. Це зазвичай стосується конфіденційних даних, таких як пароль.

$casts: Масив атрибутів, які потрібно привести до певних типів даних. Наприклад, email_verified_at приводиться до типу datetime.

Методи:

getavatarAttribute($path): Цей метод-аксесор використовується для отримання повного шляху до зображення профілю користувача. Він перевіряє, чи є у користувача google_id або facebook_id і повертає відповідне посилання, або використовує метод asset для отримання зображення зі сховища Laravel, якщо воно завантажене користувачем.

save(array $options = []): Цей перевизначений метод save перевіряє, чи файл зображення профілю завантажено за допомогою запиту request, і зберігає його у сховищі, якщо він дійсний. Він також використовує метод folderPath для визначення папки зберігання.

folderPath(): Цей метод визначає папку для зберігання зображень профілю користувачів. Шлях базується на поточному місяці та році.

save_image($request): Цей метод зберігає зображення профілю користувача у сховищі та повертає його шлях.

getCountry(): Цей метод відноситься до моделі Country за допомогою відношення hasOne. Він дозволяє отримати інформацію про країну користувача, якщо вона пов'язана.

getProductCount(): Цей метод відноситься до моделі Product за допомогою відношення hasMany. Він повертає кількість продуктів, доданих цим користувачем.

getProductSales(): Цей метод відноситься до моделі Product за допомогою відношення hasMany та виконує агрегатну функцію SUM для підрахунку загальної кількості продажів продуктів, доданих цим користувачем.

getOrders(): Цей метод відноситься до моделі Order за допомогою відношення hasMany. Він повертає список замовлень, розміщених цим користувачем, відсортованих за датою (останні замовлення відображаються першими).

scopeFilter($query): Цей метод області дії застосовує фільтри до запиту користувачів на основі параметрів request. Він може фільтрувати користувачів за різними полями, використовуючи різні оператори порівняння.

$searchable: Цей статичний масив визначає поля користувача, які можна використовувати для пошуку. Він містить ключі, що відповідають назвам полів у базі даних, та значення, що відображаються користувачеві як заголовки полів пошуку. У таблиці3.2.1.1 наведено ключові атрибути класу User.

Таблиця.3.2.1.1 Ключові атрибути класу «User».

| Атрибут | Опис |
|---|---|
| id | Унікальний первинний ключ для ідентифікації користувача. |
| role | Роль користувача (0 - адміністратор, 1 - користувач, 2 - продавець). |
| role_type | Символьний опис ролі користувача (ADMIN, USER, VENDOR). |

| Атрибут | Опис |
|---|---|
| avatar | URL або шлях до зображення профілю користувача (за замовчуванням порожній). |
| username | Ім'я користувача (за замовчуванням порожній). |
| full_name | Повне ім'я користувача (за замовчуванням порожній). |
| gender | Стать користувача (за замовчуванням порожній). |
| email | Адреса електронної пошти користувача. |
| password | Зашифрований пароль користувача. |
| mobile | Номер телефону користувача (за замовчуванням порожній). |
| birthday | Дата народження користувача. |
| address | Адреса проживання користувача (за замовчуванням порожній). |
| country_id | Ідентифікатор країни користувача (зв'язок з іншою таблицею). |
| city | Місто користувача (за замовчуванням порожній). |
| state | Штат/область користувача (за замовчуванням порожній). |
| zip_code | Поштовий індекс користувача (за замовчуванням порожній). |
| is_active | Чи активний користувач (1 - активний, 0 - неактивний). |
| is_email_verified | Чи підтверджена електронна пошта користувача (1 - підтверджена, 0 - непідтверджена). |
| email_verified_at | Дата та час підтвердження електронної пошти користувача. |
| is_mobile_verified | Чи підтверджений номер телефону користувача (1 - підтверджений, 0 - непідтверджений). |
| mobile_verified_at | Дата та час підтвердження номера телефону користувача. |
| rating | Рейтинг користувача (за замовчуванням 0). |
| created_at | Дата та час створення запису користувача. |

| Атрибут | Опис |
|---|---|
| updated_at | Дата та час останнього оновлення запису користувача. |
| deleted_at | Дата та час видалення запису користувача (використовується для м'якого видалення). |

Деякі атрибути, такі як avatar, username, full_name, address, city, state, zip_code є порожніми за замовчуванням.

Атрибути is_active, is_email_verified, is_mobile_verified мають значення за замовчуванням 1 (активний, підтверджений) 0 (неактивний, не підтверджений).

Атрибути email_verified_at та mobile_verified_at містять timestamp, коли було підтверджено електронну пошту або номер телефону.

Атрибут rating за замовчуванням 0.

Атрибути created_at, updated_at та deleted_at автоматично оновлюються системою.

### 3.2.2 Розробка міграцій

Міграція даних — це процес переміщення великих обсягів даних із одного місця до іншого [11]. Міграції - це щось подібне до системи контролю версій для вашої бази даних. Вони дозволяють команді програмістів змінювати структуру БД, водночас залишаючись у курсі змін інших учасників. Міграції зазвичай йдуть пліч-о-пліч з конструктором таблиць для більш простого поводження з архітектурою додатку.

Міграція create_users створює таблицю users для зберігання інформації про користувачів у вашій базі даних. Таблиця містить поля для ідентифікації користувача, його ролі, автентифікації, контактної інформації, рейтингу тощо.

Міграція використовує клас Migration від Laravel для визначення методу up(), який виконує створення таблиці. Ось розбивка основних рядків коду:

Schema::create('users', function (Blueprint $table) { ... }); - Цей рядок створює нову таблицю під назвою users. Аргумент function містить визначення структури таблиці.

$table->id(); - Створює стовпець id з автозбільшуваним цілим числом як первинним ключем для ідентифікації кожного користувача.

$table->tinyInteger('role')->comment('role 0 for admin , 1 for user , 2 for vendor'); - Створює стовпець role типу tinyInteger (маленьке ціле число) для зберігання ролі користувача. Значення 0 відповідає адміністратору, 1 - користувачу, 2 - продавцю. Коментар пояснює ці значення.

$table->string('role_type')->comment('ADMIN,USER,VENDOR'); - Створює стовпець role_type типу string для зберігання текстового опису ролі користувача (ADMIN, USER, VENDOR). Це поле дублює значення ролі, але може бути корисним для відображення.

Інші рядки $table визначають інші стовпці таблиці users. Деякі стовпці мають значення за замовчуванням NULL, що дозволяє зберігати порожні значення.

### 3.2.3 Розробка моделей

Laravel включає Eloquent, об'єктно-реляційний перетворювач (ORM), який дозволяє із задоволенням взаємодіяти з базою даних. При використанні Eloquent кожна таблиця бази даних має відповідну Модель, яка використовується для взаємодії з цією таблицею. Крім отримання записів з таблиці бази даних, моделі Eloquent також дозволяють вставляти, оновлювати та видаляти записи з таблиці.

Наданий приклад моделі User, представляє сутність користувача в програмі. Вона знаходиться в просторі імен App\Models та успадковує від класу Authenticatable Laravel, який дає можливість автентифікації користувачів.

Основні характеристики:

Дані користувача: Зберігає інформацію про користувача, таку як електронна пошта, ім'я, дані для входу через соціальні мережі (необов'язково), шлях до аватару та інформацію про роль.

М'яке видалення: Реалізує м'яке видалення за допомогою трейта SoftDeletes, що дозволяє логічно видаляти користувачів замість остаточного видалення.

Взаємозв'язки: Встановлює зв'язки з іншими моделями:

Country: Зв'язок типу "належить" з моделлю Country, що дозволяє асоціювати користувача з його країною.

Product: Зв'язок типу "має багато" з моделлю Product, що дозволяє користувачеві мати декілька продуктів (якщо це застосовується).

Order: Зв'язок типу "має багато" з моделлю Order, що дозволяє користувачеві мати декілька замовлень.

Спеціальні геттери: Визначає спеціальні геттери для певних атрибутів:

getavatarAttribute: Обробляє шлях до аватару, враховуючи дані для входу через соціальні мережі та розташування сховища.

Завантаження зображень та керування шляхами: Надає функціональні можливості для завантаження зображень та генерування шляхів:

save: Обробляє завантаження зображення, якщо через запит надано дійсний файл зображення. Зберігає зображення у спеціальній папці користувача.

folderPath: Генерує динамічний шлях до папки на основі поточного місяця та року для зберігання зображень користувачів.

save_image: Зберігає завантажене зображення в папку користувача.

Функціональність пошуку: Реалізує область видимості під назвою scopeFilter, щоб дати можливість шукати користувачів на основі наданих критеріїв, таких як термін пошуку, тип фільтра (містить, більше/дорівнює,

менше/дорівнює) та ключ пошуку. Модель також визначає статичний властивість $searchable, який перелічує поля, доступні для пошуку, та їх мітки.

Загалом, модель User слугує основою для керування даними користувачів, зв'язками, завантаженням зображень та функціональними можливостями пошуку в програмі.

### 3.2.4 Розробка контролерів

Замість визначати всю логіку обробки запитів як замикання у файлах маршрутів, можна організувати цю поведінку за допомогою класів «контролерів». Контролери можуть згрупувати пов'язану логіку обробки запитів на один клас. Наприклад, UserControllerклас може обробляти всі вхідні запити, що стосуються користувачів, включаючи відображення, створення, оновлення та видалення користувачів. За замовчуванням контролери зберігаються в app/Http/Controllers каталозі[12].

Розглянемо UserController, який керує автентифікацією користувачів, реєстрацією, керуванням профілями та скиданням паролів у фронт-енді додатку. Він успадковує від класу Controller Laravel та використовує різні можливості та функціональні можливості Laravel.

Ключові функціональні можливості:

Вхід користувача (метод login):

Перевіряє облікові дані користувача (електронну пошту та пароль).

Перевіряє статус активації користувача та підтвердження електронної пошти.

Виконує автентифікацію за допомогою методу Auth::attempt Laravel.

Повертає відповідні JSON-відповіді залежно від успішності або невдачі входу.

Реєстрація користувача (метод signup):

Перевіряє інформацію користувача (ім'я, електронна пошта, пароль, підтвердження пароля).

Створює новий запис користувача в базі даних за допомогою моделі User.

Надсилає електронний лист із підтвердженням електронної пошти, якщо налаштовано (використовуючи метод sendMail).

Повертає JSON-відповіді, що вказують на успіх або невдачу реєстрації.

Вихід користувача (метод logout):

Виводить користувача з системи за допомогою методу Auth::logout Laravel.

Перенаправляє користувача на попередню сторінку.

Оновлення профілю користувача (метод updateProfile):

Перевіряє інформацію профілю користувача (користувацьке ім'я, повне ім'я, необов'язковий мобільний номер, країна, місто, штат і адреса).

Оновлює дані профілю користувача в базі даних.

Повертає JSON-відповіді, що вказують на успіх або невдачу оновлення профілю, включаючи URL-адресу перенаправлення на сторінку профілю.

Зміна пароля (метод change_password):

Перевіряє старий пароль, новий пароль (включаючи вимоги до складності) та підтвердження пароля.

Перевіряє старий пароль проти поточного пароля користувача.

Оновлює пароль користувача в базі даних за допомогою хешування.

Повертає JSON-відповіді, що вказують на успіх або невдачу оновлення пароля.

Оновлення зображення користувача (метод update_user_image):

Перевіряє завантажене зображення користувача (наявність, формат, розмір).

Зберігає завантажене зображення в певній папці користувача (реалізація не показана).

Повертає JSON-відповіді, що вказують на успіх або невдачу оновлення зображення, включаючи URL-адресу перенаправлення на сторінку профілю.

Забули пароль (метод forgot_password):

Перевіряє надану адресу електронної пошти.

Перевіряє наявність запитів на скидання пароля для цієї електронної пошти.

Створює унікальний токен, якщо не існує нещодавнього запиту, і надсилає електронний лист із скиданням пароля (використовуючи метод sendMail).

Повертає JSON-відповіді, що вказують на успіх або невдачу надсилання електронного листа для скидання.

### 3.2.5 Розробка видів

Звичайно, недоцільно повертати цілі рядки HTML-документів безпосередньо з ваших маршрутів та контролерів. На щастя, уявлення надають зручний спосіб розмістити весь HTML-код в окремих файлах[13]. Уявлення відокремлюють логіку контролера / програми від логіки уявлення та зберігаються в resources/views каталозі.

Код успадковує від базового макета майстра входу (frontend.layout.login_master), який, визначає основну структуру та стилі сторінки входу.

Розглянемо деякі елементи:

Секція Header (@section('head_scripts')):

Цей розділ динамічно генерує meta-теги та властивості Open Graph (OG) для SEO та обміну в соціальних мережах:

$ASSET_URL: Встановлює базову URL-адресу для ресурсів (наприклад, зображення, JavaScript).

$setting: Отримує налаштування з сховища програми, використовуючи допоміжну функцію getsetting().

$mUrl, $mImage, $mTitle, $mDescr, $mKWords: Отримує різні налаштування з $setting (наприклад, зображення попереднього перегляду, заголовок сторінки, описи meta, ключові слова та ім'я творця сайту).

Заголовок, описи meta, ключові слова та властивості OG динамічно встановлюються на основі отриманих налаштувань.

Секція Content (@section('content')):

Цей розділ визначає основний контент, який відображається на сторінці входу:

HTML-структура для форми входу та її оточуючих елементів.

@lang директива: використовується для локалізації мови, отримуючи перекладений текст із ресурсних файлів на основі поточної мови програми (en, uk).

# 4 Використання програмного додатку

## 4.1 Клієнтська частина

Коли користувач переходить за відповідним посиланням, перед ним відкривається головна сторінка сайту (Рисунок 4.1.1), на якій знаходиться: навігаційне меню сайту, де зазначено логотип сайту, посилання на інші сторінки, перемикач мови, кнопка авторизації, пошук та банер.

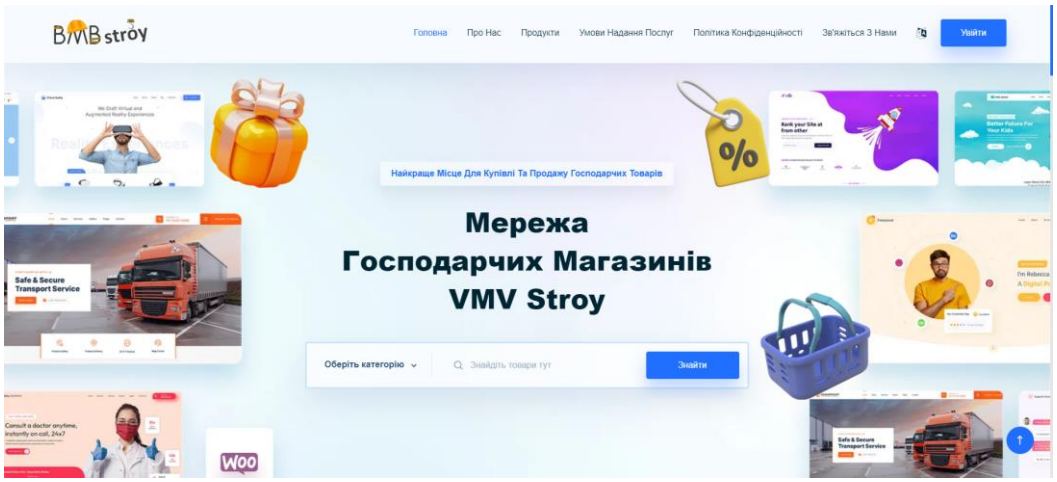



Рисунок 4.1.1 – Головна сторінка

Пошук на сайті працює, достатньо ввести запит на бажаний товар (Рисунок 4.1.2, 4.1.3).


Рисунок 4.1.2 – Пошук

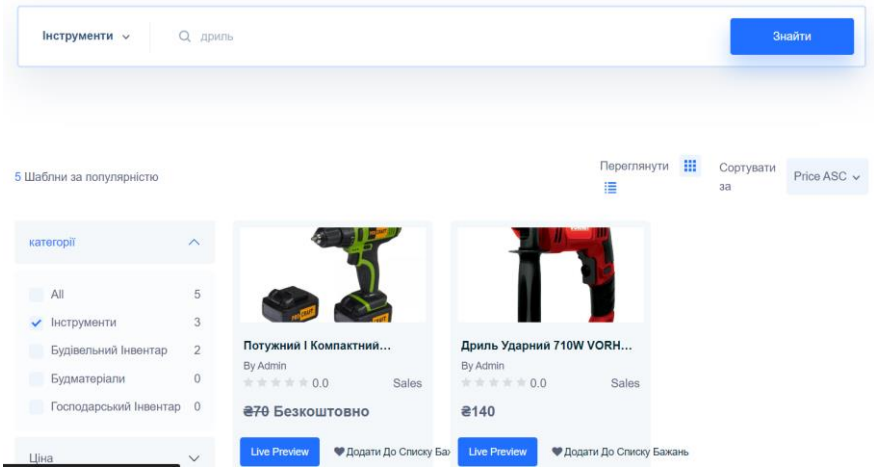



Рисунок 4.1.3 – Результат пошуку

Далі при скролі вниз користувач може побачити, список найновіших товарів, сортування товарів, топ продажів (рис 4.1.4, 4.1.5).

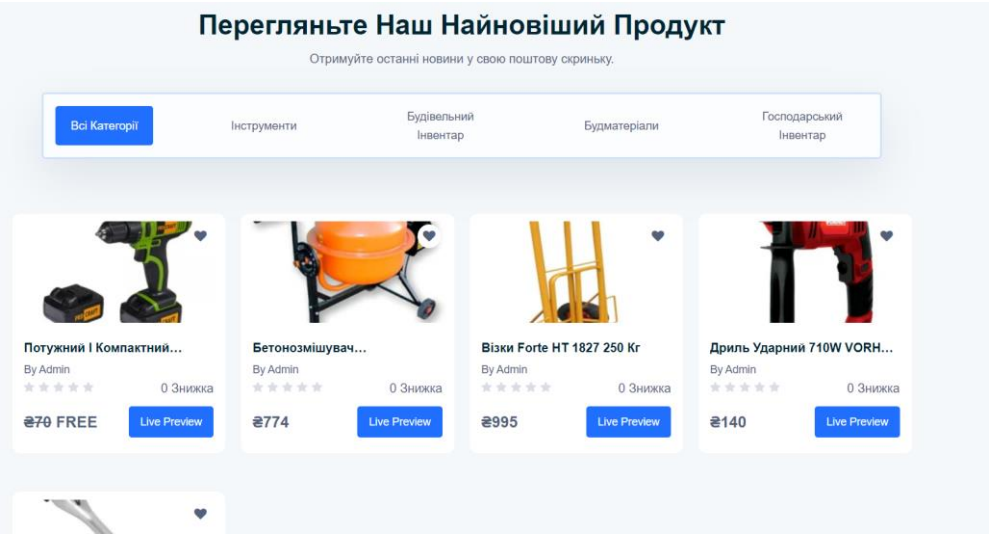



Рисунок 4.1.4 – Список нових товарів

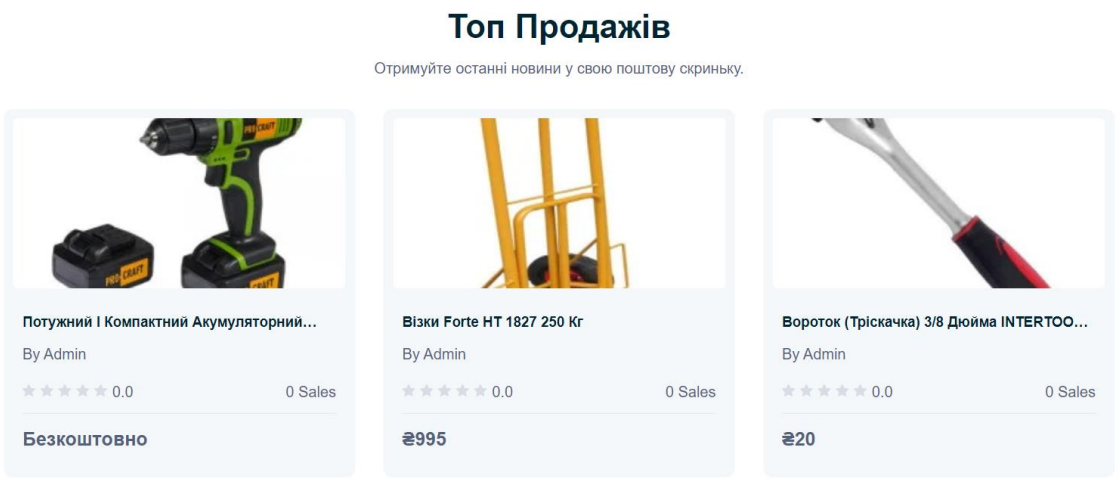



Рисунок 4.1.5 – Топ продажів

Сортування на сайті працює, достатньо обрати категорію до якої входить бажаний товар (4.1.6).

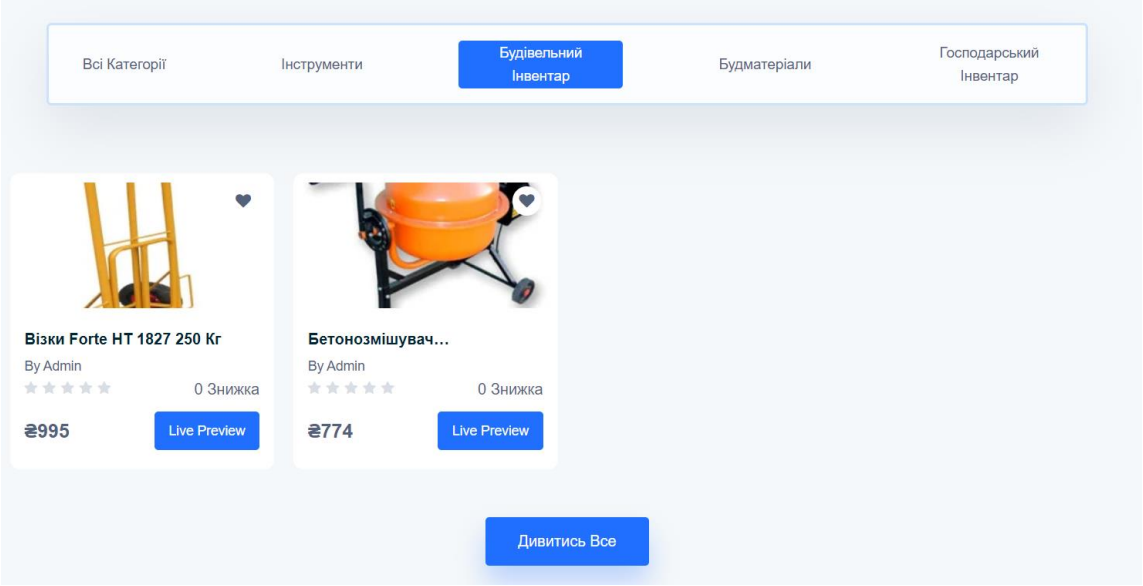



Рисунок 4.1.6 – Результати сортування

При натисканні кнопки «Продукти» перед користувачем відкриється сторінка товарів, які можна фільтрувати критеріями вказаними на Рисунок 4.1.7 та 4.1.8

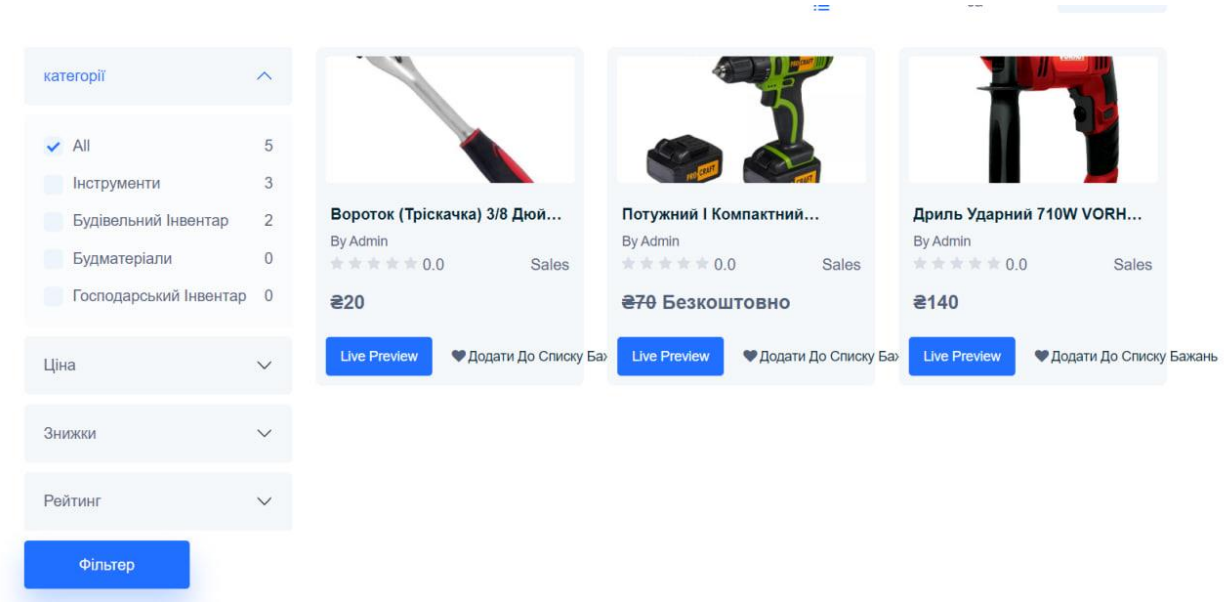



Рисунок 4.1.7 – Сторінка «Каталог»

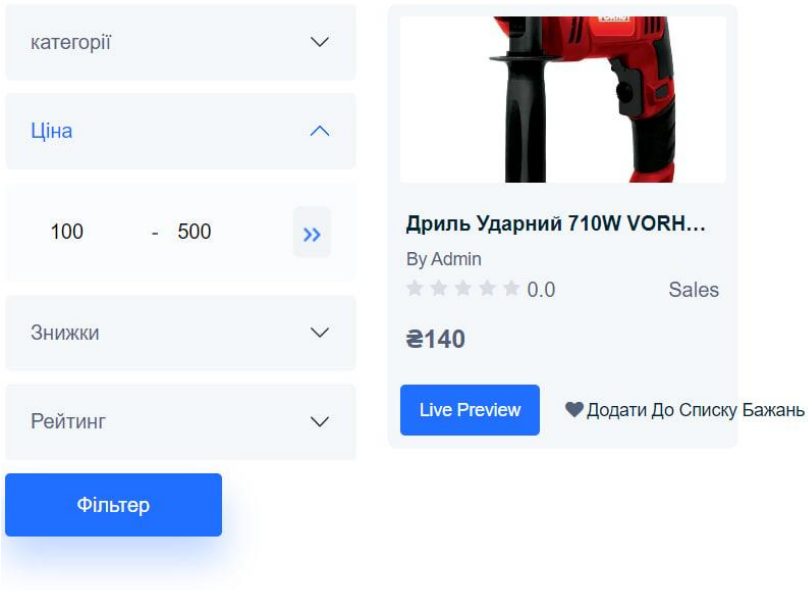



Рисунок 4.1.8 – Приклад сортування товарів

Коли користувач переходить на сторінку «Умови Надання Послуг», перед ним відкривається детальна інформація про надані гарантії та правила

повернення товару (Рисунок 4.1.9).

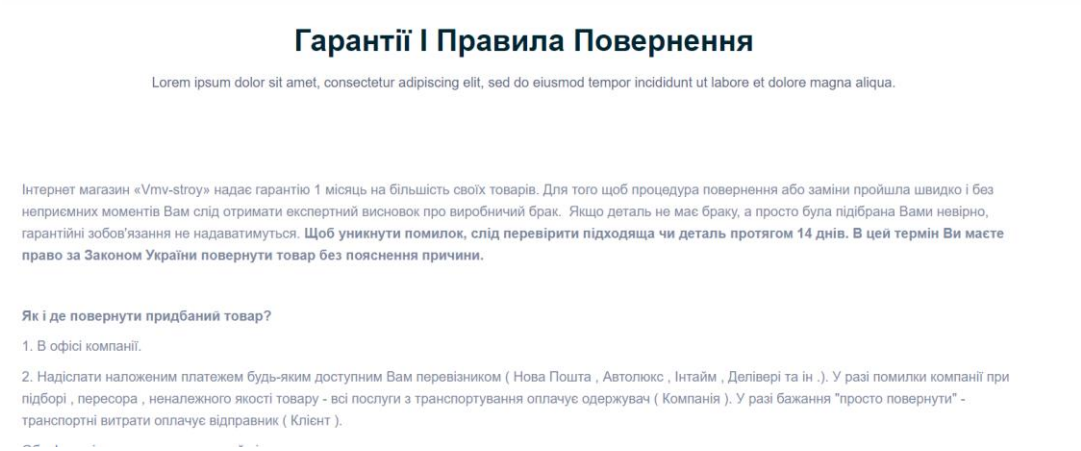



Рисунок 4.1.9 – Сторінка «Умови Надання Послуг»

На сторінці «Про нас» вказана інформація про адресу, телефон та ін. магазину (Рисунок 4.1.10).

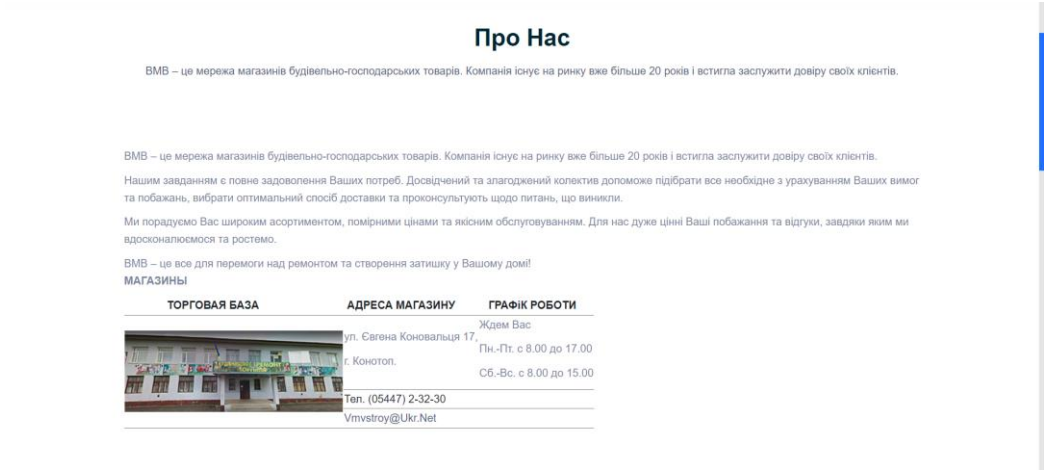



Рисунок 4.1.10 – Сторінка «Про нас»

Коли користувач переходить на сторінку «Зв'яжіться З Нами», перед ним відкривається інформація про E-mail адресу для надсилання листа, контактний норер, та форма для зв'язку (Рисунок 4.1.11).

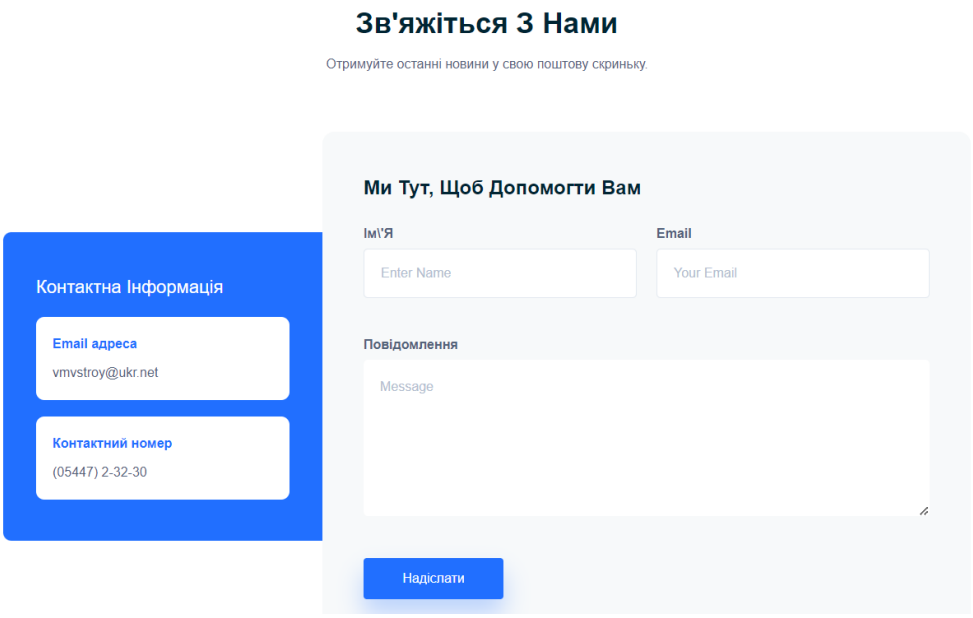



Рисунок 4.1.11 – Вигляд сторінки «Зв'яжіться З Нами»

Користувач може переглянути дані про товар, натиснувши на товар на головній сторінці або в розділі каталогу. Відображення сторінки представлено на Рисунок 4.1.12.

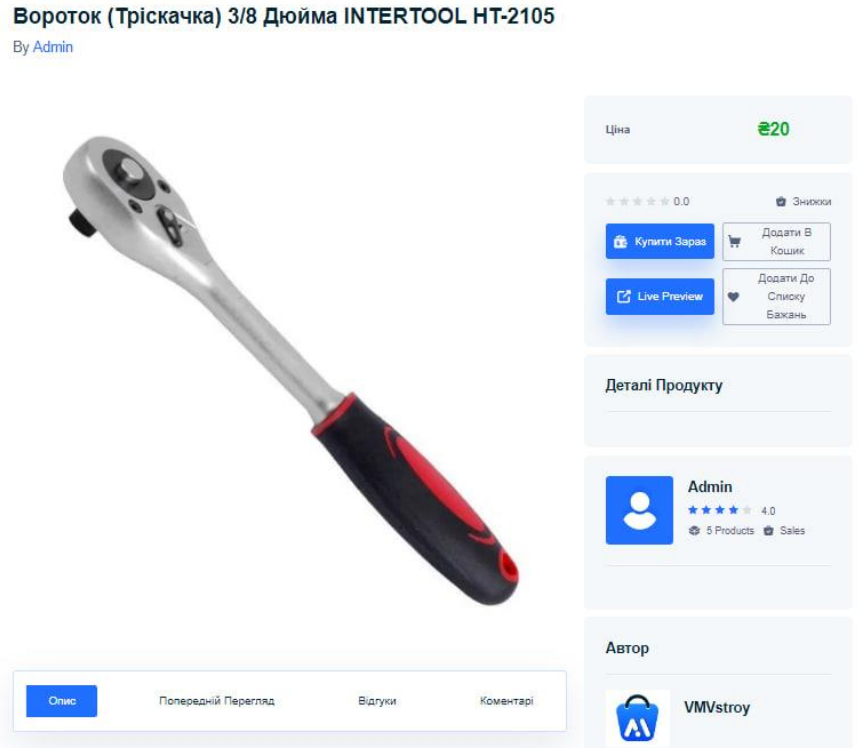



Рисунок 4.1.11 – Перегляд товару

Користувач може додати бажаний товар до кошика. На сторінці кошика (Рисунок 4.1.12) користувач може додати вибраний товар до списку бажань або видалити вибраний товар, та перейти до оформлення замовлення.

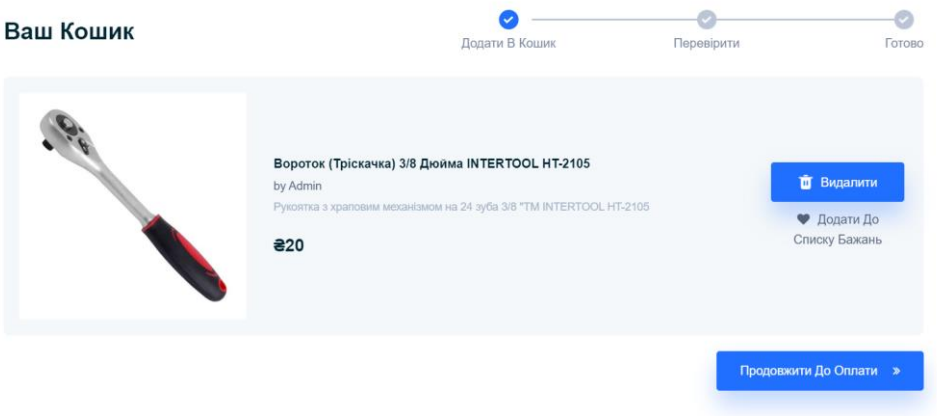



Рисунок 4.1.12 – Сторінка кошика

При натисканні кнопки «Продовжити до оплати» у випадку якщо користувач увійшов в систему, тоді він потрапить до сторінки з формою, де потрібно обрати платіжну систему, та за наявності можне скористатися знижкою (Рисунок 4.1.13), але якщо користувач не увійшов в систему на екрані з’явится форма для авторизації(Рисунок 4.1.14), Вибравши казавши всі дані, користувач може підтвердити замовлення.

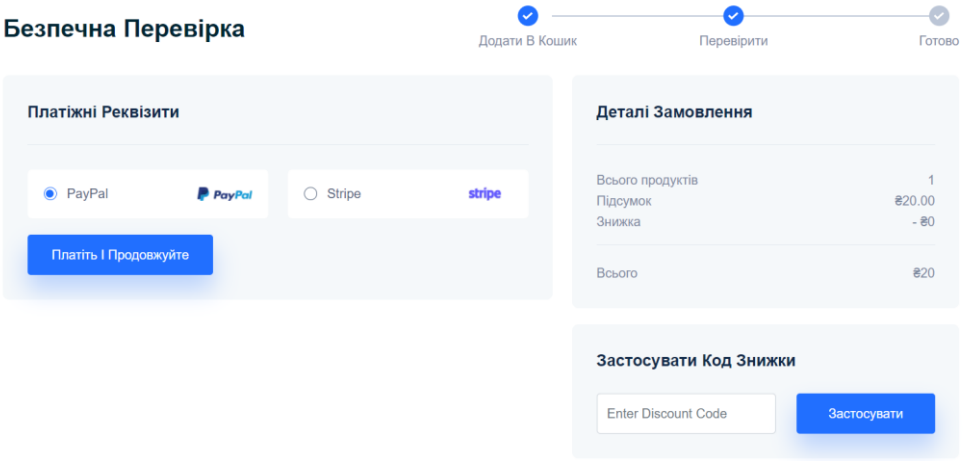



Рисунок 4.1.13 – Сторінка оплати

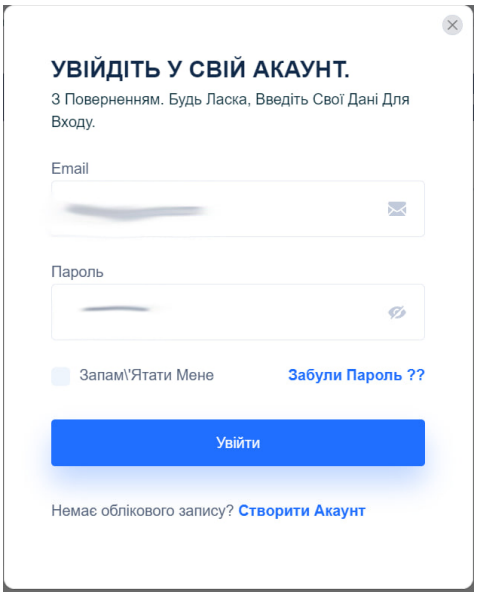



Рисунок 4.1.14 –Форма входу

За допомого навігаційного меню яке змінило вигляд після авторизації (рис 4.1.15) користувач має можливість переглянути свій профіль де може оновити інформацію,змінити пароль, перевірити замовлення (рис 4.1.16),а також переглянути список бажань (рис 4.1.17).

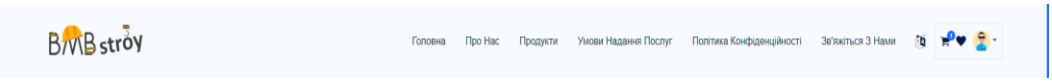



Рисунок 4.1.15 – Знаходження меню профілю

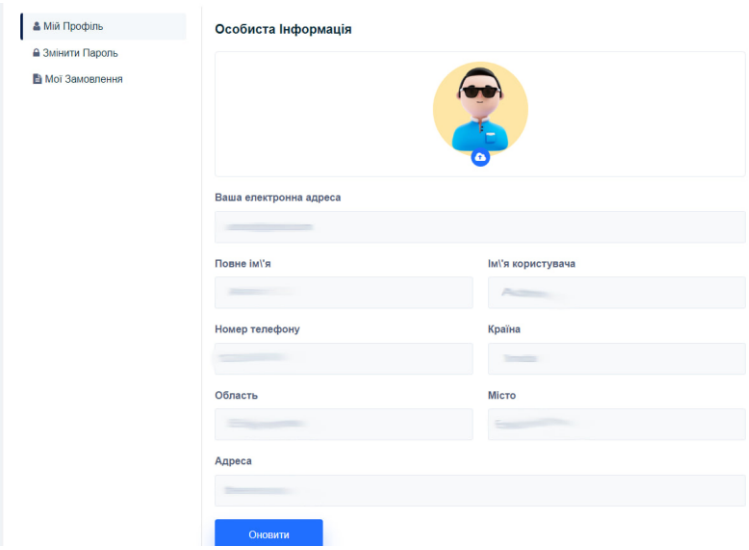



Рисунок 4.1.16 –Профіль користувача

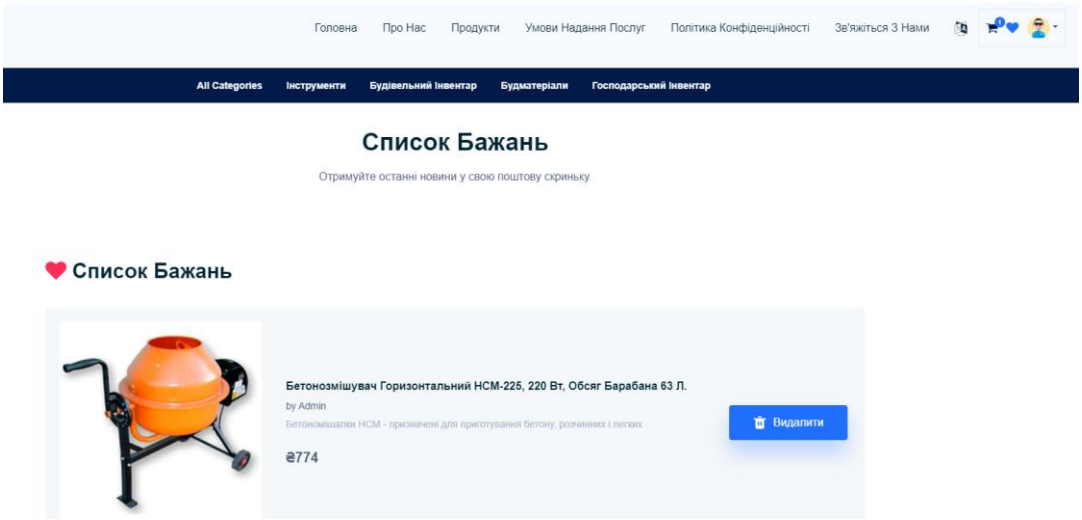



Рисунок 4.1.17 – Список бажань

## 4.2 Адміністративна частина

Щоб потрапити в адміністративну панель сайту потрібна авторизація. Знаючи логін та пароль, адміністратор сайту виконує вхід в панель (рис 4.2.1).

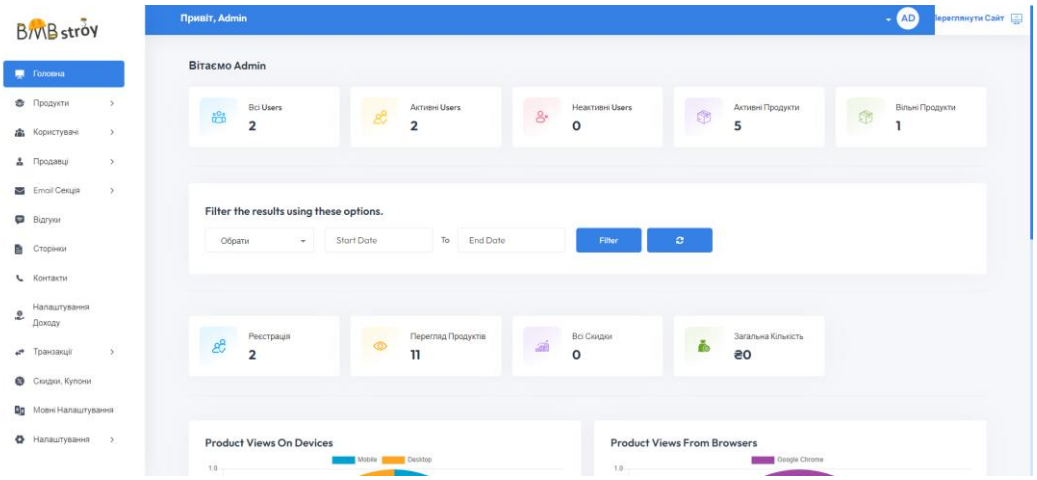



Рисунок 4.2.1 – Вигляд головної сторінки адміністративної частини 1

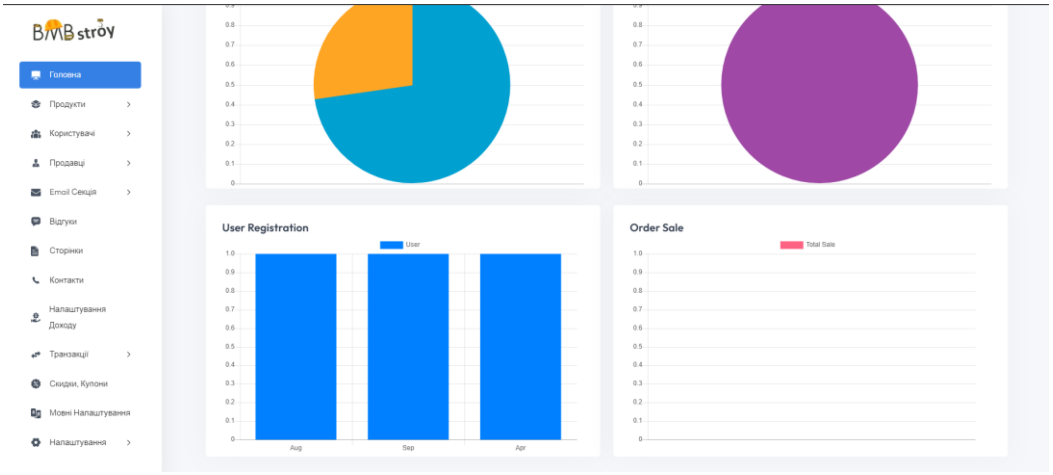



Рисунок 4.2.2 – Головна сторінка адміністративної панелі 2

Перш ніж почати додавати товар на сайт, треба створити категорії та підкатегорії, за якими будуть відображатися відповідні товари. Структура категорій є ієрархічною. Для того щоб створити категорію необхідно натиснути «Продукти – Категорії», після чого відкриється відповідна сторінка з формою створення категорії та переглядом вже існуючих (рис. 4.2.3, рис. 4.2.4).

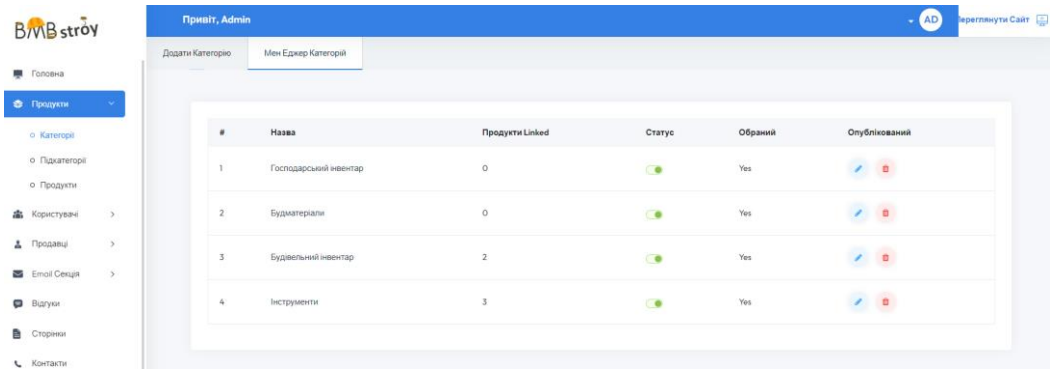



Рисунок 4.2.3 – Категорії

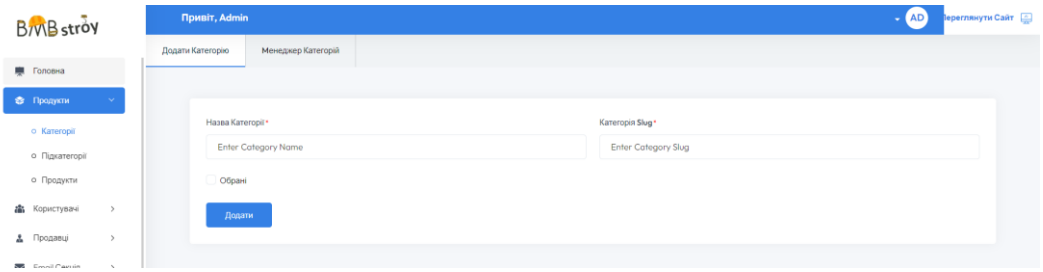



Рисунок 4.2.4 – Форма для створення категорій

Після створення категорій можна створити підкатегорії. Для того що б створити підкатегорію необхідно натиснути «Продукти – Підкатегорії», після чого відкриється відповідна сторінка з формою створення категорії та переглядом вже існуючих (рис 4.2.5, рис 4.2.6).

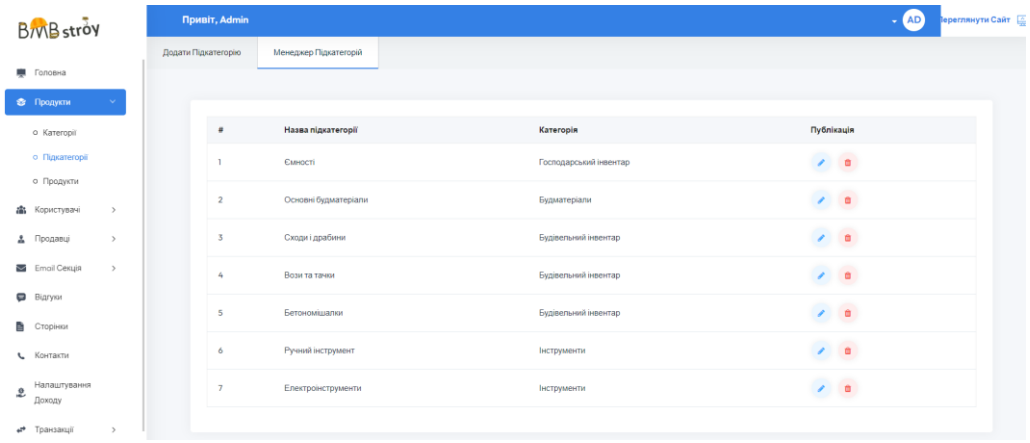



Рисунок 4.2.5 – Підкатегорії

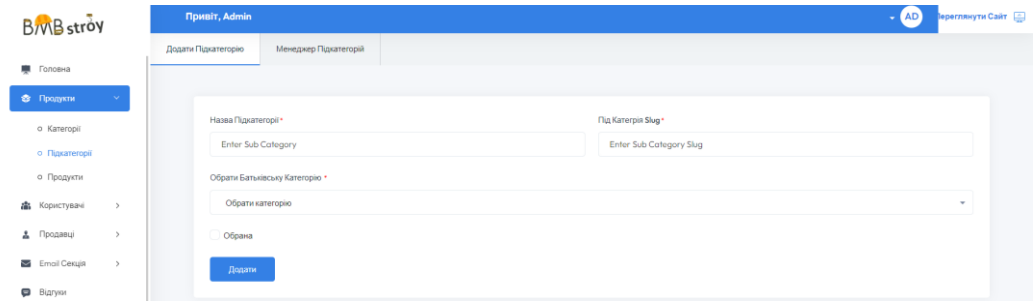



Рисунок 4.2.6 – Форма для створення підкатегорій

Після створення категорій можна додавати товари. Є тільки один спосіб це зробити:

Натиснути «Продукти – Товари– Додати новий». При переході на сторінку «Товари» також є можливість сортувати відсортувати продукти (Рисунок 4.2.7).

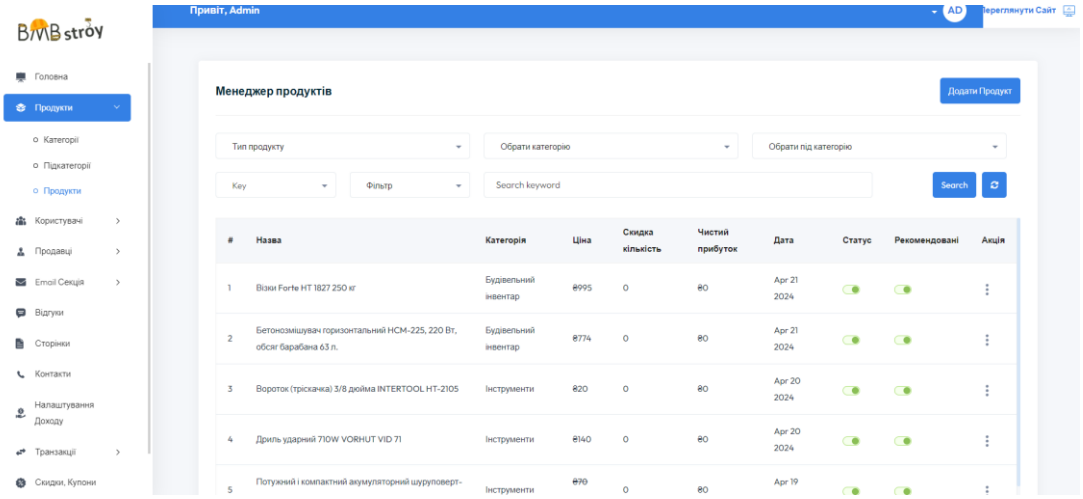



Рисунок 4.2.7 – Продукти

Натиснути «Додати Продукт». На цій сторінці потрібно все заповнювати: назва, опис, url, теги, SEO параметри, короткий опис товару, дані про товар і т.д. (Рисунок 4.2.8 Рисунок 4.2.9).

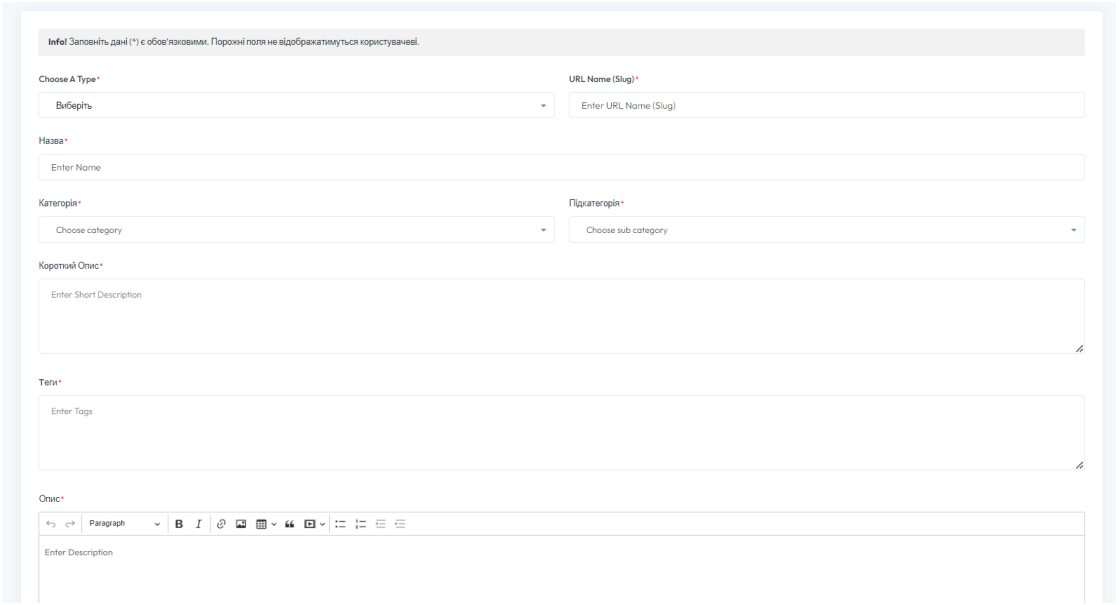



Рисунок 4.2.8 – Форма для додавання продукту 1

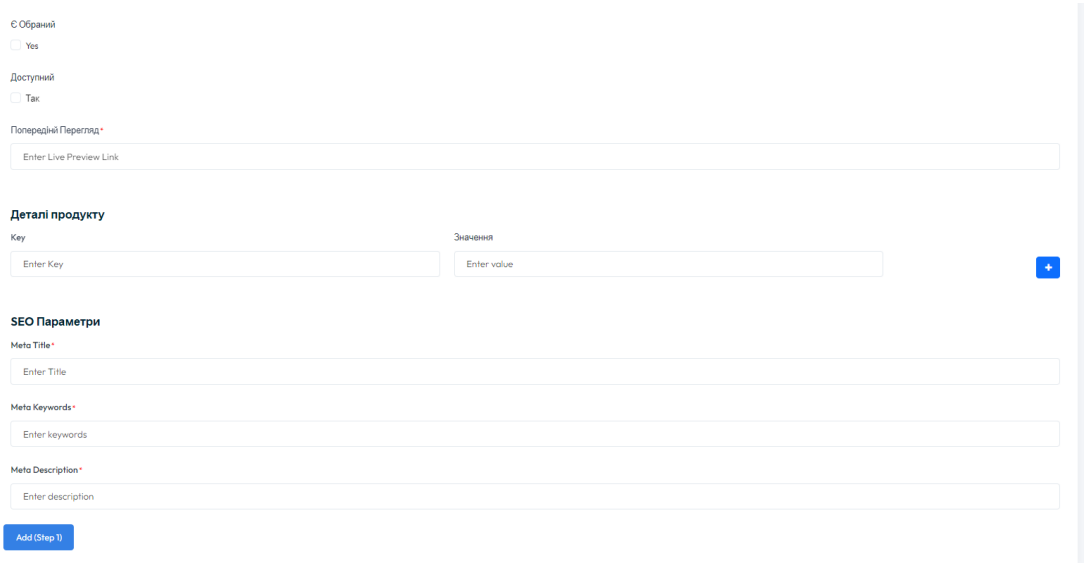



Рисунок 4.2.9 – Форма для додавання продукту 2

Натиснути Додати Наступну сторінку теж потрібно заповнити даними: вибір зображень, доступний показ товару для користувачів, ціна, знижка та фотогалерея (Рисунок 4.2.10). Натиснути кнопку «Виконати» і товар буде додано.

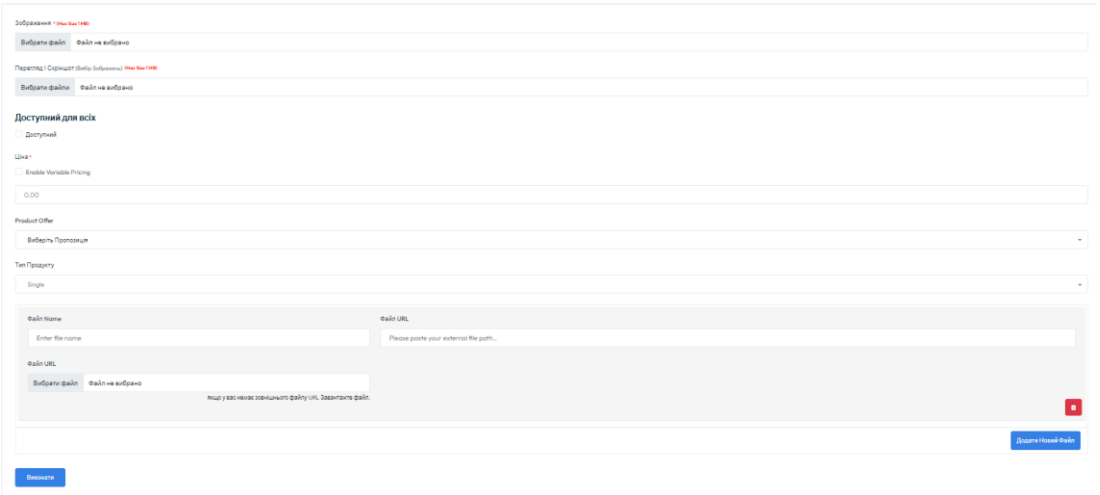
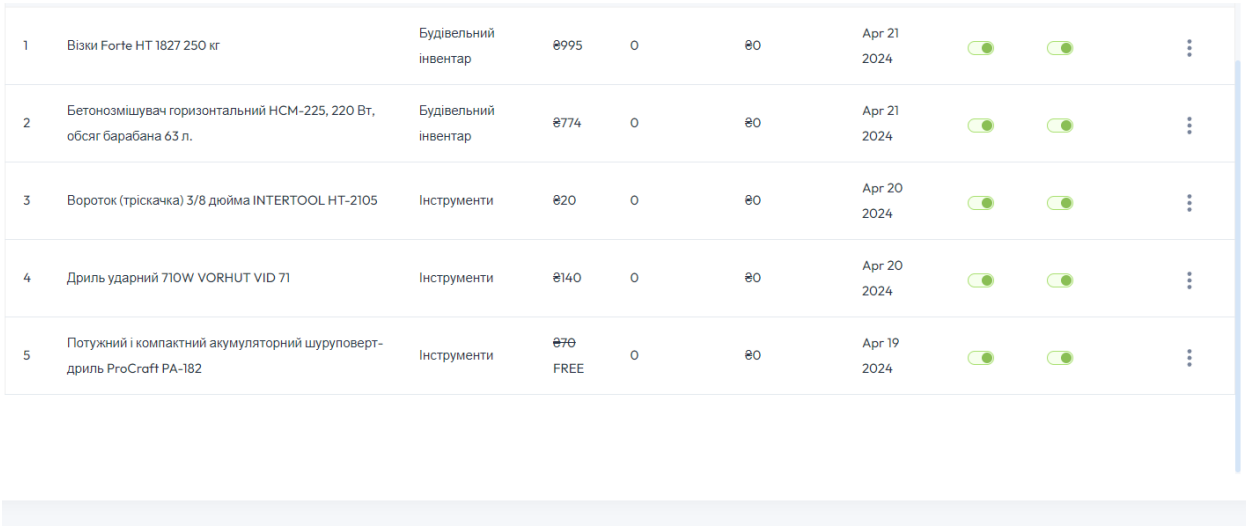



Рисунок 4.2.10 – Форма 2 для додавання продукту

Також є можливість перегляду та редагування вже доданих товарів (рис 4.2.11). При редагуванні товару відкриється те саме вікно, що і при його

додаванні.



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Візки Forte HT 1827 250 кг | Будівельний інвентар | ~~₴995~~ | 0 | ~~₴0~~ | Apr 21 2024 | | | |
| 2 | Бетонозмішувач горизонтальний HCM-225, 220 Вт, обсяг барабана 63 л. | Будівельний інвентар | ~~₴774~~ | 0 | ~~₴0~~ | Apr 21 2024 | | | |
| 3 | Вороток (тріскачка) 3/8 дюйма INTERTOOL HT-2105 | Інструменти | ~~₴20~~ | 0 | ~~₴0~~ | Apr 20 2024 | | | |
| 4 | Дриль ударний 710W VORHUT VID 71 | Інструменти | ~~₴140~~ | 0 | ~~₴0~~ | Apr 20 2024 | | | |
| 5 | Потужний і компактний акумуляторний шуруповерт-дриль ProCraft PA-182 | Інструменти | ~~₴70~~ FREE | 0 | ~~₴0~~ | Apr 19 2024 | | | |

Рисунок 4.2.11 – Перегляд всіх товарів

Переглянути та редагувати данні користувачів адміністратор може на сторінці «Користувачі» (Рисунок 4.2.12). Адміністратор може самостійно додавати користувачів, для цього потрібно: натиснути «Додати користувача» та заповнити форму (Рисунок 4.2.13 ).

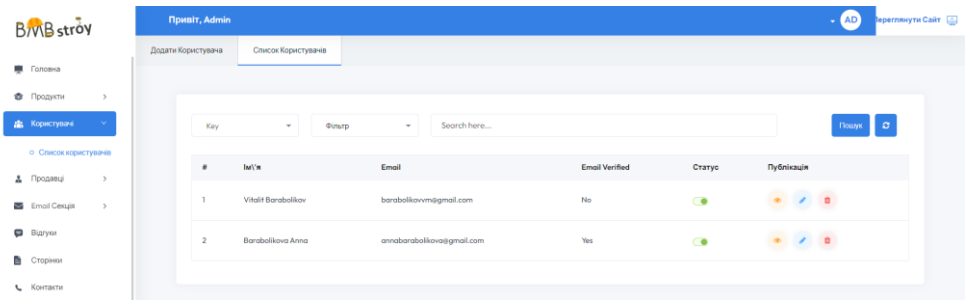



Рисунок 4.2.12 – Перегляд користувачів

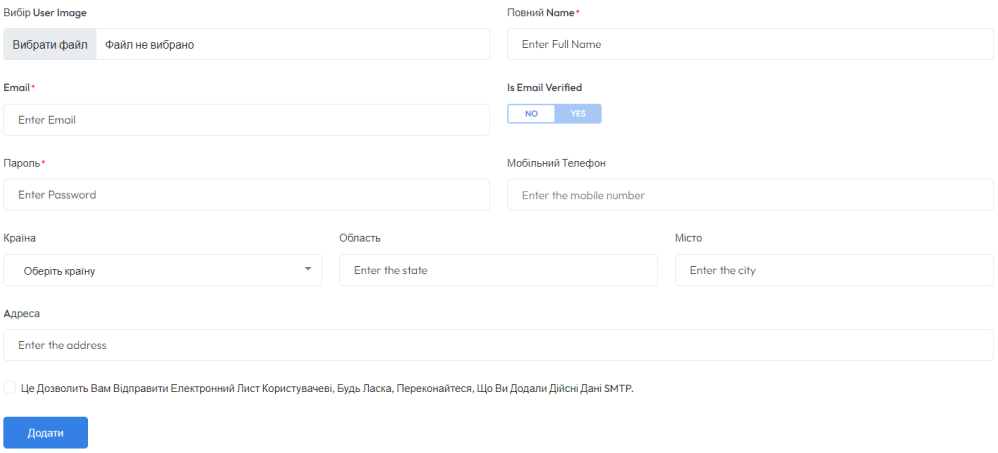



Рисунок 4.2.13 – Реєстрація користувачів адміністратором

Адміністратор може додавати та редагувати інформацію про постачальників. Для цього в лівому меню знаходиться розділ «Продавці». При натисканні підпункту «Список» відкривається сторінка з постачальниками (Рисунок 4.2.14). При кнопки «Додати Постачальника» адміністратору відкривається сторінка з формую яку потрібно заповнити та натиснути кнопку «Додати» (Рисунок 4.2.15). При редагуванні постачальника відкривається ця ж форма.

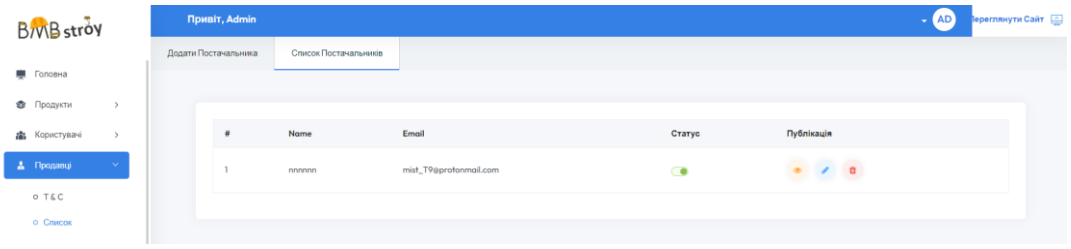



Рисунок 4.2.14 – Постачальники

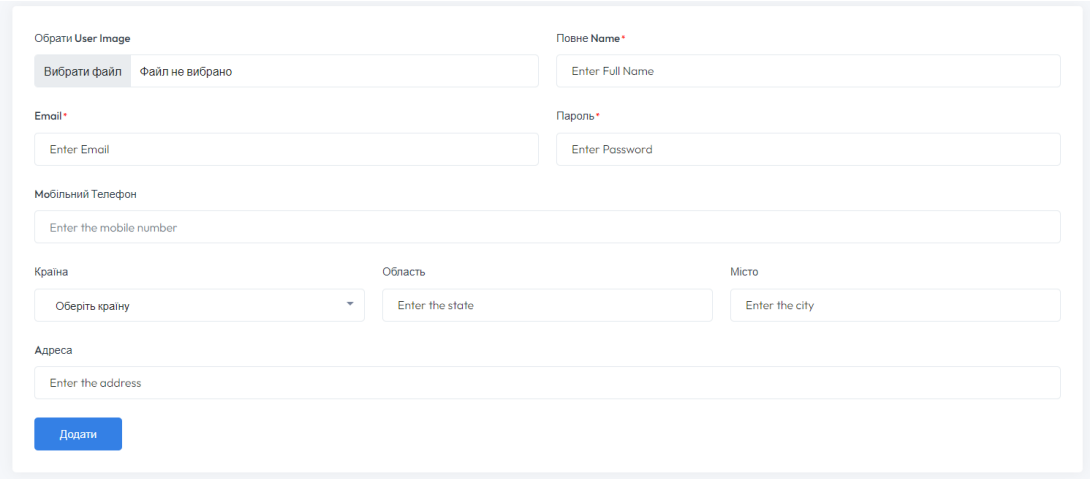



Рисунок 4.2.15 – Додавання постачальника

За допомогою E=mail секції адміністратор налаштовує шаблони повідомлень, інтеграції а також відслідковує підписки на розсилку (Рисунок 4.2.16).

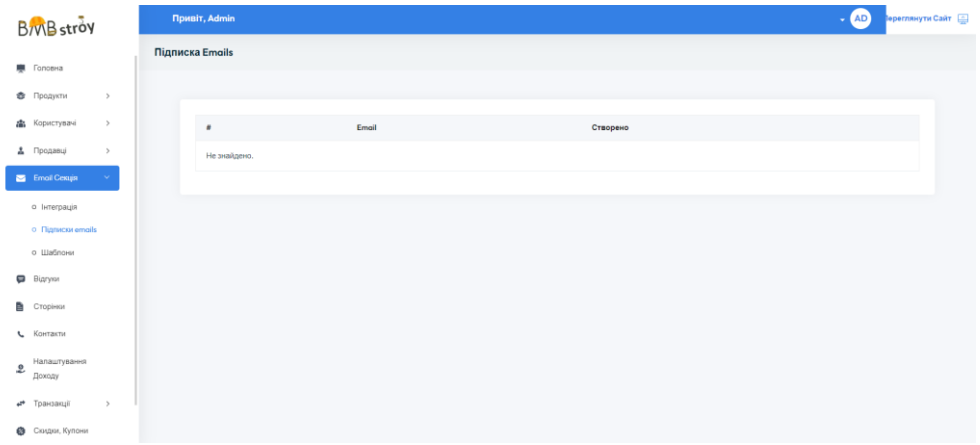



Рисунок 4.2.16 – E=mail секція

Для відстеження транзакцій адміністратору потрібно натиснути пункт меню «Транзакції» та при необхідності використовувати фільтрацію.

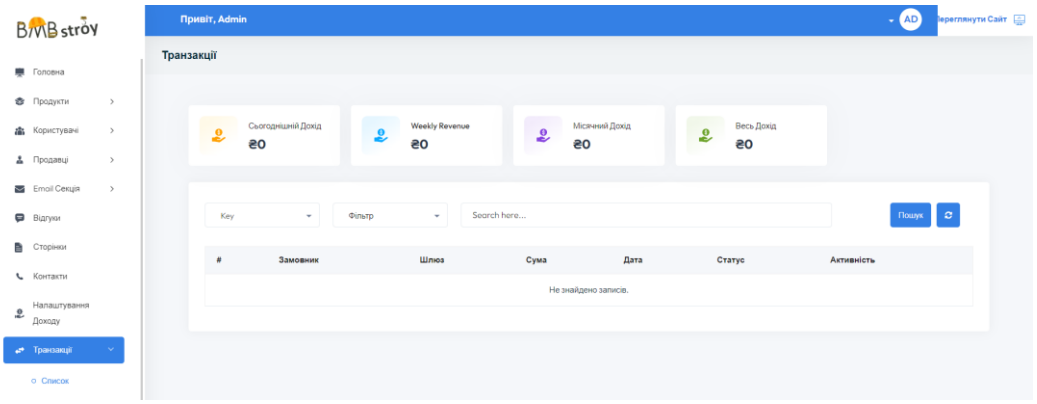



Рисунок 4.2.17 – Транзакції

Також для ведення акцій адміністратор може використовувати меню «Скидки, Купони». Сторінка купонів дозволяє редагувати та видаляти купони (Рисунок 4.2.18). Для створення купону потрібно на сторінці перейти у вкладку «Додати купон» і заповнити форму (Рисунок 4.2.19).

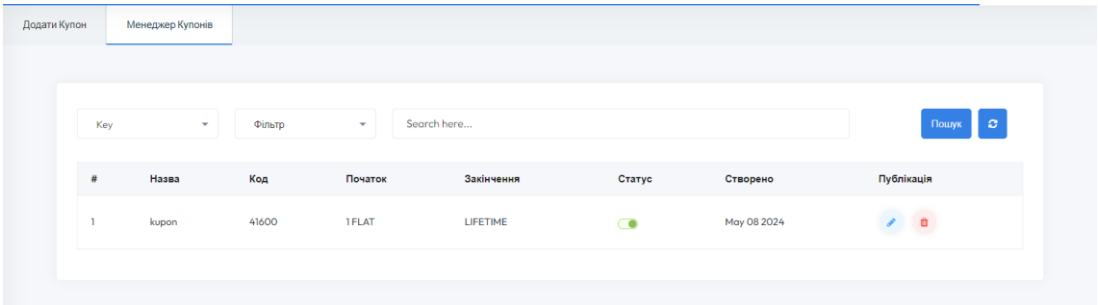



Рисунок 4.2.18 – Менеджер купонів

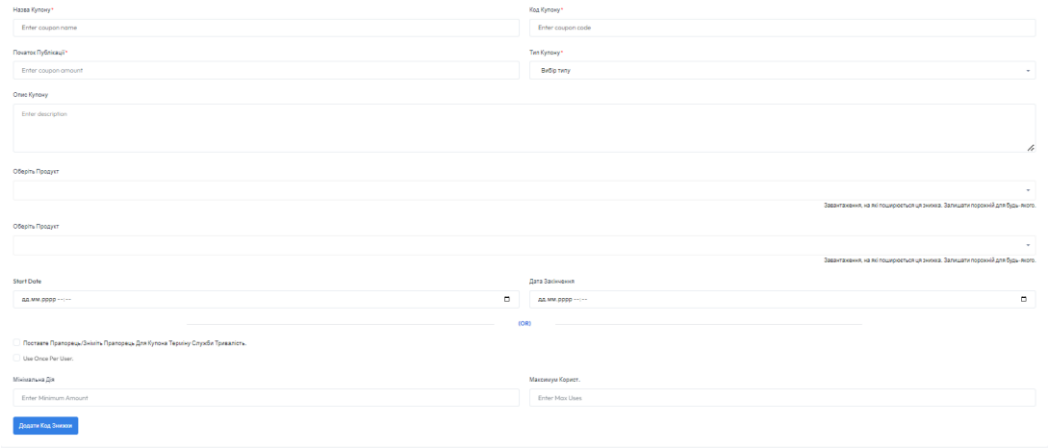



Рисунок 4.2.19 – Додавання купону

# ВИСНОВКИ

Основним завданням кваліфікаційної роботи було виконання поставленої задачі - створення інформаційної системи, яка б дозволяла ефективно управляти контентом сайту: додавати, редагувати та видаляти товари, категорії, описи, зображення, ціни тощо, отримувати та обробляти замовлення від клієнтів: реєструвати нових користувачів, відстежувати статус замовлень, надсилати повідомлення про замовлення, генерувати звіти про продажі, підвищувати зручність використання для адміністраторів та клієнтів: інтуїтивно зрозумілий інтерфейс, мобільна адаптивність, зручні форми та інструменти управління.

Актуальність даної роботи випливає з того, що зростання популярності онлайн-покупок робить необхідним використання ефективних інструментів для управління контентом та замовленнями у роздрібній торгівлі що б покращувати свою конкурентоспроможність, необхідність автоматизації та оптимізації процесів для допомогти автоматизування та оптимізувати багато рутинних завдань, пов'язаних з управлінням контентом та замовленнями, що призводить до економії часу та ресурсів, покращення якості обслуговання клієнтів.

У ході виконання кваліфікаційної роботи було виконано такі завдання:

1. Аналіз літератури та наукових джерел.
2. Вивчення існуючих систем управління контентом та замовленнями.
3. Аналіз бізнес-процесів роздрібних підприємств.
4. Проектування та розробка комплексної інформаційної системи.
5. Тестування розробленої системи.

Для реалізації веб додатку було використано такі технології:

Laravel: фреймворк PHP для розробки веб-застосунків, що забезпечує швидкість розробки, модульність, безпеку та масштабованість.

MVC (Model-View-Controller): архітектурний патерн, який розділяє логіку додатку на три компоненти: модель, вид та контролер, що покращує чіткість коду та його легкість у підтримці.

Blade: шаблонувач PHP, який використовується для динамічного генерування HTML-сторінок, що робить код більш лаконічним та читабельним.

phpMyAdmin: інструмент веб-інтерфейсу для управління базами даних MySQL, який використовується для створення, редагування та видалення даних, а також для виконання SQL-запитів.

Результатом роботи стала створена система, яка відповідає всім поставленим вимогам. Вона має зручний інтерфейс для адміністраторів та клієнтів, дозволяє ефективно управляти контентом сайту та замовленнями, а також генерувати звіти про продажі.

Практична цінність розробленої інформаційної системи в тому, що вона може бути використана для автоматизації та оптимізації процесів управління контентом та замовленнями у роздрібній торгівлі. Це може призвести до економії часу та ресурсів працівників, а також до покращення якості обслуговування клієнтів.

Крім того, дана інформаційна система може бути доопрацьована та розширена за рахунок додавання нових функцій та модулів, що робить її гнучким та масштабованим рішенням для роздрібної торгівлі.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Використання UML діаграм для аналізу і проектування інформаційних систем [Електронний ресурс] – Режим доступу до ресурсу: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://pns.hneu.edu.ua/pluginfile.php/321048/mod_resource/content/3/%D0%92%D0%B8%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D0%B0%D0%BD%D0%BD%D1%8F%20UML%20%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC.pdf.

2. ЗАСТОСУВАННЯ UML ДЛЯ МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ [Електронний ресурс] – Режим доступу до ресурсу: http://ir.stu.cn.ua/bitstream/handle/123456789/16930/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20uml%20%D0%B4%D0%BB%D1%8F%20%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D1%82%D0%B0%20%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D1%85%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC.%20.pdf?sequence=1&isAllowed=y.

3. Використання UML діаграм для аналізу і проектування інформаційних систем [Електронний ресурс] – Режим доступу до ресурсу: https://pns.hneu.edu.ua/mod/resource/view.php?id=170677.

4. ЗАСТОСУВАННЯ UML ДЛЯ МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ [Електронний ресурс] – Режим доступу до ресурсу: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://ir.stu.cn.ua/bitstream/handle/123456789/16930/%D0%97%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1

%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20uml%20%D0%B4%D0%BB%D1%8F%20%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D1%82%D0%B0%20%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%20%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B8%D1%85%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC.%20.pdf?sequence=1&isAllowed=y.

5. Blade Templates [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.com/docs/11.x/blade.

6. Що таке html? [Електронний ресурс] – Режим доступу до ресурсу: https://css.in.ua/article/shcho-take-css_3.

7. Довідник по CSS властивостям [Електронний ресурс] – Режим доступу до ресурсу: https://css.in.ua/css/properties.

8. Вступ до JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://uk.javascript.info/intro.

9. Чим PHP краще за інші мови програмування? [Електронний ресурс] – Режим доступу до ресурсу: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-php/.

10. Класи в програмуванні: занурення в об’єктно-орієнтоване програмування [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://foxminded.ua/klasy-v-prohramuvanni/.

11. Що таке міграція даних? [Електронний ресурс] – Режим доступу до ресурсу: https://experience.dropbox.com/uk-ua/resources/what-is-data-migration.

12. Контроллеры [Електронний ресурс] – Режим доступу до ресурсу: https://laravel.su/docs/10.x/controllers.

13. HTML ДОКУМЕНТ. СТРУКТУРА HTML ДОКУМЕНТА. ЩО ТЕСТУВАТИ? [Електронний ресурс] – Режим доступу до ресурсу:

https://training.qatestlab.com/blog/technical-articles/html-document-structure-what-to-test/.

# **ДОДАТОК А** Міграції

Create_users_table

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
 /**
  * Run the migrations.
  */
 public function up(): void
 {
  Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->tinyInteger('role')->comment('role 0 for admin , 1 for user , 2 for vendor');
    $table->string('role_type')->comment('ADMIN,USER,VENDOR');
    $table->string('avatar')->default('')->nullable();
    $table->string('username',100)->default('')->nullable();
    $table->string('full_name',150)->default('')->nullable();
    $table->string('gender',15)->default('')->nullable();
    $table->string('email',100)->nullable();
    $table->string('password')->nullable();
    $table->string('mobile',15)->default('')->nullable();
    $table->timestamp('birthday')->nullable();
    $table->string('address')->default('')->nullable();
    $table->integer('country_id')->nullable();
    $table->string('city')->default('')->nullable();
    $table->string('state')->default('')->nullable();
    $table->string('zip_code')->default('')->nullable();
    $table->boolean('is_active')->default('1')->nullable();
    $table->boolean('is_email_verified')->default('0')->nullable();
    $table->timestamp('email_verified_at')->nullable();
    $table->boolean('is_mobile_verified')->default('0')->nullable();
    $table->timestamp('mobile_verified_at')->nullable();
    $table->tinyInteger('rating')->default('0')->nullable();
    $table->timestamps();
    $table->softDeletes();
  });
 }

 /**
  * Reverse the migrations.
  */
 public function down(): void
 {
  Schema::dropIfExists('users');
```

```
  }
};
Create_products_table
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
  /**
   * Run the migrations.
   */
  public function up(): void
  {
    Schema::create('products', function (Blueprint $table){
      $table->id();
      $table->string('product_type',25)->nullale();
      $table->string('name')->nullable();
      $table->string('slug')->nullable();
      $table->string('image')->nullable();
      $table->integer('category_id')->nullable();
      $table->integer('sub_category_id')->nullable();
      $table->integer('user_id')->nullable();
      $table->string('tags')->nullable();
      $table->longtext('description')->nullable();
      $table->integer('quantity')->default(0)->nullable();
      $table->string('preview_link')->nullable();
      $table->string('uploaded_by')->nullable();
      $table->double('price')->nullable();
      $table->boolean('is_enable_multi_price')->default(0)->comment('1 for Yes 0 for No');
      $table->tinyInteger('file_type')->default(0)->comment('0 for single 1 for Bundle');
      $table->string('file_name')->nullable();
      $table->string('file_url')->nullable();
      $table->string('file_open_pass')->nullable();
      $table->integer('coupon_id')->nullable();
      $table->boolean('is_active')->default(1)->comment('1 for active 0 for inactive');
      $table->boolean('is_free')->default(0)->comment('1 for free 0 for No');
      $table->boolean('is_preview')->default(0)->comment('1 for Yes 0 for No');
      $table->boolean('is_featured')->default(0)->comment('1 for Yes 0 for No');
      $table->string('meta_title',100)->nullable();
      $table->string('meta_keywords')->nullable();
      $table->string('meta_desc',200)->nullable();
      $table->timestamps();
    });
  }

  /**
   * Reverse the migrations.
```

```
 */
public function down(): void
{
  Schema::dropIfExists('products');
}};
```

## **ДОДАТОК Б** Види

profile.blade

```
@php
  $ASSET_URL = asset('user-theme') . '/';
  $setting = getSetting();
  $price_symbol = $setting->default_symbol ?? '$';
@endphp
@extends('frontend.layout.master')
@section('head_scripts')
  <title>@lang('page_title.Frontend.user_profile_title')</title>
@endsection
@section('head_css')
  <link rel="stylesheet" href="{{ $ASSET_URL }}assets/css/rating.css" />
@endsection
@section('content')
  <!--===User Profile Section Start===-->
  <div class="tp_propage_wrapper">
    <div class="container">
      <div class="row">
        <div class="col-lg-12">
          <div class="tp_propage_head">
            <h2>@lang('master.user_profile.welcome_back')</h2>
            <p>@lang('master.user_profile.profile_heading')</p>
          </div>
        </div>
      </div>
      <div class="row tp_propage_text">
        <div class="col-lg-3 col-md-4 col-sm-4">
          <div class="nav nav-pills" id="v-pills-tab" role="tablist" aria-orientation="vertical">
            <button class="nav-link @if (empty(request('tab'))) active @endif" id="v-pills-profile-
tab"
              data-bs-toggle="pill" data-bs-target="#v-pills-profile" type="button" role="tab"
              aria-controls="v-pills-profile" aria-selected="true"><i class="fa fa-user"
                aria-hidden="true"></i>@lang('master.user_profile.my_profile')
            </button>
            <button class="nav-link" id="v-pills-profile-tab" data-bs-toggle="pill"
              data-bs-target="#v-pills-change_password" type="button" role="tab"
              aria-controls="v-pills-change_password" aria-selected="true"><i class="fa fa-lock"
                aria-hidden="true"></i>@lang('master.user_profile.change_password')
            </button>
            <button class="nav-link @if (request('tab') == 'my-orders') active @endif" id="v-pills-
invoice-tab"
              data-bs-toggle="pill" data-bs-target="#v-pills-invoice" type="button" role="tab"
              aria-controls="v-pills-invoice" aria-selected="false"><i class="fa fa-file-text"
                aria-hidden="true"></i>@lang('master.user_profile.my_order')
            </button>
            <button href="#v-pills-download" class="nav-link @if (request('tab') == 'my-downloads')
active @endif"
              id="v-pills-download-tab" data-bs-toggle="pill" data-bs-target="#v-pills-download"
```

```
          role="tab" aria-controls="v-pills-download" aria-selected="false"><i class="fa fa-
download"
            aria-hidden="true"></i>@lang('master.user_profile.my_download')
          </button>
        </div>
      </div>
      <div class="col-lg-9 col-md-8 col-sm-8">
        <div class="tab-content" id="v-pills-tabContent">
          <div class="tab-pane fade @if (empty(request('tab'))) show active @endif"
            id="v-pills-profile" role="tabpanel" aria-labelledby="v-pills-profile-tab">
            <div class="tp_propage_profile_wrapper">
              <div class="tp_propage_profilehead">
                <h3>@lang('master.user_profile.personal_details')</h3>
              </div>
              <form action="{{ route('frontend.update_profile', app()->getLocale()) }}"
method="Post"
                id="update_user_details">
                @csrf
                <div class="tp_propage_profile_form">
                  <div class="tp_input_main">
                    <div class="row">
                      <div class="col-lg-12 col-md-12">
                        <div class="card-body text-center">
                          <div class="tp_user_img">
                            <img src="@if (!empty($user->avatar)) {{ $user->avatar }} @else {{
$ASSET_URL . 'assets/images/user-profile.png' }} @endif"
                              alt="avatar" title="avatar" class="rounded-circle"
                              width="150px" height="150px" id="Imagepreview" />
                            <div class="tp_user_edit">
                              <i id="OpenImgUpload"
                                class="text-left fa fa-cloud-upload d-block"></i>
                            </div>
                            <input type="file" name="image" id="imgupload"
                              onchange="uploadImage('imgupload')" style="display:none" />
                          </div>
                        </div>
                      </div>
                      <div class="col-lg-12 col-md-12">
                        <div class="tp_input_text">
                          <label class="form-label">@lang('master.user_profile.contact_email')</label>
                          <input type="text" class="form-control"
                            placeholder="Enter Your Email" name="email"
                            value="{{ @$user->email }}" disabled>

                        </div>
                      </div>
                      <div class="col-lg-6 col-md-6">
                        <div class="tp_input_text">
                          <label class="form-label">@lang('master.user_profile.full_name')</label>
                          <input type="text" class="form-control"
```

```
                placeholder="Enter full Name" name="full_name"
                value="{{ @$user->full_name }}">

              </div>
            </div>
            <div class="col-lg-6 col-md-6">
              <div class="tp_input_text">
                <label class="form-label">@lang('master.user_profile.user_name')</label>
                <input type="text" class="form-control"
                placeholder="Enter User Name" name="username"
                value="{{ @$user->username }}">

              </div>
            </div>
            <div class="col-lg-6 col-md-6">
              <div class="tp_input_text">
                <label class="form-
label">@lang('master.user_profile.mobile_number')</label>
                <input type="number" class="form-control"
                placeholder="Enter Mobile Number" name="mobile"
                value="{{ @$user->mobile }}">

              </div>
            </div>
            <div class="col-lg-6 col-md-6">
              <div class="tp_input_text">
                <label class="form-label">@lang('master.user_profile.country')</label>
                <select class="form-control form-control-lg input-font"
                name="country_id">
                <option value="">@lang('master.user_profile.select_country')</option>
                @foreach ($country as $item)
                  <option value="{{ $item->id }}"
                    @if (@$user->country_id == $item->id) selected @endif>
                    {{ $item->name }}</option>
                @endforeach
                </select>
              </div>
            </div>
            <div class="col-lg-6 col-md-6">
              <div class="tp_input_text">
                <label class="form-label">@lang('master.user_profile.state')</label>
                <input type="text" class="form-control"
                placeholder="Enter Your state" name="state"
                value="{{ @$user->state }}">

              </div>
            </div>
            <div class="col-lg-6 col-md-6">
              <div class="tp_input_text">
                <label class="form-label">@lang('master.user_profile.city')</label>
```

```
                    <input type="text" class="form-control"
                      placeholder="Enter Your City" name="city"
                      value="{{ @$user->city }}">

                  </div>
                 </div>
                 <div class="col-lg-12 col-md-12">
                  <div class="tp_input_text">
                    <label class="form-label">@lang('master.user_profile.address')</label>
                    <input type="text" class="form-control"
                      placeholder="Enter Your Address" name="address"
                      value="{{ @$user->address }}">

                  </div>
                 </div>
                 <div class="col-lg-12 col-md-12">
                  <button class="btn btn-color btn-lg px-5 py-2 btn-font tp_btn"
                    type="submit"
                    id="update_user_details_btn">@lang('master.user_profile.update')</button>
                 </div>
                </div>
               </div>
              </div>
             </form>
            </div>
           </div>
           <div class="tab-pane fade" id="v-pills-change_password" role="tabpanel"
            aria-labelledby="v-pills-change_password-tab">
            <div class="tp_propage_profile_wrapper">
             <div class="tp_propage_profilehead">
              <h3>@lang('master.user_profile.change_password')</h3>
             </div>
             <form action="{{ route('frontend.change-password', app()->getLocale()) }}"
method="Post"
              id="change_password_form_id">
              @csrf
              <div class="tp_propage_profile_form">
               <div class="tp_input_main">
                <div class="tp_input_text">
                 <label class="form-label">@lang('master.user_profile.old_password')</label>
                 <input type="password" name="old_password" class="form-control"
                  placeholder="Enter Your Old Password" />

                </div>
                <div class="tp_input_text">
                 <label class="form-label">@lang('master.user_profile.new_password')</label>
                 <input type="password" name="password" class="form-control"
                  placeholder="Enter Your Password" />
                </div>
                <div class="tp_input_text">
```

```
                    <label class="form-
label">@lang('master.user_profile.confirm_password')</label>
                    <input type="password" name="confirm_password" class="form-control"
                      placeholder="Enter Your confirm password" />

                  </div>
                  <button class="btn btn-color btn-lg px-5 py-2 btn-font tp_btn" type="submit"

id="change_password_form_id_btn">@lang('master.user_profile.update')</button>
                </div>
              </div>
            </form>
          </div>
        </div>
        <div class="tab-pane fade @if (request('tab') == 'my-orders') active @endif" id="v-pills-
invoice"
          role="tabpanel" aria-labelledby="v-pills-invoice-tab">
          <div class="tp_propage_profile_wrapper">
            <div class="tp_propage_profilehead tp_propage_invoice">

              <h3>@lang('master.billing_details.orders')</h3>
            </div>
            <div class="tp_table_box tp_propage_table">
              <div class="table-responsive">
                <table id="example" class="table">
                  <thead>
                    <tr>
                      <th>#</th>
                      <th>@lang('master.billing_details.tnx_number')</th>
                      <th>@lang('master.billing_details.amount')</th>
                      <th>@lang('master.billing_details.payment_date')</th>
                      <th>@lang('master.billing_details.status')</th>
                      <th>@lang('master.billing_details.invoice')</th>
                    </tr>
                  </thead>
                  <tbody>
                    @if (isset($user->getOrders[0]))
                      @foreach ($user->getOrders as $key => $items)
                        <tr>
                          <td>{{ ++$key }}</td>
                          <td>{{ $items->tnx_id }}</td>
                          <td>{{ $price_symbol }}{{ @$items->billing_total }}</td>
                          <td>{{ set_date_format($items->created_at) }}</td>
                          <td>
                            {{ $items->status }}

                          </td>
                          <td>
                            <ul>
```

```
                        <li><a href="{{ route('frontend.download-invoice', [app()->getLocale(),
'tnx_id' => $items->tnx_id]) }}"
                            class="tp_edit" title="Edit"><i
                              class="fa fa-eye"
                              aria-hidden="true"></i></a>
                          </li>
                        </ul>
                      </td>
                    </tr>
                  @endforeach
                @else
                  <tr class="text-center">
                    <td colspan="5">@lang('master.billing_details.not_found')</td>
                  </tr>
                @endif

              </tbody>
            </table>
          </div>
        </div>
      </div>
    </div>
    <div class="tab-pane fade @if (request('tab') == 'my-downloads') show active @endif"
      id="v-pills-download" role="tabpanel" aria-labelledby="v-pills-download-tab">
      <div class="tp_propage_profile_wrapper">
        <div class="tp_propage_profilehead tp_propage_pay">
          <h3>@lang('master.billing_details.download')</h3>
        </div>

        <div class="tp_propage_download">
          <div class="row">
            @if (isset($user->getOrders[0]))
              @foreach ($user->getOrders as $key => $items1)
                @foreach ($items1->getOrderProduct as $key2 => $items2)
                  @foreach ($items2->getProduct as $key3 => $items)
                    <div class="col-lg-4 col-md-6 col-sm-12">
                      <div class="tp_pro_downbox">
                        <div class="tp_download_img">
                          <img src="{{ $items->thumb }}"
                            class="tp-animation-img" alt="project-img" />
                        </div>
                        <div class="tp_download_text">
                          <div class="tp_download_text_head">
                            <h5>{{ $items->name }}</h5>
                            <p>by {{ $items->getUser->full_name }}</p>
                          </div>
                          <div class="star_rating">
                            <div class="star_rating">
                              @include(
                                'frontend.include.rating',
```

```
                        [
                          'faRating' => true,
                          'rating' => @$items->rating,
                        ]
                       )
                      </div>
                     </div>
                     <div class="tp-dwld-toggle">
                      <div class="dropdown">
                       <button class="btn tp_btn dropdown-toggle"
                        type="button" data-bs-toggle="dropdown"
                        aria-expanded="false">
                        Downlaod
                       </button>

                       @php $fileArr = $items->getdownlaodfilelink(@$items2->variants);
@endphp
                       @if (count($fileArr) > 0)
                        <ul class="dropdown-menu">
                         @foreach ($fileArr as $key => $value)
                          <li> <a class="dropdown-item"
                              href="{{ route('frontend.download-file', [app()->getLocale(),
'file_id' => @$value['file_id'], 'tnx_id' => @$items1->tnx_id, 'pid' => @$value['product_id']])
}}">
                              <i class="fa fa-download"
                               aria-hidden="true"></i>
                              {{ @$value['file_name'] }}
                             </a>
                            </li>
                          @endforeach
                         </ul>
                        @endif
                       </div>

                       <button type="button" data-bs-toggle="modal"
                        data-bs-target="#reviewmodal"
                        class="btn tp_btn tp_rev_btn"
                        onclick="setReview('{{ $items->thumb }}',
                       '{{ $items->slug }}',
                       '{{ @$items1->tnx_id }}',
                       '{{ @$items->getUserProductReview->id }}',
                       '{{ @$items->getUserProductReview->rating }}')
                       ">Review</button>
                      </div>
                      <input type="hidden"
                       id="r_comment_{{ @$items->getUserProductReview->id }}"
                       value="{{ @$items->getUserProductReview->comment }}">
                     </div>
                    </div>
                   </div>
```

```
                    @endforeach
                  @endforeach
                @endforeach
              @else
                <p>No Downlaod Found.</p>
              @endif
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
 </div>
 <!--=== Section Start===-->
 {{-- Review model --}}
 <div class="modal fade" id="reviewmodal" tabindex="-1" aria-labelledby="reviewmodallabel"
aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered">
   <div class="modal-content bdr-radius rounded-4 overflow-hidden">
    <div class="row d-flex justify-content-center align-items-center h-100">
     <div class="col-12 col-md-12 col-lg-12 col-xl-12">
      <div class="card bg-white text-dark shadow-none tp_review_model_wrapper">
       <button type="button" class="btn-close" data-bs-dismiss="modal"
        aria-label="Close"></button>
       <div class="card-body">
        <form method="post" class="tp_review_form" id="user_set_review"
         action="{{ route('admin.rating.store', app()->getLocale()) }}">
         <div class="tp_review_box">
          <img id="md-review-img" src="" class="tp-review-img" alt="review-img">
          <div class="tp_review_box_data">
           <h3>@lang('master.review_model.product')</h3>
           <div class="tp_review_star">
            <p>@lang('master.review_model.star_rate')</p>

            <div class="col">
             <div class="rate">
              <input type="radio" id="star5" class="rate"
               name="rating" value="5" />
              <label for="star5" title="text">@lang('master.review_model.5_stars')</label>
              <input type="radio" checked id="star4" class="rate"
               name="rating" value="4" />
              <label for="star4" title="text">@lang('master.review_model.4_stars')</label>
              <input type="radio" id="star3" class="rate"
               name="rating" value="3" />
              <label for="star3" title="text">@lang('master.review_model.3_stars')</label>
              <input type="radio" id="star2" class="rate"
               name="rating" value="2">
              <label for="star2" title="text">@lang('master.review_model.2_stars')</label>
```

```
                    <input type="radio" id="star1" class="rate"
                      name="rating" value="1" />
                    <label for="star1" title="text">@lang('master.review_model.1_stars')</label>
                   </div>
                  </div>
                 </div>
                </div>
               </div>
               <div class="tp_reviewform_box">
                <div class="tp_input_text">
                 <textarea placeholder="Write a review here... " class="form-control"
id="rt_comment" name="comment" rows="3"
                   cols="50"></textarea>
                </div>
                <button id="user_set_review_btn"
                 class="btn btn-color btn-lg px-5 py-2 btn-font tp_btn" type="submit">post
                 Review</button>
               </div>
               <input type="hidden" name="pid" id="_pid" value="">
               <input type="hidden" name="txid" id="_txid" value="">
               <input type="hidden" name="rid" id="_rate_id" value="">
              </form>
             </div>
            </div>
           </div>
          </div>
         </div>
        </div>
       </div>
@endsection
@section('scripts')
 <script src="{{ asset('user-theme/my_assets/js/validation.js') }}"></script>
@endsection
```

# **ДОДАТОК В** Контроллери

UserController

```
<?php

namespace App\Http\Controllers\Frontend;
use App\Http\Controllers\Controller;
use Validator, Auth, Redirect, Mail;
use App\Models\{User};
use DB, Socialite, Hash, Response, URL, Session, DataTables, Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Crypt;
use Illuminate\Contracts\Encryption\DecryptException;
use Illuminate\Validation\Rule;
use Illuminate\Support\Str;
use App\Http\Middleware\ValidateSessionToken;

class UserController extends Controller
{
 /**
  * user login by credentails
  */
        function login(Request $request)
        {
                $rules['email'] = 'required|email';
                $rules['password'] = 'required';

                $msg['email.required'] = trans('frontend_msg.req_email');
                $msg['email.email'] = trans('frontend_msg.wrong_email_format');
                $msg['password.required'] = trans('frontend_msg.req_password');

                $validator = Validator::make($request->all(), $rules, $msg);
                if ($validator->fails())
                        return json_response(['status' => false, 'msg' => $validator->messages()-
>first()], 400);

                $userdata = ['email' => $request->email, 'password' => $request->password];
                if (Auth::attempt($userdata)) // CHECK LOGIN
                {
                        $user = Auth::user();
                        if($user->is_active == 0){ //check user is active or not
                                return json_response(['status' => false,'msg'
=>trans('frontend_msg.blockuser')], 400);
                        }
                        if($user->is_email_verified == 0){ //check user's email is verified or not
                                $this->sendMail($user,'registration'); //sent email to verify mail
                                Auth::logout(); //user logout
                          return json_response(['status' => false,'msg'
=>trans('frontend_msg.email_not_verify')], 400);
                        }
```

```
                return json_response(['status' => true, 'url' =>"", 'msg' =>
trans('frontend_msg.succ_login')], 200);
            } else {
                return json_response(['status' => false, 'msg' =>
trans('frontend_msg.wrong_credential')], 400);
            }
        }

        /**
  * user signup
  */
        public function signup(Request $request)
        {
            $rules = [
                'name'  => 'required|min:3|max:25',
                'email' => 'unique:users,email',
                'password' => 'required|min:6',
            ];
            $msg['name.required'] = trans('frontend_msg.req_name');
            $msg['email.required'] = trans('frontend_msg.req_email');
            $msg['email.email'] = trans('frontend_msg.wrong_email_format');
            $msg['email.unique'] = trans('frontend_msg.unique_email_format');
            $msg['password.required'] = trans('frontend_msg.new_password');
            $msg['confirm_password.required'] = trans('frontend_msg.req_confirmPassword');
            $msg['confirm_password.same'] =
trans('frontend_msg.password_confirmed_not_match');

            $validator = Validator::make($request->all(), $rules, $msg);
            if ($validator->fails()) {
                return json_response(['status' => false, 'msg' => $validator->messages()-
>first()], 400);
            }

            $obj = new User();
            $obj->full_name = $request->name;
            $obj->email = $request->email;
            $obj->password = bcrypt($request->password);
            $obj->role = 1;
            $obj->role_type = "USER";
            $user = $obj->save();

            $this->sendMail($obj ,'registration'); //sent email to verify mail

            if ($user)
                return json_response(['status' => true, 'url' => route('frontend.sign-
in',app()->getLocale()), 'msg' => trans('frontend_msg.succ_signup_with_email_verify')], 200);
            else
                return json_response(['status' => false, 'msg' =>
trans('frontend_msg.something_went_wrong')], 400);
```

```
        }

        /**
  * user logout
  */
        public function logout(Request $request)
        {
                Auth::logout();
                return redirect()->back();
        }

        /**
  * user update our profile
  */
        public function updateProfile(Request $request)
        {
                $rules = [
                        'username'  => ['required','min:6','max:25',Rule::unique('users')-
>where(function($query) use($request){
                                $query->where('id', '!=', Auth::id());
                        })],
                        'full_name'  => 'required|min:3|max:25',
                ];

                $msg['name.required'] = trans('frontend_msg.req_name');
                $validator = Validator::make($request->all(), $rules, $msg);

                if ($validator->fails()) {
                        return json_response(['status' => false, 'msg' => $validator->messages()-
>first()], 400);
                }

                $obj = Auth::user();
                $obj->full_name = $request->full_name;
                $obj->username = $request->username;
                if(@$request->mobile)
                $obj->mobile = $request->mobile;
                $obj->country_id = $request->country_id;
                $obj->city = $request->city;
                $obj->state = $request->state;
                $obj->address = $request->address;
                $user = $obj->save();
                if ($user)
                        return json_response(['status' => true, 'msg' =>
trans('frontend_msg.succ_update_profile'),'url'=>route('frontend.profile', app()->getLocale())],
200);
                else
                        return json_response(['status' => false, 'msg' =>
trans('frontend_msg.something_went_wrong')], 400);
        }
```

```
        // CHANGE PASSWORD
        function change_password(Request $request)
        {
                $rules['old_password'] = 'required';
                $rules['password'] = ['required', 'min:6','regex:/^(?=.*[a-z])(?=.*[A-
Z])(?=.*\d)(?=.*(_|[^\w])).+$/'];
                $rules['confirm_password'] = 'required|same:password';
                $msg['old_password.required'] = trans('frontend_msg.req_old_password');
                $msg['password.required'] = trans('frontend_msg.new_password');
                $msg['password.min'] = trans('frontend_msg.req_password_min');
                $msg['password.regex'] = trans('frontend_msg.invalid_password_format');
                $msg['confirm_password.required'] = trans('frontend_msg.req_confirmPassword');
                $msg['confirm_password.same'] =
trans('frontend_msg.password_confirmed_not_match');

                $validator = Validator::make($request->all(), $rules, $msg);
                if ($validator->fails())
                        return json_response(['status' => false, 'msg' => $validator->messages()-
>first()], 400);

                $user = User::find(Auth::user()->id);
                if (!Hash::check($request->old_password, $user->password)){// CHECK OLD
PASSWORD
                        return json_response(['status' => false, 'msg' =>
trans('frontend_msg.invalid_old_password')], 400);
                }
                if($request->old_password == $request->password){
    return json_response(['status' => false, 'msg' =>
trans('frontend_msg.old_pas_current_pass_not_same')], 400);
   }

                $user->password = bcrypt($request->password);
                $user->save();
                return json_response(['status' => true, 'msg' =>
trans('frontend_msg.succ_update_password')], 200);
        }

        //UPDATE IMAGE
        function update_user_image(Request $request)
        {
                $rules['image'] = 'required|image|mimes:jpeg,png,jpg,gif,svg|max:1000';
                $msg['image.required'] = trans('frontend_msg.req_user_image');
                $msg['image.mimes'] = trans('frontend_msg.req_user_image_mimes');
                $msg['image.max'] = trans('frontend_msg.req_user_image_max');

                $validator = Validator::make($request->all(), $rules);
                if ($validator->fails())
                        return json_response(['status' => false, 'msg' => $validator->messages()-
>first()], 400);
```

```
                $obj = Auth::user();
                $obj->save();
                return json_response(['status' => true, 'msg' =>
trans('frontend_msg.succ_update_user_image'),'url'=>route('frontend.profile', app()-
>getLocale())], 200);
        }

        // SEND MAIL FOR RESET FORGET PASSWORD
        function forgot_password(Request $request)
        {
                $rules['email'] = 'required|email';
                $msg['email.required'] = trans('frontend_msg.req_email');
                $msg['email.email'] = trans('frontend_msg.wrong_email_format');
                $validator = Validator::make($request->all(), $rules, $msg);
                if ($validator->fails())
                        return json_response(['status' => false, 'msg' => $validator->messages()-
>first()], 400);

                $user = User::where('email', '=', $request->email)->first();
                if (empty($user)) {
                        return json_response(['status' => false, 'msg' =>
trans('frontend_msg.user_not_register')], 400);
                }

                $passReset = DB::table('password_reset_tokens')->where('email', $request-
>email)->orderBy('created_at', 'DESC')->first();
                //check if the existing email created time is greater than to current time
                if (!empty($passReset) && $passReset->created_at > date('Y-m-d H:i:s')) {
                        return json_response(['status' => true, 'msg' =>
trans('frontend_msg.already_password_reset_send_mail')], 200);
                }else{
                        DB::table('password_reset_tokens')->where('email', $request->email)-
>delete();
                }

                $check = $this->sendMail($user,'forget_password'); //Send forgot password mail

                if ($check) {
                        return json_response(['status' => true, 'msg' =>
trans('frontend_msg.succ_password_reset_send_mail')], 200);
                } else {
                        return json_response(['status' => false, 'msg' =>
trans('frontend_msg.failed_password_reset_send_mail')], 400);
                }
        }

        // UPDATE RESET PASSWORD
        public function update_reset_password(Request $request)
        {
```

```
                $rules['token'] = 'required|exists:password_reset_tokens,token';
                $rules['password'] = 'required';
                $rules['confirmpassword'] = 'required|same:password';

                $msg['token.required'] = trans('frontend_msg.req_token');
                $msg['token.exists'] = trans('frontend_msg.invalid_token');
                $msg['password.required'] = trans('frontend_msg.new_password');
                $msg['password.min'] = trans('frontend_msg.req_new_password_min');
                $msg['password.regex'] = trans('frontend_msg.invalid_new_password_format');
                $msg['confirmpassword.required'] = trans('frontend_msg.req_confirmPassword');
                $msg['confirmpassword.same'] =
trans('frontend_msg.new_password_confirmed_not_match');

                $validator = Validator::make($request->all(), $rules, $msg);
                if ($validator->fails())
                        return json_response(['status' => false, 'msg' => $validator->messages()-
>first()], 400);

                $tokenData = DB::table('password_reset_tokens')->where('token', $request-
>token)->first();
                if (empty($tokenData))
                        return json_response(['status' => false, 'msg' =>
trans('frontend_msg.invalid_token')], 400);

                //updated password
                $user = User::where('email', $tokenData->email)->first();
                $user->update(["password" => bcrypt($request->password)]);
                //destroyed email
                DB::table('password_reset_tokens')->where('email', $tokenData->email)-
>delete();
                if($user->role == 0)//Check role is admin redirect to admin login
                $url = route('admin.login');
                else
                $url = route('frontend.sign-in',app()->getLocale());
                return json_response(['status' => true, 'msg' =>
trans('frontend_msg.succ_update_password'), 'url' => $url], 200);
        }

        /**
  * common function for sending mail
  */
        public function sendMail($request,$type){

                $setting = (object) DB::table('settings')->pluck('short_value','key')->toArray();
                if($setting->is_checked_smtp == 1){ //if user have set smtp credentails
                        if (DB::table('password_reset_tokens')->where('email', $request->email)-
>exists()) {
                                DB::table('password_reset_tokens')->where('email', $request-
>email)->delete();
                        }
```

```
                        $token = unique_key(); // GENERATE TOKEN
                        DB::table('password_reset_tokens')->insert(['email' => $request->email,
'token' => $token, 'created_at' => date('Y-m-d H:i:s', strtotime('+4 hour'))]);

                        if($type == 'registration'){
                                $template = $setting->registration_template;
                                $link_text = '<a href="'.route('frontend.email-verify', [app()-
>getLocale(),"token" => $token]).'">'.$setting->reg_link_text.'</a>';
                                $subject = trans('frontend_msg.subject_email_verification');
                        }
                        elseif($type == 'forget_password'){
                                $template = $setting->forget_password_template;
                                $link_text = '<a href="'.route('frontend.reset-password',[app()-
>getLocale(),"token" => $token]).'">'.$setting->forget_pass_link_text.'</a>';
                                $subject = trans('frontend_msg.subject_forget_password');
                        }
                        $final_template = "";

                        if($setting->is_checked_logo_on_mail) //if checked logo on mail
                        {
                                $final_template .= '<img src="'.\Storage::url(getsetting()-
>my_logo).'" alt="logo" width="200px" height="50px" > </br> </br>'."\r\n";
                        }

                        //get an email template from the database and replace the template content
                        $new_template = str_replace('[username]',$request->full_name ??
'username',$template);
                        $new_template = str_replace('[break]','</br>',$new_template);
                        $new_template = str_replace('[linktext]',$link_text,$new_template);

                        $final_template .= $new_template;

                        $obj = (object)[];
                        $obj->receiver_email = $request->email;
                        $obj->template = $final_template;
                        $obj->subject = $subject;
                        return send_mail($obj); //custom helper function
                }

                return false;
        }
}
```